Noname manuscript No.
(will be inserted by the editor)

Enhancing the performance of TCP over Wi-Fi Power Saving Mechanisms

Daniel Camps-Mur · Sebastià Sallent-Ribes

Received: date / Accepted: date

Abstract The Wi-Fi technology is quickly being adopted by new types of devices that pose stringent requirements in terms of energy efficiency. In order to address these requirements the IEEE 802.11 group developed in the recent years several power saving protocols, that are today widely used among devices like smartphones. In this paper we study, by means of analysis and simulation, the effect that these power saving protocols have on the performance/energy trade-off experienced by long lived TCP traffic. Our study unveils that the efficiency of Wi-Fi power saving protocols critically depends on the bottleneck bandwidth experienced by a TCP connection. Based on the obtained insights, we design and evaluate a novel algorithm, BA-TA, which runs in a Wi-Fi station, does not require any modification to existing 802.11 standards, and using only information available at layer two, improves the performance/energy trade off of long lived TCP connections, whilst also exhibiting a notable performance with Web traffic and TCP Streaming.

Keywords Wireless LAN · Power Saving · TCP · MAC Layer

1 Introduction

Since its conception the Wi-Fi technology has expanded into a wide variety of devices and applications. In particular, there is nowadays an emerging trend towards embedding Wi-Fi in all kind of battery powered devices, which pose stringent requirements on energy efficiency. Among these devices, smartphones are a prominent device class [1].

Being aware of the importance of energy efficiency, the IEEE 802.11 group developed a set of technologies tailored to fulfill this requirement. For instance, the 802.11 standard [3] defined a basic power save mode that allows stations to switch off their radio during inactivity periods in order to save power. When this protocol is used an Access Point (AP) buffers incoming data for the sleeping stations, and periodically notifies them about their buffered data using the Beacon frame, typically sent every 100ms. A power saving station wakes up to receive the Beacon frame and retrieves its buffered data by sending a PS-Poll frame to the AP. In addition, this basic power save mode was later enhanced by defining the *Unscheduled* Automatic Power Save Delivery APSD (U-APSD) protocol. The main difference between U-APSD and the original 802.11 power save mode is that in order to retrieve their buffered data, U-APSD stations can proactively trigger the AP instead of waiting for the Beacon frame. In addition, U-APSD allows an AP to deliver multiple buffered frames upon receiving a single trigger frame[1] from the station. The interested reader is referred to [4] and [5] for a detailed description of these power saving protocols.

It was however soon realized that the operation of Wi-Fi power saving protocols could be detrimental to the performance of data applications. In order to address this concern, a common practice in the industry is to configure a Wi-Fi device to operate in power save mode only until there is traffic to be sent, and then switch to active mode[2] until a timeout expires without having sent or received any traffic. An obvi-

Daniel Camps-Mur
NEC Europe Laboratories in Heidelberg, Germany.
E-mail: camps@neclab.eu

Sebastià Sallent-Ribes
Polytechnic University of Catalonia (UPC) in Barcelona, Spain. This work is partially supported by the Spanish government through project TEC2010-20527-C02-01.

[1] In U-APSD both signaling frames called QoS Nulls and regular data frames trigger the delivery of buffered data from the AP.

[2] In order to switch between active mode and power save mode, a station sets or unsets the Power Management bit in frames sent to the AP.

ous drawback of this approach though, is that when there is a continuous flow of traffic, the device stays all the time in active mode, achieving no energy saving.

Several works in the literature have tried to optimize the trade off between application performance and energy efficiency in Wi-Fi devices. Among the initial work in this field, [6] pointed out that Wi-Fi power saving protocols may heavily slow down data applications like Web, and proposed a layer two algorithm that allowed to bound the introduced slow down with a limited amount of extra power consumption. This initial work was further extended in [7]. More recently, in [8], an *Opportunistic power save method* was proposed whereby Wi-Fi stations dynamically switch between active mode and power save mode. The algorithm proposed therein though, requires modifications to the 802.11 standard and it only considers the transmission of short files. Other authors, like [9] and [10], took a different approach and proposed to enhance energy efficiency for specific applications making use of cross-layer mechanisms. For instance, in [9] the authors introduced a middle-ware entity to convey information about the activity of the applications to the MAC layer, in order to maximize sleeping periods. In [10] a cross-layer method was proposed that adjusts the advertised window field in TCP ACKs in order to shape TCP transmissions in bursts and maximize sleeping periods. Notice though that relaying on cross-layer techniques may hinder the deployment of the proposed solutions.

A different body of work has dealt with the problem of improving energy efficiency in Wi-Fi for real-time applications, e.g. VoIP and Video. In this context [11] and [12] have proposed algorithms that dynamically adapt the trigger interval used by a Wi-Fi station according to the characteristics of the applications.

Finally, the authors in [13], [14] and [15], proposed to improve energy efficiency in Wi-Fi clients by having a Wi-Fi AP schedule the moment where stations retrieve their buffered data in order to minimize the contention required to access the channel. Although this approach can certainly improve energy efficiency, it usually requires modifications to the 802.11 standard, and hence Wi-Fi clients attached to legacy APs cannot benefit from them.

Our work in this paper focuses on studying the detailed interactions between Wi-Fi power saving protocols and long lived TCP connections. Notice that the use of long lived TCP connections in battery constrained devices is gaining momentum with the advent of cloud-based applications that seamlessly synchronize content among devices[3], and with the increased usage of remote applications updates in mobile operative systems. In addition, popular content providers also distribute some forms of Video[4] as traditional TCP file transfers [16]. To the best of our knowledge this is the first paper in the state of the art attempting such a detailed study.

The main contributions of this paper are as follows:

- We study by means of analysis and simulations, the effect that Wi-Fi power saving protocols have on the performance/energy trade off of long lived TCP traffic. Our study unveils that the efficiency of Wi-Fi power saving protocols critically depends on the bottleneck bandwidth experienced by a TCP connection.
- Based on the obtained insight, we design and evaluate a novel algorithm, BA-TA, that runs in a Wi-Fi station, does not require any modification to existing 802.11 standards, and using only information available at layer two, enhances the performance/energy trade-off experienced by long-lived TCP connections, whilst exhibiting a notable performance with Web traffic and TCP Streaming.

This paper is organized as follows: Section 2 analyses the effects of Wi-Fi power saving protocols on long lived TCP traffic. Based on the insights obtained in this section, Section 3 introduces the design of our BA-TA algorithm, which is then thoroughly evaluated in Section 4. Finally, Section 5 summarizes and concludes this paper.

2 TCP performance over *Wi-Fi PSM*

In order to study the performance of TCP over Wi-Fi power saving protocols we consider the scenario depicted in Figure 1. This scenario represents a typical deployment, where a battery limited handheld device accesses the Internet through a Wi-Fi Access Point (AP), which is in turn provisioned by an access technology, typically xDSL. Our goal is to study the performance/energy trade-off achieved by Wi-Fi power saving protocols when used with data applications. In particular, we focus on long lived TCP connections[5], since the effect of power saving protocols on short lived connections, like Web, has already been thoroughly studied in the literature [6]. In our study, we have modeled a long lived TCP connection using a 50MB file transfer, which is close to the median size of a video file in the Internet [22], and have used TCP New Reno as transport protocol[6].

As it will be shown throughout this section, the performance of such a file tansfer depends on several key parameters in the scenario depicted in Figure 1, e.g. the bandwidth

[3] *DropBox* or *iCloud* are popular examples of such applications [2].

[4] The study presented in [16] reports that YouTube delivers Flash HD videos as file transfers. Other video formats though are transmitted using modified TCP connections, e.g. using bandwidth throttling or block sending.

[5] We define long lived connections as those that enter congestion avoidance.

[6] Notice that although other TCP versions like TCP Cubic [18] are currently being deployed in major operative systems, we justify our election of TCP New Reno on the fact that these new TCP versions behave in a TCP-friendly way, i.e. like New Reno, in low bandwidth delay products like the ones considered in this work.

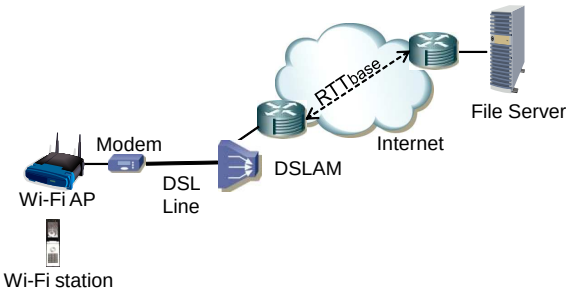


Fig. 1 Scenario under study

	DL/UL Rate	Description
Slow DSL	1Mbps/128Kbps	Close to the average connection speed in China [21].
Fast DSL	16Mbps/1Mbps	Close to the average connection speed in South Korea and also a common DSL offer in Europe [21].
WiFi Bottleneck	100Mbps/100Mbps	Speeds in this range represent FTTH deployments in South Korea or Japan [21].

Table 1 Access Technologies under study

of the access and Wi-Fi networks, the amount of available buffering, or the Round Trip Time (RTT) experienced by the connection. In order to analyze the influence of these parameters, we study the scenario depicted in Figure 1 by means of packet level simulations using OPNET [17].

Regarding the choice of access technology, we select three representative scenarios that capture today's heterogeneous broadband market. In particular, the selected scenarios span from what are nowadays considered slow access technologies to what are considered fast access technologies. These scenarios are described in Table 1.

The choice of the Wi-Fi technology is another important factor to consider when studying the performance of power saving protocols. However, there is also a wide variation in today's deployed Wi-Fi technologies, ranging from legacy 802.11b networks to 802.11n networks. Therefore, we consider in our study that the Wi-Fi network depicted in Figure 1 can operate according to two different configurations: i) a *Slow Wi-Fi* configuration, where a low over-the-air priority is used and no Transmission Opportunities (TXOP) are allowed[7], and ii) a *Fast Wi-Fi* configuration, where a high over-the-air priority is used and TXOPs are allowed. The used Wi-Fi configurations are described in Table 2.

For each of the previous scenarios we will evaluate the performance/energy trade-off of a generic file tansfer, using two different algorithms. The first one represents the common industry practice of switching to *Active Mode* once traffic is detected, which should be optimal in terms of performance. Hereafter we refer to this algorithm as the *Active Mode* algorithm. The second algorithm consists of having the station all the time in power save mode, which should be optimal in terms of energy. We refer to this algorithm as the *Wi-Fi PSM* algorithm. In particular, we use U-APSD as Wi-Fi power saving protocol, where the station wakes up at every Beacon to check if there is buffered traffic and in that case retrieves it using a single signaling trigger. More details on this U-APSD implementation can be found in [5].

[7] TXOPs allow a Wi-Fi device to transmit several frames with a single access to the channel, hence significantly reducing overhead.

	Data/Ctrl Rate	AIFS/CWmin/CWmax/TXOP
Slow WiFi	54/24 Mbps	3/15/1023/0ms
Fast WiFi	130/24 Mbps	2/7/15/3ms

Table 2 Wi-Fi Configurations under study

In order to evaluate energy consumption, we make use of a well known model for Wi-Fi chipsets that consists of four basic power states: Sleep, Listen, Reception and Transmission. Energy is computed by integrating the power that a Wi-Fi device spends in each of the previous states over a certain target time, which in our evaluation will be the time to transfer a file. The power consumption values used are shown in Table 3 [23].

Broadcom 4311^{TM}	**Sleep**	**Listen**	**Rx**	**Tx**
Power (mW)	20	390	1500	2000

Table 3 Power consumption levels used in our study

Finally, for each of the scenarios and algorithms under study, we have performed two different experiments that represent the different conditions that may be experienced by a TCP connection. Specifically:

- *Buffer Experiment*, where we configure $RTT_{base} = 20ms$ in Figure 1[8], and vary the amount of buffering available in the DSLAM or in the AP (depending on the considered scenario), from 20 to 160 packets. Notice that given the potential harm caused by large buffers, also known as *Bufferbloat* [19], [25], it is important to asses the performance delivered by the protocols under study with small buffers.
- *RTT experiment*, where we fix the buffering available in the DSLAM and AP to 50 packets and 100 packets respectively, and vary the value of RTT_{base} in Figure 1 from 10ms to 310ms.

The simulation results shown in this paper have been obtained considering at least 10 simulation runs. In addition the depicted average values are plotted with their correspondent 95% confidence intervals.

[8] Typical RTT in a domestic Internet path [20].

2.1 Buffer Experiment

We start analyzing the results of our first experiment, where we fix $RTT_{base} = 20ms$ in Figure 1 and vary the amount of buffering available in the DSLAM and in the AP in our scenarios under study.

2.1.1 Slow DSL scenario

The upper graphs in Figures 2(a) and 2(b) depict respectively the throughput and energy consumption experienced by a Wi-Fi station when retrieving a 50MB file from a server located in the Internet, as depicted in Figure 1. It is clearly seen in these figures how all the algorithms under study deliver a similar throughput, which corresponds to the bottleneck bandwidth (1 Mbps), and how being in power save mode during the file tansfer drastically reduces the energy required by the Wi-Fi station to retrieve the file (up to a five fold decrease).

In order to understand why *Wi-Fi PSM* achieves such a significant energy reduction with no penalty in terms of throughput, Figure 3(a) depicts the time evolution of the TCP congestion window (blue line), the queue occupancies at the DSLAM (red line) and the Wi-Fi AP (black line), and the power state of the Wi-Fi station (awake or sleep) during the file tansfer. We can see in the graph, how the Wi-Fi station wakes up right before the Beacon is sent (dotted vertical lines in the graph), quickly empties the AP buffer using the high bandwidth available in the Wi-Fi network, and efficiently sleeps until the next Beacon. Instead, in the *Active Mode* algorithm, the Wi-Fi station stays most of the time awake but inactive, waiting for the next TCP packet[9].

The key to understand why the *on/off* behavior introduced by the power saving protocol does not reduce performance, is to realize that in this experiment the bottleneck queue (red line in Figure 3(a)) *never gets empty*. Thus, a necessary condition to avoid degrading throughput in *Wi-Fi PSM*, is that the TCP source's congestion window (hereafter referred to as *cwnd*) needs to grow big enough in order to compensate for the increased RTT introduced by the power saving protocol, i.e. $RTT_{eff} = \lceil \frac{RTT_{base}}{BI} \rceil BI$, where RTT_{eff} stands for effective RTT, BI is the Beacon interval and RTT_{base} is depicted in Figure 1. Thus, the minimum $cwnd$, in MTU-sized packets, required to avoid any throughput degradation equals to:

$$cwnd > \frac{RTT_{eff} \times R_{DSL_{DL}}}{MTU} = \frac{\lceil \frac{RTT_{base}}{BI} \rceil BI \times R_{DSL_{DL}}}{MTU} \quad (1)$$

[9] The transmission time of a 1.5KB packet over a 1Mbps DSL line is 12ms.

Given the low speed of the DSL link ($R_{DSL_{DL}} = 1Mbps$) and the small RTT_{base} ($20ms$) considered in this scenario, Equation 1 results in $cwnd > 8.53$ packets, which is easily achieved given the considered buffer sizes.

2.1.2 Fast DSL scenario

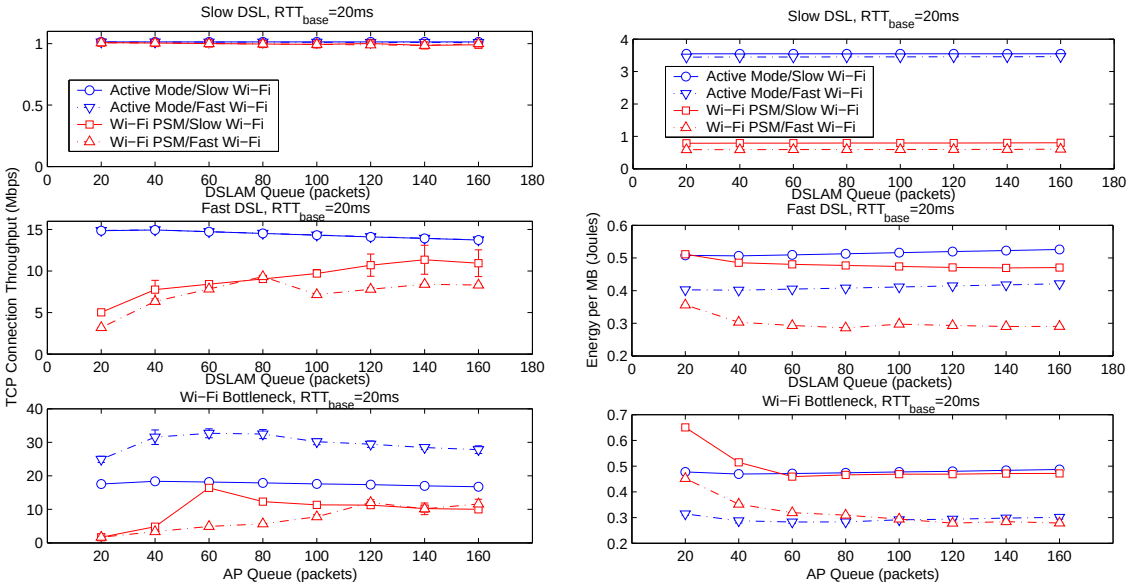
We now repeat the previous experiment in the *Fast DSL* scenario. Throughput and energy results are depicted respectively in the middle graphs of Figures 2(a) and 2(b), and reveal a very different behavior than the one observed in the *Slow DSL* scenario. In this case the file tansfer throughput significantly degrades with *Wi-Fi PSM*, specially when the DSLAM buffer is small, and interestingly it degrades even more when more bandwidth is available in the Wi-Fi network (*Fast Wi-Fi* configuration). Regarding energy, *Wi-Fi PSM* continues to be more efficient than the *Active Mode* algorithm, although a smaller gain than in the *Slow DSL* scenario is obtained.

In order to get a deeper understanding on the underlying dynamics, Figure 3(b) illustrates, when the station uses *Wi-Fi PSM*, an example of the AP and DSLAM queue dynamics (red and black lines) and the evolution of the TCP sender's *cwnd* (blue line). We can immediately see how the dynamics in this case are much more involved that in the *Slow DSL* scenario. In particular, the performance of TCP over *Wi-Fi PSM* depends in this case on the following three effects:

1. The increased bandwidth delay product introduced by *Wi-Fi PSM*.
2. Bursty increases in the DSLAM buffer due to ACK compression.
3. Extended awake periods experienced by the Wi-Fi station.

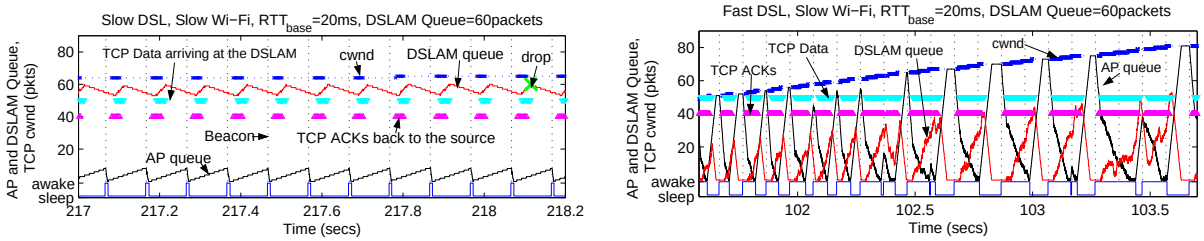
Next, we describe these effects in detail. We can clearly see in Figure 3(b), how the DSLAM queue (red line) often returns to zero leading to periods where the DSL line is underutilized. Using the lessons learned in the *Slow DSL* scenario, we can see that in order to keep the DSL line always busy in this scenario, the TCP sender's $cwnd$ should be at least $cwnd_{min} > \frac{\lceil \frac{RTT_{base}}{BI} \rceil BI \times R_{DSL_{DL}}}{MTU} = 133.3$ packets, which is not the case in Figure 3(b).

Notice thus, that when *Wi-Fi PSM* is used it is critical for the TCP sender to be able to achieve a high $cwnd$ in order to compensate for the increased badwidth delay product introduced by the power saving protocol. The maximum $cwnd$ achieved by a TCP sender depends on the bandwidth delay product of the connection and on the buffering available in the bottleneck, which explains why throughput degrades when buffering is small in *Wi-Fi PSM*. However, there is another important factor that limits the achievable $cwnd$ when *Wi-Fi PSM* is used. This factor is the *ACK compression* introduced by the power saving protocol.



(a) file tansfer throughput (b) Energy spent in the file tansfer

Fig. 2 Buffer Experiment. For each of the algorithms and scenarios under study, throughput is depicted on the left graph and energy on the right one. Notice that the same legend applies to all sub-figures.



(a) Dynamics with a Slow DSL access network (b) Dynamics with a Fast DSL access network

Fig. 3 Buffer Experiment. TCP dynamics. The left graph depicts the *Slow DSL* scenario and the right one the *Fast DSL* scenario.

We can see in Figure 3(b), how when *Wi-Fi PSM* is used, the AP holds TCP packets for the station until the Beacon transmission time, and then transmits these packets to the station at the Wi-Fi rate, which is faster than the bottleneck rate (the DSL rate). The Wi-Fi station then reflects back to the TCP source TCP ACKs at the rate of the received data, which result in a *compressed* burst of new TCP packets arriving at the DSLAM that cause a bursty queue increase, as clearly observed in Figure 3(b) (red line). This process is graphically described in Figure 4, where it can be seen how the ACK compression rate seen by the TCP source is typically limited by the upstream DSL bandwidth.

In order to understand how ACK compression affects the maximum $cwnd$ achieved by the TCP source, it is possible to compute the maximum burst size N_{max} (in MTU-sized packets) that results in no packet loss when arriving at the DSLAM buffer. Considering that, as illustrated in Figure 4, a new window of TCP data hits the bottleneck node at a rate equal to $R_{ACKcomp}$ MTU-sized packets/sec, and that the rate at which the bottleneck queue is drained is in our case $R_{DSL_{DL}}$ MTU-sized packets/sec, it is easy to see that while a new window of TCP data hits the bottleneck, this queue grows at a rate of $R_{qbott} = R_{ACKcomp} - R_{DSL_{DL}}$. Thus, if the previous queue growth is sustained during a time T_{win}, such that $T_{win}R_{qbott} > Q_{DSLAM}$, there will be a loss in the bottleneck queue limiting the size of $cwnd$. Considering that packets within a TCP window arrive at the bottleneck with an interarrival time of $T_{arrival} = \frac{1}{R_{ACKcomp}}$, with $R_{ACKcomp}$ defined in Figure 4, we have that $T_{win} = \frac{N_{max}}{R_{ACKcomp}}$, and the maximum burst size, N_{max}, that results in no drop in the bottleneck's queue, and so in no $cwnd$ reduction in the TCP connection, is:

$$N_{max} < Q_{DSLAM}\frac{R_{ACKcomp}}{R_{qbott}} = \frac{Q_{DSLAM}}{1 - \frac{R_{DSL_{DL}}}{R_{ACKcomp}}} \qquad (2)$$

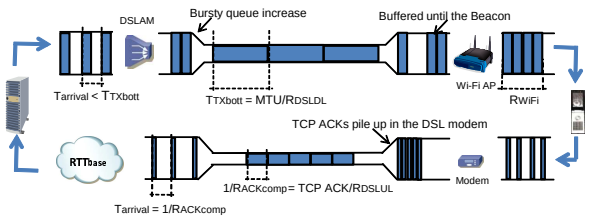


Fig. 4 ACK compression in a TCP connection caused by Wi-Fi power saving protocols.

Where the previous equation only holds if $R_{ACKcomp} > R_{DSL_{DL}}$. Notice that if for instance $Q_{DSLAM} = 40$ packets in the *Fast DSL* scenario, Equation 2 results in $N_{max} < 89$ packets, which limits the maximum value of $cwnd$ and reduces throughput. In the next sections we will study in more detail the relation between N_{max} and $cwnd$.

It is interesting to notice, looking at Equation 2, that when the amount of ACK compression increases, e.g. if the DSL upstream bandwidth increases in Figure 4, then N_{max} decreases and, contrary to intuition, the TCP connection throughput experienced by a Wi-Fi station in power save mode should degrade. Later we will evaluate the validity of this claim.

Finally, there is a third effect affecting the TCP connection throughput. Unlike the previous two effects though, this effect tends to increase TCP throughput, and is described as follows. If the time required by the Wi-Fi station after the Beacon frame to retrieve the buffered TCP packets and generate the correspondent TCP ACKs, is above RTT_{base} in Figure 1, then the Wi-Fi station is still awake when a new window of TCP data from the file server arrives at the AP, and so this new window of data will not suffer from an increased RTT. This effect is illustrated in Figure 5 and is also often observed in Figure 3(b), for instance between 103.2 secs and 103.4 secs, where the AP queue contains packets during two consecutive Beacon intervals. Notice though, that since in this scenario the Wi-Fi rate is higher than the DSL rate, the Wi-Fi station eventually manages to empty the AP queue and return to sleep, resulting again in an increased RTT and reduced throughput. In addition, notice that the previous positive effect is more likely to occur when the bandwidth in the Wi-Fi network reduces, because in this case the AP takes longer to transmit data to the station. This effect explains why when *Wi-Fi PSM* is used, the *Slow Wi-Fi* configuration outperforms the *Fast Wi-Fi* one as depicted in Figure 2(a).

To conclude the analysis of the *Fast DSL* scenario, we turn our attention to Figure 2(b) that depicts the energy spent by the Wi-Fi station to retrieve the 50MB file. The reason why the energy reduction obtained by *Wi-Fi PSM* is less significant than in the *Slow DSL* scenario, is that the DSL line is closer to saturate the Wi-Fi network, hence the station does not spend so much time idle when being in active mode. Finally, it is worth noticing that the *Fast Wi-Fi* configuration reduces energy consumption because TCP packets are transmitted faster.

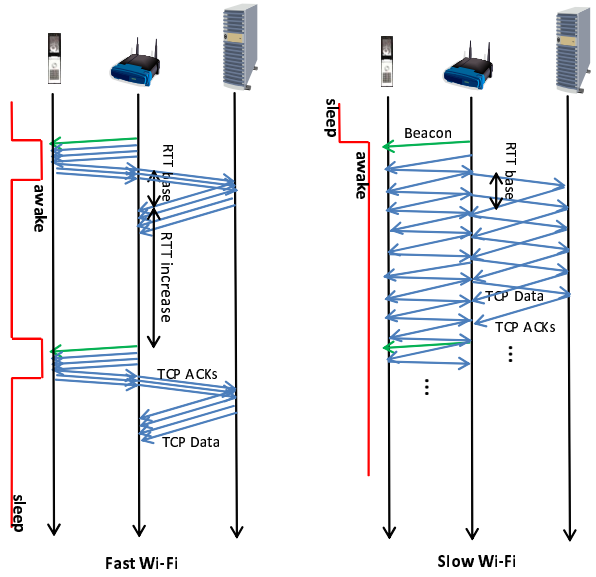
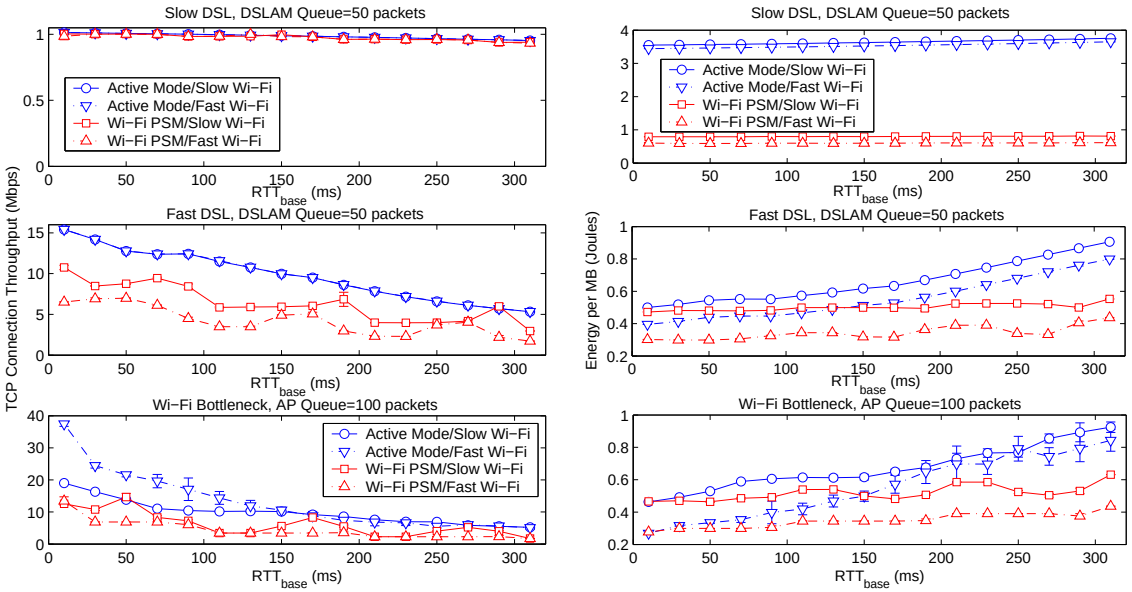


Fig. 5 Interaction between TCP and Wi-Fi PSM. In the *Fast Wi-Fi* configuration data is transmitted fast and aggregated in TXOPs in the Wi-Fi network, thus when a new burst of TCP data arrives at the AP the station is already sleeping resulting in an increased RTT. Instead in the *Slow Wi-Fi* configuration when a new burst of TCP data arrives at the AP the station might still be awake retrieving the previously buffered data, hence the RTT increase is avoided.

2.1.3 Wi-Fi Bottleneck

We finalize the analysis of our Buffer experiment looking at the throughput and energy performance obtained in the *Wi-Fi Bottleneck* scenario. These results are depicted in the lower graphs of Figures 2(a) and 2(b). Looking first at throughput, we can see that the effects observed in the *Fast DSL* scenario appear now even more magnified: i) *Wi-Fi PSM* significantly degrades throughput, specially when the buffer in the AP is small, and ii) the *Fast Wi-Fi* configuration provides the expected throughput increase when the station uses *Active Mode*, but results in a severe throughput degradation when the station uses *Wi-Fi PSM*.

Wi-Fi PSM also introduces ACK compression in this scenario, but, since upstream bandwidth is not a problem in this case, the amount of ACK compression depends on the particular mechanisms employed in the Wi-Fi network. For instance, when the *Slow Wi-Fi* configuration is used, the AP and the station compete in a fair way after each Beacon in order to transmit the buffered TCP packets and the corresponding TCP ACKs. Therefore the average time between TCP ACK and TCP data transmissions is the same, and no significant ACK compression is introduced. However, when the *Fast Wi-Fi* configuration is used, the AP first transmits to the station all its buffered TCP packets packed within a



(a) file tansfer throughput (b) Energy spent in the file tansfer

Fig. 6 RTT Experiment. For each of the algorithms and scenarios under study, throughput is depicted on the left graph and energy on the right one. Notice that the same legend applies to all sub-figures.

TXOP. This transmission is then followed by another transmission where the station packs all the TCP ACKs within a TXOP. Thus, assuming that AP and station use the same data rate, the ACK compression rate in this case can be approximated as $\frac{R_{ACKcomp}}{R_{Wi-Fi}} = \frac{Size\ TCP\ Packet}{Size\ TCP\ ACK}$.

The positive effect stemming from the overlapping between new TCP arrivals and the awake periods of the *Wi-Fi PSM* station also occurs in this case, but mostly in the *Slow Wi-Fi* configuration. The high data rates and bursty transmissions used in the *Fast Wi-Fi* configuration reduce the likelihood of this positive effect, as illustrated in Figure 5.

We highlight now another degradation[10] introduced by *Wi-Fi PSM*, that is caused by an increase in the time required by the TCP connection to complete the initial *Slow Start+Fast Recovery* phase. In our experiments, a TCP connection typically finalizes the initial Slow Start phase when tree duplicated ACKs are received at the TCP source; at that moment TCP New Reno triggers a fast retransmission and the connection enters Fast Recovery until all outstanding data is successfully transmitted. Notice that during the initial Slow Start, the TCP connection typically saturates the network, forcing the *Wi-Fi PSM* station to stay awake. However, when Fast Recovery starts, the TCP source stops transmitting data until half a window of duplicated ACKs is received [27], which allows the station in power save mode to drain the AP buffer and return to sleep. After that moment, since Fast Recovery only injects new data or retransmissions upon receiving duplicate or partial ACKs, transmissions only occur once every 100ms when the station wakes up after the Beacon, hence slowing down the completion of this phase. Figure 8(b) depicts a typical initial *Slow Start+Fast Recovery* phase when the station uses *Active Mode* and when the station uses *Wi-Fi PSM*, in an experiment where $RTT_{base} = 20ms$.

Finally, we can see in Figure 2(b), that when the Wi-Fi network is the bottleneck the *Active Mode* algorithm turns out to be the most energy efficient algorithm, because the Wi-Fi station spends all its time transmitting and receiving useful data.

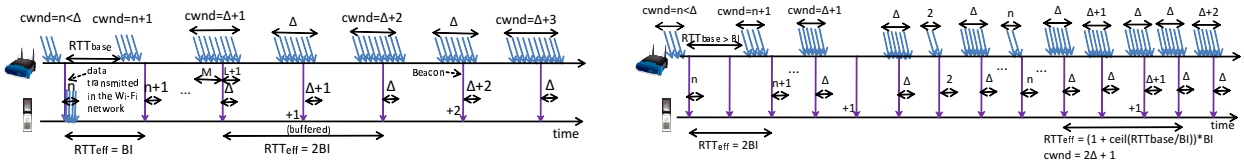
2.2 RTT Experiment

We describe now the results of our second experiment, where we fixed the maximum buffering in the DSLAM and AP and observed the throughput/energy trade-off when varying RTT_{base} from 10ms to 310ms. We considered a maximum buffer of 50 packets in the DSLAM in the *Slow DSL* and *Fast DSL* scenarios, and of 100 packets in the AP in the *Wi-Fi Bottleneck* scenario.

2.2.1 Slow DSL scenario

As illustrated in the upper part of Figures 6(a) and 6(b), the behavior of the algorithms under study in the *Slow DSL* scenario continues to be very similar than the one observed in our first experiment. Following Equation 1 we can see that

[10] This phenomena also occurred in the *Slow DSL* and *Fast DSL* scenarios, but had a smaller impact.



(a) $RTT_{base} \leq BI$ (b) $RTT_{base} > BI$

Fig. 7 TCP model. The two graphs depict the data arrivals from the DSLAM to the AP, the Beacon transmissions from the AP to the Wi-Fi station, and the delivery of the buffered data after the Beacon.

even when $RTT_{base} = 310ms$ the $cwnd$ required to keep the DSL line always busy is $cwnd > 34$ packets, which is easily achieved considering the configured DSLAM buffer of 50 packets.

2.2.2 Fast DSL scenario

The middle graph in Figures 6(a) and 6(b) depict respectively the throughput and energy performance of the algorithms under study in the *Fast DSL* scenario. It is interesting to observe in this case that when the station is in power save mode, there is a ripple effect in the connection's throughput as RTT_{base} varies. Notice that this ripple is contrary to the common knowledge that TCP throughput degrades when the RTT increases, as clearly observed when the station uses *Active Mode*. Next, we present an analytical model aimed at capturing the interactions between the TCP congestion control mechanisms and the Wi-Fi power saving protocols that result in the observed ripple effect[11]. Our simplified model is based on the following assumptions:

i. We consider only the throughput experienced by the TCP connection during the congestion avoidance phase.
ii. We assume that the Wi-Fi station retrieves all the TCP packets buffered at the AP and sends the corresponding TCP ACKs before a new window of TCP data arrives at the AP. Notice that this behavior is more likely when the Wi-Fi bandwidth is high (i.e. *Fast Wi-Fi* configuration) or when RTT_{base} increases.
iii. We consider that packet drops are only due to buffer overflow in the DSLAM, but not in the AP (i.e. Q_{AP} is big enough). In addition, we assume that no channel errors occur in the wired network, and that the wireless network is always able to recover a failed transmission using retransmissions.
iv. We assume that the rate at which TCP ACKs arrive at the TCP source is constrained by the uplink DSL modem, i.e. $T_{arrival} = \frac{Size\ TCP\ ACK}{R_{DSL_{UL}}}$ (see Figure 4).

[11] Notice that the goal of our model is not to accurately model TCP throughput over Wi-Fi PSM, but just to qualitatively understand the dynamics depicted in Figure 6.

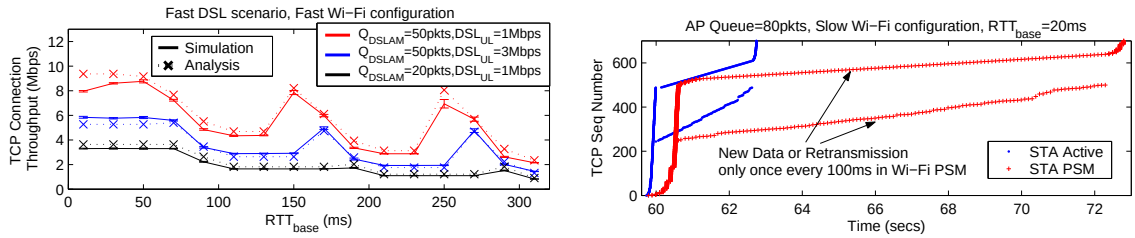
Under the previous conditions, and leveraging Equation 2, we can define $N_{max} = \frac{Q_{DSLAM}}{1-\frac{R_{DSL_{DL}}}{R_{ACK_{comp}}}}$, as being the maximum burst size (in MTU-sized packets) that can hit the DSLAM node without resulting in a packet drop. In addition, we will define Δ as being the *maximum* number of TCP data packets arriving from the DSLAM to the the AP that a Wi-Fi station can retrieve after the Beacon frame before going to sleep[12]. Δ depends on the relation between RTT_{base} and the Beacon Interval (BI), and on the Wi-Fi rate. Specifically, we will approximate Δ as $\Delta = M + L$, where $M = \frac{BI - BI \mod RTT_{base}}{T_{TX_{bott}}}$ is the maximum number of packets transferred from the DSLAM to the AP since the first packet of the new TCP window hits the DSLAM until the next Beacon time, and $L = \frac{T_{TX_{Wi-Fi}}(M)}{T_{TX_{bott}}}$ is the number of packets that can be transferred from the DSLAM to the AP while the Wi-Fi station is retrieving the first M packets; $T_{TX_{bott}} = \frac{MTU}{R_{DSL_{DL}}}$ is defined in Figure 4, and $T_{TX_{Wi-Fi}}(M)$ is the time to transmit M packets and generate the corresponding TCP ACKs in the Wi-Fi network. These variables are illustrated in the third Beacon frame in Figure 7(a).

The key to understand the ripple effect observed in Figure 6(a), is to realize that when the station is in power save mode, increasing RTT_{base} may result in the TCP source being able to achieve a higher $cwnd_{max}$ and therefore a higher throughput. Specifically, the maximum size of $cwnd$ depends on the relation between N_{max} and Δ.

We start considering the case where $N_{max} \leq \Delta$, i.e. a new window of TCP packets is entirely transferred to the station before this goes back to sleep. Under this condition $cwnd_{max} = N_{max}$, and the RTT experienced by the connection is $RTT_{eff} = \lceil \frac{RTT_{base}}{BI} \rceil BI$. This scenario is illustrated in the left hand side of Figure 7(a), from where it is easy to see that the throughput of the TCP connection in congestion avoidance can be computed as:

$$Thr = \frac{1}{2}\frac{cwnd_{max} + cwnd_{min}}{RTT_{eff}} = \frac{3}{4}\frac{N_{max}}{\lceil \frac{RTT_{base}}{BI} \rceil BI} \qquad (3)$$

[12] Notice, that if $R_{Wi-Fi} > R_{DSL_{DL}}$ there is always going to be such maximum.



(a) Validation of model accuracy

(b) Dynamics of the initial Slow Start and Fast Recovery phases

Fig. 8 TCP dynamics.

Consider now that $N_{max} > \Delta$ and $RTT_{base} \leq BI$. The dynamics in this case are again illustrated in Figure 7(a). Notice in the figure how when $cwnd$ grows above Δ, the station goes back to sleep before retrieving the complete TCP window, and a fraction of this TCP window remains in the AP and is transmitted in a subsequent Beacon (see third Beacon in Figure 7(a)). Therefore, when $cwnd > \Delta$, the TCP connection effectively transfers $cwnd + \Delta$ packets every $2BI$. Instead, when $cwnd < \Delta$ the connection transfers a $cwnd$ worth of packets every BI. Finally, notice that the maximum value of $cwnd$ is $cwnd_{max} = N_{max}$, and $cwnd_{min} = \frac{N_{max}}{2}$. Hence, after the appropriate algebraic manipulation, the throughput of the TCP connection in congestion avoidance can be computed in this case as:

$$Thr = \begin{cases} \frac{\Delta + \frac{3N_{max}}{4}}{2BI} & cwnd_{min} \geq \Delta \\ \frac{1}{2}\frac{N_{max} + (3-\alpha)\Delta + (1-\alpha)}{2BI} & cwnd_{min} < \Delta \end{cases} \quad (4)$$

Where $\alpha = \frac{\Delta - cwnd_{min}}{cwnd_{max} - cwnd_{min}}$.

Finally, let us consider that $N_{max} > \Delta$ and $RTT_{base} > BI$. The dynamics in this case are illustrated in Figure 7(b), where it can be seen that every time that $cwnd$ reaches a value multiple of Δ, i.e. $cwnd = k\Delta$ with $k \geq 1$, packets start to be transmitted in a subsequent Beacon. Thus, only when $cwnd > \lceil \frac{RTT_{base}}{BI} \rceil \Delta$ the amount of packets hitting the DSLAM after a Beacon starts growing above Δ until N_{max}. Therefore, in this case $cwnd_{max} = N_{max} + \lceil \frac{RTT_{base}}{BI} \rceil \Delta$, and the effective RTT experienced by the TCP connection is $RTT_{eff} = (1 + \lceil \frac{RTT_{base}}{BI} \rceil) BI$. Recalling that $N_{max} > \Delta$ and $RTT_{base} > BI$, it can be derived that $cwnd_{min} > \Delta$, therefore:

$$Thr = \frac{3}{4} \frac{N_{max} + \lceil \frac{RTT_{base}}{BI} \rceil \Delta}{(1 + \lceil \frac{RTT_{base}}{BI} \rceil) BI} \quad (5)$$

We can now understand the reasons behind the ripple effect depicted in Figure 6(a). We have shown how small changes in the connection parameters, e.g. RTT_{base}, can result in a different phenomena dominating the dynamics of RTT_{eff} and $cwnd$, and in a completely different throughput, as identified in Equations 3, 4 and 5. In order to validate the presented model, we compare in Figure 8(a) the throughput predicted by the model against the throughput obtained in simulations[13]. It is also interesting to notice in Figure 8(a), how as we had previously predicted, when the upstream DSL bandwidth increases ($R_{DSL_{UL}} = 3Mbps$), the connection throughput degrades due to an increase in ACK compression.

Regarding energy, it is worth to notice looking at the middle part of Figure 6(b), that an always active configuration severely penalizes energy consumption when RTT_{base} increases, because TCP can not keep the DSL line always busy and hence the station is often inactive in this case.

2.2.3 Wi-Fi Bottleneck scenario

Finally, the lower graphs in Figures 6(a) and 6(b) depict respectively for our RTT experiment the throughput and energy performance in the *Wi-Fi Bottleneck* scenario. We can see that, like in the *Fast DSL* scenario, a ripple effect appears in throughput, but this time only in the case of the *Slow Wi-Fi* configuration.

The analysis presented in the previous section can also be used in this case to understand the TCP dynamics. For instance, recall from the previous section that M packets must be buffered at the AP before a Beacon frame is sent, where $M = \frac{BI - BI \mod RTT_{base}}{T_{arrival}}$ and $T_{arrival}$ is the interarrival time of packets arriving at the bottleneck. In the *Fast DSL* scenario we had that $T_{arrival} = \frac{Size\ TCP\ ACK}{R_{DSL_{UL}}}$, however in this scenario packets may hit the bottleneck (the AP) with smaller interarrival times. For instance, as previously explained, when the *Fast Wi-Fi* configuration is used the Wi-Fi station transmits TCP ACKs within a TXOP which results in packets arriving at the AP with small $T_{arrival}$ times. Thus, if $T_{arrival}$ is small enough so that the number of TCP packets arriving at the AP before the Beacon, M, grows above Q_{AP},

[13] In our simulations, we choose the *Fast Wi-Fi* configuration, because it better fulfills condition ii. of our model, and consider only the congestion avoidance phase of the connection.

a drop will occur at the AP's power save buffer which limits the maximum $cwnd$ to $cwnd_{max} < Q_{AP}$. Indeed, this is what happens in the case of the *Fast Wi-Fi* configuration, and therefore the achieved throughput (during the congestion avoidance phase) in this case is free of ripples and is described by:

$$Thr = \frac{3}{4}\frac{Q_{AP}}{\lceil \frac{RTT_{base}}{BI} \rceil BI} \quad (6)$$

The behavior is different in the *Slow Wi-Fi* configuration, where TCP packets do not arrive at the AP so close to each other, i.e $T_{arrival}$ is bigger, and therefore the ripple effect arises. For instance, when $RTT_{base} = 110ms$ a new window of TCP data arrives at the AP approximately 90ms before the next Beacon time, which results in the AP having to be able to buffer 90ms worth of TCP Data arrivals to avoid a packet drop. This is not possible given the considered buffer of $Q_{AP} = 100$ packets, and therefore $cwnd_{max} = Q_{AP}$ in this case. However, when $RTT_{base} = 150ms$ the AP only has to be able to buffer 50ms worth of arrivals to avoid a packet drop, and so $cwnd_{max}$ can grow above Q_{AP}, obtaining a higher throughput in this case.

Regarding energy, a similar behavior like in the *Fast DSL* scenario is observed in this case.

Finally, we would like to notice that in this study we have only reported on the performance of downlink file tansfers, i.e. from the file server to the Wi-Fi station. We have however also studied how uplink file tansfers behave, observing that, given the limited upstream bandwidths in xDSL, *Wi-Fi PSM* usually delivers a fairly good throughput/energy tradeoff for uplink connections.

3 BA-TA: An Adaptive Triggering algorithm for TCP over Wi-Fi PSM

As we have seen in Section 2, the performance of TCP over *Wi-Fi PSM* critically depends on the characteristics of the network behind the AP. Therefore, our goal in this section is to design an adaptive algorithm running in a Wi-Fi station, that will tune the operation of the Wi-Fi power saving protocol according to the bottleneck bandwidth experienced by a TCP connection. Hereafter, we refer to this algorithm as the *Bottleneck Aware - Triggering Algorithm* (BA-TA).

We have seen in Section 2 that the default trigger interval used by *Wi-Fi PSM* (100ms), can severely degrade the performance of a TCP connection, specially as the capacity of the bottleneck increases. Therefore, the intuition behind BA-TA is to adapt the trigger interval in the following way. If the connection's bottleneck has a *small* bandwidth, the Wi-Fi station should use large trigger intervals which do not decrease performance and are energy efficient. On the other hand, if the connection's bottleneck has a *large* bandwidth, the Wi-Fi station should use short trigger intervals (and potentially switch to active mode), since this is the most efficient configuration in this case.

We establish the following constraints in the design of BA-TA:

- The algorithm has to run in the Wi-Fi station without any support from the Access Point, and should not require modifications to existing standards. This will allow a Wi-Fi station to immediately use BA-TA in currently deployed Wi-Fi networks.
- The algorithm should run entirely at Layer two. This will allow to easily re-use BA-TA in any device with a Wi-Fi interface.

Notice that the focus of BA-TA is to enhance the performance energy trade-off of long lived TCP connections over Wi-Fi power saving protocols. However, BA-TA should also deliver a good performance energy trade-off with elastic applications like Web browsing and also with the TCP Streaming applications widely used by popular content providers [16]. Inelastic real-time traffic though, like VoIP, is considered out of the scope of BA-TA. Therefore, BA-TA could be deployed at layer two in a Wi-Fi station that implements the classification mechanisms defined in 802.11e [3] in order to segregate elastic traffic, for which BA-TA would be applied, from inelastic real-time traffic, for which other algorithms, like those proposed in [11] and [12], could be applied.

BA-TA is composed of two independent modules. A first module that *estimates* the peak rate of the TCP connection's bottleneck, and a second module that given the previous estimation selects an appropriate trigger interval in the power saving protocol. These two modules are described next.

3.1 A bottleneck bandwidth estimation algorithm

The goal of this algorithm is to estimate in a Wi-Fi station in power save mode, the *peak* bandwidth of a TCP connection's bottleneck, e.g. 1 Mbps and 16 Mbps in our *Slow DSL* and *Fast DSL* scenarios.

The intuition behind our peak bandwidth estimation algorithm is described in Figure 9. This figure depicts the occupancy of the bottleneck line behind the AP, e.g. the DSL line in Figure 1, between two (not necessarily consecutive) trigger frames sent by the station in power save mode. Notice that when a service period completes[14], the Wi-Fi station knows that the buffer in the AP is empty. Therefore, the station can use the amount of data received between these two trigger frames in order to estimate the bottleneck's peak bandwidth in the following way. Consider the bottleneck

[14] In U-APSD a service period completes when the AP sets the EOSP bit to 1 in a transmitted frame, indicating to the station that it has no more buffered frames.

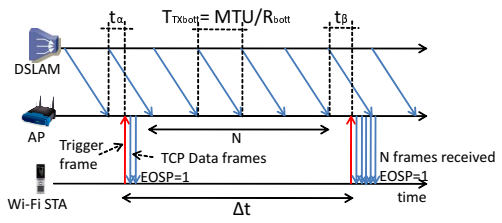


Fig. 9 Intuition behind the Peak Bandwidth Estimation Algorithm.

line to be always busy between the two triggers sent by the station. We have in this case that $N \times T_{TX_{bott}} = \Delta_t + t_\alpha - t_\beta$, where N is the number of frames received by the station between the two triggers, and $T_{TX_{bott}}$, t_α and t_β are defined in Figure 9. Notice that $0 \leq t_\alpha, t_\beta \leq T_{TX_{bott}}$.

Considering now that $T_{TX_{bott}} = \frac{MTU}{R_{bott}}$, where MTU is the size in bits of a packet transmitted over the bottleneck link and R_{bott} is the bottleneck peak transmission rate that we want to estimate, the following lower and upper bounds on R_{bott} can be derived:

$$T_{TX_{bott}} \geq \frac{\Delta_t}{N+1} \rightarrow R_{bott} \leq \frac{(N+1)MTU}{\Delta_t} \quad (7)$$

$$T_{TX_{bott}} \leq \frac{\Delta_t}{N-1} \rightarrow R_{bott} \geq \frac{(N-1)MTU}{\Delta_t} \quad (8)$$

Notice that the upper bound on R_{bott} only holds if the bottleneck line is all the time busy between the two triggers sent by the station. However, the lower bound on R_{bott} holds true even if the bottleneck is not always busy.

Thus, our peak rate estimation algorithm works in the following way. When a service period completes and more than T_{update} seconds have past from the last algorithm update, where T_{update} is an algorithm parameter, BA-TA executes the procedure described in Algorithm 1, which records a *lower bound* estimation on the bottleneck's peak rate as described in Equation 8. In addition, in order to get as close as possible to the true R_{bott}, Algorithm 1 continuously updates the bottleneck rate estimation to the highest observed R_{bott} lower bound (line 7 in Algorithm 1).

There is an extra issue to be considered in Algorithm 1. If, as it will be explained in the next section, the Wi-Fi station switches from power save mode to active mode, then the AP will start transmitting the station's remaining buffered packets at the rate of the Wi-Fi network, which can be above the bottleneck rate. In order to avoid overestimating the bottleneck rate in this case, our peak rate estimation algorithm always uses as previous reference a time where the Wi-Fi station was in power save mode, as can be seen in line 9 in Algorithm 1.

Algorithm 1: Routine executed when receiving EOSP=1, if $t_{now} > t\_last_{peak} + T_{update}$

1 – *Variable definitions*
2 $rcvd\_data_{peak} \leftarrow$ Amount of data (bits) received since last update
3 $t\_last_{peak} \leftarrow$ Time of last update
4 $first\_pkt\_size \leftarrow$ Size of the first packet (bits) received in the last interval
5 – *estimate_peak_rate()*
6 $lower\_bound \leftarrow \frac{rcvd\_data_{peak} - first\_pkt\_size}{t_{now} - t\_last_{peak}}$
7 $peak\_rate \leftarrow \max\{peak\_rate, lower\_bound\}$
8 **if** $!ActiveMode$ **then**
9 $\quad t\_last_{peak} \leftarrow t_{now}$
10 $\quad rcvd\_data_{peak} \leftarrow 0$

3.2 A trigger interval adaptation algorithm

We now describe how BA-TA controls the trigger interval used by a Wi-Fi station in power save mode. Our trigger adaptation algorithm is essentially a *proportional controller* [26], that controls the station's trigger interval in order to stabilize the connection's throughput around a configurable operation point. This operation point is an input to the algorithm, specified in terms of the desired ratio between the connection's throughput and the bottleneck peak rate, i.e. $0 < ratio_{min} < 1$. Our detailed algorithm is described in Algorithm 2, and is summarized next.

A Wi-Fi station implementing BA-TA maintains two estimates: i) a $peak\_rate$ estimate provided by our bottleneck bandwidth estimation algorithm, and ii) an estimate of the instantaneous connection throughput, $instant\_thr$. This estimate is updated using an Exponentially Weighted Moving Average (EWMA) filter with weighting factor α (line 19 in Algorithm 2).

A BA-TA station executes Algorithm 2 at regular intervals (every T_{update}), in order to update the $peak\_rate$[15] and $instant\_thr$ estimates. In addition, after $count_{max}$ consecutive estimation updates, BA-TA evaluates whether the currently used trigger interval is appropriate. Notice that the configurable $count_{max}$ parameter enables BA-TA to use different intervals to update the throughput estimates, and the used trigger interval.

In order to update the used trigger interval, BA-TA uses a proportional law. For this purpose, it computes $ratio = \frac{instant\_thr}{peak\_rate}$ as the currently used fraction of bottleneck bandwidth (line 20), and compares it to the input value $ratio_{min}$. Specifically, BA-TA updates the used trigger interval in the following way (line 31):

$$interval(n+1) = (1 + G(ratio - ratio_{min}))interval(n)$$

[15] Notice that the $peak\_rate$ estimate is only updated here when the station is in active mode, because in this case the station does not generate triggers and does not receive frames with EOSP=1.

(9)

Where G is the gain used in the control law, and $ratio - ratio_{min}$ can be understood as the current error incurred by the controller. Notice in Equation 9 that when $ratio$ is below $ratio_{min}$, the trigger interval decreases, which reduces the RTT experienced by the TCP connection and should improve throughput. In addition, when $ratio$ is above $ratio_{min}$ the trigger interval increases, which is more energy efficient. In the Appendix it is formally shown that this controller converges with no steady state error when $0 < G < \frac{2}{ratio_{min}}$. In addition, the trigger interval is only allowed to vary between a maximum and minimum trigger intervals, int_{min} and int_{max} (line 42), which are also inputs to the algorithm that can be configured according to the characteristics of each particular Wi-Fi chipset.

Having described the basic operation of BA-TA, there are several issues that have to be considered in a practical implementation. The first of these issues, is how to appropriately switch the station between active mode and power save mode. Notice that as seen in Section 2, being in active mode can potentially result in a big energy waste, therefore BA-TA should configure a station in active mode only when there is a clear throughput gain in doing so. BA-TA uses the following heuristic for this purpose:

- *Enter active mode*: If the interval selected in Equation 9 is below the minimum allowed interval (int_{min}), and the average ratio increase experienced when the station is in active mode, $\hat{\Delta}_{ratio_{AM}}$, is significantly[16] above the average ratio increase when the station is in power save mode, $\hat{\Delta}_{ratio_{PS}}$, (line 36).
- *Leave active mode*: Under two conditions. First, if $ratio$ is below $ratio_{min}$, but being in active mode is not effective, i.e. $ratio$ does not increase in at least a minimum amount (γ) (line 40). Second, when $ratio$ is above $ratio_{min}$, i.e. we have achieved the desired throughput, and the station is not saturating the network, i.e. $util_{last} \leq util_{min}$ (line 33). Recall from Section 2 that staying in active mode is effective while the station saturates the Wi-Fi network. Therefore, while in active mode a BA-TA station tracks whether its traffic is saturating the Wi-Fi network by measuring network utilization as the fraction of *used* time during the last T_{update} period (line 16). For further details on how to measure network utilization in a Wi-Fi station, the interested reader is referred to [24].

The $\hat{\Delta}_{ratio_{AM}}$ and $\hat{\Delta}_{ratio_{PS}}$ estimates are also obtained with an EWMA filter of weighting factor α, and allow BA-TA to measure the effect that being in active mode or in

[16] We force $\hat{\Delta}_{ratio_{AM}} > \hat{\Delta}_{ratio_{PS}} + |\hat{\Delta}_{ratio_{PS}}|$ as a heuristic way to ensure that switching to active mode is indeed significantly better than operating in power save mode (line 37).

Algorithm 2: BA-TA executed every T_{update}

1 – *Algorithm Parameters*
2 $T_{update} \leftarrow$ Interval between algorithm executions
3 $ratio_{min} \leftarrow$ Desired operation point
4 $int_{min}, int_{max} \leftarrow$ Minimum and maximum trigger intervals
5 $count_{max} \leftarrow$ Consecutive events between updates
6 $N_{max} \leftarrow$ Maximum number of attemtps
7 $G \leftarrow$ Controller gain
8 $\alpha \leftarrow$ Weights used in the EWMA averages
9 $\gamma \leftarrow$ Comparison margin on Δ_{ratio}
10 $n\_trig_{max}, step_{min} \leftarrow$ Avoid too agressive intervals
11 – *Variable definitions*
12 $rcvd\_data_{int} \leftarrow$ Amount of data (bits) received since last update
13 $t\_last_{int} \leftarrow$ Last time the algorithm executed
14 – *update_trigger_interval()*
15 **if** $ActiveMode$ **then**
16 $\quad util_{last} \leftarrow \frac{TX\_time+RX\_time+Backoff\_Time}{T_{update}}$
17 $\quad$ **if** $t_{now} > t\_last_{peak} + T_{update}$ **then**
18 $\quad\quad$ estimate_peak_rate
19 $instant\_thr \leftarrow \alpha \times instant\_thr + (1-\alpha)\frac{rcvd\_data_{int}}{t_{now}-t\_last_{int}}$
20 $ratio \leftarrow \min\{\frac{instant\_thr}{peak\_rate}, 1\}$
21 **if** $peak\_rate, instant\_thr > 0$ *and* $n\_trigs < n\_trig_{max}$ **then**
22 $\quad count \leftarrow count + 1$
23 $\quad$ **if** $ratio - ratio_{last} < -2\gamma$ **then**
24 $\quad\quad count \leftarrow count_{max}$
25 $\quad$ **if** $ActiveMode$ **then**
26 $\quad\quad \hat{\Delta}_{ratio_{AM}} \leftarrow \alpha\hat{\Delta}_{ratio_{AM}} + (1-\alpha)\Delta_{ratio}$
27 $\quad$ **else**
28 $\quad\quad \hat{\Delta}_{ratio_{PS}} \leftarrow \alpha\hat{\Delta}_{ratio_{PS}} + (1-\alpha)\Delta_{ratio}$
29 $\quad$ **if** $count = count_{max}$ **then**
30 $\quad\quad count \leftarrow 0$, $\Delta_{ratio} \leftarrow ratio - ratio_{last}$, $ratio_{last} \leftarrow ratio$
31 $\quad\quad interval \leftarrow (1 + G(ratio - ratio_{min}))interval$
32 $\quad\quad$ **if** $ratio \geq ratio_{min}$ **then**
33 $\quad\quad\quad$ **if** $ActiveMode = true$ *and* $util_{last} \leq util_{min}$ **then**
34 $\quad\quad\quad\quad$ Leave active mode
35 $\quad\quad$ **else**
36 $\quad\quad\quad$ **if** $ActiveMode = false$ *and* $interval < int_{min}$ *and* $rcvd\_data_{int} > 0$ **then**
37 $\quad\quad\quad\quad$ **if** $\hat{\Delta}_{ratio_{AM}} > \hat{\Delta}_{ratio_{PS}} + |\hat{\Delta}_{ratio_{PS}}|$ *or* $n_{AM} = N_{max} - 1$ **then**
38 $\quad\quad\quad\quad\quad$ Enter active mode
39 $\quad\quad\quad\quad n_{AM} \leftarrow (n_{AM} + 1) \mod N_{max}$
40 $\quad\quad\quad$ **if** $ActiveMode = true$ *and* $\Delta_{ratio} < \gamma$ **then**
41 $\quad\quad\quad\quad$ Leave active mode
42 $\quad\quad interval \leftarrow \max\{\min\{interval, int_{max}\}, int_{min}\}$
43 **else if** $rcvd\_data_{int} = 0$ *or* $n\_trigs \geq n\_trig_{max}$ **then**
44 $\quad ratio_{last} \leftarrow ratio$
45 $\quad$ **if** $ActiveMode$ **then**
46 $\quad\quad$ Leave active mode
47 $\quad$ **else**
48 $\quad\quad interval \leftarrow \min\{interval + step_{min}, int_{max}\}$
49 **else**
50 $\quad interval \leftarrow int_{max}$
51 $n\_trigs \leftarrow 0$, $rcvd\_data_{int} \leftarrow 0$, $t\_last_{int} \leftarrow t_{now}$
52 $TX\_time, RX\_time, Backoff\_Time \leftarrow 0$

power save mode has on the experienced throughput (lines 25 to 28). Notice however, that the previous estimates may contain innacuracies. For instance, if while being in active mode TCP suffers a loss and decreases its congestion window, BA-TA may observe a decrease in throughput and may wrongly believe that being in active mode is not efficient. Thus, in order to avoid BA-TA from getting permanently stalled in a sub-optimal configuration, a switch to active mode is always allowed after N_{max} refrained attempts (line 37). In addition, if a sudden[17] decrease in $ratio$ is detected, BA-TA quickly updates the used trigger interval (line 23).

Finally, another issue to consider in BA-TA is the fact that certain applications, like Web, may simply not offer enough load to achieve the desired bottleneck link utilization, i.e. maintain $ratio$ above $ratio_{min}$. The previously described logic would in this case drive a Wi-Fi station to use aggressive trigger intervals, or even switch to active mode, which is undesirable from an energy point of view. In order to prevent the previous from happening, BA-TA limits the maximum number of triggers that a station can send to the AP without receiving any data in return (n_trig_{max}). If that limit is surpassed, BA-TA considers that the used trigger interval is too small and immediately increases it by a preconfigured amount ($step_{min}$) (lines 43 to 48). Additional optimizations for TCP Streaming will be discussed in the next section.

4 BA-TA Performance Evaluation

In this section we evaluate the performance of BA-TA and study the effect of its configurable parameters. This section is divided in three sub-sections: i) a first sub-section illustrating the basic dynamics of BA-TA, ii) a second subsection studying how to configure the different BA-TA parameters, and iii) a final sub-section that extensively evaluates the performance of BA-TA compared to other algorithms in the state of the art. For the purpose of this evaluation, we consider only the *Fast Wi-Fi* configuration defined in Section 2. This configuration poses a bigger challenge to BA-TA because, as previously seen, it delivers the best energy efficiency, but results in the highest throughput degradation when used with *Wi-Fi PSM*.

Unless otherwise stated, BA-TA is configured in the following way. The maximum and minimum trigger intervals are set to $int_{max} = 100ms$, which is equivalent to the operation of *Wi-Fi PSM*, and $int_{min} = 20ms$, which is a trigger interval commonly used by current Wi-Fi chipsets with Voice traffic [5]. The update interval, T_{update}, is set to $100ms$, which is a convenient value that allows to synchronize the operation of BA-TA with other periodic operations done by a station, like receiving the Beacon frame. The parameters α and γ are empirically set by testing the behavior of BA-TA in our scenarios of interest. The weight α used in the EWMA filters is set to $\alpha = 0.9$. Finally, The parameter γ is set to $\gamma = 0.1$, which results in BA-TA requiring a 10% increase in $ratio$ when being in active mode (line 40 in Algorithm 2), and forcing a sudden interval update if $ratio$ decreases by more than a 20% (line 23 in Algorithm 2).

The effect of the rest of BA-TA parameters, i.e. $ratio_{min}$, the controller gain G, the update interval $count_{max}$, the parameter N_{max}, and the parameters n_trig_{max} and $step_{min}$, will be studied in this section.

4.1 Basic Dynamics of BA-TA

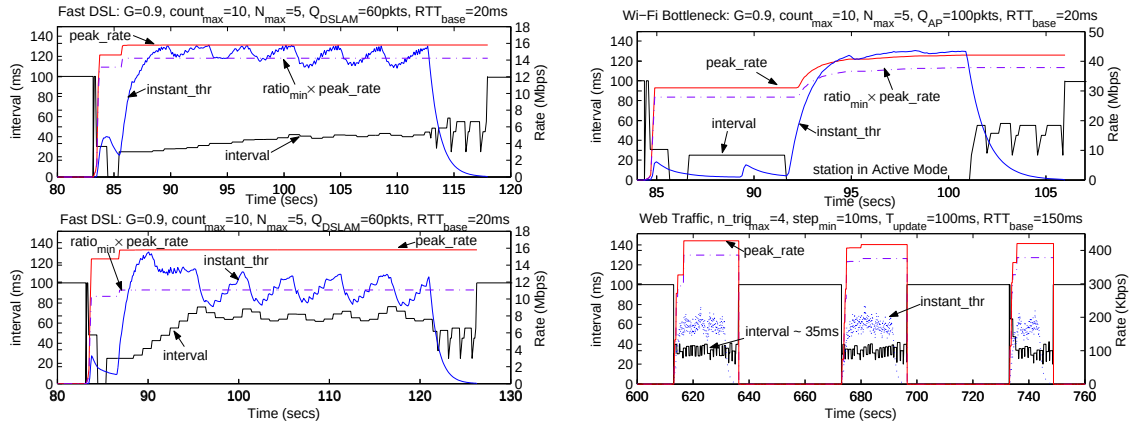
Figure 10 depicts the dynamics of BA-TA when a Wi-Fi station retrieves a 50MB file from the Internet. We start our analysis by looking at Figure 10(a) that illustrates the BA-TA dynamics in the *Fast DSL* scenario with $ratio_{min} = 0.9$ (upper graph) and $ratio_{min} = 0.7$ (lower graph). The value of the rest of configuration parameters is included on the top of each figure. Notice that a double y-axis is used in the figure, where the left y-axis plots the trigger interval used by BA-TA while retrieving the file (black line), and the right y-axis plots the bottleneck rate (red line) and the instantaneous throughput (blue line) estimated by BA-TA.

We can see in Figure 10(a) how BA-TA operates with an interval of 100ms until the file transfer begins. From that point on, BA-TA starts adjusting the trigger interval (black line) in order to drive the obtained throughput (blue line) around the desired utilization level, i.e. $ratio_{min} \times peak\_rate$ (purple dashed line). After a transient period, BA-TA converges around the configured throughput, requiring, as expected, a smaller trigger interval for $ratio_{min} = 0.9$ (upper graph) than for $ratio_{min} = 0.7$ (lower graph). In the Appendix it is formally shown that BA-TA converges to a trigger interval that is proportional to $\frac{1}{ratio_{min}}$. After the file transfer completes, BA-TA switches off and the station returns to wake up only at every Beacon frame.

The upper part of Figure 10(b) illustrates now how BA-TA behaves in the *Wi-Fi Bottleneck* scenario. We can see that after an initial transient period, TCP manages to saturate the Wi-Fi network and therefore BA-TA configures the station to operate all the time in active mode (signaled in Figure 10(b) with the black line being zero), which as seen in Section 2 is the most efficient operation mode in this case.

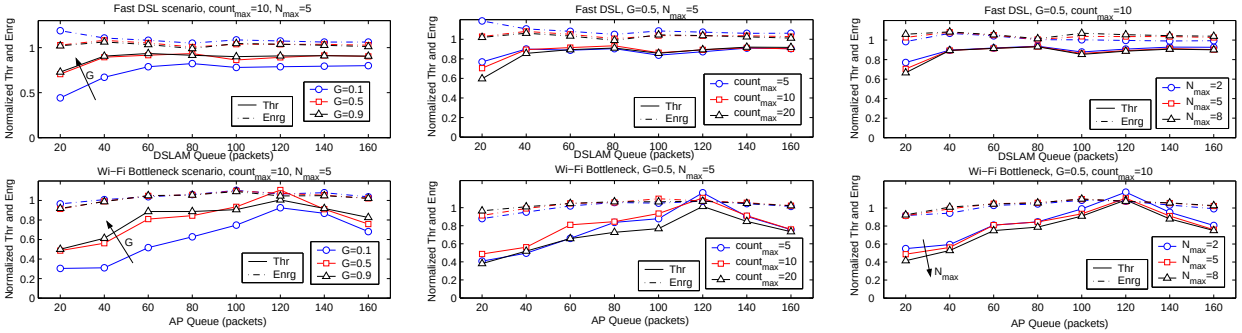
For the sake of space we do not report the dynamics of BA-TA in the *Slow DSL* scenario, where BA-TA delivered the bottleneck throughput using the maximum trigger interval, i.e. $int_{max} = 100ms$. Instead, the lower part of Figure 10(b) depicts the dynamics of BA-TA with Web traffic. Web traffic is modeled using HTTP 1.1 where the Wi-Fi station

[17] Defined as a decrease in $ratio$ above 2γ.



(a) *Fast DSL*, $ratio_{min} = 0.9$ (upper graph), $ratio_{min} = 0.7$ (lower graph) (b) *Wi-Fi Bottleneck* (upper graph), Web traffic (lower graph)

Fig. 10 Basic BA-TA Dynamics. The figures represent the variation over time of the BA-TA used interval (black line), the $instant\_thr$ estimation (blue line), and the $peak\_rate$ estimation (red line).



(a) *Exp. 1*: Effect of G (b) *Exp. 2*: Effect of $count_{max}$ (c) *Exp. 3*: Effect of N_{max}

Fig. 11 Effect of BA-TA parameters. The upper graph depicts the *Fast DSL* scenario, and the lower graph the *Wi-Fi Bottleneck* scenario. In addition, notice that the depicted throughput and energy metrics are normalized. A value of $ratio_{min} = 0.9$ is used in these experiments.

establishes a persistent TCP connection with the Web server in Figure 1, and retrieves all the objects in a Web page in a sequential way. A Web page is modeled according to the statistics provided in [22].

The figure depicts the interval used by BA-TA while retrieving three Web pages, where the value of RTT_{base} between the AP and the Web server in Figure 1 is set to $150ms$, i.e. there is a latency of at least $150ms$ between two consecutive Web page objects retrieved from the Web server. Interestingly, the interval used by BA-TA in this case converges to a value around $35ms$. The reason is the following. We configured in this scenario $n\_trig_{max} = 4$ and $step_{min} = 10ms$, recall now that if $\frac{T_{update}}{n\_trig_{max}} \leq interval \leq \frac{T_{update}}{n\_trig_{max}-1}$, and the application does not offer enough load, e.g. Web, BA-TA may receive n_trig_{max} empty triggers within a T_{update} interval and will increase the used trigger interval by $step_{min}$ (line 48 in Algorithm 2). Therefore, the used interval must converge to a value around $\frac{T_{update}}{n\_trig_{max}}$ and $\frac{T_{update}}{n\_trig_{max}-1} + step_{min}$. The previous fact can be used as a configuration guideline for the parameters n_trig_{max} and $step_{min}$, which we hereafter configure as $n\_trig_{max} = 4$ and $step_{min} = 10ms$.

4.2 Effect of BA-TA Parameters

We divide the study of the effect of the BA-TA parameters in two different parts. First, we benchmark the effect of the BA-TA parameters when a station retrieves a file from the Internet. Second, we study the effect of these parameters in a scenario where multiple flows share the bottleneck link.

4.2.1 Effects on a Single Flow

Figure 11 depicts the results of a set of experiments, where the evaluation methodology described in Section 2 is ap-

plied, while varying different BA-TA parameters. In particular, the following three experiments are shown:

- *Exp. 1*: Configure $G = 0.1, 0.5, 0.9$, which according to the analysis in the Appendix should guarantee monotonic convergence, while fixing $count_{max} = 10$ and $N_{max} = 5$ (Figure 11(a)).
- *Exp. 2*: Configure $count_{max} = 5, 10, 20$ while fixing $G = 0.5$, and $N_{max} = 5$ (Figure 11(b)).
- *Exp. 3*: Configure $N_{max} = 2, 5, 8$ while fixing $G = 0.5$, and $count_{max} = 10$ (Figure 11(c)).

For the sake of space Figure 11 only depicts the results of our *Buffer Experiment* for the *Fast DSL* (upper graph) and *Wi-Fi bottleneck* (lower graph) scenarios. In addition, the depicted throughput (solid lines) and energy (dashed lines) performance are normalized to the maximum throughput and minimum energy achieved by the algorithms considered in Section 2 for the same experiment, i.e:

$$\hat{Thr} = \frac{Thr_{BA-TA}}{max\{Thr_{Active}, Thr_{PSM}\}} \quad (10)$$

$$\hat{Enrg} = \frac{min\{Enrg_{Active}, Enrg_{PSM}\}}{Enrg_{BATA}} \quad (11)$$

Thus, ideally BA-TA would provide a performance for both indexes with a value as close as possible to 1, or bigger.

The following guidelines are identified from Figure 11. First, a too small value of the controller gain, $G < 0.5$, degrades throughput performance because the transient period takes too long in this case. Second, big values of N_{max} or $count_{max}$ slightly degrade the throughput achieved by BA-TA in the *Wi-Fi Bottleneck* scenario. The reason is that BA-TA often switches to active mode in this case, and if it gets stalled in a suboptimal configuration, due to a misestimation of $\hat{\Delta}_{ratio_{AM}}$ or $\hat{\Delta}_{ratio_{PS}}$, a time equal to $N_{max} \times count_{max} \times T_{update}$ seconds is required to recover. Thus, smaller values of N_{max} and $count_{max}$ allow BA-TA to recover earlier. Regarding energy, BA-TA is always equal or better than the most energy efficient algorithm.

The trends exhibited in this section by the different BA-TA parameters have been confirmed in other experiments where we considered multiple concurrent flows. For the sake of space though, we do not include these results here.

4.2.2 Effects on Sharing Scenarios

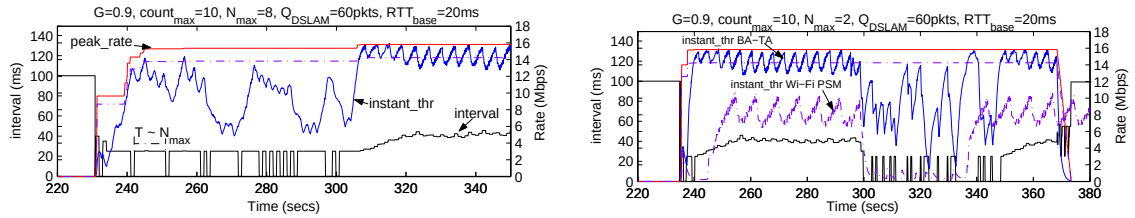
We evaluate now the performance of BA-TA in scenarios where a Wi-Fi station running BA-TA shares the bottleneck link with another station, which we configure to be in active mode since this should be the most common situation in practice. The goal of this section is to show that a station running BA-TA is able to share a bottleck link with an active mode station, without any station starving.

Figure 12(a) depicts an example where a station using BA-TA and a station in active mode share, between 230 and 300 seconds, the DSL link in the *Fast DSL* scenario. Notice that in this case, BA-TA's proportional law drives the power saving station towards using small trigger intervals or switching to active mode, because the power saving station's throughput can not reach the configured fraction of the bottleneck peak bandwidth (because the link is shared). However, being in active mode is not efficient in this case, $\hat{\Delta}_{ratio_{AM}}$ is not significantly bigger than $\hat{\Delta}_{ratio_{PS}}$, and BA-TA switches the power saving station between active mode and power save mode. Notice that it is possible to control in these sharing scenarios how often a power saving station switches to active mode by adjusting the parameter N_{max}, because BA-TA always enters active mode after N_{max} refrained attempts. Finally, when the flow from the station in active mode completes, after 300 seconds, BA-TA quickly returns to its normal operation.

Figure 12(b) depicts, between 300 and 350 seconds, an effect that occurs when a station in power save mode receives a flow through the DSL link, while another Wi-Fi station in active mode is retrieving a file from an Ethernet network behind the AP. Interestingly, if the power saving station uses *Wi-Fi PSM* (purple line), its throughput reduces to almost zero while the station in active mode is retrieving its file. This unfairness occurs because the flow from the station in active mode saturates the Wi-Fi network. Thus, when the station in power save mode wakes up and sends a trigger to the AP to retrieve its buffered frames, the AP dequeues a frame from the power saving buffer but often can not transmit it because the AP's transmission queues are full with packets from the station in active mode. This unfairness could be avoided by deploying enhanced scheduling algorithms in the AP, however these algorithms are generally not implemented in existing APs. Instead, notice in Figure 12(b) that BA-TA avoids the previous unfairness regardless of the scheduling strategy used in the AP, by maintaining the power saving station most of the time in active mode while the other flow is active.

4.3 BA-TA Extensive Evaluation

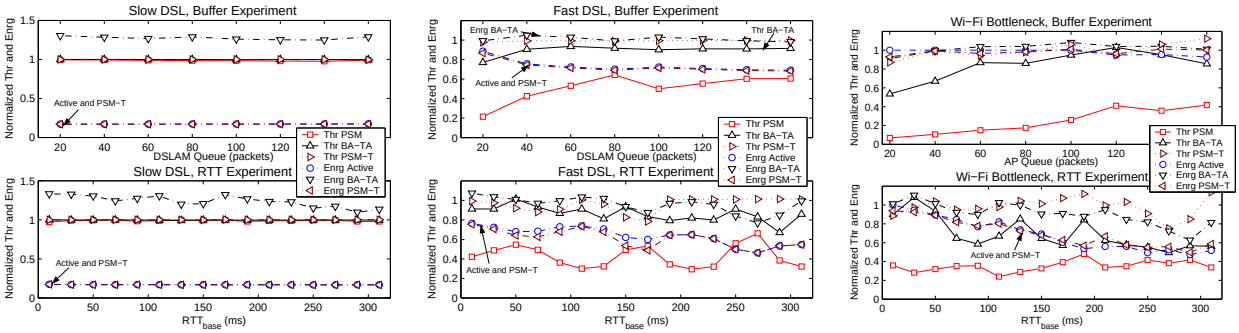
In this section we present and extensive evaluation of the performance of BA-TA. This section is composed of a first subsection, where we use the evaluation setup described in Section 2 in order to benchmark the performance of BA-TA compared to other algorithms in the state of the art. In addition, in a second subsection, we study the performance of BA-TA as the number of stations contending for the wireless medium increases, and in a third subsection we study the performance of BA-TA with TCP Streaming and Web browsing.



(a) A Wi-Fi PSM flow and an Active Mode flow through 16Mbps DSL link with $N_{max} = 8$.

(b) A Wi-Fi PSM flow through 16Mbps DSL and an Active Mode flow through 100Mbps Ethernet.

Fig. 12 BA-TA dynamics with multiple flows.



(a) Slow DSL Scenario (b) Fast DSL Scenario (c) Wi-Fi Bottleneck Scenario

Fig. 13 BA-TA Extensive Evaluation. The upper graphs depict the results for the Buffer Experiment, and the lower graphs depict the results for the RTT Experiment. In addition, notice that the depicted throughput and energy metrics are normalized.

Throughout this section and based on the insight obtained in Section 4.2 we configure BA-TA in the following way: $ratio_{min} = 0.9$, $G = 0.9$, $count_{max} = 10$, $N_{max} = 3$, $n\_trig_{max} = 4$ and $step_{min} = 10ms$.

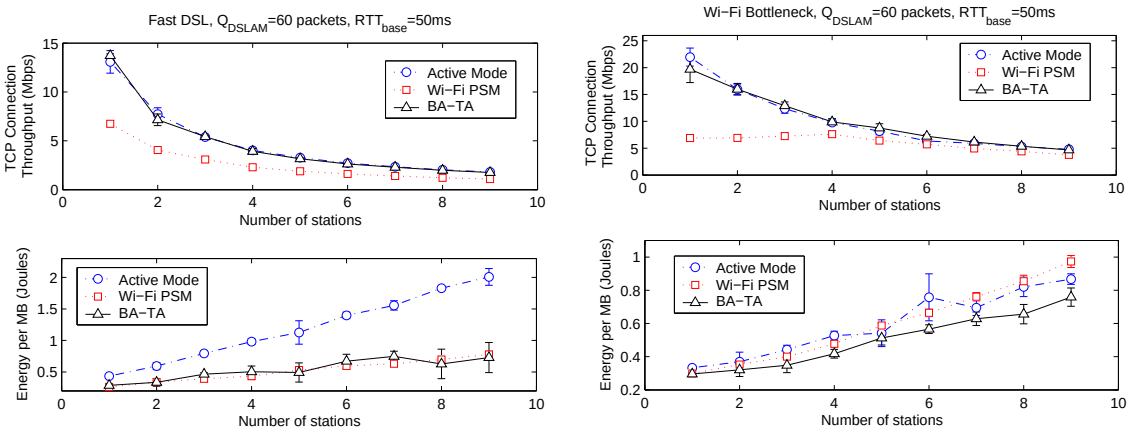
4.3.1 Benchmarking the performance of BA-TA

Figure 13 reports the performance delivered by BA-TA in the *Buffer* and *RTT* experiments defined in Section 2 for the *Slow DSL*, *Fast DSL* and *Wi-Fi Bottleneck* scenarios, where we have used the previously defined normalized throughput and energy metrics, i.e. $\hat{Thr}$ and $\hat{Enrg}$. In addition, to put BA-TA's performance in perspective, the graphs also depict the normalized throughput obtained by the *Wi-Fi PSM* algorithm, the normalized energy obtained by the *Active Mode* algorithm, and the normalized throughput and energy obtained by the *PSM-Throttling* (*PSM-T*) algorithm defined in [10].

As observed in Figure 13, BA-TA delivers a very good performance, i.e. both normalized metrics are close or above 1, for all considered scenarios and parameter range. Only when RTT_{base} increases significantly or very small buffers are considered in the *Wi-Fi Bottleneck* scenario, BA-TA struggles to match the *Active Mode* throughput, but still clearly outperforms *Wi-Fi PSM* in terms of throughput and *Active Mode* in terms of energy[18]. Therefore, we would like to notice that an important property of *BA-TA* is its ability to be energy efficient and to deliver a high throughput when the amount of buffering in the bottleneck is small, which is important to combat *Bufferbloat* [19].

PSM-T delivers in the previous experiments a performance, both in terms of throughput and energy, that is very similar to the one of the *Active Mode* algorithm. The reason for this behavior is the following. The energy efficient operation of *PSM-T* only comes into effect after an initial bandwidth estimation phase where *PSM-T* checks whether the end to end rate delivered by the application is at least half the available end to end bandwidth, i.e. if the application is throttling itself down as many streaming servers do. The previous does not happen when having a file transfer because TCP greedily uses all the available bandwidth, thus in our experiment *PSM-T* did not detect bandwidth throttling and defaulted back to *Active Mode*. Therefore, when considering bandwidth greedy applications, *BA-TA* improves upon *PSM-T* because *BA-TA* obtains enegy savings from the excess bandwidth offered by the last hop (Wi-Fi) over the end to end available bandwidth, while *PSM-T* requires an excess

[18] Notice that with a big RTT_{base}, the parameters n_trig_{max} and $step_{min}$ could be adjusted to trade off throughput and energy, as explained in the case of Web.



(a) Increasing number flows in the *Fast DSL* scenario. (b) Increasing number flows in the *Wi-Fi Bottleneck* scenario.

Fig. 14 Performance of BA-TA, Active Mode and Wi-Fi PSM when increasing the number of competing flows.

of bandwidth not only in the last hop but across all the end to end path in order to achieve energy savings. In section 4.3.3 we will compare the performance of *BA-TA* and *PSM-T* when non bandwidth greedy TCP streaming applications are used.

4.3.2 Effect of congestion on BA-TA

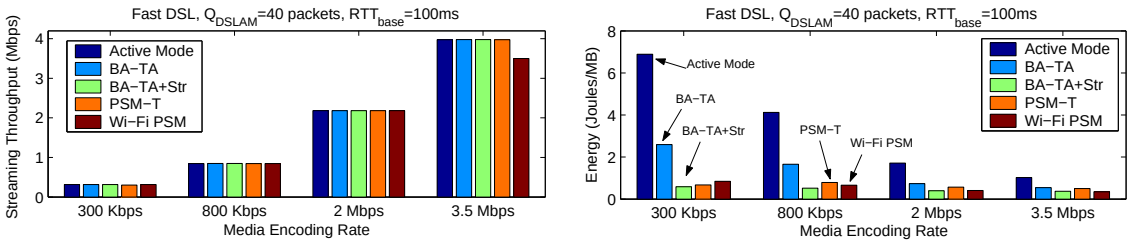
We have so far studied the performance of *BA-TA* compared to other approaches in the state of the art when the number of stations in the network is small (one or two stations). In this section we want to asses whether the performance advantages exhibited by *BA-TA* still hold when we increase the number of competing flows in the Wi-Fi network. For this purpose we consider again our *Fast DSL* and *Wi-Fi Bottleneck scenarios* and analyze the throughput and energy efficiency delivered by the *Active Mode*, *Wi-Fi PSM* and *BA-TA* algorithms when the number of competing stations in the network increases from 1 to 10 (each station retrieving a 50MB file from the file server in Figure 1). Next we report the results of an experiment where the amount of buffering in the DSLAM and in the AP are equal to 60 packets and RTT_{base} is equal to be 50 ms. We have however confirmed the obtained results for other buffer size and RTT_{base} values.

Figure 14(a) depicts the performance obtained in the *Fast DSL* scenario, both in terms of average file transfer throughput (upper subgraph) and average energy consumption (lower subgraph). As clearly seen in the figure, when the number of stations in the network increases, *BA-TA* continues to deliver a throughput very close to the one obtained by *Active Mode*, which is significantly higher than the one delivered by *Wi-Fi PSM* specially when the number of competing flows is small. Notice that when the number of competing flows increases, the bandwidth share of each flow decreases and so the degradation introduced by *Wi-Fi PSM* is not so significant, as we had already seen in the *Slow DSL* scenario. Regarding energy efficiency, we can see how the performance of *BA-TA* compares to the one of *Wi-Fi PSM* and is significantly better than the one of *Active Mode*. The reason for this behavior is that, as previously observed in Figure 12(a), when several flows implementing *BA-TA* share the bottleneck link, no single flow can achieve the bottleneck's peak rate and thus *BA-TA* operates with reduced trigger intervals that increase throughput compared to *Wi-Fi PSM*. In addition, *BA-TA* avoids keeping the station in active mode for a long time and thus, unlike the *Active Mode* algorithm, *BA-TA* turns out to be energy efficient. Notice that as the bandwidth share of each flow decreases, stations spend more time idle without transmitting or receiving and so *Active Mode* results in wasted energy.

Finally, Figure 14(b) depicts the performance obtained when increasing the number of competing flows for the *Wi-Fi Bottleneck* scenario. The performance in this case is similar to the one obtained in the *Fast DSL* scenario, with *BA-TA* achieving a throughput close to the one of *Active Mode* and much higher than the one of *Wi-Fi PSM*. Regarding energy, *BA-TA* continues to be the most energy efficient algorithm, due to its ability of switching a Wi-Fi station into active mode only when by doing so a station can saturate the Wi-Fi network.

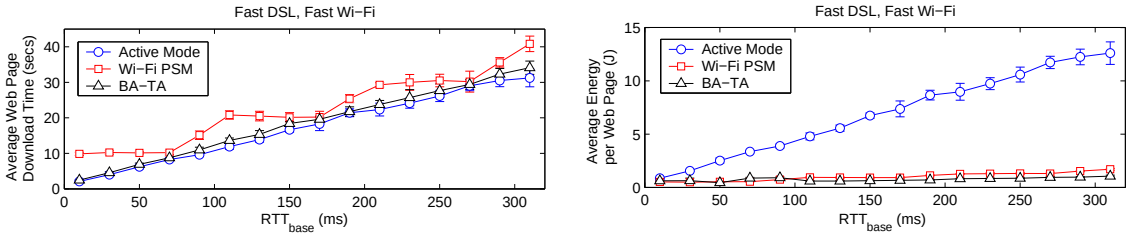
4.3.3 BA-TA Performance with TCP Streaming and Web Browsing

Although the focus of BA-TA has been on long lived TCP transfers, we study in this section how BA-TA performs with TCP Video Streaming and with Web browsing as compared to other algorithms in the state of the art.



(a) Average Streaming Throughput (b) Average Consumed Energy (J/MB)

Fig. 15 BA-TA performance with TCP Streaming. Confidence intervals are not included because they were too small.



(a) Average Web Page download time (b) Average Energy per Web Page

Fig. 16 BA-TA performance with Web Traffic. The Web traffic model described in Section 4.1 is used in this experiment.

In order to study the performance TCP Streaming we model the *block sending* TCP streaming delivery mechanism used by YouTube [16], where an initial portion of the video file is sent at the maximum speed to allow the client to build its playback buffer[19], and for the rest of the video file the application throttles iself down and transmits at the media encoding rate. In particular we will study how the algorithms under study perform both in terms of streaming throughput and energy efficiency when varying the media encoding rate between 300 Kbps and 3.5 Mbps, using values that represent different YouTube profiles [28]. For the sake of space we only report the results obtained in the *Fast DSL* scenario, although similar results were obtained for the *Slow DSL* and *Wi-Fi Bottleneck* scenarios. The algorithms considered in our study are: *Active Mode*, *Wi-Fi PSM*, *BA-TA*, *PSM-T* and *BA-TA+Str*, that is *BA-TA* with an optimization for TCP Streaming that will be described later.

Figure 15(a) illustrates the performance of the algorithms under study, where we can see that all protocols are able to deliver the required streaming rate, except for *Wi-Fi PSM* when the media encoding rate is large. The reason why *Wi-Fi PSM* is not able to deliver the maximum media encoding rate is its higher delay that is always rounded up to a multiple of the Beacon interval. Regarding energy efficiency, Figure 15(b) depicts how the *Active Mode* algorithm is the worst performing algorithm followed by *BA-TA*, while *PSM-T*, *Wi-Fi PSM* and *BA-TA+Str*, perform in a much more energy efficient way. The reason why *BA-TA* penalizes energy efficiency in the case of TCP Streaming is that the application does not offer enough load to saturate the DSL link, however having configured $ratio_{min} = 0.9$ BA-TA often tries to reduce the used trigger interval, and even switches the station to active mode, in order to try to increase the obtained throughput. A simple optimization though, can be built on top of Algorithm 2 that optimizes the behavior of BA-TA with TCP Streaming. If during a given duration, that we configure to be five seconds like in the *PSM-Throttling* bandwidth estimation algorithm [10], BA-TA observes that even when reducing the trigger interval or switching to active mode the experienced throughput remains stable, then when computing $ratio$ (line 20 in Algorithm 2) BA-TA replaces the estimated bottleneck bandwidth with the estimated application throughput. This optimized version of BA-TA is depicted in Figure 15 as *BA-TA+Str*.

Finally, Figure 16 illustrates the performance of Web traffic. In this case we consider the *Active Mode*, *Wi-Fi PSM* and *BA-TA* algorithms, since the *PSM-T* algorithm was not designed for Web traffic. Figures 16(a) and 16(b) illustrate respectively the average Web page download time and the average energy consumption per Web page, when a Wi-Fi station uses the different algorithms under study. The same model for Web traffic used in Section 4.1 is used here. Since Web traffic is mostly affected by latency, we only report the results obtained in the RTT experiment for our *Fast DSL* scenario. We can see in Figure 16(a) and 16(b) how *BA-TA* is able to deliver Web page download times similar to the ones of the *Active Mode* algorithm, at an energy cost similar to the

[19] We use 10 seconds of buffer in our experiments

one of *Wi-Fi PSM*. The reason behind the better performance of BA-TA are the dynamics depicted in the lower part of Figure 10(b). As illustated in the figure, BA-TA converges to a trigger interval small enough so that the delay experienced by the web page significantly reduces compared to Wi-Fi PSM. In addition, by sleeping between triggers BA-TA can achieve very significant energy savings. Instead, when Active Mode is used, the Wi-Fi stations spends large periods of time idle and wasting power while waiting to retrieve the objects of the web page.

5 Conclusions

Energy efficiency is becoming a matter of capital importance for the Wi-Fi technology to continue with its successful development. In this paper we have studied, by means of analysis and simulation, the effect that current Wi-Fi power saving protocols have on the throughput/energy trade-off experienced by long lived TCP traffic. Our study unveils that the efficiency of Wi-Fi power saving protocols critically depends on the bottleneck bandwidth experienced by a TCP connection.

Based on the obtained insights, we have designed and evaluated a novel algorithm, BA-TA, that runs in a Wi-Fi station, does not require any modification to existing Wi-Fi standards, and using only information available at layer two, significantly improves the performance/energy trade-off of long lived TCP connections, whilst also exhibiting a notable performance with Web traffic and TCP Streaming.

Several paths for future work may span from the work presented in this paper. First, we believe that the insights obtained in our study and the developed models may also be applicable to other MAC technologies that actively buffer data frames, like cellular technologies with power saving capabilities, e.g. WiMAX or LTE, or even optical networks like xPON. Second, while having presented a thorough simulative evaluation in this paper, the performance of BA-TA needs to be assessed in real hardware. Third, given the multiple power saving algorithms existent in the state of the art that target different traffic types or applications, a generic solution is needed for mobile devices to be able to identify the ongoing traffic at layer two and accordingly select the appropriate algorithm. Finally, another matter that deserves further study is to understand how the different power saving algorithms affect the fairness interactions between TCP flows in active mode and TCP flows in power saving.

References

1. Wi-Fi Alliance, *Thirsty for bandwidth, consumers and carriers embrace Wi-Fi phones*, March 23rd 2010, http://www.wi-fi.org.
2. Dropbox, http://en.wikipedia.org/wiki/Dropbox_(service)
3. *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-2007*.
4. X.Pérez-Costa, D.Camps-Mur and T.Sashihara. *Analysis of the Integration of IEEE 802.11e Capabilities in Battery Limited Mobile Devices*. IEEE Wireless Communications Magazine (WirComMag), special issue on Internetworking Wireless LAN and Cellular Networks, Volume 12, Issue 6, December 2005.
5. X.Pérez-Costa, D.Camps-Mur, A. Vidal. *On the Distributed Power Saving Mechanisms of Wireless LANs 802.11e U-APSD vs 802.11 Power Save Mode*. Computer Networks Volume 51, Issue 9, 20 June 2007, Pages 2326-2344.
6. R.Krashinsky and H.Balakrishnan, *Minimizing energy for wireless web access with bounded slowdown*, Proceedings of the eighth Annual International Conference on Mobile Computing and Networking (MOBICOM), September 2002.
7. D.Qiao and K.G.Shin, *Smart Power Saving Mode for IEEE 802.11 Wireless LANs*, Proceedings of IEEE INFOCOM, March 2005.
8. P. Agrawal, A. Kumar, J. Kuri, M.K. Panda, V. Navda, R. Ramjee, "OPSM - Opportunistic Power Save Mode for Infrastructure IEEE 802.11 WLAN," IEEE International Conference on Communications Workshops (ICC), 2010.
9. G.Anastasi, M.Conti, E.Gregori, and A.Passarella *802.11 power-saving mode for mobile computing in Wi-Fi hotspots: limitations, enhancements and open issues*, Wirel. Netw. 14, 6 (Dec. 2008).
10. Enhua Tan; Lei Guo; Songqing Chen; Xiaodong Zhang; *PSM-throttling: Minimizing Energy Consumption for Bulk Data Communications in WLANs*, Network Protocols, 2007. ICNP 2007. IEEE International Conference on, pp.123-132, 16-19 Oct. 2007.
11. D. Camps Mur, X. Perez-Costa and S. Sallent Ribes, *An adaptive solution for Wireless LAN distributed power saving modes*, *Computer Networks*, Volume 53, Issue 18, 24 December 2009.
12. Namboodiri, V.; Lixin Gao; *Energy-Efficient VoIP over Wireless LANs*, Mobile Computing, IEEE Transactions on , Vol.9, No.4, pp.566-581, April 2010.
13. Chai-Hien Gan, Yi-Bing Lin, "An Effective Power Conservation Scheme for IEEE 802.11 Wireless Networks," IEEE Transactions on Vehicular Technology, vol.58, no.4, pp.1920-1929, May 2009.
14. Zheng Zeng, Yan Gao, P.R. Kumar, "SOFA: A Sleep-Optimal Fair-Attention Scheduler for the Power-Saving Mode of WLANs," International Conference on Distributed Computing Systems (ICDCS), 2011.
15. Yi Xie, Xiapu Luo, R.K.C. Chang, "Centralized PSM: An AP-centric power saving Mode for 802.11 infrastructure networks, " IEEE Sarnoff Symposium, 2009.
16. Ashwin Rao, Arnaud Legout, Yeon-sup Lim, Don Towsley, Chadi Barakat, and Walid Dabbous, *Network characteristics of video streaming traffic*, In Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies (CoNEXT '11). ACM, New York, NY, USA, , Article 25 , 12 pages.
17. www.opnet.com.
18. Sangtae Ha, Injong Rhee, and Lisong Xu, *CUBIC: a new TCP-friendly high-speed TCP variant*, SIGOPS Oper. Syst. Rev. 42, 5 (July 2008), 64-74.
19. Jim Gettys and Kathleen Nichols. *Bufferbloat: Dark Buffers in the Internet*, Queue 9, 11, Pages 40 (November 2011), 15 pages.
20. A. Corlett, D.I. Pullin and S. Sargood, *Statistics of One-Way Internet Packet Delays*, 53rd IETF, Minneapolis, March 2002.
21. Akamai. The State of the Internet. 3rd Quarter 2010. http://www.akamai.com/stateoftheinternet, 2010.
22. http://www.websiteoptimization.com/.
23. *Broadcom 4311AG 802.11a/b/g*, http://www.broadcom.com.
24. P. Acharya, A. Sharma, E.M. Belding, K.C. Almeroth, K. Papagiannaki, *Rate Adaptation in Congested Wireless Networks through Real-Time Measurements*, Mobile Computing, IEEE Transactions on , Vol.9, No.11, pp.1535-1550, Nov. 2010.

25. Tianji Li, Douglas Leith, and David Malone. *Buffer sizing for 802.11-based networks*, IEEE/ACM Trans. Netw. 19, 1 (February 2011), 156-169.
26. Gene F. Franklin, J. David Powell, Abbas Emami-Naeini, *Feedback Control of Dynamic Systems*, Adison-Wesley.
27. S. Floyd , T. Henderson, *The NewReno Modification to TCP's Fast Recovery Algorithm*, RFC Editor, 1999.
28. *YouTube*, http://en.wikipedia.org/wiki/Youtube.

Appendix

In order to simplify the analysis while capturing the essence of the controller used in BA-TA we abstract the behavior of TCP in the following way. We notice that the RTT experienced by a TCP connection depends on the trigger interval in BA-TA, $int(n)$. Therefore, we assume that between interval updates in BA-TA, a long lived TCP connection delivers a throughput that is inversely proportional to the trigger interval used by BA-TA, i.e. $thr(n) = \frac{A}{int(n)}$, where A is assumed to be a constant value in this simplified model (TCP is assumed to converge within a time equal to $T_{update} \times count_{max}$), and n represents the *n-th* update interval of BA-TA. Under this premise, the $ratio$ value computed by BA-TA in the *n-th* interval update can be expressed as:

$$ratio(n) = \frac{thr(n)}{peak\_rate \times ratio_{min}} = \frac{B}{int(n)}$$

Where B is again a constant. Now recall from Equation 8 that the error incurred by BA-TA in the *n-th* interval update can be computed in the following way:

$$error(n) = ratio(n) - ratio_{min} = \frac{B}{int(n)} - ratio_{min}$$

We can now see how the proportional controller used in BA-TA evolves in time, noting that:

$$\begin{aligned} int(n+1) &= int(n)(1 + Gerror(n)) = \\ &= int(n)(1 - Gratio_{min}) + GB \end{aligned}$$

Where G is the gain used in the control law. Therefore, it can be proved by induction that:

$$\begin{aligned} int(n+k) &= int(n)(1 - Gratio_{min})^k + \\ &+ GB\sum_{j=0}^{k-1}(1 - Gratio_{min})^j = \\ &= (int(n) - \frac{B}{ratio_{min}})(1 - Gratio_{min})^k + \\ &+ \frac{B}{ratio_{min}} \end{aligned}$$

Therefore, when $k \to \infty$ the selected interval converges as $(1 - Gratio_{min})^k$, where $0 < ratio_{min} < 1$. It is then easy to see that $int(n+k)$ has the following convergence properties as $k \to \infty$:

$$\begin{cases} G \leq 0 \to Unstable \\ 0 < G < \frac{1}{ratio_{min}} \to Monotonic\ convergence \\ \frac{1}{ratio_{min}} < G < \frac{2}{ratio_{min}} \to Oscillatory\ convergence \\ G \geq \frac{2}{ratio_{min}} \to Unstable \end{cases}$$

Thus, when $0 < G < \frac{2}{ratio_{min}}$ the controller is stable and the selected interval converges to:

$$\lim_{k\to\infty} int(n+k) = \frac{B}{ratio_{min}}$$

Where, as seen in Figure 9, the selected interval logically converges to a value that is inversely proportional to $ratio_{min}$. In addition, when the algorithm converges the steady state error is:

$$\lim_{k\to\infty} error(n+k) = \frac{B}{\frac{B}{ratio_{min}}} - ratio_{min} = 0$$