Universal Cloud Classification (UCC) and its Evaluation in a Data Center Environment

Sebastian Jeuk
Cisco Systems, San Jose, USA
& UCL, London, UK
Email: sjeuk@cisco.com

Gonzalo Salgueiro
Cisco Systems
Research Triangle Park, USA
Email: gsalguei@cisco.com

Shi Zhou
Department of Computer Science
University College London (UCL), UK
Email: s.zhou@ucl.ac.uk

***Abstract*—Classification ambiguity in Cloud Computing has a catastrophic impact on cloud providers, their services and tenants. It limits the application of various network services to traffic either inside or outside a cloud. This is because IP addresses, VLANs and other transport-level technologies lack the functionality to cope with the highly dynamic, scalable and virtualized environment of cloud-enabled data centers. In this paper, we present the prototype design and discuss its features. We also evaluate the UCC proposal in a cloud-enabled data center environment. Our examination of the compatibility, performance and usability of UCC shows that this scheme is not only feasible, easy to implement, but also has significant advantages over other classification techniques. For example, it enables highly desirable functionality such as traffic volume-based data center billing. Imminent follow-up efforts include scalability testing of UCC on the open Internet. We are confident the UCC scheme can provide a long-term, practical and flexible solution for cloud classification with significant benefits.**

I. Introduction

Cloud Computing has emerged as a new IT paradigm over the last couple of years. Distinguished by its on-demand, self-service, broad network access, resource pooling and rapid elasticity, Cloud Computing enables virtualized network, compute and storage resources. Multi-tenancy, as one of the big advantages of Cloud Computing, defines the way resources are shared among services and tenants. Cloud environments suffer from ambiguity in identifying their services and tenants. Traffic from different cloud providers can not be distinguished easily on the Internet. Imagine an enterprise company trying to filter traffic according to its cloud provider or service affiliation. Filters are simply not able to obtain the provider, service and tenant identities from network packets without leveraging deep-packet inspection and risking a significant increase in latency.

To address this profound challenge, we introduce the Universal Cloud Classification (UCC) scheme that enables identification of cloud providers, their services, and tenants on the network layer. This proposal defines three optional IDs that are incorporated into an IPv6 extension header in a cloud and/or across the Internet. By segregating traffic on Layer 3, UCC enables fine-grained network services that leverage the IDs for routing decisions, security policies, or billing based on network utilization.

In this paper, we leverage the Cloud-ID/Service-ID and Tenant-ID and realize them in a prototype reflecting a data center. We outline the suggested implementation method using the IPv6 hop-by-hop option extension header. The concept is validated and evaluated against carefully selected criteria to gather scientifically meaningful and accurate results.

Our contribution in this paper is fivefold: (1) Update and consolidate the UCC concept originally introduced in previous work [1]. (2) Discuss in-depth the advantages and security features of the UCC. (3) Evaluate the feasibility and practicality of the proposal in a real cloud environment. (4) Compare the performance and impact of UCC against existing deep packet inspections mechanisms. (5) Propose a UCC-based approach for cloud billing based on network utilization.

II. Background

Identities, a way to distinguish entities, is a key element in our day-to-day lives. It distinguishes one from another and allows third-parties to associate us with our assets. This type of identification is not given in Cloud Computing due to the lack of adequate network-based classification mechanisms.

A. Ambiguity in Cloud Computing

Cloud Computing comes with many different definitions for their applications and users. Ambiguity in cloud environments is multi-fold. The first ambiguity is described by how a service is defined. Here, an application within a Cloud Provider is called a service. However, the cloud providers network can not distinguish services from services run on top of other services (e.g. SaaS on top of IaaS). Secondly, a tenant is typically defined as a user of a service (here, we ignore other meanings of tenants (for example projects in OpenStack)). That being said, a service run on top of another service can be considered a tenant of that particular service. This ambiguity makes it extremely difficult to uniquely identify services and their tenants in cloud environments. We believe that the new multi-layered service and tenant relationship is one of the most complex tasks to handle using todays technologies. In the next sub-section we highlight some of the state-of-the-art technology shortcomings. We focus our background analysis on IPs, VLANs and VxLANs as a way to currently segregate networks

B. Limitations of existing classification

1) Layer 2 segregation: Virtual LANs have been introduced to separate layer 2 traffic in a network environment. The header is added after the typical Ethernet II frame, known as 802.1q. Defined by IEEE, this standard adds an ID to enable 4096 traffic groups. Even though introduced for legacy networks, VLANs are still heavily used in Data Centers to segregate traffic. In a cloud environment this can lead to ambiguity as VLAN IDs are often re-used for traffic from multiple applications. Besides the limited address space, VLAN classifications are lost at layer-2/layer-3 boundaries. Developed for a static environment with a limited scale, VLANs can not cope with the dynamic, ever changing needs of Cloud Data Centers.

2) Layer 3 and Layer 4 segregation: Multiple segregation approaches are used on layer-3 and layer-4. We highlight IP addresses, port numbers and extensible VLANs and outline their shortcomings in classifying cloud traffic.

IP addresses, as a critical component of network traffic, are used to identify source and destination of a traffic flow. A major shortcoming of IPv4 is its limited address space. To work around this we distinguish between private and public IP addresses. Services within a Cloud environment are typically addressed by private IPs. These are masked using Network Address Translation (NAT) to a pool of external addresses. This causes ambiguity as the pool is re-used for all services offered by a cloud provider.

Port number both used with TCP and UDP allow the identification of applications in combination with IP addresses. Cloud Providers however often host multiple services that leverage the same application port. Hence, port numbers are not suitable to uniquely identify services throughout the network.

Virtual extensible LANs (VxLANs) is a MAC-in-UDP encapsulation proposed to eliminate some of the constraints described earlier. The encapsulation enables L2 subnets to span physical L3 IP networks. Even though the address space of VxLANs is large (24 bit with 16 million VxLAN segments) it does not fully solve the identification problem. VxLANs are solely used for traffic originating and terminating within Cloud Data Centers. Therefore VxLANs cannot be used to identify services and their tenants within the Internet.

3) Summary: This newly introduced three-layered hierarchical model (for now we ignore provider- and service-stacking) in cloud environments is not compatible with current network classification mechanisms. These technologies don't understand the concepts of Cloud Providers, Services or Tenants but rather perform network segmentation based on IPs, VLANs or VxLANs. However, cloud-enabled Data Centers are currently bound to those technologies due to lack of alternatives. Separating Services and their Tenants using IP addresses, VLANs, port-numbers or VxLANs results in limitations such as: (1) No end-to-end tracking of identification (often removed or changed at Layer 2 or Layer 3). (2) Identifiers used for multiple services or tenants. (3) Lack of per-service or per-tenant policies. (4) Network utilization not per-service/-tenant. For a more detailed description of legacy classification mechanisms and their drawbacks and limitations in cloud environments refer to the original document introducing UCC [1].

III. Universal Cloud Classification (UCC)

To tackle the current classification limitations seen in cloud enabled data centers we propose both Layer-2 [2] and Layer-3 IDs that are used to identify Cloud Providers, their Services and Tenants both within and outside a cloud environment. To maintain end-to-end cloud service and tenant isolation and classification we propose newly defined Cloud-IDs, Service-IDs, and Tenant-IDs carried within the Layer 3 protocols. A possible implementation for IPv6 is leveraging extension headers. IPv6 has several pre-defined headers and also a hop-by-hop option header [3] that can be used to define a fully customized header that conforms to RFC 6564[4].

A. UCC – A three-layer identification approach

The Cloud-ID is a 4 byte long integer value that needs to be unique across cloud environments. To guarantee uniqueness we propose a registration service similar to DNS. A global registrar manages existing Cloud-IDs as well as hands out new ones to cloud providers. For the purpose of this paper we ignore topics like ”ID re-use”, ”virtual Cloud Providers” or ”registrar Security”. These are discussed in the future work section.

The Service ID, a 6 byte long integer value, identifies a cloud service offered by a Cloud Provider. The ID is defined and managed within a cloud provider and therefore does not require a global registration service. Uniqueness is maintained within a Cloud Provider and across Cloud Providers by sub-categorizing it to the Cloud-ID. A Service-ID is added to the traffic stream by the Service or a tagging agent. A Service-ID only has local significance and is managed and maintained by the cloud provider.

The Tenant-ID, also a 6 byte integer, is created alongside the users account associated with a certain service. As with the S-ID, the Tenant-ID is managed within a Cloud Provider and therefore does not require a global registration service.

The overall hop-by-hop option header is 24 bytes long. This includes the UCC IDs plus several 1 byte long fields (as part of the hop-by-hop header) to define en-route behavior for the packet.

B. UCC Header size

The size of the Cloud, Service and Tenant ID, 4, 6 and 6 bytes respectively, is a critical component of our proposal. We argue that the size of the IDs is sufficient to cope with the growth of Cloud Computing in the next 5 to 10 years. By 2018, there will be 10 billion mobile devices globally [5]. The size of the Tenant-ID defines the maximum number of users for a particular service run on a particular cloud. A user will be assigned (different) Tenant IDs independently of the service used. Even if every mobile device is considered a single tenant, our Tenant-ID can store more than 10 times the projected mobile devices attached to the Internet. To uniquely identify a tenant all three IDs are required and therefore expanding the address space significantly to cope with future Cloud Computing growth. We are safe to speculate that the number of services run on a single cloud is less than the number of global users with Internet access. The 4 byte Cloud ID supports more than 4 billion Public and Private Clouds. We argue that, even without ID re-use, 4 bytes are sufficient to cope with the expected Cloud Computing growth over the next 5 to 10 years. Based on current Cloud growth projections we believe that the Cloud, Service and Tenant ID provide enough capacity for future growth without consuming unnecessary bandwidth for application data.

C. UCCs impact on Applications

To further evaluate the impact of the UCC proposal to a Data Center network we calculate the goodput for a typical application packet. Goodput defines the real application throughput per packet, excluding header overhead. In a typical DataCenter environment three standardized headers, Ethernet, IP and either UDP or TCP are used. Here, we add the proposed UCC extension header. The ethernet frame imposes a 26 byte overhead per packet, followed by IPv6 with 40 bytes and TCP with another 20 bytes. In addition, we add the UCC IPv6 extension header with an overhead of 24 bytes. This sums up to an overall header size of 110 bytes compared to 86 bytes without the UCC extension header. The typical transmission unit for Ethernet is 1500 bytes. Therefore, application information bits transmitted per packet are limited to 11120 bits or 1390 bytes. The goodput on a 10gig link using UCC is therefore limited to 9266 Mbps or 9.266 Gbps. This is a decrease in goodput of 2% compared to a normal TCP packet providing application useable throughput of 9426Mbps. To further compare the UCC proposal we use the example of VxLANs, which enable classification of traffic within Data Center environments. VxLAN differs from the proposal in the way that it is using an overlay approach. VxLAN traffic encapsulates traffic in a UDP packet. The encapsulation imposes an 70 byte overhead on every packet (Outer Ethernet Header (14) + UDP header (8) + IPv6 header (40) + VXLAN header (8)). In our calculation that calculates to VxLAN (70) + IPv6 (40) + TCP (20) an overall of 130 byte header information. The goodput for VxLAN using IPv6 is limited to 9133 Mbps. Therefore, the UCC proposal provides 2% more goodput than the state-of-the-art data center classification approach VxLAN.

D. UCC Security/Privacy

The IDs proposed in the UCC concept identify the Cloud, the Service and Tenant openly in each packet. Sending the UCC IDs in clear text across the network might raise security and privacy concerns. However, we believe that those IDs are not different to information already provided in packets to and from a Cloud Service. The Service can be identified by inspecting application data on layer 7. In addition, the user specific details are also often carried within the application payload. Tenant IDs are defined within a Cloud Providers environment and can be seen as an abstraction to security sensitive user credentials. There is no direct link between the Tenant ID and a users credentials without knowing Cloud Provider specific information. In our proposal the UCC IDs (Cloud, Service and Tenant) will be transmitted in plain text as part of the IPv6 header. Our proposal therefore provides the same level of security or privacy as existing systems.

1) Confidentiality of UCC IDs: Confidentiality is a security concern for traffic being sent across the internet and handled by multiple parties. While the UUC IDs do not offer additional confidentiality protection, they also do not introduce any security concerns or vulnerabilities. Identity protection and confidentiality should be enforced through state-of-the-art security overlays, encryption protocols and best practices applied to any IP traffic that carries sensitive security information.

The proposal is designed for the type of cloud traffic that seeks to carry its cloud identities in order to use such information for variable purposes (such as routing management, billing and firewalls). In this case, our proposal not only gives the same level of security and privacy as the existing system, but also brings significant improvements by defining cloud related identities in a standard, unified, systematic way using easy and compatible implementation with higher performance (comparing with existing cloud identity technologies). Note that the proposed UCC IDs can be used flexibly, e.g. they can be used jointly or separately and they can be added or removed at firewalls or gateway routers according to company policies.

2) Integrity of UCC IDs: It is possible to modify the UCC IDs (in plain text in IPv6 header) by a malicious attacker at any time in the network. However, this threat is always there for every IP packet, with or without the proposal. For example, any IP packet's source or destination IP address can be spoofed. For sensitive applications, security protocols (such as SSL and IPSec) can be used where the proposal is fully compatible and can enjoy the same security features.

In summary, our UCC proposal has the same level of security and privacy as existing systems while providing a wide range of crucial benefits. In fact, one of the benefits is to help/enable/improve network security. For example, firewalls can filter traffic based on not only IP addresses but also Cloud and Service IDs, which is more convenient, flexible and accurate and can be done solely at layer-3. Sensitive traffic can be fully tracked all the time cross the datacenter network in order to comply to complex access-control policies. The UCC proposal enables military based command chain protocols, which are not further discussed or investigated in this paper.

E. UCC Advantages

The UCC proposal satisfies the following requirements providing benefits to Cloud Providers, remote enterprises and Internet Service Providers. Cloud IDs are globally unique while Service and Tenant IDs are unique per Cloud. The proposal therefore guarantees uniqueness by using all three IDs together. The IDs are defined as non-mandatory fields. Network elements can choose to either use the IDs to enable additional functionality or ignore them. The IDs added to the Layer 2 or Layer 3 headers should add minimal overhead to the internal network element processing to not impact latency. UCC enables use-cases inside and outside a cloud environment that are highly desired to satisfy business requirements, such as per-tenant billing or security policies.

UCC is first and foremost a classification approach to eliminate ambiguity within Cloud data centers. However, besides this major improvement over current technologies, it also enables many highly desired network services.

IV. Evaluation of UCC in a data center environment

A. Evaluation Methodology

1) Implementation of UCC in IPv6 header: Our evaluation will focus on the IPv6 implementation of the UCC proposal, where the UCC identities are iserted in the IPv6 hop-by-hop option extension header. This header is outlined in [4]. It provides the right flexibility to implement the UCC proposal.

2) First stage of evaluation – Cloud Provider Internal: This paper presents the first stage of our evaluation, where we focus on testing UCC inside a data center environment. We will evaluate three aspects: feasibility and compatibility; performance; and functionality on billing applications.

3) Second stage of evaluation – Cloud Provider External: In our next paper we will present the second stage of our evaluation. We will then evaluate UCC outside a Cloud providers environment, where the use cases will be more diverse and complex. We plan to investigate security and routing related application for UCC.

B. Evaluation goals and requirements

1) Feasibility and compatibility: Firstly, we aim to verify the feasibility of implementing the UCC proposal in a real cloud environment. We want to show its compatibility to existing technologies without breaking or modifying a data centers setup. Here, solely focus on the compatibility and the ease of use of our proposal on the network and ignore application compatibility tests. We expect the switches to forward packets without modifying or dropping them.

2) UCCs impact on the Network: Secondly we will evaluate the performance of our UCC proposal against existing technologies. In particular we aim to study the impact of UCC on network performance. We will compare latency of UCC-enabled IPv6 traffic against plain IPv6 traffic, and then compare identity matching time by UCC-enabled IPv6 traffic against Layer-7 deep-packet-inspection of plain IPv6 traffic.

a) Performance metrics: end-to-end latency: The metrics we consider should allow us to evaluate the performance impact of our proposal. To evaluate performance, we rely on end-to-end latency between the user (or tenant) and the service. Here, the latency measured encompasses processing latency on the switches, latency introduced sending packets across the wire and matching packets against their service and tenant affiliation. We expect to see a slight increase in latency due to the additional processing time of the added 24 byte extension header.

b) Performance for different frame sizes (MTU): To gain more realistic performance measurements we introduce different frame sizes. The MTU size defines the packet size send across the wire. The minimal frame size is defined as the size of the overall Layer-3 header. Frame sizes larger than the Layer-3 header include payload data. We define a set of frame sizes ranging from the Layer 3 header size to the typical 1500 bytes (MTU={102, 256, 512, 768, 1500 and IMIX}). Different applications in cloud environments require different frame sizes and therefore performance evaluation on MTU sizes provides us with application specific results. For the purpose of this paper we ignore packet fragmentation for simplicity reasons.

c) Performance for different traffic rates (Frames per second (fps)): Another metric we leverage is the amount of frames send per second (fps). This allows us to gather latency values relative to the traffic load send across the test environment. For the proof-of-concept tests we send 10 packets per second to simplify the compatibility verification. To gather performance values we define a set of frames send per second from 10 to 100000 as Fsend={10, 100, 1000, 10000, 100000}. Data Center switches are designed to perform best under load. We can therefore expect performance improvement with increased frames send per second.

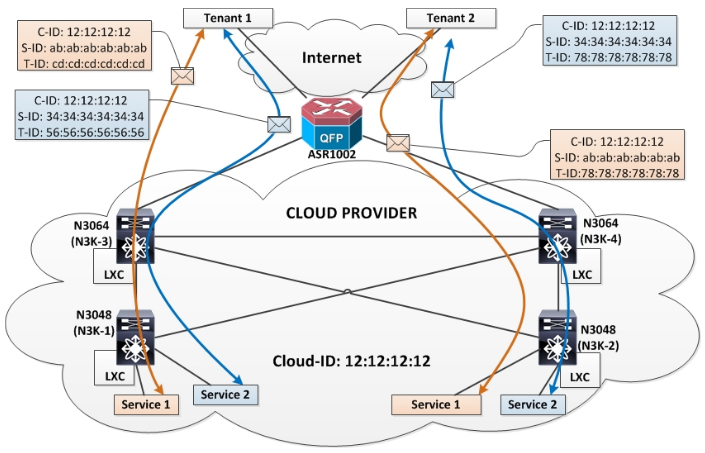

Fig. 1: **Network Topology:** The topology is defined according to a typical cloud-enabled data-center. Two streams from service to tenant and vice versa are used to simulate UCC traffic.

C. Billing as a UCC example use-case

Our UCC proposal can be leveraged in many different scenarios both within and outside a cloud providers network. Use-cases include, but are not limited to, service and tenant specific security (inside and outside a cloud) or internet routing of cloud data (outside a cloud). For the purpose of this paper we introduce and focus our evaluation on a use-case typically seen within a Cloud Providers network, such as billing.

Cloud computing should allow customers to consume resources as needed and pay only for what they use. The different Service offering models, ”Infrastructure as a Service (IaaS)”, ”Platform as a Service (PaaS)”, and ”Software as a Service (SaaS)”, leverage different resources that impact billing. However, all services use the network resources to send and receive tenant specific traffic. Currently, network utilization is gathered per interface for the whole traffic without distinguishing between services or tenants. This hinders Cloud Providers from accurately using network utilization in their billing models. Billing based on a per-service and per-tenant network resource utilization is a new way charging users for their on-demand cloud resource usage. Therefore, we focus our evaluation of the use-case on the feasibility of this UCC application. With the added matching capabilities on network devices we can both isolate traffic and define the amount of traffic processed per interface. Here, we fully rely on the Service- and Tenant-ID and ignore other Layer 2-4 identifiers.

V. Prototype Development

A. Prototype Design

To verify our solution we propose a generic data center design. Is is based on an industry standardized network design with a collapsed core, aggregation, edge and access layer. We leverage the collapsed core to provide inter-subnet routing. This allows us to verify traffic flows for services on different Layer 2/3 domains.

The functionality of intercepting packets and matching them against a set of UCC-IDs is based on software defined networking (SDN). The network consists of 4 switches that simulate the access and aggregations layers of a data center. They are connected in a nearly full-meshed fashion (no interconnection between access-layer switches). The edge is showing how the Data Center can be connected to Service Providers and the wider Internet.

IPv6 support is required end-to-end, which is realized with arbitrary IPv6 interface addresses and OSPFv3 for IPv6 routing. For our solution validation we implemented four IPv6 streams. Bi-directional traffic is simulated between two users or tenants on the Internet and two services within the cloud-enabled data center. For this paper, a logical traffic flow would look like this: User 1 send traffic to the edge router, which forwards the packets to the aggregation layer. Those two switches forward the packets to the access layer devices that connect the service to the data center.

B. Prototype Code Development

For the switches to understand the introduced IDs on layer 3 we need to either alter their Network Operating System (NOS) or use an alternative way of matching at a certain offset in a packet. We decided to use an SDN-like open network programmable interface to gain access to the data plane to do packet manipulation and monitoring.

VI. Evaluation Results

For each experiment we generate multiple traffic streams between services in the south and users in the north using different classification IDs per stream.

A. Compatibility result

With the use of IPv6's Hop-by-Hop option extension header we expect to see no incompatibility problems with existing industry standardized hardware. The Hop-by-Hop option header is designed in a way to define per hop behavior so that devices can ignore the packets in case it is not known. This makes UCC optional to intermediate devices. We have run tests showing the ease of implementing the UCC extension header without breaking underlying network principles.

We verified this behavior for typical Data Center switches seen in Cloud Provider networks and routers as seen in the WAN edge and ISP networks. This shows that the proposal not only is feasible for new cloud-enabled data center but also for established Internet routing infrastructures. The byte string as shown in Figure 3 highlights a typical IPv6 packet with the UCC proposal incorporated. We captured this byte string on one of the switches in the test bed. It demonstrates and proves compatibility to current technologies.

B. Performance result

1) End-to-end Latency vs. Packet Rates: Latency is influenced by both the MTU size and the traffic rates. Our evaluation shows that latency is the lowest at the highest traffic rates of 100k packets per second. This finding is valid for all MTU sizes tested. The behavior is also seen for both IPv6 and UCC traffic and can be explained by the hardware optimization for high traffic rates.

```
00000    60 73 5c df f3 3c 00 00 06 1e 43 9f 86 dd 60 00
00016    00 00 00 30 3c 40 20 01 ab ab 00 00 00 00 00 00
00032    00 00 00 00 00 01 20 06 ab ab 00 00 00 00 00 00
00048    00 00 00 00 00 02 3b 02 00 04 12 12 12 12 00 06
00064    ab ab ab ab ab ab 00 06 cd cd cd cd cd cd cc 0e
00080    3d 1b 2c 04 53 96 49 78 69 60 00 00 00 00 de fc
00096    04 54 8d 0c 74 73
```

Fig. 3: **Example of an IPv6 packet with UCC identifiers:** The packet is an example from the stream between User 1 and Service 1 in Fig.1. The Cloud-ID is 12:12:12:12, the Service-ID is ab:ab:ab:ab:ab:ab and the Tenant-ID is cd:cd:cd:cd:cd:cd

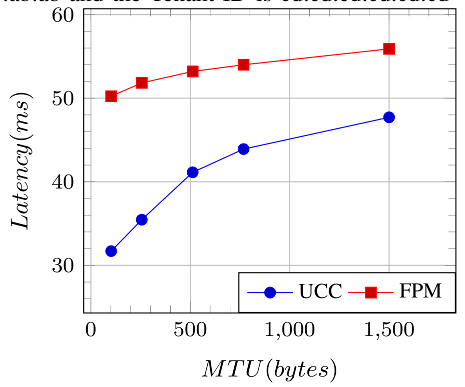

Fig. 4: **Latency difference between FPM and UCC:** The results show that UCC outperforms FPM

2) Performance for different package sizes: In this set of tests we compare the latency performance between the proposal and typical IPv6 traffic. We run traffic with different payload sizes to simulate different types of applications. The MTU size plays a big role in the forwarding behavior of switches in general and is therefore a good indication of the performance characteristics of the proposal. Figure 2 shows that our proposal has a slightly higher latency as plain IPv6 without any extension headers. This can be explained with the 24 bytes increased packet size and is expected.

3) Comparison between UCC and FPM: Flexible Packet Matching (FPM) is a technology introduced by Cisco enabling deep-packet inspection on routers. Here we compare the performance between our UCC proposal and FPM[6]. We define FPM on the edge router to match packets based on a certain offset in the HTTP payload. We use this to simulate matching packets according to their tenant and service affiliation based on Layer 7 information. For this test we run FPM separate from our proposal implementation. Ciscos FPM feature punts packets to the CPU therefore processing them in software.

Figure 4 shows that our proposal significantly outperforms FPM identifying Service and Tenant specific packets. Here, we show the results captured with a traffic rate of 100pps for both FPM and UCC. The performance difference is due to matching against a certain pattern at a different offset within a IPv6 packet. UCC starts at an offset of 54 bytes whereas HTTP service and tenant details are typically deep inside the HTTP payload.

C. Test result of classification-aware billing

With the successful implementation of matching packets against their service and tenant affiliation the proposed classification-aware billing is easily realized. Figure 1 outlines multiple flows demonstrating the importance of per-tenant

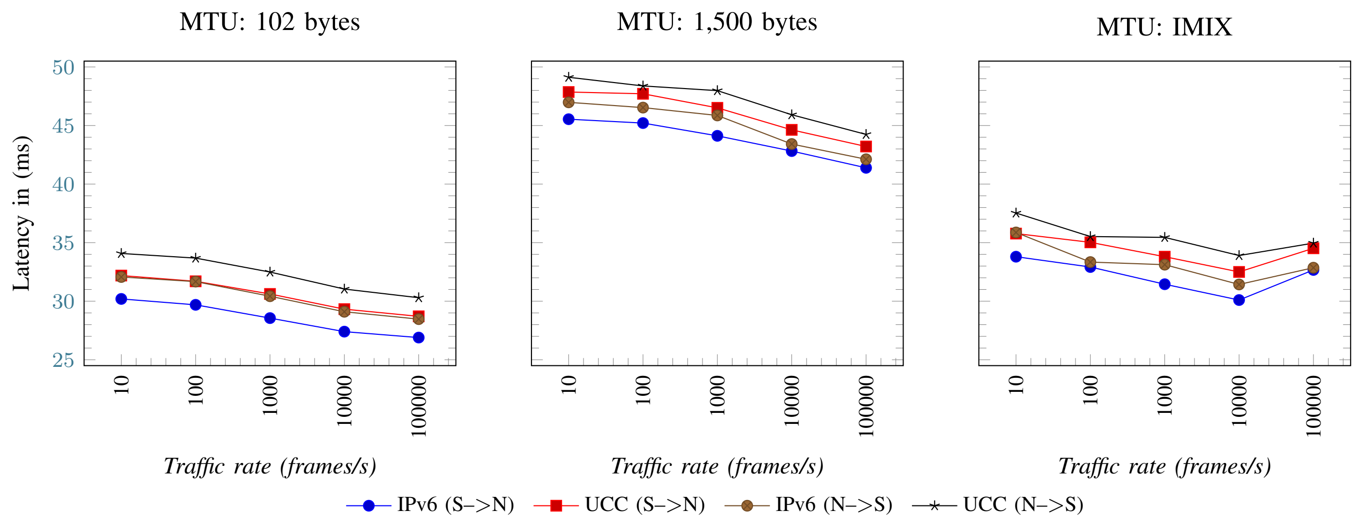

Fig. 2: **Performance results:** We compare the latency of IPv6 and IPv6+UCC packets in the south-to-north (S–>N) and north-to-south (N–>S) streams in Fig.1. Latency is measured for five different traffic rates (from 10 to 100,000 packet/s) and five different MTU sizes (from 102 to 1,500 bytes), respectively. We also study a mixed traffic flow (IMIX) with three MTU sizes (256, 512 and 1500 bytes) mixed equally. Results show that as expected, IPv6+UCC has a slightly longer latency (around 2*ms*) than IPv6, for all traffic rates and MTU sizes.

and per-service billing. Each flow takes a different path and therefore utilizes different network resources. Leveraging the UCC proposal a cloud provider can easily distinguish and isolate traffic to charge based on utilization. It also enables differentiation in charges for different services (i.e. billing Service 1 differently to Service 2 based on the traffic characteristics as traffic can be uniquely associated). This demonstrates well the flexibility and usability of the UCC proposal in an application critical for cloud providers.

VII. Discussion and Conclusion

In this paper we performed several basic tests to evaluate the UCC proposal. We consider this paper as the first in a series of evaluation papers to come. Here, we focus on the feasibility of UCC, some performance comparisons to IPv6 traffic and flexible packet matching (deep-packet inspection) and the introduction of a use-case for UCC within a Cloud Providers network. We plan to extend our evaluation of the UCC proposal in future papers to focus on more comprehensive cloud-inside and -outside scenarios.

The evaluation performed and outlined in this paper is backed-up by a Cisco based physical testbed. This gives us the most realistic and credible compatibility, performance and feasibility results compared to virtual/simulated environments. The test suite defined is well-suited for a prototype verification with its real-life performance evaluation tests. We compare our UCC proposals performance against standard IPv6 traffic and state-of-the-art technologies like flexible packet matching. The test results are reliable, consistent and repeatable as captured with a well accepted traffic generator tool.

Based on the results we can conclude that our UCC proposal is a feasible solution to isolate services and tenants within a cloud providers network. It enables highly demanded applications such as per-service and per-tenant billing based on network resource utilization. The performance impact is minimal with a 5 to 10% latency increase (for example, 29ns with plain IPv6 traffic compared to 31.5ns including our UCC proposal). This increase is solely caused by the UCC header (not due to Cisco hardware). We argue that this performance impact is acceptable in cloud-enabled data centers. We believe that the 24 byte header increase is not hindering a broad acceptance of our proposal. Current, similar approaches, such as ”Overlay Transport Virtualization (OTV)” or ”virtual extensible LANs (VxLAN)” add 42 or 50 bytes overhead respectively.

Our next paper will focus on further evaluation of a comprehensive use-case for Internet Routing of Cloud Data. Cloud-Provider, Service and Tenant isolation can be used on ISP networks to make forwarding decisions according to certain requirements of services and/or tenants. This paper will also include an evaluation of traffic send across the internet to understand the compatibility on different ISP networks. With the used hop-by-hop option IPv6 extension header we anticipate full compatibility across the internet.

References

[1] S. Jeuk, J. Szefer, and S. Zhou, “Towards cloud, service, and tenant classification for cloud computing,” in *Proceedings of 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2014.

[2] S. Jeuk, M. Rio, and S. Zhou, “Tenant-id: Tagging tenant assets in cloud environments,” in *Proceedings of 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2013.

[3] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998, updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[4] S. Krishnan, J. Woodyatt, E. Kline, J. Hoagland, and M. Bhatia, “A Uniform Format for IPv6 Extension Headers,” RFC 6564 (Proposed Standard), Internet Engineering Task Force, Apr. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6564.txt

[5] Cisco, “Cisco global cloud index: Forecast and methodology, 2012-2017,” Website. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html

[6] ——, “Cisco ios flexible packet matching (fpm),” Website. [Online]. Available: http://www.cisco.com/c/en/us/products/security/ios-flexible-packet-matching-fpm/index.html