ISSN No. 0976-5697

Volume 3, No. 7, Nov-Dec 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Dynamic Buffer Sizing Algorithms for 802.11-Based networks

Depavath Harinath
MCA,
Sreenidhi Institution of Science and Technology,
Accredited by NBA, AICTE, New Delhi - Permanently Affiliated to JNTU
Hyderabad,A.P.,India.
harinath.depavath@gmail.com

Abstract— Buffers play a key role in 802.11/802.11e Wireless networks.802.11 is a set of standards for implementing wireless local area networks (WLAN) computer communication in the 2.4, 3.6 and 5 GHz frequency bands. They are created and maintained by IEEE LAN/MAN Standards Committee (IEEE 802).Buffers are used to accommodate short-term packet bursts so as to mitigate packet drops and to maintain high link efficiency. The use of fixed size buffers in 802.11networks inevitably leads to either undesirable channel underutilization or unnecessary high delays. The main objective of this paper is to maintain high network utilization while providing low queuing delays in 802.11 wireless networks through dynamic buffer sizing algorithms.

Keywords— Buffer sizing, IEEE 802.11, wireless LANs(WLANs).

I. INTRODUCTION

In Communication networks, buffers are used to accommodate short-term packet bursts so as to mitigate packet drops and to maintain high link efficiency. Packets are queued if too many packets arrive in a sufficiently short interval of time during which a network device lacks the capacity to process all of them immediately.

For wired routers, the sizing of buffers is an active research topic [1], [2], [3], [4], [5].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 RRT) of the flows utilising this link: the Bandwidth-Delay Product(BDP) [4].This amount of buffering allows for 100% utilization of the egress link under all traffic conditions. Following this rule, most router buffers are designed to have 100-250*ms* of buffering. This, together with the TCP mechanism of congestion avoidance, ensures high link utilization. In the last few years, several studies related to buffer sizing at a congested router have occurred [6], [1], [7], [8].Round –trip time(RRT), also called round –trip delay, is the time required for a signal pulse or packet to travel from a specific source to a specific destination and back again. In this context, the source is the computer initiating the signal and the destination is a remote computer or system that receives the signal and retransmits it.

Having small buffers is attractive as it reduces the amount of memory, required physical space, energy consumption, and price of the router. According to the point of [3], the main advantage of having small buffers is the reduction in queueing delays and jitter. In the current Internet the average number of hops on a random path is about 13 [9].For a single flow with that many hops it is possible to expect several congested links on the path. Thus buffering of several hundreds *ms* at each router would imply very large queueing delays.

Recent work on buffer sizing for wired links [1] 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, RRTs, etc that change over time, adaptive buffer sizing is considered in [3] [5].

Compared to sizing buffers in wired routers, a number of fundamental new issues arise when considering 802.11-based networks. First, unlike wired networks, wireless transmissions are inherently broadcast in nature, which leads to the packet service times at different stations in a WLAN being strongly coupled. For example, the basic 802.11 DCF(Distributed Coordinated Function) which is a CSMA/CA-based algorithm ensures that the wireless stations in a WLAN win a roughly equal number of transmission opportunities [10], hence the mean packet service time at a station is an order of magnitude longer when 10 other stations are active than when only a single station is active. Consequently, the buffering requirements at each station would also differ, depending on the number of other active stations in the WLAN. In addition to variations in the mean service time, the distribution of packet service times is also strongly dependent on the WLAN offered load. This directly affects the burstiness of transmissions and hence buffering requirements. Second, wireless stations dynamically adjust the physical transmission rate/modulation used in order to regulate noncongestive channel losses. This rate adaptation, whereby the transmit rate may change by a factor of 50 or more (e.g. from 1 to 54 Mb/s in 802.11a/g), may induce large and rapid variations in required buffer sizes. Third, the ongoing 802.11n standards process proposes to improve throughput efficiency by the use of large frames formed by aggregation of multiple packets [11], [12]. This acts to couple throughput efficiency and buffer sizing in a new way since the latter directly affects the availability of sufficient packets for aggregation into large frames.

It follows from these observations that, among other things, there does not exist a fixed buffer size that can be used for sizing buffers in WLANs. This leads naturally to consideration of dynamic buffer-sizing strategies that adapt to changing conditions.

The use of fixed size buffers in 802.11 networks inevitably leads to either undesirable channel under-utilization or unnecessary high delays. The main objective of this paper is to achieve high throughput while maintaining low delay across a wide range of network conditions in 802.11 wireless networks through dynamic buffer sizing algorithms.

II. DYNAMIC BUFFER SIZING ALGORITHMS

A. The EBDP Algorithm:

The Emulating BDP is a simple adaptive algorithm based on the classical BDP rule. Although this algorithm cannot take advantage of statistical multiplexing opportunities, it is of interest both for its simplicity and because it will play a role in the more sophisticated A* algorithm.

Given an online measurement of the mean service time T_{serv} , the classical BDP rule yields the following eBDP buffer-sizing strategy. Let T_{max} be the target maximum queuing delay. 1/ T_{serv} is the mean service rate, we select buffer Q_{eBDP} according to

$$Q_{\text{eBDP}} = \min(T_{\max}/T_{\text{serv}}, Q_{\max}^{\text{eBDP}}),$$

Where $Q_{\max}^{\text{eBDP}}$ is the upper limit on buffer size. This effectively regulates the buffer size to equal the current mean BDP. The buffer size decreases when the service rate falls and increases when the service rate rises, so as to maintain an approximately constant queuing delay of T_{max} seconds. We may measure the flows' RTTs to derive the value for T_{max} in a similar way to measuring the mean service rate, but in the examples presented here we simply use a fixed value of 200 *ms* since this is an approximate upper bound on the RTT of the majority of the current internet flows.

We note that the classical BDP rule is derived from the behavior of TCP congestion control (in particular, the reduction of congestion window size, cwnd, by half on packet loss) and assumes a constant service rate and fluid-like packet arrivals. Hence, for example, at low service rates the BDP rule suggests use of extremely small buffer sizes. However, in addition to accommodating TCP behaviour, buffers have the additional role less links, short-term packet bursts and, in the case of wireless links, short-term fluctuations in packet service times. It is these latter effects that leads to the steep dropoff in throughput efficiency when there are competing uploads (and so stochastic variations in packet service times due to channel connection) plus small buffer sizes. We therefore modify the eBDP update to

$$Q_{\text{eBDP}} = \min(T_{\max}/T_{\text{serv}} + c, Q_{\max}^{\text{eBDP}}),$$

Where c is an overprovisioning amount to accommodate short-term fluctuations in service rate. Pseudocode for eBDP algorithm is shown in Algorithms 1 and 2.

Algorithm 1: Drop tail Operation of the eBDP algorithm.

```
a. Set the target queuing delay $T_{max.}$
b. Set the overprovision parameter $c$
c. for each incoming packet $p$ do
d.   Calculate
       $Q_{\text{eBDP}} = \min(T_{\max}/T_{\text{serv}} + c, Q_{\max}^{\text{eBDP}})$ ,
       where $T_{serv}$ is from MAC Algorithm 2.
e.   if current queue occupancy < $Q_{eBDP}$, then
f.     put $p$ into queue
g.   else
g.     Drop p.
h.   end if
i. end for
```

Algorithm 2: MAC operation of the eBDP algorithm.

```
a. Set the averaging parameter $W$.
b. for each outgoing packet $p$ do
c.   Record service start time $t_s$ for $p$.
d.   Wait until receive MAC ACK for $p$, record service
     end time $t_e$.
e.   Calculate service time of $p$:
     $T_{serv} = (1-W)T_{serv} + W(t_e-t_s)$.
f. end for
```

B. Adaptive Limit Tuning (ALT) Feedback Algorithm:

The main objective here it is to achieve both efficient link utilization and low delays in the face of stochastic time variations in the service time simultaneously. Intuitively, for efficient link utilization, we need to ensure that there is a packet available to transmit whenever the station wins a transmission opportunity. That is, we want to minimize the time that the station buffer lies empty, which in turn can be achieved by making the buffer size sufficiently large (under fairly general traffic conditions, buffer occupancy is a monotonically increasing function of buffer size [13]). However, using large buffers can lead to high queuing delays, and to ensure low delays, the buffer should be as small as possible. We would therefore like to operate with the smallest buffer size that ensures sufficiently high link utilization.

We now introduce the following Adaptive Limit Tuning (ALT) algorithm. Define a queue occupancy threshold q_{thr} and $t_i(k)$ (referred to as the *idle time*) be the duration of time that the queue spends at or below this threshold in fixed observation interval t, and $t_b(k)$ (referred to as the *busy time*) be the corresponding duration spent above the threshold. Note that $t = t_i(k) + t_b(k)$, and the aggregate amount of idle/busy time t_i and t_b over an interval can be readily observed by a station. Also the link utilization is lower-bounded by $t_b/(t_b+t_i)$. Let *q(k)* denote the buffer size during the k^{th} observation interval. The buffer size is then updated according to

$$q(k+1) = q(k) + a_1 t_i(k) - b_1 t_b(k)$$

Where a_1 and b_1 are design parameters. Pseudocode for this ALT algorithm is given in Algorithm 3. This algorithm seeks to maintain a balance between the time t_i that the queue is idle and the time t_b that the queue is busy. That is, it can be seen that when $a_1t_i(k) = b_1t_b(k)$, the buffer size is kept unchanged. When the idle time larger so that $a_1t_i(k) > b_1t_b(k)$, then the buffer size is increased. Conversely, when the busy time is larger enough that $a_1t_i(k) < b_1t_b(k)$, then the buffer size is decreased.

Algorithm 3: The ALT Algorithm.

```
a.  Set the initial queue size, the maximum buffer size $q_{max}$ and the minimum buffer size $q_{min}$.
b.  Set the increase step size $a_1$ and the decrease step size $b_1$.
c.  for every $t$ seconds do
d.      Measure the idle time $t_i$.
e.      $q_{ALT} = q_{ALT} + a_1 t_i - b_1(t - t_i)$.
f.      $q_{ALT} = \min(\max(q_{ALT}, q_{min}), q_{max})$
g.  end for
```

C. Combining eBDP and ALT: The A Algorithm:*

We can combine the eBDP and ALT algorithms by using the mean packet service time to calculate Q_{eBDP} as per the eBDP algorithm and the idle/busy times to calculate q_{ALT} as per the ALT algorithm. We then select the buffer size as $\min\{Q_{eBDP}, q_{ALT}\}$ to yield a hybrid algorithm, referred to as the A* algorithm, which combines the eBDP and the ALT algorithms.

When channel conditions change, the A* algorithm uses the eBDP measured service time to adjust the buffer size promptly. The convergence rate depends on the smoothing weight W. The A* algorithm can further use the ALT algorithm to fine-tune the buffer size to exploit the potential reduction due to statistical multiplexing.

The basic impetus for the design of the A* algorithm is to exploit the possibility of statistical multiplexing to reduce buffer sizes. The A* algorithm can achieve significantly smaller buffer sizes when multiplexing exists. The A* algorithm is able to achieve high throughput efficiency across a wide range of operating conditions while minimizing queuing delays. Compared to eBDP algorithm A* algorithm is capable of exploiting the statistical multiplexing where feasible. In particular, significantly lower delays are achieved with 10 download flows while maintaining comparable throughput efficiency.

III. CONCLUSION

Buffer plays a key role in 802.11/802.11e wireless networks. Buffers are used to accommodate short-term packets so as to mitigate packet drops and to maintain high link efficiency. Packets are quequed if too many packets arrive in a sufficiently short interval of time during which a network device lacks the capacity to process all of them immediately. The use of fixed size buffers in 802.11 networks in 802.11 networks inevitably leads to either undesirable channel underutilization or unnecessary high delay across a wide range of network conditions. The main objective of this paper is to maintain high network utilization while providing low queuing delays in 802.11 wireless networks through dynamic buffer sizing algorithms.

IV. ACKNOWLEDGMENT

My heartful gratitude to almighty God and my parents - father D.Chatur Naik and mother D.Ghammi Bai without whose unsustained support, I could not have completed this paper.

V. REFERENCES

[1]. G. Appenzeller, I. Keslassy, and N. McKeown, “Sizing router buffers,” in Proc. ACM SIGCOMM, 2004, pp. 281–292.

[2]. A. Dhamdher and C. Dovrolis, “Open issues in router buffer sizing,” Comput. Commun. Rev., vol. 36, no. 1, pp. 87–92, Jan. 2006.

[3]. R. Stanojevic, C. Kellett, and R. Shorten, “Adaptive tuning of drop-tail buffers for reducing queueing delays,” IEEE Commun. Lett., vol. 10, no. 7, pp. 570–572, Jul. 2006.

[4]. C. Villamizar and C. Song, “High performance TCP in ANSNET,” Comput. Commun. Rev., vol. 24, no. 5, pp. 45–60, Oct. 1994.

[5]. G. Vu-Brugier, R. Stanojevic, D. Leith, and R. Shorten, “A critique of recently proposed buffer-sizing strategies,” Comput. Commun. Rev., vol. 37, no. 1, pp. 43–48, Jan. 2007.

[6]. A. Dhamdhere, H. Jiang, and C. Dovrolis, “Buffer sizing for congested internet links,” in Proc. IEEE INFOCOM 2005, pp. 1072–1083.

[7]. D. Wischik and N. McKeown, “Part I: buffer sizes for core routers,” ACM SIGCOMM Computer Commun. Review, vol. 35, pp. 75–79, July 2005.

[8]. K. Avrachenkov, U. Ayesta, A. Piunovskiy, “Optimal choice of the buffer size in the Internet routers,” in Proc. IEEE CDC 2005, pp. 1143-1148.

[9]. Traffic measurements (online), available at: http://www.caida.org/tools/measurement/skitter/RSSAC/

[10]. D. Malone, K. Duffy, and D. J. Leith, “Modeling the 802.11 distributed coordination function in non-saturated heterogeneous conditions,” IEEE/ACM Trans. Netw., vol. 15, no. 1, pp. 159–172, Feb. 2007.

[11]. S. A. Mujtaba et al., “TGn Sync proposal technical specification,” IEEE 802.11-04/889r6, May 2005 [Online]. Available: www.tgnsync.org

[12]. T. Li, Q. Ni, D. Malone, D. Leith, T. Turletti, and Y. Xiao, “Aggregation with fragment retransmission for very high-speed WLANs,” IEEE/ACM Trans. Netw., vol. 17, no. 2, pp. 591–604, Apr. 2009.

[13]. K. Kumaran, M. Mandjes, and A. L. Stolyar, “Convexity properties of loss and overflow functions,” Oper. Res. Lett., vol. 31, no. 2, pp. 95–100, 2003.

Short Bio Data for the Author

Depavath Harinath, received the Bachelor of Science degree in computerscience from New Noble Degree college, Affiliated to Osmania University, Hyderabad, A.P, India in 2008 and received Master of Computer Applications degree from Sreenidhi Institute of Science and Technology, Accredited by NBA, AICTE, New Delhi - Permanently Affiliated to JNTU,Ghatkesar, Ranga Reddy, Hyderabad, A.P., India in 2012. My research Interests are Computer Networks, Network Security.