Adjusting the Retry Limit for Congestion Control in an Overlapping Private BSS Environment

Chang Yun Park
School of Computer Science and Engineering, Chung-Ang University, Seoul, Korea
cypark@cau.ac.kr
*Corresponding author: Chang Yun Park

Received February 11, 2014; revised April 15, 2014; accepted May 13, 2014; published June 27, 2014

Abstract

Since 802.11 wireless LANs are so widely used, it has become common for numerous access points (APs) to overlap in a region, where most of those APs are managed individually without any coordinated control. This pattern of wireless LAN usage is called the private OBSS (Overlapping Basic Service Set) environment in this paper. Due to frame collisions across BSSs, each BSS in the private OBSS environment suffers severe performance degradation. This study approaches the problem from the perspective of congestion control rather than noise or collision resolution. The retry limit, one of the 802.11 attributes, could be used for traffic control in conjunction with TCP. Reducing the retry limit causes early discard of a frame, and it has a similar effect of random early drops at a router, well known in the research area of congestion control. It makes the shared link less crowded with frames, and then the benefit of fewer collisions surpasses the penalty of less strict error recovery. As a result, the network-wide performance improves and so does the performance of each BSS eventually. Reducing the retry limit also has positive effects of merging TCP ACKs and reducing HOL-like blocking time at the AP. Extensive experiments have validated the idea that in the OBSS environment, reducing the retry limit provides better performance, which is contrary to the common wisdom. Since our strategy is basically to sacrifice error recovery for congestion control, it could yield side-effects in an environment where the cost of error recovery is high. Therefore, to be useful in general network and traffic environments, adaptability is required. To prove the feasibility of the adaptive scheme, a simple method to dynamically adjust the value of the retry limit has been proposed. Experiments have shown that this approach could provide comparable performance in unfriendly environments.

Keywords: Retry limit, Congestion control, Overlapping basic service sets, 802.11

This research was partially supported by the Chung-Ang University research grants in 2014.

http://dx.doi.org/10.3837/tiis.2014.06.004

1. Introduction

802.11 wireless LAN is so widely used that almost every home or office has its own Access Point (AP). Furthermore, a personal mobile AP also has been used for a tethering service to provide Internet access to personal devices. It is not unusual for dozens of APs to be detected at a campus, a mall or even a home, and those numbers will surely increase in the future. In our research, we call this pattern of wireless LANs the private OBSS (Overlapping Basic Service Set) environment. (More precise naming is overlapping private BSSs, but simply private OBSS is used because OBSS is already a knowledgeable term.) Here, private is the key word, which emphasizes that most BSSs are governed not by an administrator but by a user. The private AP is associated with only a few nodes or dedicated to a single node in many cases, and its range is usually very short. It mostly works in the DCF mode on a default channel without considering conflicts with other APs. Coordination from a global view of all BSSs is hard to achieve. In terms of network usage, most of the nodes in the private OBSS environment are used for instant information access from various servers and frequently connected to cloud computing services. In other words, traffics in the private OBSS environment consist mostly of download traffics from the AP to the node, and their round-trip times are usually very short.

Collisions across BSSs could be solved by channel allocation. However, this study does not consider that approach for the following two reasons. First, in a private network environment, an optimal channel allocation through global coordination is hard to expect. It is a reasonable assumption that each BSS achieves limited improvement using only local knowledge. Second, since there are a limited number of orthogonal channels (for example, three in the 802.11 2.4 GHz band), the demand of a channel always exceeds the supply. In short, channel sharing among tens of BSSs is inevitable.

The next issue, which this study focuses on, is how to efficiently share a link in the channel with other BSSs. The existing 802.11 standard mostly covers medium access in a BSS and interactions across BSSs are not considered in much detail. In overlapping wireless LANs new patterns of collisions across BSSs may occur and cause performance degradation. For example, an ACK at the receiver, normally considered collision-free, may collide with frames in other BSSs. Consequently, the performance at a BSS is degraded well below its proportional share of the link.

In fact, the performance characteristics of the private OBSS environment are somewhat different from those of a highly noisy error-prone link. For example, most 802.11 systems apply a data rate adaptation scheme, which decreases the data rate as the link quality gets worse. It has been an established rule for keeping the frame error rate stable. However, decreasing the data rate increases the transmission time of a frame, and hence the chance of collision for the frame may increase in uncoordinated accesses. In the private OBSS environment where only a few nodes are associated in a short range, the adaptation rule does not always improve performance [1]. The RTS/CTS mechanism has also been found not to be helpful in the OBSS environment because the overhead is high including collisions caused by RTS/CTS but the effect of collision avoidance across BSSs is limited without some extensions [2]. The retry limit, which is normally set high in an error-prone link, is also an interesting factor that shows performance anomalies in private OBSS environments.

Most studies on wireless LANs consider frame collision a MAC problem. There have been some efforts to enhance the RTS/CTS mechanism to reduce collisions in the OBSS environment [2-3]. However, in the private OBSS environment, inherently crowded networks of various personal devices with a lack of coordinated control, their effectiveness would be limited. In reality, frame collision is treated as a kind of link noise, and solutions for overcoming a poor link condition are simply applied. This causes performance anomalies because collision is a traffic factor rather than a physical factor. In other words, collisions in the OBSS environment should be addressed also by traffic control such as congestion control. There are few parameters for traffic control in wireless LANs, but it is believed that the retry limit could be used for congestion control at the link-level.

Conventional wisdom is that the retry limit should be set high in error-prone environments because more attempts result in more successful transmission. However, in the OBSS environment where a link is usually crowded with the frames from numerous BSSs, this might not be true. Sometimes, giving up retransmission attempt early with a low retry limit value may reduce congestion on the link. Like the policy of random early drop in the congestion control area, it can be beneficial not only to all BSSs but eventually also to the corresponding BSS.

The goal of this study is to introduce and validate the idea that in the OBSS environment reducing the link parameter of the retry limit does congestion control and improves performance. To achieve this goal, first, the effects of reducing the retry limit are figured out in detail. We have found that reducing the retry limit gives results of merging TCP ACKs and reducing HOL-like blocking time as well as invoking congestion control. Next, performance implications are checked through extensive simulation experiments. Since it is certain that reducing the retry limit does harm for error recovery, the retry limit should be used as a control knob for the trade-off between congestion control and error control. Finally, feasibility is addressed by introducing a simple adaptive scheme where the retry limit is adjusted to the link and traffic conditions.

We believe that our approach is practical in the sense that it requires neither any modification of the current medium access control (i.e., DCF), nor any cooperative management among private BSSs. The solution is novel in that the key for the solution is congestion control and it is implemented by adjusting the retry limit, which is always considered a parameter for error recovery. It is based on the observation that excessive error recovery might be ineffective in the OBSS environment.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 explains the effects of reducing the retry limit in private OBSS environments. Extensive performance evaluation has been made to validate the approach in section 4. Section 5 introduces an adaptive scheme to adjust the retry limit depending on network conditions, which could give comparable performance in the environments unlike a private OBSS. Finally, section 6 concludes the study.

2. Related Work

Most of the existing studies on the OBSS have focused on channel allocation or selection problems. Some of them have proposed a distributed channel selection method where each AP by itself decides which channel to use without coordination with other APs [4-5]. As mentioned above, considering current usage patterns of 802.11 where demands for a channel always exceed available channels, they could alleviate crowdedness but cannot solve the problem of congestion in essence. An adaptive power control for mitigating interference, for example [6], would have similar effects and limitations. Those studies could be used together with our study in an orthogonal way. Recently, there have been some studies on overlapping BSSs other than channel selection [2,7], but most of them require modification of the current access control and/or coordinated management. They also focus on not DCF access but QoS services.

The issues of overlapping BSSs have been one of the concerns of newly developing 802.11 standards, such as 802.11ac [8]. For example, [9] proposed an enhanced RTS/CTS scheme to provide protection in non-primary channels shared with overlapping non-802.11ac BSSs. However, practical effectiveness of RTS/CTS has been questionable so they are usually not used. The new schemes would protect better against collision, but it would be hard to expect immediate performance improvement in the private OBSS environment where various legacy devices with uncoordinated management coexist. Collisions on RTS might still cause congestion problems.

It is somewhat surprising that adaptive retry limits have not been much studied. It might be the reason that most agree that as the retry limit is set higher, the throughput increases though the delay may get longer. In fact, most recent studies focus on adapting the retry limit for real-time applications which are delay-sensitive [10-11].

Congestion control is usually considered not a functionality of the link layer but of the transport layer (TCP) or router. There are only a few studies on congestion control in 802.11 networks, and almost all of them have addressed how to support QoS services in congested 802.11 networks. For example, [12] proposed a method of throttling channel access among overlapping BSSs by differentiating channel access parameters such as the Arbitration Interframe Space Number (AIFN). To our best knowledge, there is no study on controlling congestion itself in 802.11 DCF networks.

We believe that our solution is novel in that it addresses the problem purely from the issue of congestion. The other difference is that utilizing the existing link parameter, it requires no modification of the current medium access control (i.e. DCF) or any cooperative management among BSSs.

3. The Effects of Reducing the Retry Limit

In order to recover transmission failures, 802.11 provides a retransmission mechanism where transmissions are attempted until the network attribute of the retry limit is reached. In most wireless LANs, the retry limit is set to 7, which means transmissions are attempted up to 6 times unless the transmission is successful. (Assuming the RTS threshold is not used, long retry limit and short retry limit are not distinguished in this study.) The conventional wisdom is that the retry limit should be set high in an error-prone environment because more attempts result in more successful transmission. However, in the OBSS environment where a link is usually crowded with the frames from numerous BSSs, this might not be true. Reducing the retry limit and ending retransmission attempts early may cause different network behaviors and performance in the OBSS environment from those in conventional non-OBSS environments.

3.1 Early Congestion Control

In the OBSS environment, transmission of a frame may fail with a high probability mostly due to collision with other frames. More crowded frames increase the failure rate. For each BSS, a retransmission attempt seems beneficial since it increases the overall chance of successful transmission. However, for BSSs sharing a link, it is questionable since it may aggravate the crowdedness of the link, especially if another collision occurs. Although the exponential back-off scheme is applied to resolve crowdedness, a retransmission after back-off may not improve the success rate greatly because some degree of crowdedness always exists in the OBSS. Increased crowdedness increases the collision probability, and it decreases the transmission success rate. As a result, a retransmission attempt may have a negative effect on the transmission of not only the frames from other BSSs but also the succeeding frames from the BSS itself. In the OBSS environment, persistent retransmission attempts up to the fixed retry limit for error recovery at each BSS, which is applied in an ordinary 802.11 LAN, increase congestion, causing time and bandwidth to be wasted in vain. This is a typical congestion situation.

To control this congestion, excessive retransmission attempts should be prohibited. A more fundamental solution is to avoid crowdedness in a link. A simple solution of reducing the retry limit can address both of them. A lower retry limit makes a frame discarded earlier, and thus excessive retransmission occurs less frequently. Dropping a frame invokes an action of TCP congestion control, and as a result, the amount of traffic at the link will be reduced. With a lower retry limit, earlier traffic reduction is possible and crowdedness could be alleviated earlier.

This approach to link crowdedness is very similar to RED (Random Early Detection) [13] at a router. The two cases are different in that meaningless resource consumption is caused by packet drop at the router in RED but by frame collision at the link in our study. Both are the same in that congestion is alleviated by dropping a packet or a frame early. They are different again in that congestion is detected by monitoring the queue size at the router in RED but by monitoring the failure of retransmission over a certain limit in our study.

Reducing the retry limit is certainly harmful for error recovery at the link layer. Since it is well known that end-to-end error recovery is more expensive than link-level recovery, performance suffers from frequent end-to-end recovery actions. However, the lower retry limit causes frames to be dropped earlier, and then invokes the data rate to decrease by TCP congestion control earlier. The less crowded link alleviates congestion and eventually reduces the link error rate due to fewer collisions. That is, a lower retry limit makes error recovery more expensive but less frequent. In other words, the retry limit could work as a control knob for the trade-off between congestion control and error control.

In order to validate the idea, a simple experiment has been made. The experiment scenario is that two mobile nodes associated with different APs in the laboratory download an infinite-size file from the different servers at the wired network. The wireless environment is a typical OBSS where about 40 BSSs are actively detected. The result is shown in **Fig. 1**. As the retry limit gets lower, the aggregate throughput of the link, the sum of IPerf [14] throughputs at the two nodes, increases. The result seems to somewhat contradict the conventional wisdom that increasing the retry limit is better in a hostile wireless link. It could support our idea that reducing the retry limit for early congestion control may improve performance, in spite of increasing the recovery cost per error. More experiments with various network parameters including the round-trip time will be made in Chapter 4.

3.2 TCP ACK Merging

It is well known that most traffic in an 802.11 LAN is download traffic, and this is also true in the OBSS environment. The upload traffic from the mobile node to the AP usually contains only the response from TCP, without any application data. In this study it is called a TCP ACK frame. The size of a TCP ACK frame is variable but is usually around 116 bytes, depending on the size of the TCP and 802.11 optional headers.

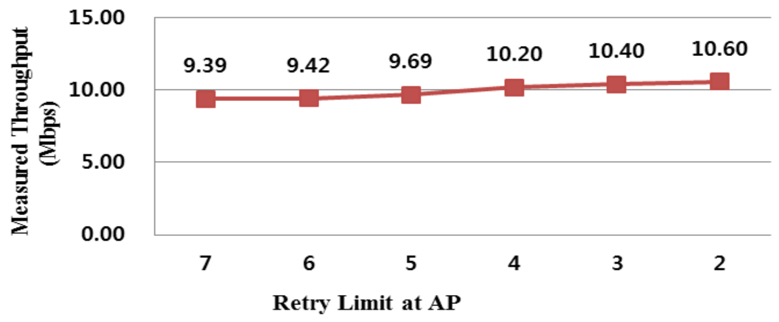
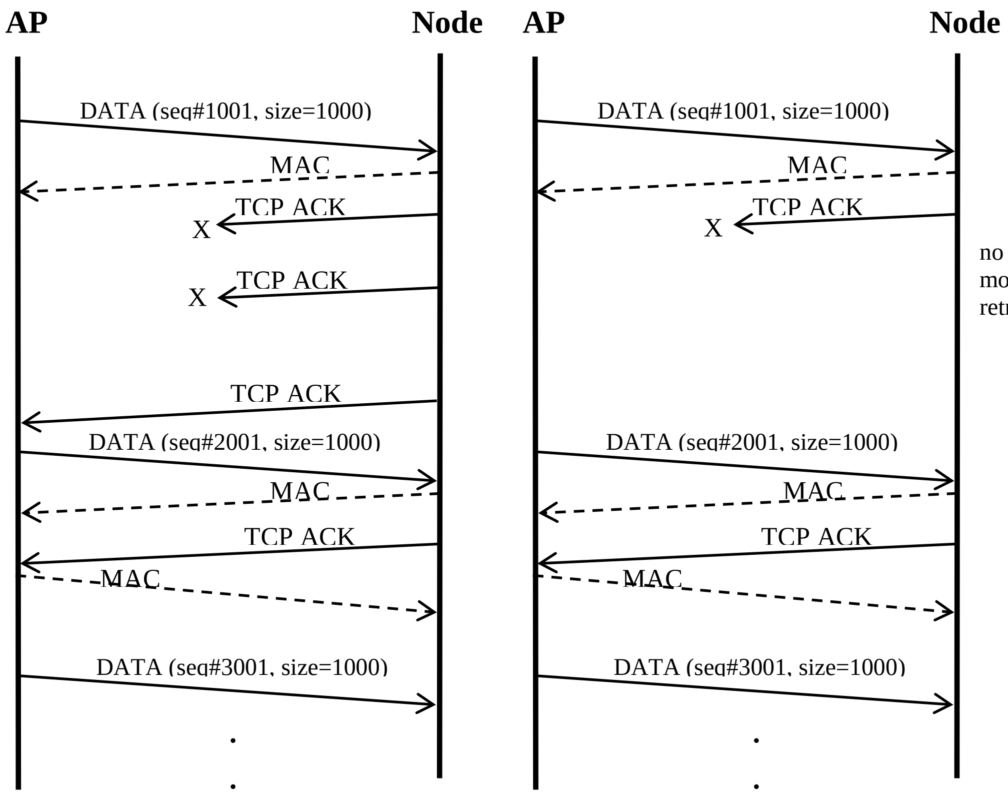


Fig. 1. Effect of Retry Limit on Performance (Measured Aggregate throughput) in A Simple OBSS Environment

Although a small frame has a better chance of avoiding collisions than a long frame, it is certain that reducing the retry limit decreases the delivery success rate of a TCP ACK frame. In other words, it has the same risk of early drop as a TCP ACK frame. However, the effect of dropping a TCP ACK frame is somewhat different from that of a normal frame containing user data. The semantics of the TCP acknowledgement scheme are cumulative in default, which means that "ACK *n*" confirms the successful reception of up to the *(n-1)*-th byte of data. Due to these semantics, loss of a TCP ACK *i* can be concealed by the delivery of a follow-up TCP ACK *i*+*m* for some positive number *m*. After the latter is delivered, the former is redundant, and thus recovery by retransmitting it is unnecessary. Since a TCP ACK frame only has a meaning of TCP acknowledgement, a drop of a TCP ACK frame could be also concealed by the successful transmission of the follow-up TCP ACK frame. In other words, the discarded TCP ACK frame is merged into the following TCP ACK frame. Owing to this merging effect, the penalty for early discard of the TCP ACK frame from the perspective of error control could be smaller than that of the data frame.

Fig. 2 shows two frame transmission scenarios where the left is a default 802.11 case with a retry limit of 7 and the right is a case with a retry limit of 2. Both are typical cases which can be found in a real traffic trace captured using AirPcap [15]. In the default case, the TCP ACK frame (TCP ACK #2001) failed twice and succeeded the third time after exponential back-offs and retransmission. In the case when the retry limit is 2, the TCP ACK frame is simply discarded with no more retransmission attempt because the retry limit is already reached. However, the progress at TCP could be the same as in the default case. TCP at the sender can transmit multiple data segments within its effective window size, and thus the reception of TCP ACK #2001 is not necessary for transmission of DATA (seq#2001). Upon sending and receiving TCP ACK #3001 which implicitly merges #2001 into its cumulative acknowledge number, #3001, the existence of TCP ACK #2001 could be ignored. That is, early discards cause TCP ACK frames to be merged into one frame, which is called TCP ACK merging in this study.

TCP ACK merging has two performance implications. First, it directly reduces link traffic. The wireless link of the 802.11 LAN is basically half-duplex, and a TCP ACK frame is a data frame in 802.11. Thus, there always exists a chance that a TCP ACK frame from a node collides with a data frame from the AP. Reducing the number of attempts to transmit TCP ACK frames is certainly a plus factor for network performance. TCP does not send an ACK on every receipt of a data segment. It is controlled in a complicated way by several rules of error, flow and congestion control (for example, [16]). In the OBSS environment, a successful reception of a TCP segment may require several transmissions of 802.11 data frames, depending on the crowdedness. Therefore, the proportion of the TCP ACK frame in the link traffic would be less than half, in terms of the number of frames. From a sample trace with a default retry limit, it is found that about 30% of data frames are TCP ACK frames. Although its frame length is shorter than that of a TCP data frame, the overhead for medium access is the same. Therefore, TCP ACK merging may save considerable amount of bandwidth.


AP
Node
AP
Node
DATA (seq#1001, size=1000)
MAC
TCP ACK
X
TCP ACK
X
TCP ACK
DATA (seq#2001, size=1000)
MAC
TCP ACK
MAC
DATA (seq#3001, size=1000)
no more retry

(a) when retry limit = 7 (b) when retry limit = 2

Fig. 2. A Scenario of TCP ACK Merging

Second, however, TCP ACK merging also has a negative effect on performance, because it basically delays reporting the receiver status. It may delay the recovery of an error and also cost the opportunity to send more data faster. In an uncongested network, an immediate TCP ACK is preferred because the TCP sending rate can be increased with the pace of ACK reception. However, in the OBSS environment, which is basically a congested network, this immediateness might not always be helpful because more traffic means more collisions. Moreover, those delay effects would be tentative only until the following TCP ACK is delivered. Therefore, as long as there exists sufficient TCP data traffic, this minus factor does not overshadow the aforementioned benefit of the early drop.

Finally, it is worthwhile to note that TCP ACK merging occurs situationally in proportion to the crowdedness of the link. In a light traffic OBSS, a TCP ACK frame is delivered to the AP with a high probability and TCP congestion control could increase the traffic rate immediately. In a heavy traffic OBSS, TCP ACK merging happens with a high probability and increasing the traffic rate could be delayed. In conclusion, the delivery time of TCP ACKs, that is, the responsiveness at TCP, could be adjusted depending on the crowdedness of the physical link.

3.3 Reduction of Head-Of-Line-like Blocking at the Access Point

Suppose there are two nodes associated in a BSS, and both nodes are actively receiving download traffic. A frame should wait in the queue at the AP until transmission of the precedent frame is finished by successfully receiving an ACK frame or reaching the retry limit. In the OBSS environment where retransmission is very frequent, this waiting might be very long. If the precedent frame is destined for the same node as the destination of the waiting frame, waiting is acceptable because logical ordering matters. However, if the destinations are different, the inevitability of waiting is questionable because it is possible to try transmitting the frame waiting in the queue to the other destination. In particular, back-off time between retransmissions of the precedent frame, which usually increases exponentially, might be useful for this alternative transmission. From the perspective of the waiting frame, its transmission is blocked not by its own logical or physical constraints but by the lack of progress of the precedent frame.

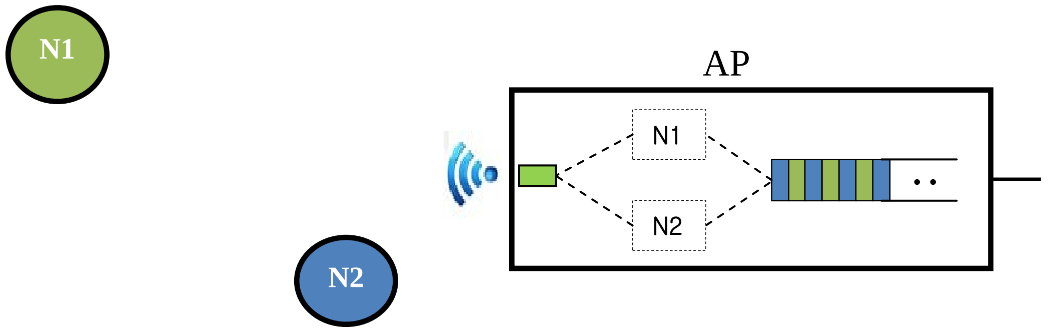


Fig. 3. A Scenario of Had-Of-Line-like Blocking

Since the situation is similar to Head-of-Line (HOL) blocking in network switching [17], it is called HOL-like blocking at the AP in this study. Fig. 3 shows a situation where HOL-like blocking occurs. Two nodes, N1 and N2 are associated with the AP and the traffic destined to them are buffered in the AP. Inside the AP, in order to remind the case of HOL-blocking, a logical switch is illustrated though the outgoing links are combined into the antenna. The frame at the head of the queue should wait until transmission of the outstanding frame is finished, although it is destined for a different node, N2. To our best knowledge, this kind of blocking at the AP has not been addressed directly in any 802.11 studies. It is probably because the outgoing link of the AP is physically unique and thus waiting is a matter of course. However, it is not nonsense that the AP has multiple logical links, logically concurrent transmission of multiple frames is possible, and then HOL-like blocking may happen.

Setting aside the validity of HOL-like blocking for a while, first observe how reducing the retry limit affects transmission of the frame waiting in the queue. Since the precedent frame is discarded earlier than in the default case, the frame waits less than the default. On transmission, the frame may have a higher probability of collision than the default case because its value for the congestion window is initialized without accounting for the collision situation while transmitting the precedent frame. In summary, reducing the retry limit may reduce blocking time in the queue but may increase frame collisions due to not performing exponential back-offs.

The issue is then whether this effect is beneficial or harmful in the OBSS environment. At first glance, it seems to be harmful because the OBSS environment is already crowded and hence sending a frame in the queue without back-offs is unlikely to succeed. It is also somewhat contradictory to our approach of improving performance by reducing crowdedness. However, it should be considered that congestion control is being performed below the surface by TCP. This is the difference from the case of linear back-offs or no back-offs. Assuming some of the APs do not have waiting frames in their queue, the collision probability of the frame sent after the early discarded frame might not be that high.

For example, suppose there are two APs, A and B. Each AP sends a frame, the frames collide, and they both discard the frames with a low retry limit. If both APs have a frame in the queue, the frames sent next at each AP will collide with a higher probability than the default case where the precedent frames are retransmitted with exponential back-offs. However, if one of the APs does have a frame in the queue, the frame in the queue at the other AP may be sent with no higher collision probability than that of retransmitting frames in the default case. In this case, reduced waiting time can be a benefit. Here it is simply assumed that other APs in the OBSS environments have the same effects in both cases. Much more analytical work would be needed to determine the detailed effects of reducing waiting time. In this study its practical feasibility will be considered in the next section through simulation experiments.

When considering the validity of the notion of HOL-like blocking, consider the session during which a frame is transmitted. In 802.11 it is basically atomic; it should not be preempted by transmission of other frames. Reducing the retry limit may cause early discard, which implies an abrupt stop of the transmission session. However, the data in the discarded frame is retransmitted by TCP, and thus the frame can be considered to be eventually transmitted again. In other words, a transmission session of a frame is interrupted, other frames may be transmitted, and the session is resumed. Assuming, for the extreme contrast, retransmission in the default case is always successful eventually, the difference is whether interleaving transmissions of frames happens or not. In summary, reducing the retry limit allows interleaving transmissions of frames, and as a result those out-of-order transmissions make it possible to overcome blocking caused by the order in the queue. This is similar to the logic for reducing HOL-blocking.

There would be an argument that interleaving transmissions of multiple frames on the same physical link has no significant meaning; it only reduces back-off times from an exponential increase to either a linear increase or no increase. However, it is not always true that two nodes in a BSS have the same collision probability at reception although they use the same physical link. Also, the practical effect of exponential back-offs is questionable in the OBSS environment. Therefore, it is believed that the case is different from existing studies on back-off time. Finally, it is worthwhile to emphasize that this study does not advocate reducing the retry limit to alleviate HOL-like blocking. It simply explains an observation that performing congestion control at the link-level by reducing the retry limit may result in reduction of HOL-like blocking at the AP.

4. Experimental Results

4.1 Experiment Methods

Since it is practically infeasible to build a controllable OBSS environment, performance evaluation has been done by simulation. OPNET [18] is selected as a simulation tool because it gives more realistic results to real measurements than ns [19]. Ns results are consistent with OPNET results, but their values are much smaller than the corresponding measurement values.

The basic experimental OBSS environment is as follows. Each BSS consists of one AP and one wireless node working in 54 Mbps 802.11g, and there are such 40 BSSs in the environment (in short notation, 40x1AP-1node-BSS is used later). The number of BSSs, 40 is selected because about 40 BSSs are detected in our laboratory. Setting the number of BSSs is important because it could decide the level of congestion. It will be addressed as the degree of overlap in the experiment later. The average range of BSS is around 40 meters, which is reasonable considering obstacles in the real world. Each AP is connected with a dedicated server through the wired network, for which latency is adjustable. With respect to traffic, each wireless node downloads files from the dedicated server. In summary, the environment consists of 40 disjoint networks of (mobile node, AP, server).

Before making performance analysis in detail, the validity of simulation results has been tested. That is, for some simplified OBSS environments, the simulation results have been compared to the real measurements. **Table 1** shows the result of one of the test cases, where two (node, AP, server) networks are fully active but the other 38 BSSs are inactive. The measurement has been made in the two BSSs in the laboratory in the middle of the night, which means that other BSSs are mostly inactive in terms of wireless traffic. In OPNET the scenario with 40 BSSs is simulated where each node in two BSSs runs the file transfer application and all other nodes run with no application.

Table 1. Comparison of Simulation Result and Measurement in a Simple OBSS Environment

Environment	Simulated Throughput	Measured Throughput
2 active BSSs and 38 inactive BSSs of 802.11g (54Mbps)	(9.23, 9.02)	(8.12, 7.81)Mbps

As mentioned in the related work (Chapter 2), it is believed that there has been no study on congestion control in DCF wireless LANs. Coordinated management methods such as bandwidth allocations are not applicable to private LANs. Other possible techniques for OBSS such as an adaptive data rate would be orthogonal to this study, and thus there would be no direct comparison. Therefore, performance evaluation is made through comparison not to other schemes but to the default retry limit.

4.2 Effects of Adjusting Retry Limit on Performance

First, the aggregate throughput of all the mobile nodes in the OBSS is analyzed with varying retry limits. The precise definition of the aggregate throughput is the sum of the goodput of the file download application at each mobile node, which indicates the real amount the OBSS system can provide to users effectively. Those redundant deliveries caused by either of link-level or transport-level retransmission should be excluded from the metric. The throughput is represented in the unit of bps (bit-per-second), which is more commonly used for link-level performance.

To check its effect at the data sender (AP) and the data receiver (mobile node) separately, experiments have been made in two cases where the retry limit of each side is adjusted disjointly. One case is where the retry limit at the AP is varied from 2 to 7 while that of the mobile node is fixed with the default value, 7. The other is the case of adjusting the retry limit of the mobile node while that of the AP is fixed with the default, 7. In the case when the retry limit is 1, retransmission never occurs by definition. Some network drivers do not allow it in reality. Hence the minimum value of the retry limit is set to 2. **Fig. 4** shows the simulation results.

In both cases, the aggregate throughput goes consistently higher as the retry limit goes lower. Reducing retry attempts at the data sender (i.e., reducing the retry limit of the AP) gives earlier effects than doing it at the receiver. Reducing the value of the retry limit consistently gives throughput improvement except the saturation from 3 to 2 at the AP. The reason for this saturation is because in case of the retry limit =3, giving up one more retry attempt decreases the success rate of frame delivery sufficiently to erase the benefit of early congestion control. For the retry limit ≥ 3, one more retry attempt does not significantly increase the success rate of frame deliver, and thus, skipping a retry attempt gives more gain than the cost of frame recovery.

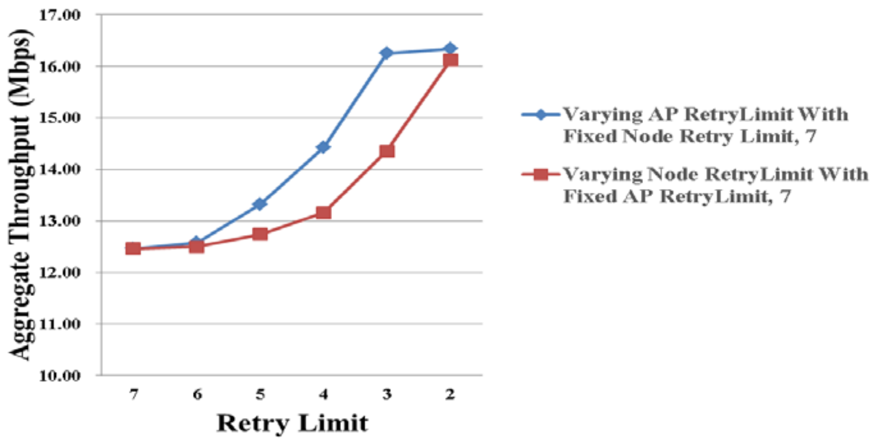


Fig. 4. Effect of Adjusting the Retry Limit on Link Throughput

Reducing the retry limit only at the receiver (i.e., the node) ultimately gives almost the same performance gain. It could be explained that TCP ACK is also a data frame at the link-level, though its length is short, competing the half-duplex wireless channel with the data sender. In addition, due to the effect of TCP ACK merging, discarding a retry attempt of a TCP ACK frame gives nothing but the benefit of congestion control except delay in the response time, as long as a subsequent data arrives. There is no saturation even from 3 to 2. These results show that in a congested network environment like OBSS, it is an acceptable idea to sacrifice error control for congestion control.

In order to evaluate the combined effects of reducing the retry limit at both sides, a total of 36 cases have been tested varying the limit from 2 to 7, independently. Only the summaries of the results are explained here. The best case with respect to throughput is 3/2, where a case *x*/*y* means when the retry limit at the AP is *x* and that at the node is *y*. It gives 17.06 Mbps on average, which wins over 2/2 and 2/3 with very little margin. It could be interpreted that reducing the retry limit at the receiver is effective only when sufficient data is delivered. From now on, case 3/2 is used as the case representing when the link congestion control is applied.

It is worthwhile to note that the important point is not the value of the best case, 3/2, but the fact that adjusting the retry limit for congestion control consistently gives better performance. 3/2 is the best case only in the given OBSS environment where traffic is homogeneously local between the nodes and the servers in LANs (precisely, RTT is 10 msec). In a different network environment, a different combination would be the best case. However, the default case of 7/7 will not be the best case in the OBSS environments. The problem of determining the value of the retry limit in a different traffic environment will be addressed later.

Next, the other important performance metric, the response time, is investigated. Specifically, when transferring small fixed-size files iteratively, their completion times are monitored in the same OBSS environment as above. We compare the average completion times for the default case and the link congestion control case while varying the file size. The results are shown in **Fig. 5**. For microscopic monitoring in a critical region, the file size is incremented by 1 Kbyte in the region from 40 to 45, while the interval is 10 K byte in the other region.

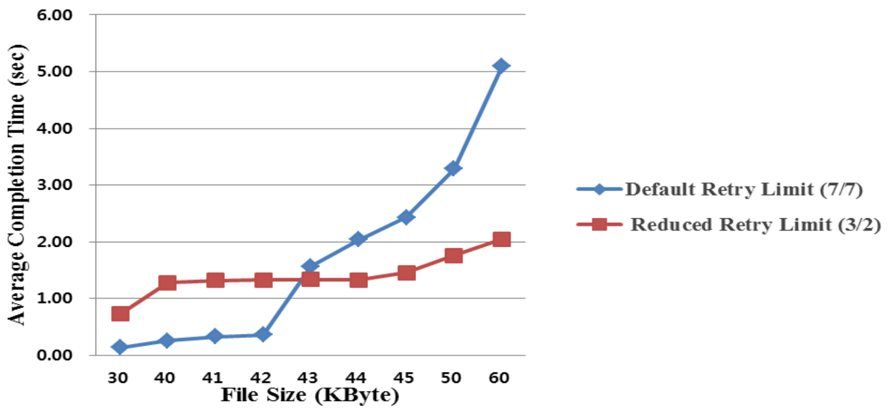


Fig. 5. Effect of Reducing the Retry Limit on Response Time (File Transfer Time)

The default case shows the typical behavior of a random access scheme, where the response time is very short in light traffic but it increases dramatically beyond some critical point, 43 Kbyte in this case. Below 43K bytes, 7/7 recovers most of frame failures by the link-level retransmission and those retransmitted frames do not make the link congested yet. Thus, the file transmission is done much faster than 3/2, which depends on retransmission at TCP. However, over 43K bytes, those retransmission frames of 7/7 make the link start to be congested. More collisions require more retransmissions, and then they yield severer congestion. It is a typical vicious cycle of congestion. Eventually, backlogging of the periodic traffic also gets started, and the response time is dramatically increased.

On the other hand, 3/2 shows behavior similar to a controlled access scheme. The response time in light traffic is relatively high due to the control overhead, that is, expensive end-to-end error recovery in this case. However, it increases almost linearly with the file size regardless of the heaviness of traffic. Therefore, over 43K bytes, it gives shorter response time than 7/7. From the perspective of evaluating a congestion control scheme, the ratio of throughput (i.e., file size) to delay (i.e., file transfer time) is almost constant. It means that our method of link congestion control by simply reducing the retry limit can sustain a performance level even with congestion.

The degree of overlap, as a measure of crowdedness, has also been investigated. Varying the number of BSSs in the environment, the aggregate throughputs of the default retry limit case and the reduced retry limit case are compared in **Fig. 6**. In the default retry limit case, the aggregate throughput is decreased as the number of BSSs is increased because more collisions cause more loss of time and bandwidth. In the reduced retry limit case, the slope of the decrease is less severe; after some point the aggregate throughput keeps a level and or rather rises sometimes (# of BSSs = 40). It can be explained as follows. The individual throughout of a BSS decreases as the number of BSSs increases. However, more BSSs mean more traffic sources, and more traffic sources often result in less idle time and waste of link bandwidth. The aggregate sum of the individual throughput at each traffic source may vary differently. In the default retry limit case, the losses due to more collisions are greater than the gains due to more traffic sources, but this might not always be true in the reduced retry limit case because collisions are controlled due to the early congestion control.

Another interesting result is when the number of BSSs is 2, which is a typical non-OBSS environment. At first glance this example is anomalous because collisions would not occur frequently and the cost of recovering them at the link should be less expensive than that of recovering them at TCP. However, early congestion control has an effect even when there are only two traffic sources, which obviously compete for the link in an aggressive manner. The short RTT in the experiment environment also enables relatively fast error recovery by TCP, which is positive for the reduced retry limit case. In summary, reducing the retry limit always gives the same or better performance than the default retry limit case regardless of the degree of overlap, at least in the private OBSS environment assumed in this study. The performances in different environments will be addressed in section 5.

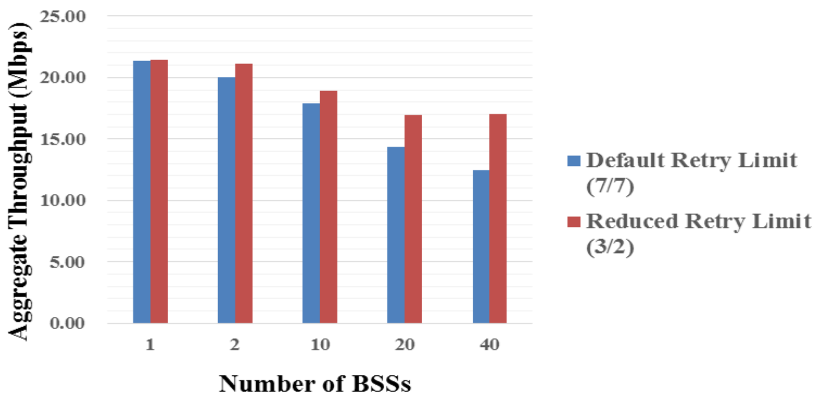


Fig. 6. Effect of the Degree of Overlap on Reducing the Retry Limit

In order to check the effect on HOL-like blocking, an experiment has been designed as follows. The basic experimental environment has been changed by adding a mobile node to each of arbitrarily chosen 10 BSSs, which is called an 1AP-2node BSS in contrast to the normal 1AP-1node BSS. Those 30 BSSs with a single node keep the link crowded and in the 10 BSSs, two different nodes share the AP for download traffic. Applying the default 7/7 retry limit and the 3/2 retry limit for link congestion control, respectively, the aggregate throughputs of each group are compared as shown in **Table 2**.

Table 2. Aggregate throughput (Average Throughput Per Node) in the OBSS Environment with 1AP-2Node BSSs for Examining the Effect on HOL-like Blocking

Retry Limit	**30 $\times$1AP-1Node BSS**	**10 $\times$1AP-2Node BSS**	**Total**
Default (7/7)	7.84 Mbps (0.26 Mbps/node)	3.01 Mbps (0.15 Mbps/node)	11.24 Mbps
Reduced (3/2)	7.62 Mbps (0.25 Mbps/node)	6.66 Mbps (0.33 Mbps/node)	15.21 Mbps

The result is somewhat surprising. Comparing the total throughputs of the link for each case, the reduced retry limit case gives better performance than the default case as expected from the effect on congestion control. However, almost all of the performance gain goes to 1AP-2node BSSs; the nodes in the 1AP-1node BSS do not have performance improvement in the previous experiments. This seems to be because those APs processing two download traffic streams aggressively utilize the time slots and bandwidth that become available due to early drops. From the synergy effect with TCP congestion control, the higher rate traffic causes higher biased access for the slots.

Looking at the result of the default case, those traffic-wise aggressive APs have only slightly higher throughput than the APs in the 1AP-1node BSS. This may be interpreted as follows. In the default case of full retransmission attempts with exponential back-offs, the average completion time of frame transmission (i.e., the service time at the AP) is not much shorter than the average inter-arrival time of each download stream. Thus, the number of attempts to access the link at the AP in the 1AP-2node BSS cannot be much higher than at the AP in the 1AP-1node BSS, although the traffic arrives from two sides and hence at twice the rate. Extra traffic had to wait in the queue at the AP.

However, in the case of a 3/2 retry limit, those early drops not only make bandwidth available but also reduce the average completion time of frame transmission. Suppose completion time is shortened by up to the half of the average traffic inter-arrival time. Then all traffic from both streams at the AP in the 1AP-2node BSS can have transmission attempted. Assuming the success ratio of frame transmission is always the same, the AP in the 1AP-2node BSS can achieve double the throughput. Combined with TCP congestion control, the throughput difference becomes more than doubles in practice. Leaving out the appropriateness of the name of HOL-like blocking, it is certain that reducing the retry limit shortens average completion time of frame transmission, and then decreases waiting time in the queue. As a result, there are more chances to attempt to access the link, and its effect on performance could be positive as in the experiment.

Again, it is worthwhile to emphasize that this study does not argue that a 1AP-2node BSS gives better performance than two 1AP-1node BSSs in the OBSS environment. In fact, it is believed that the 1AP-1node BSS is more common in the OBSS environment than the 1AP-2node BSS. This implies that reducing HOL-like blocking is one of the effects of reducing the retry limit. It is decided that validation through extensive numerical analysis is desirable but outside of the scope of this paper.

Finally, some experiments have been made in mixed OBSS environments where BSSs with the default retry limit and BSSs with the reduced retry limit coexist. Specifically, two cases have been simulated; one is the case where all 38 BSSs are working with the default 7/7 retry limit but 2 BSSs have a reduced 3/2 retry limit, and the other is the reverse, where all 38 BSSs are working with a 3/2 retry limit but 2 BSSs have the default 7/7 retry limit. All BSSs are 1AP-1Nodes.

Table 3 shows the results. While 38 nodes perform retransmission attempts as usual, the efforts of congestion control at only two nodes become meaningless to the reduction of crowdedness and result in performance degradation. In the reverse case, there exists a significant extra bandwidth saved by early drop, but it is monopolized by the two nodes with the default retry limit; other nodes with a reduced retry limit have slight throughput improvement. In summary, the effectiveness of link-level congestion by reducing the retry limit requires consensus across the OBSS environment.

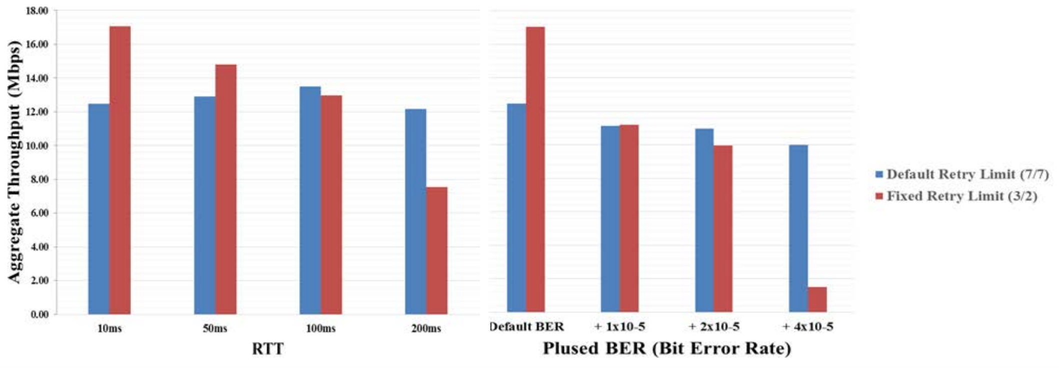
Table 3. Aggregate throughput (and Average Throughput per Node) in the OBSS Environment Mixed with the Default and Reduced Retry Limit

Case	Nodes with Default Retry Limit	Nodes with Reduced Retry Limit	Total
38 Default (7/7) + 2 Reduced (3/2)	12.13 Mbps (0.32 Mbps/node)	0.30 Mbps (0.15 Mbps/node)	12.46 Mbps
38 Reduced (3/2) + 2 Default (7/7)	3.33 Mbps (1.66 Mbps/node)	13.69 Mbps (0.36 Mbps/node)	17.02 Mbps

5. A Simple Adaptive Scheme

5.1 Experiments in Different Traffic/Network Environments

The performance results shown in Section IV are based on the assumption that there is no extra bit error in the wireless link other than attenuation by distance and all traffic is downstream from the server, which is located a short distance away. Although this environment is mostly valid for current usage of a private 802.11 BSS, there could be different traffic or wireless link conditions. Therefore, the effect of reducing the retry limit should be investigated in different traffic/network environments.



(a) RTT vs Throughput **(b) Error rate vs Throughput**

Fig. 7. Performance of Reduced Retry Limit in Various Environments

First, the case of long distance traffic has been tested. Since reducing the retry limit is basically trading off the cost of error recovery for the cost of congestion, the amount of time for recovering an error directly affects its effectiveness. The simulation result with longer RTTs (round-trip time) is shown in **Fig. 7-(a)**. As expected, the performance of the link congestion control case decreases as RTT increases.

Next, the case of lossy links has been tested with the same purpose as above. Since setting the bit error rate of the link to an absolute value does not tell us how lossy the link is, we vary the link error condition by adding some constant to the BER of the default condition where the experiment in the previous section is made. Increasing those constants, the aggregate throughputs are as shown as in **Fig. 7-(b)**. With the same reasoning as above, the bit error rate of the wireless link directly affects performance. Over some point of BER, the case of "$+4\times10^{-5}$" in the graph, the throughput of reducing the retry limit to 2/3 drops severely. It would be interpreted that our assumption on the OBSS environment that a frame error is usually due to collision rather than bit error is not true in that case.

Considering the current usage pattern of a wireless LAN, it is believed that long distance traffic and a heavily lossy link condition are rare. However, in order to be used in a general wireless network environment, the retry limit should be adjusted depending on the network and traffic conditions.

4.2 A Simple Adaptive Adjustment of the Retry Limit

Before describing our adaptive scheme in detail, the scope of the work should be described explicitly. The main theme of this paper is to introduce the idea that adjusting the value of the retry limit can be a key for congestion control in the OBSS environment. This is a notion that is relatively new and thus not validated yet. Therefore, on adaptability, it is believed that its feasibility should be addressed first. An optimal adaptive scheme, which might require a lot of analytical work, could be left to future studies after the idea is validated. In other words, our study of adaptability focuses on a simple proto-type solution rather than a complicated optimal solution.

It is clear that an adjustment method should consider BER and RTT because our scheme improves performance by sacrificing the efficiency of error recovery, which is a major factor in error recovery cost. Most wireless drivers or network adaptors estimate various error rates to utilize them in adapting network parameters such as data rate. Therefore, the approximation of BER can be determined. However, in the case of RTT, there is no easy way to figure it out. It is basically infeasible with link-level information to determine how long the data inside a frame has been traveled from the source. There might be some cross-layering techniques such as looking deep inside a frame for the source IP address. However, it does not estimate RTT directly and hence requires another complicated estimation method. Since it is against our principle of a simple solution, estimating RTT is not attempted in this study. Instead, frequency and severity of error are considered.

The following four variables are newly added in the OPNET wireless LAN library source program for adaptive adjustment of the retry limit. They have been easily computed from the existing code, and thus they could be calculable in real systems.

- `frame_size`: the length of a frame to send in bytes
- `prob_cs_succ`: the probability that the carrier sense attempt is successful
- `avg_trans_count`: average number of transmissions for finishing the process of sending a frame
- `ack_since_discard`: the number of ACK frames received between discarding retransmission attempts

`frame_size` is used to detect whether a frame is TCP ACK. `prob_cs_succ` is introduced to estimate the crowdedness of the link. It is recalculated whenever carrier sense is made. `avg_trans_count` is monitored in order to access the frame error rate of the link. This indicates the error-proneness of the link although it cannot distinguish if an error comes from collision or bit corruption. It is recalculated both when an ACK is received and when a frame is discarded.

Finally, `ack_since_discard` is introduced to include the importance of error recovery in a decision. If `ack_since_discard` is *n*, this means that *n* frames are transmitted successfully since a frame is discarded. When *n* is large, discarding a frame without retrying transmission would be tolerable as in the case of random drop at a router. However, when *n* is very small, discarding a frame might cause a majority of frames to be lost and result in a pause in data exchange. Thus, recovering the frame would be more important than when *n* is large.

Conceptually, it is similar to the “count” variable in RED [13]. `ack_since_discard` is updated when an ACK is received and reset to zero when a frame is discarded.

One subtlety in the scheme is when the value of the retry limit is adjusted. One is when a frame is inserted into the transmission queue, and the other is whenever a transmission attempt including retransmission is made. Although the former is common in implementation, in our experiments using the OPNET simulator, adjustment is made on every transmission attempt.

Using the above variables, a simple rule-based adjustment of the retry limit has been made as shown in **Table 4**. The actual code is added in the OPNET wireless LAN library (specifically, `wlan_frame_discard()`). Rule 1) is to utilize the effect of merging TCP_ACKs. A more precise condition would be equal to sizeof(TCP_ACK) but it is variable depending on the existence of the TCP optional header. The simpler condition is applied in the experiment. Rule 2) sets the base value of the retry limit to the link environment. The lower `prob_cs_succ` is, the higher the crowdedness of the link is. Thus, if `prob_cs_succ` is lower than a certain threshold (C_{OBSS}), the link is considered to be the OBSS and the retry limit is set to L_{OBSS}, which is 3 as in the experiments of the previous section. If `prob_cs_succ` is higher than $C_{NON\text{-}OBSS}$, congestion control at the link is not necessary. The retry limit has a default value of 7. When `prob_cs_succ` is in-between, the retry limit is set to L_{BASE}, which is varied between L_{OBSS} and $L_{DEFAULT}$ depending on the other conditions. Those threshold values, C_{OBSS} for 0.3 and $C_{NON\text{-}OBSS}$ for 0.4, are decided from measurements through experiments; they do not have an analytical basis. However, it is believed that they are acceptable to use for checking the feasibility of adaptation.

Table 4. Adaptation Rules for Adjusting the Retry Limit

	Condition	Adjustment
1)	`frame_size` $\leq$ sizeof(TCP_ACK)	retrylimit = L_{OBSS} [a]
2)	`prob_cs_succ` $<$ C_{OBSS}	retrylimit = L_{OBSS}
	`prob_cs_succ` $>$ $C_{NON\text{-}OBSS}$	retrylimit = $L_{NON\text{-}OBSS}$
	otherwise	retrylimit = L_{BASE}
3)	`avg_trans_count` $>$ A_{ERROR}	
	if `prob_cs_succ` $<$ C_{OBSS}	retrylimit `--`
	if `prob_cs_succ` $>$ $C_{NON\text{-}OBSS}$	increase L_{BASE} [b]
4)	`ack_since_discard` $\geq$ S_{HIGH}	retrylimit `--;` decrease L_{BASE}
	`ack_since_discard` $\leq$ S_{LOW}	retrylimit `++;` increase L_{BASE}

[a] L_{OBSS} : the constant value of the retry limit for the case of OBSS, 3;
$L_{NON\text{-}OBSS}$: the constant value of the retry limit for the case of NON-OBSS, 7;
L_{BASE}: the value of the retry limit for other cases, which is adjusted dynamically between L_{OBSS} and $L_{NON\text{-}OBSS}$, initially, $L_{NON\text{-}OBSS}$;
C_{OBSS}: the threshold of crowdedness for the case of OBSS, 0.3;
$C_{NON\text{-}OBSS}$: the threshold of crowdedness for the case of NON-OBSS, 0.4;
A_{ERROR}: the threshold of error-prone link condition in average number of transmission per frame, 2.5;
S_{HIGH}: the high-end threshold of successively successful transmissions, 6;
S_{LOW}: the low-end threshold of successively successful transmissions, 2.
[b] L_{BASE} is increased or decreased by 1 within [L_{OBSS} .. $L_{NON\text{-}OBSS}$]

Rule 3) is added to account for a lossy link. If `avg_trans_count` is higher than A_{ERROR}, the link is interpreted to be error-prone. If `prob_cs_succ` is lower than C_{OBSS}, the error comes mostly from collisions due to crowdedness and thus the retry limit is decreased for the earlier drop. When `prob_cs_succ` is higher than $C_{NON\text{-}OBSS}$, the error comes not from collisions but from bit error. Since link lossiness is a sustaining property, L_{BASE}, instead of the retry limit, is increased, which continuously affects transmission of the following frames. Finally, if `ack_since_discard` is greater than S_{HIGH}, it means failure is tolerable. Thus, opportunistically, the retry limit is decreased. When `ack_since_discard` is less than S_{LOW}, another transmission failure may cause bursty errors, which then may result in a severe drop in the data rate at TCP. The retry limit is increased. L_{BASE} is also increased or decreased in each case to maintain adaptation longer and to make the link more stable. Again those threshold values for A_{ERROR}, S_{HIGH} and S_{LOW} are decided by tuning the experiments.

As shown in **Fig. 8**, in the default OBSS environment, the adaptive scheme cuts the performance improvement from 37% to 18%, in comparison to when the fixed retry limit is reduced to 3/2. However, the adaptive scheme gives consistent performance in other environments; they are similar to the case of the default retry limit. Rule 4) in Table IV increases L_{BASE} when there are not enough successful frame deliveries between discarding a frame by reaching the retry limit. As a result, the retry limit in the adaptive scheme is usually higher than that of 3/2 fixed, 3. In the high error environments, the rate of successful frame delivery is low and L_{BASE} becomes close to $L_{NON\text{-}OBSS}$, which is the same as the default retry limit, 7. Therefore, they show similar retransmission behavior and performance.

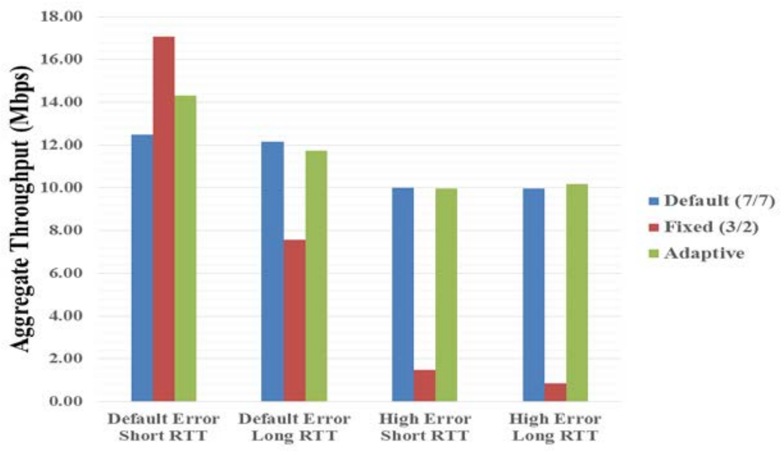


Fig. 8. Performance of a Simple Adaptive Scheme in Various Environments

This somewhat conservative behavior of the adaptive scheme is intended in fact, to argue against the belief that reducing the retry limit is disastrous in non-OBSS environments and hence useless in general. It has been shown that reducing the retry limit does nothing but improvement with some simple adaptation rules. Changing those threshold values, the adaptive scheme may work in a high-gain/high-risk OBSS-oriented mode, which is believed to be more beneficial in the current usage of wireless LANs. Some deliberate adaptation methods would work effectively in both environments, which will be left for further work in this study.

6. Conclusion

Performance degradation in the private OBSS environment has been addressed from the perspective of congestion control. By reducing the retry limit, which is known as an error control parameter, early random drops of packets occur in the wireless link and then TCP congestion control is activated. As a result, the link becomes less crowded and frame collisions could be reduced. Reducing the retry limit also gives positive effects of TCP ACK merging and a reduction of HOL-like blocking time at an AP. Extensive experiments have shown that it can consistently improve performance in environments where the OBSS consists of numerous small private BSSs and the traffic is mostly downloads to the mobile nodes from the server located at a short distance. In an environment where an end-to-end retransmission is much more expensive than a link-level retransmission, by reducing the retry limit, the cost of error recovery may overwhelm the benefit of congestion control. Therefore, a simple method of adjusting the retry limit to the network and traffic conditions has been proposed and tested. It would validate the adaptability of the idea.

Although the idea has been analyzed in behavior analysis and validated in experiments, more comprehensive numerical analysis is required to completely verify its validity. An optimal adaptation scheme based on numerical analysis is a subject for future work. The study considers TCP only because using UDP the quality of service provided, that is, reliability is different depending on the value of the retry limit. If an application is less sensitive to error, the results of the study including the adaptive scheme could be applied to it with UDP.

References

[1] Joo, Yohan, et al. "Performance Anomaly of the Overlapping BSS in the IEEE 802.11," In *Proc. of Proceedings (D) of 39th KIISE Fall Conference*, pp.179-182, 2012. Article (CrossRef Link)

[2] Fang, Yue, et al. "A two-level carrier sensing mechanism for overlapping BSS problem in WLAN," In *Proc. of Local and Metropolitan Area Networks, 2005. LANMAN 2005. The 14th IEEE Workshop on*. IEEE, 2005. Article (CrossRef Link)

[3] Chang, Zheng, et al. "Performance Analysis of IEEE 802.11 ac DCF with Hidden Nodes," In *Proc. of Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*. IEEE, 2012. Article (CrossRef Link)

[4] Leith, D. J., and P. Clifford. "A self-managed distributed channel selection algorithm for WLANs," In *Proc. of 2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, IEEE, pp.1-9, 2006. Article (CrossRef Link)

[5] Ihmig, Matthias, and Peter Steenkiste. "Distributed dynamic channel selection in chaotic wireless networks," In *Proc. of 13th European Wireless Conference*, Paris, France. 2007. Article (CrossRef Link)

[6] Oteri, Oghenekome, et al. "Advanced power control techniques for interference mitigation in dense 802.11 networks," In *Proc. of 2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, IEEE, pp.1-7, 2013. Article (CrossRef Link)

[7] Madan, Ritesh, et al., "Enhancing 802.11 Carrier Sense for High Throughput and QoS in Dense User Settings," In *Proc. of IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 254-259, 2012. Article (CrossRef Link)

[8] Ong, Eng Hwee, et al. "IEEE 802.11 ac: Enhancements for very high throughput WLANs," In *Proc. of 2011 IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, IEEE, 2011. Article (CrossRef Link)

[9] Gong, Michelle X., et al. "Channel Bounding and MAC Protection Mechanisms for 802.11 ac.," In *Proc. of Global Telecommunications Conference (GLOBECOM)*, pp. 1-5, IEEE, 2011. Article (CrossRef Link)

[10] Shadmand, Amir, and Mohammad Shikh-Bahaei, "TCP dynamics and adaptive MAC retry-limit aware link-layer adaptation over IEEE 802.11 WLAN," In *Proc. of CNSR'09. Seventh Annual Communication Networks and Services Research Conference*, IEEE, 2009. Article (CrossRef Link)
[11] Chen, Chih-Ming, et al., "Cross-layer packet retry limit adaptation for video transport over wireless LANs," *IEEE Transactions on Circuits and Systems for Video Technology*, *vol. 20 no.11*, pp. 1448-1461, 2010. Article (CrossRef Link)
[12] Han, Bo, et al., "Channel access throttling for overlapping BSS management," In *Proc. of ICC'09. IEEE International Conference on Communications*, IEEE, 2009. Article (CrossRef Link)
[13] Floyd , S. and Jacobson , V., "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking, vol. 1 no. 4*, pp.397-413, August 1993. Article (CrossRef Link)
[14] Tirumala, Ajay, et al., "Iperf: The TCP/UDP bandwidth measurement tool," http://dast.nlanr.net/Projects. 2005. Article (CrossRef Link)
[15] AirPcap, http://www.airpcap.nl/airpcap.htm. Article (CrossRef Link)
[16] Allman, Mark, et al., "RFC 2581: TCP congestion control," 1999. Article (CrossRef Link)
[17] Karo, M. et al., "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Transactions on Communications vol. 35, no. 12*, pp. 1347–1356, 1987. Article (CrossRef Link)
[18] "Documentation, OPNET Modeler", OPNET Technologies Inc., http://www.opnet.com , 2003. Article (CrossRef Link)
[19] "Network Simulator ns-2," http://www.isi.edu/nsnam/ns/. Article (CrossRef Link)

Chang Y. Park received the B.S. and M.S. degrees in computer engineering from Seoul National University, Seoul, Korea in 1984 and 1986, respectively, and received the Ph.D. in computer science from the University of Washington, Seattle in 1992. He is currently a Professor in the School of Computer Science and Engineering, ChungAng University, Seoul, Korea. His research interests include computer networks, and real-time systems.