

Distributed Delay Estimation and Call Admission Control in IEEE 802.11 WLANs

Kenta Yasukawa¹

Department of Communications and Integrated Systems
Tokyo Institute of Technology
knt@net.ss.titech.ac.jp

Andrea G. Forte and Henning Schulzrinne

Department of Computer Science
Columbia University
{andreaf,hgs}@cs.columbia.edu

I. INTRODUCTION

In recent years VoIP and IEEE 802.11 networks have seen a rapid growth. In IEEE 802.11 networks, an Access Point (AP) and the stations (STAs) it serves form a Basic Service Set (BSS). Each AP can support a limited number of concurrent voice calls; we refer to this number as the AP capacity. After the number of concurrent voice calls in the BSS surpasses the AP capacity, the communication of all users at that AP suffers from high delay, and therefore, poor quality. These are the problems we address with Call Admission Control (CAC).

We propose a mobile-station-based CAC mechanism. One of the strengths of the proposed approach is that it is very simple and yet accurate, while not requiring any probing of the medium. Our approach is entirely client-based, thus not requiring any changes in the infrastructure and the protocol.

II. PROPOSAL

In IEEE 802.11 networks using Distributed Coordination Function (DCF), the AP is the one suffering the most when congestion occurs because it has to send packets to all the stations in the BSS and still it has the same medium access priority as any other STA in the BSS. Therefore, for symmetric traffic, the down-link experiences congestion first because of the queuing delay at the AP. This means that STAs need to know how long packets are delayed at the AP in order to make CAC decisions. We introduce a method to estimate the delay in a BSS and explain how to perform CAC with it.

a) Delay Estimation using Time Between Idle Times: In 802.11 networks, the wireless medium is shared and every STA can hear packets that other STAs and the AP send and receive.

Let us consider an AP with several packets in its transmission queue. Such an AP tries to send one or more packets every time it gains access to the medium. Therefore, the AP will use any transmission opportunity it can so that no idle times will be observed on the medium. Figure 1 shows an example of an AP having four packets to send at moment t_0 . If we imagine that another packet arrives at t_1 , the packet would have to

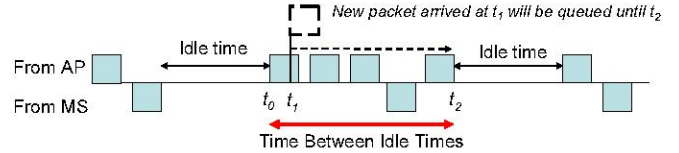


Fig. 1. Delay estimation using Time Between Idle Times (TBIT)

wait until the AP empties its queue at t_2 ($t_0 \leq t_1 < t_2$). The queuing delay for the packet is $t_2 - t_1$ and it is maximized when $t_0 = t_1$. The TBIT is equal to $t_2 - t_0$ and represents a direct measure of the maximum queuing delay at the AP at the time of the observation. This makes queuing delay estimation possible by just observing the time between idle times (TBIT).

Delay estimation using TBIT can be done by any STA in a BSS since every STA can “hear” the medium. Furthermore, the estimation can be performed anywhere in a BSS. Even in the presence of hidden nodes, this is possible because STAs can still “hear” ACK frames sent by the AP to acknowledge packets sent by other STAs, including hidden nodes. Finally, delay estimation is done without introducing any additional traffic. Therefore, delay estimation using TBIT represents a feasible way for STAs to estimate the queuing delay at their AP.

We use a threshold parameter named *idle time threshold*, I_{th} in order to ignore idle periods due to the backoff procedure in IEEE 802.11 DCF. We define I_{th} as:

$$I_{th} = T_{DIFS} + T_{Slot} \times CW_{min}, \quad (1)$$

where T_{DIFS} and T_{Slot} are the lengths of DCF Inter Frame Space (DIFS) and a time-slot duration, respectively, and CW_{min} is the minimum contention window size. These parameters are defined by the standard.

b) CAC using TBIT: We now show how TBIT can be used to make CAC decisions. Let us consider a STA making a new call. Let P and λ be its packet size and packet-rate, respectively, and let $T_{tx}(P)$ be the time needed to send a packet whose size is P including all the protocol overheads. Idle periods longer than $T_{tx}(P)$ can be considered as “service opportunities” for packets of the new call. Admitting a new call does not cause congestion if the frequency of service opportunities is higher than the packet-rate of the new flow. Therefore, if we denote the frequency of idle times that are longer

¹Work performed while at Columbia University.

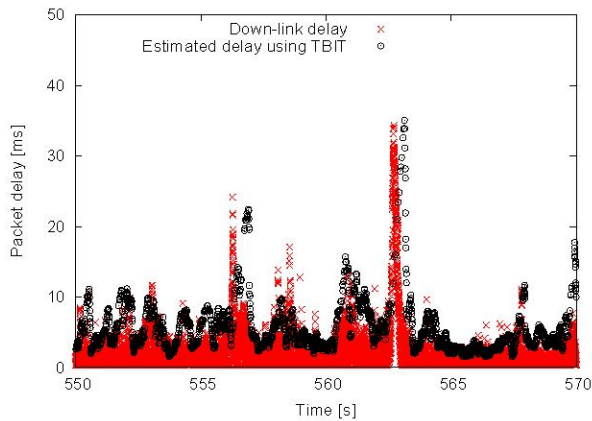


Fig. 2. Delay estimation using TBIT (G.711 VBR codec)

than $T_{tx}(P)$ by μ , we can say that the new call will not cause congestion if μ is larger than λ .

We also notice that the frequency of idle times is obviously the inverse of TBIT, so μ is obtained in the same way as in the delay estimation. The only change we need to apply here is to set the idle time threshold as $T_{tx}(P)$, that is, $I_{th} = T_{tx}(P)$. In doing so, idle periods that are not long enough to send a packet are ignored. Although $T_{tx}(P)$ is a function of P , a client which is starting a new call knows its packet size and can calculate $T_{tx}(P)$. Consequently, a STA can obtain μ by listening to the medium. By checking whether μ is larger than λ or not, a STA can make CAC decisions by itself.

III. EVALUATION

We evaluate the performance of TBIT for delay estimation and CAC through simulations, using ns-2 [1]. We use the Ethernet-to-wireless network topology and focus on the delay in the BSS. One AP is connected to the wired network and N STAs are in its service range. The STAs make VoIP calls with nodes over the wired network. One of the STAs monitors the wireless medium and computes TBIT as explained in Section II. The wireless network parameters used in our simulations are set according to the IEEE 802.11b standard.

Figure 2 shows the simulation results for delay estimation using TBIT. In the figure, the x -axis represents the time in the simulation and the y -axis shows the down-link delay. We can see from the figure that TBIT follows the actual down-link queueing delay well.

Figure 3 shows the results for CAC when using TBIT. The figure shows the 90th percentile down-link delay versus the number of calls in the BSS. Each plot is a result of multiple simulations with changing random seeds. In particular, we plotted the 90th percentile delays with 95% confidence intervals. The figure also has a second y -axis for the frequency of idle times. We define the frequency of idle times as the inverse of the average TBIT in a second.

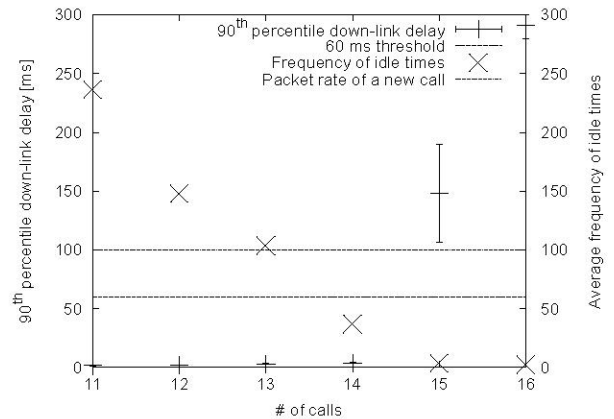


Fig. 3. VoIP capacity and frequency of idle times (G.711 CBR)

Figure 3 shows how the 90th percentile delay increases with the number of calls in G.711 codec case. As we can see, it exceeds 60ms when 15 calls are accepted, which means that 15 calls introducing an unacceptable delay. Therefore, the capacity is 14 in this case. Since the packet rate for a G.711 call is 100 [packets/s] taking into account both directions, a STA should not start a new call when the frequency of idle times is less than 100 when using TBIT. Figure 3 shows that this happens when 14 calls are accepted, i.e., just before the number of calls reaches the capacity. Therefore, we can say that STAs can make accurate CAC decisions by using TBIT.

Though the figures show results for particular scenarios, we performed simulations for various scenarios including heterogeneous scenarios in which different kinds of VoIP codecs are used, CBR and VBR are mixed, and data-traffic exist. Through the results, we made sure that TBIT can estimate the queueing delay at the AP regardless of the traffic type and that TBIT makes accurate CAC decisions also for such heterogeneous scenarios. In particular, if we define the utilization ratio U as the number of accepted calls divided by the number of calls at the capacity, U is always kept 0.9 or above, for every scenario.

IV. CONCLUSION

We introduced the concept of TBIT and showed how this can be used to estimate queueing delay and make CAC decisions in IEEE 802.11 networks. Through the simulations, we have confirmed that TBIT works well for both delay estimation and CAC.

Currently, we are building a test-bed to confirm the simulation results in a real environment. Preliminary results have proven the correctness of the simulation results. Further analysis is reserved for future work.

REFERENCES

- [1] UCB/LBNL/VINT. Network simulator ns. [Online]. Available: <http://www-mash.cs.berkeley.edu/ns/>