Virtual Speed Test: an AP Tool for Passive Analysis of Wireless LANs

Peshal Nayak *, Santosh Pandey †, Edward W. Knightly *
*Rice University, USA, †Cisco Systems, USA

***Abstract*—Internet speed tests assess end-to-end network performance by measuring throughput for 10s of MB of TCP uploads and downloads. While such tests provide valuable insights into network health, they are of little use to network administrators since (1) the results are only available on the client that performs the test and (2) the tests can saturate the network, increasing load and worsening performance for other clients. In this paper, we present virtual speed test, a measurement based framework that enables an AP to estimate speed test results for any of its associated clients without any special-purpose probing, with zero end-user co-operation and purely based on passively observable parameters at the AP. We implemented virtual speed test using commodity hardware, deployed it in office and residential environments, and conducted measurements spanning multiple days having different network loads and channel conditions. Overall, virtual speed test has mean estimation error less than 6% compared to ground truth speed tests, yet with zero overhead, and outcomes available at the AP.**

I. Introduction

TCP speed tests are end-to-end tests of network health and are available via a plethora of online apps [1], [2], [3]. As part of the measurement process, a client performs an active TCP download and an active TCP upload to a server to measure the download and upload TCP throughput respectively. Since more than 80% of current Internet traffic is transmitted via TCP [4], the performance of numerous online applications is crucially dependent on the maximum TCP throughput achievable over an underlying network path.

If a client's speed test uses a nearby server (*i.e.*, a server with minimum possible latency to the AP), the WLAN becomes the key part of the end-to-end path. Consequently, the results would be valuable to the network manager to assess WLAN performance and make decisions on network infrastructure alterations to improve the quality of service experienced by the end user. However, the results can only be seen by the end user and are unavailable to the administrator without seeking end user co-operation. Moreover, regularly performing such speed tests imposes additional traffic load on the network and hence doing so can potentially disrupt user traffic and drain the battery of mobile devices.

In this paper, we make the following contributions. First, we present a framework that enables an AP to estimate the outcome of a speed test, *i.e.*, the upload and download TCP throughputs that any of its associated STAs should obtain from a nearby server, yet, without any special-purpose probing, with zero co-operation of endpoints (*i.e.*, the server and the client), and solely based on measurements that are passively observable at the AP. We call our measurement based framework *virtual speed test*. The speed test results obtained by a STA can vary over time depending on numerous factors such as the number of active STAs, interference level, etc. Likewise, virtual speed test can enable the network manager to dynamically track any given STA's speed test results based on its own unique characteristics (e.g., via a dynamic dashboard).

Virtual speed test employs a novel L2 edge TCP model to perform throughput estimation. The key challenge for the AP to estimate these inherently bi-directional, end-to-end and layer-4 throughputs, is that the AP only has a limited view of the network. Since the AP is unaware of the presence of hidden terminals, interference from neighboring BSS to the STAs, etc. (which affect the STA's queuing delays, NAV timers and packet retransmissions), the AP cannot estimate how long it takes a STA to successfully transmit after it starts to attempt. Our design is motivated by the fact that since the WLAN is the final hop for any TCP segment directed towards a STA, this duration can also be estimated by measuring the delay incurred between the transmission of a TCP segment on the downlink to the reception of the corresponding TCP ACK on the uplink from the STA. This TCP segment, therefore, can belong to any TCP flow (e.g., a Netflix video stream) and need not be a part of a flow from a nearby server. To carry out these measurements, the AP must identify TCP flows. To this end, we leverage TCP's inherent bi-directionality and packet size signatures to spot TCP flows. Specifically the fact that TCP flows involve TCP segment traversing on the forward path and small sized TCP ACKs on the reverse path enables the AP to identify these flows and perform its measurements.

Second, we develop a virtual speed test enabled AP (VST AP) by using commodity hardware. We build APIs that enable the VST AP to passively collect a number of per packet statistics and feed them into the L2 edge TCP model to obtain throughput estimates. While virtual speed test does not require collection of STA-side statistics, for validation purposes, we also implement APIs for data collection from STAs associated with the VST AP for characterizing the operating environment and for ground truth measurement. We deploy VST AP in two environments: an office located inside a university building and an apartment in a residential complex. The VST AP is deployed in the office for a period of 2 days and in the apartment for a period of 7 days. Both deployment settings are characterized by interference from non-BSS devices co-existing in the same frequency band, human mobility and link diversity with respect to signal

Fig. 1: Enterprise WLAN scenario: bold lines indicate connectivity while dotted lines indicate interference.

propagation (*i.e.*, LoS vs non-LoS paths) and supported PHY rates. The office and the residential scenario cover a total of 36 and 49 topologies respectively with a varying number of STAs. Overall, the VST AP observes a total of 113,047 TCP flows across both deployments. These TCP flows result from multiple applications running on end devices such as video streaming, music streaming, pdf downloads and email activities. For validation, actual client-based speed tests are employed as ground truth. Virtual speed test demonstrates a high level of estimation accuracy compared with ground truth, with average estimation error under 6% for both upload and download speed estimation.

Finally, we implement virtual speed test into ns-3's source code and perform extensive simulations to investigate operating conditions beyond those encountered in our field trials. The simulation results concur with field trial conclusions demonstrating estimation errors below 5%.

To the best of our knowledge, virtual speed test is the first to estimate both upload and download TCP throughputs of STAs in the network by using passive measurement metrics at only the access point, *i.e.*, without any active probing, additional hardware infrastructure or user participation.

II. Virtual Speed Test: Scenario Description and Problem Formulation

A. Enterprise WLAN setup

We consider an enterprise WLAN environment such as illustrated in Fig. 1. As depicted, the network comprises of multiple APs. While the network may use channelization, for ease of exposition we consider only APs with at least partially overlapping channels such that they can potentially interfere with each other. Moreover, we consider that in addition to the managed infrastructure, there may be one or more non-managed WLANs that may be interfering. Such WLANs can correspond to an LTE hot spot or a neighboring WLAN under different administrative control.

Ideally, all such networks should have sufficient physical separation to enable full spatial reuse for each AP (*i.e.*, simultaneous transmission for each network). However, as depicted, the unwanted interconnectivity creates interference and contention among nodes. Moreover, inter-node connectivity can form a complex relationship: while all STAs are necessarily connected to the APs that they associate with, a particular STA may or may not be in range of other APs. Likewise, STAs may be hidden from each other or mutually in range. It is further possible that a STA is in range of other APs which are not in range of the AP that is serving it. The interference and contention possibilities are further compounded by the need to consider both downlink transmissions (AP to STA), uplink transmissions (STA to AP), and mixes.

We do not make any assumptions about the PHY layer capabilities of the AP or the STAs. For instance, the AP may have advanced physical layer capabilities such as multi-user MIMO. Likewise, the AP can have any channelization strategy, e.g., dynamically bonding channels to 80 MHz as available.

B. Background on TCP upload and download speed test

Speed tests measure a client's upload and download TCP throughput from a server on the internet.[1] If the speed test happens from a nearby server or low latency server, the WLAN becomes the key part of this end-to-end path and the network manager can use these results to assess WLAN performance. For the remainder of the paper, we focus on speed tests that happen from a nearby server. A speed test is user initiated and the results are visible to the user at the end of the measurement. Speed tests primarily consist of two phases: a setup phase during which the speed test parameters are configured and a measurement phase which involves an active TCP upload and download.

Setup phase. The setup phase begins with a server selection process which can either be manual or app driven. If this is app driven, a server is selected by probing a pool of available servers such that the backbone delay between the server and the AP is as less as possible to ensure a maximum TCP throughput [5] (with the ideal case being a server in the same LAN as the AP). Since the goal is to measure the maximum TCP throughput, while running a speed test, a STA is recommended to turn off other applications. Next, the client and server side TCP parameters are configured. The exact mechanism used for performing this configuration differs from one speed test application to another. A commonly used procedure is to conduct a test download and a test upload from the STA. For instance, for the Ookla speed test, the STA initially downloads or uploads a small file to estimate initial throughput. Following this initial phase, the STA adjusts the file size, buffer size and number of parallel TCP flows (limited to maximum of 8) to maximize the network connection usage while preventing congestion during the measurement phase [6].

Measurement phase. As shown in Fig. 2, the measurement phase consists of two sessions: an upload session and a download session. The vast majority of speed test apps available online follow a flooding based mechanism in these sessions which involves establishment of several parallel TCP flows between the server and a STA with a calculation of aggregate throughput across all the flows [7]. This ensures that the results are robust to a small TCP window size (e.g., due to loss of a

[1] Unless stated explicitly, the terms client and STA are used interchangeably.

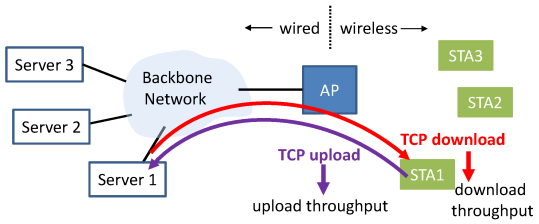


Fig. 2: Illustration of an upload and download speed test: here STA 1 performs a speed test. Server 1 is selected from a pool of servers consisting of server 1, 2 and 3. Here server 1 has minimum backbone delay to the AP.

TCP segment or a small advertised receive window size) [8]. The number of parallel flows to be established is determined in the setup phase. During the upload session, a STA performs an active upload to the selected server and measures the TCP throughput by averaging the total data transmitted end-to-end over the total time taken. During the download session, the STA performs an active download and measures throughput in a similar fashion.

C. Virtual Speed Test: High level problem definition

Analogous to online speed tests, our goal is to realize a *virtual* speed test that enables an AP to estimate the TCP download and upload throughput that a STA can achieve from a nearby server. As described in our network scenario, an AP can have an arbitrary number of STAs associated with it and the AP should be able to estimate the throughput for any of the associated STAs. The speed test results of a STA can vary as driven by factors such as number of active STAs, interference level, etc. Likewise, we target that virtual speed test also tracks the speed test results for a given STA based on its own unique characteristics. Note that the STA does not perform the actual speed test. The AP is required to make the prediction using only passively collected information available on the AP side whereas no reports are available from STAs and out-of-network APs. Further, we consider that no additional commands can be required of STAs, e.g., STAs cannot be requested to send packets for testing purposes. Moreover, STAs cannot be requested to download special purpose software or report STA-side measurements. Instead, we consider that by leveraging AP-side observables, the AP can estimate the following metrics [9].

Aggregate AP metrics. We consider that the AP can measure the airtime usage due to transmission and reception, defer time, contention time, idle time (no backlogged downlink traffic) as well as byte counts for downlink and uplink frames.

Per-STA metrics. Likewise, while the STAs do not report STA-side statistics, the AP can observe some per-STA metrics at the AP such as uplink RSSI and SNR, downlink and uplink MCS and PHY parameters including use of advanced PHY features such as channel bonding, spatial multiplexing, multi-user transmission and downlink retransmission statistics.

Non-associated device metrics. Lastly, the AP may be in range of a number of non-associated 802.11 devices that are transmitting on a different BSS. When the AP is forced to defer to a non-BSS device, it can record interferer air time consumption.

While the above may appear to be an exhaustive set of information for performance characterization, there are a number of STA-side metrics that remain unobservable by the AP. For instance, the AP does not know the STA's idle times or the STA's defer times due to NAV especially when the STA is deferring to a non-BSS device. Since the network scenario considers a complex inter-node connectivity which may lead to inter-cell interference, hidden terminals, etc., these parameters cannot be directly calculated based on the metrics mentioned above. However, the throughputs that we want the AP to estimate are inherently bi-directional, end-to-end and layer-4 and can indeed be degraded by the above factors. To this end, we infer the impact of these unknowns using the above AP-side observables with the help of techniques described in Section IV.

III. L2 EDGE TCP MODEL

To enable an AP to estimate the upload and download throughputs that a STA would obtain if it performs a speed test, we develop a novel L2 edge TCP model that uses AP-side observables as inputs.

A. Assumptions for mathematical analysis

Here, we state the key assumptions that we make to capture important aspects of the aforementioned speed test setup and measurement phases in our model.

In the measurement phase of an actual speed test, multiple TCP flows are initiated between the server and the client so that the measurement phase is not bottlenecked by the number of circulating TCP segments. Instead, we model this by representing the packet flow dynamics by a single long lived TCP flow with a maximum congestion window size of W_m which is large enough so that there are a sufficient number of TCP segments circulating in the network. We further assume that this flow does not experience any permanent packet losses. This is not to say that collisions or packet errors do not occur on the wireless channel. Rather, packets lost on the wireless channel are locally retransmitted by the MAC layer and we do account for these collisions and retransmissions in our analysis. We hereby refer to this modeled flow as the speed test flow, the STA under consideration as the target STA and the remaining STAs as non-target STAs.

In an actual speed test, the server selection process in the setup phase selects a server with minimum latency to the AP to reduce the impact of backbone elements on the measured results. Consequently, we consider backbone congestion and delays as factors that do not impact the throughput. Also, recall that the parameters of the TCP flow used during the speed test are adjusted by the STA based on an initial measurement performed to ensure that TCP does not drive the network into congestion.

B. Virtual end point representation

The discussions in this sub-section are mainly in the context of a download speed test. However, the arguments and

explanation are applicable to upload speed tests as well and will be generalized later.

In the network scenario of Fig. 1, there are no restrictions on the traffic flows of non-target STAs and STAs in neighboring BSSs and they may have UDP and/or TCP traffic going on the downlink and/or the uplink. Further, the number of these flows per device can also be variable and differ from STA to STA. Since we make no assumptions about the network topology, interfering links or the type or number of flows, it is not possible to state precisely the inputs for a queuing model. We remark that a majority of TCP models for Wi-Fi require AP-side knowledge of network topology, interfering nodes including those from neighboring BSSs, their traffic patterns, PHY capabilities, data rates, etc. Removing this requirement is vital to the realization of virtual speed test.

The modeled speed test flow comprises both its TCP segments and TCP ACKs. First, we analyze the speed test flow by considering the journey of a speed test flow segment from the server to the target STA. On the forward path, a TCP segment experiences delays on the queues of devices on the backbone.[2] When the packet enters the queue at the AP, it encounters another delay before reaching the head of the queue, part of which arises from the AP serving non-speed test flow packets. We denote the average amount of time the AP spends on non-speed test flow packets prior to serving a speed test flow packet by V. Upon reaching the head of the queue, the AP begins to contend to access the channel. It is possible that as the AP counts down, the target STA or a non-target STA or another AP wins the channel, causing the AP to defer. It is also possible that a transmission from the AP fails either due to collision or poor channel quality, forcing the AP to double its contention window size and re-contend and transmit (with the same or adapted data rates[3]). We denote the mean time the AP takes to win the channel prior to a successful transmission by d_{access} as shown in Fig. 3. Notice that the value of this parameter can vary depending on the STA being considered as the target STA. The average amount of time to transmit the TCP segment is represented by d_{tx}. This includes any MAC and physical layer overhead, MAC frame transmission time, all interframe spacings and the MAC layer acknowledgement. Just like the TCP segment, the TCP ACK also faces a similar journey back to the server. The terms u_{access} and u_{tx} are defined in a similar manner for the target STA.

For our analysis, we represent the WLAN (AP and STAs) as a virtual end-point consisting of two queues: a forward queue and a reverse queue. For now, let us assume that the non-speed test flows are non-existent and that only the speed test flow packets exist in the WLAN (we subsume the impact of non-speed test flow packets into the model parameters later). With this consideration, we can treat the virtual end point as a black box replacing the WLAN that runs a speed test. The socket level TCP client (not to be confused with the physical STA) runs on the virtual end point itself as shown

[2]A example cause of these delays is that due to cross traffic sharing a common queue on the backbone with the TCP segment.

[3]The exact rate adaptation policy is vendor implementation dependent.

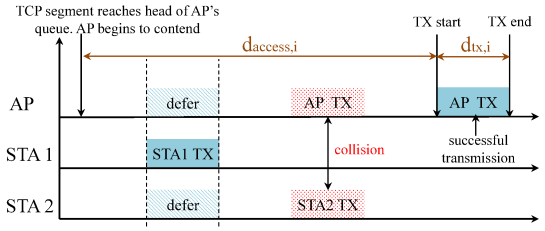


Fig. 3: Example timeline of a downlink transmission to depict $d_{\text{access,i}}$ and $d_{\text{tx,i}}$. $d_{\text{access,i}}$ and $d_{\text{tx,i}}$ denote the 'access' and 'tx' values respectively for the i^{th} downlink transmission. d_{access} and d_{tx} denote their mean values.

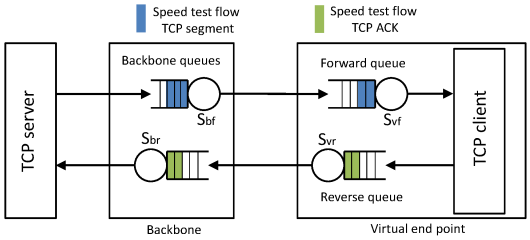


Fig. 4: WLAN representation as a virtual end point consisting of a forward and a reverse queue. The TCP client here refers to the socket level client and is not to be confused with the physical STA itself.

in Fig. 4. We can think of TCP segments and TCP ACKs as jobs circulating in the network. The service time of each job is a sum of its 'access' term and its 'tx' term. E.g., for jobs in the forward queue, the service time is a sum of d_{access} and d_{tx}. Since we account for the 'tx' term in the service time itself, the jobs themselves become indistinguishable. As we have not yet subsumed the effect of non-speed test flows, the throughput of the virtual end point is not the same as that of the target STA in our WLAN.

In our second step, we account for the impact of non-speed test flow packets on the throughput of this system by inflating the service times of each queue to account for the non-speed test flow packets. In essence, this inflation makes the effective speed of each server as seen by the speed test flow packet in the virtual end point system the same as that in the original system where some server time should have been consumed by non-speed test flow packets as well. Consequently, on the forward queue, the service time is inflated by V. However, since the target STA has no other uplink traffic while performing a speed test, the reverse queue service time requires no inflation. Similarly, we can subsume the impact of cross traffic on the backbone queues into their respective service times.

C. Throughput analysis

To analyze the throughput of the network shown in Fig. 4, we consider two cases. First we consider a case wherein TCP performs no ACK thinning. Consequently, in this case, each TCP segment received by the STA results in the generation of a TCP ACK. Next, we generalize this to account for the case of ACK thinning with an ACK thinning ratio of n. In this case, the client generates a TCP ACK following the receipt of every n^{th} TCP segment.

1) No TCP ACK thinning: Ignoring the initial transient stage during which TCP's window size grows, the speed test

flow will reach a steady state wherein TCP operates at W_{m}. Consequently, the number of packets that are contained in the speed test flow, which can either be TCP segments or TCP ACKs, remain constant and the system behaves as a closed queuing network with tandem servers and a constant number of jobs circulating inside it.

Based on the aforementioned notations, the mean service time for the forward and the reverse queue in the virtual end point (Fig. 4) is given by:

$$S_{vf} = d_{\text{access}} + d_{\text{tx}} + V \tag{1}$$

$$S_{vr} = u_{\text{access}} + u_{\text{tx}} \tag{2}$$

Let $S = S_{bf} + S_{br} + S_{vf} + S_{vr}$, $S_{max} = max(S_{bf}, S_{br}, S_{vf}, S_{vr})$ and θ denote the throughput in terms of jobs per second. It can be shown [10] that

$$\theta \leq min\left(\frac{W_{\text{m}}}{S}, \frac{1}{S_{max}}\right) \tag{3}$$

where $\frac{W_{\text{m}}}{S}$ is an asymptotic bound for small values of W_{m} and $\frac{1}{S_{max}}$ acts as an asymptotic bound for large values of W_{m}. The cases of small and large here are relative to a critical value W^*_{m} which is the point at which the asymptotes cross each other. Consequently,

$$W^*_{\text{m}} = \frac{S}{S_{max}} \tag{4}$$

To understand the physical relevance of the two components of Eq. (3), let us consider two extreme case scenarios. Let us assume that $W_{\text{m}} = 1$ which makes the number of jobs circulating in Fig. 4 the botteneck. The throughput, therefore, is given by $\frac{W_{\text{m}}}{S}$. On the other extreme, if W_{m} is sufficiently large (again large as compared to W^*_{m}) to not bottleneck the system, then the slowest queue acts as a bottleneck. In this case the slowest queue always remains busy and in accordance with the utilization law, $\theta = \frac{1}{S_{max}}$.

Recall that due to the server selection process, S_{br} and S_{bf} are not the bottleneck in the system. To understand the typical values that W^*_{m} can take, let us consider the critical point wherein $S_{br} = S_{bf} \sim max(S_{vf}, S_{vr})$. Substituting in Eq. (4), we will get $W^*_{\text{m}} = \frac{2*(S_{vf}+S_{vr})}{max(S_{vf},S_{vr})}$. The maximum value of W^*_{m} occurs when $S_{vf} = S_{vr}$ and thus $\max(W^*_{\text{m}}) = 4$. In practice, $W_{\text{m}} \gg 4$ and consequently, we can see that $\theta \leq \frac{1}{S_{max}}$ will act as a asymptotic bound on the values of θ. In fact, we find in our experimental evaluation that for a typical speed test, the values of W_{m} is extremely large as compared to 4 and θ will tend to the bound yielding

$$\theta \sim \frac{1}{S_{max}}. \tag{5}$$

2) TCP ACK thinning: Now, we extend the above to the more general case of TCP ACK thinning. For an ACK thinning ratio of n, we can view a maximum of only $\frac{W_{\text{m}}}{n}$ jobs circulating in the system and the remaining jobs can again be accounted for by further inflating the service times of each of the queues (just as for non-speed test flows). Consequently, when the wireless nodes transmit only one frame per transmission, the service times of both the forward and reverse queue in the virtual end point stretch by an amount equal to $(n-1)\times(d_{\text{access}} + d_{\text{tx}} + V)$ for the case of the download speed test. Here we inflate the service time of the reverse queue to account for the fact that the TCP ACK is not generated until the n^{th} TCP segment is received. The numerator of Eq. (5) should also be multiplied by n to compensate for the shrinking of the total number of TCP segments. For the upload speed test, the service times stretch by $(n-1)\times(u_{\text{access}} + u_{\text{tx}})$. However, when the nodes transmit multiple frames per transmission, such an inflation is not necessary since the STA receives multiple TCP segments in a single downlink transmission and there is no additional delay in the generation of a TCP ACK. These multiple frames may be transmitted using frame aggregation in single stream transmissions (e.g., SISO) or by using multi-stream transmissions (e.g., MIMO) or a combination of both frame aggregation and multi-stream transmissions. We emphasize that this is possible since typical ACK thinning ratios of TCP are much smaller than the number of frames that can be transmitted in a single transmission via the above mentioned policies under 802.11 [11], [12], [13], [14].

In summary, the throughput in bits/sec is given by

$$\theta_{\text{dl}} = \frac{\mathbb{E}[\text{TCP segment size}] \times F_{\text{AP}}}{max(S_{vf}, S_{vr})} \tag{6}$$

$$\theta_{\text{ul}} = \frac{\mathbb{E}[\text{TCP segment size}] \times F_{\text{STA}}}{max(S_{vf}, S_{vr})} \tag{7}$$

where we denote θ_{dl} and θ_{ul} as the download and upload TCP throughputs respectively. F_{AP} denotes the average number of frames transmitted by the AP in a single downlink transmission to the target STA. For the case of the upload speed test, we use F_{STA} instead.

Note that while calculating S_{vf} and S_{vr} for Eq. (6), d_{tx} is the average time to transmit F_{AP} number of TCP segments at the AP's data rate and u_{tx} is the average time to transmit F_{STA} number of TCP ACKs at the target STA's data rate. In Eq. (7), this is reversed since the target STA is now the one transmitting TCP segments and the AP is the one transmitting the TCP ACKs. S_{vf} and S_{vr} further vary depending on which STA is chosen as the target STA. Consequently, the AP has to estimate these two parameters with respect to the particular STA that is chosen as the target STA.

We remark that while the L2 edge TCP model needs to be supplemented with AP-side measurements, it is not restricted by a requirement for AP-side knowledge of inter-node connectivity or an assumption on network traffic characteristics. Next we show how the model parameters are estimated.

IV. Obtaining AP-side measurements

In this section, we show how the AP can measure all of the parameters required for the above model, thereby enabling a dynamic AP-side speed test estimate for each STA.

A. AP-side estimation problem

We observe that Eq. (6) and (7) are independent of S_{br} and S_{bf}. To estimate θ_{dl} and θ_{ul} at the AP, the key challenge is

computation of S_{vr}, as the remaining parameters are based on common AP side observables described in Sec II-C. Recall from Eq. (2) that S_{vr} is composed of u_{tx} and u_{access}. While the average uplink transmission time u_{tx} is known to the AP via per-STA metrics, the uplink access time u_{access} is known only at the STA side. Let $t_{\text{hq}}^{U,i}$ denote the time at which the i^{th} *uplink* packet reaches the head of the STA's queue, $t_{\text{ts}}^{U,i}$ denote the start time corresponding to the successful transmission of this packet and $t_{\text{te}}^{U,i}$ denote the end time of this packet transmission. By definition, $u_{\text{access}} = \mathbb{E}[(t_{\text{ts}}^{U,i} - t_{\text{hq}}^{U,i})]$. While the AP can observe $t_{\text{ts}}^{U,i}$ for any uplink transmission, $t_{\text{hq}}^{U,i}$ remains unknown. If the STA is assumed to be fully backlogged, the end time of the previous transmission can be approximated to be the time when the next packet reached the head of the queue. However, STA backlog is user activity dependent and is not known to the AP. As a result, the AP cannot estimate u_{access} by a simple observation of packets received on the uplink.

B. Snooped handshakes for estimation of uplink access time

Suppose that the client is performing a TCP download from a server (e.g., streaming a Netflix video). This can be any server on the internet with any backbone delay to the AP. The client will attempt to return a TCP ACK as fast as possible after reception of the corresponding TCP segment. This TCP ACK is "data" at layer 2. For now, consider a case where there are no other flows on the uplink from the target STA and no ACK thinning. Since the WLAN is the final hop for the TCP segment, upon reception of a TCP segment, *i.e.*, at the end of the AP's successful downlink transmission (denoted by $t_{\text{te}}^{D,i}$), the STA has the corresponding TCP ACK and begins to contend. Consequently, in this case, $t_{\text{hq}}^{U,i} = t_{\text{te}}^{D,i}$ and thus the AP will have inferred a parameter that is not directly observable. In essence, the delay incurred between the transmission of the segment to the reception of the TCP ACK enables the AP measure how long it takes the STA to successfully transmit after it starts to attempt. Thus, our general approach is to selectively sample TCP data-ACK handshakes from any TCP download performed by the target STA and use them to drive a measurement based prediction of θ_{dl} and θ_{ul}. We refer to such TCP flows as *snooped flows*.

This can be generalized under a flow hypothesis (*i.e.*, knowing that a given flow on the downlink is a TCP flow) by the following two cases.

ACK queuing. This case occurs when the target STA has other uplink flows whose packets get queued prior to the TCP ACK. Consequently, in such scenarios, $t_{\text{hq}}^{U,i} = t_{\text{te}}^{U,i-1}$. In such cases, we abuse the term $t_{\text{te}}^{U,i-1}$ to refer to the end time of transmission of the immediately preceding uplink packet.

ACK immediate. However, if the target STA has no other uplink flow, it begins to contend as soon as the TCP ACK is queued. Consequently, $t_{\text{hq}}^{U,i} = t_{\text{te}}^{D,i*n}$ where the superscript 'D' refers to a downlink transmission.

C. TCP flow inference

Because the layer four handshake is needed to estimate u_{access}, it is crucial to identify this handshake at the AP, which does not have layer four visibility. To this end, we employ IP addresses and size signatures as follows.

IP address signature. Due to the inherent bi-directionality of TCP, the source and destination addresses for TCP segments traversing on the forward path are swapped for the corresponding TCP ACKs on the reverse path. This key factor enables us to distinguish individual TCP flows and separate them from the remainder of the downlink and uplink traffic.

Packet size signature. Although the above signature enables identification of a bidirectional flow, it does not aid in spotting the forward and reverse paths distinctly. While the size of TCP segments on the forward path may fluctuate during the course of a download, the reverse path is characterized by small TCP ACKs whose size remains fixed during the entire duration of the flow. Typically a TCP ACK is 20 bytes long [15]. Having distinctly identified the forward and reverse paths, the AP can employ the u_{access} estimation process described in the previous sub-section.

V. Implementation and Experimental Methodology

In this section, we provide details of the virtual speed test enabled AP (VST AP), field trial details and our ground truth procurement methodology.

A. VST AP characterization

VST AP runs on a Linux operating system and is factory installed with 32 GB DDR4 SO-DIMM RAM, 2.4 GHz dual core CPU with a Gigabit LAN port. It is equipped with a Ralink RT3070 off-the-shelf WiFi chipset. The radio card supports IEEE 802.11b/g/n utilizing up to 40 MHz bandwidth and a peak PHY rate of 300 Mbps. To enable throughput estimation, we build APIs that enable the acquisition of a number of per packet statistics at the AP. Specifically, VST AP collects packet timestamps (available on a nanosecond granularity), source and destination IP addresses, frame sizes, and PHY rates using these APIs and feeds them into the L2 edge TCP model to estimate the throughput. As described earlier, the parameter estimation methodologies employ packet timestamps as a part of the computation process. While the absolute value of these timestamps can be impacted by system dependent offsets, their post-subtraction residue becomes negligible as they have a low second moment. The STAs associated with the VST AP are a mix of portable laptops running on either Windows or Linux OS whose network interface card supports 802.11b/g/n as well. While VST does not require collection of STA side statistics, for validation purposes, we also implement APIs and data collection capabilities for STAs associated with the VST AP to enable us to characterize the operating environment.

B. Field trials

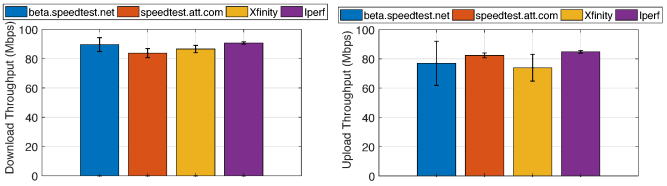
To study the estimation accuracy of virtual speed test, we deploy VST AP and STAs in two environments. The first deployment is in an office located in a 3 storied building on a university campus. In this deployment, the VST AP and

its associated STAs co-exist with a university administered enterprise network and 2 BSS deployed in nearby offices. We deploy a second network in an apartment in a 3 story residential building primarily consisting of 1 or 2 bedrooms per unit. These devices coexist with BSS from neighboring apartments. In each environment, the traffic generated by co-existing BSS is not controlled by us and is determined by user activity in those BSS. The VST AP deployed in the office performs measurements for a period of two days whereas the residential scenario measurements are carried out for a period of one week. During the entire duration of deployment, the VST AP observed a total of 113,047 snooped flows. These flows are generated by STAs associated with the VST AP which are instrumented to run web applications such as video streaming (via YouTube), music streaming (via Pandora), pdf downloads (via IEEE Xplore) and sending and receiving emails (via Gmail). The number of these applications running in parallel for each STA is also varied thereby generating cases of both single as well as mixed application traffic with a varying number of parallel running applications. The STAs use Mozilla Firefox web browser for performing these online activities. Aside from the applications mentioned above, some of the STAs have Dropbox installed on them which occasionally adds to the uplink traffic from these devices in addition to that generated by email activities.

It is important that virtual speed test is accurate in the presence of variation of the empirical L2 edge TCP model parameters which are affected by traffic characteristics, active STA count, MAC and PHY statistics, etc. Post processing traffic traces reveals that the application layer data size varied from a minimum of 20 bytes to a maximum of 1.4K bytes. Overall, the office and residential scenario covered a total of 36 and 49 different topologies respectively with a varying number of STAs associated with the VST AP in each topology and the VST AP made predictions for each associated STA. The total number of associated STAs in a topology varied between 1 and 6 in the office scenario and 1 and 7 in the residential scenario. The activity in the neighboring BSS varied in both the deployment. For instance, during approximately 30% of the flows in both the office and the residential scenario, there was no activity on the neighboring BSS. On the other hand, the maximum portion of a TCP flow's duration that a STA spent deferring to neighboring BSS was 38.8% for the residential scenario and 51.4% in the office scenario. Further details about the deployment can be found in our companion technical report available at [16].

C. Iperf for ground truth procurement

We compare the outcome of virtual speed test with that obtained from online speed test applications [1], [2], [3] as well as iperf. We focus on iperf for procuring ground truth as online tests are subject to an additional and uncontrolled source of variation due to server selection. Namely, recall that during the setup phase, the server selection process of these speed test applications tries to minimize the backbone delay between the server and the AP with the ideal case being a server in the same LAN as the AP. In practice, a server within the same LAN may not be available in the server pool probed by these applications. Consequently, results obtained from a non-local server may be affected by backbone load and delay.



(a) Download speed test (b) Upload speed test

Fig. 5: Iperf results as compared to exemplary online speed test applications.

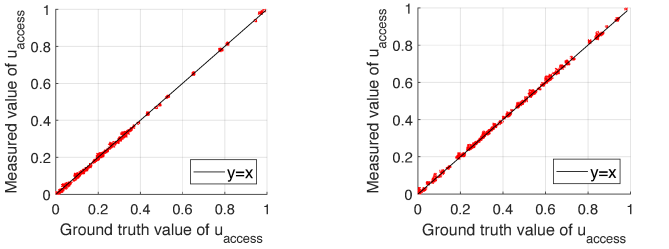
To explore this effect, we keep one 802.11n laptop associated with the AP and run multiple speed test applications 10 times with a gap of 10 mins between each run. We attempt to realize static WLAN conditions, *i.e.*, with minimal activity on co-existing BSS, no device or environmental mobility, etc., in which the outcome of speed test applications should remain stable. We compare these results with those obtained by running iperf from a local server to remove backbone and server selection effects. Fig. 5 shows the values obtained for upload and download throughput from exemplary online speed test applications as well as iperf. Compared to iperf, the speed test applications demonstrate throughput reductions and fluctuations due to backbone and server selection effects. On the other hand, the values obtained from iperf remain stable. Therefore, to obtain stable and more reliable ground truth values in our experiments, we use the iperf tool.

VI. Experimental Evaluation

In this section, we investigate the performance of virtual speed test in the two deployment scenarios. In our experiments, virtual speed test throughput estimates are obtained solely via AP measurements, i.e., by using AP-snooped TCP data - ACK handshakes from existing network traffic to estimate the parameters required by the L2 edge TCP model. Immediately following each throughput estimation, the ground truth is measured via iperf on instrumented clients. While network traffic is composed of both download and upload traffic, virtual speed test only leverages download flows for u_{access} estimation.

A. Uplink access time u_{access} estimation

Throughput and virtual service time are impacted by the uplink access time. Recall that this model parameter is not directly observable at the AP. Rather, the VST AP leverages snooped handshakes from TCP flows existing in the network to estimate uplink access time. In our deployment, the VST AP observes cases involving only single application traffic as well as those with mixed application traffic involving concurrent TCP flows with a mix of downloads and uploads. To correctly estimate the uplink access time, the VST AP needs to first identify TCP flows based on the address and size signatures and then accurately identify the occurrence of the ACK queuing and ACK immediate regimes. We first explore the efficacy of the u_{access} estimation method by comparing the



(a) Office Scenario (b) Residential Scenario

Fig. 6: Scatter plot of u_{access} measured and ground truth value in the office and residential deployments. In each sub-figure, both the axis are normalized with respect to the maximum ground truth value in that particular deployment.

estimates made by the VST AP using snooped flows with the ground truth values obtained from the STA-side.

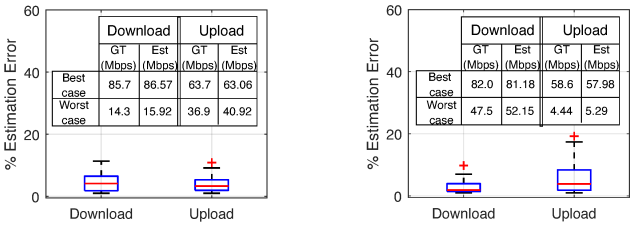
Fig. 6 shows a scatter plot of measured values of u_{access} vs. the ground truth value. For each deployment scenario, both of these values are normalized with respect to the maximum ground truth value in that particular scenario. Fig. 6 shows that as the ground truth values of u_{access} varies due to numerous factors such as background traffic characteristics, active STA count, etc., the points in the scatter plot continue to be closely located to the identity line. Fig. 6 also demonstrates the existence of homoscedasticity between estimates obtained by VST AP and the ground truth value of u_{access} obtained from the STA-side. In essence, this shows that as the ground truth value of u_{access} varies due to the numerous factors mentioned previously, the estimated value of u_{access} retains its accuracy.

B. Throughput estimation accuracy

The ultimate goal of virtual speed test is to estimate θ_{dl} and θ_{ul}. To evaluate throughput estimation accuracy, we calculate the percent estimation error between the ground truth values obtained from iperf and the estimates of virtual speed test as $\%error = \frac{|(\theta_{\text{gt}} - \theta_{\text{est}})| * 100}{\theta_{\text{gt}}}$ where θ_{gt} denotes the ground truth value of throughput and θ_{est} denotes the estimated value from virtual speed test. We remark that since the $\%error$ is calculated after weighing by θ_{gt}, it is sensitive to the value of θ_{gt}, *i.e.*, the same absolute error at a smaller ground truth value is bound to yield higher estimation error.

Fig. 7 summarizes the percent throughput estimation error statistics in both the office and residential deployments. Overall virtual speed test shows a good match against ground truth values with a mean percentage error of 4.09% and 4.3% for upload and download speeds respectively in the office scenario. In the residential scenario, these values are 5.51% and 2.9% respectively.

Virtual speed test estimates the target STA's throughput by collecting measurements from its TCP traffic. Therefore, a key factor that determines the estimation accuracy of virtual speed test is the number of measurements that the VST AP can obtain from the target STA's TCP traffic which in turn depends on the target STA's online activity. On one extreme, a large number of observations obtained from either a single large file download or a series of consecutive small file download results in refined parameter estimation. We experimentally investigate the observation set size required to avoid a deterioration in estimation

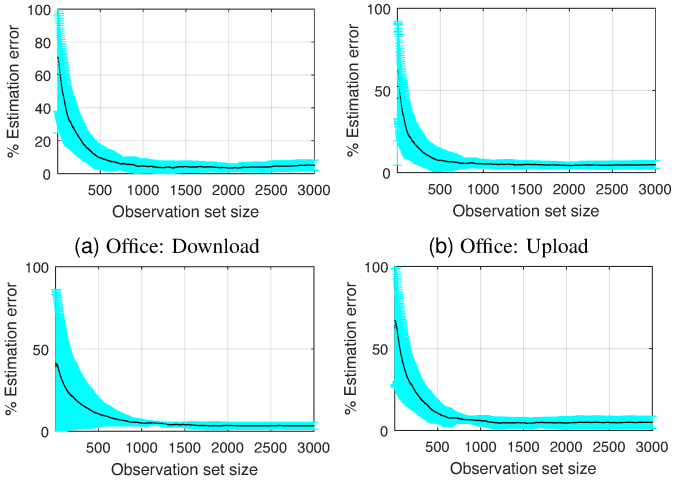


	Download		Upload	
	GT (Mbps)	Est (Mbps)	GT (Mbps)	Est (Mbps)
Best case	85.7	86.57	63.7	63.06
Worst case	14.3	15.92	36.9	40.92

	Download		Upload	
	GT (Mbps)	Est (Mbps)	GT (Mbps)	Est (Mbps)
Best case	82.0	81.18	58.6	57.98
Worst case	47.5	52.15	4.44	5.29

(a) Office Scenario (b) Residential Scenario

Fig. 7: Percent throughput estimation error statistics in the office and residential deployments. In the table, GT denotes the ground truth value and Est denotes the throughput estimated by virtual speed test.



(a) Office: Download (b) Office: Upload

(c) Residential: Download (d) Residential: Upload

Fig. 8: Throughput estimation error as a function of the observation set size available at the AP

accuracy as follows. Since the backbone delay between the snooped flow's server and the VST AP is inconsequential, we initiate a single TCP flow from a local server to the target STA and treat it as a snooped flow. We control the observation set size available at the VST AP by changing the download file size and computing the percent throughput estimation error as before.

Fig. 8 depicts the mean estimation error as a function of the size of the observation set available at the VST AP. The mean estimation error demonstrates an exponential decrease with an increasing set size. When the sample set size is on the order of a few 10s of samples, the estimation error is as high as 70%. The variance in this case is also very high. This is due to the fact that the number of measurement samples are insufficient to accurately characterize the model parameters. With an increasing sample set size, the precision in the model estimation increases as expected. Fig. 8 reveals that even when the observation set contains a few 1000s of samples (which could easily be obtained from observing the download of a research paper from IEEE Xplore), the mean estimation error of virtual speed test is under 5% in both the deployments.[4]

VII. Related Work

Analytical Models. Prior analytical models can predict throughput while incorporating various MAC and PHY layer aspects of Wi-Fi [17], [18]. Unfortunately, the models are

[4]ns 3 simulation details and results are in our companion tech report [16].

not applicable or extensible to our scenario as they require AP-side knowledge of network topology, interfering nodes including those from neighboring BSS, their traffic patterns, PHY capabilities, data rates, etc. Obtaining this information requires STA-side co-operation and regular reporting. On the other hand, virtual speed test enables throughput estimation with zero STA-side co-operation and no reporting.

Active measurements. Active probing techniques such as [19], [20] involve usage of probing packets to estimate bandwidth. However, they impose additional traffic load on the network that can disrupt user traffic or drain the battery of mobile devices. Tools such as [21] require client-side software to perform network analysis. On the other hand, virtual speed test performs passive measurement based estimation and hence does not impose any additional traffic load on the network or require any specialized client-side software.

Passive measurements. Careful deployment of sniffers can be used to make passive observations to collect traffic traces of various users to estimate throughput [22],[23]. Likewise, collection of information from co-existing BSS can also enable the AP to estimate throughput of its associated STAs [24]. However, such methods either require installation and maintenance of additional hardware or co-operation from neighboring BSS for data collection. In contrast, virtual speed test requires no additional infrastructure or any cooperation from among co-existing APs.

Training via ground truth measurements. In this approach, followed by [9], [25], [26], network clients store empirical throughput of all TCP sessions and report them to the AP to build a database of TCP throughputs. This coupled with AP-side records of wireless conditions during the TCP session (e.g., the session's MCS, busy air time, and collision rate) enable the AP to predict throughput by correlating the current conditions with historical averages corresponding to similar conditions. However, this requires client-side reporting to obtain ground truth as network and traffic conditions change, a requirement that is not allowed in our problem formulation.

TCP flow analysis. TCP flow analysis has been leveraged to understand IP and TCP statistics such as segment reordering, duplication, etc. [27], identification of malicious attacks [28] and for P2P Botnet detection [29]. In contrast, we utilize TCP flow dynamics to measure L2 parameters to facilitate upload and download throughput estimation for WLANs.

VIII. Conclusions

We presented virtual speed test - a measurement based framework that enables an AP to continuously estimate TCP speed test results for any of its associated STAs without any end-user co-operation, with no additional traffic load on the network and solely based on passively obtained AP-side observables. We deploy a VST enabled AP in a university office and in a residential apartment characterized by a variety of operating conditions including the presence of multiple co-existing BSSs, link diversity in terms of signal propagation and supported PHY rates and variation in traffic characteristics and the number of associated clients. Overall, virtual speed test exhibits high accuracy with mean estimation errors below 6%.

IX. Acknowledgements

This research was supported by Cisco and by NSF grants CNS-1801857 and CNS-1642929.

References

[1] Ookla Speedtest. http://www.speedtest.net/. Accessed: 2018-05-27.
[2] AT & T Internet Speed Test. http://speedtest.att.com/speedtest/. Accessed: 2018-05-27.
[3] Xfinity Speed Test. http://speedtest.xfinity.com/. Accessed: 2018-05-27.
[4] D. Murray, T. Koziniec, S. Zander, M. Dixon, and P. Koutsakis. An analysis of changing enterprise network traffic characteristics. In *Proc. of IEEE APCC*, 2017.
[5] Ookla SpeedTest. How does the Begin Test button select a server?, 2012.
[6] Ookla SpeedTest. How does the test itself work? How is the result calculated?, 2012.
[7] O. Goga and R. Teixeira. Speed measurements of residential internet access. In *Proc. of PAM*, 2012.
[8] E. Altman, D. Barman, B. Tuffin, and M. Vojnovic. Parallel TCP Sockets: Simple Model, Throughput and Validation. In *Proc. of IEEE INFOCOM*, 2006.
[9] A. Patro, S. Govindan, and S. Banerjee. Observing Home Wireless Experience Through WiFi APs. In *Proc. of ACM MobiCom*, 2013.
[10] M. Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.
[11] D. Murray and T. Koziniec. The state of enterprise network traffic in 2012. In *Proc. of APCC*, 2012.
[12] R. Braden. RFC-1122: Requirements for internet hosts. *Request for Comments*, pages 356–363, 1989.
[13] IEEE Std. 802.11ac-2013. Enhancements for Very High Throughput for Operation in Bands Below 6 GHz, 2013.
[14] IEEE Std. 802.11n-2009. Enhancements for Higher Throughput, 2009.
[15] P. Jon. Transmission control protocol–darpa internet program protocol specification. Technical report, RFC-793, DARPA, 1981.
[16] Companion technical report, available at. https://www.dropbox.com/s/ruf3qh60bxdwhls/tech_report.pdf?dl=0.
[17] P. Nayak, M. Garetto, and E. W. Knightly. Multi-user Downlink with Single-User Uplink can Starve TCP. In *Proc. of IEEE INFOCOM*, 2017.
[18] D. Miorandi, A. Kherani, and E. Altman. A queueing model for HTTP traffic over IEEE 802.11 WLANs. *Computer Networks*, 2006.
[19] K. Lakshminarayanan, V. N Padmanabhan, and J. Padhye. Bandwidth estimation in broadband access networks. In *Proc. of ACM SIGCOMM*, 2004.
[20] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi. Capprobe: A simple and accurate capacity estimation technique. In *Proc. of ACM SIGCOMM*, 2004.
[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] L. DiCioccio, R. Teixeira, and C. Rosenberg. Impact of Home Networks on End-to-end Performance: Controlled Experiments. In *Proc. of ACM HomeNets*, 2010.
[23] Y. Cheng, J. Bellardo, P. Benkö, A. Snoeren, G. Voelker, and S. Savage. Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis. In *Proc. of ACM SIGCOMM*, 2006.
[24] V. Shrivastava, S. Rayanchu, S. Banerjee, and K. Papagiannaki. Pie in the sky: Online passive interference estimation for enterprise wlans. In *Proc. of NSDI*, 2011.
[25] C. Rattaro and P. Belzarena. Throughput prediction in wireless networks using statistical learning. In *Proc. of LAWDN*, 2010.
[26] M. Mirza, K. Springborn, S. Banerjee, P. Barford, M. Blodgett, and X. Zhu. On the accuracy of TCP throughput prediction for opportunistic wireless networks. In *Proc. of IEEE SECON*, 2009.
[27] M. Mellia, A. Carpani, and R. Cigno. Tstat: TCP statistic and analysis tool. In *Proc. of Springer QoS-IP*, 2003.
[28] Y. Chen and K. Hwang. TCP flow analysis for defense against shrew DDoS attacks. In *Proc. of IEEE ICC*, 2007.
[29] L. Zhou, Z. Li, and B. Liu. P2P traffic identification by TCP flow analysis. In *Proc. of IWNAS*, 2006.