Re-feedback:

Freedom with Accountability for Causing Congestion in a Connectionless Internetwork

Robert Briscoe

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University of London.

Department of Computer Science

University College London

15 May 2009

To Lyn

I, Robert John Briscoe confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

_____ 15 May 2009

Abstract

This dissertation concerns adding resource accountability to a simplex internetwork such as the Internet, with only necessary but sufficient constraint on freedom. That is, both freedom for applications to evolve new innovative behaviours while still responding responsibly to congestion; and freedom for network providers to structure their pricing in any way, including flat pricing.

The big idea on which the research is built is a novel feedback arrangement termed ‘re-feedback’. A general form is defined, as well as a specific proposal (re-ECN) to alter the Internet protocol so that self-contained datagrams carry a metric of expected downstream congestion.

Congestion is chosen because of its central economic role as the marginal cost of network usage. The aim is to ensure Internet resource allocation can be controlled either by local policies or by market selection (or indeed local lack of any control).

The current Internet architecture is designed to only reveal path congestion to end-points, not networks. The collective actions of self-interested consumers and providers should drive Internet resource allocations towards maximisation of total social welfare. But without visibility of a cost-metric, network operators are violating the architecture to improve their customer’s experience. The resulting fight against the architecture is destroying the Internet’s simplicity and ability to evolve.

Although accountability with freedom is the goal, the focus is the congestion metric, and whether an incentive system is possible that assures its integrity as it is passed between parties around the system, despite proposed attacks motivated by self-interest and malice.

This dissertation defines the protocol and canonical examples of accountability mechanisms. Designs are all derived from carefully motivated principles. The resulting system is evaluated by analysis and simulation against the constraints and principles originally set. The mechanisms are proven to be agnostic to specific transport behaviours, but they could not be made flow-ID-oblivious.

Acknowledgements

I am forever indebted to my wife, Lyn, for putting up with me obsessing over this research. And to my youngest son Joe who missed some of the attention his brothers got.

Jon Crowcroft provided excellent supervision and spot-on guidance throughout. Despite switching from UCL to Cambridge he remained dedicated to helping me through, even though he didn't have to, and even though I've taken longer than any of his many PhD students. Thank you, Jon. Also thanks to Stephen Hailes for his wise advice once he took over as my first supervisor, and to Saleem Bhatti and Mark Handley, my second supervisors. And thank you to my examiners, David Clark and Frank Kelly, for their thorough review comments.

I acknowledge the funding and support of BT through Alan Steventon's Steve Wright's and Jonathan Legh-Smith's Strategic Research Programme, and the support of my managers over the duration of this research, Steve Sim and Peter Hovell. I also acknowledge the personal support of BT's CTO Matt Bross.

Specific acknowledgements have been made in each supporting publication, others by reference. Where appropriate, specific contributions have also been identified in the text.

Sébastien Cazalet and Andrea Soppera contributed to the original invention of re-feedback, and the following people have help develop it over the years: Andrea Soppera, Arnaud Jacquet, Toby Moncaster, Carla Di Cairano-Gilfedder, Alessandro Salvatori, Alan Smith, Louise Burness and Martin Koyabe. The simulations in this dissertation were produced by Carla Di Cairano Gilfedder, Alessandro Salvatori and Toby Moncaster.

All the following have given helpful comments: David Songhurst, Ben Strulo, Phil Eardley, Peter Hovell, Gabriele Corliano, Steve Rudkin, Marc Wennink, Nigel Walker, Fabrice Saffre, Cefn Hoile, Steve Wright, Don Clarke, Keith Briggs, John Davey, Nigel Geffen, Pete Willis, John Adams (BT), Sally Floyd, Scott Shenker (ICIR), Joe Babiarz, Kwok Ho-Chan (Nortel), Stephen Hailes, Mark Handley, Adam Greenhalgh, Brad Karp (UCL), David Clark, Bill Lehr, Steve Bauer, Payman Faratin, Sharon Gillett, Liz Maida (MIT) Frank Kelly (Uni Cam) and comments from participants in the CRN/CFP Broadband and DoS-resistant Internet working groups.

These acknowledgements probably miss someone who has helped. I apologise. And finally, I alone am responsible for all the mistakes, overstated claims and any comments about the work of others that may offend.

Contents

List of Figures

List of Tables

Part I

Freedom with Accountability for Causing Congestion in a Connectionless Internetwork

Chapter 1

Introduction

1.1 The Problem

This research concerns the introduction of a resource consumption metric for the datagram internetworking layer, intended to improve the current Internet architecture. The end-to-end design principle [SRC84] advises that removing unnecessarily specific functions is as important as deciding which generic functions to include—necessary but sufficient. Chosen correctly, the internetwork layer should allow communications systems to be built around it that can evolve to meet unforeseen requirements without undue complexity.

This thinking has resulted in the characteristic rudimentary network layer of the Internet that solely delivers datagrams to their destination address. It allows every end-point the freedom to communicate in any way it wants with any other end-point, using any amount of the resource pool in between. But giving all end-points such freedom allows them to conflict with the freedom of others, wherever the capacity of particular resources is insufficient for the total load focused on it.

The problem we address is to include sufficient mechanism in the network layer to transmit a trustworthy resource consumption metric, but no more than the minimum necessary to allow higher layer mechanisms to resolve resource conflicts with a wide range of resource sharing approaches.

A large part of the contribution of this research has been to identify the precise sub-problems that need solving towards this end. Therefore the following problem description becomes a sequence of continually refined sub-problem statements. Perhaps more importantly, in the process it also identifies (non-)problems that were only on the generally accepted research agenda due to unsound reasoning—they were actually huge distractions.

The current Internet architecture allows every data source the freedom to choose whatever sending rate it requires, irrespective of the congestion it may cause. Most application authors choose to use the Internet through the TCP library, which behaves very sociably by reducing its sending rate in response to congestion [Jac88]. However, applications can choose not to reduce their rate in response to congestion, some because they cannot function below a minimum rate (e.g. interactive streaming media) and others through deliberate malice (e.g. flooding attacks). If these applications compete with TCP sources their careless, selfish or malicious behaviour is rewarded further by the TCP sources, which try even harder to alleviate congestion as long as competing sources continue their aggression.

However, even if every application used TCP, or was at least TCP-friendly [FHPW00] (i.e. using roughly the same average bit-rate as a TCP source under similar conditions), although congestion collapse would be avoided, there would be no control over whether resource sharing conflicts were reconciled. TCP certainly provides a safe dynamic (second order) response to congestion, but it is a fallacy that the shares of resources (first order) that TCP allocates are in some way special. That cannot be true because it depends on how much data different users ask TCP to transfer, and how many instances of TCP they use to do it [Bri07b]. A transport protocol alone cannot and should not be expected to share resources fairly, in any sense of the word [Bri08d].

Further, it would be a mistake to solve the problem of resource conflicts by forcing every individual application to respond to congestion in a certain way. Curtailing the freedom that an application has to choose whatever sending rate it needs would limit the space for future innovation, stunting the growth of new (and existing) applications such as networked games, flurries of transactional messages or just faster than normal file downloads. Ideally we need to allow freedom within some wider bounds that encourage a generally sociable long-term and short-term sharing of resources, but with allowance for considerable give and take [GK99b].

A more fruitful approach is to view the problem as a need for accountability. We want every application to have the freedom to choose whatever rate is necessary with whatever dynamics. But, where this can restrict the freedom of others, networks need to at least be able to hold users accountable for the consequences of their actions. But, even if some networks don't hold users accountable precisely for the congestion they cause, but may-be for some poor approximation like volume, and even if some networks don't use accountability at all, then the whole system must still work for those who do care about accountability.

Accountability for resource usage was on the original 1988 agenda of requirements for the Internet protocols [Cla88], albeit last of seven in priority order. It was still an unmet requirement in the list for a new Internet architecture (NewArch) in 2000 [BCSW00], though framed as a need for a capacity allocation capability.

Only having solved the problem, do we now truly understand that the ability for networks to associate traffic with the sending user's account is neither a necessary nor sufficient form of accountability for internetworking. Firstly, the problem is one of accountability *for causing congestion*, as traffic itself is not a problem to anyone unless it contributes to congestion.

This is because the minimal accountability necessary for datagram forwarding should concern *cost* of usage. Certainly the Internet architecture should not help reveal other economic information such as consumer value. Consumers try to keep their valuation private and providers try to capture it, so it would be wrong to pre-judge the outcome of this tussle at such a low layer in the architecture. However, if the architecture doesn't reveal true usage costs, no mechanism can ensure that the cost to the consumer tends downwards over time towards the cost of provision. Cost minimisation is a generally accepted goal of all modern societies whatever mechanism is chosen, whether by encouraging competitive markets or by regulating uncompetitive markets, or even by centralised national economic planning.

The marginal cost of usage of a network resource depends entirely on the extra congestion due to the presence of the traffic. So the architecture should reveal congestion.

From the early days of the Internet, end-points were responsible for detecting and controlling congestion. Therefore, end-points place all the information they need for detecting congestion (sequence numbers) in the end-to-end transport layer. But there is no way for all the end-points to co-ordinate themselves sufficiently to hold each end-point accountable for the costs it causes to others, let alone for them to enforce any desired preventative action. Only the operator of a forwarding device can be in the natural position to do either.[1] But networks cannot see this congestion information unless end-points allow them to. It is hard for networks to measure this information reliably, because a gap in a sequence might simply imply a few packets went over a different path. And anyway, if networks did use this information against the interests of end-points, end-points could just encrypt it, or just not send it at all.

As well as which costs to consider, we have to consider who needs to associate the costs with whom. The minimum sufficient accountability requires a forwarding device to be able to associate the expected marginal costs of traffic with the entity directly causing the costs. Although the costs are originally caused by the data sender, each forwarding device directly assists in causing the costs. We now realise that it removes considerable complexity if the congestion is associated with each packet, rather than with the original sender. Then, minimally, accountability can be localised to any trust boundary across which packet traffic flows.

The advantage of making the packet, not just the sender, accountable is that we can then make each party along the forwarding path accountable for forwarding the packet across each trust boundary, localising accountability and enabling aggregation. As the packet crosses each trust boundary, the party on the receiving side can associate the costs in the packet with the party on the sending side of the boundary. Thus, a network forwarding packets on behalf of the sender can be held accountable for *allowing* the sender to cause congestion.

Localisation of accountability avoids any need for globally meaningful identities. Specifically, the validity of the sender's address in the network layer packet header becomes irrelevant for resource accountability. For wireline links this means accountability need only depend ultimately on the security of local physical connectivity.[2] For wireless links and for many virtualised wireline links, accountability will usually also have to depend on identifiers or authentication in link headers, but these need only be trusted local to the link.

The main omission prior to our research was that datagrams could only be held to account for the congestion they caused after the fact—once actual congestion had happened—because datagram transfer is inherently one-way or simplex. Instead, we ensure the sending or forwarding party can be held accountable for its *expectation* of how much congestion it will cause on the rest of the path. Then causes of excessive expected congestion can be curtailed. The problem then becomes one of ensuring that expected congestion is declared honestly, which is the subject of this dissertation.

[1] The operator might also be a consumer, as in an ad hoc or peer-to-peer network.

[2] The term wireline scales to 'wires' at a microscopic level, including data flows crossing process ownership boundaries within virtualised machines (e.g. multi-sender hosts or virtual routers).

We now understand that the problem is how to hold each self-contained datagram accountable for the congestion it expects to cause. Which leads us to have to solve the problem of how to update a packet as it traverses a network, so it always declares the congestion it is likely to cause, but only the likely congestion over the *remainder* of its journey. Laskowski & Chuang have also identified exactly this need to be able to monitor 'rest-of-path' congestion, delay etc. as the major cause of the Internet's economic problems [LC06][3].

By requiring the upstream entity to form an expectation of downstream congestion, it becomes in their interest to monitor recent downstream congestion by soliciting timely feedback. However, it would have been wrong to make feedback a necessary condition for using the Internet—a datagram must be sufficiently self-contained to be delivered alone. So we must not require end-points to depend on feedback about congestion from previous datagrams (although they can use it if they have it). Instead, an upstream entity can simply be conservative in its expectation of congestion if it chooses not to gather feedback (also essential for starting or re-starting a data flow before feedback is available).

Finally, congestion is of course caused by either excess traffic or insufficient capacity. We have so far focused on accountability for sending traffic, not for insufficiently supply capacity—dropping traffic. We believe it would be misguided to try to build a mechanism for networks to be accountable to data senders for specific instances of congestion [AMI+07, LC06]. It is sufficient for network N_B to be accountable to its upstream neighbour N_A both for any congestion within its own network and congestion in downstream networks it chooses to route through. This accountability takes a simple form. If N_B provides N_A with more expensive, more congested paths than other networks, N_A can choose not to use N_B's service. N_A can just not route via N_B, on a path-by-path basis if necessary.[4] So again, the problem is to ensure packets carry downstream congestion information. Then not only can N_B hold N_A accountable for forwarding traffic that causes congestion, but N_A can hold N_B accountable for not having provisioned sufficiently. Again, the problem is that packet networks lack visibility of downstream congestion information.

1.2 Motivation

The problem of improving the sufficiency of datagrams without sacrificing simplicity is an important scientific and engineering endeavour in its own right. But considerable social and economic problems are also at stake.

Firstly, if used as intended, the current Internet architecture allows resource allocations to become extremely sub-optimal relative to the social welfare maximisation that a perfect market would produce. Proving this is not part of the current research. But the intuition has been given above, and the author's complementary work (with co-authors) gives worked examples of how bad resource allocations can be for typical uses of the current Internet [Bri08d]. We also try to quantify the problem a little below. We

[3] In a paper published in SIGCOMM'06, articulating the problem we had provided a solution to in the same conference the year before [BJCG+05].

[4] There is an important exception where the terminating network has a monopoly on routes to the destination, which is also part of the problem we address.

can certainly say that current resource allocations are not just slightly out, but hopelessly unlike they would be if a market were allocating resources.

Unconstrained resource sharing can be beneficial in small doses, but if allowed to predominate it can stagnate market growth. If applications that want higher bit rate can help themselves without being held to account during congestion, they can effectively free-ride at the expense of other people's service impairment. Communications infrastructure, particularly the access edges of a network, requires huge levels of investment many years in advance. If free-riding predominates, the risk of investment in new infrastructure becomes too high, because there is no expectation that those most benefiting from the investment can be made to pay the returns on that investment (and usually no-one else will unless Government backing is provided). A downward spiral of declining quality and declining investment results [Gro05].

But there is considerable evidence that investment in networks is not declining. Rather than allow their network to descend into this is spiral, unsurprisingly, ISPs have found other ways to prevent the worst effects of free-riding. With no formal architectural support against free-riding, they have resorted to a hotch-potch of locally invented attempts at improvement.

This is what is happening on the Internet. It is now very common for ISPs to deploy deep packet inspection (DPI) boxes to effectively fight TCP's resource allocations. ISPs identify the application within the payload of each packet flow and throttle those that they *infer* have low value. This violates the Internet architecture. But they are trying to improve their competitive position by pleasing more of their customers more of the time, without spending excessively on capacity. They have to violate the architecture for their businesses to remain viable.

Their 'need' to violate the architecture causes unintended consequences. Those application developers most likely to be hit by throttling are obfuscating their application traffic. Many ISPs are already starting to suspect any encrypted and unidentifiable payload. Anecdotally, there is already some evidence that some applications under threat are starting to imitate the characteristics of other 'business-critical' encrypted traffic. This could lead the ISPs to throttle all unidentifiable traffic or to consider making customers seek permission to send it (possibly for a fee).

Many people don't like companies taking control of their choices, and even those who don't care get angry when ISPs infer their values wrongly. Some ISPs have a vested interest in disadvantaging certain applications or competitor services. So even if an ISP's intentions are honourable (throttling heavy users in the interests of the majority), discriminating against certain packets can be confusable with anti-competitive practice. In the US over the last three years, this has led to politicians getting involved in the details of Internet resource allocation, at which point the possibilities for further unintended consequences expand, and the chance of rational scientific debate worsens.

But what justification is there for saying Internet resource allocation has become extremely sub-optimal? If one considers that a weight could be associated with every data flow (as in weighted proportional fairness [Kel97b]), then the predominance of TCP can loosely be considered as a special case with all the weights set to one. If instead everyone was free to choose their weight, constrained only by

accountability for the congestion they caused, Kelly shows everyone would maximise aggregate social welfare by setting the weights based on their willingness to pay for the bit-rate of each application.

Market studies[5] have shown that value per bit covers a spectrum of about ten orders of magnitude, from messaging (SMS, IM) at the extreme high end to software & video downloads at the bottom, with interactive voice, Web, email, interactive video and music downloads between. Assuming the value of transferring a bit is related to the value of the bit itself, this shows that optimal weights would probably range over many orders of magnitude, so setting all the weights to one is likely to be extremely sub-optimal. Even worse, bulk transfers (a large proportion of traffic on the Internet) would probably be given a very low weight, if users were accountable. But they are currently often given a weight considerably greater than one (by the programmer opening multiple instances of TCP).

Unfortunately, bulk transfers least need a high weight. Even without considering economics, weighting small jobs is the classic way to optimise completion times in scheduling problems with a mix of job sizes [Kle76]. But if the utility of completion times is also considered, when small jobs tend to carry higher utility per unit size, weighted solutions are even more powerful.

If there were accountability for congestion, higher weight would generally be assigned to brief intermittent flows (i.e. flows of fewer bytes interspersed by periods of inactivity) because the extra cost would be easier to sustain over lower activity factors than in larger flows with higher activity factor. And if small data flows go faster they finish sooner, leaving as much capacity on average for the bigger flows over time (modulo inefficiencies due to the greater dynamic range).

1.2.1 Other Motivations

Simpler Quality of Service. Quality of Service (QoS) mechanisms, whether per-session (e.g. Intserv [BCS94]) or bulk (e.g. Diffserv [BBC+98]), have foundered once inter-domain deployment has been attempted (for years they just foundered, full-stop). There seem to be two main problems. One is at the API, the other is the need for considerable operational baggage between networks; to scalably authorise and authenticate, to provision, to monitor contracts and so forth.

The API to QoS seems to be problematic because applications can only ask for something the network knows how to offer, which often isn't really what they want (which in turn would be too complicated to express or even know clearly at design time). The industry has trained its customers to say they want bit-rate, burst size and so forth. But applications (and humans) aren't like that. Once application demands are aggregated, it starts to become easier to express what is wanted, especially in terms of expectations rather than quantitative assurances [Cla95]. But this still leaves an API gap between the application and the aggregated part of the network.

It should be fruitful to look at these QoS problems in a different light. As long as an application is given early warning of impending congestion somewhere on its path, e.g. with explicit congestion notification (ECN [Flo94]), it can take QoS for itself by just not responding as much to approaching congestion as it would otherwise.

Seen like this, the QoS problem becomes one of accountability for causing congestion anywhere

[5] Unfortunately not citable.

on the path. The problem is then not what the application can do—it can always do almost anything. The problem is what the network provider will allow the application to do and how to stop it exceeding these bounds. This becomes a lot easier if each network on the path can see the same information about congestion as the customer's machines. The network directly attached to the consumer can then set limits to the behaviour of the customer as a whole site or household and it can enforce them (or charge for exceeding them). And networks further downstream can do the same recursively against their upstream neighbour networks.

Thus, instead of arranging packets to carry QoS requests to distant networks, the problem can be seen as getting packets to carry congestion information from distant networks to the local one. Importantly, this removes any need to place significance on identifiers in packets.

This approach alone would not be expected to give QoS with strong assurances[6], but it might allow a wide range of expectations to be met without applications having to translate what they think they want into a language that doesn't have the right vocabulary.

The interface between end-point and network or between two networks would be so simple, one could hardly call it a QoS API any more. Only incipient congestion (ECN) information would need to pass across it. But the congestion information would have the additional semantic of cost—for the application to trade off against the benefit it will get from continuing to send bits.

Mitigating Bandwidth Flooding. Mitigating distributed denial of service (DDoS) attacks is another motivation for this research. The security community generally hasn't considered bandwidth flooding as a congestion accountability problem.

But, instead of the victim trying to find where attack packets are coming from, the problem can be seen as ensuring packets reveal expected congestion as they leave the sender. Then the packets headed for a flooding attack should be very obvious to the networks on the way. A high rate stream of packets heading for close to 100% congestion should stand out from everything else, as it would be very unlikely to be a genuine application. And source and networks alike could be held accountable for the congestion cost of the attack, creating strong incentives to remove it [Bri06].

1.3 Road map

The dissertation is in four parts. This first part has explained why freedom with accountability for causing congestion is important for the Internet. It now continues in more depth by surveying the seminal literature in this field followed by the main criticisms of the state of the art that motivated the present research to fill the gaps. With the background to the field explained, we then end this first part by stating the two hypotheses that focus the rest of the dissertation.

It will be more meaningful to give an outline of the approach used in the rest of the dissertation at the end of Part I, in §3.3 after the hypotheses have been introduced.

[6] It can in an edge-to-edge rather than end-to-end architecture [Ear09b], but the API gap opens up again in this case.

Chapter 2

Related Work

In retrospect, reading and thinking deeply about just the following ten or so papers would have been sufficient background for this research. Of course, other sources (extensively referenced throughout this dissertation and in supporting publications) provided necessary background understanding and ideas, as well as many false trails.

2.1 Internet Congestion Control

TCP: In 1988, Jacobson published "*Congestion Avoidance and Control*" [Jac88] to document the collection of algorithms he had implemented to provide congestion control for the transmission control protocol (TCP). Bravely, this was a wholly distributed protocol in which all aspects of resource control—efficiency, stability and fairness—were governed by the collective action of the computers comprising the Internet. Without it, or something like it, it is unlikely the Internet would have ever become widely used. TCP congestion control was produced in response to repeated congestive collapses of the whole Internet in 1986 and 1987. Router-based alternatives were being actively pursued, but Jacobson's distributed solution was such an astonishing improvement on the previous TCP that it was immediately deployed on all 30,000 or so Internet hosts, and has remained the Internet's predominant resource control mechanism to this day.

A colleague[1] recently collected results from 16 traffic characterisation studies conducted at different parts of the Internet (campus, residential and WLAN) between Jan 2003 and May 2006 in an unpublished survey. The proportion of TCP bytes measured in each study clusters around two percentages, 80% and 92%, with a clear mode of 94% Internet bytes controlled by TCP. Two outlier studies found 72% and 98% respectively. There is no significant trend up or down over the years.

Most academic focus has been on the additive increase multiplicative decrease algorithm that TCP's congestion avoidance phase borrowed from Jain *et al* [JRC87]. But probably Jacobson's most important contribution was the balance between the parameters of the initial 'slow-start' phase and the following congestion avoidance phase, which he justified with self-confessed 'hand-waving' in the paper. Internet traffic has a heavy-tailed flow-size distribution, so large numbers of flows either never reach congestion avoidance, or at least send the majority of their bytes in slow start phase. Slow start phase is a tricky

[1] Swadesh Samanta.

period for a flow as it quickly tries to find a fair operating point alongside other traffic. But the majority of bytes (not flows) in all the other traffic are in congestion avoidance phase. So the long flows must react fast enough to losses to allow in brief flows, then they must quickly converge on the new operating point together, then, as the brief flow finishes, the long flow must be able to quickly use up the freed capacity.

ECN: In 1994, Floyd published "*TCP and Explicit Congestion Notification*" (ECN) [Flo94]. It proposed a new field in the Internet protocol (IP) header, which finally reached the first 'Proposed Standard' stage of the Internet Engineering Task Force's (IETF's) standards track nearly seven years later, in 2001 [RFB01].

Prior to ECN, a queue experiencing congestion would discard some packets, then Internet congestion controls like TCP would detect the lost packets as gaps in the sequence numbers of the packet stream. The idea of ECN was to use an explicit marking on packets to indicate the onset of congestion, to try to keep the network at an operating point just below where losses started to be experienced. There is always a possibility that a gap in a packet sequence is merely a symptom of re-ordering, so a transport protocol waits for stronger evidence of a loss (further packet arrivals without filling the gap, or ultimately a timeout) before deciding congestion has really been experienced and slowing its rate. This delay due to uncertainty (which ECN solves) has a disproportionately detrimental effect on the performance of short transfers.

The reason ECN is important to the present research is an unintended but necessary side-effect of its introduction. It makes congestion visible to network devices downstream of the congested link, whereas any discards of packets by upstream devices are difficult if not impossible to monitor within the network. This is because there is no need for a sequence number space at the IP layer. So if the transport or higher layers choose not to reveal their sequence numbers (e.g. by encrypting them), the network cannot detect a gap in them. And even if they are not encrypted, a network element doesn't know whether gaps are due to re-routes or congestion. Readability of the ECN field at the IP layer is a fortunate side-effect of the need for writability of the field at the network layer.

In outline, ECN works as follows. As an ECN-enabled queue in the network starts to grow, it sets the new ECN field to a codepoint termed congestion experienced (CE), with increasing probability the longer the queue. Whenever a CE mark arrives at the receiver it notifies the sender, which can quickly and unambiguously know that congestion has been experienced. The sender is then meant to reduce its rate as if it had detected a drop (e.g. in its congestion avoidance phase TCP would halve its window).

Despite the mention of TCP in the title both of the research paper and the proposed standard, ECN was a change to the network layer's notification of congestion, which then requires any higher layer transport protocol, not just TCP, to be updated in order to understand it. TCP was merely the first transport protocol to be adapted to the new IP. This required some careful attention to backward compatibility to avoid using ECN to signal congestion to legacy transports that only understood loss as a sign of congestion.

Specifically, prior to ECN, the two bits of the ECN field had (nearly) always been left containing `00`

(now termed Not-ECT, a non-ECN-capable transport). So, for packets with the ECN field cleared to zero, even if a queue is ECN-enabled it must use drop rather than ECN to notify congestion. Also a sender-receiver pair must not use ECN unless they have established that they are both capable of understanding it, typically in the capability negotiation during the initial handshaking to start a flow. Then the sender must set the ECN field in every data packet to a non-zero value[2] to indicate to the network that the transport understands ECN (termed ECN-capable transport or ECT).

2.2 Economics of Network Congestion

Two-part congestion pricing: MacKie-Mason & Varian's "*Pricing Congestible Network Resources*" [MMV95] summarises their research in this field. It examines the tension between recovering the cost of capacity through a flat charge or through a variable usage dependent charge. It considers a range of providers available to a user, all buying capacity K [b/s] at cost $c(K)$ [¤/s].[3] It hypothesises a two-part tariff offered to customers i with a fixed subscription price q [¤/s] and a variable usage price p [¤/b] for usage x_i [b/s], such that the rate of charge [¤/s] is $q + px_i$. It considers what choice of p & q would maximise social welfare under a centrally planned economy or under competitive or monopolistic markets. The monopoly case will be set aside in this summary.

Providers vary their prices to maximise profits. Users switch between providers until the price-quality balance suits them. Quality degrades when usage starts to exceed capacity. As a result, both the non-monopoly cases arrive at the same analytical result

$$\frac{\text{usage revenue}}{\text{capacity cost}} = \frac{px_i}{c(K)} = \frac{1}{e},$$

where e is the elasticity of scale of the capacity. Elasticity of scale is solely a property of the shape of the function giving the cost of capacity at the current capacity operating point,

$$e = \frac{\text{average cost}}{\text{marginal cost}} = \frac{c(K)}{K}\frac{1}{c'(K)}.$$

For the present research, the actual result isn't so important as the order of magnitude it implies. Typical elasticities of scale for transmission equipment are of the order of 2 and they approach linear (i.e. 1) as capacity approaches technology limits (unfortunately figures are all from privately published studies of equipment costs, e.g. Lechner [Lec99] and those of Reid). So the usage element of revenue should be about 50% of total revenue—and probably increasing, given no significant cost-saving disruptions in mass transmission technology are on the horizon. This implies that, for the foreseeable future, there will be a significant element of usage pricing in competitive Internet markets, because it holds strong competitive advantage against flat pricing. Note that usage pricing schemes that roughly approximate congestion pricing could be sufficient, such as volume caps at tiered but otherwise flat prices, or with volume limiting at peak periods.[4]

[2] It should use `01` or `10`, but it can also use `11` even though it shouldn't.

[3] The paper concerned usage of general congestible resources. Applying it to specific scenarios like networking was not always natural. So, for our application of the theory to networking, units have been included in brackets to add dimensional precision. ¤ is the symbol for non-specific currency.

[4] To give a current data point, BT's 'up to 8Mb' DSL broadband pricing at Feb 2008 consists of a fixed charge of £4/month and

This formulation also clearly shows that, in a competitive market, congestion pricing will not add to an average customer's charge, rather it will substitute some part of the fixed element with a variable element.

MacKie-Mason & Varian's work also contributed the idea of shadow prices for congestion—borrowed from the classic economics literature and applied to computing and networking problems. The congestion that others experience is a negative side-effect of an individual's usage of a network (a negative externality). Shadow pricing makes an individual internalise this congestion externality. So shadow pricing is a powerful technique for dividing up the Internet's resource allocation problem across all users.

Utility functions: In the same year, Shenker published "*Fundamental Design Issues for the Future Internet*" [She95], which posited that people's utility U for bit-rate x always satiates at high bit-rates, ($\frac{\partial^2 U}{\partial x^2} \leq 0; x \to \infty$) and that utility curves fall into two main classes: elastic and inelastic, being concave ($\frac{\partial^2 U}{\partial x^2} \leq 0$) and convex-concave (sigmoid) respectively. As load increases through a capacity bottleneck, this implies there is no limit to how small a share each user of an elastic application will find useful. But for inelastic applications, there will come a point where higher value for all will result if some users have zero capacity as their share drops below the knee of the sigmoid.

If Shenker's hypothesis is correct, it implies that variable-rate congestion control suits elastic applications, but admission control is preferable for inelastic applications. Shenker also pointed out that typical bit-rate reservation systems of the time were designed as if people's utility was a step function of bit rate, which could be considered as an approximation of a sigmoid. Whereas rate-adaptive codecs would give a better approximation to a more gradually inclined utility curve.

These classes of utility curve had no experimental basis. But, since, we have validated that video utility curves are indeed sigmoid with a wide shallow sloped 'step'. We used carefully designed experiments with users paying real money, but unfortunately the results are only accessible to partners in the M3I project, given they reveal price sensitivity information [HE02]. It is possible that all utility is strictly sigmoid because, to our knowledge, the existence of elastic utility right down to zero bit rate remains unproven. Nonetheless, as long as a network rarely gets so congested that bit rates fall below the knee of typical users' utility curves, it is not cost-effective to introduce the admission control mechanisms for all traffic that some still argue for [MR99, MPCC00]—it is easier to treat the traffic as effectively elastic, which certainly leads to sub-optimal total utility during rare overload episodes, but the total loss of utility over time is probably smaller than the extra it would cost to deploy and operate an admission control mechanism.[5]

---

a variable charge of either £5, £10 or £15/month. Reverse engineering this, the lower two tariffs equate to about £1/GB of volume irrespective of congestion, while the upper, so-called 'unlimited' tariff limits heavy volume users during peak period congestion. Given the fixed element has to cover non-capacity costs as well, these figures imply BT is trying to cover about 80% of its capacity costs from usage revenues. If BT's pricing is rational and if Mackie-Mason & Varian's analysis is broadly correct, this imputes an elasticity of scale figure of perhaps 1.2 for BT's network. Or equivalently BT's network cost, $c(K) \propto K^{0.8}$.

[5] As long as congestion controls handle extreme congestion safely (e.g. TCP's exponential back-off).

Kelly: In 1998, Kelly and others published "*Rate control for communication networks: shadow prices, proportional fairness and stability*" [KMT98], which made advances on many fronts and brought all the previously mentioned research together[6]:

- It applied MacKie-Mason & Varian's shadow pricing to a network, rather than just a single resource. It proved that, where elastic applications compete for bandwidth, the total welfare of everyone using the network can be maximised if the network charges each pair of end-points a shadow price p dependent on the sum of congestion they cause on the path between them.

- It added models of each queue's pricing algorithm and each end-point's rate control algorithm, albeit abstracted and fluid.

- It proved that, given shadow pricing, if application users were modelled with a private willingness to pay per unit time w_i, and private elastic utility modelled by $U_i = w_i \ln x_i$, purely out of self-interest they would have the incentive to weight the rate of their private congestion control algorithm in proportion to their willingness to pay. Specifically, user i would have an incentive to control her rate x_i to converge on $\frac{w_i}{p}$. Kelly proposed users could do this with an equation-based additive increase multiplicative decrease algorithm of the form

$$\Delta x_i = \kappa(w_i - p x_i)\Delta t,$$

  where κ is a gain constant. Later Siris designed and implemented a window-based variant [SCM02].

- It emphasised how the proposed way to distribute the solution preserved the Internet's ability to allow new applications with new congestion control requirements to evolve. Gibbens & Kelly also restated this body of research in a more accessible paper that focused more on the evolvability aspect, "*Resource Pricing and the Evolution of Congestion Control*" [GK99b].

- It proved that such self-interested behaviour would preserve local and global network stability, as long as the gain parameter of everyone's rate control algorithms met certain constraints. Stability was proved assuming instantaneous feedback, but a number of papers later proved stability with propagation delays, each assuming various algorithms and constraints. They are reviewed in [Kel03]. In broad terms they showed the gain constant must be below a certain constraint, which must itself be inversely proportional to round trip time.

- It gave a simple mechanism to implement the proposed scheme, based on explicit congestion notification (ECN) that was in the process of standardisation into IP at the time (now proposed standard status [RFB01]). All an ISP had to do was count the number of bytes in packets arriving marked as having experienced congestion at the receiver and apply a fixed price per marked byte.

[6] Kelly and Voice extended the work to cover end-point congestion-based routing in 2005 [KV05], but the original work made the advances that are most relevant to our points.

Kelly's assumptions seem reasonable, although one continues to cause debate—not over its correctness, but over how soon it will come into play. Kelly uses the scaling arguments outlined in [Kel00, §2] to show that, whichever way that Internet scale increases in the future—whether more flows, longer flows or higher bit-rate flows—as long as scale does indeed continue to increase, congestion delays will become insignificant relative to propagation delays.

Kelly *et al*'s work raised a number of important questions about TCP's congestion control algorithms [Jac88], which dominate congestion control and resource sharing throughout the Internet.

- Firstly it introduced the possibility that the rate towards which a congestion control algorithm converges need not be limited by round trip time (RTT), as long as the algorithm's first order dynamics are limited within a constraint that is inversely proportional to RTT. For instance, with stationary congestion $\bar{p}$, the above rate control algorithm converges on $\bar{x} = \frac{w_i}{\bar{p}}$, which can be independent of the gain, κ and therefore independent of RTT[7]. More recently, FAST TCP [JWL04] has adopted a similar strategy.

- But, much more significantly, the *likely* values that self-interested users would set Kelly's weights to, given congestion pricing, would lead to extremely different capacity shares to those produced by TCP (see §1.2).

However, in the wider Internet community, the message that TCP probably leads to an extremely sub-optimal outcome got lost among the objections to Kelly's proposed means for evolving to the optimal outcome: dynamic congestion pricing.

Simple pricing: Odlyzko's paper "*A modest proposal for preventing Internet congestion*" is more well-known for its main subject, the Internet pricing proposal called Paris Metro Pricing[8] But it also contains a wealth of evidence from numerous other consumer sectors that consumers are highly averse to unpredictable pricing [Odl97, §5].[9] The section is entitled 'The irresistible force runs into the immovable object,' because it seems to be an unescapable fact that the irresistible economic logic of usage-sensitive pricing runs counter to the greater desire of consumers for pricing that is predictable and mentally undemanding. Consumers will pay a premium to not have to continuously work out how to pay less. As a result, as Odlyzko puts it, "…free enterprise companies prefer the socialist method of rationing by queue to that of rationing by price."

Congestion pricing preserves the complete freedom of application logic (under the control of the user) to change its mind at any instant—to increase or decrease spending without seeking the permission of the network. But, consumers must *also* be able to opt not to have complete freedom. Because along with total freedom comes risk—the risk that events outside the consumer's control (the discovery of some desirable information coinciding with high congestion) will tempt them into spending more than they would have wished, in hindsight.

[7] Otherwise an application's attempts to maximise utility can become confused if it doesn't compensate for RTT.

[8] Incidentally, PMP fails in a competitive market [GMS00].

[9] Earlier Barns [Bar89] had provided evidence for a desire for predictable network pricing from the defence sector.

The aim of the M3I project[10] was to produce an architecture that would enable Internet resource sharing to self-manage through a variety of pricing plans that would be able to evolve to take account of the tensions between these immovable consumer pricing preferences, their quality preferences and the irresistible logic of congestion pricing. My own summary of the projects results and their architectural implications, "*Market Managed Multi-service Internet (M3I): Architecture PtI; Principles*" [Bri02a] agreed with Odlyzko's two consumer preferences for pricing (predictable and undemanding) and added a third, transparency, in which the consumer wants to know that they are getting a known quantity of a well-understood good for a known price.

This M3I report includes a summary of how the different pricing scenarios enabled by the M3I architecture resolved all the conflicts between demand control, quality control and pricing preferences to varying extents. By the end of the M3I project, the tensions had been resolved for inelastic traffic at the expense of a little extra complexity at the network edge—a risk broker function between the user's access network and the core[11]. But the tensions remained not fully resolved for elastic traffic.

One could argue (as I did [BDT+00]) that a consumer can buy into congestion pricing but then synthesise her own flat rate pricing by mediating the risk of overspending with her own software agent that keeps congestion charges within a moving window. But, psychologically, this is still not the same as someone else sorting it all out for you. Getting Internet service at minimal cost just isn't important enough to most people who just want to pay a flat-fee and it works. Consequently, ISPs don't want to offer a pricing plan with a footnote saying "As you probably won't like this pricing plan, we also provide free software to make it acceptable."

Further reading: Costas Courcoubetis and Richard Weber, "Pricing Communication Networks" Wiley (2003) [CW03]

2.3 Internetwork Market Structure

Edge-pricing: In 1996, Shenker, Clark, Estrin and Herzog published "*Pricing in Computer Networks: Reshaping the research agenda*" [SCEH96]. It puts forward three main arguments, two of which are outlined here.[12]

Firstly it argues that there is a need to cover more than just marginal costs, so "It is important to allow prices to be based on some approximation of congestion costs, but it is important to not force them to be equal to these congestion costs." This was essentially a precursor to the principle that was better

[10] www.m3i-project.org

[11] The solution is currently being standardised in the IETF congestion and pre-congestion notification (PCN) working group [Ear09b].

[12] The second of the three arguments seems misguided in hindsight. It says that Internet service tries to be generic to all applications, so it is inherently impossible for the network to capture user utility for not having individual packets delayed or dropped, as required for congestion pricing schemes like MacKie-Mason & Varian's 'Smart Market' [MMV93] under discussion at the time. However, the point of the 'Smart Market' proposal is that utility can remain private but then the market mechanism effectively allows *users* (not the network) to sort all the demand into two sets, with utility either above or below the shadow price, in order to limit demand to supply. The argument was perhaps saying that users wouldn't be able to divide their utility down on a packet by packet basis anyway. But, this rather threw out the baby with the bath-water by eliminating the possibility that even a very rough approximation would be better than nothing.

articulated later in 'Tussle in Cyberspace': that researchers shouldn't try to dictate outcomes.

Lastly it argued that the form of pricing wrt. usage was only one aspect of pricing that needed research. Instead it argued that more attention should be given to how contractual relationships should be structured across an internetwork.

The main contribution was a description of a structure called edge-pricing. With edge-pricing, networks levy bulk fees on their neighbours (end-customers and other networks) that all taken together cover a network's costs and profits, but charges don't have to be levied on a flow-by-flow basis. The motivation of edge-pricing is to allow the forms of tariffs to be different on a pairwise basis between neighbours, encouraging evolution of tariffs structures, rather than having to embed a pricing scheme in the architecture.

Information asymmetry: In 2001 Constantiou and Courcoubetis published "*Information Asymmetry Models in the Internet Connectivity Market*" [CC01]. Although it is not a conclusive paper, in that it presents no solutions, it clarifies more precisely than other similar papers which information networks cannot see about the quality of other networks and why this is so corrosive to a successful communications value chain. More recently, Laskowski & Cheung [LC06] also highlighted the same information as the critical missing piece of the Internet, but they did not relate the problem to the economic literature on market failures due to information asymmetry.

When it comes to theoretical understanding of quality issues, basic economic theory is only just in front of the 'science' of computer communications. The detrimental effects of asymmetry of quality information were only first articulated in Akerlof's 1970 paper "*The Market for 'Lemons': Quality, Uncertainty and Market Mechanisms*" [Ake70], which led to him (with others) winning the 2001 Nobel Prize in Economics. Using the example of used car sales, it showed that the salesman's privately held knowledge of which cars were duds ('lemons') drove down the price for used cars across the whole market, because the willingness to pay of consumers would reduce once they took the average risk of buying a dud into account, even if the car in question turned out to be fine. The suppressed market price led in turn to a reduction in the incentive to supply.

One can think of a data sender, or a forwarding network, as contracting with a downstream[13] network to deliver packets. But with one-way datagram technology, the upstream network knows little about the downstream neighbours it contracts with, whereas they know their own traffic loading and distribution, available capacity, resource allocation policies, customer types and interconnection agreements. Similarly, the next network is in a similarly weak position relative to the one after.

Constantiou and Courcoubetis apply the Principal-Agent formulation to model the resulting situation. The Principal-Agent formulation has been developed in economics to model the position of the principal (upstream) and agent (downstream) parties to this contract. By attaching parameterised rewards to any measurable effort of the agent and any measurable outcome for the principal, it is possible to optimise the parameters to design a contract that minimises the negative effects of information asymmetry.

[13]Different fields use the term 'downstream' ambiguously. Communications engineering uses it to mean 'in the direction of data transmission'. In the field of industrial organisation, downstream can also have the sense of 'towards the retail end of a value chain', but that is not the intent here.

Alternatively, it is possible to predict the value of improved measurability of effort or results. As already stated, the paper is inconclusive, but it at least identifies the problem well.

Design for Tussle: Clark and others published "Tussle in Cyberspace: Defining Tomorrow's Internet" in 2002, followed by a clearer journal article in 2005 [CWSB05]. It argues that the architecture of the Internet should allow the major tensions in society and in economics to be resolved at run-time, not design time. It turns this principle into the slogan 'Design for Tussle'.

The M3I architecture mentioned earlier also espoused this principle (it was published in parallel), but Clark *et al* give a far better and more general articulation. The M3I discussion was more specific (but consequently somewhat more concrete), being based on specific examples where the Internet architecture should be changed.

The paper offers further specific design principles, one being particularly relevant here: 'Modularise along tussle boundaries'. In the context of the above two papers on edge-pricing and information asymmetry, one could interpret this as advice to ensure the intended advantage of edge-pricing (independent evolvability of each pair-wise contract) is indeed possible. And to ensure that quality information is visible to both networks at every border.

Towards the end, the paper revisits some of the old design principles of the Internet in the light of the new tussle-related principles. It tries to grapple with the tensions in the end-to-end principle [SRC84] (see §1.1). Although the discussion seems inconclusive, it concludes that "... end-to-end arguments are still valid and powerful, but need a more complex articulation in today's world." We will return to this below as we highlight the main outstanding deficiencies in all the works we have just introduced.

2.4 Critique of Existing Work

TCP: The most pernicious deficiency in existing work has been the false goal of approximately equal flow rates through a bottleneck. The idea that rate equality is a good approximation to 'fair' set in long before Jacobson adopted it for TCP (traceable at least back as far as ATM research in 1980 [Jaf80]), to the extent that he didn't even question it as a reasonable goal. The problem statement of §1.1 has already rehearsed the core arguments that instantaneous flow rate is the wrong metric to be concerned with for fairness, because a) fairness should be between users not flows and b) instantaneous flow rate doesn't take account of the proportion of time that a user (or flow for that matter) is inactive.

My recent paper "*Flow Rate Fairness: Dismantling a Religion*" [Bri07b] published in ACM CCR, is an attempt to explain why Kelly's work shows that flow rate equality through a bottleneck is a nonsensical fairness goal. It is aimed at an audience that requires implications to be spelled out bluntly and one that has an aversion to maths. It carefully builds a case to show that the idea of flow rate fairness is completely unsubstantiated dogma. In contrast Kelly's welfare maximisation is given as an example of a properly defined form of fairness built on the philosophical notion of commutative justice[14].

[14] In 350 B.C.E. Aristotle distinguished two types of justice, distributive and rectifactory (commutative) [Ari25, Book V Chapters 2, 4 & 5]. Distributive justice concerns whether a particular distribution of goods is just but has proved impossible to define convincingly (Rawls [Raw01] comes closest, but still requires all one's preconceptions to be set aside in order to judge a just distribution). Commutative justice concerns whether an action (e.g. a transfer of goods) is just, most often determined by whether

However, the paper's main message is that different forms of fairness should be possible to enforce locally[15], but this will only ever be possible if the Internet architecture as a whole supports the ability to make self-interested individuals or entities (including whole networks) accountable for the costs they cause (or allow to be caused) to others. It therefore advocates the metric of congestion-volume as a prerequisite for different forms of fairness to co-exist. Congestion-volume is defined as a count of all the bytes of dropped or congestion marked data sent by all an individual's flows i over a period of time, T:

$$\int_T \sum_{\forall i} p(t)x(t)_i dt,$$

where $x(t)_i$ is the bit-rate of flow i and $p(t)$ is the congestion it experiences.

"...Dismantling a Religion" was motivated by the extreme unfairness (defined per user and over time) that has resulted on the present Internet in the name of flow-rate equality. But it was particularly motivated by the continued use of friendliness to TCP as a goal for new congestion controls (such as TFRC [FHPW00], XCP [KHR02] and other new high speed congestion controls), which constrains the future solution space completely unnecessarily. Even though it was claimed that an XCP switch could implement different forms of fairness, "...Dismantling a Religion" explained that fairness is a property of the congestion a user causes in a whole network over time, which is not something each switch can ever hope to control by setting the relative rates of just the flows that happen to be passing through it at any particular instant.[16]

More recently, I have published an Internet draft (with others) for the IETF, "*Problem Statement: Transport Protocols Don't Have To Do Fairness*" [Bri08d] that justifies the assertion that there is extreme unfairness on the Internet, using numerical examples drawn from Internet measurements. It uses the evidence to argue that the IETF's protocol designs don't, can't and shouldn't have any control over fairness. But instead the IETF should concentrate on a protocol framework to allow fairness to be controlled at run-time (the message of 'Design for Tussle').

---

transfers are entered into voluntarily. It is alternatively termed rectifactory justice because a transfer of value (e.g. goods) in one direction that alters the balance of justice can be rectified by a transfer of value (e.g. money) in the other direction. Welfare maximisation is a result of a continuous sequence of transfers of value that are each commutatively just. If the original distribution of goods was not just (by whatever definition), a series of commutatively just transfers always improves everyone's lot in absolute terms, but it won't necessarily improve distributive justice (e.g. if defined relatively), even though progressive taxation is designed to attempt this. The only way to otherwise improve distributive justice is to somehow define a just distribution then forcibly take from the rich and give to the poor. However, further commutatively just transfers would again diverge from distributive justice, requiring continuous intervention.

[15] Both physically local and locally across a virtual grouping of users.

[16] XCP bears a superficial resemblance to re-feedback in that routers along the path decrement the change in flow rate requested in-band by the source, which is then fed back from receiver to source. However, XCP's structure is more analogous to a dynamic form of RSVP [ZDE+93]. The subtle but important difference from re-feedback is that XCP's metric quantifies the service rate (the primal variable), not the impairment introduced along the path (the dual). Even if the set of all the service rates is combined (e.g. at the customer's attachment point) nothing can be determined about whether that customer's use of the whole network is fair, because there is insufficient information about how much each flow impacts *other* users. In addition, in a non-co-operative setting, the service rate claimed in each XCP packet has to be policed at each border against the service actually provided, which requires per flow processing. This was the issue that killed the scalability of the Integrated Services architecture [BBB+97]. "...Dismantling a Religion" gives fuller discussion of these issues.

The draft accepts that some individuals aren't concerned if the Internet protocols aren't fair, so it aims to show that extreme unfairness leads to other highly detrimental concrete consequences. It uses further numerical examples to show how the inability to prevent free-riding in an architecture (extremely high allocations of congested resource for a minority of users who pay no more than others) leads to significantly higher investment risk. Because the majority will abandon a provider that continues to expect them to share the cost of its investments while receiving only a tiny share of the benefits. Using evidence that investment is still actually continuing, it explains this is because operators are throttling heavy users.

However, operators know that heavy *users* actually represent a mix of light and heavy *usage*. So rather than lose the heavy users' by limiting all their usage indiscriminately, operators are inspecting packet payloads and limiting only applications that they *infer* are causes of heavy congestion. Operators could limit overall traffic for heavy *users* and give them control over limiting their least valuable *usage*, but most users have neither the software nor the inclination to do this, so ISPs keep control themselves.

Users understandably get upset whenever their ISP's inferences are wrong. Also, however honourable the provider's intentions, their discriminatory throttling is easily confusable with anti-competitive discrimination against competitors' services, leading to the recent net neutrality debate.

The goal of flow-rate equality led to a large body of work on policing equal flow rates: Floyd and Fall's penalty box idea [FF99], Stabilized RED (SRED [OLW99]), CHOKe [PPP00], RED with Preference Dropping (RED-PD [MFW01]), Least Recently Used RED (LRU-RED [Red01]), XCHOKe [CCG+02], and Approx. Fair Dropping (AFD [PBPS03])). Because the goal of flow-rate equality is deficient, it has led these works to be triply deficient. Primarily because they are trying to police a flawed goal (per flow not per user, and instantaneous not over time). Secondly because it is easy for flows to circumvent any such policing using multiple flows on multiple paths. And thirdly because flows can simply whitewash their identifiers as soon as they are discovered, because there is no cost to creating new flow IDs.

ECN-based Congestion Pricing: Despite integrating together huge advances on many fronts, ultimately Kelly's work hit practicality problems for two entangled reasons: i) consumer aversion to dynamic congestion pricing and ii) dependence on the asymmetric structure of congestion information in the Internet. The entanglement was explained in "The case against classic feedback" in our main publication so far on re-feedback "*Policing Congestion Response in an Internetwork using Re-feedback*" [BJCG+05] (reproduced and updated slightly in §5.1.1 and outlined below.

Odlyzko's tension between the irresistible economic logic of usage-sensitive pricing and the immovable consumer desire for simpler pricing cannot be side-stepped. It must be possible for a network to ration demand by queue rather than by pricing—to slow down traffic causing congestion rather than delegate this responsibility to consumers under threat of higher charges. Of course any one user's ration will still be able to be sold at the correct congestion price. However, this will be simple, flat congestion pricing, not dynamic.

Only an ingress network, and preferably the first ingress device, can enforce congestion limiting.

But an ingress network cannot see the congestion being caused by the traffic entering the network, unless the congestion happens to be local. The classic feedback structure used by ECN, on which Kelly naturally built, cuts the upstream networks out of the loop. As explained above, ECN reveals information about congestion that was previously hidden, but not to networks upstream of the congestion. Certainly a feedback stream usually returns to the sender, but it is beyond the view of all the intervening networks—in higher layer end-to-end messages that may be encrypted, asymmetrically routed or simply omitted completely.

This is a highly unusual form of information asymmetry, where the buyer holds more information than the seller about the quality of the service. We believe it is this asymmetry that leads to all the Internet's problems of resource control economics. As we discussed in our summary of [Bri08d], this asymmetry can lead to heavily suppressed investment. This is the same outcome as for Akerlof's case where the seller holds better information about quality than the buyer. But the chain of logic is the converse to Akerlof's. Nonetheless it has the same underlying structure, in which the market price has to include a premium that averages the risk of uncertainty over each contract.

This unusual information asymmetry is solely because the Internet is simplex at the internetwork layer (one-way information flow).[17] Duplex networks don't seem to exhibit the same economic problems as the Internet because any network can see the quality of the paths into which it is sending traffic by monitoring the feedback returning along each connection and managing traffic accordingly (e.g. ATM traffic management [ITU04]).

Because only the current Internet's classic feedback arrangement was available to Kelly and co-workers, congestion pricing could not be turned into rationing by queue. It might be feasible to throttle traffic at the last egress of the internetwork, based on information emerging from upstream congestion, but only by rationing the congestion *received* by a host. However, this would be a rather odd deal for a consumer to accept as a receiver cannot stop sent traffic from entering the network, filling it with traffic and consuming the receiver's congestion ration.

Given the Internet's feedback structure, the only option available to Kelly was to charge the receiver for congestion received. Then, in order to transfer the correct incentives to the sender, the receiver had to ask the sender to reimburse its congestion costs. This would unfortunately open all receivers to 'denial of funds' attacks, as well as incurring extra transaction costs.

There is a further subtle issue with Kelly's form of congestion pricing. Kelly holds each pair of end-points accountable for *actual* congestion. Whereas MacKie-Mason & Varian's smart market proposal holds the sender accountable for her *bid* if and only if it is greater than the actual congestion price. In both schemes, the charge ends up the same. But the subtle distinction only becomes apparent by thinking at the scale of individual packets. In both schemes the sender only discovers the price after the packet is sent. But in the smart market the sender limits her exposure to the risk of a high price, and if the actual price is higher the packet is discarded—again, rationing by queue rather than by price, but at the

[17] The term connectionless is deliberately avoided because it has a slightly different meaning. For instance multi-protocol label switching (MPLS) is simplex (reverse connections are not associated with forward connections) but not connectionless (connection state is held on network elements).

microscopic scale.

A different potential problem also lurks within Kelly's approach (it actually stems from the placement of utility with respect to instantaneous bit-rate in Shenker [She95]). As Clark had pointed out in 1995 [Cla95] and Shenker *et al* had repeated [SCEH96], the utility of transfers of fixed volume objects will often depend on completion time not instantaneous bit-rate. In 1999, Key & Massoulié pointed out that the two are inversely related because completing earlier stops the congestion costs earlier [KM99]. Therefore, once congestion is above a threshold there seems to be an incentive to drive up bit-rate to the maximum possible. In these cases, Key & Massoulié seem to convincingly argue that there will be no incentive to continuously optimise instantaneous bit-rate against instantaneous congestion. If they are correct, Kelly's results would lose much of their significance, as file transfers with utility from completion time probably comprise a large proportion of elastic Internet traffic. However, Gibbens & Kelly's experiments [GK99b, §3] propose a strategy for optimising instantaneous bit-rate by adapting willingness to pay that does pay off for users transferring fixed volume files.

Despite the importance of file completion time as a metric of value being 'reinvented' recently [DM06], the implications have still not been fully worked out. However Key and others have proved that, in the presence of delays, self-interested rate control will still lead to stability as long as file transfer traffic is mixed sufficiently with other types [KMBK04].

An interesting question is whether it is myopic to solely consider each object transfer in isolation, or whether transferring each object faster necessarily leads to opportunities to transfer more objects. This would imply that fixed volume objects are part of a larger stream with an overall volume that expands with bit-rate, at least when viewed at sufficiently coarse granularity.

Edge-pricing: I developed Shenker *et al*'s edge-pricing further in "*The Direction of Value Flow in Open Multi-Service Connectionless Networks*" [Bri00], a technical report that collects together two previously published papers applied to unicast and multicast [Bri99b, Bri99a]. It questions the proposed close tie between edge-pricing and apportionment of costs between sender and receiver.

The whole reason apportionment of costs between sender and receiver is needed is because different pairs of each have different apportionments of value. If the apportionments of usage cost between sender and receiver are fixed by the network, there will often be cases where the sum of the value they both derive is greater than the sum of their costs, but the value that one alone derives is less than its fixed share of the cost. If the losing party cannot shift some of its charge to the other, the communication won't happen. In the language of industrial organisation, communications is a two-sided market, because at least two buyers are involved in each sale [FW06] (see also §12.1.2).

The Shenker edge-pricing paper argues discursively that edge-prices should embed the chosen apportionment of costs between senders and receivers, whereas "*The Direction of Value Flow...*" argues that different flows will want different apportionments between sender and receiver to match the apportionment of value each derives from the communication. "*The Direction of Value Flow...*" uses a model of internetwork pricing to show that embedding the apportionment of costs between senders and receivers solely in the edge-pricing at the network layer necessarily leads to flow-by-flow charging and

an Internet-wide pricing scheme—exactly what Shenker *et al* were trying to avoid.

"*The Direction of Value Flow...*" outlined an end-to-end clearing function to re-apportion charges between the end-points, where the difference in value apportionment from the default made the transaction cost worth it. The Shenker paper had rejected such a clearing function in a footnote.

"*The Direction of Value Flow...*" further allowed each edge price to be split down into a fixed and a variable charge, and allowed the usage charge to flow in a direction independent of the direction that fixed charges took. This model was termed split-edge pricing.

My later work, co-authored with Rudkin, "Commercial Models for IP Quality of Service Interconnect" [BR05] revisited this whole field in the light of developments like re-feedback. It also added some specific structure to Shenker *et al*'s first point (that charging should merely be based on marginal cost, not equal to it). It reasoned why we can predict that commoditisation to marginal cost will proceed faster in transit (non-access) networks, while access networks will retain a greater ability to extract profits. The reasoning was that, although end-users and software developers might be expected to drive all networking to marginal cost, many end-users do not choose to spend their time minimising their charges (Odlyzko's point again). However, access networks have the motivation and means to aggregate their knowledge of their user's demands but to hide this knowledge from transits. From the viewpoint of transit networks, access networks resemble end-users—recursively. But unlike end-users, access networks have the power of aggregation and the means to use it.

Information asymmetry: Constantiou and Courcoubetis, like other papers on accountability [AMI+07, LC06], put the problem in terms of *network* accountability. But, of course, congestion is the result of too much traffic meeting too little capacity, so it is a question of both network *and* sender accountability.[18]

However, any one source of the traffic is not wholly to blame, because they didn't necessarily know all the others would send at the same time. And the network is not wholly to blame either because traffic can adapt much more quickly to insufficient capacity than capacity can adapt to traffic.

So accountability in both directions needs to be solved. Kelly's work shows how to divide the blame among the traffic—by sharing out instantaneous congestion in proportion to instantaneous bit rate. This can then be integrated over time and each user's contribution can be summed over all queues in the network as in the formula for congestion volume earlier:

$$\int_T \sum_{\forall i} p(t)x(t)_i dt.$$

And, the same information should be used by a network N_A to hold its downstream neighbour N_B accountable for congestion within N_B or in networks beyond, that N_B has chosen to route through towards the destination (its subcontractors). In the short term, this congestion is caused by the decisions of networks like N_A to route their traffic through N_B. But if the congestion persists longer term it implies N_B is not sufficiently provisioning capacity, or it is making poor onward routing decisions into other networks that are insufficiently provisioned.

Tussle: 'Tussle:.' [CWSB05] identifies some of the symptoms of the economic tension within the end-to-end design principle [SRC84] that is central to the Internet's design. But it largely side-steps any

[18] And, as pointed out in §1.1, the problem is simplified further if it is viewed as a *traffic* accountability problem.

challenge to the fundamental economic tension between the principles of 'Design for Tussle' and of 'End-to-end Design'. We believe this tension cannot be fudged to one side with the words "…end-to-end design is still valid but needs a more complex articulation."

The end-to-end principle essentially mandates that the lion's share of the profits from the communications value chain should go to the computing sector. Whereas the message of 'Tussle' is that the Internet architecture should not prejudge the outcome of the continuing competition between the computing and communications sectors. And if the architecture does pre-judge this tussle, the communications sector will choose to serve its own interests rather than comply with the architecture, thus leading to a mess of badly interconnected patches without an architecture—the present reality of the Internet.

If the communications sector were driven to near-zero commodity profits too early, investment capital would move to other less liquid sectors. A sector that is still growing rapidly is, by definition, not a commodity sector. Notwithstanding Odlyzko's points about consumer preferences, shadow pricing of congestion is the end-game that commoditisation will drive towards. But, as well as allowing congestion control to evolve under congestion pricing [GK99b], we have to allow pricing to evolve too. Even if pricing will eventually collapse towards congestion pricing (including congestion limiting), along the way we must allow the market to experiment with other more profitable schemes and services. Consumers, not system designers, are meant to commoditise a market—when they are ready. System designers should merely ensure the architecture would *allow* a shift to commoditisation.

Even if the economics predicts that an outcome (commoditisation) seems inevitable, the architecture shouldn't prejudge how quickly the whole value chain will reach this outcome and it should be able to encompass the structures that might develop on the possibly long road to that outcome.

For instance, many telcos (particularly in the cellular sector) are still wanting to build service-oriented networks, to sell services bundled with basic networking. It might well be that the open Internet model will just steam over them as they hanker after the golden past when they could bundle everything together and lock-in their customers. But it's just as likely their mass customer base might buy into services built on service-oriented networks, then the cellular operators will have resisted the open Internet model. Considerable value would be released [BR05, BOT06] if fixed and cellular networks could converge more closely. Therefore, the lesson from "Tussle" should be that the Internet architecture must encompass service-oriented networks as well as open networks. It's not clear the authors of "Tussle" meant to go that far. But that certainly must be a goal of the present research.

2.5 Conclusions from Reviews

The literature reviewed above builds a picture of the multifaceted problem the present research aims to tackle. It can be pictured as an ancestry diagram in two cascades.

1. The ideas and deficiencies in TCP, ECN, two-part congestion pricing and bit-rate utility were all brought together into one solution by Kelly, along with Kelly's own considerable advances in network traffic modelling.

2. Then the present research brings together the ideas in Kelly with those in simple pricing, edge-

pricing, information asymmetry and tussle to identify and fix a deficiency in the feedback architecture of simplex networks.

Others either discard the power of Kelly's model because it doesn't give simple pricing, or those who identify the information asymmetry problem try to retrofit internal feedback loops within the Internet. Whereas the task we set ourselves is to keep simplex networks fully simplex end-to-end, but convolve necessary and sufficient feedback information into the forward path.

Chapter 3

Hypotheses

Hypothesis 1 (Congestion Signal Integrity). *The incentives of self-interested or malicious economic entities can be aligned to assure the integrity of indications of downstream congestion in the packets of a connectionless simplex internetwork. This can be achieved by only constraining aggregate downstream congestion-volume sent by each economic entity over time, without any dynamic congestion pricing to end-consumers, without any further constraints on transport behaviour and without any further constraints on the agents' freedom to distribute load across the internetwork, or across time.*

Hypothesis 2 (Welfare Maximising Allocation). *With a competitive market and under Assumptions 3.1 & 3.2, incentives of all parties can be aligned so that the system produces the welfare maximising allocation of resources, under all the conditions of Hypothesis 1.*

Assumption 3.1. *Each consumer's demand is small relative to aggregate load on each link.*

Assumption 3.2. *Consumer utility is for bit-rate and the internetwork operates within the concave region of everyone's utility curves or flow admission control prevents anyone operating outside their concave utility range.*

3.1 Clarifications

Can be aligned: An example scalable enforcement mechanism can be defined with acceptably low probability of false hits or false misses.

Scalable: Sub-linear complexity wrt. traffic and network topology characteristics (no. of flows, no. of networks, etc.)

Acceptably low false hits: Losses of the same order as existing losses;

Acceptably low false misses: Where attacks might be successful due to statistical variations, over time the cost of launching failed attacks must be greater than the gains from successful attacks;

Downstream congestion-volume: As defined in §6.2

$$\int \sum_{\forall i} v(t)x(t)_i dt;$$

Constraining aggregate congestion-volume over time: A congestion-volume allowance fed at a constant fill-rate into a bulk token bucket per data-sender by their access network operator, which prevents further congestion being caused below a certain level and also constrains the maximum consumption of allowance;

Sent by each economic entity: The sending end of possibly multiple applications on possibly multiple computers under the control of a single economic entity. It is assumed that economic entities behind the data-receivers may choose to share part of the data-senders' costs, and some data-senders may make sending data conditional on the receiver's contractual commitment to share costs.

Assumption 3.3. *Transaction costs between sender and receiver can be ignored.*

This assumption is invalid, but analysis of how much this higher layer issue affects the welfare maximisation and the mechanisms for sharing costs are left for further research;

Economic entity: A stakeholder with its own motivations, resources and capabilities including end-consumers and network providers (alternatively, economic agent or party);

Welfare maximising: Maximisation of the sum of the utilities of all economic entities.

The system: The combination of economic entities, the internetwork, rate control functions on host computers, congestion notification protocols, and the incentive mechanisms at the network's trust boundaries defined in this dissertation;

Connectionless simplex internetwork: A collection of network domains operated by autonomous economic entities, using only one-way self-contained end-to-end datagrams with no return channels for congestion feedback at the network layer (e.g. not in routing or congestion back-pressure messages).

Self-interested economic entity: Individuals or organisations operating with rational self-interest;

Malicious: Unbounded malice if the entity is an end-consumer or bounded malice if the entity is a network;

Bounded malice: Only willing to exploit amplifying traffic-related vulnerabilities, where the cost to the victim is strictly greater than the cost of the attack;

Unbounded malice: Willing to exploit any traffic-related vulnerabilities to cause harm to others.

Traffic-related vulnerabilities: Vulnerabilities in the re-ECN system, or in related Internet traffic control functions. Information security issues within the payload or pre-existing network security issues (e.g. routing vulnerabilities) are ruled out of scope;

Dynamic congestion pricing: Pricing proportional to congestion caused per bit;

End-consumers: The economic agents behind data-senders and data-receivers;

Further constraints: Constraints other than the aggregate constraint;

Transport behaviour: Increases and decreases in data-rate;

Distribute load across the internetwork: Send to any destination at any data rate;

Distribute load across time: Send less data now and more later or vice versa.

3.2 Significance and Rationale

The congestion signal integrity hypothesis is ambitious. Paraphrasing Popper [Pop63], safe conjectures are not interesting. For me, it is not as important to be correct as it is to be practical, as long as I'm practically correct so there is a possibility of making an impact.

Proving robustness against gaming is an ambitious and ultimately impossible goal, because one cannot know the set of all attacks that might be invented against it. However, one can create an abstract model of the solution, its incentive environment and its information flows, towards proving it has high likelihood of being robust against the attacks we know. This is believed to be a sufficient approach in computer science when proposing systems solutions to large, distributed problems.[1]

The welfare maximising hypothesis has been separated out from the integrity hypothesis. It follows fairly trivially if the first hypothesis holds, by straightforward connection to the arguments in Kelly [KMT98, KV05]. It was felt important to include a case close to the way resources are generally allocated in the world so as to link to the Internet resource allocation motivation for the work. However, it would have been wrong to tie the integrity hypothesis only to this single (albeit important) case. Congestion signal integrity is an architectural building block that would be useful for other ways of allocating resources than a market (reflecting the arguments of 'Design for Tussle').

The practicality conditions are the more challenging and interesting aspects of the integrity hypothesis. These conditions have been carefully chosen because they encapsulate a wider set of practicality constraints.

Dynamic congestion charging not required: We wanted to find a solution that did not present retail network providers with the dilemma of either having to offer an unpopular tariff or not being able to rely on their customers' natural incentives in order to share network resources fairly. We also wanted to avoid the idealistic assumption that players only act rationally, which many proofs of incentive compatibility require. So we replaced congestion pricing with engineering mechanisms that would allow networks to police their customers' responses to congestion whether they were rational or not. Enabling engineered policers, rather than relying on rationality, also protects a player against accidental misconfiguration of its own part of the system.

We know from the start that it is fruitless to align the incentives of some zealot with unbounded

[1] I have also (perhaps deliberately) engineered incentives for others to try to break my solution (by strongly criticising whole fields of other people's work, persistently claiming near-perfection in my own and challenging others to break it!). This has already led to a number of proposed attacks on re-feedback, which have helped my generalisation of possible attacks, and in some cases design changes have been necessary.

malice. However, we believe we may be able to prove our hypothesis if we require only the malice of networks to be bounded (§8.1.2), while we will allow everyone else's malice to have no bounds. This is an ambitious (and therefore interesting) attack model.

No further constraints on transport behaviour: This point aims to ensure accountability is not introduced at the expense of freedom—so that new applications with novel responses to congestion can emerge. Choosing to enforce accountability through network engineering rather than pricing would seem to imply that the congestion behaviours used by today's set of applications will become embedded within every network. But, by constraining our solution to avoid service standardisation between applications and network operators and between operators, we intend to show that new applications would be able to emerge without asking permission. Even if networks do put in certain behaviour constraints, these can be relaxed by bilateral agreement between a customer and the ingress network, without further standardisation effort across other networks downstream, which would effectively block any evolution.

Whether a system allows players the freedom to evolve is notoriously subjective and therefore both easy to prove loosely and difficult to prove conclusively. Whether service standardisation is necessary is perhaps only one aspect of evolvability, but it is at least a provable fact.

Scalability: This constraint aims to ensure that accountability is not introduced at the expense of poor network layer scaling with number of flows, users etc. in the sense used in complexity theory.

3.3 Approach

The vast majority of the rest of the dissertation is aimed at proving Hypothesis 1 (congestion signal integrity). The welfare maximising hypothesis only requires brief treatment at the end.

The following chapters are not only structured around the goal of proving the hypothesis, but at the same time having to introduce the elements of the system in an order that will be readable and interesting, and bring out all the insights learned on the way.

The dissertation proceeds in three passes: i) high level ii) abstraction; iii) concrete, because the parts of the system are interdependent, so it would otherwise have been hard to go into detail on any one part without knowing where it fitted into the whole. Experiments appear next to the aspect of the system that they test, not collected separately nearer the end. Otherwise, they would have become so distanced from the assertion they were trying to prove that the connections would have become tenuous.

Part II (next) does the first two passes. It introduces the re-feedback protocol and its incentive mechanisms in a setting that sets aside the practicalities of deployment on the Internet. In particular, it assumes the protocol can write real numbers of arbitrary precision into packet headers. It then introduces some of the possible uses of re-feedback, taking a broad brush approach, but subjecting some aspects to experiment.

Part III contains the bulk of the recent work. Not only is it grounded in the practicalities of the Internet, but it takes a more principled approach to the design of the components introduced previously.

This allows implementations to be tested against the constraints and principles they were intended to realise.

Finally, part IV ties up the proofs of the hypotheses using the material in the intervening parts. It concludes by enumerating limitations and future research directions, before listing material contributions (papers etc.) and giving concluding remarks.

Appendices are added that describe alternative approaches either deprecated or rejected, to record why they fell short, so others need not tread the same erroneous paths.

Part II

Re-feedback

Re-feedback: Summary

These chapters introduce a novel feedback arrangement, termed re-feedback. It ensures metrics in packet headers such as time to live and congestion notification will arrive at each relay carrying a truthful prediction of the remainder of their path. We propose mechanisms at network trust boundaries that ensure the dominant selfish strategy of both network domains and end-points will be to set these headers honestly and to respond in an agreed way to path congestion and delay, despite conflicting interests. Although these mechanisms influence incentives, they don't involve tampering with end-user pricing.

In these chapters mechanisms are described that use the truthful path information to police a response to congestion. We also briefly present a range of other potential uses for truthful path information showing re-feedback is a more generally useful architectural building block than just for rate policing. For instance, we believe it can help to counter flooding attacks, simplify inter-domain traffic engineering and enable inherently scalable QoS.

The re-feedback wire protocol in these chapters uses an abstraction of protocol headers, without regard to how much space is required to store path characterisation values. This is sufficient to explain the architectural intent. A concrete way to fit the protocol into the IP header is deferred until Part III.

The text of this part is largely based on text of Bob Briscoe, Arnaud Jacquet, Carla Di Cairano-Gilfedder, Alessandro Salvatori, Andrea Soppera& Martin Koyabe, , "Policing Congestion Response in an Internetwork using Re-feedback," In Proc. ACM SIGCOMM'05 (Aug 2005) [BJCG+05]. The present author's contribution was the architecture, high level design, experiment design, write-up and editing, but not the implementation and evaluation, nor the write-up of the TCP-policer and the performance evaluation. Three aspects of the original paper have been overtaken by new thinking:

- The bulk dropper design is flawed against attackers adopting the flow ID whitewashing strategy (§7.5.1). Although it has since been replaced by the more principled dropper design of §7, it is still described here unchanged (with a warning note). This dropper design and the performance experiments on it were a large part of the research and experimental endeavour behind this dissertation. Removing them would have damaged the overall dissertation too much.

- One detail of the border mechanisms (that tried to remove the effect of dishonest metrics from metering) was ineffective, and has been removed. §8.2.4 now addresses the issue it tried to solve.

- The re-ECN wire protocol originally described contained a flaw. It has been removed from the text, but its description is still recorded in Appendix B.1, where the flaw is explained.

Finally, it will be noticed that these chapters focus primarily on policing the congestion response of TCP—a rather ill-fitting centrepiece given my subsequent tirade against flow-rate fairness [Bri07b]. This example has been preserved to show how re-feedback can be put to purposes the designer didn't intend [CWSB05].[2]

[2] In 2005, my co-authors and I reluctantly decided to make TCP policing the focus despite knowing it was useless. We relegated bulk policing to a paragraph at the end. We reasoned that the paper, which was already a bit too architectural for SIGCOMM, would otherwise have been too radical to be likely to be accepted. Frustration over having to play along with the flow-rate equality game and having to hide the wider insights in our work, led to the writing of 'Flow Rate Fairness; Dismantling a Religion' [Bri07b] the following year.

Chapter 4

Receiver Aligned Re-inserted Feedback

4.1 Introduction

In 2000, capacity allocation and accountability problems helped to motivate an overhaul of the Internet architecture [BCSW00], but they remain unresolved. We believe their solution lies in a realignment of the feedback architecture.

Changing the Internet's feedback architecture seems to imply considerable upheaval. But, perhaps surprisingly, we believe a limited form of the new arrangement could be deployed incrementally at the transport layer, around unmodified routers using the existing fields in IP (v4 or v6) (see Part III). But protocol engineering isn't the focus at this stage—an idealised numeric scheme is all that is necessary to explain the concepts.

Conceptually, the solution could hardly be simpler. We propose collecting path information in packet header fields as data traverses a path, just as can already be done with time to live (TTL) or congestion notification (ECN [RFB01]). But previously, as each node added characterisation of its local hop, the header values accumulated *upstream* path knowledge. By a simple realignment, we arrange each field to characterise the remaining *downstream* path. We aim to reach a target for the metric at the destination, rather than aligning the datum at the source. For example, TTL currently always starts at the datum 255. Instead we propose it should arrive at the destination set to an agreed datum (say 16). To achieve this, each receiver needs to occasionally feed back the TTL values arriving in packets, so the sender can adjust the next attempt in order to continue to hit 16. §4.2 expands on this basic explanation with more precision.

We term this pattern 're-feedback', short for either receiver-aligned or re-inserted feedback, although it is actually similar to the ordinary feedback found in other disciplines (electronics, hydraulics, etc.). Once re-feedback is in place, each packet arrives at each network element carrying a view of its own downstream path, albeit a round trip ago. So full path congestion becomes visible at the first ingress, where a rate policer is most useful.

But we still don't seem to have solved the problem. It seems naïve to police traffic by trusting fields that depend on the honesty of both the sender and receiver—those with most to gain from lying. However, in §5.1 we explain why re-aligning feedback allows us to arrange for honesty to be everyone's dominant strategy—not only end-users, but also networks. Building on the resulting trustworthiness of

path metrics, we describe how to build a rate equation policer, using TCP as a concrete example. We generalise to any rate equation, in particular Kelly's [KMT98], showing that we can synthesise the same effect as quality of service mechanisms, but only using an ingress policer. And we briefly propose a bulk congestion policer similar to that described in detail in §11. We also describe a passive policer for inter-domain boundaries.

In §5.2 we give the results of simulations conducted to test whether the incentive mechanism really is responsive enough to ensure truthful congestion reporting. Finally we end this collection of chapters with a review of what has been achieved to that point.

The closest idea we can find to this work is Clark's proposed decrementing field representing payment as a packet traversed a path [Cla96], with receiver-initiated messages able to meet it in the middle to make up any shortfall. It may be a subtle distinction, but we would rather network layer fields represented verifiable properties of the path. Then rather than engineers defining a field as a 'price', operators could choose (or not) to apply pricing to whatever fields they wished, in order to determine cost (or even value). Of course, once a price is applied to a field, the operator may have an incentive to distort its meaning to vary the price. But the distinction between a metric and a pure price applied to a metric is still important, as it allows operators not to use the metric for pricing.

It is worth noting that connection-oriented technology such as ATM network elements send congestion back-pressure messages [ITU04] along each connection, duplicating any end to end feedback because they cannot rely on it being present. In contrast, re-feedback ensures information in forwarded packets can be used for congestion management without requiring a connection-oriented architecture and re-using the overhead of fields that are already set aside for end to end congestion control and routing loop detection.

4.2 Re-feedback

Characterising paths through networks requires more than one metric. We have chosen to explain how re-feedback works using two: congestion and delay (that is, unloaded delay not congestion delay). Re-feedback of just these two metrics helps solve a surprisingly large set of networking problems. But additional metrics might be useful in practice, e.g. hop count, unloaded loss rate etc. Delay re-feedback is a useful starting point because it is trivially simple to explain. Then we use congestion to highlight the similarities and differences that are encountered between metrics.

A pre-requisite for re-feedback is the *explicit* declaration of path metrics and their maintenance along the path. Setting aside protocol details for now, it will suffice to consider a multi-bit field for delay and another for congestion carried in future network layer packet headers[1]. Also equivalent fields will be necessary in the end-to-end back-channel from receiver to sender—sent frequently enough to control the most volatile metric (congestion). For instance, in future TCP ACKs (or RTCP receiver reports, etc.)

When starting a flow, the sender has no feedback so it will not know what to put in these fields. However we make the sender responsible for the risk during this period of uncertainty, rather than other

[1] We believe it is possible to apply re-feedback in a separate control plane, or even where control information is analogue, but for clarity we stick to one IP-based scenario.

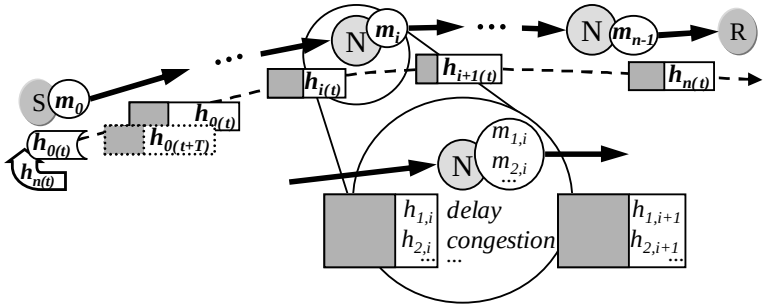


Figure 4.1: Path Characterisation Notation;
For metrics m and headers h.

users of the network. It must declare its conservative expectation of the path characteristics.

We propose an additional 'certain' flag in network layer headers, which the sender should clear at the start of a flow, when no feedback is yet available. Metric(s) carried in uncertain packets should not contribute to any bulk averaging at network equipment (e.g. see §5.1.3), but the flag is not intended to affect forwarding of the packet itself.

Fig 4.1 introduces our notation. Each path across the network consists of a sequence of resources, i_r; $0 \leq i_r < n_r$ indexed in the context of each path r from the sender S with resource $i_r = 0$ to resource $i_r = (n_r - 1)$ just before the receiver R. Whenever a single path context makes it obvious, we will drop the suffix r.

The unloaded delay header, h_1, is carried in packets from resource to resource. Each relay N characterises its local resource's contribution to the delay—perhaps by echo tests with the downstream neighbour. It contributes to the whole path delay by combining its local contribution $m_{1,i}$ with the incoming header value, $h_{1,i}$, and forwarding the updated result, $h_{1,i+1}$ (Fig 4.1). The choice of combining function depends on the metric in question. As unloaded delay is additive, subtraction is an appropriate combining function (like TTL processing), $h_{1,i+1} = h_{1,i} - m_{1,i}$.

Other packet header fields will require combining functions appropriate to the metrics they represent. The inset in Fig 4.1 shows packets carrying header fields for both delay and congestion being combined with the local metrics for each, as parallel, independent operations. Where the context is obvious, we drop the suffix that distinguishes between delay and congestion.

If we introduce feedback of unloaded delay, the receiver will report the header values it receives back to the sender. With classic feedback, the sender always initialises the unloaded delay header to a well-known value, say $h_0 = 255$, as shown in Fig 4.2a). The header will arrive at node j with a value accumulated over all the upstream resources $h_j = h_0 - \sum_{i=0}^{j-1} m_i$. We call the composition of all the local metrics m_i experienced by a packet the **path metric**

So, with classic feedback for delay, the path metric upstream of node j is $\sum_{i=0}^{j-1} m_i = h_0 - h_j$. Node j can work this out by examining h_j in packets as they arrive, because h_0 is well-known. So the receiver (with $j = n$) can characterise the whole path delay $h_0 - h_n$. If it feeds back h_n to the sender using our notional end-to-end protocol (bent arrows in Fig 4.2a)), the sender can know the path delay

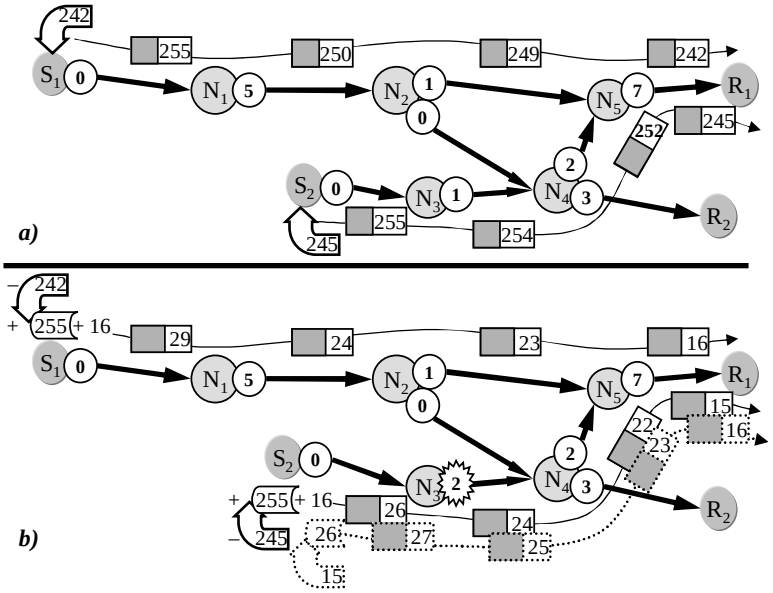


Figure 4.2: Network Flows Carrying Unloaded Delay in Packet Headers.
a) With classic feedback, sources initialise headers to 255. b) With re-feedback over the same network, sources set headers so as to reach 16 at the destination.

too. So far, we have said nothing new, merely introducing notation using a familiar example.

With re-feedback the trick is simply for the sender to choose an initial header value such that, if the path metric were to remain unchanged, the header would reach a well-known value h_z at the destination—rather than *starting* from a fixed value. In our numerical example in Fig 4.2b) we assume the industry has standardised $h_z = 16$.

Although that is really all there is to it, we will now trace through how re-feedback works step by step to be precise about the differences:

1. For now, we will assume that the source bootstraps the very first packet of a flow with the fixed value we used with classic feedback, $h_{0(t)} = 255$. (When we need to distinguish between packets, we suffix each header value with the time t at which it was originally sent.)

2. The source has to remember the initial value it chose, as depicted by the curved boxes containing 255 at each source in Fig 4.2b) and containing $h_{0(t)}$ in Fig 4.1.

3. The packet traverses the path r, combining each local delay in turn into its header, using the combining function (subtraction) already described above.

4. The receiver feeds back the resulting delay header value $h_{n(t)}$ to the sender, which arrives a round trip T_r after the first packet was sent, depicted by the bent arrows.

5. The sender initialises the delay field in the next packet (dotted) to $h_{0(t+T_r)} = h_{0(t)} - h_{n(t)} + h_z$ as well as storing this new value in place of the last one. Each initial delay header value only depends on the previous round's initial value and the value fed back—both known locally at the source.

	unloaded delay	congestion	Eqn
combining function at resource i, $h_{i+1} = g(h_i, m_i)$	$h_i - m_i$	$1 - (1 - h_i)(1 - m_i)$	(4.1)
header initialisation function at source, $h_{0(t+T)} = f(h_{0(t)}, h_{n(t)})$	$h_{0(t)} - h_{n(t)} + h_z$	$1 - \frac{(1-h_z)(1-h_{0(t)})}{1-h_{n(t)}}$	(4.4)
downstream path metric at resource j, $\rho_j(h_{j(t+T)})$	$h_{j(t+T)} - h_z$	$s\left(1 - \frac{1-h_z}{1-h_{j(t+T)}}\right)$	(4.5)

Table 4.1: Re-feedback Functions.
Summarising results from §4.2 & Appendix 4.A, where notation is formally defined. The functions $g(\cdot)$ & $f(\cdot)$ are required to implement re-feedback and $\rho(\cdot)$ to exploit it.

Now we can see that this simple shift of datum has achieved our original aim: as each packet arrives at a resource j anywhere in the network, it carries within its header h_j a prediction of its own downstream path delay, $h_j - h_z$, requiring no path state on the relay because h_z is well-known. Any packet in Fig 4.2b) illustrates this point, in that subtracting $h_z = 16$ from any header value predicts the sum of the remaining downstream resources on that path.

The second column of Table 4.1 summarises the functions to implement delay re-feedback that we have just derived. The third column gives the equivalent functions for congestion, derived in Appendix 4.A.

As with delay, the combining function for each relay to accumulate local congestion into headers (first row) must be chosen to reflect the way congestion accumulates. In Appendix 4.A.1 we define congestion as a probability, using axiomatic definitions[2]. So, as shown, we must use the function for combinatorial probability to combine congestion headers.

For either delay or congestion, the combining function at relays can be the same as for classic feedback, as the purpose is still to accumulate a path metric from local metrics. By avoiding arbitrary changes to the classic combining functions, re-feedback can be introduced incrementally, solely by arrangement between corresponding endpoints.

Each initialisation function (second row) ensures the header reaches h_z at the destination, given the way it accumulates along the path. Each function in the third row was derived from the previous two in order to predict the downstream path metric (DPM) from any node.

Note that neither prediction of DPM requires path state, only the state arriving in the packet itself. Further note that, for congestion, the DPM ρ_j also depends on the effective packet size s. For bit-congestible resources like links $s =$ actual packet size. For packet-congestible resources like forwarding look-ups $s = 1$.

Fig 4.2 also illustrates how a change on a path affects the predictions in packets traversing it. The increase in delay at resource N_3 between Figs 4.2a) & b) (highlighted as a star-burst) causes packets in flight upstream to underestimate their remaining downstream delay. Packets in flight downstream still correctly predict their downstream delay, but when feedback from them releases further packets, these

[2]In contrast to the proposed ECN standard [RFB01] where congestion is defined as the output of the RED algorithm—leaving no objective basis for improving RED.

path knowledge	align-ment	sender	relay	rcvr
up-stream	sender	n/a	$\left[0, \frac{T}{2}\right]$	$\left[0, \frac{T}{2}\right]$
	receiver	n/a	x	x
down-stream	sender	$\left[\frac{T}{2}, T\right]$	x	n/a
	receiver	$\left[\frac{T}{2}, T\right]$	$\left[T, \frac{3T}{2}\right]$	n/a

Table 4.2: Comparison of Sender and Receiver-Aligned Feedback.
By availability of path knowledge (x = not available; n/a = not applicable) and by range of timeliness (using symmetric delay).

underestimate their path delay.

With no further changes in local delays, packets in the following round (dotted) correctly predict the path again. Of course, changes in the unloaded delay at a node (e.g. due to a lower layer re-route) are rare, at least in fixed networks. However, for more volatile metrics like congestion, change is the norm. For delay, the prediction error will be $\sum_{i=0}^{n-1} (m_{i(t+T)} - m_{i(t)})$. For congestion, it is given by Eqn (4.6) in Appendix 4.A.1. In both cases, the error depends on the difference between the whole path metrics.

To put these errors in context, re-feedback causes a source to suffer the same path prediction error as classic feedback—for equivalent path changes within the last round trip. So a re-feedback source transport can extract the same information, with the same timeliness and apply the same rate control algorithms with the same dynamics. For relays, it can take up to an extra half round trip before path changes reach them. But, for relays, any downstream path prediction at all is an improvement over classic feedback, which offers none. And at the ingress, where policers are most appropriate, responsiveness will be similar to that of the source. Table 4.2 summarises the path knowledge that nodes gain or lose from re-feedback. It also quantifies the range of how long it can take for path changes to work through into correct path predictions in each case.

Previously, to achieve such knowledge at every relay would have required messages to be reverse routed hop by hop from all destinations (cf. routing messages or congestion back-pressure). Although re-feedback takes a little longer to propagate (because it travels via the source), it updates at the same *rate* as the ACK rate—as often as TCP congestion control and many orders of magnitude more often than a typical routing message rate. Also, re-feedback piggy-backs on existing data, requiring no extra packet processing.

4.A Re-feedback functions

Below, following the notation of §4.2, we derive the functions required to implement re-feedback for congestion:

- the combining function on each relay, $h_{i+1} = g(h_i, m_i)$,
- the function to initialise header values $h_{0(t+T)} = f(h_{0(t)}, h_{n(t)})$
- the downstream path metric from resource j, $\|_j^{n-1} m_{(t+T)}$.

We coin the notation $\|_a^j m$ for the **path metric**, which is the composition of all the local metrics m_i experienced by a packet along the sequence of resources $\{a, \ldots i, \ldots j\}$ using the combining function appropriate to the metric in question.

4.A.1 Congestion re-feedback

Definition 4.1. *The congestion, m_i, caused by a packet at single resource i is the probability that the event X_i will occur if the packet in question is added to the load, given any pre-existing differential treatment of packets. Where X_i is the event that another selected packet will not be served to its requirements by resource i during its current busy period.*

So, at resource i, the contribution to congestion is $m_i = P(X_i) \in [0, 1]$, which is a function of local load.

Definition 4.2. *The path congestion, $\|_a^j m$, caused by a packet traversing a sequence of resources, is the probability that the event X will occur if the packet in question is added to the loads at each resource along its path, given any pre-existing differential treatment of packets. Where X is the event that another selected packet will not be served to its requirements by any of the sequence of resources $\{a, \ldots i, \ldots j\}$ during their current busy periods.*

From definition 4.1, the function that combines the local contribution with the incoming congestion notification field must emulate combinatorial probability resulting in an outgoing header value

$$h_{i+1} = 1 - (1 - h_i)(1 - m_i). \tag{4.1}$$

$\therefore$ if the header is h_a before resource a, after node $j - 1$ it will be

$$h_j = 1 - (1 - h_a)\textstyle\prod_{i=a}^{j-1}(1 - m_i). \tag{4.2}$$

From definition 4.2 the path metric from resource a to $j - 1$,

$$\begin{aligned} \|_a^{j-1} m = P(X) &= 1 - \textstyle\prod_{i=a}^{j-1}(1 - P(X_i)) \\ &= 1 - \textstyle\prod_{i=a}^{j-1}(1 - m_i) \\ &= 1 - \frac{1 - h_j}{1 - h_a} \end{aligned} \tag{4.3}$$

A source with perfect foresight would initialise a packet header to $h^*_{0(t+T)}$ in order to reach its target value at the destination, where

$$\begin{aligned} h_{n(t+T)} &= 1 - \left(1 - h^*_{0(t+T)}\right)\left(1 - \|_0^{n-1} m_{i(t+T)}\right) \\ &= h_z \\ \therefore h^*_{0(t+T)} &= 1 - \frac{1 - h_z}{1 - \|_0^{n-1} m_{i(t+T)}} \end{aligned}$$

A practical source will use the previous path metric as an estimator for the next and set

$$\begin{aligned} h_{0(t+T)} &= 1 - \frac{1 - h_z}{1 - \|_0^{n-1} m_{i(t)}} \\ &= 1 - \frac{(1 - h_z)(1 - h_{0(t)})}{1 - h_{n(t)}} \end{aligned} \tag{4.4}$$

During sudden increases in congestion, $h_n \to 1$, but if protocol fields are bounded the source will remain responsive, but understate congestion to the network, which is the safe way round.

With hindsight, the downstream path metric from resource j

$$\|_j^{*n-1} m_{(t+T)} = 1 - \frac{\prod_{i=0}^{n-1}(1-m_{i(t+T)})}{\prod_{i=0}^{j-1}(1-m_{i(t+T)})}.$$

An efficient estimator for this metric is

$$\begin{aligned} \|_j^{n-1} m_{(t+T)} &= 1 - \frac{\prod_{i=0}^{n-1}(1-m_{i(t)})}{\prod_{i=0}^{j-1}(1-m_{i(t+T)})} \\ \text{From (4.3)} \quad &= 1 - \frac{1-h_z}{1-h_{0(t+T)}} \Big/ \frac{1-h_{j(t+T)}}{1-h_{0(t+T)}} \\ \text{From (4.4)} \quad &= 1 - \frac{1-h_z}{1-h_{j(t+T)}}. \end{aligned} \tag{4.5}$$

The prediction error $\|_j^{n-1} m_{(t+T)} - \|_j^{*n-1} m_{(t+T)}$ is

$$\frac{\prod_{i=0}^{n-1}(1-m_{i(t+T)}) - \prod_{i=0}^{n-1}(1-m_{i(t)})}{\prod_{i=0}^{j-1}(1-m_{i(t+T)})}. \tag{4.6}$$

Chapter 5

Re-feedback Incentive Mechanisms

5.1 Incentives

We aim to create an incentive environment to ensure anyone's selfish behaviour (including lying and cheating) leads to truthful declaration of downstream path characteristics.[1] Throughout this section we will focus primarily on characterisation of path congestion. This will stress re-feedback incentive mechanisms to the full in the face of conflict over scarce resources. Given most forms of fairness, including TCP's, also depend on round trip time, we will then outline how a path delay metric would be amenable to similar treatment.

Fig 5.1 sketches the incentive framework that we will describe piece by piece throughout this section. An internetwork with multiple trust boundaries is depicted. The downstream path congestion seen in a typical packet is plotted as it traverses an example path from sender S_1 to receiver R_1. They are shown using re-feedback, but we intend to show why everyone would *choose* to use it, correctly and honestly.

Two main types of self-interest can be identified:

- Users want to transmit data across the network as fast as possible, paying as little as possible for the privilege. In this respect, there is no distinction between senders and receivers, but we must be wary of potential malice by one on the other;
- Network operators want to maximise revenues from the resources they invest in. They compete amongst themselves for the custom of users.

Source congestion control: We want to ensure that the sender will be pressured to reduce its rate as downstream congestion increases. Whatever the agreed congestion response (whether TCP-compatible, flow admission control or some hybrid general reduction across all the sender's flows), to some extent it will always be against the sender's interest to comply.

Edge ingress policing/shaping: But it is in all the network operators' interests to encourage a congestion response, so that their investments are employed to satisfy the most valuable demand. N_A is in the best position to ensure S_1's compliance and it now has a choice of mechanisms across a spectrum of customer autonomy. At one extreme, N_A could give S_1 complete autonomy, but encourage responsible

[1] These mechanisms can lie dormant wherever co-operation is the social norm.

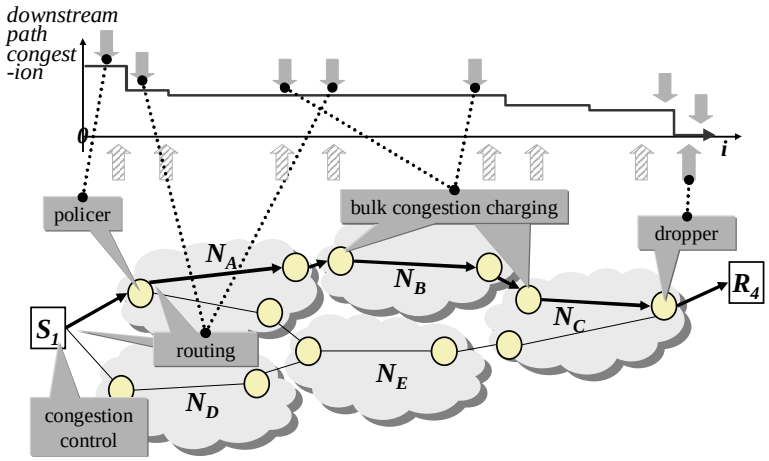


Figure 5.1: Re-feedback Incentive Framework.

behaviour by charging for the downstream congestion in packets. Or it can shape traffic directly itself, removing all S_1's autonomy. Between the two extremes, it can police a congestion response agreed upfront with S_1 (§5.1.4).

Edge egress dropper: If the source has less right to a high rate the higher it declares downstream congestion, it has a clear incentive to understate downstream congestion. But, if packets are understated when they enter the internetwork, they will be negative when they leave. So, we introduce a dropper at the last network egress, which drops packets in flows that persistently declare negative downstream congestion (see §5.1.3).

Inter-domain traffic policing: But next we must ask, if congestion arises downstream (say in N_D), what is the ingress network's (N_A) incentive to police its customers' response? If N_A turns a blind eye, its own customers benefit while other networks suffer. This is why all inter-domain QoS architectures (e.g. Intserv, Diffserv) police traffic each time it crosses a trust boundary. Re-feedback gives trustworthy information at each trust boundary so the congestion response can be policed in bulk.

Emulating policing with inter-domain congestion charging: Between high-speed (e.g. optical) networks we would rather avoid having to buffer packets while deciding whether to police them in series to forwarding. Instead, we can emulate policing using a passive, parallel monitoring function. Once re-feedback has arranged headers to carry downstream congestion honestly, N_B can contract to pay N_D in proportion to a single bulk count of the congestion metrics ρ crossing their mutual trust boundary (§5.1.5). Then N_B has an incentive either to police the congestion response of its own ingress traffic from N_A or to charge N_A in turn on the basis of congestion counted at their mutual boundary. In this recursive way, each flow's response can be precisely incentivised, despite the mechanism not recognising flows. If N_A turns a blind eye to its own upstream customers' congestion response, it will still have to pay its downstream neighbours. And if N_A lies on behalf of its customers by understating downstream congestion, packets will suffer at the dropper as if the source itself had lied.

No congestion charging to users: Bulk congestion charging at trust boundaries is passive and extremely simple, and loses none of its per-packet precision from one boundary to the next. But at any

trust boundary, there is no imperative to use congestion charging. Traditional traffic policing can be used, if the complexity and cost is preferred. In particular, at the boundary with end customers (e.g. between S_1 and N_A), traffic policing will most likely be more appropriate. Policer complexity is less of a concern at the edge of the network. And end-customers are known to be highly averse to the unpredictability of congestion charging [Odl97].

Competitive discipline of inter-domain traffic engineering: With inter-domain congestion charging, a domain seems to have a perverse incentive to fake congestion; N_B's profit depends on the difference between congestion at its ingress (its revenue) and at its egress (its cost). So overstating internal congestion seems to increase profit. However, smart border routing [GQX+04] by N_A will bias its multipath routing towards the least cost routes, so N_B risks losing all its revenue to competitive routes if it overstates congestion. In other words, N_B's ability to raise excess profits is limited by the price of its second most competitive route (but see §12.1.2 on Termination Monopolies).

Closing the loop: All the above elements conspire to trap everyone between two opposing pressures (upper half of Fig 5.1), ensuring the downstream congestion metric arrives at the destination neither above nor below zero. So we have arrived back where we started in our argument. The ingress edge network can rely on downstream congestion declared in the packet headers presented by the sender. So it can police the sender's congestion response accordingly.

5.1.1 The case against classic feedback

So why can't classic congestion feedback (as used already by standard ECN) be arranged to provide similar incentives? Superficially it can. Given ECN already existed, this was the deployment path Kelly proposed for his seminal work that used self-interest to optimise social welfare across a system of networks and users [KMT98]. The mechanism was nearly identical to volume charging; except only the volume of packets marked with congestion experienced (CE) was counted. However, relying on classic feedback meant the incentives traced an indirect path—the long way round the feedback loop. For example, if classic feedback were used in Fig 5.1, N_B would incentivise N_A via N_D, R_1 & S_1 rather than directly.

Inability to agree what happened: In order to police its upstream neighbour's congestion response, the neighbours should be able to agree on the congestion to be responded to. Whatever the feedback regime, as packets change hands at each trust boundary, any path metrics they carry are verifiable by both neighbours. But, with a classic, sender-aligned path metric, they can only agree on the *upstream* path congestion—its offset from its well-known datum at the sender.

Inaccessible back-channel: The network needs a whole path congestion metric to control the source. Classically, whole path congestion emerges at the destination, to be fed back from receiver to sender in a back-channel. But, in any data network, back-channels need not be visible to relays, as they are essentially communications between the end-points. They may be encrypted, asymmetrically routed or simply omitted, so no network element can reliably intercept them. The congestion charging literature solves this problem by treating the sender and receiver as entities with aligned incentives. Although measuring classic ECN marking rates (relative to their datum at the sender) forces a 'receiver pays'

model (at each trust boundary the downstream neighbour pays), it is argued that at least this incentivises the receiver to refer the charges to the sender.

'Receiver pays' unacceptable: However, in connectionless datagram networks, receivers and receiving networks cannot prevent reception from malicious senders, so 'receiver pays' opens them to 'denial of funds' attacks.

End-user congestion charging unacceptable: Even if 'denial of funds' were not a problem, we know that end-users are highly averse to the unpredictability of congestion charging and anyway, we want to avoid restricting network operators to just one retail tariff. But with classic feedback, we cannot avoid having to wrap the 'receiver pays' money flow around the feedback loop, necessarily forcing end-users to be subjected to congestion charging.

Receiver Policing Impractical: It might be thought that the egress network N_D could police the receiver, rather than apply a congestion charge. For instance, limiting downloads based on classic ECN marks arriving from other networks (and from within its own before the egress). Then, as the policer slowed the data flow, through the normal process of feedback the source would slow down. The first problem with this approach is that it relies circularly on the sender responding to loss in order to police the sender's response to congestion. Certainly the policer could stop the receiver benefitting from fast unresponsive downloads, but it could not stop the sender if it didn't respond to loss.

Perhaps this would at least be better than nothing were it not for there being a second problem with egress network policing. All the inter-network incentives to do policing are backwards. If there were congestion in the sender's access network N_A, the only practical way to give all the egress networks like N_D an incentive to police the flows causing congestion in N_A it would be for N_A to charge N_B for receiving congestion from N_A, then N_B would have the incentive to charge N_D, this in turn giving N_D the incentive to police the flow. Although this sounds possible when you say it fast, no network N_B would ever pay another network N_A to *receive* traffic that claimed to have experienced congestion in the originating network. N_A controls how much and which traffic it sends to N_B, so N_B would never want to advertise any routes to N_A. This would just give N_A carte blanche to print money by sending N_B congested traffic.

To summarise so far, with classic feedback, policing congestion response *requires* congestion charging of end-users and a 'receiver pays' model. Whereas, with re-feedback, incentives can be fashioned *either* by technical policing mechanisms (more appropriate for end users) *or* by congestion charging (more appropriate inter-domain) using the safer 'sender pays' model.

Impractical traffic engineering: Finally, classic feedback makes congestion-based traffic engineering inefficient too. Network N_D can see which of its two alternative upstream networks N_B and N_C are less congested. But it is N_A that makes the routing decision. This is why current traffic engineering requires a continuous message stream from congestion monitors to the routing controller. And even then the monitors can only be trusted for *intra*-domain traffic engineering. The trustworthiness of re-feedback enables *inter*-domain traffic engineering without messaging overhead (but that is out of scope of this dissertation).

We now take a second pass over the incentive framework, filling in the detail more formally.

5.1.2 The Case Against Bottleneck Policers

We borrowed ideas from policers in the literature [OLW99, FF99, PBPS03] for our rate equation policer. However, without the benefit of re-feedback they don't police the correct rate for the condition of their path. They detect unusually high *absolute* rates, but only while the policer itself is congested, because they work by detecting prevalent flows in the discards from the local RED queue. These policers must sit at every potential bottleneck, whereas our policer need only be located at each ingress to the internetwork. As Floyd & Fall explain [FF99], the limitation of their approach is that a high sending rate might be perfectly legitimate, if the rest of the path is uncongested or the round trip time is short.

XCP [KHR02] bears a superficial resemblance to re-feedback in that routers along the path decrement the change in flow rate requested in-band by the source, which is then fed back from receiver to source. However, the structure of fairness aimed for by XCP is more analogous to that aimed for by a bottleneck policer. Indeed both XCP and bottleneck policers are structurally similar to a dynamic form of RSVP [ZDE+93]. The subtle but important structural difference between XCP and re-feedback is that XCP's metric quantifies the service rate (the primal variable), not the impairment introduced along the path (the dual). Even if the set of all the service rates is combined (e.g. at the customer's attachment point) nothing can be determined about whether that customer's use of the *whole network* is fair, because there is insufficient information about how much each flow impacts *other* users at each queue. In addition, in a non-co-operative setting, the flow rate that each XCP packet claims it is part of has to be policed at each trust boundary to check it matches the flow rate actually being used, which requires per flow processing. This was the issue that killed the scalability of the Integrated Services architecture [BBB+97]. "Flow rate fairness, Dismantling a Religion" [Bri07b] gives a more extensive discussion of the deficiencies in the structure of fairness aimed for by protocols like XCP and RCP [DKZSM05].

5.1.3 Honest congestion reporting

An honest sender will declare a certain downstream path metric (DPM) ρ_0 in packets to aim for zero at the destination after allowing for path congestion. We define cheating as the difference $\Delta\rho_{0c}$ relative to this ideal, taking overstatement as positive. To rely on the DPM packets carry, we must discourage dishonesty, whether positive or negative. If the sender declares a certain DPM, a certain rate response can be policed either specifically for each flow, or an overall response envelope for all flows from a particular customer (§5.1.4). For any safe congestion response, the higher the sender's declared DPM, to some degree the slower its data rate, and the lower the value it derives. So, to the right of Fig 5.2 we can show the sender's utility strictly decreasing with overstatement.

So senders have an incentive to understate DPM, which allows them a higher bit rate. But then the DPM will turn negative before reaching the destination. If networks discard[2] negative packets, the utility to the sender of the higher bit rate will rapidly collapse, as visualised on the left of the figure. Therefore honesty at $\Delta\rho_{0c} = 0$ will be the dominant sender strategy that maximises its net utility. A receiver that

[2]Various penalties short of discard, e.g. payload truncation, can be imposed in order to preserve the feedback loop, given a packet may be wrongly penalised.

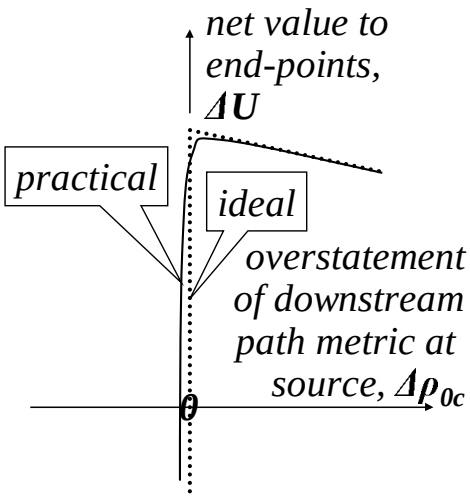


Figure 5.2: Truth Telling Incentives.

genuinely wants data to be sent as quickly as possible has incentives aligned with the sender, so honest feedback also returns the maximum net gain.

In fact, the position is complicated by continuous variability of path congestion; even honest traffic will arrive at its destination spread around zero. Below we describe a dropper that makes allowances for this variability but still detects understatement of DPM. The best dropper we can currently envisage suffers some false hits and false misses, blunting the incentive to be absolutely honest (Fig 5.2).

Adaptive dropper

If congestion didn't vary, a malicious source understating congestion by $\Delta\rho_{0c}$ (numerically negative) would cause a proportionate understatement at the destination of $\Delta\rho_{nc}$.[3] But congestion does vary, so if the probability distribution of the DPM at the destination is $P_n(\rho_n)$ for an honest sender, it will be shifted to $P_n(\rho_n - \Delta\rho_{nc})$ for the malicious sender.

We propose a dropper[2] at the last hop before the receiver. The dropper builds a model of the prevailing pattern of cheating for all packets leaving the same interface and assumes that each new packet is characteristic of this recent history; the more recent cheating, the stricter the dropper becomes. But its strictness is further modulated by how negative ρ_n is of each packet under scrutiny.

Conceptually, the bell curve in Figure 5.3 shows the probability distribution of arriving packets, exponentially weighted to favour recent arrivals. We assume this will be the probability distribution of the DPM of the next packet. Superimposed on a different vertical scale is a conjectured penalty probability function, $p(\cdot)$ intended to allow through as much negative DPM as positive, but no more. This can be achieved by ensuring that the distribution remaining after applying the penalty function is symmetric about zero (the unshaded cusp curve). So for $\rho_n < 0$:

$$\big(1 - p(\cdot)\big)P_n(\rho_n - \Delta\rho_{nc}) = P_n(\rho_n + \Delta\rho_{nc}). \tag{5.1}$$

[3] From Eqn (4.2) $\Delta\rho_{nc} = \big(1 - \|_0^{n-1} m\big)\Delta\rho_{0c}$

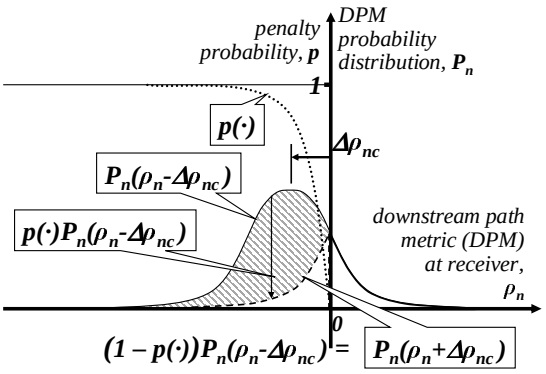


Figure 5.3: Penalising Misbehaviour Under Uncertainty.

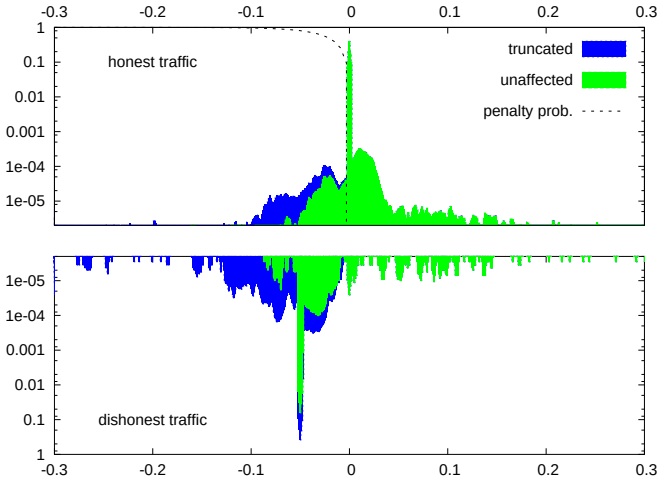


Figure 5.4: Typical simulated distributions of DPM at the destination.
From honest (top) and dishonest (bottom) sources, also showing proportion of penalised traffic (note log scale).

Initially we choose to keep state and processing to a minimum by modelling prevailing conditions with just the exponentially weighted moving average μ and EWM variance ν. So we model the prevailing distribution $P_n(\rho_n - \Delta\rho_{nc})$ as if it were the normal distribution $N(\mu, \nu)$ reconstructed from recent traffic, whatever the actual distribution (e.g. Fig 5.4).

At each packet, the EWMA & EWMV are updated:

$$\mu \leftarrow \gamma\rho_n + (1-\gamma)\mu \tag{5.2}$$

$$\nu \leftarrow \gamma(\rho_n - \mu)^2 + (1-\gamma)\nu. \tag{5.3}$$

For attack traffic $\mu \to \Delta\rho_{nc}$, converging faster by increasing γ to weight recent values $(0 < \gamma \leq 1)$. In maintaining the EWMA, positive packets with the 'certain' flag cleared (see §4.2) are ignored, incentivising correct use of the flag.

Then, using the formula for a normal distribution,

$$P_n(\rho_n - \Delta\rho_{nc}) = \frac{1}{\sqrt{2\pi\nu}} e^{-\frac{(\rho_n-\mu)^2}{2\nu}}. \tag{5.4}$$

we can derive the required penalty probability function to apply to each specific packet with DPM ρ_n, by re-arranging (5.1) and substituting from (5.4):

$$\begin{aligned} p(\rho_n, \mu, \nu) &= 0; && \mu \geq 0 \text{ or } \rho_n \geq 0 \\ &= 1 - \frac{P_n(\rho_n + \mu)}{P_n(\rho_n - \mu)}; && \mu < 0, \rho_n < 0 \\ &= 1 - e^{\frac{-2\rho_n \mu}{\nu}} \end{aligned} \tag{5.5}$$

As required, the penalty becomes stricter the worse the EWMA becomes, but flattens to zero discards when honest users keep the EWMA to zero.

Where a cheating flow is hidden in a large honest aggregate, it causes a slightly negative EWMA, leading to some dropping. After Floyd and Fall [FF99] we cache the flow identifiers of penalised packets. Once any aggregate of destination (and/or source) identifiers appears more often than would be likely by chance, a second instance of the dropper is spawned and traffic matching the identifier(s) is filtered into it.[4] Each instance of a focused dropper maintains its own EWMA[5] and may spawn further droppers. These focused droppers should be far more sensitive than the first, also shielding honest traffic from the risk of false hits.

Of course, if cheating negative traffic imitates identifiers used in honest traffic, both will be filtered into the same focused dropper, causing collateral damage to the honest traffic. But by definition the cheating traffic will tend to be more negative, which the above penalty function is designed to discriminate against.

Having isolated suspect identifiers, an egress edge dropper can send hints upstream. Any node can test hints because they point to traffic measurably below an objective threshold. And a node need only act on the hints if it has sufficient resources. So the hints need not be authenticated (unlike DoS filter push-back requests), avoiding vulnerability to floods of bogus authentication requests. Also, the hints can safely jump multiple domains without the need for a global key management infrastructure. So push-back of hints does not depend on the co-operation of high speed core networks, where operators are more wary of any additional processing.

Even if explicit congestion marking were universally deployed, buffers could still occasionally overflow. So irrespective of any hints, if a router must discard packets, clearly it should bias against any with negative DPM (§§9 & 12.1.1 develop this point).

[4] As the introduction to the current Part II of this dissertation explains, since writing this chapter we have realised that designs like this (and all bottleneck policers that inherited the same idea from Floyd & Fall) are flawed. One cannot rely on the deterrent effect of policing in bulk against flows that can switch to a whitewashed flow ID whenever they find their misbehaviour has been detected. There can be no deterrent against a cheap pseudonym that can be discarded and replaced instantly at no cost [FR98]. The new dropper design presented later in §7 fixes this vulnerability to whitewashed identifiers.

[5] From Eqn (5.5) an attacker can reduce dropping probability by increasing variance, e.g. by alternating honest & zero packets. So a focused dropper should use the EWMV of the top level dropper. We are investigating variants with varying degrees of statefulness and responsiveness.

Honest delay reporting

Congestion control and traffic engineering depend on path delay as well as congestion, so we could need header fields for both. It is possible that policing the amount of congestion caused (first order) will be sufficient so that policing the dynamics (second order) will not be necessary. But it might be necessary to police dynamics, in which case a robust way to measure downstream path delay will also be necessary. The framework we built above (§5.1.3 & Fig 5.2) to incentivise honest congestion reporting relied on two properties of congestion: it physically cannot be negative; and rising path congestion should lead to a drop in sending rate (whatever form of fairness is chosen). Delay has exactly the same properties: negative delay is physically impossible; and rising feedback delay should lead to a lower sending rate.

So, we can use a similar incentive mechanisms to that we used for congestion to ensure the sender neither overstates nor understates delay. An adaptive dropper, like the one above for the congestion field (§5.1.3), could detect and remove any negative imbalance of delay headers at the internetwork egress. And at the ingress we can use a policer like the TCP rate equation policer (§5.1.4 below) that punishes sources sending faster than the 'TCP-friendly' rate, which depends inversely on both congestion and feedback delay. Or preferably we can use the bulk congestion policer which keeps an aggregate of flows within an overall response to congestion but allows each flow to give or take from others (also see §5.1.4).

5.1.4 Policing congestion response

TCP rate equation policer

In the fastest phase of the TCP algorithm (congestion avoidance), TCP converges to the rate $\bar{x}_{TCP} \approx ks/(T\sqrt{p})$, where $k \approx \sqrt{(3/2)}$ and s, T & p are respectively the packet size, round trip time and path marking (or loss) rate [PFTK98]. Re-feedback ensures that a policer at the network ingress can derive these parameters from the metrics each packet truthfully declares. It can then calculate a compliant rate against which to compare the source's actual rate.

Previous policers had to be placed at every site of possible congestion. With re-feedback, it is sufficient to place one policer at each ingress to the internetwork. Here, downstream congestion $\rho_{2,1}$ can be assumed equal to path congestion, p. The policer can approximate the round trip delay as $T \approx T_0 + 2\rho_{1,1}$, where the upstream round trip T_0 can be found by a previous echo test against each source and the downstream delay $\rho_{1,1}$ arrives in each packet[6].

If the current TTL and ECN fields in IP were used to implement re-feedback, as sketched in the SIGCOMM paper on re-feedback [BJCG+05], an ingress policer would have enough information to mirror the TCP algorithm. Unary congestion marking can take a long time to convey an accurate congestion level.[7] Therefore, given the architectural nature of this part of the dissertation, we prefer to focus

[6] For simplicity, we choose to ignore congestion delay, because simple scaling arguments [Kel00, §2] show that as capacity continues to grow, congestion delays will become insignificant relative to fixed propagation delays.

[7] The TCP equilibrium flow rate $\bar{x}$ is proportional to $\sqrt{1/p}$ in congestion avoidance [MSMO97], where p is the marking or loss fraction. Therefore the number of packets between unary congestion marks scales $O(\bar{x}^2)$. As a numerical example, to sustain 10Gbps a flow would only sawtooth every 90mins between marks. Note that currently typical values of $\bar{x}$ double every 1.6 years or so. In contrast, in Kelly's rate control algorithm $\bar{x} \propto 1/p$, so inter-mark spacing scales $O(\bar{x})$. If packet size stays constant, the number of packets per round trip (the window) also scales $O(\bar{x})$. Therefore, if most rate control algorithms evolved from TCP to a

on multi-bit congestion and delay fields in future packet headers. §7.7 quantifies how quickly a dropper could detect misbehaving flows using a unary encoding of congestion that is all that is possible with today's IP header. The benefit of using more bits in headers to signal congestion is planned for future work, building on [GKM01, TC04, AHCC06, XSSK05].

Below we outline one possible policing algorithm. It requires per flow state, but this isn't necessarily a scalability problem at the edge of an internetwork, however it does lay the policer open to resource depletion attacks. We have also designed an unpublished variant with sub-linear scaling of flow state, but our goal here is to give a clear implementation example that is concrete but avoids gratuitous distractions.

The policer requires a token bucket per flow. It empties the bucket by the size of each arriving packet and fills it at a rate equivalent to that of a TCP compliant flow experiencing the same path conditions. It calculates this by deriving p and T from the re-feedback fields as above. In other words, when a packet arrives, the policer subtracts the packet size s from the bucket and adds $ks\Delta t/(T\sqrt{p})$, where Δt is the time since the flow's previous packet.

If the bucket empties, sanctions are applied to the flow. For instance, all future packets might be discarded, or the policer could choose to take over rate control for the flow. The depth of the bucket controls the flexibility allowed for a flow to stray from its expected throughput; it is set to $\alpha\bar{x}_{TCP}\tau$, where α is the threshold greediness for a flow to be considered non-compliant over a time τ, and $\bar{x}_{TCP}$ is an EWMA of $ks/(T\sqrt{p})$. A flow with a throughput higher than $\alpha\bar{x}_{TCP}$ will be detected in a time smaller than τ.

α is chosen so that a compliant flow is most unlikely to trigger starvation of the bucket. For instance, when p=1%, the average congestion should be 12.3 packets per round-trip. The probability of getting a window larger than 42 is smaller than 0.01%. Setting α to $42/12.3 = 3.4$ and $\tau = T$ would guarantee that less than one compliant flow in ten thousand would be subjected to sanction. Increasing α and τ would reduce false hits further.

Bulk congestion policer

If one user creates multiple flows, or runs flows for longer than another user (e.g. p2p file-sharing), per-*flow* approaches like TCP cannot arbitrate fairness between *users*. We can generalise to an adaptive policer based on MulTCP [CO98] that gives each flow an equivalent rate to w TCP flows. With the benefit of re-feedback, it can maintain a per user count of congestion sent. But, rather than levying an unpredictable charge for this congestion [KMT98], the policer can compare the count to whatever the user chooses to pay. So a flat monthly rate would effectively buy a congestion quota. The closer the internal congestion count approached this quota, the more w would be squeezed.

This style of policing is similar to that described in [JBM08] and further developed in §11 later.

Edge QoS

Our interest in solving the policing problem was not solely to police a single response to congestion, such as TCP-friendliness, although that alone is a major contribution. Once timely, truthful downstream

rate response proportional to $1/p$, as Kelly's algorithm does, the time between marks would scale $O(1)$ (i.e. stay constant) as flow rates increase.

path information is visible to ingress network operators in data packets, they can offer a spectrum of responses to incipient congestion. This is equivalent to offering different levels of QoS, perhaps ranging from a scavenger class, through best effort and premium levels of differentiated service to admission controlled bandwidth reservations (the right to zero congestion response)—all without any differential treatment on network elements beyond the first ingress (with the caveat below).

Kelly and co-workers [KMT98] pioneered this approach, proving it optimises social welfare across a network. Further its policing architecture solves the scalability problems inherent in other QoS approaches, though this is seldom appreciated.

With traditional QoS some identification convention must distinguish which traffic the edge has decided should be given which preferential treatment as it passes to interior domains. Using flow identification (like Intserv) preserves precision, but scales badly. Using class identification (like Diffserv) loses precision at scale.

With edge QoS, instead of the edge identifying the traffic's QoS for interior routers, interior routers identify the traffic's congestion for the edge. Because traffic already carries end-point identifiers, regular packet forwarding carries congestion marking to its destination end-point which in turn feeds it back to its source—the root cause. Therefore packet markings traverse deaggregation and reaggregation with absolute precision, and with no need for a separate QoS identification convention. The only unequal treatment of different traffic identities is in the policer at the first ingress to the internetwork, where customer or flow identities have local significance.

Siris [Sir02] has proven this approach through simulation. But deployment was confined to a radio network controller scenario where congestion feedback in the back-channel to the sender could be intercepted and was trusted to be correct—assumptions that can be relaxed with re-feedback, giving general applicability.

Having sung the praises of closed-loop control, a caveat is necessary. Unusual conditions (link failure or sudden traffic shifts) can cause traffic in flight to overflow queues. So, within a round trip, strong QoS assurances are only possible if each resource is capable of rudimentary local (open-loop) traffic class prioritisation until the closed-loop restores order.

Flow start incentives

At the start of each flow, a sender neither knows the state of the path to the destination nor the relative change the additional flow will cause. TCP's slow-start phase incrementally finds out both while also giving other flows time to make room for the new flow.

The re-feedback incentive framework deliberately presents a dilemma to a sender without recent path knowledge (e.g. at the first packet, or after an idle period). Sending understated DPM increases the risk of discard at the egress dropper. But sending overstated DPM increases the risk of sanction at the ingress policer as the flow rises to full rate. The strategies around this dilemma deserve a paper in their own right, so here we merely provide an outline.

We should think of TCP's exponential slow-start as dependent on an implicit evolving estimate of path congestion by the sender, starting pessimistically by assuming high path congestion. Inverting

TCP's steady state rate equation gives $\rho \propto \frac{1}{\bar{x}^2}$ to a first approximation. So rate doubling quarters the implicit path congestion estimate every round trip. To safely pass the policer and the dropper, the sender should be consistent, also using this implicit estimate of path congestion to set the DPM in each sent packet. If it reduces its path congestion estimate too quickly (increasing its rate accordingly), it will undershoot the true path congestion and risk being caught by the egress dropper.

So the re-feedback incentive framework encourages caution at the start of a flow in proportion to path uncertainty—reminiscent of TCP's slow start [KM99]. However this claim greatly depends on how quickly our mechanisms can detect and remove non-compliant behaviour.

It is well-known that repeated unary congestion feedback like ECN takes a long time to signal low congestion levels. So ECN is not a good basis on which to build responsive policing mechanisms. In the years it would take to deploy the TCP modifications needed for our re-feedback extension of ECN (§6.1.2), TCP will be hitting its own scalability limits.[7] So although we believe re-ECN could start to solve policing problems fairly quickly, we must emphasise that a multi-bit congestion field will need to be considered anyway. It would provide responsive policing even if short flows dominate the future traffic mix. And at the same time, it would help fix TCP/IP for high capacity scenarios.[8]

This still leaves the problem of whether the new flow will push a currently uncongested path into congestion.

5.1.5 Inter-domain incentive mechanisms

The overview of our incentive framework explained why bulk inter-domain congestion charging emulates policing with per-flow precision. We now describe this mechanism.

At an inter-domain interface, only a single bulk counter (and two temporary ones) per direction is needed. The main counter merely accumulates the DPM ρ in every passing packet over an accounting period T_a (e.g. a month). At the end of the month, N_A should pay N_B the charge $C_a = \lambda \sum^{T_a} \rho^+$, where λ is the fixed price of congestion agreed between them. To implement this with the re-feedback variant of ECN described in §6.1.2, the meter would simply need to increment or decrement by the size of packets marked with the Positive or Negative code-points respectively.

To protect receiving domains from 'denial of funds' attacks, any usage element of a charge should be 'sender pays'.[9] So $\lambda \geq 0$ and persistently negative ρ should be ignored, given negative congestion is physically impossible (see §8.2.4). Once neighbours agree that 'no-one pays' for persistent negative congestion, they are incentivised to introduce the dropper (§5.1.3) to remove persistent negative traffic, which no longer carries any ability to pay for further downstream congestion. 'Receiver pays' can optionally be arranged between edge operators without risk of 'denial of funds' through an end-to-end clearinghouse [BR05].

We should clarify that we neither require nor expect universal inter-domain congestion charging. However, because it exposes true costs, it is likely to emerge as the competitive equilibrium [BR05].

[8] An extra multi-bit field in IP is already proposed for the allowed congestion window in XCP [KHR02] and for the allowed sending rate in Quick-Start [FAJS07].

[9] A capacity charge made to the larger network, whatever the direction of traffic, might well complement congestion charging (or any form of usage charging).

Current tariffs such as 95th %ile peak demand or volume charging may continue. But to compete, manual price adjustments will be needed to track the congestion price. So congestion charging is likely to predominate, given it uses a simple, passive mechanism without regard to flows, but automatically adjusts the price to give the correct upstream incentives to the precise flows that deserve them.

The main alternative to usage charging is the service level agreement, where a network contracts to keep metrics within statistical limits. Currently, proving whether delay or loss (impairment) budgets have been exceeded and by whom requires a comprehensive system of trusted echo reflectors. Re-feedback greatly simplifies these problems of SLA accountability, because it ensures downstream metrics are visible purely locally at each inter-domain border.

5.1.6 Distributed denial of service mitigation

A flooding attack is inherently about congestion of a resource. Because re-feedback ensures the causes of network congestion experience the cost of their own actions, it acts as a first line of defence against DDoS. As load focuses on a victim, nearby upstream queues grow, requiring packets to be pre-loaded with a higher congestion metric. If the source does increase the initial metric, its own network's ingress policer will throttle the flow. If the source doesn't increase the initial metric, it will become negative at the congested resource, which can bias its drop against negative traffic.

Inter-domain congestion charging ensures that any network that harbours compromised 'zombie' hosts will have to pay for the congestion that their attacks cause in downstream networks. Therefore, it is incentivised to deploy our adaptive policer (§5.1.4). The adaptive policer limits hosts that persistently causes congestion to only send very slowly into congested paths. As well as protecting other networks, the extremely poor performance at any sign of congestion will incentivise the zombie's owner to clean it up.

Note, however, that delay in detecting attacks does leave re-feedback briefly vulnerable (§§5.1.4 & 5.2).

5.2 Dropper performance

The re-feedback incentive framework relies critically on how quickly the dropper (§5.1.3) can detect and isolate flows that are maliciously understating congestion, and how much collateral damage is suffered by honest packets. The error in an honest source's prediction of congestion for re-feedback (Eqn 4.6) depends on how well path congestion in one round trip correlates with congestion the next. If the correlation is weak, to avoid falsely dropping honest traffic the dropper has to heavily smooth out all the variation, making it sluggish to respond to a movement in the average due to an attack. We ran two experiments to find whether a good trade-off between false hits and false misses is possible:

1. The first experiment found the fastest smoothing coefficient that still introduced an acceptably low rate of false hits for honest flows.

2. Then the second experiment checked whether this smoothing was still fast enough to catch dishonest flows quickly.

We chose to use ns2 (v2.26) [ns2] to run a series of simulations with highly demanding sets of flows arriving at the dropper, some having traversed up to five potential bottlenecks. Below are the highlights of the experiments.

We implemented the multi-bit variant of congestion re-feedback carrying real numbers in TCP Reno using the initialisation and combining functions in Table 4.1. For the local congestion metric at each router m_i, we extracted the real value of the marking probability, p_b, used within the RED algorithm [FJ93, §4] before its transformation into a unary encoded sequence of marks. However, to be more demanding we still allowed TCP rate control to respond in its usual sawtooth way to unary ECN feedback and drops. We bounded headers h within $[-1, 1]$.

We implemented the dropper within the RED module, simulating packet truncation as its sanction—in order to preserve the feedback loop. We omitted flow-focused dropping as our initial aim was to assess feasibility. From Eqn 4.5 we approximated downstream congestion as $\rho_n \approx -h_n$, using $h_z = 0$.

Simulation model: We used a parking lot topology of 5 core nodes n_1 to n_5, connected by 10Mbps links. Queues at all core routers were RED-ECN in the direction of traffic (n_1 to n_5), and drop tail in the reverse direction with sufficiently large links to prevent ACK drops. The dropper ran on n_5. Traffic entered the network from all nodes n_1–n_4 and left it after a number of hops ranging across (1,2,3, & 5). Transmission delays between core nodes were 3ms, while edge delays defined a range of RTTs between 90–500ms, averaging $\tilde{2}$50ms. TCP flows through the dropper were grouped in three classes according to their typical RTT: low (L), medium (M), and high (U) of the order of 100, 250 & 500ms.

The traffic model consisted of 400 sources of which 110 were TCP-ECN and the rest UDP, with TCP traffic consistently $> 90\%$ of total bits. This reflected current [cla98] not necessarily future Internet traffic (when reduced TCP volume is expected). Packet sizes were all 1500B. We did not explicitly model HTTP but defined 100 TCP sources as FTP, uniformly varying sessions from small (20pkt) to large (1500pkt), with sources' average idle times exponentially distributed. The remaining 10 FTP sessions transferred infinite-sized files and traversed all core nodes. The UDP sources were packet trains with both ON and OFF times Pareto distributed with parameter 1.9. The resulting frequent short-lived and sporadic long-lived sessions reflected long-tailed Internet traffic. Traffic profiles were subject to random variations with RED queue utilisation varying from high 80s to low 100s percentages throughout. Traffic sources were initially generated at random uniformly between 0 and 20s; statistics collection began 30s into the 300s simulation. The (gentle) RED parameters were set to the currently recommended values relative to buffer size.

Simulation results: We used solely honest sources to find the dropper's baseline sensitivity under various conditions. Fig 5.5 is typical, leading us to use smoothing coefficients just below the knee of the curve for our later experiments with dishonest flows. That is $\gamma = 0.0005, 0.001$ or 0.002. Even in the last case truncation rates were only 1–7:10,000. We expected the subset of flows with below average RTT (L) to be better at predicting congestion, given it would have less time to change. In fact, they consistently suffered about 50% worse truncation rates than flows with average RTTs. Indeed, flows with average RTT were generally better at predicting the next round trip's congestion than both U and L

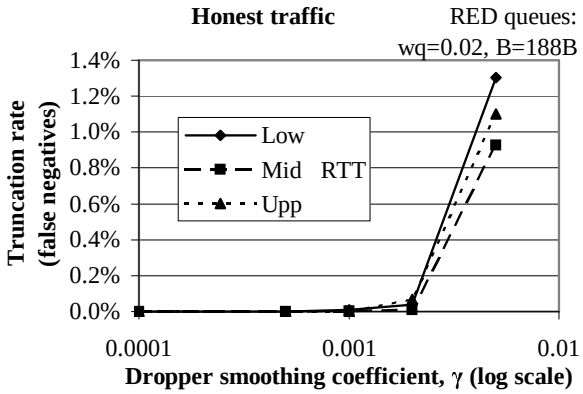


Figure 5.5: Effect of Dropper Smoothing on Truncation Rate.
For honest flows from lower, mid & upper RTT ranges (note: no focused dropper).

flows either side of them. Closed loop traffic behaviour at sub-RTT timescales is a developing field, but we are not aware of any explanation for these results.

We introduced dishonest traffic as a step under-declaring congestion by 0.1 to see how fast a large change could be detected, then ramping up to see when a small level of dishonesty became undetectable. Fig 5.6a) shows how if even 10% of flows are dishonest, high truncation peaks occur that would mark out the flow for focused treatment by a focused dropper. Note how, as levels of understatement decline, the dishonesty is lost in random fluctuations. Fig 5.6b) shows another example where 50% of flows are dishonest, thus causing strong near-immediate discrimination.

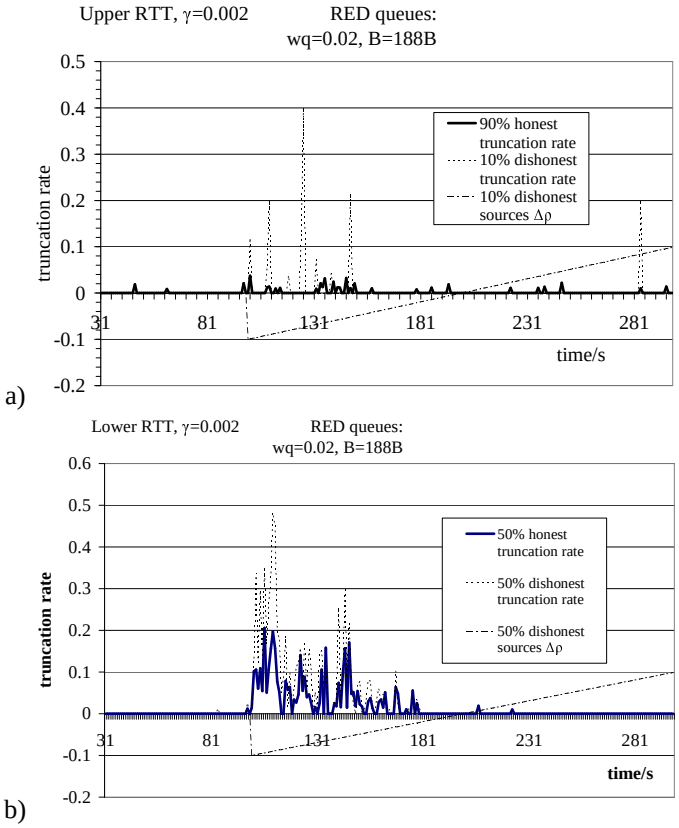


Figure 5.6: Truncation Discrimination.

With a) 10% and b) 50% of sources dishonest $\Delta\rho_{0c} = -0.1 + 0.1$ ramp (note: no focused dropper).

Re-feedback: Taking Stock

We have presented the benefits of a re-alignment of the datum of path characterisation metrics like TTL and congestion notification. Moving the datum to the destination ensures that each packet arrives at every relay carrying a view of the remaining path to be traversed by the packet, albeit a round trip delayed. Despite overhauling the underlying feedback architecture, we will see in the following chapters that a limited form of re-feedback can be deployed incrementally around unmodified forwarding elements using the existing IP header.

Once downstream information is visible, inline equipment can exercise control mechanisms that were previously impractical, such as rate policing or inter-domain traffic engineering. We describe how to police TCP's and other closed-loop rate control algorithms. Not only is it now possible to detect and remove traffic that exhibits a hostile response to congestion. It is also possible to explicitly permit applications that require such a response, perhaps given suitable payment in exchange for the enhanced quality of service.

We have introduced an incentive framework which ensures that the dominant strategy of selfish parties around the feedback loop will be to declare re-feedback honestly. It relies critically on whether malicious flows can be detected at the egress, while minimising false hits. We have simulated an adaptive dropper to show this may indeed be feasible.

Re-feedback allows senders a view of route costs, and networks a view of downstream congestion. By democratising access to path information, it enables a tussle over whether network control lies with end-points or the network [CWSB05].

Part III

Re-ECN: Unary Congestion Signal Integrity Mechanisms

Re-ECN Signal Integrity: Summary

Re-feedback of congestion signalling is a potentially powerful addition to packet networks that could be used to encourage consideration for others and curb anti-social behaviour. But it will only be truly useful if its algorithms can be designed and proved to be robust against self-interested or even malicious strategies. Previous incentive analysis of re-feedback paid no regard to current packet format constraints. This work focuses on the far more challenging case where re-feedback is applied within the constraint that no change to the standard ECN forwarding implementation of IPv4 or v6 network equipment will be necessary, implying only unary congestion signalling can be used. Further, only one extra header bit is potentially available in each packet header (an extension header is proposed for IPv6 to provide this extra bit).

The chapters in this part describe a proposed design of the re-ECN protocol and mechanisms to induce everyone to comply with it in their own interests. It aims to find the limits to the claimed benefits of re-ECN and to more rigorously prove that, within these limits, its incentives and protections work correctly.

A stated aim of re-feedback is to allow ISPs to adopt a wide range of possible actions and sanctions using the information about congestion on the path ahead that re-ECN can provide. The focus of this part is on algorithms to assure the integrity of this information. Algorithms that might *use* the information to limit or police congestion responsiveness are out of scope. However, it is very much in scope to ascertain whether the act itself of using downstream congestion signals will affect their integrity. Therefore one concrete policer design is defined to allow analysis of the system as a whole.

There follows a guide to the contents of each chapter in this part, which contains the large majority of the research in this dissertation.

Chapter 6

Re-ECN Introduction

In the interval after publication of the original re-feedback paper, the team working on re-feedback within BT thought up a number of attacks against the mechanisms that were fiendishly hard to defend against. Others in the research community came up with similar attacks and some nasty new ones too.

Bauer *et al* [BFB06] challenged the claims of incentive alignment made for re-feedback, arguing that a sender wouldn't be dissuaded by dropped packets if it had no desire to communicate with the receiver in the first place, for example in a denial of service attack. They proposed an attack that was hard to defend against and also proposed other attacks on congestion pricing in general. Further attacks have been proposed by Salvatori [Sal05], by others on various mailing lists and in denial of service research fora (e.g. unpublished attacks from Handley and Greenhalgh that we will describe later).

This part of the dissertation describes the results of a re-think of the re-ECN protocol and incentive mechanisms. Much of the original flavour remains, and the new mechanisms are, arguably, simpler.

The main difference has been in approach. Design principles have been carefully refined in order to meet a set of basic constraints. These principles resulted from an iterative process, particularly involving generalising the known attacks to understand the root of the vulnerability they were targeting. This approach allows us to test performance against the constraints and against the principles, as espoused by the design for provability movement [PLD04]. These principles are mostly specific to the dropper, although one or two have wider architectural significance, making them worth articulating in future publications. Of course, we also still aim to meet all the ambitious constraints of our original hypotheses as well.

The biggest change was at the dropper. Reluctantly, the ideal of a dropper that usually monitored an aggregate, detecting the most misbehaving flows, had to go; for the reasons concerning flow ID whitewashing, as already outlined.[1] Instead we had to accept that the dropper would need to process all behaving flows, rather than aim to process only all misbehaving flows.

Also much more thought was put into handling dummy traffic, including publication of a workshop paper [Bri06] on the subject.

No attempt is made to narrate the process that led to both the design principles and to each specific design. The design of each element actually represents the outcome of an iterative process across all the

[1] see §7.3 for the precise architectural discussion on this.

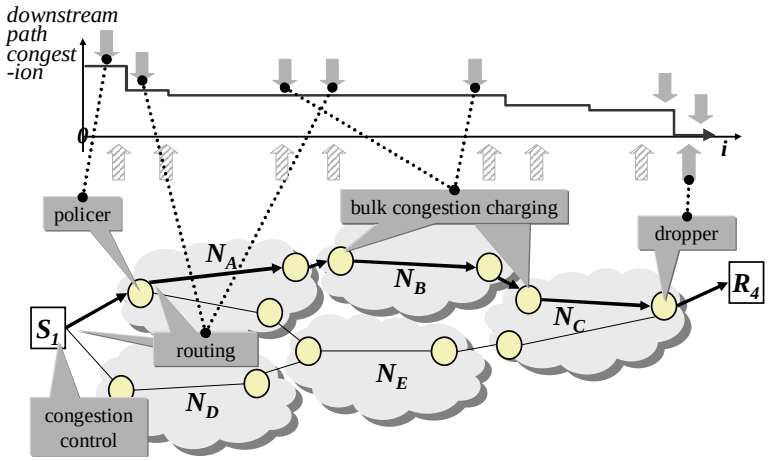


Figure 6.1: Re-ECN Incentive Framework.

elements, including occasional revisions of the re-ECN protocol that links them all together. As each new attack was identified, it revealed a gap in the understanding of the problem at hand. This led to deeper understanding of the nature of the problem, which enabled new, deeper requirements and design principles to be articulated. Then a concrete engineered design was derived from the newly framed principles.

Most of the attacks invented by the external research community were aimed at the re-feedback protocol in general, whatever the size of the protocol fields. In contrast, this part of the dissertation analyses incentives and strategies for the far more challenging case where the protocol must fit within the limited header space available in IPv4 or IPv6, without having to alter the ECN behaviour of forwarding elements. The resultant severely constrained precision brings with it new challenges and new opportunities to attack the specifics of the protocol, which we incorporate into the analysis.

The original re-feedback paper [BJCG+05] had briefly proposed a protocol called re-ECN (version-00), which could be deployed using three existing bits in the IPv4 header. However it was found to be vulnerable to attacks by unscrupulous network operators (Appendix B.1 explains how). Instead this part of the dissertation starts from the allegedly hardened (and simpler) re-ECN protocol that is specified in version 05–07 of the subsequent Internet Draft on re-ECN [BJMS09a]. Rather than reproduce it here, we have provided an abstraction of the network layer aspect in §6.1.

This part or the dissertation is organised as follows. It focuses on each of the mechanisms of the re-feedback incentive framework in turn (visualised in Fig 5.1 from §5.1, which is repeated here for convenience as Fig 6.1).

Re-ECN wire protocol: §6.1 describes the re-ECN wire protocol that links all the algorithmic mechanisms described in subsequent chapters into a complete system intended to create truth telling incentives. The full protocol details are specified in IETF Internet Draft format [BJMS09a], but here it suffices to give an abstraction of the network layer part of the protocol;

Egress dropper: §7 discusses a dropper intended to counter understatement of re-ECN signals. It would

most probably be located at the egress attachment points of the internetwork, but similar functions might be deployed at any interior location, particularly egress border routers. This chapter is the most thorough—the core of the dissertation—with pseudocode of all algorithms, analytical predictions of performance and initial experiments to verify the predictions;

Border incentive mechanisms: §8 discusses hardening of the baseline mechanism proposed in outline in §5.1.5 earlier to detect and correct understatement of re-ECN signalling before using it to support congestion-based interconnection contracts. This chapter considers strategies networks might adopt against each other, and mechanisms to ensure their incentives are aligned to help each other against attacks from the ends, even if they are commercial competitors. These functions would monitor traffic arriving at a border router of an autonomous system;

Forwarding element behaviour: §9 is a short chapter that introduces optional enhancements to active queue management (AQM) algorithms. They exploit re-ECN protocol markings in packet headers therefore they are only for elements that process the IP header. Two enhancements are proposed: i) preferential drop to improve robustness against flooding attacks and ii) marking rather than drop of flow-start packets to improve performance of short flows;

Middlebox Behaviour: §10 is another very short chapter that describes the interactions middleboxes should have with the re-ECN protocol if they hold flow state. Having reluctantly accepted that re-ECN requires flow state on its own policing elements, we wanted to provide facilities for middleboxes to handle flow state robustly and consistently as a first-class part of the Internet architecture;

Bulk Congestion Policer: §11 is yet another relatively short chapter that defines the most liberal bulk congestion policer that we believe meets all our design principles and constraints. It is intended for deployment at the ingress attachment point of an Internet access provider;

The Re-ECN System: The final chapter in this part, §12, takes an extensive view of the whole re-ECN system. First it analyses the effectiveness of the incentive mechanisms against a range of strategies and attacks designed to play off different parts of the system against each other: ends against networks, networks against ends, and ends against ends. It also includes an exhaustive check of Byzantine protocol transitions as a technique to find possible new attacks.

The main purpose of this chapter is to pull together the related parts of the system that may have been introduced in dribs and drabs throughout previous chapters. It summarises the complexity (simplicity) of the mechanisms proposed in each chapter and their outstanding vulnerabilities.

Throughout the focus is on congestion signal integrity. Mechanisms that *rely on* the integrity of these signals are out of scope. Nonetheless, we should not assume the dependency is always one-way. It may be in the interests of the end-points to understate the congestion signal and live with the resulting continuous discards of the egress dropper.

In §12.3.1 we prove that the dependency is indeed one-way. In other words, the egress dropper ensures that the dominant strategy of a sender wanting to communicate with a receiver will be to declare the

same shadow price to the network (using re-ECN) as the network declares to the receiver (using ECN), *even if* the network then uses the re-ECN shadow price to force the sender to respond to congestion.

Implementation of each element (aside from the protocol) does not need standardisation, but there are some constraints on their behaviour that do need standardisation. To help with drafting standards documents later, the relevant standards requirements are highlighted using capitals, using the terminology defined in IETF RFC2119 [Bra97].

In summary, this part proposes algorithms for each of the elements used to ensure incentives are aligned for everyone to truthfully contribute to the integrity of unary congestion marking within the re-ECN incentive framework. The purpose is both to show that the overall proposal is feasible, and to provide a relatively complete specification of an example implementation so that its effectiveness in meeting the goals stated in Hypothesis 1 (Congestion Signal Integrity) can be analysed.

6.1 Re-ECN Wire Protocol

6.1.1 Justification for Building on ECN

The re-ECN wire protocol overloads the explicit congestion notification (ECN) wire protocol [RFB01], which signals congestion in-band—within the headers of data packets. The decision to base re-ECN on ECN, was not merely to exploit backwards compatibility with an existing practice (there is precious little deployment of ECN anyway). It was because ECN reveals congestion explicitly and unambiguously in the IP header and ECN has all the mathematical and structural properties to meet our requirements, each of which are further elaborated below:

- Disambiguation of Congestion Signalling;
- Congestion Visibility to Network Nodes;
- In-Band Congestion Signalling;
- Unary Encoding.

Disambiguation of Congestion Signalling

Packet drop is a natural consequence of congestion and therefore provides an implicit signal that congestion is happening. However, a packet could be dropped for numerous reasons including:

- bandwidth congestion;
- packet-processing congestion;
- flow-state memory congestion;
- a transmission error (e.g. radio interference);
- a packet size error;
- a routing or addressing error;

- a resource consumption limit (against the sender, receiver or an intermediate network);
- some other network policy violation (perhaps even based on packet content);
- a badly designed or badly implemented middlebox;
- packet content not understood by the receiver;
- a non-existent, powered down or failed receiver.

Because congestion has considerable economic significance, the first reason for building on ECN is to have a congestion signal distinct from packet loss. A network will never be able to remove the possibility of some losses being due to congestion, but a reasonable aim would be for losses to constitute a small proportion of congestion signalling.

Congestion Visibility to Network Nodes

In the Internet architecture, drop is designed to be detected by the end-points, which notice gaps in the sequence space of their end-to-end transport protocol. Sequence numbers are not necessary for stateless packet forwarding. Therefore, by obfuscating the transport payload, end-points can hide a packet drop from all network nodes except the one that actually dropped it.

It might seem that a network would be happy to keep its congestion information private from surrounding networks, while only revealing it to the end-points causing the congestion, so they could reduce their rate in response. That would be true if a congested network could trust all the end-points causing the congestion to limit the traffic they sent, even if they were attached to other remote networks. Instead, we assume that a network will have to give its neighbouring networks and end-points incentives to limit congestion causing traffic, and neighbouring networks will then have to do likewise in turn.

Therefore, another reason we build on ECN is because we need an indication of resource congestion that is measurable by network nodes without having to inspect a packet any more deeply than the IP header.

In-Band Congestion Signalling

An ECN-enabled forwarding element marks ECN-capable packets with a probability that has a convex dependence on incipient congestion of its resources. Marking is oblivious to which flows the packets are in, but proportionately more packets will be marked in those data flows that send more packets through a resource when it is more congested.

Packets carry their markings to their destination. The destination end-point is then responsible for sending congestion feedback to the source. The feedback may or may not be sent. The source may or may not reduce its bit-rate in response to the feedback.

It may seem convoluted for a forwarding element to signal congestion in-band to the receiver, but the alternative of signalling directly back to the sender is fraught with problems. In the early Internet, Internet control message protocol (ICMP [Pos81]) datagrams of type source quench (SQ) were originally

proposed for signalling congestion notification from a router directly to the sources of the load.[2] But this was superseded by the in-band packet drop model and later by the in-band packet marking model of ECN, which is considered more robust for the following reasons:

- There is no reason to assume the addressing within the encapsulated payload of a IP packet causing congestion will be understandable to a middlebox. The middlebox may be in a tunnel that hides the original end-point identifier of the source of the packet. Packet are naturally constructed so that a successful response will be possible from the intended destination, but not necessarily from any arbitrary forwarding element;
- A response from the middle of the network to the source would need its own reliable transport, whereas in-band signals can piggy-back on the reliable delivery mechanisms of the packets they mark;
- A response from the middle of the network to the source might be rejected if it did not have the same security association with the source as the original packet (otherwise how would this message distinguish itself from denial of service traffic?), whereas in-band signals can piggy-back on the security binding of the packets they mark (whether minimal transport sequence space checks, or full cryptographic verification);

Against these points, one might argue that in-band signalling is wasteful because it requires bits to be set-aside in every packet header (re-ECN proposes an overhead of three bits in every packet). However, as long as the per-packet overhead is not too great, it is more efficient for a stressed machine to mark packets that it is already forwarding than create whole new packets to echo, which requires a transport payload to be identified and parsed.

Unary Encoding

Re-ECN signals downstream congestion using the difference between two unary signal encodings. The use of a unary encoding is to maintain similarity with the implicit congestion signal from resources that

[2] The earliest reference I can find for Source Quench is the IETF RFC famous for introducing the Angle algorithm, but also including discussion on use of Source Quench for congestion avoidance *or* recovery (about 2/3 the way through Nagle's Jan 1984 RFC896 [Nag84]). This led to the use of ICMP Source Quench as the mandatory IETF approach to congestion avoidance & control for a short while (see §2.2.3 of Postel's router requirements RFC1009 in Jun 1987 [BP87]). Even at that time, RFC 1009 allowed active queue management for congestion avoidance rather than recovery. However, it was already admitted in that RFC that SQ wasn't the ideal solution and research was continuing. The arguments against use of Source Quench that led to the change of gateway requirements are summarised in RFC1254 (§3.1, Aug 1991 [MR91]), which gives an excellent set of further references. The arguments seem to have been more a result of an unfortunate sequence of events. Essentially, there were so many different algorithms for sending source quench that it wasn't clear what a source should assume was happening when it got one - congestion onset or a router had actually run out of buffer. Packet drop, on the other hand, was a clearer indication that resources had run out. By Nov 1994 source quench was a definite 'SHOULD NOT' in the draft router requirements RFC (Almquist's RFC1716 [AK94]). The alternative approach to congestion avoidance was deliberately vague due to ongoing research, but always involved drop of packets in some form - outlined in Section 5.3.6, referring to papers of this time such as [MHR+90, Fin89, Nag85, Jac88]. SQ was described as a weak mechanism, perhaps because of the above arguments, but also perhaps because it was generally only signalled statistically to avoid congestion avalanche. Use of explicit congestion notification first appeared in [JRC87] in the DEC DNA protocol.

have to discard packets when they are congested. If a resource experiencing congestion drops packets with probability p it implicitly signals the expectation of congestion as a sequence of unary encoded numbers consisting of $n-1$ zeroes followed by a one (a drop), where $E(p) = 1/n$.

Using drop to signal '1' introduces inherent delay while the decoder decides whether a missing packet is due to reordering or drop. Explicit congestion notification (ECN [Flo94]), and the DECbit scheme [JRC87] on which ECN was based, avoided this delay by creating an explicit way for congested resources to signal a '1' but without otherwise altering the encoding.

The pragmatic aim was to ensure transports could respond to a mixture of drops and explicit signals without the strength of response having to be different for each—essentially they realised that universal agreement on the relative strength of each signal would be unlikely unless the conversion factor was 1. Other constraints that resulted in this unary encoding were:

Space Efficient: The space used in packet headers should be the minimum necessary.

Stateless: Transports only see a small subset of the packets that traverse a congested resource but the resource must not have to hold flow state to know which of the packets will be seen by which transport (anyway it cannot know which packets might subsequently be lost or re-ordered). So the encoding must be decodable from a randomly selected small subset of packets encoded by a resource.

Combinable: Packets traversing multiple congested resources will convey an encoded signal that is a known combination of the signals from each resource using the same combining function as drop; specifically, combinatorial, i.e. $p = 1 - (1 - p_1) \ldots (1 - p_n)$.

Therefore, the decision to base the re-ECN encoding on the unary signal encoding of ECN, was not merely to exploit backwards compatibility with an existing practice, it was because the rationale for that existing practice would remain valid as far as we could foresee. We cannot expect all network resources to become ECN-enabled, and even ECN-enabled resources will sometimes overrun available capacity and have to drop packets. Therefore, it will always be important to use a congestion signal like ECN that has a universally understood relationship with simple drop.[3]

The decision to use combinatorial probability to combine congestion signals from each resource ended up having a major impact on the complexity of the re-ECN system. However, if we had chosen instead to use simple addition it is likely there would have been similar or worse complexity having to cater for the possibility that protocol fields might overflow at high levels of congestion.

6.1.2 Re-ECN Network Layer Protocol

The re-ECN protocol is described in an Internet Draft proposed to the IETF [BJMS09a]. For the purposes of discussing incentives here, it will usually be sufficient to use an abstraction of the protocol at the

[3] We are working on a congestion notification encoding [Bri07a] that gives itself multiple bits per packet by using an IPv6 extension header (further extending our initial IPv6 re-ECN proposal [BJMS09a]). But it still expects most resources to signal congestion using the unary encoding of ECN. However, it outlines a way for networks to evolve towards using these multiple bits that can still work if many resources don't.

network layer without worrying about exactly which bits are set in protocol headers and without worrying about how the transport layer at the source arranges packets to be set correctly at the network layer nor how congestion is fed back from the destination to the source. These issues are specified in the above Internet Draft, but they rarely concern us here.

Re-ECN packets may be set to one of five states (enumerated in Table 6.1), each implying the bytes of the packet are 'worth' one of the three possible values, $H = -1, 0, +1$, representing the expected volume of congestion that each byte of that packet will cause downstream (or equivalently how much congestion it will experience) when averaged over a flow. Volume of congestion is defined below in §6.2. For completeness, Table 6.1 also includes the code-points used for the re-ECN states in protocol headers and it lists the legacy and unused codepoints at the bottom.

State	Notation	Worth	Variable	ECN-RE
Cautious	(+?)	+1	g	00-1
Positive	(+1)	+1	z	01-0
Neutral	(0)	0	y	01-1
Cancelled	($\pm$0)	0	c	11-0
Negative	(-1)	-1	u	11-1
Not ECN Capable	`Not-ECT`	-		00-0
Legacy ECN	`ECT(0)`	-		10-0
Currently Unused	`CU`	0		10-1

Table 6.1: Packet States in the Re-ECN Protocol.
The bottom lines tabulate legacy and unused states of the proposed extended ECN (EECN) field.
The 'Variable' column gives the name we use to represent the proportion of bits with this marking state in a set of packets.
The ECN-RE column gives the associated settings `EE-R` of the ECN field (`EE`) and RE flag (`R`).

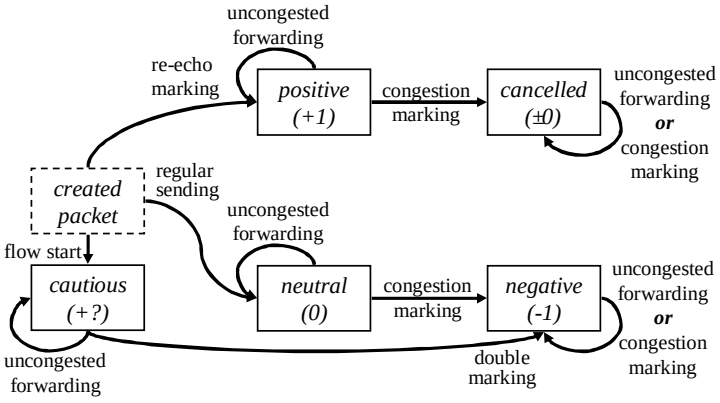


Figure 6.2: Re-ECN Expected State Transitions.

See §12.1.4 for other possible but unexpected or unusual transitions.

The re-ECN protocol arranges the five possible states of a packet so that unchanged standard ECN forwarding elements will decrement a packet's worth more often the more congested they are. That is,

a pre-existing forwarding element that complies with the ECN proposed standard [RFB01] will signal congestion by turning Neutral packets Negative (0 to -1) or cancelling the worth of Positive packets from +1 back to 0 (Fig 6.2). We term this second type of neutral as 'Cancelled' to distinguish it from a packet that started Neutral, and denote it by ± 0.

Senders may do whatever is most advantageous to them if they can get away with it, but the egress dropper (§7) is intended to encourage them to arrange that at least as much positive downstream congestion marking as negative arrives at the destination. So, normally, senders are expected to send Neutral packets (worth 0). But if a sender expects that some packets will be congestion marked to Negative (-1), it will try to balance these by sending as many bytes[4] with incremented worth (+1). All the bytes within the same packet have the same worth, so a large Negative packet would be balanced by sufficient small Positive packets, rounding up to the next whole packet to be safe.

Because of feedback delays, if the sender merely sends a packet of positive worth in response to the negative worth of each congestion feedback event, the cumulative balance will always be either negative or zero, so the moving average will always be slightly negative. Given the system is arranged so that it is in the sender's interest to maintain a balance of at least zero, the sender will want to build up some credit at the start of a flow to cover the risk of the largest packets it sends being marked Negative. Then, to maintain balance, it can re-echo the feedback from each congestion event (-1) by sending a Positive (+1) packet of the same size. The actual amount of positive credit to send at the start of a flow or hold during the flow is deliberately left as a dilemma for the sender [BJCG+05, §3.3.3]. Senders more sensitive to the risk of being sanctioned for allowing a negative balance will send more initial credit and be less concerned about recovering any credit near the end of the flow. In §7.7 we derive a default that would be reasonable to standardise into a flow control protocol like TCP.

Because the worth of the packets at the start of the flow may be redundant, or at least overstated, we introduce a fifth state that is also worth +1, but which a sender can use when it is being cautious at the start of a flow, rather than responding to actual congestion. We call this cautious positive, or 'Cautious' for short, and denote it by '+?'. It will be seen as we proceed that being able to distinguish positive worth from cautious positive worth is very useful in a number of respects.

Currently we define the value of a cautious positive byte as +1, but we recognise that it may be preferred in future to allow the relative worth of these Cautious packets to be decoupled from regular Positive packets, perhaps being determined by a separate market.

The overall effect is that senders have to pre-load enough positive downstream congestion (cf. credit) into packets to survive being decremented as they experience congestion (cf. debit), without running out of pre-loaded downstream congestion (cf. avoiding debt). The mechanisms described in this dissertation give everyone the incentives to ensure they follow such a strategy.

The protocol has been deliberately arranged so that the downstream congestion-volume caused (or experienced) by a set of packets can be metered in bulk, at least approximately, by a network element or an end-point, simply by counting the volume of packets marked +1 and subtracting the volume marked

[4] Not including headers that encapsulate the lowest internetwork (IP) layer.

-1.

6.2 Notation, Definitions and Metrics

Before we define upstream and downstream congestion, we need to define some general congestion-related concepts:

Definition (Congestion). *Both instantaneous congestion $m_i(t)$ of resource index i and path congestion $p(t)$ are defined in §4.A.1 (Definitions 4.1 & 4.2); Units: Dimensionless* [5]*;*

Definition 6.1 (Congestion-bit-rate). *Congestion-bit-rate is the interaction between bit-rate and congestion; The rate at which congestion marked bits are generated or experienced; the instantaneous product of congestion and the bit-rate of flow f, either generated by resource i as $m_i(t)x_f(t)$ or experienced over a path as $p(t)x_f(t)$ or over an aggregate set F of flows $\sum_{\forall f \in F} p(t)x_f(t)$; Typical units: [b/s];*

Definition 6.2 (Congestion-volume). *Congestion-volume is congestion-bit-rate integrated over a time period T, either for one flow f or the set F of flows $\int_T \sum_{\forall f \in F} p(t)x_f(t)dt$. The accumulated volume of congestion marked bits; Typical units: [b];*

Definition 6.3 (Congestion-intensity). *Congestion-intensity is average congestion-bit-rate over a period T; the average rate at which congestion marked bits are generated or experienced, $\int_T \sum_{\forall f \in F} p(t)x_f(t)dt/T$; Typical units: [b/s];*

In the following we will start from the marking probabilities m_i of each resource i along the path. Then we will derive an expression for the expectation of downstream congestion at an intermediate point along the network path after the ith resource, solely in terms of proportions of re-ECN markings visible locally at that resource (assuming honest marking compliant with the re-ECN protocol).

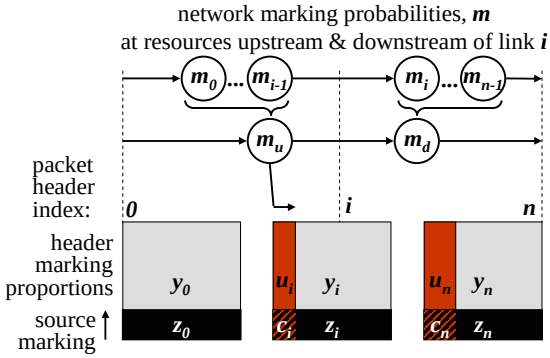


Figure 6.3: Re-ECN Markings at Intermediate Points Along a Network Path.

Upstream and downstream of this point there are congested resources that mark the ECN field with probability m_u and m_d respectively. These are the probabilities that an unmarked packet would become

[5] To check units it can help to think of a marked bit as a distinct unit from a bit. Then instantaneous congestion has the units of [(marked b)/b] or just [mark].

marked if it traversed the combination of all resources upstream or separately the combination of all resources downstream. This models the system as if there were just one resource upstream and one downstream. In other words, the combinatorial probability of ECN marking from resource 0 to $i-1$ is

$$m_u = 1 - (1 - m_0) \dots (1 - m_{i-1})$$

and the combinatorial probability of ECN marking from resource i to $n-1$ is

$$m_d = 1 - (1 - m_i) \dots (1 - m_{n-1}).$$

The top half of Fig 6.3 illustrates this congestion marking. The bottom half illustrates the resulting proportions of markings in packet headers. The area of each square represents all the packets passing the corresponding point in the path, and the different sub-areas within each square represent the proportion of each packet marking. The source determines the vertical proportions of markings, dividing the square into upper and lower parts. Then as packets traverse the path, each congested network element marks packets without regard to pre-existing markings. Therefore it superimposes the horizontal proportions of markings that divide the square into left-hand and right-hand parts. The left-hand part starts at zero size and grows as more congestion marking is experienced.

We index proportions of packet markings with the resource they are about to arrive at next. The index zero is used for the origin header before it experiences congestion m_0 at the zeroth resource (which may be within the network stack of the origin machine).

We assume for now that the source only initiates Neutral or Positive markings and we ignore Cautious markings at this stage to avoid them cluttering up the explanations, equations and diagrams.[6] It can be seen from Fig 6.3 that the re-ECN protocol has been arranged so that network marking (vertical divisions) can be orthogonal to the original source markings (horizontal divisions). Network resources apply congestion marking to packets regardless of the marking they already carry, so markings accumulate by combinatorial probability. Using this fact, Eqn (6.1) tabulates the fractions of each marking at each point in the network in terms of the Positive marking originally introduced by the source z_0 and the upstream and downstream network marking probabilities m_u & m_d.

Eqn (6.1) defines the fractions of each re-ECN marking introduced into the network by the source in the left-hand column. It then defines the resulting fractions at an intermediate point i and at the end of the path, having experienced congestion m_u and then m_d.

Positive:	z_0	$z_i = (1 - m_u)z_0$	$z_n = (1 - m_u)(1 - m_d)z_0$
Cancelled:	$c_0 = 0$	$c_i = m_u z_0$	$c_n = \big(1 - (1 - m_u)(1 - m_d)\big)z_0$
Neutral:	$y_0 = 1 - z_0$	$y_i = (1 - m_u)(1 - z_0)$	$y_n = (1 - m_u)(1 - m_d)(1 - z_0)$
Negative:	$u_0 = 0$	$u_i = m_u(1 - z_0)$	$u_n = \big(1 - (1 - m_u)(1 - m_d)\big)\big(1 - z_0\big).$

(6.1)

Fractions of each re-ECN marking at the start ($i = 0$) and end ($i = n$) of a path and at an intermediate point i.

[6] They are properly taken into account in §7.4.5

If the transport complies with the re-ECN protocol, the source will introduce as much Positive marking as the sum of Negative and Cancelled markings arriving at the destination in the previous round trip. Then, assuming congestion is stationary and therefore rises as much as it falls, on average:

$$z_0 = u_n + c_n. \tag{6.2}$$

We now use this formula to make some preparatory substitutions from Eqn (6.1) towards our goal of expressing downstream congestion m_d solely in terms of marking proportions at link i.

$$\begin{aligned} z_0 &= 1 - (1 - m_u)(1 - m_d). \\ z_i &= \big(1 - m_u\big)\big(1 - (1 - m_u)(1 - m_d)\big). \\ u_i &= (1 - m_u)(1 - m_d)m_u. \\ z_i - u_i &= (1 - m_u)m_d. \end{aligned}$$

We introduce the notation v_i for recent congestion downstream of the ith resource, where

$$\begin{aligned} v_i &= m_d \\ &= \frac{z_i - u_i}{1 - m_u} \\ &\approx z_i - u_i; \qquad m_u \ll 1 \end{aligned} \tag{6.3}$$

One can think of v_i as if it represents a proportion of **v**irtual header markings, i.e. the difference between the proportions of Positive and Negative markings.

The above approximation removes the slight inflation factor $1/(1 - m_u)$. When the condition for approximation does not hold, we want to put the precise expression for downstream congestion solely in terms of locally visible markings. Using Eqn (6.1) this can be done either in terms of recent local Positive and Cancelled markings or in terms of recent local Neutral and Negative markings:

$$\begin{aligned} v_i &= \frac{(z_i - u_i)z_0}{z_i} \\ &= (z_i - u_i)\left(1 + \frac{c_i}{z_i}\right); \end{aligned} \tag{6.4}$$

$$\begin{aligned} v_i &= \frac{(z_i - u_i)(1 - z_0)}{y_i} \\ &= (z_i - u_i)\left(1 + \frac{u_i}{y_i}\right). \end{aligned} \tag{6.5}$$

There are two independent formulae for the same thing because there is deliberate redundancy in the re-ECN encoding. This fact will be used later (§8.2.7) to double-check against cheating.

The role of the re-ECN protocol (§6.1.2) is to provide packet states that can meter recent downstream congestion v and downstream congestion-volume V.[7] We will now define what exactly would be measured to meter these characteristics, taking account of varying packet sizes.

[7] Strictly a network provider could use v to signal a downstream shadow price, but we loosely use the term downstream 'congestion' unless it is important to make the distinction (see also §12.1.2).

We must emphasise that the initial definitions given here use the above pragmatic approximation for a downstream congestion metric, which only hold for low levels of upstream congestion (Eqn (6.3)). We will return to this point in §8.2.5 where the precise correction factors just derived are used to close off a vulnerability, which can otherwise be exploited whatever the level of congestion.

The 'downstream congestion-volume' of a sequence of packets crossing a point in a network is the volume of congestion these packet cause (or equivalently experience) downstream. It can be measured in terms of re-ECN markings as

$$V_J \approx \sum_{j \in J} s_j H_j, \tag{6.6}$$

where s_j and H_j are the size[8] and 'worth' of each packet indexed conceptually (but not actually) by j and J is the sequence of indices j.[9] Note J can be any aggregate passing a point, irrespective of flow identifiers.

This compares with the offered volume over packet sequence J,

$$S_J = \sum_{j \in J} s_j, \tag{6.7}$$

The worth H of a packet is an attempt to mark the instantaneous downstream congestion caused by (or equivalently, experienced by) each byte in the packet. Note the use of bytes rather than packets—all bytes in a marked packet are defined as marked bytes. If we had enough bits per packet, we could put a real number for downstream congestion in each packet. Then instantaneous downstream congestion $v_{j,J}$ at packet index j in the sequence $J = 0 \cdots j$ would be the increase of downstream congestion-volume with respect to total offered volume,

$$v_{j,J} \approx \left. \frac{dV_J}{dS_J} \right|_j, \tag{6.8}$$

But because we have limited space per packet, it is only meaningful to measure recent downstream congestion over a finite volume of offered load using moving averages. Then recent downstream congestion up to packet index j in the sequence J is

$$\begin{aligned} v_{j,J} &\approx \left. \frac{\Delta V_J}{\Delta S_J} \right|_j \\ &\approx \frac{A^j_{k=0}(s_k H_k, a)}{A^j_{k=0}(s_k, a)}, \end{aligned} \tag{6.9}$$

where the function $A^j_{k=0}(X_k, a)$ gives the moving average of some characteristic X_k of each packet over the sequence of packet indices $0 \cdots j$ and a is the discounting factor of the moving average. The two moving average functions must be the same and both averaging functions must use the same discount factor so that the characteristics of each packet are weighted identically in the numerator and denominator.

The nub of these definitions can better be seen by suppressing all the sequence and index notation and the discount factors. Then downstream congestion-volume,

$$V \approx \sum sH, \tag{6.6′}$$

[8] Packet size includes network layer but not lower layer headers.

[9] Note that these subscripts denote the indices of packets not, as earlier, the index of the point on the path where packets are being measured.

and recent downstream congestion,

$$v \approx \frac{A(sH)}{A(s)}. \tag{6.9′}$$

Note the following:

- $A(sH)/A(s) \neq A(H)$ because H and s vary independently;
- Re-ECN encodes H into just one of the three possible encoded states (-1, 0 and 1) per packet. So the discounting factor a must be sufficiently small to give consecutive positive and negative marks a reasonably similar weight, even if interspersed by many zeros;
- The measure of congestion-volume V is deliberately independent of packet order. However *recent* downstream congestion v will be unavoidably sensitive to packet reordering, given packet order determines recency.

Chapter 7

Re-ECN Egress Dropper

7.1 Dropper Terminology

We use the term ‘drop’ and ‘dropper’ for brevity, but other sanctions may be applied depending on policy, such as payload truncation. In scenarios where trust is expected, the sanction may simply be to raise a management alarm reporting the excess of traffic above that which an honest source would have sent.

We use the term ‘egress dropper’ loosely, because the dropper will often be located at the egress of a domain and there must be a dropper at the egress of the scope of an internetwork protected by re-ECN. However, it would be valid (but not usually necessary) to locate a dropper at any network node to detect and sanction negative flows.

The term ‘flow identifier’ means the identifiers common to a sequence of packets at whatever granularity the source reveals, including within the network layer payload. §7.3 discusses our reluctant decision to detect flows in the network and our efforts to ensure only the barest minimum of per-flow constraints.

Re-ECN only needs to classify packets into ‘flows’ to check that downstream congestion in packets passing a point is consistently non-negative, given a consistently negative flow implies the transport must be misbehaving. Ideally this requires flows to be examined at the finest granularity possible. Otherwise, if only aggregates of flows are examined, a number of slightly positive microflows might mask a minority of highly negative microflows. However, in certain circumstances dependent on policy[1] it would be reasonable to detect whether an aggregate was negative before examining flows within the aggregate to find which ones to sanction. If the finest granularity visible at the edge of a network is an aggregate (e.g. an IPsec encrypted tunnel), the only option open to the network is to treat the aggregate as one flow, whether for passive detection or active sanction.

The specific identifiers used for a flow depend on which ‘next header’ field is present in the IP header. For instance, if the next header is TCP, DCCP or UDP, a flow can be defined by the five-tuple of source and destination IP addresses, the protocol ID in the IP header and the source and destination port numbers in the transport header. If the next header is IPsec AH or ESP, the flow is defined by the four-tuple of destination IP address and next header field in the IP header and the security parameter index (SPI) in the AH or ESP header. For extra entropy the IPv6 Flow Label SHOULD be used in combination

[1] Where no flows were expected to be positive, e.g. at the egress of the internetwork.

with the usual 5-tuple flow ID parameters where available (see §7.5.3). Alternatively, other flow-IDs may be used, or other ways to identify flows that are as yet to be defined.

If the next header field is unrecognised, only the 3-tuple of IP addresses and protocol ID can be used. But in general, a common 3-tuple doesn't imply that all packets belong to the same contractual entity, given tunnels, NATs and multi-user hosting machines are common on the Internet. Also, whereas all packets with a common flow ID will generally follow the same route, all packets with a common 3-tuple need not. If different proportions of one flow pass through different droppers at the same time, dropper behaviour will be very unpredictable. Switching to a new dropper mid-flow is discussed in §A.1.

If any identifiers are unrecognisable, the packet is given a default flow ID we call 'Bulk' in common with all packets not belonging to a well-behaved flow (see §12.2.2 on Forward Compatibility).

7.2 Dropper Behaviour Constraints

The egress dropper for unary marking needs to satisfy the following design constraints:

Minimal False Hits: It SHOULD introduce minimal false hits for honest flows;

Minimal False Misses: It SHOULD quickly detect and sanction dishonest flows, preferably at the first dishonest packet;

Transport Oblivious: It MUST NOT be designed around one particular rate response, such as TCP's, or one particular resource sharing regime such as TCP-friendliness [FHPW03], given an important goal is to give ingress networks the freedom to allow different rate responses and different resource sharing regimes [GK99b, Bri07b]—unilaterally without coordinating with downstream networks;

Sufficient Sanction: It MUST introduce sufficient loss in goodput so that sources cannot play off losses at the egress dropper against higher allowed throughput at the ingress policer [Sal05] (§12.1.1);

Manage Memory Exhaustion: It SHOULD be able to counter state exhaustion attacks. For instance, if the dropper uses flow-state, it should not be possible for sources to exhaust its memory capacity by gratuitously sending numerous packets, each with a different flow ID.

Identifier Accountability: It MUST NOT be vulnerable to 'identity whitewashing', where a transport can label a flow with a new ID more cheaply than paying the cost of continuing to use its current ID [FR98];

7.3 Dropper Design Principles

The following design principles have been developed for the dropper design, to satisfy the above constraints (Table 7.1 shows which principle satisfies which constraint). They are introduced briefly below, then some are discussed at greater length in the subsequent sections:

1. Source responsibility for delay allowance—aims to make false hits the responsibility of the source;
2. Sanctions proportionate to the crime (equivalence with honesty)—defines hits and misses, without being transport specific, at least within the bounds of the dropper-policer trade-off (§12.1.1);

Required Constraint	Design Principle
Minimal False Hits	Source Responsibility for Delay Allowance
Minimal False Misses	Sanction Proportionate to Crime (Equivalence with Honesty)
Transport Oblivious	
Sufficient Sanction	
Manage Memory Exhaustion	Aggregate Flow State unless Positively Flagged
Identifier Accountability	Rely on Flow ID Uniqueness not Reachability

Table 7.1: Which Design Principle Satisfies Which Constraint.

3. Aggregate flow state unless positively flagged—ensures the dropper need only allocate memory for flows that have given at least as much as they have taken;

4. Rely on flow ID uniqueness, not reachability—limits scope of accountability for resource usage to an identifier, not to the entity behind the identifier.

Sanctions Proportionate to the Crime (Equivalence with Honesty). An honest source complying with the re-ECN protocol will aim to ensure that the volume of Positive marked bytes is no less than the volume of Negative marked bytes at the egress, at least on average. The egress dropper SHOULD drop sufficient traffic so that delivered traffic conforms to that which an honest source would have sent. A dropper algorithm to achieve equivalence is derived in §7.3.1.

Neutral packets do not contribute to the balance at the egress, but they are not immune from being dropped in order to comply with the equivalence principle.

An alternative might have been to detect when the balance of a flow at the egress drops below a threshold then drop all its traffic. However, because the crime in this case has a measurable level, we can neutralise it rather than punish it. It is also safer to adopt the principle that the sanction should be proportionate to the crime, otherwise attackers may be able to deflect amplified sanctions onto others.

Merely neutralising misbehaviour creates no deterrent effect; therefore proportionate sanctions would be ineffective if only applied on a randomly sampled basis. The optimal attack strategy then would be to misbehave as much as possible, only behaving when forced to. Anyway, as soon as a misbehaving flow detects it has been picked for punitive sanctions, it can just take on a new whitewashed identity [FR98].[2] The improbability of universal worldwide source address validation [BB05] drove us to give up on source addresses having any meaning other than as a label for a sequence of packets, which in turn led to the need for proportionate sanctions and continual vigilance rather than occasional deterrent punishments.

Therefore any punishment should be applied *in addition* to neutralising misbehaviour. Neutralising misbehaviour acts as a baseline suitable for the generic internetworking layer, without precluding punishment layered on top under local policy control.

Source Responsibility for Delay Allowance. The re-ECN protocol is designed on the principle that a source carries its proportion of the risk that its packets will encounter unforeseen congestion in the round

[2] All per-flow bottleneck policer proposals suffer from this flaw.

trip time after they are sent. It is the source's responsibility, not the dropper's, to allow for feedback delay. The alternative of the network robustly determining each transport's RTT seemed infeasibly complex.

An honest source will avoid the average of recent congestion going negative under reasonable network conditions by providing sufficient positive credit at the start of the flow to allow for unforeseen events during the feedback delay. Then, as feedback informs it that congestion costs are being incurred, it will match the feedback with an equal amount of re-feedback bytes to maintain the same level of credit. Thus, during a flow, the markings of bytes should balance to no less than zero.

The dropper then has the minimal job of checking that the recent balance of a flow remains non-negative. This design choice reluctantly requires per-flow processing and state within the network. Once one accepts that it is untenable to expect hosts to co-operate in determining their own share of resources, one has to face the question of whether the network can arbitrate resource sharing with only per-packet, not per-flow processing.

In 1998 it seemed Kelly had finally devised a flow-oblivious network mechanism using ECN-based congestion pricing [KMT98]. But sanctioning the receiver created insurmountable problems in practice [BJCG+05, §3.1]. We tried many avenues to alleviate the 'receiver-pays' problem while keeping a flow-oblivious network, but re-ECN has been the closest we have been able to come to our goal.

Unlike other schemes to police congestion responsiveness[3], the network need not make any judgements about rate control behaviour, only about congestion signal integrity. The network can still leave end-points free to choose how aggressive their rate control behaviour will be, and to weigh up how much credit they are willing to spend to protect a flow from an unusual burst of congestion.

So, in summary, re-ECN can be transport-oblivious but unfortunately not flow-oblivious. Unless some completely flow-oblivious but practical network mechanism can be invented, it seems that the minimum generic network function [SRC84] will have to include per-flow testing of congestion signal integrity, at least on egress nodes in environments lacking trust. But the network doesn't need to allow for round trip delays in these signals, which can remain the responsibility of the end-points.

Aggregate Flow State unless Positively Flagged. The egress dropper cannot determine whether each packet is correctly congestion marked on its own, because of the need to allow for variation of expected downstream congestion around zero due to round-trip delays.[4] To determine whether to sanction each packet the dropper must therefore hold state on each flow. Given the dropper is only absolutely required at the egress edge of the network, we believe this is at least feasible, though not desirable.

Holding flow state could make the dropper vulnerable to state exhaustion attacks from malicious sources, because they can give a new flow ID to each packet. However, we arrange that a previously unseen flow ID alone doesn't make the dropper allocate new flow state; the source must also send a

[3] They fall into two classes: i) bottleneck policers that detect flows taking much greater than an equal rate; after Floyd and Fall [FF99]: Stabilized RED (SRED [OLW99]), CHOKe [PPP00], RED with Preference Dropping (RED-PD [MFW01]), Least Recently Used RED (LRU-RED [Red01]), XCHOKe [CCG+02], and Approx. Fair Dropping (AFD [PBPS03]) and ii) schemes that aim to isolate flows (or aggregates) from misbehaviour by enforcing equal or weighted sharing of bottlenecks based on fair queuing (FQ [Nag85]) or weighted fair queuing (WFQ [DKS89]).

[4] Initially we believed this was a limitation of unary marking, because unary markings are only meaningful when accumulated over multiple packets. However multibit markings would suffer the same limitation due to round trip delay.

Cautious (or Positive) packet for the dropper to consider a flow to be new. Because a packet must flag that it wants to be considered as a new flow, it can be held accountable for the flags it raises as it crosses every network trust boundary from the sender onwards (by the ingress policer or border mechanisms in the wider incentive framework of Fig 6.1).

Therefore, the dropper's vulnerability to state exhaustion attacks is limited because sources must consume their own 'credit' for the privilege of having the egress dropper allocate memory to allow through a new unique flow unimpeded. In addition, the regular re-feedback rules ensure each source must keep each flow's state alive, by regularly include positive worth in further packets at a rate matching the congestion markings from its network path.

Rely on Flow ID Uniqueness, not Reachability. The egress dropper uses flow identifiers solely as labels—to isolate flow IDs from each other—not to push anything back towards the source. So a flow can be made to behave correctly as long as its identifier is currently unique—it does not need to be labelled 'correctly' (i.e. with the reachable source of the packet). This is a useful property that is robust against a source that has no desire to receive a reply and therefore need not correctly identify itself in order to send datagrams to something else. Consequently, the re-ECN protocol is insensitive to whether identifiers are set honestly or correctly, except in so far as they carry a packet to the particular egress dropper in question.

Together, the two principles 'ignore reachability of the origin address' and 'source responsibility for delay allowance' ensure that packets can, in themselves, be held accountable for their resource usage.[5] As packets cross an internetwork, the packets themselves carry accountability information from the sender to the first network, then from the first to the second and so on. Each party can be made accountable to the next using the packets passing between them as intermediaries, because the packets themselves contain sufficient information about rest-of-path congestion.

The egress dropper defines a flow solely by the uniqueness of its flow identifiers while it is active, and it will not recognise a new unique flow without some investment of resources by the source (see 'Aggregate Flow State unless Positively Flagged' above). Therefore we are only concerned by spoofed source identifiers if they manage to break the uniqueness property of someone else's flow, by mimicking all its identifiers (see §7.5.3).

Note that this principle implies that the dropper will never carry over a balance (whether positive or negative) from one flow to another, even if the flow appears to belong to the same end-points—because it attaches no meaning to end-point identifiers other than their uniqueness.

7.3.1 Proportionate Sanctions (Equivalence with Honesty)

At this initial stage, we are only concerned with the integrity (honesty) of Positive markings. We assume that Negative markings are introduced as packets traverse the internetwork, with no bias towards marking any particular types of packet more than others (§12.1.2 analyses this assumption). Then an honest

[5] Multibit downstream congestion information is not necessary for this property, due to 'source responsibility for delay allowance'.

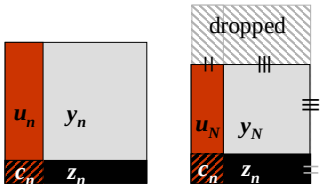


Figure 7.1: Misbehaving Traffic a) Before and b) After Discard by the Egress Dropper. Its output aims to be equivalent to traffic from an honest source.

source is defined as one that never tries to send less Positive bytes than the Negative bytes introduced by the network, and an honest flow is defined as the result.

So as not to disrupt the main flow of the explanation, we assume the source doesn't initiate any Cancelled packets (± 0), which only result from the network randomly marking packets that happen already to be Positive (+1). Until later, we also count Cautious packets (+?) in with Positive (+1). We defer deeper questioning of these assumptions to §7.4.

While the recent balance between positive and negative traffic for a flow is in credit, the dropper does nothing. But if the recent balance is in debt, by the principle of equivalence with honesty, the dropper aims to reduce the flow's traffic as a whole so that the recent volume of Negative marked traffic after drop will be no greater than the recent volume of Positive traffic before drop.

Fig 7.1 shows the concept of equivalence with honesty graphically in two dimensions, using the same visualisation conventions as in Fig 6.3. It illustrates volumes of traffic markings dividing up a) the larger square before and b) the smaller square after drop. The notation for the fractions of each marking is the same as that defined in Eqn (6.1). After the dropper, Positive and Negative markings balance, while before there is less Positive than Negative. The area of the enclosing rectangle on the right that includes the traffic discarded by the dropper (grey hatched) is the same as that of the left hand square, as are the areas of each marking within them.

By equivalence with honesty, the same proportion of Neutral markings should be dropped as Negative. This is because congestion marking should be applied randomly by upstream networks. So if we would expect to see less Negative marks from an honest source, we would also expect less traffic that would become Negative if marked, i.e. less Neutral traffic. Then, if this reduced traffic had been that sent by an honest source, as many bytes would have been marked Negative as the dropper actually received marked Positive, which leads to a dropper output that is equivalent to that from honest behaviour.

In Eqn (6.6) we defined downstream congestion-volume V so that it could be measured as the volume of Positive marked bytes less the volume of Negative marked bytes during the lifetime of a flow. An honest flow will try to ensure $V \geq 0$ throughout its lifetime. Similarly to Eqn (6.9′), we define the fraction of Positive marked bytes arriving at node index i in the recent past using the function $A(\cdot)$ for a moving average:

$$z_i = \frac{A(sH_i^+)}{A(s)} \tag{7.1}$$

and the fraction of Negative bytes similarly[6] as

$$u_i = -\frac{A(sH_i^-)}{A(s)}. \tag{7.2}$$

What 'recent' means is determined by the discounting factor a of the moving average, which we will tune to minimise false misses and false hits (§7.8). We have suppressed this factor from the notation along with all the extraneous detail of packet indexes and sequences, as was done between Eqn (6.9) and Eqn (6.9′).

From Eqn (6.3), if the moving averages of packets marked with each worth are each measured separately as they pass a point, they will sum to a moving average of recent downstream congestion,

$$v_i \approx z_i - u_i. \tag{6.3}$$

The approximation becomes an equality when testing whether $v_i = 0$.

If the recent balance of markings at the egress dropper is in debt ($v_n < 0$), the dropper makes sure delivered traffic is equivalent to that from an honest source by reducing the recently delivered fraction of Negative bytes to $u_N = z_n$, where indices n and N denote the points before and after the dropper.[7] But it only drops traffic if a flow's lifetime downstream congestion-volume $V_n < 0$ as well. This allows an honest flow to post a credit to protect itself from bursts of congestion that would otherwise make its recent balance look negative. So the drop probability of Negative bytes π_u should be such that

$$\begin{aligned} u_N &= u_n(1 - \pi_u) \\ &= z_n \\ \implies \pi_u &= 1 - \frac{z_n}{u_n}; \qquad V_n < 0, v_n < 0. \end{aligned} \tag{7.3}$$

The drop probability of Neutral bytes π_y should be the same as that of Negative bytes π_u by our earlier arguments. We don't drop any Positive packets, because they act as the baseline that would have been sent by an honest source. We also don't drop Cancelled packets, but only because we have another way to normalise the proportion of Cancelled packets that enter the dropper, which will be introduced later (§7.4.1). In summary the dropper should discard each packet marking with probability:

$$\begin{aligned} \pi_y &= \pi_u; \qquad V_n < 0, v_n < 0 \\ &= 1 - \frac{z_n}{u_n}, \end{aligned} \tag{7.4}$$

$$\begin{aligned} \pi_z &= 0, \\ \pi_c &= 0. \end{aligned} \tag{7.5}$$

§7.6 gives simple algorithms that implement these drop probabilities.

Fig 7.2 visualises the dropper reducing the various packet markings sent by an attacker understating downstream congestion. It shows re-ECN markings along a network path ending in an egress dropper.

[6] Except we negate the definition so that u_i can be numerically positive.

[7] This does not remove congestion notification signals, because packet drop is always at least an equivalent notification to marking.

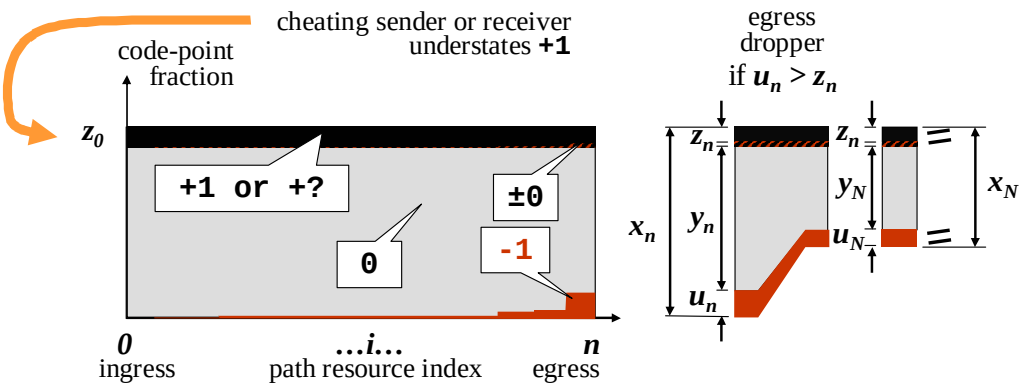


Figure 7.2: Egress Dropper for Unary Re-ECN Marking.

The transport is understating Positive markings so the dropper is removing sufficient Negative and Neutral markings so that it delivers traffic that is equivalent to that an honest source would have sent. If instead the dropper discarded packets randomly to reduce the flow as a whole, the proportion of Positive to Negative markings would incorrectly remain the same.

The converse of proportionate sanctions as a goal is stealth as a non-goal. That is, the dropper needs to actively sanction offending traffic, therefore it cannot hide its own existence. It doesn't matter if the dropper is predictable if it predictably prevents any misbehaviour. Consequently, if the dropper allows false misses (see §7.7.2), attackers will be able to probe its limits to find and exploit its trigger level.

7.3.2 Source Responsibility for Delay Allowance

The re-ECN protocol is designed on the principle that a source carries the risk of its packets encountering unforeseen congestion in the round trip time after they are sent. This allows a source the freedom to increase its window of packets in flight more aggressively than others, but it must take responsibility for the risk of exercising this freedom.

The aim is to move away from the traditional TCP constraints that all sources must increase their window equally per round trip time, but they may increase exponentially at the start. Not only does the traditional rule constrain new behaviours (e.g. attempts to improve flow start performance), but it is also being routinely flouted by applications that open multiple flows simultaneously. We need to recognise diverse application needs, rather than pretending that everyone is complying with impractically restrictive rules [CWSB05].

Because many traffic sources will all simultaneously be taking the risk of increasing their window of packets in flight, each source should carry its proportion of the risk, which depends on currently observed path congestion, its current bit-rate, and its increase in bit-rate per round trip time.

"Carries the risk" means the source pays for the risk in advance of the round trip. This distinguishes the re-ECN protocol from the form of congestion pricing suggested by Gibbens & Kelly [GK99b] in which congestion is only paid for after the event—if it happens. Under re-ECN, if an unexpected burst of congestion happens, and if the source hasn't paid to cover this risk in advance, it risks packets being

dropped. This is more generic than congestion pricing, because the source can either choose to pay credit to avoid the risk of drop, or choose to suffer some unexpected drops by spending less on credit. Once a burst of congestion has happened, the source only has to pay to balance its cost if it wishes to top up the credit that turned out to be necessary in order to to continue the flow with the same cover.

This re-ECN design choice translates into the dropper design principle that the source, not the dropper, should allow for feedback delay.[8] But, because a re-ECN source only pays its conservative expectation of congestion within the next window, this may not be sufficient to cover all possible unforeseen events within the round trip. If an unusually high burst of congestion occurs, the dropper will not deliver packets if they are not covered by sufficient credit. In other words, the source cannot expect the dropper to give credit whenever the source's own credit turns out to be insufficient. Thus a flow risks losing packets in proportion to its underestimate of the worst-case cost within a round trip.

A flow that knows it is about to complete, particularly as it approaches the final window, may recoup some of the credit it invested by not re-inserting all feedback.[9] But, even if a flow continues to re-insert feedback until the last, a debit may be left at the end of a flow if heavy congestion is experienced in the last window. Nonetheless, over many flows, it is highly unlikely that all the possible debits at the end will exceed the credits added at the start.

These considerations have an informal microeconomic interpretation. On the supply side, all the credits that turn out to be paid unnecessarily can be considered as funding sufficient extra capacity to help absorb the prevailing proportion of bytes in round trips in which flows start (or rapidly increase) versus those in steady state. While on the demand side, making sources responsible for the risk of congestion during round trip delays ensures that sources won't increase their window aggressively unless they are willing to pay their proportion of the resulting extra risk of congestion.

Conditions for stability have been derived that place limits on flow aggressiveness [KMT98, KV05]. Making sources responsible for the risk of congestion will not directly prevent them overstepping this limit. But this stability limit does seem to present a natural step in the shadow price, which should more strongly discourage sources from causing instability. If a source is aggressive enough to cause instability, the resulting oscillations will disproportionately increase the expected price. Although all competing sources will suffer, not just the one causing instability, this price step should strengthen the barrier against behaviour so aggressive it causes instability.[10]

7.3.3 Dropper State Management

Middlebox Flow-State: Pros and Cons

The design of the re-ECN egress dropper reluctantly introduces the minimum per-flow function into the network that we could achieve (see §7.3). Although re-ECN avoids embedding any subjectivity or value-judgement in the network about the required congestion responsiveness of flows, it does require the network to implement one per-flow test of behaviour: whether the balance of positive and negative

[8] However, the dropper is responsible for allowing for any delay introduced by its own algorithm (e.g. a moving average).

[9] The strategy to adopt at the end of a flow can be derived from the analysis in §7.7, but it does not affect the analysis presented here (so its details are left for further work).

[10] Except if a source is sending dummy traffic to be malicious.

bytes is in credit.

No matter how minimal this per-flow function, it necessarily requires per-flow state in the network. The 'shared fate' design principle of the Internet [Cla88] advises that the end-points alone are best placed to hold flow state, so that the state will usually only be lost if either end-point fails, in which case the flow will have failed anyway.[11]

Once we admit the need for flow state on a middlebox, we have to handle failure or startup of the middlebox during a flow, or the reroute of a flow in progress through the middlebox. Later we present two dropper algorithms:

1. the 'Continually Vigilant' algorithm primarily designed to start at the beginning of a flow (§7.6.1);

2. the 'Mid-Flow' algorithm (§A.1) designed specifically to pick up a flow in the middle and immediately start testing its behaviour.

We recommend the 'Continually vigilant' algorithm because it is more precise (better trade-off between false hits and misses) in normal operation and it will recognise packets from the middle of a flow as if they were a new flow, but only after a round trip with some degree of loss. It only recognises mid-flow packets as a new flow after it has received a Positive packet, which a flow should send at least occasionally in response to drops anyway.

The dropper could use the alternative mid-flow algorithm for flows in progress when it boots up. But it would be harder for the dropper to recognise a rerouted flow and know to use this algorithm instead of the one for a whole flow lifetime. Ideally, the dropper would have to know to expect the arrival of mid-flow packets by some out of band means—perhaps through participating in the routing system. We therefore recommend the 'Continually Vigilant' algorithm, because the alternative seems too complex.

Further, once we allow a need for network flow state, we have to admit that the network layer needs access to flow IDs. In general terms, as a minimum in order to communicate, two computing processes don't have to reveal their IDs to the network nodes between them. The Internet architecture recognises that flow IDs are not essential to the functioning of the network layer. It makes flow IDs a property of the transport layer, which can choose not to reveal them to the network layer[12]. Indeed, the facility in IPv6 that allows the transport layer to volunteer a flow label to the network has not so far been taken up.

Therefore, as we explained in §§7.1 & 7.3 the dropper only needs a flow ID to be unique; it doesn't need to identify reachable flow end-points. The best it can do is to use the most fine-grained aggregate identifier available. This means that a misbehaving flow or flows could besmirch the balance of other well-behaved flows sharing the aggregate (see §7.5.2 for details of this attack and countermeasures).

[11] One technique to avoid network flow state is to expect the end-points to regularly refresh soft flow state in the packets they send. A middlebox can even use an authenticated cookie-style mechanism [KM97] when it doesn't trust the end-points to store its state faithfully, by signing the latest state and expecting the end-points to return the signed state in future packets. But in the case of the re-ECN dropper, the network's flow state changes from packet to packet so none of these tricks will work.

[12] E.g. by aggregating many flow IDs together in some form of tunnel between two intermediate addresses between the processes and encrypting all mention of any IDs to be used for demultiplexing beyond those tunnel end-points

Dropper State Machine

Precisely, the only properties the egress dropper requires of a flow are:

- An arriving packet is considered to belong to (or match) a flow if all its identifiers match the identifiers of an active flow;[13]
- A new active flow starts when a packet carrying the Cautious (+?) or Positive (+1) marking arrives that doesn't match an existing active flow;[14]
- A flow MAY be considered to have ceased to exist if no matching packets arrive within a 'flow state minimum timeout' ([BJMS09a] fairly arbitrarily defines this timeout to be 1s—enough for about four round trips halfway round the world and back).

The above definitions allow an egress dropper (and any other elements) to manage flow state deterministically, clearing it after an idle period of 1s if necessary. If a transport re-uses its own (or someone else's) identifiers after inactivity greater than the timeout, it will require initialisation by a new packet marked positively and the egress dropper will not necessarily consider it as the same flow.

We expect ingress networks will hold sources accountable for packets marked Cautious or Positive by considering delivery of each byte in them as a cost. This will tend to limit the rate at which sources send packets that open new flows (see §12.1.3). Sending subsequent packets using the same flow ID should cost the source nothing other than further Positive packets required to balance any cost of congestion encountered. But this does place an ultimate limit on the amount of dropper state that a source can keep alive (see §12.1.3 for the worst-case scenario).

The design of a dropper is fairly hemmed in by all the constraints it faces. Nonetheless only the constraints would need standardising, not design details or implementation. We propose two example designs. We will focus here on the first design called 'Continually Vigilant' (§7.6), because it maintains state for each flow that has proved itself well-behaved, whereas the second, termed 'Mid-Flow'(§A.1) maintains less flow-state at the expense of precision.

The more stateful ('Continually Vigilant') dropper considers packets to belong to misbehaving flows if they use a flow ID that has not flagged itself as new. It treats all misbehaving flows as one 'Bulk' flow to avoid holding state for each of them and treats them all equally badly. It will only protect a new flow from the risk of the high drop rate suffered by the bulk of misbehaving flows if it starts with a Cautious (or Positive) packet and if its lifetime balance remains in credit. We categorise such flows as 'Compliant'.

There is a small chance (quantified in §7.7.1) that a Compliant flow will experience a burst of congestion marks within one window and that the dropper will wrongly consign packets to Bulk status before the source can send sufficient credit to make up the shortfall. Therefore, the egress dropper should continue to hold the state of a flow that has gone into debt for at least the flow state minimum timeout

[13] Additionally, if it is larger than any packet so far matching the flow, at least as many bytes of credit must have been posted to the flow as its size, otherwise it is not considered to match the flow. This negates flow ID whitewashing attacks (§7.5.1).

[14] To be clear, if a Cautious packet matches an active flow it doesn't start a new flow.

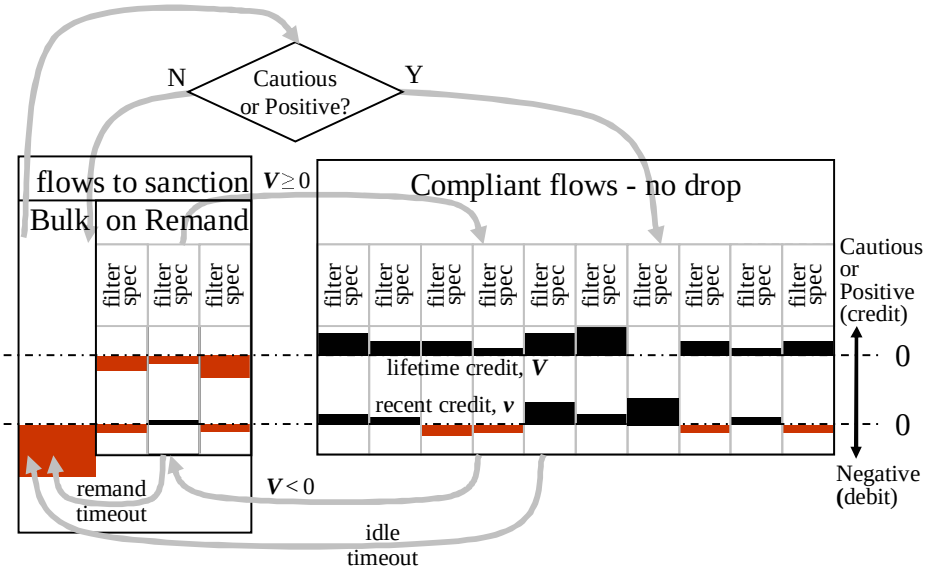


Figure 7.3: Re-ECN Dropper Flow State Machine.

(currently defined as 1s). We term this being on 'Remand'. If a flow is still on Remand after this timer expires, its flow state may be removed, leaving it to be treated with the Bulk of misbehaving flows. If sufficient Positive bytes arrive while still on Remand to make the lifetime balance $V \geq 0$, the flow returns to the Compliant status.

Even for Compliant flows, the dropper's flow state may time out if the flow has remained idle for no less than a similar duration of 1s. The path congestion that a transport algorithm learns from the network becomes stale during idle periods much longer than this anyway. Therefore a flow that is idle for at least this long MUST restart as if it were a new flow by conservatively setting Cautious in at least the first packet after the idle period. If the egress dropper has removed its state for this flow, this action will re-establish it. Whether the dropper's state has been removed or not, this action will be treated as a credit.[15]

Fig 7.3 summarises the above discussion by showing diagrammatically the state machine of a dropper design that keeps state for the lifetime of all Compliant flows. As each packet arrives, its flow ID is compared with the filter-specs already created for flows in progress. If it matches none, a new filter-spec need only be created if it is a Cautious (or Positive) packet. Otherwise it can be processed with the Bulk of unclassified packets. Having classified which flow it belongs to, the dropper then uses the packet's extended ECN field to update both the lifetime balance and the recent balance of that flow. The new value of the lifetime balance determines the larger categorisation of the flow as either Compliant, on Remand or Bulk. If the flow is then categorised as on Remand or Bulk, the packet can be sanctioned according to the recent balance held for that flow, using an algorithm such as that in §7.6.

[15]To be clear, this does not mean that a flow must send a heartbeat every second. Even if it is idle for much longer, it only needs to send one Cautious packet when it re-starts. Also note that the above restart is only with respect to the network layer; it is completely separate to any transport layer behaviour following an idle period. This may have been defined elsewhere for each transport layer protocol.

Dropper Flow-State Congestion

The re-ECN system design takes the following precautions to minimise the memory the dropper requires for flow state:

- Re-ECN flow state is only absolutely required at egress nodes at the very edge of the network where it will limit scalability less;
- The design variants we propose limit vulnerability to state exhaustion attacks by only holding state for flows that have proved they are well-behaved;
- We provide a complementary dropper design (§7.6.1) that requires per-flow state only for a sample of flows, but still has a high likelihood of catching misbehaving flows;

Nonetheless, unusually large numbers of active flows, whether malicious or innocent, might exhaust the dropper's memory capacity, however well-provisioned it is. In the re-ECN protocol, the state transition from Cautious to Negative has been reserved for any middlebox (not just the re-ECN dropper) to signal exhaustion of its flow state resources (see §10.1).

Below we consider how this signal would be used by the dropper. Although we recommend this protocol transition for signalling flow-state memory congestion, we have not taken it further than the initial design in §10.1 and we do not include it in the example algorithms.

The memory congestion function should sit logically after a dropper function on the same machine. So the dropper decision would not count any Negative marking due to its own memory congestion against a packet. If a Cautious packet arriving at the dropper did not match an existing flow it would conceptually create a credit for itself, then request long term memory to store its flow state. Approaching memory exhaustion would increase the probability of this request being rejected leading to the Cautious packet being marked Negative before forwarding.

However, if a Cautious packet arriving at the dropper did match an existing flow, it would add to the credit stored for the flow. As no request for extra long term memory would be required, it would be forwarded with its Cautious marking intact, even if memory was approaching exhaustion.

Even if the dropper had absolutely no more memory, it could still forward Cautious packets. In this case all Cautious packets for new flows would depart marked Negative, while those matching established flows would be forwarded with their Cautious marking intact.

7.4 Dropper Handling of Other Markings

This section handles a number of more detailed protocol issues.

7.4.1 Cancelled Markings

It will have been noted that the re-ECN dropper never drops Positive or Cancelled packets. It might seem reasonable to spare Positive packets, because they have unambiguously paid their way, but it seems that a source could cheat by initialising the majority of its packets to Cancelled in order to spare them all from sanction at the dropper.

The logic for the dropper not considering Cancelled packets for sanction relies on them having once been Positive and having suffered the same random congestion marking as Neutral packets. But a source that deliberately initialises packets as Cancelled breaks the basis of both these assumptions. Further, all its Cancelled packets will be immune from further congestion marking, so any congestion experienced will never need balancing with Positive markings.

§8.2.7 presents a simple solution to this apparently serious flaw, involving normalising the proportion of Cancelled packets to that expected from the proportions of other markings. This will deter a source from attempting this otherwise serious attack. Rather than explain the solution here, we defer all discussion until §8.2.7 because normalising Cancelled markings solves other problems between interconnected networks which will be described first in §8 on Border Incentives.

As well as justifying why the dropper never discards Cancelled packets, we also need to justify why the drop probability of Negative and Neutral markings in Eqn (7.4) takes no account of Cancelled bytes when it does take account of Positive. The reasoning is as follows. For equivalent behaviour to honesty, the source will send Positive bytes in response to feedback of Negative bytes u_N. If it gets feedback reporting that these Positive bytes themselves have been marked (producing a fraction of Cancelled bytes c_n) it will send further Positive bytes to top up the balance ($z_0 = u_N + c_n$). Therefore, at the egress, for stationary congestion Positive bytes should match Negative alone ($z_n = z_0 - c_n = u_N$). Diagrammatically, in terms of Figure 7.1b), this is equivalent to saying the Negative rectangle should be reduced to the same area as the Positive rectangle but the Cancelled square makes no difference.

7.4.2 Cautious Markings

There is a possibility that the first packet of a flow was originally marked Cautious by the source, but subsequently it has been over-written as Negative by a congested forwarding element (§9.2) or by a middlebox running short of memory (§10.1).[16]

It is then not possible for the dropper to know whether the flow started well-behaved or not. Such a packet will not cause creation of flow state in the dropper and will therefore have to suffer the prevailing drop rate for the bulk of misbehaving flows.

In §9.2 we recommend that congested links should re-mark Cautious packets to Negative rather than drop them. Why go to all this bother if the resulting Negative packet will be dropped anyway by the dropper further down the path? There are two reasons:

Not all Cautious packets are initial packets of flows. Probably the majority of Cautious packets will be later packets that top up flow credit. If they arrive at the dropper having been re-marked Negative, they will match pre-existing flow-state and reduce the balance as any Negative packet would. The receiver will feed the Negative marking back (not necessarily knowing it was originally Cautious). This follows the best practice of notifying congestion explicitly rather than by drop that we espoused in §6.1.1: to avoid the performance hit of timeouts, to disambiguate congestion from all the other possible reasons for drop and to make congestion visible to the network (§6.1.1);

[16] Or, a non-TCP source may start a flow with multiple packets and the first to be sent may not arrive first.

The dropper doesn't drop all initial Negative packets. Whether a dropper will drop packets in the bulk of misbehaving flows depends on whether misbehaving flows predominate. The dropper suggested in §7.6.1 folds all the left over credits at the end of behaving flows into the Bulk balance. When this results in a positive Bulk balance, initial Negative packets will get through. When it doesn't, they won't.

If a Cautious packet is the first of a flow; and if it is marked Negative; and if it does still get through the dropper, the eventual receiver will feed that fact back to the source.

As we explain in §10.1, the receiver's response to a Negative initial packet must be overloaded with the two meanings (exactly how depends on the transport):

- 'Flow state not stored' (transport layer congestion);
- and 'Echo Congestion Experienced' (network layer congestion).

Having received this feedback, the source must proceed more carefully with the subsequent flow of packets (e.g. backing off its initial window) *and* it must repeat its attempt to have flow state stored for its flow, whether on the server or on middleboxes along the way (§10.1 gives an example in the context of TCP SYN cookies).

If a source receives feedback implying a Cautious packet experienced congestion (whether by Negative marking or drop), the source SHOULD send the next packet as Cautious as well. In addition, the source SHOULD increase its count that determines how many bytes it must mark as Positive on future packets. Following this procedure ensures that eventually as many positive bytes are sent as negative bytes received as well as building up the intended credit at the dropper.[17]

The source must also check whether the receiver's response shows it supports re-ECN [BJMS09a], and if not, it must behave as if its original packet was dropped. But, because the congested link explicitly marked rather than dropped the original packet, the source doesn't have to wait in limbo, not knowing why a response hasn't been returned.

7.4.3 Legacy ECN Markings

Three codepoints of the extended ECN field fall outside the definition of the re-ECN protocol: `Not-ECT`, `ECT(0)` and `CU` (see Table 6.1 on p79).

Packets carrying the 'currently unused' or `CU` codepoint, SHOULD be treated exactly the same as Neutral packets, as recommended for forward compatibility in §12.2.2.

Packets that are not ECN-enabled (`Not-ECT` codepoint) or set to the legacy ECN codepoint (`ECT(0)`) should pass through the re-ECN dropper unscathed, even if they match a flow ID in the dropper. A network managing resource sharing using re-ECN is advised to rate-limit packets with these two codepoints at the ingress (and at some or all links if it chooses), therefore it is safe for the egress dropper to ignore them.

[17] If the Cautious packet experiencing congestion is the first of the flow, it is likely the dropper will not record the negativity of this packet against any flow ID. But the source cannot be sure of this, so it should act conservatively.

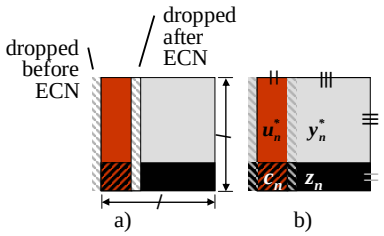


Figure 7.4: Compliant Traffic Suffering Losses Before and After ECN Marking.
Shown a) Without and b) With Repair of losses.

In addition to the above codepoints, legacy ECN flow will include some packets marked with the CE (congestion experienced) codepoint. The re-ECN protocol overloads the original congestion experienced (CE) codepoint as a Cancelled marking. Therefore Cancelled packets *that also do not match a flow ID in the dropper* should be allowed through the dropper unscathed, rather than being treated to the drop probability of the Bulk of misbehaving flows. This is safe, because re-ECN ingress and border policing already corrects for excess Cancelled packets.

It is possible that a legacy ECN flow might use the ECT(1) codepoint if the ECN nonce [SWE03] is ever implemented and deployed, but re-ECN overloads this codepoint to mean a Positive marking. The ECN nonce RFC has experimental status within the IETF standardisation process, but it has not been implemented to anyone's knowledge (except on the machine of its author). If it were implemented and deployed, it would make it look as if about 50% of the packets of a flow were Positive. This would not be a problem at the re-ECN dropper. But such flows would have to be recognised and separated out at an ingress policer and at borders, otherwise they would rapidly reduce the congestion allowance of the party sending or forwarding them.

7.4.4 Congestive Loss

So far we have only considered congestion that leads to explicit congestion notification, not congestion that leads to loss (implicit notification). Unless the network's congestion notification is explicit, re-ECN policers and droppers cannot in turn check whether the source is correctly reinserting these congestion notifications into the network. Therefore a network won't invest in re-ECN droppers and policers without also making sure ECN is turned on in its forwarding equipment.

However, some congestive loss will always be present. Older equipment may not support ECN, or may take time to be upgraded, and ECN equipment will occasionally overflow its buffers, just by happenstance.

Positive markings are introduced by the source to balance congestion marking introduced further down the path. Loss will have a different effect depending on whether it occurs before or after Negative ECN markings are introduced (Fig 7.4a):

- If loss occurs after Negative markings are introduced, on average all Positive and Negative markings will be reduced proportionately;[18]

[18] Assuming the network operator is not actively biasing its loss—see §12.1.2.

- If loss occurs before Negative markings are introduced, there will be proportionately less loss of Negative markings than Positive.

A transport can detect losses, but it cannot easily find out where on the path they are being introduced. If the transport detects loss of a packet that it originally marked Positive, its safest strategy[19] is to introduce a replacement Positive marking, as in Fig 7.4b). It will be seen that this leads to slight overstatement of Positive markings if any loss occurs before ECN markings are introduced, which is safe.

A loss may not be due to congestion, for instance it may be a deliberate sanction imposed by a re-ECN policer or dropper. However, we have made sure that the dropper doesn't drop Positive or Cautious packets, and the policer only drops Positive or Cautious packets as a last resort.[20]

7.4.5 Downstream Congestion Analysis Revisited

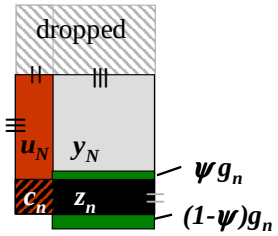


Figure 7.5: Effect of Cautious Markings on the other Re-ECN Markings after the Egress Dropper.

So far, although we have considered the effect of congestive losses informally (§7.4.4), we have not included loss in our formal analysis. Also, in §6.2, where we defined metrics for recent downstream congestion in terms of re-ECN markings, we ignored Cautious markings. But we promised to revisit our analysis to include them. The additional analysis is below, but it ends up deciding to continue to ignore these aspects as they make no difference. This section may therefore be skipped on a first pass through the document.

The bytes in Cautious packets have nominally the same worth as bytes in Positive packets. But unlike Positive packets, which become Cancelled if they experience congestion, Cautious packets can be marked Negative.[21]

Congestion marking Cautious packets to Negative is NOT a REQUIRED behaviour of forwarding elements—care has been taken to ensure no changes are required on forwarding elements between ECN and re-ECN. Therefore marking of Cautious packets introduces an additional complication because it

[19] The specification of re-ECN for TCP [BJMS09a] adopts the more generous strategy of introducing Positive bytes to balance any bytes experiencing congestion, whether signalled by ECN marking or loss. However, Positive markings to balance losses would be purely voluntary, as the re-ECN mechanisms cannot detect non-compliance.

[20] §11.3.1 introduces a way for a policer to covertly signals that it is dropping Positive or Cautious packets, otherwise a transport that accidentally exhausts its congestion allowance in the ingress policer can become trapped in an avalanche of increasing sanctions.

[21] We have already briefly mentioned this aspect of the re-ECN protocol in which forwarding elements and middleboxes can optionally be upgraded to mark Cautious packets to Negative when congested. The details of each are introduced in §9.2 (on forwarding element bandwidth congestion) and §10.1 (on middlebox memory congestion).

is separately optional from ECN marking. Before re-ECN, a box either did ECN marking or it didn't. But we will now have to take into account that some network equipment will congestion mark all five of the re-ECN codepoints while older equipment (that has been made ECN-aware but not re-ECN-aware) will congestion mark only four and drop those marked Cautious when it is congested. We will denote the proportion of traffic experiencing congestion marking of all five re-ECN codepoints as ψ and the proportion of Cautious marking arriving at the ith resource is denoted by g_i.

$$\begin{array}{lccc} \text{Neutral:} & y_0 = 1 - z_0 & y_i = (1-m_u)(1-z_0-\psi g_0) & y_n = (1-m_u)(1-m_d)(1-z_0-\psi g_0) \\ \text{Cautious:} & g_0 & g_i = (1-m_u)g_0 & g_n = (1-m_u)(1-m_d)g_0. \end{array} \tag{7.6}$$

Fractions of each re-ECN Marking where including Cautious alters Eqn (6.1)

Eqn (7.6) tabulates the proportions of Neutral and Cautious markings at the start and end of a path and at an intermediate point i. When we take Cautious markings into account, the proportions of the other markings remain unchanged from Eqn (6.1).

Cautious bytes are intended to allow the source to create credit at the egress dropper, given we have chosen to make the source responsible for any allowance for round trip delays. If the source uses Cautious markings strictly in addition to Positive markings, then none of the analysis in §6.2 changes. The source still introduces a Positive byte in response to feedback of every Negative or Cancelled byte ($z_0 = u_n + c_n$) as in Eqn (6.2). This leads to the same approximate formula ($v_i \approx z_i - u_i$) for downstream congestion in terms of the other markings as in Eqn (6.3) and the same precise formulae as Eqns (6.4) & (6.5). This is because Cautious credit requires a one-off contribution, while these formulae represent downstream congestion under stationary conditions.

The only difference will arise if the source understates Positive marks because it thinks the Cautious marks it had to send to open a flow will be sufficient to prevent its flow going into debt, without having to send a Positive byte for every Negative byte. However, our steady state equations can ignore these initialisation effects. Nonetheless, we will take the markings of single packets into account when we consider attacks (§7.5 below).

7.5 Attacks Perverting the Dropper

This section only includes attempts to attack the re-ECN egress dropper or to pervert its intended effect. Combined attacks playing off the dropper against other components of the re-ECN system are described in §12.1, "System Attacks on Congestion Signal Integrity."

7.5.1 Flow ID Whitewashing

§7.3 explained that the egress dropper defines a flow solely by the uniqueness of its flow identifiers while it is active, and it will not recognise a new unique flow without some investment of resources by the source.

This raises the question of how much initial investment is sufficient to ensure that flow ID whitewashing attacks do not pay off. We outline the attack here, and then analyse it later, in §7.7.2.

An attacker can whitewash a flow ID simply by abandoning the flow ID at the first feedback of a Negative mark and continue transferring data with a new 'whitewashed' flow ID.[22] Therefore the initial investment must be at least as great as the worth of one Negative packet.

However, an attacker could routinely start a whitewashed flow by investing credit in one tiny Cautious packet, followed by much larger subsequent Neutral packets. Then, whenever one of these packets was congestion marked, it would always have cost the network a lot more than the original credit invested, giving a considerable pay-off to the attacker over time.

This attack would be thwarted if the dropper records the largest packet so far seen in the flow and a larger packet was only deemed to match the flow ID if the credit against the flow (in bytes) was at least as large as the new largest packet.

This raises the question whether requiring just one maximum sized packet's investment is a sufficient incentive for a source to maintain the same flow ID beyond each Negative packet. We now compare the gain from the whitewashing strategy with that from behaving as re-ECN expects.

Whitewashing Strategy: We imagine a transport creates a whitewashed flow ID every time it receives feedback of a Negative packet, but it starts each whitewashed flow with a full-sized Cautious packet. It sustains its flow rate as if there had been no change of flow ID by carrying over all the congestion control state between flow IDs. Then the source will pay as much in Cautious credits as it would have to match each Negative packet with a Positive. At the dropper, each time a Negative packet arrives, the flow's balance will drop to exactly zero (which will not create any problem) while the last window of packets in flight plays out. Straight after the last packet in flight, a packet with the new flow ID would arrive at the dropper carrying with it a new Cautious credit. Thus far, the benign whitewashing behaviour would not be sanctioned any more than a normal flow, and it would cost no more to maintain (except the extra flow state for the transport and for the dropper).

However, the dropper makes this whitewashing behaviour lose out whenever there happens to be more than one congestion event per window (see §7.7.1). If two or more congestion events occurred in one window, the whitewasher's dropper balance would fall below zero and the remainder of its final window would suffer loss. The whitewasher would however avoid paying for all but one of the Negative packets when it started the new whitewashed flow.

Compliant Strategy: If, on the other hand, the source had not been continually switching flow IDs, the dropper could be much less harsh on the occasional temporary fall into debt. The Continually Vigilant dropper pseudocode in §7.6.1 allows the source to post a one-off per flow credit that protects it against occasional multiple congestion events[23].

But the Continually Vigilant dropper is designed so that a dive straight into debt early on in a flow leads to a much higher drop probability than if there had been a long record of good behaviour before the drop into debt, assuming the same initial credit in both cases. For instance, a balance of -1 caused by a Positive then two Negatives will lead to $\sim 50\%$ drop, while the same balance of -1 but preceded by 9 alternating Positive then Negatives would only lead to $\sim 10\%$ drop. This effect is illustrated in Fig 7.10

[22] A colleague, Toby Moncaster proposed this attack.

[23] And it could even recover most of the credit near the end of the flow

on p131 later.

7.5.2 Dragging Down an Aggregate

A misbehaving flow or flows could besmirch the balance of other well-behaved flows sharing an aggregate, assuming each of the flows's separate identifiers had been hidden in some form of obfuscating tunnel.[24]

This becomes problematic if impairment sanctions (e.g. drop) are used in the middle of a network, e.g. at trust boundaries (borders), where aggregates might consist of flows belonging to entities with few interests in common. However, we recommend financial sanctions in the middle of a network with drop sanctions applied closer to the edges where fine-grained flows are more likely to be visible (§8.1). In the case of a tunnel, this would mean that the tunnel decapsulator would be able to operate a dropper to isolate and sanction the offending flow(s). If it detected misbehaving flows it could also ask the encapsulator not to allow them to hide within the aggregate.

7.5.3 Dragging Down a Spoofed Flow ID

This attack uses the dropper to amplify a DoS attack on someone else. An attacker can understate congestion (e.g. not send any positive packets at all), while spoofing another flow ID. Because the attack packets match the destination IP address, they will pass through the same egress dropper with reasonably high probability whatever location they actually come from. If the dropper cannot distinguish attack from compliant traffic it will sanction both indistinguishable flows based on the proportion of Negative to Positive bytes across both taken together.

The attacker may blindly spoof other flows by a brute force search of the most highly utilised parts of the flow ID space, or it may target an attack on a flow ID known by some other nefarious means (e.g. a secondary man-in-the-middle attack).

However, it is harder for a blind attacker to launch the attack, as single attack packets have limited effect. A train of packets all with the same flow ID and all matching an active flow will cause a lot of harm, but the attack loses any effect soon after the last packet of a train. If the attacker cannot sense when the attack is having any effect (which it cannot because its spoofed source address gets no responses), it must send long trains of negative packets to the same flow ID in the hope some of them will hurt something active at the same time. If the attacker is located as a man-in-the-middle, it *can* sense if it is having an effect, but then it can just drop the traffic anyway.

Nonetheless, if an attacker knows approximately when victim flows might be active and has at least some clue of the destination identifiers, a targeted attack would be relatively trivial. The attack is much easier than hijacking a transport connection, because it only requires spoofing of the flow ID, it does not also need to spoof transport (or application) layer sequence numbers within a valid receive window. Network ingress filtering is not a universal defence, because an attacker merely has to attach to the Internet or take over a zombie where filtering is not deployed.

For instance, if there are s bits of entropy remaining in the flow ID, the probability of the event X that an attack train of n packets hits a continuously active flow would be $P(X) = 1 - (1 - 2^{-s})^{N/n}$ if

[24] A colleague, Jake Hill proposed this attack.

a total of N packets were sent. If the protocol ID, both IP addresses and the destination port number are known (or a part of the ID space is highly utilised and the attacker doesn't care which ID it hits), this would leave $s \approx 15$b of entropy in the source port if it is randomised [LG09]. Therefore only $\sim 30,000$ trains of packets with different flow IDs each containing n packets would have a 60% chance of dragging down one flow.

Beyond port randomisation, defences against this attack are limited, at least for IPv4. We do not recommend the dropper trying to communicate with the apparent source of a flow to signal that a network policing element is the cause of its high drop rate (which might otherwise help the application understand how to solve the problem). The dropper is designed on the principle of "ID uniqueness not reachability", precisely because a middlebox cannot always be expected to extract the source end-point identifier from a packet. Two possible alternative defences come to mind:

Destination Port Hop: During this attack, the destination would experience a very high drop fraction, a very high fraction of out of window packets (for TCP—or the equivalent for other transports) and potentially large numbers of packet verification errors if authentication were being used. Under these circumstances, if the destination application reset the connection and started again with different port numbers at least it would immediately find a working connection and avoid the attack continuing.

IPv6 Flow Label Nonce: For IPv6, we recommend (§7.1) that the dropper SHOULD use flow labels seen on the initial packet as part of the flow ID. A spoofing attack not using the same flow label would simply not match the flow ID. This would afford 20 extra bits of entropy. This approach is similar to Blake's proposal to overload the IPv6 flow label as a transport layer nonce [Bla08] (but we use it for what is effectively a 'flow-ID sub-layer' rather than the 'transport layer' as such).

7.6 Dropper Algorithm Implementations

We will present two possible egress dropper algorithms to give sources the incentive to comply with the re-ECN protocol:

Continually vigilant: An algorithm that requires flow state for every flow that has proved itself well-behaved from the start;

Mid-flow: An algorithm that can start mid-flow, therefore one can choose how much flow state it requires by applying it to flows picked randomly to test for compliance. But as a consequence this algorithm may allow more false hits and more false misses. That is, it could wrongly sanction some behaving flows and it could insufficiently sanction some misbehaving flows.

The latter, mid-flow algorithm can pick up a flow part-way through, therefore it doesn't know how much credit the flow has allowed itself (or needs) to stay in credit over a round trip. Therefore it cannot discourage a flow from 'whitewashing' its identity if it goes negative, whereas the former algorithm can. The latter mid-flow algorithm also has to give the source a reasonable allowance for round trip delay,

contrary to our design principle of 'Source responsibility for delay allowance' §7.3 discusses both these points in depth.

Therefore the mid-flow algorithm should not be used as an alternative to the continually vigilant one, only as a complement. For instance, the latter might be used for random flow sampling at interconnect borders, while the former could continually guard the egress of the internetwork, later on the path. Or the mid-flow algorithm might be used as an expedient when an egress dropper first boots up—to test the balance of flows already in progress. As the mid-flow algorithm introduces no new concepts over the former 'Continually Vigilant' algorithm, it is relegated to Appendix A.1.

7.6.1 Continually Vigilant Dropper Algorithm

This algorithm requires state per well-behaved flow but is expected to cause few false hits and few false misses. Its description will be built up in three stages, from low to high level:

- the algorithm for applying sanctions;
- the algorithm for maintaining the moving averages of the marking fractions;
- the management of flow state.

Algorithm for Applying Sanctions

This algorithm can be applied both to packets matching recognised flows, and to those in the bulk of misbehaving flows treated as if they all share a wild flow ID.

By Eqn (7.4) the drop probability $\pi_y = \pi_u = 1 - z/u$, where z is the recent balance of Positive markings and u the recent balance of Negative markings. But drop is only applied if both the recent balance and the lifetime balance are in debt: $(v \approx z - u < 0)$ and $(V < 0)$. The `probDrop()` algorithm below subjects each packet to the appropriate drop probability, taking the relevant `z` and `u` for the flow as inputs.

The algorithm that maintains the moving averages of `z` and `u` stores them within a flow-state structure, `fState` from one packet to the next of each flow.[25] For convenience we pass the whole structure `fState` into and out of the function that applies sanctions, even though it only uses three of its variables and only updates one. In full, the `fState` structure holds the following flow state variables:

`fID`: flow ID

`z`: recent +ve bytes per -ve mark

`u`: recent -ve bytes per -ve mark

`r`: remainder carried to next drop

`V`: downstream congestion over flow life

`lastGoodTime`: time `V` was last +ve

[25] The pseudocode variables `z` and `u` are actually scaled transformations of the z & u given in the earlier formulae (see later).

`smax`: max packet size over flow life

All the pseudocode that follows takes liberties for brevity by assuming `fState` as the context of any of these variables, unless otherwise stated.

```
/* Drop packet with probability (1-z/u)
 */
 probDrop(packet, fState) {
    if (u > z) {
        if (r == -1) {
            r = rand[0,u)
            /* Note: (0 <= r < u)
             when initialised */
        }
        r += z
        if (r > u) {
            r -= u
        } else {
            drop(packet)
        }
    }
    return(fState)
 }
```

In order to minimise per mark processor cycles, this deterministic algorithm avoids the apparent need for a division or multiplication in the drop probability formula. It uses only adds, compares and subtracts. The floating point variable `r` is the algorithm's remainder that the dropper stores to carry it over until the next packet of the same flow arrives. It also avoids running any pseudo-random functions except once during each flow, to initialise `r`. When flow state is first created, `r` is initialised to -1 as a flag to trigger its randomised initialisation the first time the function is called within a flow (see 'Flow-State Variable Initialisation' below).

We haven't thought of a way, but it might be possible for an attacker to exploit the predictability of this deterministic algorithm relative to the congestion markings it detects, in which case a randomised alternative would have to be used. The other potential danger of deterministic drop is synchronisation, but that is unlikely to be a problem for a sanction that should be only seldom applied, and which is driven from markings that are themselves spaced by a random process.

Algorithm for Maintaining the Moving Averages of the Marking Fractions

In order to ensure the current drop rate only reflects recent behaviour, it is necessary to discount events that happened further in the past using a moving average. We use an exponentially weighted moving average (EWMA) because it discounts past events evenly and it is very easy to implement.

By the design principle 'Source responsibility for delay allowance' (§7.3), the sender not the dropper is responsible for ensuring there is enough credit to survive the delay after each debit, including bursts of multiple debits within one round trip time. Therefore probabilistic drop sanctions should start to be applied as soon as the moving average goes negative (but only if the lifetime balance is also in debt). If the dropper did allow for round trip delay, it would have to track the evolution of round trip time (RTT) of each flow, and introduce a delay buffer in the Negative event stream, both of which are hard to

implement, let alone efficiently (see §A.1 for a constant conservative approximation of this strategy).

From Equations (6.9), (7.1) & (7.2) it can be seen that z & u are both measured with respect to the same metric; the moving average of packet size. But the metric with respect to which they are measured becomes irrelevant for both the purposes they are put to; when z & u are divided their identical denominators cancel out; and when their difference is compared to zero their identical denominators don't matter.

In all our algorithms we exploit the fact that z & u can be measured with respect to anything, as long as the same anything is used for both. In fact we will measure positive and negative bytes with respect to negative marks (justified below). Perhaps confusingly, we use the variables z & u in all the pseudocode, even though they are measured with respect to a denominator that is different to that of the z & u in our earlier formulae for proportionate sanctions.

The discussion in §6.2 noted that the discounting factor of the moving average to measure recent downstream congestion "must be sufficiently small to give consecutive positive and negative marks a reasonably similar weight". Given congestion and hence inter-mark spacing can vary greatly, it seems impossible to choose a discounting factor that will cater for all levels of congestion. But the trick is to use the marks themselves to clock the moving average. Thus instead of measuring downstream congestion-volume with respect to volume, with respect to the packet count, or with respect to time[26], it can be measured with respect to congestion marks themselves.

This works because the absolute level of recent downstream congestion isn't needed. It is only necessary to compare Positive with Negative, either to establish whether the balance is in debt or to know their relative proportions to determine drop probability (Eqn (6.9)). As long as each stage of the moving average discounts Positive and Negative marks the same relative to each other, they don't have to be discounted consistently from stage to stage. Instead, all that is necessary is to ensure that whenever one EWMA is updated the other is as well. Then, the EWMAs can be self-clocked by the arrival of Positive or Negative packet marks.

We decided to clock the moving average solely with respect to Negative marks. It was safer to clock the moving average on marks that are meant to be generated by the network, not the source. If we had also clocked on Positive marks, the source could have gamed the dropper by sending numerous small positively marked packets in place of bigger Positive packets. Whereas the source cannot determine the sizes of Negatively marked packets unless it sends them itself which would cost it more in Positive packets to balance them.

The principle of 'Source responsibility for delay allowance' (§7.3) implies that the source sends

[26]This is fortunate, because otherwise an event-based EWMA would be required, which is complex to implement. Often an event-based EWMA can be avoided if the arrivals of the events are Poisson [Wol82], but in the case of honest re-feedback of congestion marking, the arrivals are not Poisson. This is because the honest sender's re-ECN algorithm tends to make the arrival process of Positive marks a time-shifted version of the arrival process for Negative marks. Another reason one might think arrivals are not Poisson is that the RED algorithm used for Negative marking includes a stage to regularise the inter-mark spacing [FJ93, §4]. However, we show in §7.7 that the RED algorithm only spaces marks regularly across the aggregate of flows served by a link. But within each flow, this apparent marking regularity unravels to become i.i.d. (unless the flow is large relative to the total capacity of the congested link).

each Positive mark to counteract the *next* Negative mark, not the previous one, even though the source mechanistically sends a Positive mark in response to the previous Negative one. Therefore, the algorithm should collect up any Positive marks seen since the previous Negative mark and discount both Positive and Negative marks at the same time, when the next Negative mark appears. Even if a Negative mark never appears again (perhaps because congestion suddenly stops), any Positive markings arriving still immediately improve the flow's balance—it is only the discounting stage of both EWMAs that clocks on Negative marks.

We have to strike a balance between quickly punishing a misbehaving source and falsely punishing an innocent source that just happens to have suffered an unforeseen burst of losses. We do this by deciding whether to drop a packet *before* we use the packet's worth to update the moving averages. Then, whenever a negative packet arrives, the drop probability it experiences for itself doesn't take into account its own negativity. But its negativity is used to calculate the drop probability of subsequent packets in the flow.

In this way, the negative packet that first puts an honest source into debt will not be dropped, so it will reach the receiver and honest feedback will allow the honest source to correct the balance. Similarly, in the unlikely event of any further negative packets taking an honest source by surprise, each will be dropped with less probability than the following packets will suffer, increasing the chance that the integrity of the feedback loop holds good for an honest source.

But this leniency only lasts for one packet. So, if a misbehaving source has no intention of making up the balance sufficiently, the relative growth rates of the two recent balances will immediately increase the drop rate starting with the next packet. This doesn't quite "sanction dishonest flows, preferably at the first dishonest packet" as our "Minimal False Misses" constraint requires, but it only gives one packet's grace. This is safe, because Negative markings are under the control of the network, not the end-points, and we have ensured end-points cannot gain by sending Negative packets (§8.2.5).

We should also make sure we don't unnecessarily drop Cancelled packets, which also carry congestion notifications to the receiver. But a Cancelled packet makes no difference to the relative values of `z` and `u` so it doesn't need any extra leniency.

By delaying recalculation of the drop probability until after deciding whether to drop a packet, it seems that a positive or cautiously positive packet cannot use its own worth to save itself from being dropped. However, as in Eqn (7.4), the algorithm ensures that packets with Positive or Cautious markings are immune from drop anyway.

The resulting EWMA algorithm fits around the above drop probability algorithm as shown below. Note that `a` is the discounting factor of the EWMA. The function `readEECN(packet)` reads the extended ECN field of the `packet`. The valid states of the EECN field are:

`POSV`: Positive

`CAUT`: Cautious

`NEGV`: Negative

CANC: Cancelled (or Legacy CE if no re-ECN flow state)

NEUT: Neutral

CU: Treated as Neutral

Not-ECT: Legacy

ECT(0): Legacy

```
/* Maintain flow congestion balances
 in the fState flow state structure.
 The parameter s is the packet size.
 */
 newBal(s, fState) {
    eecn = readEECN(packet)
    if (eecn == POSV || CAUT) {
        V += s
        z += a*s
    } elseif (eecn == CANC) {
        z -= a*z
        u -= a*u
    } else {
        /* NEGV, NEUT or CU */
        if (V < 0) {
            fState = probDrop(packet,fState)
        }
        if (eecn == NEGV) {
            V -= s
            z -= a*z
            u += a*s
            u -= a*u
        }
    }
    return(fState)
}
```

The implementation of the EWMA needs a little explanation as it is unorthodox and the pseudocode is rather terse. An EWMA can be implemented[27] by the recursive formula

$$\bar{x}_{i+1} \leftarrow as + (1-a)\bar{x}_i; \qquad\qquad 0 < a < 1.$$

We are only concerned about the relative values of two EWMAs, so we can multiply $\bar{x}$ by a constant to transform it to $\bar{x}' = (1-a)\bar{x}$. Then

$$\begin{aligned} \bar{x}'_{i+1} &= (1-a)\bar{x}_{i+1} \\ &= (1-a)(as + (1-a)\bar{x}_i). \end{aligned}$$

Therefore, an alternative EWMA implementation is

$$\bar{x}'_{i+1} \leftarrow (1-a)(as + \bar{x}'_i).$$

[27] This algorithm understates a true EWMA by $\epsilon = 1 + \frac{a}{\ln(1-a)}$. But a constant understatement is irrelevant for our purposes, because we merely want to compare two EWMAs, either to find the ratio between them or to find whether one is greater than the other.

This EWMA is fastest to implement in the two half steps used in the `newBal()` pseudocode, and shown below,

```
x += a*s
x -= a*x
```

We use this transformed algorithm because, between two Negative marks, it allows us to collect any number of increments to the EWMA of Positive bytes. Then we can discount the result just once during the same step as we increment then discount the EWMA of Negative bytes, as justified earlier.

The processing cycles required for each multiplication can be minimised if a is chosen so that $a = 2^{-b}$ where b is an integer. Then we can use a right bit-shift `a*x = x>>b`.

Management of Flow State

The algorithm below maintains all the flow state. As each packet arrives it checks whether it already holds matching flow state. If so, it uses the `newBal()` algorithm above to update the balances of lifetime and recent downstream congestion-volume in the flow state. If the resulting lifetime balance is good (non-negative), the dropper stamps the flow's state with the current time. If the flow's lifetime balance subsequently goes bad (negative), this time-stamp can then be used to determine the duration since the flow was last not in debt, so it can be timed out if memory becomes in short supply (see later).

In order to protect against flows giving themselves a new ID whenever their balance goes negative ('identifier whitewashing': see §7.5.1), the egress dropper requires all new flows to lodge a positive deposit of at least as many bytes as the largest packet seen in the flow. Therefore the dropper has to record the largest packet seen in a flow. If the dropper detects a larger packet than the maximum so far and the flow's balance currently doesn't cover its size, the packet is not deemed to match the flow ID and treated as Bulk. The flow's state is not discarded, as future packets can each be judged on their merits—the offending packet may simply have arrived before a packet that carried the necessary credit but was delayed in the network.

Of course, a packet may not match any existing flow state. In such a case, the dropper only allocates flow state if a Cautious (or Positive) packet initialises the flow. Otherwise the packet experiences the prevailing (perhaps high) drop rate of the bulk of misbehaving flows.

```
/* maintainFlowState()
 Maintain flow state in fState structure
 */
 foreach packet {
    s = readLength(packet)
    eecn = read EECN(packet)
    flowID = readFlowID(packet)
    fState = matchFlowID(flowID)
    if (fState != NULL) {
        /* Existing re-ECN flow */
        if (s > smax) {
            /* Bigger max packet size*/
            if (V >= s) {
                /* Compliant */
                smax = s
```

```
            } else {
                /* Non-compliant packet
                 treat as BULK */
                fState = BULK
            }
        }
        fState = newBal(s,fState)
        if (V >= 0) {
            /* Compliant status flow */
            lastGoodTime = timeNow()
        }   /* else Remand status
            so lastGoodTime unchanged*/
    } elseif (eecn == CAUT || POSV) {
        /* New Compliant flow */
        allocate(fState)
        fID = flowID
        smax = s
        V = 0
        u = 0
        z = 0
        r = -1
        fState = newBal(s,fState)
        lastGoodTime = timeNow()
    } elseif (eecn == Not-ECT || ECT(0)
                              || CANC) {
        /* LEGACY: forward unimpeded */
    } else {
        /* New or old misbehaving flow
         set status to BULK */
        fState = BULK
        /* update balances of BULK
         and probabilistically drop */
        fState = newBal(s,fState)
    }
}
```

Initialisation of Flow State Variables

Initialisation of the variables `u`, `z` to zero is deliberate not arbitrary. Over a long-lived well-behaved flow, `u` and `z` will tend to `(1-a)`, so we could have initialised them at this value. But we initialised them at zero to counter the strategy where a flow sends an initial credit (which keeps it out of the very high bulk drop rate), but never sends another positive packet. The countermeasure of initialising `u` and `z` to zero should cause the drop probability to jump to the correct level as quickly as possible (as motivated in §7.5.1 and analysed in §7.7.2).

On the other hand, if a flow is well-behaved for a while, both `u` and `z` will grow together in the correct proportions relative to each other. Then any slight understatement of Positive markings will lead to a slow rise to the appropriate drop probability if the understatement persists, but if the understatement is quickly corrected, very little drop will have ensued.

`r` is initialised to -1 as a flag to indicate that it needs initialisation (during normal operation of the `probDrop()` function $r \geq 0$). The `probDrop()` function is designed to avoid an expensive random number generation every time it is called. Instead the algorithm is deterministic, but randomised on first

use within a flow. However, `r` should be initialised in the range $0 \leq r < u$. But because `u` grows from zero as the flow progresses, `r` cannot be randomly initialised until it is first used. Hence the need to flag that `r` needs randomised initialisation before first use.

Flow State Time Out

The re-ECN dropper deliberately makes no attempt to detect the last packet of a flow explicitly (e.g. by looking in to the transport layer headers for transport-specific indications like a FIN in TCP). This prevents the dropper getting confused due to lost final packets, or due to misbehaving transports either deliberately suppressing final packets or spoofing the final packets of other flows. Instead it deallocates all flow state by timeout.

The re-ECN protocol specification [BJMS09a] sets the rule that a flow must send a credit packet (Cautious) as if it was starting afresh whenever it has been idle for more than a specified time (currently defined as 1 sec). This allows any middlebox such as the dropper to time out flow state that has been stale for more than this timeout. A middlebox is free to hold flow state for longer, but a transport should not rely on this behaviour.

We have not written pseudocode to describe garbage collection (flow state time out), but the general idea is as follows. The flow state structure includes a variable `lastGoodTime` which is updated to hold the current time whenever the lifetime balance of a flow `V` is still positive after having updated the moving averages of the flow's balances.

Two buffers (linked lists) of pointers to flow state structures are maintained. One for flows categorised Compliant ($V \geq 0$) and another for flows categorised on Remand $V < 0$. Every time `lastGoodTime` is updated, a pointer to the flow state it belongs to is brought to the front of the relevant linked list. Every so often (or when memory usage crosses a threshold), a garbage collection process deallocates flow state, working along the linked lists from the back. Flow state can be deallocated if `lastGoodTime` shows the flow has been idle or on Remand for longer than the timeout. Complaint but idle flow state SHOULD be kept in preference to flows on Remand.

Before each flow's state is deallocated, its variables should be merged in to the Bulk flow state. This aspect of the algorithm has not been written in detail, but a reasonable strategy would be to treat the lifetime balance of the flow as the worth of a single newly arriving packet in the Bulk. This deliberately ignores the recent balance variables, because their size relative to other flows will be meaningless given the EWMA is clocked on negative marks. Further work is needed to establish whether this is a reasonable strategy. It could make misbehaviour pay off because it doesn't discount the past (for positive or negative balances).

The re-ECN protocol has been designed so any middlebox can manage approaching flow state exhaustion by marking or eventually dropping Cautious packets that would consume more memory (see §7.3.3 on Dropper Memory Congestion).

7.7 Predicted Dropper Performance

7.7.1 Predicted False Hits

Even if congestion is stationary, congestion marking is a random process, so unusually high numbers of marks will appear during some round trip delays. The design of the re-ECN egress dropper adopts the principle that the sender is responsible for allowing sufficient credit for any marking that may arise during the delay while feedback returns to the sender (§7.3.2). But, whatever level of credit an honest source posts, it will never be immune to suffering some false hits from the dropper.

Conversely, the principle of 'source responsibility for delay allowance' gives the system designer a blank cheque to meet the requirement of 'minimal false negatives'; any false negatives can always be blamed on the source posting too little credit. As we said in §7.3.2, passing off this dilemma onto the source is a deliberate design choice that has a microeconomic interpretation—sources choose how much risk they are willing to carry to get their traffic through the unknown conditions of the next round-trip.

Below we model the probability that a flow will receive more then m marks in one round trip time (RTT) in various circumstances. This analysis can then be used:

- For a transport to predict the credit level it will require to keep the incidence of false hits below a threshold;
- To predict the probability of false hits for a certain credit level, so the system designer can evaluate whether it is 'reasonable' to transfer the risk of traffic uncertainty to the end-points.

The ultimate test of what is 'reasonable' is whether end-points are willing to take on this risk by adopting the re-ECN system. If they are not, then some alternative system design, perhaps with knowledge of round-trips in the network, will have to be considered. This is essentially a question of whether the end-to-end design principle is 'reasonable'. All that can be achieved here is to quantify the effect of this design decision.

Marks per Window. As marking probability rises, a congestion responsive transport will reduce its window of packets in flight. Therefore, the marks per round trip should not rise without bound. We will now derive expressions for the marks per RTT, then apply this expression to different transports; to TCP in congestion avoidance and to a generalised weighted α-fair transport.

Assumption 7.1. *The marking fraction, p varies on timescales longer than an RTT, being stationary during round trip time R.*

This assumption is examined more closely on p118 at the end of this sub-section.

We denote the discrete random variable representing the number of marks per round trip by M. We want to find the probability $P(M>m)$ that this credit will be exhausted in one RTT if the sender invests credit m.

Independent & Identically Distributed? Before we can continue modelling the probability of a certain number of marks per RTT, we need to establish whether the probability of marking a packet will be correlated with previous markings. In order to do this, we need to justify that the following assumption is reasonable:

Assumption 7.2. *ECN marking of each packet in a flow on the production Internet will be independent and identically distributed (iid) wrt. the marking of other packets in the flow, as long as the flow rate is small relative to the capacities of links it traverses.*

Justification: For re-ECN we are *only* concerned with ECN marking, which, unlike loss, can only be generated by an active queue management algorithm, such as random early detection (RED [FJ93]). The primary aim of AQM is to prevent global synchronisation, which causes sequences of losses to hit specific flows.

Unfortunately, we have found few studies that measure whether deployment of AQM has been successful in this aim. There are certainly no measurement studies of temporal correlation of marking in production networks, because few Internet packets or queues are ECN-capable. We can find only two Internet measurement studies of temporal loss correlation where the year of data collection was after RED *could* have been deployed—after the IRTF published its strong recommendation to use RED [BCC+98] in 1998. Zhang *et al* [ZD01] (1999–2001 data) found nothing to contradict earlier studies [Pax99, YMKT99] showing losses were not iid but arrived in strongly correlated bursts. But Brosh *et al* [BLSS05, §IV] (2002 data) claimed that the distribution of about 150,000 measured loss burst lengths was consistent with iid packet loss, contradicting earlier findings of loss correlation.[28] Brosh *et al* ventured the explanation that more widespread RED deployment may have been the cause, or perhaps there were now more flows at tail drop queues.

Short of gathering empirical evidence ourselves (which is beyond the scope of the present research), our assumption will have to rely on the *intent* of known AQM algorithms. We do test how well AQM algorithms match our theoretical model of their intent, but only using simulations of AQM algorithms (§7.8), not production ones.

The intent of the recommended algorithm for RED was not only to remove the correlated losses that tail drop introduced, but to reduce bunching of marks still further—even less bunching than that expected from iid marking. Using the recommended 'Method 2' algorithm described in the original paper on RED *appears* not to mark each packet independently, because the aim was to make the distribution of inter-mark spacing uniform, rather than geometric.

RED Method 2 works as follows: We denote the discrete random variable (r.v.) representing the packets between marks in the whole aggregate as Z_a. The marking probability increases as a function of how many unmarked packets, z there have been since the last marking $P(Z_a{=}z) = p_b/(2-(z+1)p_b)$, where p_b is the target expectation of marking probability used to drive the algorithm.

Are RED Markings Uniformly Distributed? Nonetheless, this algorithm is applied to the whole aggregate at a link, oblivious to flows. By the following reasoning, we argue that, although RED 'Method 2' is ingenious, it will not achieve its intended aim of uniformly distributing markings *within the same flow*. Instead, within a flow, the marking of each packet will be independent of previous markings, as long as the flow rate is small relative to the link capacity.

Consider flow j consuming proportion r_j of the link capacity, where $0 < r_j \ll 1$, that is the flow is

[28] Unfortunately, this finding was incidental to the primary focus of the paper so the data analysis was not published.

small relative to the link.

Assumption 7.3. *The probability that a packet will arrive from flow j depends only on the relative rate of the flow r_j, and is independent of the flow IDs of previous packet arrivals.*

After flow j has been marked, we count the marks applied to any flow in the aggregate up to and including the next mark to hit flow j again. Let Y_j be the discrete r.v. representing this count. By Assumption 7.3, the probability that the next mark to hit flow j is y marks in the aggregate after the previous one is $P(Y_j{=}y) = r_j(1-r_j)^{y-1}$, because flow j will not be hit $y-1$ times with probability $1-r_j$, then once with probability r_j. In other words, Y_j follows the geometric distribution $Y_j \sim Geo(r_j)$ and therefore the expectation of the number of marks until another hits flow j is $E(Y_j) = 1/r_j \gg 1$.

The discrete r.v. representing the integer number of packets in the aggregate between marks that hit flow j is $Z_a Y_j$. And we denote the discrete r.v. representing the number of packets in flow j between marks in flow j as $Z_j = r_j Z_a Y_j$.

Thus, even though Z_a follows a uniform distribution, Z_j the inter-mark spacing within flow j, is the result of adding Y_j outcomes of Z_a together, where Y_j results from an iid process. Therefore, because $E(Y_j) \gg 1$, Z_j will follow a geometric distribution as if it were the result of a single iid process.

In summary, the 'Method 2' variant of the RED algorithm was contrived to make the inter-mark spacing follow a uniform rather than geometric distribution, but it only achieves this in the *aggregate*. The independent arrivals of packets from different flows makes the inter-mark spacing within any small *flow* revert to following a geometric distribution, unravelling RED's attempt to space the marks more uniformly. Thus RED 'Method 2' seems to be redundant, implying this extension to RED could be removed without detrimental effect wherever it has been implemented.

This concludes our rather convoluted justification for Assumption 7.2.

Distribution of Marks per Window. We can now continue to derive an expression for the probability $P(M{>}m)$ that credit m will be exhausted in one RTT.

$$\begin{aligned} P(M{>}m) &= 1 - P(M{\leq}m) \\ &= 1 - \sum_{i=0}^{m} P(M{=}i). \end{aligned} \tag{7.7}$$

Using notation W for the sending window (the number of packets sent in a RTT), the probability of exactly i marks in W packets is a binomial situation:

$$P(M{=}i) = \binom{W}{i} p^i (1-p)^{W-i}$$

Substituting in (7.7)

$$P(M{>}m) = 1 - \sum_{i=0}^{m} \binom{W}{i} p^i (1-p)^{W-i}. \tag{7.8}$$

We can now apply this formula to two transports: TCP in congestion avoidance and a generalised weighted α-fair transport. We also consider TCP slow start.

Idealised TCP in Congestion Avoidance

For any congestion control algorithm, the window, W will itself be a function of the marking fraction, p. For general congestion controls we will see later that the window is also a function of packet size, RTT and other parameters, but if we use a simplified model of TCP in its congestion avoidance phase (taking the most significant terms from [PFTK98]), the window is only a function of p:

$$W_{\mathrm{TCP-CA}} = \sqrt{\frac{3(1-p)}{2p}} \tag{7.9}$$

Substituting TCP's window formula into Eqn (7.8) results in a formula for $P(M{>}m)$ solely in terms of end-to-end marking fraction p. This formula for the probability that a stable TCP will experience more than m marks per round trip is plotted in Fig 7.6 for the first few values of $m = 0, 1, 2, 3, 4$.

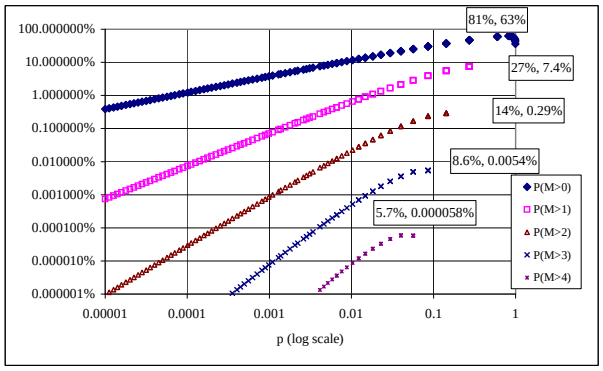
Fig 7.6a) displays the probabilities of TCP experiencing more than each number of marks, m in one congestion window. As the probabilities rapidly diminish with increasing m they are plotted on a log scale, but Fig 7.6b) shows them on a linear scale to help visualise their shape even though all the probabilities of seeing more than two marks can hardly be distinguished from the zero axis. In all cases a log scale is used for the marking fraction p. Also note that the simplified model of TCP's window is increasingly inaccurate for very high congestion levels (above around 10%). Note that only discrete points are plotted because the probability of experiencing an integer number of marks is undefined for non-integer window sizes. Fig 7.6a) also labels the coordinates of the maximum probability for each m, although we reemphasise that the accuracy of our TCP model for $p > 0.1$ becomes increasingly questionable.

For high marking fraction, $p > 0.1$ TCP's window falls to only a few packets. Clearly the probability that there are more than 2 marks in a window with only 2 packets is strictly zero, which is why each plot in Fig 7.6b) hits a peak then the next point drops to zero.[29] For instance, the plot of the probability that there are more than 2 marks per window peaks at 0.29% where the window is 3 packets and the marking fraction is 14%. The window reduces to exactly two packets by 27% marking fraction, where the probability of more than two marks will be precisely zero.

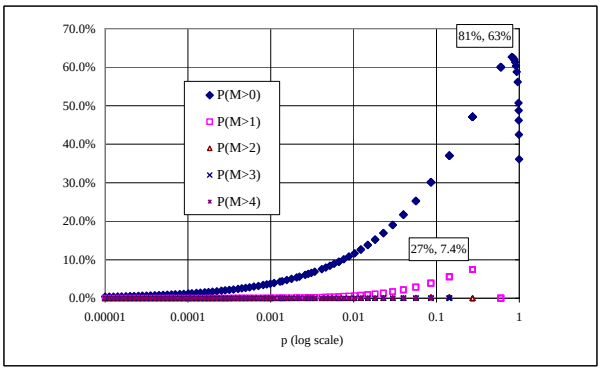
Tentative Conclusion. We can tentatively conclude that it is 'reasonable' to expect the source to be responsible for delay allowance, at least for TCP transports. If our TCP model bears even a slight resemblance to reality, a credit of just two packets would only be insufficient in just under 0.3% of round trips at a congestion level of 14%, when it would be likely that congestion itself would be causing significant levels of drop, not just ECN marking. At a more typical congestion level of 1%, this credit of two packets would only be insufficient for 0.02% of round trips (1 in 5,000). For sources highly averse to drop, a credit of 3 packets would suffice in all but 0.0054% of round trips (nearly 1 in 20,000), even with worst case congestion. And at a more typical 1% congestion, 3 credit packets would suffice in all but 0.00052% of round trips (nearly 1 in 200,000).

Justification for Assumption 7.1. It must be remembered that this interim conclusion is based on a model that is static over a round trip. The RED algorithm aims to smooth its marking fraction over RTT

[29] The points with zero probability obviously disappear off the bottom of the log-scale plots.



a)



b)

Figure 7.6: Modelled Probability of ECN Marks per Window in TCP Congestion Avoidance. Plotted against marking fraction p (model breaks down at higher p).

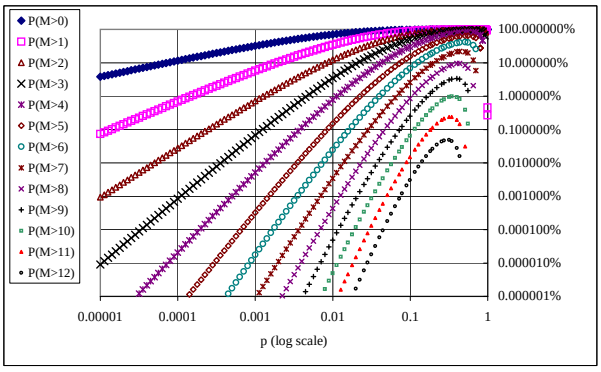


Figure 7.7: Modelled Probability of ECN Marks per Window in 10-MulTCP Congestion Avoidance. Plotted against marking fraction p (model breaks down at higher p).

timescales. Then transports don't have to respond to spikes in queue length due to coincidental bursts of packet arrivals that could disappear without any need for a congestion response. This supports our stationarity assumption.

However, flows with a wide range of round trip times may be present in the same queue. An alternative architecture could expect each transport to filter spikes in its own congestion signal (another end-to-end argument). Then shorter RTT flows would be able to more quickly compensate for congestion changes at their timescale.[30] However, a unary signal of one bit per packet already makes decoding fast changes to congestion sluggish—we wouldn't want to further damp the control loop with too much low-pass filtering.[31] Therefore low-pass filtering in the queue where there is more aggregation makes sense.

Without knowing the RTT of each flow to which each packet belongs, any AQM must choose a smoothing timescale that prevents long RTT flows reacting unnecessarily to queue spikes while not preventing short RTT flows responding to peaks and filling troughs at their timescale.[32] Floyd's advice [Flo08], informed by the considerable literature on setting RED parameters, is to aim for a smoothing time of 1s, or 30 times the propagation delay of the local link, whichever is greater. If production networks mirror this advice, even remotely, then our assumption of stationary marking rate over a round trip is eminently reasonable for typical terrestrial round trip times.

Nonetheless, in §7.8 we provide an initial validation of our model in simulations designed to create highly varying congestion levels.

Average Marks per Window. Note that for TCP in congestion avoidance phase (at least for our simplified model of it in Eqn (7.9)) the expectation of marks per RTT stays below 1 for all loss fractions:

$$\begin{aligned} E_{\mathrm{TCP-CA}}(M) &= pW \\ &= \sqrt{3p(1-p)/2}, \end{aligned} \tag{7.10}$$

which peaks at $\sqrt{3/8} = 0.61$ when $p = 0.5$.

TCP Slow-Start

The slow-start phase of TCP increases the window exponentially at the start of a flow aiming to quickly find the operating point of the path. On receiving feedback from a congestion marked packet, the TCP source considers it found its operating window a round trip ago and halves its window back to the point, thus ending slow start and starting congestion avoidance phase.

By the following argument, two packets should be sufficient credit to avoid TCP slow-start driving the balance of a flow into debt at the dropper, at least if we assume sufficient aggregation so that single

[30] Padmanabhan [PQW03] (Dec 2000 data) is often cited as evidence that loss rates remain relatively constant for several minutes. However, the definition of constant used in this paper is rather weak, only requiring the loss rate to remain within one of a set of loss-rate bands.

[31] This is another motivation for previously mentioned proposals to add more precision per packet to signal congestion.

[32] For an EWMA to smooth over time R^* requires it to smooth over CR^* packets, where C is the link packet rate. If the discount factor per packet of the EWMA is a, it will weight the packet it processed CR^* packets ago by $1/e$ if $CR^* = -1/\ln(1-a)$. Therefore setting the EWMA discount factor $a = 1 - e^{-1/CR^*}$ will discount congestion events R^* in the past by $1/e$ of their original value.

losses will be prevalent. We also assume the source has sufficient data to reach the end of the slow-start phase.

Imagine the nth packet the source sends will be the first to be congestion marked. Before feedback about this event returns to the source, it will have released as many packets into flight (unacknowledged) as have been acknowledged. So, when the feedback from the congestion event on its nth segment returns, the source will have sent n further packets that will still be in flight. At that stage, the source's best estimate of the network's packet marking fraction will be $1/n$. So, as the source will have sent about $2n$ packets, it should have already posted two packets of credit (marked two Cautious) in order to have marked a fraction of $1/n$. In fact, the first and third packets should be marked Cautious, because the first is sufficient credit to cover the second.

Generalisation to a sudden increase. In [BJMS09a, Appx D] Jacquet generalised this argument for any sudden increase in packet rate (e.g. variable bit-rate video), based on the same conservative assumption; to provide enough Cautious marking to cover the possibility that the next acknowledgement received will be congestion marked. It introduced the following notation for the numbers of markings sent or received so far (codepoint names that help explain the letters are given in parentheses):

S: segments sent

F: Cautious (FNE) segments sent

R: Positive (Re-Echo) segments sent

A: acknowledgments received

C: acknowledgments echoing congestion received

The result is the following algorithm for determining how to mark each packet from the start of a sudden increase:

```
when about to send packet (S+1)
if (R<C)
    writeEECN(packet, Positive)
elseif (F+R) < (S+1)*(C+1)/(A+1)
    writeEECN(packet, Cautious)
else
    writeEECN(packet, Neutral)
```

Applied to TCP slow-start, this indeed results in marking the first and third data packets as Cautious, as we originally argued.

One of the motivations behind re-ECN is to turn this discussion on its head and derive flow start behaviour parameterised by how much congestion the transport is willing to risk causing or equivalently how much credit it is willing to post. Thus we can have weighted flow-start as well as weighted congestion response. This will be taken up in future research (§13.2), building on [KM99].

Weighted α-Fair Transport

An important class of elastic congestion avoidance algorithms that includes TCP can be parameterised by the weight (or aggressiveness), w and curvature, α of their congestion response function [MW00]. The setting is distributed so that each resource marks traffic with a shadow price (marking fraction) p and each transport optimises the difference between its instantaneous rate of utility for bit-rate x,

$$U(x) = \frac{w^\alpha x^{1-\alpha}}{1-\alpha}, \tag{7.11}$$

and the cost rate px of congestion at that bit-rate.

The transport's optimisation will converge to the solution of $d(U - px)/dx = 0$. Therefore, bit rate x will converge to

$$x_\alpha = \frac{w}{p^{1/\alpha}}$$
$$\text{or} \qquad W_\alpha = \frac{Rw}{sp^{1/\alpha}}, \tag{7.12}$$

where W_α is the window of packets per RTT corresponding to bit-rate x_α, given packet size s and RTT R.

This formula parameterises all the main distributed elastic network resource allocations:

- With $w = 1$,

  $\alpha \to 0$ models maximum throughput;

  $\alpha \to 1$ models proportional fairness [Kel97a];

  $\alpha \to \infty$ models max-min fairness [Jaf81];

- While free choice of w provides a similar but weighted resource allocation in each case [Kel97a, SCM01];

- $w = \sqrt{3/2}(s/R)$, $\alpha = 2$ corresponds to TCP's resource allocation [MSMO97];

- And $w = n\sqrt{3/2}(s/R)$, $\alpha = 2$ corresponds to MulTCP acting as n TCP flows [CO98].

Reasonable Responsibility? To establish whether it would still be 'reasonable' to make the source responsible for delay allowance in all these cases, we need to establish that the required credit does not scale super-linearly with weight w. A transport chooses a credit, m to cover both the mean marks per window, $E(M)$ and a number of standard deviations $k\sqrt{\text{Var}(M)}$. We will say $k\sqrt{\text{Var}(M)} = \lambda E(M); \lambda > 0$ so that we can say more simply $m = (1 + \lambda)E(M)$.

A higher weight, w will lead to a proportionately higher congestion window, increasing the sample size within the window by w. With a larger sample of trials at the same marking probability p, the mean and variance of the marks per window will grow to approximately $wE(M)$ and $w\text{Var}(M)$ respectively (Central Limit Theorem).[33] Therefore, to achieve the same probability of false negatives, the weighted flow will have to post a credit

$$m' = (w + \lambda\sqrt{w})E(M).$$

[33] The approximation improves with larger window sizes.

Thus weight-normalised credit grows as

$$\frac{m'}{mw} = \frac{1 + \lambda/\sqrt{w}}{1 + \lambda}, \tag{7.13}$$

or alternatively we can say the required credit scales $O(w + \sqrt{w}) = O(w)$.

The variance ($(1 - p)/p^2$) of a geometric distribution is significantly greater than the expectation ($1/p$) for low marking probabilities, therefore the $\sqrt{w}$ term will dominate the scaling of credit until w gets extremely large.

Example. Intuition for this effect can be gained from an example. We take one value of α (TCP with $\alpha = 2$). Then we determine how much more credit a MulTCP [CO98] flow needs if its weight is $10\times$ greater than a single TCP flow. Fig 7.7 shows the probabilities of more than m marks per window for a MulTCP transport with weight $10\times$ that of TCP. We take the single TCP in Fig 7.6 with a credit of two packets as our baseline. We normalise the extra credit MulTCP needs relative to its extra weight. At $p = 0.01\%$ the 10-MulTCP only needs 20% of the weight-normalised credit of TCP, rising to 33% at $p = 1\%$ and 45% at $p = 10\%$. This increase in weight-normalised credit with marking fraction is explained by having to cover the increasing expectation of marks per round trip. But, in all cases, $10\times$ the weight requires considerably less than $10\times$ the credit, because the variance considerably dominates the expectation over the range of this example.

It is important to ensure transports will be able to use weights smaller than 1 as well as greater, otherwise weight inflation increases congestion marking (and the risk of drop) for no extra useful work. Transports that reduce their weight to less than 1 will not be able to proportionately reduce the credit they have to post. In other words, the up-front cost of such a flow will not reduce as much as its running cost will. However, it seems more likely that a low weight will be used for large transfers,[34] so the extra up-front cost should be small relative to the whole flow cost.

Average Marks per Window. For weighted α-fair congestion controls the expectation of marks per RTT

$$\begin{aligned} E_\alpha(M) &= pW \\ &= \frac{Rwp^{\left(1-\frac{1}{\alpha}\right)}}{s}. \end{aligned} \tag{7.14}$$

Reducing α from 2 (TCP) towards 1 (weighted proportional fairness [KMT98] or WPF) reduces the dependence of marks per RTT on p until the expectation for WPF

$$E_{\mathrm{WPF}}(M) = Rw/s,$$

becomes independent of the marking fraction p. Thus, a WPF transport (or any transport with $\alpha \approx 1$) does not need to adjust the credit it posts dependent on prevailing congestion conditions.

Predicted False Hits: Summary

Earlier, once we had analysed TCP, we came to the interim conclusion that the credit required for any TCP flow is not unduly onerous—two or at most three positively marked packets to protect against

[34] The converse of prioritising shortest jobs.

occasional multiple marks within one round trip. We have now provided a model that quantifies the credit required for a generalised weighted α-fair transport. And we have proved that a congestion control with weight w aggregates the risk of suffering unusual bursts of marking within a round trip—a risk that w sub-flows would otherwise each have to carry individually.

We can therefore tentatively conclude that the credit a transport requires to minimise false negatives is not unduly significant for a wide range of possible transport behaviours and therefore the dropper design principle of source responsibility for delay allowance is unlikely to be a barrier to deployment. Using simulation, §7.8 will verify whether these results remain broadly correct in highly dynamic scenarios.

7.7.2 Predicted False Misses

In this section, payoffs from gaming a re-ECN dropper are calculated, assuming the 'Continually Vigilant' algorithm in §7.6.1. Active attacks to pervert the intent of the dropper have already been assessed in §7.5. This section concerns behaviours that transports could discover (whether through malice or innocent experimentation) that might pay off even though they do not behave as the dropper intends, i.e. they do not preserve the integrity of the re-ECN congestion signal.

In all the following cases, z & u are the variables that the 'Continually Vigilant' algorithm maintains for respectively recent Positive bytes per Negative mark and recent Negative bytes per Negative mark.[35] Negative marks are indexed by i with the convention that $i = 1$ for the first Negative mark after the first missing Positive mark. a is the discount factor of the EWMA ($0 < a < 1$). To simplify the analysis, we initially consider a packet flow continuing indefinitely, with constant packet sizes s, stationary packet rate and path congestion marking fraction remaining stationary at p throughout. Then we treat dynamics discursively rather than analytically, discussing the effect of varying each parameter in turn.

Miss One Payment

We first check whether the transport gains from missing just one Positive packet.

Assume the transport had originally posted only one Cautious packet as up-front credit. Further assume the transport has been balancing every Negative mark with a Positive for sufficiently long that the dropper's recent balance of both Positive and Negative marks (after processing each Negative mark) has stabilised to $(1 - a)s$.[36] To save space, we use transformed variables $u'_r = u_r/(s(1 - a))$ and $z'_r = z_r/(s(1 - a))$, given every term of both variables in the following analysis has the factor $s(1 - a)$ in common.

After one missing Positive mark puts the balance into debt, the variables evolve as follows:

$$u'_1 = (1 - a) + a$$
$$u'_2 = ((1 - a) + a)(1 - a) + a$$
$$u'_r = 1.$$

[35] Both equally scaled relative to the z & u defined in §7.3.1, as justified when the algorithm was explained.

[36] Our algorithm produces an unconventional EWMA that is deflated by $1 - a$ (§7.6.1 explains).

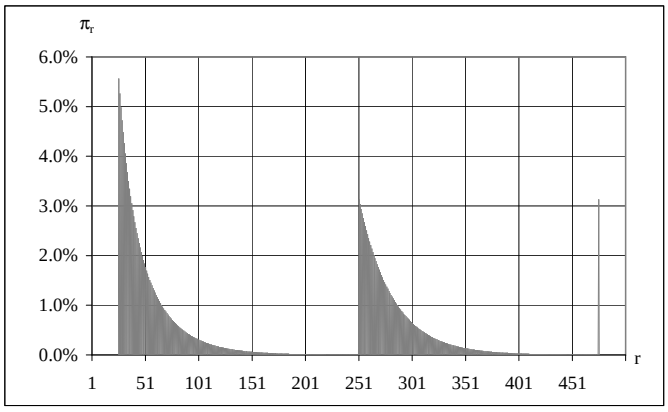


Figure 7.8: Dropper Drop Probability π_r due to Missing One Positive Mark.
A mark is missed at each of congestion mark indices, $r = 25$ & $r = 250$ (note: the drop plot bursts from zero to the top of each curve once per index); EWMA discount factor $a = 1/32$.
The event at $r = 455$ is due to a Positive mark delayed until after the next congestion mark.

The evolution of the positive balance will lack a $+a$ term at the core of the sequence, which we re-introduce and subtract again outside the sequence (the two balancing terms in each case are shown underlined:)

$$\begin{aligned}
z'_1 &= (1-a)\underline{+a} && \underline{-a}\\
z'_2 &= ((1-a)\underline{+a})(1-a)+a && \underline{-a(1-a)}\\
z'_3 &= (((1-a)\underline{+a})(1-a)+a)(1-a)+a && \\
& && \underline{-a(1-a)^2}\\
z'_r &= 1-a(1-a)^{r-1}.
\end{aligned}$$

Therefore, from Eqn (7.4), the drop probability after the rth Negative mark

$$\begin{aligned}
\pi_r &= 1 - \frac{z_r}{u_r}\\
&= a(1-a)^{r-1} \qquad (7.15)
\end{aligned}$$

After missing one payment, the source will send subsequent Positive marks as normal. Each time one arrives at the dropper, the dropper's recent balance will return to credit. But each time a further Negative mark appears, the flow's balance will fall into debt again until the next credit arrives. Therefore, after one missing payment, the dropper will inject a sequence of bursts of light loss between every Negative mark and the following Positive. Each burst will be reduced in amplitude indefinitely by the above formula.

The event after $r = 250$ in the middle of Fig 7.8 visualises the effect of missing one payment after the flow has had enough time to establish itself (in this case the EWMA discount factor $a = 1/32$).

Incidentally, the event after $r = 25$ in the same figure shows how missing a payment while the flow is establishing its record as a 'consistent payer' causes the dropper to hit the flow harder initially. The analysis of this case is not given here, but it is similar to that used in §7.7.2, when the source stops payment completely after q marks. The spike after $r = 455$ is yet another case ('Delay One Payment') discussed in the next section.

The gain the transport will make (in terms of traffic volume, not microeconomics) from the 'Miss One' behaviour is

$$G_{(-)} = 1 - \frac{\text{Traffic dropped}}{\text{Traffic that should be dropped}}.$$

The traffic dropped can be calculated by summing all the drop probabilities after each Negative mark.

To determine how much traffic should be dropped, recall that we have made the source responsible for allowing for round trip delays (§7.3.2). Therefore, by our principle of 'Equivalence with Honesty', if a payment arrives late, over time the dropper should eventually drop as much traffic as the volume forwarded between the first mark to go into debt and the next Positive mark that brings the balance into credit again. The total loss should add up to the same as if there were 100% loss in one of these intervals.

Consider the delay between a Negative packet and each subsequent Positive remains constant (i.e. the RTT remains constant). Then, by the above argument and using Eqn (7.15), the gain from the 'Miss One' behaviour,

$$\begin{aligned} G_{(-)} &= 1 - \sum_{i=1}^{r} \pi_i \Big/ 1 \\ &= 1 - a \sum_{i=1}^{r} (1-a)^{i-1}. \end{aligned}$$

This geometric progression simplifies to

$$\begin{aligned} G_{(-)} &= 1 - a(1-a^r)/a \\ &= 0; \qquad\qquad r \to \infty. \end{aligned}$$

It is not surprising that there is no gain, as the whole point of a moving average is to ensure this happens, at least in a stationary case. However, we have now checked that the 'Continually Vigilant' algorithm does indeed achieve this.

As an example, consider a continuous flow with bit-rate $x = 10$Mb/s consisting of packets of size $s = 1500$B experiencing congestion marking fraction of $p = 0.01\%$. On average there will be $1/p = 10,000$ packets or $s/p = 15$MB from one congestion mark to the next (a round), with the average duration of a round being $\tau = s/(xp) = 12$s. Imagine the source misses a Positive mark completely, then sends each subsequent Positive mark to arrive R =30ms after the Negative mark that triggers it (due to round trip delay). Then, if every subsequent payment arrives 30ms after the Negative mark it balances, the dropper should *aim* to remove $R/\tau \times 15\text{MB} = Rx = 37.5$kB of traffic (or $Rx/s = 25$ packets) eventually altogether.

Further consider that the dropper EMWA discount factor is $a = 1/32$. In the first round, the dropper will aim to discard $a \times 37.5$kB = 1.172kB, or less than one packet (i.e. it will drop one packet

with probability $1172/1500$). However, if the source repays the missing payment in the next round, all the dropper will have discarded is this fractional packet in the first round (the spike at $r = 455$) in Fig 7.8). But if the source never makes up the missing payment, the dropper will go on to remove the full 37.5kB (or 25 packets) if the flow continues for sufficient subsequent rounds (the ski-slope in the middle of Fig 7.8).

Even if it mistakenly misses a payment, a source can ensure it only ever suffers the 'spike' punishment, never the full 'ski-slope'. It can just always conservatively respond to a packet drop with a Positive packet.

Dynamics after One Missed Payment. The transport can vary the speed at which the EWMA clocks on negative marks without changing its bit-rate by sending more smaller packets or less larger packets. However, the dropper counts its balances in bytes, not packets. So if the source clocks the EWMA faster by sending smaller packets, its balances will merely evolve in smaller steps (and *vice versa*).

If, after missing one payment, the sender increases the delay between each Negative and Positive mark, it will suffer proportionately more drop, as it should. If it reduces payment delay it will suffer less drop, as it should.

The EWMA spreads out punishment for missing a mark over time. So if the marking fraction reduces after missing a payment, the punishment will merely take longer, rather than being less severe overall (assuming each round trip delay between mark and payment is the same). The converse is also true; that a higher marking fraction will drop the same volume of traffic but faster.

If the sender increases its bit-rate after missing a payment, all other things being equal, its traffic will collect congestion marks more rapidly. It will then lose the same proportion of traffic but more rapidly. However, it will be the same proportion of a higher volume of traffic, losing more overall. This is correct, because delaying payment at a different bit-rate is a different crime. Conversely, missing a payment then slowing the bit-rate will reduce the total amount of traffic removed. Again, this is correct because a source that delays each payment for less bits should be punished less, even though each delay is for the same time.

The careful (and consequently rather convoluted) logic behind this assertion is as follows. A Positive mark is meant to pay for the right to send traffic from one Negative mark to the next. Negative marks are generated based on the instantaneous product of bit-rate and resource congestion. Directly paying for Negative marks (as Gibbens & Kelly originally proposed) naturally pays for the instantaneous traffic between them. But paying for Positive marks is intended to solve all the issues with receiver-pays congestion pricing. The dropper checks that Positive marks balance Negative and that they arrive no later than the Negative mark they balance. If they do arrive later, the dropper's sanction should be to remove all the instantaneous traffic between the two events. But removing all this traffic immediately would hit accidental transgressions extremely harshly.

Therefore, instead, the dropper spreads the punishment over time. It removes a proportion of the total required punishment while the lifetime balance is negative, but it ceases punishment whenever the balance is non-negative. If the source misses one payment after the mark indexed $i = 1$, but pays for

every subsequent mark after a round trip delay, it is fair to count the payment after the second mark as if it were the missed payment after the first mark, rather than considering the first payment to have been missed forever. In other words, one missed payment is equivalent to all subsequent payments being delayed (and the last missed). Then it is appropriate that the dropper's sanction for each recent payment delay should relate to the volume of traffic sent during that recent delay, not that sent during the first payment delay.

This logic justifies the dropper algorithm remembering the proportion of traffic to remove, rather than remembering the absolute volume of traffic to remove.

Delay One Payment

If the source delays a Positive marking until after the following Negative congestion mark, from the dropper's point of view it is as if one more congestion mark has occurred within a round trip than the source posted as an initial Cautious credit. The dropper's recent balance will temporarily fall into a debt of one packet. If the source is following the re-ECN protocol it will respond to feedback about each congestion mark with a Positive packet. As soon as the first Positive marking arrives at the dropper, the balance will be returned to zero. Then further Positive marks will restore the initial credit.

We assume both the dropper's recent balances start equal to each other ($z_0 = u_0$). After one missing payment the dropper's recent balances update to

$$z_1 = (1 - a)z_0$$
$$u_1 = u_0.$$

From Eqn (7.4), the drop probability from the next congestion mark until the first Positive marking repairs the balance is

$$\pi_r = (1 - \frac{z_1}{u_1})$$
$$= a$$

For example, a spike of drop probability $1/32$ due to a delayed payment is shown at index $r = 445$ in Fig 7.8, which is unsurprising because the dropper's EWMA discount $a = 1/32$.

The source's gain from the 'Delay One' behaviour is zero, by the same argument as that for the 'Miss One' behaviour.

Pay Once Only

A flow could legitimately initialise itself with sufficient Cautious credit on its first full-sized segment, giving itself the right to flow state in the dropper, but then never send any further positive markings. At each congestion marked (Negative) packet, the transport would just continue sending Neutral packets, so the dropper would increase the proportion of drop.[37] But we need to check how much traffic the 'Continually Vigilant' dropper algorithm in §7.6.1 allows through in excess of that paid for.

No flow-state timeout is modelled, as one use for the results will be to determine the dropper's best flow-state timeout policy. So far we have assumed it can be lazy, removing stale flow state only when

[37] The 'Pay Once' behaviour obviously assumes that the transport does not respond to any subsequent drops with Positive marks.

memory is scarce. But the dropper may need to actively detect flows with a continually reducing balance and either set their drop probability to 100% or remove their flow-state.

The dropper will evolve its recent balance variables as

$$\begin{aligned} z_r &= as(1-a)^r \\ u_r &= as[(1-a) + (1-a)^2 + \ldots (1-a)^r]. \end{aligned}$$

From Eqn (7.4), the forwarding probability after the rth congestion mark

$$\begin{aligned} \phi_r &= (1 - \pi_r) \\ &= \frac{z_r}{u_r} \\ &= \frac{1}{\sum_{i=0}^{r-1}(1-a)^{-i}}. \end{aligned}$$

Simplifying this geometric progression gives

$$\begin{aligned} \phi_r &= \frac{a}{(1-a)\big((1-a)^{-r} - 1\big)} \\ &\to 0. \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad r \to \infty \end{aligned} \tag{7.16}$$

One positively marked packet pays for the average volume of traffic arriving at the dropper from one congestion mark to the next. The gain, $G_{(+)}$ to the transport from the 'Pay Once' behaviour is best stated in terms of traffic forwarded (unlike before in terms of traffic dropped):

$$G_{(+)} = \frac{\text{Total volume forwarded into future}}{\text{Volume single packet payment covers}} - 1.$$

Note that the gain is not dependent on packet size. An attacker could send larger packets after the initial Cautious packet, to get more bits through per EWMA clock cycle (from one negative mark to the next), but these would be rejected by the dropper as larger than the maximum size credit packet seen in the flow and handled instead with the Bulk of misbehaving flows.

Therefore, the gain depends on the sum of all the proportions of traffic the dropper forwards between each congestion mark (the area under ϕ_r for all r from Eqn (7.16)):

$$G_{(+)} = \frac{a}{(1-a)} \sum_{r=1}^{\infty} \left(\frac{1}{(1-a)^{-r} - 1} \right) - 1. \tag{7.17}$$

It doesn't seem possible to simplify this expression for the gain,[38] but it converges fairly rapidly against r, so numerical techniques will be accurate enough. It is plotted for a range of values of EWMA discount factor (plotted as $\lg(a)$) in Fig 7.9. $G_{(+)} = -\ln(a) - \epsilon$ seems to be a reasonable estimator for the expression, with $0 < \epsilon < 1$ over the range plotted.

Even though drop quickly rises, particularly for larger values of a, a transport adopting the 'Pay Once' behaviour obviously always gets more traffic through than it has paid for, for all a. Discussion of how to negate this positive gain is deferred until the end of the next section that models the 'Stop Payment' behaviour, which is a generalisation of 'Pay Once'.

[38] It is the most basic form of a Lambert series, which no-one has (yet) been able to simplify further.

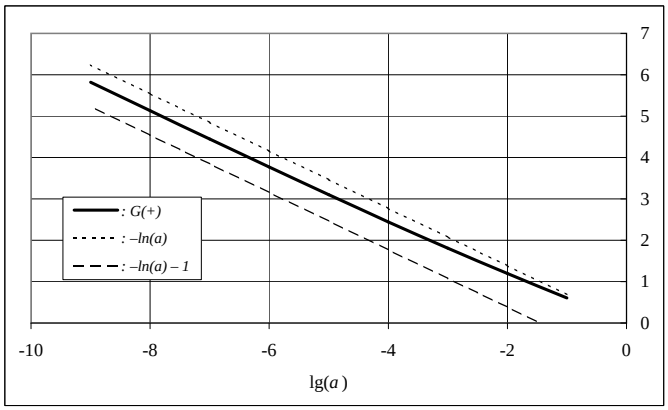


Figure 7.9: Gain from the 'Pay Once Only' Behaviour.
Plotted against $\lg(a)$, where a is the EWMA discount factor (the plot of $G_{(+)}$ is not quite straight, whereas for comparison $-\ln(a)$ is).

Stop Payment

We now establish whether and how much the source would gain from completely stopping sending any Positive marks mid-flow. In the previous section (§7.7.2) this strategy was adopted straight after the first Positive mark. This time, we assume the source pays q Positive marks (including one initial Cautious credit) to balance q Negative marks before stopping Positive marking completely. Consistent with the previous analyses, we use the convention for indexing Negative congestion marks that $i = 1$ at the first mark after payment stops. Therefore the first congestion mark of the flow is at $i = 1 - q$. The flow continues for r marks after payment stops ($i = r$).

The dropper initialises both recent balance variables $z_{-q} = u_{-q} = 0$ and they both evolve over the same geometric series while the source behaves normally, until at $i = 0$:

$$u_0 = z_0 = sa \sum_{i=1-q}^{0} (1-a)^{i+q} = s(1-a)(1-a^q).$$

Once the source stops sending Positive marks, the dropper's recent balance variables diverge until at $i = r$:

$$z_r = s(1-a)(1-a^q)(1-a)^r$$
$$u_r = s \sum_{i=1-q}^{r} (1-a)^{i+q} = s(1-a^{q+r}).$$

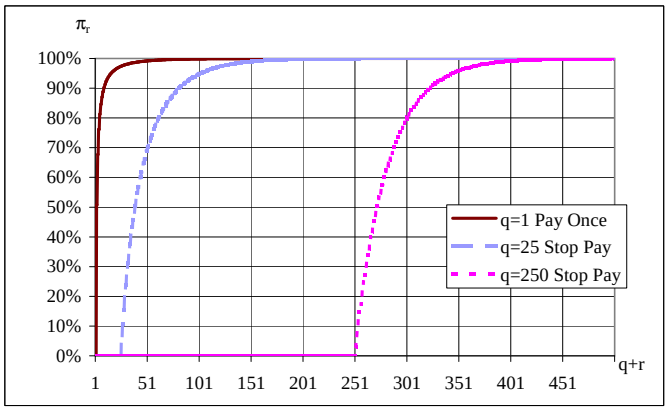


Figure 7.10: Dropper drop probability π_r due to stopping Positive Marking.
Marking stops in 3 scenarios, after congestion mark indices $r = 1, 25, 250$; EWMA discount factor $a = 1/32$.

Therefore, the forwarding probability after the rth Negative mark,

$$\begin{aligned}\phi_r &= \frac{z_r}{u_r} \\ &= \frac{(1-a^q)(1-a)^r}{(1-a^{q+r})}\end{aligned} \tag{7.18}$$

Fig 7.10 visualises the dropper's drop probability as the index of arriving congestion marks, $q + r$ rises. Three cases for when the source stops payment are shown, $q = 1$ 'Pay Once', $q = 25$ & $q = 250$. It can be seen that the sooner payment stops, the faster drop probability rises, which the algorithm was deliberately designed to do.

Similarly to the 'Pay Once' behaviour, the gain to the transport for the 'Stop Payment' behaviour is

$$\begin{aligned}G_{(0)} &= \frac{\text{Flow lifetime volume forwarded}}{\text{Volume paid for}} - 1 \\ &= \frac{\sum_{r=1}^{\infty} \phi_r}{q} \\ &= \frac{(1-a^q)}{q} \sum_{r=1}^{\infty} \frac{(1-a)^r}{(1-a^{q+r})}.\end{aligned} \tag{7.19}$$

This expression generalises the gain from the 'Pay Once' scenario. As in that case, the expression is too complex to simplify as $r \to \infty$. But again it is amenable to numerical analysis as it converges fairly fast. The headline conclusion is that gain immediately goes positive as soon as payment stops. Therefore the gain from this behaviour will need to be negated long before $r \to \infty$.

At the end of §7.6.1 we adopted the attitude that the dropper held a record of which flow-state was most stale, so it could time out this state lazily, as and when it needed more memory for new compliant flows. Our analysis of 'Pay Once' and its generalisation to the 'Stop Payment' behaviour shows that drop clocks up too slowly if the dropper just leaves its regular algorithm running for a flow that has simply

stopped paying.

Such a flow will be obvious to the dropper. The 'Continually Vigilant' algorithm categorises a flow as 'On Remand' as soon as its lifetime balance goes negative, and records the time its balance was last positive. If a flow has been on Remand for more than the flow-state minimum timeout it seems necessary to increase its drop probability much more aggressively than the algorithm currently does (assuming sufficient memory to continue to hold its flow state). We could simply increase drop of on Remand flows to 100% after the timeout, or drop could perhaps rise with time on remand.[39] This would actively drive such flows out of the system, rather than timing out their flow state and treating them with the Bulk of other misbehaving flows. Then flow-state could still be lazily recovered as needed.

The best strategy for the dropper to adopt is left for further research. We do not want to unnecessarily hit flows too hard too quickly if there is a chance an innocent flow might accidentally get into this state. And we already know further research is needed to review how we transition a flow's state when it is consigned to the Bulk (see §13.2). We also don't want to timeout the state of a flow that is obviously misbehaving if we don't need the memory—it might be better off in the Bulk, being treated with an uncontrolled mixture of Negative inputs from misbehaving traffic and Positive remainders from timed out compliant flows.

Predicted False Misses: Summary

We have now analysed a few tractable cases where the dropper might wrongly miss sanctioning transports that are not preserving the integrity of downstream congestion signals. As expected, the dropper correctly prevents any gain from a delayed payment. Where one payment is missed, the dropper's sanction is correct, although it took some effort to argue this for cases where the flow's bit-rate is non-stationary. Where payments stop completely (a missing payment taken to the extreme), the dropper's baseline sanction algorithm is insufficient. The dropper has all the information it needs to actively intervene in such a case, so it could actively introduce a more aggressive drop policy after the flow-state minimum timeout. But we have left investigation of precisely what is the best dropper policy in these cases for further research.

7.8 Simulated Dropper Performance

Simulations were conducted[40] to test the predictions of dropper performance. The simulation plan is listed below:

1. Characterise mean and variance of congestion experienced by test flows;
2. Verify theoretical model of distribution of marks per window, as used to predict false hits (§7.7.1);
3. Sensitivity analysis of false hits against EWMA weight;
4. Sensitivity Analysis of false misses against EWMA weight.

[39] The need for further research to identify a meaningful objective to judge what should be done is recorded in §13.2.

[40] The simulations were planned and designed by the present author. However, a colleague, Toby Moncaster, implemented and executed them.

In all cases, only initial runs have been completed; repeat runs are in progress so the results should only be taken as indicative at this stage.

7.8.1 Simulation Environment

The simulations were designed to create a hostile congestion environment for the dropper, with relatively high congestion that also varied considerably on fast timescales. Two scenarios were used, both with high congestion and high congestion variance, but one lower than the other:

HCHV High congestion, high congestion variance

LCLV Lower congestion, lower variance

In each case, three different test flow scenarios were used with different RTTs.

Simulated Implementation. The simulation was conducted in the ns-2 packet simulator [ns2] v2.30. Draft-07 of the re-ECN protocol in IP and TCP [BJMS09a] and the continually vigilant dropper algorithm v00R[41] (§7.6.1) were implemented in the simulator.

Unless stated otherwise, the dropper was configured to merely log that it would have dropped a packet, rather than actually dropping it. This allowed us to maintain a stable testing environment without the extra confusion of the transport's rate reduction response to each discard from the dropper as if it were congestion. Actual drop was turned on only in the experiment to find the knee of drop sensitivity against EWMA weight.

Even when the dropper was configured to actually drop packets, the transport was configured not to send a Positive packet in response to a drop. This simulated what a cheating source would most likely do. Without configuring it this way, the transport was 'too good' to ever allow any drops after the first one or two.

When we tested the transport with it configured to respond to a drop with a Positive packet, it became apparent that this would be a very robust way for a long-running honest transport to 'learn' how much credit to give to the dropper—if it had previously underestimated. One or two additional Positive packets in response to drops sufficiently topped up the credit to prevent any further packets from being discarded by the dropper—for as long as we could subsequently run the simulation. This is what we mean by 'too good'.

Simulated Topology. In order to achieve high congestion variance, a parking lot topology was arranged so that traffic bottlenecks shifted rapidly from one link to another during the simulation. Network topology consisted of 5 core routers R1–R5 connected by 4 bottleneck links. Senders generating background traffic were connected to each of the first four routers (R1–R4) and receivers were connected to each of the last four routers (R2–R5), all with 10Mbps links.

Link latencies were designed to give a range of RTTs, but shorter RTT paths were omitted to increase the chance of congestion overruns as new flows arrived. Fig 7.11 shows the topology with link

[41] This dissertation avoids details of design iterations in response to early simulation results—in fact there was only one; to randomise initialisation of the remainder in the `probDrop()` function.

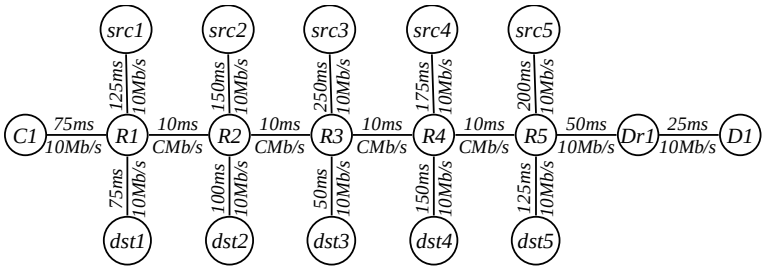


Figure 7.11: Simulation Topology to Test the Re-ECN Dropper.

latencies. Table 7.2 shows the different RTTs of the ten resulting paths through the network, from each set of sources to each set of destinations.

	dst2	dst3	dst4	dst5
src1	470ms	390ms	610ms	580ms
src2		420ms	640ms	610ms
src3			820ms	790ms
src4				620ms

Table 7.2: Simulated Round Trip Times between each Source and Destination.

Link capacities are also shown in Fig 7.11. Average congestion for the two main scenarios was controlled by altering the capacities of the bottleneck links, as specified in Table 7.3. The source test node, C1 was connected to R1 and the destination test node, D1 was connected to R5 via a re-ECN dropper, Dr1.

Model	Bottleneck capacity, C	Background on-off TCP parameters	
		Mean idle	Packet train length
LCLV	10Mb/s	5s	20–1500 pkt
HCHV	8Mb/s	2s	20–150 pkt

Table 7.3: Simulation Parameters Varied to Create the Two Traffic Scenarios.

All the queues used the RED AQM algorithm configured with the default parameters for ns-2. Specifically, this implies RED is attempting to uniformly distribute its marks, which we predict won't make any difference (§7.2), and it is scaling down the marking probability of smaller packets[42].

Three different RTT scenarios are used for the test traffic path, as we wanted to explore the behaviour of flows with RTTs comparable to the lowest, middle, and highest RTTs of competing flows (none of which were small for the reasons already given):

Short RTT 150ms

Medium RTT 380ms

[42] Which we deprecate in [Bri08a].

Long RTT 800ms

A shorter test traffic RTT of 80ms was used in one extra set of simulations. Different test flow RTTs were achieved by varying the access link latencies (Fig 7.11 shows the Medium RTT scenario).

Simulated Traffic Models. All simulated TCP traffic was ECN-capable and used a packet size of 576B. Background traffic consisted of:

- 100 on-off TCP flows (10 per route) with exponentially distributed idle times and uniformly distributed packet trains, characterised by the parameters in Table 7.3. This created the distinction between the congestion variances of the two traffic scenarios.
- 10 continuous TCP flows (1 per route) carrying traffic that modelled FTP behaviour with infinite file size so there was always data ready to be sent.
- 290 UDP flows (29 per route) carrying traffic that modelled aggregated VoIP behaviour [HGB06]: exponentially distributed on-off sources with packet size of 70B; mean burst times of 7.24s and mean idle times of 5.69s. Data rate during bursts was 10.2kb/s.

The test traffic consisted of a continuous FTP source over TCP, which ensured there was always data available to send. In all experiments, results affected by the synchronised transient of the simulation start-up were discarded.

7.8.2 Simulation Results

Characterise Mean Congestion and Variance

Without the dropper operating, the simulator was run with the traffic models specified above to record congestion experienced by the test flow. Mean congestion was measured as the total ECN marked bytes divided by total sent bytes. Variance was measured by collecting results into 1sec bins, and taking the variance of all the binned measurements. A bin size of 1sec was considered sufficiently small, because the characteristic time of RED's EWMA to smooth its marking probability (the time for the EWMA to decay events in the past to $1/e$ of their original size) was configured to the default of 1sec on all links. The overall results are tabulated below (Table 7.4). Note that the table records measurements of

Scenario	RTT/ms	$E(p)$	$\sqrt{\text{Var}(p)}$
HCHV	150	1.302%	1.676%
HCHV	380	1.223%	2.433%
HCHV	800	0.210%	2.136%
LCLV	150	0.325%	0.927%
LCLV	380	0.270%	0.841%
LCLV	800	0.219%	1.173%

Table 7.4: Congestion Mean & Variance for the 6 Simulated Scenarios.

congestion experienced by one flow (the test flow), therefore variance depends on how many packets the flow sends during the binning time. Less packets per second will naturally lead to more variance, because

the discrete steps in congestion levels are greater for smaller numbers of packets. We deliberately took this approach, because we wanted to understand how our test flow saw congestion varying.

As well as recording the above results on 'dummy runs' without the dropper operating, mean congestion was recorded during all experimental runs and generally matched these results.

Absolutely all congestion experienced by the test flow in all experiments was signalled explicitly with ECN marks—the test flows experienced no drops due to link congestion. This in itself was quite unexpected, given the relatively high level of congestion simulated. It was particularly unexpected in the HCHV scenarios with numerous short flows, where many exponential TCP slow-starts were injecting traffic bursts into the queues.

Verify Analytical False Hits Model

This simulation was run to test our theoretical analysis of the distribution of marks per window, which we used to predict likely false hit fractions in §7.7.1. We had used the analytical model as the basis for the dropper design, so we wanted to see whether it reflected reality, when large numbers of TCP and UDP flows are interacting in a highly dynamic rather than stationary environment. In particular the HCHV scenario simulated a high rate of flow-arrivals and departures, which added to the mix a high proportion of flows in TCP's exponential slow start, even though our theoretical model had only modelled TCP's congestion avoidance phase.

At first the six traffic scenarios specified above were simulated for 3000s each. Then very long runs of 15000s were repeated 8 times with different seeds for two scenarios (LCLV with 150ms & 80ms) to establish confidence intervals (see below). The dropper was configured to not drop any packets from the test flow in order to ensure its TCP algorithm only responded to congestion events. Instead the dropper was modified to measure ECN marks per window for different congestion levels. The results of an initial single run for each test are plotted in Fig 7.12. They are shown as larger symbols overlaid on the original theoretical predictions, shown as similar but smaller symbols.

The probabilities of one mark per window were close to the theoretical predictions. But probabilities of higher numbers of marks per window ($P_s(M > 1)$) were all considerably higher than the theory predicted. This was perhaps not surprising as the analytical model we used assumed stationarity, not including flow starts (or any congestion variability) which would lead to brief bursts of queue growth. This would tend to collect more of the overall number of marks into bursts within windows, with less spread evenly throughout the period.

Congestion varied considerably throughout the runs (see the previous experiment), so the average over the whole of each run was used (marked bytes to total bytes in the test flow). Analytical results are only possible for integer values of congestion window, which map to a set of discrete values of congestion p. Therefore linear interpolation between the discrete analytical results was used to produce theoretical predictions at exactly comparable values of congestion.

Further simulations are planned with completely stationary traffic to prove whether the analysis is accurately reproduced by a simulation under the same stationary conditions as assumed in the model.

In the initial 3000s runs, the probabilities of higher numbers of marks per window ($P_s(M > 2)$) are

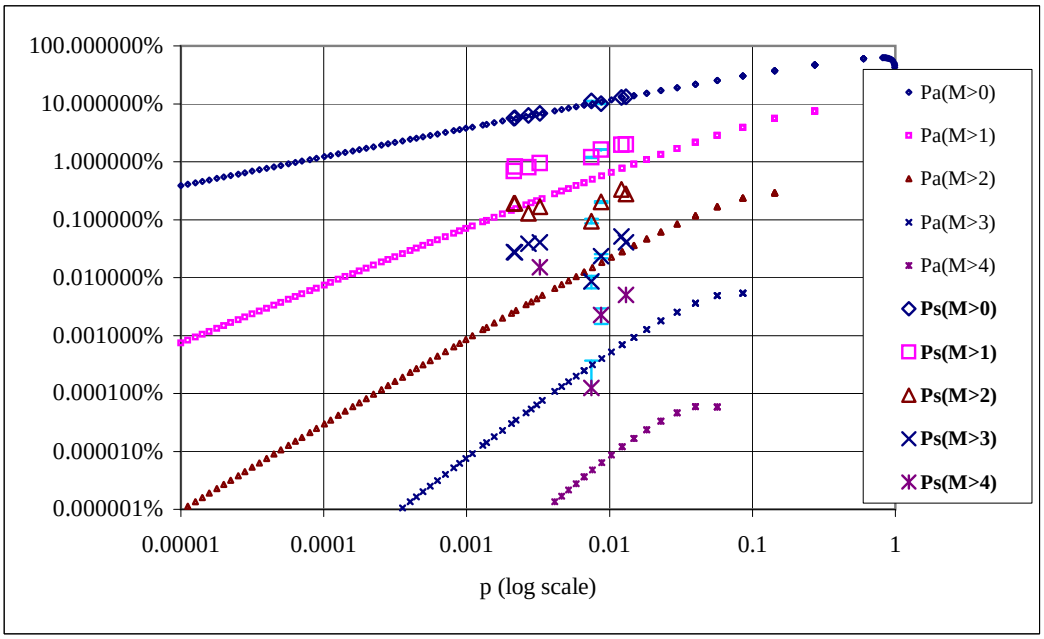


Figure 7.12: Distribution of Marks per Window for TCP against Congestion p.
(Log-log scales); Initial simulated results (the larger symbols labelled $P_s(M > m)$) superimposed over original analytical predictions for congestion avoidance phase (smaller symbols labelled $P_a(M > m)$). 95% confidence intervals are shown for the results at $p = 0.74$ & $p = 0.87$.

p /%	Scen-ario	RTT /ms	$P_s(M>m)/\%$ for $m=$ 0	1	2	3	4	$P_a(M>m)/\%$ for $m=$ 0	1	2	3	4
0.21	HCHV	800	5.65	0.19	0.028	0	0	5.51	0.15	0.0026	3.3E-5	3.1E-7
0.22	LCLV	800	5.72	0.19	0.027	0	0	5.55	0.15	0.0026	3.3E-5	3.2E-7
0.27	LCLV	380	6.35	0.13	0.039	0	0	6.16	0.19	0.0036	5.0E-5	5.3E-7
0.32	LCLV	150	6.85	0.17	0.040	0.02	0.01	6.74	0.22	0.0047	7.0E-5	8.0E-7
0.74	LCLV	150	11.17	1.19	0.095	0.009	1E-4	9.76	0.46	0.014	2.8E-4	4.1E-6
95% Confidence $\pm$			0.05	0.03	0.008	0.002	2E-4					
0.87	LCLV	80	10.02	1.63	0.203	0.023	0.002	10.70	0.55	0.018	3.8E-4	6.1E-6
95% Confidence $\pm$			0.03	0.01	0.007	0.002	7E-4					
1.21	HCHV	380	12.91	0.33	0.051	0	0	12.61	0.76	0.027	6.8E-4	1.1E-5
1.30	HCHV	150	13.18	0.28	0.040	0.005	0	13.02	0.81	0.030	7.6E-4	1.3E-5

Table 7.5: Distribution of Marks per Window for TCP against Congestion p.
Initial simulated results (labelled $P_s(M > m)$) beside original analytical predictions for congestion avoidance phase (labelled $P_a(M > m)$).

based on extremely sparse samples. Despite simulating 1.25 million packets in the test flows (alongside about half a billion in 400 other simulated flows) only nineteen runs of 3 marks, two of 4 marks and two of 5 marks were measured over all six simulations. Therefore the points plotted for runs longer than two marks should be interpreted as part of a potentially wide spread of results. In particular, it should be noted that eight points signifying zero probability of 4 & 5 mark runs could not be plotted on the log scale. The plotted data is also recorded in Table 7.5, which does show the zero results.

The much longer and repeated runs produced a significant number of RTTs with 5 marks per RTT (31) in the $p = 0.87\%$ case (LCLV with 80ms RTT) but only one in the $p = 0.74\%$ case (LCLV with 150ms RTT). Only three windows contained 6 marks, all in the 80ms case. Over 33 million packets were simulated in the test flows alone, with well over 10 billion packets in the background flows.

95% Confidence intervals are plotted for all those results involving repeat runs using different seeds.[43] Although the points without confidence intervals cannot necessarily be trusted, if those with confidence intervals sit within their respective intervals, they seem to imply that the results as a whole do not sit on a smooth line. This would be expected if stationary simulations did sit where predicted and the upward shift was due primarily to congestion variability—at least if the congestion variability correlates with the upward shifts. This suggests a further set of simulations, which are in progress and will be reported in future work.

Our original goal of limiting the false hit rate to the same order as background drop turns out to be rather challenging, given we can detect no background drop at all with ECN enabled ubiquitously, even with fairly hostile dynamic conditions. At least over the working range of these results, they imply that two packets of credit per flow would lead the dropper to introduce on the order of 0.1% additional losses in a network running at about 1% explicit congestion marking. Most current production networks (residential access in the developed world) would typically aim to provision for an order of magnitude less than 1% congestion. But these results imply it is borderline whether 2 credits will be sufficient and we may need to consider 3 as a rule of thumb. However, we must bear in mind these simulations were deliberately designed to create a very dynamic hostile congestion environment.

We can also tentatively conclude that implementations of re-ECN in TCP could reasonably post a constant hard-coded credit, rather than having to do a more complex adaptation to run-time conditions.

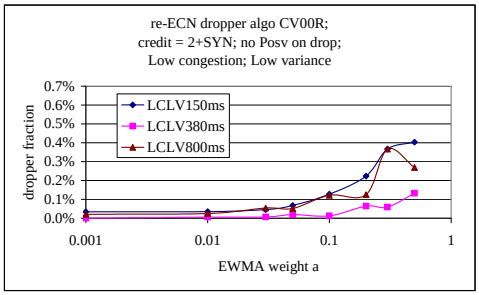
False Hits: EWMA Sensitivity

Various values of dropper EWMA weight were used with an ‘honest’ transport to try to minimise the fraction of false hits—drops punishing relatively innocent behaviour. The term ‘honest’ is strictly relative, given we make the transport responsible for allowing sufficient credit to cover its own feedback delay (§7.3.2). By ‘honest’ we mean the transport was configured to post 2.07 full-sized packets of credit (that is, two full-sized packets plus a 40B TCP SYN). A later experiment tested whether higher credit was necessary in some circumstances.

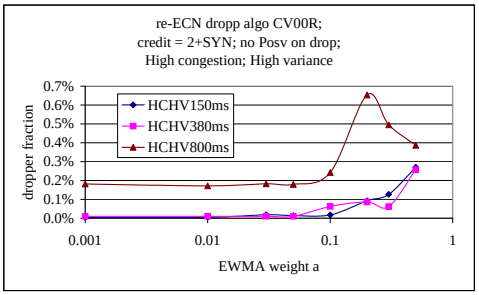
Fig 7.13 shows the results for all six scenarios in the test matrix after a calibration run had been conducted to find the general area of the knee of the curves. Figs 7.13a)&b) show the full range of

[43] The lower bound of the interval for the lowest point could not be plotted on the log scale as it was negative.

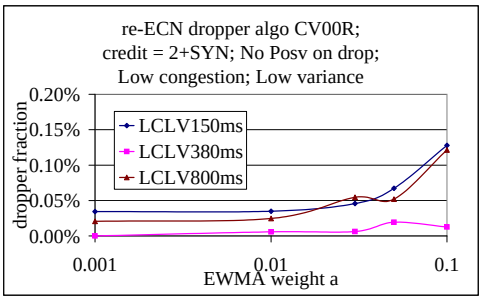
a)



b)



c)



d)

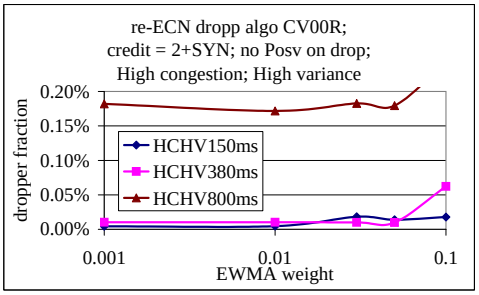


Figure 7.13: Re-ECN Dropper Sensitivity to false hits against EWMA weight a

EWMA weights tested. The wiggles for higher RTTs at higher values of EWMA weight were not due to insufficient samples, but seem to be due to some form of synchronisation. Although sufficient runs to do confidence tests are yet to be performed, these wiggles were repeatable in general form, and remained even for simulation runs an order of magnitude longer. In this experiment the dropper was configured to actually drop packets, therefore TCP was reducing its rate considerably in response to the drops. Given the extremely high EWMA weights used at the top end, it is perhaps not surprising that odd synchronisation effects appeared in the presence of intermittent very high bursts of drop. There may also have been interactions between the long RTTs used and the approximately similar smoothing time of the RED algorithm.

Figs 7.13c)&d) zoom in on the more usable results where false hits are much lower at $a \leq 0.1$. The drop fraction was expected to continue decreasing as the EWMA was decreased further (but the rate of decrease to flatten out). The results seem to flatten off completely at low EWMA weights, but repeat runs would need to be performed to test the significance of the data.

This simulation gave us sufficient confidence in the stability of the results, to be able to use only three values of EWMA from just below the knee of the curve in all our future experiments: $a = 1/16, 1/32, 1/64$ (choosing fractional powers of two reduces the complexity of EWMA implementation, as already explained).

False Misses Sensitivity

This experiment is designed to find the smallest level of cheating that the re-ECN dropper cannot detect. Rather than cheating in whole packets, it involves the source cheating a byte at a time. The source correctly sends a credit of one SYN and one full-sized segment, and it responds to feedback of every congestion mark with a Positive packet, but it stints on the size of each Positive packet.

Rather than simulating numerous scenarios with slightly different levels of cheating until the dropper notices, we fold all levels of cheating into each run of the experiment. Very gradually, the source increases the amount by which it cheats (decreasing the size of Positive packets). We call this ramp-down cheating. Then we measure how long it takes for the dropper to notice—when it starts to drop packets.

We also reverse the experiment to see how asymmetrically the dropper behaves because of the EWMA delay. That is, from a certain level of cheating, the source gradually decreases how much it cheats (increasing the size of Positive packets back to their proper full size). Then, once the source is no longer cheating, we measure how long it takes for the dropper to stop falsely hitting it with drop.

These behaviours are not particularly meant to represent a clever cheating strategy; just a way to test the sensitivity of the dropper to very small amounts of cheating.

Simulated Scenarios. The experimental set-up continues as before, but only using the HCHV scenario, not LCLV. This halves the number of scenarios to simulate, given we are more interested in the dynamic cases. A 3×3 matrix of simulation scenarios is used, with the 3 values of RTT as before, but also with the 3 values of EWMA weight found from the previous calibration experiment, $a = 1/16, 1/32, 1/64$. Full-sized packets are still 576B.

The test flow starts 20s after the simulation. Its precise cheating behaviours are as follows:

- 'Ramp-down' starts by correctly declaring one full-sized Positive packet per congestion mark. Every 20s for 2000s it decrements its Positive packet size by 1B, therefore ending up sending every Positive packet 100B smaller than full-sized (under-declaring by 17.4%)

- 'Ramp-up' starts by making each Positive packet 60B smaller than full-sized in response to each congestion mark (regularly under-declaring by 10.4%). Every 30s for 3000s it increments its Positive packet size by 1B, ending up sending Positive packets 40B larger than full-sized (a regular over-declaration of 6.9%)

In both cases, the source posts an initial credit of one full-sized packet and a TCP SYN (40B). Although the effect of this disappears into the noise relatively quickly, we realised (unfortunately after having run the initial experiments) that we should have allowed time for the initial credit to decay out of the system before the source started to ramp down. This experimental strategy had successfully been used to test a much earlier incarnation dropper, as reported in the original paper on re-feedback [BJCG+05]. It will be used in future runs.

In the ramp-up case, the test of whether a flow's lifetime balance is Positive was commented out of the simulated dropper implementation. This left the test flow only protected from the possibility of drop by its recent behaviour, not its lifetime balance.

Also, in the ramp-up case, we realised after having run the experiments that we had also wrongly changed two things at once—when the test-flow starts, it introduces a step-change in cheating at the same time as TCP starts its exponential slow-start. Future runs will allow TCP as well as the simulation to settle before introducing the initial step change in cheating. Then, as well as examining the ramp behaviour, we can also measure how quickly and how correctly the dropper responds to a step change in cheating of a certain size (equivalent to the 'Stop Payment' behaviour of §7.7.2, but with a smaller step than a complete stop).

Results. The results from the initial run could not be used for their primary intended purpose (dropper sensitivity to tiny levels of cheating), as they were marred by the above teething problems. And further runs could not be conducted given the deadline for this dissertation. Nonetheless, the results from these initial runs are displayed in Figs 7.14 & 7.15 as they reveal some interesting and unexpected effects.

Each column of figures shows all nine scenarios, grouped three per graph. On the left of each page, each group (a–c) has a common EWMA weight to compare the different RTT scenarios. On the right, each group (d–f) has a common RTT to compare the different EWMA weights. Each plot is labelled RXXX-aYY, where XXX is the RTT R in ms, and YY is the reciprocal of the EWMA weight $1/a$.

Ramp-down: The spike of drop at the start of some runs is probably a result of TCP's slow start with only one packet's initial credit to protect it. This will be shifted out of the way of the start of the ramp in future runs, so it can be analysed separately.

The plots clearly show that it takes much longer for the dropper to catch a long RTT flow. We predict this result is produced by two effects in tension against each other. TCP's packet rate is inversely proportional to RTT, so for the same level of congestion, it is picking up far fewer congestion marks over any specific duration. Therefore it will take longer for the initial credit to decay from the system. At the same time, even though it is cheating, each Positive packet partially makes up the balance at the dropper. But the longer RTT means it takes longer before it even partially makes amends. Therefore, the dropper should drop more from a longer RTT flow. But it seems the former effect dominates the latter in these experiments.

A slower EWMA at the dropper seems to harm the cheating flow more strongly. The long-RTT-slow-EWMA case (R800-a64) contradicts this trend and the slow-EWMA case with medium RTT (R380-a64) cannot make up its mind. Otherwise the trend seems to be present. This is because the ramp effectively turns into a case of continually worsening late payment (§7.7.2 explains that a missed payment is equivalent to continual late payments). A slower EWMA takes less note of a late payment. It effectively says "I'll only believe you do intend to pay in full when I see more evidence that you have".

Note that the drop fraction is significantly less than the cheating fraction in all cases. This is because the cheating fraction is nominally stated as the worst amount of cheat between each Negative mark, not the average. Each time a Positive packet arrives at the dropper, the lifetime balance of the flow is less than this worst-case number, which we have used to nominally describe the cheating level. In future runs we will plot the actual cheating level, corrected for RTT and mean inter-mark spacing.

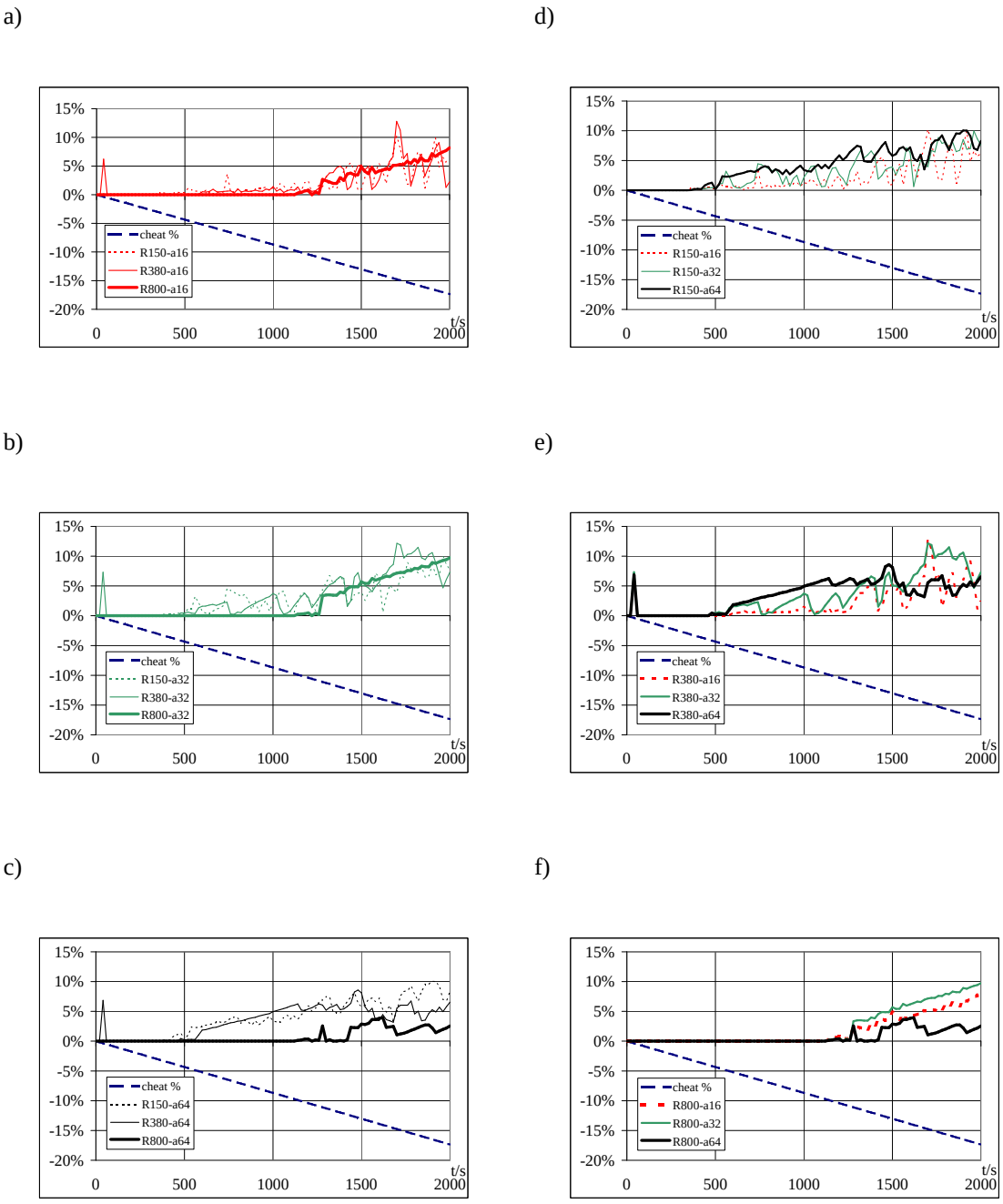


Figure 7.14: Drop Fraction against Time as the re-ECN Dropper Handles a Slowly Ramping Down Cheat.

a)–c) compare RTTs R holding EWMA weight a constant. d)–f) compare EWMA weights holding RTT constant.

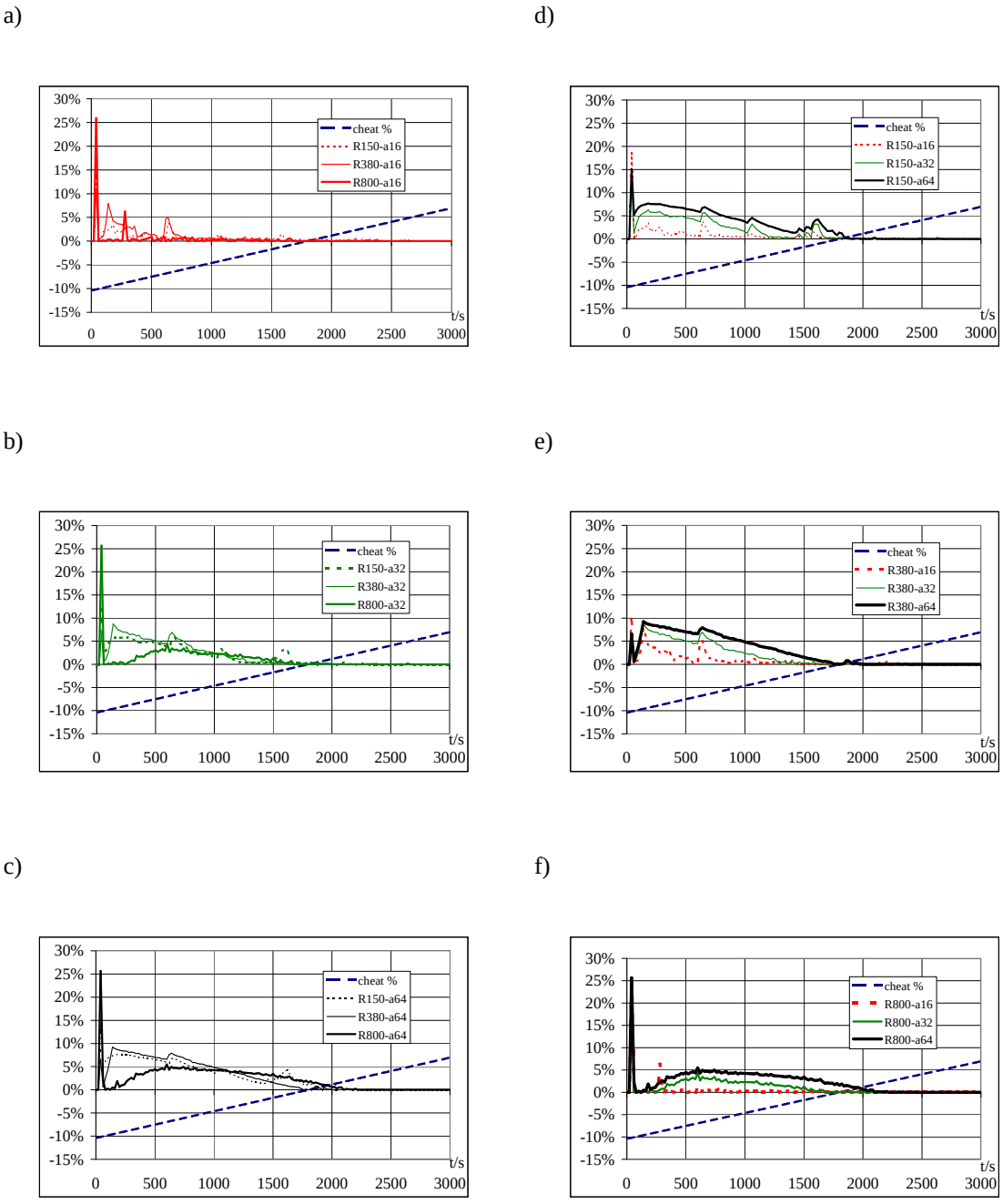


Figure 7.15: Drop Fraction against Time as the re-ECN Dropper Handles a Slowly Ramping Up Cheat. a)–c) compare RTTs R holding EWMA weight a constant. d)–f) compare EWMA weights holding RTT constant.

Ramp-up: The transient at 20s when the test flow starts (not when the simulation starts) can be explained by TCP's slow-start. We assume the receiver feeds back one or two congestion marks when slow-start finds the bottleneck operating rate. Then the TCP source will halve its rate, but at the same time it is arranged to partially under-declare how much congestion it saw. In the next few round trips, this drives the dropper balance strongly Negative, causing a huge spike of reported drops.[44] This is due to dropper being deliberately designed to be more sensitive to misbehaviour at the start of a flow (see the comparison between 'Pay Once' and 'Stop Payment' in §7.7.2).

As the source continues to partially pay its way, two effects work against each other. The dropper becomes gradually more forgiving as it gets further from the flow start. But at the same time, the flow continues to underpay, so drop becomes harsher. The two eventually balance out (which naturally takes longer for the long RTT flow). Then the drop fraction correctly tracks the reducing level of cheating (rising under-declaration) to zero.

On the way an interesting synchronisation effect seems to occur. All the short RTT flows show three shark's fin bumps at about the same times (in independent simulations), all the medium RTT flows each exhibit one bump at the same times and the long RTT flows are all smooth. These are reminiscent of 'aftershocks' from the transient, but a proper explanation eludes us at this stage.

We now move on to what we intended to learn from this experiment. The slower EWMA clearly seems to hurt the test flow more—for the same reason as given above for the ramp-down case. The fastest EWMA seems to hurt flows very little at all. This is because it takes account of a good proportion (1/16=6.25%) of each late payment for most of the duration of each inter-mark period (all of it except the first RTT). Once cheating drops below 6.25%, the dropper only sanctions the cheating flows heavily for 1RTT after each Negative packet, then not at all until the next Negative packet.

Other than delaying recovery from the transients at the start, we cannot notice a discernible effect due to RTT in these cases. More runs will be needed to tighten the experimental variance so we can interpret more predictable plots.

Once cheating falls to zero (at 1720s), periods of 20s or more with no drop at all immediately start to appear in nearly all the scenarios, except the fastest EWMA which takes 2–3 minutes (it is the fastest to punish a cheat and it takes longest to forgive). Periods of no drop lengthen from then on, but it takes about 15mins before all drop-free periods are longer than 2mins. By this point it is difficult to discern whether drop is due to the after-effects of the ramp, or the typical level of accidental false hits. The source is still some way from building up sufficient credit to protect against all false misses (see the earlier experiment designed to validate our theoretical predictions of false hits 7.8.2).

[44] Recall that we have configured the dropper to suppress actual dropping in these experiments.

Chapter 8

Re-ECN Border Incentive Mechanisms

8.1 Border Architecture

8.1.1 Baseline Border Mechanism

The ‘big idea’ of re-feedback is to include information in packets that will be updated as it traverses a network so that, at any point it will reveal the characteristics of the rest of the path. Then this information, particularly congestion information, can be used by the parties either side of a trust boundary to control each other’s incentives.

Re-feedback’s congestion signal integrity depends on the fact that congestion is a physical metric that can never be negative. Therefore a persistently negative flow can never be valid.

The ‘big idea’ of re-ECN is a practical way to instantiate rest-of-path congestion information in the IP header without having to change forwarding elements and fitting within the very limited space left in the IPv4 header. Further, re-ECN is designed to allow metering mechanisms to accumulate an aggregate of downstream congestion-volume over time with minimal complexity.

The baseline mechanism proposed to achieve this is simply to separately count the volume of packets with Positive and Negative markings in a whole aggregate crossing a trust boundary. Then downstream congestion-volume is simply the Positive count minus the Negative.

In this section we will introduce attacks that threaten the simplicity of this baseline border mechanism (§8.2). They were all proposed by others and they all, in some way, use or hide flows that are persistently negative.

We then propose defences against these attacks, which harden the simple baseline mechanism. But they also add complexity. Nonetheless, the solutions keep to design principles that we believe maintain the essence of the original simplicity.

8.1.2 Border Mechanism Constraints

The re-ECN border mechanism should meet the following constraints:

Processing Scalability: It SHOULD keep additional per-packet operations to a minimum (we cannot expect sub-linear scaling in the strict complexity theory sense, because counting marked packets is linear with packet volume).

Storage Scalability: Ideally it SHOULD require no traffic-dependent state (e.g. per flow-state). But if it uses flow-state, it must be optional, growing sub-linearly with number of flows. That is:

- it MUST NOT make flows fail if they move routes;
- how much flow-state to store MUST be controlled by policy, NOT by the number of flows potentially created by malicious sources of traffic.

Incentive Alignment for Congestion Signal Integrity: It MUST ensure that the parties either side of a border have no incentive to pervert the integrity of congestion signals and that each party in turn has the incentive to pass on this incentive to its neighbours.

Attacker Model: Mechanisms MUST be strong enough to ensure congestion signal integrity whether potential attackers are rational or malicious. While the malice of users can be unbounded, the malice of networks can be assumed to be bounded (defined below).

Below some extra detail is given on a couple of these constraints:

Scalability: A useful technology target is to require no operations that would prevent all-optical packet switching at a border with no electronics, given the likely minimal processing and storage capabilities of photonic devices in the next decade or so.

Re-ECN recognises and exploits a distinction between the typical threats posed by networks and those posed by users (defined next). This allows active policing to be located only at the interface between end-users and their network providers, shifting the active policing burden away from the more performance-critical interfaces between high-speed networks.

Attacker Types: The re-ECN framework is primarily an incentive alignment system but it can also be used to *enforce* policies. Incentive alignment is a highly desirable property of any large distributed system, but its power must not be overstated. If an attacker is immune to incentives, coercion is also necessary.

The original statement of the motivations of users and networks in our paper on re-feedback [BJCG+05] considered a range of motivations, but only explicitly enumerated two types:

Rational users: those who want to communicate with each other as fast as possible at minimal charge;

Rational network providers: those who compete amongst themselves for the custom of users by investing in network resources.

The paper hinted that it believed re-feedback could deal with DDoS attacks, but it avoided making formal claims on this front. As we wish to attempt to deal with entities sending traffic with no intent to communicate, we need to define language for a wider attack model. Beyond rational entities, we define two further types (each with user and provider sub-types):

Bounded malicious: a party that is willing to cause costs to others as long as strictly less cost to itself is involved;

Unbounded malicious: a party who is willing to cause cost to others even if the cost to itself is the same or greater.

Assumed Attack Model: In the following section attacks are proposed where one network can gain from attacking another with dummy traffic. We therefore propose the following ambitious assumption that is much harsher on the system designer than simple rationality.

Assumption 8.1. *If networks are malicious their malice is bounded, but the malice of users may be unbounded.*

We believe this assumption is a reasonable model of the real Internet. Some would consider it a somewhat paranoid assumption, as large networking organisation will usually 'play by the rules' and compete with each other for legitimate business, rather than use subterfuge against each other. However, our ambitious aim is to offer an internetworking solution that relies minimally on inter-network trust. The Internet should be able to include *ad hoc* networks of nodes operated even by single individuals, not just large corporate providers with ethical business policies.

The original re-feedback incentive framework [BJCG+05] made similar assumptions, but didn't articulate the attacker types precisely. It assumed that network providers are rational most of the time but, although most users behave rationally most of the time, we cannot rely on all users to behave rationally all the time. By omitting to define the malice of networks, it missed potential attacks by networks on each other.

Re-feedback and re-ECN are designed to provide information on top of which network operators can build engineering mechanisms like policers. Then they can physically block their locally attached unbounded malicious users from causing unreasonable harm to others. However, the original re-feedback paper omitted to check whether networks might be tempted to compromise the integrity of congestion signals in order to make gains from other networks, which in turn might compromise their ability to police end-users. The present chapter corrects that omission.

Nonetheless, it is important to recognise that the malice of networks is not likely to be unbounded. This leaves open the possibility that coercion mechanisms can be confined to the end-customer trust boundaries of the Internet. While at the higher speed borders between networks incentive mechanisms should be sufficient where it is more critical to minimise complexity.

Before we move on, a clarifying statement is necessary. The division between incentive alignment and coercion mechanisms that we propose is not set in stone. One should not confuse the mechanisms we describe with those that network operators *could* build on top of the information re-feedback provides. Network providers are not limited to applying re-feedback information with the mix of incentive and enforcement mechanisms that we propose. They can encourage sociable behaviour or use coercion wherever, however and in whatever mix they wish. Our proposed mechanisms are merely canonical examples that demonstrate how the re-ECN framework can be deployed with minimal additional complexity and minimal constraint on user freedoms while still ensuring congestion signal integrity.

Minimal constraint on user freedoms may not always be desirable. For instance, it will usually be desirable to allow a large range of innovative application behaviours, but not such extreme forms

of innovative behaviour as DDoS attacks, which most operators will want to shut down. Rather than embedding the choice of where to draw this line in the system design, we leave the choice to each network operator. But some possible example border enforcement mechanisms that use re-ECN information to detect anomalies are briefly explored in §8.2.8 (fully specified in [Bri08b, §5.7]). The preferential drop mechanisms in §9 (fully specified in [BJMS09a, §5.3]) are further examples, which we propose to use against flooding attacks in §12.1.1.

Effectively these mechanisms recognise that even entities that are ostensibly rational sometimes might behave irrationally and wish to be protected from themselves when they do. For instance, they may have misconfigured something or been infected by malware. These protections allow thresholds to be placed within the system that prevent a flow, a user or a network from running up costs at an anomalously high rate. A network provider can offer these mechanisms as a service to protect its customers or neighbouring networks from their own failings.

8.1.3 Border Design Principles

The following design principles articulate the lessons learned during the process of developing defences against the more perverse attacks against the re-ECN border mechanisms. Some deliberately run counter to currently accepted research directions. As with all good design principles, they are intended to encourage a designer to give a really good reason before contravening them; they are not intended as unbreakable rules:

Bufferless Border Control: (aka. 'Prefer Measurement to Intervention':) Measurement can be conducted passively, in parallel to transmission, while active intervention (e.g. scheduling) requires packets to be held back until each is deemed acceptable to release. Measurements can be used to influence incentives on longer timescales, creating the incentive to deal with per-packet problems closer to the edge of the internetwork.

Neutralise Don't Over-Penalise Systems shouldn't automatically unleash punitive sanctions on another network thought to be perverting the integrity of congestion information. Otherwise a third party attacker could fool network N_A into penalising N_B by spoofing an attack from N_B on N_A. If negative flows are merely neutralised instead, the problem is at least sufficiently dealt with and no such amplifying attacks are possible.

No Reliance on Push-Back Push-back is an attempt to ensure that unwanted traffic is squelched at source. However, the data plane of a packet network only reliably moves packets forwards. If a control system wants to send messages backwards along a path, it shouldn't take the source address of packets in the data plane as reliable evidence for where 'backwards' is. The route may well be asymmetric, and the source address may be spoofed.

Below we discuss each principle in a little more detail.

Bufferless Border Control. Packet networking is meant to be agnostic to underlying link technologies. Although technologies such as photonics are being developed to make packet forwarding decisions at

line speed, congestion avoidance & control becomes problematic in photonics if a buffer has to exist just in order to generate the congestion information needed to control a transmission line. If, instead, on the forwarding element itself congestion avoidance only requires passive metering, high speed interfaces can play their part in congestion control with only a tiny buffer [AKM04, GM06] or perhaps no buffer at all and just a virtual queue [CW96, KS01, Ear09a] to reduce or to eliminate the need for payload storage.

This principle implies a general structural assumption that usage of re-ECN congestion information will tend to conform to the following pattern: i) at the borders between networks congestion information will mostly be used to underpin contractual penalties based on metering ii) while between networks and their end-customers it could also drive active sanctions, such as drop.

The bufferless border control principle implies it is sufficient to passively measure traffic at borders, not actively remove it. If there is traffic that is polluting the integrity of congestion signals (by understating congestion), this further implies that it is only necessary for border mechanisms to *discount* the polluting information in the traffic, not to *remove* the traffic itself. To remove traffic requires buffering it while testing it, but metrics can be counted (or ignored) in parallel to forwarding. It is more important to ensure the integrity of the information that reveals the incentives to remove traffic and passes the incentives from network to network. Then the correct networks will have the incentive to remove the traffic itself, but each can proceed in this task more lazily.

Neutralise Don't Over-Penalise. During the development of re-ECN we proposed a (misguided) formula for use in the meter between two networks that turned any negative balance over the duration of a flow into positive. The supposed rationale was that this would not just remove the incentive to allow a flow to go negative, but it would turn any gain into an equivalent loss, thus strongly driving negative flows from the system. In effect, this earlier proposal added twice as much congestion-volume as was measured in negative flows.

However, we quickly realised that we should only neutralise the gain from negative flows, not reverse it. Otherwise we opened up a whole new set of attacks where traffic could be sent into a network so that it went negative within the network, causing the network to pay extra to its downstream neighbour. This parallels the 'Proportionate Sanctions' design principle proposed for the re-ECN dropper (§7.3).

No Reliance on Push-Back. In the original re-feedback paper [BJCG+05] we argued against excessive push-back on similar grounds to the 'Neutralise Don't Over-Penalise' principle above. If network equipment is meant to heed a message asking it to drop traffic, it must be pretty certain the message is authentic. But verification itself takes resources.

Instead we argued that re-ECN information itself provides sufficient means to test whether a flow is non-negative using only local information. Therefore we proposed the egress dropper could send hints upstream, with no danger of introducing further attacks, because they would not need authentication [BJCG+05, §3.2.1]. Rather than *instructing* an upstream network to sanction a flow, they *hinted* that it should merely check the flow for itself. However, we did not propose a mechanism that knew in which direction to send the hints, unless the source address was not being spoofed. Also hints cannot be initiated if the attacker arranges the TTL to expire just before the final egress.

We continue to hold this view as a guiding principle. The re-ECN mechanisms aim to be able to work solely on local information, or at minimum only pass information forwards, without any need for reliance on backwards reachability of the source address of packets. If hints can be sent backwards, all well and good, but we don't rely on them. This parallels the 'Source ID Uniqueness not Reachability' design principle proposed for the re-ECN dropper (§7.3).

8.2 Border Attacks and their Defences

8.2.1 Attacks and Defences: Executive Summary

The attacks in the next section (§8.2) fall into two distinct categories:

Dummy Traffic Attacks: These attacks create negative flows using dummy traffic with no intention of communicating any data.

Signal Poisoning with Cancelled Markings: These attacks exploit the fact that congestion marking probabilities combine probabilistically, not additively. The subtraction approximation of the baseline border mechanism relies on the additive approximation, which is only valid at low congestion levels. These attacks distort and exploit the error in the approximation, even at low congestion levels.

The two defences in the intervening sections address each category of attack:

- Dummy traffic attacks are handled by 'Sample-Based Downstream Congestion Inflation' (§8.2.4). The general idea is to correct the bulk packet measurement of downstream congestion taken at a border, by removing the likely contribution from negative flows. As well as bulk packet metering, a sample of flows crossing the border is taken to estimate the likely contribution to downstream congestion from negative flows. Then rather than removing the traffic in the negative flows, we explain why it is more important, and sufficient, to merely prevent the contribution from negative flows polluting the bulk measurement—not counting the information but not necessarily removing the traffic that carries it.

- Attacks that poison border marking proportions are handled by 'Normalising Cancelled Markings' (§8.2.7). Put simply, this is a way to measure downstream congestion without relying on the approximation that congestion marking combines additively. But the challenge is to preserve the simplicity of the original approximate mechanism.

8.2.2 Attack #1a: Dragging Down a Border Aggregate

In early 2006, a colleague, Salvatori, invented a class of attacks between re-ECN networks involving dummy traffic—that is, traffic sent without any desire to communicate with anyone. Network N_A can generate negative traffic itself, which it can send across a border to reduce the congestion charge it pays to its neighbour N_B. The attacking network can optionally limit the initial TTL so that it expires once

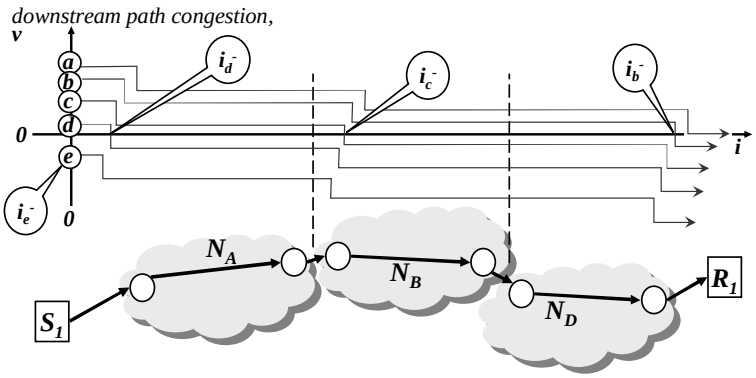


Figure 8.1: Scenarios with different levels of understatement of downstream congestion. The flow becomes negative a) never; b) in the egress network N_D; c) in transit network N_B; d) in the ingress network N_A; e) at the sender.

the traffic has crossed the border.[1] Then, if the only detection of negative flows were at egress edge droppers, nothing would even detect the attack.

Example Scenario. As examples of Salvatori's attack we will use the scenarios in Fig 8.1. The bottom half of the diagram shows the topology of network interconnection. N_A is paying congestion charges to its downstream neighbour N_B based on the downstream congestion level signalled in packets and likewise N_B is paying N_D. The top half of the diagram visualises downstream congestion-volume against resource index along the path shown. In scenarios (a)–(e) the congestion signalling introduced by the various networks is the same[2]. But in each case the source initialises Positive markings at different levels so that downstream congestion first becomes negative at the different resource indices i_b^- to i_e^- within the different networks along the way. In Salvatori's attack the source is under the control of network N_A.

Gain & Cost: If upstream network N_A contracts to pay N_B related to[3] the amount of downstream congestion it forwards across the border to N_B, then N_A gains immediately from including negative flows in traffic crossing the border, as in scenarios (d) & (e).

No network gains by accepting flows from upstream that will go negative in its own network. But any network does have an incentive to pass negative flows undetected to onward networks. For instance, network N_B will lose in scenario (c) if N_D removes any contribution to border settlements from negative flows. But if N_B can somehow conceal negative flows under the cover of other flows, network N_B will be willing to accept the flows in scenarios (c)–(e) just as readily as those in (a) & (b).

There is no immediate cost to a network like N_A from mounting dummy traffic attacks, other than the risk of sanctions if its attack is detected.

Means & Opportunity : This class of attacks can be classified into two forms:

[1] The normal behaviour of Internet routers is to decrement the TTL and discard packets when the TTL reaches zero.

[2] Each trace is slightly staggered from the next, merely to avoid confusing overlaps.

[3] We use the phrase 'related to' deliberately, because any contract based on some sanction against downstream congestion will create an incentive to cheat.

1. The introduced traffic uses flow IDs unrelated to any existing traffic;

2. The introduced traffic imitates flow IDs already present in the traffic mix, but it introduces just enough Negative marking to completely or partially negate the Positive markings already in the flow.

Flow ID imitation is easier for networks than for users because networks can trivially monitor the flow IDs of existing traffic.

Such attacks can be further classified depending on whether the attacking network arranges the TTL to expire before the destination, thus leading to the two orthogonal classifications of attack in Table 8.1.

	No expiry	Expiry
No ID imitation	1-i	1-ii
Flow ID imitation	2-i	2-ii

Table 8.1: Classes of Border Dummy Traffic Attack

Risk of Detection: If the attacking network N_A deliberately expires the TTL early in neighbouring network N_B (class ii), persistent TTL expiries can be detected by routers as anomalous, particularly if the majority of their re-ECN markings are negative.[4] This could trigger management action to trace the attack back at least to the previous upstream neighbouring network. If N_A uses N_B as a transit and arranges the TTL to expire in some network N_D beyond, the attack becomes harder to trace back especially if the source addresses used create a false trail. But the flow would already have to be negative before entering N_B (scenario d or e in Fig 8.1) otherwise N_A would not gain.

If TTL expiry is not used, the attacker has to choose the destination address it uses with care. In the form of attack using new flow IDs but no TTL expiry (class 1-i), if the attacking network sends packets to invalid destination addresses, numerous 'no route' errors from negative packets will also raise alarms. If it sends to existing but unwary hosts, they will probably silently absorb the packets in the general noise of everyday DoS attacks on the Internet. The attacking network would also have to choose a source address for its attack packets. If it chose invalid source addresses but valid destinations, the chosen destinations might again silently absorb the traffic.

In the form of attack imitating existing IDs (class 2-i), the attacking network would run a higher risk of detection given flow ID imitation is a clear contravention of accepted practices and neighbouring networks could arrange for test traffic to cross a suspected networks to detect if imitation traffic was being added.

In summary, it seems class 1-i attacks have least risk of detection, where the TTL doesn't expire and existing flow IDs are not imitated. As long as all attack traffic is sent to valid and powered up destination addresses it stands lowest risk of being detected as long as alarms raised within the chosen destinations are not reported to their network operator.

[4] Which distinguishes them from legitimate traceroute TTL expiries.

However, we will stop there, before this taxonomy of detection methods becomes too tedious, as our defence against this attack below aims to remove any gain to the attacking network from launching the attack in the first place.

As long as the likely gain can be minimised, even a tiny risk of detection is likely to deter a network operator from cheating a neighbouring network. If any subterfuge were ever discovered, significant loss of reputation would result, leading nearly all networks to refuse to interconnect with the attacker for some considerable time. Even if a network operator could invent a 'perfect crime' with no risk of detection, it would still risk whistle-blowing by disgruntled staff.

8.2.3 Attack #1b: Dummy Background Congestion

Before moving on, we note that Bauer and Faratin have proposed two dummy traffic attacks that have similarities to Salvatori's 'Dragging Down a Border Aggregate' attack.

- The first [Bau05] is a strategic attempt to use other people's money to confuse another network into investing in capacity. It is discussed under the heading 'Strategic Confusion of Investment Signals' in §12.1.1 rather than here, as it is an attack against the whole re-ECN system (indeed, against congestion charging), not just the border mechanisms.

- In the second attack [BFB06] a source sends large amounts of traffic without inserting any Positive packets, just to increase congestion costs for everyone else. Again, because it is an attack against the whole system, it is discussed in §12.1.1 under the heading 'Dummy Neutral Background Load'.

We mention these attacks here because they could be launched by networks (rather than users) against other networks. They are also mentioned here because the 'Sample-Based Downstream Congestion Inflation' defence should remove a network's motivation for these attacks in the same way as for Salvatori's (if the defence works as claimed). Although the attacks need not cost anything directly, a network is usually also concerned about the cost to its reputation if detected.

8.2.4 Defence #1: Sample-Based Downstream Congestion Inflation

We now present a solution that aims to remove the incentive for networks to include negative flows in the bulk of traffic crossing into a neighbouring network, leaving no motive for *networks* to perpetrate the above dummy traffic attacks. It doesn't directly remove the incentive for *users* to mount these attacks, but it should push back the motivation for hunting out and controlling such users to the network where a flow first becomes negative.

The proposed solution is to use per-flow state for a small but truly random sample of the traffic crossing a border. Then the bulk congestion-volume metered passing from an upstream network to a downstream neighbour can be inflated by the proportion of excess Negative bytes found in persistently negative flows in the sample.

We have not established whether sampling would work precisely enough in practice nor what size samples would be needed to give sufficient precision. However, the technique is offered more as an ar-

chitectural direction than a fully worked through mechanism. We openly admit that these ideas currently sit on weak foundations and more work is needed before we can claim there is potential in this direction.

First we will give the rationale for this solution, which runs counter to currently accepted research directions. Then we will describe the steps in the process of sample-based downstream congestion inflation, each covered in subsequent sub-sections:

1. the formula for inflating the congestion-volume
2. a random sampling mechanism
3. a process for neighbouring networks to agree on the inflated charge

Downstream Congestion Inflation: Rationale

Example Scenario. As an example we will use scenario (c) in Fig 8.1, where an attack source in network N_A arranges for a flow to go negative in the middle of network N_B. Note that this is not the same scenario as Salvatori's attack ((d) or (e)), but we use it to explain the more general benefits of downstream congestion volume inflation.

The scenario (c) flow is still positive when it crosses into network B. So, irrespective of any downstream congestion inflation, N_A would always pay N_B the same amount for this flow, which would only cover the cost of congestion just part-way through N_B's network—insufficient to cover N_B's congestion costs, let alone those of networks further downstream.

The Problem with Negative Flows. If the scenario (c) flow went undetected and was merely counted by a bulk meter without any downstream congestion inflation, N_B would pay a negative charge to N_D for the contribution of this flow. This implies money contributing to this flow would actually pass from N_D to N_B, against the data flow. We must emphasise that there would not actually be an itemised bill showing this single negative flow—there would just be one item on the bill for the bulk of all packets. We are merely saying that the contribution from this flow's metered packets would subtract from the bulk bill.

This reverse money flow would cover N_B's congestion costs, but the 'wrong' network would be paying for them. N_D would not be able to raise the revenue to pay this charge without levying a charge against the receiver. But we always want to avoid money flows having to start at receivers, otherwise they become vulnerable to denial of funds attacks. Therefore N_D bears the cost of a problem that started in N_B.

Outcome of Downstream Congestion Inflation. We now further assume that all the networks are applying sample-based downstream congestion inflation to the bulk congestion metering between them and, for now, we assume that it works correctly.

With downstream congestion inflation, N_B pays N_D nothing for flow (c). But N_B still doesn't receive enough income from N_A to cover the cost of the congestion caused by the flow. Also N_D receives nothing to cover the cost of congestion caused in its network. Therefore N_D and N_B have an incentive to sort out the upstream problem. The network within which the flow goes negative (N_B) might

only detect this negativity at its egress (if at all). Networks downstream of the point where the flow goes negative (N_D) might detect flow (c)'s negativity at their ingress and their egress (if at all).

N_A, on the other hand, has no problem and can see no problem; its congestion is fully paid for, because it gets less from the source S_1 but also pays less to N_B. If the source were the one lying about downstream congestion, N_A would allow it to send at a faster rate. But as far as N_A is concerned, it would allow S_1 to go that fast for the congestion just within its own network. N_A has no local evidence that the flow is negative, so if downstream networks have allowed the flow to go negative, N_A can rightly say that is their problem.

Therefore there is no problem until downstream of N_A. N_D has insufficient income, but its upstream neighbour, N_B already has an incentive to solve the problem. So N_B is left with a problem that no-one upstream has, so no-one except N_B and beyond is incentivised to solve it.

Summary so far. The network where a flow first goes negative is in the position where it has a problem, it knows it has a problem, and it can probably find which incoming interface is causing the problem. The network where negativity first arises is left with a problem that no-one upstream cares about, so no-one except itself and networks further downstream is incentivised to solve it. This doesn't sound good, but it gets better, and the alternatives are much worse.

Removing Inter-Network Attack Motives. What downstream congestion inflation does achieve is to remove the incentive for networks to attack *each other* with dummy traffic. In scenarios (d) or (e), N_A no longer reduces its bill by sending negative dummy traffic into N_B, for instance. If we assume the malice of a network is bounded, it will not risk being detected attacking another network for no gain. Therefore, if we assume the malice of networks is bounded while the malice of users is unbounded (Assumption 8.1), at least all networks are now either on the 'same side' or 'neutral'—none are on the 'dark side'.

An example of a 'neutral network' is N_A in scenario (c), which doesn't care about N_B's or N_D's problems. But with downstream congestion inflation at least N_A no longer has an incentive to use dummy traffic to take money from N_B using an attack scenario like (d) or (e). Examples of networks on the 'same side' are N_B & N_D in scenario (c), who both want to solve the same problem.

Downstream congestion volume inflation (if sufficiently precise) ensures all entities whose malice is bounded have the incentive to co-operate against those entities with unbounded malice. This argument generalises to any scenario that lacks congestion signal integrity, such as any of (b)–(e) in Fig 8.1.

Cement Network Co-operation First. Faced with a flow such as the one in scenario (c), we do not directly propose that N_B or N_D should ask N_A to squelch it at source. Our 'No Reliance on Push-Back' principle doesn't require that (§8.1.3). Rather we propose that it is paramount to encourage networks to co-operate against a common enemy first, by removing incentives to attack each other.

Local Solutions Second. Once aligned (by downstream congestion inflation or perhaps a better future invention), a network can discard negative flows as it detects them. But removal of unwanted traffic can

proceed lazily (i.e. not at round-trip time-scales).[5] The downstream congestion inflation process samples flows locally looking for negative ones anyway. So as it finds them it can also trigger a rule to route them to the null interface. And other ways may be found/invented to locally seek out and destroy persistently negative flows.

Trace-Back Hints Third. We can also consider sending hints backwards along a path, as briefly outlined above in §8.1.3. As we have already pointed out, 'backwards' is an ill-defined concept in a packet network. We do not propose to include trace-back solutions in this dissertation—that research field is large in itself and distinct from the work here [Bel00, SPS+02]. The one contribution we can add is that re-ECN allows the negativity (and therefore undesirability) of a flow to be tested locally. So whether trace-back solutions are hop by hop or edge-to-edge [HH07], their authentication requirements can be weak or non-existent, as the messages can merely be hints to check locally whether flows are negative.

Attack the Root Cause Lazily. Even if N_A never squelches the flow at source, everyone's traffic problem would be sufficiently solved if N_B discards the flow at its interface with N_A. N_B could then send a message to N_A across their local border interface saying "I'm discarding flow (c), so if you want to save paying me for it, you can discard it for me." Then N_B needs one less flow filter and N_A can save a little money. N_A may eventually find the source and deal with the root cause. But all this can proceed at the time-scale of a management system, rather than of packet control.

Against Punitive Sanctions. We have already argued (§8.1.3) that N_B's problem SHOULD NOT be solved by altering the border incentives any more than by neutralising negative congestion-volume. Otherwise we would introduce new possibilities for attack.

Downstream Congestion Inflation Formula

Consider a set of flows J. Each flow, index j, if metered on a per-flow basis would consist of some positively marked volume V_j^+ and some negatively marked V_j^- (the latter variable being considered numerically negative). The sum of these two volumes is the downstream congestion volume caused by each flow, $V_j = V_j^+ + V_j^-$. A set of such flows can be visualised laid out along the horizontal axis in Fig 8.2, ranked for visual convenience in order of downstream congestion-volume.

It may help to visualise this sum V_j using each shaded area shown in each flow. The inset on the left of the figure explains the graphical shorthand used for one of the flows. The left of the inset shows the actual values of $V_j^+ \& V_j^-$ and the right of the inset shows shading from zero to the midpoint, which is a useful half-scaled representation of their sum V_j. The volume of Neutral packet markings is irrelevant, so not shown.

We want the sum of downstream congestion caused by all flows, except ones that are negative[6]:

$$V_f = \sum_{\forall j \in J} (V_j^+ + V_j^-)^+. \tag{8.1}$$

[5] We shall see (§12.1.3) that a re-ECN ingress policer can considerably slow down unwanted traffic that changes flow ID continually. Therefore, to launch a more serious attack implies having to keep the same flow-ID for more packets, giving more time to remove them.

[6] The notation $(X)^+$ means X if $X \geq 0$ or zero otherwise. Perhaps confusingly, the notations V^+ & V^- use the same operator, but on a packet-by-packet basis before summing packets together.

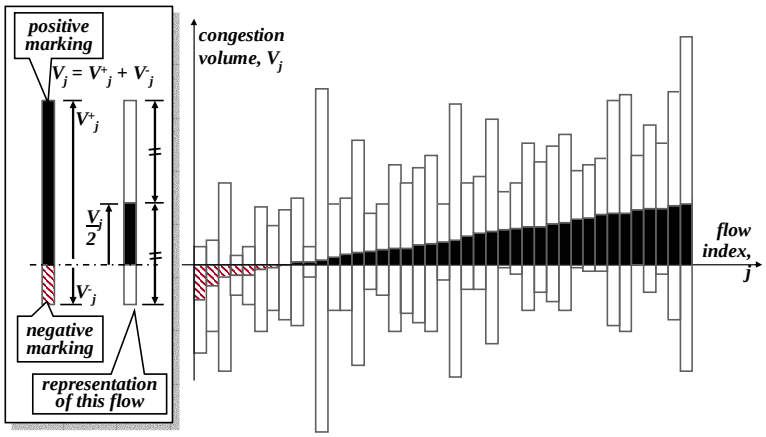


Figure 8.2: Visualisation of the Border Congestion Metering Problem.

This can be visualised as twice the black shaded area shown above the axis.

If we just meter the bulk congestion-volumes of Positive and Negative packets in all flows crossing an interface over an accounting period, we will get the (incorrect) bulk metered volume of congestion,

$$V_b = \sum_{\forall j \in J} V_j^+ + V_j^-. \tag{8.2}$$

This would be represented graphically by the black area above the axis minus the (red) striped area below it, all doubled. The problem is how to exclude the striped areas below the axis, but without accounting separately for every flow.

Assume for a moment that we can take a truly random sample of traffic comprising the subset of flows $I \subset J$. If the sample is small enough so that accounting separately for each flow is feasible, then we can measure downstream congestion both for the whole of the sample and solely for those flows in the sample that are positive. We denote the ratio between the two measures found in the sample:

$$\varepsilon_I = \frac{\sum_{\forall j \in I}(V_j^+ + V_j^-)^+}{\sum_{\forall j \in I}(V_j^+ + V_j^-)} - 1 \tag{8.3}$$

Then we can inflate the bulk downstream congestion-volume measured without regard to flows to get an estimate of downstream congestion excluding persistently negative flows.

$$E(V_f) = (1 + \varepsilon_I) V_b$$

However, even if all sources were honest, a very small proportion of downstream congestion-volume could be contained in flows that are negative overall. For instance, numerous honest flows would only contain a single datagram, and even if every single datagram flow were started with a Cautious packet, some would be congestion marked to Negative. Therefore, we actually want a deflated fraction of the above estimate to take account of the inflation factor $(1+\varepsilon_H)$ that would be found if we performed the same two measurements (sample and bulk) on a set H of purely honest flows.

$$E(V) = \frac{1 + \varepsilon_I}{1 + \varepsilon_H} V_b, \tag{8.4}$$

where ε_H is defined for the set H of purely honest flows, just as ε_I was defined for the set I in Eqn (8.3) above.

Congestion-Volume Sampling

Taking a truly random sample for inflating the bulk congestion measure requires some careful thinking. Each honest flow will tend to start with a positive balance, which it will maintain by balancing Negative with slightly delayed Positive throughout its duration, but sometimes there may not be a final balancing Positive packet. Therefore, flows cannot be picked by randomly selecting packets then looking for further packets with the same flow ID. This would tend to bias towards the end of flows, often missing off the credit at the start, while always including the debit at the end.

If we want to avoid per-flow state for all flows, we cannot randomly select from packets that start flows. This is because we cannot rely on the first packet of every flow being honestly set to Cautious, because we are trying to also detect malicious flows. And it is only possible to know that a packet starts a new flow if a list of all the currently active flows is maintained, which contravenes our original goal.

One possible sampling mechanism is to randomly pick a subset of the possible flow IDs, and detect all packets that match the subset over a period, before moving on to another subset. Then, it would be necessary to take the average of all the inflation factors from each subset weighted by the volume of traffic each subset matched. Obviously, the flow IDs used by Internet hosts are not random, because IP addresses, protocol IDs and port numbers are unevenly allocated and unevenly used, particularly because the port number space includes a number of well-known ports. Also, it is quite likely that misbehaviour is concentrated into certain parts of the flow ID space. Therefore, over an accounting period, the aim would be for the samples to have collectively covered most of the possible address space.

Even if all the flows are positive when accounted for as a whole, whenever a sampling technique only measures part of some flows, it will erroneously find some flows that appear to be negative overall. Therefore, the period over which a sample should be taken must be many times longer than the duration of flows that most traffic is in. Note the careful wording, "…the duration of flows that most traffic is in" is not the same as "…the duration of most flows".

Agreement between Neighbours

Whenever two neighbouring networks determine the size of the settlement that one must pay the other by measuring traffic crossing between them, they must trust each other or a third party. Even if they both meter the traffic, either party can simply lie about what their meter said in order to dispute the other's reading. However, even if they both don't deliberately lie, one can only build trust in the other if the other party's reading consistently agrees with the one read privately.

If the charge between neighbours depends on a bulk measure metered continuously but inflated by a sampled measure, it is important that the party in control of the traffic cannot infer when sampling is occurring. Otherwise it can condition traffic to be well-behaved during sampling and behave badly otherwise.

It may be possible for both parties to hire a trusted third party to conduct the measurements independently of each of their interests. The third party might actually operate the meter physically secured against both interested parties, or it might produce a tamper-resistant meter for them to use that neither party believes can be influenced by the other.

If sampling is used but without a third party, given neither interested party will want to inform the other when or what they are sampling, it will not be possible to ensure that both parties measure the same data. The two parties can only build trust in each other if their two readings are close, or if they are neither persistently higher nor lower than the other. The two parties readings will only be close to each other if the sampling technique is strongly representative of the traffic in total. Experiments will be necessary to establish whether this is the case.

In summary, the problem of neighbouring networks agreeing on a meter reading is not fundamentally different if sampling is used, but sampling does make it more difficult to build trust in each other's measurements if the resulting readings are unlikely to match closely.

8.2.5 Attack #2a: Signal Poisoning with Cancelled Markings

Even though it is not a normal part of the re-ECN protocol, there is nothing to stop packets being initialised with Negative or Cancelled markings (see §12.1.4 for an exhaustive check of all the possible but invalid state transitions of the re-ECN wire protocol).

In the previous section we discussed the possibility of a network sending packets that it created Negative in the first place. The protections against persistently negative flows that we describe elsewhere[7] should deal just as well with malicious Negative marking when packets are first initialised as when they are forwarded. So if any sender were to initialise packets with Negative marking, it would have to initialise as many extra packets to Positive marking to ensure the flow was not detected as persistently negative.

Cancelled as Poison. However, there are no such protections against an attack first proposed by Handley[8] involving initialisation of packets with Cancelled marking.

Cost & Gain: Any packets that a source initialises as Cancelled have no worth (± 0) so they can be sent without any cost to the sender, but they are immune to further congestion marking (they are effectively already marked), and they need no Positive packets to balance them. This seems to open up a flaw where a malicious source can initialise many, or even all, packets with Cancelled marking and achieve resistance to Negative marking. Then it can send at whatever rate it wants, and it will never have to send any subsequent Positive packets.

This attack is shown in Fig 8.3ii) relative to how the marking proportions should be in Fig 8.3i).[9] If fake cancelled markings are introduced early in the network path, they will reduce the proportion of Negative marking (because they are immune to further marking). Thus they will also reducing the need for the source to mark so many bytes Positive. Hence in Fig 8.3ii) the fake Cancelled markings have been shown replacing some Negative, some Neutral and Some Positive markings.

Risk of Detection: We introduced the Cancelled state, to remove a vulnerability of the previous re-ECN protocol coding, which had just three states, Positive, Neutral and Negative (see Appendix B.1).

[7] Congestion volume inflation at borders (§8.2.4) and the re-ECN dropper incorporated into the ingress policer (§11.3 described later).

[8] At the 5th CRN/CFP architecture working group on a Denial-of-Service Resistant Internet, Cambridge, UK, 21 Nov 2005

[9] Cautious markings are ignored as already discussed in §7.4.5.

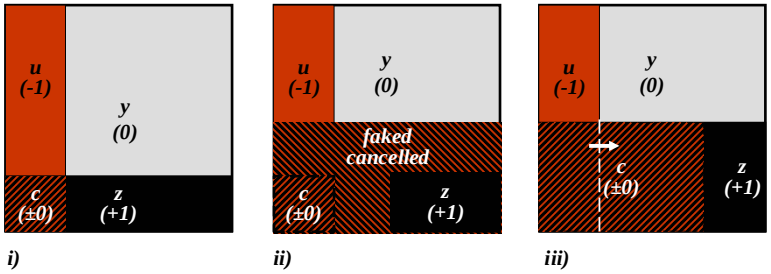


Figure 8.3: Signal Poisoning with Cancelled Markings.
i) Correct case; ii) Faked Cancelled markings reduce Negative and Positive markings; iii) Detection (see text).

The Cancelled state was also added to introduce some useful redundancy to enable detection of biased marking by networks. Happily this also allows Handley's proposed attack to be easily detected.

We refer to Fig 8.3iii), which has the same areas shaded as Fig 8.3ii) but just rearranged. If a network sees a proportion of bytes congestion marked above the horizontal line as Negative (presumed originally sent Neutral), it should see the same proportion congestion marked below the line as Cancelled (presumed originally sent Positive). Therefore the divide between Cancelled and Positive should be at the dashed line. If the proportions have moved to the right, as shown by the arrow, it knows some upstream network or user is artificially introducing more Cancelled packets. This balance between marking proportions should always exist at any point on a path, no matter how much congestion marking is still to come.

8.2.6 Attack #2b: Extreme Upstream Congestion

Even though Handley's attack can easily be detected, a different form of the attack is still possible to mount with congestion marking proportions that balance properly—passing the above test to detect Handley's attack. For instance, consider a path over two networks. The upstream network can say congestion in its network is very high, say 90.9% (conveniently chosen because it is $10/11$), much higher than in the downstream network, which for the sake of example we will say is 1%. Then the correct fractions of each marking (ignoring Cautious) to 3 significant figures will be (Fig 8.4):

- 8.27% Positive
- 82.7% Cancelled
- 0.818% Neutral
- 8.18% Negative

It can be seen that claiming high upstream congestion levels allows a network to legitimately send very high proportions of Cancelled bytes. Note that the proportions of Neutral to Negative and Positive to Cancelled are both the same (about 1:10), so the upstream network is immune from the downstream network forcing these proportions to be the same—they already are.

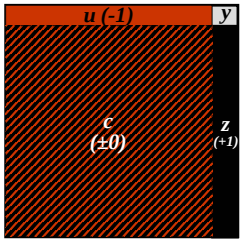


Figure 8.4: Signal Poisoning with Extreme Upstream Congestion.

The upstream network seems to do very well from this attack. All the Cancelled packets it sends cannot be further congestion marked. And if the downstream network charges it by subtracting Negative bytes from Positive, downstream congestion only appears to be 0.09%, when in fact it is 1%; eleven times greater. Therefore it seems to be very much in a network's interest to appear to be highly congested to the downstream network; by marking a lot of packets Negative and even more Cancelled. This is a particular problem if the sender operates its own network (e.g. a home or campus network). It can claim that it is experiencing very high levels of congestion as an excuse to send a very large proportion of Cancelled (and Negative) packets.

8.2.7 Defence #2: Normalising Cancelled Markings

It will be recalled that Handley's attack to poison the congestion signal by introducing faked Cancelled markings (§8.2.5) is fairly easy to detect given the deliberate redundancy in the re-ECN wire protocol encoding. However, the similar attack with extremely high congestion marking by an upstream network, although detectable, seems to be perfectly legitimate—the proportions of redundant markings can be are perfectly balanced. But the upstream network has to pay significantly less to its downstream neighbour if it is charged by Positive minus Negative bytes transferred.

Fortunately, there is a fairly simple way to thwart both these attacks. The key to the solution is *not* to use the approximate formula $z_i - u_i$ for recent downstream congestion. In §6.2 we derived the precise formula[10]:

$$v_i = \left(1 + \frac{c_i}{z_i}\right)(z_i - u_i)^+. \tag{6.4}$$

This accurate formula inflates the approximation we have used up until now by c/z (removing subscripts). In the example above, for instance, the approximate formula for downstream congestion yields 0.09%, whereas this precise formula results in an inflation factor of 11, yielding the correct answer: $(1 + 82.7/8.27)0.09\% = 1.00\%$. This removes the incentive to send Cancelled packets.

Unfortunately, this still isn't a solution. If a downstream network charges an upstream network using this formula, we will now prove that it becomes in the upstream network's interest to *reduce* the volume of Cancelled packets it sends *below* the correct proportion.

Consider a network is also the source of traffic, so it is free to alter the proportions of markings, but it is being charged for downstream congestion in traffic it forwards by Eqn (6.4). Further consider

[10] Now that we are considering attacks, we include the additional constraint $(z_i - u_i)^+ = \max(z_i - u_i, 0)$.

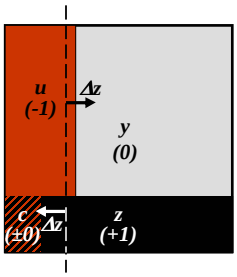


Figure 8.5: Deflating Cancelled Markings to Gain from Metering applied using Eqn (6.4).

it adopts this strategy (illustrated in Fig 8.5): it alters Positive by Δz, but it keeps $z - u$ and $z + c$ unchanged by also altering Negative by $\Delta u = \Delta z$ and altering Cancelled by $\Delta c = -\Delta z$. Given downstream congestion is

$$v = \frac{(z-u)(z+c)}{z}$$

then, by definition, the numerator is constant, therefore the change in downstream congestion that the network is charged for will be

$$\begin{aligned} \frac{dv}{dz} &= -\frac{(z-u)(z+c)}{z^2} \\ &= -\frac{v}{z}. \end{aligned} \tag{8.5}$$

This derivative being negative proves that, if the source network is charged by Eqn (6.4), its strategy of increasing Positive and Negative markings while decreasing Cancelled markings accordingly will significantly reduce the charge it pays.[11]

Nonetheless, there is an approach to thwart all these strategies, ensuring integrity of the congestion signal. §6.2 shows that we can also express the inflation factor of the subtraction approximation for downstream congestion in terms of u & y:

$$v_i = \left(1 + \frac{u_i}{y_i}\right)(z_i - u_i)^+. \tag{6.5}$$

If this formula were used to determine the upstream network's charges, the upstream network would want to shift markings back; inflating Cancelled and deflating Negative markings (moving the divisions in the *opposite* direction to those shown in Fig 8.5).

Because of the redundancy in the marking fractions, both these formulae (6.4) & (6.5) are equivalent. Therefore, the trick is for a downstream network to use them *both* and takes the higher result of the two to measure the charge its upstream neighbour should pay.

$$v_i = \left(1 + \max\left(\frac{c_i}{z_i}, \frac{u_i}{y_i}\right)\right)(z_i - u_i)^+. \tag{8.6}$$

Then the best strategy of the upstream network will be to keep the ratios of Cancelled and Negative bytes balanced. For typical low congestion conditions, these formulae produce only very small inflation

[11] This is not necessarily the network's optimal strategy, but it is always a highly gainful strategy that is simple to describe.

factors.[12]

The reason attack #2b (extreme upstream congestion) worked when charges were calculated without these formulae was because an upstream network could fake very high congestion in its own network, which led to a high level of Cancelled bytes. With the approximation, these weren't taken into account. With the precise formula, they are, thwarting the attack.

However, if just one of the formulae is used, a cheating network can still unbalance the proportions of the two pairs of markings to reduce its charge. But a cheating network cannot mount this attack when both formulae are used, because a gain on one formula leads to a loss on the other.

Precise Downstream Congestion Meter Algorithm

Below we give a pseudo-code algorithm that can output the result of Eqn (6.4) continuously as a moving average of downstream congestion from a live aggregate packet stream. It also simultaneously outputs the integral of downstream congestion-volume. It only uses single cycle machine instructions (adds, subtracts, comparisons & bit-shifts) in order to minimise processing cost. Developing a similar algorithm for Eqn (6.5) has been left for future work. It looks similar but transforming the algorithm will not be completely straightforward.

Appendix A.2 gives the pseudocode for an earlier algorithm we used to implement the full formula (8.6). The same appendix also gives the results of tests on the implementation in C of that earlier algorithm. Subsequently the simpler more elegant algorithm below was invented. It is therefore presented in preference, even though only one of the pair of formulae to compare has been coded and tested.

The new algorithm below exploits a simple trick. Rather than inflating $(z - u)$ by $(1 + c/z)$ it deflates $(z + c)$ by $(1 - u/z)$, because $(z - u)(1 + c/z) = (1 - u/z)(z + c)$.[13] The same functions, variable names and names for constants are used as in the dropper algorithms in §7.6.

```
/* Downstream congestion meter */
 meterDownCong() {
    /* Initialise variables */
    V = 0 /* downstr congestion-volume*/
    v = 0 /* recent downstr congestion*/
    z = 0 /* recent Positive markings */
    u = 0 /* recent Negative markings */
    r = 0 /* remainder */
    a = EWMA_WEIGHT
    foreach packet {
        s = readLength(packet)
        eecn = readEECN(packet)
        switch(eecn) {
        case NEGV:
            u += (s-u)*a
            z -= z*a
        case POSV:
```

[12] Recent whole path congestion (as opposed to downstream congestion) is always given by $z + c$ anywhere on the path. Path congestion, not downstream congestion, would be the measure needed to drive a per-flow rate policer (specified in [BJMS09b, Appx B2.]). But per-flow rate policing is merely an attempt to force compliance with an arbitrary standard; there is no implication that the use of path congestion can be related to incentives. Nonetheless, in such cases, the inclusion of the fraction of Cancelled bytes provides a different disincentive against re-marking packets to Cancelled when they should be Neutral.

[13] Inflating by $(1 + c/z)$ is possible, but inelegant and long-winded—that was the first attempt!

```
            z += (s-z)*a
            u -= u*a
        case CAUT:
            /* Will probably want to
             count separately */
        case (ECT(0) || Not-ECT):
            /* Depends on policy
             e.g. may rate limit */
        case default:
            /* NEUT & CU: do nothing*/
        }
        if (eecn == (POSV
                 || CANC)) {
            r += u
            if (r < z) {
                /* 1-u/z CANC or POSV
                 pkts reach here */
                V += s
                if (u < z) {
                    v += (s-v)*a
                } else {
                    /* don't count -ve
                     downstr cong */
                    v -= v*a
                }
            } else {
                r -= z
            }
        }
    }
}
```

The algorithm works broadly as follows. The assignments within the (`switch(eecn)`) logic maintain two moving averages for recent Positive and Negative markings, `z` & `u` respectively, depending on whether the packet's extended ECN marking is Positive or Negative. As with the dropper, their values are meaningless other than relative to each other. Both EWMAs clock on the same events (a Positive or Negative mark) to ensure this is so.

The most elegant part, that calculates $(1 - u/z)(z + c)$ without division or multiplication, is in the last three nested ifs. Whenever a Positive or Cancelled packet arrives it is a candidate for counting towards downstream congestion (the $(z + c)$ term). The remainder variable `r` increments by `u` each time such a candidate appears and also decrements by `z` whenever it has climbed to be greater than `z`. Therefore, `r` will climb for $(1-u/z)$ of the candidates and sawtooth down every (u/z) of the candidates. It adds all the candidate packets except those picked by the downward sawtooth to its running total of downstream congestion-volume, so it selects $(1-u/z)$ of the Positive or Cancelled packets to add. Note that whenever $u > z$ the algorithm takes downstream congestion as zero.

Fig 8.6 shows the evolution of the algorithm's main internal variables including the remainder `r` sawtoothing between `u` and `z`. The algorithm's output, recent downstream congestion v, is also shown.

It might be possible for an attacker to synchronise with this algorithm to get its smaller packets picked and its larger ones not. But if the synchronisation drifted by just one packet nothing would be

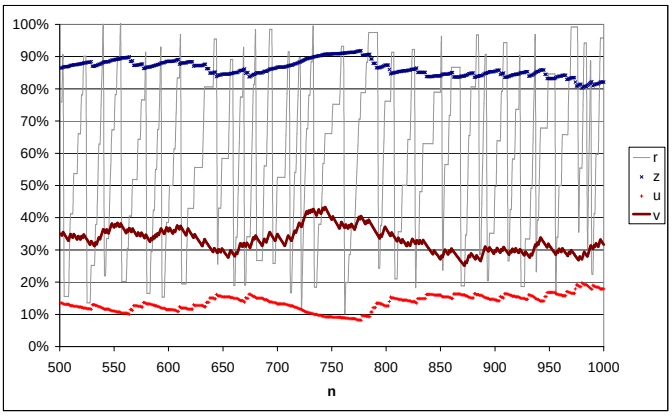


Figure 8.6: Inflation of Downstream Congestion to allow for Cancelled Markings.
Evolution of the Algorithm's Internal Variables. The horizontal axis is an index of arriving non-neutral re-ECN markings. The remainder variable r can be seen sawtoothing between u and z (see text). The downstream congestion intensity measured separately over this run was 31.6%, which the algorithm metered with a +0.11% error (EWMA weight, $a = 1/64$).

gained. If this did prove a problem, some randomisation could be added to desynchronise the deterministic sawtooth a little.

As already mentioned for previous dropper EWMA algorithms, multiplications by the EWMA weight can be implemented as bit-shifts if the EWMA weight a is chosen as an negative integer exponent of 2.

Meter Implementation and Testing

An earlier more complicated algorithm was implemented and tested satisfactorily. The simpler algorithm above has not been fully tested due to lack of time. It has been implemented and briefly tested over a stream of uniformly random markings (not in a network simulator, just a stream of numbers). Early results while the EWMAs settled were discarded. Totals of each marking were collected separately and the result from the meter compared with the result of substituting the totals directly into the Eqn (6.4).

Extremely high downstream congestion was simulated to ensure there was a difference to measure. Stationary values of 40% for path congestion and 10% for upstream congestion were used (each independently randomised), leading to expected values for z, u & c of 36%, 6% and 4% respectively and 33.33% for the output v. However, it would not have been appropriate to test the accuracy of the meter by continuous comparison against these numbers, as the randomisation should lead the instantaneous values to vary. Therefore, these values were merely used to drive a randomised marking process.

To verify the accuracy of the meter, the total downstream congestion-volume it accumulated was recorded along with the total numbers of each marking it actually generated over each run. Because the marking process was deliberately arranged to be stationary, substituting the total of each type of mark

directly into the Eqn (6.4) gave a baseline downstream congestion-volume against which the meter's gradually accumulated total could be compared.

Over 30 runs of 500 packet markings, the sum of all readings was +0.11% greater than the correct result. The standard deviation of the errors from each of the 30 runs was 2.03%. A much larger number of tests will be necessary to establish whether there is any bias in the algorithm, but these initial results seem promising.

8.2.8 Defence #3: Using Congestion Marking to Detect Anomalies

We have assumed (Assumption 8.1) that networks are often rational, but if they are malicious their malice is bounded. We have already said that some may consider this a little paranoid, but it may sometimes be too naïve. Even if a network intends to behave rationally, it can sometimes (usually accidentally) behave irrationally, often due to human error.

Re-ECN is fundamentally a system to align incentives, with each party expecting others to behave as they would expect a rational individual or organisation to behave. But the congestion signalling re-ECN provides is not just limited to providing incentives, at least as long as the congestion signals themselves are not tainted by some human error so that they always appear reasonable, even during anomalies.

As well as congestion charging mechanisms at borders, there could be additional protections that use the information re-ECN signals provide to detect seriously anomalous behaviour. For instance, a very high bit-rate of a flow or aggregate with very high congestion marking, might be considered sufficiently abnormal to trigger management action to block it.

In the specification of re-ECN for use with pre-congestion notification (PCN) [Bri08b, §5.7] we describe random selection of Negative packets and subsequent monitoring of further packets in the flow these packets belong to, in order to watch for highly negative flows. Unlike the above downstream congestion inflation mechanism, the random selection required in this case is deliberately biased towards highly negative flows by picking randomly from the subset of packets that are Negative.

The referenced section should be read for further details. This section has merely been included for completeness, highlighting that both downstream congestion inflation and anomalous flow detection are recommended at borders—one is not a substitute for the other.

In a similar vein, re-ECN information could better discriminate between otherwise similar looking anomalies. Examples already mentioned include unusual proportions of Internet Control Message Protocol (ICMP [Pos81]) 'time expired' or 'no route' errors from packets which also have a Negative re-ECN field.

8.3 Border Incentive Mechanisms: A Review

Neither of the two approaches introduced in this section to harden re-ECN's baseline border mechanism have been fully built or tested.

Defence #1: As admitted up-front, 'Sample-Based Downstream Congestion Inflation' is offered as an architectural direction. Although the architectural arguments might be plausible, the likely precision of the sampling approach seems questionable and is certainly unproven. If the approach

works, it would remove the motivation for dummy traffic attacks #1a) & #1b). Without any gain from the attacks, but with considerable risk of detection and consequent loss of reputation, we assume networks would be unlikely to mount such attacks.

Defence #2: Normalising Cancelled Markings sits on a little stronger ground. Prototypes of the algorithm have been built and it seems to work as claimed. But much more testing is required. Also the full algorithm has only been implemented rather inelegantly. Half the algorithm has been re-implemented based on a new idea producing simple elegant code, but the whole algorithm will need to be implemented and fully tested before any claims can be made with more certainty. Again, if the algorithm proves to give unbiased results under further testing, it will remove any gain from attacks #2a) & #2b) that attempt to poison congestion markings with excess Cancelled packets.

These sentiments can be formalised into a brief review of whether the two proposed extensions to re-ECN border mechanisms stay within our original constraints:

Processing Scalability:

- To normalise Cancelled marks, per-packet operations (using an admittedly incomplete algorithm) can be described as minimal. Excluding reading packet fields, the longest path through the unoptimised code (for Positive packets) is 14 operations, each a single cycle. For the majority of packets (Neutral), only one per-packet operation is required (to decide to do nothing). Negative packets require 6 single-cycle operations.
- For downstream congestion inflation, per-packet operations are only necessary for picking flow IDs to sample. Although the code has not been implemented, subsequent per-packet operations for picked flows should take just one extra operation on top of reading packet fields.

Storage Scalability:

- To normalise Cancelled marks no per-flow state is required.
- For downstream congestion inflation, an unknown proportion of sampled flow-state would be necessary. But state would very likely scale sub-linearly with number of flows and malicious traffic cannot cause state to exhaust.

Congestion Signal Integrity & Incentive Alignment: Within the attack model of Assumption 8.1 the arguments for the two schemes to harden re-ECN's border mechanisms have proved that the integrity of congestion signals is assured (if the mechanisms work as claimed).

Chapter 9

Re-ECN Forwarding Element Behaviour

We have assumed throughout that the contribution of forwarding elements to the re-ECN system is very simple: they merely implement a standard first in first out (FIFO) active queue management (AQM) algorithm such as RED [FJ93] to mark packets that are ECN-capable. As we have already made clear (§7.4.4), the re-ECN incentive framework is designed to share the resource of those parts of an internetwork with ECN deployed. It is reasonable for re-ECN to be designed on the basis that ECN is already deployed, given that we can expect any network provider that does decide to deploy re-ECN policers, droppers and border meters around its network to also turn on ECN.

Re-ECN requires no changes to forwarding elements that are already ECN-capable. However, two optional (but recommended) changes are proposed below:

- The section on 'Congestion marking of Cautious packets' proposes that forwarding elements optionally congestion mark one of the two extended ECN codepoints (Cautious) that would not be marked if the current ECN specification [RFB01] were followed;
- The section on 'Preferential Drop' is not about how an AQM algorithm *writes* congestion signals into packets, but about how it might *read* re-ECN markings—using them to determine its own drop treatment of arriving packets when under stress.

Only informal analysis of these aspects of the re-ECN protocol has been conducted (e.g. in §12.1.1 on flooding attacks).

9.1 Re-ECN Preferential Drop

As pointed out in §7.4.4, one cannot assume an ECN queue will never overflow and consequently discard packets. One can only assume drop will be rare as a normal operating behaviour. Also an ECN forwarding element might even have to discard packets for an extended episode, for instance during a concerted denial of service attack, or due to some misconfiguration.

Once traffic carries re-ECN markings, the opportunity arises for forwarding elements to use them to determine which traffic it drops first. A link under severe congestion (e.g. a DoS attack) will congestion mark most of the packets that it manages to forward, and drop the rest. A well-behaved receiver will feed these back to the source. And a well-behaved source should reduce its rate and set the Positive marking on nearly all the packets it sends in future rounds.

Therefore, if a queue is overloaded, it SHOULD drop packets with markings not used by re-ECN first (Not-ECT and ECT(0)). Then, if still stressed, it should drop Negative, Neutral and Cancelled[1] packets before finally dropping Cautious and Positive packets if absolutely necessary. These drop preferences are summarised in Table 9.1.

EECN codepoint	Drop Preference
Cautious	3
Positive	3
Neutral	2
Cancelled	2
Negative	2
CU	2
ECT(0)	1
Not-ECT	1

Table 9.1: Proposed Drop Preferences for a re-ECN-aware Forwarding Element.
1 means drop first.

A misbehaving source or receiver might not be trying to communicate data, but merely sending traffic to create congestion (a denial of service attack). It may not be concerned whether all its traffic can get through the re-ECN dropper. Therefore, it may understate the fraction of Positive markings on its path to avoid being throttled severely by an ingress policer. Therefore, if a forwarding element preferentially drops non-positive packets during severe congestion, it will tend to bias its service away from such ill-behaved sources.

Note that Table 9.1 proposes that Cautious packets should be treated to the lowest drop, along with Positive packets. This reflects their equal worth, byte-for-byte. It would be wrong to give either more preference than the other, perhaps under the mistaken impression that Cautious packets start new flows and the system should not allow in new flows when under stress. If such a policy were adopted, an attacker with sufficient fire-power might use Positive packets to block new flows from the system. Likewise *vice versa*.

Similarly, there is no reason not to treat all other re-ECN packets with equal drop preference. Negative, Neutral, Cancelled and CU[2] packets should all be treated equally. There are no grounds, for instance, for dropping Negative packets more, just because they have already experienced congestion upstream.

And finally, there are no grounds for dropping Not-ECT packets before ECT(0). Networks have no known practical way to limit the load of either any more than the other, if they are not responding to congestion.

Note that this discard behaviour is not applicable whenever a forwarding element can mark the

[1] If it were not for legacy concerns Cancelled packets could be dropped last with the same preference as Positive and Cautious. But the Cancelled codepoint overloads the congestion experienced (CE) codepoint used for legacy ECN. An operator SHOULD configure a forwarding element to treat Cancelled packets to the same drop preference as Cautious and Positive if it is certain all arriving legacy CE traffic will have been tightly rate-limited.

[2] See §12.2.2 on Forward Compatibility.

arriving workload of ECN-capable packets without any need for drop. Preferential drop of packets that would normally be ECN marked would only be relevant once the AQM algorithm was beyond its congestion avoidance operating range, where it had to drop something.[3]

Implementing and deploying preferential drop based on re-ECN markings is optional, as such forwarding elements can interwork with other routers that do not implement preferential dropping. However, forwarding elements that do not implement re-ECN-based preferential drop will simply not protect themselves (and other elements downstream) so well from DoS attacks. In addition, network operators would be strongly advised only to deploy preferential drop based on re-ECN markings where they were sure that all routes towards the queue in question were covered by a re-ECN policing function.

These preferential drop semantics are fully specified in the re-ECN Internet Draft [BJMS09a, §5.3].

9.2 Congestion Marking Cautious Packets

Although re-ECN works with unchanged forwarding elements, the Cautious marking uses a codepoint in the IP header that won't be congestion marked by existing forwarding elements, but it would be better if it was marked than dropped. Therefore, forwarding elements could be upgraded to recognise the Cautious marking as ECN-capable, and mark it as they mark other ECN-capable codepoints.[4]

The re-ECN wire protocol is arranged so that, if a forwarding element congestion marks the ECN field of a Cautious packet to `11` it will become Negative. Re-ECN transports are designed to understand what to do with such packets. Non-re-ECN transports would never send a packet with the Cautious codepoint (unless they operated some proprietary or Byzantine protocol).

§7.4.2 discusses how other system elements are expected to handle Cautious packets that have been congestion marked to Negative. It also discusses a second overloaded meaning given to Cautious packets marked Negative (see also §10.1 next); if they carry a flow-state setup message in their payload a Negative marking can also mean 'Flow-state not stored'.[5]

[3] A few days before this dissertation was due, an apparently serious flaw was noticed in the RED algorithm—at least in the algorithm that research papers discuss. The flaw concerns RED's drop behaviour rather than its marking behaviour. The outcome is that RED strongly favours unresponsive traffic if it forms a large proportion of arriving load. Therefore, this RED problem does not directly affect the re-ECN protocol as a whole. But it does strongly impact on how to modify an AQM to do preferential drop based on re-ECN markings. A quick check of open source code revealed that at least one implementer had noticed the problem and tried to work round it. Rather than try to include analysis of a hurried correction to RED in this dissertation, this section has been modified to abstract it away from reliance on RED as it stands. A summary of the discovered problem with RED is also included in Appendix C.

[4] Caution is advised on this statement. We have not decided on a good marking strategy, and if forwarding elements are being changed, changes to their AQM could also be considered. The recent realisation that the RED protocol's drop behaviour under extreme load is extremely wrong (Appendix C) warns that RED's ECN marking behaviour under extreme load might be improved too. Indeed, re-ECN would work much better under a DoS attack if RED did not mark 100% of Cautious packets. Otherwise all new flows would be starved by the dropper. If the Preferential Drop proposal of §9.1 were implemented at the same time as Cautious marking (which would make sense), then it is feasible that Positive and Cautious markings could be marked less than 100% while other packets were being dropped—given normal congestion avoidance would clearly be considered to have broken down.

[5] Indeed, this wouldn't even violate the first meaning of this protocol transition, as a forwarding element that never stores flow-state can always truthfully say 'Flow-state not stored'.

Chapter 10

Re-ECN Middlebox Behaviour

10.1 Flow-State Congestion Signalling

In §7.3 we introduced the need for the re-ECN dropper to be flow-aware (though not otherwise aware of the transport). Once a design includes flow state on a middlebox, it also has to be able to manage even innocent situations where provisioned memory becomes insufficient—a further consequence of violating the shared fate principle.

Memory exhaustion is, of course, congestion of a physical resource, so we want to be able to treat it in a similar way to congestion of network capacity. Rather than just discarding flow-start requests when memory is exhausted, an explicit signal would give a timely unambiguous indication of memory congestion. The first packet of a flow is particularly vulnerable to drop for the other reasons than congestion listed in §6.1.1, so it is doubly important to distinguish congestion signalling explicitly.

Therefore, we propose to use the re-ECN protocol to signal congestion of flow-state. Rather than confine this facility to the re-ECN dropper, there is no reason not to allow any middlebox (e.g. a network address translator) to use this facility.

If we put flow-state congestion signals in the IP header as a transport-independent mechanism, we will need to distinguish flow-state congestion from congestion of bit-capacity so that sources can respond accordingly by damping flow arrivals (rather than bit arrivals). In the re-ECN protocol, we use the transition from Cautious to Negative as a signal of flow-state memory congestion. This overloads the use of this transition for normal bandwidth congestion. The two can be distinguished by a transport, because the flow-state-related meaning only applies if the payload of the packet contains a request to store flow-state (e.g. a TCP SYN or the equivalent for other transports, whether explicit or implicit).

This proposal deliberately uses the same structure of in-band congestion signalling as ECN. That is, it provides a way for a middlebox to ask the intended receiver of a packet to issue a response to the sender saying 'flow state not stored'.[1] Even though this seems more convoluted than sending a special flow rejection signal directly back to the sender, it is much more robust for all the reasons given in §6.1.1.

Well-designed transports SHOULD provide an explicit application-independent way to reject a request to initialise flow state. A combination of TCP flags for such a flow rejection response has been

[1] The remote possibility of a flow initialisation request spanning more than one packet might have to be considered for some transports.

reserved in the specification of the TCP transport over re-ECN [BJMS09a].[2] A server could return such a rejection response for a number of reasons:

- exhaustion of its own flow-handling resources;
- exhaustion of the flow-handling resources of a middlebox on the path;
- unwillingness to allocate flow-state without more evidence that the client is genuine.

Re-ECN's in-band congestion signalling deliberately doesn't preclude a middlebox echoing a 'flow-state not stored' response on behalf of the intended receiver. Indeed, a middlebox might be deployed to spare a server the routine task of echoing challenges. For instance, the rejection response could be sent with a nonce, crypto-puzzle, TCP SYN cookie[3] or something similar [Edd07].

Note the response means 'flow state not stored,' which is subtly different from 'flow-start request rejected.' This is particularly helpful in the case of SYN cookies. SYN cookies were cleverly devised to work with unaltered clients. Unfortunately, this means that clients cannot tell whether the server has sent them a SYN cookie. So receiving a SYN cookie can lead the client to think the server has taken everything it asked for into account, when in fact it just blindly returned a SYN cookie. A response to a re-ECN client saying 'flow state not stored' would tell the client 'I've stored no more flow state than you see in this response.' Then, if the client had requested some new or esoteric TCP option, it would know the server had ignored that aspect of the request and be able to ask again.

Finally, note that re-ECN's in-band congestion signalling deliberately doesn't preclude multiple middleboxes on the path (e.g. one at the sender edge and another at the receiver edge), all possibly trying to tell the receiver how it should respond. Any one of a sequence of middleboxes can ask for a 'flow state not stored' response by switching the re-ECN field to Negative. This is why we do not advise that a server or middlebox discards non-Cautious packets just because they do not match an existing flow ID, although a machine under stress is entitled to discard them preferentially (see §12.1.3 on initial packet attacks).

Signalling flow-state congestion is a tentative part of the re-ECN protocol. Most of its many ramifications have been considered, but there are a few outstanding questions. For instance: i) should the signal mean 'Flow state stored but approaching exhaustion' or 'Flow state not stored'? ii) Should memory congestion be signalled to packets holding open existing state, or only to packets asking to allocate new state? At present, the design described above takes the latter answer in each case.

[2] The specification also allows for the possibility that the server is a legacy one that doesn't understand a congestion marking. If the response from the server shows it didn't understand the request to use ECN or re-ECN, a re-ECN client MUST follow the congestion control behaviour it would have if the first packet had been lost.

[3] SYN cookies are "particular choices of initial TCP sequence numbers by TCP servers". The server calculates the initial sequence number from information it will be able to reconstruct later from a valid acknowledgement. It then does not need to hold any connection state until it receives an acknowledgement with a sequence number one greater than the server would have created itself for what would have been the previous packet in the connection.

Chapter 11

Re-ECN Bulk Congestion Policer

11.1 Bulk Congestion Policer Model

Our high-level aim is to show that users of an internetwork can be made to pay the cost of the congestion their traffic causes to others, but without having to suffer the unpredictability of a variable usage charge. We want network operators to be able to use rationing not just pricing [Bar89, Odl97, §5] to ensure their customer's incentives to manage load can be aligned with the interests of all.

We set the constraint that the customer pays a flat monthly fee to its ISP. This is purely a commercial acceptability constraint. This funds a constant contracted rate, w_C of congestion tokens filling a token bucket[1]. Of course, ISPs may give customers the choice between different values of w_C. Unlike with a classic token bucket, only bytes marked for downstream congestion consume tokens. So tokens are not consumed based on the amount of bytes sent, but on the volume of congestion the source expects the traffic to cause. Therefore, if the customer sends flows indexed j at bit rate $x_1, x_2, \ldots x_j, \ldots$ each with an expectation of rest-of-path congestion of respectively $v_1, v_2, \ldots v_j, \ldots$, tokens will be consumed

[1] A token bucket is a model that represents a stored number f of tokens (modelled as the bucket fill level), a means for subtracting tokens and an algorithm to add tokens at a continuous rate. Usually this is implemented by storing the time, t_{r-1} when a token was last removed. Then when the next request to remove tokens arrives at time t_r (i.e. the next marked packet size s in our case) $f_r = f_{r-1} + w_C(t_r - t_{r-1}) - s$.

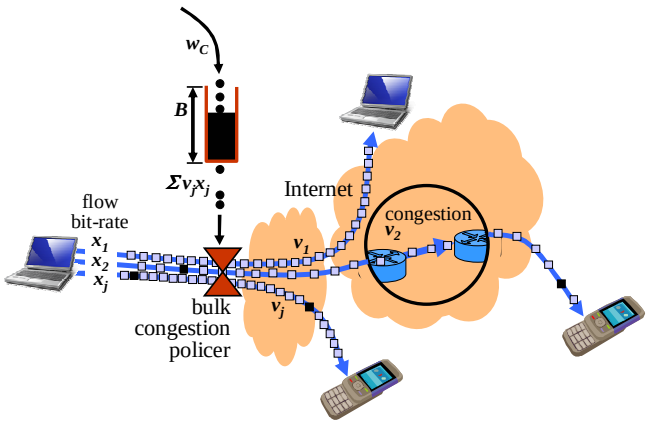


Figure 11.1: Bulk Congestion Policer in Context.

at rate $\sum v_j x_j$ (Fig 11.1). In contrast a classic token bucket would empty at rate $\sum x_j$. Consuming tokens against downstream congestion ensures a customer continually suffers the congestion cost of its behaviour on others, while only having to pay a fixed fee for a fixed token fill-rate.

So long as the customer's usage stays below the congestion allowance, the policer merely monitors the congestion bit rate passively. but whenever sustained excessive congestion bit-rate empties the bucket, the policer may penalise the whole aggregate of customer traffic entering the internetwork.

The depth B of the bucket absorbs bursts of congestion, allowing for fluctuations in network conditions and in the needs of customers and their applications. The depth may be finite or effectively infinite. If bucket depth is finite and if tokens are consumed too slowly, the stream of filling tokens may overflow and be wasted. Operators would most likely not just limit the token fill-rate but also peak token consumption using a 'dual token bucket'. That is, to consume tokens a packet must satisfy two token bucket constraints at once, one with fill-rate w_C and depth B, the other with fill-rate $\widehat{w}_C \gg w_C$ but depth of just one MTU (the maximum transmission unit of the link). With the second token bucket, there would be less need to limit the first bucket's depth. To be concrete for this dissertation we assume $B \to \infty$ and that the operator limits peak token consumption to $\widehat{w}_C$.

Unlike a traditional traffic conditioner based on bit-rate, congestion bit-rate has largely the same value and meaning however much it is shifted in time or space, therefore the bucket can potentially absorb bursts over periods of hours or weeks. The depth B of the bucket is more a commercial policy issue, effectively limiting how much unused congestion-volume quota the ISP allows the customer to 'roll-over'. However, the depth of all the other customers' buckets on the Internet does determine the predictability of any single customer's service experience.

The peak token consumption rate $\widehat{w}_C$ primarily limits anomalous behaviour, whether malicious (malware), accidental (buggy applications) or deliberate (highly inconsiderate use - e.g. scripted execution of vast numbers of flow instances).

The particular design of congestion policer we propose here is not intended to be generic to all ISPs. It is a design of policer that satisfies the flat monthly fee constraint of a particular commercial approach, while still aligning incentives to respond to congestion. It was the primary aim of this research to be able to offer flat-fee Internet access *and* align incentives, and it is an important contribution if re-ECN has enabled such a simple policer design to achieve this aim.

But we have another reason for proposing this particular design in this dissertation. We need to establish whether re-ECN can assure the integrity of congestion signals *even when they are used* (see §12.3.1). We have proposed this particular policer design as one that we believe gives the maximum freedom to a customer and to developers of new application and transport behaviours, but within the constraints of a flat fee and incentive alignment.

11.2 Policer Diversity

Re-ECN could be used by ISPs to give customers *more* freedom. Re-ECN could be used for sender-pays congestion charging, much as Kelly suggested ECN could be used for receiver-pays congestion charging [Kel97b]. And customers who prefer flat fees could discipline themselves to ensure their congestion

charges regularly result in the same usage fee each month.

But we propose this congestion policer because we want to define a concrete artefact that ISPs could use to offer *and enforce* a moving congestion limit. This approach allows network operators to impose limits when incentives are not enough; for instance in times of crisis, or against customers whose machines have become infected with malware. It is likely to be more commercially viable than dynamic congestion charging, which seems to offer *too much* freedom to be palatable to both providers and customers alike.

It should be emphasised that this policer is an *application* of the congestion signal integrity that re-ECN aims to provide. An architectural aim of re-ECN is to encourage diversity of policer designs to meet different commercial objectives; to encourage 'Tussle in Cyberspace' [CWSB05]. For instance, we have proposed other design examples ourselves:

- In a workshop paper co-authored with Jacquet and Moncaster [JBS05], we proposed a similar bulk token bucket congestion policer to the one described here, but one which discards all flows equally when the bucket empties. That proposal aims to give a customer the incentive to self-police each flow (whether elastic or inelastic) without the policer having to. The discards when the token bucket is empty effectively add a cross-flow element to the apparent congestion that each flow experiences. Customers who fail to self-police bring this cost on themselves, but it also allows a customer to choose to allow inelastic flows to complete even if their cost to the customer's other activities increases after they have started;

- The Internet Draft Motivating re-ECN [BJMS09b, Appx B2.], also co-authored with others, proposes a per-flow policer that aims to force each flow to respond to congestion as TCP or a TCP-friendly [FHPW00] transport should;

- Network operators might deploy 'service-oriented' [WK08] congestion policers that specialise a virtual network to a certain service or services;

- Network operators might enforce a 'walled-garden' around their networks to limit customers to only a certain set of applications identified by their rate response to congestion signals in combination with other information perhaps gathered using deep packet inspection (DPI) technology to inspect the payloads of packets.

We hope and expect congestion policers will become an important policy enforcement point (PEP) at the ingress to the internetwork, not just for flow admission control [YPG00], but to control traffic admission at any granularity from packet [Cla95] to service. Re-ECN should allow some networks to adopt liberal policies on what traffic they allow (as the policer proposed here would do), while simultaneously other networks can enforce conservative policies, without all having to lose the benefits of full interconnection. Networks with conservative policies will still be able to protect their services and their investments using the border mechanisms of §8. And any network will be able to decide to change its whole culture from liberal to conservative, merely by altering its policing policy.

Effectively, re-ECN turns the end-to-end principle with respect to transport control into an optional policy rather than a design principle embedded deep in the architecture. And this policy is applied once per customer, at the outer edge of the internetwork. Thus it is amenable to change through the simple change of an individual bipartisan contract, without depending on changes to any contracts with operators in the rest of the internetwork.

Different token bucket fill-rates (w_C) effectively provide a differentiated spectrum of network services without explicitly using differentiated scheduling (Diffserv [BBC+98]) (for a technology advantage of this approach see 'Bufferless Border Control' in §8.1.3). But downstream congestion information complements and improves Diffserv too. New forms of traffic conditioning agreement (TCA) for any or all classes of service could be based on downstream congestion.

Currently, if a Diffserv TCA polices traffic without regard to destination, each network link has to be provisioned statistically based on the expected variation of the whole traffic matrix. The advantage of enforcing a TCA against downstream congestion information would be to manage network load even if the traffic matrix was unusual. This would also allow cheaper, less generous provisioning by leveraging most customers' abilities to respond elastically to temporary congestion most of the time.

Downstream congestion has another potential advantage over straight bit-rate for policing. TCAs based on bit-rate are only useful at high levels of aggregation—for larger sites comprising many sources of load. Bit-rate TCAs are hard if not impossible to apply for small sites characterised by long periods of complete inactivity and occasional intense flurries—network provisioning would either be prohibitively expensive, or the assurances that could be offered would be worthlessly poor. However, TCAs based on downstream congestion integrate properly over long periods of time. Thus congestion-based TCAs are potentially useful for consumer as well as business customers.

We have deliberately used the term 'customer' rather than 'end-user' throughout this discussion to denote a 'locally attached contractual entity'. The customer relationship need not be commercial although it often will be; for instance it may be a relationship between an academic institution and its internal network operations department. The term customer includes customers with large numbers of users as well as sole-user customers. A customer could even operate a commercial network themselves, in which case this policer would effectively be a border mechanism that is actively intervening in the packet stream (§8.2.8), in contrast to the passive border mechanisms recommended in the rest of §8.

Finally, we propose this bulk congestion policer design because it would be extremely cheap to implement. Therefore it could be suitable for distribution to low-cost access equipment at every attachment point of an access network, e.g. on the input ports of a digital subscriber line access multiplexers (DSLAM), hybrid-fibre-coax node (HFC-node) or cellular base-station (Node B in UMTS). It is usually more cost-efficient to centralise the implementation of policy controls because the flexibility they need tends to lead to complexity. But it is preferable for a policing function to be cheap enough to distribute right to the network edge, in order to physically block unruly incoming traffic from causing severe congestion.

11.3 Bulk Congestion Policer Design

The bulk congestion policer proposed here (from here on just called 'the policer') puts no individual constraints on the transport behaviours of different flows, other than limiting total long running expected downstream congestion for all flows as a whole.

Like the egress dropper (§7), and unlike the bulk congestion policer we proposed in [JBS05], if the aggregate bucket has emptied, each flow is sanctioned distinctly. Thus flows expecting no congestion on their path can continue unaffected by the policer's lack of tokens, while flows causing considerable congestion will be hit the hardest. The policer effectively consists of two modules, a bulk token bucket meter for all flows called on by per-flow instances of droppers that re-use the principles, design and implementation of the dropper we have already described for the egress.

The meter module uses the algorithm in §8.2.7 based on Eqn (6.4) to normalise any poisoned cancelled markings in arriving packets. This produces a 'virtual packet marking' of the moving average of recent downstream congestion, v, which determines how many tokens each packet consumes. If the network(s) between the source and the policer are uncongested, this will be equivalent to counting just packets marked Positive. But if there has already been some congestion (e.g. in the customer's own network, or into the line from the customer to their provider), the policer removes it from its count by calculating downstream congestion v.

How Much Drop? We now consider how much drop is appropriate if there are insufficient tokens in the bucket to 'pay' for the markings of expected downstream congestion. Rather than just drop everything as suggested in [JBS05, BJMS09b, Appx B1.], conceptually we follow the principle of 'Proportionate Sanctions' (§7.3).

The dropper module can assume that every incoming Positive mark was triggered by an earlier Negative mark that it would have been able to count if it were at the egress. So it maintains a per-flow moving average of Negative marks counting real Negative marks as well as Positive and Cancelled marks (but not Cautious) as if they *also* represent Negative markings.

If the meter module (the bucket) is empty, we propose that the dropper module should change an arriving Positive marking to Neutral. We call this 'covert' marking' because we do not want it to be seen as a congestion mark by the transport (see below).

The rationale for this behaviour is as follows. An arriving Positive marking can be likened to the customer saying to the network, "Charge this to my account." When the bucket is not empty the network operator can forward the Positive mark saying in turn to the next network's border meter, "Charge this to my account." But if the original customer's account is empty, the first network says "Your account is empty; I refuse to commit to paying the next network if you haven't paid me enough.[2]" At this stage no packets have been dropped—the dropper is still measuring what to do.

Even if the bucket is empty when one Positive marking arrives, it may be refilling sufficiently quickly for some Positive markings to be allowed through unchanged. The policing module therefore maintains a moving average of Positive markings, not counting any it has changed to Neutral.

[2] Different policies may allow certain customers some credit, or course.

The policer having removed some Positive markings, the flow will be understating its expectation of downstream congestion, v. The policer then determines drop as a proportionate sanction in exactly the same way as the egress dropper of §7 does.[3] From Eqn (7.3) the drop probability for Negative and Neutral[4] marked packets is $\pi_u = \pi_y = 1 - z/u; (\text{if } z < u \,\&\, V < 0)$, while packets with other markings (Positive, Cautious and Cancelled) are not dropped at all. Packets marked with legacy markings (`Not-ECT` and `ECT(0)`) are likely to be rate-limited separately as we have already said.

11.3.1 Covert Marking as Policer Signals

This is a tentative extension to the re-ECN wire protocol that is not yet documented in the formal re-ECN protocol specification, and may be changed.

It may have been noted that we have chosen to remove Positive markings by re-marking them to Neutral, which is not a valid re-ECN protocol transition (§6.1.2). All the valid transitions in the re-ECN wire protocol only require the network to alter the ECN field, not the RE flag. The wire protocol has been designed so that end systems can double-check that the RE flag is unchanged, to give some assurance that the network has not cheated by making a protocol transition that it should not normally make (§12.1.4).

The policer deliberately uses this 'illegal' transition as a way for it to signal to the transport that the drops being experienced are most likely due to policing, not congestion. This keeps to our design principle of using in-band not backwards signalling (§6.1.1). Assuming some covertly marked packets are not dropped by the policer,[5] the end-to-end transport can detect a change to the RE flag on packets it originally sent as Positive. It can then respond to drops differently, suspecting they are due to policing.

The precise response to policer-induced drops is up to each transport. Each will probably still respond to these policer drops by reducing its rate, but it would be advised not to respond to each drop with a Positive marking in the next round. This would otherwise drive it deeper into debt with the token bucket.

The approach of each transport to being policed is out of scope of this dissertation. Also out of scope are the details of whether the token bucket toggles to covert marking in one step, or as a gentle ramp as the bucket empties. This dissertation only needs to say enough about the policer to give:

- an existence proof of flat-charging with aligned incentives;
- a concrete instance of the most liberal policer we believe is possible, so that we can model it in §12.3.1.

[3] And, of course, if real Negative markings arrive with insufficient Positive markings to cover even them, then an even higher level of drop will automatically ensue.

[4] Neutral also implies Currently Unused (CU see §12.2.2).

[5] Because the policer re-uses the algorithms of the regular dropper from §7.6 it gives one packet's grace to covert markings, tending to drop subsequent packets instead.

Chapter 12

The Re-ECN System

12.1 System Attacks on Congestion Signal Integrity

The rudimentary goal of re-ECN is to ensure that the integrity of congestion signalling information in re-ECN packets can be trusted to be accurate. This section aims to quantify or at least estimate the limits under which this property of re-ECN will still hold under various attacks.

12.1.1 Endpoints Against Networks

FEC Trade-Off Attack

Salvatori [Sal05] describes this attack against the re-ECN system, which involves the source playing off the egress dropper against the ingress policer. A similar attack was also mentioned independently the following year in [BFB06]. The source inserts erasure codes [BLMR98] (forward error correction or FEC)[1] into the data stream which inflates the bit-rate, but allows the original data to be reconstructed if a proportion is lost. The source then understates downstream path congestion knowing the dropper will discard some data, but this also consumes less tokens at the ingress policer so it can go faster for the same 'charge'. We call this the 'FEC trade-off' attack.

The FEC trade-off attack was aimed at an earlier variant of egress dropper design, which was consequently updated to that in §7. We will now prove that the updated design prevents the FEC trade-off attack from giving a source-destination pair any gain from understating expected downstream congestion at the network ingress relative to congestion marking emerging at the egress. We will first define the scenario and its notation.

FEC added by the source inflates the data rate to Φ times its original bit-rate, where $\Phi \geq 1$. We assume the FEC source does not re-transmit any repairs of losses, as this would cost even more on top of the FEC overhead. We compare cases where FEC is added and where it isn't. Variables are subscripted with f in the case where FEC is added, and not when it isn't.

As previously defined, the recent fractions of Positive and Negative re-ECN marked bytes are z and u_i at node index i along the network path (as the dropper does not drop Positive packets z remains unchanged along the path). Index $i = 0$ represents the source before FEC is added, $i = 1$ represents arrival at the ingress policer, which is also equivalent to the point of departure from the source after

[1] Note, not error-correcting codes, which are different.

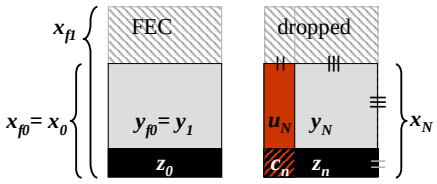


Figure 12.1: The Futility of the FEC Trade-Off Attack.

any FEC is added. $i = n$ on arrival at the egress dropper and $i = N$ on departure for delivery to the destination. To simplify matters, but without loss of generality, we will assume that $u_1 = 0$ at the ingress policer, that is, there is no congestion marking within the source's own network.

Theorem 12.1. *An end-to-end flow with useful bit-rate x_0 and free choice in declaring expected downstream congestion z cannot gain by sending bit-rate x_1 which includes added redundant FEC data, as long as the network makes the user accountable for cost zx_1 at the ingress and discards non-Positive packets with probability $\pi_d = 1 - z/u_n$ $(z < u_n)$ at the egress.*

Proof. Whether in the FEC case or not, we assume no drop of the ECN-capable packets between ingress and egress, that is $x_1 = x_n$. We also note that all packet marking fractions and bit-rates cannot physically be less than zero.

In the base case with no FEC, the source ensures $z = u_n$ so that there is no drop at the dropper. Therefore $x_N = x_n = x_1 = x_0$.

In the FEC case, as shown in Fig 12.1, we split the data flow into two parts: i) a bit-rate equivalent to the base data the source wishes to send $x_{f0} = x_0$ and ii) the added FEC $x_{f1} - x_0$. To ensure bit-rate x_0 gets through the dropper with no loss, a Positive bit-rate of zx_0 needs to be sent where $z = u_n$, given the marking fraction introduced along the path is u_n. Thus the base data can be considered self-contained with just enough Positive bit-rate to sustain its delivery.

Considering the additional FEC data as a separate flow, if none of it is marked Positive, it will all be dropped. It would be pointless to mark any of the additional FEC data as Positive just in order to deliver additional data that isn't needed, as this would cost more than if FEC had not been used at all. This proves that a drop probability of $\pi_d = 1 - z/u_n$ $(z < u_n)$ is sufficient to ensure no gain can be made from the FEC trade-off attack.

Put another way, any additional Neutral bytes attract proportionately more Negative markings. So, unless the source marks more packets Positive, the dropper will discard all the additional Neutral packets sent. □

Salvatori's FEC trade-off attack originally played off the ingress policer against the egress dropper (hence its inclusion in this section on attacks against the whole system). However, the above argument proves that the new egress dropper design alone is sufficient to negate any gain from the attack.

For those who would prefer a less terse proof, the following may help.[2]

[2] Arnaud Jacquet suggested this second form of proof in place of my earlier erroneous proof.

Proof. Consider that the source sends a bit-rate of zx_0 Positive packets, which will be forwarded by the dropper with no loss. If the source uses FEC data to inflate the remaining bit-rate by Φ, then the bit-rate of Neutral packets leaving the source will be $\Phi(1-z)x_0$.

Once proportion u of these Positive and Negative packets have been marked, the bit-rates of Positive and Negative packets will respectively be $(1-u)zx_0$ and $u\Phi(1-z)x_0$. Therefore the drop fraction of non-Positive packets will be

$$\pi_d = 1 - \frac{(1-u)zx_0}{u\Phi(1-z)x_0} \tag{12.1}$$

Adding the unscathed bit-rate of originally Positive packets zx_0 to the bit-rate of originally non-Positive packets forwarded by the dropper, the overall delivered bit-rate through the dropper will be

$$\begin{aligned} x_N &= zx_0 + (1-\pi_d)\Phi(1-z)x_0 \\ &= \frac{z}{u}x_0. \end{aligned} \tag{12.2}$$

This is clearly independent of the FEC inflation factor Φ, proving again that addition of FEC without any Positive markings can have no effect on delivered throughput. □

The source always has to conservatively estimate the amount of FEC to add because it doesn't know in advance the rate of congestion marking, which the dropper algorithm uses to determine the drop rate. Therefore the FEC trade-off attack will actually always result in worse goodput than not using FEC at all, aside from any additional overhead or inefficiency in the FEC scheme.

It is interesting to note that Raghaven & Snoeren's 'Decongestion Control' [RS06] also falls foul of the same unpredictability problem with FEC. Decongestion control involves everyone always sending at maximum line rate and everyone varies the redundancy of their erasure codes, in place of the rate response of traditional congestion controls. Decongestion and congestion control would be equivalent were it not for the extra redundant data needed to cater for the unpredictability of loss. This has another interesting analogy with the per-flow credit that the re-ECN dropper requires due to the 'Source Responsibility for Delay Allowance' principle. In place of decongestion control's redundant overhead packets, re-ECN unnecessarily consumes downstream congestion quota. Both redundancies can only be identified in hindsight.

Strategic Confusion of Investment Signals

Bauer and Faratin [Bau05][3] claim to identify a strategising attack against congestion charging in general, that they term the 'Capacity Expansion Game'. They argued that congestion-based incentive mechanisms are only myopic incentive-compatible, not long-term incentive-compatible. Although this attack is not specifically against re-ECN[4] we deal with it here, as re-ECN depends in turn on accountability for congestion in order to align incentives. We only argue discursively about this attack, as it is exceedingly hard to pose a realistic scenario in which it might work and that is also tractable.

[3] There is no formal reference to cite for the 'Capacity Expansion Game' aspect of the authors' work other than this slide presentation.

[4] Their specific attack against re-ECN is dealt with elsewhere.

In the capacity expansion game, a strategising player predicts that it will have higher demand in the future. It sends a small amount of dummy traffic when it has no demand, which it properly pays for. It aims to cause other players who do have demand at those times to pay more in congestion charges than they would have otherwise. It banks on this collective behaviour eventually sending an investment signal to the operator to invest in more capacity, using the funds mostly provided by customers other than the strategising player. After the investment in more capacity, the strategising player can satisfy its own higher demand as predicted, but at a lower congestion charge. A simulation shows the strategy works for heavy users against light users, even if many players adopt it.

This game is effectively a strategising attack not just on congestion charging but on microeconomics itself. It is equivalent to gaming a vendor of apples (or any perishable good). Strategising players who know they will want more apples in the future can reduce the future price of apples by buying a few more than they need now, thus increasing supply and reducing the price for the future.

It is certainly true that this game can succeed sometimes. For instance the authors pointed to certain players like Content Distribution Networks that might be in a strategic position to sufficiently predict future demand of themselves and others.[5] But this strategy can also fail. It is possible that there could be situations where a player finds itself in a position it can take advantage of. But the strategy is only a fundamental problem if it can be systematically exploited.

The probability of success depends on:

- the relative elasticities of demand (the primal rate response to congestion marking) and of price (the dual marking response to congestion);
- whether congestion charges are likely to be higher or lower in the future.

On the demand side, if one user tries to increase the price by increasing demand, other users will reduce their demand, thus reducing the price (this was not included in Bauer & Faratin's model). Therefore, strategising players might have to spend a lot to get the market as a whole to spend enough extra to trigger significantly earlier investment.

On the supply side, in a growing market, congestion charges are likely to be lower in the future anyway if there is an economy of scale (the marginal cost of capacity reduces as more capacity is installed). The game assumed that capacity expansion would occur in steps leading to a temporary uncharacteristic drop in price. However, it would be as likely that network operators would mark based on gradually increasing logical limits, rather than actual physical capacity constraints. One can think of this as a counter-strategy against the game by network operators.

Anderson *et al* [AKS06] suggests another potential operator counter-strategy. The paper recognises that pure congestion charging alone can be exploited by strategising players at the expense of myopic ones. It proposes that network operators could offer fixed price bandwidth contracts, but allow buyers freedom to use more or less bandwidth in the resource pool. The operator applies balancing charges (positive or negative) at the end of each contracting round so that its total revenue remains fixed, but

[5] We should add that players using significant market power to distort market signals are generally expected to be handled by market regulators, not the market mechanism.

customer charges end up proportionate to the amount of congestion they caused. The paper proves that each customer's incentives are still aligned with the interests of others, but they also have no incentive not to be honest in their estimation of future bandwidth demand.

Returning to Bauer & Faratin's contribution, in general it is unlikely that any player will have sufficiently perfect information to be able to predict a network's capacity expansion plans, which depend on competition, equipment cost movements, internal company politics and performance, growth in demand, variations in the cost of borrowing and so forth.

Similarly, strategising players would have to predict the future behaviour of other players, to predict overall future demand. The strategy was shown to succeed for the set of heavy users against the set of light users, each acting independently. But it conveniently didn't allow users to switch providers, while it expected the provider to invest in capacity as if it were under the competitive threat of its customers switching to an alternative supplier. If some light users became fed up with the generally high level of congestion being caused by the strategising players, they might leave for another network, removing the need for the provider to invest in capacity. Then the strategising players would lose all they had invested in the game without getting the provider to expand capacity before their demand increased as they knew it would, causing them further congestion costs.

We have not, and cannot, prove whether the capacity expansion game could be systematically successful. If it were successful in some scenarios, it would imply that the integrity of re-ECN pricing signals can be confused by a strategising player. But we hope that we have cast sufficient doubt to place it in the background, as a possible opportunistic attack rather than a systematic flaw requiring a systematic solution.

Bandwidth Flooding Attacks

The re-ECN protocol introduces the possibility of some protection against bandwidth flooding attacks. No hard claims can be made about re-ECN's ability to mitigate DDoS, but it does have the potential to considerably raise the bar against attacks and to transfer liability for the cost of attacks back towards the source, providing information and incentives to deal with them at source.

Considerable attention has been paid to DDoS issues since the brief explanation of re-feedback's DDoS mitigation capabilities in the original re-feedback paper [BJCG+05, §3.5], including publication of a workshop paper on the issue [Bri06], the work in §8.2 on networks attacking each other with dummy traffic and the work below.

Bauer and Faratin [BFB06] published a dummy traffic attack on re-feedback that they mentioned could be used to deny service. However, their primary target was to show that strategic users could increasing congestion marking for others. This class of attacks is dealt with in §8.2

Re-ECN's defence against bandwidth flooding relies on the flooded forwarding element(s) implementing the optional preferential drop scheme recommended in §9.1. Then, whenever a queue is highly congested to the point it is dropping packets, it will preferentially enqueue arriving Cautious and Positive packets. This protects the forwarding service at the very first step of the queuing process. Therefore, any packets not marked positively by an attacker would contribute nothing to the force of an attack.

This is because a highly congested queue will be marking most packets that it does serve as either Negative or Cancelled. So a well-behaved re-ECN source would have to respond to the resulting continuous congestion feedback by sending all packets as Positive (after the initial Cautious packets).[6]

We will now derive formulae for the force of such an attack, then roughly quantify the mitigating effect of the re-ECN system.

Initially, we will assume the case of a malicious source sitting behind the most liberal re-ECN congestion policer we believe will be deployed (§11). The policer limits the source's congestion-bit-rate to a contracted peak of $\widehat{w}_C$ and to a contracted fill rate $w_C \ll \widehat{w}_C$ once any built-up congestion allowance has been used up.

As we have shown that only positively marked packets can carry any attack force, the contracted *congestion*-bit-rate limits essentially act as limits on the *total* bit-rate x_A that an attack source can contribute to a bandwidth flooding attack, that is

$$x_A \leq \widehat{w}_C$$

and once any stored allowance is used up

$$x_A \leq w_C.$$

A typical contracted congestion-bit-rate w_C will allow for path congestion conditions averaged over a month $\bar{p}$; a reasonable activity factor over the month υ; and a reasonable expected proportion η of peak access bit-rate X when active, such that

$$w_C = \bar{p}\upsilon\eta X. \tag{12.3}$$

Typically, a site's contracted peak congestion-bit-rate would prevent it causing unreasonably high congestion levels $\widehat{p}$, even at its full access bit-rate.

$$\widehat{w}_C = \widehat{p}X. \tag{12.4}$$

And, if a site were allowed to store up B unused congestion allowance (in units of bucket fill-time), it could sustain this peak congestion-bit-rate for a maximum time (from full bucket to empty) of

$$\begin{aligned} \widehat{T}_C &= \frac{w_C}{\widehat{w}_C} B \\ &= \frac{\bar{p}}{\widehat{p}} \upsilon\eta B. \end{aligned} \tag{12.5}$$

For a typical residential customer, the following 'ball-park' values for these parameters seem reasonable:

$\bar{p}$ average congestion: from 0% to 3% at worst, with perhaps 0.2% a reasonable average [CC08];

υ activity factor: perhaps 20%;

[6] Assuming there is a re-ECN dropper function after the congested link's marking process, sending any Neutral packets would risk running into debt at the dropper, without easily being able to balance it with a Positive packet, because all Positive packets get cancelled by the congestion.

X peak access bit-rate: 100Mb/s;

η expected proportion of access rate attained when active: perhaps 70%;

$\widehat{p}$ maximum allowed congestion: 10%;

B stored unused congestion allowance: 6hrs fill-time.

Thus giving estimates of

	$x_A \leq \widehat{w}_C$	$\approx$ 10Mb/s
for duration	$\widehat{T}_C$	$\approx$ 1minute
and thereafter	$x_A \leq w_C$	$\approx$ 30kb/s.

Even 100Mb/s customers expecting 100% of their access rate all the time ($\eta = \upsilon = 100\%$) would only need to contract for a congestion policer fill rate $w_C \approx 200$kb/s. So if the scenario were otherwise the same, their token bucket would be able to sustain a peak congestion-bit-rate 10Mb/s attack for at most about 7 minutes before limiting the attack to 200kb/s.

Thus, a single attacker with a 100Mb/s access link would only be able to sustain a flooding attack in the tens or low hundreds of kb/s. Therefore, to saturate a 10Gb/s link, it would take a botnet of about 100,000 residential sites. Although botnets of this size are not uncommon, the take-away point is that, if congestion policing were prevalent, botnets would need to be about three orders of magnitude larger to achieve the same effect.

Of course, candidate hosts for a botnet can be selected from those not behind residential congestion policers. Various attack strategies that might be adopted are tabulated in Table 12.1 against a suitable remedy in each case.

Active Detection and Intervention. So far we have considered how the re-ECN incentive framework strongly damps flooding attacks merely as a fortunate side-effect of getting end-points to manage their regular every-day congestion. Nonetheless, networks can further exploit the congestion information re-ECN reveals to them. Flooding attacks appear as a stream of traffic towards a common destination address or prefix with marking approaching 100% downstream congestion (i.e. nearly all positively marked packets). Essentially, re-ECN information greatly amplifies the visibility of flooding attacks at source for attack detection systems to pick up, and it provides an incentive for the network at the source to do so.

This was the theme of the workshop paper by the present author "Using Self-Interest to Prevent Malice; Fixing the Denial of Service Flaw of the Internet" [Bri06]. Active detection of DoS attacks is outside the scope of this dissertation, but the general idea will be briefly summarised here.

The goal of re-ECN is to allow evolution of a wide range of innovative transport behaviours in response to congestion, while providing incentives that cajole all approaches towards a responsible, responsive middle ground—neither insensitive nor unnecessarily over-sensitive.

One could describe a flooding attack as an innovative response to congestion. (Using other people's congestion allowances is even more innovative :) But on the spectrum of responses to congestion, one

Strategy	*Remedy*
Attack sources could be selected in networks not policing congestion, and still programmed to send Cautious or Positive re-ECN packets.	Networks using re-ECN to limit congestion would deploy bulk congestion policers at their borders with any network not deploying re-ECN congestion policers.
Attack sources could send legacy packets without re-ECN markings.	On congested forwarding elements employing preferential drop based on re-ECN markings, legacy packets would have no attack force against re-ECN packets, as already discussed. Also, one would expect bulk congestion policers around a network to also limit the straight bit-rate of packets without re-ECN markings.
Attack sources on larger sites (e.g. enterprise networks or campuses) might sit behind a bulk congestion policer for the whole site that would less stringently limit the congestion-bit-rate they could cause.	But, of course, this would limit the force that could be amassed from a number of attack sources in one site. Indeed, one would expect a contracted peak congestion-bit-rate to be much lower for a congestion policer serving a large aggregate, because the variance in congestion-bit-rate over time should be greater for single users than aggregates.[7]

Table 12.1: Further DDoS Attack Strategies and Remedies.

would expect a wide expanse of clear water between even an inelastic transport and a DDoS flooding attack. The difference is only apparent if congestion is compared with bit-rate. An inelastic transport might temporarily exhibit no response to congestion when congestion is low (perhaps having used congestion-based flow admission control). But if both congestion and bit-rate are extremely high, a transport that is still unresponsive will be highly suspect.

If a host attached to network N_A is part of a 'zombie' army (controlled by a botnet master) causing a high volume of congestion in network N_B, the re-ECN border mechanisms (§8) will ensure the network N_A harbouring the source recompenses network N_B.

But this cost, in itself, is not a sufficient incentive to make network N_A want to act against the offending traffic, because the re-ECN incentive framework ensures that a source cannot cost a network more than it has paid. Even if the victim of the attack is in the same network (N_A) as the source, one could argue that network N_A is fully recompensed for the reduced network quality that its other customers experience, by the allowance it draws from its 'zombie-customer'.

However, network N_A does have a strong incentive to help its zombie-customer; either silently, by discarding the attack traffic before it uses up the customer's allowance; or more actively, by helping the customer remove the offending malware. The zombie-customer gets no utility from the attack traffic and effectively loses some access rate. So a network that continues to cash-in from such traffic will merely appear more expensive and lower quality than a competitor.

[7] Congestion is non-linear, so it would be unlikely to be normally distributed over time. If it were, its variance would scale $O(1/\sqrt{n})$ with n users by the Central Limit Theorem.

Effectively, the network has two customers in one: a white-market and a black-market customer. But the network cannot gain from the parasitic black-market without risking losing a white-market customer. Although black-market revenues could be quite large, they would never be worth the risk of losing much greater white-market revenues, because a parasite cannot survive without a healthy host.

Single Packet Flow Attacks

Negative Single Packet Flows. An attacker (or an accident) might create many single packet flows, each consisting of a Negative packet. These would not decrement the token bucket at the ingress bulk congestion policer. Indeed it might seem as if they would conceptually consume negative tokens and therefore act to fill the bucket. They would eventually reach the egress dropper and be treated with the Bulk of misbehaving packets. This would rapidly cause the dropper's Bulk flow state to accumulate a highly negative balance and discard them all. But on the way, it seems they could wreak considerable havoc traversing intervening networks. They could drag down the metering of downstream congestion at any borders they crossed, at least until they were picked up in a sample of the Downstream Congestion Inflation mechanism.

Happily, the proposed ingress policer of §11 should stop Negative packet attacks at source. It includes a module to determine how much to discard which is identical to the per-flow egress dropper of §7. This dropper would treat all these Negative single packet flows in the Bulk of misbehaving flows, just as an egress dropper would; rapidly leading it to discard them all. Alternative ingress policer designs SHOULD make sure they also included dropper behaviour that would block such attacks.

Cancelled Single Packet Flows. It might seem that numerous flows of single Cancelled packets could be used as a dummy traffic attack (or as the result of an error). The egress dropper (§7) deliberately allows them through untouched, on the grounds that they might be congestion marks within legacy ECN traffic. However, the ingress policer of §11 comes to the rescue again in this case. Its algorithm for metering downstream congestion to consume tokens from the bucket counts Cancelled packets as part of its normalisation of downstream congestion (§8.2.7). Thus, such attacks would be blocked as the token bucket rapidly emptied.

Dummy Neutral Background Load

Bauer & Faratin [BFB06] pointed out that sources could be motivated to send dummy traffic at little or no cost to themselves just to increase congestion marking levels within some interior network, not intending to cause any drop. Such sources might be motivated by a desire to confuse the network into investing in capacity at other people's expense (§12.1.1), or simply to cause expense to others (whether targeted or random others).

A source could simply send Neutral packets whenever it had nothing else to communicate. Neutral packets might not get very far if they all used separate flows IDs, because the dropping module within the ingress policer would include them in the Bulk of misbehaving flows, perhaps dropping most of them. But, for the cost of a single Cautious packet, valid flow-state could be opened in the ingress dropper allowing through large numbers of Neutral packets. There would be no need to send any Positive markings in such flows, as long as the flow was sufficiently Positive to get through the dropper within the

ingress policer. The source would not need to balance any Negative congestion marking added later on the path. Its nefarious purpose would have been met if it caused congestion somewhere in the middle of the network. The source wouldn't care if the flow became Negative to be discarded by an egress dropper before leaving the Internet.

Bauer & Faratin correctly point out that, for a flow causing some congestion, no network element before the first dropper after the congestion can rely on the expected downstream congestion the flow declares. Therefore, to strictly assure congestion signal integrity, every network element would have to run a per-flow dropper. Taken to this extreme, the re-ECN wire protocol would allow a negative flow to be immediately identified locally at whatever point it became persistently negative.

However, the 'Sample-Based Congestion Volume Inflation' process (§8.2.4) is proposed as a pragmatic alternative to ubiquitous deployment of droppers. It aims to eventually remove under-declaring flows, at least from the point where they become negative, but only lazily. Push-back further isn't required, but is optional using hints. The most important aim, though, is to remove the polluting effect of negative flows from border measurements. This removes any financial motivation for networks to attack each other.

This process accepts that not all congestion signals will always be sound. It prioritises aligning all the networks' incentives to remove negative traffic above actually removing it. A detailed rationale is given in §8.2.4, which admits that it is more an architectural direction than a detailed mechanism proposal at this stage. Briefly, the following steps are proposed:

1. Remove inter-network attack motives, align the incentives of all networks to remove negative flows;
2. Use localised sampling solutions to detect negative flows;
3. Optionally pass on trace-back hints;
4. Attack the root cause lazily.

12.1.2 Networks Against Endpoints

'Faked' Congestion Marking

A network operator being able to fake 'congestion' marking is a feature of the re-ECN framework, not an attack. When we first defined 'congestion signal integrity', we warned that it should strictly be termed 'shadow-price signal integrity'. If packet marking is used for market pricing, it is only equivalent to congestion marking in a perfectly competitive market [MMV95]. Then the market price will tend towards the congestion price.

In a network, some links may suffer less competition than others, and competitiveness may change over time. Certain key links may provide the only route to a particular part of the world. Or there may be a temporary surge in interest in communicating between two parts of the world, but no quick way of providing extra capacity. There may be commercial or political barriers to entry that make it hard to compete against certain links.

In such cases of imperfect markets for specific connectivity, an incumbent network operator might well ECN mark packets more often than they would need to if they were only signalling congestion. One could say this 'faked' the existence of congestion in order to raise the price. But, if a market is generally competitive, one cannot really argue that the originator of a price signal shouldn't be allowed to set the price as they see fit. A high price for scarce but popular connectivity encourages competitors to invest in alternative routes, which is all part of the competitive process that drives the price downwards towards the marginal cost of capacity, below which no competitor wants to go—at least not for sustained periods.

One motivation for the contract & balancing mechanism mentioned earlier [AKS06] is to prevent networks profiteering from congestion marking once they have signed up their customers. The operator's total revenue is fixed at the initial contract stage, because the balancing charges must all sum to zero. However, no mechanism is proposed (and none seems viable) for customers to check whether the operator has indeed set all the balancing charges so they sum to zero as promised.

It would be possible for strategising networks to attempt the 'Predation Game' [BFB06] where they route traffic onto paths heavily used by competitors to increase their price disproportionately. However, this game assumes a convenient network topology for the attacker, so that the 'predatee' network cannot itself react to the high price and move its traffic to other paths— indeed it may both reduce its own price and counter-attack against the predating network, reflecting the original strategy.

In general, if congestion signalling is widely used to align user and network incentives, one would expect traffic to continually shift [KV05], continually aiming to converge towards equal congestion at all links [WHBB08]. As soon as any one network attempted the predation game, end-points and other networks would shift traffic around to continue to try to converge towards equalised congestion.

Thus, routing is the foundation of competition in networks. And routing based on packet marking information would tend to create a highly competitive market. Indeed, by revealing the price from any point in the network interior to a destination, re-ECN could oil the wheels of competition further. However, routing needs to see the cost of all paths, while re-ECN doesn't reveal a price unless a path carries traffic. Nonetheless it would potentially be possible for the local routing process to 'fill in' the cost of jumps from paths where flowing data would reveal the downstream congestion cost to all the other points without data flowing to that prefix (described in more detail in an early technical report on re-feedback [BCSJ04]). By adding up measured congestion at each link along the jumps, the routing system could gradually find lower cost paths than those currently offered by the routing system. However, no detailed research on this topic has been attempted, and further discussion on using re-ECN to offer new lower cost routes is outside the scope of the present dissertation.

Although competition is generally intensified through congestion-informed routing, there is one case of particular concern; the termination monopoly, discussed below.

Termination Monopoly

The term 'termination monopoly' originated in telephony markets. Most individuals only have one telephony provider at a time. Once an individual has chosen their access provider, the only route to that individual from elsewhere is via that provider. Therefore, if part of the charge for others to make a call

to that individual is set by the terminating access provider, it has what is termed a termination monopoly.

This charge by the monopolist may not be separately visible to the party paying for the call; it may only be visible as an internal charge on the interconnection market. But overall, retail prices will increase to cover the monopolist's internal charge.

The re-ECN framework is vulnerable to exploitation by termination monopolies. Re-ECN deliberately uses a 'sender-pays' model, to avoid 'denial of funds' attacks against receivers. This is similar to the 'originator-pays' model common in telephony, but at the packet rather than the call level. Once a receiver has chosen its Internet access provider, senders have to route packets through that ISP to reach the receiver; the ISP has a termination monopoly. If the monopolist ISP increases packet marking, the sender has no choice but to consume more tokens if it wants to communicate with that receiver.

We argue that the termination monopoly problem should not be solved by re-ECN *per se*, but by adding an end-to-end receiver-pays model over re-ECN—at a higher layer. Below, we outline multiple reasons why a mix of sender-pays and receiver-pays is necessary and desirable, including to solve the termination monopoly problem. But we argue that it could be dangerous to add a 'receiver-pays' model at the packet network layer.

Termination monopolies are a consequence of two factors, each of which will be dealt with separately below:

- The tendency for network access to be a natural monopoly;
- The economic externality of the receiver's choice on the sender in a two-sided market.

Natural Monopoly of Network Access. Simple geography works against competition for physical network access to a dispersed set of customers. Consumers can get little benefit from a second competing access provider. Competition may bring down monopoly prices, but the base-cost per customer increases when two (or more) physical networks cover an area—each network has to cover the same geographical area, but for a sparser set of customers [Com02]. This argument applies even for competing technologies, e.g. cellular wireless and copper access.

Because access network monopoly has a natural cause, remedies usually need to be regulatory. In some jurisdictions, the regulator simply sets retail price caps. A successful regulatory strategy used in the UK residential market is for the incumbent network operator to be forced to lease its lines to competing retailers at a regulated price (termed local loop unbundling). Typically, conditions are also placed on the time and effort required for a customer to change providers (termed 'switching costs'). Thus, local loop unbundling creates largely unregulated retail competition by creating regulated wholesale competition [DR04].

Internet access is a logical layer on top of physical network access. If physical access is monopolistic, the monopolist can use its advantage to dominate in the Internet access market as well. But if physical access is properly regulated, Internet access is not a natural monopoly in its own right. As long as there are not other lock-in factors (like non-portability of network addresses) it is relatively straightforward for multiple Internet access providers to compete to provide service through one single physical attachment point (or a small number of access points).

Although not common (yet), it is possible for a customer to multi-home Internet access over a single-homed physical access. At the Internet layer this completely removes any cost and effort barrier to switching providers. Multiple providers would be accessible simultaneously, so switching providers can be achieved on a per-packet basis. This could be achieved by a transport protocol in the end-point that could use multiple interfaces simultaneously [WHBB08]. Rather than each provider selling a fixed portion of the access capacity and losing the benefits of multiplexing, all providers could offer the whole of the available bandwidth. Then the customer could use any of the providers to access the Internet (in either direction), only noticing any difference in how packets are routed and in available capacity beyond the access link. Effectively, the multi-provider Internet resource pool could start at the home access router [JBM08].

Two-Sided Market. Even if the receiver is multi-homed, this alone doesn't solve the termination monopoly problem. The problem is a consequence of communications markets being naturally two-sided [FW06]. In a two-sided market a platform sells a product to two types of customer. Customers of one type (e.g. senders) only get any value from the product if they can use it to access the other type (e.g. receivers).

The re-ECN requirement to support 'sender-pays' as a default does not stem from a commercial requirement, it stems from a technology constraint. Re-ECN is designed to ensure that the party directly causing a cost can be held accountable.[8] In a packet network, the sender directly causes the costs—the receiver may ask the sender to cause the costs, but ultimately the sender can choose whether to or not.

It is necessary to make 'sender-pays' the default model for usage at the packet internetworking layer. Otherwise, any charge to the receiver opens it to having to pay for unsolicited packets, while the senders bears none of the cost [Bri99b].

Nonetheless, commercially it is desirable and necessary to support a mix of sender-pays and receiver pays. Most transmissions are not unsolicited, and therefore both parties accrue some value from them. It can often happen that the sum of the utilities of both sender and receiver is greater than the marginal cost of a transmission, but neither alone are greater than this cost. Therefore if the sender, say, has to bear all the cost, it will choose not to communicate on many occasions when it would have otherwise—if the cost had been shared [Bri99b].

This problem can largely be solved by covering usage costs with a subscription, rather than per-transaction charges [Arm06]. There is a subtle distinction between an unlimited-use subscription and an allowance or quota. An unlimited use subscription solves the two-sided market problem but opens up a new free-riding problem. A quota (as used by our congestion policer in §11) prevents free-riding but doesn't solve the two-sided market problem, because one party bears the cost of each transaction.[9] Therefore, we still need a receiver-pays model.

We will now explain why support for 'receiver pays' can solve the termination monopoly problem

[8] The abominable term 'cost-causation' is used in the telephony interconnection literature, where the call originator is the analogue of the packet sender.

[9] However, psychologically, a bulk quota can tend to merge the gains and losses from individual transactions in different directions, smudging each per-transaction decision in favour of more communication.

too. When a receiver chooses its access provider or providers, its choice represents an economic externality for senders who will have to pay to communicate with the receiver. Senders not receivers can experience higher costs as a side-effect of the receiver's choice. The receiver has no direct incentive to choose a provider that minimises costs to senders.

Certainly, if the terminating ISP introduces higher packet marking, the sender's congestion control will tend to respond by reducing its rate. This seems to give receivers an incentive to choose an ISP with lower downstream marking—a quality rather than price incentive. But the receiver can compensate for this quality degradation by downloading in parallel from multiple senders (known as a swarming download). If many receivers do this, marking will increase in the terminating network further still. But until the marking represents real congestion (i.e. drop) receivers can always improve quality while senders have to pay for it.

Therefore the only sure way to give a receiver the incentive to choose an access ISP that doesn't inflate its marking is to make it possible for a sender to ask the receiver to pay for (or at least contribute to) the cost of a download. To avoid vulnerability to denial of funds attacks, it should be possible to arrange 'receiver-pays' for a whole flow, but it would rarely be appropriate for a single packet flow, except where trust prevails. That is why we argue it is only appropriate to arrange 'receiver-pays' at a higher layer than the packet network, and for 'sender-pays' to otherwise be the default.

Arranging for the receiver to be accountable for the congestion markings on packets within a particular session is an interesting question, and there are a number of ways it might be done, but they are out of scope of this dissertation. Here we only need to know that it is not appropriate to solve the termination monopoly problem in the network layer re-ECN protocol, and that it should be possible to solve this important problem at higher layers.

Biased Congestion Marking

So far we have assumed that network AQM algorithms do not discriminate between packets when marking the presence of congestion. But we should consider whether adding price semantics to different markings gives a network any incentive to mark some re-ECN packet types more than others. We are solely concerned here about whether a network would want to discriminate how it spreads a total amount of bytes of marks that it has already decided to apply. Given a network can chose the overall amount of marked bytes at each link, there is no reason why it would want to mark more or less than this overall. Therefore below we consider whether there is any motivation to ECN mark packets arriving with particular re-ECN codepoints more or less than others.

Marking a Positive or Neutral packet decrements the worth of either by one (to Cancelled or Negative respectively). So byte-for-byte there cannot be any reason to mark one more than another. Anyway, there is an incentive not to mark them with different probabilities, because the downstream congestion metering formula of Eqn (8.6) takes the maximum of the two marking fractions if they differ.

Congestion marking either Negative or Cancelled packets has no effect, because they are already marked. One might therefore imagine that a network would prefer to mark other packets. Alternatively, one might imagine that a network would just inflate its general level of marking to allow for hitting some

packets that were already marked. In fact neither view is correct.

A packet that is already marked on arrival at a link in network N_B, may either have been marked by an upstream network or by an upstream resource within N_B. If the packet stream in question entered N_B from N_A at the upstream border with N_A, previously marked upstream congestion costs should have been taken into account using the precise downstream congestion formula (8.6) from §8.2.7.[10] Similarly, at the next downstream border, say into network N_C, this precise formula will also be used.

These precise formulae inflate downstream congestion allowing for the combinatorial probability that previously marked packets cannot be marked again. Therefore, if a link wants to add a certain amount of bytes of marking, its simplest strategy is to add that amount of marking to packets of all four markings discussed so far with the same probability as each other. This will result in less bytes of added marking. But the border formulae will inflate their meter readings so that the outcome results in the required amount of marked bytes being considered to have been added.

There is no incentive to treat Negative and Cancelled packets with anything other than the same probability, because neither marking has any affect anyway.

Marking a Cautious packet decrements its worth by two, rather than one (because a Cautious marking is worth +1 and when marked it becomes a Negative packet worth -1). If a network just marks all packets with equal probability, the greater the proportion of Cautious packets in the traffic mix, the more the network will overstate its marking relative to the amount it intended. On these grounds, a network might wish to bias packet marking away from Cautious packets.

The proportion of bytes marked Cautious on the Internet is likely to be low, because the large majority of bytes has always resided in large flows [Wis07], which should only have a few Cautious packets at the start. Therefore a network may not consider it important to worry unduly about reduced marking of Cautious packets, which would probably be complex to implement.

Marking Cautious packets has a higher chance of delaying a new flow from starting than marking other packet types. In times of low congestion a network would not want to disproportionately mark packets that might bring in new traffic. In times of high congestion, the network might consider marking Cautious packets a useful strategy that afforded some degree of flow admission control.

Discriminatory marking of Cautious packets is deferred for further research. As far as can be briefly ascertained, it does not seem to cause any particular harm to mark Cautious packets with the same probability as other packets.

Of course, there are pre-existing incentives to bias congestion marking against packets in more valuable flows—simple price discrimination. Although this is an interesting subject, the introduction of re-ECN doesn't change the incentives, therefore such bias is out of scope of this dissertation. Nonetheless, the availability of transparent congestion markings should allow customers to more easily test which particular operators are biasing congestion marking against certain types of traffic.

[10] This also applies if the upstream border is with an end-customer—we recommended congestion policers should also use the precise formula.

12.1.3 Ends Against Ends

SYN and Initial Packet Attacks

Public servers can be high profile targets, particularly when the information they hold is in high demand (e.g. during flash crowds). At such times it takes minimal extra effort for an attack to push a server into overload and causes maximum loss in value to the victim service and its clients.

Public servers (or stateful middleboxes protecting them) that intentionally listen for connection attempts from any address (an 'unbound' LISTEN) cannot discriminate between initial packets of legitimate flows and initial packets that they will later discover were unwanted. Both necessarily contain unfamiliar source addresses so they cannot know whether the source address in the request might be spoofed or whether the source intends to continue the connection attempt. At least they cannot know without responding to the request and checking for a valid response. Initial packet attacks can have either or both of two intended effects:

- they can exhaust connection setup memory on a server (or on a stateful middlebox);
- and they can exhaust link bandwidth (either of a server's interface or further into the network).

§12.1.1 outlined how re-ECN considerably raises the bar against bandwidth exhaustion attacks. The present section discusses how re-ECN is also intended to help mitigate memory exhaustion attacks.

With intimate knowledge of the transport semantics, it is sometimes possible to prevent flow initiation packets consuming memory on the server by turning round the connection state into a well-crafted response, e.g. using SYN cookies in TCP or similar approaches in more modern transport protocols like SCTP or the HIP base exchange [Edd07]. It would be useful to be able to offload this initial connection validation to middleboxes, but we want to avoid this resulting in middleboxes that never allow new transport protocols to evolve.

Deciding whether a communication is unwanted is inherently a multi-layer process. For instance, an e-commerce site might consider sessions as unwanted if they seem not to be heading towards a sale [CP98]. Nonetheless, each layer should detect and reject any activity that does not meet the basic requirements of communication at that layer. And it should contain any generic hooks that will help the next layer up expedite such a decision.

The introduction of the Cautious marking into re-ECN has been overloaded with more than just network layer semantics in order to help higher layers—but taking care not to do anything that is not generic for all higher layers.

To a server, any packet *without* the Cautious marking means, "If your storage resources are stressed, and if you have no matching flow state, this packet can be discarded, whether or not it claims at the higher layer that it is the start of a flow." Then a stressed server can concentrate on those initial packets that show they were willing to 'pay their way' to get across the internetwork. This does not imply ISPs *have to* charge or account for Cautious packets. But servers that want this protection can attach to ISPs that limit Cautious packets at their outer borders.

Also the Cautious marking has been defined so that a server can offload initial packet processing to

a middlebox. They can either turn round initial packets on behalf of servers, or they can forward on valid initial packets to the server. The architecture supports chains of middleboxes, so that there is no problem if independent deployments place more than one middlebox on a path. Middleboxes can protect their own memory in the same way that re-ECN is designed to protect a server's memory resource.

The various elements that re-ECN provides to help higher layers with initial packet have been introduced in a rather piecemeal way throughout this dissertation. Therefore, §12.2.1 entitled 'Flow Start Architecture Revisited' brings all the elements together. We defer further discussion to that section. Suffice to say, the new architecture gives higher layers as much generic help as possible with packets that might start new flows. However, the IP header remains generic—as stated above, the Cautious marking doesn't even mean 'flow start'.

We have also defined a congestion signalling channel to support this relationship between middleboxes and servers. It allows a middlebox to warn a server if it has not stored flow-state for an initial packet due to memory congestion (§10.1). And forwarding elements can also congestion mark initial packets if their bandwidth is congested (§9.2). Then the server can feedback the congestion status of an initial packet request, so that valid clients quickly know what has happened and what hasn't happened.

State Keep-Alive Attack

This attack can be launched by an end-point against any machine that holds flow-state, whether a server or a middlebox such as the re-ECN dropper itself. It will be described as an attack against the re-ECN dropper to be concrete.

A malicious source can try to exhaust the memory of the dropper by creating many single small packet flows, with Cautious markings. The dropper could be forced to create as much flow state as could be triggered by single packet flows sent within one idle flow state minimum timeout. One source with uplink capacity C could hold open memory $S = MCT_i/s_{\min}$, where M is the memory held per flow entry, T_i is the minimum idle flow-state timeout and $s_{\min}$ is the minimum packet size that can legally initiate flow-state.

This would be costly to the source, given it is likely to be limited by its network provider in how much Cautious traffic it can send in a certain period (for example, using the congestion policer of §11, in which case C would be taken as the maximum rate of Cautious packets). But a source may still achieve its malicious objectives to launch the attack for a short while without regard to its own long term welfare. Particularly, if the attack is launched from a zombie host under the control of some other malicious party.

The maximum memory that a set of misbehaving source can cause the dropper to allocate is ultimately limited by how much they can hold open, not how much they can open in the first place. This is a much more efficient attack, because each memory entry only has to be paid for once. Having opened some flow-state with a Cautious packet, at least one packet per flow-state entry per timeout period is needed to hold the memory open (the sender need not 'pay' anything more for these so-called 'keep-alive attack packets'). Even if Cautious packets are rate limited, more and more memory can be opened as tokens become available for further Cautious packets, as long as sufficient capacity is left for all the keep-alive attack packets. However, if there is also path congestion between an attacker and the dropper,

it will have to consume Positive quota to hold open dropper memory as well as Cautious packets to open new flows.

Ultimately, the maximum memory that can be held open will still be that predicted by the above formula for S; at least one packet per flow per timeout period. But C should not then be taken as the maximum rate of Cautious packets. Instead it will be the maximum line rate for Neutral keep-alive attack packets (for a whole set of attacking sources). Beyond this point, a set of misbehaving sources would be unable to make the dropper consume more memory; in order to cause the dropper to allocate more memory they would simultaneously have to release the same amount.

Receiver Suppression of Feedback

For a source-destination pair that are trying to communicate, re-ECN creates an incentive for the receiver, not just the source, to assure the integrity of the downstream congestion signal declared to the network by the source. If the receiver suppresses congestion feedback in the hope the source will go faster, the dropper will still detect that the source is under-declaring actual path congestion and proportionately discard traffic. In other words, download rate depends as much on honest feedback as honest re-feedback.

However, the dropper's sanction only hurts if the receiver *wants* the downloaded data. If, instead, the receiver merely wants to make the sender consume its own congestion quota, or its own bandwidth resources, it can do so by suppressing congestion feedback. This class of attacks potentially harms the integrity of the re-ECN congestion signal, although that is likely to be a side-effect, not its intent.

The attacking receiver doesn't gain itself from this attack, which merely moves money into the pocket of a network operator at the expense of a remote sender. But even though the receiver doesn't gain, it's attack could be motivated by a grudge against the server.[11] Further, the receiver and the gaining network might be acting in collusion or, indeed, they may actually be one and the same entity.

Before we point to solutions to this class of problems, we will argue next that the end-to-end transport or application layer is the appropriate place for solutions, not the network layer.

An End-to-End Problem. Ultimate control over sending lies with the sender. If a receiver fools a server into dedicating a large fraction of its own resources (whether bandwidth or congestion quota) to that receiver, the server only has itself to blame. A server doesn't have to apportion its resources according to the congestion experienced across the network path of each connection. That is probably a sensible strategy if the server is not the bottleneck. But otherwise, the server should consider its own congestion as well. And there is no reason why the server cannot determine its own sharing policies.

A server might use a congestion control that always consumes a constant token rate for each connection (i.e. proportionally fair flow rate). Then if a receiver under-declares path congestion, the source will go faster for that connection, but still consume tokens at the same rate per connection. But one destination may open many connections from the same server, but make the server think they are all different. Or one destination might open many more connections over time than others, but use different addresses for each one. The network itself might be creating fake requests in order to swell its revenues by triggering multiple payments for deliveries of content from the server to itself.[12]

[11] Recall by Assumption 8.1 our attack model allows end-points unbounded malice.

[12] Precise accounting for marginal costs highlights this problem, but it exists otherwise. For instance, if a server's network

If a server were concerned about such 'Sybil' (split identity) attacks, it has it within its power to counter them, perhaps using end-to-end authentication to tie each connection to a real-world identity, or some pragmatic alternative[13]. This is an end-to-end authentication matter that we shouldn't try to solve in a network layer protocol.

It has become common for many public servers to offer content to all-comers. If there are attendant delivery costs, the server chooses to take the risk of paying for all the delivery costs even if some transfers to some receivers don't warrant any payment, or if some receivers overuse the service. If a server is concerned about abuses of its generosity, it shouldn't offer to pay for delivery to all-comers without first establishing the real identity of its customers. This is an end-to-end matter that is likely to be separate from the network layer accountability that re-ECN offers.

Why Not an End-to-Middle Problem? As we have seen, a receiver that is only interested in causing costs to a server can under-declare path congestion. The tokens the server consumes will only cover the cost of congestion up to a point part way along the path. Networks up to this point do not suffer any harm as they are fully recompensed for the externality of their congested transmission. Networks beyond this point can detect there is a problem with negative flows and remove the offending traffic to protect themselves against losses, albeit not instantly (see §8.2.4 on Sample-Based Downstream Congestion Volume Inflation).

Therefore, it seems that the hapless server is left exposed. Networks can cover their backs, either ensuring their costs are covered, or removing traffic if they are not. But networks *seem* to have no incentive to advise the server that it is paying them to transmit data partway across the Internet only for it to be discarded in the middle.

In fact, networks *do* have an incentive to help servers pay them less; by helping to remove unwanted traffic. My paper on "Using Self-Interest to Prevent Malice" [Bri06] explains that a network that helps a sender stop spending on useless data transfers gains a competitive advantage over another network that quietly pockets the proceeds, as summarised in §12.1.1—gains from the black market aren't worth losing a much greater volume of white market business. Therefore networks have an incentive to use the 'lazy' process of tracing back negative flows to their root cause, as described in §8.2.4. If this process leads a network to accuse an innocent sender of being the root cause of attacks to certain destinations, it should be within that sender's power to test and black-list the offending destinations.

As we show below, it is indeed within the sender's power to test the integrity of congestion feedback from any destination on an end-to-end basis without any help from the network.

Proposed Solutions. The ECN nonce [SWE03] is an experimental network layer solution to this problem. The ECN field provides the redundancy of two states to mean 'not congestion marked but able to understand marking'. The source can weave a nonce into a stream of packets by setting either of these states in a pattern it stores. Then it can detect if the receiver or any network element has tried to

charges depend only on its access capacity, the network can generate fake server demand to increase the capacity the server needs to buy.

[13] For example, from the author's personal experience, Google servers under stress from a perceived DDoS attack issue CAPTCHAs—Completely Automated Public Turing test to tell Computers and Humans Apart.

'unmark' a congestion mark, or deny a drop. A congestion mark sets the ECN field to a third codepoint. For every acknowledgement claiming no congestion was experienced, a TCP[14] receiver is asked to feed back a one-bit sum of all the nonces seen since the last congestion event. So if anyone[15] tries to unmark a packet, or deny a drop, they have to guess which of the two ECT states the sender originally set. The sender can then detect whenever they guess wrongly with a 50:50 chance per acknowledgement.

Due to pressure on IPv4 header space, re-ECN had to re-use the same codepoint as the ECN nonce for one of its encoding states (Positive). Therefore both cannot be used simultaneously in the same internetwork (for IPv4 at least). The reasoning for proposing to sacrifice the ECN nonce was that re-ECN covered a much greater attack space and, as far as anyone can ascertain, the nonce has never been implemented anyway.

However, although the attack space re-ECN covers is much larger, it is not a complete superset of that covered by the ECN nonce. The one part of the attack space covered by the ECN nonce but not by re-ECN is that described above—where the receiver doesn't care about getting any data, it just wants to cause expense to the sender (see 'Why Not an End-to-Middle Problem?'). Although re-ECN networks should eventually be able to trace the attack back to the sender, if the sender knows it is innocent it needs a way to check whether the receiver is indeed the root cause of the attack, or whether it has been falsely accused by its network provider. If such an end-to-end test were possible, the sender could obviously use it to identify a non-compliant receiver at any time, not just in response to an accusation from the network.

Moncaster *et al* [MBJ07][16] proposes such an end-to-end test. It is specific to TCP and has the advantage that it does not require a receiver implementation to be modified in order to be found guilty. A modified source picks a receiver to test, then occasionally randomly adds a small delay to a segment, so it appears slightly out of order at the receiver. If a receiver reports receipt of the segment in order it comes under suspicion, but is not proven guilty. The second phase of the test delays a packet by more than an RTT, which can prove the receiver is lying—if it acknowledges receipt of an unsent packet.

Moncaster *et al* also provides a survey of this class of attacks and other end-to-end defence techniques, including Savage *et al*'s end-to-end transport layer nonce proposal [SCWA99].

To summarise, receivers can launch some very nasty attacks on senders that threaten the integrity of congestion signals. However, solutions to receiver suppression of end-to-end feedback should be end-to-end, not in the network layer. Good end-to-end solutions are available that complement re-ECN congestion signal integrity protections at the network layer. Therefore, further detailed coverage of this issue can be ruled out of scope for the present dissertation, as it is sufficiently covered by the above references.

Single Packet Reflection Attacks

This class of attacks is targeted at servers that are designed to automatically respond to datagram queries with reasonably large datagram responses (e.g. DNS servers). A client can send high volumes of small

[14] The 1-bit nonce only works for transports like TCP that provide in-order delivery.

[15] Whether receiver or network

[16] Co-authored by the present author.

datagram queries and cause the server greater cost by responding to each query with a larger response datagram. To get these datagram responses through the re-ECN system it has to mark them as Cautious. The client could spoof different source IP addresses each time.[17]

This attack actually exposes two separate problems.

- Services that are expected to turn round quick, small responses to queries are vulnerable to spoofed queries. This is a problem that predated re-ECN and has many proposed solutions. This first aspect of the problem is therefore out of scope of this dissertation.

- Re-ECN requires flows to start with a Cautious marking, the relative cost of which increases excessively for shorter flows.

So far, we have conveniently been able to dismiss this cost, tending to amortise it over a mix of longer and shorter flows. However, if large numbers of single packet flows all have to use the Cautious marking, it could make running a directory server far more expensive than would be warranted by the actual costs of congestion even if the uncertainty of predicting congestion for single packets is added.

We have already mentioned (§6.1.2) that the worth of a Cautious marked byte need not equate to a Positive Byte, but instead find its own market price. This is the proper solution to this problem, but our dropper design would have to be considerably modified as it assumes these two markings are equivalent. The problem this issue really highlights is lack of space in a packet header to allow anything other than a unary price for each byte in a packet.

Other work-rounds are possible, but they all move the problem around rather than solving it properly.

- One could argue that re-ECN merely exposes the cost of opening many single packet flows without knowing the state of resources on their path. A directory server often has long-lived relationships with a set of clients. If re-ECN gave it the incentive, it may well switch to maintaining TCP connections with these, rather than using UDP datagrams.

- A directory server serving a mix of datagram and connection clients might rate limit use of Cautious markings so it would prioritise long-lived relationships when under stress.

- The directory server could mark some responses as Neutral when under stress and risk them being dropped by a re-ECN dropper;

- A directory server might expect clients to 'pay' from their quota to assure a response, using an end-to-end quota transfer mechanism (see §12.1.2 on solving Termination Monopolies);

- §12.1.4 includes speculation that a proxy might aggregate short flows into fewer larger flows, using its aggregated knowledge of congestion on fewer shorter paths. In turn queries to and responses

[17] A colleague, Adam Greenhalgh first pointed out this issue at the 5th CRN/CFP architecture working group on a Denial-of-Service Resistant Internet, Cambridge, UK, 21 Nov 2005. Mark Handley also pointed out an amplification of this attack that he had previously identified (separately from re-ECN): a client can greatly amplifying the necessary size of responses by requesting DNSSEC authentication of a response without authenticating its own identity.

from directory servers could be directed via such proxies, which might reduce the need for single packet flows by aggregating together queries and responses to and from numerous directory servers. However, for each connection with a directory server that is split by a proxy, two concatenated connections will generally result. Only the split connections from the proxy to the directory can be aggregated, while the individual connections between each client and the proxy would all still be on separate paths. Thus the cost of opening multiple connections would simply shift from the directory server to the proxy.

Further research is therefore needed on whether re-ECN's high relative cost to services offering single packet responses can be mitigated. directory services.

12.1.4 Byzantine State Transitions

End-to-End Integrity Checks

The encoding of the re-ECN wire protocol [BJMS09a] has been devised so that all legal transitions within the protocol do not change the RE flag (the extra header bit used to extend the two-bit ECN field) once it has been set by the source. This allows an end-to-end transport to detect if any illegal transitions have occurred in transit, by comparing the RE flag sent and received.

In IPv4, the RE flag is considered mutable (even though a use for it has not previously been defined). Therefore it is masked out before applying IPsec authentication, so it is not covered by end-to-end integrity checks. In IPv6, we also arrange for the extension header used for the RE flag to be considered mutable by IPsec. We took this approach to allow proxies to change the flag on behalf of the sender.

Therefore, if the source or the transport wishes to check the integrity of the RE flag setting, it will need to arrange it's own higher layer integrity check. This would be relatively easy to do, but we have not made any specific proposals on this. Such a proposal would either require the sender to trust the receiver's feedback or to arrange to include this bit in end-to-end integrity checks (§12.1.3).

It would seem that simple feedback without any cryptographic checks would typically be sufficient. In the re-ECN protocol, there are no cases where the sender relies on feedback from the receiver of a change to the RE flag (because it is not expected to change). Therefore if the source asks a receiver it doesn't trust to feed back the setting of the RE flag it received, receivers seem to have little independent incentive to lie—other than to falsely accuse the network of changing the flag, or to collude with the network to hide a change in the flag.

The tentative 'Covert Marking' proposal in §11.3.1 to extend the re-ECN protocol is an exception to the above. The policer changes the RE flag on Positive packets to warn the transport that the drops it is experiencing are probably due to policing, not congestion. A robust feedback mechanism for this extension in the presence of an untrusted receiver is yet to be designed (see §§12.1.4 & 13.2).

Re-ECN Unexpected State Transitions

Figure 12.2 shows unexpected transitions between re-ECN protocol states. It was compiled by taking all the possible transitions between the five re-ECN codepoints and subtracting those defined for the re-ECN wire protocol (see Fig 6.2), leaving the remainder shown in the diagram.

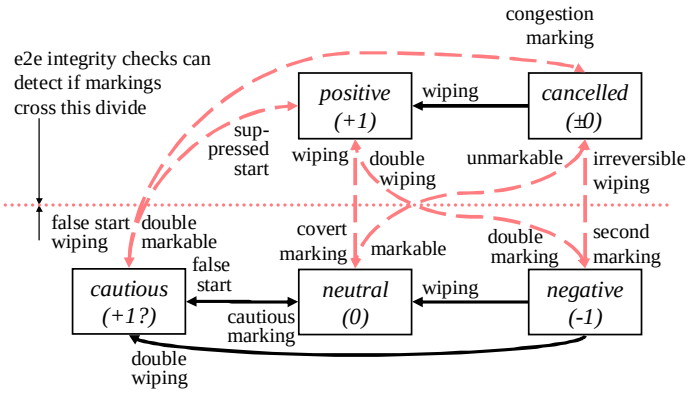


Figure 12.2: Re-ECN Unexpected State Transitions.

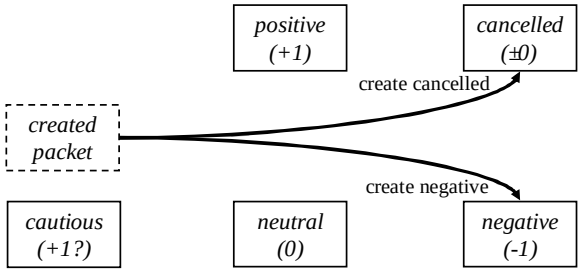


Figure 12.3: Re-ECN Unexpected State Initialisation.

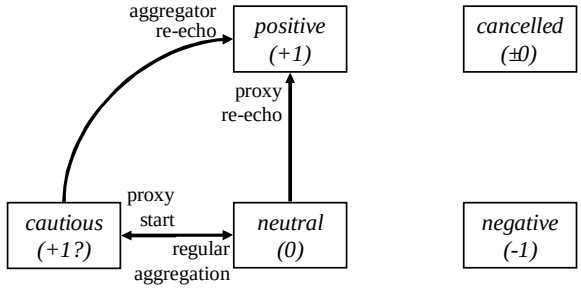


Figure 12.4: Re-ECN Expected Proxy State Transitions.

The horizontal (dotted) divide through the middle of the diagram separates two sets of codepoints that are distinguished by different settings of the binary RE flag. The wire protocol has been devised so that all legal transitions within the protocol do not cross this divide. Then an end-to-end transport can detect whether any unexpected transitions have occurred in transit (see 'E2E Integrity Checks' in §12.1.4 above).

It will be noted that the large majority of unexpected transitions are those that cross this divide (shown dashed). There is no harm in the end-points making these transitions; the source would only be fooling itself and the receiver is not relied on to report anything that would be affected by any of these transitions (as just pointed out under 'E2E Integrity Checks').

The only exception to this statement is the tentatively proposed use of 'Covert Marking' as a signalling channel from a policer to the transport. This is also discussed under 'E2E Integrity Checks' in §12.1.4 above.

There are two remaining types of unexpected transition that are less easy for the transport or the source alone to detect, as they do not cross the horizontal divide:

'Wiping', 'Double Wiping' or 'False Start': Wiping or Double wiping reverses congestion marking. §12.1.3 on 'Receiver Suppression of Feedback' discusses this issue and points to end-to-end solutions, arguing this should not be treated as a network-layer problem. Networks could theoretically reverse congestion markings, but they have no incentive. A marking inserted by a forwarding element earlier in the path can either have been added by the local network (N_B) or an upstream network (say N_A). The local network N_B has no incentive to remove its own markings, and it has no incentive to remove markings already added by an upstream network such as (N_A. If N_B removes Negative markings the feedback loop reduces Positive markings, making the whole effect equivalent to N_B giving money to its upstream network N_A. Regarding 'False Start', a network has no incentive to revert a Neutral packet to Cautious either (again, equivalent to paying money upstream).

The anomaly detection mechanisms introduced in §8.2.8 are designed to protect against accidental wiping of markings due to misconfiguration or other human error.

'Cautious Marking': This is the one transition a network could do that doesn't change the RE flag, but isn't covered by re-ECN's wider incentive framework.[18] It involves changing a Cautious marking to Neutral, without actually congestion marking the packet (i.e. the protocol doesn't define a Neutral packet as a congestion mark, even though the end-points could work out between them that it used to be a Cautious packet). We cannot see any motive for a network to make this transition, but if a well-motivated attack is identified, we will have to find a way to protect against it. If a receiver made this transition it would have no effect (because no-one relies on the receiver telling them it has received a Cautious mark).

[18] But see the potential use of this transition as 'Regular Aggregation' in the discussion of 'Proxy State Transitions' below.

Re-ECN Unexpected State Initialisation

Figure 12.3 shows initial packet markings that a re-ECN source would not be expected to set if it complied with the protocol. It was compiled by taking all the five possible initialisations of re-ECN codepoints and subtracting those defined as valid for the re-ECN wire protocol (see Fig 6.2), leaving the remainder shown in the diagram.

This section is merely included to show that an exhaustive check has been made of all possible state transitions. Both the resulting unexpected state transitions have already been covered in §12.1.1 on attacks using Negative and Cancelled packets.

Re-ECN Unexpected State Removal

Unexpected removal of the state of a wire protocol is merely a convoluted way of saying 'packet drop'. Re-ECN has been designed to be robust to drop, by carrying all numeric values as unary encodings (justified in §6.1.1). Treatment of drop as implicit congestion signals is discussed in §7.4.4 on 'Congestive Loss'.

Re-ECN Proxy State Transitions

Figure 12.4 shows a subset of the unexpected state transitions of Fig 12.2 that would probably be necessary for a proxy to act on behalf of a re-ECN source. The transitions shown are speculative, given definition of a re-ECN proxy is for future study. This figure is merely included for completeness.

The transition denoted 'Regular Aggregation' is worthy of note. The re-ECN wire protocol was speculatively designed so that a proxy could silently intercept re-ECN flows and aggregate them into fewer larger flows, given its position closer to the middle of the Internet. The thinking was to provide the incentives to keep the Internet's congestion accountability mechanisms safe and efficient even if the traffic mix became dominated by short flows, including single packet flows (a possible prerequisite for the vision of ubiquitous computing [Wei91]). Otherwise such a traffic mix would leave the Internet without the benefit of path feedback for most of its traffic. There would be a strong incentive to operate such proxies if the proxy could absorb the excess value of all the Cautious packets initiating the small flows, because it would need less Cautious packets for the aggregate it could create from all the tiny flows. However this is all for further study because the receiver side of this proxy would require a different commercial arrangement with the network from that we have described so far for a regular receiver (zero income even from positive flows).

Tunnelling Re-ECN Codepoints

Tunnelling in the context of the re-ECN protocol is defined in the protocol specification [BJMS09a]. The proposed extended ECN (EECN) field in the re-ECN wire protocol is deliberately divisible into the ECN field for forwarding elements to alter, and the RE flag that they do not touch.

Because congestion is exhaustion of a physical resource, if the transport or higher layers are to deal with congestion, congestion notification must propagate upwards; from the physical layer to the transport layer. The transport layer can directly detect loss of a packet (or frame) by a lower layer. But if a lower layer marks rather than drops a forward-travelling data packet (frame) in order to notify

incipient congestion, this marking has to be explicitly copied up the layers at every header decapsulation. So, at each decapsulation of an outer (lower layer) header a congestion marking has to be arranged to propagate into the forwarded (upper layer) header. It must continue upwards until it reaches the destination transport. Then typically the destination feeds this congestion notification back to the source transport.

As IP packets traverse the internetwork, they can be tunnelled within outer IP headers, or the headers of any other protocol including, of course, logical link protocols. If any of these outer protocol headers support explicit congestion marking, it is necessary to arrange similar propagation of congestion notification up the layers. For instance, ECN and its propagation up the layers has recently been specified for MPLS [DBT08][19], and forward explicit congestion notification is being considered for Ethernet protocols.

It is not necessary to propagate the RE flag down the layers and back up again, as it should not change end-to-end. However, it must be visible whenever a packet crosses a trust boundary, so that downstream congestion can be correctly metered if required. It is assumed that IP headers will generally be visible at trust boundaries, given IP is intended as the internetworking protocol. Therefore, re-ECN border mechanisms should all work correctly as long as the RE flag is copied on encapsulation by another IP header. Header copying is specified in the re-ECN protocol spec. [BJMS09a], but it is typically the default behaviour for IP tunnel encapsulators anyway.

However, during standardisation of the ECN protocol in IP, an exception to header copying was specified for tunnel encapsulators. The reason was a perceived security requirement that turned out later not to be a concern. IPsec tunnelling has since been re-specified to mandate header copying. The IETF Transport Area is currently working on a draft to make header copying the default for all IP in IP tunnelling [Bri09]. Without this change, tunnels might reveal incorrect ECN values to re-ECN border mechanisms. This draft (written by the present author) explains all the issues with ECN tunnelling.

An analysis of possible opportunities for tunnelling to be used to cheat the re-ECN protocol is provided in [Bri08b, §6]. The broad conclusion is that there are no new threats as long as tunnel endpoints disappearing or emerging in the middle of networks are treated as potential border gateways, which is already standard security practice.

12.2 Re-ECN Protocol Reconsolidated

12.2.1 Re-Architecting Flow Start

This section brings together various parts of the new flow-start architecture that have been hinted at in a piecemeal way throughout this dissertation. It complements the initial definition of the re-ECN wire protocol in §6.1.2.

The initial packet of a flow is, by definition, a flow-related concept that doesn't seem to belong in the internetwork layer of the current Internet architecture. However, we have to revisit the current architecture because it gives the providers of network resources no role in mediating the conflicting

[19] Co-authored by the present author.

resource demands of hosts. The minimum resource sharing function we have been able to muster requires per-flow checking of congestion signal integrity at the network layer (see §7.3). Therefore, although we intend to prove that the resource sharing function can be transport oblivious (§12.3.1), we believe it cannot be flow-oblivious. This is a similar position to the recent proposal that end-point identifiers should be considered as a sub-layer between the network layer and the transport layer (an end-point ID sub-layer [FI08][20]).

Having added per flow checks to the network layer, we also had to add the minimum necessary support at the network layer to protect this per-flow function against cheating. To this end, we provided the Cautious marking that a source MUST use on the first packet of a flow if it intends to use other re-ECN codepoints. It MAY also mark some subsequent packets as Cautious. For re-ECN, the Cautious marking has the semantic "Each byte of this packet can be considered as a credit, which may be counted towards an account identified by the packet's flow ID."

But we also deliberately designed the Cautious marking so that it could be used as a transport-independent facility to flag that the packet may contain a request to set-up flow state. A network-layer state set-up bit has been argued for independently by Handley & Greenhalgh [HG04], who in turn acknowledged Clark as the originator of the idea.

Architecturally, we place the Cautious marking in the IP header because it provides a transport-independent way for servers and middleboxes to handle the resource congestion aspect of flow state, as part of the resource congestion signalling function of the IP header[21].

In contrast to the Handley proposal, the flow-start meaning of the Cautious marking is idempotent; a source can send more than one Cautious packet with the same flow ID to signify the start of a flow, and the first to be received implies a flow start, while any others have no extra effect[22]. This allows sources to insure against loss of Cautious packets and it allows the Cautious marking to be overloaded with two meanings: i) idempotent flow start and ii) re-ECN credit (which is additive not idempotent).

Also, rather than using a whole bit, the Cautious marking uses just one codepoint of the 3-bit extended ECN field ($3/8$ of a bit), given none of the other codepoints would be relevant on the first packet of a flow.

Because we broadly agree with the motivations for a flow-start flag given in the Handley proposal, they are listed below.[23] Annotation in square brackets highlights where we differ slightly:

- a transport independent way for servers or stateful middleboxes to identify packets needing special validation;
- a [stressed] server or middlebox receiving a connection set-up request with this flag not set would [could] simply discard the packet;

[20] However, agreeing to this sub-layer doesn't imply agreeing to the flow-regulation sub-layer idea in the same proposal!

[21] We avoid the term 'network layer header', given we believe there are generic aspects of the transport layer that are also best carried in the IP header [Day07], such as resource congestion.

[22] If a node times out its flow state, then receives another Cautious packet with the same flow ID, the new Cautious packet will, of course, have an extra effect; the node will consider a new flow has started.

[23] One of their motivations is omitted completely, being specific to another proposal in their paper—separation of client and server address space.

- packets without this flag set can [may] be discarded by [stressed] servers or stateful middleboxes if no matching flow state is found;
- Sites [networks] might rate-limit state-setup packets sent by some clients at their outgoing [incoming] edge;
- a way for stateful middleboxes (e.g. firewalls) to permit evolution of network protocols without always needing to know the protocol semantics, and to do some degree of transport-independent validation of encrypted traffic.

We prefer not to automatically discard non-Cautious packets with no matching flow state or non-Cautious flow start requests. Instead a middlebox or server MAY discard them before Cautious requests, particularly if its resources are under stress. This allows middleboxes to pick up mid-flow following a reroute. It also entertains the possibility that middleboxes could explicitly mark and forward rather than discard flow-start requests (flow admission control—see §7.3.3). We believe this would be a more principled way to explicitly signal rejection of a flow-start packet. The alternatives are either i) to drop the packet which is confusable with all the other reasons for dropping a packet or ii) to bounce the request with a rejection packet from the middle of the network, which may not have access to or understand the end-to-end transport (the same justifications as given already in §6.1.1). Instead, we send unambiguous explicit feedback via the receiver, which will understand the origin address of the transport.

The Cautious marking does not replace the SYN flag for TCP flows, but it complements it. The SYN flag has a stronger meaning of flow *re-start*; "Every time you receive a SYN, discard all your previous flow state and start a new TCP flow using the enclosed initial sequence number." In contrast, any packet *without* the Cautious marking means, "If your storage resources are stressed, if you have no matching flow state, this packet can be discarded, whether or not it claims at the higher layer that it is the start of a flow."

12.2.2 Forward Compatibility

It is important to ensure that a protocol contains some space for adding things that become important in the future.

re-ECN Wire Protocol. In IPv6, the re-ECN wire protocol [BJMS09a] uses one bit of a proposed IPv6 extension header, with 127b reserved for other uses (we have some in mind already). However, in the IPv4 header we have more of a space problem. Extending the ECN field uses up the very last unused bit in the IPv4 header.

Re-ECN uses five of the eight possible states of this proposed three bit extended ECN field. The original ECN wire protocol uses three states, but we have ensured re-ECN usage overloads two. A further state indicates a non-ECN-capable packet. Thus one 3-bit codepoint is left 'currently unused' or 'CU': `10-1`, where the notation `XX-Y` means an ECN field of `XX` and an RE flag of `Y`.

One codepoint isn't much, but it's not a bad trade for the last bit, given all the re-ECN achieves. But, if the Internet community decides to go ahead with re-ECN, there is another forward compatibility issue with this CU codepoint: what should equipment do if a packet arrives with a CU codepoint?

- It would be wrong to advise that such packets should be dropped. That would create a huge bootstrap problem for any future use of the codepoint.
- It would be wrong to say nothing. That would leave unnecessary uncertainty for any future use of the codepoint. It would also subtly imply that any such packets might present a security risk, which would probably lead to security equipment dropping them anyway.
- A better approach is to specify that, in the interim until it is defined, CU should be treated the same as another codepoint. But which one? Candidates are:

  Not-ECT (`00-0`) We can reject Not-ECT on the assumption that any future use would more likely build on the latest protocol features than the oldest, so it would be unlikely to want congested legacy forwarding equipment to drop rather than mark CU

  Legacy ECN (`10-0`) If re-ECN is deployed, it is likely that network operators will rate limit non-re-ECN packets (otherwise they would represent a loophole to avoid re-ECN policing). Therefore, CU packets would probably get rate-limited if they were given the same semantics as legacy ECN packets.

  Neutral re-ECN (`01-1`) Therefore, in the absence of any better suggestion, we recommend that equipment SHOULD treat CU packets as if they were re-ECN packets marked Neutral. Network equipment MAY log the presence of CU packets. But it SHOULD NOT treat them as any more of a security risk than a Neutral packet.

Future Role of Legacy ECN. If re-ECN is adopted, there will be no future role for the ECT(0) codepoint currently used by ECN, but not by re-ECN. Today only a tiny proportion of packets carry this codepoint, due presumably to the vanishingly small chance of one of a small proportion of ECN clients talking with one of the small (but larger) proportion of ECN servers. If this situation remains, it may be decided that those sources that upgraded to ECN early can be encouraged to upgrade to re-ECN too. Then the ECT(0) codepoint could be made available for some other use in the future, along with the CU codepoint (`10-0` & `10-1` respectively).

Flow IDs. The re-ECN dropper effectively recognises flow IDs as a sub-layer of the internetwork layer. However, the Internet architecture should allow novel end-to-end protocols to make-up their own flow ID syntax and semantics. We have had to recommend (§7.1) that the dropper treats any unrecognisable flow IDs with the bulk of misbehaving flows. This is at least better than recommending such packets should be dropped, but it is not much better. This effectively raises a barrier against any future end-to-end transport protocol using a new protocol identifer at the IP layer.

Even if re-ECN dropper can be updated to understand a new protocol identifier (e.g. a new protocol ID for DCCP has recently been standardised), the flow-ID uniqueness requirement of the dropper also places a requirement that packets of the same inter-process flow should all contain the same ID. For instance, if some novel security protocol wished to obfuscate the fact that a series of packets all belong to the same flow, it would not be able to use a pseudorandom sequence as a flow ID, without sharing the key to the sequence with the re-ECN dropper.

We can think of no way round these issues, but we felt they should at least be stated.

12.3 Re-ECN System Properties

12.3.1 Transport Oblivious Congestion Signal Integrity

When constraints on the re-ECN dropper were listed (§7.2), the two paraphrased here appeared to be irreconcilably in tension:

Sufficient Sanction: The dropper MUST introduce sufficient loss in goodput so that sources cannot play off losses at the egress dropper against higher allowed throughput at the ingress policer;

Transport Oblivious: It MUST NOT be designed around one particular rate response. An important goal is to give ingress networks the freedom to allow different rate responses and different resource sharing regimes.

There is another way of stating the same problem. It seems one can only know whether an individual finds it worthwhile to play off drop sanctions against the policer, if one knows how strongly the individual values increasing bandwidth. From Mo & Walrand's work [MW00][24], we know that the curvature of an individual's marginal utility for bandwidth gives the individual a direct incentive to use a transport with a particular rate response to congestion. Therefore it seems the question of whether dropper sanctions are sufficient may *not* be oblivious to the transport.

To claim re-ECN can be transport oblivious, we need to establish whether the re-ECN system creates a one-way barrier against individuals perverting the integrity of congestion signals, however much the individual values bit-rate, and however strongly their valuation grows with increasing bit-rate. To prove the re-ECN incentive framework can be transport oblivious, we use the following model and assumptions.

We use the bulk congestion policer of §11 as a canonical example of a transport oblivious policer. When we described it, we justified its inclusion in this dissertation by saying "We will need to establish whether re-ECN can assure the integrity of congestion signals *even when they are used*." The particular policer design we proposed allows customers freedom to choose whatever short-term response to congestion they wish, except they know that they are held accountable for the congestion they cause, because their long-term congestion bit-rate is limited.

As previously defined, the recent fractions of Positive and Negative re-ECN marked bytes are z and u_i at node index i along the network path (as the dropper does not drop Positive packets, z remains unchanged along the path). Index $i = 1$ on arrival at the ingress policer, $i = n$ on arrival at the egress dropper and $i = N$ on departure for delivery to the destination. To simplify matters, but without loss of generality, we will assume that $u_1 = 0$ at the ingress policer, that is, there is no congestion marking within the source's own network.

[24] The weight and utility curvature in Mo & Walrand's model parameterise a similar space to Bansal & Balakrishnan's increase and decrease parameters published the following year in their set of Binomial Congestion Control Algorithms [BB01]. We use Mo & Walrand's model because it affords a direct link with a wider economic model that we can use to compare net utility gains.

It is reasonably assumed that the user's utility for bit-rate x satiates, therefore the user's utility function for delivered bit-rate $U(x_N)$ is assumed concave over the operating region of the network (see later for further justification). The notation $\Upsilon(x)$ is used for the user's net utility (utility of delivered bit-rate x_N minus the cost of the quota used by bit-rate x_1 at the ingress policer). In our distributed setting, the source optimises the net utility of its delivered bit-rate $\Upsilon(x_N) = U(x_N) - zx_1$ as its primal part of the wider system optimisation [KMT98].

Theorem 12.2. *A customer with concave utility for the delivered bit-rate x_N of an end-to-end flow and free choice to declare expected downstream congestion z has no incentive to misrepresent z relative to path congestion u_n if the network makes the user accountable for cost zx_1 at the ingress and discards non-Positive packets from arriving bit-rate $x_n(= x_1)$ with probability $\pi_d = 1 - z/u_n$ $(z < u_n)$ at the egress.*

Proof. In the setting of Kelly's optimisation, there is no dropper and the customer pays for the bytes actually congestion marked after having traversed the network. A customer with utility $U(x_n)$ uses a rate controller that chooses optimal bit-rate x_n^* to maximise net utility $\Upsilon(x_n) = U(x_n) - u_n x_n$, which occurs where

$$U'(x_n^*) = u_n. \tag{12.6}$$

In the setting of the re-ECN framework, the customer's transport can choose to mark any fraction z of sent bit-rate x_1 and only use up quota zx_1. We will consider two cases: $z \leq u_n$ and $z > u_n$.[25]

Understated downstream congestion $z \leq u_n$. In this case the dropper will reduce the bit-rate delivered to

$$x_N = \frac{z}{u_n} x_1. \tag{12.7}$$

The price p of delivered bit-rate that a re-ECN sender effectively experiences is the quota used divided by the delivered bit-rate:

$$\begin{aligned} p &= \frac{zx_1}{x_N} \\ &= u_n. \end{aligned} \tag{12.8}$$

Therefore, when a re-ECN customer uses up quota zx_1, it is equivalent to being charged $u_n x_N$. In turn, when this customer optimises net utility $\Upsilon(x_N) = U(x_N) - zx_1$ it is equivalent to optimising $\Upsilon(x_N) = U(x_N) - u_n x_N$, which is identical to Kelly's optimisation. Therefore the customer's rate controller will choose an identical optimal bit-rate x_N^* such that

$$U'(x_N^*) = u_n. \tag{12.9}$$

It seems as if net utility is unaffected and the chosen optimal delivered bit-rate is unaffected whether $z < u_n$ or $z = u_n$. However, if $z < u_n$, to compensate for expected discards at the dropper the sender

[25] $z = u_n$ is strictly a third case, but it acts as a limit to either case, so for brevity we treat it within the first case.

has to send faster (which it can afford to do because it understates path congestion). While if $z = u$ the dropper discards nothing. The two are clearly not equivalent in practice, because the sender cannot know in advance which data the dropper will drop. Therefore there will always be more delay introduced while lost data is repaired, or more overhead will be necessary to send forward error correction (FEC) conservatively (see a similar argument in §12.1.1).

Overstated downstream congestion $z > u_n$. If the customer chooses to use more quota than is necessary to ensure traffic traverses the dropper without loss, nothing is gained and more quota is lost.

Therefore, through testing both cases we have proved that the customer's best strategy is always to choose $z = u_n$, rather than choosing a z that misrepresents u_n. This means the re-ECN framework can be oblivious to the transport while still assuring downstream congestion signal integrity—on condition that the customer's utility function is concave. □

Examples over a range of concave utility functions. We now give some examples to show that it is advantageous to maintain downstream congestion signal integrity for a whole range of transports—transports that would be chosen by users with a whole range of concave utility functions.

Functions of utility against delivered bit-rate x_N represent human judgements so they can only be determined empirically and they will not follow any simple mathematical form. However, after Mo & Walrand [MW00], we can parameterise a set of utility functions that might approximate different people's utility, while confining ourselves to a set of functions that are continuously differentiable for positive bit-rate x:

$$U(x_N) = \begin{cases} \frac{w^\alpha x_N^{(1-\alpha)}}{(1-\alpha)}, & \alpha \geq 0, \alpha \neq 1; \\ w \ln(x_N), & \alpha = 1. \end{cases} \tag{7.11}$$

This is called α-utility.[26] The parameter α represents the concavity of the utility function and, as utility is assumed concave, ($\alpha \geq 0$). The exception at $\alpha = 1$ fills the discontinuity in U that would otherwise result. w represents the weight of the utility. Both α and w are intended to be held constant over a flow of data, but they may be allowed to adapt slowly.[27]

We will consider the three cases where downstream congestion claimed by the source is correct, overstated and understated. The special case where $\alpha = 1$ will be dealt with separately later.

Correct downstream congestion ($z = u_n$)

In this baseline case, net utility ($\alpha \geq 0, \alpha \neq 1$) is

$$\begin{aligned} \Upsilon_{(=)}(x_N) &= \frac{w^\alpha}{(1-\alpha)} x_N^{(1-\alpha)} - z x_1 \\ &= \frac{w^\alpha}{(1-\alpha)} x_N^{(1-\alpha)} - u_n x_N. \end{aligned}$$

Differentiating, finding the bit-rate x_N that maximises net utility, and substituting back into this equation

[26] See also Eqn (7.11) in §7.7.1

[27] The denominator $(1-\alpha)$ and the exponent α of w merely simplify the form of the ultimate results.

gives optimum net utility in this baseline case:

$$\Upsilon^*_{(=)} = w u_n^{(1-1/\alpha)} \frac{\alpha}{(1-\alpha)}. \tag{12.10}$$

Overstated downstream congestion ($z > u_n$)

This case is nearly identical to the base case, except u_n cannot be substituted for z in the net utility formula:

$$\begin{aligned} \Upsilon_{(>)}(x_N) &= \frac{w^\alpha}{(1-\alpha)} x_N^{(1-\alpha)} - z x_N, \\ \Upsilon^*_{(>)} &= w z^{(1-1/\alpha)} \frac{\alpha}{(1-\alpha)}. \end{aligned} \tag{12.11}$$

Understated downstream congestion ($z < u_n$)

This case is also nearly identical to the base case, except x_N cannot be substituted for x_1 in the net utility formula, because the dropper reduces it by u_N/z:

$$\begin{aligned} \Upsilon_{(<)}(x_N) &= \frac{w^\alpha}{(1-\alpha)} x_N^{(1-\alpha)} - z x_1 \\ &= \frac{w^\alpha}{(1-\alpha)} x_N^{(1-\alpha)} - z x_N \frac{u_N}{z}. \\ \Upsilon^*_{(<)} &= w u_n^{(1-1/\alpha)} \frac{\alpha}{(1-\alpha)}. \end{aligned} \tag{12.12}$$

The three cases for $\alpha = 1$ can be calculated similarly. Then the gains in optimum net utility[28] from over or understatement of expected downstream congestion z are:

$$\begin{aligned} \Delta\Upsilon^* &= \Upsilon^* - \Upsilon^*_{(=)} \\ &= \begin{cases} w \frac{\alpha}{(1-\alpha)} (z^{(1-1/\alpha)} - u_n^{(1-1/\alpha)}), & z > u_n,\ \alpha \neq 1; \\ w \ln\left(\frac{u_n}{z}\right), & z > u_n,\ \alpha = 1; \\ 0, & z \leq u_n. \end{cases} \end{aligned} \tag{12.13}$$

Over their given valid ranges, none of these expressions are greater than zero. These examples support our proof of Theorem 12.2; that the re-ECN mechanisms at ingress and egress remove any incentive to misrepresent downstream congestion for anyone as long as their utility for bit-rate is concave.

If we use a normalised measure of net utility gain $\Delta Y^* = \Delta\Upsilon^* u_n^{(1/\alpha)-1}/w$, we can derive its slope as

$$\frac{d}{d(z/u_n)} \Delta Y^* = -\left(\frac{z}{u_n}\right)^{-1/\alpha}. \tag{12.14}$$

Therefore, for all α, this normalised net utility gain has slope -1 when $z \to u_n$ from above. Fig 12.5[29] plots normalised net utility gain against overstatement of downstream path congestion, for various values of concavity of utility α. When z is understated (the heavy horizontal plot along the horizontal axis to the

[28] Note that these values of net utility are meaningless except for comparisons between two values from the same utility curve. In economic theory, utility is generally considered ordinal, not cardinal. α-utility curves with $\alpha \geq 1$ do not even pass through the origin, and their numeric values are negative (in the $\alpha = 1$ case, both positive and negative). The value and sign of $\Upsilon_2 - \Upsilon_1$ is meaningful if they come from the same utility curve, but $(\Upsilon_2 - \Upsilon_1)/\Upsilon_1$ is meaningless.

[29] Cf. Fig 5.2 on p58.

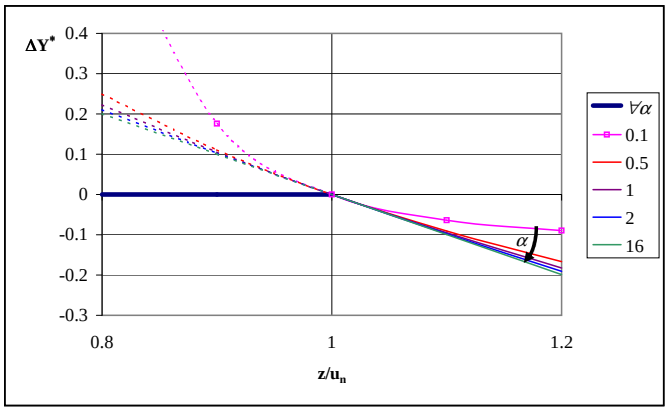


Figure 12.5: Normalised Net Utility Gain;
ΔY^* is plotted against overstatement of downstream path congestion z/u_n for various concavities of utility α. For $z/u_n < 1$, plots are shown both with the dropper (heavy flat line) and without (dashed plots).

left), it can be seen that there is no net utility gain relative to the case where $z = u_n$. Plots of normalised net utility gain without the re-ECN dropper are shown dashed above it, to show how the dropper removes any gain from understating z.

We must emphasise at this point that our proof of Theorem 12.2 is not intended to imply anything about whether the engineering of the re-ECN mechanisms is robust against gaming. For instance it does not imply that the mechanisms can remove incentives to misrepresent downstream congestion immediately, or against someone with no utility at all for transferring any bit-rate. It does however prove that the re-ECN mechanisms do not depend on how concave the utility function is, as long as it is concave, i.e. the re-ECN mechanisms proposed in this dissertation are indeed transport-oblivious. Put another way, this proves that in any circumstance where the proposed re-ECN mechanisms protect the integrity of congestion signals, they do so with a one-way incentive barrier—a barrier that no rate response to congestion can overcome.

Note that the phrase ‘no incentive to misrepresent downstream congestion’ is not as strong as ‘an incentive to correctly represent it’, which we have not strictly proved, although we have given a discursive argument why this is so. Our dropper design principle of ‘Proportionate Sanctions’ only removes any gain from understating downstream congestion, witnessed by the flat plot to the left of Fig 12.5—theoretically, a customer would be equally willing to sit anywhere along that flat line. But even the tiniest additional latency due to unnecessary dropper discards or the tiniest additional punishment beyond neutralising the gain from understatement will push a customer’s incentives towards stating downstream congestion correctly.

The uncertainty of knowing which packets might be dropped would be sufficient to tip this

balance—one-off Cautious credits per-flow represent a customer's willingness to pay to avoid this uncertainty. And an active sanction at the dropper against flows that had been negative for more than the minimum idle flow-state timeout would provide a very strong incentive not to stray into understatement. §7.3 on the principle of 'Proportionate Sanctions' gave two main reasons why neutralising any gain is preferable to punitive sanctions: i) infeasibility of punitive sanctions against cheap pseudonyms (i.e. flow IDs) and ii) the risk of the mechanisms being perverted to amplify third party attacks.

Note that the proof of Theorem 12.2 would actual hold for all convex not just concave utility functions. However, the economic optimisation on which the re-ECN framework is based is only proven for concave utility [KMT98]. Below we discuss how to handle applications for which utility can be expected to be convex over part of their range.

Inelastic Applications. It is convincingly argued that certain types of so-called inelastic application [She95] exhibit concave utility over their operating range of bit-rates, but below a certain bit-rate utility becomes convex.[30]

Flow admission control can be used to ensure inelastic flows are only admitted to the system when it is in their concave operating region. Congestion marking can determine the initial admission decision [GK99a], but then the flow is not expected to reduce its rate if congestion in the system subsequently moves out of its operating region. At least three strategies are possible for engineering such a system:

- Inelastic flows can be isolated in a separate traffic class, and admission controlled into the capacity assigned to the class by network-controlled gateways. This is the approach adopted[31] and being standardised by the IETF in its pre-congestion notification (PCN) working group [Ear09b, Ear09a]

- Inelastic and elastic flows can be mixed. If congestion rises so that inelastic flows would reduce below their operating range, admitted flows can remain while elastic flows adapt to the temporary shortage, but new inelastic flows will not enter the system. Our paper [JBM08] on a bulk congestion policer similar to that proposed in §11 analyses how it makes users sacrifice the bit-rate of their own elastic applications to preserve their ongoing inelastic flows, by creating cross-flow congestion as if it were real congestion.

- Alternatively, inelastic and elastic flows can share the same capacity but packets have separately identifiable classes. The marking of the two classes of packet is linked [GK02] so that both classes optimally share the available capacity on longer timescales, but the inelastic traffic is allowed to keep its admitted bit rate once it has entered the system and it is served with very low latency. The PCN architecture discussed above can be used to implement this approach [SEB+06].

The re-ECN incentive framework has been proposed as a way to allow pre-congestion-based admission control systems based on network gateways to scale to an internetwork operated by network providers that do not trust each other [Bri08b]. It uses exactly the same pattern of incentives as the

[30] Experiments on people paying to view videos in Lab conditions have shown that people do indeed have such convex-concave utility functions for video streaming, but unfortunately the experiment results remain confidential [HE02].

[31] In work by the present author as a co-author with others.

re-ECN framework proposed in the present dissertation, but the system is contained within gateways around the edges of the internetwork, which are equivalent (in incentive terms) to endpoints in our more general re-ECN framework.

Section Summary. To summarise this section, we have proved that the re-ECN system creates an incentive for bandwidth consumers to truthfully declare their expectation of downstream congestion, whatever their utility function if it is concave. And further, we have briefly introduced engineering mechanisms that allow the same congestion marking and incentive mechanisms to be used for inelastic applications—those applications characterised by utility functions that are not concave at low bit-rates.

12.3.2 Algorithm Complexities

This section brings together statistics on the complexities of each of the proposed re-ECN mechanisms. Figures for numbers of processor operations have been established merely by tracing through the unoptimised pseudocode.

1. Egress Dropper Complexity.

   Cycles per packet: Typ: 5, Max: 16 (non-compliant flow, Negative packet) plus 2 header field reads and a flow ID match;

   Extra cycles per new flow: 6;

   Storage per compliant flow: ~16B plus flowID;

   Total storage for the set of non-compliant flows: Two extra compliant flows' worth of storage.

   See §7.6.1 for details.

2. Extra Border Complexity.

   Cycles per packet: Typ: 1, Max: 14 (Positive packet) plus a header field read, and 1 extra cycle if packet picked for sampling. Note: all border processing can be in parallel to forwarding.

   Storage per border interface: 2 counters (32B?), plus the flow ID ranges to be sampled and two counters per sampled flow.

   §8.3 summarises additional border complexity a little less tersely.

3. Optional Extra Forwarding Element Complexity.

   Cycles per packet: Probably Typ 1, Max 2 for preferential drop; Marking Cautious should be possible by altering an existing test with zero extra cycles.

   Storage per interface: Zero.

4. Ingress Policer Complexity.

   The baseline ingress congestion policer would be similar in complexity to the dropper plus the same downstream congestion normalisation as the basic border mechanism above (without flow sampling) plus an additional bulk token bucket. Altogether (without even having written the pseudocode) this is likely to add up to the following:

Cycles per packet: Typ: 10, Max: 26 (Negative packet) plus a header field read;

Storage per flow: ∼16B plus flowID;

Storage per policed interface: 16B (token bucket) plus two extra flow's worth of storage.

12.3.3 Performance

Initial performance results from simulations of the dropper are given in §7.8. So far, too few runs have been attempted to make any definitive statements, but we have tentatively concluded that a hard-coded up-front credit of two or possibly three packets should prevent the dropper introducing all but a residual additional drop fraction for TCP flows in reasonable network environments. Dropper performance has been predicted analytically in §7.7. The other parts of the system are yet to be implemented.

12.3.4 Outstanding Vulnerabilities

Attack Model. This dissertation has attempted to subject the re-ECN system to an ambitious attack model in which potential attackers are either rational or malicious, on condition that the malice of network operators is bounded (defined in §8.1.2).

Caveats and Concerns. The least solid defence we have proposed is 'Sample-Based Congestion Volume Inflation' (§8.2.4). It is designed to remove the motivation for networks to attack each other using dummy traffic. At this stage, this idea is no more than an architectural direction, backed by some rationale and theory. Defending against the 'Dummy Neutral Background Load' attack depends on this direction being fruitful (§12.1.1). This attack aims to raise congestion costs for other users.

The attack 'Dragging Down a Spoofed Flow ID' (§7.5.3) is as easy (or hard) to mount as it already is to mount flow ID hijacking attacks. Addition of a re-ECN dropper to a network would add a new way to use this attack against the flows of others.

The need to start a flow with a Cautious packet to get it through the re-ECN dropper could increase the cost of reflection attacks against any servers routinely giving responses either as single packets or very short flows (e.g. DNS servers, see §12.1.3).

The flow time-out mechanisms of the dropper (§7.6.1) and a tentative proposal to use 'Covert Marking' as a signalling channel from policers to end-points (§11.3.1) have yet to be fully explored. These may reveal further vulnerabilities. The possibilities for attacks using legacy codepoints in the extended ECN field and using tunnels have not been exhaustively explored, although these issues have been thoroughly considered for most parts of the design.

Aside from these outstanding caveats, we believe we have shown that the proposed re-ECN mechanisms are otherwise robust against all the other attacks on the integrity of congestion signals identified in this dissertation.

Of course new attacks may be identified in the future. However, over the years, re-ECN has attracted considerable attention from researchers trying to break down its defences with new invented attacks. These have been generalised into the range of attacks enumerated in this dissertation. This is certainly not claimed as any form of 'proof of security'. However, even proofs of security are only as good as the deviousness of their attack model.

Better Defence against Attacks on the Existing Internet. We cannot claim re-ECN fully prevents distributed bandwidth flooding attacks (DDoS). However it should considerably raise the bar against them—the brief analysis in §12.1.1 estimates that a botnet army would have to be two or three orders of magnitude larger to sustain an attack of the force it can muster today without re-ECN in place. The Re-ECN system also shifts liability for the congestion cost of a flooding attack to the ingress network. It also throws likely attack traffic into sharp relief throughout the network path. This both provides a strong incentive for other active preventative measures, and provides the evidence needed for them to be deployed close to the root cause where they will be most useful.

The re-ECN system also provides additional protection against initial packet attacks (e.g. TCP SYN flooding—see §12.1.3) and 'State Keep-Alive' attacks (§12.1.3).

Part IV

In Closing

Chapter 13

Conclusions

13.1 Closing Arguments

Hypothesis 1: Congestion Signal Integrity

Proof. We will take Hypothesis 1 (p37) phrase by phrase.

"***The incentives of self-interested malicious economic entities can be aligned to assure the integrity of indications of downstream congestion in the packets of a connectionless simplex internetwork…***"

We can only prove this part of the hypothesis with strong conditions.

Lazy Removal of Dummy Traffic. In §12.3.4 we have outlined our concerns and caveats regarding outstanding vulnerabilities. If any of those issues are not resolved, including discovery of new successful attacks, the hypothesis falls. In particular, those concerns include the question-mark hanging over 'Sample-Based Congestion Volume Inflation' to align the incentives for networks not to attack each other at borders. Indeed the whole system must be considered riddled with concerns, given some parts still only exist on paper.

We should therefore add the form of condition suggested by Bauer & Faratin that: "*No network element before the first dropper after congestion can rely on the expected congestion declared for a flow.*" Wherever there are dummy traffic attacks in progress that have not been detected and removed, the integrity of congestion signals in their packets will not be assured.

Although this sounds like it kills the hypothesis, we can continue if the argument is acceptable that we can still assure signal integrity *wherever it is used*, with no need to assure it otherwise. All the information is available to detect and remove dummy traffic attacks at any node in the network. One could take the direction we have outlined to ensure that reasonable (lazy) effort would get rid of negative flows relatively quickly. Or, with greater effort, one could do much better. The above condition still strictly states the limit to congestion signal integrity, but only because the architecture deliberately allows an operator to avoid expending effort unless the problem warrants it. The architecture at least provides the information to test whether there is a problem and whether the effort expended has solved it.

Dynamics. If we take the liberty of assuming that all the above concerns prove unfounded, we have shown that the re-ECN dropper can remove the gain from misrepresenting downstream congestion. But the implemented algorithm only completely removes gain over a long period, not immediately (§7.7.2) . As soon as someone starts to gain from cheating, the system only starts to remove the gain increasingly strongly. This is sufficiently fast for some attacks (e.g. 'Flow ID Whitewashing') but not others (e.g. 'Stop Payment'). We had to allow a trade-off to avoid false hits being too harsh, by using moving averages (at least we found we could set the characteristic period of the EWMA to move fairly fast $a = 1/16$ or perhaps 1/32). This trade-off was ultimately necessary because re-ECN markings use the same unary encoding as ECN, which is a very slow signalling channel.

However, these short-run gains are only possible because we recommended, but did not implement, a more active timeout mechanisms. With a timeout mechanism, any flow persistently negative, even slightly, for more than the timeout (perhaps 1 sec) could very simply be subjected to arbitrarily high drop rates and, ultimately, complete blocking.

Thus any small gains are potentially limited to the timeout period. Attackers cannot exploit multiple timeout periods (or even one) without 'spending' a Cautious packet for the right to each timeout (only one timeout is available per flow). To gain, an attacker would have to be able to gain from the 'Flow ID Whitewashing' attack. But, with the addition of active flow state timeout, we have shown that isn't possible.

Slight additional punishment. In §12.3.1 we proved that the combination of the policer and the dropper gives self-interested entities 'no incentive to misrepresent downstream congestion', at least not in the long-run (and we have just handled short-run issues). We went on discursively to show that any additional punishment beyond neutralising the gain would turn this into 'an incentive to correctly represent downstream congestion', as long as the punishment increases with understatement of congestion. Timeout punishments alone would fit this description.

Scalable enforcement mechanism. §12.3.2 summarises the complexities of the proposed algorithms for the various components of the re-ECN system. The ingress and egress components exhibit sub-linear scaling with number of flows, but only just. Flow state is required for every active behaving flow, also held for a short timeout once they become inactive. Assuming there are some misbehaving flows, this represents sub-linear scaling with flows (just). From a practical angle, exhaustive flow-state is not required except at the outer edges of the internetwork. At every other trust boundary, only sampled flow state is required.

Additional per-packet processing can be truthfully described as minimal in all cases.[1] The most processor intensive component is the policer at 10 cycles per packet typical with a maximum of 26 cycles. Border elements require one extra cycle per typical packet (max 14). Importantly, border operations can all be in parallel to data forwarding. An optional addition to forwarding elements would require two cycles per packet maximum. For the parts that are yet to be designed in detail we assume our complexity estimates are reasonably accurate.

[1] Strictly any additional per-packet cycles make the system scale super-linearly with packet load.

"***. . . This can be achieved by only constraining aggregate downstream congestion-volume sent by each economic entity over time, without any dynamic congestion pricing to end-consumers, without any further constraints on transport behaviour and without any further constraints on the agents' freedom to distribute load across the internetwork, or across time.***"

Bulk Congestion Policing. The fact that re-ECN allows the design of bulk congestion policer in §11 satisfies all these conditions. The policer only constrains aggregate downstream congestion-volume over time. Theorem 12.2 in §12.3.1 proves the policer combines with the rest of the system to be transport oblivious. And the policer sets no constraint on where traffic goes, and no constraint on when it is sent as long as aggregate downstream congestion-volume over time fits into its fill-rate and peak-rate.

Flat Pricing. Within the proof of Theorem 12.2 (Transport Oblivious), we saw that, in the long run, proportionate sanctions at the dropper made Positive marked bytes at the ingress z equivalent to Negative at the egress u_n, even if Positive markings were under-declared. Therefore the bulk policer can be considered to give equivalent incentives to congestion pricing, but at the same time it can be fed by a constant stream of tokens that represent a flat charge, not a dynamic one.

Thus, Hypothesis 1 is not proven unconditionally and conclusively. But with all the above strong conditions concerns and caveats, it can still be valid. □

Note that the strong conditions required on this proof could be considered as evidence that the hypothesis is too ambitiously worded. However, it is preferred to preserve the hypothesis as an enduring statement of the goal (*quod errat*) and continue to try to improve the re-ECN system to soften the caveats, or otherwise find a concrete refutation of the hypothesis in the future.

Hypothesis 2: Welfare Maximising Allocation

Proof. Again we will work through Hypothesis 2 (p37) phrase by phrase:

"***With a competitive market and under Assumptions 3.1 & 3.2 (p37), incentives of all parties can be aligned so that the system produces the welfare maximising allocation of resources, under all the conditions of Hypothesis 1.***"

If Hypothesis 1 holds, Positive marked bytes at the ingress z are equivalent to Negative at the egress u_n. The assumptions referred to in this hypothesis are equivalent to Kelly's assumptions in [KMT98]. We need the additional assumptions:

Assumption 13.1. *All network operators on the Internet choose to use the re-ECN bulk congestion policer, or an equivalent.*

Assumption 13.2. *The consumer perceives paying for tokens that pay for congestion as equivalent to paying directly for congestion.*

Then the incentives in the re-ECN system are equivalent to those in Kelly's SYSTEM. Then a Welfare Maximising proof for the re-ECN system follows by equivalence with Kelly's welfare maximising proof for users with concave utility [KMT98].

Therefore, under these assumptions, the shares of network resources that users will choose to use in the re-ECN system will be the welfare maximising allocation. □

Note that Kelly's proof of welfare maximisation requires the condition of concave utility, which is a sub-set of the range of utility functions our proof of Theorem 12.2 holds for. Therefore, unlike the proof of our first hypothesis, welfare maximisation only applies for elastic traffic.

It is unlikely that Assumption 13.1 will hold, as many operators will prefer to charge by value rather than cost, by discriminating prices for different types of session. This situation will prevail wherever competition is weak enough. The re-ECN framework is designed to merely provide congestion information to reveal marginal costs as a floor for pricing—it is not the designer's role to insist that pricing should track congestion costs, but pricing must *be able to* as competition intensifies. Of course, network operators should heed the corollary of Hypothesis 2; that any other pricing scheme will lead to resource allocations that do *not* maximise welfare. Or put another way, a network operator can do no better to satisfy its customers than to use congestion policing.

In §12.3.1 we also outlined how flow admission controls can prevent inelastic flows being admitted when the shadow price of the system is outside their region of concave utility. We are not claiming that these mechanisms would take the system to the welfare optimum for users with both concave and convex utilities. But if the shadow price did not change drastically for the duration of each flow, it would be fairly close.

Also note that the condition of a competitive market is necessary to ensure that congestion signals tend towards shadow prices for capacity; a representation of the marginal cost of the capacity needed to alleviate the congestion.

13.2 Re-ECN Limitations and Further Work

Architectural Issues. The need for a end-point flow ID sub-layer in the Internet architecture needs to be justified by and incorporated with wider concerns than just resource sharing.

The re-ECN architecture does not allow self-congestion beyond the egress of the provided internetwork to be discounted (except by tunnelling).

The question of how an egress dropper can allow for packets that arrive mid-flow due to reroutes (§7.3.3) has been addressed but full assessment remains outstanding. This could be somewhat of a problem for multi-homed receivers or mobile receivers during hand-overs.

Although the high level implications of re-ECN on routing & traffic engineering have been articulated, they need to be more deeply considered (see [BCSJ04, §4]).

An architecture for proxying re-ECN senders and/or receivers needs to be defined (see §12.1.4).

Management diagnosis of misbehaving / malconfigured policers / droppers will need to be considered.

Accountability for causing congestion has only been addressed for the unicast mode of communications. For multicast and anycast, superficially at least, the accumulated congestion information provided by simple ECN is sufficient, because control is at the receiver.[2] However, receiver control is only per

[2]But current multicast forwarding duplicates congestion markings, massively 'double-accounting' for each congestion event. This could be solved with the form of multicast forwarding proposed in an expired Internet Draft and associated Technical Report [BC01a, §5] that I co-authored with Jon Crowcroft. However this would require different and more complicated forwarding

session while sender control is per packet. So one really needs sender accountability for packet rate, with receiver accountability only at the session level. If one wanted to hold the sender accountable with re-feedback, the expected congestion a multicast packet will cause should be the sum of all the congestion events in every branch. One could achieve this using the unary congestion encoding scheme in [BC01a, §5], which already picks an acker for each separate instance of a congested link. But a mechanism to ensure the integrity of such congestion signals and to meter downstream congestion at intermediate points such as borders are very much open issues.

Re-ECN Protocol: Network Layer. There are outstanding questions to answer if flow-state congestion signalling is to be considered for full inclusion in the re-ECN protocol (§10.1).

The tentative proposal to include a signalling channel from policers to the transport via covert marking needs to be further considered (§11.3.1).

Re-ECN Transport Protocol Extensions. The effect of the current re-ECN dropper design on transports with infrequent feedback needs to be considered in depth; even delayed ACKs in TCP present some problems. (Guidelines on re-feedback design for a range of transports other than TCP are included in [BJMS09a], but no detailed design, performance analysis or testing has been done.)

More comprehensive end-to-end integrity checks need to be defined in the presence of untrusted receivers (§12.1.4).

Protocols for end-to-end transfer of congestion quota need to be designed.

Economic & Security Analysis. Precisely what the dropper should do with outstanding credit or debit when a flow times out (§7.6.1) requires further consideration. It seems to be a matter of policy, but it also has a more general economic interpretation.

Fuller analysis of the effect of re-ECN on internetwork competition and termination monopolies is required (§12.1.2), taking account of any ability to transfer congestion quota end-to-end.

Incremental & partial deployment scenarios have been proposed in an Internet Draft [BJMS09b] and a workshop paper has been prepared to lay the groundwork to analyse deployment incentives [Bri06]. But a fuller incremental deployment plan and analysis is needed.

Opportunities to exploit legacy fields in the protocol and legacy behaviours have often been included in the economic & security analysis of re-ECN, but not always. Congestion signal integrity needs to be assured in a partially deployed setting. And the extra opportunities for illegal protocol transitions with legacy codepoints need to be considered.

Evaluation. Some of the initial dropper simulations need to be repeat tested to derive confidence intervals. Further interpretation of the results (§7.8.2) is needed, with possible further iterative design and even re-consideration of architectural choices.

The sample-based congestion volume inflation ideas (§8.2.4) need to be developed further, and put to the test.

The full precise downstream congestion meter formula (§8.2.7) needs to be implemented and fully stress tested.

---

implementations.

Example ingress policer designs need to be tested against the rest of the system.

Congestion Control Evolution. Numerous open research issues remain in congestion control research, particularly now the goal can be more usefully stated as congestion accountability (not TCP-friendliness) as well as prevention of congestion collapse. An attempt to document open research issues is currently in progress [WPSB09] as a work item of the Internet Congestion Control Research Group (ICCRG) of the Internet Research Task Force (IRTF).

Contractual Transparency. The requirement for contract transparency remains the most unsatisfactory aspect of the present research. It is solved in all but a psychological respect. It is certainly now possible to apply a simple tiered flat pricing scheme to a good that is under the full control of the consumer but the good, expected traffic congestion, is not a natural one for consumers to understand.

The consumer's software is in full control of its declarations of congestion likely to be experienced by each packet. So certainly this good is immune to whatever unpredictable congestion occurs. But most experts, let alone consumers, in the industry aren't yet even aware that congestion has a very specific definition. Congestion is perceived as a vague state of a system, possibly even binary—either congested or not. The idea that congestion can be precisely counted is unfamiliar to people.

This weakness could possibly be solved through education, just as consumers were educated about the previously unfamiliar concept of data volume and bytes before volume capping was introduced around 2001. However, this is a rather ambitious hope.

Nonetheless, UK road tax licenses are now priced based on the volume of an economic externality of the vehicle's performance that no human can even sense ($mgCO_2/lt$ of fuel), so perhaps, as long as the good being priced is generally recognised as valid, it doesn't matter if it's not tangible.

13.3 Material Contributions

Material contributions are divided between those directly relevant to this dissertation and those that have provided background context.

13.3.1 Direct contributions

Accountability for congestion with freedom: A protocol has been invented[3] that enables[4] an ingress access network to constrain the overall congestion an attached data sender can cause anywhere in a connectionless internetwork, without any further constraints on the user's freedom to distribute load across this internetwork nor across time. This contribution is summarised in the recent workshop publication "*Policing Freedom to Use the Internet Resource Pool*" [JBM08] (co-authored[5]).

Accountability for congestion robust to gaming: Robustness to gaming can never be proven conclusively, but we have at least outlined the re-ECN system's robustness in the face of currently foreseen attacks, and stated the limits of its vulnerability.

[3] With co-inventors.

[4] Subject to further successful performance experiments.

[5] I contributed the overall ideas the structure and most of the text, except the central section analysing the effect on flow congestion controls and cross-flow interactions.

Simple pricing: The tension between the irresistible economic logic of usage-sensitive pricing and the immovable consumer desire for simple and predictable pricing [Odl97] has been resolved. Minimal constraint can be applied to consumers to align their incentives within a tiered flat rate pricing plan, without dynamic congestion pricing. Nonetheless, 'sender-pays' dynamic congestion pricing with simple bulk accounting is now possible as well—at any trust boundary—so it is free to develop in wholesale and interconnect markets if desired.

Necessary but sufficient mechanism: The alteration to feedback transparency at the network layer claims to be no more than the minimum necessary to offer generic support to a wide range of higher layer resource sharing approaches—it solely reveals expected rest-of-path congestion. The mechanisms suggested to ensure truth-telling and provide sender and network accountability are not embedded in the network layer—they are merely optional *applications* of the protocol mechanism.

Identified fundamental problem: The problem has been narrowed down to a lack of information about quality that is necessary for efficient contracting: expected downstream congestion.

Localisation of resource accountability: By adding congestion accountability to self-contained datagrams, identification of the entities local to each trust boundary is sufficient to create strong chains of precise accountability, without requiring a global identity infrastructure. Either party at a trust boundary can hold the other accountable for causing congestion—whether A forwarded too much traffic, or B provided too little capacity.

Identified the flow rate equality problem as a nonsensical distraction: "*Flow Rate Fairness: Dismantling a Religion*" [Bri07b] (ACM CCR journal) and [Bri07c] (individual IETF Internet draft) explained far more clearly and simply than before (indeed bluntly) why flow rate equality is a non-goal and why fairness should be measured in terms of congestion volume in order to ensure fairness on a global scale between different local definitions of fairness. "*Problem Statement: We Don't Have To Do Fairness Ourselves*" [Bri08d] (individual IETF I-D in progress with co-authors[6]) explained the mechanisms by which flow rate equality is leading the Internet into a highly suboptimal state, backing up the assertions in "*. . . Dismantling a Religion*" with more concrete evidence. This in turn has led to positive coverage in the technical media, including being invited to write an article on the subject (and on re-feedback) for the Dec 2008 issue of IEEE Spectrum Magazine [Bri08c], and invitations to present the work in several international fora.

Placed solution in commercial context: "*Commercial Models for IP Quality of Service Interconnect*" [BR05] (BT Technology Journal with co-author[7]—presented in two international industry fora).

Articulated solution, rationale and evaluation: "*Policing Congestion Response in an Internetwork*

[6] Co-authors contributed considerable editing, restructuring & reviewing

[7] Co-author contributed editing & reviewing

using Re-feedback" [BJCG+05] (ACM SIGCOMM conference paper with co-authors[8]) described the solution for any abstract connectionless internetwork, and for the Internet specifically.

Full protocol specification: "*Re-ECN: Adding Accountability for Causing Congestion to TCP/IP*" [BJMS09a] (individual IETF I-D in progress with co-authors[9]—presented to IETF six times).

Applicability & rationale for re-ECN: "*Re-ECN: The Motivation for Adding Accountability for Causing Congestion to TCP/IP*" [BJMS09b] (individual IETF I-D in progress with co-authors).

Enabled both open and closed models to interwork: The bulk and per-flow policers described in the "*re-ECN*" I-D [BJMS09b, Appx B] enable open and closed models respectively, and the border arrangements allow both models to fully interwork. Thus the tussle between service-oriented networks and open Internet access can be fought out at run-time without losing the value of full interconnectivity, because the service-oriented networks can protect their interests against excess congestion caused by traffic from their open neighbours (and *vice versa*).

First design of scalable internetwork admission control: "*Emulating Border Flow Policing using Re-ECN on Bulk Data*" [Bri08b] (individual IETF I-D in progress—presented twice) uses re-feedback with infrequent congestion feedback at reservation refresh signalling time-scales to create incentives for network operators to admission control traffic that would otherwise cause 'pre-congestion' in other operators' downstream networks.

First know treatment of DDoS as a congestion policing problem: "*Using Self-interest to Prevent Malice; Fixing the Denial of Service Flaw of the Internet*" [Bri06] (paper for Int'l Workshop on the Economics of Securing the Information Infrastructure).

Outlined incremental deployment incentives: "*Using Self-interest to Prevent Malice...*" [Bri06] outline the incentives for initial deployment, incremental adoption and the convex increasing incentives towards complete deployment. Also outlines the strong incentives to deploy other DDoS solutions. The "*re-ECN Motivation*" I-D [BJMS09b] also outlines incremental deployment issues and incentives.

Patent filings and gifts: I have filed six patent applications covering the present research with co-inventors[10] (under BT's ownership). All are now published and have so far survived searches. The primary patent of the re-feedback mechanism was recently granted in Europe, the others remain pending. Nonetheless, the present author persuaded BT to gift[11] free of royalties any aspect necessary to comply with our IETF standards contribution (the wording of BT's IPR declaration takes precedence if there is any conflict with this wording) [Orm05].

[8] Co-authors contributed co-invention, text on TCP policing and conducted & documented performance experiments, which I designed except for the co-author's choices of topology & traffic models

[9] Co-authors finessed protocol design and contributed appendices on policing & edits throughout

[10] The nub of the re-feedback idea was the result of a truly collaborative discussion between the co-inventors, but I originally laid out the problem space and developed the newly formed idea to reconcile it with the structure of classic feedback.

[11] The most challenging aspect of this whole endeavour!

13.3.2 Background contributions

The Market-Managed Multi-service Internet (M3I) project: [12] A medium-sized (EUR3.7M) EC Information Society Technologies Fifth Framework project that I initiated and led, to take up Kelly's work and build practical network controls and pricing schemes around it. "*A Market Managed Multi-service Internet (M3I)*" [BDH+03] (Computer Communications journal paper with many co-authors) summarises the project.

M3I Architecture: Principles & Components: "*Market Managed Multi-service Internet: Architecture Pt I; Principles*" [Bri02a] and "*... Pt II; Construction*" [Bri02b] (technical reports).

Split-edge pricing and end-to-end clearing structure: "*The Direction of Value Flow in Open Multi-service Connectionless Networks*" [Bri00, Bri99b, Bri99a] (technical report combining two papers, one for the Int'l Conf on Telecoms & E-commerce, the other an invited paper for the Int'l Workshop on Networked Group Communications).

Extending ECN and its economic effects to protocols below the network layer: "*Explicit Congestion Marking in MPLS*" [DBT08] (co-authored IETF Proposed Standard RFC), "*Layered Encapsulation of Congestion Notification*" [Bri09] (IETF I-D accepted as working group business—in progress), "*Service Differentiation in Third Generation Mobile Networks*" [SBS02b] (co-authored Quality of Future Internet Services (QoFIS) int'l workshop paper) and "*Economic Models for Resource Control in Wireless Networks*" [SBS02a] (co-authored Personal, Indoor and Mobile Radio Communications (PIMRC) int'l conference paper).

Applying policy control to congestion control: "*Market Managed Multi-service Internet: Pricing Mechanisms; Price Reaction Design*" [BDT+00] (main author of technical report).

Fixing important but detailed aspects of congestion notification: "*Byte and Packet Congestion Notification*" [Bri08a] (IETF I-D accepted as working group business—in progress) and "*An Open ECN service in the IP layer*" [BC01b] (expired co-authored individual IETF I-D used to finesse the proposed ECN standard).

Synthesising admission control from congestion notification: "*Pre-Congestion Notification Architecture*" [Ear09b] (contributor to IETF working group I-D in last call for Informational RFC status), "*Marking behaviour of PCN-nodes*" [Ear09a] (contributor to I-D on the IETF standards track to standardise virtual queue congestion marking), "*Guaranteed QoS Synthesis for Admission Control with Shared Capacity*" [SEB+06] (co-authored technical report), "*Guaranteed QoS synthesis - an example of a scalable core IP quality of service solution*" [HBC05] (co-authored BT Technology Journal article), "*An edge-to-edge Deployment Model for Pre-Congestion Notification: Admission Control over a DiffServ Region*" [BES+06] (co-authored individual IETF I-D in progress).

[12] www.m3i-project.org

13.4 Concluding Remarks

This concluding section aims to bring out the concepts contributed during this period of doctoral research.

Architectural and economic: A primary economic contribution has been to highlight from Kelly's work the importance of congestion-volume as a metric for trading bandwidth usage that is both location and time independent; unlike bit-rate or volume, congestion-volume represents the same cost wherever and whenever it is used in a competitively provided internetwork. It seems many missed these subtle but pivotal insights in Kelly's model, which have been instantiated in the bulk congestion policer of §11: location independence allows the shares of flows traversing all different resources in the internetwork to be controlled in one bulk mix in each policer; and time independence allows spending to be shifted back and forth in time using a token bucket buffer.

Another contribution (both economic and architectural) has been to follow a trail of multidisciplinary research to expose and articulate a major structural problem with the Internet architecture—an unusual form of information asymmetry where each of the providers in the value chain cannot determine the quality of their own product whereas the end-consumer can. Theoretically, this information asymmetry could in turn cause market failure, where all of quality, price and investment decline. In the case of the Internet, rather than a market failure, it seems to be causing network providers to violate the architecture that is causing the problem. This in turn is causing further problems as the resulting mess ossifies the ability of the Internet to evolve.

To solve this problem a novel feedback pattern called re-feedback has been proposed. It induces buyers to reveal the quality of the product to providers, both at the edges of the network, and at the borders between networks. It enables self-contained datagrams to carry an expectation of the characteristics of the rest of the path. This reveals information that could be used to solve the information asymmetry problems of packet networks. A way called re-ECN has also been invented to deploy this idea without having to change the forwarding elements of the Internet, through a protocol.

Re-feedback had the potential to align the incentives of all the stakeholders on the Internet to truthfully reveal the previously hidden price/quality information. The primary task of this dissertation has been to establish whether this is so for the much more challenging case of re-ECN where re-feedback ideas have to fit into the remaining one bit of space left in the IP header. The arguments in §13.1 that attempt to prove the hypotheses show that this has been tenuously proven, with strong caveats, concerns and conditions. In truth, it remains inconclusive. It is likely to remain inconclusive unless tested on a real internetwork against truly motivated attackers, although a lot more experimental work can be done.

However, on a more positive note, no clear flaws or vulnerabilities exist in the design, in the sense that none of the proposed attacks have no defence, or at least no direction in which further work can proceed to establish a possible defence. Given the severe constraints on header space and the highly ambitious attack model, this is certainly a nontrivial achievement.

A particular contribution is a policer design that represents a desirable contractual proposition for network operators. The proposed bulk congestion policer has the potential to allow a flat fee contract to

be offered to any size Internet consumer, while at the same time aligning their incentives (and in turn the incentives of their application developers) to take account of the congestion costs they are causing others. In Odlyzko's words, this resolves tension between the irresistible force of usage-sensitive pricing and the immovable object of consumer desire for simple predictable pricing.

Also, the policer sets no particular constraints on transport behaviour, thus potentially enabling easy evolution of new behaviours without having to ask permission of the network. A proof of this transport-oblivious property has been provided.

However, although oblivious to transport behaviours, we reluctantly could not make the re-ECN system oblivious to flow IDs. The only way to test for cheating is to look for flows that consistently under-declare expected congestion relative to actual. This can be tested locally, so it does not need to know where the flow is coming from or going to (therefore no need to rely on push-back), but it does need to match packets to consistent flow IDs and hold flow state. Knowing that the shared fate Internet design principle advised against this, we have designed a fully fledged flow ID sub-layer into the architecture, based on partial soft-flow-state, with its own flow-state congestion signalling.

Along the way, a few other architectural insights have been articulated and incorporated into the design:

- Datagram resource accountability: Rather than holding end-point identifiers to account for resource usage (infeasible globally), holding the datagrams themselves to account enables accountability to transfer from one party to the next across each contractual boundary between networks in turn.
- Principled adherence to the use of explicit in-band signalling for all congestion-related functions, to decouple the congestion system from any reliance on reverse reachability semantics of flow-IDs;
- Create a cost for using pseudonyms: Given flow IDs are zero cost pseudonyms, if flow IDs are to be punished, a transport must be made to invest a small cost in using each new ID.
- Proportionate sanctions: A policing mechanism embedded at the lower layers should take great care to only neutralise any gains as a minimum architectural contribution, not to over-punish misbehaviour. Otherwise the amplified punishment could be turned against others by spoofing their identity.
- Bufferless border control: Traffic should not have to be held back in a buffer while it is tested for compliance at high speed (e.g. photonic) interfaces. The system should be able to work with virtual queues and minimal or zero actual buffering.

Myths Slain: Some long-standing fallacies have been dismantled.

- Flow-rate equality & TCP friendliness; for two main reasons:
  - Fairness must be between economic entities not arbitrarily chosen pseudonyms;
  - Fairness must incorporate a time dimension.

- Per-flow policing, for four reasons:
  - Particular transport behaviours should not be embedded in the network (see 'transport obliviousness' above);
  - Flows can split and follow stepping stones to game the system;
  - Sampling cannot be used as a deterrent when a flow can switch pseudonyms as soon as it detects it has been caught;
  - And fourthly, given flow-rate equality is a meaningless goal, rate comparisons between flows are just as meaningless anyway.
- Flow isolation harmful: In our most recent publication, 'Policing Freedom' [JBM08], we argue that flow isolation (a goal of WFQ etc.) makes traffic that would be willing to shift in time unaware that others want it to—it muffles congestion signals that would otherwise allow better re-allocation of the resource pool. Providing signals of incipient congestion without actually introducing any impairment is more useful.
- RED: Three aspects have been found suspect, the last two of which open up major DoS vulnerabilities:
  - Uniformly distributing spacing of marks in the aggregate just wastes lots of valuable interface cycles, because they becomes geometrically distributed again within each flow;
  - Packets should be marked independent of their size;
  - The goal of 100% drop above a threshold queue size leads to forwarding absolutely nothing and gives unresponsive flows a huge advantage;

Modelling: The components of the re-ECN system have generally been designed for provability—to meet principled objectives. The objectives are sufficiently non-arbitrary that the results of *initial* experiment so far have validated the models fairly closely.

A model of the distribution of congestion events within a window has been created. An initial experimental proof of its validity implies it could be very accurate.

Algorithms: Compact algorithms for metering and manipulating unary encodings have been developed. One in particular to meter precise downstream congestion is rather pleasing. It outputs the unary encoded product of a unary encoded sum and difference (which itself contains a quotient), using only adds, compares and shifts on a small minority of packets (maximum 14 cycles on a Positively marked packet).

Generalisation & Prospects: This dissertation has focused on policing downstream congestion-volume as a way to encourage congestion responsive transports without restricting their freedom. If we really have solved this problem, we have also provided a basis for at least two other 'applications':

- Mitigating bandwidth flooding and initial packet attacks;

- an extremely simple quality of service mechanism that naturally 'just works' as it is extended to multiple domains, with surrounding mechanisms for security, monitoring, pricing, mobility, multihoming etc. thrown in for free.

This dissertation has discussed the prospects for mitigating flooding a little, but it has not particularly explored the applicability of the work to QoS. However, it is certainly not to be dismissed.

Simplicity? Reflecting back on this research endeavour, an appropriate question to raise is "Has the initial simplicity of the idea been lost as defences have been added against each new attack?" The sheer length of the dissertation in order to accommodate the discussion of each attack makes the outcome feel complicated. But, on reflection most of the space is used to prove or show that the basic mechanisms are robust enough against each attack—very little has been *added* to counter each attack:

- The need to check individual flows for negativity, although always recognised as necessary in some form, is the least satisfactory aspect of the whole scheme. Compared to Kelly's simple charging of ECN marked bytes to the receiver, this clearly adds unwelcome complexity. But, in return, operators can fully align incentives without all having to conform to a single dynamic pricing plan—that few, if any, customers would accept. And applications do not have to ask permission of the network to behave in novel, unexpected ways. The core of the hypothesis is that it is worth striking this Faustian bargain;

- The addition of flow state on middleboxes brought with it the need to manage potential memory congestion. But it was recognised that the facility to manage end-point flow state congestion was a missing piece of the Internet architecture anyway. So memory congestion control has been added for both middles and ends, with hardly any more complexity than would have been necessary to add it to end-points alone.

- Sampling flows at borders to introduce an inflation factor for negative flows is probably the most complicated (and still unproven) addition;

- The introduction of Cancelled packets created extra potential attack possibilities that all had to be checked through. But, overall, this act probably removed much more complexity than it added;

- The algorithm to meter precise downstream congestion rather than using the simple difference between volumes of Positive and Negative packets certainly adds complexity, but the resulting algorithm can hardly be called overly complicated;

A summary of the computational complexity of each part of the re-ECN framework is provided in the concluding section (§12.3.2) of Part III.

Perhaps accusations of complexity can be rebutted, but the word 'brittle' seems to justifiably apply to the end result. The incentives seem to balance on a knife-edge, and the mechanisms seem to *only just* work. But this is unsurprising. The aim was to balance maximum freedom against minimum accountability, all with minimum complexity. The chosen method was to design at the knife's edge—to probe

the limits—not to recommend that production networks operate at these limits. In practice, implementers should have cycles spare to beef up the incentive mechanisms or add their own special features.

Final Words. In the long term, all that can be hoped is that the main contribution will be seen as having identified the problem. Even if there's a better way to solve the problem than re-feedback, then it will have been worth it. But, on reflection, it is quite incredible what can be achieved by judicious use of one extra bit in packet headers—and how much can be written about it.

This dissertation set itself ambitious goals. More was bitten off than chewed, but enough was chewed to make it very long and very late. So I shall stop.

Appendix A

Design Alternatives

A.1 Mid-Flow Dropper Algorithm

It is possible for the egress dropper to bound the amount of flow state it uses by monitoring the packet stream for packets marked Negative and randomly picking one every so often. It can then create flow state for the flow from which the packet was selected, with a higher chance of picking flows causing a higher volume of congestion. It would then run the routine listed below that we call `newRecentBal()`. It is similar to `newBal()` used in §7.6.1, but it doesn't maintain or check the lifetime balance.

Instead of remembering the credit a flow might have given when it started, `newRecentBal()` makes an arbitrarily conservative allowance for round trip delay to allow Positive marks to catch up with the Negative ones they should balance. It doesn't act on incoming Negative marks, but instead stores them for a short fixed period—long enough to allow for all reasonable round trip times, e.g. 1 sec. After the fixed period, the dropper inputs the delayed mark as if it were applied to the next incoming packet.

The pseudocode for `newRecentBal()` is given below. The constant `rttAllce` determines how long to buffer Negative marks (it may be possible to implement a fixed delay buffer more efficiently with hardware support).

The flow state of each sampled flow no longer stores the lifetime balance of the flow `V` or the maximum packet size `smax`, but instead it holds pending Negative marks in the buffer structure `negvBuffer`. The functions `enQueue()` and `deQueue()` add data to and remove data from opposite ends of the buffer. The function `readQueue()` reads the oldest data from the buffer without dequeuing it.

```
/* Maintain flow congestion balances
 in the fState flow state structure.
 The parameter s is the packet size.
 */
 newRecentBal(s, fState) {
    eecn = readEECN(packet)
    /* Buffer NEGV & CANC marks
     and wipe as if not marked */
    if (eecn == NEGV || CANC) {
        if (eecn == NEGV) {
            eecn = NEUT
        } else {
            eecn = POSV
```

```
        }
        /* Store time-stamp in delay
         buffer */
        enQueue(timeNow(), negvBuffer)
    }
    /* If a buffered mark older than
     rttAllce, treat as if marked*/
    if (readQueue(negvBuffer)
            >= timeNow() - rttAllce) {
        deQueue(negvBuffer)
        switch(eecn) {
            case POSV:
                eecn = CANC
            case CAUT || NEUT || CU:
                eecn = NEGV
            /* Otherwise no action */
        }
    }
    if (eecn == POSV || CAUT) {
        z += a*s
    } elseif (eecn == CANC) {
        z -= a*z
        u -= a*u
    } else {
        /* NEGV, NEUT or CU */
        fState = probDrop(packet,fState)
        if (eecn == NEGV) {
            z -= a*z
            u += a*s
            u -= a*u
        }
    }
    return(fState)
 }
```

The following pseudocode would be used to maintain recent state of sampled flows with the help of `newRecentBal()`.

```
/* maintainSampleFlowState()
 Maintain flow state in fState structure
 */
 foreach packet {
    s = readLength(packet)
    eecn = read EECN(packet)
    flowID = readFlowID(packet)
    fState = matchFlowID(flowID)
    if (fState != NULL) {
        /* Existing flow */
        fState = newRecentBal(s,fState)
        if (z >= u) {
            /* Compliant status flow */
            lastGoodTime = timeNow()
        }   /* else Remand status
            so lastGoodTime unchanged*/
    } elseif (eecn == CAUT || POSV) {
        /* New Compliant flow */
        allocate(fState)
        fID = flowID
```

```
        u = 0
        z = 0
        r = -1
        fState = newRecentBal(s,fState)
        lastGoodTime = timeNow()
    } elseif (eecn == Not-ECT || ECT(0)
                              || CANC) {
        /* LEGACY: forward unimpeded */
    } else {
        /* New or old misbehaving flow
         set status to BULK */
        fState = BULK
        /* update balances of BULK
         and probabilistically drop */
        fState = newRecentBal(s,fState)
    }
}
```

A.2 Precise Downstream Congestion Meter Algorithm

The algorithm presented below seems naïve relative to that in §8.2.7 that was developed later. However, the algorithm below has been fully implemented and more thoroughly tested.

We developed this original algorithm to implement the accumulation of congestion-volume based on Eqn (8.6) on p162 avoiding using any multiplication or division operations. Both parts of the max function are implemented simultaneously, so that they both accumulate downstream congestion-volume in real time each using one of the alternative parts of the formula. Then the maximum of the two results can be used. This discourages the upstream network from cheating, because its dominant strategy will be to try to keep $z/c = u/y$, which should be the case for random congestion marking.

Given the algorithms approximate the true inflation factors by sampling, taking the maximum of two approximations could introduce a persistent positive bias that would disadvantage even an honest network. We tested the algorithm measuring downstream congestion with stationary but random congestion at 1% and 97% neutral packets and no malicious packet marking. We found that, over 40 samples of 100M packets, the error of either algorithm had a mean and standard deviation of about 0.005% and 0.02% respectively. Therefore, for an honest user, taking the maximum of the two will not disadvantage an honest user. However, the experimental conditions were fairly ideal with no variation in congestion. A more rigorous experiment would have to be conducted to be certain that taking the maximum of two alternate formulae would not disadvantage honest networks.

Below we give a pseudo-code algorithm for just Eqn (6.4). C source is also available. The algorithm for the other formula is nearly identical. It works by accumulating $z_i - u_i$ as normal, but also adding an additional sample of $z_i - u_i$ taken over c bytes out of every z. Remainders after rounding are always preserved by using them as initial values for the next round.

```
int V=0;         /* uninflated downstr
                    congestion-volume*/
int Vc=0; /* inflation increment for
             downstr congestion-volume*/
```

```
int zc=0, uc=0, rc=0; /* increments of
                              z, u & c */
#define SI MAX_MTU /* sample increment*/
int sic=SI;  /* sample increment index*/

int zs=SI;          /* last sampled zc*/
for every packet {
   s = readLength(packet);
   if readEECN(packet) == POSV {
       zc += s;
       V  += s;
   } elseif readEECN(packet) == NEGV {
       uc += s;
       V  -= s;
   } elseif readEECN(packet) == CANC {
       rc += s;
       if (rc > zs) {
           zc = 0;
           uc = 0;
           rc  -= zs;
           sic += SI;
           once = 1;
       }
       if  (sic > 0) {
           sic -= s;
       } elseif (once == 1) {
           zs  = zc;
           Vc += zc - uc;
           once = 0;
       }
   }
}
```

Whenever a positive or negative packet arrives, its size is added or subtracted from the running total of downstream congestion-volume, V. But another running total is maintained of the amount of inflation Vc required on top of this downstream congestion-volume.

Two incremental variables zc & uc are also maintained to hold how much positive and how much negative volume has arrived since they were last zeroed. We will start the explanation from when they are both zeroed (they are always zeroed together). While they are gradually incrementing as positive and negative packets arrive, the algorithm keeps track of how many bytes of Cancelled packets are arriving in a third incremental variable rc. As soon as more than an arbitrarily set amount MAX_MTU of Cancelled bytes has arrived, the difference between the two incremental variables for positive and negative bytes can be added to the amount of inflation required.

A snapshot of the incremental positive variable zc is also stored in zs, which will determine how long the algorithm ignores positive and negative arrivals before it zeroes them again and the cycle repeats.

The two incremental variables go on accumulating positive and negative volume, but once the snap shot has been taken, their values are no longer used. The incremental variable for Cancelled volume also continues to increase and when it finally exceeds the snapshot value taken of positive congestion zs, the incremental variables are zeroed and the sampling period starts again.

This mechanism ensures that the whole cycle repeats every zs bytes of Cancelled packets, where

`zs` was the incremental volume of positive bytes at the snapshot. Then it doesn't actually matter how many bytes there are before the snapshot, it just has to be some known number of bytes that is bigger than any single packet (the value `MAX_MTU`). At this snapshot, z/c more positive packets will have arrived than Cancelled. So `zs =` z/c `* MAX_MTU`. Then, by waiting until `zs` Cancelled bytes have arrived, we will have waited for z/c times more Cancelled bytes than there were up to the snapshot. So the difference between the two incremental variables for positive and negative bytes taken at the snapshot will have been c/z of the difference over the whole cycle. Accumulating these snapshot differences gives the required amount of inflation of downstream congestion-volume, $(z - u)c/z$.

Note that Cancelled bytes are accumulated then tested against thresholds (`MAX_MTU` or `zs`), but they are never zeroed which would lead to a bias in one direction due to rounding errors. We have been careful to always subtract the threshold from the count of Cancelled bytes, so that the remainder is carried forward into the next round.

Appendix B

Rejected Design Alternatives

B.1 Rejected: Three Primary Marking States

We introduced the Cancelled codepoint to remove a vulnerability of the previous re-ECN wire protocol encoding, which had just three states, Positive, Neutral and Negative.[1]

The currently standardised ECN field [RFB01] provides the redundancy of two states to mean 'not congestion marked but able to understand marking' (termed ECN-capable transport or ECT and the two states are called ECT(0) and ECT(1)). They were intended to allow a nonce [SWE03] to be woven into a stream of packets by the source, so it could detect if the receiver or any network element had tried to 'unmark' a congestion mark. A congestion mark set the ECN field to a third codepoint (congestion experienced or CE). So if anyone tries to unmark a packet, they have to guess which of the two ECT states the sender originally set. And the sender can detect when they guess wrongly.

The original re-ECN wire protocol [BJCG+05] used the codepoint of the ECN field that was reserved for the ECN nonce [SWE03].[2] ECT(1) was used for a Neutral marking and ECT(0) for Positive. And CE meant Negative. The unused bit in the IPv4 header was used to indicate something akin to what is now the 'Cautious' marking. As it is still, the charge for downstream congestion was measured by subtracting Negative from Positive bytes, whether actually translated into money, or subtracted from a user's 'congestion quota'.

This all seemed nice and simple, but problems piled on problems because both Neutral and Positive packets changed to Negative when congestion marked. Firstly, no-one could tell whether a Negative packet had previously been Positive, so a source had to introduce a relatively complicated inflation of the amount of Positive it sent to allow for some being marked. Secondly, networks could bias their Negative congestion marking against Positive packets in order to reduce the charge they had to pay for downstream congestion. As they gained two points of worth for marking a Positive packet against one for a Neutral packet, they had a strong incentive to do this. Although this led to a negative balance at the egress dropper, it was not possible to attribute the blame to any one network. Then, the source couldn't be sure how much inflation it should add, and the problems descended in a vicious spiral.

It became clear that we had to introduce the Cancelled state, making the re-ECN wire protocol as

[1] Setting aside the complication of the changes we made to the Cautious state at the same time.

[2] Re-ECN provides a superset of the capabilities of the ECN nonce, so we claimed the nonce is no longer necessary.

described in this dissertation. With the introduction of the Cancelled state, a Positive packet can now be congestion marked to remove one unit of worth, which is gained by the network doing the marking. But it does not lose two units of worth as in the previous scheme, and it is still possible to infer that a Cancelled packet probably started as a Positive packet. These changes have three useful effects:

- there is no longer an incentive for a network to bias its marking against Positive packets
- end-points can detect if a network changes packet markings in contravention of the protocol (see §12.1.4)
- there is no longer a need for the sender to do any complicated marking inflation

B.2 Rejected: Using Positive Not Cautious

The role of Cautious packets seems only marginally different from Positive packets. This raises the question of whether the functions of the two could be overloaded into one codepoint—the codepoint currently used by Positive—leaving the Cautious codepoint spare for future use. The primary reason for keeping the Cautious codepoint is a rather irritating and possibly minor backward compatibility issue.

As we have pointed out, the primary reason for introducing re-ECN into the internetwork layer is to provide principled control of sharing and congestion of the resource pool. When ECN was first defined, of course, there was no control of excessive use of resources. The solution for controlling unwanted traffic was thought to lie in anomalous flow detection, either those using excessive network resources, or those being rejected by hosts that didn't want them, perhaps using push-back messages, capabilities or other ideas at the time. This left the thorny problem of the initial packets of flows—they had to be assumed valid.

Those who standardised ECN knew that hosts sending unwanted traffic could claim that they would respond to ECN markings to gain an advantage over other legacy traffic that didn't. They also knew that the sender of the initial packet of a flow couldn't claim for certain that the receiver would understand or respond to ECN markings, because it hadn't yet established communication with it to find out its capabilities. Therefore they mandated that the initial packet of a flow MUST NOT be marked ECN-capable.[3]

The thinking was that servers and stateful firewalls could then immediately discard TCP SYN packets that claimed to be ECN capable, thus thwarting any advantage there was in using ECN-capability in a SYN attack.

With re-ECN, we have added control of resource sharing at the packet level. So we don't need any special measures to prevent initial packets gaining advantage from congestion marking. And we have devised another way to handle a re-ECN source being able to claim ECN-capability on the first

[3]The letter of the specification solely stipulates this for TCP transports, saying "A host MUST NOT set ECT on SYN ...packets". But the spirit of the specification seemed to intend a similar sentiment to apply to all transports. However, the DCCP transport, which became a proposed standard in 2006 (5 years after ECN) assumes support for ECN but says next to nothing about it, let alone whether the initial packet should claim ECN support.

packet—before it knows whether the receiver will understand a congestion mark from the network.[4]

However, if a re-ECN sender uses anything but Not-ECT (`00`) in the ECN field of the first packet of its flow, it is bound to hit problems with firewalls and servers who have followed the ECN specification and built in rules to discard TCP SYNs with ECN-capability.

We contrived the re-ECN protocol so that the Cautious codepoint made the packet look not ECN-capable (Not-ECT) to equipment that only checked the ECN field. However, Cautious is distinguished from Not-ECT by setting the RE flag. Therefore, as forwarding equipment is upgraded to understand re-ECN it can mark the ECN field in Cautious packets rather than drop them.

If Positive packets had served both the Positive and the Cautious functions, the protocol would have become trapped in a Catch-22. A source would not be able to send a Positive packet to start a flow on some paths, if a firewall or the server itself discarded ECN-capable SYNs. But if the source sent a non-Positive SYN, a re-ECN dropper would not necessarily set up flow state without any credit.

With the addition of the Cautious codepoint, a source can set-up flow-state with a packet that looks as if it isn't ECN-capable to legacy equipment and servers (even though it actually is) and it can have positive worth.

There is a strong possibility that some firewalls or servers will discard Cautious packets as a precaution too (because the RE flag has not traditionally been set). However, re-ECN won't work at all through such firewalls anyway, because it relies on using the RE flag. We can only hope and beg that firewalls and servers are upgraded to allow re-ECN capable packets. But we cannot ask that the rule to discard ECN-capable SYNs is removed from firewalls because, even if re-ECN is deployed, that firewall rule will have to stay as the only protection against DoS attacks using legacy ECN packets.

It might seem that packets with the codepoint used for Cautious could serve both Positive and Cautious functions instead. But then, every time a re-ECN flow needed to re-echo a positive marking, it would have to send a Cautious packet, which appears not to be ECN-capable to legacy forwarding equipment—a situation that would persist for many years, and probably decades. This would leave positive packets with a much greater chance of drop than other packets, making the resulting alternative re-ECN protocol highly flaky.

There are other reasons for introducing Cautious not just Positive markings:

- The most principled, but least concrete, argument is that a separate marking for Cautious from Positive allows the economic values of the two markings to diverge in the future given their different semantics. For instance, policers might rate-limit Cautious packets more strictly than Positive packets to throttle the opening of new flows.

- Markings in response to actual congestion (Positive) can be separated from markings that don't indicate actual congestion (Cautious) when monitoring packet aggregates within the network, perhaps for traffic engineering or SLA monitoring.

[4] If the receiver's initial response shows it doesn't speak re-ECN, the sender is advised to proceed cautiously *as if* the receiver had fed back that the first packet was congestion marked.

Finally, we should point out that we RECOMMEND that servers and middleboxes allow flow state to be set up by a request in either a Positive packet or a Cautious packet, but it is only MANDATORY to set up flow state when requested in a Cautious packet. Therefore hosts would be advised to use a Cautious marking to be sure.

One advantage the Positive marking does have is during a DoS attack. A highly congested forwarding element could be re-marking all Cautious packets to Negative and all Positive to Cancelled. A re-ECN dropper further along the path that was stressed by the same DoS attack would probably drop all the Negative packets, but it would forward the Cancelled packets unscathed. However, even a highly congested forwarding element shouldn't be congestion marking 100% of Cautious and Positive packets (see §9).

Note that congestion marking a Positive packet to Cancelled MUST NOT be taken to mean 'Flow state not stored'. This would add too much ambiguity to the protocol.

It is compliant with the ECN specification [RFB01] for a host to send the codepoint `ECT(1)` (termed Positive in the re-ECN protocol) in order to weave a nonce into the ECN fields of a stream of packets [SWE03]. If these arrived at the re-ECN dropper, it would not necessarily have to set up flow state, even if they hadn't been preceded by a Cautious packet with the same flow ID. But if a flow were re-routed into a re-ECN dropper from another path, it would be good if the first Positive packet in the stream did set up flow state, even if the initial Cautious packet had previously passed through a different dropper.

Appendix C

RED under Extreme Load

The drop probability of RED [FJ93] (and AQM's derived from it such as WRED [Sys02] or RIO [CF98]) is defined to rise to 100% in its congestion control phase. Therefore unresponsive traffic would make the link forward precisely nothing.

The core of the RED algorithm is meant to have three phases dependent on the size of the averaged queue:

- Normal $[0, \texttt{minth})$
- Congestion avoidance $[\texttt{minth}, \texttt{maxth})$
- Congestion Control $[\texttt{maxth}, \infty)$

The problem concerns the drop (not marking) behaviour of RED in its congestion control phase.[1] When the load L on a link is *above* the link capacity C, the average drop fraction in order to shed sufficient load only needs to be $p_n = 1 - C/L$. Fig C.1 shows p_n for a range of overloaded utilisation $u = L/C$ from 100–1000%. As soon as $u \geq 100\%$, the average queue maintained by RED will rapidly rise past `maxth`. RED is intended to drop 100% in its congestion control phase, when the average queue, $\bar{q} \geq \texttt{maxth}$ (or $\bar{q} \geq 2\texttt{maxth}$ for `gentle_RED`). Thus, with sufficient unresponsive traffic to fill the link, we assume RED will allow nothing through at all, as the conjectured plot of p_R in Fig C.1 shows.

RED will be likely to strongly favour unresponsive traffic over responsive as it approaches congestion control phase. If utilisation were hovering around 100%, causing drop to hover around 100% also, any responsive traffic would slow itself to a trickle. Therefore if, say, the average drop probability were 95%, the remaining 5% of capacity would virtually all be taken by unresponsive traffic. Whereas, if the queue were just naturally shedding load with no AQM, the drop probability could be relatively low (at least low relative to 95%) with such a load. For instance, even at 110% utilisation, natural drop would be 9%. Without RED if the arriving load were half responsive and half unresponsive half and half would get through.

[1] By 'RED' we mean how researchers describe it. It is possible that some implementers of RED have noticed this problem and invented *ad hoc* work-rounds. For instance, the ALTQ code for BSD Unix switches to tail drop when $\texttt{ave}_\texttt{q} > \texttt{maxth}$. An older random drop behaviour can still be configured instead. The research community still routinely models RED with 100% drop in congestion control, e.g. ns-2 [ns2] has no other behaviour.

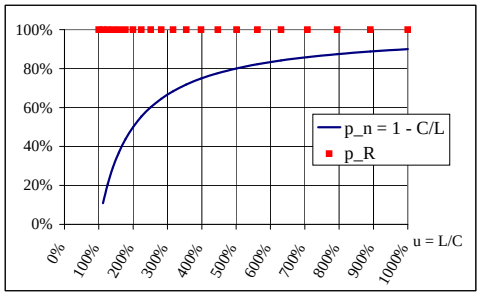


Figure C.1: Drop at an Overloaded Queue.
Utilisation $u = C/L > 100\%$. p_n: drop needed to shed sufficient arriving load; p_R: conjectured RED drop in its congestion control phase.

Note that 'unresponsive traffic' doesn't have to imply a DDoS attack. It could just be a flash crowd of inelastic traffic on a link. We will need to redesign an AQM that is robust against unexpected traffic loads, rather than forwarding nothing at all.

It could make sense for RED to increase *ECN marking* to 100% in the congestion control phase. But it doesn't seem to makes sense to increase *drop* to 100%. In fact, the problem seems to be that the drop behaviour mimics the marking behaviour when it shouldn't. The ECN specification, RFC3168 [RFB01] says that when RED moves into congestion control phase, marking should be turned off and it should only do drop. This is clearly intended to prevent unresponsive traffic gaining by abusing the ECN capability. However, it would only be a sensible strategy if drop mode itself didn't so strongly favour unresponsive traffic.

Bibliography

[AHCC06] Lachlan L.H. Andrew, Stephen V. Hanly, Sammy Chan, and Tony Cui. Adaptive deterministic packet marking. *IEEE Comm. Letters*, 10(11):790–792, November 2006.

[AK94] P. Almquist and F. Kastenholz. Towards requirements for IP routers. Request for comments 1716, Internet Engineering Task Force, November 1994. (Obsoleted by RFC1812) (Status: informational).

[Ake70] G.A. Akerlof. The market for 'lemons': Quality, uncertainty and market mechanisms. *Quarterly Journal of Economics*, 84:488–500, August 1970.

[AKM04] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing router buffers. *Proc. ACM SIGCOMM'04, Computer Communication Review*, 34(4), September 2004.

[AKS06] Edward Anderson, Frank Kelly, and Richard Steinberg. A contract and balancing mechanism for sharing capacity in a communication network. *Management Science*, 52:39–53, 2006.

[AMI+07] Katerina Argyraki, Petros Maniatis, Olga Irzak, Subramanian Ashish, and Scott Shenker. Loss and delay accountability for the internet. In *Proc. IEEE International Conference on Network Protocols*. IEEE, October 2007.

[Ari25] Aristotle. *Ethica Nicomachea*. Clarendon Press, Oxford, 1925. Translated by W.D.Ross.

[Arm06] Mark Armstrong. Competition in two-sided markets. *RAND Journal of Economics*, 37(3):668–691, Autumn 2006.

[Bar89] W. Barns. Defense data network usage accounting enhancement approaches. Technical report, The MITRE Corporation, 1989.

[Bau05] Steve Bauer. Incentive misalignment under congestion-based pricing. URL: `http://cfp.mit.edu/CFP_WG_WS/BBWG_NOV_2005/Steven_Bauer_11-05.pdf`, November 2005.

[BB01] Deepak Bansal and Hari Balakrishnan. Binomial congestion control algorithms. In *Proc. IEEE Conference on Computer Communications (Infocom'01)*, pages 631–640. IEEE, April 2001.

[BB05] Rob Beverly and Steve Bauer. The spoofer project: Inferring the extent of source address filtering on the Internet. In *Proc. Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI 2005)*, pages 53–59. USENIX, July 2005.

[BBB+97] F. Baker, B. Braden, S. Bradner, A. Mankin, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang. Resource ReSerVation protocol (RSVP) — version 1 applicability statement; Some guidelines on deployment. Request for comments 2208, Internet Engineering Task Force, January 1997.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Request for comments 2475, Internet Engineering Task Force, December 1998.

[BC01a] Bob Briscoe and Jon Crowcroft. An open ECN service in the IP layer. Technical Report TR-DVA9-2001-001, BT, February 2001.

[BC01b] Bob Briscoe and Jon Crowcroft. An open ECN service in the IP layer. Internet draft, Internet Engineering Task Force, February 2001. (Expired).

[BCC+98] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the Internet. Request for comments 2309, Internet Engineering Task Force, April 1998.

[BCS94] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: an overview. Request for comments 1633, Internet Engineering Task Force, June 1994.

[BCSJ04] Bob Briscoe, Sébastien Cazalet, Andrea Soppera, and Arnaud Jacquet. Shared control of networks using re-feedback; an outline. Technical Report TR-CXR9-2004-001, BT, September 2004.

[BCSW00] Robert Braden, David Clark, Scott Shenker, and John Wroclawski. Developing a next-generation Internet architecture. White paper, DARPA, July 2000.

[BDH+03] Bob Briscoe, Vasilios Darlagiannis, Oliver Heckman, Huw Oliver, Vasilios Siris, David Songhurst, and Burkhard Stiller. A market managed multi-service internet (M3I). *Computer Communications*, 26(4):404–414, February 2003.

[BDT+00] Bob Briscoe, Konstantinos Damianakis, Jérôme Tassel, Panayotis Antoniadis, and George Stamoulis. M3I pricing mechanism design; Price reaction. Deliverable 3 Pt II, M3I Eu Vth Framework Project IST-1999-11429, July 2000.

[Bel00] Steve M. Bellovin. ICMP traceback messages. Internet Draft draft-bellovin-itrace-00.txt, Internet Engineering Task Force, March 2000. (Work in progress).

[BES+06] Bob Briscoe, Philip Eardley, David Songhurst, Francois Le Faucheur, Anna Charny, Jozef Babiarz, Kwok-Ho Chan, Stephen Dudley, Georgios Karagiannis, Attila Bader, and Lars Westberg. An edge-to-edge deployment model for pre-congestion notification: Admission control over a DiffServ region. Internet Draft draft-briscoe-tsvwg-cl-architecture-04.txt, Internet Engineering Task Force, October 2006. (Work in progress).

[BFB06] Steve Bauer, Peyman Faratin, and Robert Beverly. Assessing the assumptions underlying mechanism design for the Internet. In *Proc. Workshop on the Economics of Networked Systems (NetEcon06)*, June 2006.

[BJCG+05] Bob Briscoe, Arnaud Jacquet, Carla Di Cairano-Gilfedder, Alessandro Salvatori, Andrea Soppera, and Martin Koyabe. Policing congestion response in an internetwork using re-feedback. *Proc. ACM SIGCOMM'05, Computer Communication Review*, 35(4):277–288, August 2005.

[BJMS09a] Bob Briscoe, Arnaud Jacquet, Toby Moncaster, and Alan Smith. Re-ECN: Adding accountability for causing congestion to TCP/IP. Internet Draft draft-briscoe-tsvwg-re-ecn-tcp-07.txt, Internet Engineering Task Force, March 2009. (Work in progress).

[BJMS09b] Bob Briscoe, Arnaud Jacquet, Toby Moncaster, and Alan Smith. Re-ECN: The motivation for adding congestion accountability to TCP/IP. Internet Draft draft-briscoe-tsvwg-re-ecn-tcp-motivation-00.txt, Internet Engineering Task Force, March 2009. (Work in progress).

[Bla08] Steven Blake. Use of the IPv6 flow label as a transport-layer nonce to defend against off-path spoofing attacks. Internet Draft draft-blake-ipv6-flow-label-nonce-01, Internet Engineering Task Force, November 2008. (Work in Progress).

[BLMR98] John Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. *Proc. ACM SIGCOMM'98, Computer Communication Review*, 28(4), September 1998.

[BLSS05] Eli Brosh, Galit Lubetzky-Sharon, and Yuval Shavitt. Spatial-temporal analysis of passive TCP measurements. In *Proc. IEEE Conference on Computer Communications (Infocom'05)*, pages 949–959. IEEE, March 2005.

[BOT06] Bob Briscoe, Andrew Odlyzko, and Benjamin Tilly. Metcalfe's Law is Wrong. *IEEE Spectrum*, Jul 2006:26–31, July 2006.

[BP87] R. T. Braden and J. Postel. Requirements for Internet gateways. Request for comments 1009, Internet Engineering Task Force, June 1987. (Obsoleted by RFC1812) (Status: historic).

[BR05] Bob Briscoe and Steve Rudkin. Commercial models for IP quality of service interconnect. *BT Technology Journal*, 23(2):171–195, April 2005.

[Bra97] Scott Bradner. Key words for use in RFCs to indicate requirement levels. BCP 14, Internet Engineering Task Force, March 1997. (RFC 2119).

[Bri99a] Bob Briscoe. The direction of value flow in connectionless networks. In *Proc. 1st International COST264 Workshop on Networked Group Communication (NGC'99)*, volume 1736. Springer LNCS, November 1999. (Invited paper).

[Bri99b] Bob Briscoe. The direction of value flow in multi-service connectionless networks. In *Proc. International Conference on Telecommunicatons and E-Commerce (ICTEC'99)*, October 1999.

[Bri00] Bob Briscoe. The direction of value flow in open multi-service connectionless networks. Technical Report TR-NZG12-2000-001, BT, August 2000.

[Bri02a] Bob Briscoe. M3I Architecture PtI: Principles. Deliverable 2 PtI, M3I Eu Vth Framework Project IST-1999-11429, February 2002.

[Bri02b] Bob Briscoe. M3I Architecture PtII: Construction. Deliverable 2 PtII, M3I Eu Vth Framework Project IST-1999-11429, February 2002.

[Bri06] Bob Briscoe. Using self-interest to prevent malice; Fixing the denial of service flaw of the Internet. In *Proc Workshop on the Economics of Securing the Information Infrastructure*, October 2006.

[Bri07a] Bob Briscoe. Fast congestion notification (FCN). Tech report TR-CXR9-2006-003, BT, May 2007. (unpublished work in progress).

[Bri07b] Bob Briscoe. Flow rate fairness: Dismantling a religion. *ACM SIGCOMM Computer Communication Review*, 37(2):63–74, April 2007.

[Bri07c] Bob Briscoe. Flow rate fairness: Dismantling a religion. Internet Draft draft-briscoe-tsvarea-fair-02, Internet Engineering Task Force, July 2007. (Expired).

[Bri08a] Bob Briscoe. Byte and packet congestion notification. Internet Draft draft-ietf-tsvwg-byte-pkt-congest-00.txt, Internet Engineering Task Force, August 2008. (Work in progress).

[Bri08b] Bob Briscoe. Emulating border flow policing using re-PCN on bulk data. Internet Draft draft-briscoe-re-pcn-border-cheat-02.txt, Internet Engineering Task Force, September 2008. (Work in progress).

[Bri08c] Bob Briscoe. A fairer, faster internet protocol. *IEEE Spectrum*, Dec 2008:38–43, December 2008.

[Bri08d] Bob Briscoe. Problem statement: Transport protocols don't have to do fairness. Internet Draft draft-briscoe-tsvwg-relax-fairness-01, Internet Engineering Task Force, July 2008. (Work in progress).

[Bri09] Bob Briscoe. Tunnelling of congestion notification. Internet Draft draft-ietf-tsvwg-ecn-tunnel-02.txt, Internet Engineering Task Force, March 2009. (Work in progress).

[CC01] Ioanna D. Constantiou and Costas A. Courcoubetis. Information asymmetry models in the Internet connectivity market. In *Proc. 4th Internet Economics Workshop*, May 2001.

[CC08] Denis Collange and Jean-Laurent Costeux. Passive estimation of quality of experience. *Journal of Universal Computer Science*, 14(5):625–641, March 2008.

[CCG+02] Parminder Chhabra, Shobhit Chuig, Anurag Goel, Ajita John, Abhishek Kumar, Huzur Saran, and Rajeev Shorey. XCHOKe: Malicious source control for congestion avoidance at Internet gateways. In *Proc. IEEE International Conference on Network Protocols (ICNP'02)*. IEEE, November 2002.

[CF98] David D. Clark and Wenjia Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[Cla88] David D. Clark. The design philosophy of the DARPA internet protocols. *Proc. ACM SIGCOMM'88, Computer Communication Review*, 18(4):106–114, August 1988.

[Cla95] David D. Clark. A model for cost allocation and pricing in the internet. *Journal of Electronic Publishing*, 1(1&2), January–February 1995.

[Cla96] David D. Clark. Combining sender and receiver payments in the Internet. In G. Rosston and D. Waterman, editors, *Interconnection and the Internet*. Lawrence Erlbaum Associates, Mahwah, NJ, October 1996.

[cla98] kc claffy. The nature of the beast: Recent traffic measurements from an Internet backbone. In *Proc. INET'98*. ISOC, 1998.

[CO98] Jon Crowcroft and Philippe Oechslin. Differentiated end to end Internet services using a weighted proportional fair sharing TCP. *Computer Communication Review*, 28(3):53–69, July 1998.

[Com02] Computer Science and Telecommunications Board (CSTB). *Broadband; Bringing Home the Bits*. National Academy Press, Washington D.C., 2002.

[CP98] L. Cherkasova and P. Phaal. Session-based admission control — a mechanism for improving performance of commercial web sites. In *Proc. International Workshop on QoS (IWQoS'99)*. IEEE/IFIP, June 1998.

[CW96] Costas Courcoubetis and Richard Weber. Buffer overflow asymptotics for a switch handling many traffic sources. *Journal Applied Probability*, 33:886–903, 1996.

[CW03] Costas Courcoubetis and Richard Weber. *Pricing Communication Networks*. Wiley, 2003.

[CWSB05] David Clark, John Wroclawski, Karen Sollins, and Robert Braden. Tussle in cyberspace: Defining tomorrow's Internet. *IEEE/ACM Transactions on Networking*, 13(3):462–475, June 2005.

[Day07] John Day. *Patterns in Network Architecture: A Return to Fundamentals*. Prentice-Hall, 2007.

[DBT08] Bruce Davie, Bob Briscoe, and June Tay. Explicit congestion marking in MPLS. Request for comments rfc5129.txt, Internet Engineering Task Force, January 2008.

[DKS89] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair-queueing algorithm. *Computer Communication Review (SIGCOMM'89)*, 19(4):1–12, September 1989.

[DKZSM05] Nandita Dukkipati, Masayoshi Kobayashi, Rui Zhang-Shen, and Nick McKeown. Processor sharing flows in the internet. In *Proc. International Workshop on QoS (IWQoS'05)*, June 2005.

[DM06] Nandita Dukkipati and Nick McKeown. Why flow-completion time is the right metric for congestion control. *ACM SIGCOMM Computer Communication Review*, 36(1):59–62, January 2006.

[DR04] Ian Dobbs and Paul Richards. Innovation and the new regulatory framework for electronic communications in the EU. *European Competition Law Review*, 25(11):716–730, 2004.

[Ear09a] Marking behaviour of PCN-nodes. Internet Draft draft-ietf-pcn-marking-behaviour-02.txt, Internet Engineering Task Force, March 2009. (Work in progress).

[Ear09b] Pre-congestion notification architecture. Internet Draft draft-ietf-pcn-architecture-11.txt, Internet Engineering Task Force, April 2009. (Work in progress).

[Edd07] Wes Eddy. TCP SYN flooding attacks and common mitigations. Request for comments RFC4987, Internet Engineering Task Force, August 2007.

[FAJS07] Sally Floyd, Mark Allman, Amit Jain, and Pasi Sarolahti. Quick-Start for TCP and IP. Request for comments rfc4782.txt, Internet Engineering Task Force, January 2007.

[FF99] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[FHPW00] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. Equation-based congestion control for unicast applications. *Proc. ACM SIGCOMM'00, Computer Communication Review*, 30(4):43–56, October 2000.

[FHPW03] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. TCP friendly rate control (TFRC): Protocol specification. Request for comments rfc3448.txt, Internet Engineering Task Force, January 2003.

[FI08] Bryan Ford and Janardhan Iyengar. Breaking up the transport logjam. In *Proc. IETF-73 Transport Area Open Meeting*. Internet Engineering Task Force, November 2008. (Presentation slides).

[Fin89] G. Finn. A connectionless congestion control algorithm. *ACM SIGCOMM Computer Communication Review*, 19(5), October 1989.

[FJ93] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[Flo94] Sally Floyd. TCP and explicit congestion notification. *ACM SIGCOMM Computer Communication Review*, 24(5):10–23, October 1994. (This issue of CCR incorrectly has '1995' on the cover).

[Flo08] Sally Floyd. RED (random early detection) queue management; setting parameters. Web page `http://www.icir.org/floyd/red.html#parameters`, November 2008. (Last accessed Jan 2009).

[FR98] E. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 1998.

[FW06] Peyman Faratin and Tom Wilkening. Interconnection discrimination: A two-sided markets perspective. In *Proc. ACM Hot Topics in Networking (HotNets-V)*. ACM, November 2006.

[GK99a] Richard J. Gibbens and Frank P. Kelly. Distributed connection acceptance control for a connectionless network. In *Proc. International Teletraffic Congress (ITC16), Edinburgh*, pages 941–952, 1999.

[GK99b] Richard J. Gibbens and Frank P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35(12):1969–1985, December 1999.

[GK02] Richard J. Gibbens and Frank P. Kelly. On packet marking at priority queues. *IEEE Transactions on Automatic Control*, 47(6):1016–1020, June 2002.

[GKM01] Ayalvadi Ganesh, Peter Key, and Laurent Massoulié. Feedback and bandwidth sharing in networks. In *Proc. 39th Annual Allerton Conference on Communication, Control and Computing*, 2001.

[GM06] Yashar Ganjali and Nick McKeown. Update on buffer sizing in Internet routers. *ACM SIGCOMM Computer Communication Review*, 36, October 2006.

[GMS00] Richard Gibbens, Robin Mason, and Richard Steinberg. Internet service classes under competition. *IEEE Journal on Selected Areas in Communications*, 18(12):2490–2498, 2000.

[GQX+04] David K. Goldenberg, Lili Qiu, Haiyong Xie, Yang Richard Yang, and Yin Zhang. Optimizing cost and performance for multihoming. *Proc. ACM SIGCOMM'04, Computer Communication Review*, 34(4):79–92, October 2004.

[Gro05] Broadband Working Group. The broadband incentive problem. White paper, MIT Communications Futures Programme and Cambridge University Communications Research Network, September 2005.

[HBC05] Peter Hovell, Bob Briscoe, and Gabriele Corlianò. Guaranteed QoS synthesis (GQS): An example of a scalable core IP quality of service solution. *BT Technology Journal*, 23(2), April 2005.

[HE02] David Hands (Ed.). M3I user experiment results. Deliverable 15.2, M3I Eu Vth Framework Project IST-1999-11429, February 2002. (M3I partner access only).

[HG04] Mark Handley and Adam Greenhalgh. Steps towards a DoS-resistant Internet architecture. In *FDNA '04: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pages 49–56, New York, NY, USA, 2004. ACM Press.

[HGB06] H. Hassan, J.M. Garcia, and C. Bockstal. Aggregate Traffic Models for VoIP Applications. In *Proc. International Conference on Digital Telecommunications (ICDT'06)*, page 70, 2006.

[HH07] Felipe Huici and Mark Handley. An edge-to-edge filtering architecture against DoS. *Computer Communication Review*, 37(2):39–50, 2007.

[ITU04] Traffic control and congestion control in B-ISDN. Recommendation I.371 (03/04), ITU-T, March 2004.

[Jac88] Van Jacobson. Congestion avoidance and control. *Proc. ACM SIGCOMM'88 Symposium, Computer Communication Review*, 18(4):314–329, August 1988.

[Jaf80] J.M. Jaffe. A decentralized, "optimal", multiple-user, flow control algorithm. In *Proc. Fifth Int'l. Conf. On Computer Communications*, pages 839–844, October 1980.

[Jaf81] J. M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, 29(7):954–962, July 1981.

[JBM08] Arnaud Jacquet, Bob Briscoe, and Toby Moncaster. Policing Freedom to Use the Internet Resource Pool. In *Proc Workshop on Re-Architecting the Internet (ReArch'08)*. ACM, December 2008.

[JBS05] Arnaud Jacquet, Bob Briscoe, and Alessandro Salvatori. A path-aware rate policer: Design and comparative evaluation. Technical Report TR-CXR9-2005-006, BT, October 2005.

[JRC87] R. Jain, K. Ramakrishnan, and D. Chiu. Congestion avoidance in computer networks with a connectionless network layer. Technical report DEC-TR-506, Digital Equipment Corporation, 1987.

[JWL04] Cheng Jin, David Wei, and Steven Low. FAST TCP: Motivation, architecture, algorithms, performance. In *Proc. IEEE Conference on Computer Communications (Infocomm'04)*. IEEE, March 2004.

[Kel97a] Frank P. Kelly. Charging and accounting for bursty connections. In Lee W. McKnight and Joseph P. Bailey, editors, *Internet Economics*, pages 253–278. MIT Press, 1997.

[Kel97b] Frank P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997. A version with a correction by Ramesh Johari and Frank Kelly to distinguish flows with zero weight and using a better structure of proof is available from URL: http://www.statslab.cam.ac.uk/~frank/elastic.html.

[Kel00] Frank P. Kelly. Models for a self-managed Internet. *Philosophical Transactions of the Royal Society*, 358(1773):2335–2348, August 2000.

[Kel03] Frank Kelly. Fairness and stability of end-to-end congestion control. *European Journal of Control*, 9:159–176, 2003.

[KHR02] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. *Proc. ACM SIGCOMM'02, Computer Communication Review*, 32(4):89–102, October 2002.

[Kle76] Leonard Kleinrock. *Queuing Systems, Vol II: Computer Applications*. Wiley, New York, 1976.

[KM97] D. Kristol and L. Montulli. HTTP state management mechanism. Request for comments 2109, Internet Engineering Task Force, February 1997.

[KM99] Peter Key and Laurent Massoulié. User policies in a network implementing congestion pricing. In *Proc. Workshop on Internet Service Quality and Economics*. MIT, December 1999.

[KMBK04] Peter Key, Laurent Massoulie, Alan Bain, and Frank Kelly. Fair Internet traffic integration: Network flow models and analysis. *Annales des Télécommunications*, 59:1338–1352, 2004.

[KMT98] Frank P. Kelly, Aman K. Maulloo, and David K. H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[KS01] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. *Proc. ACM SIGCOMM'01, Computer Communication Review*, 31(4), October 2001.

[KV05] Frank Kelly and Thomas Voice. Stability of end-to-end algorithms for joint routing and rate control. *ACM SIGCOMM Computer Communication Review*, 35(2):5–12, April 2005.

[LC06] Paul Laskowski and John Chuang. Network monitors and contracting systems: Competition and innovation. *Proc. ACM SIGCOMM'06, Computer Communication Review*, 36(4):183–194, September 2006.

[Lec99] Robert Lechner. Competitive network evolution towards broadband IP. In Bruce Wiltshire, editor, *Proc. BT Alliance Engineering Symposium (AES'99)*, page Session 1A Paper 2. BT Alliance, June 1999. (not publicly accessible).

[LG09] Michael Vittrup Larsen and Fernando Gont. Port randomization. Internet Draft draft-ietf-tsvwg-port-randomization-03, Internet Engineering Task Force, March 2009. (Work in Progress).

[MBJ07] Toby Moncaster, Bob Briscoe, and Arnaud Jacquet. A TCP test to allow senders to identify receiver non-compliance. Internet Draft draft-moncaster-tsvwg-rcv-cheat-02.txt, Internet Engineering Task Force, November 2007. (Work in progress).

[MFW01] Ratul Mahajan, Sally Floyd, and David Wetheral. Controlling high-bandwidth flows at the congested router. In *Proc. IEEE International Conference on Network Protocols (ICNP'01)*, 2001.

[MHR+90] A. Mankin, G. Hollingsworth, G. Reichlen, K. Thompson, R. Wilder, and R. Zahavi. Evaluation of Internet performance — FY89. Technical report MTR-89W00216, MITRE Corporation, February 1990.

[MMV93] Jeffrey K. MacKie-Mason and Hal Varian. Some economics of the Internet. In *Proc. Tenth Michigan Public Utility Conference at Western Michigan University*, March 1993.

[MMV95] Jeffrey K. MacKie-Mason and Hal Varian. Pricing congestible network resources. *IEEE Journal on Selected Areas in Communications, "Advances in the Fundamentals of Networking"*, 13(7):1141–1149, 1995.

[MPCC00] Richard Mortier, I. Pratt, C. Clark, and Simon Crosby. Implicit admission control. *IEEE Journal on Selected Areas in Communications*, 18(12):2629–2639, December 2000.

[MR91] A. Mankin and K. Ramakrishnan. Gateway congestion control survey. Request for comments 1254, Internet Engineering Task Force, July 1991. (Status: informational).

[MR99] Laurent Massoulié and Jim W. Roberts. Arguments in favour of admission control for TCP flows. In *Proc. Int'l Teletraffic Congress (ITC16)*, June 1999.

[MSMO97] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *SIGCOMM Comput. Commun. Rev.*, 27(3):67–82, 1997.

[MW00] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, October 2000.

[Nag84] J. Nagle. Congestion control in IP/TCP internetworks. Request for comments 896, Internet Engineering Task Force, January 1984. (Status: unknown).

[Nag85] J. Nagle. On packet switches with infinite storage. Request for comments 970, Internet Engineering Task Force, December 1985. (Status: unknown).

[ns2] Network simulator. URL: `http://www.isi.edu/nsnam/ns/`.

[Odl97] Andrew Odlyzko. A modest proposal for preventing Internet congestion. Technical report TR 97.35.1, AT&T Research, Florham Park, New Jersey, September 1997.

[OLW99] Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. SRED: Stabilized RED. In *Proc. IEEE Conference on Computer Communications (Infocom'99)*, pages 1346–1355. IEEE, March 1999.

[Orm05] Ralph Orme. British Telecommunications plc's statement about IPR claimed in draft-briscoe-tsvwg-re-ecn-tcp-00.txt. URL: `https://datatracker.ietf.org/public/ipr_detail_show.cgi?&ipr_id=651`, November 2005.

[Pax99] Vern Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):227–292, June 1999.

[PBPS03] Rong Pan, Lee Breslau, Balaji Prabhaker, and Scott Shenker. Approximate fairness through differential dropping. *ACM SIGCOMM Computer Communication Review*, 33(2):23–40, April 2003.

[PFTK98] Jitendra Padhye, V. Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput: A simple model and its empirical validation. *Proc. ACM SIGCOMM'98, Computer Communication Review*, 28(4), September 1998.

[PLD04] Antonis Papachristodoulou, Lun Li, and John C. Doyle. Methodological frameworks for large-scale network analysis and design. *ACM SIGCOMM Computer Communication Review*, 34(3):7–20, July 2004.

[Pop63] Karl Popper. *Conjectures and Refutations*. Routledge & Kegan Paul, Abingdon, Oxon, England, 1963.

[Pos81] Jon Postel. Internet control message protocol. Request for comments 0792, Internet Engineering Task Force, September 1981.

[PPP00] R. Pan, B. Prabhakar, and K. Psounis. CHOKe, A stateless active queue management scheme for approximating fair bandwidth allocation. In *Proc. IEEE Conference on Computer Communications (Infocom'00)*. IEEE, March 2000.

[PQW03] Venkata N. Padmanabhan, Lili Qiu, and Helen Wang. Server-based inference of Internet performance. In *Proc. IEEE Conference on Computer Communications (Infocomm'03*. IEEE, April 2003.

[Raw01] John Rawls. *Justice as Fairness: A Restatement*. Harvard University Press, Cambridge, MA, 2001.

[Red01] Smitha A. L. Narasimha Reddy. LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers. In *Proc Globecomm'01*, November 2001.

[RFB01] K. K. Ramakrishnan, Sally Floyd, and David Black. The addition of explicit congestion notification (ECN) to IP. Request for comments 3168, Internet Engineering Task Force, September 2001.

[RS06] Barath Raghavan and Alex C. Snoeren. Decongestion control. In *Proc. ACM Hot Topics in Networking (HotNets-V)*. ACM, November 2006.

[Sal05] Alessandro Salvatori. Closed loop traffic policing. Master's thesis, Politecnico Torino and Institut Eurécom, September 2005.

[SBS02a] Vasilios A. Siris, Bob Briscoe, and Dave Songhurst. Economic models for resource control in wireless networks. In *Proc. 13th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2002)*. IEEE, September 2002.

[SBS02b] Vasilios A. Siris, Bob Briscoe, and Dave Songhurst. Service differentiation in third generation mobile networks. In *International workshop on Quality of future Internet Services (QofIS'02)*, volume 2511, pages 169–178. COST263, Springer LNCS, October 2002.

[SCEH96] Scott Shenker, David Clark, Deborah Estrin, and Shai Herzog. Pricing in computer networks: Reshaping the research agenda. *ACM SIGCOMM Computer Communication Review*, 26(2), April 1996.

[SCM01] Vasilios A. Siris, Costas Courcoubetis, and George Margetis. Service differentiation in ECN networks using weighted window-based congestion control. In *Proc. 2nd International workshop on Quality of future Internet Services (QofIS'01)*. COST263, September 2001.

[SCM02] Vasilios A. Siris, Costas Courcoubetis, and George Margetis. Service differentiation and performance of weighted window-based congestion control and packet marking algorithms in ECN networks. *Computer Communications*, 26(4):314–326, 2002.

[SCWA99] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. TCP congestion control with a misbehaving receiver. *ACM SIGCOMM Computer Communication Review*, 29(5):71–78, October 1999.

[SEB+06] David J. Songhurst, Phil L. Eardley, Bob Briscoe, Carla Di Cairano Gilfedder, and June Tay. Guaranteed QoS Synthesis for admission control with shared capacity. Technical Report TR-CXR9-2006-001, BT, February 2006.

[She95] Scott Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176–1188, 1995.

[Sir02] Vasilios A. Siris. Resource control for elastic traffic in CDMA networks. In *Proc. ACM International Conference on Mobile Computing and Networks (MobiCom'02)*. ACM, September 2002.

[SPS+02] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, 10(6):721–734, 2002.

[SRC84] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984. An earlier version appeared in the Second International Conference on Distributed Computing Systems (April, 1981) pages 509–512.

[SWE03] Neil Spring, David Wetherall, and David Ely. Robust explicit congestion notification (ECN) signaling with nonces. Request for comments RFC3540, Internet Engineering Task Force, June 2003. (Status: Experimental).

[Sys02] Cisco Systems. Distributed weighted random early detection. Release Note Cisco IOS Release 11.1 CC and Feature Modules, Cisco Systems, 2002.

[TC04] R.W. Thommes and M.J. Coates. Deterministic packet marking for time-varying congestion price estimation. In *Proc. IEEE Conference on Computer Communications (Infocomm'04)*. IEEE, March 2004.

[Wei91] Mark Weiser. The computer for the 21st Century. *Scientific American*, 265(3):94–104, September 1991.

[WHBB08] Damon Wischik, Mark Handley, and Marcelo Bagnulo Braun. The Resource Pooling Principle. *SIGCOMM Comput. Commun. Rev.*, 38(5):47–52, October 2008.

[Wis07] Damon Wischik. Short messages. In *Proc. Workshop on Networks: Modelling and Control*. Royal Society, September 2007.

[WK08] John Wittgreffe and Kashaf Khan. Orchestrating end-to-end network and resources according to application level service level agreements. *BT Technology Journal*, 26(1):46–57, September 2008.

[Wol82] Ronald W. Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2):223–231, March–April 1982.

[WPSB09] Michael Welzl, Dimitri Papadimitriou, Michael Scharf, and Bob Briscoe. Open Research Issues in Internet Congestion Control. Internet Draft draft-irtf-iccrg-welzl-congestion-control-open-research-03, Internet Research Task Force, April 2009. (Work in progress).

[XSSK05] Yong Xia, Lakshminarayanan Subramanian, Ion Stoica, and Shivkumar Kalyanaraman. One more bit is enough. *Proc. ACM SIGCOMM'05, Computer Communication Review*, 35(4):37–48, 2005.

[YMKT99] Maya Yajnik, Sue B. Moon, James F. Kurose, and Donald F. Towsley. Measurement and modeling of the temporal dependence in packet loss. In *Proc. IEEE Conference on Computer Communications (Infocom'99)*, pages 345–352. IEEE, 1999.

[YPG00] R. Yavatkar, D. Pendarakis, and R. Guerin. A framework for policy-based admission control. Request for comments 2753, Internet Engineering Task Force, January 2000.

[ZD01] Yin Zhang and Nick Duffield. On the constancy of Internet path properties. In *Proc. 1st SIGCOMM Workshop on Internet Measurement (IMW '01)*, pages 197–211, New York, NY, USA, 2001. ACM.

[ZDE+93] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: A new resource ReSerVation protocol. *IEEE Network*, September 1993.