uScope: a Tool for Network Managers to Validate Delay-Based SLAs

Peshal Nayak
Samsung Research America
USA
p.nayak@samsung.com

Edward W. Knightly
Rice University
USA
knightly@rice.edu

ABSTRACT

This paper presents *uScope*, an AP-side framework for validation of delay-based SLAs. Specifically, *uScope* enables in the estimation of WLAN uplink latency for any of the associated STAs and decomposition into its constituent components. *uScope* does not require any form of active probing, no special purpose software installations on the STAs, nor any additional infrastructure to collect more information, and makes estimations solely based on passive AP-side observations. We implement *uScope* on a commodity hardware platform and conduct extensive field trials on a university campus and in a residential apartment complex. In over 1 million tests, *uScope* demonstrates a high estimation accuracy with mean estimation errors under 10% for all the estimated parameters.

CCS CONCEPTS

• **Networks** → **Transport protocols**; **Link-layer protocols**; **Cross-layer protocols**; **Network experimentation**; *Network performance modeling*; **Network performance analysis**; *Network measurement*;

ACM Reference Format:
Peshal Nayak and Edward W. Knightly. 2021. *uScope*: a Tool for Network Managers to Validate Delay-Based SLAs. In *The Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '21), July 26–29, 2021, Shanghai, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3466772.3467046

1 INTRODUCTION

A service level agreement (SLA) is a contract between a provider and clients detailing the performance assurances given by the network provider. These guarantees are given with respect to the numerous performance metrics. Failing to meet these assurances can have serious implications for the service providers. Therefore, SLA compliance monitoring is of significant importance to network managers. Further, identifying the root causes of poor performance metric values is critical to identify the diagnostic actions needed to meet SLA assurances.

One of the key metrics used in service level agreements is the WLAN latency [1, 2]. WLAN latency comprises three key components – channel access delay (which is further composed of 802.11 contention, retransmissions, and defer delays), queuing delays, and transmission delays (as determined by chosen data rates, transmission modes, overhead, etc.). Remotely monitoring WLAN latency for each client device in the network and decomposing it into its constituent components can enable service providers with SLA validation as well as to identify root causes of poor latencies for each client. Unfortunately, for cost and logistics reasons, the network service provider can deploy APs, but not additional devices (e.g., probing stations). Moreover, the service provider cannot expect clients to download special-purpose software to aid this endeavor. While a small subset of users might be willing to aid the provider, our objective is to monitor all clients, and we cannot practically expect all clients to provide such assistance and trust to their network provider.

WLAN latency and its components are determined by the joint effect of a number of factors such as the number of conflicting nodes, their traffic load, air time utilization, etc., whose impact can be directly observed at the transmitter. Therefore, remotely computing WLAN *downlink* latency and decomposing it into its constituent components is trivial based on direct observations obtained from the AP logs. However, an AP-side estimation of uplink latency is challenging for two reasons - (i) the factors affecting uplink latency are only known at the STA (*i.e.*, at the transmitter) (ii) the factors and magnitude of their impact can be different for different STAs in the network.

In this paper, we make the following contributions.

First, we present *uScope* (**u**plink latency micro**scope**), an AP-side framework for passive monitoring and analysis of WLAN uplink latency. While management and inference of WLANs and ad hoc network parameters has been the focus of intense research for decades, *uScope* is the first to enable estimation of WLAN uplink latency and breakdown into its constituent components. While doing so, *uScope* does not require any active measurements, special purpose software installation on the STAs (and hence no additional messages between the AP and STAs), nor any additional hardware infrastructure to collect more information. *uScope* estimates and decomposes uplink latency solely based on passive observations made from a single AP.

uScope employs virtual probing to enable measurement based analysis of uplink latency. The key idea in virtual probing is to employ layer-4 handshakes of the STA as virtual probes. Since the WLAN is the final hop for any TCP segment intended for a STA, the duration between transmission of a TCP segment on the downlink to reception of the TCP ACK on the uplink exposes the total WLAN uplink latency for that STA. Because virtual probing leverages the fundamental closed loop property of TCP, it can employ the layer-4 handshake of any TCP download (e.g., a Netflix video stream) to estimate WLAN uplink latency. Further, virtual probing does not impose any additional traffic load on the network as it uses the layer-4 handshakes that occur due to TCP.

However, the virtual probe only reveals the total uplink latency. To further decompose it into its constituent components *uScope* leverages the transmissions received from the STA during the handshake. *uScope* analyses the packet timestamps of these transmissions to decompose the total uplink latency. The packet timestamp methodology leverages the fact that the STA is guaranteed to be backlogged with at least one packet (the TCP ACK) in the duration between the end time of transmission of the TCP segment to the reception of the TCP ACK. Consequently, any intermediate transmission that occurs from the STA in this duration exposes the time when the TCP ACK reaches the head of the queue. Thus, by leveraging the timestamps of reception start and reception end of packets from the STA, *uScope* can estimate its queuing and access delays. Finally, *uScope* uses a novel estimation technique that couples virtual probing with knowledge of 802.11 protocol rules, to estimate the average number of retransmissions and the average defer delays faced by the STA.

Next, we implement *uScope* on an 802.11ac compliant off-the-shelf Access Point. The implemented framework comprises of over *7,000 lines* of code in Python to process the AP log and implement *uScope*. While *uScope* does not require any STA side information, for the purpose of evaluation, we also build APIs to collect STA side observations from portable laptops for measurement of ground truth values.

Finally, we deploy our commodity hardware based testbed on a university campus and in a residential apartment and perform a total of *1,296,000 tests* to validate *uScope*. Both of these trials are characterized by interference from co-existing BSSs, light user and environmental mobility, diversity with respect to links (*i.e.*, LoS and non-LoS paths), supported PHY rates, varying levels of traffic load, etc. In these field trials, the STAs run various internet applications performing video streaming, music streaming, pdf downloads, email activities, etc. Our field trials reveal that the estimation accuracy of *uScope* is dependent on the number of TCP handshakes that the AP observes. However, even with as few as 1,000 observed handshakes, *uScope* demonstrates an estimation error under 10% across all the parameters.

To the best of our knowledge, *uScope* is the first AP-side framework that passively estimates and decomposes WLAN uplink latency for any STA in the network. While doing so, *uScope* does not require any active measurements, special purpose software installations on the STA, additional hardware infrastructure and can make estimates solely based on passive AP side observations.

2 NETWORK SCENARIO AND PROBLEM FORMULATION

2.1 Network Scenario

We consider a network scenario with multiple basic service sets (BSSs) competing for spectral resources as shown in Fig. 1. While some BSSs can be a part of a managed infrastructure, there may be one or more non-managed interfering BSSs. Example of such non-managed BSSs could be personal hotspots, BSSs falling under a different network manager, etc. While the network manager can access the logs of the APs falling under the managed infrastructure, observations made by non-managed APs may not be accessible or readily available.

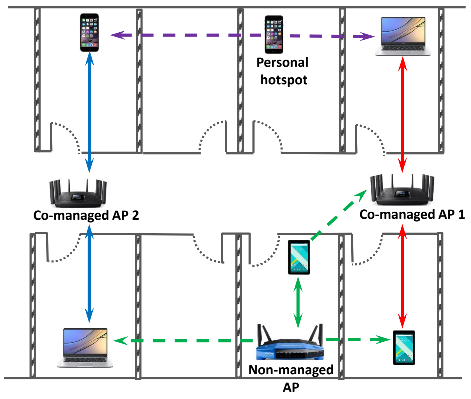


Figure 1: WLAN scenario comprising of a managed infrastructure co-existing with non-manged wireless devices. Solid lines indicate connectivity between devices whereas dotted lines indicate interference.

Ideally, all co-managed BSSs should operate on non overlapping channels thereby not interfering with each other. However, we consider that due to limited number of channels, some of the co-managed BSSs could co-exist together. Moreover, the connectivity among STAs and APs can form an arbitrary graph, allowing for complex interference and contention relationships. For instance, a subset of nodes that a particular STA can sense may be hidden from the AP. Likewise, some of the devices that the AP can sense may be hidden from some of its associated STAs. As a result, information obtained via logs of one or more managed APs may not fully characterize the contention and interference possibilities for each of their associated STAs. We make no assumptions about the physical layer capabilities of the nodes or their traffic loads. For instance, some APs and STAs may support advanced transmission modes such as MU-MIMO whereas others may have only legacy modes supported. Consequently, the channel utilization time for different STAs may be different for both uplink as well as downlink transmissions. The number and type of active applications may vary from STA to STA. Further, these applications can perform one or more uploads or downloads over TCP or UDP.

2.2 High level problem formulation

Let t_{enq} denote the time at which a particular packet to be analyzed is enqueued at the STA and $t_{\text{rx\_end}}$ denote the time when the AP successfully receives the packet. Our goal is to enable the AP to passively compute the mean uplink latency ($\bar{L}_{\text{uplink}}$), *i.e.*, the average duration between t_{enq} and $t_{\text{rx\_end}}$ for any of its associated STAs. Further for each associated STA, the AP should break down the mean uplink latency into its constituent components. While doing so, the AP cannot perform any active measurements (such as probing), cannot seek any STA side co-operation, does not have any additional hardware infrastructure (e.g., a network of sniffers) for collecting extra measurements and is thus constrained to make an estimate solely based on passive AP side observations.

For ease of discussion, the associated STA under consideration is hereby referred to as the target STA. Our network scenario comprises multiple BSSs co-existing together. Remaining STAs from the same and neighboring BSSs as well as APs from neighboring BSSs are hereby referred to as non-target STAs.

To understand the components of $\bar{L}_{\text{uplink}}$ and the challenges in estimating $\bar{L}_{\text{uplink}}$ and these components based on passive AP-side observations, consider the journey of a packet from when it gets enqueued at the target STA until it reaches the AP successfully. It is possible that when the packet gets enqueued, the target STA's queue already has a backlog of previously queued packets that are waiting to get serviced. Consequently, prior to reaching the head of the queue at time denoted by t_{head}, the packet will experience a queuing delay of $t_{\text{head}} - t_{\text{enq}}$. The average queuing delay is denoted by $\bar{\Phi}_{\text{queuing}}$. If the target STA has multiple flows pushing packets into the layer 2 queue, the average queuing delay $\bar{\Phi}_{\text{queuing}}$ will have contributions from each of these flows and consequently, $\bar{\Phi}_{\text{queuing}} = \sum_{i=1}^{N} \bar{\Phi}_{q,i}$ where $\bar{\Phi}_{q,i}$ denotes the average contribution to $\bar{\Phi}_{\text{queuing}}$ from the i^{th} flow from the target STA.

When the packet reaches the head of the queue at t_{head}, the STA chooses a random backoff and begins to contend in accordance with the rules of 802.11 to gain access to the wireless channel. The target STA contending for channel access might have to defer as some non-target STA captures the channel. It is also possible that the target STA captures the channel and makes an unsuccessful transmission (due to collisions, channel errors, etc.) forcing the STA to double its contention window size, choose another random backoff and attempt again. Finally, after several retransmission attempts the AP successfully receives the packet. The start time of this reception is denoted by $t_{\text{rx\_start}}$. We define the duration between t_{head} and $t_{\text{rx\_start}}$ as the uplink channel access delay with its mean value denoted by $\bar{\Phi}_{\text{access}}$.

Recall that the AP is constrained to estimate $\bar{L}_{\text{uplink}}$, $\bar{\Phi}_{\text{queuing}}$, $\bar{\Phi}_{q,i}$ and $\bar{\Phi}_{\text{access}}$ based on passive observations. Consequently, the AP encounters a few challenges. First, these parameters are determined by the joint effect of a number of factors such as network topology, interfering links to the target STA (from same BSS as well as neighboring BSSs), user activity, traffic load of interfering nodes, their PHY capabilities, data rates, etc. These factors are not directly observable by the AP. Further, for each packet received on the uplink by the AP, it cannot directly observe t_{enq} and t_{head}. As a result, the AP cannot estimate $\bar{L}_{\text{uplink}}$, $\bar{\Phi}_{\text{queuing}}$ and $\bar{\Phi}_{\text{access}}$ based on direct observation of uplink transmissions from the target STA. Since the AP is unaware of t_{enq} and t_{head}, for a packet received on the uplink from a target STA, the AP cannot directly observe which of the target STA's flows contributed to the queuing delay of the observed packet and the amount of contribution made.[1]

The mean uplink channel access delay ($\bar{\Phi}_{\text{access}}$) is affected by two key factors - (i) the number of retransmissions the STA has to make and (ii) the amount of time the STA defers prior to $t_{\text{rx\_start}}$. We denote the average number of retransmissions and the mean defer time per packet transmission by $\bar{R}$ and $\bar{\Psi}_{\text{defer}}$ respectively. Passive estimation of these parameters is challenging as the AP is unaware of which non-target STAs the target STA defers to as some of them may be hidden from the AP. This prevents the AP from directly observing the amount of time the channel is kept busy by hidden non-target STAs. The AP also cannot directly count the number of retransmissions from a target STA. Consequently, based on passive observations, the AP cannot directly compute $\bar{\Psi}_{\text{defer}}$ and $\bar{R}$ for a target STA.

Finally, as the STA makes a successful transmission, it occupies the channel for a time which includes any MAC layer overhead, interframe spacings, data transmission time and the time to send the MAC layer acknowledgement. The average duration between the start of the successful transmission and its completion is the mean transmission delay ($\bar{\Phi}_{\text{tx}}$).

Thus the mean uplink latency ($\bar{L}_{\text{uplink}}$), which is the average duration between t_{enq} and $t_{\text{rx\_end}}$, has contributions from three components as follows

$$\bar{L}_{\text{uplink}} = \bar{\Phi}_{\text{queuing}} + \bar{\Phi}_{\text{access}} + \bar{\Phi}_{\text{tx}}. \quad (1)$$

The goal of this paper is to validate delay-based SLAs for WLANs and assist the network manager in diagnostics solely based on AP-side observables. In other words, our objective is to enable the AP to passively estimate $\bar{L}_{\text{uplink}}$ and decompose it into its components. Since $\bar{\Phi}_{\text{tx}}$ can be directly observed by the AP, we focus only on $\bar{\Phi}_{\text{queuing}}$ and $\bar{\Phi}_{\text{access}}$. As stated previously, the AP should be able to further decompose these two components. Thus, the target STA can have an arbitrary number of flows contributing to $\bar{\Phi}_{\text{queuing}}$ and the AP should be able to infer their individual contributions ($\Phi_{q,i}$). Further, the AP should also be able to estimate $\bar{R}$ and $\bar{\Psi}_{\text{defer}}$ which affect $\bar{\Phi}_{\text{access}}$ for the target STA. Note that $\bar{L}_{\text{uplink}}$, $\bar{\Phi}_{\text{access}}$, $\bar{\Psi}_{\text{defer}}$, $\bar{R}$, $\bar{\Phi}_{\text{queuing}}$ and $\bar{\Phi}_{q,i}$ can vary from STA to STA depending on their traffic load and the network conditions experienced by the individual STA.

3 uSCOPE FRAMEWORK

3.1 TCP handshake as a virtual probe

Consider a layer-4 handshake: The STA receives a TCP segment from the AP on the downlink sent by any arbitrary server on the internet. Since the WLAN is the last hop for this TCP segment, reception of this segment results in the generation of a TCP ACK. This layer-4 handshake exposes three key attributes that are not directly observable to the AP.

(i) Uplink enqueue timestamp. The STA will attempt to return the TCP ACK as fast as possible. In other words, when the handshake occurs, the time when the TCP ACK gets enqueued is approximately the time when the TCP segment is received on the downlink from the AP [3]. Therefore, the enqueue timestamp for the TCP ACK can be inferred by the AP as $t_{\text{enq}}^{\text{ack}} \approx t_{\text{tx\_end}}^{\text{seg}}$.

(ii) Uplink aggregate layer-2 delay. After the layer-4 ACK gets enqueued at the STA, it experiences queuing delay as the STA transmits previously backlogged packets, if any. As the ACK reaches the head of the queue, it experiences additional delays from uplink contention, deferral, retransmissions and uplink transmission prior to reaching the AP. Thus, the duration between the transmission

[1]Under the assumption that the target STA is fully backlogged, t_{head} for a given packet is indeed the end time of the preceding packet's transmission. However, devices can also have idle times based on user activity.

of a TCP segment to reception of TCP ACK is a sum of all layer-2 delays that any packet transmitted by the STA would encounter.

(iii) Uplink backlog state indicator. In the time interval between transmission of the TCP segment to reception of the TCP ACK, the STA is guaranteed to be backlogged with at least one packet, the TCP ACK. As a result, until the AP receives the TCP ACK, the STA's queue can be considered to be in a backlogged state. Any other packet received from the STA in this period of time can be considered to be already present in the queue when the TCP ACK gets enqueued.

uScope leverages these three inferences obtained from a layer-4 handshake to estimate total uplink latency and its constituent components as described in the following subsections. Thus, the methodology used in *uScope* is to perform an assessment of uplink latency by leveraging layer-4 handshakes from TCP flows in which the target STA receives data. Analogous to active probing (e.g, ping) where the AP generates probe requests and the STA sends probe responses, *uScope* leverages TCP handshakes as virtual probes. However, unlike active probing, this methodology does not increase the traffic load as it leverages layer-4 handshakes that would anyways occur due to TCP. Further, since *uScope* leverages the fundamental closed loop property of TCP (which still carries more than 80% of the internet traffic today [4–8]) the handshakes can be a part of *any* TCP flow in which the target STA receives data (e.g., a Netflix video stream).

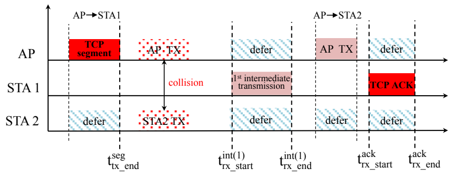
3.2 Timing analysis on virtual probes for delay estimation

For every packet received on the uplink, the AP can directly observe the timestamps for reception start ($t_{\text{rx\_start}}$) and reception end ($t_{\text{rx\_end}}$). Further, for each packet transmitted on the downlink, the AP can observe the timestamp corresponding to the end of the transmission ($t_{\text{tx\_end}}$). This section describes how virtual probing enables *uScope* to estimate the total uplink latency, access delay and queuing delay based on these timestamps for TCP segments and ACKs.

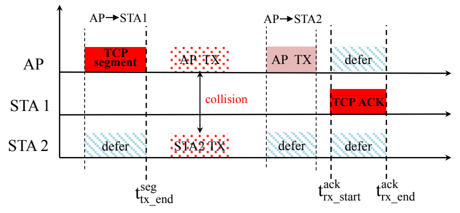
3.2.1 Total uplink latency estimation. The key challenge faced in estimation of the total uplink latency ($\bar{L}_{\text{uplink}}$) is that the AP cannot not directly observe the enqueue timestamp (t_{enq}) for a packet received on the uplink from a target STA. Inferring t_{enq} for all or a subset of packets received on the uplink would enable the AP to use $t_{\text{rx\_end}}$ (a directly observable parameter at the AP-side) and t_{enq} corresponding to such packets to obtain $\bar{L}_{\text{uplink}}$. However, as stated previously, when a layer-4 handshake occurs, the time at which the TCP ACK gets enqueued is approximately the time at which the TCP segment is received, *i.e.*, $t^{\text{ack}}_{\text{enq}} \approx t^{\text{seg}}_{\text{tx\_end}}$. Therefore, for these TCP ACKs, the enqueue timestamp can be inferred at the AP-side. *uScope* leverages this key idea to estimate the total uplink latency for a target STA as

$$\bar{L}_{\text{uplink}} = \overline{(t^{\text{ack}}_{\text{rx\_end}} - t^{\text{seg}}_{\text{tx\_end}})}.$$

3.2.2 Uplink access delay estimation. To estimate the access delay, the key information missing at the AP is the time when the packet reached the head of the STA queue (t_{head}). Suppose that there is a time interval in which the target STA's queue is backlogged with


AP→STA1
AP→STA2
AP
STA 1
STA 2
TCP segment
AP TX
defer
AP TX
defer
collision
1st intermediate transmission
TCP ACK
defer
STA2 TX
defer
defer
defer
$t^{\text{seg}}_{\text{tx\_end}}$
$t^{\text{int(1)}}_{\text{rx\_start}}$
$t^{\text{int(1)}}_{\text{rx\_end}}$
$t^{\text{ack}}_{\text{rx\_start}}$
$t^{\text{ack}}_{\text{rx\_end}}$

(a) **Queued ACK handshake**


AP→STA1
AP→STA2
AP
STA 1
STA 2
TCP segment
AP TX
AP TX
defer
collision
TCP ACK
defer
STA2 TX
defer
defer
$t^{\text{seg}}_{\text{tx\_end}}$
$t^{\text{ack}}_{\text{rx\_start}}$
$t^{\text{ack}}_{\text{rx\_end}}$

(b) **Immediate ACK handshake**

Figure 2: Timeline to show queued ACK handshake and immediate ACK handshake. In the illustration, STA 1 is the target STA and STA 2 is the non-target STA. In queued ACK handshake shown in (a), there is one intermediate transmission between $t^{\text{seg}}_{\text{tx\_end}}$ and $t^{\text{ack}}_{\text{rx\_start}}$. In the immediate ACK handshake shown in (b), the AP does not receive any intermediate transmissions from the target STA before receiving the TCP ACK.

a few packets. Each packet will reach the head of the queue at the end time of the transmission of the previous packet and thus the AP can infer t_{head} for each packet received in this duration on the uplink. This inference, combined with the time of reception start ($t_{\text{rx\_start}}$), which is directly observable at the AP, can help compute the access delay that the STA encounters. Unfortunately, the AP cannot directly observe time intervals when a STA's queue is backlogged as the STA can have some idle times based on user activity. However, virtual probing provides a unique opportunity to the AP: Recall that the STA is guaranteed to be backlogged until the TCP ACK is received, *i.e.*, in the interval from $t^{\text{seg}}_{\text{tx\_end}}$ to $t^{\text{ack}}_{\text{rx\_end}}$. Therefore, by using packets received on the uplink from the target STA in this duration, *uScope* can estimate the uplink access duration as follows.

Let us say that the AP observes N transmissions from the target STA between $t^{\text{seg}}_{\text{tx\_end}}$ and $t^{\text{ack}}_{\text{rx\_start}}$. We term such transmissions as intermediate transmissions. Handshakes with $N > 0$ are referred to as queued ACK handshakes and those with $N = 0$ are termed as immediate ACK handshakes. Fig. 2 shows an illustration of these two handshakes. *uScope* analyses the timestamps of the packets received from the target STA during these handshakes to obtain instantaneous values of access delay as follows. $\bar{\Phi}_{\text{access}}$ is computed by averaging over all these values.

(a) Queued ACK handshake: In such a case, the STA queue is not empty when the TCP ACK is enqueued. Consequently, the AP witnesses transmission of previously queued packets as intermediate transmissions prior to reception of the TCP ACK on the uplink. The queued ACK handshake provides instantaneous values of uplink access delay in two ways. One measurement is provided by each of the intermediate transmissions and then one comes from the TCP ACK as follows. For the j^{th} intermediate transmission ($1 < j \leq N$), $t^{int(j)}_{\text{head}} = t^{int(j-1)}_{\text{rx\_end}}$. Consequently, the instantaneous value of uplink access delay is given by $\left(t^{int(j)}_{\text{rx\_start}} - t^{int(j-1)}_{\text{rx\_end}}\right)$. The TCP ACK reaches the head of the queue after the N^{th} intermediate transmission. Consequently, for the TCP ACK, the instantaneous value of uplink access delay is given by $\left(t^{\text{ack}}_{\text{rx\_start}} - t^{\text{int(N)}}_{\text{rx\_end}}\right)$.

(b) Immediate ACK handshakes: In this case, the STA queue is empty when the TCP ACK gets enqueued and hence the TCP ACK does not wait prior to reaching the head of the queue, *i.e.*, $t^{\text{ack}}_{\text{head}} = t^{\text{ack}}_{\text{enq}}$. Recall that for all handshakes, $t^{\text{ack}}_{\text{enq}} = t^{\text{seg}}_{\text{tx\_end}}$ and hence the instantaneous value of uplink access delay for the immediate ACK handshake is given by $\left(t^{\text{ack}}_{\text{rx\_start}} - t^{\text{seg}}_{\text{tx\_end}}\right)$.

3.2.3 Queuing delay estimation and decomposition. Recall that our goal is to estimate the average queuing delay ($\bar{\Phi}_{\text{queuing}}$) and further decompose it into contributions coming from individual uplink flows. To this end, *uScope* leverages the two classes of TCP handshakes mentioned above as follows.

Consider the case of immediate ACK handshake. In this case, the target STA's queue is empty when the TCP ACK gets enqueued. Consequently, the ACK immediately reaches the head of the queue (*i.e.*, $t^{\text{ack}}_{\text{enq}} = t^{\text{ack}}_{\text{head}}$) and the queuing delay is 0.

However, in the case of a queued ACK handshake, the TCP ACK experiences a delay equal to the amount of time required to transmit the N packets queued before it. Consequently, the ACK reaches the head of the queue after the N^{th} intermediate transmission ($t^{\text{ack}}_{\text{head}} = t^{\text{int(N)}}_{\text{rx\_end}}$) and hence the instantaneous value of queueing delay is given by $\left(t^{\text{int(N)}}_{\text{rx\_end}} - t^{\text{seg}}_{\text{tx\_end}}\right)$. As before, $\bar{\Phi}_{\text{queuing}}$ can be computed by averaging over all the instantaneous values obtained from the two types of handshakes.

The net queuing delay has contributions from each flow coming on the uplink from the target STA. Each packet delays the TCP ACK from reaching the head of the queue by an amount of time equal to the duration between when the packet reaches the head of the STA queue to when the packet gets transmitted successfully. As shown in Fig. 2a, the contribution to the net queuing delay from packets of the i^{th} flow is given by $\left(t^{int(j-1)}_{\text{rx\_end}} - t^{int(j)}_{\text{rx\_end}}\right)$ if the j^{th} intermediate packet belonged to the i^{th} flow from the target STA.

3.3 Retransmission and defer delay estimation

The value of $\bar{\Phi}_{\text{access}}$ is influenced by two key factors, (i) $\bar{\Psi}_{\text{defer}}$ which is the amount of time the medium is sensed as busy by the target STA as it attempts to transmit and (ii) $\bar{R}$ which is the average number of retransmissions attempts per packet that the target STA makes prior to the successful transmission.

The AP's lack of knowledge of the connectivity links and contention relationships for a target STA prevents it from knowing which nodes a target STA defers to. In the network scenario we consider, due to asymmetry in the contention relationships for the downlink and uplink, the target STA may be deferring to nodes that are hidden from the AP. Likewise, the AP may hear nodes that are hidden from the target STA. Further, the AP cannot directly observe and record each failed transmission of the target STA. As a result, both $\bar{\Psi}_{\text{defer}}$ and $\bar{R}$ can be directly observed only at the STA and are unknown to the AP. This subsection presents a technique that *uScope* leverages to perform joint estimation of $\bar{\Psi}_{\text{defer}}$ and $\bar{R}$ passively at the AP.

Suppose that the STA has to make K attempts on average prior to a successful transmission (K is not known or observable at the AP). Let $\bar{Z}_k$ denote the average duration for the k^{th} attempt ($1 \le k \le K$). Each attempt duration consists of contention time, defer time and transmit time. Therefore,

$$\bar{A}_k = \bar{\theta}_{\text{contn,k}} + \bar{\theta}_{\text{defer,k}} + \bar{\theta}_{\text{occ,k}} \tag{2}$$

where $\bar{\theta}_{\text{contn,k}}$ represents the mean contention time, $\bar{\theta}_{\text{defer,k}}$ denotes the mean defer time and $\bar{\theta}_{\text{occ,k}}$ denotes the mean transmission time in the k^{th} attempt. We assume that $\bar{\theta}_{\text{occ,k}}$ remains fixed until the successful transmission and hence $\bar{\theta}_{\text{occ,k}} \approx \bar{\Phi}_{\text{tx}}$. Further, we also assume that $\bar{\theta}_{\text{defer,k}} \approx \bar{\theta}_{\text{defer,1}}$. These simplifying assumptions indeed result in an error. However, we show how *uScope* compensates for it later. While limited to first order effects, these assumptions lead to a simple methodology to estimate $\bar{\Psi}_{\text{defer}}$ and $\bar{R}$ that nonetheless lead to accurate results (as shown in later sections).

The number of such attempt durations that can fit within the average access delay ($\bar{\Phi}_{\text{access}}$) is equal to the average number of transmission attempts or in other words the average number of retransmission attempts ($\bar{R}$) plus one. The sum of defer time across all these intervals is the average defer delay ($\bar{\Psi}_{\text{defer}}$). Therefore, to estimate $\bar{R}$ and $\bar{\Psi}_{\text{defer}}$, *uScope* must estimate the number of attempt durations that can fit within the average access delay (a value that can be computed based on previously described techniques). However, notice that for doing this, *uScope* must have knowledge of the average duration of each attempt for a given STA. Unfortunately, the average attempt durations are not directly observable at the AP. Therefore, *uScope* must estimate the average duration for each attempt for a given STA. To this end, *uScope* uses a protocol based inference methodology coupled with virtual probing to make estimates for each attempt duration.

3.3.1 Protocol based inference. Under the above formulation, the duration of each attempt differs from the others based on the value of contention time. However, contention time can be estimated by leveraging knowledge of the 802.11 standard. Recall that the standard defines the rules governing the contention process. In each round of attempt, the target STA chooses a random number that is uniformly distributed in $[0, W-1]$ where W is the maximum contention window size. $W = 2^A$ where A starts with an initial value of 4 for the first attempt and increments for each round of transmission attempt. Therefore,

$$\bar{\theta}_{\text{contn,k}} = \frac{(2^{3+k} - 1) * \sigma}{2} \tag{3}$$

where σ is the slot duration. Based on our assumptions and from Eq. (3) and Eq. (2), we get

$$Z_k = \frac{(2^{3+1} - 1) * \sigma}{2} + \bar{\theta}_{\text{defer,1}} + \bar{\Phi}_{\text{tx}} \tag{4}$$

where $\bar{\theta}_{\text{defer,1}}$ is the average defer delay faced by the STA during first attempt. Therefore, to obtain an estimate for each Z_k, the only unknown left is $\bar{\theta}_{\text{defer,1}}$. However, the key challenge to estimate $\bar{\theta}_{\text{defer,1}}$ is that the AP cannot directly measure who the target STA defers to and for how long. Further, the AP also cannot directly observe when each retransmission attempt of the target STA started. As a result, $\bar{\theta}_{\text{defer,1}}$ is not directly known at the AP either. Next, we show how *uScope* leverages virtual probing to overcome this challenge.

3.3.2 Virtual probing for first attempt defer delay. Recall that virtual probing provides instantaneous values of uplink access delays as stated in the previous subsection. We restate that each uplink access delay is a sum of total contention time, defer delays and transmission delays. Consider TCP handshakes whose TCP ACK was successfully transmitted by the STAs in the first attempt. Notice that the average defer delay faced by such ACKs is the average defer delay for the first attempt, *i.e.*, $\bar{\theta}_{\text{defer,1}}$. Since the TCP ACK is transmitted in the first attempt, the average contention time can be computed based on Eq. (3) with $k = 1$. Further recall that transmission delays are directly observable to the AP. By subtracting these two quantities from the average defer delay of such TCP ACKs, *uScope* can estimate $\bar{\theta}_{\text{defer,1}}$.

To identify such handshakes, *uScope* uses the retry bit in the 802.11 MAC header of the packet carrying the TCP ACK. The retry bit indicates if the current packet experienced any retransmissions and is set to 1 for all retransmitted packets and 0 for those that are successfully transmitted in the first attempt. Note that the retry bit does not indicate the number of retransmissions experienced by the packet. Based on such handshakes, *uScope* estimates $\bar{\theta}_{\text{defer,1}}$ as

$$\bar{\theta}_{\text{defer,1}} = \overline{t^{\text{ack}}_{\text{rx\_end}} - t^{\text{ack}}_{\text{head}}} - (\bar{\theta}_{\text{contn,1}} + \bar{\theta}_{\text{tx}}). \tag{5}$$

Finally, to compensate for any errors arising from the iid assumption made about $\bar{\theta}_{\text{defer,k}}$, *uScope* adds the residual after subtraction of all possible Z_ks from $\bar{\Phi}_{\text{access}}$ to the $\bar{\Psi}_{\text{defer}}$ estimate.

4 uSCOPE IMPLEMENTATION

4.1 System implementation

The system consists of two main parts, a *stats manager module* which gathers AP side observations and a *stats processor module* which implements the *uScope* core.

Stats manager. The stats manager runs on the AP and implements the framework for collection of AP side observation logs. This module employs a libpcap engine [9] to collect logs from the network interface card of the device. The logs consists of aggregate transmission (TX) and reception (RX) statistics which include packet timestamps, 802.11 radio tap headers and packet headers for transmissions and receptions. Further, the logs also contain neighboring BSS's statistics which comprise of timestamps and airtime utilization for transmissions from co-existing BSSs.

Stats processor. The information gathered by the stats manager is operated upon by an instance of the stats processor. The stats processor is a Python based framework with over *7,000 lines* of code to implement the *uScope* core. This module implements techniques to parse the AP-side log and separate individual flows. The parsed log is piped into the *uScope* core which implements the techniques described in the previous section to estimate $\bar{L}_{\text{uplink}}$, $\bar{\Phi}_{\text{access}}$, $\bar{\Psi}_{\text{defer}}$, $\bar{R}$, $\bar{\Phi}_{\text{queuing}}$ and $\bar{\Phi}_{q,i}$.

4.2 Testbed characterization

Access Point. We use the Linksys WRT3200ACM as the Access Point. The device is an Armada-385 based router running a variant of Linux operating system. The AP has a dual architecture design wherein a general purpose system on chip (SoC) controls the mother board and a peripheral SoC executes the 802.11 PHY and MAC protocols. The dual band radio card has 4 external antennas providing antenna gains of 2.52 dBi in the 2.4 GHz band and 3.81 dBi in the 5 GHz band. Antennas with such gains are typically needed for WLAN operations [10–14]. The AP is IEEE 802.11ac compliant and supports up to 3 simultaneous spatial streams and up to 160 MHz bandwidth. The peak data rates supported on the AP are 2.6 Gbps (802.11ac, 5 GHz), 600 Mbps (802.11n, 2.4 GHz) and 54 Mbps (802.11a/g). The AP is equipped with a 1.8 GHz dual core ARM based CPU, 256MB Flash, 521MB DDR3 RAM, 4 Gigabit LAN ports and a Gigabit WAN port. While the evaluation has been performed on a 802.11ac compliant access point, *uScope* is not limited to 802.11ac. Note that the key principles involved in *uScope* do not depend on any 802.11ac specific feature and can thus be extended to upcoming Wi-Fi standards as well.

The complexity of the operations involved in *uScope* increases linearly with the number of devices for which the AP makes an estimate. While the computational capability of the AP is sufficient to support on board operation of both the stats manager and stats processor, the AP's on board memory becomes a critical bottleneck. Like most commodity hardware platforms, the AP's on board memory is optimized by the vendor to suffice the execution of only legacy functionalities. Consequently, less than 10% of the memory ($\approx$ 23MB) is available for implementing *uScope*. To overcome this bottleneck, we offload the stats processor module to a remote server which has sufficient memory for storage of AP logs whereas the stats manager runs on the AP. We further modify the stats manager in the following way. The statistics collected by the libpcap engine are temporarily stored in the kernel space of the AP which is probed by a periodic transfer routine. When a packet log is encountered by the routine, it fires a transfer command which sends the log to the remote server for storage. The estimates from the *uScope* core are stored in the output log on the remote server. The work flow is illustrated in Fig. 3. It is important to note that such an architecture increases the robustness of *uScope* in scenarios involving an AP failure. Since the logs are stored on the remote server, a device failure does not cause any loss of information. The time delay incurred by offloading the stats processor module to a remote server is determined by the wired delay on the link connecting the AP to the remote server.

STAs. The STAs are portable laptops running either Windows or Linux operating system. The STAs are upgraded to support IEEE 802.11ac by using an open source IEEE 802.11ac capable Edimax EW-7822ULC Wi-Fi chipset. The radio card has 2 internal antennas and supports communication on both 2.4 and 5GHz bands. The wireless interface has peak rates of 144, 300 and 867 Mbps while using the 20, 40 and 80MHz bandwidth respectively. Web activities on the STAs are performed by using the Mozilla Firefox web browser.

An instance of the stats manager also runs on the STAs to gather STA side observation log and stores it locally on the device for post-experiment retrieval. While *uScope* does not require STA side log for making estimations, this information helps in characterization of ground truth which is required for validation of *uScope*.

5 FIELD TRIALS OVERVIEW

Next we perform extensive field trials in which a number of factors co-exist together thereby enabling us to validate the performance

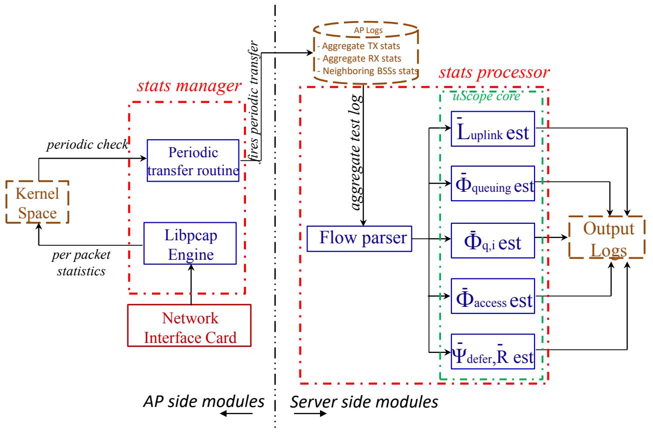


Figure 3: The workflow of commodity hardware based implementation of *uScope*. The implementation comprises of two key modules: (i) stats manager running on the AP and (ii) stats processor running on a remote server. The stats manager implements key functionalities for recording AP side observations which are sent to the remote server and piped into the *uScope* core for obtaining the final estimates.

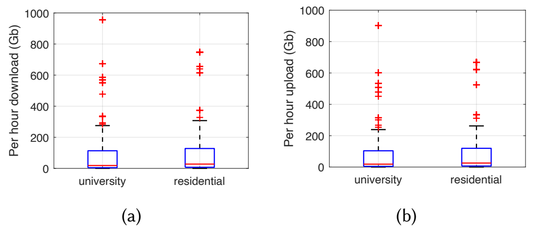


Figure 4: Aggregate per hour download and upload statistics in the university and residential deployments (a) Per hour download distribution, (b) Per hour upload distribution.

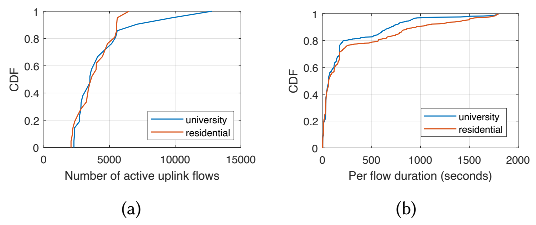
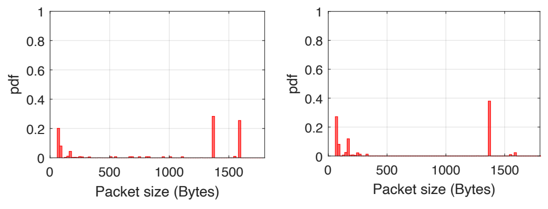
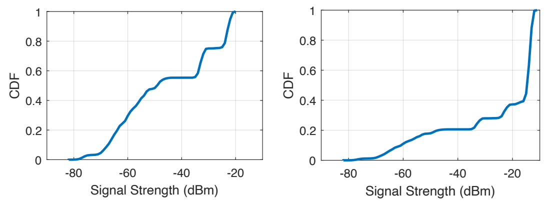


Figure 5: Aggregate uplink flow statistics in the university and residential deployments. (a) Distribution of number of active uplink flows per hour (b) Distribution of the duration of an active flow.



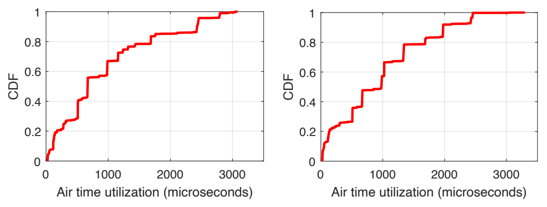
(a) **University deployment** (b) **Residential deployment**

Figure 6: Packet size distribution for the university and residential deployments. In each of these deployments, the minimum packet size encountered is around 200 bytes and the maximum packet size is around 1.6KB.



(a) **University deployment** (b) **Residential deployment**

Figure 7: Distribution of STA's signal strength in the university and residential deployments.



(a) **University deployment** (b) **Residential deployment**

Figure 8: Airtime utilization of devices in co-existing BSSs in the university and residential deployments.

of *uScope* in complex scenarios representative of a typical multi-BSS environment. The empirical values of total uplink latency and its constituent components are affected by a number of factors such as traffic load, network topology as well as MAC and PHY statistics. It is important that as these factors vary, *uScope* is able to accurately estimate total uplink latency as well as decompose it into is constituent components. This section characterizes the diverse operating conditions encountered in these tests.

Deployment overview. We deploy our AP and STAs in two locations. The first location is a university campus. Here the AP is deployed in a 3m x 5m office located in a 3 story building. The STAs are deployed both inside the office as well as outside the office at different locations on the same floor. The second deployment is an apartment located in a residential complex which comprises 1 or 2 bedroom units. Both of these environments comprise multiple BSSs co-existing together. In the university deployment, our devices co-exist with a university administered enterprise WLAN and 4 student deployed APs in nearby offices. Whereas in the residential environment, our devices co-exist with 12 APs from neighboring apartments. In each of these deployments, our AP has a total of 10 STAs associated with it. Each of these scenarios are characterized by a diversity in links (*i.e.*, LoS and non-LoS paths), light human and environmental mobility as well as device mobility. We validate *uScope* via an extensive number of tests conducted in these deployments. Specifically, the total number of tests run were *1,296,000* of which 576,000 were run in the university deployment whereas 720,000 were run in the residential deployment.

Traffic statistics. The STAs in our deployments run online internet applications that perform video streaming (using YouTube and Amazon Prime Video), music streaming (via Pandora), pdf downloads (from IEEE Xplore), email activities (using Gmail) and Gigabit file downloads and uploads to Dropbox and Google Drive. The tests involve cases with both single application traffic where only one of the above applications is run at a time as well as mixed application traffic were a number of these applications run in parallel. These internet applications are run using Mozilla Firefox web browser. In addition, some of the STAs also performed UDP downloads and uploads to local servers. A distribution of the amount of per hour download and upload is shown in Fig. 4 and a distribution of the number of active uplink flows and their duration is shown in Fig. 5. This variation causes a fluctuation in the queuing delays experienced by the STAs as well as a variation in the per flow queuing delay. Variation in packet sizes affects air time utilization causing a fluctuations in defer and queuing delays. The packet size distributions encountered in each of these scenarios are shown in Fig. 6. The minimum packet size is around 200 bytes and the maximum is around 1600 bytes.

Network topology. The network topology determines the number of interfering nodes which affects the retransmission rate and defer delays. These in turn affect the uplink access delay. The residential and university scenarios cover a total of *250* and *200* number of topologies respectively. These topologies arise from a combination of weak links, strong links, near and far away nodes, hidden terminals, etc. Further, the tests cover cases with active STA sets of all possible sizes and the AP makes an estimate for each associated STA.

MAC and PHY statistics. A diversity in both the network topology as well as the traffic characteristics of the STAs result in a variation in the MAC and PHY statistics of the devices in our network. This variation affects the total uplink latency as well as its constituent components. Fig. 7 shows the distribution of rssi values for the STAs during the experiments. Rssi affects selected data rates which affects transmission delays and hence total uplink latency. Transmission delays of non-target STAs affects the amount of time that the target STA defers thereby affecting its defer delays and access delays.

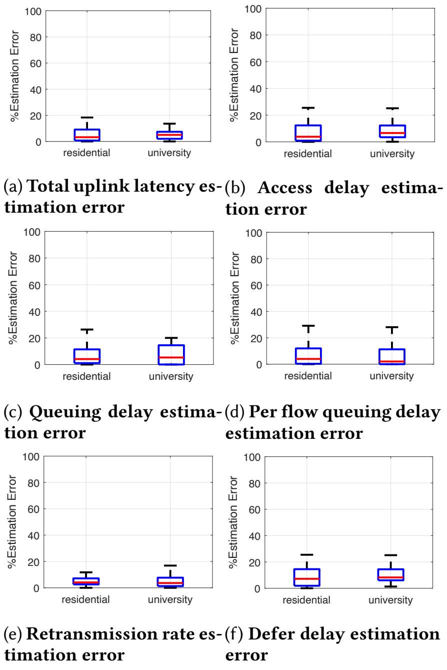
Both deployment scenarios are characterized by co-existing BSSs. As a result, the STAs in our network defer to these nodes depending on their air time utilization which affects their defer delays and access delays. Fig. 8 shows the distribution of air time utilization of devices in the co-existing BSSs.

6 EXPERIMENTAL EVALUATION

In this section, we investigate the performance of *uScope* in the two deployment scenarios.

6.1 Parameter estimation accuracy

First we experimentally evaluate the parameter estimation accuracy of *uScope*. Recall that the key idea in *uScope* is to treat TCP handshakes as virtual probes and use them to drive measurement based estimation of total uplink latency and its constituent components. In our deployment, the system estimates total uplink latency and its components for each associated STA every 30 seconds. Therefore, in this test duration, the system gathers AP logs and pipes them into the *uScope* core to obtain estimates. These estimates are timestamped and stored on the remote server. Recall that the instance of stats manager running on the STAs locally stores the STA log. Similar to the AP, this log is collected every 30 seconds. This log captures the ground truth observed by the STAs. In the deployment scenarios, the STA uses each packet transmitted on the uplink as a recording of ground truth. However, the AP only uses the TCP handshake coupled with the techniques described in the Sec. 3 to estimate the parameters.

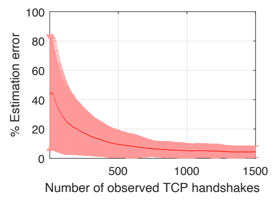


(a) **Total uplink latency estimation error** (b) **Access delay estimation error** (c) **Queuing delay estimation error** (d) **Per flow queuing delay estimation error** (e) **Retransmission rate estimation error** (f) **Defer delay estimation error**

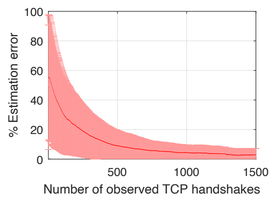
Figure 9: Parameter estimation accuracy of *uScope* for the university and residential deployments. The average estimation error across all the parameters is less than 10%.

We compare the estimates for $\bar{L}_{\text{uplink}}$, $\bar{\Phi}_{\text{access}}$, $\bar{\Phi}_{\text{queuing}}$, $\bar{\Phi}_{q,i}$, $\bar{R}$ and $\bar{\Psi}_{\text{defer}}$ made by the *uScope* system with those obtained from the STA side. To evaluate the parameter estimation accuracy, we compute the percent error in the estimate calculated as $\frac{|\beta_{est}-\beta_{gt}|*100}{\beta_{gt}}$ where β_{est} is the estimate for a parameter obtained from the *uScope* system and β_{gt} is the ground truth for the parameter obtained from the STA side.

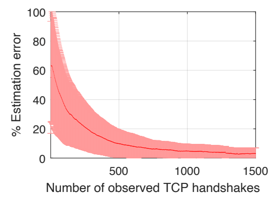
Fig. 9 summarizes the estimation error statistics for both the university and the residential scenario. Overall, *uScope* demonstrates a mean estimation error below 10% across all parameters. However, Fig. 9 reveals that in some cases, the worst case estimation errors are much larger compared to the average. This primarily occurs due to the following reason. Recall that *uScope* is driven by measurements collected from TCP handshakes at the AP side. As a result, for an accurate estimation, the AP must observe a sufficiently large number of TCP handshakes. In our deployments, the number of handshakes observed by the AP in any given test duration is dependent on the activity of the internet application running on the STA. Consequently, in test intervals where the AP observes an insufficient number of TCP handshakes, the estimate made by the AP demonstrates a high error. We further explore how estimation error varies as a function of number of TCP handshakes in the next subsection.



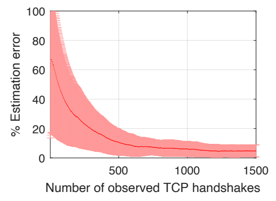
(a) **Total uplink latency estimation error**



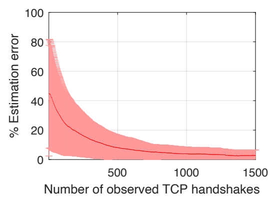
(b) **Access delay estimation error**



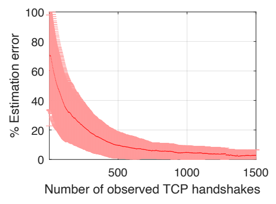
(c) **Queuing delay estimation error**



(d) **Per flow queuing delay estimation error**

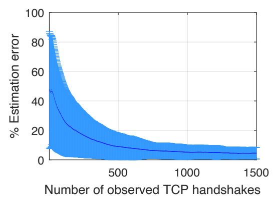


(e) **Retransmission rate estimation error**

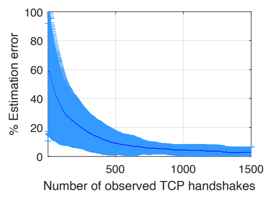


(f) **Defer delay estimation error**

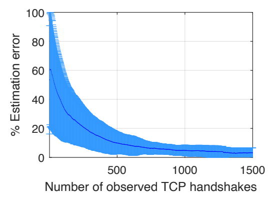
Figure 10: Mean estimation error for total uplink latency and its constituent components as a function of the number of TCP handshake measurements available to the AP while making an estimate in the residential deployment.



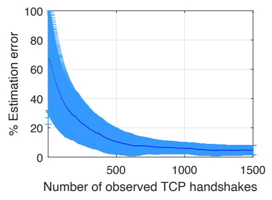
(a) **Total uplink latency estimation error**



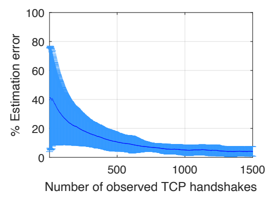
(b) **Access delay estimation error**



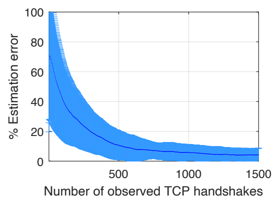
(c) **Queuing delay estimation error**



(d) **Per flow queuing delay estimation error**



(e) **Retransmission rate estimation error**



(f) **Defer delay estimation error**

Figure 11: Mean estimation error for total uplink latency and its constituent components as a function of the number of TCP handshake measurements available to the AP while making an estimate in the university deployment.

6.2 Estimation error characterization

uScope is a measurement driven framework. These measurements are collected by observing TCP handshakes from TCP flows of a STA. Consequently, the parameter estimation error is a function of the number of TCP handshakes that the AP observes. Here we experimentally investigate the relationship between the estimation error and the number of TCP handshakes observed by the AP.

In order to do so, we leverage the logs collected from the AP and all the STAs in the field trials. We modify the stats processor to iterate through the log and pass a specified number of consecutive TCP handshakes to the *uScope* core. As a result, *uScope* is constrained to make an estimate solely based on these samples. This number is varied to understand the relationship between estimation error and number of observed TCP handshakes.

Fig. 10 and 11 depict the estimation error as a function of the number of observed handshakes in the residential and the university scenario respectively. The estimation error demonstrates a similar trend across all the parameters. When the number of TCP handshakes observed by the AP are on the order of a few 100s, the mean estimation error is extremely high. This is because the number of measurements is not sufficient to characterize the total uplink latency and its constituent components. In this regime, the variance in the error is also high. Both the estimation error and the variance demonstrate a rapid decay with an increase in the number of observed TCP handshake. Fig. 10 and 11 reveal that even with over 1,000 observed TCP handshakes, the estimation error of *uScope* across all the parameters is less than 10%.

uScope can enable a network manager to estimate WLAN uplink latency and its constituent components for any STA in the network. This information is critical to the network manager to understand if the network infrastructure meets the performance assurances provided to the client in the SLA. Further, the latency breakdown provided by *uScope* can aid in building diagnostic models to identify causes and origins of performance problems for users in the network. For instance, does a STA experience poor TCP performance because it is deferring longer than expected? Does a STA experience poor performance due to a contention disadvantage resulting from an asymmetric network topology? Does the network overall experience a poor performance due to deferring to a co-existing BSS? Not just identifying a performance problem using *uScope* but also gaining an insight into the nature of the problem will enable network managers to make effective decisions on network infrastructure alterations.

Further, periodically storing *uScope* results for users in the network will provide the network manager with a rich dataset for user behavior characterization. For instance, application of machine learning techniques on such data sets could aid in the identification of diurnal patterns. Such patterns would be valuable especially in the flagging an abnormal network condition to sound an alarm to the network manager. Another application of user behavior analysis would be to cluster users based on their typical experienced performance. For instance, users could be grouped as those experiencing a "worse than normal" performance and their potential causes.

Knowledge of poor performance experienced by the user, its underlying causes and historical information of the network creates possibilities to perform corrective measures in an automated fashion from the AP itself. For instance, based on historical information and current estimated performance, an AP can be instrumented to predict if any network optimization strategy in the form of channelization strategies to alter interference, prioritization strategies, etc. could improve the performance of a particular user. Consequently, the AP can be empowered to dynamically optimize the network performance by leveraging the diagnostic models. In general, capability of such a what-if analysis will serve as a valuable tool to enable the network manager to predict the outcome of a network infrastructure alteration decision.

7 RELATED WORK

Active measurement based tools. Techniques such as [15–17] make use of AP initiated probing can be used to measure and decompose WLAN uplink latency. Likewise, tools such as [18–22] can be used to collect STA side information via user initiated measurements and reporting to analyze uplink latency. Unfortunately, these tools involve active measurements which increases the traffic load on the network. Consequently, their usage for periodic monitoring can potentially disrupt user traffic thereby worsening latencies for other users and draining the battery of mobile devices. Further, STA-side tools also require special purpose software installation on the STAs for data collection and reporting which end users may be unwilling to install. In contrast, *uScope* is completely passive, does not require any special purpose STA side software and makes an estimate solely based on AP side observations.

Passive measurement approaches. Deploying a network of sniffers can enable collection of packet traces which could facilitate a passive inference of the inter-node connectivity in the network and activity of interfering nodes for a given STA [23–25]. Such insights can enable estimation and decomposition of WLAN uplink latency for all STAs that the sniffers can sense. Unfortunately, such passive approaches involve deploying an additional hardware infrastructure which adds to the cost of network deployment and maintenance. On the other hand, *uScope* does not require any additional hardware infrastructure and can make an estimate solely based on passive observations from a single AP.

TCP based network analysis. TCP flows have been used to collect information that is valuable for network analysis, performance monitoring as well as detection of security threats. Tools like [26] analyze TCP headers to provide several IP and TCP statistics such as segment reordering, duplication, etc. which aids network measurement research whereas tools like [27] leverage observation of TCP flows in the network to predict achievable TCP throughput for all STAs in the network. On the other hand, [28, 29] utilizes TCP flow characteristics to identify security threats. In contrast, *uScope* leverages TCP's layer 4 handshake to measure and decompose WLAN uplink latency.

8 CONCLUSION

This paper presents *uScope*, a tool for validation of delay-based SLAs. In particular, *uScope* enables an estimation of WLAN uplink latency and breakdown into its constituent components solely based on passive AP side observations. We implement *uScope* on commodity hardware platform and run extensive field trials by deploying our AP on a university campus and in a residential apartment complex. In over 1,296,000 tests performed in these deployments, *uScope* demonstrates a mean estimation error under 10% across all the parameters.

REFERENCES

[1] ATT Service Level Agreement. http://cpr.att.com/pdf/se/0001-0003.pdf.

[2] Comcast Service Level Agreement. https://www.comcasttechnologysolutions.com/resources/service-level-agreement.

[3] S. Larsen et al. Architectural breakdown of end-to-end latency in a tcp/ip network. *International journal of parallel programming*, 2009.

[4] A. Marnerides et al. Internet traffic characterisation: Third-order statistics & higher-order spectra for precise traffic modelling. *Computer Networks*, 2018.

[5] P. Nayak, M. Garetto, and E. W. Knightly. Multi-user downlink with single-user uplink can starve TCP. In *IEEE INFOCOM*. IEEE, 2017.

[6] P. Nayak. *AP-side WLAN Analytics*. PhD thesis, Rice University, 2019.

[7] P. Nayak, M. Garetto, and E. W. Knightly. Modeling Multi-User WLANs Under Closed-Loop Traffic. *IEEE/ACM Transactions on Networking*, 2019.

[8] P. Nayak. Performance Evaluation of MU-MIMO WLANs Under the Impact of Traffic Dynamics. Master's thesis, 2016.

[9] V. Jacobson and S. McCanne. Libpcap: Packet Capture Library. *Lawrence Berkeley Laboratory, Berkeley, CA*, 2009.

[10] P. B. Nayak, S. Verma, and P. Kumar. Multiband fractal antenna design for Cognitive radio applications. In *Proc. of ICSC*. IEEE, 2013.

[11] P. B. Nayak, S. Verma, and P. Kumar. A novel compact tri-band antenna design for WiMax, WLAN and bluetooth applications. In *Proc of NCC*. IEEE, 2014.

[12] P. B. Nayak, R. Endluri, S. Verma, and P. Kumar. Compact dual-band antenna for WLAN applications. In *Proc. of PIMRC)*. IEEE, 2013.

[13] P. B. Nayak, S. Verma, and P. Kumar. Ultrawideband (UWB) Antenna Design for Cognitive Radio. In *Proc. of CODEC*. IEEE, 2012.

[14] R. Endluri, P. B. Nayak, and P. Kumar. A Low Cost Dual Band Antenna for Bluetooth, 2.3 GHz WiMAX and 2.4/5.2/5.8 GHz WLAN. *International Journal of Computer Applications*.

[15] K. Sui et al. Characterizing and Improving Wi-Fi Latency in Large-Scale Operational Networks. In *Proc. of MobiSys*, 2016.

[16] F. Cangialosi et al. Ting: Measuring and Exploiting Latencies Between All Tor Nodes. In *Proc. of ACM IMC*, 2015.

[17] N. Baranasuriya et al. QProbe: Locating the Bottleneck in Cellular Communication. In *Proc. of ACM CoNEXT*, 2015.

[18] R. Das et al. Informed Bandwidth Adaptation in Wi-Fi Networks using Ping-Pair. In *Proc. of ACM CoNEXT*, 2017.

[19] N. D. Mickulicz et al. Zephyr: First-Person Wireless Analytics from High-Density In-Stadium Deployments. In *Proc. of IEEE WoWMoM*, 2016.

[20] Kyung-Hwa Kim et al. MoT: A Collaborative Network Troubleshooting Platform for the Internet of Things. In *Proc. of IEEE WCNC*, 2014.

[21] K. Kim, H. Nam, and H. Schulzrinne. WiSlow: A Wi-Fi network performance troubleshooting tool for end users. In *Proc. of INFOCOM*, 2014.

[22] S. Rosen et al. MCNet: Crowdsourcing Wireless Performance Measurements through the Eyes of Mobile Devices. *IEEE Communications Magazine*, 2014.

[23] N. Mishra et al. Usage of 802.11n in Practice: A Measurement Study. In *Proc. of IEEE COMSNETS*, 2015.

[24] W. Zeng et al. Delay monitoring for wireless sensor networks: An architecture using air sniffers. *Elsevier Ad Hoc Networks*, 2014.

[25] U. Paul et al. Passive Measurement of Interference in Wi-Fi Networks with Application in Misbehavior Detection. *IEEE Trans on Mobile Computing*, 2013.

[26] M. Mellia et al. Tstat: TCP Statistic and Analysis Tool. In *Proc. of Springer QoS-IP*, 2003.

[27] P. Nayak, S. Pandey, and E. W. Knightly. Virtual Speed Test: an AP Tool for Passive Analysis of Wireless LANs. In *Proc. of IEEE INFOCOM*, 2019.

[28] Y. Chen and K. Hwang. TCP Flow Analysis for Defense against Shrew DDoS Attacks. In *Proc. of IEEE ICC*, 2007.

[29] L. Zhou et al. P2P Traffic Identification by TCP Flow Analysis. In *Proc. of IWNAS*, 2006.