A Survey of Network Analysis of TCP ECN, Mobile Traffic, and Cloud Infrastructure with Direction for Future Research

Giuseppe Di Natale
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599
Email: gdinatal@cs.unc.edu

Jay Aikat
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599
Email: aikat@cs.unc.edu

Abstract—**Network measurement and analysis is a very active area of research that has grown due to the gradual evolution of the Internet and the ever larger quantity of traffic being carried by the network. Much of this growth has come from an ever increasing number of mobile devices on the Internet along with various cloud deployments, such as Amazon EC2 or Microsoft Azure, gaining traction as a method of deploying services. Other areas of interest involve improving the utilization of the underlying network with research on how various AQM schemes and TCP ECN could impact network performance. In this paper, we begin by providing a survey of the fundamentals of active and passive measurement. We also provide information about publicly available datasets of interest in the networking community and discuss the basic issues of precision and accuracy in regard to network measurement and analysis. Next, we present an overview of network measurement methodologies presented at conferences within approximately the last three years in the areas of TCP ECN support and usage on the Internet, IaaS cloud usage, and mobile traffic analysis. Finally, we propose some direction for future research which is related to the three areas reviewed.**

I. Introduction

Over time, the Internet evolves to continue to serve the ever changing needs of the billions of servers, desktops, laptops, and mobile devices attempting to communicate with each other. Network measurement and analysis aides in the understanding of how devices behave and interact with each other and the network. Over time, it is also possible to understand how the landscape of the Internet is changing.

In this paper, we will survey the two types of measurement techniques called active and passive measurement. We will present some important datasets which the network measurement community actively uses to perform studies. We also cover the basic concept of a network flow and cover some details regarding TCP and IP flows. Along with the basic methodology and terms, we will cover various concerns when performing network measurement and analysis and how to account for these concerns. We then present the methodologies from recently published research in the areas of TCP Explicit Congestion Notification (ECN), mobile traffic analysis, and usage of the cloud. Finally, we present active and passive measurement methodologies to extend and combine research in the areas reviewed.

II. Types of Measurement

In this section, we present an overview of the two methods used to perform network measurements. The two methods are passive and active measurement. Each method can yield different sets of data and both have their strengths and weaknesses.

A. Passive Measurement

Passive measurement is a method of measuring traffic on a network without generating or modifying existing traffic the network carries [1]. This method involves monitoring or collecting traffic carried by network links, delivered to various endpoints, routed by routers in the network, or any combination of the aforementioned. Examples of information that can be collected from passive methods are packet timings, protocol mixes, and packet rates [1].

Tools that can help collect, monitor, or help process traffic include, but are not limited to, Wireshark [2], tcpdump [3], libpcap [3]. Some specialized hardware often used to collect traffic includes fiber taps and DAG cards [4]. Fiber taps allow for a copy of duplex traffic over a physical link to be sent to a machine for capture with minimal degradation of the original signal on the link. Using DAG cards ensures that traffic capture is reliable at high link rates. Alternatively, if the traffic is being collected over from a WIFI signal, wireless adapters, such as Riverbed AirPcap [5], can be purchased and used to collect 802.11 traffic. If traffic is being collected at a router, it may support copying traffic to a mirror port for capture.

Passive measurement gives you a single vantage point within the network. If you deploy a passive monitor with a DAG card, it will only have the view of the network as seen by the tapped link. Similarly, if you perform monitoring on a router, you will only have the view of the network from that router. In order to remedy the problem, monitoring can be deployed at different points in the network, but this practice is infeasible for very large networks.

The biggest benefit of passive measurement is that it has virtually no impact on network performance. There is also the additional benefit of being able to collect network traffic once and running different analyses on the single dataset. Also, since packets are collected and stored, experiment replication becomes possible for other researchers. Passive methods also enable network application debugging because entire packets can be collected and verified [1].

One major concern when passively collecting traffic is user privacy. Traffic may contain personal information, passwords, and other personal information if the packet payload is not encrypted. A method to prevent invading the privacy of the network users is to obfuscate the IP addresses in the IP header. The anonymization of IP addresses removes the primary source of personally identifying information. In order to avoid payloads with user identifiable information, only the IP and TCP headers could be collected from monitoring machines. Of course, there is a trade off, the more you obfuscate or anonymize the data, the higher the chance for loss of potentially insightful information. An example of a case where IP addresses need to be preserved for a reverse DNS lookup to retrieve the domain name. As a note, there may be cases where a full packet trace (without obfuscation or anonymization) may be necessary and institutional review may be necessary.

A drawback of this technique is the high cost of the specialized hardware to collect traffic on high rate links and the costs increase if multiple monitor machines need to be deployed. Another drawback is the need of large volumes of disk space to store large amounts of collected traffic for analysis and reuse. Other drawbacks include inaccuracies due to packet loss, timestamp inaccuracies, and internet routing differences between the send and receive paths [1].

B. *Active Measurement*

Active measurement is a method of measuring some aspect of the network by generating traffic which the network will carry between two endpoints. Some classic examples of active measurement include ping, traceroute, and ICMP probing. Another more unconventional example of active measurement would be DNS querying as seen in [6]. Examples of data collected from active measurement are packet round trip times, average packet loss, and bandwidth [1]. A common tool which performs basic active measurement is called iperf. Iperf can measure UDP and TCP performance and can measure bandwidth, packet loss, and jitter [7].

There are also cases where packets must be manually built and transmitted over the wire for probing. Scapy is a tool which enables a user to manipulate or craft a packet and transmit it to a destination [8]. This allows a user to set desired flags in a packet and can be used to test specific functionality provided by protocols such as TCP's ECN (see [9] for more details).

Active measurement gives you information regarding a particular route through the network as opposed to a single point in the network [1]. There are research testbeds that can be used to deploy measurement tools across the globe allowing for location to be factored into active measurement. Perhaps one of the biggest testbeds is PlanetLab which currently is comprised of over 1300 nodes at over 650 sites [10].

Unlike passive measurement, there is no issues with user privacy. All traffic is generated by the measurement tool and no traffic is collected from the network.

The main concern with active measurement is that traffic is being generated for the network to carry. Depending on the amount of probing, network performance can be impacted which can invalidate any measurements taken [11]. Another concern is that the network, in some cases, may treat the probing traffic differently which can also invalidate experimental measurements [11].

III. Publicly Available Datasets

Data sets are an important part of network measurement research and are released by fellow researchers or organizations. Researchers often times release data sets generated from projects so others can replicate or leverage these data sets in their own research. In this section, we will only discuss the two most common types of data sets which are website rankings and network traffic captures from other locations around the Internet.

A. *Website Rankings*

Typically, researchers are interested in potential trends or behavior that exist for the most popular websites on the Internet. A popular website ranking data set is provided by Alexa Internet, Inc. [12]. The data set provides the top one million websites based on one month of average traffic ranking. Website ranking data sets, such as Alexa, are often leveraged in active measurement studies and are used to determine which services to probe or request information from. A couple of examples of studies using both active measurement and site ranking are [13] and [6].

B. *Network Traffic Captures*

Since passive measurement studies usually are limited to one vantage point on the network, it would be useful to have other network traces to validate if an observed pattern or trend exists from some other vantage point. Various organizations provide network traces from the edge of their networks so researchers can have other vantage points to test their theories against. An older collection of network traces is provided by the WAND Network Research Group from the edge of the University of Auckland [14]. A more recent collection of traces is provided by the Center for Applied Internet Data Analysis (CAIDA) from its passive monitors [15].

IV. Network Flows

A network flow is defined as a collection of IP packets from a specific time period which all have a set of common properties and are collected from the same observation point [16]. These common properties can include various packet header fields such as destination and source address, port number, protocol type, or even application level header fields [16].

Network measurement studies are usually interested in either IP or TCP flows. These two flows differ in the amount of granularity they provide. When doing flow analysis, the difference between these two types of flows comes down to which headers are collected. Below we discuss the differences in more detail. Also, depending on the circumstances, the concept of a *valid* flow may have to be defined. For example, a flow is valid only if it contains more than a specified number of packets or it is valid if the flow was terminated correctly (i.e. does not experience a TCP RESET).

A. IP Flows

IP flows consist of a collection of time ordered IP headers which are collected from a passive monitor. The IP header gives you source and destination IP addresses, various options (including ECN codepoints), protocol, and TTL along with a other miscellaneous information [17]. IP flows can only provide very limited information regarding the nature of the flow and at times it may be even difficult to calculate simple metrics such as average bandwidth. Usually IP flows are collected in one direction only due to limitations including routing differences to and from the remote endpoint or some monitor limitations. IP flows also allow for researchers to still analyze network traffic at some level if user privacy is a major concern for the network.

Tools also exist to help collect statistics of IP flows on networks. Some universities and organization leverage Cisco's NetFlow to collect information such as the type of traffic is on their network, traffic volume, and protocol usages [18].

B. TCP Flows

TCP flows consist of a collection of time ordered TCP header or packets, usually including the IP header, which are collected by a passive monitor. TCP flows provide a much finer granularity of information. Assuming the flows include IP header information, the flows provide other information regarding TCP configuration, source and destination port numbers, and sequence numbers [19]. Also, when collecting TCP flows, usually flows in both directions are collected. Also, TCP flows can be used to generate the same metrics as an equivalent IP flow, thus IP flows provide a subset of the information that TCP flows provide.

TCP flows allow you to see how individual TCP connections are started and which options are negotiated. During a flow, you can get detailed insights on TCP behavior such as window size changes, when congestion is detected if ECN is enabled, and detect losses and retransmissions. Using the TCP destination port number, it is possible to infer what services are used (i.e. HTTP is usually port 80).

V. Concerns When Performing Network Measurement

In this section, we will discuss the basic concerns of network measurement in very brief detail. In order to get more information regarding this matter, please see [20].

A. Precision

Precision is defined as the ”maximum exactness” a tool can achieve [20]. The common example of precision, or resolution, of system clocks. Most modern operating systems have a clock resolution of a millisecond or less [21]. In passive monitoring of high speed links, a much higher resolution clock is required to ensure that correct packet ordering is preserved. In order to gain this higher resolution, GPS receivers or DAG cards, which usually have special timing hardware, can be used [4]. Also, when performing a packet trace, it is usually considered more precise to collect the entire packet as opposed to just the packet's header [20]. Precision should also be carried forward when performing calculations and analysis (referred to as meta-data) [20].

Another source of possible precision issues are IP address and domain name mappings. IP addresses are often assigned (or leased out) to domains for varying amounts of time. Once this time passes, the IP address is then placed back into the pool of unassigned IP addresses to be later assigned out again. If a measurement requires obtaining the domain name assigned to a given IP address, a reverse DNS lookup is requested for the address. Theoretically, it is possible to obtain the incorrect domain name for an IP address if enough time has lapsed between discovering the IP address and performing the DNS reverse lookup. To resolve the issue of obtaining potentially incorrect domain names, it is possible to design tools using APIs, such as CoralReef [22], to perform a DNS lookup immediately after an IP address is obtained from the initial measurement.

B. Accuracy

Accuracy is defined as a measure's ”closeness” to the actual phenomenon being measured [20]. In network measurement, there are two concerns; clock accuracy and packet loss.

Clock accuracy is important for passive monitors or computers generating probes to measure phenomenon. There are a number of protocols that attempt to ensure that a computer's clock is accurate. The Network Time Protocol (NTP) is used to synchronize computer clocks over the Internet and ensures a clock resolution that is under a nanosecond [23]. The Precision Time Protocol (PTP) can synchronize time across an entire network and can be used to synchronize a network of monitor machines [24]. Again, there are also hardware solutions, such as GPS receivers, which will ensure time remains accurate.

When running a passive monitor on a network, packet loss becomes a major concern. The task of a network monitor is to provide an accurate representation of the traffic crossing the link or being processed by a router. The task of collecting traffic becomes a bit more difficult with higher rate links. DAG cards are are commonly deployed because they guarantee little to no loss during traffic capture [4]. For active measurement, packet loss is less critical and is normally handled by implementing a retry mechanism before definitively saying a probe attempt has failed.

VI. Methodologies Presented in Recent Research

In the three following sections, we discuss the various passive and active techniques used in recent research (within the last three years) for three topics in network traffic analysis. The three areas we will explore are TCP ECN deployment and usage, mobile traffic analysis, and usage of the cloud.

VII. Survey of TCP ECN Deployment and Usage

A. Background & Motivation

Conventional Transmission Control Protocol (TCP) shrinks the congestion window when packet loss occurs. By waiting for loss, lower network utilization results because the network has already reached a high level of congestion. Explicit Congestion Notification (ECN) is an attempt to have the network routers notify the endpoints of a TCP connection when congestion is beginning to present itself in the network. Once the endpoints see an ECN notification, they shrink their congestion window before packet loss occurs with the goal of preserving network utilization.

In order to achieve this system of notification, ECN is built as a two part solution. First, both endpoints of a TCP connection must support and both must agree to enable ECN on a per connection basis. Second, some routers along the path between the endpoints must run an Active Queue Management (AQM) scheme to mark packets, or if necessary intelligently drop packets, if the router becomes congested.

In the IP header, ECN uses the two least significant bits of the differentiated services codepoint in the IP header [25]. An IP header with ECN codepoint with value 00 indicates a packet that is not capable of ECN transport (Non-ECT) and routers should not flag the packet. An ECN codepoint with value 11 (CE) is a packet that encountered a router with congestion while traversing the path between endpoints. An ECN codepoint value of 01 (ECT(1)) or 10 (ECT(0)) allows any routers running an AQM scheme on the path between the endpoints to mark that packet with the CE codepoint to indicate to the endpoints that congestion exists on the path.

In the TCP header, there are three bits dedicated to ECN functionality. The three bits are ECN-nonce (NS) [26], ECN-Echo (ECE) flag [25], and Congestion Window Reduced (CWR) flag [25]. When a TCP endpoint receives an IP header with the CE codepoint set, it must inform the sender of the network congestion by setting the ECE flag in the TCP header of the next acknowledgment (ACK) sent. The endpoint will continue to send ACKs with the ECE flag set until it receives a packet with the CWR flag set. The CWR flag is used to signal the other endpoint that the ECE flag was received and that its congestion window was reduced accordingly.

ECN having been introduced over a decade ago, has seen slow adoption across the endpoints of the Internet [13]. Even though ECN is enabled by default on most modern operating systems, ECN negotiation will only happen when the remote host requests a connection with ECN enabled [9]. Also, most modern routers have AQM schemes built in, but AQM schemes are seldom enabled by default which has hindered deployment.

B. Review of On the State of ECN and TCP Options on the Internet

In *On the State of ECN and TCP Options on the Internet* [9], the authors were interested in testing the deployment of various TCP features. The TCP features of interest were TCP's selective acknowledgments (SACK), timestamps (TS), window scaling (WS), and ECN. In this survey, they present two active measurement methods and utilize passive monitoring on the SWITCH network.

Active probing was done by generating a SYN packet with SACK, TS, and WS enabled and attempting to establish a connection with the servers on the Alexa top 100,000 webservers list. All connections were immediately closed with a FIN packet. The SYN/ACK responses were collected with tcpdump [3] and analyzed offline using scapy [8].

The other active probing tool is generated based on scapy [8] to determine if ECN is viable for each target webserver. A SYN is generated with ECN negotiation and sent to the target server. If the target server responds and is ECN capable, a packet is sent with the CE codepoint set and determines if the corresponding ACK has ECE set.

The IP Time to Live (TTL) value was used to estimate which operating system was running on the target server. Table I contains which TTL value corresponds with which operating system.

Operating System	TTL
Linux	<64
Windows	128
Solaris/Google	255

TABLE I
IP TTL values associated to various Operating Systems

The passive portion of the survey leveraged NetFlow [18] version 9 flow data from the edge of the Swiss national research and education network called SWITCH. The network contains over two million IPv4 addresses and contains both servers and clients. The amount of traffic generated by users of the network is on the order of 100TB. Since the TCP ECN negotiation is not available, the authors decided to watch all the IP flows for ECT(0) or ECT(1) codepoints and if encountered, would mark the source of the packet as ECN capable if the underlying flow is using TCP. From here, they collected statistics from the resulting dataset.

C. Review of Enabling Internet-Wide Deployment of Explicit Congestion Notification

In *Enabling Internet-Wide Deployment of Explicit Congestion Notification* [27], the authors were interested in testing remote servers for ECN capability and various paths for ECN interference. The testing was performed at vantage points in London, New York, and Singapore on three different occasions in 2014. The Alexa top million website list was used to determine suitable targets for testing. All domains on the list are resolved to at most one IPv4 and one IPv6 address.

The authors designed a tool called ECN Spider that is implemented in Python 3 and utilizes Linux's sysctl utility to enable or disable ECN. The tool spawns two threads and initiates a TCP connection from each thread with one performing a normal TCP negotiation and the other performing ECN negotiation. If a connection is established from a thread, then an HTTP request (without redirects) is performed. The tool is able to determine if a connection with or without ECN fails, but it cannot determine if ECN is actually negotiated and used. A passive monitor is introduced to collect the traffic produced by ECN Spider which provides all the TCP flags and ECN signaling per flow.

Using Linux's iptables, the authors were able to test ECN functionality with the various target servers. TCP maximum segment size (MSS) was set to 300 bytes to ensure that the HTTP requests were broken down into multiple packets. In order to test ECE responses, all outgoing packets were marked with the CE codepoint. To test CWR responses, all incoming packets were marked with the CE codepoint so the local machine sends ECE to the remote server in the following ACK. Finally, CE and ECT blackhole testing was done by marking the SYN packet with the CE, ECT(0), and ECT(1) codepoints to ensure marked packets were not dropped.

D. Related Research

A common issue with AQM and TCP ECN is middlebox interference. Any router or machine on the path between two TCP endpoints qualifies as a middlebox. There is active research into methods of detecting middlebox interference and methods of determining which middlebox on the path is faulty. An example of a middlebox detection tool is tracebox [28]. Tracebox works in a similar fashion to other ICMP probing tools, such as traceroute, and is able to determine which hop is causing the interference. The tool is able to catch interference since more routers on the Internet are becoming RFC1812 compliant which means some routers will return full ICMP packets to the source of the probe [29].

VIII. Survey of Mobile Traffic Analysis

A. Background & Motivation

The number of mobile devices on the planet now exceeds the human population [30]. Many individuals now access services from a mobile device on a frequent basis whether it be to access social media, email, visit websites, or play a game. These mobile devices exhibit different traffic patterns from their desktop machine counterparts. Also, with the number of mobile devices in use, studying how the large volume of traffic impacts the network is essential as well.

B. Review of Network performance of smart mobile handhelds in a university campus WiFi network

In *Network performance of smart mobile handhelds in a university campus WiFi network* [31], the authors were interested in the performance of mobile devices while they are utilizing a WiFi connection. IPhones, iPods, Android phones, Windows phones, and Blackberry phones are considered to be mobile devices and are the devices of interest for this study. A monitor was made using a standard PC and a DAG card and it was placed at the edge of the University of Connecticut (UCONN) network. The monitor collected up to 900 bytes of each packet which includes application, TCP, and IP level headers. The UCONN campus network has separate IP address pools for Wireless LAN (WLAN) and Ethernet, so packet collection was filtered to only catch WLAN traffic.

A majority of the traffic on the UCONN network is TCP traffic. In order for a TCP flow to be *valid*, it must finish the TCP three-way handshake and the connection must not experience a TCP RESET. The port number in the TCP header is used to determine what kind of traffic is being carried by the flow. The authors decided to focus on HTTP flows (port 80) and have left the performance of the other protocols as future work.

In order to distinguish which HTTP flows were mobile device flows, the authors look at the HTTP header. The user agent field in the HTTP header is used to determine if the HTTP flow is associated with a mobile device. By associating an HTTP flow with a mobile device, the authors can then determine the device's IP address at that point in time. Any flows with the same IP occurring concurrently with the current flow are then also associated with the device.

Once the authors filter all mobile HTTP traffic, they begin to classify the TCP flows based the destination. Using the destination IP address in the TCP header, a reverse DNS lookup is used to determine the domain name of the destination. They compare the top services contacted with those of non-mobile devices from the same trace.

They report varying statistics regarding TCP flows from mobile devices including RTT, loss rates, and initial congestion windows among other things. For more details please read [31].

IX. Survey of Usage of the Cloud

A. Background & Motivation

On today's Internet, it is common for companies to deploy their web services on infrastructure as a service (IaaS) clouds. IaaS clouds offer other services including, but not limited to, load balance, content distribution networks (CDN), and DNS hosting. A couple of the more popular IaaS clouds are Amazon's EC2 [32] and Microsoft's Azure [33].

Recent research has shown interest in how many top websites leverage the IaaS cloud and to what degree do these websites depend on the cloud. Many companies are now beginning to leverage IaaS cloud services, such as EC2 and Azure, for rapid deployment, services offered, and flexibility. Some recent research has also started looking into the performance of services which run on the cloud as well.

B. Review of Next Stop, the Cloud: Understanding Modern Web Service Deployment in EC2 and Azure

In *Next Stop, the Cloud: Understanding Modern Web Service Deployment in EC2 and Azure* [6], the authors were interested in determining who is using IaaS and how services are using the cloud. The authors use the Alexa top one million [12] and leverage the published public IP ranges for EC2 and Azure.

The first dataset is a list of subdomains derived from Alexa's top one million websites that leverage the cloud. The subdomain dataset is generated by a couple of active probing methods. The first method is through the performa a DNS zone transfer query (query type AXFR) for each site from Alexa's one million list [34]. This type of DNS query is normally used by administrators to replicate DNS databases across DNS servers [34]. The second method is subdomain brute-forcing which is nothing more than using a list of possible subdomains and performing basic DNS queries. The tool used in the paper for subdomain brute-forcing is called dnsmap [35]. It is worth noting, that subdomain brute-forcing is not perfect and can miss some subdomains. Once a subdomain list is generated, DNS lookups are used to obtain an IP for each subdomain which is compared to the public IP ranges for Amazon's EC2 and Microsoft's Azure IaaS platforms. If the IP falls within the public IP range of either cloud, then the subdomain is labeled as cloud using.

The second dataset is derived from packet traces collected at the border of the University of Wisconsin - Madison campus network. The packet trace from the edge of the UW-Madison campus contained full packets. The tool Bro [36], a traffic analysis and network monitoring tool, was used to analyze the packet trace.

Determining what type of service running on EC2 is a matter of analyzing the DNS records once a subdomain is established as cloud using. A subdomain has a VM front end in EC2 if the DNS query returns only an IP since it is assumed those setups are directly reachable from the outside. A platform as a service (PaaS) front end is running if the DNS query for the subdomain returns a CNAME including, but not limited to, 'elasticbeanstalk', 'heroku.com', 'herokuapp', 'herokucom', and 'herokussl'. A load balancer as a front end contains a CNAME with 'elb.amazonaws.com'. The Amazon CDN uses an entirely different IP range than EC2.

Again, determining the type of service running on Azure is reflected in the DNS records. Azure has a different architecture from EC2 and thus is handled differently. In Azure, VMs and PaaS services are hidden behind logical "Cloud Service" (CS), therefore there is no way to distinguish between these. If the DNS query for an Azure subdomain returns an IP address or a CNAME containing 'cloudapp.net', then it is a CS front end. If the DNS record of an Azure subdomain contains a CNAME which ends with 'trafficmanager.net', then it is a traffic manager. A subdomain uses the Azure CDN if the DNS record contains CNAMEs containing 'msecnd.net'.

C. Review of WhoWas: A Platform for Measuring Web Deployments on IaaS Clouds

In *WhoWas: A Platform for Measuring Web Deployments on IaaS Clouds* [37], the authors were interested in determining how IaaS cloud usage changes over time. In order to determine how the IaaS landscape was changing, they developed an active measurement tool call WhoWas to periodically collect data on the state of the cloud.

The first part of WhoWas is the scanner. The tool is provided a range of IP addresses to probe. In order to probe EC2 and Azure, the tool is provided the publicly published IP ranges for both clouds. WhoWas also allows for IPs to be blacklisted to prevent probing IPs that should not be. For each IP address in the range (and not blacklisted), WhoWas sends a TCP SYN for ports of commonly used services. The ports scanned are port 80 (HTTP), port 443 (HTTPS default), and then port 22 (SSH). Each probe of a port is timed out if a response does not arrive in two seconds and does not retry failed probes. If any probes to an IP address succeed, the address is labeled as *responsive*, otherwise it is *unresponsive*.

If WhoWas determines that an HTTP or HTTPS server is running at a specific IP, it will perform a GET request by concatenating "http://" or "https://" (depending on which port was successful) and then concatenating "robots.txt" to the end of the URL. Depending on the content of the robots.txt file, a second GET request will be performed. For each request, WhoWas records HTTP response codes, response headers, or error information into a MySQL database. If the GET request yields that downloadable content exists, it will then fetch the content and store it locally.

If a probe results in a content download, WhoWas will then perform feature extraction. The tool will extract various fields from the HTTP response including the "x-powered-by" field, the description from the "description" tag in the HTML, a string of all field names in the HTTP response header, the "generator" tag, the server type from the HTTP response header, "keywords" tag from the HTML, and the Google Analytics ID from the HTML.

The first major limitation of the tool is that it assumes default ports for HTTP, HTTPS, and SSH. There is no guarantee that service administrators will run all services on their default assigned ports. The other major limitation is that some service administrators require that content be access by specific URLS where WhoWas simply uses the IP address in hopes of getting a default page.

X. Passive Monitor - Bloodhound

We are planning to launch a server, named Bloodhound, at the edge of the University of North Carolina - Chapel Hill (UNC) campus network to collect network traffic generated by real users on the campus network. The monitor is currently equipped with an Endace 9.2X2 DAG card which will allow for link rate (approximately 10Gb/s) traffic capture [4] and is currently running Ubuntu 12.04 LTS with DAG software version 5.2.1. Bloodhound has an approximately 1TB drive to store active captures and over 7TB for long term storage of captures. For more details on installed packages and sample scripts, please visit the Bloodhound help page [38] for more information on our passive monitor setup.

Also, according to the UNC IT department, mobile devices are allocated an IP from a different IP range than other devices (such as desktop PCs) on the UNC network. Therefore, mobile

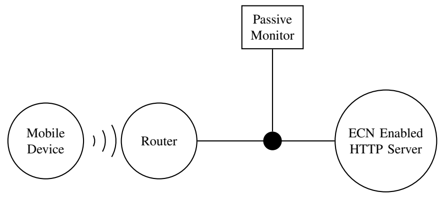

Fig. 1. Network topology of ECN testbed for mobile devices.

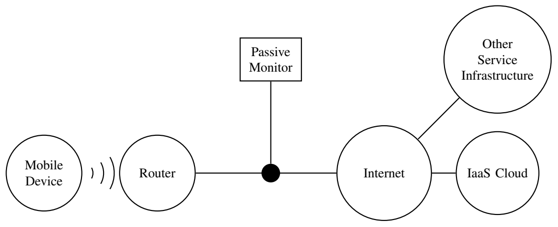

Fig. 2. Network topology of mobile application traffic collection.

traffic generated on the UNC network can be easily filtered from the rest of the traffic after collection based on their unique IP range. This is particularly useful in cases when mobile traffic is to be studied.

XI. Potential Future Research

In this section, we will discuss potential future topics of study along with a high level outline of the potential methodology for each topic.

A. *Comprehensive Survey of Mobile Application Traffic*

As previously mentioned, the number of mobile devices has skyrocketed in recent years [30]. Plus, mobile devices are now generating a significant portion of traffic on the Internet [39]. Much of current research still focuses on other conventional areas while mobile devices continue to grow in number. It is becoming increasingly important to address what type of traffic mobile devices are generating, the utilization of advanced TCP features, and what infrastructure backs services available to mobile users.

For the purpose of this study, a mobile device would be considered a device with a mobile operating system and falls under the category of a phone or tablet. Some examples of these devices would include iPhones, iPads, Windows phones, Android phones, and Android tablets running various recent builds of their respective mobile operating system. We restrict the study to these devices because other mobile devices such as laptops run larger, feature rich operating systems such as Mac OS or Windows and can be viewed as a conventional PC from the point of view of our study.

With modern desktop operating systems supporting ECN, how well do mobile devices support it? ECN capability can be tested by setting up a simple testing infrastructure. As depicted in Figure 1, the testbed would consist of a wireless router (802.11), an ECN enabled HTTP server, and a passive monitor. The wireless router provides a mobile device the ability to connect to our testbed thus allowing it to communicate with the HTTP server. The HTTP server will be ECN enabled and will host a sufficiently large HTML page to ensure that TCP flows between a mobile device and the server are larger than a single packet.

The first task is to determine which mobile devices are ECN capable. To achieve this, each mobile device will simply request the HTML file from the HTTP server with the passive monitor capturing traffic. We then analyze the TCP three way handshake performed by each device and determine if ECN was successfully negotiated for each. In the case of each device, we can explore options which would allow users to enable ECN. In the case of Android devices, if ECN cannot be enabled, it could be possible to enable it on a rooted device. If any device is ECN capable, ECE and CWR testing can be performed. Both tests can be accomplished by leveraging Linux's iptables on the HTTP server. For the ECE test, all non-control packets should have the CE codepoint set in the IP header with the ECN capable mobile device responding with the ECE codepoint set in the TCP header of every ACK. For the CWR test, the server should set the ECE codepoint in the TCP header of each ACK it sends and the mobile should reduce its CW and set the CWR flag in the next TCP packet it sends. It also may be worth testing malformed TCP flows where a mobile device receives ECE or CWR in a TCP packet when ECN was not negotiated during the TCP three way handshake.

Next, we would like to determine what kind of traffic some of the more popular mobile applications generate and whether they communicate with an IaaS cloud. In order to determine which mobile applications to test, we can look at application services such as App Annie [40] which provide market data (ranking based on downloads for paid and free applications and top grossing applications) for Android, Windows phone, and iOS devices. For each mobile operating system, a device can be used to generate traffic from the core functionality (i.e. such as the News Feed from Facebook's mobile application) of the top 2 or 3 applications in each category using a setup similar to the one shown in Figure 2. A mobile device will connect to the Internet via a wireless 802.11 router while a passive monitor will collect all traffic passing from the router to the Internet. A mobile application is IaaS supported if some or all of the TCP flows it produces are destined for an IP address in the published IP range of EC2 or Azure. A mobile application is ECN enabled if any TCP connections are negotiated to have ECN enabled via the TCP three way handshake.

It is worth noting that determining core functionality could be difficult since how an application may be perceived to be used can in fact be quite different from how it is actually used. Instead of guessing what core functionality of an application may be, a user study could be performed where users can use either a provided device or their own personal device and connecting it to they wireless router. They are then given approximately five to ten minutes to use each mobile application of interest. Gaining IRB approval may be difficult since our passive monitor would be collecting entire packets

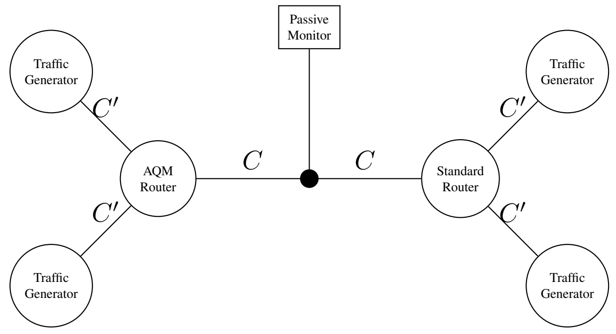

Fig. 3. Network Topology of the NS2 simulator used to model other delays present in a physical experiment.

in which unencrypted data could be present. The risks of breaching a study participants privacy can be mitigated by simply collecting only packet headers.

Finally, on a network such as the UNC campus network where mobile devices are placed in their own IP range, we can generate a data set of just mobile traffic from a trace. For each TCP flow generated by a mobile device, we can tie the flow to a particular service by performing a reverse DNS lookup to determine what domain name the service is running under. We can also determine if the service provider is in the cloud by again comparing the destination IP address of the TCP flow to the public IP ranges of EC2 and Azure. We can also determine if any mobile devices in the wild negotiate ECN and which service ECN was negotiated with allowing us to validate our findings in our testbed setups.

B. TCP RAPID and AQM Schemes

Introduced in 2009, TCP RAPID, a major variation on the TCP protocol, employs a packet scale bandwidth estimation algorithm [41]. Research on improving TCP RAPID has been on going. To our knowledge, TCP RAPID's bandwidth estimation has not been tested with routers employing an AQM scheme. We feel that exploring the various possible configurations of the network nodes in between TCP RAPID endpoints considering there has been significant research into AQM schemes, such as CODEL and PI-E, and a much larger push for deployment of AQM on the Internet.

The primary method of testing these configurations would be to deploy a small network with a dumbbell topology as seen in Figure 3. One router would run a conventional routing protocol such as drop tail and the other router would run an AQM scheme such as CODEL or PI-E. Each end of the network would have the same number of endpoints and each would have a link with capacity C' between them and their respective router. The link in between the routers would have a capacity of C such that $C \ll C'$ in order to maintain a bottleneck.

Realistic traffic generation from each endpoint could be accomplished with a software such as tmix [42]. The amount of utilization on the bottleneck link would have to be varied to determine if bandwidth estimation is impacted. The passive monitor would provide the actual amount of utilization across

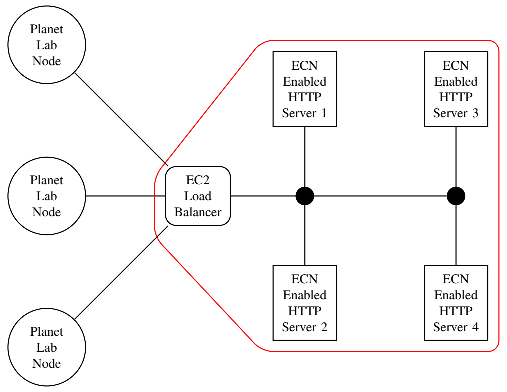
Planet Lab Node
Planet Lab Node
Planet Lab Node
EC2 Load Balancer
ECN Enabled HTTP Server 1
ECN Enabled HTTP Server 2
ECN Enabled HTTP Server 3
ECN Enabled HTTP Server 4

Fig. 4. Diagram depicting the experimental setup for ECN testing incorporating the EC2 Load Balancer. The portion surrounded in red represents the IaaS cloud.

the bottleneck link. For each experiment, the the number of pairs generating TCP RAPID flows can be changed and the estimated bandwidth of each pair can be logged. Any nodes not running TCP RAPID could run a more mainstream TCP implementation such as New Reno.

C. ECN and IaaS Clouds

Recent research has been working towards improving AQM and recent studies have shown the slow deployment of ECN across the Internet [9], [27] with some studies revealing that there is ECN interference by middleboxes [28]. With IaaS clouds playing a very large role as the backbone for some of the most popular web services utilized by users [6], it is important to determine what kind of role these clouds play, if any, in ECN interference. In this section, we present a methodology which can be leveraged to test both Amazon's EC2 and Microsoft's Azure. Unfortunately, there is no way to determine if all routes through an IaaS cloud have been tested (the architecture of the network is not public), but it will help researchers gain insight on how the cloud behaves with ECN negotiations and notifications present.

First, multiple instances running a recent Linux distribution should be deployed on the target IaaS cloud. The instances (henceforth called the servers) will be the target of probing and should have ECN enabled. The servers should host an HTML file with sufficient text to span multiple TCP packets. Each server can also run a packet capture software such as tcpdump [3] in order to record traffic to and from each server. Having multiple servers allows for probing to target multiple physical servers thus providing different routes through the IaaS network. In the case of EC2, servers can be deployed across all the different zones offered by Amazon which allows for testing from one zone to another.

Next, a tool, similar to ECN Spider in [27], should be developed that can perform a TCP three way handshake with both ECN enabled and disabled. If the tool successfully negotiates ECN with a server, it should perform an HTTP request for the HTML file hosted by the server. This tool could be built on top of a tool such as scapy [8] similar to the tool presented in [9].

A client instance running a recent Linux distribution (henceforth called the client) should be deployed within the same IaaS cloud in order to perform ECN testing from within. The client will run the ECN tool with the deployed servers as the targets and the client can also run tcpdump [3] to collect traffic between it and the servers. The client can leverage Linux's iptables to perform ECE and CWR testing in the same fashion as [27]. To test ECE, the client will have to send packets which are all marked with the CE codepoint in the IP header. In order to test CWR, the client must send ACKs with the ECE codepoint set in the TCP header. In order to test ECN across zones on EC2, a client should be deployed in each zone and each client should target all servers in all zones.

To test how EC2's Elastic Load Balancing (ELB) impacts ECN, the server set up would slightly change. As depicted in Figure 4, a set of servers would need to be placed behind an ELB and all clients internally would have to then target the load balancer. They would perform the same tests as the conventional client to server testing. To further test the ELB setup, PlanetLab [10] nodes could be leveraged with the same client setup and again the ELB setup would be used as a target.

Once experiments are completed, the TCP flows within the packet captures can be analyzed to determine if ECN behaved correctly. In the case of ECN negotiation, we would like to determine if the middleboxes within the cloud are modifying the TCP header to force non-ECN connections to be negotiated. If ECN is enabled, for the ECE test, we would like to see the server respond with an ECE if the CE codepoint is set in the IP header. Again if ECN is enabled, for CWR testing, we would like to see the server return a CWR in the TCP header upon receiving an ECE in a TCP ACK.

Additionally to performing active probing in the cloud, a passive monitor, such as Bloodhound (see section X for more details), could be leveraged to collect traffic destined for an IaaS cloud to help determine if external connections to the cloud are leveraging ECN. The passive monitor would be deployed on the edge of a network and would use a filter to collect TCP flows which have an endpoint in either Amazon's EC2 cloud or Microsoft's Azure cloud. The filtering could be accomplished with the published IP ranges for both clouds. A valid TCP flow would have successfully completed a three way handshake with an endpoint in one of the IaaS clouds and A TCP flow would be considered ECN enabled if ECN negotiation was successful during the three way handshake at connection setup. If a TCP packet has the ECT(0) or ECT(1) codepoint in the IP header set, then the associated TCP flow could be considered ECN capable since ECN is not required to be used if enabled. Finally, we would label a TCP flow as ECN utilized if an ECE codepoint is seen in the TCP header of a packet in the flow. The output of analysis would be statistics regarding the count of ECN enabled, capable, and utilized TCP flows within a collection period.

XII. Conclusion

In this paper, we have covered the basic concepts of active and passive measurement and reviewed key datasets important in network measurement. We have review the basic concepts of network flows and discussed the important details regarding TCP and IP flows. We presented and discussed the concerns of accuracy and precision and discussed how to alleviate these concerns. We presented motivation, background, and methodologies for recent research in the areas of TCP ECN deployment and usage, mobile traffic analysis, and cloud usage. We presented methodologies which can provide some insight on mobile TCP ECN capability, if mobile applications leverage services hosted by IaaS clouds, and what mobile traffic looks like. We also presented a methodology which can provide insight into how well IaaS clouds support (or interfere) with TCP ECN. Finally, we presented a methodology to analyze newer TCP protocol designs, such as TCP RAPID, on a network which contains routers running an AQM scheme.

References

[1] J. Curtis, "Passive measurement," Jan 2000, http://wand.net.nz/pubs/19/html/node9.html.

[2] "Wireshark," 1998, https://www.wireshark.org/.

[3] "tcpdump & libpcap," 2010-2015, http://www.tcpdump.org/.

[4] "Emulex corporation," 2013, http://www.emulex.com/.

[5] "Riverbed," http://www.riverbed.com/.

[6] K. He, A. Fisher, L. Wang, A. Gember, A. Akella, and T. Ristenpart, "Next stop, the cloud: Understanding modern web service deployment in ec2 and azure," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 177–190.

[7] "iperf," https://iperf.fr/.

[8] "Scapy," http://www.secdev.org/projects/scapy/.

[9] M. Kühlewind, S. Neuner, and B. Trammell, "On the state of ecn and tcp options on the internet," in *Passive and active measurement*. Springer, 2013, pp. 135–144.

[10] "Planetlab," https://www.planet-lab.org/.

[11] A. Cecil, "A summary of network traffic monitoring and analysis techniques," 2006, http://www.cse.wustl.edu/~jain/cse567-06/ftp/net_monitoring/index.html.

[12] "Alexa," http://www.alexa.com/.

[13] S. Bauer, R. Beverly, and A. Berger, "Measuring the state of ecn readiness in servers, clients, and routers," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 171–180.

[14] "Wits: Waikato internet traffic storage," http://wand.net.nz/wits/catalogue.php.

[15] "Caida: Center for applied internet data analysis," http://www.caida.org/.

[16] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917 (Informational), Internet Engineering Task Force, Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3917.txt

[17] J. Postel, "Internet Protocol," RFC 791 (INTERNET STANDARD), Internet Engineering Task Force, Sep. 1981, updated by RFCs 1349, 2474, 6864. [Online]. Available: http://www.ietf.org/rfc/rfc791.txt

[18] Wikipedia, "Netflow — Wikipedia, the free encyclopedia," 2015, http://en.wikipedia.org/wiki/NetFlow.

[19] J. Postel, "Transmission Control Protocol," RFC 793 (INTERNET STANDARD), Internet Engineering Task Force, Sep. 1981, updated by RFCs 1122, 3168, 6093, 6528. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[20] V. Paxson, "Strategies for sound internet measurement," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 263–271.

[21] Wikipedia, "System time — Wikipedia, the free encyclopedia," 2015, http://en.wikipedia.org/wiki/System_time.

[22] "Coralreef software components," http://www.caida.org/tools/measurement/coralreef/components.xml.

[23] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905 (Proposed Standard), Internet Engineering Task Force, Jun. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5905.txt

[24] Wikipedia, "Precision time protocol — Wikipedia, the free encyclopedia," 2015, http://en.wikipedia.org/wiki/Precision_Time_Protocol.

[25] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168 (Proposed Standard), Internet Engineering Task Force, Sep. 2001, updated by RFCs 4301, 6040. [Online]. Available: http://www.ietf.org/rfc/rfc3168.txt

[26] N. Spring, D. Wetherall, and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces," RFC 3540 (Experimental), Internet Engineering Task Force, Jun. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3540.txt

[27] B. Trammell, M. Kühlewind, D. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger, "Enabling internet-wide deployment of explicit congestion notification," in *Proceedings of the 2015 Passive and Active Measurement Conference*, New York, 03/2015 2015.

[28] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 1–8.

[29] F. Baker, "Requirements for IP Version 4 Routers," RFC 1812 (Proposed Standard), Internet Engineering Task Force, Jun. 1995, updated by RFCs 2644, 6633. [Online]. Available: http://www.ietf.org/rfc/rfc1812.txt

[30] Z. D. Boren, "There are officially more mobile devices than people in the world," October 2014. [Online]. Available: http://www.independent.co.uk/life-style/gadgets-and-tech/news/there-are-officially-more-mobile-devices-than-people-in-the-world-9780518.html

[31] X. Chen, R. Jin, K. Suh, B. Wang, and W. Wei, "Network performance of smart mobile handhelds in a university campus wifi network," in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 315–328.

[32] "Amazon ec2 product details," http://aws.amazon.com/ec2/details/.

[33] "What is microsoft azure?" http://azure.microsoft.com/en-us/overview/what-is-azure/.

[34] Wikipedia, "Dns zone transfer — Wikipedia, the free encyclopedia," 2014, http://en.wikipedia.org/wiki/DNS_zone_transfer.

[35] "dnsmap," https://code.google.com/p/dnsmap/.

[36] "The bro network security monitor," https://www.bro.org/.

[37] L. Wang, A. Nappa, J. Caballero, T. Ristenpart, and A. Akella, "Whowas: A platform for measuring web deployments on iaas clouds," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 101–114.

[38] "Bloodhound passive network monitor," http://networking.web.unc.edu/?page_id=74.

[39] A. Dusto, "Mobile devices account for nearly a third of web traffic," February 2014. [Online]. Available: https://www.internetretailer.com/2014/02/06/mobile-devices-account-nearly-third-web-traffic

[40] "App annie," https://www.appannie.com/.

[41] V. Konda and J. Kaur, "Rapid: Shrinking the congestion-control timescale," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 1–9.

[42] M. C. Weigle, P. Adurthi, F. Hernández-Campos, K. Jeffay, and F. D. Smith, "Tmix: a tool for generating realistic tcp application workloads in ns-2," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 3, pp. 65–76, 2006.