Private and Censorship-Resistant Communication over Public Networks

Michael John Rogers

Thesis submitted for the degree of
Doctor of Philosophy
of
UCL

Department of Computer Science
University College London
University of London

2010

I, Michael John Rogers, 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.

Abstract

Society's increasing reliance on digital communication networks is creating unprecedented opportunities for wholesale surveillance and censorship. This thesis investigates the use of public networks such as the Internet to build robust, private communication systems that can resist monitoring and attacks by powerful adversaries such as national governments.

We sketch the design of a censorship-resistant communication system based on peer-to-peer Internet overlays in which the participants only communicate directly with people they know and trust. This 'friend-to-friend' approach protects the participants' privacy, but it also presents two significant challenges. The first is that, as with any peer-to-peer overlay, the users of the system must collectively provide the resources necessary for its operation; some users might prefer to use the system without contributing resources equal to those they consume, and if many users do so, the system may not be able to survive.

To address this challenge we present a new game theoretic model of the problem of encouraging cooperation between selfish actors under conditions of scarcity, and develop a strategy for the game that provides rational incentives for cooperation under a wide range of conditions.

The second challenge is that the structure of a friend-to-friend overlay may reveal the users' social relationships to an adversary monitoring the underlying network. To conceal their sensitive relationships from the adversary, the users must be able to communicate indirectly across the overlay in a way that resists monitoring and attacks by other participants.

We address this second challenge by developing two new routing protocols that robustly deliver messages across networks with unknown topologies, without revealing the identities of the communication endpoints to intermediate nodes or *vice versa*. The protocols make use of a novel unforgeable acknowledgement mechanism that proves that a message has been delivered without identifying the source or destination of the message or the path by which it was delivered. One of the routing protocols is shown to be robust to attacks by malicious participants, while the other provides rational incentives for selfish participants to cooperate in forwarding messages.

Contents

Acknowledgements

Spending eight years on a puzzle is a selfish use of time, and of the many people I would like to thank for their support, advice, encouragement and inspiration during the time it has taken to write this thesis, there are some who first deserve an apology. Laura, Alan and Dan, I'm sorry for too often letting work get in the way.

My thanks go first to my parents, who always encouraged me to be curious and to enjoy learning, although they may wish by now that they had emphasised the benefits of employment instead. I would also like to thank my colleagues at UCL and elsewhere – especially Denise, Piers, Manish, Torsten, Mo, Daniele, Toad and Matteo – for everything they taught me.

To the friends in the CBR who pulled me through some difficult times and difficult haircuts – HB, Itch, Russell, Phil, Jimbo, Jamie, Mike – thank you. Lorena, I wouldn't have understood the full horror of what it is to be a PhD student without our conversations. To the crew of the good ship LimeWire, now holed below the waterline, I offer my thanks for making me into a better coder and a worse drinker.

I would also like to thank my supervisors, Steve Hailes, Cecilia Mascolo, Licia Capra, and above all Saleem Bhatti, without whose patience, generosity, insight, wise advice and good humour this long random walk would not have reached its destination.

$\kappa\alpha\lambda\lambda\iota\sigma\tau\eta$

Chapter 1

Introduction

> "Now, I been in jail when all my mail showed
> That a man can't give his address out to bad company"
> – Bob Dylan, *Absolutely Sweet Marie*

1.1 Background

Scientific research always exists within a social context. The goals of the present work – privacy and censorship-resistance – are social as well as technical concepts, and before defining them in technical terms we must first consider something of their social meaning.

Censorship

Censorship, narrowly defined, is the official examination and suppression of material that is to be published. However, in the context of Internet censorship, this definition immediately raises a number of questions. Can an automatic Internet filter be said to perform an official examination? Is denying access to published material from a particular location – such as a school, a library, or even an entire country – equivalent to prohibiting its publication? Can material that circulates between friends be said to have been published?

Jansen [1] argues for a broader conception of censorship as "socially structured silence", recognising that power relations may influence discourse in many ways. These include official and unofficial punishment and intimidation; social norms that deny certain individuals a public voice; institutional practices that limit the spread of information; and the power of markets and media owners to decide what is published. Whether such influence is active, as in the case of a government banning a newspaper, or passive, as in the case of a newspaper declining to cover a controversial story, the effect of censorship is to reinforce the relations of power that produce it.

If censorship is an effect of power relations then we might suppose that it can be resisted by placing communication beyond the reach of power. Habermas [2] imagines an *ideal speech situation* where "no force except that of the better argument is exercised" and "all motives except that of the cooperative search for truth are excluded." In this ideal situation, speakers seek universal consensus in "an atmosphere free of coercion or dependencies (inequalities) that would incline individuals toward acquiescence or silence" [3]. Habermas admits that this ideal is never achieved in practice, but it might be argued that even as an ideal, the notion of universal consensus implicitly deprecates dissent. We might also ask how consensus can be distinguished from mere compliance – Strauss [4] describes how the threat of persecution has led many authors to "write between the lines", creating an appearance of agreement by concealing subversive messages within apparently conformist texts.

Lyotard [5] rejects the goal of universal consensus in favour of a contestational model of discourse as a set of incommensurable *language games*, where "every utterance should be thought of as a 'move' in a game." For Lyotard, disagreement and dissent are enlivening rather than threatening; nevertheless, power can still intrude into language games in the form of *terror*: "eliminating, or threatening to eliminate, a player from the language game one shares

with him." Thus, whether discourse is seen as the search for consensus or as a competitive and constantly evolving game, the influence of power relations cannot be ignored.

Recent analyses of Internet censorship reveal a wide range of practices being used to control online discourse, from the simple filtering of Internet traffic to complex socio-technical systems of monitoring, self-regulation, intimidation and propaganda that blur the line between repressive and productive forms of power [6, 7, 8, 9, 10]. Clearly, contemporary censorship is not merely a matter of preventing access to information: it is a matter of shaping discourse in accordance with the demands of power.

Surveillance and Privacy

According to Murakami Wood [11], "Where we find purposeful, routine, systematic and focused attention paid to personal details, for the sake of control, entitlement, management, influence or protection, we are looking at surveillance." If censorship is socially structured silence, then surveillance is socially structured visibility.

Foucault [12] uses the Panopticon, Bentham's proposal for a circular prison in which the inmates are constantly and unobtrusively observed from a central watchtower [13], to illustrate the relationship between power and visibility. According to Foucault, those who are under surveillance internalise the power of the observer; control becomes self-control. "He who is subjected to a field of visibility, and who knows it, assumes responsibility for the constraints of power; he makes them play spontaneously upon himself; he inscribes in himself the power relation in which he simultaneously plays both roles; he becomes the principle of his own subjection."

In recent years, surveillance has moved beyond the walls of the institutions described by Foucault and has become ubiquitous. It is now common for public spaces to be monitored through video cameras, and for electronic interactions of all kinds to be recorded, collated and analysed, constantly and automatically, in what has been called "the surveillance society" [11, 14, 15]. Without doubt, what Foucault would refer to as the *disciplinary* effect of surveillance – the projection of a normalising influence onto those observed – is an intended outcome of deploying surveillance systems, perhaps even outweighing the desire to identify and punish offenders. So much is readily admitted with respect to video cameras [16]. We are less eager to admit that the widespread monitoring of personal communication might also have a normalising effect.

Nock [17] offers a contrasting view of surveillance as a necessary response to what he sees as the unprecedented degree of privacy in modern life. "If personal privacy is defined as the legitimate denial of access to, or scrutiny of, one's behaviours, then it is clear that we enjoy more privacy today than did our ancestors." According to Nock, social control in premodern societies was exercised through the family and the community, but this is no longer possible in contemporary societies, with their mobile and anonymous populations; surveillance has become necessary to establish and maintain reputations among strangers.

Nock's definition of privacy as "denial of access" is reminiscent of Cooley's famous "right to be let alone" [18]. Yet there is more to privacy than excluding others. Pedersen [19] identifies six distinct types of privacy – reserve, isolation, solitude, anonymity, intimacy with family and intimacy with friends – and shows that individuals who desire one type of privacy do not necessarily desire the others. Feldman [20] observes that "individuals live their lives in a number of social spheres, which interlock, and in each of which they have different responsibilities, and have to work with people in relationships of varying degrees of intimacy." For Feldman, privacy means being able to choose the social spheres in which we live, and to control the flow of information between them. "The core of privacy as a civil liberty, then, is the entitlement to dignity and autonomy within a social circle." The private and the social are not opposed, but fundamentally linked.

The expectations of intimacy, confidentiality and trust within personal relationships can make them a strong foundation for resisting censorship, as in the *samizdat* (self-publishing) network in the Soviet Union, which was used by artists and dissidents to distribute subversive texts [21]. Yet, paradoxically, those same qualities make personal relationships into attractive targets for surveillance. Even when the content of a person's communication cannot be intercepted, the pattern of communication may be revealing. Analysing the social networks revealed by surveillance has become a significant research topic in recent years [22, 23, 24, 25, 26, 27].

In the context of political resistance, Beam [28] argues that large, hierarchical organisations will always be vulnerable to surveillance of their social structures; as an alternative he proposes *leaderless resistance* in small, independent cells. At the opposite end of the political spectrum, the Critical Art Ensemble reaches the same conclusion: "leftist political

action must reorganize itself in terms of anarchist cells, an arrangement that allows resistance to emerge from many different points" [29].

Bey [30] describes cellular anarchist communities that aim to evade institutional power rather than confronting it directly, occupying "cracks and vacancies" in the territory of the state and surviving through mobility, invisibility and evasion. This may seem to be a feeble form of resistance, but if the mere fact of visibility creates a power relationship, as Foucault argues, then becoming invisible dissolves that relationship. In the struggle against surveillance, evading observation is not a means to an end: it is an end in itself.

1.2 Goals

From a technical perspective, the goals of the present work – private and censorship-resistant communication over public networks – can be defined as follows: we aim to enable individuals to communicate across a public network such as the Internet, in the presence of a powerful adversary such as a national government, without the adversary being able to prevent the individuals from communicating, or to discover the content, volume, or timing of their communication.

The goals of privacy and censorship-resistance are interlinked: by ensuring that the adversary cannot prevent communication we resist censorship in the narrow sense of suppression of material, while by protecting the communicating parties from the effects of surveillance and intimidation we resist censorship in the broad sense of socially structured silence.

1.3 Assumptions

Since a public network such as the Internet cannot be assumed to provide the privacy or censorship-resistance properties described above, we will attempt to provide those properties by designing a *censorship-resistant communication system* that uses the public network as a communication substrate. Designing and evaluating such a system will require some assumptions to be made about the users of the system and the adversary we intend it to resist.

1.3.1 The Adversary

Any discussion of censorship-resistance must assume the existence of a powerful and well-funded adversary, whose goal may be to monitor people's communication, to prevent certain people from communicating, or to deny access to certain information. The adversary may be *internal or external* to the communication system; may have *local or global* access to it; may use a *static or adaptive* behaviour pattern; and may be *active or passive* in relation to the system [31]. Passive adversaries are sometimes referred to as *observers* or *eavesdroppers*, while active adversaries are generally referred to as *attackers*.

One way to choose an appropriate threat model would be to specify an adversary with restricted capabilities, design a system that could resist that adversary, then incrementally strengthen the threat model and repeat the process. Another approach would be to describe a realistic adversary from the start, make some progress towards resisting that adversary, then incrementally improve the design. This work takes the latter approach. The threat model described below is similar to the strongest threat models used in the existing literature, and we believe that recent events have shown it to be realistic. That does not mean, unfortunately, that we claim to offer a complete solution to resisting an adversary of this kind.

- **A global observer.** We assume the adversary can passively monitor all Internet traffic at a national or global scale. For example, the adversary may control a national Internet service provider. The alleged monitoring of the Internet backbone [32, 33, 34] and the proliferation of national firewalls [35, 36] have made it clear that such wholesale surveillance is a realistic threat.
- **A limited internal attacker.** The adversary is assumed to be capable of infiltrating the censorship-resistant communication system to a limited extent, either by compromising some fraction of the users' computers or by persuading some fraction of the users to trust the adversary's agents.

- **A limited active attacker.** We assume that the adversary can drop, delay, modify and generate Internet traffic. For example, the adversary may block all communication between specific IP addresses, or flood specific connections with traffic. It is assumed, however, that the adversary is unable or unwilling to cause widespread disruption – active attacks must be targetted and limited in scope. We do not consider how to resist an adversary who is willing to shut down Internet access at a regional or national level, although that is, unfortunately, a realistic threat [37, 38].

- **A limited physical attacker.** In addition to attacks involving Internet traffic, the adversary is assumed to have a limited ability to coerce or intimidate the users of the censorship-resistant communication system, if such users can be identified. As discussed in section 1.1, this is an important aspect of censorship in the real world.

- **Access to IP addresses.** We assume that the adversary controls enough IP addresses with distinct prefixes to prevent such addresses or prefixes from being treated as scarce identifiers.

- **Access to computing power.** The adversary is assumed to have sufficient computing power to solve computational problems at a faster rate than all of the users of the censorship-resistant communication system combined. However, we assume the adversary cannot break standard cryptographic primitives.

- **Able to identify users.** We assume the adversary can identify the user of any computer given its IP address; this is commensurate with the abilities of many governments [7, 15, 39].

Although the above assumptions focus on the Internet, we assume similar capabilities with respect to other public networks, such as the telephone system.

1.3.2 The Users

We must also make some assumptions about the people who might wish to use the censorship-resistant communication system.

- **Internet access.** We assume that individuals who wish to use the system have access to personal computers and Internet connections, and that, apart from the limited infiltration described above, they are able to keep their computers secure and their cryptographic keys secret.

  This is a strong assumption, but it is one upon which many existing communication systems depend. Protecting the information stored on personal computers against malware, intrusion and theft is an important task, but one that goes beyond the scope of this thesis.

- **Diversity of trust.** It is assumed that every user trusts some other users of the system and is willing to reveal her use of the system to them. We refer to these people as the user's *friends*, and we use that term exclusively to indicate a trust relationship of this kind. It is not assumed that users will only wish to communicate with their friends.

  Crucially, we do not assume that any person or organisation is trusted by all of the users of the system.

- **Social networks.** It cannot generally be assumed that all users of a given communication system are members of a single social group or *vice versa*. However, in this work we make the strong and perhaps unrealistic assumption that any two people who wish to communicate using a particular system are connected by a chain of friends (in the sense defined above) who are also users of that system. This assumption may be more reasonable for systems that grow by personal invitation than it is for communication systems in general.

- **Shared secrets.** We assume that any two people who wish to communicate, whether directly or indirectly, have some way to establish a shared secret out-of-band, for example by meeting in person or by communicating through a mutually trusted third party.

- **Selfishness.** It is assumed that some fraction of the users are selfish, meaning that they are more strongly concerned with the level of service they receive from the communication system than with the level of service received by other users. We do not assume that people in general are selfish, only that some selfish people exist.

1.4 Requirements

Based on the goals defined in section 1.2 and the assumptions described in section 1.3, we can identify the following requirements for a censorship-resistant communication system:

- **Confidentiality.** The users of the system should be able to communicate without revealing the content of their communication to anyone. This requirement follows from the goal of privacy. The assumption of an internal attacker entails that the content of communication must be concealed not only from external observers, but also from other users.
- **Unlinkability.** The adversary should not be able to determine which users are communicating with each other. Like confidentiality, this requirement follows from the goal of privacy, and as with confidentiality the assumption of an internal attacker entails that communication patterns must be concealed from other users as well as from external observers.
- **Robustness.** The adversary should not be able to destroy the system or prevent it from being used. This requirement follows from the goal of censorship-resistance and the assumption of an active attacker. As stated above, we do not attempt to resist an adversary who can shut down Internet access entirely, but many existing communication systems can be disabled by far less drastic measures.
- **Decentralisation.** The system should not depend on any centralised technical or administrative components, since that would require all users to trust a single entity. This requirement follows from the assumption of diversity of trust. Centralisation would create a single point of failure, so this requirement also follows from the assumption of an active attacker.
- **Covert membership.** The system should limit, as far as possible, the number of people who can discover that any given person is using the system. This assumption follows from the goal of privacy and the assumption of an attacker who can coerce or intimidate known users.

  In addition to undermining the privacy of individual users, a list of users may serve as a starting point for attacks against the communication system itself, such as blocking or interfering with traffic, and may be used to analyse the users' long-term communication patterns. This requirement is therefore connected to the requirements of robustness and unlinkability discussed above.

  In practice it may not be possible to conceal the membership of an Internet-based system from an adversary who can monitor all Internet traffic and distinguish the system's protocols from other traffic. Nevertheless, it is possible and worthwhile to conceal such information from less powerful adversaries.
- **Resource contribution.** The resources needed for communication should be provided collectively by the users rather than by a separate entity. This requirement follows from the assumption of diversity of trust. The system should allocate the resources needed for communication in a way that encourages selfish users to contribute resources and limits the impact of resource-depletion attacks. This requirement follows from the goal of censorship-resistance, the assumption of selfishness, and the assumption of an internal, active attacker.

In Chapter 2 we review existing work related to the goals and requirements listed above, which spans the fields of peer-to-peer networks, robust storage and routing, anonymous communication, and incentive mechanisms.

1.5 Thesis Statement

The thesis of this work is that private, censorship-resistant communication is possible over public networks such as the Internet, through the construction of distributed overlay networks in which users relay encrypted messages for one another. Even in the presence of a powerful adversary who can monitor all traffic in the underlying network and make targetted attacks, robust and unlinkable communication can be maintained across such overlays without central control, coordination or infrastructure, and without requiring users to communicate directly with anyone they do not trust. Attackers with a limited ability to infiltrate such overlays can be prevented from disrupting the communication of other users, and selfish users who would prefer not to contribute resources can be given rational incentives to do so.

1.6 Contributions

This work makes the following contributions to the study of censorship-resistant communication:

1. Chapter 4 describes a new game theoretic model of cooperation under conditions of scarcity, *the sharer's dilemma*, and an *expected utility strategy* that robustly encourages cooperation between selfish players.
2. Chapter 5 describes a mechanism for producing *unforgeable acknowledgements* to prove that messages have been delivered unmodified to their intended destinations across an untrusted network, without requiring an identity infrastructure, and without revealing the identities of the communication endpoints to untrusted parties.
3. Chapter 6 describes two *address-free routing protocols* for untrusted networks, which maintain unlinkability between sources and destinations without requiring an identity infrastructure. The first protocol is shown to be robust to active internal attacks against routing, while the second is shown to create an incentive for selfish users to cooperate in message forwarding.

1.7 Previously Published Work

Some parts of the work presented in Chapters 2, 4, 5 and 6 were previously published as conference papers and a book chapter [40, 41, 42, 43]. In all cases the author of the present work was the main author of the publications, which were co-authored by Saleem Bhatti, acting in a supervisory capacity.

Chapter 2

Related Work

> "On those remote pages it is written that animals are divided into (a) those that belong to the Emperor, (b) embalmed ones, (c) those that are trained, (d) suckling pigs, (e) mermaids, (f) fabulous ones, (g) stray dogs, (h) those that are included in this classification, (i) those that tremble as if they were mad, (j) innumerable ones, (k) those drawn with a very fine camel's hair brush, (l) others, (m) those that have just broken a flower vase, (n) those that resemble flies from a distance."
> – Jorge Luis Borges, *The Analytical Language of John Wilkins*

This chapter gives a brief survey of several fields that are relevant to private and censorship-resistant communication. Some of the work reviewed here is directly related to the technical contributions described in later chapters, while other work is included to demonstrate why the general approach of the present work was chosen over the alternatives.

It should be noted that much of the work surveyed here is based on goals and assumptions different from those described in Chapter 1. Censorship-resistance involves an unusually strong threat model and requirements that would unnecessarily constrain the design space of systems intended for other purposes. However, since the goal of this chapter is to identify techniques that are suitable for censorship-resistant communication, existing work is evaluated using the threat model and requirements described in Chapter 1.

We begin by reviewing some background issues concerning network structure and identity that are relevant to many of the systems discussed in later sections. Section 2.2 discusses peer-to-peer networks, while section 2.3 describes systems for robust storage and routing. Anonymous communication is discussed in section 2.4, and section 2.5 describes ways to encourage users to contribute resources to distributed systems. The chosen categories inevitably overlap, so in some cases different aspects of a piece of work are discussed in different sections.

2.1 Background

2.1.1 Network Structure

Baran [44] was perhaps the first researcher to study the effect of a network's structure on its robustness to attack. In his seminal work on survivable communication networks, Baran distinguishes between *centralised* networks with a single hub, *decentralised* networks with many hubs forming a tree-like structure, and *distributed* networks with redundant links forming a mesh. To maximise survivability, Baran recommends building distributed networks from inexpensive, unreliable, yet redundant components.

Large networks can be classified using statistics that summarise their structure. Three notable classes of distributed network are *random networks* [45], in which every node is connected to every other node with equal probability; *small world networks*, which are highly clustered; and *scale-free networks*, which have highly skewed degree distributions.

Small World Networks

Watts and Strogatz [46] investigate the *small world phenomenon*, in which members of a large social network turn out to be connected by a short chain of acquaintances [47, 48, 49]. Networks that replicate this effect can be created by adding random connections to a regular lattice: like lattices, the resulting networks are highly clustered, but like random networks, there is a small average distance between nodes. Watts [50] shows that this combination of clustering and short paths appears in a wide range of natural and technological networks.

Kleinberg [51, 52] examines the problem of finding short paths in small world networks using only local information. The networks in question are augmented lattices similar to those described by Watts and Strogatz: each node u has short-range connections to every node within p lattice steps, as well as q long-range connections such that each long-range neighbour v is selected with a probability proportional to $r^{-\alpha}$, where r is the lattice distance between u and v. Kleinberg shows that no local greedy search algorithm can find a short path between two nodes in polylogarithmic time unless α is equal to the dimensionality of the underlying lattice; such networks are called *navigable small worlds*.

Liben-Nowell *et al.* [53] find that the social network of the LiveJournal website is navigable using a greedy search algorithm that only considers the geographic distance between users, suggesting that navigable small worlds may be useful as synthetic models of social networks.

Scale-Free Networks

A scale-free network has a degree distribution that follows a power law. Many real networks, including social networks, exhibit highly skewed degree distributions that are often said to follow power laws, although the goodness-of-fit may sometimes be overstated [54].

One simple way to generate a scale-free network is to simulate a growth process that attaches each new node to several existing nodes such that the probability of attaching to a node is proportional to its degree [55]. This process is known as *preferential attachment*. Sarshar and Roychowdhury [56] show that networks generated in this way do not retain their scale-free structure if nodes are randomly deleted as the network evolves, so preferential attachment may not be suitable for modelling networks with constantly changing populations. Bauke and Sherrington [57] describe a *local attachment* rule that maintains a power law degree distribution regardless of the rate at which nodes are deleted: each neighbour of a new node is selected by choosing a random neighbour of a randomly chosen node.

Albert *et al.* [58] show that scale-free networks are resilient to the removal of randomly chosen nodes, but vulnerable to attacks that target high-degree nodes, since many links are removed by such attacks. Motter *et al.* [59] find that while long-range links are responsible for the small average distance between nodes in the small world model of Watts and Strogatz, the same is not true for scale-free networks: removing short-range links has a greater effect on the average distance between nodes than removing long-range links, because short-range links tend to connect high-degree nodes and therefore tend to be included in a large number of shortest paths. Thus scale-free networks may be vulnerable to targetted attacks on either links or nodes.

2.1.2 Identity

The open membership policies of many distributed communication networks, such as peer-to-peer overlays and mobile *ad hoc* networks, make it difficult to establish a one-to-one relationship between users and network identities, and this uncertainty can be exploited by attackers.

Some systems use IP addresses to identify users, since most individuals have access to a limited number of addresses. IP addresses are not secure identifiers, however: an attacker who is on the communication path between two hosts may be able to spoof the IP address of each host when communicating with the other [60, 61]. Feamster *et al.* [62] point out than an attacker who controls a firewall, such as the national firewalls discussed in section 2.3.4, can spoof traffic from any address on the other side of the firewall. A powerful attacker might alternatively gain access to a large number of IP addresses by using a botnet [63].

The *Sybil attack* involves the maintenance of multiple identities by a single user. Douceur [64] shows that such attacks are possible in almost any system without a central administrative component. Introducing barriers to the maintenance of multiple identities, such as computational puzzles [65, 66], can limit the extent of Sybil attacks, but cannot prevent

them entirely: if there is significant heterogeneity in nodes' capabilities, a single strong node can maintain as many identities as several weak nodes.

SybilGuard [67] uses the structure of social networks to limit Sybil attacks. Users construct a *web of trust* in which the nodes represent identities and the edges represent trust relationships. An attacker may create any number of Sybil nodes with any number of edges between them, but if few users trust the attacker, the Sybil nodes will be linked to the rest of the network by relatively few edges. SybilGuard uses long pseudo-random routes to sample the web of trust and determine whether a given identity is likely to be a Sybil node.

SybilLimit [68] uses shorter random routes in parallel and provides tighter bounds on the number of Sybil nodes, while Gatekeeper [69] tightens the bounds further. The random routes used by SybilGuard, SybilLimit and Gatekeeper must be recalculated whenever the web of trust changes, and all nodes must be online to participate in the recalculation, so these solutions may not be suitable for highly dynamic networks.

SybilInfer [70] uses random walks and Bayesian inference to detect Sybil nodes in social networks.

Viswanath *et al.* [71] show that SybilGuard, SybilLimit and SybilInfer all work by detecting a cluster of honest nodes around the node performing the evaluation. Standard local clustering algorithms are shown to work equally well; however, all of these approaches break down when the non-Sybil region of the network contains multiple clusters.

Apart from the Sybil attack, other identity-related attacks include *whitewashing*, in which a user changes identities frequently to escape the consequences of past actions [72]; and the *eclipse attack*, which involves manipulating the node discovery process so that victims are disproportionately likely to discover corrupt nodes [73]. Even if victims of the eclipse attack initially have an unbiased view of the network, they can be surrounded by corrupt nodes over time.

Certified Identities

Many systems attempt to deal with identity-related attacks by introducing an online or offline *certificate authority* that issues *certified identities* to authorised users after verifying their real identities. A centralised certificate authority may be considered unsuitable for censorship-resistant communication, since it creates a single point of attack and relies on all users trusting the same authority.

Zhou and Haas [74] describe a way to share responsibility for generating certificates among multiple nodes. The certifying key is split into *shares* using a secret sharing algorithm [75]; each share can only generate partial signatures, several of which must be combined to produce a valid certificate [76]. The key can be re-shared to invalidate the shares of compromised nodes, assuming they can be detected and excluded from the re-sharing process.

Quercia *et al.* [77] show how groups of users can combine partial signatures to create a certified pseudonym for each group member. To prevent an individual from obtaining more than one pseudonym, members verify the owner's identity out-of-band before signing. Certification requests are blinded so that a pseudonym cannot be linked to its owner, except perhaps by observing a new pseudonym becoming active shortly after a new user joins the group.

2.2 Peer-to-Peer Networks

A peer-to-peer network is an Internet overlay composed of connections between autonomous nodes. Some peer-to-peer networks use central servers to coordinate communication between nodes [78, 79]; others construct a multi-hop overlay that allows indirect communication between any pair of nodes. In many systems the overlay is only used to allow nodes that wish to communicate directly to find one another and establish a direct connection.

Peer-to-peer networks are often divided into two families: *unstructured peer-to-peer networks*, in which the topology of the overlay is only loosely controlled, and *structured peer-to-peer networks*, which attempt to guarantee scalability and performance through tighter control of the topology. Two further families have emerged in recent years: *semantic peer-to-peer networks*, which use information about the contents of shared files to guide searches, and *private peer-to-peer networks*, which are restricted to invited users.

Risson and Moors [80] give a comprehensive survey of the peer-to-peer search literature, while Androutsellis-Theotokis and Spinellis [81] survey peer-to-peer content distribution networks. Lua *et al.* [82] provide a general survey of peer-to-peer systems.

2.2.1 Connectivity and Churn

Firewalls and network address translators create significant problems for peer-to-peer networks. As the number of Internet-connected devices increases, a growing proportion of users are separated by 'middleboxes' of one kind or another. Techniques for establishing connections across middleboxes are not perfectly reliable and often require communication with a third party, which may have privacy implications [83, 84, 85].

The unreliability of nodes themselves is also an important design consideration for peer-to-peer networks – in large networks there is constant *churn* as nodes join and leave. Measurement studies by Saroiu *et al.* [86] and Stutzbach *et al.* [87] show that the distribution of session lengths is highly skewed; Bustamente and Qiao [88] and Guha *et al.* [89] find that the distribution follows a power law. The mean session length is on the order of a few hours, with daily and weekly cycles in the number of available nodes [86, 89].

Evans *et al.* [90] distinguish between *join-leave churn*, in which nodes join the network for a single session before leaving permanently, and *leave-join churn*, in which nodes leave the network temporarily and then rejoin. The difference may be important if nodes have persistent identifiers or store information across sessions.

2.2.2 Unstructured Peer-to-Peer Networks

Nodes in the Gnutella file sharing network [91] form an unstructured overlay in which messages are flooded for a limited number of hops. The overlay is primarily used for searching; files are transferred by direct connections between requesters and responders. The original Gnutella software has been withdrawn, but many other file sharing programs implement the Gnutella protocol.

The use of flooding limits Gnutella's scalability: if the query rate is too high then low-capacity nodes begin to drop messages, effectively causing the overlay to fragment [92]. Restricting the range of queries raises the query rate at which fragmentation occurs, at the cost of giving each node access to only a fraction of the files available in the network.

Techniques to improve the scalability of unstructured search overlays include constructing spanning trees [93, 94]; using random walks instead of flooding [95, 96, 97, 98, 99]; using flow control tokens and topology adaptation to avoid forwarding queries to overloaded nodes [96, 97]; replicating information about the files held by neighbouring nodes [100]; and iteratively searching small regions of the overlay [101, 102].

In YAPPERS [103], node and data identifiers are hashed into a small number of *buckets*, and publishers place pointers to their data on local nodes with matching buckets. This reduces the number of nodes that need to be queried when performing an exhaustive search. The protocol requires only local knowledge of the network and is suitable for use over arbitrary topologies. LMS [104] similarly replicates data and queries across multiple locally optimal locations. Each replica is randomly forwarded for a small number of hops and then deterministically forwarded to the nearby node with the identifier that most closely matches its key.

Roussopoulos and Baker [105] propose that information about files should be 'pushed' across the search overlay towards requesters. This will reduce traffic on an overlay connection if the pushed information would have been requested at least once on that connection, so the decision to propagate should be based on expected popularity.

Konspire2b [106] uses a pure push model: nodes subscribe to authenticated broadcast channels and form *ad hoc* distribution trees when broadcasters wish to send files.

BitTorrent [107] is a file distribution protocol in which nodes that are downloading the same file form an unstructured overlay called a *swarm* to exchange pieces of the file. Nodes discover other members of the swarm through a central server or a structured overlay. BitTorrent's incentive mechanism is discussed in section 2.5.4.

Two-Tier Networks

Gnutella takes advantage of heterogeneity among nodes by using a two-tier structure [108]: nodes with high bandwidth that are able to accept incoming connections join the upper tier, while low-bandwidth nodes and those behind firewalls join the lower tier. Nodes in the lower tier only connect to nodes in the upper tier, reducing the load on low-capacity nodes and minimising the impact of firewalls. The proprietary FastTrack protocol used by KaZaA, Grokster and iMesh appears to have a similar two-tier structure [109], as does the Skype protocol [89]. Yang and Garcia-Molina

[110] develop a detailed cost model for a Gnutella-like protocol and use it to derive rules of thumb for the design of two-tier networks.

Ledlie *et al.* [111] describe a two-tier network that uses Bloom filters [112] to summarise the files held by each node; files are identified by their hashes. Nodes organise themselves into trees, with the root of each tree maintaining a combined Bloom filter summarising the files held by the entire tree. The filters are used to direct searches within and between trees.

Scale-Free Peer-to-Peer Networks

Adamic *et al.* [113] present a search algorithm for scale-free peer-to-peer networks: each query contains a list of previously visited nodes and is forwarded to the highest-degree unvisited neighbour of the current node. The algorithm depends on high-degree nodes also having high capacity, since they carry most of the query traffic. The authors argue that power law degree distributions in peer-to-peer networks reveal a power law distribution of capacity. Ripeneau *et al.* [114] find a power law degree distribution in Gnutella, but Stutzbach *et al.* [115] show that this may be a measurement error caused by crawling the network too slowly.

Sarshar *et al.* [116] describe a search algorithm for scale-free networks that uses random walks to propagate queries and lists of files. It is shown that a short random walk on a scale-free network will usually reach a high-degree node, and that the high-degree nodes will usually form a connected subgraph. Probabilistic flooding within this connected subgraph can be expected to reach all of the high-degree nodes using a relatively small number of messages, allowing any query to reach any list of files. As with the algorithm of Adamic *et al.*, the high-degree nodes must have high capacity.

2.2.3 Structured Peer-to-Peer Networks

Structured overlays aim to provide scalable and predictable performance by controlling the topology of the overlay, typically by assigning each node a unique identifier that determines which nodes should be its neighbours.

Many structured overlays implement a *distributed hash table* abstraction for storing arbitrary key/value mappings: the overlay is structured so that all lookups for a given key converge on the same small set of nodes, which are responsible for storing the corresponding value. Lookups can be implemented recursively – the current node forwards the lookup to the next node – or iteratively – the current node returns the address of the next node.

Tapestry [117] and Pastry [118] are structured overlays based on the prefix routing algorithm of Plaxton *et al.* [119]. Node and data identifiers are expressed as d-digit numbers in base b, and each node maintains connections to db neighbours chosen so that any lookup can be forwarded to a node that matches the requested identifier in more leading digits than the current node. Law and Siu [120] show how to construct and maintain regular random networks that can be used for prefix routing.

Prefix routing allows considerable flexibility in the choice of those neighbours that have short matching prefixes. Castro *et al.* [121] use this flexibility to match the topology of the overlay to the topology of the underlying network, minimising the bandwidth overhead and latency of the overlay, while Kademlia [122] uses it to tolerate churn by exploiting the discovery of Saroiu *et al.* [86] that old nodes are likely to remain in a peer-to-peer network for longer than new nodes. Each Kademlia node sorts its potential neighbours by age, with new nodes added to the tail of the list and unresponsive nodes flagged and eventually removed. Node addresses are learned from queries, so nodes that issue large numbers of queries will tend to receive large numbers of queries in return, as long as they remain responsive. This could allow Kademlia to benefit from heterogeneity in node capabilities, whereas most other distributed hash tables aim for uniform load balancing. Kademlia overlays scale well in practice and are used in several large file sharing networks [123, 124, 125].

Chord [126] is a distributed hash table in which node and data identifiers are hashed into a circular *key space*. Each node is assigned a location by hashing its IP address, and takes responsibility for the arc of the key space between itself and the nearest anticlockwise node. Each node maintains connections to the nodes responsible for $O(logN)$ keys chosen so that the clockwise distance to the first key is half the circumference of the circle, the distance to the second is a quarter of the circumference, and so on. Nodes periodically use a *stabilisation protocol* to ensure they are connected to the correct neighbours.

CAN [127] uses a multidimensional key space corresponding to the surface of a torus rather than the perimeter of a circle. Nodes take the latency of links into account when choosing locations, with the aim of decreasing the load on the underlying network by assigning adjacent regions to nearby nodes.

Freedman and Vingralek [128] propose a *distributed trie* that caches key/value mappings on nodes that request them. Routing information is updated lazily by piggybacking updates on responses to queries.

Most structured overlay designs assume that nodes join an established network by contacting an existing member; they do not address the problem of merging multi-node components, which might be necessary if the network is temporarily partitioned, or if a link is created between two established overlays. SkipNet [129] is a structured overlay that tolerates network partitions by arranging nodes in the overlay according to their DNS names. Requests for an item located within the same organisation as the requester do not pass outside the organisation, so local queries can be routed while the organisation is disconnected from the rest of the network, and the disconnected segment can later reconnect to the rest of the overlay.

Firewalls and other connectivity restrictions can be particularly problematic for structured overlays, which typically require specific nodes to connect to one another. Sandberg [130] describes a way to assign node identifiers in any navigable small world network to support efficient lookups without modifying the topology. Virtual Ring Routing [131] implements a distributed hash table over an arbitrary topology by adding virtual links between nodes that may be separated by multiple hops in the underlying network.

Unmanaged Internet Protocol (UIP) [132] uses a prefix routing algorithm for packet-switching over a mixture of physical and overlay links. Each node is identified by the hash of its public key, as in the Host Identity Protocol [133], allowing end-to-end authentication and encryption without a central certificate authority.

Techniques for coping with faulty or corrupt nodes in structured overlays are discussed in section 2.3.2.

2.2.4 Semantic Peer-to-Peer Networks

Semantic peer-to-peer networks use information about the content of shared files to improve the efficiency of searches, for example by adapting the overlay to bring users with shared interests closer together [134, 135, 136, 137, 138, 139], by routing queries according to the keywords they contain [140, 141, 142], or by introducing additional metadata such as directory trees [143, 144, 145] or topic hierarchies [146, 147, 148, 149, 150, 151, 152].

Because they rely on exposing information about users' interests, semantic overlays are not generally suitable for private, censorship-resistant communication. Two interesting exceptions, Clouds and UQDT, are described in section 2.4.8.

2.2.5 Private Peer-to-Peer Networks

The peer-to-peer networks discussed so far are designed to be open to the public: anyone can join a network simply by obtaining the addresses of other participants, which in many cases are available from public servers. Even systems designed to protect the privacy of their users often have open membership policies (see sections 2.4.5 and 2.4.6). This openness enables wide participation, ensuring that a large amount of content is available in file sharing networks, for example, but it also allows attackers to monitor, join and disrupt peer-to-peer networks [153, 154, 155, 156, 157].

A *private peer-to-peer network* can be defined as an Internet overlay in which the resources and infrastructure are provided by the users, and new users may only join the network by personal invitation. This definition excludes systems that rely on public servers, but it does not necessarily imply decentralisation – some private peer-to-peer networks use central servers, but access to those servers is restricted to invited users, and the servers are operated by users of the network.

Several public peer-to-peer networks have been shut down by attacking their central servers [78, 79, 156], leading to the perception that centralised designs are fragile and should be avoided. However, the risks may be different in private networks, where servers can be hidden from untrusted parties. Centralisation can make it easier for users to manage their identities, exchange cryptographic keys, and learn the current addresses of other users, provided all users trust the operator of the server – a requirement that may become increasingly problematic as the network grows.

Some private peer-to-peer networks allow direct connections between any pair of users, while others only allow direct connections between users who know one another. We refer to the former as *group-based networks* and the latter

as *friend-to-friend networks* [158, 159]. This distinction becomes important in larger networks, because any user may invite a friend into the network who does not know all of the other users. In a group-based network, the new user will be able to connect to any existing user, so group-based networks become less private as they grow, whereas friend-to-friend networks can remain private at any scale.

Biddle *et al.* [160] refer to private networks collectively as "the darknet" – not a single global network, but rather a patchwork of local networks, technologically separate but socially connected through users who belong to more than one network.

Middleboxes and dynamic network addresses are especially problematic for private peer-to-peer networks, since they may prevent users from reconnecting to the network after spending time offline. In a group-based network, at least one member of the group must have a stable address; the current addresses of other members can be learned from that member. In a friend-to-friend network, every user needs at least one friend with a stable address. Some systems cope with this problem by using internal or external lookup services, such as distributed hash tables, for middlebox traversal and address discovery; others require users to exchange updated addresses out-of-band if they cannot reconnect. By monitoring or participating in a lookup service, an observer may be able to discover who connects to whom, so the decision to use such services involves a tradeoff between privacy and ease of use.

Confidentiality and authentication are areas where private peer-to-peer networks have an advantage over public networks. Because the users know one another, it is feasible for them to exchange cryptographic keys out-of-band; private peer-to-peer networks could even be bootstrapped using existing keys and trust relationships, such as those recorded in the PGP web of trust [161].

One serious disadvantage of friend-to-friend networks is that they reveal the users' social relationships to anyone who can observe the overlay topology; to a lesser extent the same is true of group-based networks. Social network analysis can then be used to target important members of the network [22, 26, 162, 163]. Narayanan and Shmatikov [164] show that many of the users in an anonymised social network can be identified by comparing the network's structure to that of a separate, non-anonymised network, even if there is limited overlap between the two networks' sets of users.

Pouwelse *et al.* [139] identify five research challenges for peer-to-peer networks: decentralising functionality; maintaining availability in the face of churn; ensuring the integrity of data and metadata; creating incentives to contribute resources; and achieving network transparency across middleboxes. They suggest that all five challenges can be addressed by encouraging users to form cooperative social groups. Nagaraja [165] and Li and Dabek [166] also argue that users of friend-to-friend networks will be more willing to contribute resources than users of public networks. Zhang *et al.* [167] find that private BitTorrent servers can maintain higher sharing ratios than public servers, by evicting users with low ratios and giving preferential treatment to users with high ratios.

Despite being limited to invited users, private peer-to-peer networks may be vulnerable to Sybil attacks and whitewashing. For example, an attacker might automatically 'invite' Sybil identities into a group more quickly than the other users can evict them. Such attacks can be prevented by requiring a certain fraction of existing members to approve each invitation [77], but this approach does not scale to large groups where not all users know one another.

Friend-to-friend connections are not a panacea for identity-related attacks, but they guarantee that every identity within a local scope corresponds to a different individual, which prevents whitewashing; as discussed in section 2.1.2, it might also be possible to use the structure of social networks to limit the impact of Sybil attacks.

The strongest protection against identity-related attacks comes from central servers, which allow the server's operator to verify that no individual uses more than one identity.

Group-Based Networks

One of the first private peer-to-peer networks was Groove [168], a group-based collaboration tool for creating 'shared spaces'. Each member of the group maintains a copy of the shared space's state, and encrypted updates are transmitted to other members when the state changes. Members who are unable to communicate directly can exchange messages through dedicated relays. Users can only join groups by invitation, and any member can evict any other member from a group by creating a new group key and transmitting it to all members except the evicted member.

WASTE [169, 170] is a group-based network that uses flooding and reverse path forwarding to implement file sharing and chat. Nodes can relay one another's messages if all-to-all connectivity is not possible. Links are encrypted and optionally padded to a constant traffic level, but there is no end-to-end encryption or authentication, so users can eavesdrop on one another and spoof messages.

Strufe and Reschke [171] describe a peer-to-peer overlay that stores group information as well as file information, so users can create authenticated groups for file sharing and instant messaging. Each group has a single owner who is responsible for adding and removing members. Other group-based file sharing systems include Shinkuro [172], PowerFolder [173] and Octopod [174].

N2N [175] is a group-based virtual private network that uses relay nodes for address discovery and middlebox traversal.

Direct Connect [176, 177] requires one member of each group to run a server, which is used for address discovery, keyword searches and chat.

Friend-to-Friend Networks

Turtle [178] is a friend-to-friend file sharing network designed for censorship-resistance. Searches are flooded across a multi-hop overlay, search results are forwarded back along the reverse path, and virtual circuits can be established for anonymous file transfer. The virtual circuit architecture is also capable of supporting other applications [179].

Turtle uses a novel key agreement protocol in which friends exchange personal questions over an insecure channel, the answers to which are assumed to be known to both friends but not to eavesdroppers. This avoids the need for out-of-band key exchange, but the strength of the resulting keys will depend on the extent of the eavesdropper's knowledge about the users.

SockeToome [180] enables friend-to-friend file transfers between users with dynamic IP addresses, although it is arguably not a peer-to-peer network since no overlay is constructed. Easter [181] uses email as a substrate for friend-to-friend file sharing, which makes it possible to circumvent many firewalls.

In anoNet [182], virtual private network (VPN) connections between friends are used to construct an encrypted overlay. The overlay uses standard Internet protocols such as BGP, and even has an internal DNS hierarchy. A reserved network prefix is used to avoid accidentally routing packets between the overlay and the public Internet.

Figueiredo *et al.* [183] describe how to establish VPN connections between users of social networking websites. The websites act as trusted third parties for key exchange, so it might be argued that this is not strictly a friend-to-friend design. More recent versions of the system can use public or private servers for key exchange [184].

CSpace [185] is a general-purpose friend-to-friend connection service based on a distributed hash table. Participants in the distributed hash table can observe who connects to whom, which may have implications for privacy. PeerSON [186], a friend-to-friend social networking service, uses a distributed hash table for address discovery and to store messages for users who are offline.

The Retroshare [187] instant messaging and file sharing network also uses a distributed hash table for address discovery. Users can communicate indirectly through mutual friends, which may allow them to build up trust in one another before requesting direct connections. Galet [188], Alliance [189], and Cryptic6 [190] allow friends-of-friends to communicate in a similar way, but they do not use distributed hash tables or relay servers for address discovery, so addresses and encryption keys must be exchanged out-of-band or through mutually trusted friends.

Tsnecv [191] is a private peer-to-peer system for local area networks in which users may assign different access levels to friends, friends-of-friends and strangers. Gazzera [192] also supports different trust levels, allowing the friends of highly trusted friends to connect directly. Users who are not directly connected can share files anonymously through the friend-to-friend overlay. Tonika [193] is a distributed social networking service in which users may communicate indirectly across a friend-to-friend overlay.

F2F [166] and Friendstore [194] are peer-to-peer backup systems in which friends store each other's data.

Vasserman *et al.* [195] describe three *membership-concealing overlays* that support scalable communication while revealing each user's participation to only a constant number of other users, all of whom are the user's friends or friends-of-friends. The networks are double overlays: the upper overlay is a distributed hash table, each link of which comprises one or more links in the lower, friend-to-friend overlay. Onion encryption (see section 2.4.3) conceals the content of messages in the upper overlay from intermediate nodes in the lower overlay.

A central authority is required for identity management and overlay construction, which proceeds in two phases: first the central authority builds the friend-to-friend overlay, then each node uses flooding to discover static routes to its neighbours in the distributed hash table. The routes do not adapt to changes in the social network, so routing breaks down if too many nodes go offline.

To ensure a well-connected network and prevent targetted attacks against high-degree nodes, each node must have between $\lceil k/2 \rceil$ and k neighbours in the friend-to-friend overlay, for some system-wide constant k. In a sampled topology from the Orkut social networking website, this restriction prevented 15% of users from joining the overlay.

The peer-to-peer anonymity systems Freenet, GNUnet, MUTE and OneSwarm can be configured to connect only to trusted friends (see section 2.4.6).

2.3 Availability

Censorship-resistance can be seen as an extreme example of the well-studied problem of *availability*: that is, ensuring that information remains reachable in the face of intentional attacks and accidental failures. Availability requires robust storage of information and robust communication between those who are storing the information and those who wish to access it.

2.3.1 Robust Client-Server Storage

Anderson [196] describes the Eternity Service, a network of servers that store anonymously published files for a publisher-specified period in exchange for anonymous payment in a digital currency. Publishers cannot discover which servers are storing their files. Anderson argues that censorship-resistant systems should not allow files to be modified or deleted by their publishers before their specified expiry times, since this would encourage censors to identify publishers and force them to remove their files.

WikiLeaks [197] is a website that publishes leaked official documents, which can be submitted anonymously through Tor (see section 2.4.4) or by postal mail. The WikiLeaks servers are spread across multiple jurisdictions. Martus [198] enables human rights workers to store encrypted backups of their records on servers hosted around the world. Publishers can 'seal' their records so that they cannot later be forced to delete them.

In Free Haven [199], publishers, readers and storage servers remain anonymous to one another by communicating through a mix network (see section 2.4.3). Files are published by uploading them to any server, which splits them into redundant *shares* that it trades with other servers. Download requests can be sent to any server; they are flooded to all other servers, which send their shares through the mix network to the requester. Servers test each other to ensure shares are not discarded before their publisher-specified expiry times, but since the servers are anonymous to one another, any that are caught discarding shares may be able to re-enter the network under new identities. The use of flooding to distribute download requests and server announcements could limit Free Haven's scalability and expose it to denial-of-service attacks.

In Publius [200, 201], a small number of servers store encrypted files on behalf of clients. To publish a file, a client encrypts it with a secret key and then splits the key into n shares, any $k < n$ of which can be used to reconstruct the key. Each share is stored on a different server together with a copy of the encrypted file. No single server can decrypt the file it is holding, but a client that retrieves enough shares (using a multi-part URL which is assumed to be unknown to the servers) can reconstruct the key and thereby decrypt and verify the file. The list of servers must be known and agreed by all clients, so central administration is required.

Servers in Tangler [202] can join and leave the network dynamically, but there must still be general agreement about the list of participating servers. Server and data identifiers are hashed into a circular key space and each server takes responsibility for several regions of the key space selected by hashing its public key and the date, so that each region changes hands periodically. A server can make itself responsible for an arbitrary region if it can create a suitable public key.

Tangler's most interesting feature is *document entanglement*, a way of combining a new file with previously published files using a secret sharing algorithm [75] so that reconstructing the new file requires access to the files with which it was entangled. This makes it difficult to delete specific files from the system without affecting others, although a file is not protected until newer files have been entangled with it. Dagster [203] and the Owner-Free File System [204] use similar techniques to make it difficult to delete specific files.

Tahoe [205] is a distributed storage system that uses erasure coding [206] to provide robustness against failed or corrupt servers. Cryptographic keys can be used to grant read-only or read-write access to files.

Song *et al.* [207] describe methods for searching an encrypted file without revealing the search terms or the contents of the file to the server performing the search. Blindfold [208] achieves a similar goal while preventing automated dictionary attacks, by using two servers that do not communicate directly and which may be located in different jurisdictions: the *index server* stores mappings from hashed keywords to captchas [209], while the *content server* stores mappings from hashed captcha solutions to files encrypted with the captcha solutions and the original keywords, which are not revealed to either server. The use of captchas prevents an attacker from enumerating all of the content in the system, but a censor could force the index server to remove its mappings for certain keywords without needing jurisdiction over the content server.

2.3.2 Robust Peer-to-Peer Storage

Usenet [210, 211] is a public discussion forum divided by topic into *newsgroups*. Messages are published, propagated, and read using a simple peer-to-peer flooding mechanism. Back [212] describes a system loosely based on Anderson's Eternity Service design that uses Usenet to store anonymously published web pages.

Usenet's message identifiers are not cryptographically secure, so a message can be prevented from propagating by sending out a different message with the same identifier. Messages can also be 'cancelled' by their authors, and it is possible to forge cancellation requests. Nodes generally store the most recent messages in each newsgroup and discard old messages to save space, so an attacker may be able to force a message to expire by sending a large volume of messages to the same newsgroup.

PAST [213] is a distributed storage system based on the Pastry structured overlay. Files are replicated across multiple nodes to keep them available in the face of attacks and failures; requests are routed to the nearest replica according to a metric such as geographic distance or network latency. To prevent nodes from using more storage than they contribute, PAST has a quota system based on certified identities.

FARSITE [214] also uses a certificate authority to control access to a distributed file system, while OceanStore [215] takes a federated approach in which multiple operators buy and sell capacity among themselves. CFS [216] enforces a storage quota for each IP address, which could make it difficult to publish files anonymously, and would be open to abuse by an attacker with access to a large number of IP addresses.

HiveCache [217] is a distributed backup system that uses content hashes to identify files on a local area network and distributes copies of unique files to machines with free storage space. Mnet [218] is a wide-area distributed storage system based on similar principles. Nodes in the Pastiche backup system [219] use a distributed hash table to find 'backup buddies' with which they have many files in common.

Mnemosyne [220] aims to give nodes *plausible deniability* regarding the files they are storing: files are divided into blocks, redundantly encoded [221], encrypted, and stored in a distributed hash table. Blocks may collide, so files will eventually be corrupted if they are not republished. Mnemosyne does not use certified identities and would be vulnerable to Sybil attacks.

Serjantov [222] suggests publishing and retrieving encrypted files in PAST through a mix network (see section 2.4.3) so that servers do not know what they are storing.

Castro *et al.* [223] consider attacks against distributed hash tables, such as a node adopting a particular identifier to make itself responsible for a particular key, which would allow the node to delete, modify or monitor the corresponding value. They propose a solution based on certified identities and redundant parallel lookups that is suitable for storing values that can be verified by the requester.

Myrmic [224] uses *neighbourhood certificates* signed by a central authority to verify that structured overlay lookups terminate at the correct nodes.

Kuhn *et al.* [225] describe a distributed hash table that is robust to the adversarial removal of $O(logN)$ nodes, while Awerbuch and Scheideler [226] present join and leave algorithms that prevent an attacker who can force any sequence of nodes to leave and rejoin a distributed hash table from controlling arbitrarily large regions of the key space. Fiat and Saia [227] describe a distributed hash table that uses clustering to survive the adversarial removal of up to half the nodes; however, without certified identities the design would be vulnerable to a Sybil attack.

Sit and Morris [228] describe some security measures for distributed hash tables that do not require certified identities, such as verifying node addresses to prevent IP spoofing, checking that iterative lookups make progress towards the key at every hop, using multiple hash functions to store redundant copies of data under different keys, and assigning

node identifiers in a verifiable way. Wang *et al.* [229] show that a popular implementation of Kademlia is vulnerable to attack because nodes' identifiers are not verifiable.

Halo [230] provides robust lookups over an unmodified distributed hash table by looking up nodes that are likely to have direct links to the target node due to the structure of the overlay. Unlike simple parallel lookups, these lookups are unlikely to converge on the same set of potentially corrupt nodes until the last hop.

SPROUT [231] augments a distributed hash table with connections between users who trust one another. Lookups use trusted connections where possible, falling back to untrusted connections if necessary.

Danezis *et al.* [232] address Sybil attacks against distributed hash tables by considering the *introduction graph* of the network, where an edge exists between two nodes if one of them introduced the other to the network. A node performing a lookup attempts to avoid relying on nodes that are linked to it through similar paths in the introduction graph.

Lesniewski-Laas [233] describes a Sybil-proof distributed hash table based on a social network. Each node keeps a *successor table* containing its successors in the circular key space and a *finger table* containing random nodes selected through short random walks on the social network. The nodes in the finger table are likely to be honest if the number of social relationships between honest nodes and Sybil nodes is small (this is similar to the clustering-based Sybil defenses described in section 2.1.2). To perform a lookup, the requester tries the nodes in its finger table in order of increasing distance from the target until an honest node with the target in its successor table is found.

Another way to limit the number of Sybil identities is to introduce an artificial cost for creating or maintaining identities. Borisov [234] describes a protocol for distributing computational puzzles in a distributed hash table such that a node performing an iterative lookup can verify that every node it contacts has recently solved a puzzle that was partly determined by the requester. New nodes are avoided during lookups until they have had time to receive and solve puzzles from the rest of the network.

Endsuleit and Mie [235] use the secure lookup scheme of Castro *et al.* in a censorship-resistant publishing system in which files are identified by sets of keywords. A file is published by encrypting it with a key derived from its keywords and splitting it into two fragments, both of which are needed to reconstruct it. The fragments are signed with the publisher's private key, which allows the publisher to update or delete the file, and are stored in a distributed hash table at independent locations that are derived from the keywords. Readers only need to know the keywords to retrieve, reconstruct and decrypt the file; however, unless a reader has the publisher's public key there is no way to tell whether a corrupt node has modified the file.

To prevent the certificate authority from linking users to their files or nodes, signatures of publisher and node identifiers are blinded. Unfortunately this violates the requirement of Castro *et al.* that nodes should not be able to choose their own identifiers.

The deterministic placement of data in distributed hash tables may be useful for censorship-resistance if it prevents corrupt nodes from making themselves responsible for storing particular files. Danezis and Anderson [236] argue, on the other hand, that users may have a greater incentive to participate in censorship-resistant systems if they can choose to store and redistribute files that match their personal beliefs.

2.3.3 Robust Routing

Surviving external attacks has been a design goal since the first routing protocols were developed [44], yet many protocols are vulnerable to internal attacks by participating nodes. In this section we consider internal attacks against routing protocols and discuss protocols designed to resist disruption by faulty or misbehaving nodes.

2.3.3.1 Mobile Ad Hoc Networks

A mobile *ad hoc* network is a group of wireless nodes that communicate among themselves without infrastructure or manual configuration. Routing protocols for such networks must be able to cope with a constantly changing topology. Most protocols use one of two approaches: *table-driven* or *proactive* protocols such as DSDV [237] and OLSR [238] attempt to maintain an up-to-date route to every destination, while *on-demand* or *reactive* protocols such as DSR [239, 240] and AODV [241] do not discover routes until they are needed. Hybrid protocols such as ZRP [242] combine elements of both approaches.

Many on-demand protocols use flooding for route discovery, which has the advantage that it finds the quickest route from the source to the destination. However, flooding is inefficient in areas of high node density. Several probabilistic and deterministic techniques have been proposed to reduce broadcast redundancy [243, 244, 245, 246]. Wu and Li [247], and Das and Bharghavan [248], consider techniques for selecting sets of nodes to act as routing backbones for *ad hoc* networks.

Perkins [249] and Hong *et al.* [250] survey the *ad hoc* routing literature, while Broch *et al.* [251] and Lee *et al.* [252] compare the performance of various *ad hoc* routing protocols at different levels of mobility.

2.3.3.2 Probabilistic Routing Protocols

Another way to discover routes across a dynamic network is to forward packets probabilistically, using feedback to adjust the probabilities in order to optimise some metric such as reliability or latency.

ARA [253] is a probabilistic routing protocol for *ad hoc* networks inspired by the collective foraging behaviour of ants. Parallel routes are discovered on demand by flooding, and data packets are forwarded probabilistically along them, reinforcing the routes they follow unless a loop or a broken link is detected.

Several other ant-based routing protocols have been proposed for wired and wireless networks [254, 255, 256, 257, 258, 259, 260]. MUTE, an anonymous peer-to-peer network, also uses ant-inspired routing (see section 2.4.6).

Q-routing [261] uses reinforcement learning to find fast routes across dynamic networks. Xie *et al.* [262] show that a protocol based on Q-routing converges to an equilibrium where no source can unilaterally improve the latency of its traffic by choosing a different route. Cognitive packet networks [263] use a variety of learning techniques to find routes that meet user-specified quality of service constraints.

Electric routing [264] is a probabilistic protocol that calculates the 'electric potential' of each node and the 'current' on each link with respect to each destination. When a node needs to route a message towards a destination, it chooses an outgoing link probabilistically according to the current flowing along each link. This is essentially a probabilistic form of distance vector routing.

2.3.3.3 Attacks Against Routing Protocols

In all of the protocols described above, nodes base their decisions on the assumption that other nodes will behave correctly, making the protocols unsuitable for use in adversarial environments.

Many on-demand protocols use flooding to discover routes; to reduce collisions, each node waits for a random period before propagating a flooded route request packet. In the *rushing attack*, the attacker propagates flooded packets immediately, causing a disproportionate number of routes to pass through corrupt nodes [265].

Other protocols use explicit distance or cost metrics to select routes; an attacker may be able to manipulate route selection by advertising false metrics. Values that are implicitly used as metrics, such as the number of hops in a route, can also be manipulated. In the *tunnelling attack*, two corrupt nodes encapsulate data passing between them to create the appearance that they are neighbours [266]. The route between the corrupt nodes appears shorter than it actually is, which will cause many routing protocols to prefer it.

In the related *wormhole attack* against wireless networks, the attacker relays signals from one point to another so that two nodes believe themselves to be neighbours; unlike the tunnelling attack, this can be done without compromising any nodes. Wormhole attacks can be prevented with tight clock synchronisation [267] or by accurately measuring the round-trip time at the link layer [268].

In the *black hole* attack, the attacker behaves correctly during route discovery and route maintenance but silently drops data packets [269].

2.3.3.4 Byzantine Robustness

Perlman [270] considers the problem of routing in the presence of faults that produce arbitrarily complex behaviour, which are known as *Byzantine faults* [271]. A network layer is said to be *Byzantine robust* if it can deliver any packet

provided there is a fault-free route between the source and the destination. This implies that faulty components must not be able to interfere with the discovery or use of fault-free routes.

Perlman presents a Byzantine robust flooding protocol that uses reserved buffers, round-robin scheduling, non-wrapping sequence numbers and digital signatures to ensure that the most recent packet from every authorised source reaches every node to which a fault-free route exists. To allocate resources correctly, every node must have a complete list of authorised sources and their public keys. These lists are distributed by one or more *trusted nodes* using the robust flooding protocol; the public keys of the trusted nodes are manually installed at each node.

If a trusted node develops a Byzantine fault, the public key lists received from different trusted nodes may disagree. The definition of Byzantine robustness used in this situation is more restrictive: a node is only considered non-faulty if there is a non-faulty route from at least one non-faulty trusted node to the node in question. To limit the damage that can be caused by faulty trusted nodes, a node will not accept a public key list that contains too many entries, or a list that does not contain its own identity and public key.

Perlman's flooding protocol is robust but inefficient – every packet must be transmitted and authenticated on every link. A second protocol, robust link state routing, provides higher efficiency but lower robustness. Every node uses the robust flooding protocol to broadcast a signed link state packet listing its neighbours. As with the robust flooding protocol, the public key lists distributed by different trusted nodes may not agree, so each node constructs a separate view of the network for each public key list. A link between two nodes is added to a view if both nodes report the link and both nodes are in the view's public key list. Each view is then used to find node-disjoint routes between the source and the destination, and signed packets are source routed along these routes.

The link state protocol is less robust than the flooding protocol because a fault-free route will be rejected if it shares one or more nodes with a faulty route.

2.3.3.5 Fault Detection

Avramopoulos *et al.* [272, 273, 274] propose using message authentication codes instead of digital signatures to authenticate packets in Perlman's robust link state routing. This decreases the computational overhead but increases the bandwidth overhead, as a separate message authentication code is needed for each node along the route. Acknowledgements, timeouts and fault announcements are proposed to detect and avoid faulty links. The timeout at each node is based on the maximum round-trip time, which grows linearly with the number of nodes in the network because every link has a reserved buffer for every node.

ODSBR [269, 275] uses probe messages to detect and avoid faulty links, protecting against black hole, rushing and tunnelling attacks.

Sprout [276] uses Perlman's robust flooding protocol to distributed signed link state packets for probabilistic source routing. Every minimal route between the source and destination is chosen with non-zero probability, where a route is minimal if no node can be removed from it without disconnecting the route. End-to-end signed acknowledgements are used to find fast, reliable routes, which are subsequently selected with higher probability.

SMT [277] uses an information dispersal algorithm [221] to split packets into redundant pieces that are sent over node-disjoint routes. The reliability of each route is measured using end-to-end acknowledgements and the encoding is adjusted accordingly.

Awerbuch *et al.* [278] describe a protocol in which sources use reinforcement learning to select routes based on secure acknowledgements from relays and destinations.

All of these protocols require certified identities to prevent an attacker from advertising an unlimited number of nodes and links, which the attacker can then discard when they are detected as faulty.

2.3.3.6 Secure Ad Hoc Routing Protocols

There have been many proposals to improve the robustness of routing protocols for mobile *ad hoc* networks. Several of the proposals use *hash chains* and *delayed disclosure* as security primitives.

Hash chains were first proposed by Lamport [279]; given a random value r and a one-way hash function h, the i^{th} element of the chain is $h^i(r)$. Due to the one-way nature of h, anyone can verify that a given element belongs to the same chain as any later element, but no earlier element of the chain can be guessed.

In delayed disclosure [280], each element of a hash chain is used as a secret authentication key for a single packet, then revealed once the packet has been delivered. This allows message authentication codes to be used in place of expensive digital signatures, but requires clock synchronisation between the sources and destinations of packets.

Link State Procotols

Hauser *et al.* [281] use hash chains to authenticate link state updates; separate chains are used to indicate whether a link is active or inactive. Cheung [282] suggests using hash chains and delayed disclosure to authenticate link state updates, which requires loose clock synchronisation and optimistic acceptance of unverified updates. Nodes in SLSP [283] maintain topological information for a local zone; hash chains are used to prevent signed link state updates from propagating out of the zone.

All of these protocols require certified identities, and all are vulnerable to tunnelling and black hole attacks.

Distance Vector Protocols

Smith *et al.* [284] suggest using digital signatures to authenticate the immutable fields of distance vector updates and a signed predecessor field to authenticate the mutable hop count field. SAODV [285, 286] similarly uses digital signatures to authenticate immutable fields. Hash chains are used to prevent corrupt nodes from decreasing the hop count, although they cannot be forced to increment it. SEAD [287] uses hash chains in place of destination sequence numbers.

As with the link state protocols, all of these protocols require certified identities and are vulnerable to tunnelling and black hole attacks.

Source Routing Protocols

Ariadne [288] uses hash chains and delayed disclosure to authenticate an on-demand source routing protocol. This requires loose clock synchronisation and certified identities.

Papadimitratos and Haas [289] describe an on-demand source routing protocol in which route requests and replies are authenticated end-to-end using message authentication codes. The source and destination do not need to share keys with intermediate nodes, so certified identities are not necessarily required. However, the protocol is vulnerable to tunnelling and black hole attacks.

ARAN [266] is an on-demand protocol that finds the quickest (rather than the shortest) route from the source to the destination, preventing tunnelling attacks. Certified identities are required, and black hole attacks are possible.

Jeong *et al.* [290] use hash chains to prevent an attack against on-demand route discovery in which an attacker replies on behalf of the destination, which they call an I^2 *black hole attack*. Their mechanism does not defend against black hole attacks in general.

2.3.4 Circumvention Systems

Many governments have in recent years begun to filter international Internet traffic, preventing their citizens from accessing certain IP addresses, domain names or URLs, and in some cases terminating connections that contain certain keywords [35, 36]. In response, various systems for circumventing censorship firewalls have been devised, most of which depend on users connecting to censored servers via *proxy servers* [291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301]. Roberts *et al.* [302] evaluate the security, performance and usability of several deployed circumvention systems.

Since proxies can in turn be blocked by censors, circumvention systems face the problem of allowing users, but not censors, to discover proxies, even though censors may pose as users. Feamster *et al.* [62] call this the *proxy discovery problem*. Solutions include privately sharing proxy addresses with trusted friends [292, 297, 303, 304]; distributing proxy addresses by email [292, 304]; requiring clients to solve computational puzzles [62]; requiring users to solve captchas [305]; and using the structure of social networks to limit the number of proxies an attacker can discover [306].

Köpsell and Hillig [305] argue that designers of circumvention systems should start with simple designs and see how censors respond before making improvements. This assumes, however, that users are not placed in danger when a censor defeats a deployed design.

Infranet [303] disguises proxy connections as ordinary web browsing traffic, encoding clients' requests for censored information in sequences of requests for uncensored information and steganographically concealing the censored responses in uncensored images. Haystack [307] appears to use similar techniques, although no details of its design have been published.

Collage [308] avoids the need for proxies altogether by using steganography to pass messages through websites that host user-generated content. Clayton *et al.* [309] show the keyword filter of the Chinese national firewall can be circumvented without a proxy by ignoring TCP reset packets.

Many of the anonymous communication systems described in sections 2.4.4 and 2.4.5 can also be used for firewall circumvention.

2.4 Anonymity

Even if a communication network is robust, it is not censorship-resistant if the people who wish to use it can be identified and punished for doing so. Anonymous communication systems aim to conceal the identities of communicating parties and the relationships between them.

Pfitzmann and Köhntopp [310] define anonymity as "the state of not being identifiable within a set of subjects, the *anonymity set*". A more general property, *unlinkability*, refers to an observer's inability to connect actors or actions to other actors or actions. In particular, *sender-recipient unlinkability* means that an observer cannot determine whether two people have communicated [311].

Many anonymous communication systems support cryptographic *pseudonyms*, which allow users to establish long-term relationships without knowing one another's real identities. A pseudonym may allow an observer to link a series of actions to the same unknown actor, which has important implications for anonymity, because observers can often learn more from a series of linked actions than from any single action.

Dingledine and Mathewson [312] discuss the relationship between usability, configurability and security in anonymising networks. They argue that software that is more usable will tend to attract more users, increasing the size of each user's anonymity set. The anonymity set is said to be *partitioned* if an observer can distinguish between two sets of users based on their behaviour. To avoid partitioning, it is argued that anonymous communication software should have few configuration options.

Borisov *et al.* [313] show the effectiveness of selective denial-of-service attacks against a wide range of anonymity systems. Their strategy is to block communication attempts that the attacker cannot monitor, forcing users to try again until the attacker succeeds in monitoring them.

Danezis and Díaz [314], and Edman and Yener [315], survey the state of the art in anonymous communication.

2.4.1 Measuring Anonymity

Several ways have been proposed to measure the degree of anonymity provided by a communication system. In all of them, anonymity is measured from the perspective of a particular observer, and an action may have different degrees of anonymity from the perspectives of different observers.

The simplest measure of anonymity is the size of the anonymity set – that is, the number of actors who may have performed a given action [316]. A system may be described as *k-anonymous* if the anonymity set of every action contains at least k actors [317]. Berthold *et al.* [318] measure the degree of anonymity in bits, as the base two logarithm of the size of the anonymity set.

Reiter and Rubin [319] consider the degree of anonymity as a continuum between *absolute privacy* and *provable exposure*. Two interesting points along this continuum are *probable innocence*, where the observer believes that a given actor is less likely to have performed a given action than not, and *possible innocence*, where the observer believes that there is a non-negligible probability that the actor did not perform the action.

Díaz *et al.* [320] point out that for a system with a large number of actors, the largest probability assigned to any actor may be smaller than $1/2$, which is the threshold for probable innocence, yet much larger than the probability assigned to any other actor. Thus the absolute probability assigned to an actor may not be sufficient to describe the actor's degree of exposure. Serjantov and Danezis [321] suggest defining anonymity as "the amount of information the attacker is missing to uniquely identify an actor's link to an action" [321]. This is equal to the *entropy of the probability distribution* assigned by the observer to the set of actors. The entropy of the distribution can be divided by the base two logarithm of the number of actors, which is the maximum possible entropy, to give a value between zero, meaning that the correct actor is identified with certainty, and one, meaning that all actors are assigned equal probabilities [320].

Tóth *et al.* [322] argue that the observer's goal may simply be to assign a high probability to the correct actor, regardless of how the other probabilities are distributed. They define a communication system as *source-hiding with parameter* p if no possible sender of a message is assigned a probability higher than p, and *destination-hiding with parameter* p if the same applies to recipients. In a system that is source-hiding with parameter p, the entropy of the observer's probability distribution is at least that of a uniform distribution over $1/p$ possible senders.

Moskowitz *et al.* [323] and Zhu and Bettati [324] view anonymous communication systems as *covert channels* [325] that may leak information about anonymous actions to an observer. The *mutual information* between the anonymous actions and the observations is a measure of how much the observer can learn, and the worst-case mutual information gives the channel capacity. Chatzikokolakis *et al.* [326] present a simulation-based method for calculating the capacity of such channels; the number of samples required is proportional to the number of possible hidden actions times the number of possible observations, which makes the method impractical for large systems. Chen and Malacaria [327] show how to incorporate prior knowledge about the probabilities of hidden actions into the calculation of channel capacity.

If a communication system operates probabilistically, different runs of the system may provide different degrees of anonymity for the same action. In such cases we may be interested in the probability distribution of the degree of anonymity provided by the system across different runs (as distinct from the probability distribution the observer assigns to the actors in a given run). In particular, we may wish to know the worst-case anonymity. Borisov [328] uses Monte Carlo simulations to estimate the *conditional entropy* of anonymous actions given the adversary's observations, with a focus on the worst-case anonymity. This technique can be applied to systems that are too complex to analyse mathematically.

If the observer has some prior knowledge about the probability of each actor performing a given action, *relative entropy* can be used to quantify the amount of information the observer gains by observing the system [329].

Edman *et al.* [330] propose a system-wide anonymity metric that measures the degree of anonymity collectively provided to all users of a communication system, where the observer's goal is to measure the volume of communication between each pair of users. The metric is based on counting the possible matchings between messages entering and leaving the system. Grégoire and Hamel [331] show how a similar metric can be calculated more efficiently by treating all matchings that assign the same number of messages to each pair of users as equivalent.

2.4.2 Traffic Analysis

Even when all communication in a network is encrypted, an observer may be able to learn a considerable amount from the timing, duration and volume of communication; this is known as *traffic analysis*. Danezis and Clayton [332] review the history of traffic analysis and describe some current techniques.

Traffic analysis can be made more difficult by generating *cover traffic* to conceal the volume of genuine traffic on each network link; by *rerouting* traffic through intermediate nodes; and by *reordering or delaying* traffic. Raymond [31] refers to the property of completely concealing traffic patterns as *network unobservability*: "not only does the system conceal who is communicating with whom, but it also hides which users sent or received a message during a period of observation."

Newman *et al.* [333] model cover traffic and rerouting as transformations on a *traffic matrix* representing all of the traffic in an anonymity system during a period of observation. Given an *observed traffic matrix* representing the volume of traffic sent across each link, an observer may attempt to determine the *actual traffic matrix* representing the end-to-end communication between each pair of nodes. Absent cover traffic and rerouting, the matrices would be identical.

Even if the volume of traffic on every link is kept constant through cover traffic, an observer can learn something about the actual traffic matrix from the observed traffic matrix: the actual traffic entering or leaving any node or group of nodes cannot exceed the observed traffic. Whether or not the system uses cover traffic, the observer can enumerate the actual traffic matrices that are possible given the observed traffic matrix, and may assign them different probabilities using prior knowledge of likely communication patterns or concealment strategies. The entropy of the resulting probability distribution can be used as a system-wide measure of anonymity.

Dependent link padding [334, 335] conceals traffic patterns without using constant rate cover traffic by disguising the relationships between the flows entering and leaving a node. When a node needs to forward a received packet, it first checks whether it has scheduled a cover packet for the same flow. If so, the received packet is sent instead. Otherwise, the received packet is scheduled for later transmission, and a cover packet is scheduled for every other flow the node is forwarding. Once a cover packet has been transmitted, it remains with the flow until it reaches its destination, and may cause further cover traffic to be generated; Díaz *et al.* [336] show that this can lead to feedback that results in links being filled to capacity indefinitely. Dependent link padding only conceals packet forwarding: it does not conceal the fact that a source node is generating traffic, so it does not provide unobservability.

Black Box Attacks

Long-term observation of an anonymity system may reveal communication patterns that are not apparent in the short term. The *intersection attack* [31, 318] exploits the fact that users of an anonymising network are active at different times. To identify the user responsible for a series of actions, an observer records which users are active each time one of the actions is observed and intersects the sets. In the related *disclosure attack* [337] and *hitting set attack* [338], an observer attempts to link pairs of communicating users by intersecting the sets of users who are active at the same time as a target user. Statistical variants of the disclosure attack can identify the target's correspondents with high probability using fewer observations than the original attack [339, 340]. An observer can make better use of the available information by considering all pairs of users at once [341], or by using Bayesian inference to incorporate any prior expectations the observer may have about the users' communication patterns [342].

Danezis and Díaz [314] refer to this family of attacks as *black box attacks* because they are based on observing the inputs and outputs of an anonymity system. Feamster and Dingledine [343] show that such attacks are not restricted to global adversaries: traffic entering popular anonymity systems is often carried by the same Internet service provider as traffic exiting those systems, creating a single point at which black box attacks could be carried out.

Köpsell and Hillig [305] point out that an attacker might use a firewall to mount active attacks on anonymity, for example by cutting off a particular user's traffic and looking for corresponding changes in anonymous traffic.

The Predecessor Attack

The *predecessor attack* [344] applies to networks where multiple communication paths can be linked to the same anonymous initiator. The initiator must appear on all of the paths, whereas any other node will tend to appear less frequently, so if every corrupt node on any of the paths records the identity of its immediate predecessor, the initiator is likely to be recorded more often than any other node. The probability of false positives is lower in larger networks, because nodes other than the initiator are less likely to appear more than once. Initiators can resist the predecessor attack by always choosing the same first node for their communication paths; however, this will not always be possible if nodes are unreliable [345].

Wright *et al.* [345] show that the number of times a given user visits a given website follows a power law: most relationships between users and sites are short-lived, but long relationships, which would be susceptible to predecessor and intersection attacks, are not uncommon.

Low-Latency Traffic

In many cases it is possible to identify the websites a user is visiting through an encrypted proxy, simply by comparing the number of bytes transferred in each request with the sizes of the pages and images on each website [346]. Liberatore and Levine [347] show that the same applies if packets are counted rather than bytes, which calls into question the usefulness of padding packets to conceal their sizes.

Partridge *et al.* [348] show that an external observer can determine how data is routed through an encrypted wireless network without decrypting any packets. Adding random delays does not effectively prevent traffic analysis: the delays appear as noise and can be filtered out using standard signal processing techniques.

Zhang and Paxson [349] use traffic analysis to detect *stepping stones*, where a user connects to a host, and from there to a second host. Their algorithm looks for pairs of connections with correlated idle periods. Shmatikov and Wang [350] apply similar techniques to low-latency mix networks (see section 2.4.4). Levine *et al.* [351] suggest adding cover traffic that can be dropped by intermediate nodes to reduce the correlation between the traffic patterns on different links.

Wang and Reeves [352] show that low-latency traffic can be *watermarked* by manipulating inter-packet delays. Forward error correction can be used to cope with queueing delays and other sources of noise, enabling watermarks to survive across multiple hops. Kiyavash *et al.* [353] present a method for detecting and removing watermarks by comparing multiple flows, which can be defeated by an improved watermarking scheme [354]. Spread spectrum techniques can be used to create watermarks with extremely small delays that are hard to detect [355].

Karagiannis *et al.* [356] show that traffic can be classified more effectively by examining the behaviour of nodes, or groups of nodes, than by examining individual flows. Even unknown protocols can often be broadly classified based on their connection patterns. Gong [357] presents some simple rules for identifying peer-to-peer traffic: a peer-to-peer node typically produces a burst of equal-sized packets to different destinations as it connects to the network, followed by traffic between a single local address and port and a large number of remote addresses and ports.

2.4.3 High-Latency Mix Networks

Much of the research into anonymous communication has focussed on sender anonymity for email. An *anonymous remailer* is a server that delivers email while concealing the sender's identity. The simplest kind of remailer just removes the sender's address and other identifying headers from the message before delivering it. To prevent an observer from linking the incoming and outgoing messages, the incoming message (including the recipient's address) may be encrypted with the remailer's public key. However, the remailer's operator still knows the identities of the sender and the recipient, and may be pressured to reveal them [358].

To avoid the risk of relying on a single party, a message can be relayed through a chain of remailers using *onion encryption*: the message and the recipient's address are encrypted with the public key of the last remailer in the chain, the resulting ciphertext and the address of the last remailer are encrypted with the public key of the second-to-last remailer, and so on. Each remailer removes a layer of encryption and delivers the revealed message to the revealed address. Only the first remailer knows the sender's identity, and only the last remailer knows the identity of the recipient.

Chaum [359] described the first design of this kind, the *mix*, which pads messages back to their original size after removing each layer of encryption and waits for several messages to arrive before processing and outputting them as a batch, concealing the relationships between incoming and outgoing messages. Serjantov *et al.* [360] describe several batching strategies for mixes and analyse the best and worse case anonymity provided by each. Díaz and Serjantov [361] propose an improved strategy in which each message is flushed independently with a probability that depends on the number of messages in the mix.

Danezis and Díaz [256] survey a number of mix-like systems and attacks against them, many of which involve manipulating the inputs to a mix in order to identify a particular message among its outputs.

To prevent such attacks, Stop-and-Go mixes [362] delay each message for a period chosen by the sender from an exponential distribution, so the sender can predict when his or her message will leave each mix, but to an observer the departure times appear random. The expected time of arrival at each mix is encoded in the message, allowing mixes to detect attacks that work by delaying or replaying messages, but not those that work by dropping messages altogether.

An RGB-mix [363] sends encrypted 'heartbeat' messages to itself through the mix network to detect when its input messages are being delayed or dropped. If a mix detects an attack, it generates additional dummy messages to disguise its genuine output messages.

Mix techniques can also be applied to other protocols; the rewebber network [364] uses onion-encrypted URLs to retrieve onion-encrypted files through a chain of HTTP proxies, while Nonesuch [365] uses steganography to hide messages intended for a mix network in Usenet newsgroups. The steganography scheme used by Nonesuch gives each

mix a limited ability to distinguish hidden messages from background traffic, protecting against black box attacks even if the attacker controls some of the mixes.

Reply Blocks

Chaum's original mix design includes *reply blocks*, which allow the recipient of an anonymous message to reply through the mix network by attaching a message to an onion-encrypted header created by the sender. The sender's identity and the route through the mix network are concealed from the recipient.

Reusable reply blocks can be used to carry out a *replay attack* in which many messages are sent with the same reply block to trace the route back to the sender through traffic analysis. Mixminion [366] prevents replay attacks by providing single-use reply blocks: a mix will not process two blocks with the same header even if they have different bodies. Mixes can only store the hashes of previously seen blocks for a limited time, so to prevent long-term replay attacks a mix must change its public key before discarding the hashes of previously seen blocks. This also provides *forward secrecy*: if a mix is compromised, its current key cannot be used to decrypt old messages. Mixminion uses encrypted, authenticated connections between mixes to prevent attackers from recognising their own outgoing messages.

A nymserver [366, 367] is an email server that provides mailboxes to anonymous users, who register their accounts and collect their email using a mix network. Each mailbox is associated with a reusable reply block, or with a public key that can be used for authenticating single-use reply blocks.

Cascades and Restricted Routes

As an alternative to the *free route* mix networks described above, in which a message may pass through any sequence of mixes, Berthold *et al.* [318] propose *mix cascades*, in which every message passes through the same sequence of mixes. Unlike a free route network, a cascade remains secure when an attacker controls all but one of the mixes, and because of the restricted topology it is feasible to provide cover traffic between mixes. However, users cannot choose to avoid mixes they do not trust, and blocking or monitoring single entry and exit points may be easier than blocking or monitoring every mix.

Danezis [368] investigates the possibility of restricting the available routes within a mix network, which would make it feasible to provide cover traffic between mixes. Sparse expander graphs are shown to have good anonymising properties, in the sense that a short random walk will reach any node with nearly equal probability. Nagaraja [165] shows that the social network of the LiveJournal website approximates an expander graph and can be used to construct a mix network, although more cover traffic is required than would be the case for an ideal expander graph, due to the skewed degree distribution.

2.4.4 Low-Latency Mix Networks

The mixes described in the previous section introduce substantial delays that make them unsuitable for applications that require low latency, such as web browsing and instant messaging. A parallel strand of research has focussed on anonymising low-latency traffic, starting with Pfitzmann *et al.* [369], who describe a mix network for ISDN communication.

Low-latency mix systems usually establish *circuits* that carry messages in both directions, so the terms *initiator* and *responder* are used in this context instead of sender and recipient. Because low-latency mixes do not delay or reorder messages, they generally provide weaker anonymity guarantees than high-latency mixes. In particular, most low-latency mix systems do not aim to provide anonymity against a global observer.

Onion Routing [370] is a mix network for TCP communication between anonymous clients and non-anonymous servers. A client creates a circuit through a series of mixes using an onion-encrypted message that establishes a symmetric key at each mix. The last mix in the circuit makes an ordinary TCP connection to the server, and the symmetric keys are used to onion-encrypt the data passing in both directions through the circuit. As with Chaum's mix network design, only the first mix knows the identity of the client and only the last mix knows the identity of the server.

Freedom [371] is a commercial anonymity system with a design similar to that of Onion Routing.

Tor [372], a second-generation Onion Routing system, constructs its circuits telescopically: the initiator contacts the first mix in the circuit directly, then contacts the second mix through the first mix, and so on. This enables the use of ephemeral keys that can be discarded when the circuit is torn down, providing forward secrecy in case a mix is later compromised.

To prevent an observer from distinguishing between initiators based on their choice of mixes, which would partition the anonymity set [373, 374], all initiators must update their lists of mixes regularly. Tor publishes the list on a small number of trusted *directory servers*, which are a central point of failure for the system. They could also be attractive targets for observation: if all initiators were to contact the directory servers directly, an observer monitoring the servers would have an up-to-date list of active initiators that could be used for intersection attacks. Tor uses mixes as directory caches to make such monitoring more difficult. Mittal *et al.* [375] suggest using private information retrieval to obtain randomly selected mixes from the directory servers without downloading the complete list.

Mixes that report high uptime and bandwidth to the directory servers are selected more often to participate in circuits, allowing attackers to increase the probability of their mixes being chosen. Bauer *et al.* [376] propose a reputation mechanism for verifying uptime and bandwidth claims, but this would not protect against attackers who genuinely controlled high-performance mixes.

Mutual Anonymity

As well as initiator-anonymous Internet access, Tor supports mutually anonymous *hidden services*. Contact details for hidden services are currently published by the directory servers, but there are plans to decentralise this function [377]. The protocol for contacting a hidden service uses four circuits: one between each party and an *introduction point* chosen by the responder, and one between each party and a *rendezvous point* chosen by the initiator. This helps to prevent denial-of-service attacks, because the responder can selectively ignore connection requests.

ScatterChat [378] and TorChat [379] use hidden services to implement decentralised, mutually anonymous instant messaging.

Øverlier and Syverson [380] present a combined predecessor and traffic analysis attack against hidden services, and show that the attack can be prevented if a service always chooses the same first mix for its circuits. Recent versions of Tor select reliable mixes to act as *entry guards*, making such attacks more difficult; each client periodically chooses three entry guards to use as the first mixes in all its circuits [381].

Cascades and Restricted Routes

Back *et al.* [382] describe two attacks against low-latency mix networks that depend on initiators being able to choose which mixes their traffic will pass through. The first attack involves comparing the latency of a user's circuits to the latency of circuits built by the attacker, while the second involves manipulating the load on mixes and looking for corresponding changes in the throughput of connections entering or leaving the mix network. Murdoch and Danezis [383] demonstrate a similar attack against Tor: a corrupt server sends an identifiable traffic pattern to an anonymous client and uses a one-hop *probe circuit* through each mix to look for corresponding changes in load. McLachlan and Hopper [384] apply this attack to peer-to-peer mix networks (see section 2.4.5) and show that stochastic fair queueing can make the attack more expensive by requiring more probe circuits.

As with high-latency mixes, some designers have advocated cascade topologies to prevent attacks of this kind. JAP [385, 386] is a client for anonymous web access through AN.ON, a low-latency mix cascade. A *cache-proxy* at the exit of the cascade parses the requested web pages, removes content that might compromise the client's anonymity, requests embedded objects such as images, and returns the entire page in a single response. This prevents fingerprinting of the requested page (see section 2.4.2), but requires the cache-proxy to have access to every client's unencrypted traffic. Tor uses a client-side proxy called Privoxy [387] to remove content that might endanger the user's anonymity.

All JAP initiators build and tear down their circuits at synchronised times to prevent an observer from using circuit lifetimes to link initiators and responders. To prevent denial-of-service attacks, a certificate authority issues signed tickets authorising initiators to send traffic through the mix cascade, although it is not clear how Sybil attacks could be prevented without requiring users to prove their real identities to the anonymity system. The JAP design calls for end-to-end cover traffic between the client and the cache-proxy, but this has not been implemented [315].

Díaz *et al.* [336] analyse the costs and benefits of providing cover traffic in four low-latency mix topologies: a free route network; a cascade; a stratified topology where each mix serves as either an entry, middle or exit node; and a restricted stratified topology where only certain combinations of entry, middle and exit nodes may be chosen. The stratified and restricted stratified topologies are shown to provide better tradeoffs between anonymity and bandwidth overhead than free routes or cascades. Modifications to the Tor protocol to support stratified topologies and cover traffic are presented.

Blocking Resistance

When mix networks are used to circumvent censorship firewalls, they encounter the same *proxy discovery problem* as the circumvention systems discussed in section 2.3.4: censors can block access to any mixes they can discover, and to any means of discovering mixes. Köpsell and Hillig [305] describe a blocking resistance strategy for JAP in which clients can volunteer as untrusted relays between other clients and the mix cascade. Captchas are used to prevent censors from discovering relays automatically. Tor has recently added support for *bridge relays*, which are mixes that are not announced through the directory servers but are revealed to users through a variety of other channels, such as email and word of mouth [304].

Vasserman *et al.* [195] show that bridge relay addresses can be collected by running a mix: any connection from an address not listed in the directory either comes from a client or from a bridge relay, and bridge relays can often be distinguished from clients by connecting back to them on commonly used ports. McLachlan and Hopper [388] show that the operators of bridge relays are exposed to additional attacks against their anonymity.

End-to-End Issues

If the first and last mixes in a circuit can recognise that they are participating in the same circuit, they can the link the initiator to the responder. Bauer *et al.* [376] show that Tor's telescopic circuit construction produces a distinctive traffic pattern that can be used by corrupt mixes to determine whether they belong to the same circuit. Corrupt mixes might also try to signal to one another that they belong to the same circuit by watermarking traffic (see section 2.4.2) or by modifying packets [389]. In general it is very difficult to eliminate all covert channels from a system [325], so it seems unlikely that low-latency mix networks can prevent corrupt mixes from realising that they are participating in the same circuit.

Hopper *et al.* [390] show that the end-to-end latency of anonymised connections can be used to reduce the initiator's anonymity, while Manils *et al.* [391] describe an attack that identifies Tor users through information leaks in application layer protocols.

An incident reported by Zetter [392] highlights an important weakness of systems that provide initiator-anonymous Internet access: the connection between the last mix and the responder is not encrypted by the anonymity system, so unless the initiator and responder use end-to-end encryption and authentication, the last mix can monitor and tamper with their communication.

Another issue with anonymising access to existing Internet services is that clients may anonymously attack Internet hosts; mix operators might even be held responsible for the actions of anonymous clients. Tor allows mix operators to set *exit policies* that specify which services may be accessed through their mixes. Many operators choose not to allow exit traffic at all [393].

Nymble [394] addresses this problem by issuing each user with a pseudonym that can be used to obtain single-use tickets for accessing servers anonymously. If a server reports an abusive user to the ticket issuer, the user is warned that she has been reported, and her subsequent connections to that server become linkable.

2.4.5 Peer-to-Peer Mix Networks

In the mix networks described so far, anonymity is provided to a large number of clients by a relatively small number of servers. This approach has certain advantages: the servers can be maintained by experts, it may be possible to exclude faulty or untrustworthy servers, and the servers are easy for clients to find. On the other hand, the servers are obvious targets for attack, access to the servers can be blocked, the servers must contribute enough resources to

support all of the clients, and clients only participate in the system when they are communicating, which provides a starting point for intersection, disclosure and timing attacks.

Tarzan [395, 396] is a peer-to-peer low-latency mix network that aims to avoid these weaknesses by decentralising the mix discovery process and requiring all users to run mixes, making it hard for an observer to distinguish initiators and responders from relays.

Each Tarzan mix communicates with a small number of other mixes called its *mimics*, which are selected verifiably at random. The restricted topology makes it feasible to use cover traffic between mimics. Circuits are constructed by randomly choosing the next mix from among the current mix's mimics. The last mix in the circuit acts as a proxy that can establish outgoing connections to Internet servers and listen for incoming connections from Internet clients, allowing anonymous users to act as initiators or responders.

To avoid building circuits that are entirely controlled by an attacker, IP addresses are hashed in such a way that a mix and its mimics tend to have distinct network prefixes. The assumption is that an attacker may control any number of machines but is likely to have access to a limited number of unique network prefixes; however, as discussed in section 2.1.2, this may not be true for powerful attackers.

Tarzan uses a gossip protocol for mix discovery. To prevent an observer from identifying initiators through their choice of mixes, every initiator must know about every mix with high probability, and must validate every mix's address and public key before propagating them. This could make it prohibitively expensive to maintain the network in the face of churn. It might also be unacceptable to users who do not want their use of the system to be widely known. Due to the peer-to-peer nature of the network, the set of active mixes includes the set of active users, which could be used to carry out intersection attacks.

The advantage of gossip is that a mix cannot partition the anonymity set by giving different public keys to different mixes. However, a mix can behave differently when different mixes attempt to validate its address and public key. For example, a corrupt mix might respond to validation requests from some mixes but not others, creating an inconsistent view of the network that could partition the anonymity set.

MorphMix [397, 398] is a similar system that does not require every mix to know about every other. Instead, MorphMix attempts to prevent a single entity from controlling the entire circuit by employing randomly chosen *witnesses* during circuit construction, and by compiling statistics on how often mixes suggest one another for inclusion in circuits. Tabriz and Borisov [399] show that colluding mixes can avoid detection by modelling the internal state of the collusion detection mechanism.

In Cebolla [400], initiators retrieve lists of mixes from one or more sources and check them for consistency, since inconsistent lists could be used to partition the anonymity set. Encryption is optional, which provides a small performance advantage to initiators with low anonymity requirements at the cost of reducing the size of the anonymity set for encrypted traffic.

I2P [401] provides mutual anonymity in a similar way to Tor's hidden services: each user constructs separate inbound and outbound circuits and publishes the entry point of the inbound circuit in a distributed hash table under a pseudonym. The distributed hash table also stores the addresses and keys of mixes, removing the need for directory servers. When constructing new circuits, users contact the distributed hash table through their existing circuits to prevent their lookups from being observed. Circuits are rebuilt frequently, which might enable predecessor attacks if the timing or other characteristics of encrypted traffic can be used to identify circuits that are likely to have the same owner.

Coping with Churn

To prevent circuits from breaking when one of the participating mixes leaves the network, Zhu and Hu [402] suggest anonymously storing the symmetric circuit keys in a distributed hash table. Any node responsible for storing a replica of the first key can act as the first mix in the circuit, any node responsible for the second key can act as the second mix, and so on. When a node leaves the network, any of the other replica-holders can take its place. Churn and replication will reveal each key to more nodes over time, so initiators must replace their circuits periodically.

Cashmere [403] handles churn by replacing individual mixes with groups of nodes that share an identifier prefix in a structured overlay. Each group is issued a public/private key pair by a central certificate authority. The first node in each group to receive a given message removes a layer of encryption and broadcasts the message to the rest of

the group; the destination may belong to any group along the path. End-to-end acknowledgements are used to route around failed or corrupt nodes.

Selecting Random Mixes

Salsa [404] addresses the problem of selecting random mixes to participate in circuits without knowing the address of every mix in the network. This can be achieved by looking up a random key in a structured overlay, but lookups may be misdirected by corrupt nodes. The Salsa overlay is structured so that nodes that are near each other in the key space are likely to have disjoint paths to any given key, so a node can use nearby nodes to perform redundant lookups. Every node's identifier is the hash of its IP address, so the initiator can check that the distance between the node returned by a lookup and the requested key matches the density of nodes in its local region of the key space.

To prevent an observer from linking initiators to their chosen mixes, Salsa constructs parallel circuits where the nodes in each layer perform redundant lookups for the nodes in the next layer. One node from each layer is then chosen to participate in the circuit. Mittal and Borisov [405] show that the redundant lookups used by Salsa increase its robustness to active attacks but leak identifying information to passive attackers, creating a tradeoff between robustness to passive and active attacks.

Torsk [393] selects mixes at random using Myrmic, a secure distributed hash table (see section 2.3.2). Each node uses random walks on the structured overlay to choose a series of *buddies*, which each perform one lookup on its behalf, preventing the node from being linked to the mixes it selects. Wang *et al.* [406] describe a *buddy exhaustion attack* that uses floods of lookup requests to prevent initiators from building circuits that are not controlled by the attacker.

NISAN [407] also uses a structured overlay to select mixes at random. Rather than relying on a certificate authority as Myrmic does, NISAN uses the verifiable neighbour relationships of Chord (see section 2.2.3) to prevent lookups from being captured by corrupt nodes. The key being looked up is not revealed to the nodes visited during the lookup; instead, the initiator asks each node for its list of neighbours, verifies that they are close to the expected locations in the key space, and selects those nearest the key for the next round. Wang *et al.* [406] show that this nevertheless leaks information about the key.

ShadowWalker [408] is a structured overlay that uses *shadow nodes* to certify that each node's neighbour list contains the correct entries. A node's shadows are chosen verifiably at random, making it unlikely that an attacker controls a node and all its shadows; a mechanism for preventing Sybil attacks is assumed. As with Chord, each node must periodically run a stabilisation protocol to find its correct neighbours; the addition of shadow nodes makes this more expensive.

Tran *et al.* [409] describe selective denial-of-service attacks against mix networks based on structured overlays. By blocking the construction of circuits for which the attacker cannot link the initiator and responder, the attacker is able to link the initiators and responders of a disproportionate fraction of the remaining circuits. Attacks against Salsa and Cashmere are demonstrated.

Friend-to-Friend Overlays

Mittal *et al.* [375] briefly describe two designs that use random walks to select mixes. The first design is based on a degree-restricted friend-to-friend overlay; to prevent corrupt nodes from capturing random walks, a central authority enforces symmetric neighbour relationships. The second design is based on a structured overlay, with neighbour locations verified in a similar way to NISAN.

Drac [410] aims to provide *unobservability* for low-bandwidth applications such as instant messaging, meaning that a global observer should not be able to tell whether a given user is communicating. Each user is assumed to have a number of friends and a number of secret *contacts*; friends are known to the observer, while contacts must be concealed.

Drac uses friend-to-friend connections carrying constant *heartbeat traffic* as signalling channels to set up onion-encrypted circuits through which users can communicate with their friends and contacts. All circuits are constructed and torn down in synchronised *epochs*; in each epoch, a user constructs a circuit for the following epoch by taking a random walk on the heartbeat channels and exchanging ephemeral session keys with the mixes along the walk. The last mix becomes the *entry point* of the circuit.

To communicate with a contact, a user connects anonymously to a *presence server* and sends the contact an encrypted message containing the entry point of her circuit for the next epoch. If the contact wishes to communicate, she prepares a *bridge connection* between the entry points of the two users' circuits. When the next epoch begins, the circuits and the bridge connection become active and the users are able to communicate.

Circuits use constant rate cover traffic, so an observer can tell how many circuits each friend-to-friend connection is carrying, but cannot trace individual circuits. Since bridge connections are created between arbitrary mixes, it is assumed that an observer can distinguish them from other connections. This allows the observer to assign probabilities to the possible initiators of the bridged circuits by enumerating the random walks that could have terminated at their respective entry points. Over time, as pairs of contacts bridge different pairs of entry points, they may be revealed as likely correspondents. The authors argue, however, that the probabilities assigned to users who are near one another in the social network will tend to be correlated, so even when multiple entry points are considered it may be hard to distinguish a communicating user from those near her in the social network.

This raises an important issue for anonymity systems based on social networks: whether it is more desirable to be anonymous among a group of friends or among a group of strangers. It could be argued that a group of friends is more resistant to coercion, while a group of strangers is more resistant to inference: friends are less likely to betray one another's anonymity, but an observer might learn more from the fact that a group of students contacted a group of journalists, for example, than from mixed groups of the same size.

2.4.6 Other Peer-to-Peer Anonymity Systems

This section discusses peer-to-peer anonymity systems that do not use onion encryption.

Random Walks

Crowds [319] uses random walks to provide initiator anonymity for web browsing. Initiators form a peer-to-peer *crowd* coordinated by a central membership server, and each initiator constructs a path through the crowd by sending a request to a randomly chosen node, which decides randomly whether to forward the request to another member of the crowd or to act as a web proxy, communicating with web servers on the initiator's behalf.

All communication is sent along the same path until the initiator receives a synchronisation message from the membership server, which tells all nodes to construct new paths. Synchronisation has two purposes. First, it prevents new nodes from being linked to new paths, because a new node does not construct its first path until it receives its first synchronisation message, at which point all other nodes are also constructing paths. Second, it prevents initiators from constructing paths too frequently, which would expose them to predecessor attacks.

Connections between nodes are encrypted using keys obtained from the membership server, but there is no end-to-end encryption, and no reordering or padding to prevent traffic analysis. As with Tarzan, every user has a complete list of other users, which may enable intersection attacks.

Hordes [411] is a similar system that uses IP multicast to return information to the initiator. The web server must run a Hordes-aware proxy to handle the asymmetric communication. Unlike Crowds, each packet is routed independently, so Hordes is vulnerable to predecessor attacks. Mantis [412] is a Crowds-like system that uses IP spoofing [60, 61] to return information from anonymous responders to non-anonymous initiators.

Muñoz-Gea *et al.* [413] provide an alternative path selection algorithm for Crowds that reduces the variation in path length, but Danezis *et al.* [414] show that this reduces anonymity and exposes the initiator to predecessor attacks.

Unstructured Overlays

MUTE [415] is an anonymous packet switching network based on an unstructured overlay; users may optionally connect only to their friends. Each node is identified by a random overlay address, and packets are routed using a probabilistic reverse path forwarding protocol inspired by the collective foraging behaviour of ants. Packets addressed to unknown destinations are flooded to discover short routes.

Chothia [416] describes a spoofing attack that can be used to determine whether a given MUTE overlay address belongs to a given neighbour. Version 0.5 of MUTE prevents this attack by deriving each node's overlay address

from the hash of its public key. All packets are timestamped and numbered to prevent replay attacks, the public key is included in the packet header, and the packet is signed with the private key. This causes significant per-packet processing and bandwidth overhead.

MUTE's reverse path forwarding is vulnerable to selective filtering, because it implicitly assumes that a good route *from* a node is also a good route *to* that node. If an attacker drops packets addressed to a target node but continues to forward packets sent by the target, neighbouring nodes will not attempt to route around the attacker.

ANts P2P [417] adds end-to-end encryption to MUTE, but intermediate nodes can carry out a man-in-the-middle attack unless one of the endpoints obtains the other's public key out-of-band. ANts P2P uses a two-tier structure to improve scalability (see section 2.2.2).

MuON [418] uses a gossip protocol to disseminate messages in an unstructured overlay. The header of every message is gossiped to the entire network. With a certain probability, each node apart from the intended recipient retrieves the message body from the sender and marks itself as the sender before gossiping the message header. The intended recipient always retrieves the message body, so the sender and recipient of a series of messages can identify each other using predecessor attacks. No other node can link the sender and recipient, however, because messages and headers are encrypted end-to-end.

Sneakernet [419] uses small data-carrying devices such as mobile phones and memory sticks to pass encrypted messages between friends. A gossip protocol allows messages to travel over multiple hops between a trusted Internet gateway and anonymous users.

OneSwarm [420] supports private file sharing between friends and anonymous file sharing through an unstructured overlay. Users with high security requirements may choose to connect only to their friends, while those with lower security requirements can discover peers from public or private servers, and may even join non-anonymous BitTorrent swarms. To make it difficult for attackers to collect large numbers of addresses from public servers, each node is consistently matched with a small number of others using the hash of its IP address. OneSwarm's search protocol uses pseudo-random delays to conceal the identities of nodes responding to searches; to prevent statistical attacks, a responder always uses the same delay for a given file. Files can be downloaded from multiple sources in parallel, reducing the impact of slow overlay paths.

SSMP [421] provides mutual anonymity for initiators and responders in an unstructured search overlay by splitting each query into n shares, any $k < n$ of which can be used to reconstruct the query. Each share is flooded probabilistically, and any node that receives k or more shares reconstructs the query and floods it on behalf of the initiator. Reverse path forwarding can then be used to deliver query results and transfer files. Rumor Riding [422] uses a similar technique based on random walks.

Katti *et al.* [423] show that *network coding* [424] can be used as an alternative to onion encryption in anonymising networks. The sender constructs a layered route with several relays in each layer, with the recipient at a random position along the route. Each message is encoded into multiple *slices* that travel through different combinations of relays. No single relay can decode the messages or identify any other node participating in the route, except the nodes in the previous and next layers. However, a local observer who can monitor all of the recipient's connections can decode the received messages, and a local observer who can monitor all of the sender's connections can also link the sender to the recipient.

Structured Overlays

In many peer-to-peer anonymity systems it is possible for a node to learn the network address of any other node, and some systems require every node to know every other node's address. This information may be useful to an attacker: a list of active nodes could provide information for intersection attacks, targetted surveillance or blocking. In some jurisdictions mere participation in an anonymous communication network might be illegal.

Hazel and Wiley [425] describe Achord, a modified version of Chord in which each node only reveals its participation in the overlay to $O(logN)$ other nodes. An attacker cannot discover the address of the node that is responsible for a given key unless the attacker is one of the node's $O(logN)$ neighbours, which can be checked by hashing its IP address. However, this does not provide much protection against an attacker with access to a large number of IP addresses. If some of the participants use dynamic IP addresses then the attacker can learn additional addresses over time.

The routing protocol used by Chord and Achord does not provide strong anonymity for initiators, because the clockwise distance to the requested key decreases at each hop, allowing each node to estimate the probability that the previous node is the initiator. O'Donnell and Vaikuntanathan [426] show that the average size of the anonymity set grows linearly with the size of the network, but Borisov [328] demonstrates that the worst-case anonymity set may be much smaller than the average. The worst-case anonymity can be improved if each node adds a small amount of randomness to the locations it uses to select its neighbours [427].

AP3 [428] uses random walks on a structured overlay to provide anonymity. To send a message anonymously, a sender chooses a random key from the overlay's key space and routes the message deterministically to the node responsible for that key, which decides randomly whether to deliver the message to the recipient or forward it to another randomly chosen key. As in Crowds, the probability of forwarding creates a tradeoff between efficiency and anonymity.

AP3 does not use onion encryption, so every node forwarding a message can see the identity of the recipient. Each message follows an independent random path, so a predecessor attack can be used to identify a sender who sends a series of messages to the same recipient.

To receive messages anonymously, a recipient creates a *channel* by sending a message to a randomly chosen key, which becomes the channel's address; messages sent to the channel are forwarded along the reverse path to the anonymous recipient. Channels must be rebuilt periodically to cope with churn, exposing their owners to predecessor attacks.

AP3 uses the secure lookup protocol of Castro *et al.*, which requires certified identities (see section 2.3.2). Mittal and Borisov [405] show that the redundancy of the secure lookup protocol leaks information about the source of the lookup.

Borisov [328] proposes an anonymising overlay in which each message follows a random walk before being routed deterministically to its destination. The overlay is based on Koorde [429], which is shown to provide optimal anonymity using shorter random walks than any other structured overlay.

Safebook [430] is a social networking service based on a friend-to-friend overlay and a distributed hash table. A central authority is required for identity management, but the authority does not have access to users' data, which is stored in encrypted form on their own nodes and mirrored by their friends. To communicate with untrusted parties, users build paths through the friend-to-friend overlay and publish the entry points of their paths in the distributed hash table; this is similar to the approach used by Drac, but the paths are not onion encrypted.

Distributed Caches

Freenet [431] is an anonymous publishing system based on a peer-to-peer distributed cache. Different versions of the software use different routing and caching algorithms.

Freenet 0.5 [432, 433] is a loosely structured overlay where each file is identified by a unique key. Each node keeps a limited number of *routing hints* associating keys with node addresses, and files are published and retrieved using a depth-first search algorithm. When a node receives a request to publish or retrieve a file, it sorts its hints by their lexicographic distance from the file's key and forwards the request to the address associated with the closest hint. If the request travels in a loop or reaches a dead end, the next-closest hint is tried. A probabilistic hop counter is used to limit the range of searches.

Nodes cache the files that are published and retrieved through them, so popular files will tend to be widely replicated. However, no single node is responsible for storing any file, so there is no way to ensure that a published file will be retrievable in the future. In theory any node can store a permanent copy of any file, but requests for that file will not necessarily reach that node.

When a file is successfully retrieved, each node that forwards the file back towards the requester stores a routing hint associating the file's key with the address where the file was found. Each node therefore tends to receive requests for keys that are lexicographically close to those it has supplied in the past, causing its routing and data caches to specialise in certain areas of the key space.

However, if every node knew the location of every file then searches would only travel one hop, destroying anonymity. Similarly, if every node were to specialise in a few areas of the key space then any node sending a search for a key outside its specialisation would be identifiable as the initiator, since no other node would send it such a search. To ensure that paths do not become too short or nodes too specialised, occasionally a node that is returning a requested file will replace the address where the file was found with its own address, hiding the true source of the file.

Zhang *et al.* [434] propose that each Freenet node should actively specialise in a particular region of the key space. This would improve the efficiency of routing and caching, but might also allow an attacker to censor particular keys by becoming responsible for them. The proposed solution to this problem, in which a node's neighbours collectively choose its specialisation, is flawed because multiple nodes may be controlled by a single attacker.

Freenet 0.6 [435] aims to speed up searches by estimating which node will be able to respond to a search for a given key most quickly. This involves incorporating many factors – such as the distance between keys, fluctuations in load, and differences in capacity and reliability among nodes – into a single estimate of overall response time, which is then used to guide the depth-first search.

Freenet provides some degree of anonymity against a local, internal observer, because a node receiving a search cannot tell whether the search originated at the previous node or was forwarded on behalf of another node. However, Borisov [328] shows that Freenet 0.5 and 0.6 provide little anonymity against a global eavesdropper, while Dhamija [436] describes several attacks against routing, anonymity and node discovery.

Freenet 0.7 [437] is a distributed hash table that achieves efficient routing by changing the nodes' locations in the key space rather than by controlling the topology. Each node starts at a random location and uses a stochastic algorithm to swap locations with other nodes until the network converges on a suitable arrangement for efficient routing [130]. The location-swapping algorithm can create a distributed hash table from any navigable small world network (see section 2.1.1).

Evans *et al.* [90] show that in a network with a few long-lived nodes and a constant influx of short-lived nodes, Freenet's location-swapping algorithm causes the locations of the long-lived nodes to collapse into tight clusters. This leaves a few nodes responsible for large areas of the key space, which may cause them to become overloaded. The same effect can be produced by one or more corrupt nodes regularly resetting their locations to fixed values and then swapping as normal. The problem can be mitigated if all nodes regularly reset their locations to random values, but this prevents the swapping algorithm from converging.

Freenet 0.7 was originally intended to be a friend-to-friend network [437, 438], but the design does not allow groups of friends to establish separate networks and merge them later – all new users must join the main network. Potential users might not know any users of the main network, or their social connections might not have the small world structure needed for efficient routing, so recent versions of Freenet optionally create additional connections between strangers to produce a suitable topology [439].

FASD [440] is a distributed, anonymous search engine built on top of Freenet.

GNUnet [441, 442] is an anonymous file sharing system in which files are encrypted and broken into equal-sized blocks, each of which is identified by a separate key. The overlay is unstructured; queries are routed using probabilistic flooding, with blocks being returned and cached along the reverse path. Kügler [443] describes a predecessor attack that makes it possible to determine whether a series of related queries is likely to have originated at a neighbouring node or to have been forwarded on behalf of another node. Similar attacks might be possible against Freenet 0.7, which also stores each block under a separate key. Users of both networks can choose to connect only to their friends, which might help to prevent predecessor attacks.

2.4.7 Anonymous Ad Hoc Routing Protocols

Several protocols have been proposed to provide unlinkability between sources and destinations in mobile *ad hoc* networks. All of the proposed protocols have an on-demand structure similar to that of DSR or AODV (see section 2.3.3.1): a *route request* is flooded to discover a route from the source to the destination, which sends a *route reply* back along the reverse path, and *route identifiers* established at the intermediate nodes during route discovery are used to forward data packets.

Nodes in MASK [444] are issued with certified pseudonyms by an offline certificate authority, which they use instead of their real identities, periodically changing pseudonyms so they cannot be tracked. A secret handshake protocol enables neighbouring nodes to authenticate each other and establish pairwise keys for link encryption without exposing their real identities or revealing their membership in the network to non-members.

Route requests in MASK contain the destination's real identity; to avoid linking its identity to its current pseudonym, the destination rebroadcasts the route request like any other node. Any node can send a route reply on behalf of the destination, replacing any routes created by replies with lower sequence numbers; a corrupt node can therefore prevent

route discovery by waiting for any genuine route replies to reach the source, then sending a false route reply with a higher sequence number.

Route replies with equal sequence numbers establish parallel routes, and a node with multiple routes to the destination selects one at random for each packet. This balances load and could also make traffic analysis more difficult. MASK also uses random delays to hinder traffic analysis. Anonymous route error messages are used to report broken links; these are not authenticated, so an internal or external attacker can use them to delete any route.

Nodes in SDAR [445] monitor the forwarding behaviour of their neighbours and classify them into three *trust levels*. Each node distributes a link encryption key to all its neighbours in each trust level, and sources can request routes that only use nodes at a certain trust level.

Route requests contain the destination's identity encrypted with its public key, so every node must attempt to decrypt every route request. Each route request also contains a single-use public key that is used to establish symmetric keys between each relay and the endpoints. The destination uses the symmetric keys to onion-encrypt the route reply, and the source uses them to onion-encrypt data packets.

AnonDSR [446] uses onion encryption and single-use public keys in a similar way to SDAR. There is no link encryption between neighbouring nodes, so an external observer can trace route requests through the network. Route replies and data packets are onion encrypted, but all packets on a given route carry the same unencrypted route identifier on each link.

AnonDSR also includes a protocol to establish shared secrets between sources and destinations. The protocol reveals the identities of both endpoints, so unlinkability would be undermined unless every pair of nodes executed the protocol, which would not be scalable. However, many other designs simply assume that a shared secret already exists.

In ANODR [447, 448] the first route request between a given source and destination contains the destination's public key, so other nodes need not attempt to decrypt it; however, this allows internal or external observers to identify the destination. Subsequent route requests between the same endpoints are encrypted using a shared secret. Each relay adds a single-use public key to the route request, which is used to establish a symmetric link encryption key during the route reply phase. Route requests are not link encrypted, so they can be traced by an external observer. Like MASK, ANODR uses anonymous route error messages that allow an internal or external attacker to delete routes.

Sources and destinations in ASR [449] are anonymous to relays and *vice versa*. The size of route request and reply messages does not change along the route, so relays cannot tell how may hops separate them from the source or destination, except perhaps by measuring the round-trip time. Parallel route replies are used to discover node-disjoint routes, which the source can select between to avoid faulty nodes (see section 2.3.3.5). As in ANODR, single-use public keys are used to establish symmetric link keys for each route, but route requests are not link encrypted and can be traced. The link keys are also used to authenticate route error messages, preventing external attackers from deleting routes.

ASR and ANODR delay and reorder packets and use cover traffic to make traffic analysis more difficult.

ARM [450] is broadly similar to AnonDSR, but it uses link encryption and a large amount of cover traffic: route replies and data packets are flooded for a limited number of hops by nodes that are not on the route.

In ANODR, ASR and ARM, each route request contains a computational puzzle that can only be solved by the destination, so relays can verify that the route reply was generated by the intended destination. This is somewhat similar to the unforgeable acknowledgement mechanism described in Chapter 5; however, the puzzle does not authenticate any of the other fields in the route reply, so corrupt nodes can tamper with them to prevent route discovery.

All of the protocols described in this section are vulnerable to black hole attacks in which the attacker correctly forwards route requests and route replies but drops data packets. MASK and ASR gain some protection by using parallel routes, but such defences are vulnerable to Sybil attacks.

2.4.8 Anonymous Broadcast Networks

The systems described above provide anonymity by relaying messages so that the sender and recipient do not communicate directly. An alternative way to provide anonymity is to create *broadcast groups* in which every user receives the messages sent by every other user.

The *dining cryptographers protocol* [316] provides provable sender and recipient anonymity within a broadcast group. Each pair of users shares a secret key that is used to seed a pseudo-random number generator. In each round of the

protocol, each user XORs the outputs of her generators. A user who does not wish to send a message transmits the result, while a user who wishes to send a message transmits the XOR of the message and the result. Each user XORs the values she receives and the value she transmitted. If no user sent a message, the result is all zeroes, because each pseudo-random bit was transmitted twice. If one message was sent, it is received by all users but none of them can identify the sender. If more than one message was sent, the result is unreadable.

The protocol provides perfect anonymity for senders and recipients, but scales poorly and is vulnerable to a simple denial-of-service attack: any user can anonymously jam the channel by transmitting in every round.

A variant of the dining cryptographers protocol can be used to communicate over a spanning tree [451].

Herbivore [452] is a peer-to-peer network in which small cliques of nodes implement the dining cryptographers protocol. A structured overlay is used to route messages between cliques; an eavesdropper can observe communication between cliques, but cannot tell which member of each clique is communicating. Anonymous connections can also be made to Internet servers. (CliqueNet [453] is an earlier proposal by the same authors.)

Herbivore uses computational puzzles to prevent nodes from joining arbitrary cliques, but the average clique contains 128 nodes, so an attacker who controls a small fraction of the nodes in the network has a reasonable chance of having a node in any given clique, allowing the attacker to jam the clique's communication anonymously. For example, in a network of 100 cliques (12,800 nodes), an attacker who controls 1% of the nodes can jam an expected $1 - (99/100)^{128} = 72\%$ of the cliques. Nodes move to new cliques if they cannot transmit, but this exposes them to intersection attacks.

Golle and Juels [454] present a variant of the dining cryptographers protocol that makes it difficult for users to jam the anonymous channel without being identified. Each round of the protocol is divided into several transmission slots. Before each round, each user broadcasts a challenge that commits her to revealing the seeds of her pseudo-random number generators for, on average, half the slots. If a user wishes to transmit in a given slot, she may search for a challenge that does not commit her to revealing her seed for that slot, which would reveal that she had transmitted a message. The difficulty of doing so grows exponentially with the number of slots she wishes to use, so an attacker can only jam a limited number of slots in each round without being exposed.

P^5 [455] uses broadcast groups to provide mutual anonymity, but does not use the dining cryptographers protocol. Each node joins several randomly chosen broadcast groups with variable-length identifiers. Messages are addressed to prefixes, and are received probabilistically by every group with an identifier matching the prefix. This allows each user to choose a suitable tradeoff between anonymity and efficiency: by revealing a longer prefix to a sender, a recipient reduces the size of her anonymity set, but also reduces the number of groups that will receive the sender's messages, improving the signal to noise ratio.

P^5 also supports onion encryption, allowing messages to be forwarded through multiple groups to hide the sender's identity. Public keys are distributed by broadcast, which might allow man-in-the-middle attacks.

In Agyaat [456], each node in a structured overlay joins one or more unstructured *clouds*, and messages are addressed to clouds rather than nodes. Routing begins with a random walk in the sender's cloud to conceal the initiator's identity; the message is then forwarded through the structured overlay to the *rendezvous node* of the recipient's cloud, which broadcasts the message through the cloud, concealing the identity of the recipient. To balance load among cloud members, Agyaat uses multiple structured overlays with independent key spaces; a given cloud may have a different rendezvous node in each overlay.

Nodes in Clouds [457] use profiles describing the files they are sharing to find nodes with similar content, which become their neighbours in a semantic overlay (see section 2.2.4). As in Agyaat, queries, responses and download requests are routed between clouds rather than individual nodes, making it hard to censor queries and providing anonymity for requesters and responders. Randomness in the cloud formation protocol prevents attackers from surrounding targetted nodes, but the protocol is vulnerable to Sybil attacks. While queries and responses cannot be linked to particular nodes, profiles are public knowledge: some users might not wish to publish profiles describing the content they are sharing.

UQDT [458] uses overlay multicast trees to route keyword queries to relevant nodes; each node maintains a Bloom filter [112] describing the files held by the nodes in its subtree. Parallel trees are used to balance load, with each node serving at different levels in different trees. The leaves of the trees are broadcast groups whose members calculate a collective index of their files using secure multi-party computation [459] such that no single user can be associated with any file, although a Sybil attack could defeat this protection. Responses are sent back through an anonymous channel, the design of which is not described.

Beimel and Dolev [460] describe a family of anonymous communication protocols based on the metaphor of *buses* touring a network. Each seat on a bus is assigned to a sender and contains either random data or an onion-encrypted message from that sender; a layer of encryption is removed at each node the bus visits, and if the result is a valid message then it is intended for the current node, which replaces it with random data.

The use of a reserved buffer for each sender somewhat resembles Perlman's scheme for Byzantine robust routing (see section 2.3.3), and indeed one of the bus protocols is shown to protect against a limited number of Byzantine faulty nodes. All of the protocols require nodes to have global knowledge of the network, although one of them is suitable for a dynamic topology over a known set of nodes.

2.5 Cooperation

Any system in which the infrastructure is provided collectively by the users faces the problem of encouraging users to contribute resources as well as consuming them.

Nielson *et al.* [461] classify the various 'rational attacks' that might be carried out by self-interested nodes in a communication network. The authors distinguish between *altruistic* or *obedient nodes*, which obey the system's protocols regardless of their own self-interest; *malicious nodes*, which attempt to damage the system; and *rational nodes*, which attempt to benefit from the system, possibly by disobeying its protocols.

Adar and Huberman [462] show that the majority of Gnutella nodes make no files available for download, and the majority of downloads are from a tiny minority of nodes. It is suggested that this *free riding* will eventually lead to the collapse of the network, although the large number of downloads observed could also be seen as evidence of the network's continuing health, albeit at the expense of a small number of altruists. Hughes *et al.* [463] argue that Gnutella developers have little incentive to solve this problem, since implementations that do not force their users to cooperate will become more popular than those that do. Krishnan *et al.* [464] show that sharing files is not necessarily altruistic: it is rational for some users to share files if doing so reduces the load on nodes from which they wish to download.

Huang *et al.* [465] question the usefulness of incentive mechanisms in mobile *ad hoc* networks. They argue that incentives to cooperate are unlikely to be necessary during the early stages of adoption, when most users are likely to be enthusiasts who will not mind incurring costs to use a new technology; incentive mechanisms may even hinder adoption by increasing complexity and adding unnecessary performance overheads. It is suggested that incentives could be implemented later at the application layer if they turn out to be necessary.

It is not clear, however, that application-layer incentives could achieve network-layer goals such as encouraging nodes to forward packets, since nodes sharing the same network layer may be involved in a variety of different applications [466]. Application-layer incentives for network-layer cooperation would seem to imply the construction of a separate network for each application, whereas network externalities suggest that a single shared network would be more valuable [467].

Four main approaches have been taken to the problem of encouraging resource contribution in distributed communication networks:

1. **Micropayments:** nodes pay one another for services using a digital currency.
2. **Reputations:** nodes report one another's behaviour and combine other nodes' reports with their own direct observations.
3. **Audits:** nodes test one another's behaviour and punish or exclude misbehaving nodes.
4. **Reciprocation:** nodes adjust the level of service they provide to one another based on the level of service they receive.

2.5.1 Micropayments

Buttyán and Hubaux [468] propose a virtual currency called *nuglets* to encourage nodes to forward packets in mobile *ad hoc* networks; the currency is implemented with tamper-resistant hardware and certified identities. Chandan and

Hogendorn [469], Crowcroft *et al.* [470], Anderegg and Eidenbenz [471], and Xue *et al.* [472] develop traffic pricing models for mobile *ad hoc* networks. Feldman and Chuang [473] point out that if a packet fails to reach its destination in a multi-hop network, the source of the packet may not know which node is responsible for dropping it. Nevertheless they show that if the source offers a payment to each relay that is conditional on the end-to-end delivery of the packet, it is in the interest of rational nodes to forward packets.

Micropayments have also been used to support cooperation in peer-to-peer networks, starting with Mojo Nation [474], which used micropayments to encourage users to share files. Golle *et al.* [475] describe a micropayment system for peer-to-peer file sharing, while Ioannidis *et al.* [476] propose micropayments for peer-to-peer storage. Li *et al.* [477] use micropayments to encourage message forwarding in peer-to-peer overlays. KARMA [478] is a general-purpose micropayment framework for peer-to-peer networks. Osipkov *et al.* [479] suggest using randomly chosen witnesses and an offline certificate authority to prevent double spending in a peer-to-peer environment.

Some digital currencies support anonymous payments; Figueiredo *et al.* [480], Androulaki *et al.* [481], and Jansen *et al.* [482] describe anonymous micropayment systems to encourage resource contribution in anonymity systems.

To prevent double spending, micropayment systems require some combination of centralised banks, certified identities and tamper-resistant hardware. The Trusted Computing Group [483] aims to make tamper-resistant hardware widely available, but this solution might be considered unsuitable for censorship-resistant communication because it is centrally controlled [484].

2.5.2 Reputations

Resnick *et al.* [485] provide an overview of reputation systems on the Internet and identify three requirements for a successful reputation system:

1. Long-lived entities that inspire an expectation of future interaction.
2. Capture and distribution of feedback about interactions.
3. Use of feedback to guide trust decisions.

As discussed in section 2.1.2, the first requirement is perhaps the hardest to guarantee in an open system; positive reputations may be vulnerable to Sybil attacks, while negative reputations may be vulnerable to whitewashing. Another potential problem is *slander*, in which an attacker makes false negative reports about nodes in order to harm their reputations.

CONFIDANT [486] is a reputation system for mobile *ad hoc* networks in which nodes exchange observations of good and bad behaviour with their neighbours. To prevent slander, each node will only accept negative reports from a manually configured list of trusted friends. CORE [487] uses positive and negative direct observations, but only accepts positive reports from other nodes. Buchegger and Le Boudec [488] propose a system in which nodes combine others' reports with their own observations using a Bayesian rule designed to minimise the effect of slander. A more recent proposal from the same authors uses separate performance and honesty metrics [489]: a node is considered honest if its reports of other nodes' performance are consistent with first-hand observations. If a node does not meet a certain threshold of honesty, its reports are not incorporated.

P-Grid [490] is a peer-to-peer network incorporating a reputation system [491]. Performance reports are stored in a structured overlay and can be retrieved by any interested node. Each report is weighted according to the reporter's trustworthiness, which is based on first-hand experience of its performance, and the weighted reports are aggregated using maximum likelihood estimation.

EigenTrust [492] is a distributed algorithm for calculating a global reputation value for each node in a peer-to-peer file sharing system.

Marti *et al.* [493] describe a reputation system for mobile *ad hoc* networks in which nodes monitor the wireless channel to overhear their neighbours forwarding packets. Nodes that forward less than a certain fraction of packets are reported to the source, and routes are selected so as to avoid uncooperative nodes. It might be argued that this is exactly what uncooperative nodes want, since they will save bandwidth and battery power by being excluded from forwarding. However, cooperative nodes also benefit because they avoid retransmitting lost packets.

None of the reputation systems described above is designed to deal with attackers who collude or use multiple identities. Cheng and Friedman [494] show that any reputation system that attempts to calculate a global value for each node's reputation is vulnerable to Sybil attacks. However, some systems in which each node makes its own evaluation of other nodes' reputations can resist such attacks; Cheng and Friedman present an example based on maximum flow. Feldman *et al.* [495] evaluate a reputation system of this kind and find that it is robust to collusion and whitewashing, although reputations must be discounted over time to prevent an attacker from building up a good reputation and then exploiting it.

Dell'Amico [496] shows how to use a local, partial view of a reputation graph, in which a directed edge between two nodes represents a report of an interaction, to approximate the reputation values that would be calculated from the complete graph.

2.5.3 Audits

Ngan *et al.* [497] describe a peer-to-peer storage system in which every node publishes a signed list of the files it is storing for other nodes and the files other nodes are storing on its behalf. Random audits are used to detect and evict nodes that publish false claims.

Blanc *et al.* [498] propose an incentive mechanism for peer-to-peer networks in which a node that fails to cooperate is punished for a certain period, during which it must always cooperate and tolerate defection by other nodes, otherwise the period restarts. It is shown that if the punishment period is long enough, rational nodes can be induced to cooperate. However, the mechanism is vulnerable to noise and assumes regular traffic patterns: if a node can compress its traffic into bursts, it can defect at the end of each burst and then wait for the punishment period to expire before sending another burst.

Catch [499] is a protocol that encourages forwarding in multi-hop wireless networks by making it risky for nodes to lie about their connectivity to avoid forwarding. Nodes use anonymous challenges to test their neighbours' connectivity; nodes that fail to retransmit challenge messages are permanently evicted from the network. Similarly, nodes in ARA [500] monitor their neighbours to ensure that they are forwarding packets, and misbehaving nodes are evicted.

SHARP [501] is a general framework for peer-to-peer resource trading: digitally signed tickets are used to reserve and claim resources such as storage, bandwidth and computation. If a node fails to provide the resources it has promised, the tickets it has signed can be used as evidence to evict it from the network.

Punishment and eviction depend on the assumption that nodes cannot rejoin the network under new identities. Even if a node only has access to a limited number of identities, it may be able to avoid temporary punishment by using those identities in rotation, leaving each identity idle until its punishment is over.

2.5.4 Reciprocation

Reciprocation between neighbouring nodes does not require central record-keeping or identities with global scope, making it attractive from a privacy perspective. However, as with reputations and audits, there must be an expectation of future interaction, and it may be difficult to sustain cooperation if nodes can easily generate new identities.

Reciprocation in Download Swarms

BitTorrent [107] uses reciprocation to encourage nodes in file sharing swarms to upload as well as downloading. Each node maintains a small *active set* of connections, with all other connections *choked*, meaning that nothing is uploaded. Nodes update their active sets periodically, unchoking those connections that have recently provided the best download speeds, so nodes that upload more quickly are more likely to be unchoked by their neighbours. The active set also includes one randomly chosen connection; when all connections appear to be choked at the other end, additional connections are randomly unchoked to avoid deadlock. The size of the active set must be chosen carefully: if the set is too small, a large fraction of the node's upload bandwidth will be spent on randomly unchoked neighbours, but if the set is too large it may include poorly performing neighbours.

The average download speed in BitTorrent swarms is often higher than the average upload speed, because nodes that have finished downloading may remain in the swarm as *seeds*, uploading to other nodes in round-robin order [502].

This altruistic behaviour seems to indicate that many users are willing to repay the network for resources they have used, even if they may be reluctant to contribute resources in advance.

BitTyrant [503] is a modified BitTorrent implementation designed to obtain the maximum download speed for a given upload speed, which can be seen as a form of *utility-maximising behaviour* (see Chapter 4). BitTyrant exploits the observation that a standard BitTorrent node uploads at the same speed to every unchoked neighbour, so beyond a certain threshold it is not possible to obtain a faster download from a standard node by uploading to it more quickly. BitTyrant calculates this threshold for each of its neighbours, sorts the neighbours according to their thresholds, and uploads to as many neighbours as necessary to saturate its upstream connection.

BitTyrant performs well in swarms of standard BitTorrent nodes and in single swarms of other BitTyrant nodes, but when each node is allowed to participate in more than one swarm, the average performance decreases as high-capacity nodes allocate their resources efficiently across several swarms to the detriment of the low-capacity nodes in each swarm.[1]

Tribler [139] achieves faster BitTorrent downloads through cooperative downloading: *collectors* who are interested in obtaining a file ask their friends to act as *helpers*, downloading pieces of the file on their behalf. It is not clear whether this benefits the swarm as a whole – seeds upload to other nodes in round-robin order, so a collector with helpers will receive more from the seeds than an ordinary node, to the detriment of other downloaders. This could even be viewed as a form of Sybil attack. However, cooperative downloading could be beneficial in the longer term if the helpers remain in the swarm as seeds.

The eDonkey2000 [504] and eMule [505, 506] file sharing networks also allocate bandwidth in a reciprocal fashion.

Payment Balances

GNUnet's excess-based economic model [507] aims to allocate resources in such a way that people with no resources to offer can use the network's spare capacity, without making the network vulnerable to resource-depletion attacks. Nodes 'pay' their neighbours to forward queries, but unlike micropayment systems, no currency is required: payment balances exist between pairs of nodes but are not transferable. Priority is given to paid queries, while spare bandwidth and storage are allocated to unpaid queries. Thus a denial-of-service attack can only use the network's excess resources, unless the attacker maintains a positive payment balance by contributing resources to the network, which would defeat the purpose of the attack.

SWIFT [508] uses payment balances to encourage file sharing nodes to upload. Each node counts the bytes sent and received on each of its connections and uploads in round-robin order to those neighbours with positive credit balances. Simulations show that nodes benefit by distributing a small fraction of their bandwidth equally among all neighbours because this helps to avoid deadlock.

Ostra [509] uses payment balances to limit the amount of data that can be sent across each link of a social nework before receiving an acknowledgement from the destination that the data was wanted. In the centralised variant of the scheme, a trusted server finds routes from sources to destinations across the social network; the decentralised variant uses a distributed routing protocol. Unacknowledged messages tie up credit until they expire, so it might be possible for an attacker to disable links by continually sending messages to nonexistent destinations.

Reciprocal Storage

Cooper and Garcia-Molina [510, 511] describe a reciprocal backup system in which nodes bid for storage space on other nodes by offering space in return. There is no mechanism for dealing with nodes that delete the files they have agreed to store.

Storage relationships in the Samsara backup system [512] do not have to be symmetric – instead, each node that is asked to store a block of data issues an equal-sized *claim* in return. Each claim contains pseudo-random data generated using a secret key, so the owner can regenerate it on demand and does not need to keep a copy. Nodes that are storing blocks periodically test the holders of the corresponding claims, discarding the blocks if the claims have been discarded.

[1] There is an interesting parallel here with the free movement of international investment capital, where swarms are analogous to countries.

Claims consume up to half of Samsara's storage capacity, so to save space a node can forward an existing claim when it is asked to store a block, rather than creating a new claim. If a claim is forwarded in a loop then it occupies no space because the owner does not need to store it, but simulations show that this rarely happens in practice. Samsara's claim mechanism is implemented in the Flŭd distributed backup system [513].

Reciprocation in Rings

Ackemann *et al.* [514] describe how reciprocation can be extended to multiple services by looking for *bartering rings* in which each node provides a service to the next node at a level determined by the quality of service received from the previous node. Gauthier *et al.* [515] suggest a similar scheme, while Anagnostakis and Greenwald [516] and Bocek *et al.* [517] describe protocols for finding suitable rings in peer-to-peer networks.

PledgeRoute [518] combines rings with payment balances to enable indirect reciprocation. Each node uses biased random walks to sample the *contribution graph* between nodes. When a node i wants to request a service from a node j that does not owe it any reciprocation, i uses its local view of the contribution graph to find a path to j in which every node owes a debt to the previous node. These debts are then reduced by an equal amount, and a new debt is created from j to i for the same amount, without any node's overall wealth changing. i can then request a service from j to cancel the debt. Since this process removes debt from the graph and thus makes subsequent paths harder to find, nodes also look for rings in the contribution graph where each node can create an artificial debt to the next node without any services being exchanged.

Contribution Ratios

SLIC [519] is an incentive mechanism for peer-to-peer search overlays in which each node allocates capacity to its neighbours' queries based on the number of search results they have supplied in the recent past. Simulations show that malicious nodes cannot flood the network by allocating most or all of their capacity to their own queries, since by doing so they decrease their neighbours' satisfaction with their performance and thus decreases the capacity allocated to them. However, rational nodes can increase the number of results they receive by sending slightly more queries than the average. It is not clear what determines the level at which the number of results starts to decrease again due to decreased cooperation from neighbours, or what would be the effect of all nodes gradually increasing their query rates.

SLIC addresses the problem of whitewashing by assigning a value to new connections that depends inversely on the node's level of satisfaction with its current connections: a node that is satisfied gives a low value to new connections, while a dissatisfied node is more likely to take a risk on an unknown neighbour. This would seem to create a risk of positive feedback, with dissatisfied nodes becoming increasingly vulnerable to whitewashers.

OneSwarm [420] uses BitTorrent's reciprocation algorithm to encourage cooperation between members of anonymous download swarms who may be separated by multiple hops. However, this does not create an incentive for intermediate nodes to forward data between the members of the swarm. Nodes allocate capacity to their neighbours in proportion to their contribution ratios, but the authors note that while forwarding data from one neighbour to another increases a node's contribution ratio at the downstream neighbour, it harms the node's ratio at the upstream neighbour, so the ratios may not create any incentive to forward data.

Feldman *et al.* [495] investigate a reciprocative decision function in which a node providing and requesting services in a peer-to-peer network uses its own generosity to judge the generosity of other nodes. If p_i and c_i are the number of services provided and consumed, respectively, by node i, then $g(i) = p_i/c_i$ is i's generosity, and $g_j(i) = g(i)/g(j)$ is i's normalised generosity as perceived by node j, which can use $g_j(i)$ to decide whether to provide services to i. Simulations show that this decision function performs well in small networks, but cannot scale to large networks with high node turnover; when the expected number of interactions with a given node is low, reciprocation does not provide a strong enough incentive to outweigh the cost of cooperating.

Three techniques for mitigating this problem are described. First, nodes selectively request services from nodes with which they have successfully interacted in the past, either as providers or consumers. Second, nodes share information about their past interactions – a reputation system based on maximum flow is presented and shown to be robust to collusion. Third, nodes adapt their behaviour towards strangers, becoming less generous in the presence of whitewashers. This can cause cooperation to collapse, as reciprocative players become unwilling to risk cooperating

with one another. However, when combined with the reputation system, it is shown that a small number of altruistic nodes can sustain cooperation by allowing reciprocative players to earn good reputations.

Lai *et al.* [520] find that reciprocation cannot sustain cooperation in large networks where any node is free to request service from any other. Cooperation can be maintained through a combination of reputations for established identities and an adaptive policy towards new identities that reduces cooperation in the presence of whitewashers.

Friedman and Resnick [72] show that rational nodes have an incentive to keep the same identities if established nodes can force newcomers to "pay their dues" by providing resources without reciprocation. Newcomers can later regain some but not all of the dues they have paid by demanding dues from others. This scheme requires agreement about which nodes are newcomers, however; if a new connection is created between two established nodes, it is not clear which of them should pay dues, but if neither pays then neither has an incentive to risk cooperating.

Reciprocation in Packet Forwarding

Srinivasan *et al.* [521] ask whether it is rational for nodes in multi-hop networks to forward packets. Nodes are assumed to benefit from the delivery of the packets they generate and to incur a cost when they forward packets. Each node uses its own throughput to decide probabilistically whether to cooperate with forwarding requests, without distinguishing between cooperative and uncooperative neighbours. Under these assumptions it is shown to be rational for all nodes to cooperate, since any defection causes cooperation to collapse throughout the network, harming the defector's throughput. (It follows that malicious nodes can deliberately cause throughput to collapse.)

Félegyházi *et al.* [522] use a similar model: each node divides the number of packets delivered on its behalf by the number of packets it has forwarded, and cooperates if the result is above a certain threshold. Again, cooperation is sustained by the threat of universal defection. Higher thresholds are needed to sustain cooperation in topologies that place uneven forwarding load on the nodes; in a simulated mobile *ad hoc* network, the required threshold is shown to decrease as mobility increases, since mobility reduces the likelihood that a node will have to forward more than its fair share of packets for a long period.

In the model of Urpi *et al.* [523], nodes can distinguish between cooperative and uncooperative neighbours. Mobility has a different effect here than in the model of Félegyházi *et al.*: the incentive to cooperate decreases with increased mobility, as nodes interact for shorter periods, reducing the probability that they will be rewarded for cooperating or punished for defecting. Urpi *et al.* assume that nodes can tell whether their neighbours are forwarding packets, which may not be possible in real wireless networks due to interference, variable-power transmitters and hidden terminals. They also use a utility function that could be said to beg the question: a node benefits from the delivery or further forwarding of the packets it forwards, regardless of their origin, as well as from its own throughput.

An important observation made in this paper is that a rational, resource-limited node can benefit not only by avoiding forwarding packets for other nodes, but by avoiding sending its own packets if there is a low probability that they will reach their destinations.

Game theoretic models of reciprocation are discussed further in Chapter 4.

Chapter 3

Analysis and Proposed Design

> "Persecution, then, cannot prevent independent thinking. It cannot even prevent the expression of independent thought. For it is as true today as it was more than two thousand years ago that it is a safe venture to tell the truth one knows to benevolent and trustworthy acquaintances, or more precisely, to reasonable friends."
> – Leo Strauss, *Persecution and the Art of Writing*

This chapter assesses the extent to which the systems surveyed in Chapter 2 are suitable for private, censorship-resistant communication, and outlines the design of a new censorship-resistant communication system based on friend-to-friend networks. The technical contributions presented in Chapters 4, 5 and 6 could serve as building blocks in the design outlined here, or in other designs.

3.1 Evaluation of Existing Work

The systems surveyed in Chapter 2 are intended for a range of different purposes, but when assessing their suitability for private and censorship-resistant communication, a number of common issues become apparent.

- **Certified identities.** Many designs rely on certified identities to prevent Sybil, eclipse and whitewashing attacks. It is possible for small groups of users to employ certified identities without relying on a central certificate authority, but this depends on every user being able to verify every other user's identity out-of-band (see section 2.1.2). In networks where such verification is impractical due to scale or undesirable due to privacy concerns, certified identities require centralisation.
- **Address harvesting.** By collecting the IP addresses of the people participating in an Internet-based communication system, an adversary may be able to block access to vital parts of the system, or may intimidate or punish its users. With enough effort, a global observer may be able to enumerate the participants in any Internet-based system, but some systems, such as friend-to-friend networks, make this task more difficult than others.
- **Verifiability.** Many of the attacks described in the previous chapter, from black hole attacks against routing protocols to lookup capture in structured overlays, are caused by nodes relying on unverifiable information supplied by other nodes, or making unsupported inferences from verifiable information. For example, victims of the black hole attack assume that nodes that participate in route discovery will also participate in data delivery (see section 2.3.3). Recent work on secure lookups in structured overlays demonstrates the difficulty of making distributed protocols verifiable at every step (see section 2.4.5).
- **Long-term traffic analysis.** Even in the strongest anonymity systems, long-term intersection and disclosure attacks can eventually link communicating users (see section 2.4.2). In client-server systems these attacks can be carried out externally by monitoring the servers, while in some peer-to-peer systems they can be carried out internally by collecting the addresses of active nodes. The best current defence against such attacks would seem

to be latency: to the extent that communicating users do not have to be active at the same time, they are harder to link.

- **Short-term traffic analysis.** With the exception of the high-latency mix networks discussed in section 2.4.3, few systems attempt to disguise the timing or volume of traffic from a global observer. In a system with a restricted topology, such as a cascade, it is feasible to use cover traffic to conceal the volume of genuine traffic on each link, but anonymity systems with entry and exit points are still vulnerable to end-to-end traffic analysis.
- **Resource contribution.** Four approaches to encouraging resource contribution were described in section 2.5: micropayments, reputations, audits and reciprocation. Of these approaches, only reciprocation appears to be suitable for private, censorship-resistant communication: micropayments require certified identities, centralised infrastructure or special hardware, while reputations and audits reveal information about users' interactions that they may wish to keep private.

3.2 Outline of the Proposed Design

In this section we sketch the high-level design of a censorship-resistant communication system motivated by the issues with existing systems that were summarised above. The design sketch provides a frame of reference for the technical contributions presented in later chapters, although they could also be used in other contexts.

3.2.1 Roles

The proposed system is a peer-to-peer network consisting of autonomous nodes. Each node is a software process running on an Internet-connected computer on behalf of an individual user, who is referred to as the node's owner. As stated in section 1.3, each user is assumed to have one or more friends, whom she trusts not to reveal her participation in the censorship-resistant communication system. Each node maintains application-layer connections across the Internet to the nodes of its owner's friends, which are referred to as the node's neighbours.

3.2.2 Infrastructure

The infrastructure required by the proposed system consists of Internet-connected computers and standard transport-layer protocols for establishing connections between processes running on those computers.

Each user must have access to an Internet-connected computer, such as a personal computer or workstation, and the ability to run software on that computer. We do not mandate any particular transport-layer protocol for the connections between neighbouring nodes; standard protocols such as TCP and UDP would be suitable. Privileges that are usually granted only to administrators, such as accessing raw sockets or binding low-numbered TCP or UDP ports, are not required.

It is assumed that any computer may be offline for a significant fraction of the time, and that the node software may not be running at all times. Whenever a node is online, it will attempt to maintain connections to its neighbours by initiating outgoing connections and accepting incoming connections.

3.2.3 Security Contexts

Each pair of friends must establish a shared secret in order to ensure the confidentiality, integrity and authenticity of the communication between their nodes. Similarly, a shared secret must be established between each pair of users who wish to communicate indirectly, to ensure the confidentiality, integrity and authenticity of their communication.

Any computer on which a node is running must ensure the secrecy and integrity of the keys used by the node and its owner in the above contexts. As discussed in section 1.3, this may be difficult to guarantee in practice, but solutions to the huge and varied problems of secure computer administration are outside the scope of the present work.

3.2.4 Features of the Proposed Design

- **Friend-to-friend overlays.** The connections between neighbouring nodes in the proposed system form one or more friend-to-friend overlays. As noted in section 1.3, we do not assume that all users of the system belong to a single social network, so the system may consist of many separate overlays, but any two users who wish to communicate with one another are assumed to belong to the same social network.

  When two users first decide to create a connection between their nodes, they must establish a shared secret. To do this, each user generates a public/private key pair and gives the public key to the other user, either face-to-face or through a mutually trusted third party. The users then follow a standard protocol such as Station-to-Station [524] to calculate a shared secret, and encryption and authentication keys for the connection are derived from this secret. The calculation of the shared secret is only necessary the first time a connection between two users is established.

  The proposed system does not provide a lookup service, such as a distributed hash table, for nodes to discover the current addresses and ports of their neighbours, so friends must initially exchange these details out-of-band. If two neighbouring nodes go offline simultaneously, their owners may need to exchange updated addresses out-of-band when their nodes come back online. Unlike the initial exchange of public keys, this exchange does not need to happen face-to-face or through a mutually trusted third party; encryption and authentication keys derived from the shared secret can be used to send the updated addresses securely across any available channel, such as email, even if the channel does not provide its own confidentiality, integrity or authenticity guarantees.

- **Cover traffic.** Each node in the proposed system transmits packets[1] to its neighbours in round-robin order at a constant rate. The size of each packet and the rate at which packets are transmitted are independent of the amount of data waiting to be sent.

  When a packet is being prepared for transmission, as much waiting data as possible is added to the packet. Any remaining space is then filled with *padding* before encrypting and authenticating the packet. A packet may consist entirely of padding if there is no data waiting to be sent.

  Padding cannot be distinguished from data by an external observer, but a node receiving a packet can recognise and discard the padding after authenticating and decrypting the packet. Padding is not forwarded across the overlay.

  The round-robin policy means that each node divides its outgoing bandwidth equally among its online neighbours. Due to variations in the number of online neighbours, a node's transmission rate to each neighbour may vary even though its overall transmission rate remains constant.

- **Address-free routing.** We introduce the term *address-free routing* to denote any protocol that delivers data across a multi-hop network without relying on identifiers with global scope or explicit knowledge of the topology. The proposed system uses address-free routing to enable any two users who belong to the same friend-to-friend overlay to communicate across the overlay. Any node can act as a source, destination or relay in the address-free routing protocol.

  As mentioned in section 3.2.3, users who wish to communicate indirectly must first establish a shared secret. As with the direct connections between friends, this is done by exchanging public keys out-of-band or through a mutually trusted third party and then following a standard protocol such as Station-to-Station to calculate a shared secret, from which encryption and authentication keys are derived. The shared secret only needs to be calculated once, the first time two users communicate.

- **Private file sharing.** Like many of the systems described in section 2.2.5, the proposed system supports private file sharing between neighbouring nodes. File transfers are assumed to be latency-insensitive, so they are given lower priority than the messages of the address-free routing protocol when selecting data for transmission to neighbouring nodes.

- **Reciprocation.** To prevent resource-depletion attacks and to encourage users to contribute the resources needed for communication, the proposed system uses an incentive mechanism based on reciprocation between neighbouring nodes. The incentive mechanism allocates resources in such a way that contributors receive better service from the system than non-contributors.

[1] We use the generic term *packet* to refer to any transport layer protocol data unit, such as a segment in the case of TCP or a datagram in the case of UDP.

3.3 Rationale

3.3.1 Friend-to-Friend Overlays

The friend-to-friend structure of the proposed system contributes to achieving the goals defined in section 1.2 in four ways:

1. By restricting their direct connections to people they know and trust, users reveal their participation in the censorship-resistant communication system to as few people as possible, which helps to protect them from intimidation and coercion, as well as protecting the system as a whole from attacks based on address harvesting.
2. By running their own autonomous nodes, users minimise their dependence on infrastructure that is outside their control, and do not need to place their trust in any single entity.
3. Because friend-to-friend connections are based on the identities of known individuals, they are not susceptible to whitewashing or eclipse attacks.
4. Because friend-to-friend connections are based on long-term relationships, reciprocative incentive mechanisms can be used to encourage resource contribution.

In principle, a friend-to-friend overlay can be constructed over any mixture of networks to which the users have access, such as the Internet, the telephone system, and local wireless networks – a possibility we call a *hybrid overlay*. We assume, however, that connections across the Internet will be the easiest for users to set up and maintain, and we concentrate on Internet-based overlays in the present work.

Freenet's experiment with a global friend-to-friend structure shows that the potential users of a communication system do not necessarily belong to a single, well-connected social network, especially during the early stages of the system's adoption. Network designers must therefore choose between building a single global network that may require connections between strangers, as Freenet has done, or allowing users to build separate local networks that can later be merged if their memberships overlap. Both approaches have their advantages: a global network may increase the size of each user's anonymity set, while a local network can retain a purely friend-to-friend structure and may attract less attention. In this work we choose the local approach, which restricts our design choices somewhat – for example, we cannot use protocols that would break down if connections were created between two mature networks.

Middleboxes, dynamic IP addresses and churn are likely to make strict friend-to-friend overlays fragile: neighbouring nodes may not be able to locate each other after moving to new IP addresses, or they may not be able to traverse a firewall or network address translator. However, relying on an internal or external lookup service, such as a distributed hash table, could undermine the privacy advantages of the friend-to-friend approach, so the proposed system does not use such a service.

3.3.2 Cover Traffic

Each node in the proposed system sends traffic at a constant rate to prevent an external observer from determining the volume of genuine traffic passing across any friend-to-friend connection. The size and timing of packets is kept independent of the amount of data waiting to be sent, preventing an external observer from recognising the correspondence between traffic entering and leaving a node.

This is an expensive defence, but less so in a friend-to-friend network than in many other kinds of network: the restricted topology means that it is only necessary to provide cover traffic on a limited number of links, while private file sharing provides a plentiful source of latency-insensitive traffic that can be used to keep the volume of traffic constant without using padding, whenever a file transfer is in progress. The true cost of cover traffic will therefore depend on the popularity of private file sharing and other forms of direct communication between friends, relative to indirect communication.

Due to variations in the number of online neighbours, a node's transmission rate to each neighbour may vary even though its overall transmission rate remains constant. Neighbours and external observers may thereby be able to tell when a node's number of online neighbours changes, and the throughput or latency of any traffic the node is generating

or forwarding across the overlay may be affected. In future work we plan to investigate whether this variation reveals any information to an observer that would be concealed by keeping the transmission rate on each link constant, which would be a less efficient strategy than keeping each node's overall transmission rate constant, since bandwidth allocated to neighbours that are currently offline cannot be reallocated to other links.

Further work is also needed to determine whether fixed or random packet sizes and inter-packet delays provide better protection against an observer attempting to trace messages through the overlay by combining internal and external observations.

Limitations of Cover Traffic

The cover traffic strategy described above is designed to ensure that an external observer learns nothing about the volume of genuine traffic on any friend-to-friend connection from the observable traffic on that connection. There are some limits to what cover traffic can achieve, however. First, the observer knows that the genuine traffic on a connection cannot be greater than the observable traffic. Second, as noted by Newman *et al.*, the genuine traffic entering or leaving any node or group of nodes cannot exceed the observable traffic (see section 2.4.2). Third, nodes that are offline cannot send or receive traffic, so the observer can use nodes' periods of uptime and downtime for coarse-grained timing analysis – for example, two users whose nodes are never online at the same time cannot be communicating by any low-latency protocol.[2]

Traffic Classification

Encryption and cover traffic may not prevent an observer from recognising that an individual is using a censorship-resistant communication system: encrypted protocols usually have an unencrypted handshaking phase during which encryption and authentication parameters are negotiated [525, 526, 527], and traffic characteristics such as the number and duration of connections can be used to distinguish between application classes (see section 2.4.2).

It is probably impossible to prevent a human observer who can examine an individual's Internet traffic from determining whether that person is using a given application, but we should aim to make automatic determination as difficult as possible, by disguising the individual connections as well as the fact that the connections belong to the same application.

Individual connections can be disguised by encrypting all application-layer data, including handshaking messages and headers; however, this requires the encryption and authentication parameters to be agreed in advance, and the absence of unencrypted headers might in itself help to identify the protocol. Alternatively, connections can be disguised as other applications with similar traffic patterns, such as peer-to-peer file sharing or video conferencing, which often use long-lived encrypted connections with little variation in the transfer rate. If each connection is disguised as a different application and uses a different port number, it might also be hard for an observer to determine that the connections belong to the same application.

In the long term, however, any attempt to conceal the system's traffic can only result in an arms race between concealment techniques and detection techniques, and since we cannot be sure that we are ahead in the race, we must assume that the adversary can identify the system's traffic.

3.3.3 Address-Free Routing

A friend-to-friend overlay only permits direct communication between users who are friends; in order to allow users who are not friends to communicate, the proposed system must support indirect communication across the friend-to-friend overlay.

We approach this as a routing problem: the challenge is to find paths across a network with an unknown structure that is subject to constant change due to churn and the evolving social relationships between users. In an unusual twist, we would like to *avoid* discovering the structure of the overlay during routing, because if such information were available, it could be used by an internal adversary to carry out long-term traffic analysis and to identify important members of the social network.

[2] Users should therefore be encouraged to keep their nodes online even when they are not communicating.

The systems described in Chapter 2 use six methods to find routes across restricted topologies: structured overlay routing, source routing, table-driven routing, probabilistic routing, on-demand routing, and flooding.

1. **Structured overlay routing**, as used by Sandberg (see section 2.2.3) and by Vasserman *et al.* (see section 2.2.5), builds a structured overlay across a restricted topology and uses the structured overlay's routing algorithm to connect sources and destinations. There are three issues with using this approach in friend-to-friend networks. The first is that the topology may not be suitable: Vasserman *et al.* require every node to have between $\lceil k/2 \rceil$ and k neighbours, for some system-wide constant k, while Sandberg requires the network to be a navigable small world (see section 2.1.1).

   The second issue is that the structured overlays used by Vasserman *et al.* and Sandberg do not allow mature networks to be merged, although the example of SkipNet suggests that it might be possible to overcome this problem by using hierarchical identifiers (see section 2.2.3).

   The third and most serious issue is that Sybil attacks can be used to occupy an arbitrarily large fraction of the structured overlay's key space, thus capturing an arbitrarily large fraction of routes.

2. **Source routing** is used in Perlman's robust link state protocol (see section 2.3.3), and in the restricted stratified mix topology described by Díaz *et al.* (see section 2.4.4). In source routing, the source needs to discover the topology between itself and the destination in order to select a route. However, as shown in section 2.3.3, without certified identities it may not be possible to discover the topology in a robust way: tunnelling attacks can create the appearance of short routes through corrupt nodes, while Sybil attacks can defeat fault detection mechanisms and the use of redundant routes.

3. **Table-driven routing**, which is used by certain *ad hoc* routing protocols (see sections 2.3.3.1 and 2.3.3.6), maintains a table at each node indicating the best next hop for each destination. The difficulty with this approach in an adversarial context is that the routing tables are based on information provided by other nodes, which may not be trustworthy. Even if the information in the routing tables is correct, table-driven routing is vulnerable to black hole attacks.

4. **Probabilistic routing**, which includes ant-inspired protocols such as ARA and learning-based protocols such as Q-routing and Cognitive Packet Networks, uses exploration and feedback to guide a probabilistic forwarding process (see section 2.3.3.2). As with table-driven routing, the difficulty is that nodes may disrupt routing by supplying false information about the network. MUTE uses digital signatures and timestamps to authenticate source addresses in a probabilistic routing protocol, but this only proves that the path is reliable in the direction leading *away* from the node in question.

5. **On-demand routing**, which is used by anonymous *ad hoc* routing protocols such as ASR, does not require explicit knowledge of the topology: routes can be discovered by flooding without identifying the endpoints to the relays or *vice versa*, making global identifiers unnecessary and thus limiting the scope of identity-related attacks (see section 2.4.7). The private peer-to-peer network WASTE uses flooding in a similar way, while Turtle uses flooding to establish virtual circuits (see section 2.2.5). However, all of these protocols are vulnerable to black hole attacks.

6. **Flooding** can also be used on its own – it requires no knowledge of the topology and is robust to black hole attacks, since there is no distinction between route discovery and data delivery. Unfortunately, its scalability is poor. This motivates our search for a routing protocol that is more scalable than flooding and more robust than on-demand routing, while sharing their resistance to identity-related attacks.

We refer to the process of routing without relying on global identifiers or explicit knowledge of the topology as *address-free routing*. Flooding and on-demand routing are examples of existing techniques for address-free routing. The proposed system uses an address-free routing protocol to support indirect communication across the friend-to-friend overlay. Any node can act as a source, destination or relay in the protocol, making it difficult for external observers to distinguish sources and destinations from relays through traffic analysis. Because the protocol does not use identifiers with global scope, it maintains unlinkability between sources and destinations.

The *end-to-end principle* suggests that networks should provide a simple, general-purpose communication service, minimising the complexity of relays and allowing new applications to be deployed incrementally [466]. The proposed system therefore uses address-free routing to provide an unreliable datagram service similar to that provided by the Internet Protocol. We leave the design of higher protocol layers for future work.

Onion Encryption

Unlike many existing systems that aim to provide anonymity or unlinkability (see sections 2.4.3 and 2.4.4), the proposed system does not use onion encryption. If the adversary controls more than one node on a path used by the address-free routing protocol, the corrupt nodes will be able to tell that they are on the same path.

Onion encryption provides two benefits in existing systems: it prevents an external observer from recognising the correspondence between traffic entering and leaving a mix, and it prevents corrupt mixes from recognising that they belong to the same communication path.

The first benefit is already provided by the proposed system because each friend-to-friend connection is encrypted and authenticated using a different shared secret. The second benefit is only worthwhile if there is no other way for corrupt mixes to recognise that they belong to the same communication path. In low-latency mix networks, corrupt mixes may be able to use the timing or volume of traffic to distinguish between circuits, or they may be able to manipulate those characteristics – for example, by embedding watermarks in inter-packet delays – to signal to one another that they belong to the same circuit (see section 2.4.2). We therefore consider it doubtful whether onion encryption provides sufficient benefits in a low-latency system to outweigh the problems of key distribution and mix selection that it introduces.

3.3.4 Private File Sharing

The *samizdat* publishing network in the Soviet Union has shown that censored information can spread through social networks even when copying is laborious [21], which suggests that secure friend-to-friend connections could be useful for censorship-resistance even if they only allowed users to share documents with their friends. To take advantage of this possibility, the proposed system supports private file sharing between directly connected friends. As files are re-shared and disseminated through the overlay, anonymous or pseudonymous authors may be able to reach audiences beyond their immediate friends. Because files are stored at intermediate nodes, authors and readers who are not directly connected need not be active at the same time, protecting them against long-term traffic analysis.

As a method of indirect communication, however, disseminating files in this way has several disadvantages: it is likely to be slow, it depends on each author's readers forming a connected subgraph in the social network, and it does not provide end-to-end confidentiality. These shortcomings indicate the need for a general-purpose method of indirect communication across the overlay, such as the address-free routing described above, in addition to private file sharing.

3.3.5 Reciprocation

Of the four approaches to encouraging resource contribution discussed in section 2.5, reciprocation is the most suitable approach for the proposed system: it relies on long-term relationships between neighbouring nodes, which friend-to-friend networks are likely to provide, but it does not require global identities or centralised infrastructure.

There are two problems to be solved in the design of a reciprocative incentive mechanism, however. First, a node must be able to measure the level of service received from each neighbour, and second, the incentive mechanism must resist manipulation by selfish users.

In the case of private file sharing the first problem is straightforward: the level of service received can be measured either by the amount of data received from each neighbour, as in SWIFT, or by the rate at which data is received, as in BitTorrent (see section 2.5.4). In the case of forwarding data across the overlay, however, it is more difficult to tell whether a neighbour is cooperating. Even in unencrypted wireless networks it may not always be possible to tell whether a neighbour has retransmitted a packet, due to effects such as interference, and in an Internet overlay with encrypted connections there is no direct way to monitor a neighbour's forwarding behaviour.

The second problem – designing an incentive mechanism that resists manipulation by selfish users – is difficult even in the case of file sharing, where cooperation is easily measured: the success of BitTyrant and Tribler at outperforming standard BitTorrent implementations (see section 2.5.4) demonstrates that robust incentive mechanisms cannot be based on heuristic methods of reciprocation.

3.4 Design Strategy

Two major problems remain unsolved in the proposed design outlined above: we require an address-free routing protocol that maintains unlinkability between sources and destinations, and a reciprocative incentive mechanism that can measure the cooperation received from neighbouring nodes and reciprocate in a way that resists manipulation. Both solutions must be robust to arbitrarily complex behaviour by faulty nodes and selfish users.

Our general strategy for solving these problems is *failure mode reduction*: we try to reduce complex faults to simpler, detectable faults. We then avoid relying on faulty nodes, and limit the resources they can consume. This minimises the impact of any attacks faulty nodes might carry out, while creating an incentive for selfish users to operate reliable nodes.

The first building block of this strategy is *information restriction*: by limiting the information available to nodes we attempt to restrict the complexity of the faulty behaviour they can exhibit. Information restriction also helps to protect privacy.

The second building block is *verifiability*: by making correct behaviour verifiable, we enable nodes to measure the reliability of their neighbours. Crucially, we do not attempt to distinguish between a neighbour that is itself faulty, and a neighbour that is unreliable because it depends on faulty nodes.

The third building block is *reciprocation*: by preferentially allocating resources to reliable neighbours, we prevent faulty nodes and selfish users from exhausting the network's resources.

In Chapter 6 these building blocks are combined in the design of two address-free routing protocols, both of which use the unforgeable acknowledgement mechanism described in Chapter 5 for reliability measurement, and one of which uses the model of reciprocation developed in Chapter 4 to allocate resources preferentially to reliable nodes.

Chapter 4

Cooperation in Distributed Networks

> "Hence I learn to do a service to another, without bearing him any real kindness; because I foresee, that he will return my service, in expectation of another of the same kind..."
> – David Hume, *A Treatise on Human Nature*

4.1 Game Theoretic Models of Cooperation

Many researchers have used *game theory* to study the problem of encouraging cooperation in networks that depend on resources provided collectively by the users. If a network is viewed as a group of self-interested individuals interacting according to rules specified by the protocol designer, then game theory provides tools for modelling the behaviour of rational participants, and *mechanism design* can be used to create protocols that reward cooperation, encouraging rational participants to behave in ways that benefit the network [528, 529, 530].

It might seem reductive to regard the participants in a communication network as simple egoists, but when modelling a communication network we are concerned with the *behaviour of nodes* rather than the *intentions of users*: game theory is not appropriate for modelling all human interactions, but it is well suited to modelling those interactions in which humans delegate routine decisions to software, reducing complex social considerations to a choice between programmatic 'strategies'. Programs might not be capable of fully or accurately reflecting their users' intentions, but since it is programs rather than users that implement the network's protocols, it is the concrete behaviour of programs that we must attempt to model.

While game theory is useful for reasoning about the behaviour of self-interested individuals, it does not depend on the assumption that everyone is selfish. Even if most participants in a network are not selfish, game theory allows us to assess the network's vulnerability to exploitation by a selfish or malicious minority.

4.1.1 The Prisoner's Dilemma

Simple games can embody surprisingly complex problems, and perhaps no simple game has received more attention than *the prisoner's dilemma*, a single-round game for two players. Each player chooses between two actions, *cooperation* and *defection*, and receives a payoff that depends on the choices of both players: T is the 'temptation' payoff for unilateral defection, R is the 'reward' payoff for mutual cooperation, P is the 'punishment' payoff for mutual defection, and S is the 'sucker' payoff for unilateral cooperation [531].

The dilemma arises because $T > R > P > S$, which means a rational player will defect regardless of her opponent's choice. The players cannot escape the dilemma by communicating about their intentions, because a rational player will claim that she intends to cooperate, but then defect. Thus rational players always defect, leading to a suboptimal payoff $P < R$ for both players.

The prisoner's dilemma has been used to model a wide range of situations in nature and society where the benefit of cooperation is greater than the cost. Wahl and Nowak [532] describe the prisoner's dilemma in terms of the cost of

cooperating, c, and the benefit of receiving cooperation, b. The restriction $b > c > 0$ leads to the payoff structure described above. Roberts and Sherratt [533] describe the dilemma using a single parameter $k = b/c$.

Public goods problems, social dilemmas [534] and reciprocal altruism [535] find natural expression in the form $b > c > 0$, but not all prisoner's dilemmas can be expressed in this way: for example, many studies use the payoffs $T = 5, R = 3, P = 1, S = 0$. In this chapter we will only consider dilemmas that arise from the costs and benefits of cooperating, and can therefore be expressed in the form $b > c > 0$.

4.1.2 The Shadow of the Future

Although rational players always defect in the single-round prisoner's dilemma, it may be possible to establish cooperation if the game is repeated for more than one round. Players who expect to interact for many rounds must consider the long-term effects of their short-term decisions: automatic defection is no longer necessarily the best strategy, because players have the chance to recognise cooperative opponents and gain a higher payoff through mutual cooperation (although each player will still be tempted to defect once her opponent has started to cooperate).

Simple strategies for the repeated prisoner's dilemma include Tit For Tat, which cooperates in the first round and thereafter copies its opponent's action from the previous round [536]; Win Stay Lose Shift, which cooperates in the first round and thereafter repeats its previous action if it receives cooperation, or switches to the other action if it suffers defection [537]; and Stochastic Tit For Tat, which cooperates with a probability equal to the fraction of rounds in which the opponent has cooperated [538]. Each of these strategies creates an incentive for rational opponents to cooperate while minimising losses against uncooperative opponents.

4.1.3 Multi-Player Dilemmas

When a dilemma involves more than two players, the model must specify which actions affect which players. If each player chooses one action that affects all her opponents, the situation is a *social dilemma* [534]; encouraging cooperation in such situations is harder than in two-player games, because it is not possible to cooperate with cooperators while defecting against defectors [521, 522, 539].

In a *networked social dilemma*, each player chooses one action that affects her neighbours in a spatial lattice or other network [540, 541, 542, 543]. Cooperation can succeed if cooperative players' interactions with other cooperators sufficiently outnumber their interactions with defectors.

Finally, if each player can choose a different action for each opponent, the situation can be modelled as a *network of two-player games*. Several networked variants of the prisoner's dilemma have been developed [544, 545, 546, 547], all based on the assumption that a player's choices and payoffs in her pairwise games are independent. However, if the payoffs represent the costs and benefits of cooperating, we may ask whether the assumption of pairwise independence is always realistic: there may be situations in which a player has limited resources for cooperation, but can allocate them freely among her pairwise interactions. In such cases no two-player strategy indicates how best to allocate her scarce resources.

We argue that many of the situations modelled as networked dilemmas fit this pattern: players can allocate their resources unevenly, cooperating more with some opponents than with others. Hunters sharing food, animals grooming one another, and peer-to-peer nodes uploading files are all faced with opportunities to strengthen or weaken cooperative relationships by choosing how much to share, and with whom.

4.1.4 The Sharer's Dilemma

To explore the problem of allocating resources in networked dilemmas we propose a simple extension to the prisoner's dilemma, incorporating scarcity into the game by limiting the number of times a player can cooperate in each round. We call this new game *the sharer's dilemma*. The prisoner's dilemma can be viewed as a special case of the sharer's dilemma in which the limit is high enough that cooperation with every opponent is possible in every round. Strategies from the prisoner's dilemma can be adapted to the sharer's dilemma by specifying how to choose between opponents when multiple opponents are eligible for cooperation.

As with the prisoner's dilemma, many variants of the new game are possible, but here we will only consider the simplest case: in each round, a player can either cooperate with one chosen opponent or defect against them all. While simple, this starting point captures the essential problem of cooperation under scarcity: when resources are limited, the question is not only how to establish and sustain cooperation with each opponent, but how to prioritise the demands of different opponents in order to maximise the total cooperation received.

In this initial exploration of the sharer's dilemma we make the following assumptions, which are also commonly used when studying networked variants of the prisoner's dilemma:

- Players can recognise opponents with whom they have interacted in the past. Players cannot change their identities at will or maintain multiple identities.
- The structure of the network is stable enough to provide a reasonable expectation that any two players who interact in a given round will interact again in the next round.
- Players always carry out their chosen actions, and a player's actions can always be correctly recognised by her opponents.

Analysing the sharer's dilemma is, unfortunately, more difficult than analysing the prisoner's dilemma, precisely because the sharer's dilemma captures that fact that under conditions of scarcity, a player's pairwise interactions are connected. Her chosen action in a given round may therefore depend on the actions of all her opponents in previous rounds, which may in turn depend on the actions of all *their* opponents in previous rounds, and so on. Standard analytical tools such as game dynamics [548] that make the simplifying assumption of independent pairwise interactions cannot be applied to such situations: to model the dynamics of the sharer's dilemma we would need to consider not only the history of a given interaction, but the history of the entire network. Such an approach would be practical only for very small networks.

Since analytical results are difficult to obtain, we explore the sharer's dilemma from two alternative directions: reasoning from first principles about the behaviour of self-interested players, and simulating networks of players who follow various strategies. The first approach leads to the expected utility strategy described in the next section, which demonstrates an interesting connection between the sharer's dilemma and rational decision theory, while the second leads to the simulations presented in section 4.2, which allow us to study the game dynamics on a scale that would not be feasible through analysis.

4.1.5 The Expected Utility Strategy

Savage [549] introduced the concept of *subjective expected utility* as a basis for rational decision-making. When an agent is considering an action x with several possible outcomes, the agent's subjective assessments of the probability, $p(x_i)$, and utility, $u(x_i)$, of each possible outcome x_i can be combined to calculate the expected utility of the action, $u(x)$:

$$u(x) = \sum_i u(x_i)p(x_i). \tag{4.1}$$

If we assume that it is possible to express costs and benefits in the same units then the utility of an outcome can be defined as its benefit minus its cost, $u(x_i) = b(x_i) - c(x_i)$:

$$u(x) = \sum_i (b(x_i) - c(x_i))p(x_i). \tag{4.2}$$

In Savage's model of decision-making, when more than one action is available to an agent, the rational choice is the action with the highest expected utility. The application of this principle to the sharer's dilemma is clear: in each round, a rational player should calculate the expected utility of cooperating with each neighbour and the expected utility of defecting, and choose the action with the highest expected utility. If a player expects that cooperating with an opponent will result in a higher level of cooperation in return, she can weigh the benefit of the opponent's expected reciprocation against the cost of cooperating, comparing the incentives offered by different opponents to obtain the

greatest benefit in return for her cooperation. This idea is the basis of our *expected utility strategy* for the sharer's dilemma.

A player using the expected utility strategy estimates the benefit of cooperating with each opponent under the assumption that all of the benefit received from the opponent so far is a result of reciprocation – in other words, she attributes the cooperation received to the cooperation given. The benefit of all the cooperation received in previous rounds divided by the number of times the player has cooperated in previous rounds is the *expected benefit* of cooperating in the current round. The player cooperates with whichever opponent offers the highest expected benefit, unless the highest expected benefit is less than the cost of cooperating, in which case she defects.

More formally, consider a player with n opponents, $o_1 \dots o_n$. In round t of the game, the player may cooperate with any opponent o_i, an action that we denote $a_i(t)$, or she may defect, which we denote $a_0(t)$. As in Wahl and Nowak's formulation of the prisoner's dilemma, we use b to represent the benefit of receiving cooperation and c to represent the cost of cooperating. The cost of defecting is 0, and the restriction $b > c > 0$ creates the dilemma. The cost of cooperating is the same for any opponent, so $u(a_i(t)) = b(a_i(t)) - c, \forall i \in \{1 \dots n\}$, while $u(a_0(t)) = b(a_0(t))$.

We now consider the expected utility strategy. Let $c_{ij} = 1$ if the player cooperated with o_i in round j, otherwise $c_{ij} = 0$. Let $b_{ik} = b$ if the player received cooperation from o_i in round k, otherwise $b_{ik} = 0$. Then $b_i(t)$, the expected benefit of cooperating with o_i in round t, is defined as:

$$b_i(t) = \frac{\sum_{j=1}^{t-1} b_{ij}}{\sum_{k=1}^{t-1} c_{ik}}. \tag{4.3}$$

The assumption made by the expected utility strategy that cooperation received is a result of cooperation given can be expressed as $b(a_i(t)) = b_i(t)$ and $b(a_0(t)) = 0$, from which it follows that $u(a_i(t)) = b_i(t) - c$ and $u(a_0) = 0$. Thus, if cooperation received is attributed to cooperation given, it is rational to cooperate with whichever opponent maximises the value of $b_i(t)$, unless $b_i(t) < c, \forall i \in \{1 \dots n\}$, in which case it is rational to defect.

When comparing her opponents in this way, a player does not need to know the cost or benefit of cooperation from any opponent's point of view – she only needs to estimate the cost to herself of cooperating, and the benefit to herself of the expected reciprocation. Costs and benefits in this model can therefore be subjective.

The expected utility strategy's attribution of cooperation received to cooperation given implies that continuing to cooperate with the same average frequency as in past rounds can be expected to result in receiving cooperation with the same average frequency as in past rounds, while increasing or decreasing the frequency of cooperation can be expected to result in a comcomitant increase or decrease from the opponent. In a sense, this amounts to an assumption that the opponent is also playing the expected utility strategy, or something very like it. We should therefore expect the strategy to perform well when most or all of its opponents are, in fact, playing the same strategy.

4.1.6 Bootstrapping Cooperation

Like any cooperative strategy, the expected utility strategy faces the problem of *bootstrapping*: when two players first meet, one or both of them must risk cooperating without knowing how much reciprocation (if any) will result. In the prisoner's dilemma, the successful strategies Tit For Tat and Win Stay Lose Shift take the simple approach of always cooperating in the first round, but this may not be possible in the sharer's dilemma, due to the limit on the amount of cooperation per round. One possible solution would be for the expected utility strategy to assign a high expected benefit to first-time interactions, but this might be vulnerable to exploitation by whitewashers who can continually change identities [550]; alternatively, the benefit of a first-time interaction could be estimated using the average benefit of previous first-time interactions [520].

Uncertainty about the duration of the game can be incorporated into the strategy by applying a *discount factor* to future payoffs, reducing the expected benefit of reciprocation if games tend to be short-lived [550]. The discount factor need not be the same for all opponents; if old players can be expected to outlive new players, as in many peer-to-peer networks, then it may be appropriate to use a heavier discount factor for new opponents (see section 2.2.1).

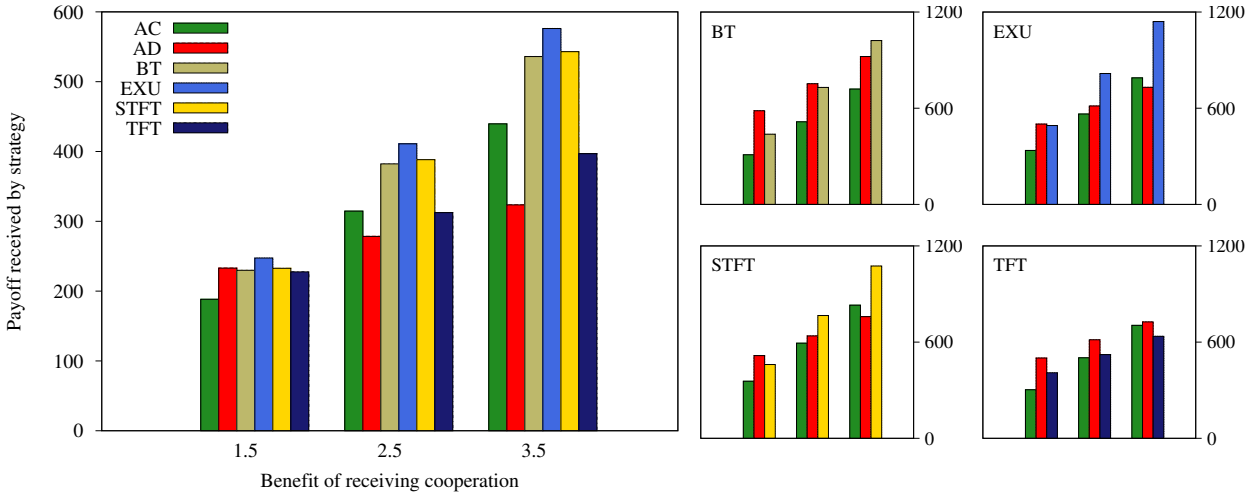


Figure 4.1: Fixed population proportions. The payoff received by each strategy as a function of b, the benefit of receiving cooperation, averaged over 20 runs. The large figure shows a population containing all six strategies; the small figures show populations containing AC, AD and each of the four reactive strategies.

4.2 Simulations

This section describes simulations to compare various strategies for the sharer's dilemma. Our model is a population of n players connected uniformly at random so that each player has d neighbours on average. In each round of the game, each player either cooperates with one of her neighbours, increasing the neighbour's payoff by b, or defects, increasing her own payoff by c. The players make their choices in a random order each round. In all of the simulations presented here, $n = 1,000$, $d = 10$ and $c = 1$. Three different values of b are simulated, as explained below.

We simulate two strategies adapted from the prisoner's dilemma: the first, Tit For Tat (TFT), cooperates with a neighbour chosen uniformly at random from those that cooperated with it in the previous round, or defects if no neighbours cooperated in the previous round. The second, Stochastic Tit For Tat (STFT), cooperates with a randomly chosen neighbour, choosing each neighbour with a probability proportional to the fraction of rounds in which the neighbour has cooperated. To bootstrap cooperation, TFT and STFT treat new neighbours as if they cooperated in the previous round.

We also simulate a strategy based on BitTorrent's incentive mechanism (see section 2.5.4). In each round the BitTorrent (BT) strategy cooperates with a randomly chosen member of its *active set*, which contains one randomly chosen neighbour and the s other neighbours that have provided the most cooperation in recent rounds. A new active set is chosen every r rounds, using an exponentially weighted moving average to measure the cooperation received from each neighbour. The results presented here use $s = 1$ and $r = 5$, which appear to give the best payoff for the BT strategy in this setting.

Finally, we simulate the expected utility (EXU) strategy described in the previous section, which cooperates with whichever neighbour provides the greatest expected benefit in return for cooperation, unless the cost of cooperating exceeds the expected benefit, in which case it defects. Cooperating with a neighbour lowers the expected benefit of cooperating with it in subsequent rounds, while receiving cooperation raises the expected benefit. To bootstrap cooperation, EXU treats new neighbours as though they have cooperated once and received cooperation once.

4.2.1 Fixed Population Proportions

Each simulation consists of 20 independent runs of 2,000 rounds each. At the end of each round a randomly chosen player is removed and replaced with a new player using the same strategy, who is connected to d randomly chosen neighbours. To prevent the initial conditions from influencing the results, no measurements are taken during the first half of the run. The payoff received by each strategy is averaged over the second half of the run.

In the first set of simulations, the population contains equal proportions of the four reactive strategies described above, along with the simple strategies Always Cooperate (AC) and Always Defect (AD). We vary the strength of the dilemma by simulating three different values of b, the benefit of cooperation; the cost $c = 1$ is held constant. The first frame of Figure 4.1 shows the payoff received by each strategy for $b = 1.5$, $b = 2.5$ and $b = 3.5$. When b is close to c, meaning that there is little benefit to be obtained by establishing cooperation, AD narrowly outperforms all of the reactive strategies except EXU. Increasing b favours the cooperative strategies at the expense of AD. For all values of b, EXU receives the highest payoff of any strategy.

We also evaluate each reactive strategy separately against AC and AD, as shown in the small frames of Figure 4.1. Each population contains equal proportions of AC, AD and the strategy being tested. Once again the outcome depends on the value of b. For $b = 1.5$, none of the reactive strategies does better than AD, although EXU comes close. However, as b increases, all of the reactive strategies except TFT do better than AD.

The poor performance of TFT can be ascribed to its short memory: even in a perfectly cooperative population, the probability of receiving cooperation from a given neighbour in a given round is only $1/d$. TFT only considers the previous round, so it is often unable to distinguish defectors from cooperators that happen to have cooperated with another player in the previous round. The other reactive strategies evaluate their neighbours over longer periods.

All of the results shown in Figure 4.1 are significant at the 99% level using the two-tailed Mann-Whitney U test. Student's t-test would not be suitable for making comparisons between strategies because the samples are not independently and randomly drawn from normally distributed populations. The nonparametric Mann-Whitney U test was chosen instead because it depends on fewer assumptions about the population distribution [551].

4.2.2 Evolutionary Simulations

To further investigate how each strategy fares in a mixed population, we now allow the population proportions to evolve. As before, at the end of each round a single player is chosen uniformly at random and replaced with a new player, but now the new player's strategy is chosen using the *roulette wheel method*: the probability of the new player adopting each strategy is proportional to the total payoff received by that strategy in the previous round. Thus the mixture of strategies in the population evolves according to the payoffs received: a player's payoff can be interpreted as her *reproductive fitness* [552].

To prevent any strategy from becoming extinct, if the strategy of the player being replaced is used by five or fewer players, the new player always adopts the endangered strategy. This allows strategies that are unsuccessful under certain conditions to re-emerge later in the game, and prevents strategies from being eliminated by random drift.

Figure 4.2 shows the evolution of the population for $b = 1.5$, $b = 2.5$ and $b = 3.5$, averaged over 50 independent runs of 100,000 rounds each. AC is quickly pushed to the edge of extinction: by cooperating equally with all of its neighbours, it wastes resources on AD that could have been used to earn reciprocation from reactive neighbours. TFT has trouble distinguishing between cooperators and defectors due to its short memory, and it too is quickly defeated. This is not a weakness of the Tit For Tat strategy as such, but only of our method of adapting it to the sharer's dilemma; STFT, which is also based on the Tit For Tat strategy from the prisoner's dilemma, does not have the same weakness.

Once AC has been eliminated, the other reactive strategies all outperform AD. It might seem paradoxical that any strategy would benefit from the elimination of altruists, but in evolutionary simulations it is relative payoffs, rather than absolute payoffs, that are decisive: the defeat of AC makes the environment harsher for the reactive strategies, but harsher still for AD, which receives most of its cooperation from AC.

Despite their selfishness, BT, EXU and STFT succeed in establishing almost full cooperation: after 20,000 rounds the fraction of players cooperating in each round is always above 95%. As in the simulations with fixed population proportions, EXU outperforms all of the other strategies for all values of b.

To test the significance of the results we use the two-tailed Mann-Whitney U test to compare the number of players using each strategy at the end of the 50 runs. All of the differences between strategies are significant at the 99% level, with the exception of AD and TFT, which are not significantly different when $b = 1.5$ or $b = 2.5$.

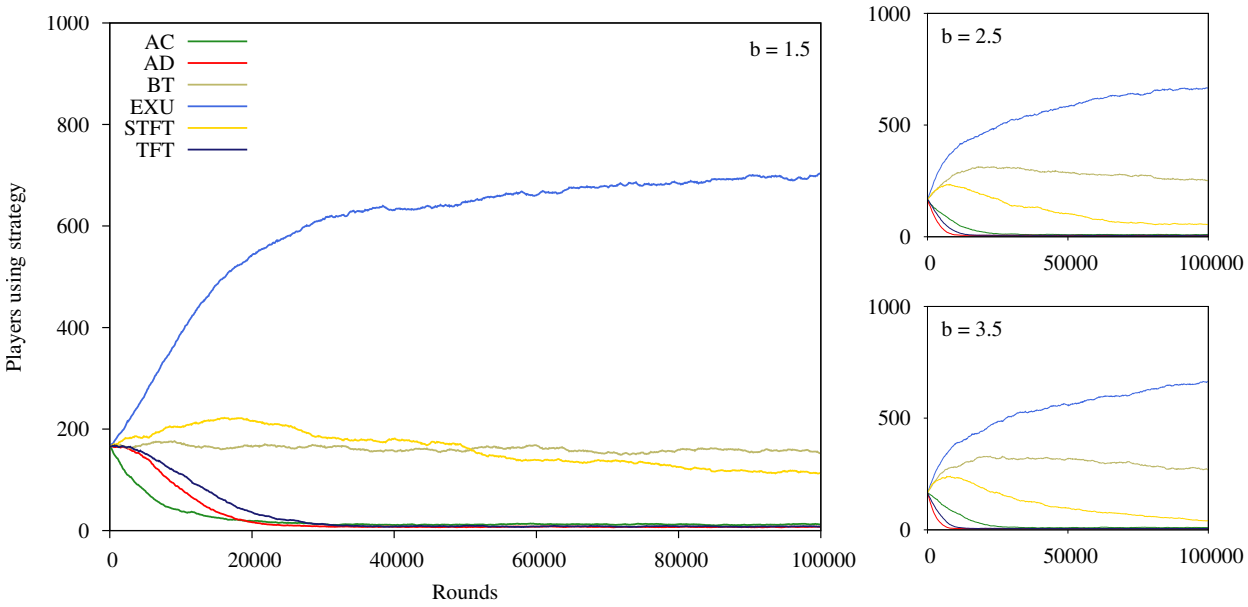


Figure 4.2: Evolution of the population. The number of players using each strategy as a function of the number of rounds, averaged over 50 runs, for $b = 1.5$ (left), $b = 2.5$ (top right) and $b = 3.5$ (bottom right).

4.2.3 Invasion Simulations

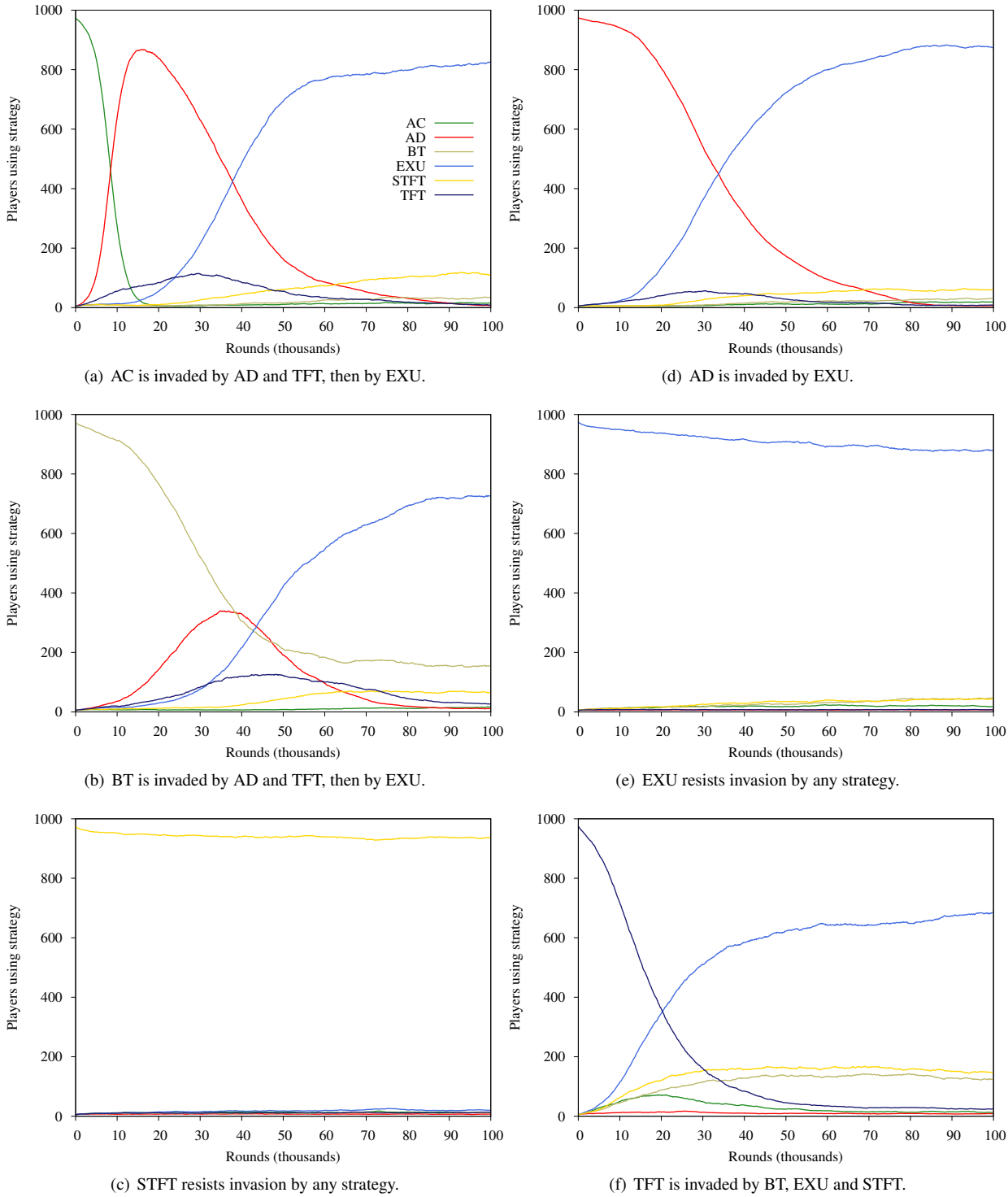
The results so far have shown that the expected utility strategy performs well when played by a substantial fraction of the population, but we would also like to know whether it is suitable for use in populations dominated by other strategies. To find out, we simulate the evolution of six populations, each containing 975 players of one strategy and five players of each of the other strategies. As before, no strategy is allowed to drop below five players. The results are averaged over 50 runs of 100,000 rounds each.

The results for $b = 1.5$ are shown in Figure 4.3. EXU is able to invade any strategy except STFT. This is a strong result – clusters of Tit For Tat players can invade other strategies in networked variants of the prisoner's dilemma [536], but in our simulations there are no clusters: a new player is connected to each existing player with equal probability.

STFT and EXU can each resist invasion by any other strategy. In 50 longer runs of one million rounds each (not shown here), EXU stabilises at 80% of the population when it starts as the dominant strategy, while STFT stabilises at 90% when it is initially dominant. Informally we might say that both strategies appear to be *evolutionarily stable*, at least among the strategies considered here. Formally speaking, however, existing definitions of evolutionary stability are based on the assumption of independent pairwise encounters, such as those that occur in the prisoner's dilemma [548, 552]. A formal analysis of evolutionary stability in the sharer's dilemma will require a new and more general definition of evolutionary stability, a task that we leave for future work.

4.3 Discussion

So far we have only made a preliminary exploration of the sharer's dilemma – many interesting aspects of the game remain to be investigated. For example, we have assumed that all players possess equal resources for cooperation, but the dynamics of cooperation may change when some players have more resources than others. Piatek *et al.* have shown that when high-capacity BitTorrent nodes are able to participate in multiple swarms they benefit from allocating their resources between swarms, to the detriment of the low-capacity nodes in each swarm (see section 2.5.4). The sharer's dilemma gives us a theoretical framework for investigating whether such issues apply to cooperation under scarcity in general. We can simulate differences in capacity by updating the players at different rates, with high-capacity players making their choices more frequently than low-capacity players, allowing them to cooperate more often in a given period of time.



(a) AC is invaded by AD and TFT, then by EXU.

(d) AD is invaded by EXU.

(b) BT is invaded by AD and TFT, then by EXU.

(e) EXU resists invasion by any strategy.

(c) STFT resists invasion by any strategy.

(f) TFT is invaded by BT, EXU and STFT.

Figure 4.3: Invasion simulations. The number of players using each strategy as a function of the number of rounds, averaged over 50 runs, for $b = 1.5$. EXU can invade any strategy except STFT; EXU and STFT both resist invasion.

Similar issues arise when we consider moving beyond the simple random networks simulated here to more realistic models of social or other networks. Like variation in capacity, variation in degree could benefit some nodes relative to others. High-degree players will have more opponents from whom to receive cooperation than low-degree players, but on the other hand they will have to share their own resources among a larger number of potential beneficiaries; it is possible that this might cause difficulties with bootstrapping, since it will take a high-degree player many rounds to cooperate at least once with every potentially cooperative opponent.

Another simplifying assumption we have made concerning network structure is that a new player is connected to each existing player with equal probability. This rules out the formation of clusters, for example, which might help to establish cooperation in otherwise hostile networks; on the other hand, if players are able to choose their neighbours then defectors might be able to exploit the first-time cooperation of reactive strategies by continually connecting to new neighbours. The structure and dynamics of the network are clearly relevant to the outcome of the game, so when using the sharer's dilemma to model any scenario we will need to make sure that we are modelling the network, as well as the individual players, realistically.

In the simulations presented here we have also assumed that all players assign the same subjective cost to cooperating and the same subjective benefit to receiving cooperation. This restriction is not required by the model, and we would like to explore the effect of variation within the population: for example, free riding might be an appealing strategy to players who consider the cost of cooperating to be high, while altruism might appeal to those who consider the cost to be low. However, when players can receive different payoffs from the same outcomes, it becomes difficult to compare the success of different strategies in a meaningful way, and the use of evolutionary simulations becomes problematic; we will need to consider new ways of comparing strategies before we can explore subjective payoffs.

4.4 Extensions to the Model

4.4.1 Incommensurable Costs and Benefits

In section 4.1.5 we assumed that the cost of cooperating and the benefit of receiving cooperation could be expressed in comparable terms. Even in situations where costs and benefits are not commensurable, a player can still use the expected utility strategy to maximise the benefit obtained for a given cost – or, equivalently, to minimise the cost of obtaining a given benefit – by cooperating with whichever opponent will provide the greatest benefit in return for her cooperation.

Whereas in section 4.1.5 we subtracted the cost of cooperating from the expected benefit, in the case of incommensurable costs and benefits we consider the *expected benefit/cost ratio* instead, defined as the total benefit received from an opponent divided by the total cost of cooperating with the opponent:

$$b'_i(t) = \frac{\sum_{j=1}^{t-1} b_{ij}}{\sum_{k=1}^{t-1} c'_{ik}}, \tag{4.4}$$

where $b_{ij} = b$ if the player received cooperation from o_i in round j, otherwise $b_{ij} = 0$, and $c'_{ik} = c$ if the player cooperated with o_i in round k, otherwise $c'_{ik} = 0$. When costs and benefits are incommensurable, the expected utility strategy defines $u(a_i(t)) = b'_i(t)/c$. The value of $u(a_0(t))$ is a parameter of the strategy, because the cost of defecting is zero and therefore the benefit/cost ratio is undefined. Comparing benefit/cost ratios cannot tell a player whether to cooperate or defect; it can only tell a player who chooses to cooperate which opponent to choose.

To express the concept of expected benefit/cost ratio in more general terms, let $b(x_i)$ and $c(x_i)$ denote the benefit and cost, respectively, of outcome x_i of some action x. Following Savage's definition of expected utility, we define the expected benefit/cost ratio of x, $u'(x)$, as the ratio of the sum of the benefits of all possible outcomes, weighted by their probabilities, to the sum of the costs of all possible outcomes, weighted by their probabilities:

$$u'(x) = \frac{\sum_i b(x_i)p(x_i)}{\sum_i c(x_i)p(x_i)}. \tag{4.5}$$

Note that the expected benefit/cost ratio of an action with no possible cost is undefined: when costs and benefits are incommensurable, there is no basis on which to choose between an action with a possible cost and an action with no

possible cost.[1]

4.4.2 Multiple Services

So far we have assumed that there is only one form of cooperation – that is to say, only one service that players can provide to one another – and that any player can in principle provide that service to any neighbour. We have therefore used an undirected network to represent the possible interactions between players. To model more complex situations in which players can provide services of different kinds, we can use networks with directed, labelled edges, such that a player who is capable of providing a service X can only provide it over outgoing edges labelled X, and a player who is capable of receiving a service Y can only receive it over incoming edges labelled Y. The problem that then arises is how to sustain cooperation between neighbours who cannot provide the same service to one another.

The existing work reviewed in section 2.5.4 takes two approaches to this problem. The first is to look for *rings* in which each player provides a service to the next player in the ring and receives a service, possibly of a different kind, from the previous player in the ring. A ring can make use of a directed edge even when there is no edge with any label in the opposite direction, potentially creating an incentive to cooperate with a neighbour who cannot directly reciprocate in any way, as long as a ring can be found that supports indirect reciprocation.

The second approach taken by existing work is *delegation*: a player who is asked to provide a service may either provide it herself or delegate the request to another player, who may provide the service or delegate the request again. A player receiving a request may not know whether her upstream neighbour is the original requester, and likewise a player requesting a service may not know whether her downstream neighbour is the final provider. Message forwarding in multi-hop networks can be seen as a form of delegation in which the service being delegated is the delivery of a message to a particular destination.

We can model the delegation of requests by creating a directed, labelled edge from each player who is capable of requesting a given service to each player from whom she can request it, even if the latter player cannot directly provide the service. If a path exists from a given player to a provider of the service then a request from that player can in principle be delegated to that provider, although in practice the player may not know that the path exists, or may not be able to find it.

Unlike a player in a ring, the final provider of a delegated request does not immediately receive any benefit from providing the requested service: instead, neighbours balance their contributions over time, as in the original, undirected model of reciprocation, although possibly by exchanging services of different kinds. Payment balances and contribution ratios have been proposed to ensure fairness between neighbours providing delegated services (see section 2.5.4), but as we have argued previously, incentive mechanisms must be grounded in the self-interest of players if they are to resist manipulation. Rather than insisting on fairness, it is rational for a player to consider the expected benefit and cost of cooperating with each neighbour, and to allocate her resources accordingly.

In order for a service to support delegation, any player who originates or delegates a request must be able to tell whether the requested service has been performed. Each player can then reciprocate with her immediate neighbours without needing to determine whether they are the original requesters or final providers of the services being exchanged. GNUnet and OneSwarm use cryptographic hashes for this purpose: each request contains the hash of a block of data, so anyone who has seen the request can verify that the correct data has been sent in response. Ostra uses digital signatures: each request identifies the recipient of a message, who sends a signed response to show that the message was wanted. The acknowledgement mechanism presented in the Chapter 5 combines the advantages of both techniques: like a hash, it can be verified without identifying the requester or provider, and like a signature, it can be used in situations where the requester wishes to send data rather than receiving it. For the present discussion, however, we only need to assume that the final provider of a service is able to create some abstract *proof of service* that can be verified by the original requester, as well as by any player who has delegated the request.

[1] For example, imagine that a student is given a choice of two wagers, each based on the toss of a fair coin. In the first wager, if the coin comes up heads then the student will receive a cabbage; if it comes up tails, she will have to spend three years writing a PhD thesis. In the second wager, if the coin comes up heads, the student will receive two cabbages; if tails, she will have to spend eight years writing. Even if the student cannot express the value of her time in terms of cabbages, the expected benefit/cost ratio tells her to prefer the first wager; it cannot, however, tell her whether to decline both wagers.

4.4.3 Indirect Utility

Given a proof of service mechanism, we can extend the sharer's dilemma to encompass delegation. In the original model, a player considers the cost of cooperating and the benefit of the reciprocation she expects to receive in return. In the extended model, a player may have the option of delegating a request rather than fulfilling it herself, in which case she must consider the cost of forwarding the request and the response, rather than the cost of performing the service. The possibility that the delegated request will not be fulfilled must also be taken into account; recall that the concept of expected utility allows a player to assign a probability, as well as a utility, to each possible outcome.

Let x denote the action of delegating a request to a neighbour, where x_1 is the outcome that the request is fulfilled and x_2 the outcome that it is not. Let $p(x_1)$ and $p(x_2)$ denote the subjective probabilities the player assigns to the outcomes, where $p(x_1) + p(x_2) = 1$, since the outcomes are exhaustive. We represent the cost of forwarding the request by c_x, the cost of forwarding the response by c_{x_1}, and the benefit of the reciprocation that is expected if the request is fulfilled by b_{x_1}.

If costs and benefits are commensurable then we can use Equation 4.2 to calculate the expected utility of x, $u(x)$, by subtracting the cost of each outcome from its benefit:

$$u(x) = \sum_i (b(x_i) - c(x_i))p(x_i) = (b_{x_1} - c_x - c_{x_1})p(x_1) - c_x p(x_2) = (b_{x_1} - c_{x_1})p(x_1) - c_x. \tag{4.6}$$

If costs and benefits are not commensurable then we can use Equation 4.5 to calculate the expected benefit/cost ratio of x, $u'(x)$, rather than the expected utility:

$$u'(x) = \frac{\sum_i b(x_i)p(x_i)}{\sum_i c(x_i)p(x_i)} = \frac{b_{x_1}p(x_1)}{(c_x + c_{x_1})p(x_1) + c_x p(x_2)} = \frac{b_{x_1}p(x_1)}{c_{x_1}p(x_1) + c_x}. \tag{4.7}$$

Extending the model in this way raises the interesting possibility of a rational player cooperating with a neighbour in order to receive a service that has no direct benefit to herself, but which enables her to cooperate with another neighbour in order to receive another service, eventually obtaining a service that has a direct benefit. We might say that some services have *direct utility* or *use value*, while others have *indirect utility* or *exchange value*. Although it is not a currency, a proof of service mechanism enables the transition from use value to exchange value.

Under some circumstances a player may need to choose between actions that have direct utility and actions that have indirect utility, and the expected utility strategy can encompass such decisions.

4.5 Conclusion

We have seen that a simple extension to a well-known game can provide a new perspective on the problem of cooperation in networks: incorporating scarcity into the prisoner's dilemma reframes the problem of cooperation as a problem of prioritisation and suggests new strategies based on maximising expected utility. The sharer's dilemma provides a game theoretic model for many situations in nature and society where the benefit of cooperation is higher than the cost, and where resources for cooperation are scarce.

It is easy to see how the strategies described in this chapter could be applied to peer-to-peer file sharing networks such as BitTorrent. Our simulations, although simple, show that the expected utility strategy performs well in a mixed population of other strategies, indicating that it may be possible to deploy it incrementally in existing networks.

We are also interested in the possibility of using the expected utility strategy to create incentives for selfish nodes to forward messages in multi-hop networks. In the next chapter we describe an unforgeable acknowledgement mechanism that can be used to create proofs of service for message forwarding, and in Chapter 6 we develop a routing protocol that uses the unforgeable acknowledgement mechanism and the expected utility strategy to create incentives for selfish nodes to forward messages.

Chapter 5

Unforgeable Acknowledgements

> "Outside our small safe place flies Mystery"
> – A.S. Byatt, *Possession*

This chapter describes a mechanism for proving that a message has been delivered unmodified to its intended destination across an untrusted network, without relying on any identity infrastructure to establish the identities of the nodes or links of the network. We assume that only the *source* and *destination* of each message trust each other, that there is no other trust relationship within the network, and that the source and destination may not be able to see or verify the identities of any other nodes in the network. Nodes that forward a message but are not the source or the destination are termed *relays*. We assume that any node may act as a source, destination or relay.

Unlike existing end-to-end acknowledgement mechanisms that only provide proof of delivery to the source of an acknowledged message, the mechanism described in this chapter provides proof of delivery to the source and to every relay that forwarded the message.

Previous work in this area has relied on establishing a certified identity for each node (see section 2.3.3). However, in many environments it may not be practical to insist on establishing the identity of every member of the network. For example, the membership of the network may be changing constantly, a certificate authority may be unavailable or may not be trusted by all users, or users may wish to remain anonymous. Our mechanism can operate in networks with unknown or varying membership and with no limits on the creation of new identities, and it does not reveal any information to relays about the identities of the endpoints.

The mechanism is based on end-to-end (destination to source) *unforgeable acknowledgements (U-ACKs)* that can be verified by any node that has seen the acknowledged message. Unlike a digital signature scheme, relays do not need to share any keys with the source or destination, or to know their identities. Nodes perform only relatively lightweight operations per message.

In Chapter 6 we use the U-ACK mechanism as a building block in two address-free routing protocols. However, U-ACKs are a general mechanism that could be used in other contexts, in conjunction with a suitable application-specific dependability metric based on the messages sent and the U-ACKs received.

5.1 The Unforgeable Acknowledgement Mechanism

The U-ACK mechanism involves two kinds of data: *messages* and *acknowledgements*. A message is a general-purpose unit of communication consisting of a header and a data payload. Depending on the application and the layer of operation in the communication stack, a message could be a frame, a packet, or an application data unit such as a block in a file transfer. Each acknowledgement confirms delivery of a single message.

The source and destination of each message must share a secret key that is not revealed to any other node, and each message sent between the same endpoints must contain a unique serial number or nonce to prevent replay attacks. This number need not be visible to intermediate nodes, and indeed the U-ACK mechanism does not reveal any information

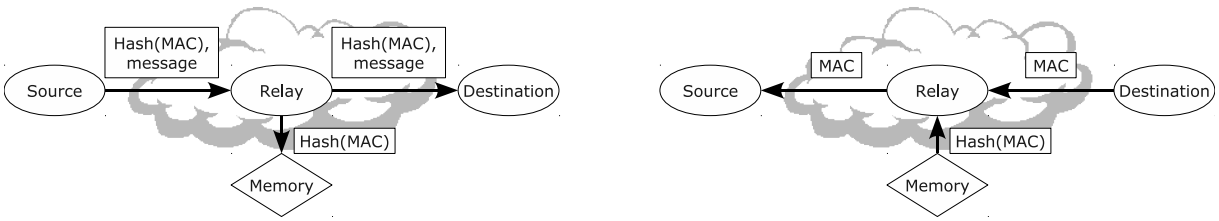


Figure 5.1: The U-ACK mechanism: relays use the hash attached to the message to verify the acknowledgement.

that can be used to determine whether two messages have the same source or destination, although such information might be revealed by traffic analysis or by other protocol layers.

We assume that the source and destination have some way of establishing a shared secret key before they begin communicating. The U-ACK mechanism does not rely upon or mandate any particular key management scheme or key exchange mechanism; any existing scheme appropriate to the application can be used.

5.1.1 Description of the Mechanism

Unforgeable acknowledgements (U-ACKs) make use of two standard cryptographic primitives: *message authentication codes (MACs)* and *collision-resistant hashing*. Before transmitting a message, the source computes a MAC over the message using a secret key, k, shared with the destination. Instead of attaching the MAC to the message, the source attaches the *hash of the MAC* to the message. Relays store a copy of the hash when they forward the message. If the message reaches its destination, the destination computes a MAC over the received message using the secret key, k, shared with the source. If the hash of this MAC matches the hash received with the message then the message is valid, and the destination *sends the MAC as an acknowledgement*, which is forwarded back along the path taken by the message. Relays can verify that the acknowledgement hashes to the same value that was attached to the message, but they cannot forge acknowledgements for undelivered messages: they lack the secret key, k, to compute the correct MAC from the message, and they cannot invert the hash function to derive the correct MAC from the hash.

More formally, let $H(x)$ denote the hash of x, let $MAC(y, z)$ denote a message authentication code computed over the message z using the key y, and let $\{a, b\}$ denote the concatenation of a and b. Let k be the secret key shared by the source and destination, and let d be the data to be sent. The relays between the source and destination are denoted $r_1 \ldots r_M$. The U-ACK mechanism works as follows:

1. The source first attaches a unique nonce or serial number, s, to the data, to produce the payload $p_1 = \{s, d\}$.
2. The source calculates $h_1 = H(MAC(k, p_1))$. The source stores h_1 and sends $\{h_1, p_1\}$ to relay r_1.
3. Each relay r_i stores an identifier (such as the network address) of the previous node under the hash h_i, and forwards $\{h_{i+1}, p_{i+1}\}$ to the next node, where $h_{i+1} = h_i$ unless r_i modifies the header, and $p_{i+1} = p_i$ unless r_i modifies the payload.
4. On receiving $\{h_{M+1}, p_{M+1}\}$ from r_M, the destination calculates $H(MAC(k, p_{M+1}))$ and compares the result to h_{M+1}. If the result does not match, then either $h_{M+1} \neq h_1$ or $p_{M+1} \neq p_1$ – in other words either the header or the payload has been modified by one of the relays – and the destination does not acknowledge the message.
5. If the message has not been modified, the destination returns the acknowledgement $a_{M+1} = MAC(k, p_{M+1})$ to relay r_M.
6. Each relay r_i calculates $H(a_{i+1})$. If the result matches a stored hash then r_i forwards a_i to the previous node stored under the hash, where $a_i = a_{i+1}$ unless r_i modifies the acknowledgement; r_i then discards the hash.
7. On receiving a_1 from r_1, the source checks that $H(a_1) = h_1$. If so, the message has been acknowledged.

When a relay receives an acknowledgement whose hash matches the stored hash corresponding to a message it previously forwarded, it knows that the message was correctly delivered, and that neither the header, the payload, nor the acknowledgement was modified by any node between itself and the destination.

Likewise, when the source receives an acknowledgement whose hash matches the stored hash corresponding to a message it previously generated, it knows that the message was correctly delivered to the destination, since only the destination could have generated the acknowledgement.

5.2 Security

In this section we examine the security properties of the U-ACK mechanism.

5.2.1 Unforgeability

We begin by showing that relays cannot forge acknowledgements as long as the underlying cryptographic primitives are secure. Four specific properties of the underlying primitives are listed below. These properties are commonly accepted, and are based on the design goals of those primitives:

1. It is not feasible to recover the secret key k by observing any sequence of authenticated messages, $\{MAC(k, m_1), m_1\} \ldots \{MAC(k, m_n), m_n\}$.
2. It is not feasible to calculate $MAC(k, m)$ for a given message m without knowing the secret key k.
3. It is not feasible to find the preimage x of a given hash $H(x)$.
4. It is not feasible to find a second preimage $y \neq x$ for a given preimage x, such that $H(y) = H(x)$.

The first two properties are standard requirements and design goals for MAC functions, while the last two properties (inversion resistance and second preimage resistance) are standard requirements and design goals for cryptographic hash functions. These properties are not affected by recent collision search attacks on cryptographic hash functions [553]. As long as these properties are true for any specific MAC and hash function used to implement the U-ACK mechanism, we argue that U-ACKs are unforgeable.

First we show that the U-ACK mechanism does not reveal the secret key, k. If an eavesdropper could recover the secret key from some sequence of messages, $\{H(MAC(k, m_1)), m_1\} \ldots \{H(MAC(k, m_n)), m_n\}$, and their acknowledgements, $MAC(k, m_1) \ldots MAC(k, m_n)$, then the attacker could also recover the key from $\{MAC(k, m_1), m_1\} \ldots \{MAC(k, m_n), m_n\}$ by hashing the MACs, contradicting the first property above.

Next we show that an attacker cannot forge acknowledgements without the secret key. Assume that an attacker succeeds in forging an acknowledgement. Either the forged acknowledgement is identical to the genuine acknowledgement or it is different. If it is identical then either the attacker has succeeded in calculating $MAC(k, m)$ from m without knowing k, which contradicts the second property above, or the attacker has inverted $H(MAC(k, m))$ to obtain $MAC(k, m)$, which contradicts the third property. On the other hand if the forged acknowledgement is different from the genuine acknowledgement, the attacker has found a second preimage $y \neq x$ such that $H(y) = H(x)$, which contradicts the fourth property.

5.2.2 Faulty Nodes

Although they cannot forge acknowledgements, faulty nodes can interfere with the operation of the U-ACK mechanism in other ways. Here we consider the most general class of faults: *Byzantine faults*, in which the faulty nodes may exhibit arbitrarily complex behaviour, including collusion.

A faulty *source* may replay a message that it previously generated, or may generate a message that cannot be acknowledged by any destination.

A faulty *destination* may replay an acknowledgement; may fail to acknowledge a message; may create an invalid acknowledgement for a message for which it is, or is not, the intended destination; may send an acknowledgement to a neighbour that did not send it the corresponding message; or may, in collusion with another faulty node, create a valid acknowledgement for a message for which it is not the destination.

A faulty *relay* may replay a message or acknowledgement; may fail to forward a message or acknowledgement; may modify a message or acknowledgement before forwarding it; may forward an acknowledgement to a neighbour that did not send it the corresponding message; or may, in collusion with another faulty node, forward a valid acknowledgement for a message that it did not forward.

The U-ACK mechanism does not attempt to prevent or even distinguish between the faults listed above; nor does it attempt to identify faulty or non-faulty nodes – it only attempts to provide non-faulty nodes with proof that specific messages have been delivered.

While messages may in principle carry source or destination addresses, the U-ACK mechanism does not authenticate such addresses. If source and destination addresses are used, faulty nodes upstream and downstream from a given relay may collude to generate and acknowledge messages with spoofed addresses.

Similarly, faulty nodes upstream and downstream from a given relay might collude to replay messages and acknowledgements generated by themselves or by other (faulty or non-faulty) nodes. The U-ACK mechanism does not prevent such behaviour: if the messages that a non-faulty node receives from a neighbour are not unique, then the acknowledgements returned to that neighbour are not guaranteed to be unique either.

As noted above, a faulty relay may modify the header of a message before forwarding it. However, since the destination will not acknowledge the modified message, the faulty relay will be unable to provide a suitable acknowledgement to its upstream neighbour, and thus from its neighbour's point of view the faulty relay will effectively have dropped the message and transmitted one of its own instead, albeit one with an identical payload. The faulty relay's upstream neighbour will not consider it to have delivered the message.

Likewise, if a faulty relay modifies the payload instead of the header, the destination will not acknowledge the message and again the faulty relay will be unable to provide an acknowledgement to its upstream neighbour. Thus with regard to proof of delivery, any modification to a message or acknowledgement is equivalent to dropping the message. This is in line with the strategy of failure mode reduction described in section 3.4.

5.3 Applicability

The U-ACK mechanism is not a complete routing protocol, but rather a building block that provides proof of delivery. The method by which sources and destinations establish shared secret keys is not specified here, because the acknowledgement mechanism is independent of the key exchange mechanism; similarly, end-to-end encryption is not discussed, although we would expect it to be used by parties requiring confidentiality and unlinkability.

In principle, unforgeable acknowledgements can operate at the network layer instead of the application layer. There are no dependencies between messages, only between a message and its acknowledgement, so each message can be treated as an independent datagram; retransmission, sequencing and flow control can be handled by higher protocol layers.

5.3.1 Reverse Path Forwarding

We have assumed that the forward path of the message is the same path that will be followed, in reverse, by the U-ACK. This may not be possible in all networks – for example some wireless networks may contain unidirectional links. Where the assumption of reverse path forwarding does not hold, acknowledgements may still be delivered by a different path. In that case there may be some relays that receive proof of delivery, while others do not, at least not for all messages. This information may be sufficient for some applications.

Note that routing asymmetries such as those commonly found in the Internet do not prevent reverse path forwarding at the application layer: each relay stores the identity of the previous node when forwarding a message, so provided bidirectional communication is possible between each pair of nodes, the acknowledgement can always pass through the same nodes as the message.

Asymmetric link bandwidth does not prevent reverse path forwarding as long as it is possible to return one acknowledgement for each message sent in the opposite direction.

5.3.2 Gateways, Proxies and Middleboxes

The U-ACK mechanism does not require the source or destination to share keys with the relays, but it can easily be generalised to situations where the source wishes to direct traffic through a certain trusted gateway, proxy, or other middlebox. To set up forwarding through a trusted gateway, the source must exchange keys with the gateway and the gateway must exchange keys with the destination. The gateway then acknowledges messages from the source and forwards them to the destination with new headers, while the destination acknowledges messages from the gateway. The key shared by the source and the gateway is independent from the key shared by the gateway and the destination, so it is possible for the gateway to re-encrypt the messages before forwarding them. Indeed, onion encryption could be combined with the U-ACK mechanism to provide anonymity for the source as well as proof of delivery. If the source trusts the gateway to deliver messages then the source and destination can remain mutually anonymous.

5.3.3 Non-Unicast Communication

So far we have only considered unicast communication, but there may be further considerations if non-unicast mechanisms are used for message delivery. For example, in protocols that use flooding or gossiping, multiple copies of a message may reach a destination or relay node by different paths. The source only requires one acknowledgement for each message, so one possibility would be to acknowledge the first copy of each message and discard duplicates (the protocols described in Chapter 6 follow this approach). Another possibility would be to maintain a one-to-one association between messages and U-ACKs by returning a copy of the acknowledgement to every neighbour from which a copy of the message was received.

Another issue to consider is one-to-many or many-to-many communication, such as network-layer or application-layer multicast. Here, a single message may have many destinations, and a naive application of our mechanism would require each of those destinations to send an acknowledgement. Reliable multicast is an area of ongoing research [554, 555, 556], but it is known to be impractical to use per-destination acknowledgements for large groups.

In a tree-based multicast scheme, one possibility would be for certain nodes in the tree to act as trusted gateways, as described in the previous section. Each gateway would be responsible for receiving U-ACKs from nodes below itself in the multicast tree and sending aggregate U-ACKs to a node above itself in the tree. An aggregate U-ACK would indicate delivery of a message to all intended recipients.

We consider the use of U-ACKs in non-unicast communication to be a topic for further study.

5.4 Engineering Considerations

5.4.1 Timeouts

Relays cannot store hashes indefinitely while waiting for acknowledgements – at some point, unacknowledged hashes must be discarded. A relay that receives an acknowledgement after discarding the corresponding hash cannot verify or forward the acknowledgement, so there is no reason for a relay to store a hash for longer than its upstream or downstream neighbours. The most efficient solution would be for all relays along the path of a given message to discard the associated hash at the same time, after waiting an appropriate amount of time to receive a U-ACK. However, using a separate synchronisation protocol to determine when to discard hashes could be very complicated in an untrusted scenario, and adding a time-to-live field to messages would undermine source-destination unlinkability by allowing relays to estimate the distance to the source.

Fixed timeouts avoid these problems while ensuring that adjacent relays discard each hash at approximately the same time. The length of the timeout creates a tradeoff between the maximum end-to-end latency the network can tolerate and the number of outstanding hashes each relay must store, so the choice of an appropriate timeout will depend on the application. The simulations in Chapter 6 use a timeout of five seconds for peer-to-peer overlays of up to 2,000 nodes.

5.4.2 Overhead

The computation and bandwidth overheads of the U-ACK mechanism are modest, particularly when compared to the normal cost of using MACs for end-to-end authentication. The source and destination must each perform one hash computation per message in addition to the normal cost of using MACs, and each relay must perform a single hash computation and table lookup per acknowledgement. Every message requires a separate acknowledgement, but since acknowledgements are small and there is at most one per message they could be piggybacked on messages travelling in the other direction to reduce transmission costs.

The storage overhead, on the other hand, may be considerable. The source and each relay must store one hash per outstanding message, so the storage overhead depends on four factors: the bandwidth of the node's outgoing link, b; the timeout for stored hashes, t; the hash size, h; and the message size, m. We can approximate each node's storage requirement, s, as:

$$s = \frac{b.t.h}{m}$$

So, with a 60 second timeout and a minimum message size of 100 bytes including headers, a typical peer-to-peer node with a 64 kB/s outgoing link may need to store up to 38,400 outstanding hashes. If each record occupies 50 bytes, that would require nearly 2 MB of memory. This represents the worst case, however, when all messages have the minimum size and all acknowledgements time out; in a more realistic scenario with a mean message size of 500 bytes and an average round-trip time of five seconds, the storage overhead would be just 32 kB. Nevertheless it is clear that this overhead could make the U-ACK mechanism unsuitable for use on resource-constrained devices such as wireless sensors.

An attacker might attempt to exhaust a relay's memory by flooding it with messages, forcing it to store a large number of hashes. This attack could be mitigated by allocating a separate storage quota to each neighbour; a neighbour that exceeded its quota would then simply cause its own hashes to expire early.

In some applications it may be acceptable to reduce the bandwidth and storage overheads of the mechanism by truncating the output of the hash function – this is often done with the hash-based message authentication code HMAC [557], for example. Truncation will weaken the inversion resistance and second preimage resistance properties of the hash function, but these properties only need to be strong enough to resist an attack during the short time before the hash expires.

5.5 Experimental Evaluation

To evaluate the cost of the U-ACK mechanism in a realistic setting, we created a simple, unoptimised implementation in Java, using HMAC-MD5 as the MAC algorithm and SHA-1 as the hash algorithm. HMAC-MD5 has a 128-bit output and SHA-1 has a 160-bit output, so the bandwidth overhead is 20 bytes per message for the hash and 16 bytes per acknowledgement for the preimage.

We also created two alternative implementations against which to compare the U-ACK mechanism: a *plain* implementation with no acknowledgement mechanism, and a *sequence number* implementation in which each message carries a unique four-byte sequence number and is acknowledged by echoing the sequence number in a four-byte selective acknowledgement. These two alternatives allow us to isolate the overhead of using acknowledgements from the overhead of the U-ACK mechanism specifically. Apart from the generation and handling of acknowledgements, all of the implementations use the same code. Each message and acknowledgement is sent as a separate UDP datagram. We do not include the overhead of UDP or IP headers in our results.

To simulate packet loss, the receiver side of each implementation randomly drops a configurable fraction of messages without acknowledging them; we call this fraction the loss rate. Unacknowledged messages are not retransmitted. For the sequence number and U-ACK implementations, a cleanup task runs every five seconds on the sender side to discard state associated with messages that have not been acknowledged after ten seconds.

In our experiments, each implementation was used to transfer 1 MB of random data at an average rate of 50 kB/s across a wide-area link with a round-trip time of 14 ms (standard deviation 0.4 ms). The loss rate and payload size were varied to investigate their impact on the bandwidth, memory and processing costs of the implementations. For each condition,

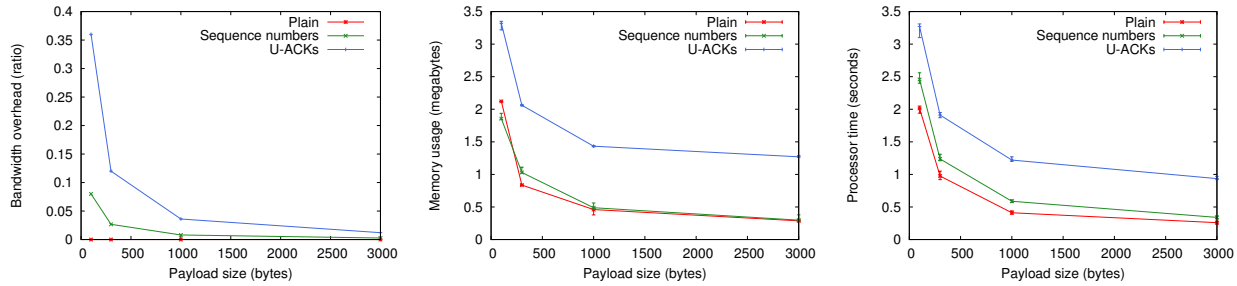


Figure 5.2: The effect of payload size. Bandwidth overhead (left), memory usage (centre), and processor time (right), as functions of the data payload per message. Smaller payloads increase the number of messages that must be transmitted and therefore increase the overhead of using acknowledgements.

the results were averaged over ten runs. Memory usage was measured by the Java virtual machine every ten seconds after invoking the garbage collector; for each run we recorded the maximum memory usage measurement taken during the run. Processing time was measured by the operating system, including time spent executing system calls.

The experiments were conducted on a 32-bit Acer AS1410 notebook running Linux 2.6.31 and Sun Java 1.6.0_20. This is a low-end consumer device, but the experiments show that it is easily capable of supporting the U-ACK mechanism.

5.5.1 Results

Figure 5.2 shows the effect of payload size on the bandwidth, memory and processing costs of the three implementations. The error bars indicate the maximum and minimum values obtained in any of the ten runs. For this experiment, the loss rate was set to zero and the payload size was varied from 100 to 3,000 bytes per message. The sequence number implementation adds four bytes to each message, while the U-ACK implementation adds 20. Acknowledgements use another four bytes for the sequence number implementation and 16 for the U-ACK implementation.

The left frame of Figure 5.2 shows the bandwidth overhead of each implementation, defined as the ratio of non-data bytes sent and received to data bytes sent. The ratio is always zero for the plain implementation, since it only sends data. For small message sizes, the overhead of using acknowledgements is high: when the payload size is 100 bytes, the U-ACK mechanism uses an additional 0.36 bytes for every byte of data (36% overhead). As the payload size increases, however, the overhead drops, reaching 0.012 bytes per byte of data (1.2% overhead) at a payload size of 3,000 bytes. The overhead of sequence numbers is smaller by a constant factor, but scales in the same way.

The central frame of Figure 5.2 shows the memory usage of the three implementations. Smaller payload sizes cause higher memory usage for all of the implementations, but the U-ACK implementation uses consistently more memory than the others at all payload sizes. The difference is approximately constant and appears to be caused by loading cryptographic libraries that are not required by the other implementations. While this overhead is not specific to the U-ACK mechanism, it should be borne in mind as a cost of cryptographic solutions in general.

The processor time used by each implementation is shown in the right frame of Figure 5.2. Again, smaller payloads lead to higher costs for all of the implementations, but the U-ACK implementation has the highest cost, using 1.6 times as much computation as the plain implementation at a payload size of 100 bytes and 3.6 times as much computation at a payload size of 3,000 bytes. While the U-ACK mechanism is expensive in comparison to the plain and sequence number implementations, it is lightweight in comparison to other cryptographic proof of delivery mechanisms. According to the OpenSSL benchmarking tool, the test machine takes 6.7 μs to calculate the MD5 digest of 1024 bytes of data followed by the SHA-1 digest of 16 bytes of data, which gives us a rough idea of the cost of creating a U-ACK for a 1 kB message using a fully optimised implementation. In contrast, the OpenSSL benchmark's 1024-bit DSA signature operation takes 1.7 ms on the same machine, while the 1024-bit RSA signature operation takes 3.7 ms – 549 times as long as creating a U-ACK.

We would expect packet loss to increase the memory usage of the sequence number and U-ACK implementations, since they keep state for each unacknowledged message for at least ten seconds, but surprisingly the data show that a loss rate of up to 10% does not cause a significant change in the memory usage of any of the implementations ($p > 0.01$ using the two-tailed Mann-Whitney U test). This suggests that either the cost of keeping state for each message is not

significant in comparison to the fixed costs of the implementation, or the virtual machine is not immediately reclaiming the memory associated with acknowledged messages.

5.6 Related Work

The cryptographic primitives used in the U-ACK mechanism – message authentication codes and one-way hash functions – have previously been used in a variety of ways for authenticated communication and proof of delivery.

5.6.1 Authenticated Acknowledgements

IPSec [558] uses message authentication codes for end-to-end authentication at the network layer, which makes it possible to authenticate transport-layer acknowledgements as well as data. Unlike U-ACKs, however, the MACs used by IPSec can only verified by the endpoints, not by third parties such as relays.

TLS [526] uses MACs at the transport layer. TCP headers are not covered by the MAC calculation, however, so it is possible for relays to forge TCP acknowledgements for TLS connections. As with IPSec, the MACs used by TLS cannot be verified by relays.

In the 2HARP routing protocol [559], each relay or destination node that receives a packet sends a signed acknowledgement to the previous two nodes on the path, allowing each node to verify that its downstream neighbour forwarded the packet. It is assumed that each node has only one identity, so nodes cannot create acknowledgements on their own behalf. As discussed above, the computational cost of using signatures is much higher than that of using U-ACKs, although keys of less than 1024 bits might be sufficient for an application of this kind, which would reduce the cost of creating and verifying signatures.

Some of the robust routing protocols discussed in section 2.3.3.5 use MACs to acknowledge messages and to detect faulty links and nodes. Fault detection requires some way to limit the creation of identities, such as issuing a certified identity to each user, to prevent attackers from creating arbitrary numbers of node and link identifiers. The use of MACs for fault detection also rules out source-destination unlinkability, since the endpoints must identify themselves to the relays in order to establish MAC keys with them.

5.6.2 Authentication Using One-Way Functions

Gennaro and Rohatgi [560] describe two methods for authenticating streams using one-way functions. The first scheme uses one-time signatures [279, 561]: each block of the stream contains a one-time public key, and the corresponding private key is used to sign the next block. The first block carries a conventional asymmetric signature. One-time signatures are large, so this scheme has a considerable bandwidth overhead. The computational cost of verifying a one-time signature is comparable to that of an asymmetric signature, but signing is more efficient.

The second scheme uses chained hashes: each block contains the hash of the next block and the first block carries an asymmetric signature. In this scheme, the entire stream must be known to the source before the first block is sent.

The Guy Fawkes protocol [562] also uses chained hashes. The source does not need to know the entire stream in advance, but each block must be known before the previous block is sent. Each block carries a preimage and a hash that are used to verify the previous block, and a hash that commits to the contents of the next block. The first block carries a conventional signature. To prevent forgery, each block must be received before the next block is sent.

Several of the *ad hoc* routing protocols described in section 2.3.3.6 use hash chains to reduce the number of asymmetric signature operations. Others use delayed disclosure, in which a hash and its preimage are sent by the same party at different times, requiring loose clock synchronisation. In the U-ACK mechanism the preimage is not sent until the hash is received, so no clock synchronisation is required.

The signature schemes described above use similar cryptographic techniques to the U-ACK mechanism, but their aims are different. Whereas the aim of a signature scheme is to associate messages with a source, the aim of the U-ACK mechanism is to associate an acknowledgement with a message *without identifying the source or destination of the message*. The signature schemes mentioned above therefore require an initial asymmetric signature to identify the source, whereas the U-ACK mechanism does not require asymmetric cryptography.

GNUnet [443] uses a private search protocol in which the requester searches for a keyword that has been hashed multiple times; the responder returns the preimage of the requester's query, proving that the responder knows the requested keyword. The protocol is vulnerable to dictionary and replay attacks, since a given keyword will always produce the same request and response. The U-ACK protocol, in contrast, uses each hash and preimage only once, and does not generate them from guessable keywords.

5.7 Discussion

5.7.1 The Puzzle-Solution Pattern

While we have described the U-ACK mechanism in terms of a particular implementation based on message authentication codes and one-way hash functions, the principle underlying the mechanism is more general: it consists of associating each message with a *puzzle* that only the intended destination can solve, but for which any trusted or untrusted party can recognise the correct solution.

Using a MAC algorithm to generate puzzles has the advantage that the destination can recognise modified messages without an additional authentication mechanism, but if another end-to-end authentication mechanism is already in use then we may not need this feature, and other implementations of the puzzle-solution pattern can be considered. We have used cryptographic hash functions for their one-way properties, but there are many other functions that are hard to calculate by brute force but easy to verify, which might serve in place of hash functions in the puzzle-solution pattern.

For example, the pattern can be implemented using a block cipher: each puzzle is a ciphertext produced by encrypting some well-known plaintext with a single-use key, and the solution to the puzzle is the single-use key. Given the ciphertext and the well-known plaintext, it is a standard design requirement of block ciphers that nobody can determine the key, but once the key is revealed, anyone can verify that it encrypts the well-known plaintext to produce the previously seen ciphertext. We do not consider this implementation of the pattern in any further detail here, but merely offer it as an example.

5.7.2 Key Management

The U-ACK mechanism requires the source and destination of each message to establish a shared secret key before communicating. Although the manner in which this is done does not affect the design of the mechanism itself, it raises some issues that must be addressed by any complete system that uses U-ACKs. These issues are key negotiation, key compromise, and key replacement.

Key Negotiation

In general, key negotiation may be conducted in-band or out-of-band – that is to say, using the same channel over which the key will be used or some other channel. If the channel used for key negotiation is susceptible to monitoring by an adversary then the negotiation protocol must not send the key in the clear, and if the adversary can tamper with the channel then the negotiation protocol must also detect man-in-the-middle attacks.

The simplest form of out-of-band key negotiation involves a face-to-face meeting between the communicating parties, but that may not be practical for all applications, or for all users. A more commonly used approach is to exchange public keys in-band, verify them out-of-band, then use the public keys to authenticate an in-band key negotiation protocol that establishes a secret key. If a public key infrastructure is available then a certificate verification step may replace the out-of-band key verification step. This is the approach used by TLS with X.509 certificates, for example [526].

The secure shell protocol [527] makes out-of-band key verification optional, but caches public keys so that man-in-the-middle attacks against subsequent connections can be detected. PGP uses a *web of trust* [161] for key verification: users sign each other's public keys, ideally after verifying them face-to-face, which makes it possible to obtain a key from an untrusted source and verify it with a degree of confidence that depends on the verifier's trust in the users who have signed the key. ZRTP [563] verifies keys in-band by relying on the communicating parties' ability to recognise each other's voices.

Any of these approaches would be suitable for use with the U-ACK mechanism, depending on the application in which the mechanism is used. Once the source and destination have verified each other's public keys, a standard authenticated key negotiation protocol such as Station-to-Station [524] can be used in-band to establish a shared secret key.

We briefly sketch a procedure for augmenting a key negotiation protocol with U-ACKs, if the application requires key negotiation messages to be acknowledged:

1. The source of each key negotiation message creates a unique *temporary key* for use with that message.
2. The source encrypts the temporary key with the destination's public key and attaches the encrypted temporary key to the message.
3. The source uses the temporary key to compute a MAC over the message, including the encrypted temporary key, and attaches the hash of the MAC to the message.
4. On receiving the message, the destination decrypts the encrypted temporary key and computes a MAC in the same way.
5. If the hash of the resulting MAC matches the hash attached to the message, the destination sends the MAC as an acknowledgement.

This procedure is much more computationally expensive for the endpoints than the ordinary use of U-ACKs, since it involves asymmetric encryption, but we emphasise that it is only necessary the first time two users communicate, and only if the application requires key negotiation messages to be acknowledged. The procedure involves no greater overhead for relays than the ordinary use of U-ACKs, and does not provide any way for relays to determine that the messages being exchanged are key negotiation messages.

Key Compromise

If the shared secret key used in the U-ACK mechanism becomes known to any party other than the source and destination, that party will be able to carry out two attacks. First, the attacker will be able to forge U-ACKs, causing the source and relays to believe that the destination is receiving messages when it is not. Second, the attacker will be able to forge MACs, causing the destination to believe that messages created or modified by the attacker were created by the source. It is therefore vital to protect the shared secret key.

We argued in section 5.2.1 that the U-ACK protocol does not leak any more information about the shared secret key than MACs used in the standard way, but even so, it is generally considered good practice to replace keys periodically to protect against possible information leaks in cryptographic primitives and to limit the damage caused by exposed keys. We return to the issue of key replacement below.

Another way for an attacker to obtain the key would be to compromise one of the endpoints' computers. Protecting against threats of this kind is a task that goes far beyond the scope of this thesis, and includes testing and examining software to detect security flaws, keeping third-party software up-to-date with the latest security patches, and controlling physical access to the devices on which keys are stored.

Key Replacement

To prevent replay attacks, no two U-ACKs should be generated for identical payloads using identical keys. The description of the U-ACK mechanism given in section 5.1 specifies that a unique nonce or serial number should be attached to the data to ensure that each payload is unique; if the range of the serial number is not sufficient to prevent it from wrapping during the lifetime of the relationship between the communicating parties then the key must be replaced before the serial number wraps. This is a standard requirement for encryption and authentication protocols, and the standard solution is for both endpoints to use the shared secret key to generate matching sequences of keys, each of which is used until the serial number is about to wrap, at which point the key is replaced with the next key in the sequence [525]. Replacing keys periodically can also help to limit the damage caused by information leaks and exposed keys, as discussed above.

A related issue that requires consideration is how to manage the nonces or serial numbers themselves. Unlike keys, serial numbers do not need to be agreed between sources and destinations, but each source must ensure that it does not reuse the same combination of key and serial number. If a node crashes then it may lose track of which keys and serial numbers it has already used. In such a scenario it may be necessary to repeat the key negotiation procedure to obtain a new shared secret key, from which a new sequence of keys can be derived as described above. Any sequence number can then be safely used with the keys from the new sequence.

5.8 Conclusion

We have described the U-ACK mechanism, a lightweight technique for proving that a message has been delivered unmodified to its intended destination across an untrusted network. The acknowledgements created by the mechanism are unforgeable but can be verified by untrusted third parties. The only nodes that need to be able to verify one another's identities are the source and destination of each message.

The mechanism has broad applicability: it can operate at the network layer or in a peer-to-peer overlay, and does not require relays to establish a security association with the endpoints, or to be aware of the details of higher-layer protocols. U-ACKs can be seen as an instance of a more general puzzle-solution pattern for proving that a message has been delivered to its intended destination.

In the next chapter we will use the U-ACK mechanism as a building block in two address-free routing protocols designed for robust, unlinkable communication across untrusted networks.

Chapter 6

Routing Protocols for Untrusted Networks

> "Wanderer, your footsteps are the road, and nothing more;
> Wanderer, there is no road, the road is made by walking."
> – Antonio Machado, *Proverbs and Songs*

This chapter describes two routing protocols, Darknet and Exu, that use the U-ACK mechanism described in Chapter 5 to discover reliable paths across untrusted networks. Nodes in the Darknet protocol attempt to optimise the probability that the messages they transmit will be delivered, regardless of whether those messages were generated locally or are being forwarded on behalf of other nodes, while nodes in the Exu protocol attempt to optimise the benefit they obtain from the messages they transmit, by prioritising their own messages and those of cooperative neighbours.

Both protocols are examples of *address-free routing*, an approach to communication across multi-hop networks that does not require identifiers with global scope or explicit knowledge of the network topology. The protocols combine path discovery with data delivery, so that every message produces feedback about the reliability of the path it follows.

We use extensive simulations to compare the Darknet and Exu protocols to two existing methods of address-free routing: on-demand routing and flooding. Although the Darknet protocol does not perform as well as on-demand routing under ideal conditions, we find that it is robust to internal attacks by misbehaving nodes, while the Exu protocol creates strong incentives for selfish users to cooperate in message forwarding.

6.1 Assumptions and Terminology

The scenario considered in this chapter is similar to that considered in Chapter 5: an untrusted network with no central administration or admission control, where nodes cannot establish the identities of any nodes other than their immediate neighbours, and do not trust any nodes other than those with which they wish to communicate. Each communication exchange involves a series of *messages* sent from a *source* to a *destination* by relying on the forwarding behaviour of intermediate *relays*, and a series of corresponding *acknowledgements* that follow the reverse paths of the acknowledged messages. Any node may act as a source, destination or relay.

As in the previous chapter, each source and destination are assumed to have some way to establish a shared secret key before communicating; the discussion of key negotiation, key compromise and key replacement from the previous chapter also applies here.

We assume that nodes and links are *unreliable*, and that some participants in the routing protocol may be *selfish*, meaning that they would prefer to avoid the cost of forwarding messages, while others may be *malicious*, meaning that they may try to disrupt communication. Other participants may wish to monitor the traffic passing across the network. The goal of the protocols presented in this chapter is to deliver messages from sources to destinations in the presence of unreliable components and selfish or malicious users, while maintaining *unlinkability* between sources and destinations.

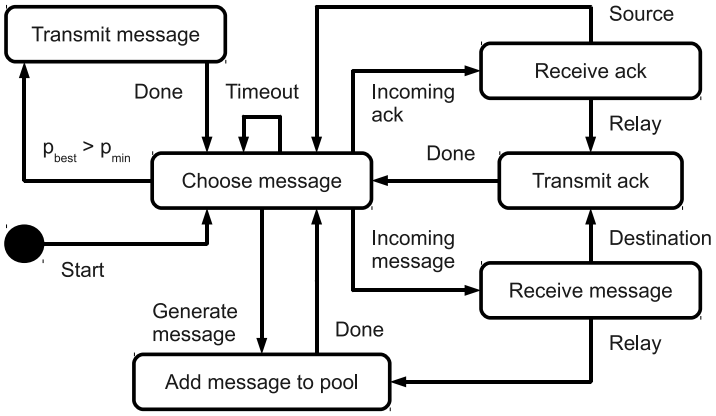


Figure 6.1: UML state machine for the Darknet protocol.

6.2 The Darknet Protocol

The Darknet protocol (Distributed Adaptive Routing without Knowledge of the Network) uses the U-ACK mechanism described in the previous chapter to discover reliable paths between sources and destinations. In keeping with the design strategy described in section 3.4, the protocol does not attempt to distinguish between accidental, selfish and malicious failures; nor does it attempt to identify faulty components. Instead, it routes around faults by preferentially selecting paths that reliably deliver messages end-to-end. By using the feedback provided by U-ACKs, each node attempts to learn which messages should be transmitted to which neighbours in order to maximise the probability of delivery, without explicit knowledge of the paths by which messages are being delivered.

At a high level, the Darknet protocol is similar to the probabilistic routing algorithms surveyed in section 2.3.3.2. The source of each message estimates the probability that the message will be acknowledged if it is transmitted to each neighbour, and accordingly transmits the message to zero or more neighbours. Each of those neighbours makes its own estimates and transmits the message to zero or more of its own neighbours, and so on. If the message is successfully delivered, the first copy to reach the destination is acknowledged. The acknowledgement follows the reverse path of the acknowledged copy to the source, updating the reliability estimators of the nodes along the path. Timeouts are used to update the reliability estimators corresponding to unacknowledged copies and undelivered messages.

The result of this feedback process is that when no path for a message is known, the protocol may send redundant messages along many paths, but when a reliable path is discovered, it is chosen in preference to alternative paths for as long as it remains reliable.

Every message in the Darknet protocol is used for reliability measurement; there are no dedicated route discovery or probe messages, so faulty nodes cannot selectively drop certain messages without affecting the measured reliability.

6.2.1 Local Identities

Most routing protocols use a globally unique name or address to identify each node; it is usually assumed that accidental identity collisions can be resolved (for example, by choosing a new identity when a collision is detected [564]) and deliberate collisions can be prevented (for example, by using certified identities). However, as discussed in section 2.1.2, these assumptions may not hold in an untrusted network with no central administration. Even if no identity-related attack is taking place, in some cases it may not be practical for every node to be aware of the membership and structure of a constantly changing network, and for privacy reasons it may be undesirable for protocols to require such knowledge.

The Darknet protocol does not use end-to-end addresses or any other global identifiers – in fact we assume that each node knows nothing about the network beyond its immediate neighbours. Each node must be able to distinguish

between its neighbours, but identities with local scope are sufficient for this purpose; a node is free to use a different identity when dealing with each neighbour.

How these local identities are implemented will depend on the context in which the Darknet protocol is used. For example, if the Darknet protocol is used at the network layer, each node will require a link-layer identifier for each of its neighbours, whereas if the Darknet protocol is used in an application-layer overlay, each node will require a transport-layer identifier for each neighbour. The establishment of suitable local identifiers occurs at a lower layer of the protocol stack than the Darknet protocol itself, so we do not consider it in detail here.

For the purposes of the Darknet protocol, nodes that are only connected by unidirectional links are not considered to be neighbours – the protocol cannot discover or use paths that contain unidirectional links.

6.2.2 The Message Pool

Each node in the Darknet protocol keeps a small pool of messages that are waiting for transmission to neighbouring nodes; these may include messages it has generated as a source and messages it has received as a relay. Messages are selected from the pool for transmission so as to maximise the probability that the transmitted messages will be acknowledged.

Algorithm 6.1 Choosing a message from the pool for transmission.

```
p_best ← 0
for all m ∈ pool do
  for all n ∈ neighbours do
    if ¬ previouslyReceivedFrom(m, n) ∧¬ previouslyTransmittedTo(m, n) then
      p ← estimateProbabilityOfAcknowledgement(m, n)
      if p > p_best then
        p_best ← p
        m_best ← m
        n_best ← n
      end if
    end if
  end for
end for
if p_best > p_min then
  transmitMessage(m_best, n_best)
end if
```

The state machine for the Darknet protocol is shown in Figure 6.1. Whenever a node enters the *choose message* state, it uses Algorithm 6.1 to select a message from the pool for transmission. Note that if $p_{best} \leq p_{min}$, no message is transmitted; thus the parameter p_{min}, the minimum acceptable probability of acknowledgement, controls how aggressively the protocol uses the available bandwidth.

The size of the message pool is limited: when the pool is full and a new message is added to the pool, the message with the lowest remaining probability of being acknowledged (which may be the new message) is discarded. Messages that have been acknowledged are removed from the pool, as are messages that have timed out (see section 6.2.3).

Duplicate messages are discarded to prevent routing loops and to reduce redundancy. Before discarding a duplicate message, a node records the neighbour from which the message was received, to avoid transmitting the message back to that neighbour. A node may transmit multiple copies of a given message to different neighbours, but it will never transmit the same message to a given neighbour more than once; nor will it transmit a message to a neighbour from which it has received a copy of that message.

Acknowledgements take priority over messages and are not held in the message pool.

6.2.3 Timeouts

Nodes store information about the messages they have transmitted in the *message table*. Each record in the table includes the hash of the expected acknowledgement, the local identifier of the neighbour from which the message was first received (unless the message was locally generated), and pointers to any relevant reliability estimators. When an acknowledgement is received or a timeout occurs, all the relevant reliability estimators are updated and the record is removed from the table.

As described in section 5.4.1, fixed timeouts are a simple way to ensure that adjacent relays along a given path discard their records at approximately the same time, thus minimising wasted storage. Timestamps would require clock synchronisation – a difficult problem in an untrusted network – and might allow relays to determine their distance from the source, so messages in the Darknet protocol do not carry timestamps. Each relay uses a fixed timeout based on the time at which it first received the message, while the source uses a fixed timeout based on the time at which the message was generated.

6.2.4 Reliability Estimators

Within the general framework described above there are many possible ways to estimate the probability that a message will be acknowledged. To achieve the best results, any information that may indicate the message's relationship to earlier messages should be taken into consideration. Even without end-to-end addresses, the local identities of the previous and next hops provide some information that can be used to distinguish between messages. Timing is also significant: changes in the network topology and traffic levels, even if they are not directly visible to the node, make new information more relevant than old information when estimating reliability. Thus our first attempt at a reliability estimator is a simple exponentially weighted moving average for each ordered pair of neighbours (previous hop and next hop). The moving average is adjusted upwards whenever a message is acknowledged (Equation 6.1), and downwards whenever a message times out (Equation 6.2):

$$x_{i+1} = (1-\alpha)x_i + \alpha \tag{6.1}$$

$$x_{i+1} = (1-\alpha)x_i, \tag{6.2}$$

where x_i is the estimate before updating the moving average and x_{i+1} is the estimate afterwards. The parameter α determines the sensitivity of the estimator to changes in reliability; in the simulations presented below we set α to 0.1. We will see in section 6.3.4 that even this simple reliability estimator provides a considerable improvement in efficiency when compared with flooding; further improvements may be possible by designing more sophisticated estimators.

The reliability estimators for locally generated messages are similar to those for forwarded messages: a node that is acting as a source keeps one estimator per neighbour to estimate the probability that locally generated messages will be acknowledged if they are transmitted to that neighbour.

6.2.5 Flow Labels

In addition to discovering implicit relationships between messages, the efficiency of routing can be improved if related messages are explicitly grouped together. We define a *flow* as any sequence of messages that have the same source and destination and that are semantically related in some way, such as the messages that make up a single file transfer. If relays can identify flows then they may be able to use the reliability information gathered from previous messages in a flow to improve their routing decisions for subsequent messages in the same flow.

To indicate the existence of flows we introduce *flow labels*. By marking all messages in a flow with the same arbitrary label, the contents of which are not significant, the source of a flow can provide a hint to relays that the messages are related. If for any reason the source does not want to reveal that two messages are related, the source is free to mark them with different labels. The flow label is not covered by the message authentication code in the U-ACK mechanism.

Flow labels have local scope: as a message travels across the network, it will be assigned a different label on each link it traverses. However, messages belonging to the same flow should have matching labels on any given link, and each flow traversing a link must be assigned a label that distinguishes it from any other flows currently traversing the same link. In particular, flows arriving at a node from different upstream neighbours *must* be assigned distinct labels on any downstream link, even if they happen to have matching labels on their respective upstream links.

The use of flow labels with local scope is similar to the use of label-swapping in Multiprotocol Label Switching [565], but there is no requirement to establish state in the relays before data transfer begins – labels can be assigned to new flows on the fly.

Although they do not identify the endpoints, flow labels enable fine-grained reliability measurement: messages arriving from the same neighbour with the same flow label are likely to have the same (unknown) source and destination, so the reliability of earlier messages in a flow can be used to estimate the reliability of later messages. Nodes can make use of this information by keeping a separate reliability estimator for each active flow. As with the simple per-pair estimators described above, new information is more likely to be relevant than old information, so our first attempt at a per-flow estimator once again uses an exponentially weighted moving average, as described in Equations 6.1 and 6.2.

To estimate the reliability of new flows, nodes also need to keep per-pair estimators that are only updated by the first message in each flow. These are used to estimate the reliability of a message *given that it is the first message in a new flow* – a per-pair estimator updated by every message would tend to direct new flows along the same paths as established flows, undermining the purpose of flow labels. The per-pair estimators are used to initialise per-flow estimators, which are thereafter updated independently.

As with per-pair estimators, the per-flow reliability estimators for locally generated messages are similar to those for forwarded messages: a node that is acting as a source keeps one estimator per neighbour, which is updated by the first message in each locally generated flow and then used to initialise an independent per-flow estimator for the remainder of the flow.

6.2.6 Aging and Discounting

From the description given at the start of section 6.2 it might appear that Darknet routing is likely to produce a large number of redundant messages. Aging and discounting are two techniques designed to improve the efficiency of the protocol by reducing the likelihood of transmitting redundant messages.

The technique of *aging* is based on the observation that an acknowledgement arriving after the timeout will not be recognised, since the corresponding record will have expired from the message table. Similarly, an acknowledgement arriving near the timeout is likely to miss the timeout at the next node. Thus the probability that a message will be acknowledged *and the acknowledgement will reach the source* decreases during the time the message is held in the pool, reaching zero at the timeout. Aging accounts for this effect by reducing each message's reliability linearly from the time it enters the pool to the time it expires.

The second technique, *discounting*, is based on the observation that each additional copy of a message that is transmitted is increasingly likely to be redundant. The higher the estimated reliability of the copies transmitted so far, the more likely it is that an additional copy will be redundant, so an additional copy should only be transmitted if the reliability of the copies transmitted so far is low. This has the effect of causing more exploration when the path to the destination is unknown or unreliable, and less exploration when a reliable path exists.

Ideally we would like to calculate the conditional probability of an additional copy of the message being acknowledged, given that none of the previous copies is acknowledged first. However, it would be impractical to store all of the information needed to estimate the conditional probability for each neighbour given any possible combination of previous neighbours, so in practice it is necessary to treat the probabilities as independent.

Let x_i denote the probability of the i^{th} copy of the message being acknowledged, and let y_i denote the conditional probability of the i^{th} copy being acknowledged, given that none of the previous copies is acknowledged first. Then, under the simplifying assumption of independence:

$$
\begin{aligned}
y_1 &= x_1 \\
y_i &= (1 - \sum_{j=1}^{i-1} y_j) x_i, \qquad (6.3)
\end{aligned}
$$

where x_i is the estimate before discounting and y_i is the discounted estimate. As the sum of the estimates for previous copies approaches one, meaning that an acknowledgement is expected as a result of the copies already transmitted, the discounted estimate for an additional copy approaches zero. The calculation can be made more efficient by keeping a running total.

6.2.7 Storage Overhead

The storage overhead of the U-ACK mechanism was discussed in general terms in section 5.4.2. Here we give some more specific figures for a peer-to-peer scenario.

The size of a node's message table depends primarily on its outgoing bandwidth. If we assume a timeout of 60 seconds and a typical message size of 1,000 bytes, a node may have up to 60 messages outstanding for every kB/s of outgoing bandwidth. If each record occupies 100 bytes, a typical peer-to-peer node with 64 kB/s of outgoing bandwidth would need to allocate 384 kB of storage for its message table.

Flow labels introduce a second source of storage overhead: the *flow table*. For each flow a node is currently forwarding, the node must record the mapping between the label on the upstream link and the label on the downstream link, together with a per-flow reliability estimator. If we assume that mappings are discarded after 60 seconds of inactivity, then the number of active mappings is limited by the node's outgoing bandwidth, since each outgoing message reactivates one mapping. If each mapping occupies 100 bytes, the typical peer-to-peer node described above would need a further 384 kB of storage for its flow table.

All of the information in the flow table is soft state: it does not need to survive across restarts, and information about inactive flows can be discarded to reclaim space. When information about a flow is discarded, the flow is not cut off, but the per-flow reliability estimator and the downstream flow label are lost, so if the flow later becomes active again it will be treated like a new flow and assigned a new label on the downstream link.

6.2.8 Attacks on Flow Labels

We stated above that it is "likely" that messages arriving from the same neighbour with the same flow label have the same source and destination. In fact, in an adversarial scenario there is no guarantee that this is the case – an internal attacker may assign any label to any message. By giving a message a new label, the attacker will cause downstream nodes to route the message separately from the other messages in its flow, which may prevent the message from reaching its destination. Any attacker who can modify a message's label, however, can just as easily drop the message, which would also be possible in the absence of labels. Regardless of whether the attacker drops messages or modifies them so that they do not reach their destinations, upstream nodes will tend to prefer more reliable paths that bypass the attacker, if any such paths exist.

A more subtle type of attack involves *adding* messages to a flow. The added messages may have the same destination as the original messages, a different destination, or no valid destination at all. They may be generated by the attacker or taken from other flows the attacker is forwarding.

If an attacker mixes messages with no valid destination into a flow, the added messages will not be acknowledged, reducing the measured reliability of the flow at every node between the attacker and the destination of the original messages. This may cause nodes downstream from the attacker to forward fewer of the original messages than they would otherwise have done. In that case, nodes upstream from the attacker will see the reliability of the flow decreasing on paths that pass through the attacker, while the reliability of the flow will be unaffected on paths that bypass the attacker, if any such paths exist. Thus, to the extent that the attacker succeeds in causing downstream nodes to stop forwarding the flow, upstream nodes will tend to route around the attacker. Note that this happens without any knowledge that an attack is taking place, and without any attempt to identify the attacker.

If the messages mixed into a flow have a valid destination that is different from the destination of the original messages, nodes downstream from the attacker will receive mixed signals as to which is the best path for the flow: they may forward the flow towards the destination of the original messages, the destination of the added messages, or both. To the extent that they stop forwarding the original messages to their correct destination, the result will be the same as described above: nodes upstream from the attacker will prefer paths that bypass the attacker, if any such paths exist.

It is for this reason that we mandate that flows arriving from different upstream neighbours must be assigned distinct flow labels on any downstream link, even if they happen to have matching labels on their respective upstream links: if a flow is forwarded along two paths, one of which passes through an attacker while the other bypasses the attacker, and if those paths meet at some node downstream from the attacker, the attacker's manipulation of the first path must not affect the downstream node's reliability estimate for the second path. Otherwise an attacker would be able to interfere with the discovery of fault-free paths.

Finally, we consider an attack that does not try to prevent messages from being delivered, but instead tries to determine their destination. By mixing messages from a flow with a known destination into a flow with an unknown destination, an attacker may try to determine whether the two flows follow the same downstream path. If the reliability of the flows is not affected by mixing then it is likely that the downstream paths are the same, at least as far as the nearer of the flows' destinations (the destinations may be the same). The attacker can test whether mixing affects reliability by alternating between periods of mixing and not mixing while looking for corresponding changes in the reliability of the flows.

To prevent this attack, destinations should check the flow labels of received messages; any message that has the same flow label as a message the destination has previously acknowledged, but that does not come from the same source as that message, should not be acknowledged or forwarded.

6.2.9 Comparison Between Flow Labels and Circuits

To understand what flow labels achieve, it is useful to compare them with the circuits used by low-latency mix networks such as Tor (see section 2.4.4). Like circuits, flow labels are used to indicate a relationship between messages that share the same source and destination, without identifying the source or destination to intermediate nodes. Both mechanisms increase the efficiency of routing – in the case of flow labels, by allowing fine-grained reliability measurement, and in the case of circuits, by reducing the use of asymmetric cryptography.

There are some significant differences between the two mechanisms, however. A Tor circuit follows a single, pre-established path: the source must select trustworthy and reliable relays when the circuit is set up, and the path is fixed for the lifetime of the circuit. A flow in the Darknet protocol, on the other hand, may follow any number of paths, with flow labels being allocated on demand on each link the flow traverses. Messages belonging to the same flow may follow different paths during the lifetime of the flow in response to changing network conditions, and may even follow different paths in parallel, if those paths are equally reliable.

6.2.10 Recognising Delivered Messages

Since the Darknet protocol does not use end-to-end addresses, every node that receives a message must check whether it is the intended destination of the message. A simple but inefficient way to do this is to attempt to decrypt and verify the message using the secret key shared with each source with which the destination communicates. (Note that the number of such sources may be far smaller than the number of sources in the network as a whole.)

The computational cost of decrypting and verifying every message in this way is certainly not trivial, but it is comparable to the costs incurred by mixes in low-latency mix networks (see section 2.4.4). Some of the routing protocols described in section 2.4.7 require every node to check whether it is the intended destination of every route request by performing an expensive asymmetric decryption; by contrast, the check described above only requires relatively inexpensive symmetric operations.

If the overhead of checking every message is unacceptable then it is easy to imagine a more efficient solution involving pseudo-random 'tags' that can be predicted by the intended destination, but which appear random to any other node. However, since the negotiation and management of the keys and counters needed to generate and recognise such tags would introduce additional complexity, we do not consider such a solution in detail here; we assume that the simple 'brute force' method is used.

6.3 Simulations

To evaluate the suitability of the Darknet protocol for censorship-resistant communication we carried out detailed simulations to compare its performance with that of two existing approaches to address-free routing: on-demand routing and flooding. As discussed in section 3.3.3, these approaches do not require identifiers with global scope or explicit knowledge of the network topology, so they are potentially suitable for use in robust, unlinkable communication.

To discover the best results that any protocol could achieve in the simulated scenarios we also simulated an ideal routing protocol that uses an oracle to select short, uncongested paths between sources and destinations.

To evaluate the suitability of the protocols for private and censorship-resistant communication, we compared them in a setting similar to the one proposed in Chapter 3: a friend-to-friend Internet overlay using constant rate cover traffic.

The protocols were evaluated for their scalability; for their ability to discover short, low-latency paths; for their sensitivity to the topology and bandwidth distribution of the network; and for their sensitivity to flow length. Sections 6.4 and 6.5 present further simulations evaluating the protocols' vulnerability to malicious and selfish behaviour.

6.3.1 The Simulator

The following requirements were identified for the simulator:

1. It should capture the temporal behaviour of the protocols as accurately as possible.
2. It should be capable of simulating unstructured peer-to-peer overlays with thousands of nodes.
3. It should realistically model bandwidth constraints and congestion, both in the underlying network and in the overlay.
4. It should be capable of simulating multiple protocols under the same conditions, including mixtures of faulty and non-faulty nodes.

As reported by Naicken *et al.* [566] in their review of peer-to-peer network simulators, there is no existing simulator that meets these requirements, except for low-level simulators such as NS-2 [567], which would require the protocols to be simulated at a prohibitively costly level of detail. We therefore developed a new discrete event-based simulator that models Internet overlays at the packet level. The simulator is described in Appendix A, together with the default parameters used to simulate each of the routing protocols.

All simulations were allowed to reach a steady state before any measurements were taken, and all results were averaged over ten runs. Error bars indicate the maximum and minimum values obtained in any run.

6.3.2 System Model

The simulated networks are friend-to-friend Internet overlays. Each simulated node represents a personal computer with a typical domestic broadband connection, while the links between nodes represent social relationships between the respective users. We assume that the social network does not change during the course of the simulations, but nodes may come online or go offline at any time; in other words we simulate leave-join churn but not join-leave churn (see section 2.2.1). The periods of uptime and downtime have exponentially distributed durations, each with a mean of two hours. The exponential distribution captures the fact that very short and very long uptime durations are observed in peer-to-peer networks. It was chosen over the more realistic power law distribution (see section 2.2.1) because it is memoryless; simulations with power law uptime or downtime distributions would have taken an extremely long time to reach a steady state.

Traffic in the network consists of flows between sources and destinations that are chosen uniformly at random from the nodes that are currently online. Each flow contains a geometrically distributed number of messages[1]. When a flow ends, or when either of its endpoints goes offline, it is immediately replaced with a new flow between randomly chosen

[1]The geometric distribution is the discrete counterpart of the exponential distribution, and is similarly memoryless.

endpoints. Within each flow, messages are generated at a constant rate. We obtained similar results, not shown here, for exponentially distributed inter-message intervals.

Choosing sources and destinations uniformly at random is arguably unrealistic – one might expect that users who want to communicate with one another will tend to be closer in the social network, and therefore closer in a friend-to-friend overlay, than users chosen at random. The relationship is not a simple one, however. Golder *et al.* [568] report that while the majority of messages in a social networking website are exchanged between users who identify themselves as friends, only a minority of pairs of friends exchange such messages, while Bigwood *et al.* [569] find significant differences between self-reported social networks and those based on encounters.

Having no simple model for users' communication preferences, we therefore use a random model to avoid overestimating the amount of local traffic, which would under-emphasise the importance of scalable routing.

Terminology

In the following descriptions of the simulated protocols, we distinguish between *data messages*, which carry end-to-end data payloads; *messages* in general, which include data messages and information used by the routing protocol, such as route requests and acknowledgements; and *packets*, which transport messages across the underlying network between neighbouring nodes. Each packet contains zero or more complete messages.

Cover Traffic

In section 3.2.4 we specified that every node should transmit packets at a constant rate, using padding to keep the volume of traffic that is visible to an external observer independent from the volume of genuine traffic. To model cover traffic in the simulations, each online node transmits packets to its online neighbours in round-robin order. Messages waiting for transmission to each neighbour are held in a drop-tail queue. When it is a neighbour's turn to receive a packet, as many complete messages as possible are removed from the head of the neighbour's queue and placed in the packet; the remainder of the packet is filled with padding. If there are no messages waiting, the entire packet is filled with padding. The neighbour that receives the packet can recognise and discard the padding – padding is not forwarded across the overlay.

Once the packet has been transmitted, a packet is created for the next neighbour in the same way. Thus an online node is always transmitting to some neighbour, as long as it has any neighbours online. The frequency with which packets are transmitted to each neighbour varies according to the number of online neighbours.

6.3.3 The Protocols

Flooding

Flooding is the simplest of the simulated protocols. When a source generates a data message it queues the message for transmission to each online neighbour. Any node that receives a data message for which it is not the destination queues the message for transmission to each online neighbour, except the one from which the message was received. Duplicate messages are discarded.

Flooding is included here mainly as a benchmark against which to compare the performance of the other protocols. It is known to perform poorly in large networks, but it can be very robust in small networks: when the offered load is low enough, every message travels across every link, ensuring that a fault-free path will be found if any such path exists.

On-Demand Routing

Several on-demand routing protocols for unlinkable communication were described in section 2.4.7. Rather than focussing on the details of a particular system, we simulate a generic on-demand protocol that captures the common features of the systems described in that section. No global identifiers such as addresses are used; instead, anonymous route identifiers are used to distinguish between routes without identifying the endpoints to the relays or *vice versa*. As with Darknet routing, we assume that any two users who wish to communicate across the overlay have established a shared secret out-of-band in advance.

As its name suggests, on-demand routing only discovers routes when they are needed. To discover a route, a source generates a *route request message* that is flooded through the overlay. When the destination receives the route request it generates a *route reply message* that follows the reverse path of the route request, establishing an anonymous route identifier at intermediate nodes. *Data messages* follow the route identifier from the source to the destination. If a node is unable to forward a data message because the next node along the route has gone offline, the node creates a *route error message* that is forwarded back to the source, which repeats the route discovery process with a new route identifier. During route discovery and rediscovery, sources queue any data messages they generate. If a route request times out without receiving a reply, another request is generated with a new route identifier.

To maintain the anonymity of the source, route requests do not have a time-to-live counter limiting their range.

Darknet Routing

As described in section 6.2.2, each node in the Darknet protocol keeps a small pool of data messages that are waiting to be transmitted, including those it has generated as a source and those it has received as a relay. When a packet is being prepared for transmission to a neighbour, any waiting acknowledgements are first removed from the neighbour's queue and added to the packet; if there is enough space left for a data message, the best message is selected from the pool.

Cover traffic makes the procedure for selecting a message somewhat simpler than that described in section 6.2.2: the neighbour that will receive the next packet has already been chosen, so it is only necessary to estimate the probability of each message receiving an acknowledgement if it is transmitted to the chosen neighbour; messages that have already been sent to or received from the chosen neighbour can be skipped. Since the packet will be padded to the same size regardless of whether it contains a data message, we set p_{min} to zero, so any message with a non-zero probability of being acknowledged is considered for transmission.

Two variants of the Darknet protocol are simulated: Darknet/Pair, which uses one reliability estimator per pair of neighbours and does not use flow labels; and Darknet/Flow, which uses flow labels and one estimator per flow, as described in section 6.2.5.

Oracle Routing

The last of our simulated protocols is intended to give an upper bound on the performance of any possible protocol under the simulated conditions. An oracle with complete knowledge of the network, including the state of every node's internal message queues, selects an independent path for each data message, avoiding faulty nodes and congested links. If no suitable path can be found, the message is dropped at the source to avoid placing unnecessary load on the network.

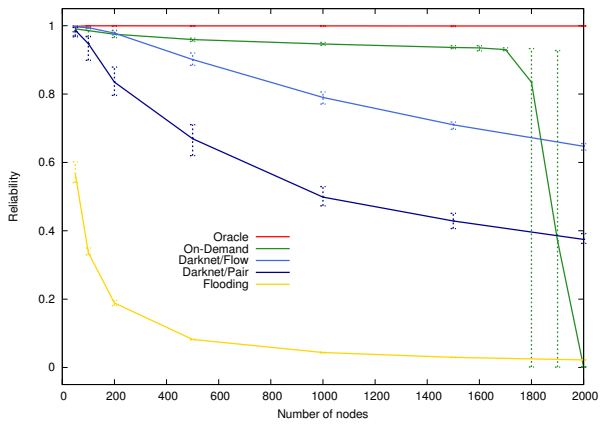
If several paths are available, the oracle chooses the shortest, breaking ties by congestion. The congestion of each path is measured by finding the largest number of messages queued for transmission on any link along the path; the least congested path is chosen. This tends to balance load across paths of equal length.

6.3.4 Scalability

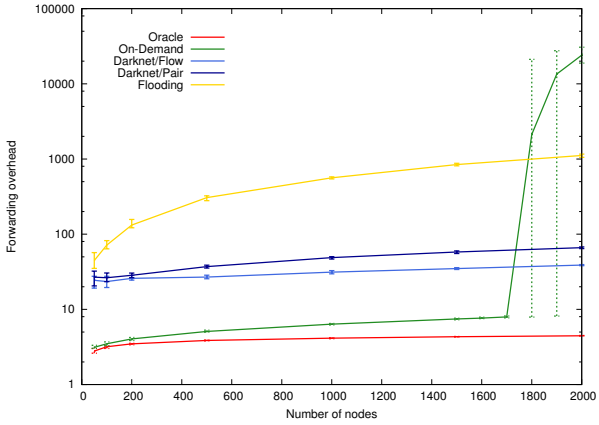
The first set of simulations evaluates the scalability of the protocols. We simulate networks of various sizes, from 50 to 2,000 nodes. In each run we measure the *reliability*, defined as the fraction of data messages successfully delivered, and the *forwarding overhead*, defined as the number of bytes transmitted by the network, excluding padding, divided by the total size of the data messages successfully delivered. We exclude padding from the overhead calculations to discover how much capacity each protocol leaves for background communication between neighbouring nodes, such as the private file sharing described in section 3.3.4.

The results of the scalability simulations are shown in Figure 6.2. As expected, oracle routing scales perfectly, achieving 100% reliability at all network sizes, whereas flooding does not scale well at all. Even in networks of just 50 nodes, the offered load is too high for flooding, and only 57% of messages reach their destinations; as the size of the network increases, flooding's reliability drops rapidly.

On-demand routing scales well up to 1,700 nodes, with high reliability and low overhead. However, at 1,800 nodes the variation between runs becomes extremely large, with some runs achieving 93% reliability and others less than



(a) Reliability as a function of the network size.



(b) Forwarding overhead as a function of the network size.

Figure 6.2: Reliability and forwarding overhead as functions of the network size. On-demand routing scales well up to 1,700 nodes, but then suffers from congestion collapse due to route request retransmissions.

0.2%. At 2,000 nodes every run achieves less than 0.2% reliability. Figure 6.2(b) shows that the overhead increases rapidly as the reliability drops – at 2,000 nodes, the overhead of on-demand routing is more than twenty times that of flooding.

Further investigation shows that the cause of this sudden change is *congestion collapse*: when the network becomes sufficiently congested that some route requests fail to reach their destinations, the automatic retransmission of route requests adds to the congestion, creating a positive feedback loop that overwhelms the network with route requests, destroying reliability. The threshold at which congestion collapse occurs can be raised by increasing the retransmission timeout of route requests, but the underlying problem still exists. A better strategy might be to use an adaptive timeout, backing off exponentially after each retransmission, as is done in AODV (see section 2.3.3.1).

Since all of the remaining simulations use networks of 1,000 nodes, we do not investigate this problem further; we assume that on-demand routing can be made to scale adequately beyond the network sizes considered here, through the use of adaptive timeouts or other techniques.

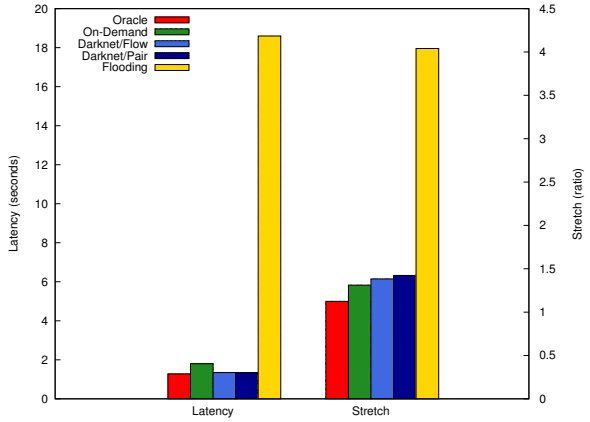
Darknet routing with per-pair estimators scales much better than flooding, delivering 99% of messages in networks of 50 nodes and 37% in networks of 2,000 nodes. Flow labels and per-flow estimators significantly improve the protocol's scalability: Darknet/Flow achieves 65% reliability in networks of 2,000 nodes, nearly twice the reliability of Darknet/Pair. Up to the point of on-demand routing's congestion collapse, however, Darknet/Flow's scalability is far below that of on-demand routing, and the downward trend suggests that neither variant of the Darknet protocol is likely to be suitable for use in very large networks. We do not expect Darknet routing to suffer from congestion collapse at any scale, since it does not retransmit messages, but if higher-layer protocols use retransmission, the possibility of congestion collapse must be borne in mind.

All of the simulations in the remainder of this chapter use networks of 1,000 nodes.

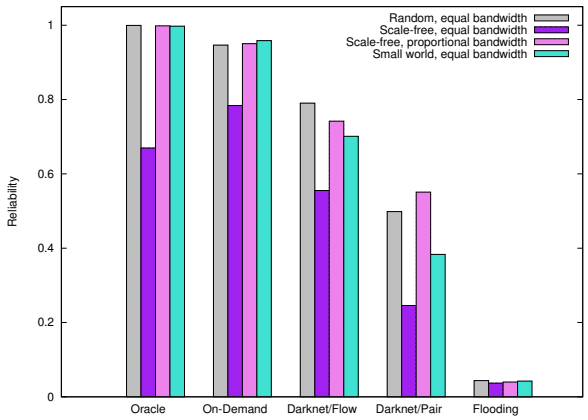
6.3.5 Latency, Stretch and Coverage

To assess how good the protocols are at finding short paths between sources and destinations, we measure the *latency* and *stretch* of delivered messages. Latency is the elapsed time between a data message being transmitted by its source and received by its destination. Stretch is the length of the path taken by a data message, divided by the ideal path length, which is the length of the shortest path between the source and the destination.

Figure 6.3(a) compares the latency and stretch achieved by the four protocols. The poor performance of flooding is obvious: the paths it uses are on average four times the ideal length, which is one of the factors contributing to an end-to-end latency ten times higher than that of any other protocol; the other factor is congestion, which causes long queueing delays. It should be noted, however, that when the offered load is low, flooding is good at discovering short,



(a) Latency and stretch.



(b) The effect of network topology and bandwidth distribution.

Figure 6.3: Left: Flooding has much higher latency and stretch than the other protocols. The Darknet protocol uses somewhat longer paths than on-demand routing. Right: The effect of network topology and bandwidth distribution. None of the protocols performs well in scale-free networks unless bandwidth is proportional to degree.

low-latency paths, because it tries every possible path in parallel. On demand-routing, which uses flooding for route discovery, has latency and stretch values comparable to the ideal values achieved by oracle routing.

Both variants of Darknet routing have lower latency than on-demand routing but higher stretch, which would seem to suggest that they deliver data messages more quickly than on-demand routing, but over longer paths. This paradoxical effect is more pronounced for Darknet/Pair than for Darknet/Flow. The extra round-trip for route discovery in on-demand routing is unlikely to be the cause, since route discovery only delays the first few messages of each flow.

One possible explanation is that Darknet routing tends to be more successful at delivering messages between endpoints that are close together in the network, which would reduce the average latency of successfully delivered messages relative to a protocol that delivered all messages. To determine whether this is the case, we measure the *coverage* of each protocol, defined as the ratio of the mean ideal path length of delivered messages to the mean ideal path length of generated messages. A low coverage value indicates that a protocol is biased towards endpoints that are close together in the network.

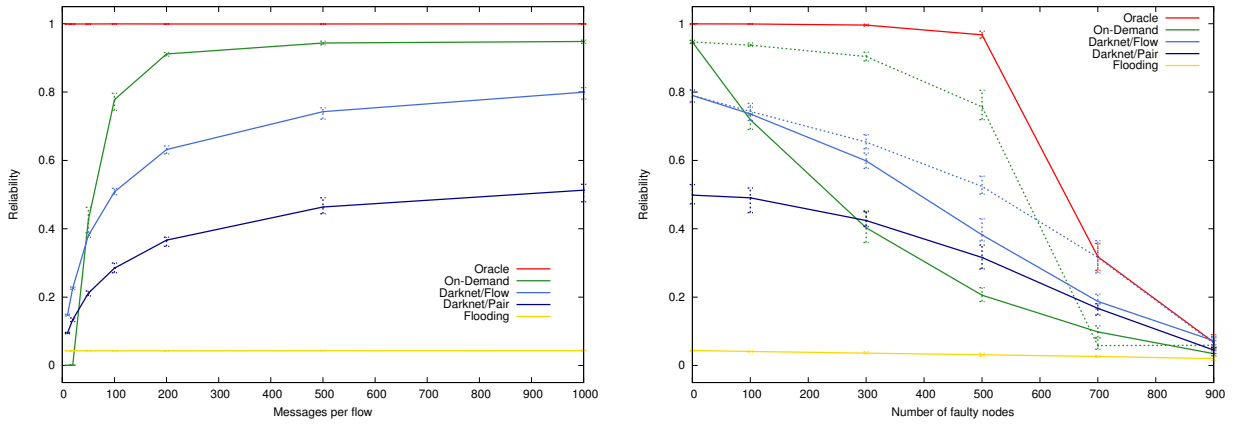
In networks of 1,000 nodes, oracle and on-demand routing achieve 100% coverage; Darknet/Flow achieves 99%; Darknet/Pair, 97%; and flooding, 94%. It therefore seems likely that Darknet routing's imperfect coverage contributes to its apparently low latency.

6.3.6 Topology and Bandwidth Distribution

As discussed in section 2.1.1, online social networks tend to exhibit highly skewed degree distributions, with a small number of nodes having far more than the average number of neighbours. It might be argued that the degree distribution is likely to be less skewed in networks designed for censorship-resistance than in networks designed for casual socialising, since users of the former are unlikely to connect to dozens of distant acquaintances. Nevertheless, we need to determine whether the Darknet protocol is suitable for use over a wide range of topologies.

To measure the impact of network topology on the performance of the four routing protocols, we simulate three types of topology: random networks, in which every pair of nodes is connected with equal probability; scale-free networks, which are generated with Bauke and Sherrington's local attachment model; and small world networks, which are generated with Kleinberg's navigable small world model. (See section 2.1.1 for details of the scale-free and small world constructions.) Scale-free networks have power law degree distributions, in contrast to the Poisson degree distributions of random networks, while small world networks are more highly clustered than random networks.

When using round-robin cover traffic (see section 6.3.2), a node's degree affects the amount of bandwidth it allocates to each link: a high-degree node will devote a smaller fraction of its bandwidth to each of its links than a low-degree node, so the outgoing links of a high-degree node may become congested unless the node has correspondingly high



(a) Reliability as a function of flow length.

(b) The impact of black hole attacks.

Figure 6.4: Left: Short flows increase route discovery costs, adversely affecting all of the protocols except oracle routing and flooding. Right: The impact of black hole attacks. The dotted lines show the effect of the attacker dropping all messages.

bandwidth. Similarly, a high-degree node's incoming bandwidth may be exhausted by traffic from its many neighbours, especially those that have low degree and therefore allocate large amounts of bandwidth to each link.

To separate these effects from any other effects of the degree distribution, we simulate two bandwidth distributions for scale-free networks. In the first, all nodes have equal bandwidth, as in the random and small world topologies, while in the second, each node's bandwidth is proportional to its degree. The total bandwidth across all nodes is the same in all cases.

The results of the simulations are shown in Figure 6.3(b). All of the protocols perform worse in scale-free networks with equal bandwidth than in any of the other conditions. Even oracle routing's reliability falls from 100% to 67%. Surprisingly, on-demand routing performs better than oracle routing under these conditions, suggesting that we could improve the heuristics oracle routing uses to avoid congestion. Even with a perfectly optimised implementation, however, there may be limits to what any routing protocol can achieve under these conditions: in a scale-free network, the links attached to high-degree nodes make up a significant fraction of all links, so if those links are too congested to be usable for routing, it may not be possible to find alternative paths between every pair of nodes. If we consider the high-degree nodes to be excluded from routing entirely then we are in the situation studied by Albert *et al.*, who found that removing a small number of high-degree nodes from a scale-free network can disconnect the remaining nodes (see section 2.1.1).

Scale-free structure is not in itself a problem: all of the protocols perform well in scale-free networks when bandwidth is proportional to degree. It would seem, therefore, that if friend-to-friend networks have highly skewed degree distributions without correspondingly skewed bandwidth distributions, they may not function efficiently as multi-hop overlays, especially when cover traffic is used. This could turn out to be a fundamental weakness of our chosen approach. It might be possible to mitigate the problem by allowing users to make new friend-to-friend connections only if they have enough bandwidth to prevent their existing connections from becoming congested, but that would inconvenience users and might weaken the overlay by reducing the number of links. To understand the effect of such a policy on the network's structure we would need to model the growth of the network. We leave the investigation of this important issue for future work.

6.3.7 Flow Length

The results presented so far have been based on an average of 1,000 messages per flow. Figure 6.4(a) shows the effect of changing the average flow length.

Unsurprisingly, oracle routing and flooding are unaffected by flow length, since they treat every message independently. On-demand routing performs slightly worse with shorter flows, due to the overhead of more frequent route

requests, and below 200 messages per flow its performance begins to collapse.

Both variants of Darknet routing perform substantially worse with shorter flows. Since the Darknet protocol works by identifying implicit and explicit relationships between messages, its performance depends on the number and strength of those relationships. Long flows give nodes time to identify reliable paths, and the initial cost of path discovery can be amortised over the lifetime of the flow. These results suggest that Darknet routing is more likely to be useful for applications that produce long flows of messages between the same endpoints – such as video conferencing, file transfer and instant messaging – than for applications that produce short flows between a large number of endpoints, such as web browsing.

6.3.8 Summary

We have seen that the performance of the simulated protocols is strongly dependent on the network conditions: large networks and short flows create problems for all of the simulated protocols except oracle routing, while skewed degree distributions make routing difficult even for an omniscient oracle, unless the high-degree nodes have correspondingly high bandwidth.

The performance of Darknet routing is better than that of flooding under all conditions, although that is not much of a boast. Flow labels improve the scalability of the Darknet protocol, making it comparable to on-demand routing, although its overhead is higher. The Darknet protocol appears to be best suited to networks of moderate size (up to a few hundred nodes), and to applications that create long flows of messages between the same endpoints.

6.4 Robustness

In this section we simulate active internal attacks against routing, varying the number of faulty nodes to measure each protocol's resistance to attack. The faulty nodes are chosen at random and have the same bandwidth and average degree as other nodes.

We do not simulate Sybil or tunnelling attacks, since the simulated protocols do not make use of global identifiers. Similarly, we do not simulate eclipse attacks or whitewashing, since they are prevented by the use of friend-to-friend connections. We concentrate on two attacks that apply to friend-to-friend overlays without global identifiers: black hole attacks and denial-of-service attacks.

6.4.1 Black Hole Attacks

The black hole attack was described in section 2.3.3: the attacker behaves correctly during route discovery but drops data messages. The details of the attack are different for each of the simulated protocols.

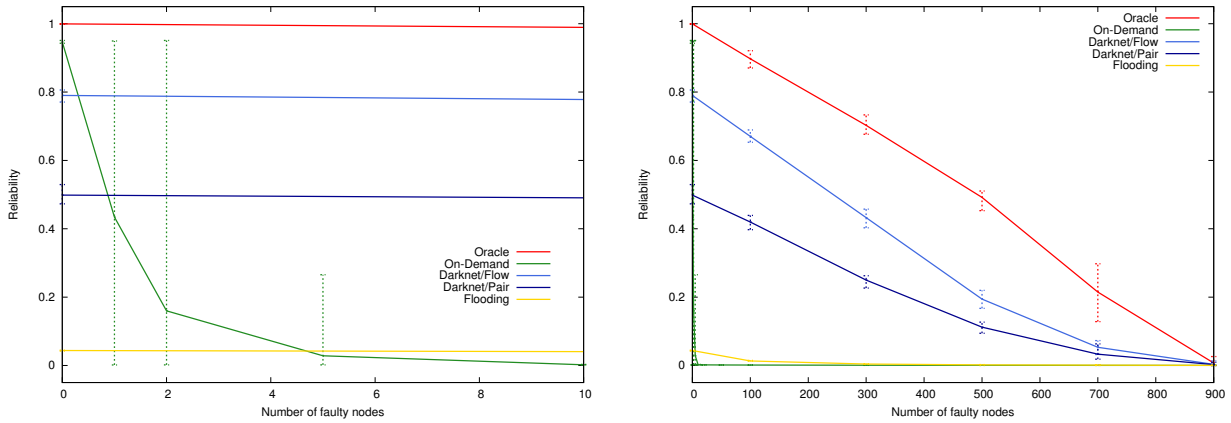
In on-demand routing, the faulty nodes forward route requests and route replies, but drop data messages. (A faulty node will never receive a route error message because they are only sent in response to data messages.)

In Darknet/Flow, the faulty nodes forward the first message in each flow but drop subsequent messages, exploiting the fact that per-pair estimators are only used for new flows (see section 6.2.5). Nodes upstream from a faulty node will therefore update their per-pair estimators when the first message is acknowledged, but subsequent losses will only affect their per-flow estimators, so they will continue to route new flows through the attacker.

In Darknet/Pair, all messages affect the per-pair estimators of upstream nodes, so there is no way for an attacker to drop selected messages without being routed around. The faulty nodes therefore drop all messages. Flooding similarly provides no basis on which to target particular messages, so the faulty nodes drop all messages.

In oracle routing, we assume that the oracle can identify faulty nodes and route around them. This reduces the number of available paths, so the attack affects reliability when the number of faulty nodes is large.

Figure 6.4(b) shows the impact of the black hole attack. For on-demand routing and Darknet/Flow, the dotted lines show the effect of the attacker dropping all messages, demonstrating that the targetted attack is more effective than dropping all messages.



(a) Denial-of-service attacks with 0 to 10 attackers.

(b) Denial-of-service attacks with 0 to 900 attackers.

Figure 6.5: The impact of denial-of-service attacks. Left: On-demand routing collapses immediately. Right: No protocol is unaffected by the attacks, but the robustness of Darknet routing is close to optimal.

On-demand routing is highly vulnerable to the black hole attack, which cuts its reliability in half when 30% of the nodes are faulty, compared to a 4% reduction in reliability when the same number of faulty nodes simply drop all messages.

Darknet/Flow is also somewhat vulnerable: when the majority of nodes are faulty, the black hole attack reduces its performance to not much better than that of Darknet/Pair. This highlights the importance of information restriction, part of the design strategy described in section 3.4: the more information a protocol reveals, the more vulnerable it is to complex forms of misbehaviour. Flow labels provide valuable information for efficient routing, but the same information can be used by an attacker to choose which messages to drop. Darknet/Flow still outperforms Darknet/Pair for any given number of faulty nodes, however, showing that resilience to misbehaving nodes must be weighed against the efficiency gained by providing information to well-behaved nodes.

When up to 10% of the nodes are faulty, on-demand routing has the best performance of any protocol except oracle routing. Beyond that point, Darknet/Flow outperforms all protocols except oracle routing.

6.4.2 Denial-of-Service Attacks

We next consider a denial-of-service attack in which faulty nodes flood the network with messages to prevent users from communicating. The faulty nodes also drop all incoming packets – they do not take part in forwarding, and messages sent to faulty destinations are not delivered or acknowledged. As before, the faulty nodes are chosen at random and have the same bandwidth and degree as other nodes.

The details of the attack again depend on the protocol. In on-demand routing, the faulty nodes constantly generate route requests for nonexistent destinations, exploiting the protocol's flooding-based route discovery phase to consume as much bandwidth as possible.

The Darknet protocol does not have a separate route discovery phase, so the faulty nodes generate data messages for nonexistent destinations. In Darknet/Flow, the attacker gives every message a different flow label, causing other nodes to treat it as the first message in a new flow, which makes them more likely to forward multiple copies of the message and prevents them from using flow labels to identify undeliverable traffic.

Darknet/Pair does not use flow labels, so the faulty nodes simply generate data messages for nonexistent destinations. In flooding they do the same. In all protocols, the faulty nodes generate messages as quickly as they can transmit them.

To provide a benchmark against which to compare the other protocols, we do not simulate any attack traffic in oracle routing, but the faulty nodes still drop all incoming packets. As with the black hole attack, we assume that the oracle can identify faulty nodes and route around them. Messages sent to faulty destinations are still lost, however.

Figures 6.5(a) and 6.5(b) show the impact of the denial-of-service attack. The performance of oracle routing degrades

linearly when up to 50% of the nodes are faulty, reflecting the probability of faulty nodes being chosen as destinations; thereafter the rate of degradation increases as it becomes difficult to find paths between non-faulty nodes.

Darknet routing does not quite degrade linearly, but its robustness is impressive. In both variants of the protocol, per-pair estimators detect that faulty nodes are not delivering messages, allowing the neighbours of faulty nodes to route around them. Per-pair estimators also allow the neighbours of faulty nodes to recognise that the attacker's messages are not being acknowledged, making them progressively less likely to forward the attacker's messages. (One could imagine an alternative attack strategy in which the faulty nodes acknowledged each other's messages – but in that case the protocol would use the acknowledgements to find paths between the faulty nodes, minimising the resources consumed by the attack.)

As the number of faulty nodes increases, the performance of flooding degrades more quickly than that of Darknet routing: when 30% of the nodes are faulty, flooding is one tenth as reliable as it is in the absence of attackers (0.43% versus 4.4%), whereas Darknet/Flow is more than half as reliable (43% versus 79%).

On-demand routing is extremely vulnerable to denial-of-service attacks. As shown in Figure 6.5(a), a single faulty node can cut the network's reliability in half. When there are five or more faulty nodes, on-demand routing performs worse than flooding, because for a data message to be delivered, three messages – a route request, a route reply, and the data message itself – must cross the congested network, whereas for flooding only the data message itself needs to cross the network.

The attack against on-demand routing is made even more effective by congestion collapse, which causes innocent nodes to amplify the attack by retransmitting their own route requests, as described in section 6.3.4. Adaptive retransmission timeouts might prevent accidental congestion collapse, but they would not protect against deliberate denial-of-service attacks: by increasing their retransmission timeouts, non-faulty nodes would avoid contributing their own route requests to the congestion, but they would still forward the attacker's route requests. The fundamental problem is that a single faulty node can produce enough traffic to exhaust the bandwidth of every non-faulty node, due to the use of flooding in route discovery.

For any number of faulty nodes, Darknet/Flow has the best performance of any protocol except oracle routing.

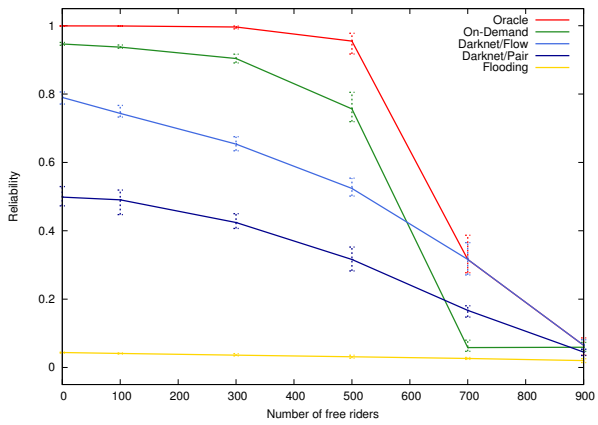
6.4.3 Summary

We have seen that the Darknet protocol is robust to active attacks against routing by misbehaving nodes. Its performance degrades under attack, but the degradation is less severe than that experienced by either on-demand routing or flooding, and when large numbers of nodes are faulty, Darknet routing outperforms both the existing protocols. Flow labels expose the Darknet protocol to black hole attacks, but the impact of such attacks is not large enough to negate the performance benefits of using flow labels.

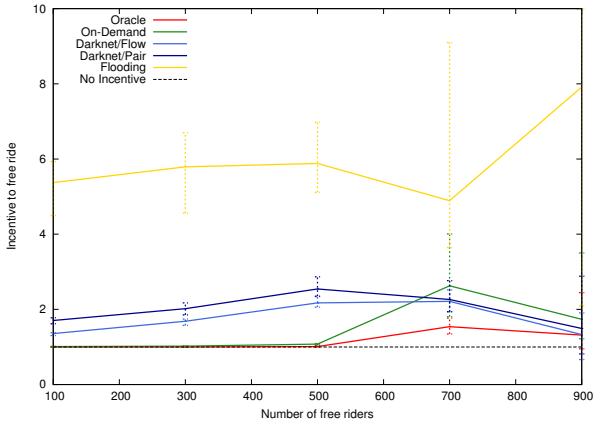
6.5 Cooperation

In this section we investigate the impact of selfish behaviour on the simulated protocols. We assume that some or all of the users are selfish, meaning that they are interested in obtaining the best possible service from the network without regard to their impact on other users. Since locally generated messages often compete for resources with messages being forwarded on behalf of other users, we investigate whether selfish users can increase the likelihood of their own messages being delivered by refusing to forward messages – a behaviour we refer to as *free riding*. We assume that selfish users have the same privacy requirements as other users, so free riders still use round-robin cover traffic, but when selecting messages for transmission they only consider locally generated messages. The neighbours of a free rider cannot trivially detect its misbehaviour, since it will still cooperate in the routing protocol when acting as a source or destination.

Figure 6.6(a) shows that free riding seriously harms the reliability of all the simulated protocols. Oracle routing chooses paths that do not include free riders, but as the number of free riders increases the remaining paths become congested and reliability drops. On-demand routing similarly collapses when the majority of nodes are free riders. The performance of Darknet routing degrades more steadily, but the final result is the same: no protocol achieves more than 6% reliability when 90% of the nodes are free riders.



(a) The impact of free riding.



(b) Incentives to free ride.

Figure 6.6: Free riding. Left: As the number of free riders increases, reliability drops. Right: All of the simulated protocols reward free riding by providing higher reliability to free riders than cooperators.

To determine whether such high levels of free riding are a realistic threat, we compare the reliability experienced by free riders with that experienced by cooperative nodes, to see whether selfish users would benefit from switching to free riding implementations of the node software. Figure 6.6(b) shows the ratio of the reliability experienced by free riders to that experienced by cooperators, as a function of the number of free riders in the network. With one exception, all of the protocols have ratios greater than one at all levels of free riding, meaning that there is an incentive for selfish users to switch to free riding. The only exception is oracle routing with up to 30% free riders, which successfully delivers all messages, creating no incentive either way. In general, the lower a protocol's reliability at a given level of free riding, the stronger the incentive to free ride.

Taken together, the two figures represent a serious threat to the use of peer-to-peer networks for censorship-resistance. Figure 6.6(b) shows that users can obtain better service from all of the simulated protocols by free riding, while Figure 6.6(a) shows that by doing so they seriously harm the network. The damage that can occur is only limited by the number of users who are willing to use node implementations that give them better service at the cost of other users.

As discussed in section 2.2.5, several authors have argued that users of private peer-to-peer networks will be less likely to behave selfishly than users of public networks. The argument remains unproven, however. While users of private BitTorrent servers have been shown to maintain higher sharing ratios than users of public servers, it is not clear whether they are doing so because of increased altruism or because of the threat of eviction by the central server.

In the next section we describe a protocol that is designed to counter the threat of free riding by creating incentives for selfish users to cooperate in routing.

6.6 The Exu Protocol

The Exu protocol combines the basic approach of the Darknet protocol – address-free routing using the feedback provided by U-ACKs – with the expected utility strategy described in Chapter 4. Like the Darknet protocol, the Exu protocol attempts to route around unreliable nodes, but it also gives selfish users an incentive to operate reliable nodes by providing a higher level of service to neighbours that successfully deliver messages.

Unlike many of the incentive mechanisms described in section 2.5, the Exu protocol does not depend on *ad hoc* rules that might be exploited by selfish users: every action taken by the protocol is based on strict utility-maximisation, so selfish users have no reason to deviate from the protocol, while unselfish users can use the protocol to discourage selfish users from harming the network by free riding.

6.6.1 Direct Utility

In what follows, we will refer to nodes, rather than users, as being selfish or unselfish, and as receiving benefits or incurring costs; it is assumed that users will choose implementations of the node software that represent their own preferences as far as possible.

In common with the work of Srinivasan *et al.* and Félegyházi *et al.* discussed in section 2.5.4, we assume that selfish nodes are only interested in the level of service they receive from the network, as measured by the successful delivery of the data messages they generate as sources. A node is assumed to receive a *subjective benefit* when a locally generated data message is delivered, and to incur a *subjective cost* when it transmits or receives any message.

For simplicity we will assume that a given node assigns the same benefit to the delivery of every message it generates, and that transmission and reception costs do not vary with time. These restrictions are not required by the model, but they simplify the presentation. We use b to represent the subjective benefit of a locally generated data message being delivered, c_m and c'_m to represent the subjective costs of transmitting and receiving a data message, respectively, and c_a and c'_a to represent the subjective costs of transmitting and receiving an acknowledgement, respectively.

When a node considers performing an action, these costs and benefits can be combined with the estimated probabilities of the action's possible outcomes to calculate the *expected utility* of performing the action. We first consider actions that have a direct benefit for the node performing them, and then move on to actions that derive their expected benefit from reciprocation.

Let x denote the action of transmitting a locally generated data message, where x_1 is the outcome that the message is delivered, x_2 is the outcome that it is not, and $p(x_1)$ is the estimated probability of delivery. A benefit of b is received if the message is delivered, along with a cost of c'_a to receive the acknowledgement. Transmitting the message costs c_m regardless of whether it is delivered. If costs and benefits can be expressed in the same units then we can use Equation 4.2 to calculate the expected utility, $u(x)$, of transmitting the message:

$$u(x) = \sum_i (b(x_i) - c(x_i))p(x_i) = (b - c_m - c'_a)p(x_1) - c_m p(x_2) = (b - c'_a)p(x_1) - c_m. \tag{6.4}$$

Note that if $p(x_1)$ is sufficiently small, $u(x)$ may be less than zero. In that case an action with no benefit and no cost, which we call the *null action*, is preferable to x. As in the model of Urpi *et al.* described in section 2.5.4, it may be rational for a node not to transmit a locally generated message if the probability of its delivery is sufficiently low.

6.6.2 Indirect Utility

The expected utility strategy described in Chapter 4 estimates the benefit of cooperating with a neighbour by dividing the total benefit received from the neighbour by the number of times cooperation has been given to the neighbour. In the Exu protocol, which bases its decisions on the expected utility strategy, we define the total benefit received from a neighbour as the benefit of all the locally generated messages delivered by the neighbour, where the delivery of each message provides a benefit of b. The number of times cooperation has been given is defined as the number of messages delivered on the neighbour's behalf, without knowing whether the neighbour was the source of some or all of those messages. Both values depend on the number of messages *delivered* rather than the number of messages transmitted. In other words, cooperation is measured through acknowledgements.

This has two interesting consequences. First, a node may cooperate with a neighbour simply by acknowledging a message for which the node itself is the destination. Second, if a node forwards a message but does not return an acknowledgement, it has not cooperated. It may have *attempted* to cooperate, but since there is no proof of the attempt, and since the neighbour does not benefit from the attempt, no reciprocation can be expected. The cost of attempting and failing to cooperate can be taken into account by assigning a cost but no benefit to the outcome that no acknowledgement is received for a forwarded message.

More formally, consider a node with k neighbours, $n_1 \dots n_k$. The total benefit received from n_i is denoted b_i, while the number of acknowledgements returned to n_i is denoted a_i. Let y_i denote the action of forwarding a message on behalf of n_i, where y_{i_1} is the outcome that the message is delivered, y_{i_2} is the outcome that it is not, and $p(y_{i_1})$ is the estimated probability of delivery. From Equation 4.2 we can calculate the expected utility of forwarding a message, $u(y_i)$, as follows:

$$u(y_i) = \sum_j (b(y_{i_j}) - c(y_{i_j}))p(y_{i_j}) = (\frac{b_i}{a_i} - c_m - c'_a - c_a)p(y_{i_1}) - c_m p(y_{i_2}) = (\frac{b_i}{a_i} - c'_a - c_a)p(y_{i_1}) - c_m. \quad (6.5)$$

Similarly, let z_i denote the action of returning an acknowledgement to n_i, where the acknowledgement may have been generated locally or may have been received from another neighbour as proof of delivery of a forwarded message. If we assume that the links between neighbouring nodes are reliable, there is only one possible outcome for z_i, so $p(z_{i_1}) = 1$ and we can calculate $u(z_i)$ as follows:

$$u(z_i) = \sum_j (b(z_{i_j}) - c(z_{i_j}))p(z_{i_j}) = b(z_{i_1}) - c(z_{i_1}) = \frac{b_i}{a_i} - c_a. \quad (6.6)$$

A selfish node now has everything it needs to compare the actions available to it and choose the action with the greatest expected utility, and that is exactly what the Exu protocol does.

6.6.3 Selecting a Message for Transmission

Like the Darknet protocol, the Exu protocol keeps a pool of messages that are waiting for transmission, but it uses expected utility rather than reliability to select messages from the pool. The state machine for the Exu protocol is almost identical to the state machine for the Darknet protocol, as shown in Figure 6.1, but whenever an Exu node enters the *choose message* state it uses Algorithm 6.2 to choose a message from the pool for transmission.

Algorithm 6.2 Choosing a message from the pool for transmission.

```
u_best ← 0
for all m ∈ pool do
  for all n ∈ neighbours do
    if ¬ previouslyReceivedFrom(m, n) ∧¬ previouslyTransmittedTo(m, n) then
      p ← estimateProbabilityOfAcknowledgement(m, n)
      if locallyGenerated(m) then
        u ← expectedUtilityOfTransmittingOwnMessage(p)
      else
        u ← expectedUtilityOfForwardingMessage(p, m)
      end if
      if u > u_best then
        u_best ← u
        m_best ← m
        n_best ← n
      end if
    end if
  end for
end for
if u_best > 0 then
  transmitMessage(m_best, n_best)
end if
```

In Algorithm 6.2 the procedure expectedUtilityOfTransmittingOwnMessage(p) evaluates Equation 6.4 for a locally generated message, while the procedure expectedUtilityOfForwardingMessage(p, m) evaluates Equation 6.5 for a forwarded message. Note that if the highest expected utility, u_{best}, is not greater than zero, the node chooses the *null action* and no message is transmitted.

As with the Darknet protocol, acknowledgements are given priority over data messages and are not kept in the pool. In theory, the expected utility of returning an acknowledgement should be calculated using Equation 6.6 and compared with the expected utility of transmitting each message in the pool, but in practice the cost of transmitting a U-ACK

is smaller than the cost of transmitting a message, the benefit is similar (since the benefit of forwarding a message is equal to the benefit of returning an acknowledgement for that message), and there is no need to estimate reliability for acknowledgements, so the simplification of giving U-ACKs priority over data messages makes little practical difference.

Aging and discounting are applied in the same way as for the Darknet protocol, as described in section 6.2.6. When the message pool is full and a new message is added to the pool, the message with the lowest remaining expected utility (which may be the new message) is discarded.

6.6.4 Incommensurable Costs and Benefits

In section 4.4.1 we showed that even in situations where costs and benefits cannot be expressed in the same units, it is still possible to make a rational choice between available actions by calculating the *expected benefit/cost ratio* of each action rather than its expected utility. The Exu protocol can be adapted for use in such situations by replacing the expressions for $u(x)$, $u(y_i)$ and $u(z_i)$ with the following expressions for $u'(x)$, $u'(y_i)$ and $u'(z_i)$:

$$u'(x) = \frac{\sum_i b(x_i)p(x_i)}{\sum_i c(x_i)p(x_i)} = \frac{bp(x_1)}{(c_m + c'_a)p(x_1) + c_m p(x_2)} = \frac{bp(x_1)}{c'_a p(x_1) + c_m}, \tag{6.7}$$

$$u'(y_i) = \frac{\sum_j b(y_{i_j})p(y_{i_j})}{\sum_j c(y_{i_j})p(y_{i_j})} = \frac{(b_i/a_i)p(y_{i_1})}{(c_m + c'_a + c_a)p(y_{i_1}) + c_m p(y_{i_2})} = \frac{b_i p(y_{i_1})}{a_i((c'_a + c_a)p(y_{i_1}) + c_m)}, \tag{6.8}$$

$$u'(z_i) = \frac{\sum_j b(z_{i_j})p(z_{i_j})}{\sum_j c(z_{i_j})p(z_{i_j})} = \frac{b(z_{i_1})}{c(z_{i_1})} = \frac{b_i}{a_i c_a}. \tag{6.9}$$

The procedure for selecting a message from the pool in this scenario is the same as that described in section 6.6.3, except that the expected benefit/cost ratio of the null action is undefined – as we saw in Chapter 4, the expected benefit/cost ratio cannot be used to decide whether an action with a possible cost is preferable to an action with no possible cost. To replace the null action in such situations we therefore define a minimum expected benefit/cost ratio, u'_{min}, which is a parameter of the protocol. When costs and benefits are incommensurable, a node will not take any action unless its expected benefit/cost ratio is greater than u'_{min}. Just as costs and benefits may be subjective, different nodes may use different values of u'_{min}.

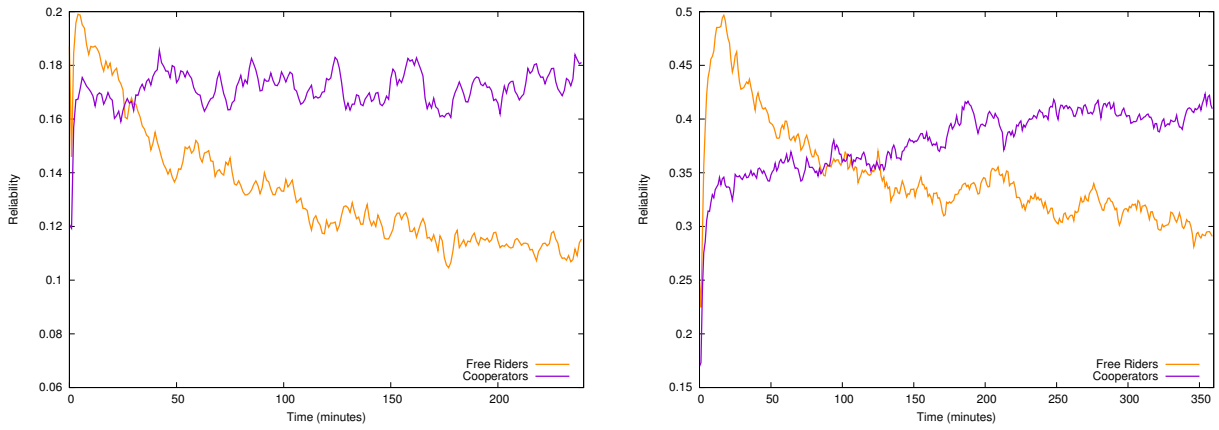
6.6.5 Bootstrapping Cooperation

The problem of bootstrapping cooperation in the expected utility strategy was discussed in Chapter 4: the EXU strategy simulated in that chapter treated each new opponent as though it had cooperated once and received cooperation once. We have found through experimentation that the same approach performs poorly in Exu routing. Whereas in the games simulated in Chapter 4 a player was free to cooperate with any of her opponents in any round, in Exu routing a node may only have the opportunity to cooperate with certain neighbours at certain times, depending on the flows that exist in the network.

If a node initially treats each neighbour as though the neighbour has delivered one locally generated message and received one acknowledgement – the equivalent of the approach used in Chapter 4 – then returning a single acknowledgement to a new neighbour will halve the expected utility of forwarding another message on that neighbour's behalf. The unintentional effect of this sudden change resembles the black hole attack: a node will forward the first message from each new neighbour but will then become much less likely to forward subsequent messages.

This problem can be solved by initialising a_i to a large value, so that each acknowledgement makes a small proportional difference to a_i. That gives rise to a second problem, however: nodes become slow to distinguish between cooperative and uncooperative neighbours. Initialising b_i to a small value solves the second problem by ensuring that the first few increases to b_i, which occur when cooperation is first received from a neighbour, make a large proportional difference.

Combining a large initial value of a_i with a small initial value of b_i will tend to make cooperation with new neighbours unattractive, but in the experiments we have conducted so far it seems to be more effective to start from a low expected utility and adapt upwards for cooperative neighbours than to start from a high expected utility and adapt downwards



(a) Reliability for free riders and cooperators in Exu/Pair.

(b) Reliability for free riders and cooperators in Exu/Flow.

Figure 6.7: Exu routing creates strong incentives against free riding. Left: Exu/Pair adapts to free riders within 30 minutes. Right: Exu/Flow takes 100 minutes to adapt.

for uncooperative neighbours. In the simulations presented in section 6.6.6 we initialise a_i to 1,000 and b_i to $b/1,000$, so the expected benefit of forwarding a message is one millionth of the expected benefit of transmitting a locally generated message. The result is that nodes only use their spare capacity to forward their neighbours' messages, unless and until they receive cooperation, which rapidly raises the value of b_i. Under these conditions there is a strong incentive to cooperate, as shown in the next section. This can be compared with GNUnet's strategy of allocating spare resources to 'unpaid' queries (see section 2.5.4).

Further investigation is needed of the impact of these parameters on the incentive to cooperate, the time required to establish reciprocation between cooperative nodes, and the protocol's resistance to whitewashers.

6.6.6 Simulations

To see whether the Exu protocol creates incentives against free riding, we simulate it under the same conditions used in section 6.5. As with the Darknet protocol, two variants of the Exu protocol are simulated: Exu/Pair, which uses one reliability estimator per pair of neighbours and does not use flow labels, and Exu/Flow, which uses flow labels and one estimator per flow.

Bootstrapping

We first examine the process of bootstrapping cooperation by simulating networks of 1,000 nodes, half of which are free riders. At the end of each minute of simulated time we record the reliability experienced by free riders and that experienced by cooperators. Results are averaged over ten runs. As in the previous simulations, we are interested in discovering whether selfish users have an incentive to switch between cooperative and free riding implementations of the node software.

Figure 6.7 shows that both variants of the Exu protocol create strong incentives for selfish users to switch to cooperative node implementations. Free riders experience higher reliability than cooperators during the first few minutes of simulated time, when the cooperative nodes have not yet been able to distinguish between their cooperative and uncooperative neighbours, but after around 30 minutes in the case of Exu/Pair and 100 minutes in the case of Exu/Flow, cooperators begin to outperform free riders, and after 150 minutes both variants of the protocol create strong incentives against free riding. The strength of the incentives continues to grow throughout the period of simulation, suggesting that the Exu protocol would be particularly effective at discouraging free riding in networks with long-term relationships between neighbours.

Exu/Flow adapts to the presence of free riders more slowly than Exu/Pair, which can be explained by considering the forwarding behaviour of each variant of the protocol. Exu/Pair has less information on which to base its forwarding

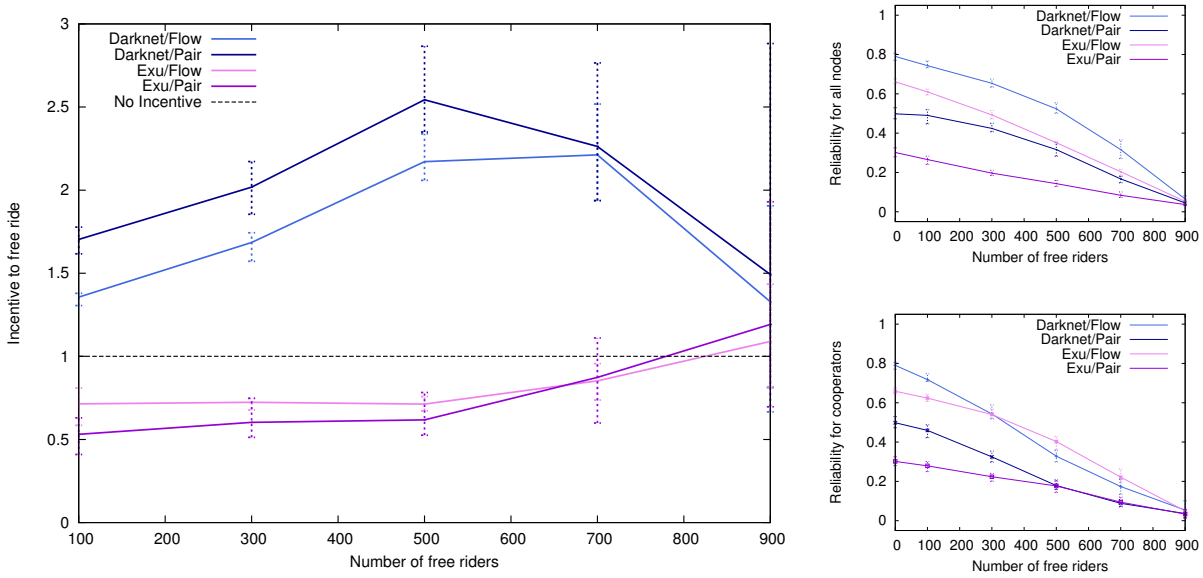


Figure 6.8: Free riding in the Darknet and Exu protocols. Left: Exu routing creates incentives against free riding when up to 80% of the nodes are free riders. Top right: Overall reliability is lower in Exu routing than in Darknet routing. Bottom right: At high levels of free riding, reliability for cooperators is higher in Exu routing than in Darknet routing.

decisions than Exu/Flow and therefore tends to send more redundant messages and to discover longer paths (by way of analogy, compare the overhead of Darknet/Pair and Darknet/Flow in Figure 6.2(b) and their stretch in Figure 6.3(a)). Thus a cooperative Exu/Pair node tends to participate in more paths than a cooperative Exu/Flow node, allowing it to 'sample' the reliability of its neighbours more evenly, and thus to distinguish more quickly between neighbours that are uncooperative and those that are just unable to deliver a particular flow. This can be seen as an instance of the well-known *exploration/exploitation tradeoff* in adaptive systems: flow labels allow Exu/Flow to exploit discovered paths more efficiently, but in their absence, Exu/Pair explores more paths and therefore learns about its neighbours more quickly.

Incentives to Cooperate

To determine whether the Exu protocol creates incentives against free riding regardless of the number of free riders, we repeat the simulations from section 6.5 using Exu routing. Due to limits on the resources available for simulation we make an exception to the usual rule of running simulations until they reach a steady state before taking measurements, and the following results are based on a settling time of 180 minutes of simulated time for Exu/Pair and 300 minutes for Exu/Flow. As Figure 6.7 shows, at the end of the settling period the incentives to cooperate are still growing stronger and the reliability experienced by cooperative nodes is still increasing, so the following results may underestimate the effectiveness of Exu routing.

The left frame of Figure 6.8 compares the incentives for cooperation provided by the Darknet and Exu protocols. Cooperators in Exu routing experience substantially better reliability than free riders when up to 70% of the nodes are free riders. When 80% of the nodes are free riders, cooperators and free riders experience roughly equal reliability, and beyond that point there is an incentive for selfish users to switch to free riding. The success of the Exu protocol in discouraging free riding therefore depends on the initial conditions: if more than 80% of the users initially select free riding node implementations, there is an incentive for the remaining users to do likewise, but if less than 80% are initially free riders, there is an incentive for the initial free riders to switch to cooperative implementations.

Exu routing's protection against free riders comes at a cost, however. The top right frame of Figure 6.8 shows that the overall reliability of the Exu protocol is lower than that of the Darknet protocol at all levels of free riding. It should be noted that these figures include the reliability experienced by free riders: to the extent that the Exu protocol succeeds in creating an incentive against free riding by giving free riders a diminished level of service, it is bound to harm

overall reliability.

To exclude this effect, the bottom right frame of Figure 6.8 compares the reliability experienced by cooperative nodes in the Darknet and Exu protocols. Unfortunately, even for cooperative nodes, the performance of Exu routing is worse than that of Darknet routing under some conditions. Cooperators receive higher reliability in Exu/Flow than in Darknet/Flow when more than 30% of the nodes are free riders, and essentially the same reliability in Exu/Pair and Darknet/Pair when more than 50% of the nodes are free riders, but at low levels of free riding the Exu protocol performs worse for cooperative nodes than the Darknet protocol does.

Perhaps this result is not surprising, since the quantity being measured – system-wide reliability – is the quantity that Darknet nodes are trying to optimise, whereas each Exu node is trying to optimise its own utility. Nevertheless, it is clear that resilience to selfish behaviour, like resilience to malicious behaviour, carries a cost in terms of performance.

6.6.7 Discussion

The simulations presented here use the system model described in section 6.3.2, and they reflect the assumptions of that model – in particular, the assumption that the connections between neighbouring nodes are based on long-lived social relationships between users. That assumption becomes particularly important when considering the incentives for cooperation created by the Exu protocol – as we saw in section 4.1, establishing cooperation between self-interested parties depends on 'the shadow of the future', and if nodes are free to adopt new identities then the incentives created by the Exu protocol may be undermined.

Figure 6.7 shows that free riders experience higher reliability than cooperators during the first few minutes of the Exu protocol's operation; if free riders could change identities every few minutes then it might be possible for them to maintain this advantage indefinitely. Whereas we argued in section 6.2.1 that the Darknet protocol does not place any particular requirements on the local identities provided by lower protocol layers, the same is not true of the Exu protocol. In order for the Exu protocol to resist whitewashing, the network must provide conditions similar to those described in section 4.1.4:

- Nodes must be able to recognise neighbours with which they have interacted in the past.
- Nodes must not be able to change their local identities at will.
- There must be a reasonable expectation that any two nodes that are presently neighbours will remain neighbours in the near future.

These additional requirements restrict the applicability of the Exu protocol. The requirements are met by the friend-to-friend overlays described in Chapter 3 and simulated in the present chapter, but they might not be met in other settings, or additional mechanisms might be needed in order to meet them.

6.6.8 Summary

We have described the Exu protocol, a variant of the Darknet protocol that uses the expected utility strategy to select messages for transmission. Every action taken by a node following the Exu protocol is based on an attempt to maximise its own utility, either by transmitting locally generated messages or by earning reciprocation from its neighbours by forwarding messages for them and returning acknowledgements.

Unlike the Darknet protocol, the Exu protocol creates strong incentives against free riding by providing a higher level of service to neighbours that reliably deliver messages. However, the performance of the protocol in the absence of free riders is worse than that of the Darknet protocol. We have yet to investigate the Exu protocol's robustness to the active attacks simulated in section 6.4, but it seems possible that its strategy of giving priority to reliable neighbours could help to limit the impact of denial-of-service attacks by restricting the resources available to faulty nodes.

Further work is needed to investigate the Exu protocol's performance under a range of conditions, and to determine how various parameters affect the incentives to cooperate and the time required to establish cooperation.

6.7 Conclusion

Address-free routing in an adversarial environment with selfish users is a huge and complex problem, and we have only scratched the surface of it here. Within the scope of the present work many interesting engineering problems remain to be solved, such as the design of effective reliability estimators, a fuller investigation of realistic traffic patterns, and the characterisation of the loss and delay properties of the Darknet and Exu protocols. A real implementation of either protocol would require sensitivity analysis of all the relevant parameters, not just those examined here.

There are also many opportunities for architectural improvements that go beyond the scope of the present work, including the use of techniques such as multi-path routing or forward error correction to cope with losses, and the design of robust higher-layer protocols.

One issue that requires quantitative investigation is the degree of unlinkability provided by the Darknet and Exu protocols. Although the protocols do not use end-to-end addresses, the very fact that relays can infer relationships between messages shows that implicit information about end-to-end communication patterns (as well as explicit information, in the form of flow labels) is available to internal observers. By combining external observations of the structure of the overlay with internal observations such as the round-trip time of acknowledgements, an adversary may be able to assign higher probabilities to some suspected sources and destinations than others, and in the long term may be able to identify pairs of communicating users with high confidence. We consider this issue further in section 7.3.

Chapter 7

Conclusions and Future Work

> "Of course some methods are mastered, some results are verified. Often it's amusing. But so many things we wanted have not been attained, or only partially and not like we imagined. What communication have we desired, or experienced, or only simulated? What real project has been lost?"
> – Guy Debord, *Critique of Separation*

7.1 Summary of Results

We set out to demonstrate that private, censorship-resistant communication is possible over public networks such as the Internet, even in the presence of a powerful adversary who can monitor all traffic in the underlying network and make targetted attacks, and even when there is no central entity that is trusted by all of the users.

We have presented the following results in support of our thesis:

1. A new game theoretic model of cooperation under scarcity, *the sharer's dilemma*. By making a simple extension to a well-known game we gained a new perspective on the problem of cooperation in networks: incorporating scarcity into the prisoner's dilemma reframes the question of cooperation as a question of prioritisation. The sharer's dilemma provides a formal model of many situations in nature and society where the benefit of cooperation is higher than the cost, and where resources for cooperation are scarce.

2. An *expected utility strategy* for the sharer's dilemma that robustly encourages cooperation between selfish players, together with a model of rational decision-making in situations where costs and benefits are incommensurable, which uses the concept of *expected benefit/cost ratio* as an alternative to Savage's expected utility.

3. A lightweight cryptographic mechanism for producing *unforgeable acknowledgements* to prove that messages have been delivered unmodified to their intended destinations across an untrusted network, without revealing the identities of the communication endpoints to the relays or *vice versa*. The acknowledgements created by the mechanism can be verified by untrusted third parties, but the only parties who need to be able to verify one another's identities are the source and destination of each message. The mechanism can operate at any layer of the protocol stack, and does not require relays be aware of the details of higher-layer protocols. The implementation we described, which is based on one-way hash functions and message authentication codes, can be seen as an instance of a more general *puzzle-solution pattern* for proving that a message has been delivered to its intended recipient.

4. The *Darknet protocol*, a routing protocol that discovers reliable paths across untrusted networks while maintaining unlinkability between sources and destinations. The protocol is an example of *address-free routing*, an approach to communication across multi-hop networks that does not require identifiers with global scope or explicit knowledge of the topology, making it possible to operate without a trusted identity infrastructure. The Darknet protocol is suitable for use in networks of moderate size, especially for applications that create long flows of messages between the same endpoints, and is robust to active internal attacks against routing that badly affect existing address-free routing protocols.

5. The *Exu protocol*, an alternative method of address-free routing that combines the adaptive path discovery techniques of the Darknet protocol with the expected utility strategy to create a protocol in which every action taken by a node is strictly utility-maximising. By using the unforgeable acknowledgement mechanism to measure the cooperation received from neighbouring nodes, the Exu protocol provides strong incentives for selfish users to cooperate in message forwarding.

7.2 Discussion

Although the results summarised above can be slotted into similarly shaped holes in the design sketch from Chapter 3, we have not described or evaluated a complete system. The simulations presented in Chapter 6 were necessarily limited in scope and detail – they omitted many factors that would be relevant to the performance of a real system, such as usage patterns, heterogeneity among nodes and users, implementation quality, and usability. Some of those factors could not have been evaluated under experimental conditions even if we had used emulation rather than simulation to evaluate the protocols, and would require real-world testing to understand their impact.

Implementing the Darknet and Exu protocols would be a reasonable next step to take, since the effect on usability of factors like latency and cover traffic is harder to evaluate using cold figures than through direct experience. However, developing and deploying an application that would allow a meaningful evaluation of such factors is a large task, and one that goes beyond the scope of the present work.

The performance of the expected utility strategy in BitTorrent swarms could also be investigated through a real implementation, perhaps by modifying an existing open source client. Since the implementation would not require any modifications to the BitTorrent protocol, it could be tested experimentally in real swarms without requiring widespread deployment.

In Chapter 1 we made a number of assumptions that strongly influenced the direction of the work presented here. Some, like the powerful threat model described in section 1.3.1, were pessimistic, ruling out the use of existing techniques; others, like the assumptions about social connectivity and shared secrets in section 1.3.2, were optimistic. If those assumptions do not hold then the approach to censorship-resistant communication that we have followed may not be viable.

Another issue that calls into question the chosen approach is the difficulty of routing across scale-free networks where the high-degree nodes do not have correspondingly high capacity. As we saw in section 6.3.6, even an omniscient oracle may have trouble finding uncongested routes across such networks. If the solution proposed in that section – limiting each node to as many connections as its bandwidth can support – does not produce routable overlays, we may need to reconsider whether routing is the appropriate model for indirect communication across friend-to-friend networks. Other approaches, such as the high-latency publish-subscribe approach taken by Usenet (see section 2.3.2), may cope better with congestion and poor connectivity than low-latency routing can.

While the simulations presented in Chapter 4 showed that the EXU strategy performs well under a range of conditions, we have not proven that it is the optimal strategy for the sharer's dilemma, nor that it is evolutionarily stable. Existing definitions of evolutionary stability are based on the assumption of pairwise encounters, an assumption that does not hold for the sharer's dilemma. To prove or disprove the stability of the EXU strategy, we would first need to develop a new and more general definition of evolutionary stability. Such a definition could have broad application beyond the present work.

Similarly, when considering free riding in Chapter 6, we did not examine all the strategies a selfish player might adopt; we only showed that under certain conditions, the Exu protocol rewards complete cooperation more highly than complete free riding. Given the complexity of the model used in Chapter 6, it seems unlikely that we will be able to prove that the Exu protocol is optimal, but we could evaluate a wider range of strategies, as we did for the sharer's dilemma. We also need to examine the effect of the Exu protocol's incentive mechanism on nodes that are poorly connected to the rest of the network, and on nodes that do not act as sources or destinations.

7.3 Future Work

7.3.1 Evaluation of Unlinkability

The most important unresolved question in the evaluation of the Darknet and Exu protocols is the degree of unlinkability they provide. Borisov [328] has demonstrated that simulations can be used to measure the anonymity provided by systems that are too complex to model analytically, but he only considers the observation of one message at a time. The long-term traffic analysis techniques described in section 2.4.2 show that an adversary can learn much more by combining observations over time than by considering each message separately, so we are interested in extending Borisov's simulation-based approach to multiple messages.

As in Borisov's simulations, this can be done by nominating certain nodes as corrupt and recording their internal observations of the system. Those observations can then be combined with the adversary's external observations of the overlay to determine which nodes are possible sources and destinations of each message from the adversary's point of view. Whenever the adversary is able to determine that two messages have the same (as yet unidentified) source and destination, the anonymity sets of those messages can be intersected. For example, in the Darknet protocol, all messages that reach a corrupt node over the same upstream link with the same flow label must have the same source and destination, as must any other copies of those messages observed elsewhere in the network. Furthermore, the path from the source to the corrupt node must have remained unchanged since the flow label was first observed.

Given a set of possible sources and destinations for each message, there are many ways to quantify the anonymity and unlinkability provided by the system, as discussed in section 2.4.1. One could look, first of all, at the distribution of anonymity set sizes across messages, as Borisov does, although in this case the sets would sometimes be smaller than in the single-message case, due to intersection. One could also consider the degree of exposure of individual users by looking at the distribution of anonymity set sizes across each user's messages, or the distribution across users of the minimum anonymity set size per user. For each of these measures of anonymity, unlinkability could be quantified by considering the number of possible source-destination pairs for each message.

For any of the above metrics, the value of the metric under a single-message analysis could be compared to its value under a multiple-message analysis of the same observations to quantify the routing protocol's susceptibility to long-term traffic analysis. For a protocol such as flooding that gives no indication of the connection between messages, the two values would be identical.

We are also interested in developing system-wide anonymity metrics to measure the extent of the adversary's partial knowledge of the overall communication pattern. Edman *et al.* [330] and Grégoire and Hamel [331] have developed system-wide anonymity metrics for 'black box' observations of messages entering and leaving an anonymity system, but new metrics are needed for peer-to-peer architectures where the endpoints are located inside the system.

7.3.2 Cover Traffic Strategies

New anonymity metrics might also allow us to determine whether the round-robin cover traffic strategy described in section 3.2.4 provides optimal protection against a combined internal and external observer, or whether another strategy – such as keeping the transmission rate of each connection, rather than each node, constant – would provide better protection.

It is also important to consider how cover traffic interacts with active internal attacks such as watermarking (see section 2.4.2). A design that protects optimally against passive attacks may be vulnerable to active attacks or *vice versa*, as in the case of structured overlay lookups (see section 2.4.5). The choice of an appropriate cover traffic strategy may therefore depend on the chosen threat model.

7.3.3 Overlay Construction and Maintenance

The networks simulated in Chapter 6 are idealised friend-to-friend overlays where any two users who wish to establish a direct connection can do so. In practice there are many obstacles to creating a connection between two personal computers, including dynamic IP addresses, firewalls, and network address translators.

In a public peer-to-peer network the process of connecting to the overlay can be automated by embedding the addresses of a few dedicated bootstrap nodes in the node software. Private peer-to-peer networks, on the other hand, require manual configuration. Not only must users exchange contact details when they first connect – which may require multiple round-trips if a middlebox is present – but they may also need to do so whenever one of them moves to a new network address.

Manual overlay maintenance not only places a burden on users, it also encourages them to rely on insecure channels to exchange contact details, which could place them at increased risk of monitoring or man-in-the-middle attacks. We must therefore consider ways to automate overlay maintenance as far as possible.

As discussed in section 2.1.1, many social networks are highly clustered, meaning that any two neighbouring nodes are likely to share a common neighbour. It might be possible to exploit this property by using mutual neighbours to exchange up-to-date contact details when nodes reconnect to the network. It would still be necessary to exchange addresses manually if all of a node's neighbours had moved to new addresses, however, and the protocol for discovering mutual neighbours would have to be designed carefully to avoid leaking private information about users' friendships.

7.3.4 Parameter Sensitivity

In Chapter 6 we found that all of the simulated protocols were sensitive to parameters such as flow length, network size and degree distribution. Even within the simplified setting of the simulations there are many other factors that we did not explore. Each protocol has parameters such as timeouts and queue sizes that can result in threshold behaviour, as we saw with the unexpected congestion collapse of on-demand routing. Unfortunately the number of possible combinations of parameters is too large to allow a full exploration of the parameter space, but we can examine the sensitivity of each parameter individually.

The Exu protocol in particular requires further investigation: the initial values of b_i and a_i, and their evolution over time for cooperative and uncooperative neighbours, will affect the strength of the incentives to cooperate and the time required to establish cooperation, and may also determine whether the Exu protocol can achieve the same performance as the Darknet protocol in the absence of free riders. Packet-level simulations of the protocol's long-term behaviour would be very resource-intensive, however, so we may need to develop a simpler model that isolates the parameters of interest.

7.3.5 Improved Estimators

So far we have only considered simple reliability estimators based on exponentially weighted moving averages, using the same parameter $\alpha = 0.1$ for all estimators under all conditions. This is an area where it is likely that engineering improvements can be made.

Kalman filters [570] are used in many time series estimation and control problems, and may be suitable for use as reliability estimators, although the assumption of Gaussian noise in the original Kalman filter design must be treated with caution: in an adversarial scenario, an attacker may be able to model the internal state of the filter and manipulate the 'noise' by selectively dropping messages in order to cause a node to make poor routing decisions. The unscented Kalman filter [571] may be able to prevent such attacks, since it does not assume Gaussian noise.

7.3.6 Architectural Extensions

In addition to engineering improvements such as reducing the time required to establish cooperation and increasing the accuracy of estimators, there are many opportunities for extensions to the architecture of the Darknet and Exu protocols.

Multi-Path Routing

We have not yet studied the loss characteristics of the Darknet and Exu protocols to determine whether losses are evenly spread across flows, or whether some flows bear disproportionate losses. The latter possibility seems more likely than the former, since there is a positive feedback loop between messages being acknowledged and subsequent

messages covered by the same reliability estimator being transmitted, which could reinforce small initial differences in reliability between flows that might occur by chance.

If it does turn out to be the case that some flows fare better than others for essentially stochastic reasons, sources might benefit from using multiple flows in parallel when trying to reach a given destination, assigning messages to one flow or another depending on the measured reliability of each flow. Relays could also use the same technique for forwarded messages, since they are free to modify flow labels.

If losses turn out to be more evenly distributed, on the other hand, endpoints may benefit from using forward error correction to minimise the number of unrecoverable losses.

Higher Protocol Layers

So far we have concentrated on providing an unreliable datagram service across friend-to-friend overlays, but a complete censorship-resistant communication system would most likely require a reliable, stream-based transport layer above the datagram layer, since most applications cannot cope with message loss or reordering.

Although TCP is designed to cope with accidental loss and reordering, it is extremely vulnerable to *adversarial* loss and reordering, as well as to manipulation of the round-trip time variance [572, 573]. Such attacks would not be easy to prevent with the U-ACK mechanism, since a small adversarial loss rate is sufficient to damage TCP throughput severely, while reordering and moderately delaying messages would not be detectable by the U-ACK mechanism at all.

The design of transport-layer protocols that can cope robustly with adversarial loss, reordering and delay is an interesting research challenge, and one that is not restricted to the context of the present work.

7.3.7 Hybrid Overlays Revisited

In section 3.3.1 we introduced the concept of a *hybrid overlay* composed of connections across a mixture of underlying networks, which might include local wireless networks as well as public networks like the Internet. While we have concentrated on Internet overlays in the subsequent investigation, hybrid overlays containing short-range wireless links could have a significant advantage over pure Internet overlays for censorship-resistance: short-range links cannot be monitored from a central point in the way that Internet links can, which could help to meet the requirement of covert membership described in section 1.4, as well as resisting attacks on anonymity that rely on knowledge of the overlay topology.

To pursue this advantage as far as possible, we might consider building censorship-resistant networks entirely from short-range links between personal devices carried by the users. That would require a major shift of focus, however, since each link would be active only intermittently and nodes would spend the majority of their time disconnected from their neighbours. Combining the already considerable challenges of message delivery in intermittently connected networks [574, 575] with the security requirements of censorship-resistant communication seems likely to provide enough food for at least another eight years of thought.

References

[1] S.C. Jansen. *Censorship: The Knot that Binds Power and Knowledge*. Oxford University Press, 1988.

[2] J. Habermas. *Legitimation Crisis*. London: Heinemann Educational, 1976.

[3] M.E. Warren. The self in discursive democracy. In S.K. White, editor, *The Cambridge Companion to Habermas*, pages 167–200. Cambridge University Press, 1995.

[4] L. Strauss. *Persecution and the Art of Writing*. University of Chicago Press, 1952.

[5] J.F. Lyotard. *The Postmodern Condition: A Report on Knowledge*. Manchester University Press, 1984.

[6] R. MacKinnon. China's censorship 2.0: How companies censor bloggers. *First Monday*, 14(2), February 2009.

[7] Freedom House. *Freedom on the Net: A Global Assessment of Internet and Digital Media*. Washington, DC: Freedom House, March 2009.

[8] F. Fassihi. Iranian crackdown goes global. *The Wall Street Journal*, December 2009.

[9] A. Soldatov. Kremlin.com. *Index on Censorship*, 39(1):70–78, March 2010.

[10] R.J. Deibert, J.G. Palfrey, R. Rohozinski, and J. Zittrain, editors. *Access Controlled: The Shaping of Power, Rights, and Rule in Cyberspace*. MIT Press, April 2010.

[11] D. Murakami Wood, editor. *A Report on the Surveillance Society*. London: Information Commissioner's Office, November 2006.

[12] M. Foucault. *Discipline and Punish: The Birth of the Prison*. Penguin, 1979.

[13] J. Bentham. Panopticon. In M. Božovič, editor, *The Panopticon Writings*, pages 29–95. London: Verso, 1995.

[14] J. Stanley. *The Surveillance-Industrial Complex: How the American Government is Conscripting Businesses and Individuals in the Construction of a Surveillance Society*. New York: American Civil Liberties Union, August 2004.

[15] G. Crossman, H. Kitchin, R. Kuna, M. Skrein, and J. Russell. *Overlooked: Surveillance and Personal Privacy in Modern Britain*. London: Liberty, December 2007.

[16] M. Gill and A. Spriggs. *Assessing the Impact of CCTV*. London: The Home Office, February 2005.

[17] S.L. Nock. *The Costs of Privacy: Surveillance and Reputation in America*. Aldine de Gruyter, 1993.

[18] T.M. Cooley. *A Treatise on the Law of Torts*. Chicago: Callaghan and Company, 2nd edition, 1888.

[19] D.M. Pedersen. Dimensions of privacy. *Perceptual and Motor Skills*, 48:1291–1297, 1979.

[20] D. Feldman. Secrecy, dignity or autonomy? Views of privacy as a civil liberty. *Current Legal Problems*, 47(2):41–71, 1994.

[21] J. Telesin. Inside 'samizdat'. *Encounter*, 40(2):25–33, February 1973.

[22] M.K. Sparrow. The application of network analysis to criminal intelligence: An assessment of the prospects. *Social Networks*, 13(3):251–274, 1991.

[23] V. Krebs. Uncloaking terrorist networks. *First Monday*, 7(4), March 2002.

[24] B. Reed. *Formalizing the Informal: A Network Analysis of an Insurgency*. PhD thesis, Department of Sociology, University of Maryland, June 2006.

[25] US Department of the Army. Social network analysis and other analytical tools. In D.H. Petraeus and J.F. Amos, editors, *Field Manual FM 3-24: Counterinsurgency*, appendix B. Washington, DC: Department of the Army, December 2006.

[26] J.D. Farley. *Toward a Mathematical Theory of Counterterrorism: Building the Perfect Terrorist Cell*. Pennsylvania: US Army War College, December 2007.

[27] S. Gorman. NSA's domestic spying grows as agency sweeps up data. *The Wall Street Journal*, March 2008.

[28] L. Beam. Leaderless resistance. *The Seditionist*, 12, February 1992.

[29] Critical Art Ensemble. *Electronic Civil Disobedience and Other Unpopular Ideas*. New York: Autonomedia, 1997.

[30] H. Bey. *TAZ: The Temporary Autonomous Zone*. New York: Autonomedia, 2nd edition, 2003.

[31] J.F. Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, volume 2009 of *Lecture Notes in Computer Science*, pages 10–29, July 2000.

[32] E. Lichtblau. Senate approves bill to broaden wiretap powers. *The New York Times*, July 2008.

[33] G. Schmid, editor. *Report of the Temporary Committee on the ECHELON Interception System*. European Parliament, July 2001.

[34] R. Singel. Whistle-blower outs NSA spy room. *Wired*, April 2006.

[35] S.J. Murdoch and R. Anderson. Shifting borders. *Index on Censorship*, 36(4):156–159, November 2007.

[36] R.J. Deibert, J.G. Palfrey, R. Rohozinski, and J. Zittrain, editors. *Access Denied: The Practice and Policy of Global Internet Filtering*. MIT Press, February 2008.

[37] C. Hogg. China restores Xinjiang internet. *BBC News*, May 2010.

[38] S. Wang and S. Nagaraja. *Pulling the Plug: A Technical Review of the Internet Shutdown in Burma*. OpenNet Initiative, 2007.

[39] Electronic Privacy Information Center and Privacy International. *Privacy and Human Rights 2006: An International Survey of Privacy Laws and Developments*. Washington, DC: Electronic Privacy Information Center, December 2007.

[40] M. Rogers and S. Bhatti. Private peer-to-peer networks. In X. Shen, H. Yu, J. Buford, and M. Akon, editors, *Handbook of Peer-to-Peer Networking*. Springer, 2010.

[41] M. Rogers and S. Bhatti. Cooperation under scarcity: The sharer's dilemma. In *Proceedings of the 2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2008), Bremen, Germany*, volume 5127 of *Lecture Notes in Computer Science*, pages 28–39, July 2008.

[42] M. Rogers and S. Bhatti. A lightweight mechanism for dependable communication in untrusted networks. In *Proceedings of the 37th International Conference on Dependable Systems and Networks (DSN 2007), Edinburgh, UK*, pages 430–439, June 2007.

[43] M. Rogers and S. Bhatti. An adaptive routing protocol for censorship-resistant communication. In *2nd International Conference on Information Society (i-Society 2007), Merrillville, IN, USA*, October 2007.

[44] P. Baran. On distributed communications networks. Paper P-2626, RAND Corporation, September 1962.

[45] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[46] D.J. Watts and S.H. Strogatz. Collective dynamics of small world networks. *Nature*, 393(6684):440–442, June 1998.

[47] S. Milgram. The small world problem. *Psychology Today*, 1:61–67, May 1967.

[48] P.S. Dodds, R. Muhamad, and D.J. Watts. An experimental study of search in global social networks. *Science*, 301(5634):827–829, August 2003.

[49] J. Leskovec and E. Horvitz. Worldwide buzz: Planetary-scale views on an instant-messaging network. Technical Report MSR-TR-2006-186, Microsoft Research, June 2007.

[50] D.J. Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.

[51] J.M. Kleinberg. Navigation in a small world. *Nature*, 406(6798):845, August 2000.

[52] J.M. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *32nd ACM Symposium on Theory of Computing, Portland, OR, USA*, May 2000.

[53] D. Liben-Nowell, J. Nowak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences USA*, 102(33):11623–11628, August 2005.

[54] M.L. Goldstein, S.A. Morris, and G.G. Yen. Problems with fitting to the power-law distribution. *European Physical Journal B*, 41(2):255–258, September 2004.

[55] A.L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[56] N. Sarshar and V. Roychowdhury. Scale-free and stable structures in complex ad hoc networks. *Physical Review E*, 69(026101), 2004.

[57] H. Bauke and D. Sherrington. Local attachment in networks under churn, May 2007. arXiv preprint, available from `http://arxiv.org/abs/0706.0018`.

[58] R. Albert, H. Jeong, and A.L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):387–482, 2000.

[59] A.E. Motter, T. Nishikawa, and Y.C. Lai. Range-based attacks on links in scale-free networks: Are long-range links responsible for the small-world phenomenon? *Physical Review E*, 66(065103), 2002.

[60] B. Claerhout. A short overview of IP spoofing: Part I, 1996. Available from `http://staff.washington.edu/dittrich/papers/IP-spoof-1.txt`.

[61] B. Claerhout. A short overview of IP spoofing: Part II, 1997. Available from `http://staff.washington.edu/dittrich/papers/IP-spoof-2.txt`.

[62] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger. Thwarting web censorship with untrusted messenger discovery. In *Proceedings of the 3rd Workshop on Privacy Enhancing Technologies, Dresden, Germany*, volume 2760 of *Lecture Notes in Computer Science*, pages 125–140, March 2003.

[63] P. Bächer, T. Holz, M. Kötter, and G. Wicherski. Know your enemy: Tracking botnets. Technical report, The Honeynet Project, October 2008. Available from `http://www.honeynet.org/papers/bots/`.

[64] J.R. Douceur. The Sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260, March 2002.

[65] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference (CRYPTO '92), Santa Barbara, CA, USA*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147, 1992.

[66] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Network and Distributed System Security Symposium, San Diego, CA, USA*, 1999.

[67] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. In *SIGCOMM 2006, Pisa, Italy*, September 2006.

[68] H. Yu, P.B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social network defense against Sybil attacks. In *IEEE Symposium on Security and Privacy, Oakland, CA, USA*, May 2008.

[69] N. Tran, J. Li, L. Subramanian, and S.S.M. Chow. Brief announcement: Improving social-network-based Sybil-resilient node admission control. In *29th Symposium on Principles of Distributed Computing (PODC 2010), Zurich, Switzerland*, July 2010.

[70] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil nodes using social networks. In *16th Annual Network and Distributed System Security Symposium (NDSS '09), San Diego, CA, USA*, February 2009.

[71] B. Viswanath, A. Post, K.P. Gummadi, and A. Mislove. An analysis of social network-based Sybil defenses. In *SIGCOMM 2010, New Delhi, India*, August 2010.

[72] E. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[73] A. Singh, T.W. Ngan, P. Druschel, and D.S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *INFOCOM 2006, Barcelona, Spain*, April 2006.

[74] L. Zhou and Z.J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, November 1999.

[75] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.

[76] Y. Desmedt. Some recent research aspects of threshold cryptography. In *Proceedings of the 1st International Information Security Workshop (ISW '97), Tatsunokuchi, Japan*, volume 1396 of *Lecture Notes in Computer Science*, pages 158–173, 1997.

[77] D. Quercia, S. Hailes, and L. Capra. TATA: Towards anonymous trusted authentication. In *Proceedings of the 4th International Conference on Trust Management (iTrust 2006), Pisa, Italy*, pages 313–323, May 2006.

[78] Napster website, archived March 2001, available from `http://web.archive.org/web/20010322195829/www.napster.com/index.html`.

[79] Aimster website, archived August 2001, available from `http://web.archive.org/web/20010801151157/aimster.com/`.

[80] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: Search methods. Technical Report UNSW-EE-P2P-1-1, University of New South Wales, September 2004.

[81] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. In *21st International Conference on Data Engineering, Tokyo, Japan*, 2005.

[82] E. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(2), 2005.

[83] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. RFC 3489: STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs), March 2003.

[84] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *USENIX Annual Technical Conference, Anaheim, CA, USA*, April 2005.

[85] S. Guha and P. Francis. Characterization and measurement of TCP traversal through NATs and firewalls. In *Internet Measurement Conference (IMC 2005), Berkeley, CA, USA*, October 2005.

[86] S. Saroiu, P. Krishna Gummadi, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking (MMCN '02)*, January 2002.

[87] D. Stutzbach and R. Rejaie. Towards a better understanding of churn in peer-to-peer networks. Technical Report UO-CIS-TR-04-06, Department of Computer Science, University of Oregon, November 2004.

[88] F.E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *8th International Workshop on Web Content Caching and Distribution, Hawthorne, NY, USA*, September-October 2003.

[89] S. Guha, N. Daswani, and R. Jain. An experimental study of the Skype peer-to-peer VoIP system. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS '06), Santa Barbara, CA, USA*, pages 1–6, February 2006.

[90] N.S. Evans, C. Gauthier-Dickey, and C. Grothoff. Routing in the dark: Pitch black. In *23rd Annual Computer Security Applications Conference (ACSAC), Miami Beach, FL, USA*, December 2007.

[91] The Gnutella protocol specification v0.4. Available from `http://www.stanford.edu/class/cs244b/gnutella_protocol_0.4.pdf`.

[92] J. Ritter. Why Gnutella can't scale. no, really, February 2001. Available from `http://www.darkridge.com/~jpr5/doc/gnutella.html`.

[93] T. Miconi. I jumped in the GnutellaNet and what did i see? lessons from a simple Gnutella network simulation, October 2002. Available from `http://thomas.miconi.free.fr/TSN3.pdf`.

[94] S. Jiang, L. Guo, and X. Zhang. LightFlood: An efficient flooding scheme for file search in unstructured peer-to-peer systems. In *Proceedings of the 32nd International Conference on Parallel Processing (ICPP '03), Kaohsiung, Taiwan*, pages 627–635, October 2003.

[95] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th International Conference on Supercomputing, New York, NY, USA*, June 2002.

[96] Q. Lv, S. Ratnasamy, and S. Shenker. Can heterogeneity make Gnutella scalable? In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, volume 2429 of *Lecture Notes in Computer Science*, pages 94–103, March 2002.

[97] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *SIGCOMM 2003, Karlsruhe, Germany*, August 2003.

[98] D. Tsoumakos and N. Roussopoulos. Adaptive probabilistic search (APS) for peer-to-peer networks. Technical Report CS-TR-4451, Computer Science Department, University of Maryland, February 2003.

[99] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *INFOCOM 2004, Hong Kong*, March 2004.

[100] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *22nd International Conference on Distributed Computing Systems, Vienna, Austria*, July 2002.

[101] A. Fisk. Gnutella dynamic query protocol v0.1, May 2003. Archived March 2005, available from `http://web.archive.org/web/20050316171004/www.limewire.com/developer/dynamic_query.html`.

[102] B. Yang, P. Vinograd, and H. Garcia-Molina. Evaluating GUESS and non-forwarding peer-to-peer search. In *24th International Conference on Distributed Computing Systems (ICDCS), Tokyo, Japan*, March 2004.

[103] P. Ganesan, Q. Sun, and H. Garcia-Molina. YAPPERS: A peer-to-peer lookup service over arbitrary topology. In *INFOCOM 2003, San Francisco, CA, USA*, March-April 2003.

[104] R. Morselli, B. Bhattacharjee, M.A. Marsh, and A. Srinivasan. Efficient lookup on unstructured topologies. In *24th Annual ACM Symposium on Principles of Distributed Computing, Las Vegas, NV, USA*, July 2005.

[105] M. Roussopoulos and M. Baker. CUP: Controlled update propagation in peer-to-peer networks. In *USENIX Annual Technical Conference, San Antonio, TX, USA*, June 2003.

[106] Konspire2b website, `http://konspire.sourceforge.net/`.

[107] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*, June 2003.

[108] A. Singla and C. Rohrs. Ultrapeers: Another step towards Gnutella scalability, December 2001. Archived February 2005, available from `http://web.archive.org/web/20050209124117/www.limewire.com/developer/Ultrapeers.html`.

[109] T. Hargreaves. The FastTrack protocol, August 2003. Archived November 2006, available from `http://web.archive.org/web/20061130141151/gnunet.org/papers/FAST-TRACK-PROTOCOL`.

[110] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *IEEE International Conference on Data Engineering*, March 2003.

[111] J. Ledlie, J. Taylor, L. Serban, and M. Seltzer. Self-organization in peer-to-peer systems. In *10th ACM SIGOPS European Workshop, Saint-Emilion, France*, September 2002.

[112] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[113] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power-law networks. *Physical Review E*, 64(46135), 2001.

[114] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

[115] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern P2P file-sharing systems. In *Internet Measurement Conference (IMC 2005), Berkeley, CA, USA*, October 2005.

[116] N. Sarshar, P.O. Boykin, and V. Roychowdhury. Scalable percolation search in power law networks. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P 2004), Zurich, Switzerland*, pages 2–9. IEEE Computer Society Press, August 2004.

[117] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, EECS Department, University of California, Berkeley, April 2001.

[118] P. Druschel and A. Rowstron. Pastry: Scalable distributed object location and routing for large-scale peer-to-peer systems. In *18th IFIP/ACM Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany*, November 2001.

[119] C. Plaxton, R. Rajaram, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, June 1997.

[120] C. Law and K.Y. Siu. Distributed construction of random expander networks. In *INFOCOM 2003, San Francisco, CA, USA*, March-April 2003.

[121] M. Castro, P. Druschel, Y.C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. Technical Report MSR-TR-2002-82, Microsoft Research, 2002.

[122] P. Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, volume 2429 of *Lecture Notes in Computer Science*, pages 53–65, March 2002.

[123] D. Stutzbach and R. Rejaie. Improving lookup performance over a widely-deployed DHT. In *INFOCOM 2006, Barcelona, Spain*, April 2006.

[124] J. Falkner, M. Piatek, J.P. John, A. Krishnamurthy, and T. Anderson. Profiling a million user DHT. In *Internet Measurement Conference (IMC 2007), San Diego, CA, USA*, October 2007.

[125] S.A. Crosby and D.S. Wallach. An analysis of BitTorrent's two Kademlia-based DHTs. Technical Report TR-07-04, Department of Computer Science, Rice University, June 2007.

[126] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM 2001, San Diego, CA, USA*, August 2001.

[127] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM 2001, San Diego, CA, USA*, August 2001.

[128] M. Freedman and R. Vingralek. Efficient peer-to-peer lookup based on a distributed trie. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, volume 2429 of *Lecture Notes in Computer Science*, pages 66–75, March 2002.

[129] N.J.A. Harvey, M.B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *4th USENIX Symposium on Internet Technologies and Systems, Seattle, WA, USA*, March 2003.

[130] O. Sandberg. Distributed routing in small-world networks. In *8th Workshop on Algorithm Engineering and Experiments (ALENEX06), Miami, FL, USA*, January 2006.

[131] M. Caesar, M. Castro, E.B. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: Network routing inspired by DHTs. In *SIGCOMM 2006, Pisa, Italy*, September 2006.

[132] B. Ford. Unmanaged internet protocol: Taming the edge network management crisis. In *2nd Workshop on Hot Topics in Networks (HotNets-II), Cambridge, MA, USA*, November 2003.

[133] R. Moskowitz and P. Nikander. RFC 4423: Host identity protocol (HIP) architecture, May 2006.

[134] E. Cohen, A. Fiat, and H. Kaplan. A case for associative peer to peer overlays. In *1st Workshop on Hot Topics in Networks (HotNets-I), Princeton, NJ, USA*, October 2002.

[135] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM 2003, San Francisco, CA, USA*, March-April 2003.

[136] N. Ambastha, I. Baek, S. Gokhale, and A. Mohr. A cache-based resource location approach for unstructured P2P network architectures. In *Graduate Research Conference, Department of Computer Science, Stony Brook University, NY, USA*, May 2003.

[137] V. Cholvi, P. Felber, and E. Biersack. Efficient search in unstructured peer-to-peer networks. In *Proceedings of the 16th ACM Symposium on Parallel Algorithms and Architectures*, pages 271–272, 2004.

[138] H. Cai and J. Wang. Foreseer: A novel, locality-aware peer-to-peer system architecture for keyword searches. In *Proceedings of the 5th International Middleware Conference (Middleware 2004), Toronto, Canada*, volume 3231 of *Lecture Notes in Computer Science*, pages 38–58, 2004.

[139] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips. Tribler: A social-based peer-to-peer system. In *5th International Workshop on Peer-to-Peer Systems (IPTPS '06), Santa Barbara, CA, USA*, February 2006.

[140] M.T. Prinkey. An efficient scheme for query processing on peer-to-peer networks, 2001. Available from `http://aeolusres.homestead.com/files/index.html`.

[141] C. Rohrs. Query routing for the Gnutella network, May 2002. Archived September 2007, available from `http://web.archive.org/web/20070927043455/www.limewire.com/developer/query_routing/keyword+routing.htm`.

[142] S. Joseph. NeuroGrid: Semantically routing queries in peer-to-peer networks. In *International Workshop on Peer-to-Peer Computing, Pisa, Italy*, May 2002.

[143] P.J. Keleher, B. Bhattacharjee, and B.D. Silaghi. Are virtualized overlay networks too much of a good thing? In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, volume 2429 of *Lecture Notes in Computer Science*, pages 225–231, March 2002.

[144] B. Silaghi, S. Bhattacharjee, and P. Keleher. Routing in the TerraDir directory service. In *SPIE ITCom, Boston, MA, USA*, July 2002.

[145] C. Peery, F.M. Cuenca-Acuna, R.P. Martin, and T.D. Nguyen. Collaborative management of global directories in P2P systems. Technical Report DCS-TR-510, Department of Computer Science, Rutgers University, November 2002.

[146] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. Edutella: A P2P networking infrastructure based on RDF. In *11th International World Wide Web Conference (WWW 2002), Honolulu, HW, USA*, May 2002.

[147] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *22nd International Conference on Distributed Computing Systems, Vienna, Austria*, July 2002.

[148] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *Proceedings of the 12th International World Wide Web Conference (WWW 2003), Budapest, Hungary*, pages 536–543, May 2003.

[149] C. Tempich, S. Staab, and A. Wranik. REMINDIN': Semantic query routing in peer-to-peer networks based on social metaphors. In *Proceedings of the 13th International World Wide Web Conference (WWW 2004), New York, NY, USA*, pages 640–649, May 2004.

[150] P. Haase, J. Broekstra, M.Ehrig, M. Menken, P.Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *3rd International Semantic Web Conference, Hiroshima, Japan*, November 2004.

[151] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building internet-scale semantic overlay networks. In *3rd International Semantic Web Conference, Hiroshima, Japan*, November 2004.

[152] A. Löser, C. Tempich, B. Quilitz, W.T. Balke, S. Staab, and W. Nejdl. Searching dynamic communities with personal indexes. In *4th International Semantic Web Conference, Galway, Ireland*, November 2005.

[153] J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in P2P file sharing systems. In *INFOCOM 2005, Miami, FL, USA*, March 2005.

[154] N. Christin, A.S. Weigend, and J. Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *ACM Conference on Electronic Commerce, Vancouver, Canada*, June 2005.

[155] A. Banerjee, M. Faloutsos, and L.N. Bhuyan. The P2P war: Someone is monitoring your activities! In *Proceedings of the 6th International IFIP-TC6 Networking Conference (NETWORKING 2007), Atlanta, GA, USA*, volume 4479 of *Lecture Notes in Computer Science*, pages 1096–1107, May 2007.

[156] Electronic Frontier Foundation. RIAA v. the people: Four years later, August 2007. Available from `http://w2.eff.org/IP/P2P/riaa_at_four.pdf`.

[157] S. Le Blond, A. Legout, F. Le Fessant, W. Dabbous, and M.A. Kaafar. Spying the world from your laptop: Identifying and profiling content providers and big downloaders in BitTorrent. In *3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '10), San Jose, CA, USA*, April 2010.

[158] D. Bricklin. Friend-to-friend networks, August 2000. Available from `http://www.bricklin.com/f2f.htm`.

[159] L. Gonze. Friendnet, December 2002. Available from `http://www.oreillynet.com/pub/wlg/2428`.

[160] P. Biddle, P. England, M. Peinado, and B. Willman. The darknet and the future of content protection. In *Proceedings of the 2nd International Workshop on Digital Rights Management (DRM 2002), Washington, DC, USA*, volume 2696 of *Lecture Notes in Computer Science*, pages 155–176, 2003.

[161] J. Cederlöf. Web of trust statistics and pathfinder. Available from `http://www.lysator.liu.se/~jc/wotsap/`.

[162] D. Knoke and J.H. Kuklinski. *Network Analysis*, volume 28 of *Quantitative Applications in the Social Sciences*. London: Sage Publications, October 1982.

[163] S. Nagaraja and R. Anderson. The topology of covert conflict. Technical Report UCAM-CL-TR-637, University of Cambridge Computer Laboratory, July 2005.

[164] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy, Oakland, CA, USA*, May 2009.

[165] S. Nagaraja. Anonymity in the wild: Mixes on unstructured networks. In *Proceedings of the 7th Workshop on Privacy Enhancing Technologies (PET 2007), Ottawa, Canada*, pages 254–272, June 2007.

[166] J. Li and F. Dabek. F2F: Reliable storage in open networks. In *5th International Workshop on Peer-to-Peer Systems (IPTPS '06), Santa Barbara, CA, USA*, February 2006.

[167] C. Zhang, P. Dhungel, D. Wu, Z. Liu, and K.W. Ross. BitTorrent darknets. In *INFOCOM 2010, San Diego, CA, USA*, March 2010.

[168] J. Udell, N. Asthagiri, and W. Tuvell. Security. In A. Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 18. O'Reilly, March 2001. This chapter describes Groove.

[169] WASTE website, `http://waste.sourceforge.net/`.

[170] M. Ek, F. Hultin, and J. Lindblom. WASTE peer-to-peer protocol, March 2005. Reverse-engineered protocol documentation, available from `http://prdownloads.sourceforge.net/j-waste/waste_documentation-1.1.pdf?download`.

[171] T. Strufe and D. Reschke. Efficient content distribution in semi-decentralized peer-to-peer networks. In *Proceedings of the 8th International Netties Conference, Ilmenau, Germany*, pages 33–38, September-October 2002.

[172] Shinkuro website, `http://shinkuro.com/`.

[173] PowerFolder website, `http://www.powerfolder.com/`.

[174] Octopod website, `http://sysnet.ucsd.edu/octopod/`.

[175] L. Deri and R. Andrews. N2N: A layer two peer-to-peer VPN. In *Proceedings of the 2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2008), Bremen, Germany*, volume 5127 of *Lecture Notes in Computer Science*, pages 53–64, July 2008.

[176] NeoModus Direct Connect website, archived June 2005, available from `http://web.archive.org/web/20050627012020/www.neo-modus.com/`.

[177] DC++ website, `http://dcplusplus.sourceforge.net/`.

[178] B.C. Popescu, B. Crispo, and A.S. Tanenbaum. Safe and private data sharing with Turtle: Friends team-up and beat the system. In *12th International Workshop on Security Protocols, Cambridge, UK*, April 2004.

[179] P. Matějka. Security in peer-to-peer networks. Master's thesis, Department of Software Engineering, Charles University, Prague, December 2004.

[180] SockeToome website, archived July 2007, available from `http://web.archive.org/web/20070730103712/www.ziggy.speedhost.com/bdsock.html`.

[181] Easter website, `http://easta.sourceforge.net/`.

[182] anoNet website, http://anonet.org/.

[183] R. Figueiredo, O. Boykin, P. St. Juste, and D. Wolinsky. Social VPNs: Integrating overlay and social networks for seamless P2P networking. In *Workshop on Collaborative Peer-to-Peer Systems (COPS), Rome, Italy*, June 2008.

[184] Social VPN website, https://socialvpn.wordpress.com/.

[185] CSpace website, http://www.cspace.in/.

[186] S. Buchegger, D. Schiöberg, L.H. Vu, and A. Datta. PeerSON: P2P social networking – early experiences and insights. In *2nd Workshop on Social Network Systems (SocialNets 2009), Nürnburg, Germany*, March 2009.

[187] Retroshare website, http://retroshare.sourceforge.net/.

[188] Galet website, http://galet.sourceforge.net/.

[189] Alliance website, http://www.alliancep2p.com/.

[190] Cryptic6 website, http://cryptic6.sourceforge.net/.

[191] L.A. Fredriksen. Securing private peer-to-peer networks. Master's thesis, University of Tromsø, August 2007.

[192] Gazzera website, https://code.google.com/p/gazzera/.

[193] Tonika website, http://5ttt.org/.

[194] D.N. Tran, F. Chiang, and J. Li. Friendstore: Cooperative online backup using trusted nodes. In *1st International Workshop on Social Network Systems (SocialNets 2008), Glasgow, UK*, April 2008.

[195] E. Vasserman, R. Jansen, J. Tyra, N. Hopper, and Y. Kim. Membership-concealing overlay networks. In *16th ACM Conference on Computer and Communications Security (CCS 2009), Chicago, IL, USA*, November 2009.

[196] R. Anderson. The Eternity Service. In *Proceedings of the 1st International Conference on the Theory and Applications of Cryptology, Prague, Czech Republic*, pages 242–252, October 1996.

[197] Wikileaks website, http://wikileaks.org/.

[198] The Benetech Initiative. Martus human rights bulletin system. Available from http://www.martus.org/dl/martus-overview.pdf.

[199] R. Dingledine, M.J. Freedman, and D. Molnar. The Free Haven project: Distributed anonymous storage service. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, volume 2009 of *Lecture Notes in Computer Science*, pages 67–95, July 2000.

[200] M. Waldman, A. Rubin, and L.F. Cranor. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, August 2000.

[201] M. Waldman, L.F. Cranor, and A. Rubin. Publius. In A. Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 11. O'Reilly, March 2001.

[202] M. Waldman and D. Mazières. Tangler: A censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001), Philadelphia, PA, USA*, pages 126–135, November 2001.

[203] A. Stubblefield and D.S. Wallach. Dagster: Censorship-resistant publishing without replication. Technical Report TR01-380, Department of Computer Science, Rice University, July 2001.

[204] Owner-Free File System website, http://offsystem.sourceforge.net/.

[205] Z. Wilcox-O'Hearn and B. Warner. Tahoe – the least-authority filesystem. In *4th International Workshop on Storage Security and Survivability (StorageSS 2008), Alexandria, VA, USA*, October 2008.

[206] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, June 1960.

[207] D.X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA*, pages 44–55, May 2000.

[208] R.S. Peterson, B. Wong, and E.G. Sirer. Blindfold: A system to "see no evil" in content discovery. In *9th International Workshop on Peer-to-Peer Systems (IPTPS '10), San Jose, CA, USA*, April 2010.

[209] L. von Ahn, M. Blum, N.J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2003), Warsaw, Poland*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311, May 2003.

[210] B. Kantor and P. Lapsley. RFC 977: Network news transfer protocol, February 1986.

[211] R. Allbery and C. Lindsey. RFC 5537: Netnews architecture and protocols, November 2009.

[212] A. Back. The eternity service. *Phrack*, 7(51), September 1997.

[213] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *8th Workshop on Hot Topics in Operating Systems, Elmau, Germany*, May 2001.

[214] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R.P. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI '02), Boston, MA, USA*, pages 1–14, December 2002.

[215] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An architecture for global-scale persistent storage. In *9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), Cambridge, MA, USA*, November 2000.

[216] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *18th ACM Symposium on Operating Systems Principles, Banff, Canada*, October 2001.

[217] HiveCache website, archived October 2003, available from `http://web.archive.org/web/20031004124554/hivecache.com/`.

[218] Mnet website, `http://mnet.sourceforge.net/doc.php`.

[219] L.P. Cox, C.D. Murray, and B.D. Noble. Pastiche: Making backup cheap and easy. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI '02), Boston, MA, USA*, pages 285–298, December 2002.

[220] S. Hand and T. Roscoe. Mnemosyne: Peer-to-peer steganographic storage. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, volume 2429 of *Lecture Notes in Computer Science*, pages 130–140, March 2002.

[221] M.O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, April 1989.

[222] A. Serjantov. Anonymizing censorship resistant systems. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, volume 2429 of *Lecture Notes in Computer Science*, pages 111–120, March 2002.

[223] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *5th Symposium on Operating Systems Design and Implementation, Boston, MA, USA*, December 2002.

[224] P. Wang, N. Hopper, I. Osipkov, and Y. Kim. Myrmic: Secure and robust DHT routing. Research Report 2006/20, Digital Technology Center, University of Minnesota, November 2006.

[225] F. Kuhn, S. Schmid, and R. Wattenhofer. A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In *4th International Workshop on Peer-to-Peer Systems (IPTPS '05), Ithaca, NY, USA*, February 2005.

[226] B. Awerbuch and C. Scheideler. Towards scalable and robust overlay networks. In *6th International Workshop on Peer-to-Peer Systems (IPTPS '07), Tampa, FL, USA*, February 2007.

[227] A. Fiat and J. Saia. Censorship resistant peer-to-peer content addressable networks. In *13th Annual ACM Symposium on Discrete Algorithms, San Francisco, CA, USA*, 2002.

[228] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, volume 2429 of *Lecture Notes in Computer Science*, pages 261–269, March 2002.

[229] P. Wang, J. Tyra, E. Chan-Tin, T. Malchow, D. Foo Kune, N. Hopper, and Y. Kim. Attacking the Kad network. In *4th International Conference on Security and Privacy in Communication Networks (SecureComm 2008), Istanbul, Turkey*, September 2008.

[230] A. Kapadia and N. Triandopoulos. Halo: High-assurance locate for distributed hash tables. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS '08), San Diego, CA, USA*, pages 61–79, February 2008.

[231] S. Marti, P. Ganesan, and H. Garcia-Molina. SPROUT: P2P routing with social networks. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004), Heraklion, Crete, Greece*, volume 3268 of *Lecture Notes in Computer Science*, pages 425–435, 2004.

[232] G. Danezis, C. Lesniewski-Laas, M.F. Kaashoek, and R. Anderson. Sybil-resistant DHT routing. In *10th European Symposium on Research in Computer Security (ESORICS 2005), Milan, Italy*, September 2005.

[233] C. Lesniewski-Laas. A Sybil-proof one-hop DHT. In *1st International Workshop on Social Network Systems (SocialNets 2008), Glasgow, UK*, April 2008.

[234] N. Borisov. Computational puzzles as Sybil defenses. In *Proceedings of the 6th International Conference on Peer-to-Peer Computing (P2P 2006), Cambridge, UK*, pages 171–176, September 2006.

[235] R. Endsuleit and T. Mie. Censorship-resistant and anonymous P2P filesharing. In *Proceedings of the 1st International Conference on Availability, Reliability and Security (ARES 2006), Vienna, Austria*, pages 58–65, April 2006.

[236] G. Danezis and R. Anderson. The economics of censorship resistance. In *3rd Annual Workshop on Economics of Information Security, University of Minnesota, MN, USA*, May 2004.

[237] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *SIGCOMM 1994, London, UK*, August 1994.

[238] T. Clausen, P. Jacquet, A. Laouiti, P. Mühlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proceedings of the 5th International Multi-Topic Conference (INMIC 2001), Lahore, Pakistan*, pages 62–68, December 2001.

[239] D.B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, USA*, pages 158–163, December 1994.

[240] D.B. Johnson, D.A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In C.E. Perkins, editor, *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[241] C.E. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99), New Orleans, LA, USA*, pages 90–100, February 1999.

[242] Z.J. Haas. A routing protocol for the reconfigurable wireless networks. In *Proceedings of the 6th International Conference on Universal Personal Communications (ICUPC '97), San Diego, CA, USA*, pages 562–566, October 1997.

[243] W. Peng and X.C. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2000), Boston, MA, USA*, pages 129–130, August 2000.

[244] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical Report RR-3898, INRIA, February 2000.

[245] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *7th International Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[246] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. Technical Report IC/2002/54, EPFL, July 2002.

[247] J. Wu and H. Li. On calculating connected dominating sets for efficient routing in ad hoc wireless networks. In *3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Seattle, WA, USA*, August 1999.

[248] B. Das and V. Bharghavan. Routing in ad hoc networks using minimum connected dominating sets. In *IEEE International Conference on Communications*, June 1997.

[249] C.E. Perkins, editor. *Ad Hoc Networking*. Addison-Wesley, 2001.

[250] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network*, 16(4):11–21, July 2002.

[251] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th International Conference on Mobile Computing and Networking (MobiCom), Dallas, TX, USA*, pages 85–97, October 1998.

[252] S.J. Lee, M. Gerla, and C.K. Toh. A simulation study of table-driven and on-demand routing protocols for mobile ad hoc networks. *IEEE Network*, 13(4):48–54, July 1999.

[253] M. Günes, U. Sorges, and I. Bouazizi. ARA: The ant-colony based routing algorithm for MANETs. In *Proceedings of the 31st International Conference on Parallel Processing Workshops (ICPP 2002 Workshops), Vancouver, BC, Canada*, pages 79–85, August 2002.

[254] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. Technical Report HPL-96-76, HP Labs, May 1996.

[255] E. Bonabeau, F. Héneaux, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in telecommunications networks with ant-like agents. In *Proceedings of the 2nd International Workshop on Intelligent Agents for Telecommunications Applications (IATA '98), Paris, France*, volume 1437 of *Lecture Notes in Computer Science*, pages 60–72, July 1998.

[256] G. DiCaro and M. Dorigo. Ant colonies for adaptive routing in packet-switched communications networks. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, Amsterdam, The Netherlands*, volume 1498 of *Lecture Notes in Computer Science*, pages 673–682, September 1998.

[257] S. Marwaha, C.K. Tham, and D. Srinivasan. Mobile agents based routing protocol for mobile ad hoc networks. In *IEEE Global Telecommunications Conference (GLOBECOM 2002), Taipei, Taiwan*, November 2002.

[258] M. Roth and S. Wicker. Termite: Emergent ad-hoc networking. In *2nd Mediterranean Ad Hoc Networking Workshop (MedHocNet 2003), Mahdia, Tunisia*, June 2003.

[259] S. Rajagopalan and C.C. Shen. ANSI: A unicast routing protocol for mobile ad hoc networks using swarm intelligence. In *Proceedings of the International Conference on Artificial Intelligence (ICAI 2005), Las Vegas, NV, USA*, pages 104–110, June 2005.

[260] G. DiCaro, F. Ducatelle, and L.M. Gambardella. AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5), October 2005.

[261] J.A. Boyan and M.L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. *Advances in Neural Processing Systems*, 6:671–678, 1994.

[262] H. Xie, L. Qiu, Y.R. Yang, and Y. Zhang. On self adaptive routing in dynamic environments. In *Proceedings of the 12th International Conference on Network Protocols (ICNP 2004), Berlin, Germany*, pages 12–23, October 2004.

[263] G. Sakellari. The cognitive packet network: A survey. *The Computer Journal*, 53(3):268–279, March 2010.

[264] J. Kelner and P. Maymounkov. Electric routing and concurrent flow cutting. In *20th International Symposium on Algorithms and Computation (ISAAC 2009), Hawaii, USA*, December 2009.

[265] Y. Hu, A. Perrig, and D.B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe '03), San Diego, CA, USA*, pages 30–40, September 2003.

[266] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E.M. Belding-Royer. A secure routing protocol for ad hoc networks. In *10th International Conference on Network Protocols (ICNP 2002), Paris, France*, November 2002.

[267] Y.C. Hu, A. Perrig, and D.B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. Technical Report TR01-384, Department of Computer Science, Rice University, September 2002.

[268] J. Eriksson, S.V. Krishnamurthty, and M. Faloutsos. TrueLink: A practical countermeasure to the wormhole attack in wireless networks. In *14th International Conference on Network Protocols (ICNP 2006), Santa Barbara, CA, USA*, November 2006.

[269] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens. On the survivability of routing protocols in ad hoc wireless networks. In *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005), Athens, Greece*, pages 327–338. IEEE Computer Society Press, 2005.

[270] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, August 1988.

[271] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[272] I. Avramopoulos, H. Kobayashi, and R. Wang. A routing protocol with Byzantine robustness. In *IEEE Sarnoff Symposium*, March 2003.

[273] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Highly secure and efficient routing. In *INFOCOM 2004, Hong Kong*, March 2004.

[274] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Amendment to: Highly secure and efficient routing. Available from `http://www.cs.princeton.edu/~rywang/papers/infocom04b/amendment.pdf`.

[275] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to Byzantine failures. In *Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe '02), Atlanta, GA, USA*, pages 21–30, September 2002.

[276] J. Eriksson, M. Faloutsos, and S.V. Krishnamurthy. Routing amid colluding attackers. In *15th International Conference on Network Protocols (ICNP 2007), Beijing, China*, October 2007.

[277] P. Papadimitratos and Z.J. Haas. Secure message transmission in mobile ad hoc networks. In *Proceedings of the ACM Workshop on Wireless Security (WiSe '03), San Diego, CA, USA*, pages 41–50, September 2003.

[278] B. Awerbuch, D. Holmer, R. Kleinberg, and H. Rubens. Provably competitive adaptive routing. In *INFOCOM 2005, Miami, FL, USA*, March 2005.

[279] L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Palo Alto, CA, USA, 1979.

[280] A. Perrig, R. Canneti, J.D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *CryptoBytes*, 5(2):2–13, 2002.

[281] R. Hauser, T. Przygienda, and G. Tsudik. Reducing the cost of security in link-state routing. In *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 1997.

[282] S. Cheung. An efficient message authentication scheme for link state routing. In *Proceedings of the 13th Annual Computer Security Applications Conference (ACSAC), San Diego, CA, USA*, pages 90–98, December 1997.

[283] P. Papadimitratos and Z.J. Haas. Secure link state routing for mobile ad hoc networks. In *IEEE Workshop on Security and Assurance in Ad Hoc Networks, Orlando, FL, USA*, January 2003.

[284] B. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves. Securing distance-vector routing protocols. In *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 1997.

[285] M.G. Zapata. Secure ad hoc on demand distance vector (SAODV) routing, October 2001. Available from `ftp://manet.itd.nrl.navy.mil/pub/manet/2001-10.mail`.

[286] M.G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe '02), Atlanta, GA, USA*, pages 1–10, September 2002.

[287] Y.C. Hu, D.B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, June 2002.

[288] Y.C. Hu, A. Perrig, and D.B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *8th International Conference on Mobile Computing and Networking (MobiCom)*, September 2002.

[289] P. Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, USA*, January 2002.

[290] J. Jeong, G.Y. Lee, and Z.J. Haas. Prevention of black hole attack using one-way hash chain scheme in ad hoc networks. In *Proceedings of the International Conference on Information Networking (ICOIN 2007), Estoril, Portugal*, pages 546–573, January 2007.

[291] Anonymizer website, `http://www.anonymizer.com/`.

[292] Circumventor website, `http://www.peacefire.org/circumventor/`.

[293] Freegate website, `http://us.dongtaiwang.com/loc/download_en.php`.

[294] GTunnel website, `http://gardennetworks.org/`.

[295] PaperBus website, `http://www.getpaperbus.com/`.

[296] Peekabooty website, `http://sourceforge.net/projects/peekabooty/`.

[297] Psiphon website, `http://www.civisec.org/software/psiphon`.

[298] SafeWeb website, archived December 2001, available from `http://web.archive.org/web/20011214023631/fugu.safeweb.com/sjws/solutions/safeweb.html`.

[299] TriangleBoy website, archived August 2001, available from `http://web.archive.org/web/20010822143132/fugu.safeweb.com/sjws/solutions/triangle_boy.html`.

[300] UltraSurf website, `http://www.ultrareach.com/`.

[301] J. Finkle. U.S. tests technology to break foreign web censorship. *Reuters*, August 2009.

[302] H. Roberts, E. Zuckerman, and J. Palfrey. *2007 Circumvention Landscape Report: Methods, Uses, and Tools*. Berkman Center for Internet and Society, March 2009.

[303] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In *11th USENIX Security Symposium, San Francisco, CA, USA*, August 2002.

[304] R. Dingledine and N. Mathewson. Design of a blocking-resistant anonymity system. Available from `https://svn.torproject.org/svn/projects/design-paper/blocking.pdf`.

[305] S. Köpsell and U. Hillig. How to achieve blocking resistance for existing systems enabling anonymous web surfing. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004), Washington, DC, USA*, pages 47–58, October 2004.

[306] Y. Sovran, A. Libonati, and J. Li. Pass it on: Social networks stymie censors. In *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008), Tampa, FL, USA*, February 2008.

[307] Haystack website, `http://www.haystacknetwork.com/`.

[308] S. Burnett, N. Feamster, and S. Vempala. Chipping away at censorship firewalls with user-generated content. In *19th USENIX Security Symposium, Washington, DC, USA*, August 2010.

[309] R. Clayton, S.J. Murdoch, and R.N.M. Watson. Ignoring the Great Firewall of China. In *6th Workshop on Privacy Enhancing Technologies, Cambridge, UK*, June 2006.

[310] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, volume 2009 of *Lecture Notes in Computer Science*, pages 1–9, July 2000.

[311] A. Pfitzmann and M. Waidner. Networks without user observability. *Computers and Security*, 2(6):158–166, 1987.

[312] R. Dingledine and N. Mathewson. Anonymity loves company: Usability and the network effect. In L.F. Cranor and S. Garfinkel, editors, *Security and Usability*. O'Reilly, August 2005.

[313] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? how attacks on reliability can compromise anonymity. In *14th ACM Conference on Computer and Communications Security (CCS 2007), Alexandria, VA, USA*, October 2007.

[314] G. Danezis and C. Díaz. A survey of anonymous communication channels. Technical Report TR-2008-35, Microsoft Research, January 2008.

[315] M. Edman and B. Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Computing Surveys*, 42(1), 2010.

[316] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[317] L. Sweeney. k-Anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness ad Knowledge-Based Systems*, 10(5):557–570, 2002.

[318] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free mix routes and how to overcome them. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, volume 2009 of *Lecture Notes in Computer Science*, pages 30–45, July 200o.

[319] M.K. Reiter and A.D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and Systems Security*, 1(1):66–92, 1998.

[320] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Proceedings of the 2nd International Workshop on Privacy Enhancing Technologies (PET 2002), San Francisco, CA, USA*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68, April 2003.

[321] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the 2nd International Workshop on Privacy Enhancing Technologies (PET 2002), San Francisco, CA, USA*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53, April 2003.

[322] G. Tóth, Z. Hornák, and F. Vajda. Measuring anonymity revisited. In *Proceedings of the 9th Nordic Workshop on Secure IT Systems, Espoo, Finland*, pages 85–90, November 2004.

[323] I.S. Moskowitz, R.E. Newman, and P.F. Syverson. Quasi-anonymous channels. In *3rd IASTED International Conference on Communication, Network, and Information Security (CNIS 2003), New York City, NY, USA*, December 2003.

[324] Y. Zhu and R. Bettati. Anonymity vs. information leakage in anonymity systems. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS 2005), Columbus, OH, USA*, pages 514–524, June 2005.

[325] V.D. Gligor. A guide to understanding covert channel analysis of trusted systems. Technical Report NCSC-TG-030, National Computer Security Center, November 1993.

[326] K. Chatzikokolakis, T. Chothia, and A. Guha. Statistical measurement of information leakage. In *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2010), Paphos, Cyprus*, volume 6015 of *Lecture Notes in Computer Science*, pages 390–404, March 2010.

[327] H. Chen and P. Malacaria. Quantifying maximal loss of anonymity in protocols. In *Proceedings of the 4th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2009), Sydney, Australia*, pages 206–217, March 2009.

[328] N. Borisov. *Anonymous Routing in Structured Peer-to-Peer Overlays*. PhD thesis, University of California, Berkeley, May 2005.

[329] Y. Deng, J. Pang, and P. Wu. Measuring anonymity with relative entropy. In *Proceedings of the 4th Workshop on Formal Aspects in Security and Trust (FAST '06), Hamilton, ON, Canada*, volume 4691 of *Lecture Notes in Computer Science*, pages 65–79, 2006.

[330] M. Edman, F. Sivrikaya, and B. Yener. A combinatorial approach to measuring anonymity. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI 2007), New Brunswick, NJ, USA*, pages 356–363, May 2007.

[331] J.C. Grégoire and A.M. Hamel. A combinatorial enumeration approach for measuring anonymity, February 2009. arXiv preprint, available from `http://arxiv.org/abs/0902.1663`.

[332] G. Danezis and R. Clayton. Introducing traffic analysis. In A. Acquisti, S. di Vimercati, S. Gritzalis, and C. Lambrinoudakis, editors, *Digital Privacy: Theory, Technologies, and Practices*. Auerbach Publications, 2007.

[333] R.E. Newman, I.S. Moskowitz, P. Syverson, and A. Serjantov. Metrics for traffic analysis prevention. In *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany*, volume 2760 of *Lecture Notes in Computer Science*, pages 48–65, March 2003.

[334] P. Venkitasubramaniam and L. Tong. Anonymous networking with minimum latency in multihop networks. In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA*, pages 18–32, May 2008.

[335] W. Wang, M. Motani, and V. Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008), Alexandria, VA, USA*, October 2008.

[336] C. Díaz, S.J. Murdoch, and C. Troncoso. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *Proceedings of the 10th Privacy Enhancing Technologies Symposium (PETS 2010), Berlin, Germany*, volume 6205 of *Lecture Notes in Computer Science*, pages 184–201, July 2010.

[337] D. Kesdogan, D. Agrawal, and S. Penz. Limits of anonymity in open environments. In *Proceedings of the 5th International Workshop on Information Hiding (IH 2002), Noordwijkerhout, The Netherlands*, volume 2578 of *Lecture Notes in Computer Science*, pages 53–69, October 2002.

[338] D. Kesdogan and L. Pimenidis. The hitting set attack on anonymity protocols. In *Proceedings of the 6th International Workshop on Information Hiding (IH 2004), Toronto, Canada*, pages 326–339, May 2004.

[339] G. Danezis. The statistical disclosure attack. In *Proceedings of the 18th International Conference on Information Security (SEC2003), Athens, Greece*, pages 421–426. Kluwer, 2003.

[340] N. Mathewson and R. Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure, May 2004. Available from `http://www.freehaven.net/doc/e2e-traffic/e2e-traffic.pdf`.

[341] C. Troncoso, B. Gierlichs, B. Preneel, and I. Verbauwhede. Perfect matching disclosure attacks. In *8th Privacy Enhancing Technologies Symposium (PETS '08), Leuven, Belgium*, July 2008.

[342] G. Danezis and C. Troncoso. Vida: How to use Bayesian inference to de-anonymize persistent communications. In *9th Privacy Enhancing Technologies Symposium (PETS '09), Seattle, WA, USA*, August 2009.

[343] N. Feamster and R. Dingledine. Location diversity in anonymity networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004), Washington, DC, USA*, pages 66–76, October 2004.

[344] M. Wright, M. Adler, B. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. In *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 2002.

[345] M. Wright, M. Adler, B. Levine, and C. Shields. Defending anonymous communication against passive logging attacks. In *IEEE Symposium on Security and Privacy, Berkeley, CA, USA*, May 2003.

[346] A. Hintz. Fingerprinting websites using traffic analysis. In *Proceedings of the 2nd International Workshop on Privacy Enhancing Technologies (PET 2002), San Francisco, CA, USA*, volume 2482 of *Lecture Notes in Computer Science*, pages 171–178, April 2003.

[347] M. Liberatore and B.N. Levine. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, VA, USA*, pages 255–263, October 2006.

[348] C. Partridge, D. Cousins, A.W. Jackson, R. Krishnan, T. Saxena, and W.T. Strayer. Using signal processing to analyze wireless data traffic. Technical Memorandum 1321, BBN Technologies, Cambridge, MA, USA, May 2002.

[349] Y. Zhang and V. Paxson. Detecting stepping stones. In *Proceedings of the 9th USENIX Security Symposium, Denver, CO, USA*, pages 171–184, August 2000.

[350] V. Shmatikov and M.H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of the 11th European Symposium on Computer Security (ESORICS 2006), Hamburg, Germany*, volume 4189 of *Lecture Notes in Computer Science*, pages 18–33, September 2006.

[351] B.N. Levine, M.K. Reiter, C. Wang, and M. Wright. Timing attacks in low-latency mix systems (extended abstract). In *Proceedings of the 8th International Financial Cryptography Conference (FC 2004), Key West, FL, USA*, volume 3110 of *Lecture Notes in Computer Science*, pages 251–265, February 2004.

[352] X.Y. Wang and D.S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003), Washington, DC, USA*, pages 20–29, October 2003.

[353] N. Kiyavash, A. Houmansadr, and N. Borisov. Multi-flow attacks against network flow watermarking schemes. In *Proceedings of the 17th USENIX Security Symposium, San Jose, CA, USA*, pages 307–320, July 2008.

[354] A. Houmansadr, N. Kiyavash, and N. Borisov. Multi-flow attack resistant watermarks for network flows. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2009), Taipei, Taiwan*, pages 1497–1500, April 2009.

[355] A. Houmansadr, N. Kiyavash, and N. Borisov. RAINBOW: A robust and invisible non-blind watermark for network flows. In *16th Annual Network and Distributed System Security Symposium (NDSS '09), San Diego, CA, USA*, February 2009.

[356] T. Karagiannis, D. Papagiannaki, and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In *SIGCOMM 2005, Philadelphia, PA, USA*, August 2005.

[357] Y. Gong. Identifying P2P users using traffic analysis, July 2005. Available from `http://www.securityfocus.com/print/infocus/1843`.

[358] Johan Helsingius closes his internet remailer, August 1996. Electronic Frontier Foundation press release, available from `https://w2.eff.org/Censorship/Foreign_and_local/Finland/960830_penet_closure.announce`.

[359] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), February 1981.

[360] A. Serjantov, R. Dingledine, and P. Syverson. From a trickle to a flood: Active attacks on several mix types. In *Proceedings of the 5th International Workshop on Information Hiding (IH 2002), Noordwijkerhout, The Netherlands*, volume 2578 of *Lecture Notes in Computer Science*, pages 36–52, October 2002.

[361] C. Díaz and A. Serjantov. Generalising mixes. In *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany*, volume 2760 of *Lecture Notes in Computer Science*, pages 18–32, March 2003.

[362] D. Kesdogan, J. Egner, and R. Buschkes. Stop-and-go-mixes: Providing probabilistic anonymity in an open system. In *Proceedings of the 2nd International Workshop on Information Hiding (IH '98), Portland, OR, USA*, volume 1525 of *Lecture Notes in Computer Science*, pages 83–98, 1998.

[363] G. Danezis and L. Sassaman. Heartbeat traffic to counter (n-1) attacks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003), Washington, DC, USA*, pages 89–93, October 2003.

[364] I. Goldberg and D. Wagner. TAZ servers and the rewebber network: Enabling anonymous publishing on the world wide web. *First Monday*, 3(4), April 1998.

[365] T.S. Heydt-Benjamin and A. Serjantov ad B. Defend. Nonesuch: A mix network with sender unobservability. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2006), Alexandria, VA, USA*, pages 1–8, October 2006.

[366] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA*, pages 2–15, May 2003.

[367] N. Mathewson. Underhill: A proposed Type 3 nymserver protocol specification, November 2005. Available from `http://mixminion.net/nym-spec.txt`.

[368] G. Danezis. Mix-networks with restricted routes. In *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany*, volume 2760 of *Lecture Notes in Computer Science*, pages 1–17, March 2003.

[369] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems, Mannheim, Germany*, pages 451–463, February 1991.

[370] D. Goldschlag, M. Reed, and P. Syverson. Onion Routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):39–41, February 1999.

[371] P. Boucher, A. Shostack, and I. Goldberg. Freedom System 2.0 architecture, December 2000. Zero Knowledge Systems white paper, available from `http://osiris.978.org/~brianr/crypto-research/anon/www.freedom.net/products/whitepapers/`.

[372] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium, San Diego, CA, USA*, August 2004.

[373] G. Danezis and R. Clayton. Route fingerprinting in anonymous communications. In *Proceedings of the 6th International Conference on Peer-to-Peer Computing (P2P 2006), Cambridge, UK*, pages 69–72, September 2006.

[374] G. Danezis and P. Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In *8th Privacy Enhancing Technologies Symposium (PETS '08), Leuven, Belgium*, July 2008.

[375] P. Mittal, N. Borisov, C. Troncoso, and A. Rial. Scalable anonymous communication with provable security. In *5th USENIX Workshop on Hot Topics in Security (HotSec '10), Washington, DC, USA*, August 2010.

[376] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against anonymous systems. Technical Report CU-CS-1025-07, University of Colorado at Boulder, February 2007.

[377] K. Loesing. Distributed storage for Tor hidden service descriptors, December 2007. Available from `https://tor-svn.freehaven.net/svn/tor/trunk/doc/spec/proposals/114-distributed-storage.txt`.

[378] ScatterChat website, archived September 2007, available from `http://web.archive.org/web/20070908191613/www.scatterchat.com/`.

[379] TorChat website, https://code.google.com/p/torchat/.

[380] L. Øverlier and P. Syverson. Locating hidden servers. In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA*, pages 100–114, May 2006.

[381] K. Bauer, J. Juen, N. Borisov, D. Grunwald, D. Sicker, and D. McCoy. On the optimal path length for Tor. In *3rd Hot Topics in Privacy Enhancing Technologies (HotPETs 2010), Berlin, Germany*, July 2010.

[382] A. Back, U. Moller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proceedings of the 4th International Workshop on Information Hiding (IH 2001), Pittsburgh, PA, USA*, volume 2137 of *Lecture Notes in Computer Science*, pages 245–257, April 2001.

[383] S.J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, USA*, pages 183–195, May 2005.

[384] J. McLachlan and N. Hopper. Don't clog the queue: Circuit clogging and mitigation in P2P anonymity schemes. In *12th International Conference on Financial Cryptography and Data Security (FC 2008), Cozumel, Mexico*, January 2008.

[385] O. Berthold, H. Fedderath, and S. Köpsell. Web mixes: A system for anonymous and unobservable internet acess. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, volume 2009 of *Lecture Notes in Computer Science*, pages 115–129, July 2000.

[386] JAP website, http://anon.inf.tu-dresden.de/index_en.html.

[387] Privoxy website, http://www.privoxy.org/.

[388] J. McLachlan and N. Hopper. On the risks of serving whenever you surf: Vulnerabilities in Tor's blocking resistance design. In *Workshop on Privacy in the Electronic Society (WPES 2009), Chicago, IL, USA*, November 2009.

[389] W. Dai. Two attacks against ZKS Freedom, November 1999. Available from http://weidai.com/freedom-attacks.txt.

[390] N. Hopper, E.Y. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007), Alexandria, VA, USA*, pages 82–91, October 2007.

[391] P. Manils, A. Chaabane, S. Le Blond, M.A. Kaafar, C. Castellucia, A. Legout, and W. Dabbous. Compromising Tor anonymity exploiting P2P information leakage. Technical Report 00471556, INRIA, April 2010.

[392] K. Zetter. Rogue nodes turn Tor anonymizer into eavesdropper's paradise. *Wired*, September 2007.

[393] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with Torsk. In *16th ACM Conference on Computer and Communications Security (CCS 2009), Chicago, IL, USA*, November 2009.

[394] P.P. Tsang, A. Kapadia, C. Cornelius, and S.W. Smith. Nymble: Blocking misbehaving users in anonymizing networks. Technical Report 2008-637, Computer Science Department, Dartmouth College, December 2008.

[395] M.J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC, USA*, November 2002.

[396] M.J. Freedman. A peer-to-peer anonymizing network layer. Master's thesis, Massachusetts Institute of Technology, May 2002.

[397] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-peer based anonymous internet usage with collusion detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002), Washington, DC, USA*, pages 91–102, November 2002.

[398] M. Rennhard and B. Plattner. Practical anonymity for the masses with MorphMix. In *Proceedings of the 8th International Financial Cryptography Conference (FC 2004), Key West, FL, USA*, volume 3110 of *Lecture Notes in Computer Science*, pages 233–250, February 2004.

[399] P. Tabriz and N. Borisov. Breaking the collusion detection mechanism of MorphMix. In *Proceedings of the 6th International Workshop on Privacy Enhancing Technologies (PET 2006), Cambridge, UK*, volume 4258 of *Lecture Notes in Computer Science*, pages 368–383, June 2006.

[400] Z. Brown. Cebolla: Pragmatic IP anonymity. In *Ottawa Linux Symposium, Ottawa, Canada*, June 2002.

[401] I2P website, http://www.i2p2.de/.

[402] Y. Zhu and Y. Hu. TAP: A novel tunneling approach for anonymity in structured P2P systems. In *Proceedings of the International Conference on Parallel Processing (ICPP 2004), Montreal, Canada*, pages 21–28, August 2004.

[403] L. Zhuang, F. Zhou, B.Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *2nd Symposium on Networked Systems Design and Implementation, Boston, MA, USA*, May 2005.

[404] A. Nambiar and M. Wright. Salsa: A structured approach to large-scale anonymity. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, VA, USA*, pages 17–26, October-November 2006.

[405] P. Mittal and N. Borisov. Information leaks in structured peer-to-peer anonymous communication systems. In *15th ACM Conference on Computer and Communications Security (CCS 2008), Alexandria, VA, USA*, October 2008.

[406] Q. Wang, P. Mittal, and N. Borisov. In search of an anonymous and secure lookup. In *17th ACM Conference on Computer and Communications Security (CCS 2010), Chicago, IL, USA*, October 2010.

[407] A. Panchenko, S. Richter, and A. Rache. NISAN: Network information service for anonymization networks. In *16th ACM Conference on Computer and Communications Security (CCS 2009), Chicago, IL, USA*, November 2009.

[408] P. Mittal and N. Borisov. ShadowWalker: Peer-to-peer anonymous communication using redundant structured topologies. In *16th ACM Conference on Computer and Communications Security (CCS 2009), Chicago, IL, USA*, November 2009.

[409] A. Tran, N. Hopper, and Y. Kim. Hashing it out in public: Common failure modes of DHT-based anonymity schemes. In *Workshop on Privacy in the Electronic Society (WPES 2009), Chicago, IL, USA*, November 2009.

[410] G. Danezis, C. Díaz, C. Troncoso, and B. Laurie. Drac: An architecture for anonymous low-volume communications. In *Proceedings of the 10th Privacy Enhancing Technologies Symposium (PETS 2010), Berlin, Germany*, volume 6205 of *Lecture Notes in Computer Science*, pages 202–219, July 2010.

[411] B. Levine and C. Shields. Hordes: A multicast based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[412] S.C. Bono, C.A. Soghoian, and F. Monrose. Mantis: A lightweight, server-anonymity preserving, searchable P2P network. Technical Report TR-2004-01-B-ISI-JHU, Information Security Institute, Johns Hopkins University, June 2004.

[413] P. Muñoz-Gea, J. Malgosa-Sanahuja, P. Manzanares-Lopez, C. Sanchez-Aarnoutse, and J. Garcia-Haro. A low-variance random-walk procedure to provide anonymity in overlay networks. In *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS 2008), Malaga, Spain*, volume 5283 of *Lecture Notes in Computer Science*, pages 238–250, October 2008.

[414] G. Danezis, C. Díaz, E. Kasper, and C. Troncoso. The wisdom of Crowds: Attacks and optimal constructions. In *Proceedings of the 14th European Symposium on Computer Security (ESORICS 2009), St Malo, France*, volume 5789 of *Lecture Notes in Computer Science*, pages 406–423, September 2009.

[415] MUTE website, `http://mute-net.sourceforge.net/`.

[416] T. Chothia. Securing pseudo identities in an anonymous peer-to-peer file-sharing network. In *3rd International Conference on Security and Privacy in Communication Networks (SecureComm 2007), Nice, France*, September 2007.

[417] ANts P2P website, `http://antsp2p.sourceforge.net/`.

[418] N. Bansod, A. Malgi, B. Choi, and J. Mayo. MuON: Epidemic based mutual anonymity. In *13th International Conference on Network Protocols (ICNP 2005), Boston, MA, USA*, November 2005.

[419] Sneakernet website, `https://code.google.com/p/sneakernet/`.

[420] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-preserving P2P data sharing with OneSwarm. In *SIGCOMM 2010, New Delhi, India*, August 2010.

[421] J. Han, Y. Liu, L. Xiao, R. Xiao, and L.M. Ni. A mutual anonymous peer-to-peer protocol design. In *19th International Parallel and Distributed Processing Symposium (IPDPS 2005), Denver, CO, USA*, April 2005.

[422] J. Han and Y. Liu. Rumor Riding: Anonymizing unstructured peer-to-peer systems. In *Proceedings of the 14th International Conference on Network Protocols (ICNP 2006), Santa Barbara, CA, USA*, pages 22–31, November 2006.

[423] S. Katti, J. Cohen, and D. Katabi. Information slicing: Anonymity using unreliable overlays. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI '07), Cambridge, MA, USA*, pages 43–56, April 2007.

[424] T. Ho, M. Médard, J. Shi, M. Effros, and D.R. Karger. On randomized network coding. In *41st Annual Allerton Conference on Communication, Control and Computing, Monticello, IL, USA*, October 2003.

[425] S. Hazel and B. Wiley. Achord: A variant of the Chord lookup service for use in censorship resistant peer-to-peer publishing systems. In *1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, March 2002.

[426] C. O'Donnell and V. Vaikuntanathan. Information leak in the Chord lookup protocol. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P 2004), Zurich, Switzerland*, pages 28–35. IEEE Computer Society Press, August 2004.

[427] G. Ciaccio. Improving sender anonymity in a structured overlay with imprecise routing. In *Proceedings of the 6th International Workshop on Privacy Enhancing Technologies (PET 2006), Cambridge, UK*, volume 4258 of *Lecture Notes in Computer Science*, pages 190–207, June 2006.

[428] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. Wallach. AP3: A cooperative, decentralized service providing anonymous communication. In *11th ACM SIGOPS European Workshop, Leuven, Belgium*, September 2004.

[429] M.F. Kaashoek and D.R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA*, volume 2735 of *Lecture Notes in Computer Science*, pages 98–107, February 2003.

[430] L.A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communicaions Magazine*, December 2009.

[431] I. Clarke. A distributed decentralised information storage and retrieval system. Technical report, Division of Informatics, University of Edinburgh, 1999. Available from `http://freenetproject.org/papers/ddisrs.pdf`.

[432] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66, July 2000.

[433] I. Clarke, S.G. Miller, T.W. Hong, O. Sandberg, and B. Wiley. Protecting free expression online with Freenet. *IEEE Internet Computing*, 6(1), January 2002.

[434] H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve Freenet performance. In *INFOCOM 2002, New York, NY, USA*, June 2002.

[435] I. Clarke. Freenet's next generation routing protocol, July 2003. Available from `http://freenetproject.org/ngrouting.html`.

[436] R. Dhamija. A security analysis of Freenet, December 2000. Archived December 2007, available from `http://web.archive.org/web/20071205125308rn_2/people.ischool.berkeley.edu/~rachna/courses/cs261/paper.html`.

[437] I. Clarke and O. Sandberg. Routing in the dark: Scalable searches in dark P2P networks. In *DefCon 13, Las Vegas, NV, USA*, July 2005.

[438] I. Clarke. Project status update, and request for your help, September 2005. Archived August 2007, available from `http://web.archive.org/web/20070830223351/archives.freenetproject.org/message/20050914.103042.4b8aac35.en.html`.

[439] O. Sandberg and I. Clarke. The evolution of navigable small-world networks. Technical Report 2007:14, Department of Computer Science and Engineering, Chalmers University of Technology, April 2007.

[440] A.Z. Kronfol. *FASD: A Fault-Tolerant, Adaptive, Scalable, Distributed Search Engine*. PhD thesis, Princeton University, May 2002.

[441] K. Bennett, C. Grothoff, T. Horozov, and I. Patrascu. Efficient sharing of encrypted data. In *7th Australasian Conference on Information Security and Privacy (ACISP 2002), Melbourne, Australia*, July 2002.

[442] K. Bennett and C. Grothoff. GAP - practical anonymous networking. In *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany*, volume 2760 of *Lecture Notes in Computer Science*, pages 141–160, March 2003.

[443] D. Kügler. An analysis of GNUnet and the implications for anonymous, censorship-resistant networks. In *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, Germany*, volume 2760 of *Lecture Notes in Computer Science*, pages 161–176, March 2003.

[444] Y. Zhang, W. Liu, W. Lou, and Y. Fang. MASK: Anonymous on-demand routing in mobile ad hoc networks. *IEEE Transactions on Wireless Communications*, 5(9):2376–2385, September 2006.

[445] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. SDAR: A secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN 2004), Tampa, FL, USA*, pages 618–624, November 2004.

[446] R. Song, L. Korba, and G. Yee. AnonDSR: Efficient anonymous dynamic source routing for mobile ad-hoc networks. In *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2005), Alexandria, VA, USA*, pages 32–42, November 2005.

[447] J. Kong and X. Hong. ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003), Annapolis, MD, USA*, June 2003.

[448] J. Kong, X. Hong, and M. Gerla. An identity-free and on-demand source routing scheme against anonymity threats in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 6(8):888–902, August 2007.

[449] B. Zhu, Z. Wan, M.S. Kankanhalli, F. Bao, and R.H. Deng. Anonymous secure routing in mobile ad-hoc networks. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN 2004), Tampa, FL, USA*, pages 102–108, November 2004.

[450] S. Seys and B. Preneel. ARM: Anonymous routing protocol for mobile ad hoc networks. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA 2006), Vienna, Austria*, pages 133–137, April 2006.

[451] S. Dolev and R. Ostrovsky. Xor-trees for efficient anonymous multicast and reception. Technical Report DIMACS 98-54, Rutgers University, 1998.

[452] S. Goel, M. Robson, M. Polte, and E.G. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical Report 2003-1890, Cornell University, February 2003.

[453] E.G. Sirer, M. Polte, and M. Robson. CliqueNet: A self-organizing, scalable, peer-to-peer anonymous communication substrate, December 2001. Available from `http://www.cs.cornell.edu/People/egs/papers/cliquenet-iptp.pdf`.

[454] P. Golle and A. Juels. Dining cryptographers revisited. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2004), Interlaken, Switzerland*, volume 3027 of *Lecture Notes in Computer Science*, pages 456–473, May 2004.

[455] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A protocol for scalable anonymous communication. In *IEEE Symposium on Security and Privacy, Berkeley, CA, USA*, May 2002.

[456] A. Singh, B. Gedik, and L. Liu. Agyaat: Mutual anonymity over structured P2P networks. *Internet Research*, 16(2):189–212, 2006.

[457] M. Backes, M. Hamerlik, A. Linari, M. Maffei, C. Tryfonopoulos, and G. Weikum. Anonymity and censorship resistance in unstructured overlay networks. Technical Report TR-MPI-Clouds08, Max-Planck-Institut für Informatik, 2008.

[458] E. Curtmola, A. Deutsch, K.K. Ramakrishnan, and D. Srivastava. Censorship-resistant publishing. Technical Report CS2010-0956, University of California, San Diego, March 2010.

[459] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or, a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York, NY, USA*, pages 218–229, 1987.

[460] A. Beimel and S. Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 16(1):25–39, 2003.

[461] S. Nielson, S. Crosby, and D. Wallach. A taxonomy of rational attacks. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS '05), Ithaca, NY, USA*, volume 3640 of *Lecture Notes in Computer Science*, pages 36–46, February 2005.

[462] E. Adar and B. Huberman. Free riding on Gnutella. *First Monday*, 5(10), October 2000.

[463] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on Gnutella revisited: The bell tolls? *IEEE Distributed Systems Online*, 6(6), June 2005.

[464] R. Krishnan, M.D. Smith, Z. Tang, and R. Telang. The impact of free-riding on peer-to-peer networks. In *Proceedings of the 37th Hawaii International Conference on System Sciences, Big Island, HW, USA*, pages 199–208, January 2004.

[465] E. Huang, J. Crowcroft, and I. Wassell. Rethinking incentives for mobile ad hoc networks. In *SIGCOMM 2004 Workshop on Incentives and Game Theory in Networked Systems, Portland, OR, USA*, August-September 2004.

[466] J.H. Saltzer, D.P. Reed, and D.D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[467] S.J. Liebowitz and S.E. Margolis. Network externalities (effects). In *New Palgrave Dictionary of Economics and the Law, MacMillan*, 1998.

[468] L. Buttyán and J.P. Hubaux. Nuglets: A virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical report, Swiss Federal Institute of Technology, January 2001.

[469] S. Chandan and C. Hogendorn. The bucket brigade: Pricing and network externalities in peer-to-peer communications networks. Working Paper 2001-001, Department of Economics, Wesleyan University, October 2001.

[470] J. Crowcroft, R. Gibbens, F. Kelly, and S. Östring. Modelling incentives for collaboration in mobile ad hoc networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '03), Sophia-Antipolis, France*, March 2003.

[471] L. Anderegg and S. Eidenbenz. Ad hoc VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *ACM Mobicom*, 2003.

[472] Y. Xue, B. Li, and K. Nahrstedt. Price-based resource allocation in wireless ad hoc networks. In *Proceedings of the 11th International Workshop on Quality of Service (IWQoS 2003), Berkeley, CA, USA*, volume 2707 of *Lecture Notes in Computer Science*, pages 79–96, 2003.

[473] M. Feldman and J. Chuang. Hidden-action in multi-hop routing. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[474] Mojo Nation website, archived January 2002, available from `http://web.archive.org/web/20020124143240/http://www.mojonation.net/`.

[475] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge. Incentives for sharing in peer-to-peer networks. In *Proceedings of the 2nd International Workshop on Electronic Commerce (WELCOM 2001), Heidelberg, Germany*, volume 2232 of *Lecture Notes in Computer Science*, pages 75–86, 2001.

[476] J. Ioannidis, S. Ioannidis, A.D. Keromytis, and V. Prevelakis. Fileteller: Paying and getting paid for file storage. In *Proceedings of the 6th International Financial Cryptography Conference, Southampton, Bermuda*, pages 282–299, March 2002.

[477] C. Li, B. Yu, and K. Sycara. An incentive mechanism for message relaying in peer-to-peer discovery. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[478] V. Vishnumurthy, S. Chandrakumar, and E.G. Sirer. KARMA: A secure economic framework for P2P resource sharing. In *Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*, June 2003.

[479] I. Osipkov, E. Vasserman, N. Hopper, and Y. Kim. Combatting double-spending using cooperative P2P systems. In *27th International Conference on Distributed Computing Systems (ICDCS '07), Toronto, Canada*, June 2007.

[480] D. Figueiredo, J. Shapiro, and D. Towsley. Incentives to promote availability in peer-to-peer anonymity systems. In *13th International Conference on Network Protocols (ICNP 2005), Boston, MA, USA*, November 2005.

[481] E. Androulaki, M. Raykova, S. Srivatsan, A. Stavrou, and S.M. Bellovin. PAR: Payment for anonymous routing. In *8th Privacy Enhancing Technologies Symposium (PETS 2008), Leuven, Belgium*, July 2008.

[482] R. Jansen, N. Hopper, and Y. Kim. Recruiting new Tor relays with BRAIDS. In *17th ACM Conference on Computer and Communications Security (CCS 2010), Chicago, IL, USA*, October 2010.

[483] Trusted Computing Group website, `http://www.trustedcomputinggroup.org/`.

[484] R. Anderson. Trusted computing frequently asked questions, August 2003.

[485] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwahara. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[486] S. Buchegger and J.Y. Le Boudec. Performance analysis of the CONFIDANT protocol (cooperation of nodes fairness in dynamic ad-hoc networks). In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Lausanne, Switzerland*, pages 226–236, June 2002.

[487] P. Michiardi and R. Molva. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the 6th Joint Working Conference on Communications and Multimedia Security, Portoroz, Slovenia*, pages 107–121, September 2002.

[488] S. Buchegger and J.Y. Le Boudec. The effect of rumor spreading in reputation systems for mobile ad hoc networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '03), Sophia-Antipolis, France*, March 2003.

[489] S. Buchegger and J.Y. Le Boudec. A robust reputation system for P2P and mobile ad hoc networks. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[490] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, R. Schmidt, and J. Wu. Advanced peer-to-peer networking: The P-Grid system and its applications. *Praxis der Informationsvevarbeitung und Kommunikation*, 26(3), 2003.

[491] Z. Despotovic and K. Aberer. Maximum likelihood estimation of peers' performance in P2P networks. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[492] S.D. Kamvar, M.T. Schosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *12th International World Wide Web Conference, Budapest, Hungary*, May 2003.

[493] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom), Boston, MA, USA*, pages 255–265, August 2000.

[494] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 3rd Workshop on Economics of Peer-to-Peer Systems, Philadelphia, PA, USA*, pages 128–132, 2005.

[495] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the 5th ACM Conference on Electronic Commerce, New York, NY, USA*, pages 102–111, 2004.

[496] M. Dell'Amico. Neighbourhood maps: Decentralised ranking in small-world p2p networks. In *3rd International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P), Rhodes Island, Greece*, April 2006.

[497] T.W. Ngan, D.S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA*, volume 2735 of *Lecture Notes in Computer Science*, pages 149–159, February 2003.

[498] A. Blanc, Y.K. Liu, and A. Vahdat. Designing incentives for peer-to-peer routing. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[499] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining cooperation in multi-hop wireless networks. In *2nd Symposium on Networked System Design and Implementation, Boston, MA, USA*, May 2005.

[500] M. Ham and G. Agha. ARA: A robust audit to prevent free-riding in P2P networks. In *5th International Conference on Peer-to-Peer Computing (P2P 2005), Konstanz, Germany*, August-September 2005.

[501] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An architecture for secure resource peering. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), Bolton Landing, NY, USA*, pages 133–148, October 2003.

[502] A. Legout, G. Urvoy-Keller, and P. Michiardi. Understanding BitTorrent: An experimental perspective. Technical Report 00000156, INRIA, November 2005.

[503] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent? In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI '07), Cambridge, MA, USA*, pages 1–14, April 2007.

[504] eDonkey2000 website, archived August 2006, available from `http://web.archive.org/web/20060825034330/www.edonkey2000.com/`.

[505] eMule website, `http://www.emule-project.net/`.

[506] Y. Kulbak and D. Bickson. The eMule protocol specification. Technical report, School of Computer Science and Engineering, Hebrew University of Jerusalem, January 2005.

[507] C. Grothoff. An excess-based economic model for resource allocation in peer-to-peer networks. *Wirtschaftsinformatik*, 45(3):285–292, June 2003.

[508] K. Tamilmani, V. Pai, and A.E. Mohr. SWIFT: A system with incentives for trading. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[509] A. Mislove, A. Post, P. Druschel, and K.P. Gummadi. Ostra: Leveraging trust to thwart unwanted communication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08), San Francisco, CA, USA*, pages 15–30, April 2008.

[510] B.F. Cooper and H. Garcia-Molina. Peer-to-peer resource trading in a reliable distributed system. In *1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, March 2002.

[511] B.F. Cooper and H. Garcia-Molina. Bidding for storage space in a peer-to-peer data preservation system. In *22nd International Conference on Distributed Computing Systems, Vienna, Austria*, July 2002.

[512] L.P. Cox and B.D. Noble. Samsara: Honor among thieves in peer-to-peer storage. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA*, pages 120–132, October 2003.

[513] Flůd website, `http://www.flud.org/`.

[514] T. Ackemann, R. Gold, C. Mascolo, and W. Emmerich. Incentives in peer-to-peer and grid networking. Technical report, Department of Computer Science, University College London, February 2004.

[515] P. Gauthier, B. Bershad, and S.D. Gribble. Dealing with cheaters in anonymous peer-to-peer networks. Technical Report 04-01-03, University of Washington, January 2004.

[516] K.G. Anagnostakis and M.B. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 2004), Tokyo, Japan*, pages 524–533. IEEE Computer Society Press, 2004.

[517] T. Bocek, W. Kun, F.V. Hecht, D. Hausheer, and B. Stiller. PSH: A private and shared history-based incentive mechanism. In *Proceedings of the 2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2008), Bremen, Germany*, volume 5127 of *Lecture Notes in Computer Science*, pages 15–27, July 2008.

[518] R. Landa, D. Griffin, R. Clegg, R. Mykoniati, and M. Rio. A sybilproof indirect reciprocity mechanism for peer-to-peer networks. In *Proceedings of INFOCOM 2009, Rio de Janeiro, Brazil*, pages 343–351, April 2009.

[519] Q. Sun and H. Garcia-Molina. SLIC: A selfish link-based incentive mechanism for unstructured peer-to-peer networks. In *24th International Conference on Distributed Computing Systems*, 2004.

[520] K. Lai, M. Feldman, I. Stoica, and J. Chuang. Incentives for cooperation in peer-to-peer networks. In *Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*, June 2003.

[521] V. Srinivasan, P. Nuggehalli, C.F. Chiasserini, and R. Rao. Cooperation in wireless ad hoc networks. In *INFOCOM 2003, San Francisco, CA, USA*, March-April 2003.

[522] M. Félegyházi, J.P. Hubaux, and L. Buttyán. Equilibrium analysis of packet forwarding strategies in wireless ad hoc networks - the dynamic case. Technical Report IC/2003/68, EPFL, November 2003.

[523] A. Urpi, M.A. Bonuccelli, and S. Giordano. Modelling cooperation in mobile ad hoc networks: a formal description of selfishness. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt '03), Sophia-Antipolis, France*, March 2003.

[524] B. O'Higgins, W. Diffie, L. Strawczynski, and R. do Hoog. Encryption and ISDN: A natural fit. In *International Switching Symposium (ISS '87), Phoenix, AZ, USA*, March 1987.

[525] N. Ferguson and B. Schneier. *Practical Cryptography*. Wiley, 2003.

[526] T. Dierks and C. Allen. RFC 2246: The TLS protocol, January 1999.

[527] T. Ylonen and C. Lonvick. RFC 4251: The secure shell (SSH) protocol architecture, January 2006.

[528] R. Gibbons. *Game Theory for Applied Economists*. Princeton University Press, 1992.

[529] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, 2002.

[530] J. Shneidman and D.C. Parkes. Rationality and self-interest in peer to peer networks. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA*, volume 2735 of *Lecture Notes in Computer Science*, pages 139–148, February 2003.

[531] S. Kuhn. Prisoner's dilemma. In E.N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. 2003. Available from `http://plato.stanford.edu/archives/fall2003/entries/prisoner-dilemma/`.

[532] L.M. Wahl and M.A. Nowak. The continuous prisoner's dilemma: I. linear reactive strategies. *Journal of Theoretical Biology*, 200:307–321, 1999.

[533] G. Roberts and T.N. Sherratt. Development of cooperative relationships through increasing investment. *Nature*, 394(6689):175–179, July 1998.

[534] R.M. Dawes. Social dilemmas. *Annual Review of Psychology*, 31:169–193, January 1980.

[535] R.L. Trivers. The evolution of reciprocal altruism. *Quarterly Review of Biology*, 46(1):35–57, March 1971.

[536] R. Axelrod. *The Evolution of Cooperation*. New York: Basic Books, 1984.

[537] M.A. Nowak and K. Sigmund. The alternating prisoner's dilemma. *Journal of Theoretical Biology*, 168(2):219–226, May 1994.

[538] M.A. Nowak and K. Sigmund. Oscillations in the evolution of reciprocity. *Journal of Theoretical Biology*, 137(1):21–26, March 1989.

[539] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster. To share or not to share: An analysis of incentives to contribute in collaborative file sharing environments. In *Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*, June 2003.

[540] M.A. Nowak and R.M. May. Evolutionary games and spatial chaos. *Nature*, 359(6398):826–829, October 1992.

[541] M.A. Nowak, S. Bonhoeffer, and R.M. May. Spatial games and the maintenance of cooperation. *Proceedings of the National Academy of Sciences USA*, 91:4877–4881, July 1994.

[542] J.M. Epstein. Zones of cooperation in the demographic prisoner's dilemma, 1997. Santa Fe Institute Working Paper 97-12-094.

[543] H. Ohtsuki, C. Hauert, E. Lieberman, and M.A. Nowak. A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441:502–505, May 2006.

[544] P. Grim. The greater generosity of the spatialized prisoner's dilemma. *Journal of Theoretical Biology*, 173(4):353–359, April 1995.

[545] M.D. Cohen, R.L. Riolo, and R. Axelrod. The role of social structure in the maintenance of cooperative regimes. *Rationality and Society*, 13:5–32, 2001.

[546] R. Axelrod, R.L. Riolo, and M.D. Cohen. Beyond geography: Cooperation with persistent links in the absence of clustered neighborhoods. *Personal and Social Psychology Review*, 6(4):341–346, 2002.

[547] M.A. Nowak and K. Sigmund. Evolutionary dynamics of biological games. *Science*, 303:793–799, February 2004.

[548] P.D. Taylor and L.B. Jonker. Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, 40:145–156, July 1978.

[549] L.J. Savage. *The Foundations of Statistics*. Wiley, 1954.

[550] M. Feldman and J. Chuang. The evolution of cooperation under cheap pseudonyms. In *7th International IEEE Conference on E-Commerce Technology, Munich, Germany*, July 2005.

[551] S. Siegel. *Nonparametric Statistics for the Behavioural Sciences*. London: McGraw-Hill, 1956.

[552] J. Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[553] X. Wang, Y.L. Yin, and H. Yu. Finding collisions in the full SHA-1. In *25th Annual International Cryptology Conference, Santa Barbara, CA, USA*, August 2005.

[554] IETF Reliable Multicast Transport (RMT) Working Group website, `http://www.ietf.org/html.charters/rmt-charter.html`.

[555] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.

[556] K. Ostrowski, K. Birman, and A. Phanishayee. QuickSilver scalable multicast. Technical Report TR2006-2063, Cornell University, April 2006.

[557] H. Krawczyk, M. Bellare, and R. Canetti. RFC 2104: HMAC: Keyed-hashing for message authentication, February 1997.

[558] S. Kent and R. Atkinson. RFC 2401: Security architecture for the internet protocol, November 1998.

[559] P.W. Yau and C.J. Mitchell. 2HARP: A secure routing protocol to detect failed and selfish nodes in mobile ad hoc networks. In *Proceedings of the 5th World Wireless Congress, San Francisco, CA, USA*, pages 1–6, 2004.

[560] R. Gennaro and P. Rohatgi. How to sign digital streams. In *Proceedings of the 17th Annual Cryptology Conference (CRYPTO '97), Santa Barbara, CA, USA*, volume 1294 of *Lecture Notes in Computer Science*, pages 180–197, 1997.

[561] R. Merkle. A digital signature based on a conventional encryption function. In *Proceedings of the Conference on the Theory and Applications of Cryptographic Techniques (CRYPTO '87), Santa Barbara, CA, USA*, volume 293 of *Lecture Notes in Computer Science*, 1988.

[562] R.J. Anderson, F. Bergadano, B. Crispo, J.H. Lee, C. Manifavas, and R.M. Needham. A new family of authentication protocols. *Operating Systems Review*, 32(4):9–20, October 1998.

[563] P. Zimmermann, A. Johnston, and J. Callas. ZRTP: Media path key agreement for unicast secure RTP, June 2010. IETF Internet Draft.

[564] S. Cheshire, B. Aboba, and E. Guttman. RFC 3927: Dynamic configuration of IPv4 link-local addresses, May 2005.

[565] E. Rosen, A. Viswanathan, and R. Callon. RFC 3031: Multiprotocol label switching architecture, January 2001.

[566] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *ACM SIGCOMM Computer Communications Review*, 37(2):95–98, April 2007.

[567] NS-2 website, `http://www.isi.edu/nsnam/ns/`.

[568] S.A. Golder, D. Wilkinson, and B.A. Huberman. Rhythms of social interaction: Messaging within a massive online network. In *3rd International Conference on Communities and Technologies (CT 2007), East Lansing, MI, USA*, June 2007.

[569] G. Bigwood, D. Rehunathan, M. Bateman, T. Henderson, and S. Bhatti. Exploiting self-reported social networks for routing in ubiquitous computing environments. In *Proceedings of the 1st International Workshop on Social Aspects of Ubiquitous Computing Environments (SAUCE), Avignon, France*, pages 484–489, October 2008.

[570] P.S. Maybeck. *Stochastic Models, Estimation, and Control*. Academic Press, 1979.

[571] S.J. Julier and J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls (AeroSense), Orlando, FL, USA*, April 1997.

[572] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and the elephants). In *SIGCOMM 2003, Karlsruhe, Germany*, August 2003.

[573] I. Aad, J.P. Hubaux, and E.W. Knightly. Denial of service resilience in ad hoc networks. In *ACM Mobicom, Philadelphia, PA, USA*, September 2004.

[574] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *6th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM '05), Taormina, Italy*, June 2005.

[575] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *INFOCOM 2006, Barcelona, Spain*, April 2006.

[576] M. Claypool, R. Kinicki, M. Li, J. Nichols, and H. Wu. Inferring queue sizes in access networks by active measurement. In *Proceedings of the 5th Passive and Active Measurement Workshop (PAM 2004), Antibes Juan-les-Pins, France*, volume 3015 of *Lecture Notes in Computer Science*, pages 227–236, 2004.

[577] CAIDA Skitter project website, `http://www.caida.org/tools/measurement/skitter/`.

Appendix A

The Simulation Framework

This appendix describes the discrete event-based simulator that was used to conduct the simulations presented in Chapter 6. A new simulator was developed for this purpose because existing simulation frameworks either provided too much detail, making it impractical to simulate networks with thousands of nodes, or too little detail, making it impossible to capture realistic effects such as congestion. The simulator was implemented in Java.

Simulation Parameters

The default simulation parameters are summarised in Table A.1. Unless otherwise noted, the simulated networks are classical random graphs containing 1,000 nodes, with an average of 20 neighbours per node. The nodes' periods of uptime and downtime each have a mean duration of two hours, so on average each node spends half its time online. Each online node is the source of one flow at a time on average, with a mean flow length of 1,000 messages and an interval of one second between messages.

In simulations where the attacker controls some fraction of the nodes, the attacker-controlled nodes are selected at random. Similarly, when some of the nodes are free riders they are chosen randomly.

Settling Times

Simulations were allowed to reach a steady state before any measurements were taken. We allowed one hour of simulated time for Darknet routing to reach a steady state, and ten minutes for oracle routing, on-demand routing and flooding. The settling times for Exu routing are discussed in section 6.6.6. For all protocols, the measurement period was one hour of simulated time.

Parameter	Value
Number of nodes (mean online)	1,000 (500)
Mean neighbours per node (online)	20 (10)
Mean uptime duration	2 hours
Mean downtime duration	2 hours
Network topology	Erdös-Rényi
Number of simultaneous flows	500
Mean flow length	1,000 messages
Interval between messages	1 second
Duration of simulation (after settling)	1 hour

Table A.1: Default simulation parameters

Message type	Protocols	Size (bytes)
Data message	All	1,000
Route request	On-demand routing	200
Route reply	On-demand routing	50
Route error	On-demand routing	50
Acknowledgement	Darknet and Exu routing	50

Table A.2: Message sizes

Duplicate Detection

All of the simulated protocols except oracle routing detect and discard duplicate messages by storing the unique identifiers of received messages for a limited time. (In the case of on-demand routing, it is route requests rather than data messages that may be duplicated.) A duplicate detection timeout of 180 seconds was used for flooding, 90 seconds for on-demand routing, and 60 seconds for Darknet and Exu routing. Assertions in the simulator code ensured that these timeouts were adequate to detect all duplicates, except in the denial-of-service attack simulations described in section 6.4, where the attacker's ability to cause mechanisms such as duplicate detection to break down was considered to be an aspect of the attack that should be simulated.

Message and Packet Sizes

The simulator distinguishes between *packets* in the underlying network and *messages* in the overlay. Some of the simulated protocols use only data messages, which carry end-to-end data payloads, while others use additional message types, such as acknowledgements or route requests, to carry routing information. The sizes of these messages are shown in Table A.2.

The cover traffic strategy described in section 6.3 sends packets to online neighbours in round-robin order. The size of each packet is chosen uniformly at random between 1,300 and 1,400 bytes, of which network-layer and transport-layer headers are assumed to occupy 50 bytes. The remaining space is available for messages; any unused space is filled with padding to conceal the size of the payload from external observers.

Network Model

We assume that the access links connecting peer-to-peer nodes to the Internet are the bottleneck links. This is a realistic assumption for current domestic Internet connections, and it allows us to simulate the access links realistically while using a simple model for the wide-area network. Each simulated node has transmission and reception queues that represent the router queues at either end of its access link. A packet's journey time between two neighbouring nodes in the overlay comprises five elements:

1. Time spent in the transmission queue of the sender's access link, which is proportional to the amount of data in the queue.
2. Transmission time on the sender's access link, which is directly proportional to the packet's size and inversely proportional to the sender's upstream bandwidth.
3. Transit time across the wide-area network.
4. Time spent in the reception queue of the receiver's access link, which is proportional to the amount of data in the queue.
5. Reception time on the receiver's access link, which is directly proportional to the packet's size and inversely proportional to the receiver's downstream bandwidth.

Thus the overall latency is affected by congestion on the access links as well as by their bandwidth.

Parameter	Value
Upstream bandwidth	32 kB/s
Downstream bandwidth	128 kB/s
Transmission queue size	16 kB
Reception queue size	64 kB
Wide-area network latency	100 ms

Table A.3: Access link parameters

The transmission and reception queues on each access link are limited in size. The size of the transmission queue is set to 16 kB and the size of the reception queue to 64 kB, based on a study of home broadband connections [576]. When a queue is full, packets that do not fit in the queue are dropped. We do not implement any transport-layer retransmission or congestion control mechanisms, since the overlays being simulated are designed to provide an unreliable datagram service (see section 3.3.1). However, even without explicit congestion control, some of the simulated protocols tend to avoid overloaded links as a side-effect of discarding duplicate messages, since messages that follow multiple paths will arrive later over congested paths than uncongested paths.

In addition to the transmission and reception queues on its access link, each node maintains an internal queue of messages that are waiting for transmission to each neighbour. (Note that these queues hold messages rather than packets.) The size of each queue is 4 kB, which was found to be optimal for the reliability of the flooding protocol; the queue size has little effect on the other protocols, except when they are under attack. As with the transmission and reception queues, messages that do not fit in a neighbour's message queue are dropped.

In the scenarios we have simulated, the cover traffic strategy ensures that the transmission queues of access links are never congested, since each node transmits packets only as quickly as its link can handle them. Congestion can still occur in the access links' reception queues, however, and in the nodes' internal message queues.

Each node's access link has 32 kB/s of upstream bandwidth and 128 kB/s of downstream bandwidth, which are meant to represent a typical domestic broadband connection. For the scale-free networks with proportional bandwidth described in section 6.3.6, the bandwidths were adjusted to make each node's upstream and downstream bandwidth proportional to its degree, while keeping the mean upstream and downstream bandwidth unchanged.

For simplicity we assume that the transit time across the wide-area network is 100 ms for any pair of nodes. A more realistic distribution of transit times could be generated from the Skitter dataset if required [577].