A Tutorial on Network Latency and its Measurements

Minseok Kwon
Dept. of Computer Science, Rochester Institute of Technology

ABSTRACT

Internet latency is crucial in providing reliable and efficient networked services when servers are placed in geographically diverse locations. In order to support server selection in distributed computing, measuring accurate latency becomes extremely important. The trend of mobile computing like iPhone and Android-based smartphones only accelerates the importance of accurate latency measurement due to its nature of rapidly changing locations and interactivity. Accurately measuring network latency, however, is not an easy task due to lack of testing resources, the sheer volume of collected data points, the tedious and repetitive aspect of measurement practice, and clock synchronization. In addition, the time that latency is measured affects measurement results significantly due to network dynamics, volatile traffic conditions, and network failures. In this chapter, we discuss the techniques that use PlanetLab to measure latency in the Internet, its underlying infrastructure, representative latency results obtained from experiments, and how to use these measure latencies. We discuss 1) details of using PlanetLab for latency measurement, 2) the underlying infrastructure of the Internet that causes the discrepancy between local and global latencies, and 3) measured latency results from our own experiments and analysis on the distributions, averages, and their implications.

Keywords: *Network Latency; Application Response Time; Planetlab; Latency Profiling;*

INTRODUCTION

Internet latency is crucial in providing reliable and efficient networked services such as online retails (e.g., Amazon), multimedia streaming (e.g., Netflix), and social networking (e.g., Twitter). For example, Netflix runs its servers on the Amazon cloud in geographically diverse locations, and provides video streams from the server that can deliver the content to a client in the shortest time. In order to support this server selection in distributed computing, measuring accurate latency becomes extremely important. The trend of mobile computing like iPhone and Android-based smartphones only accelerates the importance of accurate latency measurement due to its nature of rapidly changing locations and interactivity. Accurately measuring network latency, however, is not an easy task due to lack of testing resources, the sheer volume of collected data points, the tedious and repetitive aspect of measurement practice, and clock synchronization. In addition, the time that latency is measured affects measurement results significantly due to network dynamics, volatile traffic conditions, and network failures. Hence, it is critical to measure latencies over a wide span of time and days to acquire a representative and comprehensive view.

In the literature, latency measurement has been studied extensively. For accurate measurements, end hosts at strategic locations are used (Francis, et al., 2001), latency is estimated based on coordinated using landmark hosts (Ng & Zhang, 2001) or DNS queries (Gummadi, Saroiu, & S., 2002), or routing topology (Dabek, Cox, Kaashoek, & Morris, 2004) (H. Madhyastha, 2006). Measured latencies also help select target servers to achieve specified optimization goals in routing, content distribution networks, and cloud computing (Wendell, Jiang, Freedman, & Rexford, 2010) (Khosla, Fahmy, & Hu, 2012) (Ding, Chen, T., & Fu, 2010). Their objective function is to minimize the total execution time and select best replica servers considering server loads, cost and locations. Recently, mobile-cloud computing has gained considerable attention in which a mobile device offloads its computation to servers in the cloud seamlessly and transparently. Again, choosing right servers is critical to lower execution time, and we can use profiling that estimates execution time as a function of latency, loads, program, and network conditions.

The primary focus of this chapter is to discuss the techniques that use PlanetLab (PlanetLab, 2014) in order to measure latency in the Internet, its underlying infrastructure, representative latency results obtained from experiments, and how to use these measure latencies. As PlanetLab provides a world-scale network testbed, it has become a popular platform for network measurement. Based on our own experience, we discuss details regarding PlanetLab: 1) the fundamentals of PlanetLab, 2) how to establish a PlanetLab site and connect to its network, 3) how to manage an account, nodes, and slices, 4) how to run a server and manage tens or hundreds of servers, and 5) how to collect measurements like latency. Our results imply significant discrepancy between global and local latencies. Here, global latencies refer to latency of a long distance connection like transoceanic network link while local latencies are network delay between regional end-hosts. We look into the causes that make such discrepancy; this helps understand the underlying cause. Additionally, we present measured latency results from our own experiments and analysis on the distributions, averages, and their implications.

This chapter is organized as follows: First, we discuss the basics of PlanetLab and using PlanetLab for network latency measurement purposes. Second, we discuss the underlying network infrastructure that causes different latency measurements for global and local connections. Third, we report measured latencies from our experiments and analyze their causes and effects. Finally, we conclude this chapter with discussion on using measured latencies for applications and systems.

LATENCY versus THROUGHPUT

What is Latency?

It would be ideal if data in the Internet services can move from one node to another instantaneously at the speed of light. This is, unfortunately, not possible in reality. As a packet travels from a host (the sender) to subsequent routers or another host (the receiver), the packet experiences different types of delays at hosts, routers, and network links. Those delays include 1) processing delay, 2) queuing delay, 3) transmission delay, and 4) propagation delay. A host or a router needs time to process an incoming packet (processing delay) for packet forwarding such as reading a packet header and searching the routing table to determine the next hop. The packet also often needs to wait in the queue to be transmitted onto the link (queuing delay). Transmission delay (also known as the store-and-forward delay) is the time incurred as a node pushes all the bits of the packet onto the link. Finally, propagation delay is the time required for a bit to propagate over the link from a node (either host or router) to another. End-to-end latency is the sum of such delays incurred either at the end-hosts or in transit. When it comes to end-to-end latency, note that an end-host can impose significant delay as well, e.g., dial-up modems, cable modems, and DSL.

What is Throughput?

Another important and closely related performance metric is throughput that indicates the rate of file transfer. End-to-end throughput is defined as the amount of data received at the destination per unit time. For instance, when it takes T seconds to transfer all M bits of a file, the average throughput is M/T bps. Throughput is closely related to link bandwidth because higher throughput can be achieved often when larger link bandwidth is available. In most cases, an end-to-end network path consists of multiple hops, and the bottleneck link with the smallest bandwidth among those multiple hops determines the end-to-end throughput. We can also define instantaneous throughput as the rate at which a receiver receives data at any given time.

Relationship between Latency and Throughput

TCP mitigates network congestion by reducing its sending rate, and increases throughput as it sends more data when more network capacity becomes available. One critical component is self-clocking that synchronizes the sending rate and the rate at the bottleneck link. TCP infers the bottleneck link capacity by observing the rate of acknowledgments returned by the receiver (Fahmy & Karwa, 2001). The TCP congestion control mechanism affects end-to-end latency significantly since it changes the sending rate dynamically. There have been attempts to understand the effects of TCP on throughput, latency, and their relationships (Padhye, Firoiu, Towsley, & Kurose, 1998). TCP throughput is computed as a function of round-trip time (*rtt*) and the packet loss rate (*p*): $\boldsymbol{hroughput} = \frac{\sqrt{1.5}}{rtt\sqrt{p}}$. Note that *rtt* is the end-to-end round-trip time computed as the sum of end-to-end latency in both directions, and *p* is the one-way packet loss rate. The throughput function implies that throughput increases in inversely proportion to latency and vice versa.

End-to-End Latency Dynamics

The behaviors, dynamics, and effects of end-to-end latency have been studied from routing and TCP's perspectives. In a comprehensive measurement study (Paxson, 1997), Paxson finds that a significant portion of end-to-end latency suffers from several non-negligible routing pathologies including routing loops, erroneous routing, infrastructure failures, mid-stream altered connectivity, and temporary outages. Another similar study (Savage, Collins, Hoffman, Snell, & Anderson, 1999) discovers that in 30-80% of the cases, end-to-end latency can be shortened if packets follow an alternate path with superior quality.

Paxson also finds in another study that variations in end-to-end latency indicate congestion periods that span over a wide range (Paxson, 1996).

TRANSPORTING DATA OVER THE INTERNET

In this section, we will be discussing how data is transported over the Internet. First, a brief description of the TCP/IP (Transmission Control Protocol / Internet Protocol) will be provided, which is the backbone of all Internet communications. We will also provide a discussion on the providers of data transportation: Internet Service Providers (ISPs) and Telephone Service Providers (TSPs). We will elaborate on how these entities affect the aforementioned network latency and network throughout.

TCP/IP Protocol

The Internet uses the TCP/IP protocol for communication between its entities such as hosts on the network edge and routers at the network core. More protocols (other than TCP and IP) are involved in Internet communications, but TCP and IP are considered the most critical ones. The original development of TCP/IP was funded by DARPA, an agency of the United States Department of Defense around 1960-70, e.g., ARPANET was launched in 1966. Now IETE (Internet Engineering Task Force) maintains TCP/IP and related protocols whose details are available in RFCs. The main goal of TCP/IP is to provide internetworking technologies that enable communications between two hosts on different networks. Their design principles include 1) packet switching, 2) autonomy (no internal changes required to interconnect networks), 3) the end-to-end principle, 4) minimalism and a best-effort service model, 5) stateless routers, and 6) decentralized control. The end-to-end principle states that complexity and intelligence should be put at the edges (hosts), not at the network core. This principle is indeed related to minimalism, a best-effort service model, and stateless routers. TCP/IP defines today's Internet architecture, and specifies how data should be addressed, end-to-end connectivity is provided, and packets are routed.

In an effort to tackle the complexity of the network system, the TCP/IP protocol adopts a layering structure with abstraction of services. Each layer is service encapsulation that isolates from other layers (protocols) and hides their details, so that designing and maintenance become significantly simple. From top to bottom, there are application, transport, network, data link, and physical layers. In the application layer, a user runs network applications and application-layer protocols such as Web and HTTP; the transport layer provides end-to-end message (also known as segments) delivery in two modes—reliable (TCP) and unreliable (UDP) data transfers; the network layer is responsible for transmitting packets (also known as datagrams) from one node to another (either hosts or routers), so that a datagram can travel from the source to the destination through a sequence of routers; the link and physical layers help transmit datagrams actually over the link within a same network. As a packet goes through these protocols stacks top to bottom at the sender, each layer prepends additional information (header), and this header is stripped off as the packet arrives and goes through the protocols stack backward at the receiver.

The network layer, specifically the IP protocol, in the Internet protocol suite has two main functions, namely forwarding and routing, which contribute to reducing network latency. In forwarding, a router passes a packet arrives at the input link to the relevant output link toward the next hop (either router or host). To forward the packet, the router needs to search a forwarding table for the target outgoing interface (next hop) using the destination IP address as the keyword. This search must be performed swiftly, ideally faster than the link speed; otherwise, arriving packets will be queued and eventually dropped as the queue becomes full. Since 10-100 Gbps link bandwidth at the backbone networks is not far-fetched, the routers should be able to process and forward packets at least at a comparable speed. Routing algorithms also reduce network latency as they compute the shortest paths from the source to the destination. Popular protocols are OSPF and RIP. In these protocols, each router communicates with neighboring routers, and finds collectively the shortest paths from itself to all the

other destinations using Dijkstra's shortest path algorithm or distance-vector routing in graph algorithms. TCP runs on hosts at the network edge only, and helps reduce latency by controlling the sending rate as it predicts incipient network congestions. One challenge is the lack of information and signals from the core network regarding the network conditions (e.g., ICMP source quench packets are no longer in use). With no help from routers, TCP infers the network conditions and congestion by analyzing the pace of returned acknowledgments from the destination. Specifically, the sender aligns its sending rate with the acknowledgment rate (called self-clocking) in order to synchronize its sending rate to the bottleneck link rate. Additionally, TCP changes the amount of data it can send at one time (sliding window) depending on the inferred network congestion—more rapidly initially, and slowly as it nears the full capacity.

Data Transportation over Cable (DOCSIS Standard)

Cable operators, such as Time Warner Cable, transport internet data over the same cable that is used to transport the cable TV signals. The standard that allows this is *Data Over Cable Service Interface Specification (DOCSIS)*. Cable operators already own a high speed HFC (hybrid fiber coaxial) network, which is good enough for transporting internet traffic up to 100 Mbps. The DOCSIS standard went through multiple revisions over the past two decades, starting at DOCSIS 1.0 in 1997. With the introduction of DOC 2.0, internet speeds upto 10 Mbps became available. The newest standard, at the time of the preparation of this document, is DOCSIS 3.0 which allows internet speeds up to 50 or 100 Mbps. With such speeds, users can download movies over the internet without any slowdown and watch them in real-time. DOCSIS3.1 is expected to reach up to Gbps speeds and exceed it.

Data Transportation over Telephone Networks

Today most people purchase a smartphone (e.g., iPhone or Android phones) with a data plan through which a user can send videos, audios, or other kinds of data. Data are transported over wireless phone networks such as 3G, 4G, 5G, and LTE. 3G is the third generation of mobile telecommunications technology whose standards are defined by IMT-2000. The specifications of IMT-2000 are in turn defined by the ITU (International Telecommunication Union). While ITU does not clearly specify the data rate for 3G, its data rates usually range from 200 kbps to 2 Gbps, and the applications span voice over IP, Internet access, video conferencing, and even mobile TV. Successful 3G standards include UMTS in Europe, Japan, an China, CDMA2000 in North America and South Korea, and EDGE (Cingular in the United States). 4G is the next generation mobile telecommunication technology succeeding 3G, and provides ultra-broadband Internet access. The target applications include high-definition mobile TV, 3D TV, cloud computing, and video games. The 4G standards (IMT-Advanced) require the data rates to be 100 Mbps and 1 Gbps for high and low mobility communications, respectively. 4G technologies use MIMO, OFDM, and OFDMA for physical layer transmission, and support Mobile IP and IPv6. Early successful versions are 3GPP LTE, Mobile WiMAX, LTE Advanced.

DSL is arguably the most successful technology that transports data over telephone networks in the wired networks. In the United States, about a half of residential broadband access use DSL, and in some countries in Europe, more than 90% of the residential connections are DSL. By using higher frequency bands, the DSL service can share the telephone line with landline phone service. The data rate of DSL ranges from 256 kbps and can reach even over 100 Mbps downstream. Access networks or technologies including DSL are critical in determining end-to-end latency. Imagine that backbone and regional networks enjoy plenty of available network bandwidth and uncongested network conditions due to the efficiency and correctness of the TCP and IP protocol. Even in this case, if the last mile capacity or network conditions in access networks are not sufficiently large, an application suffers from long end-to-end latency.

Internet Service Providers (ISPs)

Computers (also called end hosts or end systems in networking) are connected to the Internet through Internet Service Providers (ISPs). Examples of ISPs are AT&T, Sprint, Verizon, ComCast, and Time Warner. There are ISPs at several different levels—ISPs that run the backbone networks, corporate ISPs, university ISPs, residential ISPs, and ISPs that provide WiFi hotspots at airports, coffee shops, hotels, and even on the streets. In many cases, these ISPs are cable or telecommunication companies, e.g., in Rochester, NY, residential or small business users can access the Internet via Time Warner Cable or Frontiers (phone). The lower-tier ISPs enable residential broadband network access via DSL or cable modem, high-speed local area network access, wireless access (WiFi), or even relatively slow dial-up modem access to customers. These lower-tier ISPs are connected to upper-tier ISPs that usually is responsible for running the core network in which high-speed fiber optic links (1-10 Gbps) are connected through high-speed routers nationally and internationally.

As indicated earlier, ISPs are organized hierarchically as they provide access to the Internet. A tier-1 ISP runs Internet backbones, a tier-2 ISP is in charge of regional or national coverage and needs a service from a tier-1 ISP, a tier-3 ISP has a smaller area coverage and is a customer of a tier-2 ISP, etc. An ISP is connected to other ISP via a Point of Presence (POP) that is a set of routers to which other ISP's routers can connect. In general, when an ISP wants to access a larger area, the ISP establishes a contract with an upper tier ISP, leases communication links, and connect its routers to the upper tier ISP's routers through POPs. In this contract, the upper tier ISP becomes a provider and the lower tier ISP is a customer. If two ISPs are connected at the same tier level, they are peers with each other.

Telephone Service Providers (TSPs)

A telephone service provider (TSP) provides traditional telecommunication services to users. The examples are Verizon, AT&T, Vonage in the United States, and British Telecom in Britain. Although many of them also provide Internet services, TSPs usually mean only phone-related communication services excluding Internet or TV cable or satellite services. Interestingly, portions of the local wired phone infrastructure are heavily utilized in order to provide access to the Internet. These access technologies include dial-up (mostly in 1990s) and DSL (Digital Subscriber Line, often called broadband access) that use phone lines to access the Internet ISP networks. If a household buys a DSL line, it uses the phone network for both data and voice signals. Today, more popular technologies are wireless communication services like WiFi, 3G/4G, and WiMAX. WiFi (or Wireless LAN or IEEE 802.11) is an inexpensive technology where a user can send and receive data through an access point that is connected to the wired Internet. One weakness of WiFi is its limited coverage within a few tens of meters of the access point. TSPs also provide wide-area wireless Internet access using 3G/4G cellular technologies and base stations at speeds in excess of 1Mbps.

USE OF PLANETLAB FOR NETWORK LATENCY MEASUREMENT

Network traffic measurement is useful to accurately diagnose the causes of phenomena experienced in the network without clear reasons. The accurate diagnosis is later used for addressing anomalies or improving network performance. The measurement usually includes throughput (or bandwidth), latency (or delay, round-trip time), packet loss rates, network topology, packet jitters, traffic types, link utilization, and energy usage. The analysis of these data sets helps give insights on network behaviors and structures, its causes, and reasons behind perceived phenomena. Discoveries from network measurements can also inspire novel systems and algorithmic techniques that solve current obstacles.

It is, however, not easy to collect real measurement data from a large-scale operational networks. To this end, network researchers should have access to computing resources all over the world that allow experimental systems to run disruptive technologies. ISPs (Internet Service Providers) understandably are

not fond of disruptive technologies, and therefore are not supportive of these measurement activities. Motivated by this challenge, PlanetLab (PlanetLab, 2014) was created to provide a worldwide network test-bed to network researchers, so that they can try out their experimental systems and algorithms without disrupting operational networks. We will look into the details of PlanetLab, and how to measure latencies using PlanetLab in this section.

What is PlanetLab?

PlanetLab consists of over a thousand machines dispersed around the world providing a large-scale network-testing environment to researchers. All the machines are connected to the Internet, and they are hosted by research institutions, universities, and routing centers (a few of them). Researchers use PlanetLab to conduct experiments with new systems and services under real-world conditions. These systems and services include multicast overlays, content distribution networks, network-embedded storage, distributed file sharing, and network measurement. The long-term vision of PlanetLab is beyond a simple network test-bed serving as a deployment platform that enables a seamless transition from early prototype to operational systems. Three benefits are highlighted: 1) a large set of machines distributed in different geographical locations, 2) a realistic network substrate that experiences congestion, failures and link behaviors, and 3) a realistic client workload.

A PlanetLab machine runs a Linux operating system together with mechanisms for remote bootstrapping, distributing software updates, system management, auditing malicious activities, key distribution, and user accounts. The software is bundled as a package called MyPLC. The software supports distributed virtualization, so that an application can run on multiple PlanetLab machines at any given time while other applications are running on the same machines. All of the functions in MyPLC and a PlanetLab machine are controlled remotely. This helps easy management and lowers security risks.

Another goal is to understand how the next generation Internet should be designed to support overlay networks and disruptive technologies. This idea is ironically inspired by the successful Internet that is subject to ossification like other commercially successful systems, e.g., UNIX. The ossified Internet hampers network researchers' new attempts to evolve the underlying protocols and mechanisms in the Internet to solve problems (e.g., security vulnerabilities) and address new challenges (e.g., real-time high-volume data transfer). Overlay networks are flexible enough to try out new protocols and capabilities without modifying the core network functions. If these disruptive technologies turn out to be useful on the PlanetLab overlay, they can be adopted as a new feature at commercial routers; if they are complex despite its usefulness, they can continue to be provided as a part of overlays.

How to Use PlanetLab?

When you have a project that you want to use PlanetLab for, a PI (Principal Investigator) in your institution (either you or someone else) first needs to create a PlanetLab site. A site is a physical location where PlanetLab nodes are located; a node is a dedicated server that runs PlanetLab services. Creating a site requires two actions: 1) signing the consortium membership agreement and 2) connecting two or more nodes to PlanetLab. The PI is responsible for 1) overseeing all slices, 2) account management, and 3) node management at your site where a slice is a set of allocated resources distributed across PlanetLab (a UNIX shell access to many nodes). For example, the PI is responsible for addressing any complaints on malicious activities originated from one of your slices, creating and deleting slices, assigning users to slices, enabling and disabling user accounts, and physical maintenance of the nodes at your site.

Once your site is up and running, you can sign up for an account to start using PlanetLab. In your registration process, you are required to designate which site you belong to. It is important to create an SSH key pair for authentication since remote access to PlanetLab nodes requires SSH login using RSA authentication. After you generate the key pair, you keep your private key file and upload the public key to the PlanetLab website to be populated to other nodes. This SSH key setup helps automate the login

process to a group of nodes when you populate your slice to those nodes. Without this setup, you need to login manually each machine, which is practically impossible.

Now you can ask your PI to create a slice for you, or connect your account with an existing slice. After associating with a slice, it takes up to an hour for your slice to be created on all nodes and the public key to propagate to those nodes. Then you can start accessing your slice created on all nodes (of course, except down ones). You log in the node using your slice name as the login name, e.g.,

ssh –l princeton_test1 –I ~/.ssh/id_rsa planetlab-1.cs.princeton.edu.

The machine you will get is a minimal Fedora Core 8 Linux installation with basic software and system libraries installed. You can su to root, add new users, install new packages, and mount directories, but cannot do most privileged administrative operations like network configuration. Note that your slice is set to expire after two months. You will be notified of any upcoming expiration, and can extend the expiration date of your slice by renewing the slice. There are also limits on resources including disk space, memory usage, file descriptors and bandwidth that are assigned on a per-slice basis.

How to Measure Latency with PlanetLab?

When you want to measure latency (or other performance metrics) using PlanetLab, the basic mechanism is to use network programming via sockets at application layers, measure latency between two endpoints, and extend it to a set of nodes in PlanetLab. A major challenge in this process is scalability while extending the experiment to tens to hundreds of machines. One critical preliminary step is to create a list of machines that are alive. To this end, you can utilize the information available on the PlanetLab website that maintains a list of nodes available. Unfortunately, some of these supposedly available nodes do not respond for unknown reasons depending on your locations. Hence, the only way to ensure whether a node is accessible is to manually check the node by using either SSH or ping. This is a tedious but necessary process. After this step, we can login each node using the SSH command discussed earlier with our login name (or slice name). Making use of scripts (e.g., UNIX shell scripts) would help automate the whole process. For instance, we can write a shell script to SSH each node on the list, install packages on each node again using SSH, and then run an application there on each node. Once the experiments are complete, we can collect results by transferring files from each node to the main server. This collection and analyzing the results can all be done automatically again using scripts.

UNDERLYING INFRASTRUCTURE

As we measure latencies, we naturally wonder what network links and points (i.e., routers or switches) packets go through resulting in such measurements. In this section, we briefly discuss the underlying network infrastructure that helps understand the measured latency results. The questions that we will address are: 1) Which network links and components contribute to long latency? 2) What is the bandwidth and latency of a long-distance network link? 3) How is a packet delivered between two continents? and 4) What delay and loss events occur and why?

Why are Global and Local Latencies Different?

Local area networks (LANs) are connected to the backbone network of its Internet Service Provider (ISP). This connection point is called a Point of Presence (POP). These POPs are then connected to high-level networks via Network Access Points (NAPs), and NAPs are in turn connected to the Internet backbone networks. As local latencies are the sum of link propagation delay and transmission delays at switches, global latencies include relatively long propagation delay at backbone links and delays at other connection points such as POP, routers, and NAPs.

Today, 99% of international traffic travels over submarine communication cables under the sea while only 1% utilizes overseas satellite links. The submarine cables provide considerably more reliable communication with multiple backup paths. The bandwidth of submarine cables is higher than that of

satellites, and the propagation delay is smaller. However, laying submarine cables on the seabed costs more than satellites. The bandwidth has increased drastically recently from 100Gbps to a few terabits per second with the advance of fiber optics cable technology. Satellite links are also used as an alternative route when submarine communication cables fail for some reason. Repeaters connect point-to-point submarine cable links and amplify light signals in transit.

As expected, global latencies are significantly larger than local ones simply because of its longer propagation and transmission delays. Here by global latencies, we mean coast-to-coast in the United States or transcontinental connection latencies, and local latencies refer to connection delay within same regions like a same network or a same ISP. For example, round trip time over the transcontinental connections go easily over 100ms while it usually takes less than 30ms for local latencies (Markopoulou, Tobagi, & Manour, 2006). Following is a traceroute result from a computer in Rochester, NY, in the United States to a website in South Korea:

```
 1  * * 172.16.0.1 (172.16.0.1)  3.534 ms
 2  bundle1.rochnyhly-ubr02.nyroc.rr.com (67.246.240.1)  44.703 ms  26.420 ms  40.392 ms
 3  gig9-5.faptnyal-rtr002.wny.northeast.rr.com (24.93.9.78)  11.023 ms  12.773 ms  15.652 ms
 4  rdc-72-230-153-12.wny.east.twcable.com (72.230.153.12)  29.129 ms *  18.283 ms
 5  rdc-72-230-153-243.wny.east.twcable.com (72.230.153.243)  35.792 ms  31.104 ms *
 6  be45.cr0.chi10.tbone.rr.com (107.14.19.106)  37.432 ms *
    ae-3-0.cr0.chi10.tbone.rr.com (66.109.6.72)  34.341 ms
 7  ae-6-0.cr0.sjc30.tbone.rr.com (66.109.6.14)  82.626 ms  80.971 ms  83.404 ms
 8  ae-0-0.pr0.sjc20.tbone.rr.com (66.109.6.139)  81.536 ms  82.834 ms  88.399 ms
 9  66.109.10.206 (66.109.10.206)  83.414 ms *  82.225 ms
10  112.174.87.153 (112.174.87.153)  202.003 ms  203.643 ms  206.637 ms
11  112.174.83.161 (112.174.83.161)  220.072 ms
    112.174.83.73 (112.174.83.73)  406.919 ms
    112.174.83.161 (112.174.83.161)  206.341 ms
12  112.174.8.221 (112.174.8.221)  260.756 ms
    112.174.48.153 (112.174.48.153)  210.390 ms
    112.174.8.125 (112.174.8.125)  205.498 ms
13  112.174.63.74 (112.174.63.74)  596.896 ms
    112.174.22.226 (112.174.22.226)  207.012 ms
    112.174.23.14 (112.174.23.14)  610.016 ms
14  112.188.240.54 (112.188.240.54)  429.317 ms  211.567 ms *
15  112.175.105.10 (112.175.105.10)  311.066 ms  726.723 ms  226.981 ms
```

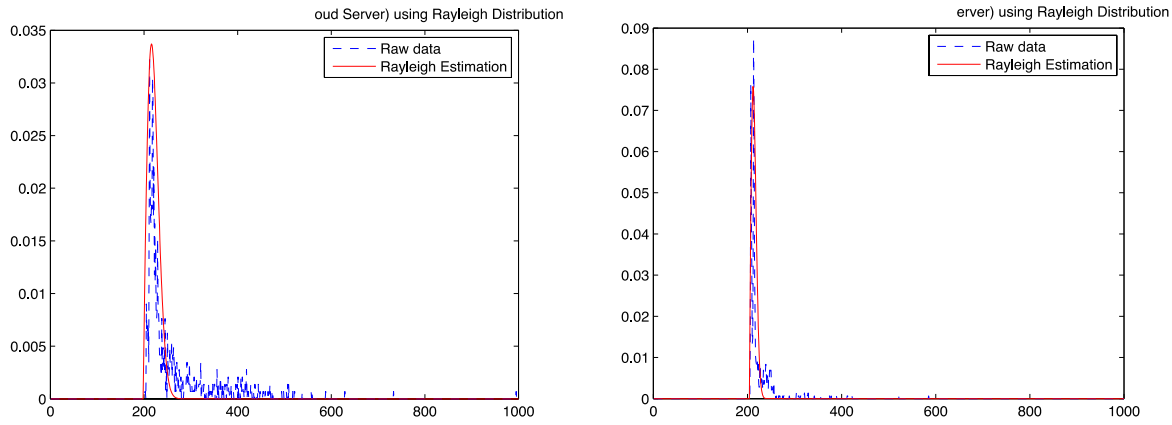
In the above example, delay from Hop #9 66.109.10.206 (Virginia, US) to Hop #10 112.174.87.153 (Seoul, Korea) is approximately 120ms, which accounts for the transpacific portion of this connection.

Delay and Loss Events and Their Causes

Prior studies discover that a non-negligible percent of network connections experience on occasion extremely long delay and high loss rates even in the backbone network **(Paxson, End-to-End Routing Bahavior in the Internet, 1996) (Savage, Collins, Hoffman, Snell, & Anderson, 1999)**. Moreover, it is possible to find alternatives shorter than the current routes for a substantial percent of the routes computed by routers. The probable causes are packet drops in the buffer (packet losses), routing anomaly (suboptimal routes), router reconfiguration or errors in router configuration (packet losses or suboptimal routes), and link failures (packet losses or suboptimal routes). These events occur periodically, and in particular are detrimental for real-time traffic and applications like VoIP and video streaming.

LOCAL and GLOBAL LATENCY

In this section, we discuss communication latency using real measurements in the cloud environment, and study the measurements through statistical analysis. Our goal is to gain insight into the performance of the cloud through these measurements and analysis. This analysis enables us to optimize our algorithms and system design based on the observed latencies and processing times.



(a) (b)

Fig. 1. *Histograms of latencies at Oregon datacenter on the weekend (a) and a weekday (b).*

In order to measure latencies over wired connections, we ran a simple program that sends ping packets from a client computer to servers in cloud datacenters in January and February 2012. The client computer was located in Rochester, New York, in the United States, and we used the five datacenters available in AWS (AWS, 2012), which are all located in geographically different regions, namely in Virginia and Oregon in the United States, Ireland in Europe, Sao Paolo in South America, and Singapore in Asia. We ran two sets of experiments, one on a weekday and the other over the weekend, to collect data with different network traffic conditions. In each experiment, we measured the round-trip time once every minute for 24 hours (thus the total number of latencies in each experiment is $6 \times 24 = 1{,}440$.

TABLE I

Average and standard deviation of latencies over wired connections (in millisecond). Virginia, Oregon, Ireland, Sao Paolo, Singapore

Measured Time	Weekend					Weekday				
Datacenter	VA	OR	IRE	SAO	SING	VA	OR	IRE	SAO	SING
Mean	122	322	294	389	580	42	223	196	389	546
Std Dev	124	525	201	166	242	18	41	25	166	34

Table I shows the mean and standard deviation of these latencies for the AWS datacenters. The latencies to the Virginia datacenter are the shortest while those to Singapore are the longest, as easily conjectured based on their geographical distances from Rochester. In Figure 1, we show the histograms of latencies for the Oregon AWS datacenter on both the weekend and a weekday, where the x-axis denotes the latencies and the y-axis represents the frequencies of such latencies. The latency distributions for all of the other four datacenters behave similarly to the Oregon data, and they are omitted here due to space limitations (refer to (Project, 2012) for all the measurement data). Note that we limit the range of latencies from 0 to 1000ms. While there were a few latencies over 1000ms, they were negligible. In our measurements, the weekend data show higher variance than the weekday data. While we can surmise a variety of causes for this difference including a local network failure, a high volume of traffic at a certain point of the network, etc., our goal here is obtaining real measurement data to enrich our simulations rather than measuring comprehensive data sets and identifying their causes. All of the histograms show a virtually identical shape, which is similar to a Rayleigh distribution with a peak close to the average and

then gradually decreasing as latency increases (Wikipedia, 2012). The figure also shows the x and σ values used for the Rayleigh distribution (refer to Equation 1 for details).

TABLE II

Average and standard deviation of latencies over wireless connections (in millisecond). Virginia, Oregon, Ireland, Sao Paolo, Singapore

Measured Time	Weekend					Weekday				
Datacenter	VA	OR	IRE	SAO	SING	VA	OR	IRE	SAO	SING
Mean	253	389	293	434	697	930	817	798	872	1061
Std Dev	470	635	520	704	1278	595	710	915	1079	2060

In addition, we measured latencies from a mobile device to servers running at these five different datacenters of AWS. We used an Android phone for the mobile device, and tested over both WiFi and 3G communication links. Similar to the wired connection experiment, the program on the Android phone periodically sent ping packets to the servers (200 times), and recorded their round-trip time. Table II presents the averages and standard deviations of these round-trip times. As expected, the latency to Virginia is the lowest and the one to Singapore is the highest. The latencies of WiFi are smaller than those of 3G, and both are larger than the latencies of wired connections (see Table I). Figure 2 depicts these latency distributions for the AWS Oregon datacenters. These latencies can be approximated by a Rayleigh distribution, as with the latencies over wired connections, although in this case the fit is not as good.

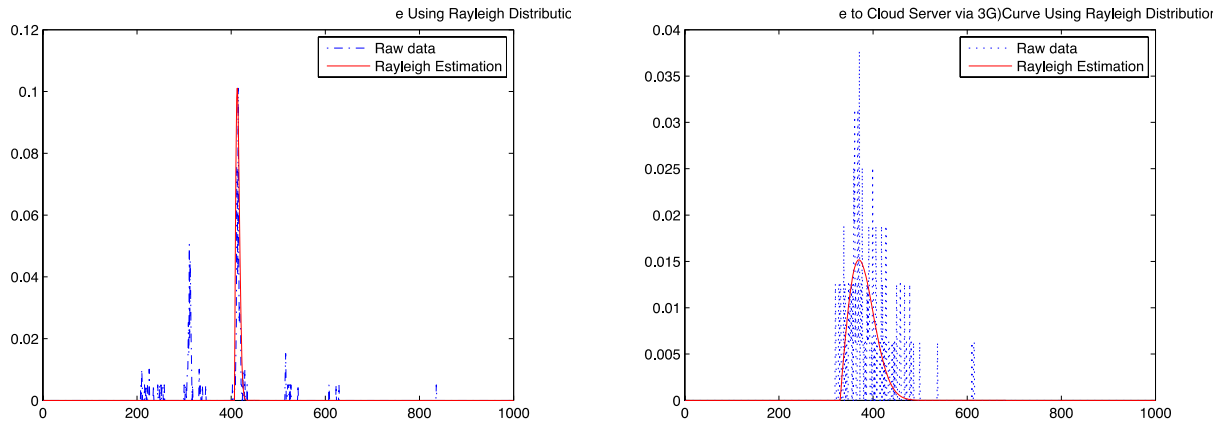


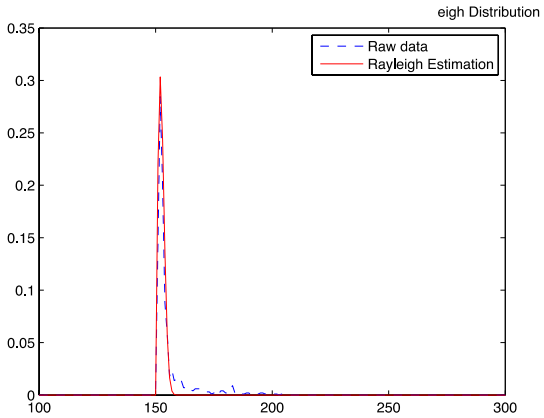
Fig. 2. *Histograms of latencies at Oregon datacenter via WiFi (a) and 3G (b).*

We also measured the processing times of different AWS instances at these five datacenters. In theory, the same kind of instances at different datacenters should have approximately the same processing power. Of course, the processing time is different for different programs. Our goal here is to measure relative and comparative processing powers of these instances. We executed an odd-even transposition sorting program with $O(n^2)$ time complexity for n=10,000 1,000 times and averaged the measured data (see (Knuth, 1973) for the detailed algorithm). As shown in Table III, the average processing time of the micro instance is much larger than that of the small instance, and the time of the small instance is nearly double that of the large instance regardless of the datacenter. As explained on the AWS web site (AWS, 2012), the micro instance is well-suited for a process in short bursts, but not for long consistent computing. This characteristic contributes to a high standard deviation for the micro instance, e.g., 12,221 at Virginia, and results in high and inconsistent average processing time as well.

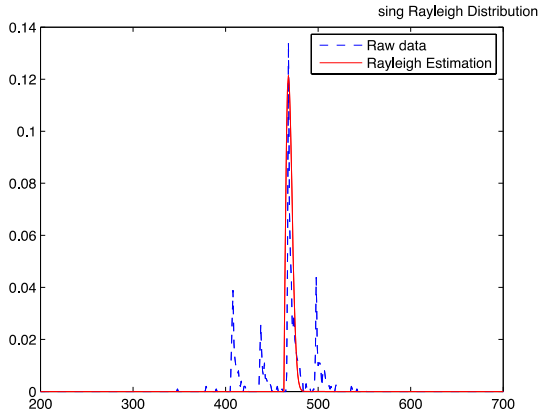
TABLE III

Cloud server processing time (in milliseonds)

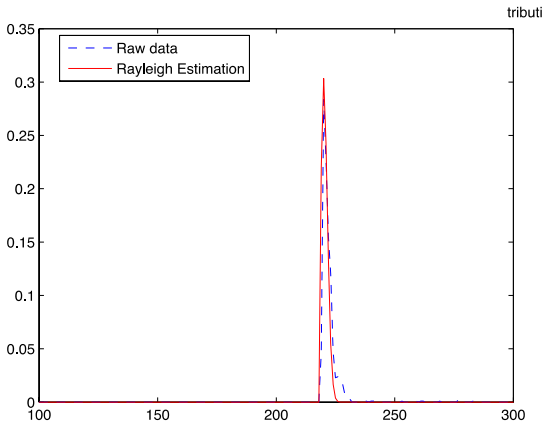
Instances	Virginia	Oregon	Ireland	Sao Paolo	Singapore
Micro	11,267	4,297	8,166	3,947	3,221
Small	1,565	1,565	1,560	1,557	1,647
Medium	744	751	706	751	712



(a)



(b)



(c)

Fig. 3. *Histograms of processing time for the micro (a), small (b), and medium (c) instances in Virginia for the lightweight application.*

In addition to the mean, we study the distributions of the processing times as well. In Figure 3, we show the distributions of processing times for three kinds of AWS instances (micro, small and medium) measured at the Virginia datacenter for a lightweight application. Figure 4 illustrates the processing time distribution for a heavyweight application only for the micro instance at the Virginia datacenter. We use the same sorting application with n=5,000 for the lightweight and n=10,000 for the heavyweight applications. Again, the distributions of the other four datacenters and the rest of the data sets for the

heavyweight application are available at (Project, 2012). The results show that the processing times for the micro instance are fairly consistent for the lightweight application while they are varied with a few bursts for the heavyweight application, as explained on the AWS web page (AWS, 2012). The processing times at the small instance closely follow a Gaussian distribution with several anomalies and bursts, and the times for the medium instance concentrate on a small set of data points for both the lightweight and heavyweight applications.

Another distribution depicted in Figure 5 compares the processing times measured with applications with different sizes. We used the same sorting application with n=800, n=3,000, and n=8,000. The figure only shows the processing times measured on the small instance; the data for the micro and medium instances are available at (Project, 2012). All of these data were measured in the Virginia datacenter on a weekday. The figure indicates that the processing times become varied as the size and computing time of the application increases for the small instance (a similar phenomenon was observed for the micro and medium instances). As more computation is required, the performance is more vulnerable to external conditions like available computing resources, the number of other jobs, and network bandwidth.

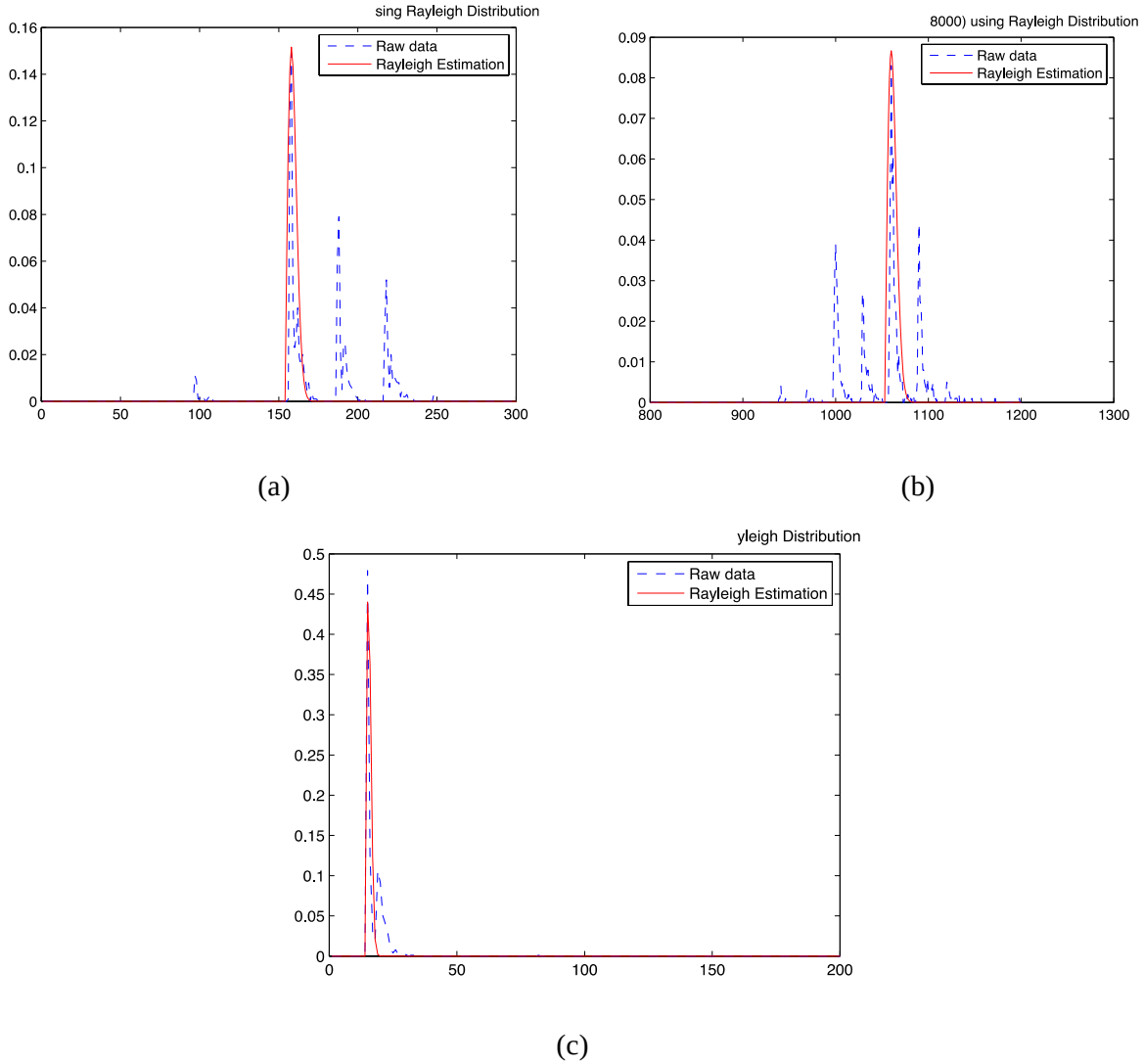


Fig. 4. *Histograms of processing time for different application sizes on the small instance at the Virginia datacenter.*

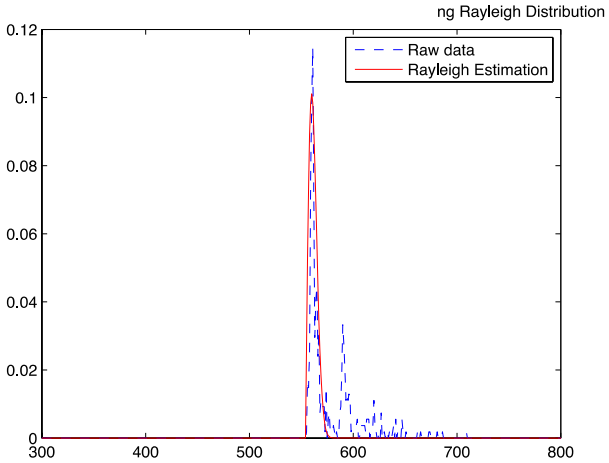


Fig. 5. *Histograms of processing time for the micro instance at the Virginia datacenter for the heavyweight application.*

As implied previously, we have found that all of the latency distributions, namely 1) cloudlet to the cloud and 2) mobile to the cloud over either 3G or WiFi, can be approximated (some better than the others) by a Rayleigh distribution with a shifted x-axis. The processing time distributions under different loads are also similar to a Rayleigh distribution. The probability density function of a Rayleigh distribution is

$$\frac{x}{\sigma^2} e^{x^2/2\sigma^2}, \quad x \geq 0 \qquad (1)$$

where σ denotes the shape of the distribution (either narrow or wide) and x adjusts the distribution into a different location along the x-axis. We fit curves to the measured data by adjusting the values of σ and x in the Rayleigh distribution to best match the measured data, and we use these models for latency in our simulations in the next section.

Interestingly, we can observe several sub-distributions in some of the probability curves from the measured data, e.g., the mobile to cloud latencies via WiFi for the Oregon datacenter, and some processing time distributions—Figure 3(b), Figures 5(b) and (c). Each sub-distribution is also well-fitted to a Rayleigh distribution. We surmise that locality exists in both latency and processing time due to more than one route in the network and workload locality with the CPUs.

The cloud can be modeled as a graph using the vertices as cloud server locations and the edges as the network latencies (Soyata & Friedman, 1994) (Soyata, Friedman, & Mulligan, 1993) (Soyata & Friedman, 1994) (Soyata, Friedman, & Mulligan, 1995) which will allow sophisticated combinatorial optimization algorithms for task scheduling in the cloud (Soyata & Friedman, 1994) (Soyata, Friedman, & Mulligan, 1997) (Soyata T. , 2000). Additionally, rather than pure-software methods, hardware accelerators can be utilized for reducing the computational latency (Li, Ding, Hu, & Soyata, 2014) (Guo, Ipek, & Soyata, 2010) (Soyata & Liobe, 2012). While this will provide a speed-up and mitigate some of the negative impact of the network latencies, the next section focuses on the general effects of the latency on certain latency-sensitive applications.

PERFORMANCE IMPLICATIONS ON APPLICATIONS

To understand the performance implications of network latency, the authors of (Kwon, et al., 2014) compare three data partitioning and server selection algorithms, namely random, fixed, and greedy using estimated latencies. The random algorithm sends tasks to a group of randomly selected servers; the fixed algorithm distributes tasks evenly among servers; and the greedy algorithm selects the server that can finish the task as quickly as possible. In the greedy algorithm, the amount of time to be completed is estimated based on measured latencies in the profile. The results show the clear advantage of using latency profiling to achieve the minimal response time.

Applications that benefit from latency profiling include bandwidth-hungry or latency-sensitive ones (Page, Kocabas, Soyata, Aktas, & Couderc, 2014), (Wang, Liu, & Soyata, 2014), (Soyata, et al., 2012) (Fahad, et al., 2012), (Hassanalieragh, Soyata, Nadeau, & Sharma, 2014), (Nadeau, Sharma, & Soyata, 2014). An example of a bandwidth-hungry application is a long-term health monitoring system using homomorphic encryption (Kocabas O. , et al., 2013), and an example for the latter is face detection or recognition (Soyata, Muraleedharan, Funai, Kwon, & Heinzelman, 2012).

Face detection and recognition applications are tested in a mobile-cloudlet-cloud computing setting in which mobile users can offload applications using virtual resources available on the cloud infrastructure (Kocabas & Soyata, 2014) (Soyata, Ba, Heinzelman, Kwon, & Shi, 2013) (Project, 2012). The cloud servers exhibit diverse latencies and processing times for different instance types and geographical locations, resulting in a wide spectrum of system performance. Figure 6 shows the response times for the Greedy algorithm (Kwon, et al., 2014) run over a narrow-range real cloud environment, when the cloudlet emulator and the cloud servers are located within the same city. To demonstrate the utility of the cloudlet, the results were reported with and without the cloudlet emulator. As evidenced from both plots, the TCP overhead is a significant burden on the mobile. By performing pre-processing on the data received from the mobile and intelligently hiding the TCP overhead partially, the cloudlet is able to provide a 6x speed-up over a naive implementation on the mobile.

The homomorphic encryption application (Kocabas O. , et al., 2013) is tested in an Amazon Cloud environment (AWS, 2012) to determine the cost of running this application in a regular outsourced cloud computing scenario. It has been determined that, this application is intensive in **incoming traffic**, thereby creating an advantage in terms of bandwidth billing. In other words, a significant amount of data is pushed into the cloud, whereas the results are substantially smaller in size, almost by 2-3 orders of magnitude. Operators such as AWS do not charge for the incoming bandwidth, whereas the outgoing traffic is billable. In addition to bandwidth billing, the cost of computation (i.e., GHz/month billing) as well as the cost of storage have also been analyzed in this study.

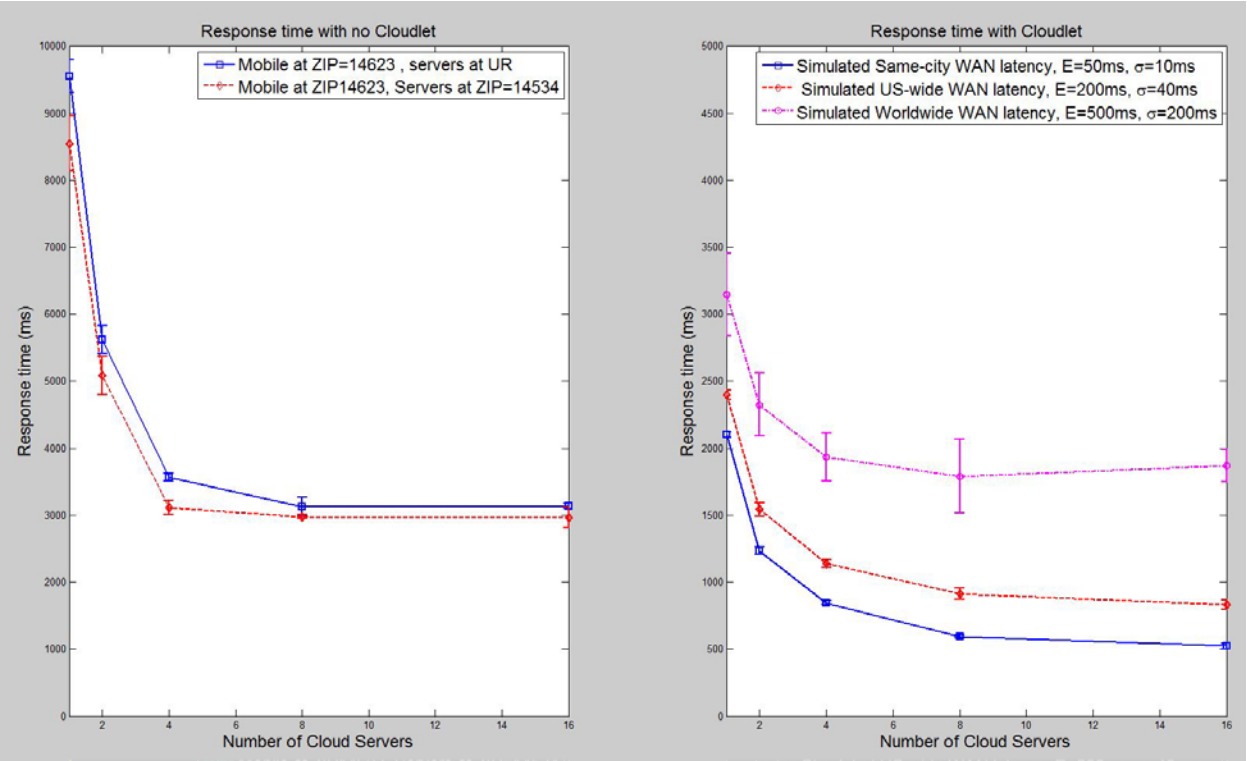


Fig. 6. *Measurement of the response times over actual and simulated links without the cloudlet (left) and with the cloudlet (right).*

REFERENCES

AWS. (2012). *http://aws.amazon.com*. Retrieved from Amazon Web Services.

Dabek, F., Cox, R., Kaashoek, F., & Morris, R. (2004). Vivaldi: A Decentralized Network Coordinate System. *ACM SIGCOMM*, (pp. 15-26).

Ding, C., Chen, Y., T., X., & Fu, X. (2010). CloudGPS: A Scalable and ISP-friendly Server Selection Scheme in Cloud Computing Environments. *IEEE IWQoS*, (pp. 1-9).

Fahad, A., Soyata, T., Wang, T., Sharma, G., Heinzelman, W., & Shen, K. (2012). SOLARCAP: Super Capacitor Buffering of Solar Energy for Self-Sustainable Field Systems. *Proceedings of the 25th IEEE International System-On-Chip Conference (SOCC)*, (pp. 236-241). Nigara Falls, NY.

Fahmy, S., & Karwa, T. (2001). *TCP Congestion Control: Overview and Survey of Ongoing Research.* CSD-TR-01-016, Purdue University, Computer Science.

Francis, P., Jamin, S., Jin, C., JIn, Y., Raz, D., Shavitt, Y., & Zhang, L. (2001, Oct.). IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking, 9*, 525-540.

Gummadi, K., Saroiu, S., & S., G. (2002). King: Estimating Latency between Arbitrary Internet End Hosts. *ACM IMW*, (pp. 5-18).

Guo, X., Ipek, E., & Soyata, T. (2010). Resistive Computation: Avoiding the Power Wall with Low-Leakage, {STT-MRAM} Based Computing. *Proceedings of the International Symposium on Computer Architecture (ISCA)*, (pp. 371-382). Saint-Malo, France.

H. Madhyastha, T. A. (2006). A Structural Approach to Latency Prediction. *ACM IMC*, (pp. 99-104).

Hassanalieragh, M., Soyata, T., Nadeau, A., & Sharma, G. (2014). Solar-Supercapacitor Harvesting System Design for Energy-Aware Applications. *Proceedings of the 27th IEEE International System-On-Chip Conference.* Las Vegas, NV.

Kaehler, G., & Bradski, A. (2008). *OpenCV Computer Vision with OpenCV Library.* O'Reilly.

Khosla, R., Fahmy, S., & Hu, Y. (2012). Content Retrieval Using Cloud-based DNS. *IEEE Global Internet Symposium*, (pp. 1-6).

Knuth, D. (1973). *The Art of Computer Programming: Sorting and Searching.* Reading, MA: Addison-Wesley.

Kocabas, O., & Soyata, T. (2014). Medical Data Analytics in the Cloud Using Homomorphic Encryption. In P. Chelliah, & G. Deka, *Handbook of Research on Cloud Infrastructures for Big Data Analytics* (pp. 471-488). IGI Global.

Kocabas, O., Soyata, T., Couderc, J., Aktas, M., Xia, J., & Huang, M. (2013). Assessment of Cloud-based Health Monitoring Using Homomorphic Encryption. (pp. 443-446). IEEE ICCD.

Kwon, M., Dou, Z., Heinzelman, W., Soyata, T., Ba, H., & Shi, J. (2014). Use of Network Latency Profiling and Redundancy for Cloud Server Selection. *IEEE CLOUD.*

Li, P., Ding, C., Hu, X., & Soyata, T. (2014). LDetector: A Low Overhead Race Detector for GPU Programs. *5th Workshop on Determinism and Correctness in Parallel Programming (WODET2014).*

Markopoulou, A., Tobagi, F., & Manour, K. (2006, June). Loss and Delay Measurements of Internet Backbones. *Computer Communications, 29*(10), 1590-1604.

Nadeau, A., Sharma, G., & Soyata, T. (2014). State of Charge Estimation for Supercapacitors: A Kalman Filtering Approach. *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, (pp. 2213-2217). Florence, Italy.

Ng, E., & Zhang, H. (2001). Predicting Internet Network Distance with Coordinate-based Approaches. *IEEE INFOCOM*, (pp. 170-179).

Padhye, J., Firoiu, V., Towsley, D., & Kurose, J. (1998). Modeling TCP Throughput: A Simple Model and its Empirical Validation. *ACM SIGCOMM*, (pp. 303-323).

Page, A., Kocabas, O., Soyata, T., Aktas, M., & Couderc, J. (2014). Cloud-Based Privacy-Preserving Remote ECG Monitoring and Surveillance. *Annals of Noninvasive Electrocardiology (ANEC)*.

Paxson, V. (1996). End-to-End Routing Bahavior in the Internet. *ACM SIGCOMM*, (pp. 25-38).

Paxson, V. (1997). End-to-End Internet Packet Dynamics. *ACM SIGCOMM*, (pp. 139-152).

PlanetLab. (2014). *PlanetLab*. Retrieved from http://www.planet-lab.org

Project, T. M. (2012, February). *AWS Measurements*. Retrieved from http://www.themochaproject.com

Savage, S., Collins, A., Hoffman, E., Snell, J., & Anderson, T. (1999). The End-to-End Effects of Internet Path Selection. *ACM SIGCOMM*, (pp. 289-299).

Soyata, T. (2000). *Incorporating circuit-level information into the retiming process.* University of Rochester.

Soyata, T., & Friedman, E. (1994). Retiming with non-zero clock skew, variable register, and interconnect delay. *Proceedings of the IEEE Conference on Computer-Aided Design (ICCAD)*, (pp. 234-241).

Soyata, T., & Friedman, E. G. (1994). Synchronous Performance and Reliability Improvements in Pipelined ASICs. *Proceedings of the IEEE ASIC Conference (ASIC)*, (pp. 383-390).

Soyata, T., & Liobe, J. (2012). pbCAM: probabilistically-banked Content Addressable Memory. *Proceedings of the 25th IEEE International System-on-Chip Conference (IEEE SOCC)*, (pp. 27-32). Niagara Falls, NY.

Soyata, T., Ba, H., Heinzelman, W., Kwon, M., & Shi, J. (2013). Accelerating Mobile-Cloud Computing: A Survey. In H. Mouftah, & B. Kantarci, *Communication Infrastructures for Cloud Computing* (pp. 175-197). IGI Global.

Soyata, T., Friedman, E. G., & Mulligan, J. H. (1995). Monotonicity constraints on path delays for efficient retiming with localized clock skew and variable register delay. *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, (pp. 1748-1751). Seattle, WA.

Soyata, T., Friedman, E. G., & Mulligan, J. H. (1997, January). Incorporating Interconnect, Register, and Clock Distribution Delays into the Retiming Process. *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, 16*(1), 105-120.

Soyata, T., Friedman, E., & Mulligan, J. (1993). Integration of Clock Skew and Register Delays into a Retiming Algorithm. *Proceedings of the IEEE International Symposium on Circuits and Systems*, (pp. 1483-1486).

Soyata, T., Muraleedharan, R., Ames, S., Langdon, J., Funai, C., Kwon, M., & Heinzelman, W. (2012). COMBAT: mobile-Cloud-based cOmpute/coMmunications infrastructure for BATtlefield applications. *Proceedings of SPIE*, (pp. 84030K-13). Baltimore, MD.

Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., & Heinzelman, W. (2012). Cloud-Vision: Real-Time Face Recognition Using a Mobile-Cloudlet-Cloud Acceleration Architecture. *Proceedings of the 17th IEEE Symposium on Computers and Communications (IEEE ISCC)*, (pp. 59-66). Capadoccia, Turkey.

Wang, H., Liu, W., & Soyata, T. (2014). Accessing Big Data in the Cloud Using Mobile Devices. In P. Chelliah, & G. Deka, *Handbook of Research on Cloud Infrastructures for Big Data Analytics* (pp. 444-470). IGI Global.

Wendell, P., Jiang, J., Freedman, M., & Rexford, J. (2010). DONAR: Decentralized Server Selection for Cloud Services. *ACM SIGCOMM*, (pp. 231-242).

Wikipedia. (2012). *Rayleigh Distribution*. Retrieved from http://en.wikipedia.org/wiki/Rayleigh_distribution

KEY TERMS & DEFINITIONS

Network Latency: Time interval measured from the source sending a packet to the destination receiving it. It can be measured either one-way or round-trip time that is the one-way from the source to destination plus the reverse one-way, and measuring round-trip time is more straightforward due to clock synchronization. Ping is a popular service used often for round-trip time measurement.

Network Throughput: The rate of data transmission over a communication channel in the network. It indicates the amount of data received per unit time and usually is measured in bits per second or bps. Note that data loss lowers network throughput because such packets do not arrive successfully at a receiver.

Round-Trip Time (rtt) : The time it takes for a packet to travel from a sender to a receiver, and then back to the sender. The rtt time consists of propagation and queuing delays at the core networks, and processing delays at end-hosts.

Network Router: A server that connects more than one network and forwards packets between networks. A router is equipped with specialized hardware and software platforms for fast packet forwarding. Routers also communicate with other routers to compute routes, i.e., the next hop for a specified destination.

Network Switch: A computing device that connects more than one local area network (LAN), and forwards packets between LANs. A switch is a simpler version of a router in the LAN environment, and its main tasks are routing and forwarding.

Hop: A part of the route between the source and the destination, specifically a link between two consecutive routers. A hop occurs each time a packet travels from one router to the next one. We can use traceroute or ping to find the number of hops from one host to another.

Multi-Hop Connection: A network connection that consists of more than one hop. This implies that the connection includes more than one router and links between two routers. Multi-hop connection requires routing decisions at each router to reach the destination.

Internet Service Provider (ISP): A commercial or non-commercial organization that enables users to access the Internet services. ISPs are a multi-tired organization, and pay upstream ISPs with a larger network for Internet access. The Internet services include Internet access, e-mails, domain name registration, web hosting, and Internet transit.

Telephone Service Provider (TSP): A service provider that allows access to phone and communication services. The examples are AT&T, Verizon, Frontier, and CenturyLink.

Application Response Time: The time taken for an application to respond to a user from the user's point of view. While network response time is about how fast the network responds, application response time is end-user perceived time that includes network response time. Application response time is the sum of network response time and transaction response time where transaction response time is the time taken for the server and client to process a request.

Mobile Computing: Computing and communication activity in which a computer is mobile and transported during its usage. Mobile devices include mobile computers like laptops, mobile phones like smartphones, and wearable computers. The challenges include power consumption, mobile management, communication ranges, security and privacy, and transmission interferences.

Cloud Computing: A type of computing where computing, storage, and networking resources are delivered to users as a utility over the network. The cloud provides an illusion of infinite computing resources, and eliminates an upfront capital expense since users can pay the cost on a need basis. Cloud computing is also used in a pay-as-you-go manner, i.e., elasticity that removes the risks of overprovisioning and underprovisioning.

Cloudlet: A computationally highly capable intermediate device placed between a mobile device and the cloud servers. Mobile-cloud computing benefits from having a cloudlet as the cloudlet can provide far higher computational power with minimal latencies. The cloudlet can also pre-process data sent by a mobile and pass even smaller data to the cloud.

Mobile-Cloud Computing: A type of computing in which a mobile client offloads some of its computing to servers in the cloud. As mobiles and cloud computing become more prevalent, mobile-cloud computing becomes popular with high potential for applications. One challenge is how to divide and distribute data from the mobile to servers in the cloud seamlessly and transparently.