PacketCloud: an Open Platform for Elastic In-network Services

Yang Chen[1], Bingyang Liu[2], Yu Chen[1], Ang Li[1], Xiaowei Yang[1], Jun Bi[2]
[1]Department of Computer Science, Duke University, Durham, NC 27708, USA
[2]Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China
{ychen,yuchen,angl,xwy}@cs.duke.edu,
liuby@netarchlab.tsinghua.edu.cn, junbi@tsinghua.edu.cn

ABSTRACT

The Internet was designed with the end-to-end principle where the network layer provided merely the best-effort forwarding service. This design makes it challenging to add new services to the network layer. However, as the Internet connectivity becomes a commodity, users and applications increasingly demand new in-network services. This paper proposes PacketCloud, a cloudlet-based open platform to host elastic in-network services. PacketCloud can help both Internet Service Providers (ISPs) and emerging application/content providers deploy their services to strategic network locations. We have implemented a proof-of-concept prototype of PacketCloud based on the MobilityFirst architecture. PacketCloud introduces a small additional delay, and can scale well to handle high-throughput data traffic.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network communications

Keywords

Cloud computing, in-network services, open platform, elasticity, future Internet architecture, MobilityFirst

1. INTRODUCTION

The Internet was designed with the end-to-end principle where the network layer offered merely the best-effort forwarding service. Many important communication functions such as reliability and state management are placed at the end points [1]. This design provides low complexity and high robustness at network routers, and is necessary to meet the technology constraints of the 1970s when the Internet was first introduced.

However, as the Internet connectivity becomes a commodity, more and more users and applications increasingly demand new functions and services to be added to the network, such as host mobility support [2], efficient delivery for frequently accessed content [3], on-path storage [4], or malicious traffic blocking [5]. In general, these services can either enhance the user experience while surfing the Internet, or optimize the traffic in the network. Unfortunately, it is difficult to add new services or functions to the network layer of the existing Internet, due to the end-to-end design principle. The nuts and bolts of the Internet are not designed to be easily extensible. Some researchers have coined this problem as the Internet's "ossification" problem [6].

There are a few existing proposals for in-network services. Overlay networks such as Internet Indirection Infrastructure (i3) [7] suffer from detours during data delivery. CoMb [8] can consolidate in-network middleboxes in enterprise networks. However, CoMb needs thorough knowledge of deployed applications for performance optimization, and is not open to emerging third-party content/application providers. APLOMB [9] can outsource in-network processing tasks in enterprises to the public clouds. Still, APLOMB may generate unwanted interdomain traffic even when both the source and the destination are in the same domain. Also, letting public cloud providers access the traffic data may introduce data ownership problems [10].

We propose an architectural solution that integrates "cloudlets" [1] into the Internet's routing infrastructure. Our solution, PacketCloud, accommodates a distributed resource pool for in-network service providers. ISPs can choose suitable Point of Presences (PoPs) to deploy cloudlets. Compared with widely-used standalone middleboxes, the available resources can be shared among services. Also, every deployed service can achieve an elastic resource allocation. PacketCloud is compatible with different underlying network architectures including today's Internet Protocol (IP), and is chosen as the "computing layer" of the MobilityFirst architecture [11] to accommodate in-network services.

As an open platform, PacketCloud benefits both ISPs and third-party application/content providers. We show in § 3.4 that PacketCloud can help ISPs host value-added services to satisfy their customers. Also, PacketCloud can help the ISPs tailor the network traffic. By placing traffic optimizers/firewalls at data forwarding paths, duplicated/malicious traffic can be identified and eliminated. For third-party providers, PacketCloud allows them to deploy their services in the network infrastructure, by renting resources of the in-network cloudlets. This new collaboration method helps the third-party providers utilize the strategic network locations in a convenient way, while providing viable economic

[1]In our framework, a cloudlet is a computer cluster consists of tens of commodity PCs.

MobiArch'13, October 4, 2013, Miami, Florida, USA

http://dx.doi.org/10.1145/2505906.2505910 ...$15.00.

rewards for ISPs. Compared with today's practices, PacketCloud leads to a win-win solution for both ISPs and third-party providers. For example, Netflix, which consumes more than 30% of peak traffic on US ISP networks, begins to provide content caching servers for ISPs for traffic optimization [12]. If PacketCloud was deployed, Netflix could simply rent the resources in selected cloudlets to host the caches instead of delivering physical servers to ISPs.

We have built a proof-of-concept prototype of PacketCloud, and four representative use cases. We use the protocol translator as an example to demonstrate the benefits of PacketCloud services. Also, our evaluation results show that PacketCloud only introduces a small delay penalty, which is acceptable for numerous Internet applications. PacketCloud is scalable and can handle high-throughput data traffic.

2. BACKGROUND AND RELATED TECHNOLOGIES

2.1 Underlying Network Architecture

PacketCloud does not have any particular requirement on the underlying network architecture. It is compatible with conventional architectures such as today's Internet Protocol (IP), and clean-slate architectures such as MobilityFirst (MF) [11]. Due to the page limit, we pick MF as the underlying network architecture. The main reason for choosing MF is that the mobile platforms and applications are developing rapidly. According to a report from Walker Sands [2], the percentage of website traffic coming from mobile devices becomes 23.1% in Q4 2012, demonstrating an 84% increase from 12.6% in Q4 2011. It is anticipated that by 2015, emerging mobile data will significantly exceed the traffic among fixed hosts on the Internet [11]. MF is prominent in handling host mobility and wireless data delivery.

MF supports the ubiquitous mobility by the separation of permanent identifiers from topology-dependent addresses [11]. Every network-attached entity can be identified by a globally unique identifier (GUID). Different from topology-dependent network addresses (NAs), the GUID of an entity will stay the same when this entity travels among different networks. Another key feature of MF is the optimized reliable data transmission mechanism. The reliable data transmission in MF is robust to data links with varying qualities [4, 11]. The unit of reliable transmission in MF is a chunk, i.e., a large segment of contiguous packets. Similar to [4], every chunk is delivered in a hop-by-hop manner, and on-path routers store received chunks temporarily. If we need to retransmit a chunk due to temporary disconnection or bad link quality, we can start from the one closest to the destination among the routers who have received the chunk. Meanwhile, the acknowledgment messages is generated in chunk level. This can reduce the number of acknowledgments by using TCP and the idle time of waiting for acknowledgments [4].

PacketCloud has become a key component of MF to host emerging in-network services. Based on MF, we propose the detailed design of PacketCloud in § 3.

2.2 Service Isolation and Virtualization

To avoid the unnecessary resource waste of using standalone, specialized middleboxes, we allows one PC to host multiple in-network services simultaneously. Among existing proposals, CoMb [8] runs multiple services on the same PC without using virtualization. That is because CoMb hosts trusted in-network functions. However, PacketCloud is an open platform to host services from different service providers, the deployed services could be experimental, or even malicious. Virtualiaztion is necessary to provide isolation among them. We demonstrate how to use virtualization to ensure the reliability and security of PacketCloud in § 3.5.

Among different virtulization technologies, we choose operating system-level virtualization to host multiple isolated user-space instances. Compared with whole-system virtualization (e.g., VMWare vSphere) or paravirtualization (e.g., Xen), operating system-level virtualiaztion has little overhead. For our Linux-based prototype, we use lxc (LinuX Containers) for virtualization [3].

3. SYSTEM DESIGN

This section describes the design of PacketCloud. We give an informative overview of PacketCloud architecture in § 3.1. As the deployed cloudlets form an elastic resource pool, we investigate the resource allocation in § 3.2. We present the service identification and discovery policy in §3.3. A number of use cases are shown in § 3.4. We describe the reliability and security considerations in § 3.5.

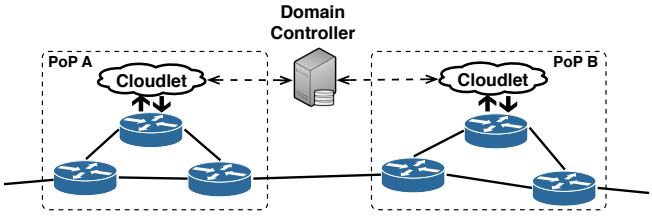


Figure 1: Cloudlets in an ISP

3.1 Overview

We assume that an ISP owns p in-network cloudlets, namely C_1, C_2, ..., C_p. These general-purpose cloudlets can be placed at PoPs chosen by the ISPs incrementally. To use all p cloudlets as a resource pool, the ISP hosts a logically centralized *domain controller*. As shown in Fig. 1, this domain controller manages all deployed cloudlets. Every cloudlet sends a resource report to the domain controller periodically, including the reserved and available resources (CPU, memory, disk, and I/O). Therefore, the domain controller maintains a local database to keep an aggregate view of the resources of all p cloudlets. This sole interface helps both ISPs and third-party service providers to schedule their services among available cloudlet resources.

A cloudlet is a functional unit in the resource pool. As shown in Fig. 2, a cloudlet is composed of a cloudlet controller, and a number of general-purpose computation nodes. The computation nodes are used for hosting in-network services, and the cloudlet controller is responsible for resource allocation and management. There are two physical network interfaces (NICs) in every computation node, i.e., a data NIC and a control NIC. The data NIC exchanges data chunks between the computation node and the co-located

[2]http://www.walkersands.com/quarterlymobiletraffic

[3]http://lxc.sourceforge.net/

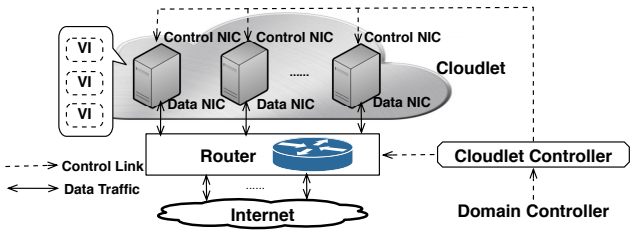


Figure 2: Building Blocks of a Cloudlet

router. The control NIC handles the management messages between the computation node and the cloudlet controller. We assume that a router selects x ports to devote to in-network services, and each of them connects to a switch. If by average every switch has y available ports to connect to computation nodes, this router's co-located cloudlet can accommodate at most $x \times y$ computation nodes. As will be shown in § 4.2, a cloudlet with tens of nodes is already capable to handle high-throughput traffic requesting computationally intensive services. The cloudlet controller serves as an agent for the domain controller, and actively monitors the resources of every computation node within its cloudlet.

Similar to public cloud services, every deployed service is contained in a lxc based VI. PacketCloud provides multiple types of virtual instances (VIs). A service provider can reserve any VI that matches its requirement. The ISP can also suggest a cost-effective VI type for the service provider based on benchmarking results [13], e.g., the number of chunks an intrusion detection system (IDS) can inspect per second using a certain type of VI. To handle high-throughput traffic, a service can even reserve multiple computation nodes.

3.2 Resource Allocation

To reserve cloudlet resources from an ISP, a reservation request should be filed to the domain controller. This request should include a desired VI type and quantities, a requested time slot, and optionally, a preferable cloudlet C_i. Upon receipt of a request, the domain controller checks the local database to list the cloudlets that satisfy the request, and further checks with corresponding cloudlet controllers to confirm which cloudlet will host the service. Note that the confirmation procedure can address some prospective information inconsistency, e.g., discovering some unexpected server crash occurs after the latest resource report was sent.

The cloudlet controller releases the resources occupied by a certain service after the reservation slot expires. According to the record provided by the domain controller, the third-party content/application providers need to pay the corresponding ISPs in a "pay-as-you-go" fashion.

As the traffic data rate of a certain service may vary over time, a pre-selected VI may not have enough capability during peak hours, and thus has to discard extra chunks that it cannot handle. To address this problem, we allow a service provider to request to scale up and down the capability of a reserved VI. If there are enough available resources, the service providers can ask for more computation or storage resources for the VI when the data throughput becomes larger. Also, it can choose to just keep minimal resources during off-peak hours. In short, PacketCloud is able to host elastic in-network services to serve the dynamic Internet traffic.

For a certain computation node, *scaling conflict* occurs when one or multiple VIs are demanding new resources, while the node is not able to allocate enough resources for them. We suggest to use a migration-based conflict handling method, i.e., by migrating some VIs to the least loaded node of the cloudlet. We leave the detailed conflict handling policy as an interesting future work.

3.3 Service Identification and Discovery

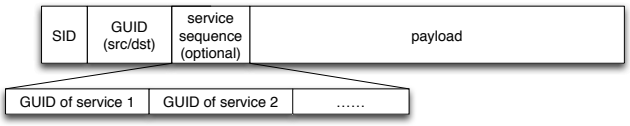


Figure 3: MobilityFirst Chunk Format

PacketCloud supports two categories of services, i.e., *user requested services* and *transparent services*. The difference lies on who can activate the service.

A user requested service is activated by end users. Every deployed PacketCloud service can be identified by a globally unique and routable service identifier, so-called service ID (SID). The SID acts as the GUID of a deployed service. Fig. 3 shows the format of a typical MF data chunk. The first packet of the chunk has a *network header*, including a SID field, and a field of the GUIDs of source/destination entities. The sender needs to explicitly insert the desired service's SID into every chunk he sends. Upon the sender's request, the network will *demultiplex* and forward the traffic to the desired service first, and then deliver the processed chunks to the destination. If a sender desires multiple services one after one, he needs to set the SID field to a specified value, indicating "multiple service desired". This specified SID enables the optional *service sequence field*. The chunk sender can insert a sequence of SIDs into this field. The network will send a data chunk to every desired service sequentially according to the service sequence field.

Transparent services are activated by the ISPs by intercepting and further processing legacy traffic. The sender and the receiver may not even be aware of the existence of these services. By applying some pre-configured *demultiplexing rule* (e.g, a specified source/destination GUID, a specified pattern in chunk payload), a router can intercept selected chunks and forward them to a co-located transparent service without notifying the sender and the receiver. An intercepted chunk may either be dropped, be forwarded further to next hop in an "as is" fashion, or be delivered further with an updated payload. Third-party service providers can collaborate with ISPs to launch transparent services.

A router may find a received chunk related to multiple deployed services in its co-located cloudlet. For example, the chunk sender requests a transcoding service in this cloudlet, while the ISP wants the chunk to be inspected by an IDS service in this cloudlet as well. The cloudlet controller has a priority configuration to judge which service should be conducted earlier. According to this configuration, the router will deliver the chunk to the unexecuted service with highest priority, and so on and so forth. If a service with higher priority decides to discard this chunk, there will be no further actions for other services.

3.4 Use Cases

We list a number of use cases as follows. First, let us show some user requested services.

Protocol translator: Backward compatibility is critical for boosting a new architecture such as MobilityFirst [11], as a new architecture will always be deployed gradually. We deploy protocol translators at the boundary between a new infrastructure and the current IP infrastructure. Therefore, a user in a new network can access resources in IP networks through a translator.

Context-aware service: Most of the Internet users are not computer professionals, technical terminologies like GUIDs are hard for them to remember. PacketCloud is a natural place to translate human-readable contexts into GUIDs. Context-aware services can help mobile users to find nearby bus stops, explore new places, and communicate with friends, without knowing the GUIDs of the communication objects.

Transcoding: Similar to PC-based users, online video watching is an important activity for mobile device users. However, as the bandwidth of a mobile device using 3G/4G network may fluctuate constantly, the available bit-rate varies over time. A transcoding service can measure the available bandwidth periodically. If the bit-rate provided by the video publisher is too high for the latest available bandwidth between the video publisher and the video viewer, the service can dynamically transcode the video stream according to the measured available bandwidth.

Delay Tolerant Content Delivery: To handle network congestion, upgrading the capacity of corresponding links can be expensive, not to mention the unavoidable over-provisioning during off-peak hours. Alternatively, ISPs may want to host some in-network storage instances [14] to store delay-tolerant contents. An end user can choose to send data through a topologically-closest storage instance. The data will be stored temporarily if the path between the instance and the destination is congested. When the network becomes lightly loaded, the content can be sent out to the destination. This approach can leave room for time-sensitive traffic. ISPs can provide some rewards to users who utilize this service to deliver less time-sensitive content.

Anonymous communication: Anonymous communication is a vital key for anti-censorship, and onion routing is a popular method. Today's end hosts-based Tor overlay suffers from heterogeneous access bandwidth of end hosts [15]. With PacketCloud, onion routers can be placed in the network infrastructure to attain higher network capacity. To avoid ISP-based traffic observation for both ends of an anonymous connection, an ISP-aware onion routing path selection algorithm [16] can be used.

Also, we enumerate the following transparent services.

On-path Encryption/Decryption: An organization may have multiple branches in different locations. Confidential and unencrypted data may go through untrusted networks between two branches. Using PacketCloud, the two branches can transparently deploy an encryption service for outgoing traffic, and one decryption service for incoming traffic. This effectively reduces the encryption/decryption cost at the end hosts. For example, for mobile device users, the limited battery resources can be saved.

WAN Optimizer: People always transfer uncompressed data through the Internet. Such data traffic can be represented more efficiently using compression technologies. An ISP can perform WAN optimization between two strategic network locations at the data forwarding path. The redundant contents can be compressed before the delivery between these two locations.

Intrusion detection system (IDS): An ISP's PoPs are natural locations to deploy in-network IDSes to identify malicious/unwanted traffic. ISPs can dynamically adjust the filtering policies, according to some field values in the chunk header, or content patterns in the chunk payload. Identified traffic will be either blocked, or rate limited.

Load Balancer: A load balancer is a proxy to distribute the workload across multiple servers with a shared publicly known GUID. The load balancer can divide all chunks going to this GUID into multiple flows, each of which will be redirected to one server.

Content caching: To enhance content delivery efficiency, caches can be deployed at path aggregate points, keeping local copies of frequently requested contents. These caches are not only important to ISPs, but also attractive to content providers. The Netflix case in § 1 is an example.

3.5 Reliability and Security

Both reliability and security are very important for PacketCloud as it works inside the Internet infrastructure. There are several issues we need to consider in the design.

Unexpected service failures: To limit the impact of unexpected failures, PacketCloud uses lxc virtualization for service isolation. Therefore, any unexpected crash will only fail the corresponding service within the VI without troubling other services running on the same computation node. Moreover, the physical decoupling between the cloudlet and the router ensures that the router will operate robustly under any service crash. This feature is important for deploying new experimental services.

Malicious demultiplexing rules: To decide which chunks should be forwarded to a deployed service, the service provider needs to provide some demultiplexing rules to the corresponding ISPs. A malicious rule can demultiplex some legitimate traffic, sniff the traffic, and even discard some data chunks intentionally. To address this, PacketCloud inspects every third-party provider's demultiplexing rule. For a user requested service, the service provider needs to show the ownership of the specified SID. For a transparent service, the service provider must be the owner of the specified destination GUID. Ownership verification of GUIDs/SIDs in MF is easy, as the GUIDs/SIDs are self-verifying [11].

Malicious services: A third-party service may include some malicious functions in its code. To secure the whole cloudlet, PacketCloud only allows the corresponding ISP to own the root permission of every computation node. As every deployed service is restricted to be executed inside a VI, the ISP can inspect all or a sampled subset of chunks to/from the instance. If there is any abnormal activity, the ISP can restrict the resources of a suspected service, or even terminate the service.

Excessive resource usage: Although PacketCloud is designed for handling high-throughput traffic, still, a cloudlet's processing capability is not as high as a data center. We want deployed computation resources to be utilized efficiently. We should not allow a malicious third-party provider to occupy a significant part of a cloudlet with a low financial overhead. Our solution is two-fold. On one hand, a fixed amount of dedicated resources are reserved for services deployed by the ISPs. On the other hand, we introduce a

tiered pricing policy [17]. A high price will be applied to punish the third-party providers who intend to occupy an unreasonable amount of resources.

4. EVALUATION

4.1 Implementation of Use Cases

We implement a proof-of-concept prototype of PacketCloud based on the latest MF prototype [11], which is developed using Click modular router [4]. We integrate PacketCloud services into the MF prototype, mainly by adding the service identification and discovery module. Among the use cases we introduce in § 3.4, we have implemented four of them, i.e., MF-IP protocol translator, WAN optimizer, intrusion detection system, and secure communication.

We pick the MF-IP protocol translator as an example to show the benefits of PacketCloud services. This translator does on-the-fly translation between MF chunks and IP packets, and it can help MF users access resources in the IP network. We use a simplified four node topology in DeterLab testbed [5]. There are two end hosts A and B, and one router R stays in between. The router R connects to a co-located node C, which acts as a simplified *cloudlet* with one computation node. The network capacity, latency, and packet loss rate of the links $A-R$, $R-B$, $R-C$ are shown in Fig. 4(a). Among all available node types in DeterLab, we adopt the "bpc2133" nodes in DeterLab for these four nodes, as these nodes use quad core processors while other nodes use dual core processors. Each bpc2133 node has an Intel Xeon X3210 quad core processor running at 2.13 Ghz. All data NICs and router ports works at a rate of 1Gbps.

We have the following three scenarios. The first scenario shows a pure IP environment, as A, B, and R all run IP protocol. Then we measure the goodput of downloading a 1GB file from B to A using TCP, and the measured result is 4.30Mbps. The second scenario shows a pure MF environment, as A, B, and R all run MF protocol. Then the goodput of getting a 1GB file from B to A becomes 26.14Mbps using MF's reliable data transmission mechanism [11]. This demonstrates that MF performs better than IP in reliable data delivery. Note that we do not involve the cloudlet C in the these two scenarios. In the third scenario, we assume that node A belongs to an MF network, and node B belongs to an IP network. Node R is a dual-stack router, which locates at the boundary between the MF network and the IP network. The protocol translator service locates at the cloudlet C. This service can fetch the desired 1GB file from node B using TCP, and send every retrieved data segment of this file back to node A using MF's reliable data transmission manner. The measured goodput is as large as 7.10Mbps, 65.12% higher than the goodput in the first scenario. In other words, when an MF user downloads resources in IP networks, the reliable data transmission manner can help him experience good downloading efficiency. This is also one motivation for IP users to upgrade to MF.

Due to the space limit, we briefly describe the other three implemented services. To build the WAN optimizer, we utilize the GNU zip library, a fast data compressor, to compress the payload of selected chunks. For the encryption and decryption services, we use the OpenSSL library to support different encryption/decryption algorithms. For the intrusion detection system, it allows the corresponding ISP to define some rules to detect and discard unwanted data chunks. The rules can include selected source/destination GUIDs or specified chunk payload patterns. All these services have been deployed and successfully tested on DeterLab testbed.

[4]http://www.read.cs.ucla.edu/click/click

[5]http://www.isi.deterlab.net/index.php

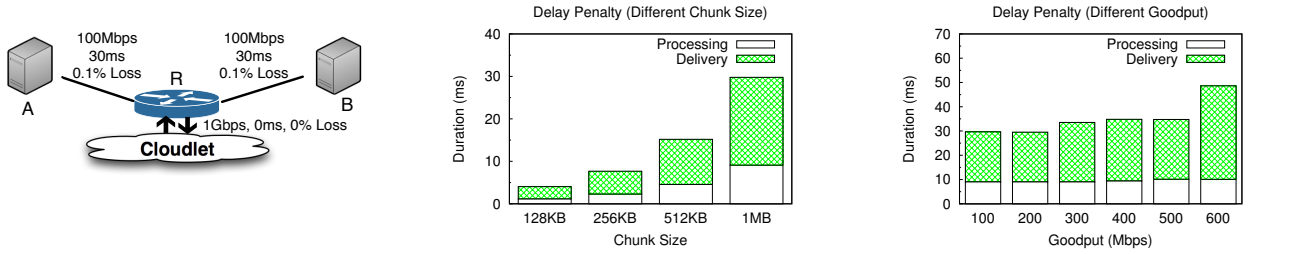
4.2 Delay Penalty and Scalability

We conduct some performance-oriented microbenchmarking with two major focuses. First, *delay penalty*. We define the delay penalty as the time duration between a router starts to forward a chunk to its co-located cloudlet for processing, and the router receives the entire processed chunk. We require this delay penalty to be small, as the end-to-end delay is critical for many Internet applications. Second, *scalability*. A cloudlet should be able to handle high-throughput data traffic.

The *delay penalty* of using a PacketCloud service is composed of a *processing duration* and a *delivery duration*. The processing duration indicates the duration of conducting a selected service for a data chunk. For the same hardware device, the processing duration is determined by the complexity of the desired service, as different services require different amount of computation resources like CPU cycles. The delivery duration is defined as the delay penalty minus the processing duration. It indicates the sum duration of sending a chunk from the router to a computation node, and sending a processed chunk back to the router. The delivery duration is caused by data transmission and chunk buffering.

For our evaluation, we study the AES encryption function (using AES-128-CBC mode) of the implemented secure communication service, as this encryption function is widely used and computationally intensive. Using the topology in Fig. 4(a), we let the router R forward fixed size chunks to the computation node C at a slow rate of one chunk per second. The chunks are processed at C and returned to R. By varying the chunk size and repeating our experiment for 10 times, we can obtain the average processing duration and the average delay penalty as in Fig. 4(b). We can see even when the chunk size is as large as 1MB, the average per-chunk delay penalty is still less than 30ms. This number is much smaller than the additional delay of sending an individual IP packet using 3G [18]. Therefore, the delay penalty is acceptable for numerous Internet applications. Note that this number is modest. As the CPU hardware we use for computation nodes was released in Jan. 2007, we can easily reduce the processing duration by using a newer CPU. Similarly, if we launch a service which is less computationally intensive, the processing duration can be reduced as well. The delivery duration is bounded by the data rate of the data NIC. We foresee that the delivery duration can be reduced significantly by adopting a 10Gbps data NIC for computation nodes. Particularly, for possible deployment of PacketCloud in IP networks, the delay penalty would be significantly smaller, as the protocol data unit in the network layer IP is a packet instead of a chunk.

To fully understand how well a cloudlet can deal with high-throughput data traffic, it is important to investigate the capacity of a single computation node as a first step. We let the router R send chunks to computation node C at different rates. We vary the arriving goodput of the data chunks, from 100Mbps to 600Mps, at an interval of



(a) Topoogy (b) Delay Penalty (Different Chunk Size) (c) Delay Penalty (Different Goodput)

Figure 4: Evaluation on Deterlab

100Mbps. The chunk size is configured as 1MB. The resulted delay penalty values are shown in Fig. 4(c). When the data rate is under 300Mbps, the average delay penalty stays stable. When the data rate is as large as 500Mbps, the average delay penalty increases to 34.76ms, 17.08% more than the delay penalty when the data rate is 100Mbps. Such increase is because of the thread scheduling, and the additional buffering time for receiving/sending chunks. When the data rate is as high as 600Mbps, the average delay penalty is 48.62ms, still, this value is smaller than the additional per-packet delay of using 3G. We can also see that the increase in the delivery duration is more significant than in the processing duration. Therefore, the network interface between R and C becomes the bottleneck. When the data rate becomes larger than 650Mbps, the delay penalty jumps quickly to several seconds. Therefore, we do not suggest one computation node to handle a goodput larger than 600Mbps. To conclude, a bpc2133 node can handle traffic as fast as 500Mbps∼600Mbps with an acceptable delay penalty. If we can have 20 such computation nodes for a cloudlet, and every node handles a goodput of 500Mbps, the cloudlet can handle up to 10Gbps traffic. If we introduce better hardwares, or, accommodate services with less computational complexity, we are able to handle a higher goodput using the same number of computation nodes.

5. CONCLUSION AND FUTURE WORK

This paper proposes PacketCloud, an architectural solution to host elastic services in the network infrastructure. As an open platform, PacketCloud helps both ISPs and third-party providers deploy in-network services conveniently and incrementally. Our evaluation demonstrates the usefulness of PacketCloud. PacketCloud has a small delay penalty, and can scale well for high-throughput data traffic.

In the next step, we plan to tackle some practical challenges for deploying PacketCloud. One prospective topic is the cloudlet deployment strategy. Several viable issues should be considered carefully as an integrated whole, including network topology, user behavior, and resource availability. We plan to seek for a robust, flexible, and evolvable strategy to satisfy both ISPs and third-party providers. The other possible trend is the economic issues. As PacketCloud introduces new financial links among different Internet entities, i.e., users, ISPs, and third-party providers, economic issues would be very interesting to investigate. We would like to develop an economic model to guide different Internet entities' financial investments.

6. ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1040043. We thank Dr. Kiran Nagaraja and Prof. Dipankar Raychaudhuri from Rutgers University for their comments.

7. REFERENCES

[1] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[2] T. Vu, A. Baid, and et al. Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet. In *Proc. of IEEE ICDCS*, 2012.

[3] F. Dogar, A. Phanishayee, and et al. Ditto: A system for opportunistic caching in multi-hop wireless mesh networks. In *Proc. of ACM Mobicom*, 2008.

[4] M. Li, D. Agrawal, and et al. Block-switched networks: A new paradigm for wireless transport. In *Proc. of NSDI*, 2009.

[5] X. Liu, X. Yang, and Y. Lu. To filter or to authorize: Network-layer dos defense against multimillion-node botnets. In *Proc. of ACM SIGCOMM*, 2008.

[6] Constantine Dovrolis and J. Todd Streelman. Evolvable network architectures: what can we learn from biology? *SIGCOMM Comput. Commun. Rev.*, 40(2):72–77, 2010.

[7] I. Stoica, D. Adkins, S. Zhuang, and et al. Internet indirection infrastructure. *IEEE/ACM Trans. on Networking*, 12(2):205–218, 2004.

[8] V. Sekar, N. Egi, and et al. Design and implementation of a consolidated middlebox architecture. In *Proc. of NSDI*, 2012.

[9] J. Sherry, S. Hasan, and et al. Making middleboxes someone else's problem: Network processing as a cloud service. In *Proc. of ACM SIGCOMM*, 2012.

[10] M. Satyanarayanan, P. Bahl, and et al. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[11] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *SIGMOBILE Mob. Comput. Commun. Rev.*, 16(3):2–13, 2012.

[12] Netflix open connect content delivery network. `https://signup.netflix.com/openconnect`.

[13] A. Li, X. Yang, and et al. Cloudcmp: Comparing public cloud providers. In *Proc. of ACM IMC*, 2010.

[14] N. Laoutaris and P. Rodriguez. Good things come to those who (can) wait. In *Proc. of ACM HotNets*, 2008.

[15] M. Sherr, A. Mao, and et al. A3: An extensible platform for application-aware anonymity. In *Proc. of NDSS*, 2010.

[16] Matthew Edman and Paul Syverson. As-awareness in tor path selection. In *Proc. of ACM CCS*, 2009.

[17] V. Valancius, C. Lumezanu, and et al. How many tiers? pricing in the internet transit market. In *Proc. of ACM SIGCOMM*, 2011.

[18] J. Huang, Q. Xu, and et al. Anatomizing application performance differences on smartphones. In *Proc. of ACM Mobisys*, 2010.