Performance Improvement of 802.11 Wireless Access Network with TCP ACK Agent and Auto-Zoom Backoff Algorithm

Qixiang Pang[1], Soung C. Liew[2], Victor C.M. Leung[1]

[1] Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, BC, Canada V6T 1Z4

[2] Department of Information Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong

***Abstract*—This paper proposes two schemes to improve the TCP performance over 802.11 WLAN access protocol. In the standard 802.11 and TCP protocols, a TCP data segment is acknowledged twice, once at the MAC layer, and once at the TCP layer. By a simple cross-layer design, a TCP ACK agent is installed at the WLAN AP. When a MAC acknowledgment is received by the AP, the AP generates a TCP ACK on behalf of the WLAN stations. Together with the TCP ACK agent scheme, an auto-zoom backoff algorithm is proposed to further improve the access performance. In the auto-zoom backoff, the contention windows can be progressively reduced to a very low value in case of light or asymmetric traffic, and return to normal quickly when collision occurs. With the TCP ACK agent and auto-zoom backoff, we show that significant improvement in TCP throughput performance can be achieved in typical Internet application scenarios.**

Keywords - IEEE 802.11, Cross-Layer Protocols, WiFi, WLAN, MAC, TCP over WLAN, data access

I. INTRODUCTION

The use of wireless local area networks (WLANs) in homes, offices, and public areas is growing by leaps and bounds [1][2]. The Media Access Control (MAC) and Physical (PHY) layers in the WLAN are defined by the IEEE 802.11 standard [2]. Most WLAN equipment uses the Distributed Coordination Function (DCF) defined in the standard to coordinate channel access. DCF belongs to the class of algorithms that employ Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). In DCF, the basic access is a two-way handshake procedure. A successful data frame transmission is acknowledged by the receiver.

Current use of WLANs for Internet access from wireless stations is dominated by downlink TCP traffic; however, the characteristics of TCP have not been sufficiently taken into account for data transport over WLANs. One characteristic of TCP transport over WLAN is the duplication of acknowledgment (ACK) functions at the MAC and transport layers. Each TCP data segment transmitted across the WLAN will cause a TCP ACK to be transmitted later, in addition to the MAC-layer ACK that is transmitted by the receiver immediately. Although TCP ACKs are small in size, when the overhead at the MAC layer is added, they are not negligible and can degrade the throughput performance significantly. We investigate a scheme that eliminates the duplication of the two acknowledgements on the WLAN. We show that this scheme produces significant performance improvements together with an auto-zoom backoff algorithm, particularly in the infrastructure-mode WLAN in which all traffic flows through an access point (AP).

The remainder of this paper is organized as follows. Section II discusses performance limiting factors in 802.11 WLAN. Among them, the most severe one is the duplication of TCP ACK function and MAC layer ACK function. Section III proposes two complementary schemes – TCP ACK agent and auto-zoom backoff to solve the issues discussed in Section II. Numerical results are presented to demonstrate the significant performance improvements that can be realized. Section IV concludes the paper.

II. PERFORMANCE LIMITING FACTORS IN 802.11 WIRELSS ACCESS NETWORKS

The 802.11 WLAN is widely used for wireless access to the Internet. However, although the Internet is dominated by TCP traffic, the characteristics of TCP have not been sufficiently taken into account in the original 802.11 WLAN design.

Figure 1 shows the most common application scenario of 802.11 WLAN as a wireless access network. When TCP traffic (such as from Web browsing, FTP, and e-mail) is transmitted over an infrastructure WLAN, four performance limitations are as follows:

1) Functional duplication between TCP and MAC layers

When a TCP DATA segment is sent by the AP, two ACKs are generated by the receiver: one at the MAC layer and one at the TCP layer. The TCP ACK is not needed in principle, since the MAC ACK confirms that the TCP DATA has indeed been received by the MAC receiver, which in this case is also the end point of the TCP connection. In 802.11, header and other MAC layer overhead are considerable. Even though the payload of a TCP ACK is small, the corresponding MAC DATA frame for the TCP ACK can use up a considerable amount of airtime. Take 802.11b. The values of relevant

This work was sponsored by the AoE scheme under the University Grant Committee of Hong Kong (AoE/E-01/99) and the Canadian Natural Sciences and Engineering Research Council through grant STPGP 257684-02.

parameters of 802.11b are listed in Table I. Assuming the payload of the MAC DATA frame for TCP DATA is 1500 bytes, is the airtime used up by the MAC DATA frame and the returning ACK frame is

$$50_{\text{DIFS}} + 192_{\text{PHY}} + 28 \times 8/11_{\text{MAC}} + 1500 \times 8/11_{\text{data}} + 10_{\text{SIFS}} + 192_{\text{PHY}} + 14 \times 8/2_{\text{ACK}} \approx 1620\mu s$$

Similarly, the airtime required for transporting the TCP ACK is:

$$50_{\text{DIFS}} + 192_{\text{PHY}} + 28 \times 8/11_{\text{MAC}} + 40 \times 8/11_{\text{data}} + 10_{\text{SIFS}} + 192_{\text{PHY}} + 14 * 8/2_{\text{ACK}} \approx 560\mu s$$

Elimination of the TCP ACKs can lighten the airtime usage by more than 25%.

2) Asymmetric traffic and unbalanced load

The original MAC protocol in the 802.11 standard does not exploit the asymmetry of traffic flows to optimize performance. All nodes are treated the same way as far as the MAC protocol is concerned. When the traffic is mostly downlink TCP data traffic from an AP, the uplink traffic will be mostly TCP ACKs.

3) Channel contention and collisions among AP and stations

The downlink TCP DATA and the uplink TCP ACK from the stations contend for the wireless channel. Collisions of the MAC DATA frames in two directions may result in further airtime wastage.

4) Unnecessarily long backoff time

In the original 802.11 algorithm, after a successful packet transmission, the station must wait for an average of (CWmin–1)/2 time slots before transmitting the next packet. If most of the WLAN traffic comes from the AP, the number of active stations at any particular moment in time can be quite low. Therefore setting CWmin at 32 and 16, as in 802.11b and 802.11a/11g, respectively, is not optimal.

The issues listed above have not been carefully considered. Some related work is briefly reviewed as follows. References [3]-[8] discussed many variations of TCP to improve performance over wireless links, including TCP SACK, Snoop, Split-connection, TCP Veno, and TCP Westwood. The focus was on how to distinguish packet loss due to transmission errors from packet loss due to traffic congestion, so as to prevent TCP from adjusting its congestion window in an inappropriate manner. No particular attention was paid to the specifics related to the 802.11 WLAN operation. Reference [9] proposed a mechanism to improve TCP performance over WLAN. However the scheme in [9] only discussed how to avoid collisions between stations or between the AP and stations. It requires tightly coupled timing relation between the MAC and TCP layers and it is not applicable to the more common asymmetric TCP traffic scenarios considered in this paper. In particular, the duplication of TCP ACK and MAC ACK was not considered in [9], therefore its improvement is far from optimal.

In this paper, we consider the elimination of the above performance limiting factors using a simple cross-layer design. As far as we know, this is a first investigation in this direction. As will be demonstrated, significant throughput enhancements can be obtained

III. TCP ACK AGENT AND AUTO-ZOOM BACKOFF SCHEMES

1) TCP ACK Agent

The observations in Section II suggest the use of a TCP ACK agent to improve performance. The TCP ACK agent is a cross-layer entity located at the AP. As usual, when a station correctly receives a MAC data frame, it returns a MAC ACK frame to the AP. However, the transmission of TCP ACK from the station to the AP is eliminated. The TCP ACK agent generates TCP ACKs on behalf of the TCP receiver (station), and sends the TCP ACKs to the TCP sender located in the wired network.

In a real system, the reception of data at the MAC layer does not guarantee the reception at the TCP layer due to possible software errors, buffer overflows, etc. To tackle the potential unreliability between the MAC and TCP layers, two solutions are proposed as follows. Both use two pairs of virtual TCP receivers/senders in the WLAN – one at the AP WLAN NIC and the other at the station NIC.

The first solution uses a buffer at the MAC layer of the station (TCP client). The mechanism is depicted in Figure 2. When the MAC layer has forwarded the received data frame to its upper layer (the TCP layer), instead of dropping it immediately, the MAC layer buffers the data frame (TCP data enclosed) until the TCP ACK for it is received. When the MAC layer receives the TCP ACK from the TCP layer locally, both the TCP ACK and the buffered data frame are discarded. Of course, when the buffer is full, the newly arriving data frame from the AP will be discarded and no MAC ACK will be generated.

In the second solution, no MAC layer buffer is needed. Two types of MAC ACK frames are defined. The two ACK frames are differentiated by a "TCP ACK agent flag". Only when the AP receives a MAC frame with TCP ACK agent flag being ON will the TCP ACK agent generate a TCP ACK. When the flag in the received MAC ACK is OFF, the TCP ACK agent at the AP will not generate a TCP ACK. By default, the flag is set to OFF. At the station, when a MAC data frame is received, it is forwarded to the TCP layer. After that, the station will wait for a certain time (≤SIFS). If within the expiry of the time, MAC layer receives the TCP ACK for the TCP data just sent to the higher layer, a MAC ACK frame with the "TCP ACK agent flag" = ON will be sent to the AP when SIFS expires. Otherwise, a default MAC ACK (the flag is OFF) will be sent. Thus, we can avoid the problem mentioned above. When the station is installed with a slow processor (e.g., PDA), due to processing delay, it may not be able to send TCP ACK to the MAC layer within SIFS time (10 μs in 802.11b and 16 μs in 802.11a/11g) even if the TCP layer successfully receives the TCP data. The solution is to use differentiated SIFS as in 802.11e. The station may notify the AP its selected SIFS according to its processing power beforehand. The communication between the AP and the station follows the selected SIFS that is specific to the station. The duration field in the MAC data frame from the AP can tell other stations how long they should wait.

From the descriptions of the two solutions, it can be seen that the second one involves more modification to the existing MAC layer and therefore the first one is preferred.

2) Auto-Zoom Backoff

When a TCP ACK agent is deployed, the standard backoff algorithm can be further improved with an auto-zoom backoff algorithm. The improvement is mainly based on observation (4) in Section II. In the auto-zoom backoff algorithm, besides CWmin and CWmax, one more parameter, minCWmin < CWmin, is added. The minCWmin is used to specify the minimum value that the new contention window can take.

As shown in Figure 3, if consecutive transmissions are successful, the contention window can be progressively reduced to values below CWmin. Consecutive successful transmissions mean the contention intensity is light or the traffic is asymmetric. Therefore it is not necessary to adopt the relatively high CWmin value (32 in 802.11b and 16 in 802.11a/11g) for a new contention window. A smaller value should be adopted instead. However, the new value should be carefully selected since the contention window below CWmin is a critical region. In our algorithm, the contention window evolution below CWmin follows a linear relation. For each successful transmission in this region, the contention window is reduced by 1 until minCWmin is reached. If a collision occurs while the contention window is in this region (smaller than CWmin), the contention window is doubled and if the doubled value is still smaller than CWmin, CWmin is adopted. By doing so, the contending stations can quickly evolve to a large contention window when contention is high. Using the quick-jump and linear decrease mechanisms, even when the number of contending stations is large, there is no detrimental effects on performance. On the other hand, when the contention is light, significant improvements can be obtained as will be shown.

Figure 4 compares our new schemes with the original scheme. Note that when the AP is the only node sending TCP data on the WLAN, there is no collision and the contention window of AP operates at the minimum value - minCWmin.

3) Performance Study

The performance of the new schemes is examined as follows. Since we are mainly concerned with the WLAN as the bottleneck, the bandwidth between the TCP server and AP is assumed to be infinite in our simulation. TCP Reno is used. The TCP DATA segment size is 1480 bytes, which includes a 20-byte TCP header. The IP header is an additional 20 bytes. The minCWmin is assigned a value of 2. The NS-2 simulator is used [10]. In our simulation, we assume that the potential unreliability between the MAC and TCP layers have been removed using one of the two methods proposed in the previous section.

Figure 5 and Figure 6 give the throughput comparison in 802.11b and 802.11a/g WLANs respectively. Only downlink TCP flows from the AP to the stations are activated. There is one TCP connection to each WLAN station. As can be seen from Figure 5 and Figure 6, substantial throughput improvements can be achieved by the TCP ACK Agent and auto-zoom backoff. Specifically, compared with the standard 802.11b, 60% increase in throughput can be achieved; and compared with the standard 802.11a/g, 65% throughput increase can be achieved.

Figure 5 and Figure 6 also present the TCP throughput when the AP running the standard backoff algorithm is assigned a smaller CWmin (which is 2). Doing so may actually result in worse throughput than that of the standard algorithm. The reason is that if TCP ACKs are transmitted on the WLAN, there will be collisions. Setting small CW_{min} will cause the collision probability to increase.

The TCP throughput when TCP ACK agent and auto-zoom backoff are applied can be estimated by the following equation [11][1]:

$$S_{TCP} = TCP\_DATA \Big/ \left(\frac{(TCP\_DATA + IP + MAC)}{DataRate} + \frac{ACK}{BasicRate} + 2PHY + SIFS + DIFS + T_{slot} \frac{(\min CW_{\min} - 1)}{2} \right)$$

where,

TCP_DATA is the TCP DATA segment size including TCP header; constant size is assumed;
IP is the size of IP header (i.e., 20 bytes);
MAC is the MAC header in bits (28×8=224 bits);
DatsaRate is the physical layer data rate;
PHY is the overhead in physical layer;
SIFS is the time of SIFS;
DIFS is the time of DIFS;
ACK is the size of MAC ACK in bits (14×8=112 bits);
BasicRate is the rate for MAC ACK transmission;
T_{slot} is the time of one slot.

The throughput improvement is even higher when TCP DATA segment size is smaller (Figure 7). For example, when TCP DATA size is 500 bytes, more than 100% improvement is obtained. The reason is that with smaller TCP DATA size, the TCP ACK overhead will be more significant comparatively, and therefore the elimination of TCP ACKs will lead to higher improvements.

We believe that the asymmetric scenario as depicted in Figure 1, or one in which there is little uplink TCP traffic compared to the downlink TCP traffic, is one that arises most often in practice. However, it would still be interesting to investigate situations in which downlink and uplink TCP traffic is symmetric to see if the new schemes continue to work well.

We study a scenario in which there are 20 downlink TCP flows and 20 uplink TCP flows over an 802.11b infrastructure WLAN. Altogether, there are 40 wireless stations in addition to the AP. Only the uplink TCP ACKs are eliminated through the TCP ACK agent mechanism; the downlink TCP ACKs for the uplink TCP DATA are not eliminated. With the TCP ACK agent and auto-zoom backoff, the total throughput is about 6 Mbps. The original scheme can only produce a throughput of 4.4 Mbps. Again, the throughput improvement is obvious.

IV. CONCLUSIONS

Two complementary schemes for transporting TCP traffic over 802.11 WLAN have been proposed in this paper to

improve TCP throughput over WLAN. The enhancements are motivated by noting the duplication of the functionality of TCP ACKs and MAC ACKs in WLAN; and by the observation that TCP traffic in WLAN can be highly asymmetric, with mostly downlink traffic and little uplink traffic.

The first scheme consists of a TCP ACK agent at the AP that sends TCP ACKs to the servers within the Internet on behalf of the wireless stations. In this way, uplink TCP ACKs over the WLAN can be eliminated. The second scheme consists of an auto-zoom backoff algorithm that reduces long backoff time in the multiaccess protocol when traffic in the WLAN is mostly generated by the AP. Integration of the two scheme results in significant TCP throughput improvements – more than 50% - as compared to the standard 802.11 protocol.

Our proposed schemes apply to infrastructure networks. Whether similar schemes can be devised for multi-hop networks [13] [14] is an interesting subject for further study.

REFERENCES

[1] M. S. Gast, 802.11 Wireless Networks: The Definitive Guides, O'Reilly, 2002.

[2] IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ISO/IEC 8802-11:1999E), Aug. 1999.

[3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," IEEE/ACM Trans. Networking, vol.5, no.6, pp.756-769, December 1997.

[4] V. Jacobson, "Congestion avoidance and control," ACM SIGCOMM 1988, pp.314-329.

[5] W. R. Stevens, "TCP/IP illustrated," Addison-Wesley, 1994.

[6] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: end-to-end congestion control for wired/wireless networks," Wireless Networks, vol.8, no.5, pp.467-479, 2002.

[7] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for wireless access networks," IEEE J. Select. Areas Commun., vol.21, no.2, pp.216-228, February 2003.

[8] Q. Pang, S. C. Liew, C. P. Fu, W. Wang, and V. O. K. Li, "Performance study of TCP Veno over WLAN and RED router," IEEE GLOBECOM 2003, pp.3237-3241.

[9] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, "Performance of reliable transport protocol over IEEE 802.11 wireless LAN: analysis and enhancement," IEEE INFOCOM 2002, pp.599-607.

[10] NS-2, URL http://www.isi.edu/nsnam/ns.

[11] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," IEEE/ACM Trans. Networking, vol.8, no.2, pp.133-145, Apr. 2000.

[12] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," Comput. Commun. Rev., vol.27, no.3, pp.67-82, July 1997.

[13] C. Ng, and S. C. Liew, "Offered Load control in IEEE 802.11 Multi-hop Ad-hoc Networks," The 1st IEEE International Conference on Mobile Ad-hoc and Sensor System, Oct 2004.

[14] C. Ng and S. C. Liew, "Re-routing Instability in IEEE 802.11 Multi-hop Ad-hoc Networks, " *The 4th IEEE International Workshop on Wireless Local Network, Nov 2004*

TABLE I. PARAMETERS OF 802.11B, 802.11A/G IN NS-2 SIMULATION

	802.11b	802.11a/g
CWmin	32	16
CWmax	1024	1024
SlotTime	20 μs	9 μs
CCATime	15 μs	3 μs
RxTxTurnaroundTime	5 μs	2 μs
SIFSTime	10 μs	16 μs
PHY overhead	192 μs	20 μs
MaxPropagationDelay	2 μs	0.5 μs
DataRate	11 Mbps	54 Mbps
BasicDataRate	2 Mbps	6 Mbps

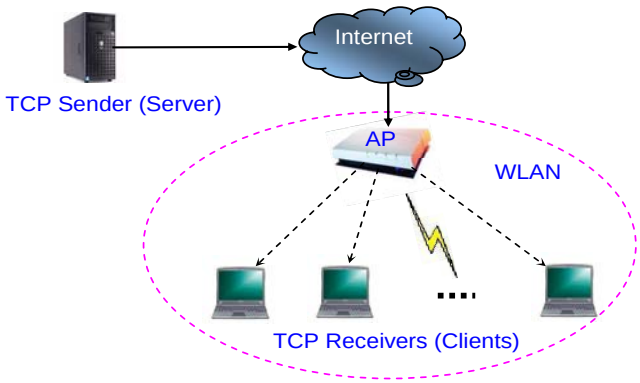


Figure 1. TCP traffic flows over infrastructure WLAN

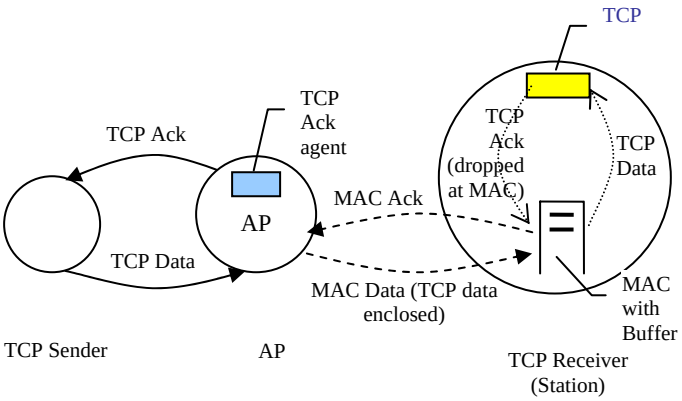
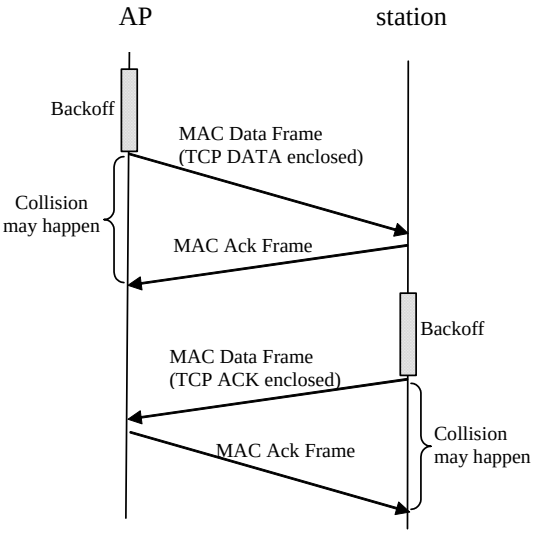


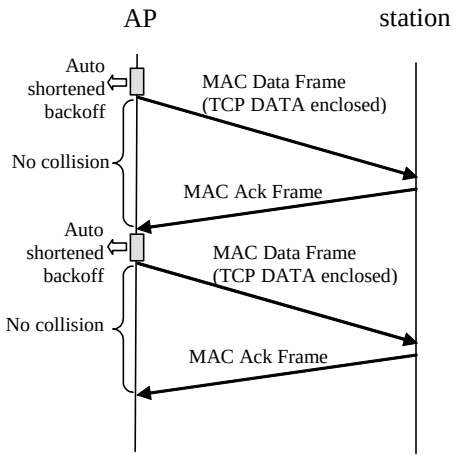
Figure 2. Implementation of TCP ACK agent scheme

- $cw \leftarrow CW_{min}$ *if (success and* $cw > CW_{min}$*).*
- $cw \leftarrow max\ (--CW, minCW_{min})$, *if (success and* $cw \leq CW_{min}$*).*
- $cw \leftarrow min\ (CW_{max}, max\ (cw*2, CW_{min}))$, *if collision.*

Figure 3. The auto-zoom backoff algorithm



(a) original scheme



(b) TCP ACk agent + auto-zoom

Figure 4. Comparison of traffic flows

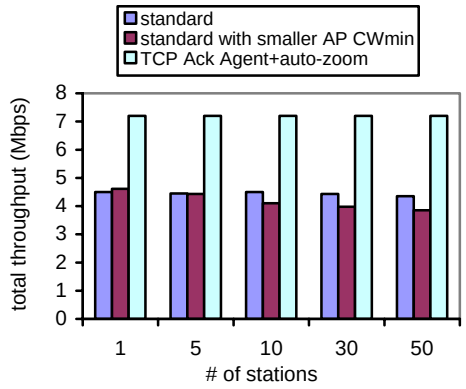


Figure 5. Downlink TCP throughput comparison – 802.11b

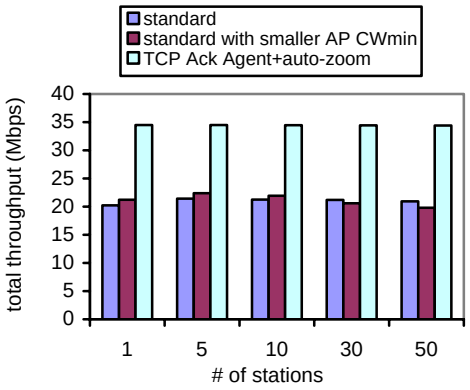


Figure 6. Downlink TCP throughput comparison – 802.11a/11g

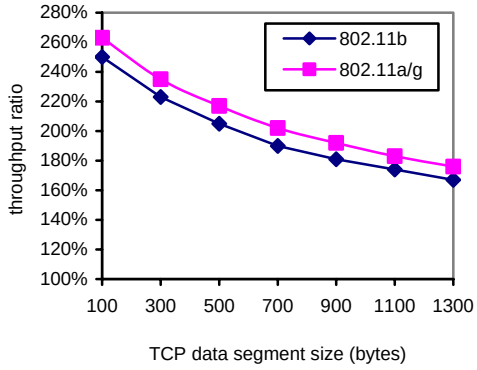


Figure 7. TCP throughput increment versus TCP data segment size