Adaptive Buffer Sizing for TCP Flows in 802.11e WLANs

Tianji Li, Douglas Leith
Hamilton Institute, National University of Ireland at Maynooth, Ireland
Email: {tianji.li, doug.leith}@nuim.ie

***Abstract*—We consider the provision of Access Point buffers in WLANs. We first demonstrate that the default use of static buffers in WLANs leads to either undesirable channel under-utilisation or unnecessary high delays, which motivates the use of dynamic buffer sizing. Although adaptive algorithms have been proposed for wired Internet, a number of fundamental new issues arise in WLANs which necessitates new algorithms to be designed. These new issues include the fact that channel bandwidth is time-varying, the mean service rate is dependent on the level of channel contention, and packet inter-service times vary stochastically due to the random nature of CSMA/CA operation. We propose an adaptive sizing algorithms which is demonstrated to be able to maintain high throughput efficiency whilst achieving low delay.**

I. Introduction

We consider WLANs where the access point (AP) acts as a wireless router between the WLAN and the Internet. As TCP flows account for the vast majority (more than 90% [21]) of current Internet traffic and also of WLAN traffic [17], we consider buffer sizing for TCP flows.

Buffers are traditionally sized with two primary objectives in mind.

(i) *Accommodating short-term packet bursts.* Due to the nature of TCP, internet traffic tends to be bursty. Should too many packets arrive in a sufficiently short interval of time then a network device may lack the capacity to process all of the packets immediately. The first job of the buffer is to mitigate packet losses due to bursts by accommodating these packets in a buffer until they can be serviced.

(ii) *Ensuring AIMD throughput efficiency.* Most of the traffic on networks continues to be carried by the TCP protocol. The AIMD congestion control algorithm used by TCP reduces the number of packets in flight by half on detecting network congestion. If buffers are too small, this backoff action will cause them to empty with a corresponding reduction in link utilisation.

The classical rule of thumb is to provision buffers to be equal to the *bandwidth* of the link multiplied by the average *delay* (which is typically described by round trip time or RTT) of the flows utilising this link: the *Bandwidth-Delay Product* (BDP) [18]. Recent work on buffer sizing for wired links [2] shows that the BDP rule can be overly conservative, and suggests sizing buffers to $\frac{BDP}{\sqrt{n}}$ instead where n is the number of flows traversing a link. This exploits the statistical multiplexing when many flows share a link. Since real-world traffic patterns are often extremely complex, including a mix of connection sizes, RTTs, etc that change over time, adaptive buffer sizing is considered in [14] [19].

A number of fundamental new issues arise in 802.11 WLANs. Firstly, the mean service rate at a wireless station is strongly dependent on the level of channel contention and thus on the number of active stations and their load. Secondly, even when the network load is fixed, the packet inter-service times at a station are not fixed but vary stochastically due to the random nature of the CSMA/CA operation. As a result, neither the bandwidth nor the delay in 802.11 WLANs are constant, in contrast to the wired links. We therefore do not have a fixed BDP value available to provide a basis for sizing buffers. In our previous work [9], we design an emulating BDP algorithm to size buffers according to this BDP rule.

In this paper, we design a new algorithm to exploit statistical multiplexing gains in WLANs, besides providing high throughput and low delay. This involves feedback control of buffer size based on measurements of the buffer idle and busy time. In particular, we observe the buffer occupancy over an interval of time. If the buffer rarely empties, we decrease the buffer size to avoid high delay. Conversely, if the buffer is empty for too long a period, we increase the buffer size to maintain high throughput. The effectiveness of the algorithm is demonstrated with extensive simulations.

II. Related Work

The classical approach to sizing Internet router buffers is the BDP rule proposed in [18]. Recently, in [2] it is argued that the BDP rule can be overly conservative on links shared by a large number of flows. In this case it is unlikely that TCP congestion window sizes (cwnd) evolve synchronously and due to statistical multiplexing of cwnd backoffs, the combined buffer requirement can be considerably less than the BDP. The analysis in [2] suggests that it may be sufficient to size buffers as the BDP divided by the square root of the number of active flows n. The potential may therefore exist to reduce buffer sizes by two to three orders of magnitude. This work is extended in [12], [5] and [20] to consider the performance of TCP congestion control with many connections under the assumption of small, medium and large buffer sizes. Several authors have pointed out that the value n can be difficult to

This work is supported by Irish Research Council for Science, Engineering and Technology and Science Foundation Ireland Grant 03/IN3/I396.

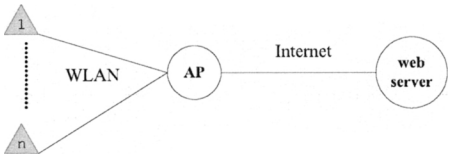

Fig. 1. Topology used for buffer sizing in WLANs. Each station in WLAN is the source/destination of a single TCP flow. MAC parameters of the WLAN are listed in Table I.

T_{SIFS} (μs)	10
Idle slot duration (σ) (μs)	9
Retry limit	11
Packet size (bytes)	1000
PHY data rate (Mbps)	54
PHY basic rate (Mbps)	6
PLCP rate (Mbps)	6

TABLE I
MAC/PHY PARAMETERS USED, CORRESPONDING TO 802.11G.

determine for realistic traffic patterns, which not only include a mix of connections sizes and RTTs, but are also strongly time-varying [4], [19]. In [19], it is observed that in a production link, traffic patterns vary significantly, and may contain a complex mix of flow connection lengths and RTTs. It is demonstrated in [4] that use of very small buffers can lead to an excessive loss rate. Motivated by these observations, in [14] [6] a measurement-based adaptive buffer size tuning method is therefore proposed.

The foregoing work is in the context of wired links, and to our knowledge the question of buffer sizing for 802.11 wireless links has received almost no attention in the literature. Besides our previous work on emulating BDP in WLANs [9], other notable exceptions include [11] [13] [15]. Sizing of buffers for voice traffic in WLANs is investigated in [11]. The impact of fixed buffer sizes on TCP flows is studied in [13]. In [15], TCP performance with a variety of AP buffer sizes and 802.11e parameter settings is investigated. With regard to the current state of the art, some vendors use small static buffers (e.g., in Proxim APs, up to 8 packets can be temporarily stored in the interface buffer), while others use large static buffers (e.g., in Atheros APs, up to 399 packets can be held.).

III. SETUP

We consider the topology shown in Fig. 1 where the AP acts as a wireless router between the WLAN and the Internet. Upload flows originate from stations in the WLAN on the left and are destined to server(s) in the wired network on the right. Download flows are from the server(s) to stations in the WLAN. We ignore differences in wired bandwidth and delay from the AP to the servers which can cause TCP unfairness issues on the wired side (an orthogonal issue) by using the same wired-part RTT for all flows.

We note that in WLANs, TCP ACK packets can be easily queued/dropped due to the fact that the basic 802.11 DCF ensures that stations win a roughly equal number of transmission opportunities. For example consider n stations each carrying one TCP upload flow. The TCP ACKs are transmitted by the AP. While the data packets for the n flows have an aggregate $n/(n+1)$ share of the transmission opportunities the TCP ACKs for the n flows have only a $1/(n+1)$ share. Issues of this sort are known to degrade TCP performance significantly as queuing and dropping of TCP ACKs disrupt the TCP ACK clocking mechanism. Following [8], we address this problem using 802.11e. At the AP and each station we treat TCP ACKs as a separate traffic class, collecting them into a queue which is assigned high priority via $CW_{min} = 3$, $CW_{max} = 7$, $AIFS = 2$. TCP data packets are transmitted by another queue with parameters $CW_{min} = 31$, $CW_{max} = 1023$ and $AIFS = 6$. This makes use of 2 out of the 4 available queues in 802.11e.

We use IEEE 802.11g parameters as shown in Table I. For TCP traffic, the widely deployed TCP Reno with SACK extension is used. The TCP slow start threshold is set to be 64 packets [16], the maximum congestion window size is set to be 4096 packets (each has a payload of 1000 bytes) which is the default size of current Linux kernels (2.6.xxx).

IV. PERFORMANCE WITH FIXED BUFFERS

In contrast to wired networks, the mean service rate at a wireless station is not fixed but instead depends upon the level of channel contention and the network load. This is illustrated in Fig. 2 where the throughput and delay of a download flow are plotted as a function of AP buffer size when the number of competing upload flows (with one upload flow per wireless station) is varied. Similarly to wired networks, the throughput always increases monotonically with the buffer size, reaching a maximum above a threshold buffer size. However, it can also be seen that the download throughput falls as the number of competing uploads increases. The variation in throughput can be substantial, e.g., in this example the maximum throughput falls from 14Mbps to 1.25Mbps as the number of competing uploads increases from 0 to 10. As a result, the BDP – marked by vertical lines in Fig. 2 – also varies significantly and this is reflected in buffering requirements. For example, it can be seen from Fig. 2 that with no competing uploads the threshold buffer size above which the AP achieves maximum throughput is around 300 packets, while for 10 competing uploads this buffer size falls to approximately 70 packets.

In addition to variations in the mean service rate, the random nature of 802.11 CSMA/CA operations leads to short time-scale stochastic fluctuations in service rate. This is fundamentally different from wired networks and directly impacts buffering behaviour. For example, from Fig. 2 with 10 uploads the maximum download throughput is 1.25Mbps, yielding a BDP of 31 packets. However, it can be seen that at this buffer size the achieved download throughput is only about 60% of the maximum – a buffer size of at least 50 packets is required to achieve 100% throughput. Stochastic fluctuations in service rate can lead to early queue overflow or even TCP retransmission timeouts (RTOs) unless buffer provisioning over and above the BDP is used. For example, Fig. 3 illustrates

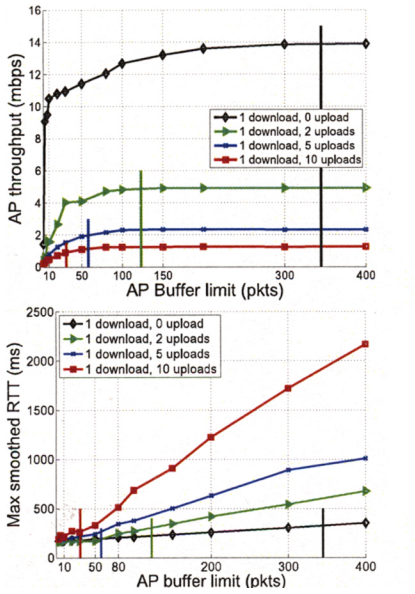

Fig. 2. WLAN buffer sizing requirements. Data is shown for 1 download flow and 0, 2, 5, 10 competing uploads. Corresponding BDP values are marked by vertical lines. "Max smoothed RTT" denotes the maximum TCP *srtt* value observed. Wired backhaul link bandwidth 100Mbps, RTT 200ms.

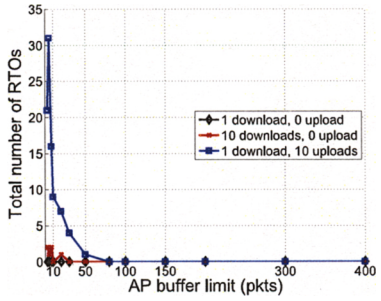

Fig. 3. Number of TCP RTOs vs AP buffer size. Values collected in simulations run for 400 seconds.

the total number of TCP RTOs as the buffer size is varied. It can be seen that the number of RTOs decreases as the buffer size is increased. The buffer size required to minimise RTOs is dependent on network conditions.

One possible buffer sizing approach in WLANs is to provision buffers based on worst case conditions, i.e., based on the conditions requiring the largest buffering to achieve high throughput. However, while ensuring high throughput, this comes at the cost of high latency. For example, it can be seen from Fig. 2 that when a fixed buffer size of 300 packets is used (which in this example ensures maximum throughput regardless of the number of contending uploads), the round-trip latency experienced by the download flow is about 300ms with no uploads but rises to around 2s with 10 contending upload stations. This occurs because TCP's congestion control algorithm probes for bandwidth until packet loss occurs and so flows will tend to fill buffers, regardless of their size.

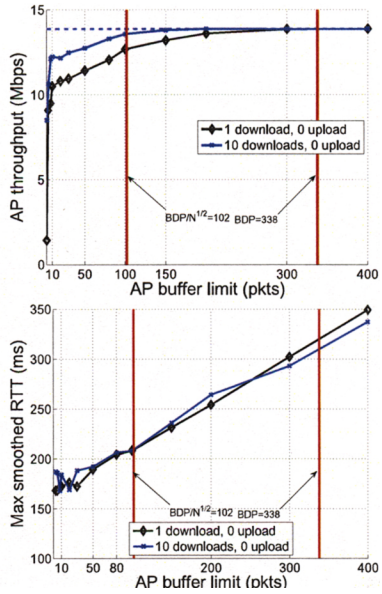

Fig. 4. Impact of statistical multiplexing. There are 1/10 downloads and no uploads. Wired backhaul link bandwidth 100Mbps, RTT 200ms.

Conversely, sizing the buffer to achieve lower latency across all network conditions comes at the cost of reduced throughput, e.g., a buffer size of 30 packets ensures latency of 200-300ms for up to 10 contending upload stations but when there are no contending uploads the throughput of a download flow is only about 75% of the maximum achievable.

Therefore, neither fixed long nor fixed short buffers can maintain both high throughput and low delays at the same time, motivating consideration of adaptive approaches to buffer sizing.

The potential exists to lower the buffer size (thereby reducing latency) without loss of throughput. This can be seen from the example in Fig. 4 that while a buffer size of 300 packets is needed to maximise throughput with a single download flow, this falls to around 100 packets when 10 download flows share the link. In this section we consider the design of measurement-based algorithms that are capable of taking advantage of such statistical multiplexing opportunities.

V. Adaptive Buffer Limit Tuning

A. The Algorithm

Our objective is to simultaneously achieve both high throughput efficiency and low delay. Intuitively, in order to ensure efficient link utilisation, the buffer should not lie empty for too long a time. Increasing the buffer size tends to reduce the link idle time. However, to ensure low delays, the buffer should be as short as possible and a trade-off therefore exists. This intuition suggests the following approach. We observe the buffer occupancy over an interval of time. If the buffer rarely empties, we decreases the buffer size to avoid high delay. Conversely, if the buffer is empty for too long a period, we increase the buffer size to maintain high throughput.

In more detail, we consider an adaptive buffer tuning procedure as follows. Let $t_i(k)$, $t_b(k)$ be the durations of idle and busy time in an observation interval t, i.e., $t = t_i(k) + t_b(k)$, and $q(k)$ be the buffer limit during the k-th observation interval. The buffer limit is updated according to

$$q(k+1) = q(k) + at_i(k) - bt_b(k), \quad (1)$$

where a and b are design parameters. Pseudo-code for this procedure is given in Algorithm 1.

Assuming q converges, then we have that $at_i = bt_b$, i.e., $t_i = \frac{b}{a}t_b$ and the link utilisation is therefore lower bounded by

$$\frac{t_b}{t_i + t_b} = \frac{1}{1 + b/a}. \quad (2)$$

Choosing $\frac{b}{a}$ to be small then ensures high utilisation.

The duration of the observation/update interval t should be so selected to reflect timely changes on buffer usage which are nonpredictable in reality (see for example [19] and the references therein). Too small t can therefore yields similar/repeated observations if traffic patterns are varying slowly, too large t will likely miss bursty changes. Here we choose the safer option, i.e., a short $t = 1$ second is used.

Let α_i be the rate in $packet/s$ at which flow i increases its congestion window and $\alpha_T = \sum_{i=1}^{n} \alpha_i$ be the aggregate rate at which flows increase their congestion windows, in packets/s. Standard TCP increases the flow congestion window by one packet per RTT, in which case $\alpha_i \approx 1/T_i$ where T_i is the RTT of flow i. In real world Internet, T_i is upper-bounded by 200ms in most of the cases [21]. We then have that a rough lower bound on α_T is 5 (corresponding to 1 flow with RTT 200ms). As the update duration used is 1 second, we choose $a = 10$ to be slightly larger than the lower bound of α_T, and $b = 1$ to ensure $\frac{b}{a}$ to be small in this paper. A formal analysis as to the selection of a and b is given is our technical report [10].

It is prudent to constrain q to lie between the minimum and the maximum values q_{min} and q_{max}. In the following, the maximum limit q_{max} and the minimum buffer limit q_{min} are set to be 400 and 30 packets, respectively. We use 400 packets as the maximum buffer limit to facilitate comparison with the fixed buffer scheme used in Atheros chip sets, and 30 packets as the minimum buffer limit to avoid frequent RTOs.

This procedure is shown in Algorithm 1.

Algorithm 1 : The adaptive buffer tuning algorithm.

```
Set the initial queue limit, the maximum buffer limit q_max
  and the minimum buffer limit q_min.
Set the increase step size a and the decrease step size b.
for Every t seconds do
   Measure the idle time t_i.
   q_new = q + a t_i - b(t - t_i).
   if q_new < q_max then
      if q_new < q_min then
         q ← q_min
      else
         q ← q_new
      end if
   else
      q ← q_max
   end if
end for
```

B. Results

The effectiveness of the ALT algorithm is illustrated in Fig. 5. Recall that (Figs. 2 and 4) 330, 100 and 70 packets are the approximate threshold buffer sizes above which the AP achieves maximum throughput for perspectively 1 download only, 10 downloads and 1 download plus 10 upload flows. It can be seen that using the ALT algorithm, the desired buffer limits are reached.

In Figs. 6 and 7, we plot more results for cases when the RTT in the wired part of the network is varied from 50 to 300ms and when the number of upload flows is increased from 0 to 10. The metrics uses are throughput and maximum smoothed RTT percentage. The throughput percentage is the ratio between throughput achieved using the ALT algorithm and that with a 400-packet buffer. The maximum smoothed RTT percentage is defined in a similar way. That is, these two metrics are intentionally defined to compare the ALT algorithm with fixed buffers. It can be seen from the figures that the ALT algorithm maintains high throughput efficiency across the entire range of operating conditions. This is achieved while maintaining low delays compared with fixed buffers. We can also see that the ALT algorithm is capable of exploiting the statistical multiplexing where feasible. In particular, significant lower delays are achieved with 10 download flows whilst maintaining high throughput efficiency.

VI. Conclusions

We have shown that the use of static buffers in WLANs leads to either undesirable channel under utilisation or unnecessary high delays. We have proposed an adaptive algorithm that achieves both high throughput efficiency and low delays simultaneously. Future work includes theoretical analysis of the proposed algorithm, and test-bed implementation is in progress.

References

[1] Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Medium Access Control (MAC) Quality of Service (QoS) Enhancements, IEEE 802.11e/D8.0, Feb. 2004.

[2] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," in *Proc. of ACM SIGCOMM*, 2004. pp. 281–292.

[3] C. Chatfield, *The Analysis of Time Series, An Introduction*, CRC Press 2004.

[4] A. Dhamdher and C. Dovrolis, "Open Issues in Router Buffer Sizing," in *Computer Communication Review*, Jan. 2006.

[5] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with Very Small Buffers," in *Proc. of IEEE INFOCOM*, Dec. 2006.

[6] C. Kellett, R. Shorten, and D. Leith, "Sizing Internet Router Buffers, Active Queue Management, and the Lur'e Problem," in *Proc. of IEEE CDC*, 2006.

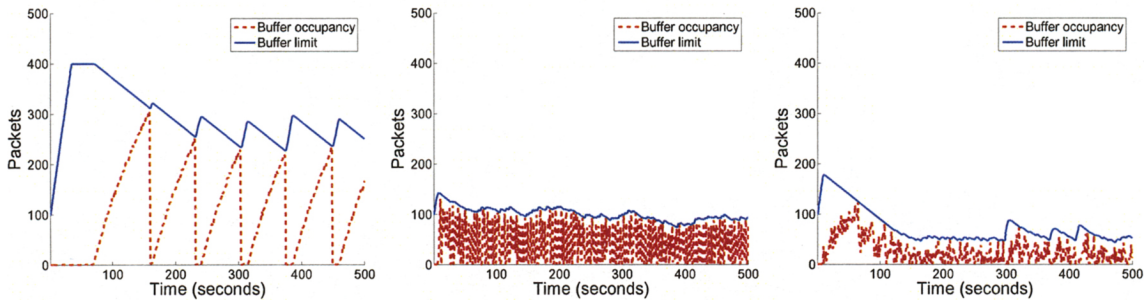

Fig. 5. Buffer histories with the ALT algorithm.

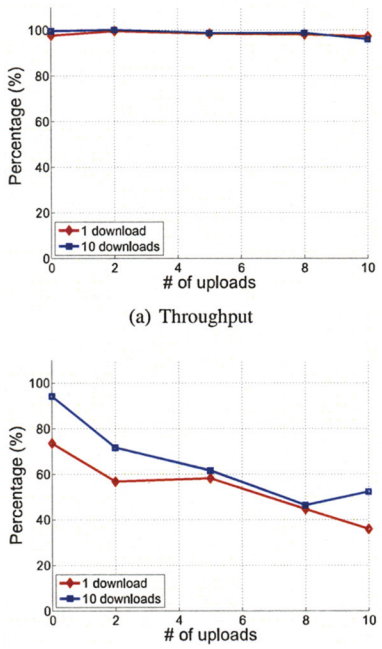

Fig. 6. Performance of the ALT algorithm as the number of upload flows is varied. The wired RTT is 200ms.

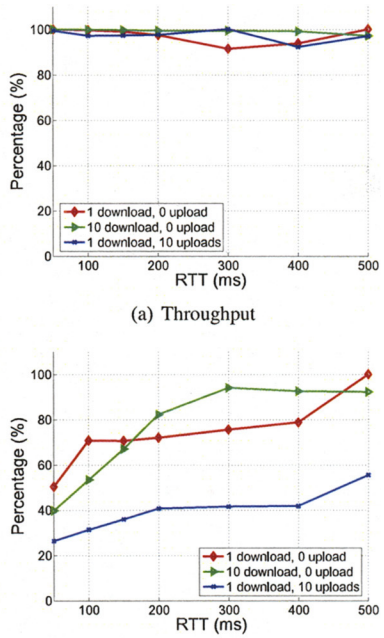

Fig. 7. Performance of the ALT algorithm as the wired RTT is varied.

[7] S. Floyd, and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[8] D. Leith, P. Clifford, D. Malone, and A. Ng, "TCP Fairness in 802.11e WLANs," *IEEE Communications Letters*, vol. 9, no. 11, pp 964–966, Jun. 2005.

[9] T. Li and D. Leith, "Buffer Sizing for TCP Flows in 802.11e WLANs," *IEEE Communications Letters*, to appear.

[10] T. Li and D. Leith, "Buffer Sizing Access Point in WLANs," Technical report of Hamilton Institute, NUI Maynooth, Ireland. Nov. 2007.

[11] D. Malone, P. Clifford, and D. J. Leith, "On Buffer Sizing for Voice in 802.11 WLANs," *IEEE Communications Letters*, vol. 10, no. 10, pp 701–703, Oct. 2006.

[12] G. Raina and D. Wischik, "Buffer Sizes for Large Multiplexers: TCP Queueing Theory and Instability Analysis," in *Proc. of EuroNGI*, Jul. 2005.

[13] S. Pilosof, et. al., "Understanding TCP fairness over Wireless LAN," in *Proc. IEEE INFOCOM* 2003.

[14] R. Stanojevic, C. Kellett, and R. Shorten, "Adaptive Tuning of Drop-Tail Buffers for Reducing Queueing Delays," *IEEE Communications Letters*, vol. 10, no. 7, pp 570–572, Jul. 2006.

[15] M. Thottan, and M. C. Weigle, "Impact of 802.11e EDCA on mixed TCP-based applications," in *Proc. IEEE WICON* 2006.

[16] A. S. Tanenbaum, *Computer Networks* Fourth Edition. New Jersey, Prentice Hall PTR, 2003.

[17] D. Tang and M. Baker, "Analysis of A Local-Area Wireless Network," in *Proc. of ACM MobiCom*, Aug. 2000.

[18] C. Villamizar and C. Song, "High Performance TCP in ANSNET," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 45–60, Oct. 1994.

[19] G. Vu-Brugier, R. Stanojevic, D. Leith, and R. Shorten, "A Critique of Recently Proposed Buffer-Sizing Strategies," *ACM Computer Communication Review*, vol. 37. no. 1, Jan. 2007.

[20] D. Wischik and N. McKeown, "Part I: buffer sizes for core router," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, Jul. 2005.

[21] Z. Zhao, S. Darbha, and A. L. N. Reddy, "A Method for Estimating the Proportion of Nonresponsive Traffic At a Router," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 708–718, Aug. 2004.