Improving Throughput Performance of the IEEE 802.11 MAC Layer Using Congestion Control Methods*

Song Ci
CS Department
University of Michigan-Flint
Flint, MI48502
cisong@umich.edu

Guevara Noubir
College of CS
Northeastern University
Boston, MA02115
noubir@ccs.neu.edu

Hamid Sharif
CEEN Department
University of Nebraska-Lincoln
Omaha, NE68182
hsharif@unl.edu

Abstract

In this paper, we will present and analyze adaptive fragmentation algorithms for enhancing throughput performance under a slow fading channel. We will first study the similarities between the TCP protocol and MAC protocol. Next, we will propose an adaptive approach to change the fragmentation size dynamically according to variations of the wireless channel quality. Simulation results show that the proposed algorithms can greatly improve the throughput performance of the IEEE 802.11 wireless LAN.

1 Introduction

There is an increasing demand for QoS provisioning in the IEEE 802.11 wireless LAN. As in other wireless networks, the IEEE 802.11 WLAN faces problems such as fading channel, interference, power efficiency, and so forth. Additionally, the IEEE 802.11 network is more complex than other wireless data networks, since it uses the random access mechanism and the half duplex channel. This makes its performance more susceptible to be affected by retransmissions than other wireless data networks such as cellular networks. In general, retransmissions in the IEEE 802.11 wireless LAN are caused by collisions and bit errors. Collisions are caused by the characteristics of random backoff algorithm and other factors such as hidden terminals; frame errors are caused by interference, fading, and noises in wireless channels. Normally, collisions also appear as frame errors at the receiver end.

Unlike its wired counterpart, a wireless channel is error-prone and time-varying due to slow fading, fast fading, path loss, shadowing, noise, interference, and so forth. This causes a very high frame error rate at the receiver and results in a lot of retransmissions. As a consequence, the channel efficiency is severely degraded [8, 5]. On the other hand, current wireless systems are generally designed to use fixed MAC parameters such as frame size. This is obviously not efficient in terms of the channel utilization [4]. Even though some optional measures, such as data rate drafting and power control, have been mentioned in the IEEE 802.11 standards [1], no further specification on these issues are defined.

Link adaptation techniques can improve the system performance of wireless LANs by changing the protocol parameters, according to the channel quality and the network load. These techniques have been studied in many research works. In [8, 7], the optimal frame size prediction in wireless networks has been studied. In [8], there are two approaches proposed for obtaining the optimal frame size with

*This paper was partially published on proceeding of the 16th ACM Symposium on Applied Computing, Las Vegas, NV, 2001.

the wireless ARQ protocol. In [7], the trade-off between the protocol overhead and the payload size is studied by gathering physical measurements with the Lucent WAVELAN radio. Generally speaking, the idea behind these works is to achieve maximum throughput by dynamically changing the frame size according to variations in wireless channel quality. When the channel quality is good, a longer frame size could be used; similarly, when the channel quality is bad, a shorter frame size is used to lower the number of retransmissions. In [6], link adaptation under a Rayleigh fading channel is addressed on frame size, equalizer, and power control.

In this paper, we will propose and analyze several adaptive fragmentation algorithms for enhancing throughput performance under a slow fading channel. The paper is organized as follows. A comparison of TCP protocol and MAC protocol is given in section 2. The proposed fragment adaptation algorithms are presented in section 3. A simulation is described and the results are given and analyzed in section 4.

2 Characteristics of TCP Protocol and MAC Protocol

The TCP protocol is used to provide end-to-end error-free data transfer services. When transmitting data of connection-oriented applications across connectionless networks such as the Internet, we cannot detect whether or not congestions occur inside the network. Congestion is usually caused by unpredictable network load that may trigger buffer overflows in network nodes. As a consequence, packets are discarded whenever a congestion occurs. The discarded packets will be retransmitted after the timeout occurs at the sender. In the TCP protocol, all packet losses are assumed to be caused by congestions.

On the other hand, the MAC protocol defines a set of functionalities to decide when and how a station can access and use the physical medium. In the MAC protocol, packet losses are assumed to result from collisions that occur during transmissions.

The MAC protocol, in terms of resource allocation, is similar to the TCP protocol. In both protocols, the primary concern is how to allocate the network resources efficiently. In the TCP protocol, end-to-end congestion-control algorithms have been developed for improving the throughput performance, where the congestion window size is used as a unit of resource allocation. Through dynamically changing congestion window sizes at the sender and the receiver, congestions will be controlled or avoided. In the MAC protocol, the frame size can be used as the unit of resource allocation, since it is closely related to the overall channel access time and channel fading rate.

In both the TCP protocol and MAC protocol, when a packet is not acknowledged successfully or is discarded, a retransmission will be scheduled. Retransmissions are the main causes of poor throughput performance and channel utilization. Due to the end-to-end nature and coarse-scale timer used in the TCP protocol, the cost of transport layer retransmissions is much higher than that of MAC layer retransmissions. Note that even though the MAC ACK can shorten the delay overhead of retransmissions, it is still a main factor of performance degradation, especially in the wireless environment, where retransmissions are mainly caused by frame errors rather than collisions.

3 Adaptive Fragmentation Algorithms for Throughput Enhancement

In the IEEE 802.11 wireless LAN, its MAC protocol is much different from its wired counterpart such as Ethernet. There is only one radio in a wireless LAN terminal, which can either transmit or receive but cannot do both simultaneously. Therefore, unlike in Ethernet where collisions can be detected right away after frames having been sent, it is very difficult to detect collisions occurring in a wireless LAN in a timely manner. Moreover, the hidden station problem increases the number of collisions due to impairments on channel sensing mechanisms and interferences [2]. Additionally, wireless channels are

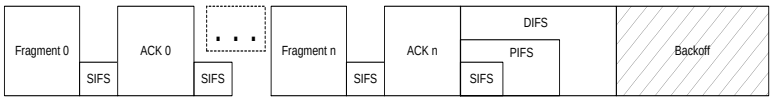

Figure 1: Basic Access Method of MAC Fragmentation

especially error-prone and bursty, often causing serious degradation in channel utilization. All of these factors make the design of the wireless LAN MAC layer protocol difficult.

In the current IEEE 802.11 MAC protocol [1], Distributed Coordination Function (DCF) and Point Coordination Function (PCF) are defined to provide a collision-free multiple access environment by physical carrier sensing mechanism plus virtual carrier sensing mechanism. Several performance enhancement methods specific for the wireless LAN, such as MAC layer Acknowledgement (ACK), MAC layer fragmentation, etc., are proposed in the current standard [1]. Essentially, all these measures focus on improving link reliability and reducing the number of retransmissions.

The goal of the MAC layer fragmentation is to lower the frame error rate when interferences caused by microwave ovens, Bluetooth terminals, and cordless phones occur. In general, the channel efficiency has the following relation with the frame size

$$\gamma = \frac{1}{(1-P_b)^{-L}} \tag{1}$$

here, $L = l + h$ is the total size of a frame, l is the payload size of a frame, and h is the total overhead that includes header size of a frame, acknowledgement and time of waiting an acknowledgement. P_b is the bit error rate under a given channel quality. Thus, the frame size should be determined in such a way that it satisfies the specified frame error rate. This could be done by fragmenting a longer frame into several shorter fragments when the channel quality is bad. This is illustrated in Figure 1.

Although fragmentation is useful to improve the channel throughput, there is no specification on how to choose the fragmentation threshold in the current standard [1]. The drawback of fragmentation is that it will increase the overall overhead. There is always a balance between the fragmentation overhead and the throughput improvement. Based on exploring the similarities between the TCP protocol and the MAC protocol, new adaptive fragmentation algorithms are proposed in a heuristic manner of studying congestion control algorithms [4].

ADAPTIVE FRAGMENTATION ALGORITHM 1:

Input: θ_k is the fragment size at time k
FRG_{min} and FRG_{max} are respectively
the minimum frame size and the maximum frame size
Parameters of uniform distributions ν and ω

Output: The fragmentation size used in the next transmission.

Local Vars: n and m are uniform r.v.

if ACK times out
generate $n \in [1, \nu]$ with a uniform dist.;
$\theta_{k+1} = \theta_k \div n$;
if $(\theta_{k+1} < FRG_{min})$
$\theta_{k+1} = FRG_{min}$;

else

generate $m \in [1, \omega]$ with a uniform dist.
$\theta_{k+1} = \theta_k \times m$;
if $(\theta_{k+1} > FRG_{max})$
$\theta_{k+1} = FRG_{max}$;

end-if

---

where ν and ω are the maximum backoff window sizes used by the increase and decrease procedures respectively. n and m are the backoff slots used in time $k+1$. Note that hereafter we use the time slot to represent the transmission opportunity for simplicity. θ is the adaptive fragmentation threshold used by the sender. In the above algorithm, the fragmentation threshold is increased or decreased exponentially in a random manner [3].

ADAPTIVE FRAGMENTATION ALGORITHM 2:

Input: θ_k is the fragment size at time k
FRG_{min} and FRG_{max} are respectively
the minimum frame size and the maximum frame size
Output: The Fragmentation size used in the next transmission.
if ACK times out
$\theta_{k+1} = \theta_k \div 2$;
if $(\theta_{k+1} < FRG_{min})$
$\theta_{k+1} = FRG_{min}$;
else
$\theta_{k+1} = \theta_k \times 2$;
if $(\theta_{k+1} > FRG_{max})$
$\theta_{k+1} = FRG_{max}$;
end-if

The algorithm 2 is designed by using the method behind the slow-start congestion-control algorithm that is widely adopted in the TCP protocol [9]. If ACK is lost or timed out, the adaptive fragmentation threshold will be decreased by half; in other words, in time $k + 1$, the threshold will be one-half of the threshold in time k. Similarly, if ACK is received successfully, the fragmentation threshold will be doubled in time $k + 1$. In this algorithm, the fragmentation threshold should be less than the maximum frame size and larger than the minimum frame size.
Both the algorithm 1 and 2 adopt the exponential increase and decrease of the fragmentation threshold. This may cause an overreaction due to the nature of exponential algorithm, especially when the channel quality is changing slowly. In order to mitigate or to avoid this deficiency, we design the following fragmentation adaptation algorithms.

ADAPTIVE FRAGMENTATION ALGORITHM 3:

Input: θ_k is the fragment size at time k
FRG_{min} and FRG_{max} are respectively
the minimum frame size and the maximum frame size
δ is the fragment increase step

The parameter of a uniform distribution ν

Output: The fragmentation size used in the next transmission.

Local Vars: n and m are uniform r.v.

if ACK times out
  generate $n \in [1, \nu]$ with a uniform dist.;
  $\theta_{k+1} = \theta_k \div n$;
  if $(\theta_{k+1} < FRG_{min})$
    $\theta_{k+1} = FRG_{min}$;

else
  $\theta_{k+1} = \theta_k + \delta$;
  if $(\theta_{k+1} > FRG_{max})$
    $\theta_{k+1} = FRG_{max}$;

end-if

In algorithm 3, the fragmentation threshold is decreased in a random exponential way; but increased in an additive way. The algorithm 4 is designed by using an idea behind an improved version of slow-start algorithm [9]. In algorithm 4, a preset limit ϵ is specified to slow down the further increase of fragmentation threshold beyond the limit.

ADAPTIVE FRAGMENTATION ALGORITHM 4:

Input: θ_k is the fragment size at time k
FRG_{min} and FRG_{max} are respectively
the minimum frame size and the maximum frame size
δ is the fragment increase step
Optimal fragmentation threshold ϵ under a certain channel quality
The parameter of a uniform distribution ν

Output: The fragmentation size used in the next transmission.

Local Vars: n and m are uniform r.v.

if ACK times out
  generate $n \in [1, \nu]$ with a uniform dist.;
  $\theta_{k+1} = \theta_k \div n$;
  if $(\theta_{k+1} < FRG_{min})$
    $\theta_{k+1} = FRG_{min}$;

else
  if$(\theta_k < \epsilon)$
    $\theta_{k+1} = \theta_k \times 2$;
  else
    $\theta_{k+1} = \theta_k + \delta$;
  if $(\theta_{k+1} > FRG_{max})$
    $\theta_{k+1} = FRG_{max}$;

end-if

---

where ϵ is the global optimal fragmentation threshold under a given network scenario. When the last frame is acknowledged correctly, the fragmentation threshold will be increased exponentially until the

fragmentation threshold reaches a preset limit. After that, the fragmentation threshold will be increased additively. In algorithm 5, the global optimal fragmentation threshold is always used for retransmissions while the same increasing scheme as in algorithm 4 is adopted.

ADAPTIVE FRAGMENTATION ALGORITHM 5:

Input: θ_k is the fragment size at time k
FRG_{min} and FRG_{max} are respectively
the minimum frame size and the maximum frame size
δ is the fragment increase step
Optimal fragmentation threshold ϵ under a certain channel quality
The parameter of a uniform distribution ν
Output: The fragmentation size used in the next transmission.
if ACK is timeout
$\theta_{k+1} = \epsilon$;
else
if($\theta_k < \epsilon$)
$\theta_{k+1} = \theta_k \times 2$;
else
$\theta_{k+1} = \theta_k + \delta$;
if ($\theta_{k+1} > FRG_{max}$)
$\theta_{k+1} = FRG_{max}$;
end-if

---

4 Results and Analysis

System Modeling and Simulation Configuration

The measurement results of indoor channel quality, as introduced in [10], are adopted for this simulation. Figure 2 shows the variation of the wireless channel quality. Simulations have been conducted under two channel quality scenarios corresponding to the different noise floors. In this simulation, they are -85dB and -95dB, respectively. Note that even at a same noise floor, the channel quality may still be time-varying.

The parameters chosen for simulations are as follows, in accordance with the IEEE 802.11 MAC standard. The fixed long frame size is set to 1500 bytes. The values of ν and ω used in the proposed algorithms are set to 4. δ is set to 150 bytes and ϵ is set to the optimal fragmentation threshold. According to the current IEEE 802.11 standard, the maximum fragment size should be less than the frame size; the minimum fragment size should be larger than the one-sixteenth of the frame size. In this simulation, the FRG_{min} is 150 bytes and FRG_{max} is 1500 bytes.
Since our primary concern is to evaluate the performance of proposed adaptive fragmentation algorithms, the means of determining the optimal fragment size is beyond the scope of this paper. In this simulation, the optimal fragment size for different channel quality scenarios is derived by running simulations with different fragmentation thresholds. We choose the best one that can achieve the highest throughput as the optimal fragment size for the proposed algorithms [4]. We first run the simulation iteratively from

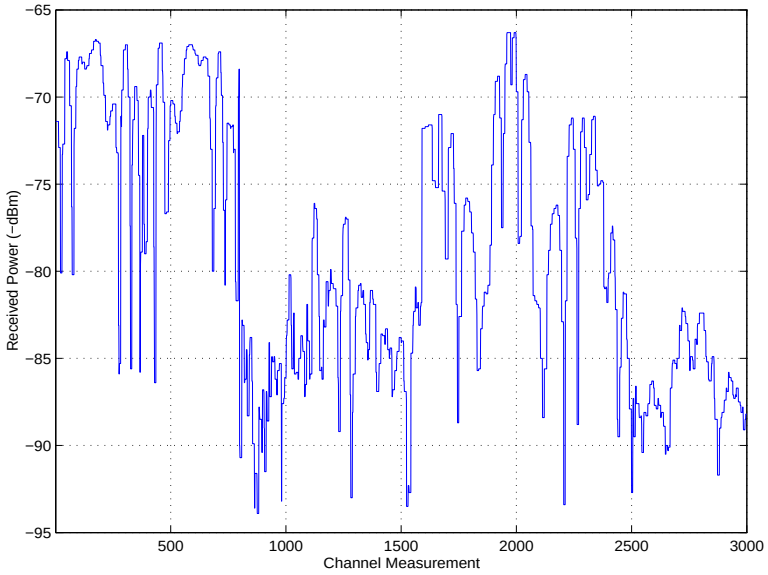

Figure 2: Measurements of Indoor Wireless Channel

the fragment size 100 Bytes to 1500 Bytes with an increase step of 10 Bytes, and then we pick the one that achieves the best throughput performance.
According to the simulation results, the optimal fragment size under the given frame size and given network scenarios is 750 Bytes, which is one-half of the selected long frame size. This is because under the given channel quality and network load, fragmenting a large frame just into two fragments results in the lowest overhead, as illustrated in Figures 3, 4, 5 and 6.
We evaluate the average throughput performance of the proposed algorithms by using the algorithms to transfer files with different file sizes. We run all simulations ten times and compute the mean and confidence intervals of the average good throughput.

Result Analysis

We first evaluated the average good throughput performance by transferring a 100 KB file. Table 1 shows the comparison of throughput performance with different fixed fragmentation threshold, where the noise floor (NF) is -85dB and the number of stations (M) is 5. In this case, the algorithm 5 achieves the best throughput performance due to its constant decrease and slow-start increase procedure. Actually, algorithm 1 can also achieve a better average throughput performance than the fixed fragment size scheme or the non-fragment scheme. In addition, all proposed algorithms can achieve a higher average good throughput than the fixed size scheme. But note that the lower bounds of other algorithms fail to meet the lower bound of 95% confidence interval of the fixed fragment scheme, even though its average performance is better than that of the fixed fragment scheme.

Table 2 is derived under a scenario in which NF=-85dB, M=20. In this case, algorithm 4 achieves the best throughput performance. This is probably due to the fact that it has aggressive decrease and slow-start increase procedure. The lower bound of algorithm 1 is smaller than the lower bound of 95% confidence intervals of the fixed fragment scheme, even though its average performance is better than that of the fixed fragment scheme.
Table 3 is derived for the case where NF=-95dB, M=5, and Table 4 is derived under the scenario with

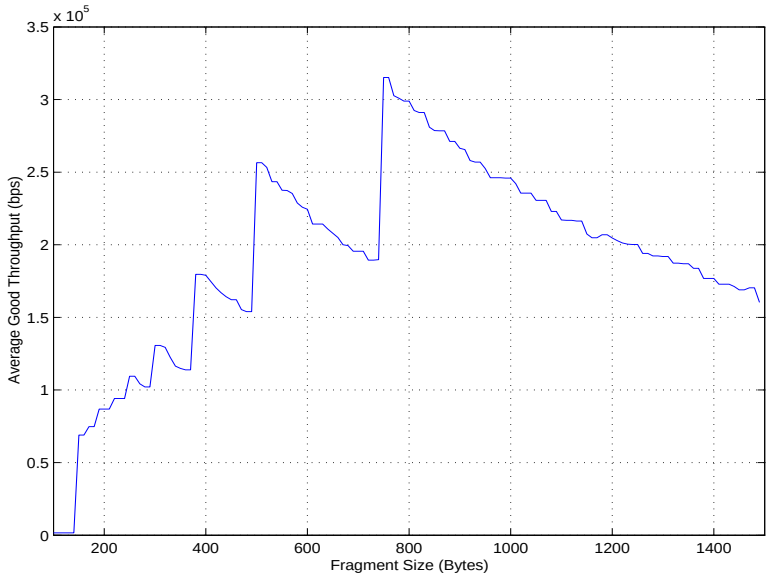

Figure 3: Throughput Performance of Fixed Fragment Scheme (NF=-85dB, M=5)

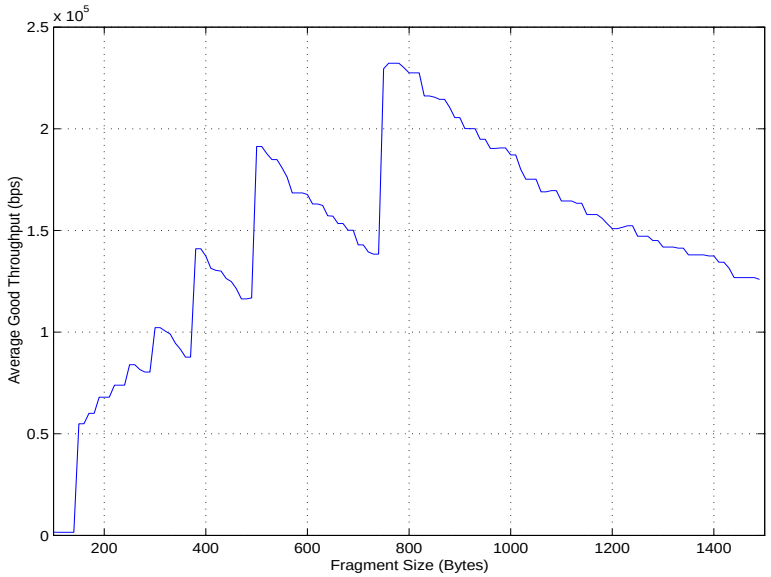

Figure 4: Throughput Performance of Fixed Fragment Scheme (NF=-85dB, M=20)

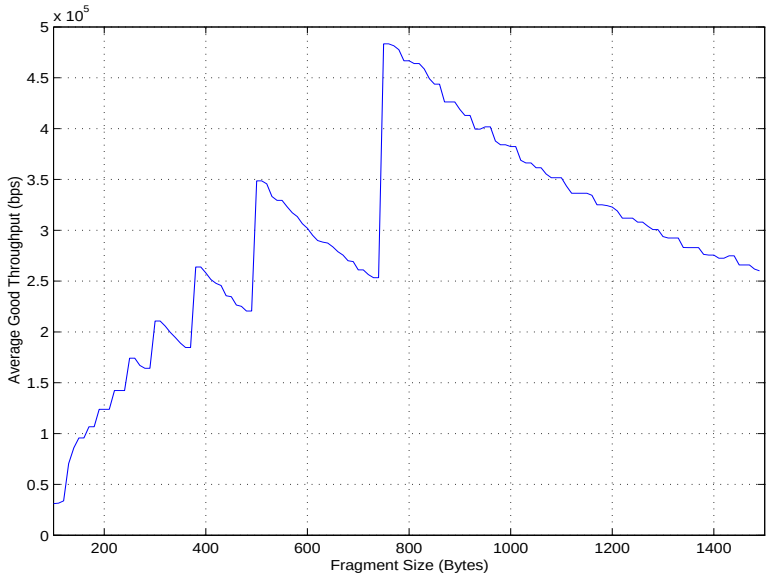

Figure 5: Throughput Performance of Fixed Fragment Scheme (NF=-95dB, M=5)

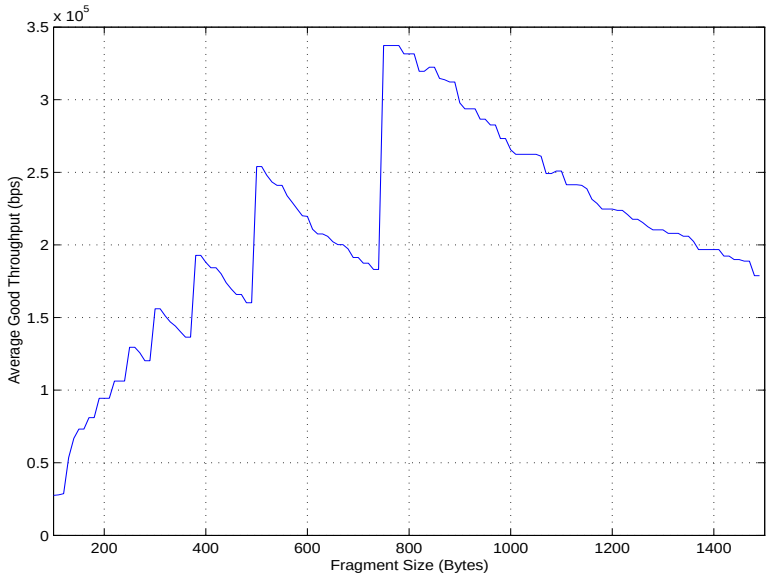

Figure 6: Throughput Performance of Fixed Fragment Scheme (NF=-95dB, M=20)

Table 1: Comparison of Throughput Performance of Different Algorithms with NF=-85dB and M=5

Algorithm	Average good throughput (1×10^5 bps) (95% confidence intervals)
Algorithm 1	3.9254 (3.3569, 4.4940)
Algorithm 2	3.8202 (3.0954, 4.5449)
Algorithm 3	3.9132 (3.0785, 4.7480)
Algorithm 4	3.8075 (2.7885, 4.8265)
Algorithm 5	4.1057 (3.1490, 5.0625)
Fixed fragment size	3.1541 (2.9493, 3.3589)
Non-fragment scheme	3.4923 (3.1245, 3.8601)

Table 2: Comparison of Throughput Performance of Different Algorithms with NF=-85dB and M=20

Algorithm	Average good throughput (1×10^5 bps) (95% confidence intervals)
Algorithm 1	2.8499 (1.9872, 3.7127)
Algorithm 2	2.9253 (2.3825, 3.4681)
Algorithm 3	2.8522 (2.3752, 3.3291)
Algorithm 4	3.0816 (2.3669, 3.7964)
Algorithm 5	3.0030 (2.6169, 3.3890)
Fixed fragment size	2.3551 (2.0618, 2.6484)
Non-fragment scheme	2.6356 (2.2863, 2.9849)

Table 3: Comparison of Throughput Performance of Different Algorithms with NF=-95dB and M=5

Algorithm	Average good throughput (1×10^5 bps) (95% confidence intervals)
Algorithm 1	7.6955 (7.1333, 8.2577)
Algorithm 2	7.7038 (6.9168, 8.4908)
Algorithm 3	7.6565 (7.0586, 8.2545)
Algorithm 4	7.7267 (6.9739, 8.4795)
Algorithm 5	7.5866 (6.7529, 8.4203)
Fixed fragment size	5.0287 (4.8283, 5.2292)
Non-fragment scheme	7.6232 (6.9281, 8.3183)

NF=-95dB, M=20. We can conclude that when the channel quality is good, all proposed algorithms achieve a better performance than the fixed fragment size scheme. But we should note that when the number of users increases, the performance of fragmentation gradually decreases in that collisions become the main reason of retransmissions. Under this situation, continuing to use fragmentation will make the network more crowded. For example, with NF=-95dB and M=20, the non-fragment scheme may perform better than all the other fragment schemes.

Table 4: Comparison of Throughput Performance of Different Algorithms with NF=-95dB and M=20

Algorithm	Average good throughput (1×10^5 bps) (95% confidence intervals)
Algorithm 1	5.4794 (4.9741, 5.9848)
Algorithm 2	5.2076 (4.6464, 5.7688)
Algorithm 3	5.3714 (4.4442, 6.2985)
Algorithm 4	5.4757 (4.7394, 6.2119)
Algorithm 5	5.2403 (4.5296, 5.9510)
Fixed fragment size	3.4729 (3.2192, 3.7267)
Non-fragment scheme	5.6929 (4.8772, 6.5085)

In general, from above analyses, we can conclude that by using the proposed algorithms, we can achieve a much better throughput performance than what is achieved by the conventional fix fragment scheme. But the proposed algorithms have a larger variance of average good throughput than the fixed fragment algorithm. In addition, using the proposed algorithms to transfer a small file like a web page, the maximum improvement on the average good throughput is 50%.

Recall that TCP congestion control algorithms are optimized to the TCP's coarse timer (500ms) and end-to-end delay, we evaluate the performance of the proposed algorithms by running simulations, where we use the algorithms to transfer a large file (1 MB).

Table 5 shows the comparison of throughput performance with different fixed fragmentation threshold,

Table 5: Comparison of Throughput Performance of Different Algorithms with NF=-85dB and M=5

Algorithm	Average good throughput (1×10^4 bps) (95% confidence intervals)
Algorithm 1	1.6600 (1.5935, 1.7265)
Algorithm 2	1.6576 (1.6068, 1.7084)
Algorithm 3	1.6538 (1.5709, 1.7367)
Algorithm 4	1.6680 (1.6064, 1.7296)
Algorithm 5	1.6758 (1.6137, 1.7380)
Fixed fragment size	0.2534 (0.2382, 0.2686)
Non-fragment scheme	1.0090 (0.9117, 1.1003)

where the noise floor is -85dB and the number of stations is 5. Table 6 is derived for the case where NF=-85dB, M=20; Table 7 is derived under the scenario with NF=-95dB, M=5; Table 8 is derived under the scenario with NF=-95dB, M=20.

Table 5 and 6 show that by using the proposed algorithms, the throughput performance can be greatly improved, especially when the channel quality is bad. In Table 5, the average good throughput can be improved more than seven times by using the proposed algorithms. From this table, we can observe that we can achieve a much better throughput performance than with the fixed fragment scheme. The overall system overhead is greatly increased by using the fixed fragment size scheme due to a large number of retransmissions. We can also conclude that in the given network scenario the adaptive fragment scheme can achieve a better average throughput performance than the non-fragment scheme. We can get similar

Table 6: Comparison of Throughput Performance of Different Algorithms with NF=-85dB and M=20

Algorithm	Average good throughput (1×10^4 bps) (95% confidence intervals)
Algorithm 1	1.5588 (1.4629, 1.6547)
Algorithm 2	1.5344 (1.4360, 1.6329)
Algorithm 3	1.5476 (1.4946, 1.6006)
Algorithm 4	1.5735 (1.5084, 1.6386)
Algorithm 5	1.5717 (1.5277, 1.6157)
Fixed fragment size	0.2297 (0.2202, 0.2391)
Non-fragment scheme	0.9149 (0.8355, 0.9942)

Table 7: Comparison of Throughput Performance of Different Algorithms with NF=-95dB and M=5

Algorithm	Average good throughput (1×10^5 bps) (95% confidence intervals)
Algorithm 1	7.3280 (7.0156, 7.6403)
Algorithm 2	7.4590 (7.2219, 7.6961)
Algorithm 3	7.3748 (7.1472, 7.6024)
Algorithm 4	7.4013 (7.1283, 7.6742)
Algorithm 5	7.3783 (7.1400, 7.6167)
Fixed fragment size	2.2700 (2.0340, 2.5059)
Non-fragment scheme	7.4987 (7.2240, 7.7734)

Table 8: Comparison of Throughput Performance of Different Algorithms with NF=-95dB and M=20

Algorithm	Average good throughput (1×10^5 bps) (95% confidence intervals)
Algorithm 1	5.2583 (4.9172, 5.5994)
Algorithm 2	5.2813 (5.0000, 5.5627)
Algorithm 3	5.3194 (5.1358, 5.5030)
Algorithm 4	5.2653 (5.0680, 5.4626)
Algorithm 5	5.1898 (4.9052, 5.4744)
Fixed fragment size	1.7086 (1.4189, 1.9984)
Non-fragment	5.4049 (5.1392, 5.6706)

conclusions from Table 6, except that the improvements is higher.

Tables 7 and 8 demonstrate that when the channel quality is good, the proposed algorithms can triple the throughput performance of the fixed fragment scheme. These results verify the fact that using the proposed adaptive fragmentation algorithms under a favorable channel quality results in declining improvement of throughput performance by using the proposed algorithm. In fact, when the channel quality is good, fragmentation gives a worse throughput performance than does the non-fragment scheme

due to the increase of overall system overhead.
In general, from above tables, it is shown that the performance of binary exponential algorithm (algorithm 2) is really close to that of the random exponential algorithm (algorithm 1). In fact, the performance of all proposed algorithms are very close. But algorithm 4, which has a slow-start increase and random exponential decrease, can in general achieve a slightly better performance than the others.
We also consider the average throughput performance of the proposed algorithms with different file sizes. Figure 7 is derived under NF=-85dB and M=5. We normalize the average good throughput of the proposed algorithms with the maximum throughput of the fixed fragment sizes. From this figure, we can observe that the performance of the proposed algorithms vary according to different file sizes.
As previously mentioned, when using the fixed frame size scheme to transfer a given file, the total number of frames is determined. If fragmentation is used to transmit each frame, each fragment of the frame will experience a different channel quality. In other words, a larger file has more transmissions than a smaller file. On the other hand, as shown in Figure 2, the channel quality is time-varying. Hence, the larger file has more chances to experience serious channel quality degradations than a smaller file. As a result, its average good throughput is lower than that of a smaller file. For example, in Figure 7, using the proposed algorithms to transfer 1 MB file achieves a lower throughput performance than with a 100 KB file. Furthermore, we can observe that there is a big drop in the average throughput between 900 KB and 1 MB in that the channel quality experiences a severe degradation when transferring 1 MB file. In general, by using the proposed algorithms, the improvement of throughput turns into more and more with the increase of the file size.
Similar results are derived under NF=-95dB and M=5, shown in Figure 8. We normalize the average good throughput of the proposed algorithms with the maximum throughput of the fixed fragment sizes. Note that there is no big drop between 900 KB and 1 MB in this figure. This is because that the noise floor in this simulation is -95dB. In this case, the bad effect of variations of the channel quality is smoothed out by dropping the noise floor.
From the above simulations and analysis, we can conclude that under the same channel condition and

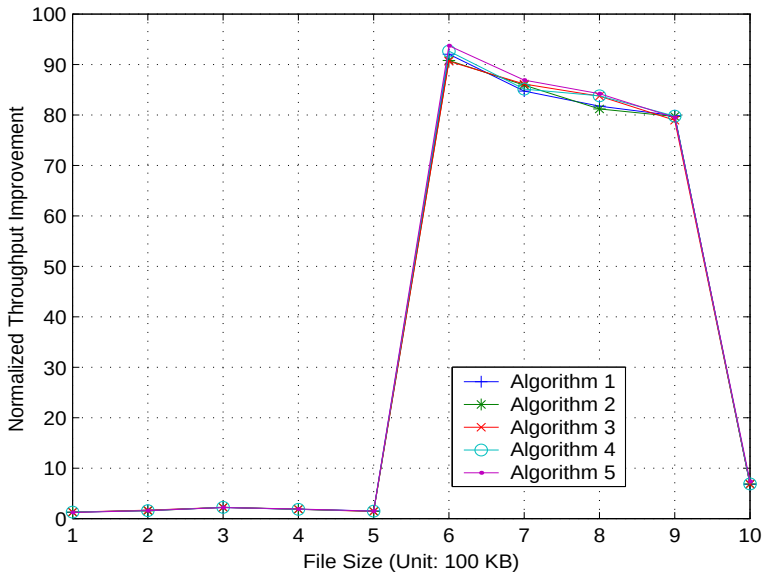

Figure 7: Throughput Performance with Different File Sizes with NF=-85dB (Normalized to the Max. Throughput of Fixed Fragment Scheme)

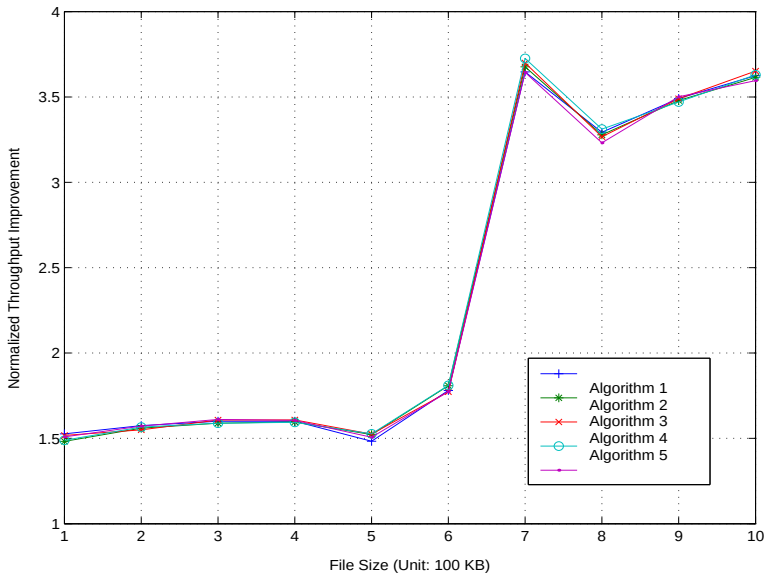

Figure 8: Throughput Performance with Different File Sizes with NF=-95dB (Normalized to the Max. Throughput of Fixed Fragment Scheme)

network load, using the proposed adaptive fragmentation algorithms can improve the good throughput performance significantly, especially under an unfavorable channel quality. This improvement can be attributed to the exploration of the similarities between the TCP protocol and MAC protocol in terms of adaptivity.

However, we should note that the proposed algorithms cannot achieve the best throughput performance under all network scenarios. This is because fragmentation will increase the overall system overhead, even though it can enhance the reliability of transmissions. Moreover, the congestion control algorithms in the TCP protocol are proposed and optimized for the characteristics of the TCP protocol; for example, TCP congestion control algorithms have to accommodate factors such as end-to-end nature, the coarse timer, and the assumption of having a reliable medium. Since TCP congestion control algorithms are optimized for a coarse timer, the proposed adaptive fragmentation algorithms cannot keep track of variations in channel quality in a timely manner. As a result, there is not too much margin on the throughput improvement in some cases, especially with ftp or downloading a small file such as a web page.

5 Conclusion

In this work, we improved the throughput performance of the IEEE 802.11 wireless LAN with link adaption algorithms at the MAC layer. We investigated the effect of adapting frame fragmentation sizes to channel variations. We proposed and studied several adaptive fragmentation algorithms by exploring the analogies between the TCP protocol and the MAC protocol. We have shown by simulating our proposed algorithms greatly improved the good throughput.

References

[1] ANSI/IEEE Std 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. http://www.ieee.org, 1999.

[2] H.S. Chhaya and S. Gupta. Performance Modeling of Asynchoronous Data Transfer Methods of IEEE 802.11 MAC Protocol. *Wireless Networks*, 3:217–234, 1997.

[3] S. Ci and H. Sharif. Adaptive approaches to enhance throughput of ieee 802.11 wireless lan with bursty channel. *The 25th Annual IEEE Conference on Local Area Network (LCN'2000)*, 2001.

[4] S. Ci, H. Sharif, and G. Noubir. Improving Performance of MAC Layer by Using Congestion Control/Avoidance Methods in Wireless Network. *The 16th ACM Symposium on Applied Computing (ACM SAC 2001)*, 2001.

[5] S. Ci, H. Sharif, and A. Young. A Link Adaptation Approach for QoS Enhancement in Wireless Networks. *The 26th Annual IEEE Conference on Local Area Network (LCN'2001)*, 2001.

[6] C. Chien et al. Adaptive Radio for Multimedia Wireless Links. *IEEE Journal on Selected Areas in Communications*, 17(5):793–813, 1999.

[7] P. Lettieri and M.B. Srivastava. Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency. *IEEE INFOCOM '98*, 2:564–71, 1998.

[8] E. Modiano. An Adaptive Algorithm for Optimizing the Packet Size Used in Wireless ARQ Protocols. *Wireless Networks*, 5:279–286, 1999.

[9] L.L. Perterson and B.S. Davie. *Computer Network: A Systems Approach.* Morgan Kaufmann Pulishers Inc., 1996.

[10] N. VanErven, L. Sarsoza, and B. Yarbrough. Fading Channel vs. Frequency Band, Receiver Antenna Height, Antenna Orientation in Office Environment. *Technical Report of Radio System Engineering Group of 3COM Corporation*, 2001.