The Design, Implementation and Performance Evaluation of Xylophone: A Scalable QoS-enabled WiFi Locator Service

Xinyu Xing[1], Jianxun Dang[2], Shivakant Mishra[1], Xue Liu[2]

[1]University of Colorado at Boulder, [2]McGill University

Contact: xingx@colorado.edu

Department of Computer Science

University of Colorado at Boulder

Technical Report CU-CS 1072-10

July 2010

Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 00 (2010) 1–22
Procedia Computer Science

The Design, Implementation and Performance Evaluation of *Xylophone*: A Scalable QoS-enabled Wifi Locator Service

Xinyu Xing[a], Jianxun Dang[b], Shivakant Mishra[a], Xue Liu[b],

[a]*Department of Computer Science, University of Colorado at Boulder, United States*
[b]*School of Computer Science, McGill University, Canada*

Abstract

WiFi access points that provide Internet access to users have been steadily increasing in urban areas. Different access points differ from one another in terms of services that they provide, including available upstream and downstream bandwidths, overall network capacity, open/blocked ports, security features, and so on. However, there is no reliable service available at present that can aid a user in selecting an access point from the many that are available. This paper presents *xylophone*, a WiFi locator service that enables users to select commercial hotspots in accordance with the current quality of service of hotspot access points. The primary research challenge in *xylophone* is how to accurately estimate current quality of service of various access points in an efficient manner without requiring any installation of special software on the access points, and not burdening the WiFi subscribers to perform any communication or computation intensive task. *Xylophone* has been extensively evaluated via a prototype implementation in an indoor testbed and in the Amazon's EC2 platform. The evaluation demonstrates low latencies experienced by WiFi subscribers to measure DHCP connectivity, authentication and association process, and discover open/blocked ports. Also, the bandwidth measurement component of *xylophone* exhibits high measurement accuracy, low latency, high scalability, and minimal intrusiveness.

Keywords: wireless access point selection, bandwidth measurement, scalability, cloud

1. Introduction

Commercial hotspots that offer Internet access over a wireless local area network through the use of a router connected to an ISP have been increasingly becoming popular in most public areas. These hotspot WiFi services are fully-featured and time-tested. However, the New York Times and The Wall Street Journal reported in 2005 that these commercial hotspots may not be able to satisfy clients' connectivity requirements. Consider the following scenario. When a business traveler checks into a hotel, connects to a WiFi network in the hotel and attempts to

upload financial statements to a centrialized file server, he realizes that his WiFi device is experiencing poor connectivity due to limited Access Point (AP) backhaul bandwidth[1] and firewall restrictions. So, the traveler decides to use a WiFi locator service to search for alternative WiFi networks nearby. Generally, he can find several alternatives with ease as commercial hotspots are available in abundance in most locations in urban areas. However, apart from location and navigation information, current locator services do not provide any other information. As a result, the traveler has no way to find out in advance, i.e. before he pays for a connection, which AP provides the best quality of service as per his needs.

To address this problem, [3] introduced a collaborative service called WiFi-reports that provides WiFi clients with historical information about hotspot AP performance before the clients decide to pay for an access. However, [3] depends on voluntary collaboration among users, requiring users to report hotspot AP performance. It requires that volunteer users measure hotspot AP performance (connectivity capacity, bandwidth etc.) by running a third-party measurement software on their mobile devices while accessing the Internet for their computing needs. The users then need to report their measurement results to a central server that maintains a history of performance of various APs. Utility of such a historical based mechanism is based on the assumption that either the volunteer users report hotspot AP performance quite regularly or the hotspot AP performance remains approximately constant. Unfortunately, in the absence of any incentives, it is unlikely that users will voluntarily report hotspot AP performance quite regularly. Furthremore, hotspot AP performance, particularly bandwidth can change quite frequently in general. Finally, bandwidth measurement is a communication and computation intensive task, and so, only a few users will be willing to run such software. When only a few clients participate, such collaborative services fail to provide an accurate and up-to-date perfomance characteristics of various APs.

This paper introduces *xylophone* – a system prototype for commercial hotspot services – that operates as an extension to hotspot WiFi locator service. Like a hotspot WiFi locator service[4][5][6], *xylophone* allows a hotspot subscriber to search for available hotspots in a given area. In addition, *xylophone* enables hotspot subscriber to select commercial hotspots based on their current quality of service. *Xylophone* addresses the limitations of WiFi-reports by moving the computation and communication intensive tasks such as bandwidth measurement to a separate cloud platform. This is done by introducing a novel method for estimating the current bandwidth of APs. Specifically, *xylophone* has several unique features. First, it provides the current quality of service of hotspot APs in terms of current bandwidth as well as connectivity capacity, open/blocked ports, etc. Second, unlike [3], *xylophone* does not require clients to perform any compute or communication intensive tasks to measure bandwidth. Third, *xylophone* introduces a novel methodology for estimating hotspot AP backhaul bandwidth that does not require voluntary user participation or hotspot APs to run any special software. Fourth, *xylophone* is highly scalable in terms of simultaneously providing up-to-date performance information of a significantly large number of APs by utilizing cloud services. Finally, *xylophone* utilizes efficient techniques to estimate current performance, resulting in performance measurements completed within a few seconds.

While the goal of *xylophone* is to enhance selectivity for commercial hotspot APs, it can also be operated as a part of hotspot management system to monitor hotspot AP performance and diagnose network faults. The contributions of the paper are listed as follows.

[1] Unless otherwise stated, bandwidth refers to the available bandwidth as defined in [1][2].

(1) A novel bandwidth estimation methodology is presented. This methodology not only estimates the upstream and downstream bandwidths accurately, but also it is efficient (low latency), scalable and less-intrusive.

(2) A prototype of *xylophone* has been implemented on Amazon's EC2 platform and its client component on a mobile device.

(3) An extensive evaluation of *xylophone* in terms of measurement accuracy, latency, scalability and intrusiveness has been done.

The rest of the paper is organized as follows. In Section 2, we summarize related work in the areas of wireless AP selection and bandwidth measurement. In Section 3, we present an overview of *xylophone* and in Section 4, we describe *xylophone* in detail. In Section 5, we present a detailed evaluation of *xylophone*. Finally, in Section 6, we conclude the paper.

2. Related Work

There are two areas of research that *xylophone* builds on: improving wireless network selection and bandwidth estimation.

Improving Wireless Network Selection. Several researchers have explored wireless network selection in recent years. From Judd and Steenkiste's load-sensitive AP selection algorithm[7] to WiFi-Reports[3], the objective has been to enable WiFi clients to choose hotspot APs that provide better performance. Virgil[8] exploits applications mounted on reference servers and clients to collaboratively measure end-to-end bandwidth between a client and a reference server. It then uses this end-to-end bandwidth to choose the best hotspot AP that can be detected on a scan. However, Virgil does not allow users to evaluate hotspot APs that are not within their signal range. [9] relies on a passive bandwidth measurement to select optimal hotspot APs. However, this passive measurement requires modifications in the wireless driver to estimate upstream bandwidth. Furthermore, it also does not help a client to evaluate hotspot APs that are not within their signal range.

WiFi-Reports[3] harnesses user-submitted historical information about hotspot AP performance to help users determine which hotspot APs will be sufficient to run their applications. It enables users to search for the best performance hotspot AP in a certain geographical region from a centralized database. WiFi-Reports improves upon earlier approaches in that it allows users to evaluate APs that may not be within the signal range of the users. However, WiFi-Reports suffers from several limitations. First, WiFi-Reports relies on voluntary participation of users. The key issue is that there is no direct incentive for users to participate. Why will users run computation and communication-intensive tasks to measure the bandwidth on their devices? Indeed, it has been shown that selfish (rational) users tend not to share their resources or time unless there is an external incentive. Second, this lack of incentives will likely result in very few users participating voluntarily. As a result, the central database that contains performance information of various APs will very likely contain outdated information. This is problematic because performance factors such as bandwidth can change quite frequently. Finally, because WiFi-Reports relies on user participation, it needs to deal with the difficult issues of privacy of the users, and security and trustworthiness of the information submitted by the users.

Xylophone is designed to address the limitations of these earlier approaches. Like some of the earlier approaches, *xylophone* does not require participation of APs (in terms of running any special software) to estimate their current performance. Also, like WiFi-Reports, *xylophone* adopts the strategy of a central database that users can query to obtain the performance of an

AP that may not be within their signal range. However, unlike WiFi-Reports, *xylophone* does not require a WiFi client to perform computation and communication-intensive tasks. Instead, it runs computation and communication-intensive tasks on a cloud platform. Also, unlike [9], *xylophone* does not require any modification to the wireless driver.

Bandwidth Estimation. There have been several methodologies and tools developed over the last ten years for estimating bandwidth along a path. Most prior work requires installing special software at both ends of the path to be measured [10][11][1] [2][12][13]. This requirement significantly limits their applicability for our system, because it is unrealistic to install measurement software on each hotspot AP. Our design options are confined to a non-cooperative environment where measurement software is mounted solely on a cloud platform. To the best of our knowledge, only a few measurement methodologies[14][15] can be utilized in a non-cooperative environment.

Abget[14] exploits the properties of TCP, forces a remote host to generate a traffic flow at a certain rate and thus approximately estimates a range of end-to-end bandwidth. However, our previous study[15] indicates that Abget usually suffers from irreparable weakness especially when the storage locations of large files alter. In addition, the applicability of Abget is greatly dependent upon large accessible files and it is impractical to crawl a large accessible file from each hotspot AP. Therefore, Abget is not suitable for *xylophone*. Abode[15] exploits ICMP messages to probe hotspot AP backhaul bandwidth. However, it suffers from two fundamental limitations. First, several empirical-based studies have shown that most ISPs block or constrain ICMP messages. This can cause a serious bandwidth underestimate. Second, Abode is designed for only upstream bandwidth estimation.

Bandwidth measurement methodology of *xylophone* addresses these limitations. It does not require any software installation on APs, does not involve crawling large files from APs, and does not rely on ICMP messages. Furthermore, it can accurately estimate the bandwidth in both upstream and downstream directions in relatively smaller time periods.

3. *Xylophone* Overview

3.1. Challenges

To obtain performance metrics of each commercial hotspot AP, there are several unique challenges confronting *xylophone*.

(1) Computation and communication-intensive measurement tasks: One of the most significant tasks for *xylophone* is to estimate hotspot AP backhaul bandwidth. Though a WiFi user is able to perfrom bandwidth measurement by running a bandwidth estimation software [14][1] on his mobile device, continuous bandwidth measurement severely depletes user's computation and communication resources. This may give rise to a situation where a hotspot subscriber may not be willing to sacrifice his limited resources to perform computation and communication-intensive bandwidth measurement. Furthermore, bandwidth measurement may fail if two hotspot subscribers simultaneously estimate bandwidth via a same hotspot AP[16]. Finally, bandwidth measurement launched by hotspot subscribers may involve potential security threats, because the inbound traffic to clients' mobile devices may compromise clients' security.

(2) Measurement intrusiveness: While some measurement tasks such as SNR measurement and security measurement can be performed passively, others such as bandwidth estimation have to be performed actively. Bandwidth estimation usually involves some probe messages. When measurement latency increases, this may significantly impact hotspot subscribers' connectivity

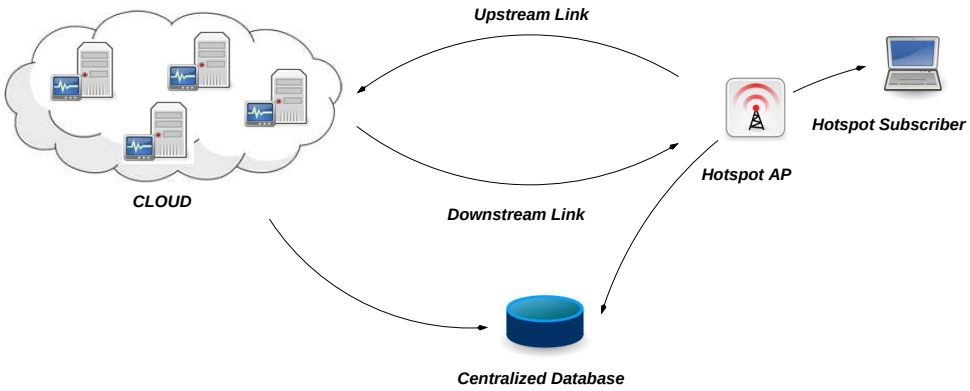

Figure 1: The *xylophone* architecture.

performance. Therefore, bandwidth estimation has to be efficient with low measurement latency and sampling rate of bandwidth measurement cannot be high.

(3) Interference from middleboxes: Middleboxes, such as load balancers, intrusion detection systems and web proxies have an impact on the latency and flow of probe traffic sent as part of bandwidth measurement and the corresponding response messages. As a result, they introduce inaccuracy in the bandwidth being measured[17]. Thus, the bandwidth measurement must account for these middleboxes, and compensate for the inaccuracies introduced.

(4) Scalability: According to a recent survey from Jiwire[6], there may be hundreds of hotspots in a certain region, e.g. there are 877 of hotspots in New York. Since *xylophone* assigns bandwidth measurement task to a centrialized infrastructure, it becomes challenging to estimate bandwidth of these large number of hotspot APs simultaneously. Simply using one single server to perform bandwidth measurement can be problematic due to limited outbound bandwidth and computation resources.

3.2. The xylophone architecture

There are three logical entities in the *xylophone* architecture (see Figure 1): (1) a centralized database; (2) wireless hotspot subscribers; and (3) one or more bandwidth measurement servers in the cloud. The centralized database stores a history of performance characteristics of all APs. It exports two sets of interfaces: one interface that bandwidth measurement servers and hotspot subscribers can use to update its content with new performance values of various APs; and the other interface that clients may use to query the current and past performance of various APs. The centralalized database of *xylophone* is built using standard database techniques, and we will not discuss it anymore in the paper.

Performance characteristics of APs provided by *xylophone* are classified into two categories: collaborative and non-collaborative. Performance characteristics in the collabortive category are those that require low communication and computation for measurement, and whose values remain relatively stable, e.g. set of blocked TCP ports, DHCP acquisition latency, authentication and association latency, and wireless factors[2]. A subscriber generates a measurement report

[2]A number of tools[18] exist to estimate these wireless factors. These tools are outside the scope of this paper.

containing the values of these performance characteristics, and automatically submits it to the centralized database. This design ensures that hotspot subscribers do not have to run computation or communication intensive software on their mobile devices. Furthermore, since they report only those performance characteristics whose values are relatively stable, *xylophone* does not require frequent reports from these subscribers, or even a participation from a large number of subscribers.

We have developed a client module to run on mobile devices. This module initiates a measurement when a mobile device launches a connection to a hotspot WiFi network. Performance information about wireless factors, DHCP acquisition latency, authentication and association latency is collected passivley during the process of connection establishment. No intrusive traffic is injected into hotspot WiFi network to collect these information. In addition, information about connectivity capacity – Block/Open ports – is collected by sending probes to several application ports on the cloud under our control. Our current implementation sends probes to 127 common application ports in parallel. These include FTP, SSH, IMAP, BitTorrent, World of Warcraft etc. The basic operation of port scan is as follows. A SYN packet is sent to our cloud. Since all the common application ports are open in the cloud, the cloud should respond with a SYN-ACK packet. However, the client module will receive a RST packet, if the corresponding hotspot AP filters packets sent to a certain port. Since port scan requires connecting to the cloud, this can be done only after the mobile device has successfully connected to the Internet.

Performance characteristics in the non-collaborative category are those that require high communication and computation for measurement, and whose values change frequently. These include upstream and downstream bandwidths. Bandwidth measurement servers reside on a cloud and run *xylophone*'s bandwidth measurement software. They update the contents of the centralized database with new bandwidth measurements of various APs. We discuss the design and implementation of these bandwidth servers in detail in Section 4.

3.3. Trust assumption

Xylophone assumes that hotspot subscribers do not submit fraudulent measurement results. We expect this trust to result from an existing falsification detection scheme [3]. By using this scheme, *xylophone* can identify fraudulent submissions as well as corresponding counterfeiters, and thus blacklist the counterfeiters. In addition, we further assume that hotspot subscribers cannot masquerade other subscribers to submit fraudulent measurements, as commercial hotspot providers usually assign their subscribers a unique user name and password to control their access.

3.4. Network model

Most commercial hotspots use ADSL technique to connect their local wireless networks to the Internet[3]. As a result, IP addresses assigned to hotspot APs can change frequently. ISPs usually use *SessionTimeout* to limit the maximum lifetime of an IP address and typically sets it to 24 hours[19]. In a commercial hotspot scenario, commercial hotspots usually belong to a certain ISP. So, even though a hotspot AP has dynamic IP address, *xylophone* that is deployed by an ISP can still track its current address.

[20] illustrates that the bandwidth bottleneck of the Internet is usually on the edge of the Interenet (i.e. 2-hop away) and relatively persistent. This observation was further verified in

[3] AT&T hotspots where we conducted our experiments use ADSL technique to connect to the Internet.

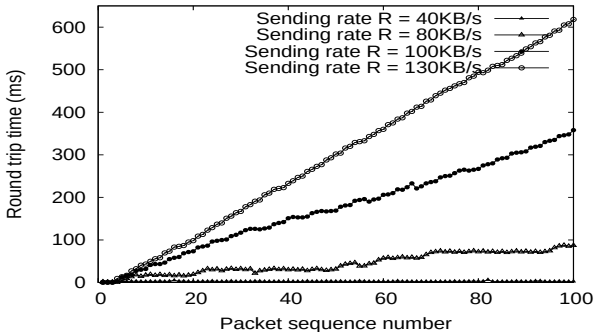

Figure 2: RTT variation with varying TCP ACK data rate.

the context of wireless networks in [15]. With this observation, *xylophone* estimates hotspot AP backhaul bandwidth instead of wireless throughput. Another reason that we estimate hotspot AP backhaul bandwidth is that the ADSL-support maximum throughput is far less than wireless throughput, even after taking the potential wireless signal interference into consideration. Additionally, the distinguishing characteristic of ADSL technology is that bandwidth is greater in the downstream direction than the reverse[21].

Behind hotspot APs, ISP usually deploys a firewall to block unauthorized access. We assume that this firewall does not block our probe messages. In all of our experiments, this assumption has been true. In case *xylophone* is deployed by an ISP, the ISP may set firewall rules to ensure that these messages are not blocked.

4. Bandwidth measurement servers in the cloud

The design, implementation and evaluation of the bandwidth measurement servers are major contributions of this paper. Recall that one major goal is to build a bandwidth measurement module that is highly accurate and scalable, and has low latency and minimal intrusiveness. We first describe the methodology used to measure the upstream and the downstream bandwidths and then describe a prototype implementation in the Amazon's EC2 platform.

4.1. Bandwidth measurement methodology

Bandwidth measurement in *xylophone* is based on TCP protocol behavior. In general, when a host sends a TCP acknowledgement (TCP ACK) packet to another host and if a TCP connection has not been established between them, a TCP reset (TCP RST) message with the fixed length of 40 bytes is sent in response regardless of the size of the TCP ACK packet. Since TCP ACK and RST packets traverse the network in two different directions (i.e. TCP ACK packets are transmitted from the bandwidth measurement server to hotspot APs - downstream link; TCP RST packets are transmitted in the reverse direction - upstream link), we can saturate either downstream or upstream link by adjusting the size of the TCP ACK packet or the interval between back-to-back ACK packets. Bandwidth measurement methodology consists of increasing the sending data rate either by increasing the ACK packet size or the interval between back-to-back ACK packets, and observing the round-trip time (RTT). Here, we define RTT of a TCP ACK packet as the time elapsed between the time when the ACK packet is sent and the time when

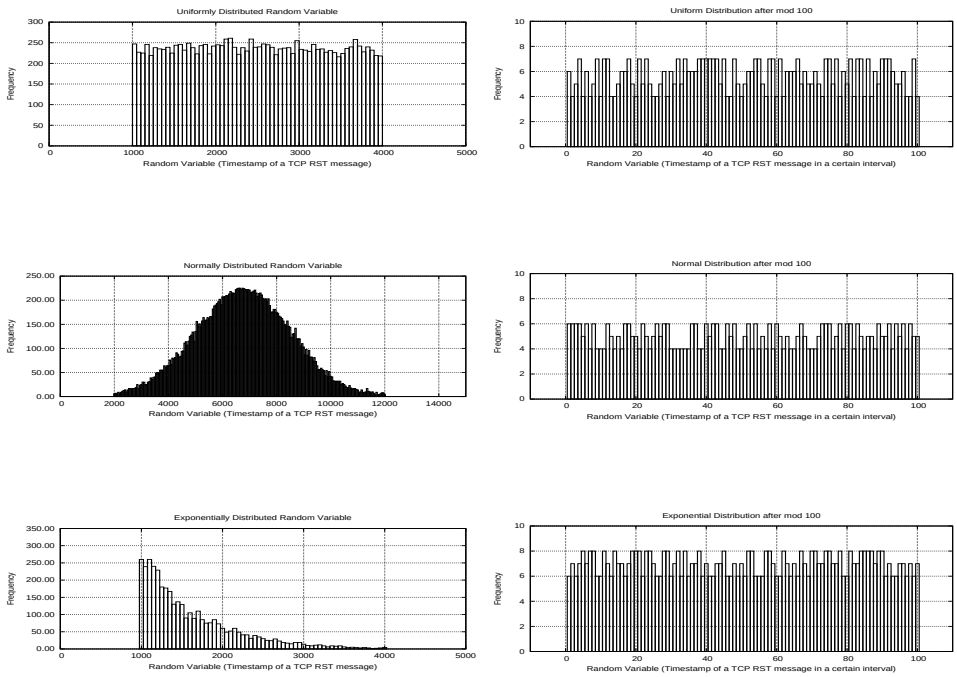

Figure 3: Remainder Distribution of modulo operation over uniformly, normally, exponentially distributed random variable.

the corresponding RST packet is received. The key idea is that when none of the links are saturated, RTT does not change with increase in the data rate. On the other hand, when one of the links is saturated, RTTs between successive TCP ACK packets will have an overall increasing trend. This is because queuing delays start increasing when one of the link has been saturated. The bandwidth of the saturated link is then estimated as the sending data rate when RTT starts increasing from being same.

As an example, Figure 2 illustrates the variation in RTT for a sequence of ACK packets for different data rates. In this experiment, a bandwidth measurement server in the cloud sends 100 TCP ACK packets at different data rate on a 75 KB/s link. As shown in the figure, when sending rate is lower than 75 KB/s, i.e. for data rate 40 KB/s, the RTT of each TCP ACK packet is same for ACK packets. On the other hand, RTTs increase for each successive ACK packet when data rate is above 75 KB/s, i.e. for data rates 80, 100 and 130 KB/s. Furthermore, this increase in RTT is higher for higher sending data rates.

As mentioned earlier, upstream link bandwidth is lower than downstream link bandwidth. So, to measure the upstream link bandwidth, we saturate that link by increasing the rate at which 40B ACK packets are sent. As this rate is increased, the rate at which RST is sent also increases. Since upstream link bandwidth is lower than the downstream bandwidth, it saturates before the downstream link. On the other hand, to saturate the downstream link bandwidth, we saturate that link by increasing the size of the ACK packet sent at some predetermined rate. This ensures that

the data rate on the upstream link doesn't change, but the data rate on the downstream link keeps increasing.

In practice, a traffic flow may not follow the following two assumptions that most bandwidth measurement tools make: (1) FIFO queuing is adopted at all routers along a path, and (2) a traffic flow follows a single routing path between the two end hosts. If the communication path between two hosts varies significantly in a short space of time, e.g. flow splitting and merging incur message out-of-order, we may see unexpected fluctuations in RTT. To address this, we adopt the concept of *time centroid*.

Assume a TCP RST stream, generated at a certain rate, contains $n + 1$ TCP RST messages: $m_0, m_1, m_2, ..., m_n$. Let $t_i (i = 0, ..., n)$ denote the absolute time stamp of TCP RST message m_i. Hence, $\Delta t_i = t_i - t_0$ is the relative time stamp of m_i from the time instance of the first TCP RST message m_0. In addition, Δt_i also represents the offset of TCP RST message m_i from the starting point of the TCP RST stream.

Given any particular sequence of relative time stamps of TCP RST messages $\Delta t_0, \Delta t_1, \Delta t_2, ..., \Delta t_n$ and a random interval length T ($T > 0$ and $T \ll \Delta t_n - \Delta t_0$), $\Delta' t_i = \Delta t_i \bmod T$ represents TCP RST message m_i's offset from the starting point of its interval. Further, $\Delta' t_i$ is approximately uniformly distributed in the range $(0, T)$. For example, Figure 3 shows the empirical distributions of the remainders of modulo 1000 operations over uniformly, normally and exponentially distributed random variables respectively. As we can see from Figure 3, the remainder of modulo operation over random variables of different distributions is approximately uniformly distributed. Consequently, for a TCP RST stream with sufficiently large number of TCP RST messages, at any interval length T which has $T > 0$ and $T \ll t_n - t_0$, the relative positions of TCP RST messages within their respective intervals ($\Delta' t_i$) are uniformly distributed.

Since a TCP stream experiences queuing delays when sending rate of the TCP stream goes above maximum bandwidth, estimating bandwidth is to determine whether a TCP stream experiences queuing delays. Hence, we define the centroid of a time interval as follow:

$$cent(T_j) = \frac{1}{n_j} \sum \Delta' t_i \tag{1}$$

where T_j is the jth time interval and $j = 1, ..., \frac{\Delta t_n}{T}$[4]; n_j is the number of TCP RST messages in the jth time interval and $\sum_{j=1}^{\frac{\Delta t_n}{T}} n_j = n$. Furthermore, $\Delta' t_i$ ranges from 0 to T. In case there is no TCP RST messages in the interval T_j, we define $cent(T_j)$ to be $\frac{T}{2}$. Extending the concept of centroid to a complete TCP RST stream, we then have

$$cent = \frac{T}{\Delta t_n} \sum_{j=1}^{\frac{\Delta t_n}{T}} cent(T_j) \tag{2}$$

Due to the fact that each interval T_j is uniformly distributed when queuing delays do not occur and a TCP stream follows a single routing path, the centroid of any time interval is approximately in the middle of the interval, i.e. $cent(T_j) = \frac{T}{2}$. Furthermore, the centroid of a complete TCP RST stream is equal to $\frac{T}{2}$. This property is stated and proved as follows.

Theorem 1. *If the sending rate of a TCP ACK stream R_0 is less than maximum bandwidth A (i.e. $R_0 \leq A$), then $cent = \frac{T}{2}$.*

[4]We tune the value of T and make Δt_n divisible by T.

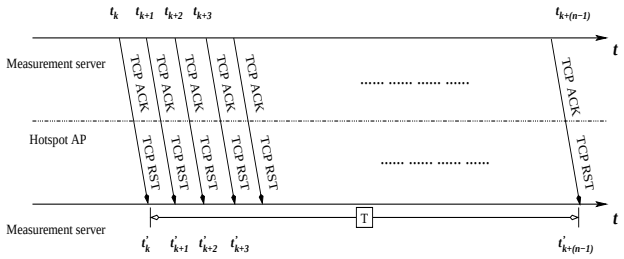

Figure 4: A bandwidth measurement server in the cloud sends a TCP ACK stream to a hotspot AP, and then the hotspot AP responds back to the server with a TCP RST stream, when $R_0 \leq A$.

Proof. Suppose that t_k is the absolute time stamp of TCP ACK message k and δ_k is the RTT of the TCP ACK message. Thus, the arrival time of the corresponding TCP RST message $t'_k = t_k + \delta_k$. When $R_0 \leq A$, $\delta_k = d_0$ for $k \in 1, 2, ...$ Since the sending interval for back-to-back TCP ACK messages is $\tau = t_{k+1} - t_k$, the arrival interval for back-to-back TCP RST messages is $t'_{k+1} - t'_k = t_{k+1} + \delta_{k+1} - t_k - \delta_k = \tau$. Furthermore, we assume time interval T. Over the interval $[t'_k, t'_k + T)$, n TCP RST messages arrive (see Figure 4) and so,

$$cent(T) = \frac{1}{n} \cdot \sum_{j=0}^{n-1} t'_{k+j} = \frac{1}{n} \cdot \frac{n \cdot (n-1)}{2} \cdot \tau = \frac{(n-1) \cdot \tau}{2} = \frac{T}{2} \tag{3}$$

Consequently, the centroid of a TCP RST stream is

$$cent = \frac{N \cdot cent(T)}{N} = \frac{T}{2} \tag{4}$$

where N is the number of interval in a TCP RST stream.

Recall the assumption that we makes in Theorem 1: a TCP stream follows a single routing path. However, this assumption is not always true in a real network [17]. A TCP stream may be split or merged, which causes slight transmission delays are imposed upon some packets in the TCP stream. The following theoretical proof illustrates that the centroid of a stream approximately remains $\frac{T}{2}$ even though stream splitting and merging happen.

Theorem 2. *When the sending rate of a TCP ACK stream R_0 is less than maximum bandwidth A (i.e. $R_0 \leq A$), stream splitting and merging will not skew the centroid of a TCP RST stream.*

Proof. Suppose the centroid of an interval fluctuates, the centroid of the interval becomes $\frac{T}{2} + o$ where o is the offset of the expected centroid. Then the centroid of a TCP RST stream is

$$cent = \frac{T}{2} + \frac{T}{\Delta t_n} \cdot o \tag{5}$$

If $\frac{\Delta t_n}{T}$ is large enough, the value of $\frac{T}{\Delta t_n} \cdot o$ approaches zero.

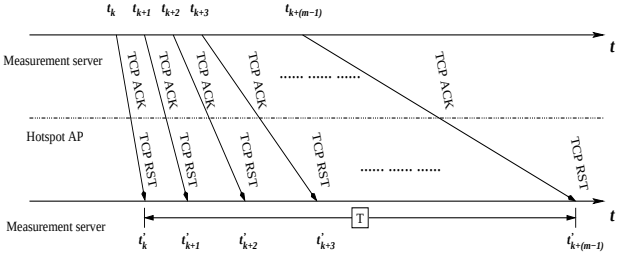

Figure 5: A bandwidth measurement server in the cloud sends a TCP ACK stream to a hotspot AP, and then the hotspot AP responds back to the server with a TCP RST stream, when $R_0 > A$.

Intuitively, when a bandwidth measurement server saturates either upstream or downstream channel of a hotspot AP, a queuing delay is imposed upon a TCP stream. Therefore, there is an obvious centroid offset for a TCP RST stream. This property is stated and proved formally as follows.

Theorem 3. *When the sending rate of a TCP ACK stream R_0 is greater than maximum bandwidth A (i.e. $R_0 > A$), the centroid of a TCP RST stream skews from $\frac{T}{2}$ to $\frac{T}{2} + d$, where d is an offset of the centroid.*

Proof. In this case, the arrival time t'_k for TCP RST message k is the sum of the absolute time stamp of corresponding TCP ACK message k and RTT of the TCP ACK δ_k, i.e. $t'_k = t_k + \delta_k$ where t_k is the absolute time stamp of corresponding TCP ACK message k (see Figure 5). Different from Theorem 1 where δ_k is a constant (i.e. $\delta_k = d_0$ for $k \in 1, 2, ...$), δ_k is a variable when $R_0 > A$, since each TCP message may experience a variable queueing delay. Therefore, the arrival time for back-to-back TCP RST messages is denoted as follow

$$t'_{k+1} - t'_k = t_{k+1} + \delta_{k+1} - t_k - \delta_k = \tau + (\delta_{k+1} - \delta_k) = \tau + \alpha_k \tag{6}$$

where $\tau = t_{k+1} - t_k$ and $\alpha_k = \delta_{k+1} - \delta_k$ with $\alpha_k > 0$. Over the interval $[t'_k, t'_k + T)$, the centroid for m TCP RST messages is

$$cent(T) = \frac{1}{m} \cdot \sum_{j=0}^{m-1} t'_{k+j} = \frac{1}{m} \cdot [\frac{m \cdot (m-1)}{2} \cdot \tau + \sum_{j=1}^{m-1} \alpha_j] = \frac{\tau \cdot (m-1)}{2} + \frac{1}{m} \cdot \sum_{j=1}^{m-1} \alpha_j = \frac{T}{2} + d_i \tag{7}$$

where $d_i = \frac{1}{m} \cdot \sum_{j=1}^{m-1} \alpha_j$ and $d = \frac{\sum_{i=1}^{N} d_i}{N}$ (N denotes the number of intervals in a TCP RST stream). Consequently, the centroid of a TCP RST stream is

$$cent = \frac{N \cdot \frac{T}{2} + \sum_{i=1}^{N} d_i}{N} = \frac{T}{2} + d \tag{8}$$

4.1.1. Verification

To verify our theorems, we investigate two hotspot APs that have the same downstream bandwidth (450 KB/s) and different upstream bandwidths (130 KB/s and 65 KB/s). Two measurement

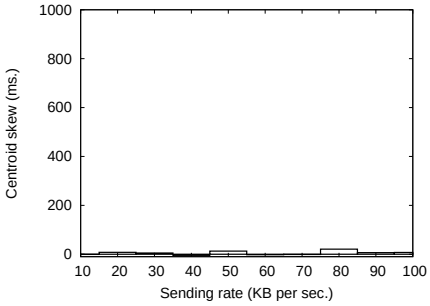

Figure 6: Centroid skew when bandwidth is 130 KB/s.

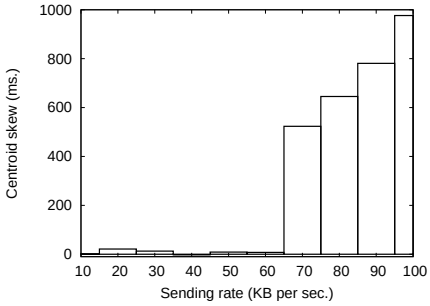

Figure 7: Centroid skew when bandwidth is 65 KB/s.

servers in the Amazon's EC2 platform send TCP ACK packets of size 40 bytes to the two hotspot APs respectively with the varying data rate, ranging from 10 KB/s to 100 KB/s. Figure 6 and 7 show the fluctuation of the centroid of the TCP RST stream around the time centroid for the AP with 130 KB/s and 65 KB/s upstream bandwidths respectively. We notice that this fluctuation is on both sides of the time centroid for 130 KB/s upstream link, where the sending rate is less than the bandwidth. On the other hand, for the 65 KB/s upstream link, when the sending rate is increased to 70 KB/s, the centroid of the TCP RST stream experiences a sudden jump. In addition, the centroid skew continues an upward trend beyond this sending rate. Therefore, to estimate the bandwidth, the measuremet servers monitor the variation of the centroid of the TCP RST stream and locate the point of sudden jump.

To locate the point of sudden jump in centroid, we use a binary search algorithm. Initially, we set lower and upper bounds (R_{lower} and R_{upper}) and a termination condition ($R_{upper} - R_{lower} \leq R_0$, where R_0 is the measurement resolution, which is the nearest bounding range between the lower and upper bound). The specific values for R_0, R_{lower} and R_{upper} are discussed in the next section. The search process iteratively adjusts the lower and upper bounds until $R_{upper} - R_{lower}$ is less than the measurement resolution R_0. The measurement server sends a TCP ACK stream at a certain rate $R(n) = (R_{upper} + R_{lower})/2$. If a centroid skew appears, the server adjusts the upper bound to $R_{upper} = (R_{upper} + R_{lower})/2$. On the other hand, if no centroid skew appears, the server adjusts the lower bound to $R_{lower} = (R_{upper} + R_{lower})/2$. This process is repeated until the termination condition is satisfied. The sending rate at this time is the estimated bandwidth.

4.2. Prototype implementation

We have implemented and experimented with two different prototypes of our bandwidth measurement servers. The first one is implemented in an indoor testbed at McGill University. We have used this prototype to evaluate the accuracy of our measurement methodology and estimated the value of measurement resolution, R_0. Next, to test the scalability of our methodology, we have implemented a prototype in the Amazon's EC2 platform [22]. Current EC2 platform offers 250 Mbps bandwidth for each instance (virtual machine), which guarantees that the bandwidth estimation component running on each instance can obtain a large enough outbound bandwidth. Since the outbound bandwidth of an instance is far larger than the outbound and inbound bandwidth of a hotspot AP (250 Mbps vs. 12 Mbps), an instance is capable of estimating hotspot AP backhaul bandwidth.

Our *xylophone* system in the cloud consists of three components: the hardware infrastructure, a distributed framework and a bandwidth estimation component. The hardware infrastruc-

ture serves as the underpinning of the cloud, providing support for computing, networking and storage. EC2 is employed in our implementation as the hardware infrastructure. We harness 11 medium High-CPU instances (one instance serves as master, the other ten serve as slaves), each of which serves as a measurement server in the cloud. Each instance is a 32-bit platform with 5 EC2 Compute Units[5] (2 virtual cores with 2.5 EC2 Compute Units each), 1.7 GB of Memory, 350 GB of local instance storage and 250 Mbps network capability [23]. To organize and manage these computing resources, we need a distributed framework to connect them together. Thus, our prototype implementation chooses Hadoop [24] to serve as our distributed framework. Finally, the bandwidth measurement component runs on each instance.

At a high level, our *xylophone* system in the cloud works as follows. We first assign measurement tasks to instances. In our prototype implementation, measurement tasks are described in a set of text files. Each of the files contains the IP addresses of hotspot APs and corresponding measurement parameters (e.g. measurement sampling rate and measurement resolution etc.). The master instance in the cloud (Hadoop's JobTracker) uniformly assigns the text files (i.e. measurement tasks) to those slave instances in the cloud (Hadoop's TaskTrackers). Following the description in the text file that the master instance assigns, a Hadoop's TaskTracker runs the bandwidth measurement component, collects hotspot AP backhaul bandwidth and outputs estimation results to another text file, in which bandwidth estimation results and estimation latencies are listed. Since output files are stored in Hadoop Distributed File System (HDFS), the Hadoop's JobTracker is able to access these output files, incorporate them into a measurement report and store the report into our centralized database. While the operation process is relatively straightforward, several important practical issues must be taken into consideration.

(1) Automatic task assignment in Hadoop platform is dependent upon the size of the input file. By default, HDFS cuts a huge file into 128 MB chunks. Each chunk is automatically assigned to one TaskTracker. However, *xylophone* cannot utilize automatic task assignment. Recall that the purpose of our *xylophone* system is to estimate hotspot AP backhaul bandwidth. Assume that tens of thousands of measurement tasks are described in a single file. The size of the file is at most several megabytes. This can give rise to a curious situation where all the measurement tasks are assigned to one TaskTracker to handle. To address this problem, we manually divide all the measurement tasks into a set of files. Thus, Hadoop platform can treat these files as separate HDFS chunks and distribute the files (i.e. measurement tasks) to multiple TaskTrackers. As a future work, we will automate this process.

(2) Since our measurement tasks are performed by instances in the cloud, it is possible that multiple instances share a single physical machine. Hence, instances may experience suspending and resuming. Once an instance is suspended, the measurement component running on the instance will be interrupted until the instance resumes. In practice, we observe that inter-instance suspension times for EC2's small default instances are large enough to invalidate our bandwidth estimation results, whereas medium High-CPU instances have much shorter suspension times. Consequently, we choose medium High-CPU instances, rather than the small default ones.

(3) The NICs (Network Interface Cards) in EC2 instances are also virtual devices. Therefore, the driver and the networking protocol stack in EC2 might be slightly different from those regular driver and networking stack in physical machines. According to our experiment, we observe that a virtual NIC driver does not timestamp incoming packets. This can cause the incoming packets stay at the system buffer for a long time without time stamps. To address this problem,

[5]One EC2 Compute Unit equals 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

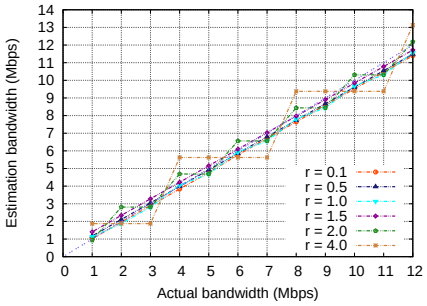

Figure 8: Bandwidth estimation.

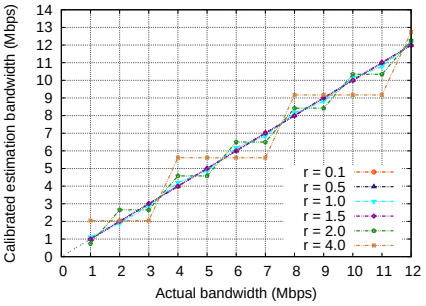

Figure 9: Calibrated bandwidth estimation.

we implement dual threads – one thread for sending outgoing TCP ACK packets and the other for receiving incoming TCP RST packets. This enables us to timestamp each incoming packet instantly and precisely.

(4) To enhance reliability of *xylophone* system, some environment parameters of Hadoop platform must be set properly. In particular, *mapred.task.timeout* should be set to 300, indicating that a measurement task should be canceled by the TaskTracker if it cannot output measurement results within 300 seconds. Note that a measurement task contains at most 5 hotspot APs in our current deployment and estimating bandwidth of one hotspot AP at most takes 60 seconds. In addition, we further set *setNumMapTasks* to 1. It indicates that at most one measurement task runs on a TaskTacker simultaneously. This is because if multiple measurement tasks run on one instance simultaneously, they may experience interference with each other.

5. Evaluation

We have evaluated *xylophone* extensively using our two prototype implementations. We have evaluated the following aspects – measurement accuracy, latency, scalability, intrusiveness and sampling rate.

5.1. Testbed

We have deployed an indoor testbed in McGill university to experiment with and evaluate the bandwidth estimation component of *xylophone* system in terms of accuracy, latency and intrusiveness. In addition, we have used this testbed to derive optimal values of some of the parameters used. The testbed consists of a Cisco Aironet 1131 AG series wireless AP, a Linux box and a Dell OptiPlex 980 Mini Tower. The wireless AP is used as a hotspot AP in our experiments and a virtual machine running on the Mini Tower is used to run the bandwidth estimation task. To emulate an asymmetric connection, we used the linux box as a traffic shaper. It bridges the wireless AP and the Mini Tower. The traffic passing through the Linux box is shaped using the Linux bandwidth (tbf) and latency (netem) shaping modules. In addition, we also experimented with 50 AT&T commercial hotspot APs in Denver, CO area to examine the performance of *xylophone* system in the context of real commercial hotspot environment.

Resolution ($r = \alpha$)	*Correlation coefficient*	$a_{r=\alpha}$	$b_{r=\alpha}$ *(Mbps)*
$\alpha = 0.1$	1.000000	0.9447	0.0755
$\alpha = 0.5$	1.000000	0.9375	0.2344
$\alpha = 1.0$	1.000000	0.9463	0.0891
$\alpha = 1.5$	1.000000	0.9375	0.4687
$\alpha = 2.0$	0.990300	0.9572	0.3409
$\alpha = 4.0$	0.953400	0.9572	0.3409

Table 1: The corelation coefficients between the actual bandwidth and estimation and calibration coefficients for different values of measurement resolution.

5.2. Measurement accuracy

We investigate the accuracy of bandwidth estimation using our indoor testbed. By using the bandwidth shaping module (tbf), we emulate different hotspot AP backhaul bandwidth. In this experiment, we set up upstream bandwidth of the AP to 500 Kbps and vary downstream bandwidth of the AP from 1 Mbps to 12 Mbps. The bandwidth estimation component running on the Mini Tower sends probes to estimate downstream bandwidth with different measurement resultions. Figure 8 plots our estimated bandwidth as a function of actual bandwidth for different values of measurement resolution. Overall, higher the measurement resolution is, less accurate our estimation is. In addition, we make an interesting observation from this figure. The estimation errors for each measurement resolution are consistent over bandwidth variation. Thus an interesting question is: *Are the estimations linearly correlated with each other?* To answer this question, we take the actual bandwidth in this experiment as the reference and calculate *Pearson product-moment correlation coefficients* between the actual bandwidth and our estimation for different values of measurement resolution. The calculation results are shown in the second column of Table 1, which validates that estimation results are highly correlated with actual bandwidth. Taking the actual bandwidth as reference, we can thus calibrate our estimation results through linear transformations. Let $B'_{r=\alpha}$ and $B_{r=\alpha}$ denote the calibrated estimation result and original estimation results for measurement resolution $r = \alpha$ respectively. The linear transformation from $B_{r=\alpha}$ to $B'_{r=\alpha}$ is

$$B'_{r=\alpha} = a_{r=\alpha} \cdot B_{r=\alpha} + b_{r=\alpha} \tag{9}$$

called calibration formula, where $a_{r=\alpha}$ and $b_{r=\alpha}$ are the calibration coefficients for measurement resolution $r = \alpha$. Using linear regression, we can calculate the value of $a_{r=\alpha}$ and $b_{r=\alpha}$. The last two columns of Table 1 list the calibration coefficients. Figure 9 shows the estimated bandwidth using this calibration.

For further comparison, we calculate the mean square error (MSE). Figure 10 plots the cumulative distribution of MSE before and after calibration. We can see that our estimation accuracy is significantly improved through calibration. These calibration formulas that we obtain from our indoor testbed will be further utilized in our real deployment. In addition, we can see that the calibrated results do not indicate a great improvement when the measurement resolution goes above 1.5. Therefore, our prototype implementation uses *measurement resolution=1.5*.

To further investigate the accuracy of our bandwidth estimation component, we have experimented with measuring the bandwidths of several commercial hotspot APs that are not under our control. Since these APs are not under our control, we have no way of knowing the actual hotspot

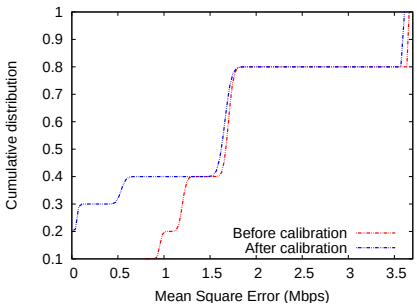

Figure 10: Mean square error of bandwidth estimation before and after calibration.

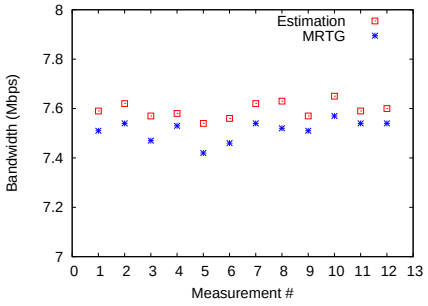

Figure 11: Comparison between bandwidth measured using MRTG and *xylophone*.

AP backhaul bandwidths that we can compare with our measurements. Instead, we compare our measurement results with bandwidth measured using another popular tool, MRTG[25]. We run MRTG behind a hotspot AP. MRTG measures hotspot AP backhaul bandwidth by monitoring incoming and outgoing traffic within a 5-minute time slot. Notice that bandwidth estimation component of *xylophone* takes far less estimation latency. To compare these short-term estimates with MRTG measurement, we continuously perform bandwidth estimation from an instance in the cloud for 5 minutes. Based on these measurements over a 5-minute time slot, we calculate an average 5-minute-bandwidth, $\bar{R}$ as follows:

$$\bar{R} = \sum_{i=1}^{K} \frac{T_i}{5} \times R_i \tag{10}$$

where K is the number of times the bandwidth is estimated in the 5-minute interval; T_i and R_i are the estimation latency and the estimated bandwidth respectively in the i^{th} time during the 5-minute interval.

Figure 11 shows the bandwidth estimated using MRTG during a 5-minute interval and the average 5-minute bandwidth during the same 5-minute interval for 12 different 5-minute intervals for one commercial hotspot AP. The main observation is that *xylophone* is able to accurately estimate AP backhaul bandwidth. MRTG measures the bandwidth right behind the hotspot AP, while our estimation probe messages traverse the Internet and finally arrive at the hotspot AP. The observation that these two measurements match fairly well confirms the result that the bandwidth bottleneck of a hotspot AP is usually located at the edge of the Internet [26][20].

5.3. Latency

5.3.1. Latency at WiFi subscribers end

We first evaluate the latency experienced by the hotspot subscribers when they measure performance values such as DHCP connectivity latency, open/blocked ports, and authentication and association times. Figure 12 shows the measured DHCP latency for 16 AT&T commercial hotspot APs in the Boulder, CO region over a week. The main observation is that the maximum latency for DHCP connectivity measurement is below 15 seconds for all APs except one. The average DHCP measurement latency is about 5 seconds. Also, we notice that some hotspot APs (hotspot AP 8 and 16) usually experience some DHCP problems. To measure the latency of port scans, we performed port scan of 127 common application ports in parallel. It took at most

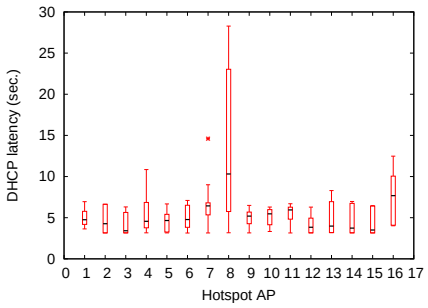

Figure 12: DHCP measurement on 16 hotspot APs over a week.

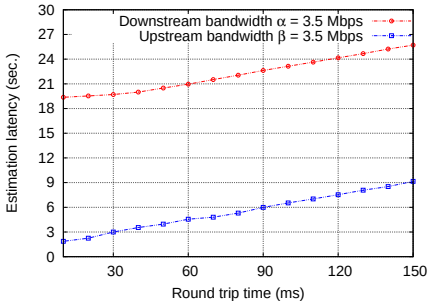

Figure 13: Measurement latency at varying transmission delays.

170 milli-seconds to complete this port scan for the 16 commercial AT&T hotspot APs. Finally, we also investigated the latency of the authentication and association mechanisms for different 802.11 security mechanisms as part of a previous project, see [27]. The maximum latency for authentication and association is about 2.49 seconds. Overall, we observe that the latencies for collecting the performance values on the WiFi hotspot subscribers end are quite low. Considering that these measurements require only minimal communication and computation, our conclusion is that *xylophone* design on WiFi hotspot subscribers end is quite reasonable.

5.3.2. Latency at bandwidth measurement servers

The Internet is highly dynamic in terms of transmission latency, bandwidth between the cloud and a hotspot AP, etc. In general, there is no way to predict bandwidth estimation latency. Instead of investigating overall latency for bandwidth estimation, we examine several factors that cause bandwidth estimation latency variation. By using latency shaping module (*netem*) in our indoor testbed, we emulate different transmission delays between the cloud and a hotspot AP. In one set of experiments we measure the upstream bandwidth by setting upstream bandwidth equal to 3.5 Mbps, and in another set of experiments, we measure the downstream bandwidth by setting downstream bandwidth equal to 3.5 Mbps. Figure 13 shows the estimation latency for estimating upstream and downstream bandwidths as a function of transmission delay between the measurement server and the hotspot AP varies from these two sets of experiments. The first observation here is that the estimation latencies are reasonably low, less than 27 seconds for round trip times as high as 150 ms. Second, we notice that the measurement latency shows approximately a linear increase with respect to increase in RTT. Based on this, we can say that the measurement latency will remain reasonably low even when the RTT goes beyond 150 ms.

Finally, we notice that measurement latency is significantly different for upstream bandwidth estimation and downstream bandwidth estimation. The main reason for this is that the TCP ACK packets that the instance sends out have different packet sizes while measuring upstream and downstream bandwidths. Recall that the instance uses 40-byte ACK packets to estimate upstream bandwidth but larger ACK packets (>>40 bytes) to estimate downstream bandwidth. Thus it takes much longer to send the same number of ACK packets while measuring downstream bandwidth than upstream bandwidth.

To understand the impact of packet size on bandwidth estimation latency, we conducted another set of experiments. In these experiments, the downstream bandwidth was set to three different values: 3.5 Mbps, 5.5 Mbps and 7.5 Mbps. By using different size of TCP ACK packets

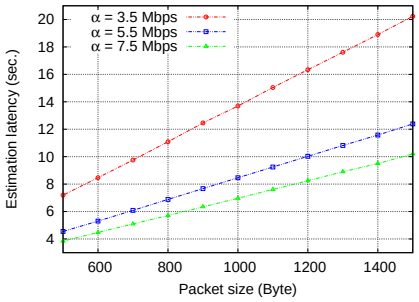

Figure 14: Measurement latency at varying size of a TCP ACK packet.

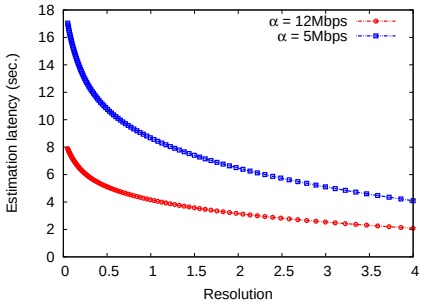

Figure 15: Measurement latency at different resolutions.

to estimate downstream bandwidth in our indoor testbed with a fixed RTT, we can observe in Figure 14 that the bandwidth estimation latency shows a linear increasing trend. The reason is that less delays are involved when an instance sends out a TCP ACK stream at a certain data rate using smaller TCP ACK packets. Another interesting observation in Figure 14 is that lower bandwidth shows higher estimation latency when an instance uses same size of TCP ACK packets to estimation bandwidth. The reason for this is quite straightforward – an instance spends less time in sending the same number of TCP ACK packets if these packets are transmitted through a high-speed link. This explanation is also validated in Figure 15.

Finally, we examine the effect of measurement resolution on bandwidth estimation latency (See Figure 15). In this experiment, we vary measurement resolution from 0.1 to 4.0. Note that the estimation latency exponentially decreases as resolution value increases. To understand this, recall that measurement resolution is part of the termination condition of our bandwidth estimation. Since bandwidth estimation is the process of a binary search, larger the resolution is, faster the search process will terminate. Therefore, we can easily reduce bandwidth estimation latency by choosing a greater value of measurement resolution. As we discussed in Section 5.2, choosing a greater value of resolution, however, may harm the accuracy of our bandwidth estimation. In order to balance this tradeoff between estimation accuracy and latency, the prototype implementation of *xylophone* uses the configuration – *measurement resolution = 1.5*.

5.4. Scalability

One practical issue with *xylophone* is how well will it scale when the number of APs is very high, e.g. 1000 APs? Clearly, we need multiple instances to handle such a large number of APs. To evaluate this, we implemented the bandwidth estimation component on Amazon's EC2 platform. In this experiment, we first perform our measurement task (i.e. estimating hotspot AP backhaul bandwidth of 50 AT&T hotspot APs in Denver and Boulder region) from a single instance. Then, we increase the number of instances and evenly divide bandwidth measurement tasks of 50 AT&T hotspot APs. Figure 17 shows the latency of measuring upstream and downstream bandwidths of all 50 APs for three different configurations: 1 instance, 5 instances and 10 instances. We notice that the measurement latency is significantly reduced when the number of instances in the cloud is increased from 1 to 10. This result is quite encouraging. In particular, with 10 instances, we can measure the upstream and downstream bandwidths of all 50 APs in less than 100 seconds. So, if we have to update the bandwidths of APs every 30 minutes, ten

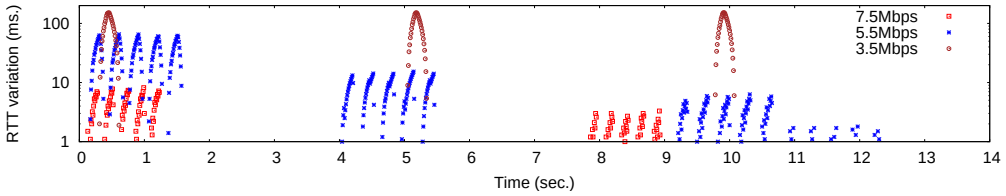

Figure 16: RTT variation during bandwidth estimation.

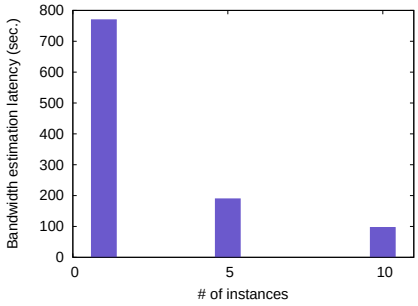

Figure 17: *Xylophone* scalability in terms of bandwidth estimation.

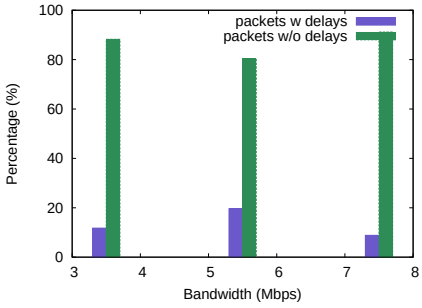

Figure 18: The impact of bandwidth estimation on crossing traffic.

instances can manage as many as 900 APs. The issue of how frequently the bandwidths should be measured is discussed in Section 5.6.

Another observation from Figure 17 is that though the number of instances are increased to 5 times and 10 times, the bandwidth estimation latency is not decreased by 5 times and 10 times. Two major factors contribute to this. The first is the overhead of managing all instances in the cloud. The other is the RTT between an instance and hotspot AP are not equal for different hotspot APs, so the actual distribution of which set of APs is assigned to different instances can impact the latency.

5.5. Intrusiveness

Since bandwidth estimation of *xylophone* system is based on the concept of self-induced congestion (i.e., an instance saturates hotspot AP backhaul bandwidth in the process of the bandwidth estimation.), an important question is how does the bandwidth estimation of *xylophone* impact the performance of ongoing user computations? For example, does it result in increase in transmission delays for hotspot subscribers? To quantitatively investigate this potential impact on transmission delays, we conducted an experiment in our indoor testbed. A client uses *hping2* to generate a TCP packet every 100 milliseconds through the AP we deployed in our testbed. Over this peroid of time, the instance probes the AP backhaul bandwidth. Figure 16 shows the transmission delays for some TCP packets from the client side increases. Furthermore, we can see that overall transmission delays show an obvious increasing trend when the TCP stream from the client side passes through a low-bandwidth link. The reason is that when TCP packets are buffered in a queue, lower bandwidth routers need longer time to empty the packet buffered in the queue. Another important observation from Figure 16 is that there are no drop-off packets.

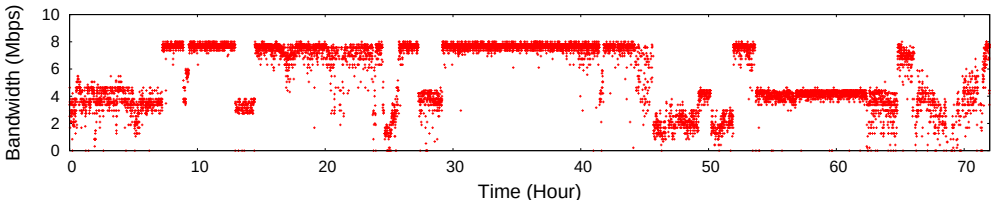

Figure 19: Time series plot of bandwidth at a hotspot AP in Boulder region, Colorado.

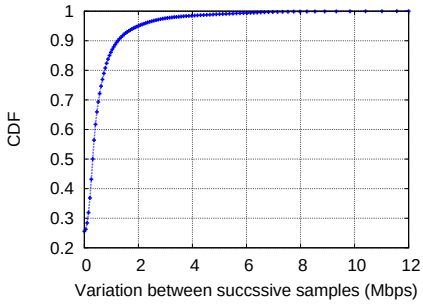

Figure 20: CDF of relative variation in bandwidth between successive samples.

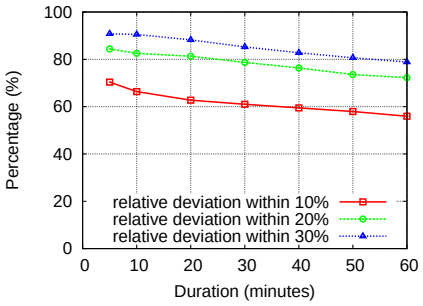

Figure 21: Percentage of slide windows in which the relative deviation is within 10%, 20% and 30%.

To examine the overall impact of bandwidth estimation on a TCP stream sent from a client side, we investigate what percentage of TCP packets from the client side experience longer transmission delays. Figure 18 indicates that only 10%–20% TCP packets generated from the client side experience longer transmission delays. The reasons for this relatively lower percentages are: (1) bandwidth estimation does not cause a persistent increase in queue size because a probe stream is never sent before the previous probe stream has been acknowledged; and (2) the process of bandwidth estimation is a process of binary search, and therefore, not every probe stream will saturate hotspot AP backhaul bandwidth.

5.6. Sampling frequency

As we discussed in Section 5.5, bandwidth estimation impacts the performance of hotspot subscribers in terms of increasing the transmission delays for a relatively small percentage of packets. Therefore, *xylophone* must reduce the frequency of bandwidth estimation of an AP. However, lower frequency may degrade the applicability of our system. For instance, suppose *xylophone* probes hotspot AP backhaul bandwidth every 60 mintues. During this period of time, hotspot AP backhaul bandwidth may vary significantly. Therefore, it is important to choose an optimal sampling frequency for bandwidth estimation. To do this, we need an understanding of how bandwidth changes over a given time period. So, we conducted a long-term experiment to analyze the temporal stability of bandwidth. The purpose of this analysis is to (1) explore how the bandwidth observed at a hotspot AP varies as a function of time; and (2) identify the relative stability of bandwidth over different time scales.

Data collection: We collected hotspot AP backhaul bandwidth information from 50 commercial hotspot APs in Denver and Boulder region. These hotspot APs were accessed and used daily by

hotspot subscribers. They vary widely from one another in terms of the number WiFi subscribers. Some of them are used by hundred and even thousand of hotspot subscribers daily (e.g. airport hotspot), while some others are used by less than 10 subscribers, e.g. a coffee shop. We estimated bandwidth $B(t)$ of these hotspot APs every 30 seconds. To describe the temporal stability of bandwidth, we select one of the most dynamic hotspot AP in terms of bandwidth variation. A snapshot of the time series plot of bandwidth from this hotspot AP is shown in Figure19 for a 70-hour time period.

Persistence: We first take a look at the amount by which successive bandwidth samples change. If many abrupt changes over short time scales occur, the bandwidth is unstable. We compute the Cumulative Distribution Function (CDF) of the amount of bandwidth change between successive samples and analyze it. Figure 20 shows the CDF of persistence for the measured hotspot AP, whose time series plot is shown in Figure 19. We find that successive samples of the hotspot AP vary by less than 8 Mbps over the entire trace period. However more than 90% of the successive bandwidth samples vary by less than 0.8 Mbps. Therefore, we can conclude that the measured bandwidth $B(t)$ is relatively persistent.

Time scales of stability: A persistent stable bandwidth $B(t)$ might not be stationary stable, where the mean and variance do not change over time or space. For example, consider a stochastic process X(t) in which variable X_i keeps increasing by a small amount with increase in i. In this case, $X(t)$ is persistent, but not stationary stable. If we take a close look at the time series plot in Figure 19, we notice that the bandwidth varies quite a lot over the entire observation period. Indeed, $B(t)$ is not stationary stable in general. We need to verify whether $B(t)$ is piecewise stationary, i.e. whether $B(t)$ is stationary stable over shorter time scales.

To check the piecewise stationary stability of $B(t)$, we set a time window w of some length, and slide this window over the measurement interval over the entire trace period. For each instance of slide window, we calculate mean and relative deviation. Figure 21 shows the percentage of slide windows in which the relative deviation is less than 10%, 20% and 30% for different window sizes ranging from 5 minutes to 60 minutes. For example, when we set time window $w = 20$ minutes, 81% windows on the trace have a relative deviation less than 20%. This means that the chance that the bandwidth varies by more than 20% with in a 20-minute period is about 19%. As expected, we note that the bandwidth is less stationary stable for larger window sizes. These results show that $B(t)$ is piecewise stationary for time intervals of the order of 20 to 30 minutes.

The results reported here are from a snapshot of the most dynamic hotspot AP in our experiment, and so this observation cannot be generalized to all the hotspot APs. However, other hotspot APs, especially for those hotspot APs with less dynamic behaviors in terms of bandwidth variation, show more stable behaviors. With this observation, our prototype implementation of *xylophone* uses 30 minutes as our bandwidth estimation interval, i.e. duty cycle for estimating hotspot backhaul bandwidth is 30 minutes.

6. Conclusion

This paper introduces *xylophone*, a hotspot WiFi locator service that allows a hotspot subscriber to search for available hotspots in a given area and enables hotspot subscriber to select commercial hotspots based on their current quality of service. The quality of service parameters include upstream and downstream bandwidths, connection delays in terms of DHCP connectivity latency, authentication and association latencies, and open/blocked ports. A novel bandwidth estimation methodology is introduced. Extensive evaluations of *xylophone* are provided which include evaluation from a prototype implementation in an indoor testbed as well as a prototype

implementation in the Amazon's EC2 platform. Evaluation results demonstrate low latencies experienced by WiFi subscribers to measure DHCP connectivity, authentication and association process, and discover open/blocked ports. Also, the bandwidth measurement component of *xylophone* exhibits high measurement accuracy, low latency, high scalability, and minimal intrusiveness.

References

[1] M. Jain, C. Dovrolis, End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput, IEEE/ACM Transactions on Networking (TON) 11 (4) (2003) 537–549.

[2] J. Strauss, D. Katabi, F. Kaashoek, A measurement study of available bandwidth estimation tools, in: Proc. of ACM SIGCOMM conference on Internet measurement, 2003, pp. 39–44.

[3] J. Pang, B. Greenstein, M. Kaminsky, D. McCoy, S. Seshan, Wifi-reports: improving wireless network selection with collaboration, in: Proc. of ACM MOBISYS, 2009, pp. 123–136.

[4] http://www.att.com/gen/general?pid=5949.

[5] https://content.hotspot.t-mobile.com/assetprocess.asp?asset=com.defau lt.main.001.

[6] http://v4.jiwire.com/search-hotspot-locations.htm.

[7] G. Judd, P. Steenkiste, Fixing 802.11 access point selection, ACM SIGCOMM Computer Communication Review 32 (3) (2002) 31–31.

[8] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, D. Wetherall, Improved access point selection, in: Proc. of ACM MOBISYS, 2006, pp. 233–245.

[9] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, D. Towsley, Facilitating access point selection in ieee 802.11 wireless networks, in: Proc. of ACM IMC, 2005, pp. 293–298.

[10] G. Kola, M. K. Vernon, Quickprobe: available bandwidth estimation in two roundtrips, in: SIGMETRICS Performance Evaluation Review, 2006, pp. 359–360.

[11] N. Hu, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS 21 (6) (2003) 879–894.

[12] S.-R. Kang, D. Loguinov, Imr-pathload: Robust available bandwidth estimation under end-host interrupt delay, in: Proc. of PAM, 2008.

[13] P. Papageorge, J. McCann, M. Hicks, Passive aggressive measurement with mgrp, in: Proc. of ACM SIGCOMM, 2009.

[14] D. Antoniades, M. Athanatos, A. Papadogiannakis, E. P. Markatos, C. Dovrolis, Available bandwidth measurement as simple as running wget, in: Proc. of PAM, 2006.

[15] X. Xing, S. Mishra, Where is the tight link in a home wireless broadband environment?, in: Proc. of IEEE/ACM MASCOTS, 2009, pp. 49–58.

[16] D. Crocey, M. Melliay, E. Leonardiy, The quest for bandwidth estimation techniques for large-scale distributed systems, in: ACM SIGMETRICS Performance Evaluation Review, 2010, pp. 20–25.

[17] X. Luo, E. W. Chan, R. K. Chang, Design and implementation of tcp data probes for reliable and metric-rich network path monitoring, in: Proc. of USENIX Annual Technical Conference, 2009, pp. –.

[18] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, V. Mhatre, Mdg: measurement-driven guidelines for 802.11 wlan design, in: Proc. of ACM MOBICOM, 2007, pp. 254–265.

[19] G. Maier, A. Feldmann, V. Paxson, M. Allman, On dominant characteristics of residential broadband internet traffic, in: Proc. of ACM IMC, 2009, pp. 90–102.

[20] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, J. Wang, A measurement study of internet bottlenecks, in: Proc. of IEEE INFOCOM, 2005, pp. –.

[21] K. Fukuda, K. Cho, H. Esaki, The impact of residential broadband traffic on japanese isp backbones, in: ACM SIGCOMM Computer Communication Review, 2005, pp. 15–22.

[22] http://aws.amazon.com/ec2/.

[23] http://en.wikipedia.org/wiki/amazon_elastic_compute_cloud/.

[24] http://en.wikipedia.org/wiki/hadoop.

[25] http://oss.oetiker.ch/mrtg/index.en.html.

[26] M. Dischinger, A. Haeberlen, K. P. Gummadi, S. Saroiu, Characterizing residential broadband networks, in: Proc. of ACM SIGCOMM conference on Internet measurement, 2007.

[27] X. Xing, S. Mishra, X. Liu, Arbor: hang together rather than hang separately in 802.11 wifi networks, in: Proc. of IEEE INFOCOM, 2010.