Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2013, Article ID 752363, 12 pages
http://dx.doi.org/10.1155/2013/752363

Research Article

Optimizing TCP Performance in Multi-AP Residential Broadband Connections via Minislot Access

Domenico Giustiniano,[1] Eduard Goma,[2] Alberto Lopez Toledo,[2] and George Athanasiou[3]

[1] *Communication System Group, Gloriastrasse 35, 8092 Zurich, Switzerland*
[2] *Telefónica I+D, S.A.U., Pl. Ernest Lluch i Martin 5, Barcelona, Spain*
[3] *Automatic Control Department, Electrical Engineering School, ACCESS Linnaeus Center, KTH Royal Institute of Technology, Osquldas väg 10, 100-44 Stockholm, Sweden*

Correspondence should be addressed to George Athanasiou; georgioa@kth.se

Received 10 December 2012; Accepted 22 March 2013

Academic Editor: Juan Reig

The high bandwidth demand of Internet applications has recently driven the need of increasing the residential download speed. A practical solution to the problem is to aggregate the bandwidth of 802.11 access points (APs) backhauls in range. Since 802.11 devices are usually single radio, the communication to APs on different radio channels requires a time-division multiple access (TDMA) policy at the client station. With an in-depth experimental analysis and a customized 802.11 driver, in this paper, we show that the usage of multi-AP TDMA policy may cause degradation of the TCP throughput and an underutilization of the AP backhauls. We then introduce a simple analytical model that accurately predicts the TCP round-trip time (RTT) with a multi-AP TDMA policy and propose a resource allocation algorithm to reduce the observed TCP RTT with a very low computational cost. Our proposed scheme runs locally at the client station and improves the aggregate throughput up to 1.5 times compared to state-of-the-art allocations. We finally show that the throughput achieved by our algorithm is very close to the theoretical upper bound in key simulation scenarios.

1. Introduction

Asymmetric digital subscriber line (ADSL) has become the "de-facto" standard for residential broadband access to the Internet. In addition, the density of ADSL deployments with 802.11 wireless local area network (WLAN) connectivity tends to be high, especially in urban areas [1]. The interplay between these two technologies introduces interesting technical challenges and opportunities that can be exploited. First, WLAN access rates are typically an order of magnitude higher than the bottleneck of the end-to-end path, which is either the ADSL [2] or the backbone [3]. Second, the set of ADSL links in the neighborhood are generally underutilized [2]. As a consequence, there is a potential to bundle the access points (APs) backhaul bandwidth via 802.11 connections. However, (i) APs usually operate on independent radio channel, and (ii) users typically connect to these APs with single-radio commodity 802.11 cards. Because a single-radio card cannot simultaneously connect to more than one AP, it has been proposed to rely on the standard 802.11 power saving (PS) mode to implement a time-division multiple access (TDMA) policy. Stations spend enough time to either collect all the bandwidth from each AP [4] or to provide a fair sharing of the aggregated resources [5] by sequentially cycling through the APs in a round-robin fashion [6].

Unfortunately, multi-AP TDMA policies hurt the throughput performance of single TCP flows by increasing their round-trip time (RTT). To illustrate this effect, we consider the scenario in Figure 1, where one station is connected to N APs. We focus on the time that the station spends connected to one of the APs, say AP_1. When the station is connected to AP_1, it starts receiving the buffered TCP data packets. While connected, the station normally receives TCP data and replies with TCP ACKs. These TCP ACKs will trigger the transmission of new TCP data from the sender. Because of the end-to-end wired delay, these TCP

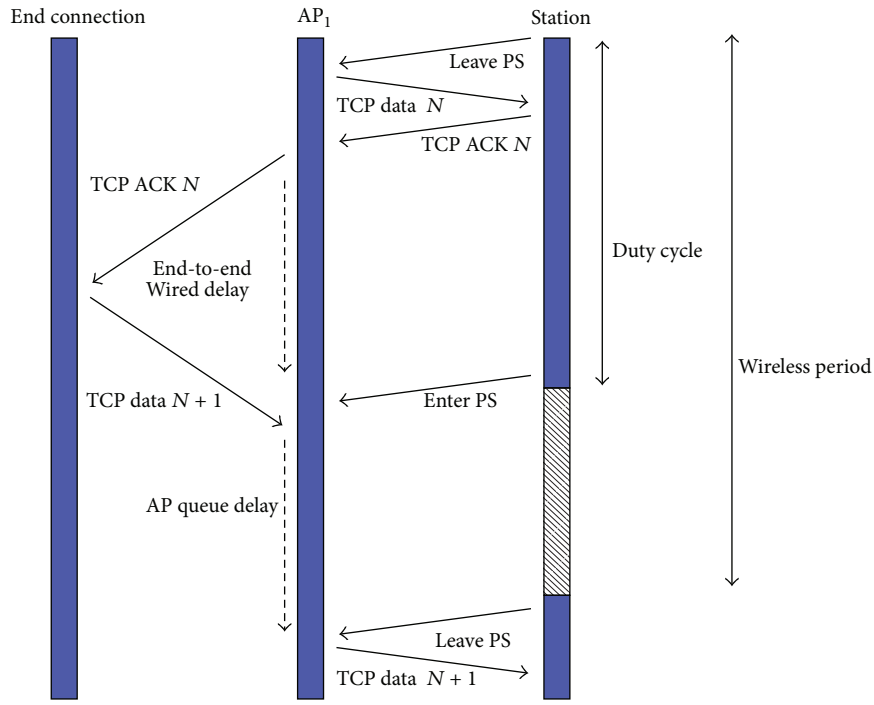


Figure 1: Relation between TCP congestion control and time-division access to multiple APs.

data may arrive to AP_1 right after the station has already moved to the next AP. These packets will be buffered by AP_1 until the station connects again to it. The result is that the time duration of the connectivity and nonconnectivity periods may result in TCP flows observing an RTT artificially larger than the actual end-to-end wired delay.

Motivated by the aforementioned problem, the main contributions of this paper are as follows:

(1) an in-depth analysis of the effect of multi-AP TDMA on TCP flows by performing numerous experiments using our prototype station that connects to multiple Aps,

(2) an analytical model that correlates the TCP congestion control with the multi-AP TDMA policy,

(3) a cost-efficient resource allocation algorithm, named *min-max disconnection time*, that distributes the time spent by the station to the APs in minislots (part of the TDMA slot) to minimize the total disconnection time from the APs.

We evaluate our model and show that it accurately fits the experimental results. We further study the algorithm via extensive simulations, and we show that it performs very close to the theoretical upper bound. The rest of the paper is organized as follows. Section 2 reviews related work, and Section 3 presents our prototype implementation of multi-AP TDMA used in the experimental tests. Then, Section 4 investigates the performance degradation of TCP on the multi-AP TDMA scenario, and it introduces an accurate analytical model. Section 5 validates the analytical model via both experiments and simulations. The resource allocation algorithm is introduced and validated in Section 6. Finally, Section 7 concludes the paper.

2. Related Work

The need for 802.11 resource allocation schemes has been extensively studied in the literature [7–9]. Many of the proposed schemes rely on either nonstandard compliant features [10], or completely develop an entire new MAC protocol [11]. Both strategies may be undesirable, and so we avoid them. Given that the resource allocation scheme that more closely relates to ours is in [12] that studied the problem of absence of application-specific 802.11 resource allocation schemes. As a solution, they designed and implemented an overlay MAC layer (OML) to divide the time into slots of equal size. Then, they used a distributed algorithm to allocate the slots across the competing nodes, where each competing node receives a number of slots proportional to its weight function. However, the authors let as an open issue the understanding of the increased delay for TCP flows in the presence of the slotted mechanism [12].

Although overlay solutions are easy to be implemented, they are often suboptimal and difficult to scale because of the overlapping and duplication of similar functionalities at different layers (e.g., in the driver and in the card firmware). The VirtualWiFi project [6] proposed an architecture that abstracts a single 802.11 WLAN card to appear as multiple virtual clients to the user. Each client instance adopts standard PHY/MAC protocols, but it can be separately

Table 1: Main variables used.

AP_i	ith AP
N	Number of AP backhauls
T (ms)	Wireless period
$VSTA_i$	ith virtual station, associated to AP_i
d_i	End-to-end wired delay
f_i (≤ 1)	Duty cycle for the ith virtual station
g_i (≥ 1)	Number of slots for the ith virtual station
G	Total number of slots
C_j	Disconnection cost for the jth slot
SlotTime	Minimum slot size

configured at the driver level. An interesting application was the idea of connecting to multiple APs through a single radio interface. The authors rely on the 802.11 power save (PS) mode feature to switch among different 802.11 WLAN nodes in a time-division fashion. A station can inform the current 802.11 WLAN node that it is going into PS mode—so that it can buffer packets directed to it—and switch the radio frequency to other 802.11 WLAN nodes, only to come back to the original node before the PS period expires.

Based on the above PS mechanism, FatVAP [4] studied the problem of ADSL bandwidth aggregation via wireless connectivity. The authors introduced a local scheduler that computes the percentage of time to dedicate to each AP in order to maximize the aggregate throughput at each client station. The solution leverages on the fact that the high speed wireless card at the station needs to be connected to each AP for a short period of time in order to collect all the pending data. THEMIS [5] reformulated the problem considering that gross unfairness would be generated in the above setting. Their approach achieved weighted proportional fairness, and they experimentally validated it with a multistory building and real ADSLs, showing that it outperformed previous solutions. However, both FatVAP and THEMIS do not explore TCP latency-related problems for single TCP flows. They essentially limited the analysis to scenarios with sufficient number of TCP flows, such that the ADSL bandwidth is saturated, or with short-lived TCP flows, where the congestion control phase does not trigger. Finally, Juggler [13] proposed an architecture similar to one in [4], and it focused on the support of a seamless handoff between WLAN APs.

3. Connecting to Multiple APs with Off-the-Shelf Hardware

In this section, we briefly describe WiSwitcher [14], the experimental 802.11 station that we have implemented, and it will be used in the rest of this paper as the basis for the experimental tests.

In WiSwitcher, the wireless driver of the single radio interface is *virtualized*; that is, it appears as independent virtual stations ($VSTA_i$) associated to their respective access points AP_i. Each $VSTA_i$ connects to Internet via AP_i, regardless of the AP_i radio frequency. Let us consider the scenario with one station in Figure 2(a), connected to three APs. In this example, WiSwitcher configures 3 virtual stations: $VSTA_1$, $VSTA_2$, and $VSTA_3$. Each of these VSTAs connects to one ADSL via its AP in range.

As we can see in Figure 2(b), WiSwitcher assigns the control of the card to a $VSTA_i$ for a given percentage of time, called *duty cycle* f_i (with $\sum_i f_i = 1$). During this time, $VSTA_i$ transmits and receives frames using the AP_i backhaul, while the other VSTAs (and the corresponding APs) can only buffer packets. We denote *wireless period* T as the amount of time to cycle through all the VSTAs. A summary of the main variables used in this section, and the rest of the paper is given in Table 1.

3.1. MAC Protocol. WiSwitcher manages the multiple backhaul connections relying on the 802.11 PS mechanism. Particularly, referring to the example in Figure 2(b); we will notice the following.

(1) During the reserved duty cycle, $VSTA_1$ transmits and receives data according to the 802.11 distributed coordination function (DCF) protocol. The other VSTAs are dormant in PS mode, and hence they (and the corresponding APs) can only buffer packets.

(2) When the duty cycle expires, $VSTA_1$ sends a frame to inform AP_1 that is going to PS mode and waits for its MAC ACK. According to the 802.11 protocol, AP_1 starts to buffer the packets directed to it.

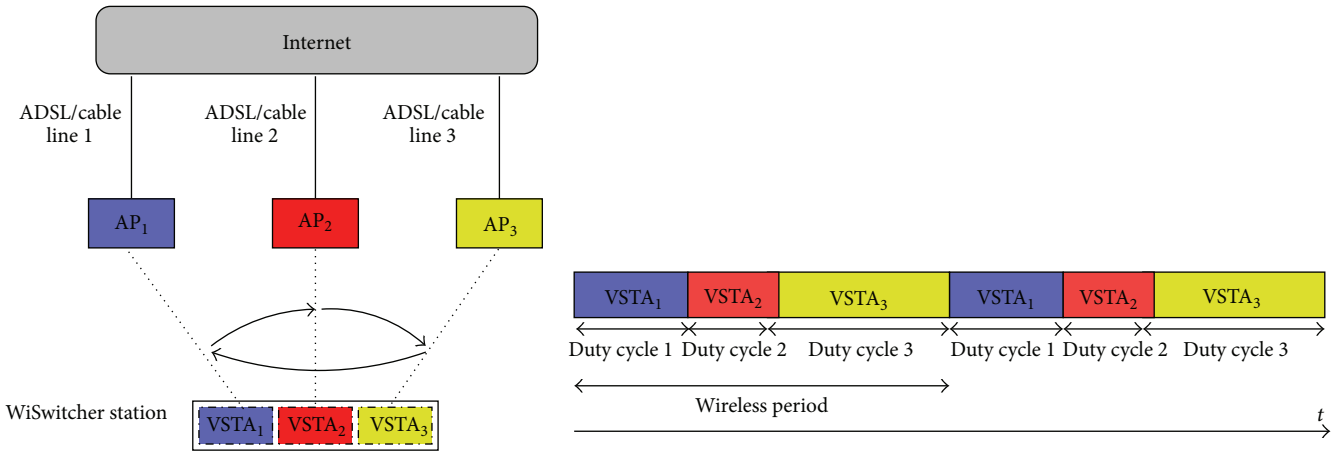
(3) WiSwitcher assigns the control of the card to $VSTA_2$ and switches to the AP_2 radio frequency.

(4) $VSTA_2$ sends a frame to announce that it can send and receive traffic and it waits for its MAC ACK.

(5) The process continues until the station has cycled through all the VSTAs (a wireless period T).

In the implementation, we incur in a channel-switching cost—that is, the time where WiSwitcher cannot send and receive any traffic—of 1.2 ms for uplink traffic and 1.5 ms for downlink traffic. This cost is less than half of the one obtained in the time-division implementation given in [4, 13]. This result has been achieved using 802.11 standard-compliant solutions, such as the MAC virtual queue per AP, and an efficient management of a hardware buffer size of one (1) data packet. The bulk of the cost of WiSwitcher is caused by the hardware operation delay, which is in the order of 800 μsec in our Atheros chipset-based cards. This cost is hardware dependent, and in other chipset implementations is reduced to 200–500 μsec [13].

Furthermore, the implementation achieves a fine-grained timing at MAC/PHY level, thus avoiding any additional variance of the delay in the packet transmission, and it considers an independent instance of the rate selection algorithm for each VSTA. This allows us to connect to APs with different link qualities. The reader is referred to [14] for an in-depth description of the MAC implementation.



(a) Topology

(b) Relation between duty cycle and wireless period

Figure 2: Time division access to multiple Aps.

3.2. Network Layer Functionalities. At the network layer, three functionalities are needed.

Scheduler. It calculates the percentage of time (duty cycle) to spend on each AP in order to maximize some utility function. In this work, the duty cycles are fixed via user-space commands.

Load Balancer. It assigns the new TCP flow from the upper layers to the different VSTAs so that the total load received from each AP maintains the proportions indicated by the scheduler. Since the load balancer is not the main objective of this work, we use the same per-flow basis scheduler presented in [4].

Reverse-NAT. In order to guarantee transparency to higher layers, we implement a reverse-network address translation (NAT) module with two functions: (i) ensure that the packets leave the host with the correct source IP address (i.e., the one corresponding to the outgoing VSTA, as assigned by the AP) and (ii) that the incoming packets are presented to the OS with the expected IP address, a dummy IP address in our implementation. Reverse-NAT modules were also present in [4, 13].

4. TCP over Multi-APs TDMA

In this section, we first show an experimental test that enlightens the correlation between the end-to-end throughput and the delay added by the TDMA policy, and then we introduce an analytical model that characterizes the TCP RTT for a station connected to multiple APs. The importance of the model is that it not only gives insights into the problem, but it will be also used to validate the resource allocation algorithm later defined in the paper.

4.1. Example of TCP Throughput over Multi-AP TDMA Policy. We experimentally test the side effect of the multi-AP TDMA policy on a (long-lived) TCP session. Figure 3 shows the average TCP throughput obtained by a station spending 50% of its time (i.e., duty cycle of $f_1 = 0.5$) to one AP, as a function of the end-to-end wired delay d_1. For the test, we consider a wireless period $T = 100$ ms, which gives a connection of $f_1 T = 50$ ms on AP_1. Each point is the average TCP throughput obtained over 5 independent tests of 300 s. In the tests, the average experimental congestion signal rate we measured is of ≈0.4%. These losses are likely generated on the wireless link (as a result of using SACK, TCP congestion signals are mainly caused by fast retransmissions due to duplicated ACKs because its goal is to avoid retransmission timeouts. For all SACK-based TCPs, multiple losses within one RTT are treated as a single congestion signal. In this paper, we use congestion signal and packet loss interchangeably and always refer to losses at TCP layer).

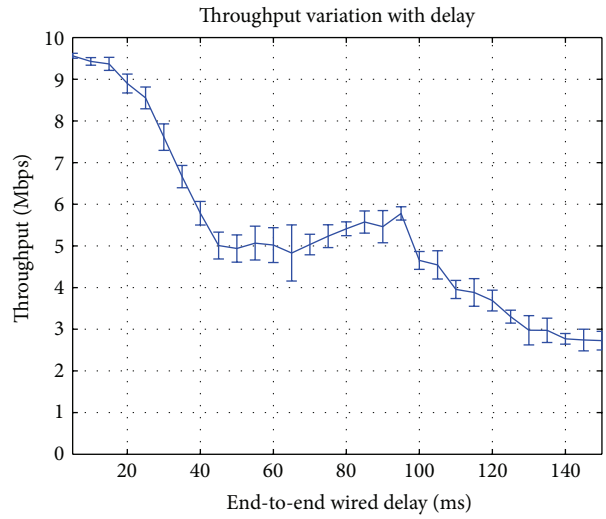


Figure 3: Experimental throughput connected 50% of time to one AP.

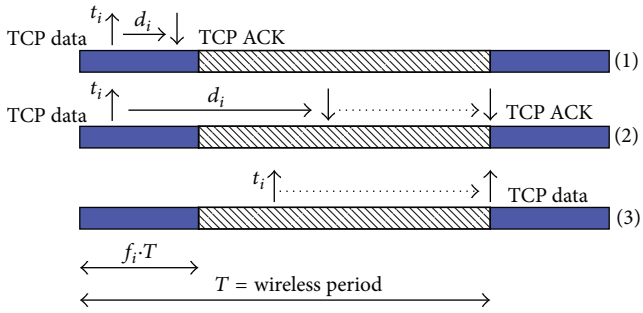


Figure 4: Model of the relation between TCP congestion control and duty cycle.

We can see from Figure 3 that the station gets a similar throughput for both $d_1 = 50$ ms and $d_1 = 100$ ms. This is caused by the similar RTT observed for 50 and 100 ms. The result is that the TCP data arriving at the AP with 50 ms of end-to-end wired delay have to wait for an extra-buffering time at the AP, due to the disconnection of $(1 - f_1)T = 50$ ms. This reduces the throughput observed by the TCP flow for $d_1 = 50$ ms. More exactly, there are small valleys in the throughput: a disconnection of 50 ms increases the buffer size of the AP. By further increasing the end-to-end wired delay from 50 to 100 ms, there is a higher and higher probability that downlink packets will arrive at the AP when the station is connected to it. This will reduce the buffering time at the AP (and thus the observed RTT) that will deliver the packets in a shorter time, with a slight increase of the throughput (note that there are slight variations in the packet losses observed in the experimental tests, which translate in variations (represented by error-bars in the figure) of the average throughput observed between different experiments).

4.2. Modeling the TCP RTT over Multi-AP TDMA. We can model the dependency of the TCP RTT on the end-to-end delay and the duty cycle by observing all the possible cases in which TDMA affects the observed RTT. In what follows, we consider the uplink case (e.g., the VSTA is sending data to a remote server), but it is straightforward to see that the RTT computations are symmetric for both the uplink and downlink cases. We distinguish three conditions.

(1) We consider the case of Figure 4(1) in which the station sends the TCP data at time t_i, during its duty cycle. Also, we assume that the end-to-end delay d_i is such that the TCP ACK arrives from the TCP server before the station disconnects from the AP. In that case, we see that the observed RTT from TCP is d_i.

(2) Next, we consider the case of Figure 4(2), where the station sends the TCP data at time t_i, during its active period, but the end-to-end delay d_i is such that the TCP ACK arrives from the server during the time reserved to other VSTAs. The AP_i will buffer the packet in its queue until the station reconnects again at time 0 of the next wireless period. In this case, the observed RTT for the TCP packets is $T - t_i$, where T is the wireless period. Note that, as long as $(1 - f_i)T > (d_i \mod T)$, there will be always some packet that will wait in the AP_2 downlink buffer because of the disconnection period.

(3) Finally, we consider the case of Figure 4(3), wherein the TCP data is buffered at the station at time t_i, during the sleeping period in the AP. However, we experimentally verified by monitoring the AP queues that case (3) does not occur in the TCP steady state, and no new TCP data is buffered during the sleeping period. The reason is that in the TCP steady state case, new TCP data can only be sent when a TCP ACK is received from the server. But as we have seen, the TCP ACKs can only arrive to the station when it is connected to the AP.

Finally, in order to take into account that the station takes some time for processing and transmitting the TCP ACKs, as verified experimentally, we consider that (i) the TCP ACKs arrive exponentially distributed over the duty cycle because there is a high probability that some TCP ACK is waiting in the AP_i downlink buffer when the station reconnects to it (in the beginning of a duty cycle); (ii) one TCP DATA is sent right after the reception of a TCP ACK. This assumption considers the case when, in average and steady state, throughput does not either increase or decrease; (iii) when $f_i T$ is very small, some of the TCP data scheduled during the connection period will inevitably be sent at the next connectivity period, due to buffering delay. Based on these assumptions, we statistically calculate the RTT distribution, considering as input the wireless period T, the duty cycle f_i and the end-to-end delay d_i. Despite the simplicity of the model, Section 5.1 will show that the model matches the experimental results.

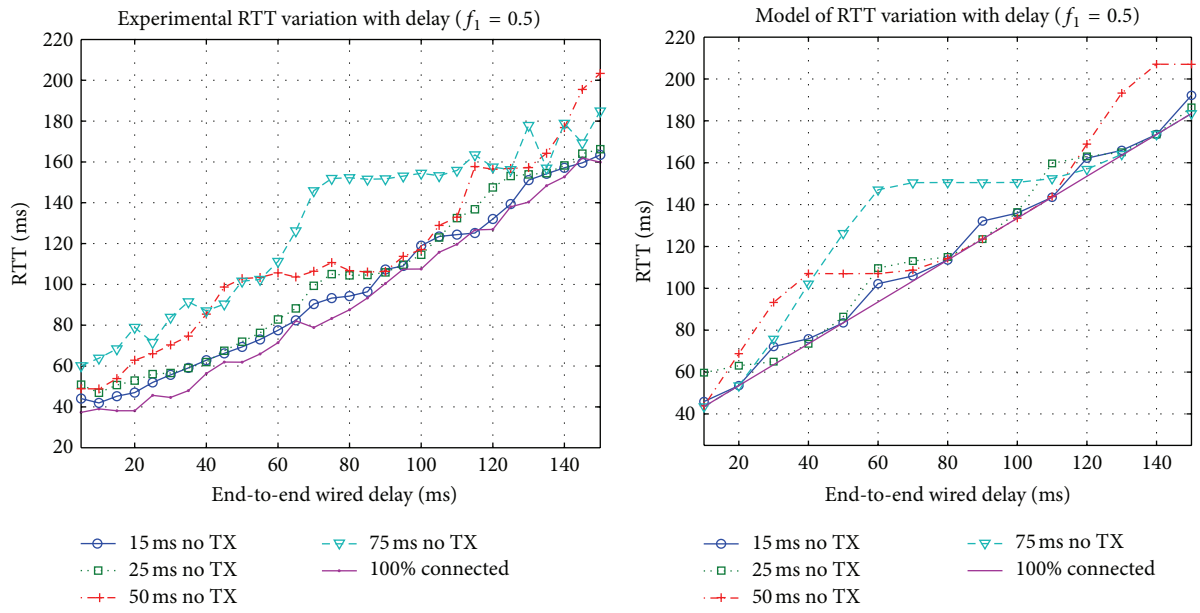
Mapping the Modeled TCP RTT to Throughput. Although a wide variety of TCP algorithms are used on the Internet, the current most popular implementation is TCP Reno [15]. Then, in order to map the RTT given by the model to throughput, we use the Mathis TCP model [16], which is intended to predict TCP end-to-end throughput as $BW \leq (\text{MSS}/\text{RTT}) \cdot (1/\sqrt{p})$, where RTT is the round-trip time observed by the station, MSS is the TCP maximum segment size, and p is the packet loss rate (this model applies to long lived connections over nearly all implementations of TCP Reno with SACK TCP. Note that in order to use this model, the packet loss rate should be smaller than 2%, condition verified in all the experimental tests in this paper).

5. Evaluation

In this section:

(i) we validate the accuracy of the TCP RTT model presented in the previous section, comparing it with experimental results;

(ii) we show that long disconnection time severely affects the TCP throughput;



(a) Experimental RTT (b) Analytical RTT

FIGURE 5: Downlink RTT for $f = 0.5$ to one AP. The station is not connected to the AP for a time equal to 0, 15, 25, 50, and 75 ms, respectively. We see that the analytical model accurately predicts the measured TCP RTT.

(iii) we demonstrate that for any duty cycle, the best strategy is to reduce the wireless period T as much as possible;

(iv) we show that the selection of the wireless period T must be done based on the smallest duty cycle $\min_i f_i$ of the station.

In what follows, we discuss the details of the experimental and simulation setup.

Experimental Setup. In each controlled test, we use laptops with Atheros-based chipsets running WiSwitcher as described in Section 3 and off-the-shelf APs (Linksys) with DD-WRT v24sp1 firmware. On the wireless station, automatic rate selection, wireless multimedia extensions, and the RTS/CTS mechanism are disabled. In the experimental tests of this paper, the 802.11 PHY rate is fixed to 54 Mbps. Other tests were performed in other configurations (e.g., with automatic rate selection enabled) and showed similar results to the ones presented in this paper. Any no-802.11 standard compliant features at the MAC level were also disabled. Our station uses a hardware queue with best effort parameters.

For the transport layer, we use a Linux standard TCP Reno with SACK and delayed ACK options enabled. TCP parameters are monitored using a modified version of the TCP probe kernel module and the kernel patch Web100. For each test, we establish one TCP connection over an AP backhaul (note that the link utilization can increase establishing more than one TCP connection over each AP [14], which is out of the scope of this paper), and we collect statistics using the *iperf* tool. Regarding the wired connections, we emulate the AP backhaul links through the *tc* Linux traffic shaper, varying the delay using the *netem* tool.

For each experimental test, we establish one TCP Reno connection over each AP, ran 5 independent tests of 300 secs and plot the average values obtained. To achieve independent tests, the station is configured so that the TCP metrics are reset after each test.

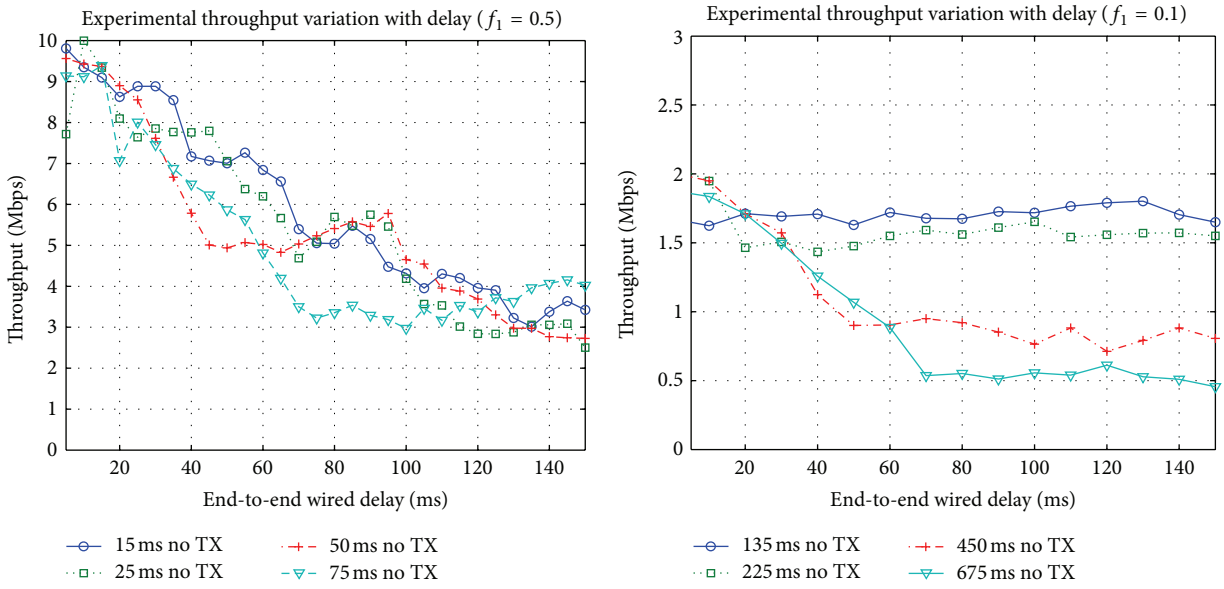
Simulation Setup. The simulations are performed using the model described in Section 4.2, implemented in MATLAB using as input the experimental values of MSS and TCP congestion signals rate per packet for the Mathis formula defined in Section 4.2 [16].

5.1. TCP RTT Model Validation. In this section, we compare the RTT values achieved experimentally with the ones of the model. For brevity, we only show one scenario, but similar finding has been achieved with several other setting (e.g., with different values of duty cycle).

Figure 5(a) shows the average RTT values obtained experimentally as a function of the end-to-end wired delay, when the WiSwitcher station spends 50% of its time connected to an AP, and it observes a disconnection time of 0, 15, 25, 50, and 75 ms, respectively. The plot shows that the increase of disconnection time significantly affects the measured TCP RTT.

Figure 5(b) shows the observed TCP RTT calculated using the model for the same scenario. We can see that the analytical model accurately predicts the measured TCP RTT. The observed differences are the result of the variable losses observed in the experiments and the expected noise in the experimental environment.

Furthermore, there are specific situations where a smaller disconnection time results in a higher RTT. As an example,



(a) Throughput with 50% of connection ($f_1 = 0.5$) to AP_1

(b) Throughput with 10% of connection ($f_1 = 0.1$) to AP_1

Figure 6: Throughput per TCP-flow with different duty cycles and disconnection times.

we consider the RTT in Figure 5(b), for a disconnection time of 50 and 75 ms and an end-to-end wired delay of 125 ms. Here, the interplay of the disconnection time and the delay causes a higher TCP RTT for a disconnection time of 50 ms compared to the 75 ms case. This phenomenon gets less important at smaller disconnection times (see, e.g., the RTT observed with a disconnection time of 15 and 25 ms).

5.2. Impact on Throughput per TCP Flow. In this section, we study the throughput observed by a TCP flow opened on one AP. Figure 6(a) depicts the experimental throughput when the *duty cycle* of one AP is 50%. We observe that even for small delays, the throughput performance may be dramatically affected. As an example, when we operate with a disconnection time of 75 ms, we observe a quasiconstant throughput when the end-to-end wired delay spans from 75 to 150 ms. This is caused by the similar RTT observed at 75 and 150 ms of end-to-end wired delay.

Even more evident is the case where the VSTA is connected for the same amount of time—hence for a connection time of $f_1T = \{15, 25, 50, 75\}$ ms—*but* for a smaller *duty cycle* to AP_1, for example, 10% of its time. We can also see from Figure 6(b) that the penalty in throughput is more severe as the disconnection time grows. For example, when the disconnection time is 675 ms, the average throughput is more than three times smaller than the throughput achieved when the disconnection time is 135 or 225 ms.

5.3. Reducing the Wireless Period. Based on the analysis in the previous section, in order to reduce the impact of TDMA on the TCP throughput, we have to keep the disconnection time as small as possible. Since the disconnection time is equal to $T - f_iT = (1 - f_i)T$, this also implies that for a fixed f_i, the wireless period T should be kept small. We study this issue with experimental tests. Figure 7 shows the throughput achieved as a function of the percentage of time connected to one AP. In the tests, we use different wireless periods ($T = \{30, 50, 100, 150\}$ ms), and we fix the end-to-end delay to 100 ms. The figure shows that similar performance is achieved with wireless periods of $T = 50$ ms and $T = 100$ ms, while throughput can be severely affected choosing a wireless period of $T = 150$ ms, once the time of connection is above 45%.

However, T cannot be reduced as much as we want. For $T = 30$ ms, the station gets slightly less throughput for small *duty cycles*. In fact, there are two limiting factors: (i) there is a time spent by the 802.11 card to switch AP (called *switching cost*, as described in details in Section 6.1); (ii) the frequent AP switching introduces an extra congestion signal rate of ≈0.01–0.08%, caused by an inefficient management of the transmission queues at the driver in off-the-shelf APs [14]. As a practical design aspect, since the congestion signal rate and the switching cost affect more severely smaller duty cycles, we can conclude that the selection of the wireless period T should be based on the smallest duty cycle at the station. In the next section, we provide an accurate description of the selection of the wireless period T and its correlation with the other parameters.

6. Increasing the Aggregated Throughput

In this section, we aim at improving the aggregate throughput at the station by (i) introducing the assignment of slots to each VSTA and (ii) allocating the slots via a distributed resource allocation algorithm. The objective is to minimize the total disconnection time such as the TCP throughput of single flows is increased.

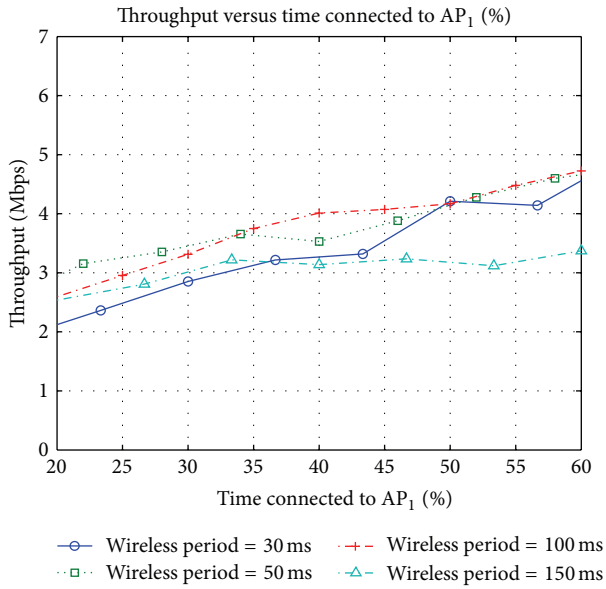


FIGURE 7: Experimental downlink throughput connected 50% of time to one AP for an end-to-end delay of 100 ms.

6.1. Concept of Slotted Operation. Instead of connecting to each AP_i for a consecutive amount of time $f_i \cdot T$, we introduce the concept of slot assignment, and we give $g_i \geq 1$ slots to each $VSTA_i$. For this scope, according to the analysis in Section 5.3, we first define SlotTime as the minimum amount of time allowed in the system at which the effect of the switching cost and the packet losses can be neglected on the connection with the smallest duty cycle.

In order to select such a SlotTime, based on our empirical data, we get at least 85% of the expected throughput—defined as the throughput that would be achieved without any cost of switching—with only 6 ms of connection time over a wireless period of 12 ms, at least 90% with 10 ms over 20 ms and at least 95% of the expected throughput with a connection time of 15 ms over a wireless period of 30 ms (see [14] for details). These values are lower bounds because achieved when the AP queue is constantly backlogged. With Internet traffic, APs are backlogged only in the beginning of the duty cycle (because of the downlink packets already in the AP queue at the time of starting the duty cycle), while instead transmit at the end-to-end transmission rate for the rest of the duty cycle [4]. Then, SlotTime = 10–15 ms safely gets the expected throughput.

However, an allocation of slots of equal length SlotTime would increase the wireless period T, with immediate drawbacks on the performance. As an example, if the station is connected to two APs with scheduler output $f_1 = 0.27$ and $f_2 = 0.73$, we would need 27 slots for AP_1 and 73 slots on AP_2. A SlotTime set to 15 ms would result in a wireless period of $15 \cdot 100 = 1500$ ms, which is computational inefficient.

Driven by the experiments and simulations of the previous section, we resolve this problem with the following principles:

(i) we calculate the wireless period as $T = \text{SlotTime}/\min_i f_i$; that is, the procedure reduces the wireless period T, based on the smallest duty cycle of the station (as demonstrated in Section 5.3);

(ii) we derive the number of slots locally assigned to each $VSTA_i$ as $g_i = \lfloor (f_i T)/\text{SlotTime} \rfloor$, for a total number of slots of $G = \sum g_i$;

(iii) we determine the slot size per $VSTA_i$ as $\text{SlotTime}_i = f_i T / g_i$. This may give slots of different sizes among different VSTAs.

Note that the solution can be transparently applied to the systems proposed in [4, 5, 13], considering the different switching costs of these systems to compute SlotTime.

Once selected T, $\{g_i\}$, and $\{\text{SlotTime}_i\}$, our objective is to construct a resource allocation algorithm that, given the set of duty cycles f_i provided by the upper-layer scheduler, it assigns the set of slots to the APs in order to minimize the overall disconnection time for all the APs. For a rigorous analysis of the resource allocation algorithms, some definition is needed, as introduced in the next paragraph.

Disconnection Cost. Let us define $\mathbf{S}_i = [S_i(1), S_i(2), \ldots, S_i(g_i)]$, the vector that indicates the slot positions in the range $[1, G]$ for $VSTA_i$, with $S_i(g_i + 1) = S_i(1)$ and $S_i(l) \neq S_j(m)$ for any $i, j = 1, \ldots, N$, with $i \neq j$, $l = 1, 2, \ldots, g_i$, and $m = 1, 2, \ldots, g_j$. Besides, we define the cost (slot duration) of each slot as the slot size of the $VSTA_i$ that uses the slot:

$$C_{S_i(l)} = \text{SlotTime}_i \quad \forall i. \tag{1}$$

In order to measure the disconnection cost of the $VSTA_i$ during two transmissions in the slots $S_i(l)$ and $S_i(l+1)$, we take into account the costs of the intermediate slots $C_{S_i(l)+1}, \ldots, C_{S_i(l+1)-1}$. Therefore, we introduce the following cost function:

$$c_{i,l} = \sum_{j=S_i(l)+1}^{S_i(l+1)-1} C_j \quad l = 1, 2, \ldots, g_i. \tag{2}$$

Example. Let us suppose that $N = 3$ and that the slots are allocated as follows: [$VSTA_1$ $VSTA_2$ $VSTA_3$ $VSTA_1$ $VSTA_2$ $VSTA_1$]. This gives $S_1 = [1\ 4\ 6]$, $S_2 = [2\ 5]$, and $S_3 = [3]$. Furthermore, we suppose that $\text{SlotTime}_3 = 10$ ms, $\text{SlotTime}_1 = 12$ ms, and $\text{SlotTime}_2 = 15$ ms. Then, we calculate the disconnection cost between $S_1(1) = 1$ and $S_1(2) = 4$ as $c_{1,1} = C_2 + C_3 = 15 + 10 = 25$ ms.

6.2. Resource Allocation Algorithm. We now present three different, fully decentralized, slot allocation mechanisms with different performance and computational costs that aim to reduce the impact that the multi-AP TDMA has on single TCP flows (we also tested an allocation mechanism with random assignment of the slots, used as a constraint that each slot is assigned to a given $VSTA_i$ with a probability equal to f_i. Although this random assignment may decrease the buffering time at the AP in certain configurations, we found that it generally increases the jitter observed by TCP and then reduces the observed downlink throughput).

Blind Resource Allocation. We have seen in Section 5.1 that a multi-AP TDMA policy increases the observed RTT of

the TCP packets. We have also seen that this increase is exactly the disconnection time in the worst case. In other terms, for VSTA_i, and given an allocation that produces a disconnection time of $\max_{l=1,2,\ldots,g_i} c_{i,l}$, we would have

$$\text{RTT}_i = d_i + \max_{l=1,2,\ldots,g_i} c_{i,l}. \tag{3}$$

The TCP throughput achieved by the above allocation can be approximated as

$$\frac{\text{MSS}}{\left[d_i + \max_l c_{i,l}\right] \cdot \sqrt{p_i}}, \tag{4}$$

where MSS and p_i are the parameters of the Mathis model defined in Section 4.2 [16]. It follows that in order to minimize the throughput penalty caused by disconnection, we need to solve the following problem:

$$\min \sum_i^N \left(\frac{\text{MSS}}{d_i \cdot \sqrt{p_i}} - \frac{\text{MSS}}{\left[d_i + \max_l c_{i,l}\right] \cdot \sqrt{p_i}} \right). \tag{5}$$

The slot assignment obtained from solving (5) depends on the correct estimation of the loss rates $\{p_i\}$ and end-to-end delays $\{d_i\}$. In a realistic deployment, an accurate prediction of these values may be not available. In the absence of any end-to-end delay information, we can reformulate the problem simply as the minimization of the inverse of the maximum disconnection times as follows:

$$\begin{aligned} \max_{S_i(l)} \quad & \sum_i^N \frac{1}{\left(\max_l c_{i,l}\right)}, \\ \text{s.t.} \quad & \sum_{i=1}^G C_i = T, \\ & f_i T = g_i \cdot \text{SlotTime}_i \quad \forall i, \\ & S_i(l) \in \{1, G\} \quad \forall i, l, \end{aligned} \tag{6}$$

where the variables C_i, SlotTime, f_i, and g_i are defined in Table 1.

Min-Max Disconnection Time Allocation Algorithm. The blind resource allocation algorithm defined above can be prohibitively expensive. In order to reduce its complexity, we define a *min-max disconnection time* heuristic approach. This algorithm considers that, in average, the TCP throughput is more severely affected by the amount of time that each VSTA_i is *not connected* to the corresponding AP_i. Therefore, the algorithm tries to minimize the disconnection time, starting with the connection with the largest duty cycle (i.e., the AP backhaul with the highest utilization). The algorithm operates as follows.

(1) First, it allocates the slots to the VSTA with $\max(g_i)$.

(2) Next, the VSTAs with lower number of slots will be served one by one. At each step, the selected ith VSTA_i analyzes *only* the slots not previously assigned, and it calculates the vector $\mathbf{S}_i$ to satisfy the condition: $\min \max_{l=1,2,\ldots,g_i} (S_i(l+1) - S_i(l))$; that is, it selects the g_i slots to minimize the maximum distance between each pair of consecutive slots assigned to the VSTA_i.

(3) Finally, at the last step, the remaining set of slots are assigned to the VSTA with $\min(g_i)$. The last VSTA_i to allocate is the one with the smallest duty cycle. For that one, SlotTime is already chosen such as its performance is not affected.

Upper Bound. We also calculate the upper bound for the TCP aggregate throughput: for each delay, we compute the TCP aggregate throughput for all the feasible solutions and select the one that achieves the maximum throughput. Note that this upper bound cannot be calculated in practice and we include it for comparison purposes.

6.3. Simulation Results. We now evaluate the above algorithms—and particularly the aggregate throughput achieved by the *min-max disconnection time* allocation—via simulations in different scenarios: (i) high number of Aps, (ii) VSTAs with different duty cycle and delay, and (iii) different slot size per VSTA. In the tests, we consider one long-lived TCP flow for each VSTA, and we suppose that the end-to-end communication is limited by the RTT delay so that we do not reach the maximum capacity of the end-to-end path. For each test, we generate 10000 samples of RTT (similar results were achieved with 1000 samples. More samples then 10000 can be used, but it would adversely affect the simulation time, without impact on the results). and we use a congestion signal rate of 0.32%, as the one measured experimentally by connecting to one AP in range with high signal-to-noise ratio and measuring the average number of congestion signals per acknowledged packet (we use the congestion signal rate connecting to one AP because, as discussed in Section 5.3, current off-the-shelf APs implementations add at each AP a certain packet loss rate that may limit the performance in experimental implementations. The reader is referred to [14] for more details. In this work, we do not consider these implementation issues (that, however, do not currently allow a validation of the resource allocation algorithm with real experiments) and assume that the limiting factor is the time spent by the 802.11 card to switch AP). Note that each TCP flow experiences a different RTT, according to the specific duty cycle and slots' assignment. Note also that more TCP flows may be sent per AP, and other congestion signal rates may be used (even higher, according to the link quality and the 802.11 PHY rate) but do not add a new contribution to this section.

Case 1 (high number of Aps). Figure 8(a) shows that a station is connected to 5 APs, with the scheduler selecting the following duty cycles: $f_1 = 0.5$, $f_2 = 0.125$, $f_3 = 0.125$, $f_4 = 0.125$, and $f_5 = 0.125$. The corresponding number of slots, based on the slot calculation presented in Section 6.1, is $g_1 = 4$, $g_2 = 1$, $g_3 = 1$, $g_4 = 1$, and $g_5 = 1$, and the total number of slots is $G = 8$. In the model, we use SlotTime = 15 ms, which gives a wireless period of 15 ms · 8 slots = 120 ms. The algorithm minimizes the time without

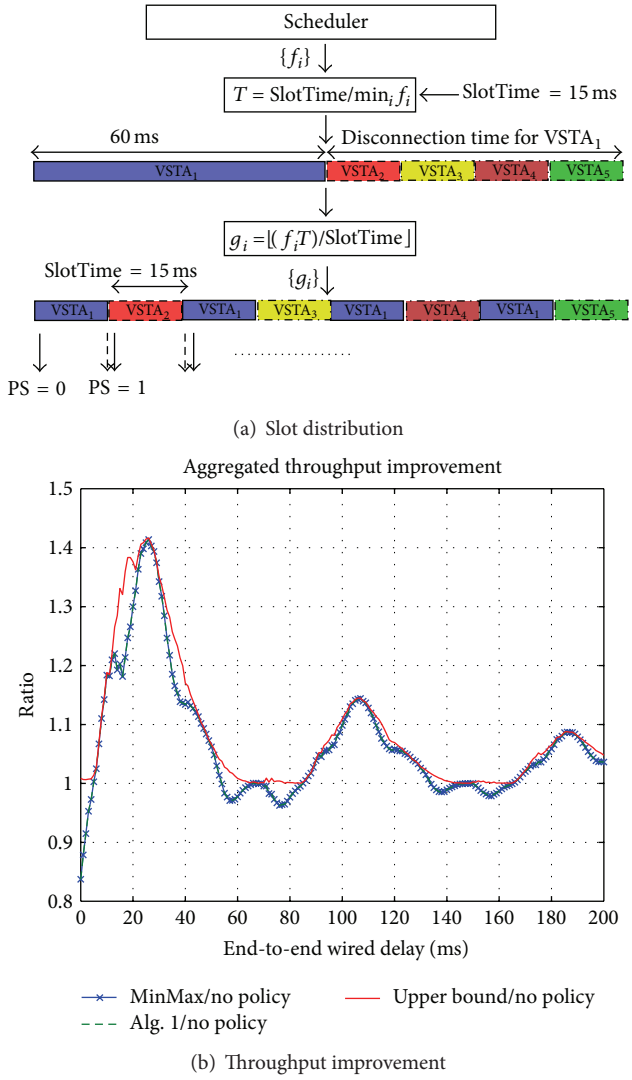


FIGURE 8: Min-max disconnection time allocation algorithm. Case 1.

transmission allocating first the slot to $VSTA_1$, then to $VSTA_2$, $VSTA_3$, $VSTA_4$, and $VSTA_5$.

Figure 8(b) depicts the throughput improvement versus the end-to-end delay, obtained comparing the proposed allocation algorithm (labeled "MinMax") with no resource allocation (labeled "no policy"), that is, spending consecutive 60 ms on $VSTA_1$ and then 15 ms on $VSTA_2$, $VSTA_3$, $VSTA_4$, and $VSTA_5$, sequentially.

We observe that the *min-max disconnection time* allocation improves the throughput in all the cases thanks to the reduction of the disconnection time. The min-max algorithm improves the throughput by up to 1.5 times with respect to the case without any resource allocation. Note that for an end-to-end wired delay of 0–5 ms, the *min-max disconnection time* algorithm has a slight lower aggregate throughput. This is because, with this very small delay, the higher number of AP switching increases the probability that the TCP packet needs to wait at the next connection period before being ACKed.

Figure 8(b) also depicts the throughput achieved by running the algorithm in (6) (labeled "Alg. 1"). We can see that the heuristic approach performs identically to the blind resource allocation.

Finally, we run a test with all possible slot allocations. We verify for each delay the configuration that achieves the upper bound, according to the methodology given in Section 6.2 (labeled "upper bound"). We observe that despite the high cost and the need for an optimal calculation of the end-to-end delay per connection and the packet loss rate, the upper-bound algorithm only slightly increases the aggregate throughput observed by the station with respect to the *min-max disconnection time* approach. The main reason behind this result is that the key parameter that affects the end-to-end TCP throughput is the disconnection time from the AP, which is taken into account in the *min-max disconnection time* approach, rather than the extra-buffering time at the AP.

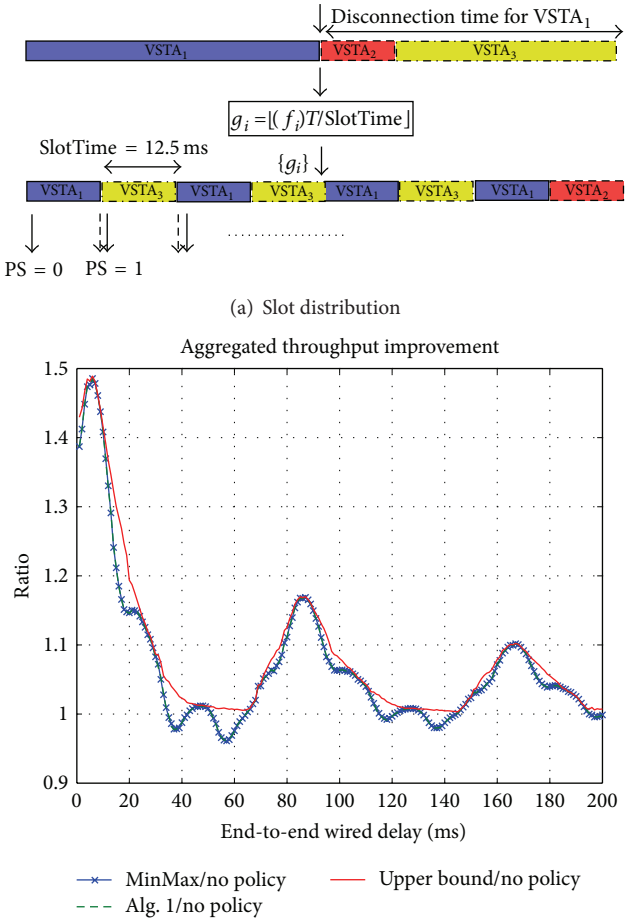
Case 2 (VSTAs have different duty cycle and different delays). We now consider a station connected to 3 APs, with the scheduler giving an output the set: $f_1 = 0.5$, $f_2 = 0.125$, and $f_3 = 0.375$. The corresponding number of slots given by the resource allocation algorithm are $g_1 = 4$, $g_2 = 1$, and $g_3 = 3$. We also suppose that SlotTime = 12.5 ms.

In this example, we consider that the VSTAs experience different delays. Particularly, we suppose that for a given delay x on $VSTA_1$, the delay on $VSTA_2$ is $x + 20$ ms, and the delay on $VSTA_3$ is $x + 40$ ms. We then calculate the aggregated throughput summing the throughput achieved on the three VSTAs at a function of the delay x.

Without using the resource allocation algorithm, $VSTA_1$ would be disconnected 50 ms, $VSTA_2$ for 87.5 ms and $VSTA_3$ for 62.5 ms. The *min-max disconnection time* algorithm increases the granularity of the AP assignment so that $VSTA_1$ will be disconnected for 12.5 ms, $VSTA_2$ (still) for 87.5 ms, and $VSTA_2$ for at most 37.5 ms. Note that the TCP throughput at $VSTA_2$ can be improved only by reducing the SlotTime, since it uses just one slot per period and a resource allocation algorithm cannot contribute to improve $VSTA_2$ throughput.

In Figure 9(b), we observe that the ratio between the aggregated throughput obtained by the *min-max disconnection time* algorithm over the one obtained without any algorithm is higher (up to 1.5 times) in most cases. Particularly, the *min-max disconnection time* allocation algorithm gets higher throughput corresponding to the scenarios where the delay added by the disconnection periods causes higher RTT. In general, the resource allocation algorithm can significantly improve the performance, since the VSTAs have different throughput demands. Our resource allocation algorithm tries to meet these demands by selecting the slot combination that minimizes the disconnection time.

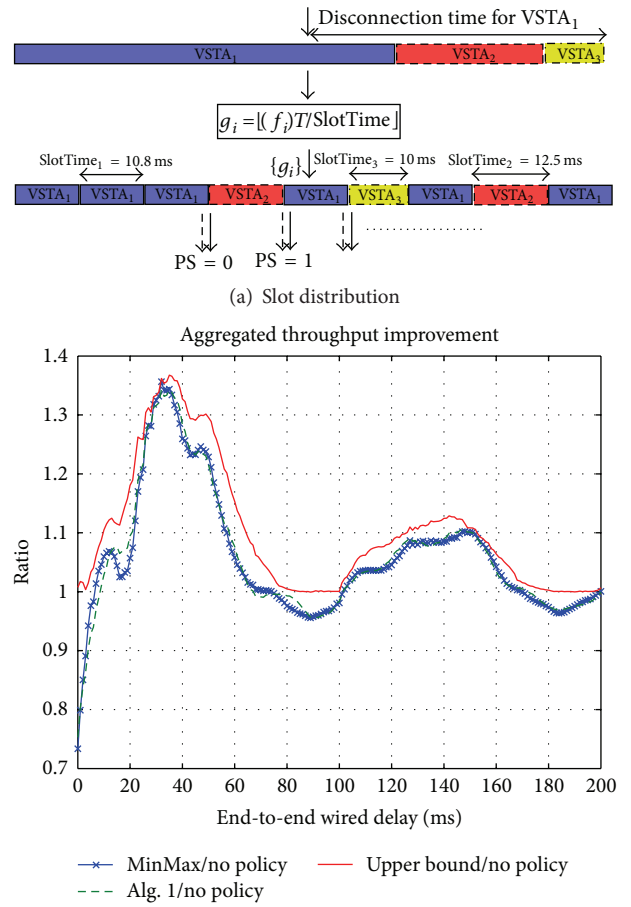
Besides, the "Alg. 1/NoPolicy" line in Figure 9(b) is identical to the one achieved running the algorithm in (6) that needs 280 runs to analyze all the feasible solutions, with respect to the 6 runs needed by the *min-max disconnection time* algorithm. Finally, "Upper Bound/NoPolicy" line shows that the throughput improvements achieved with the optimum solution (upper-bound algorithm) is negligible.



(a) Slot distribution

(b) Throughput improvement

FIGURE 9: Min-max disconnection time allocation algorithm. Case 2.



(a) Slot distribution

(b) Throughput improvement

FIGURE 10: Min-max disconnection time allocation algorithm. Case 3.

Case 3 (different slot size per VSTA). We finally consider a scenario where the slot time length are different per each VSTA, caused by a set of duty cycle equal to $f_1 = 0.65$, $f_2 = 0.25$, and $f_3 = 0.10$. The corresponding slot distribution is given in Figure 10(a), supposing SlotTime = 10 ms. According to the slot allocation procedure defined in Section 6.1, the VSTAs use $\text{SlotTime}_1 = 10.8$ ms, $\text{SlotTime}_2 = 12.5$ ms, and $\text{SlotTime}_3 = 10$ ms.

The results in Figure 10(b) show that the *min-max disconnection time* algorithm still achieves the best tradeoff between performance and computational cost. Particularly, we can see from "Alg. 1/NoPolicy" line that the throughput achieved by the *min-max disconnection time* algorithm is very similar to the one defined in (6). Besides, in some scenario, the *min-max disconnection time* algorithm achieves a slightly higher throughput, despite the cost of only 7 runs compared to the 252 runs needed in (6).

7. Conclusion

The aggregation of the ADSL bandwidth via 802.11 wireless communication and multi-AP TDMA could dramatically increase the RTT observed by TCP flows. In this paper, we studied this problem, both via extensive experiments with our prototype implementation and via a simulator that accurately correlates the TCP RTT with the time spent by the wireless station on each AP. We presented a simple model that accurately predicts the main effects caused by TDMA schemes on the observed TCP RTT, and we introduced a resource allocation algorithm that improves the aggregated throughput with respect to state-of-the-art approaches with a complexity that grows linearly with the number of APs. Our solution does not require modifications to the rest of the network, and it can be applied to existing solutions that aggregate the AP backhaul bandwidth. Furthermore, we showed that its throughput performance is very close to the theoretical upper bound for a number of key scenarios. We believe that our approach will help to provide an efficient solution to aggregate multiple AP backhauls independently of the type of TCP traffic.

References

[1] D. Han, A. Agarwala, D. G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan, "Mark-and-sweep: getting the "inside" scoop on neighborhood networks," in *Proceedings of*

the 8th ACM SIGCOMM Conference on Internet Measurement (IMC '08), pp. 99–104, ACM, New York, NY, USA.

[2] M. Siekkinen, D. Collange, G. Urvoy-Keller, and E. W. Biersack, "Performance limitations of ADSL users: a case study," in *Proceedings of the 8th Passive and Active Measurement Conference (PAM '07)*, vol. 4427 of *Lecture Notes in Computer Science*, pp. 145–154, Louvain-la-neuve, Belgium.

[3] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference (IMC '09)*, pp. 90–102, ACM, New York, NY, USA, 2009.

[4] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi, "FatVAP: aggregating AP backhaul capacity to maximize throughput," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08)*, pp. 89–104, USENIX Association, San Francisco, Calif, USA, April 2008.

[5] D. Giustiniano, E. Goma, A. Lopez Toledo, I. Dangerfield, J. Morillo, and P. Rodriguez, "Fair wlan backhaul aggregation," in *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking (MobiCom '10)*, pp. 269–280, ACM, New York, NY, USA, 2010.

[6] R. Chandra and P. Bahl, "Multinet: connecting to multiple ieee 802.11 networks using a single wireless card," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 2, pp. 882–893.

[7] G. Athanasiou, T. Korakis, O. Ercetin, and L. Tassiulas, "Dynamic cross-layer association in 802.11-based mesh networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 2090–2098, Anhcorage, Ala, USA, May 2007.

[8] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[9] G. Athanasiou, T. Korakis, O. Ercetin, and L. Tassiulas, "A cross-layer framework for association control in wireless mesh networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 1, pp. 65–80, 2009.

[10] C. Doerr, M. Neufeld, J. Fifield, T. Weingart, D. C. Sicker, and D. Grunwald, "MultiMAC—an adaptive MAC framework for dynamic radio networking," in *Proceedings of the 1st IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN '05)*, pp. 548–555, November 2005.

[11] A. Sharma and E. M. Belding, "FreeMAC: framework for multi-channel mac development on 802.11 hardware," in *Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO '08)*, pp. 69–74, ACM, August 2008.

[12] A. Rao and I. Stoica, "An overlay MAC layer for 802.11 networks," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys '05)*, pp. 135–148, ACM, June 2005.

[13] A. J. Nicholson, S. Wolchok, and B. D. Noble, "Juggler: virtual networks for fun and profit," *IEEE Transactions on Mobile Computing*, vol. 9, no. 1, pp. 31–43, 2010.

[14] D. Giustiniano, E. Goma, A. Lopez, and P. Rodriguez, "Wiswitcher: an efficient client for managing multiple aps," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO '09)*, pp. 43–48, ACM, New York, NY, USA, 2009.

[15] W. R. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," 1997.

[16] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *Computer Communication Review*, vol. 27, no. 3, pp. 67–82, 1997.