Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?

Shugong Xu and Tarek Saadawi, City University of New York

ABSTRACT

The IEEE 802.11 MAC protocol is the standard for wireless LANs; it is widely used in testbeds and simulations for wireless multihop ad hoc networks. However, this protocol was not designed for multihop networks. Although it can support some ad hoc network architecture, it is not intended to support the wireless mobile ad hoc network, in which multihop connectivity is one of the most prominent features. In this article we focus on the following question: Can the IEEE 802.11 MAC protocol function well in multihop networks? By presenting several serious problems encountered in an IEEE 802.11-based multihop network and revealing the in-depth cause of these problems, we conclude that the current version of this wireless LAN protocol does not function well in multihop ad hoc networks. We thus doubt whether the WaveLAN-based system is workable as a mobile ad hoc testbed.

INTRODUCTION

Wireless ad hoc networks are required where a fixed communication infrastructure, wired or wireless, does not exist or has been destroyed. In a multihop ad hoc network, nodes communicate with each other using multihop wireless links, and there is no stationary infrastructure such as a base station. Each node in the network also acts as a router, forwarding data packets for other nodes. A central challenge in the design of ad hoc networks is the development of dynamic routing protocols that can efficiently find routes between two communication nodes. A *mobile ad hoc networking* (MANET) working group has been formed within the Internet Engineering Task Force (IETF) to develop a routing framework for IP-based protocols in ad hoc networks [1].

However, in this article we discuss another challenge in wireless multihop ad hoc networks: medium access control (MAC). Since media is a shared and scarce resource in a wireless network, efficiently controlling access to this shared media becomes a complicated task. A great deal of effort has been made in this field, and many MAC layer protocols have been proposed. However, few of them were designed to be used in multihop wireless links, and few of them have been evaluated in multihop networks. In this article we focus on the IEEE 802.11 MAC layer protocol and examine the problems caused by multihop wireless links. Since IEEE 802.11, "Distributed Foundation Wireless Media Access Control (DFWMAC)," is the standard for wireless ad hoc and infrastructure LANs [2] and is widely used in almost all of the testbeds and simulations for wireless ad hoc network research, an important and natural question is whether the IEEE 802.11 MAC protocol works well in multihop ad hoc networks.

This question arises when we evaluate the performance of TCP in IEEE 802.11-based wireless ad hoc networks. TCP is the prevalent transport layer protocol used in the IP world today. It provides reliable data transfer and congestion control. As a transport layer protocol, it runs above the network and MAC layers. Thus, MAC layer protocols of ad hoc networks should support TCP. If the MAC layer protocol (in our case, DFWMAC) cannot support TCP very well, it is probably not advisable to use this protocol in this kind of network, even if it works well in typical wireless LANs. As we will show in the following parts of this article, TCP traffic intensifies the problem in this MAC layer protocol when it is used in IEEE 802.11-based multihop ad hoc networks.

In this article we present the problems in the IEEE 802.11 MAC protocol, which are encountered and exacerbated when this protocol works with TCP in a wireless ad hoc network. By analyzing the multilayer traces from the simulation, we reveal the in-depth causes of these problems and offer possible solutions. Before doing this, however, we will give an overview of IEEE 802.11.

0163-6804/01 $10.00 © 2001 IEEE

An Overview of the IEEE 802.11 Standard [2]

Like any 802.x protocol, the 802.11 protocol covers the MAC and physical layers. The standard currently defines a single MAC which interacts with three PHYs (all of them running at 1 and 2 Mb/s) as follows: frequency hopping spread spectrum in the 2.4 GHz band, direct sequence spread spectrum in the 2.4 GHz band, and infrared.

The MAC layer defines two different access methods, the distributed coordination function (DCF) and point coordination function (PCF). We now describe the DCF in detail (since the PCF cannot be used in ad hoc networks, it is not described here).

The basic access mechanism, the DCF, is basically a carrier sense multiple access with collision avoidance (CSMA/CA) mechanism. CSMA protocols are well known in the industry, the most popular being the Ethernet, which is a CSMA with collision detection (CSMA/CD) protocol. A CSMA protocol works as follows. A station desiring to transmit senses the medium. If the medium is busy (i.e., some other station is transmitting), the station defers its transmission to a later time. If the medium is sensed as free, the station is allowed to transmit. These kinds of protocols are very effective when the medium is not heavily loaded, since it allows stations to transmit with minimum delay. But there is always a chance of stations simultaneously sensing the medium as free and transmitting at the same time, causing a collision. These collision situations must be identified so the packets can be retransmitted by the MAC layer, rather than by the upper layers. The latter case will cause significant delay. In order to overcome the collision problem, the 802.11 uses a CA mechanism coupled with a positive acknowledge scheme, as follows:

- A station wanting to transmit senses the medium. If the medium is busy, it defers. If the medium is free for a specified time, called the distributed interframe space (DIFS) in the standard, the station is allowed to transmit.
- The receiving station checks the cyclic redundancy check (CRC) of the received packet and sends an acknowledgment packet. To distinguish this MAC layer ACK from upper layer acknowledgments, we designate it M-ACK. Receipt of the M-ACK indicates to the transmitter that no collision occurred. If the sender does not receive the M-ACK, it retransmits the frame until it receives an M-ACK or throws it away after a given number of retransmissions. According to the standard, a maximum of seven retransmissions are allowed before the frame is dropped.

In order to reduce the probability of two stations colliding due to not hearing each other, the well-known "hidden node problem," the standard defines a virtual CS mechanism: a station wanting to transmit a packet first transmits a short control packet called request to send (RTS), which includes the source, destination, and duration of the intended packet and ACK transaction. The destination station responds (if the medium is free) with a response control Packet called clear to send (CTS), which includes the same duration information.

All other stations receiving either the RTS and/or the CTS set their virtual CS indicator, called a network allocation vector (NAV), for the given duration and use this information together with the physical CS when sensing the medium. The physical layer carrier sensing function is called clear channel assessment (CCA). The NAV state is combined with CCA to indicate the busy state of the medium. This mechanism reduces the probability of the receiver area collision caused by a station that is "hidden" from the transmitter during RTS transmission, because the station overhears the CTS and "reserves" the medium as busy until the end of the transaction. The duration information on the RTS also protects the transmitter area from collisions during the M-ACK (from stations that are out of range of the acknowledging station). It should also be noted that, due to the fact that the RTS and CTS are short frames, the mechanism also reduces the overhead of collisions, since these short transmissions allow faster recognition of collisions than would be possible for the transmission of an entire packet.

As we know, besides the hidden node problem, wireless packet networks also face the exposed node problem. A hidden node is one that is within the interfering range of the intended destination but out of the sensing range of the sender. Hidden nodes can cause collisions on data transmission. Exposed nodes are complementary to hidden nodes. An exposed node is one that is within the sensing range of the sender but out of the interfering range of the destination. If exposed nodes are not minimized, the available bandwidth is underutilized. However, in the 802.11 MAC layer protocol, there is almost no scheme to deal with this problem. This might cause a serious problem when it is used in multihop wireless networks. We will discuss this in more detail in the next sections.

The Simulation Environment and Methodology

Before proceeding further we need to introduce the simulation environment and methodology. The results reported in this article are based on simulations using the NS2 network simulator from Lawrence Berkeley National Laboratory (LBNL) [3], with extensions from the MONARCH project at Carnegie Mellon [4]. The extensions include a set of mobile ad hoc network routing protocols and an implementation of BSD's ARP protocol, as well as an 802.11 MAC layer and two radio propagation models. For more information about this software, we refer the reader to [3, 4].

The link layer of the simulator implements the complete IEEE 802.11 standard MAC protocol DCF in order to accurately model the contention of nodes for the wireless medium. All nodes communicate with identical, half duplex, wireless radios that are modeled after

IEEE 802.11 is widely used in almost all of the testbeds and simulations for wireless ad hoc network research.

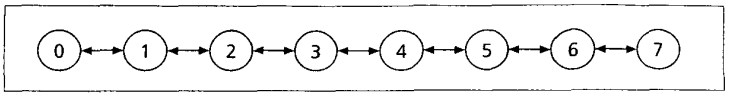

Figure 1. *A string topology.*

the commercially available 802.11-based WaveLan wireless radios which have a bandwidth of 2 Mb/s and a nominal transmission radius of 250 m.

With a few exceptions, we chose to keep most of the parameters of the simulations used in [4]. The following is the description of our simulation setup. Each node has a queue (called IFQ) for packets awaiting transmission by the network interface that holds up to 50 packets and is managed in a drop tail fashion. DSR routing protocol was used.

We consider one type of network topology: a string topology with eight nodes (0 through 7) as shown in Fig. 1. It is a good example for multihop connectivity. Only a portion of the nodes in this network are involved in each experiment. The distance between any two neighboring nodes is equal to 200 m, which allows a node to connect only to its neighboring nodes. In other words, only those nodes between which a line exists can directly communicate. The same distances between neighboring nodes ensure that the nodes act equally in the simulation. Nodes are static. We do not address the link failure problem, which is caused by mobility. Our target network is a wireless multihop network, which is the basis of wireless mobile ad hoc networks.

In this article we use TCP traffic to show the problems existing in the MAC layer. We assume that these TCP connections carry large file transfers (i.e., infinite backlog; the TCP sender always has data to send out). Now we offer more explanation of the reason we use TCP in this article.

WHY TCP?

First, the Transmission Control Protocol (TCP) is the prevalent reliable transport protocol used in the Internet today. To make this network function well, TCP must be supported.

Second, TCP has another important advantage: it can adapt to the network condition and do congestion control. Therefore, it can use almost all the available bandwidth without causing congestion. We use this feature to examine the MAC layer protocol. As mentioned above, TCP traffic enlarges the problems of the MAC layer protocol.

Following, we present a brief introduction to TCP. It is a window-based ACK-clocked flow control protocol. (Note that here ACK means a TCP layer acknowledgment from a TCP destination.) It uses an additive-increase/multiplicative-decrease strategy for changing its windows according to network conditions. Starting with one packet (or a larger value in some TCP versions), the window is increased exponentially by one packet for every nonduplicate ACK until the resource estimate of network capacity is reached. This is the slow start (SS) phase, and the capacity estimate is called the *slow start threshold*. Once this threshold is reached, the source (sender) switches to a slower rate of increase in the window by one packet for every window's worth of ACKs. This phase, called *congestion avoidance* (CA), aims to slowly probe the network for any extra bandwidth. Window increase will stop when it reaches the maximum TCP window size, which is defined when the connection starts. Otherwise, window increase is interrupted when a loss is detected. Either the expiration of a retransmission timer or the receipt of three duplicate ACKs (fast retransmit) could result in such a loss. There is little difference among different TCP versions in the processing method for a loss. The source supposes that the network is in congestion and sets its estimate of the capacity to half the current window. TCP Reno, which is now the most popular version, has a fast recovery algorithm to retransmit losses. We will use this TCP variance as an example. For more information about TCP, please refer to [5] and the references therein.

In this article we present two problems existing in IEEE 802.11-based multihop wireless ad hoc networks. They are the TCP instability problem and the unfairness problem. From the forthcoming description, we will demonstrate that these problems are rooted in the MAC layer. The IEEE 802.11 MAC protocol will be shown to function poorly when it is used in a multihop environment. After a description of each problem, we illustrate its underlying cause by showing the multiple layer traces.

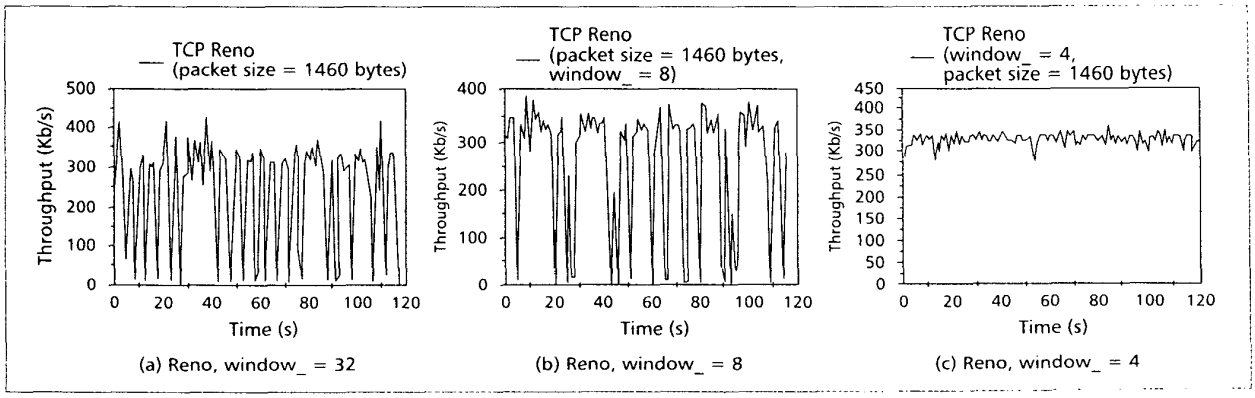

Figure 2. *The instability problem in the four-hop TCP connection (from node 1 to node 5). TCP packet size = 1460 bytes.*

THE TCP INSTABILITY PROBLEM AND ANALYSIS

In the first set of experiments, we set up a single TCP connection between a chosen pair of sender and receiver nodes, and measured the successively received packets over the lifetime of the connection. The network topology is shown in Fig 1. The TCP session is the only traffic in that network; no background traffic exists. Hence, there are no network condition changes in the whole lifetime of each experiment. As mentioned earlier, TCP can adaptively adjust its transmission rate according to network condition. If the network condition does not vary, the TCP throughput should stay stable within some range. More specifically, the Reno TCP version, which has a fast recovery algorithm, should achieve more stable throughput than the former version, Tahoe. So in each of our experiments we expect steady throughput in the connection lifetime. However, this does not seem to be the case. In the following part of this section we use a four-hop TCP connection as an example. The source node is 1, destination node 5. The TCP packet size is 1460 bytes. Note that the current version of NS2 can only support fixed-size TCP packets in each simulation. This does not hurt the universality of our conclusions.

Figure 2 shows the related results. It includes three small figures, each illustrating the measured throughput variations during the lifetime of one simulation run. The plotted values of the throughput are measured over 1.0 s intervals. We count the successively received TCP packets in each 1.0 s interval and transfer it into the throughput in that interval. Let us take a look at Fig. 4a. In the 120 s lifetime of this connection, there are 20 times when the throughput reached or neared zero. In those 1.0 s intervals, almost no TCP packets were successively received, which means that TCP performance degraded seriously. Every time after this, TCP restarted using slow start. Since only one connection exists in the experiment, this kind of pause is not expected. This oscillation can only be explained by this TCP version not working well in the IEEE-802.11 based wireless multihop network. We call this *instability* of TCP in this specific kind of network. The TCP parameter, known as maximum window size (*window_*), has an effect on this problem. As explained above, it is the limit of the real transmission window size in a TCP connection. In fact, the TCP instability problem can be lessened or eliminated with a smaller maximum window size. In Fig. 2a this parameter is set as 32. Figure 2b demonstrates serious oscillation with a *window_* of 8 and a packet size of 1460. The results are better than those associated with *window_* = 32, but the oscillation is still very serious. In the 120 s lifetime of this TCP connection, the throughput reached or neared zero 16 times! Figure 2c shows the case with *window_* = 4. No serious instability problem occurs at this level. After describing these phenomena, we offer our analysis of the problem.

By analyzing the traces, we find this problem is always due to one node failing to reach its adjacent node. This triggers a route failure. If it is an intermediated node, this node drops all queued packets (ACK in most cases) to that adjacent node and reports a route failure to the source. Here *source* means data packet source: either the TCP sender or receiver. After the source receives this message, it starts route discovery. Before a route is found, no data packet can be sent out. Usually, this causes a timeout in the TCP sender. Then the TCP session has to wait before a route becomes available again.

Now we will look at the cause of route failure, focusing on the case shown in Fig. 2b. We look into the simulation traces of this run. This TCP connection is from node 1 to 5. As we can see in the figure, the throughput falls to zero at around 4.0 s and 20.0 s. We focus on the packet trace of the latter case. By analyzing the simulation trace, we find it is rooted in the MAC layer. Node 1 cannot reach node 2. After node 1 tries to contact node 2 and fails seven times, the MAC layer reports a link breakage. Note that a limit of seven retries is defined in IEEE 802.11. A part of the MAC layer packet trace is shown in Fig. 3. In this figure, *data* means TCP packet

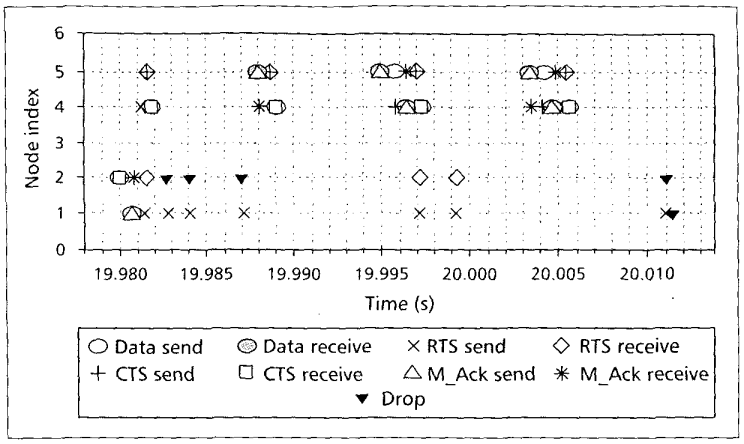

Figure 3. *Part of the MAC layer packet trace for the session in Fig. 2b; Reno, window_ = 8, packet size = 1460 bytes.*

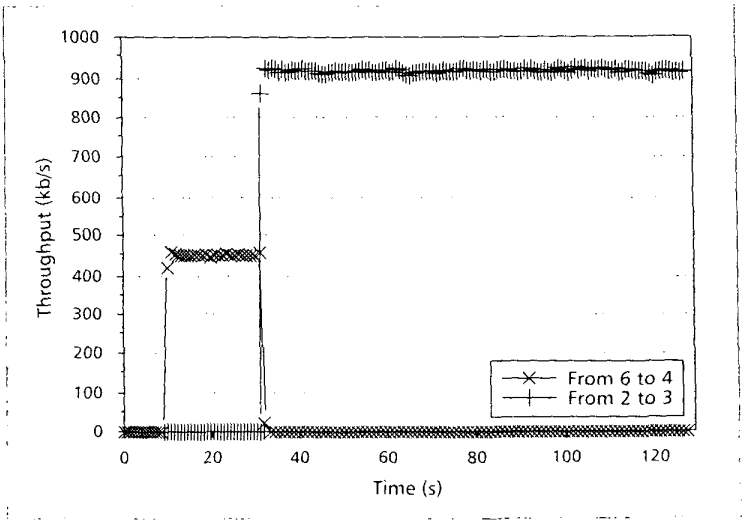

Figure 4. *Throughput of two TCP connections with different sender and receiver, window_=4.*

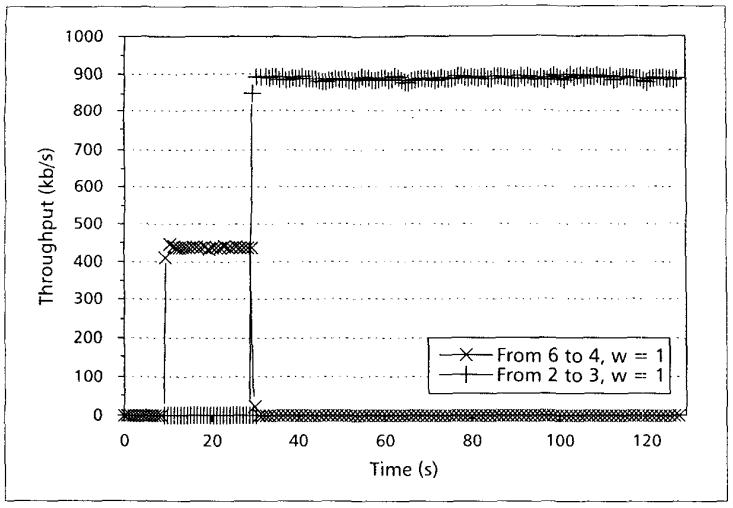

Figure 5. *The throughput of two TCP connections with different sender and receiver, window_ = 1.*

or TCP ACK packet. In reference to the MAC layer, they are all data from an upper layer. (Refer to [3, 4] for more details about implementation of the IEEE 802.11 MAC layer in NS2 software.) From this figure, we find that the collision and the exposed station problem in node 2 prevent node 1 from reaching node 2. Since node 2 can sense node 4, it has to defer when node 4 is sending. The result is that node 2 cannot send back CTS even if it receives the RTS from node 1 correctly. After failing to receive CTS from node 2 seven times, node 1 quits and reports a link breakage to its upper layer. Then a route failure event occurs.

It must be stated that, in a CS wireless network, the interfering range (and sensing range) is typically larger than the range at which receivers are willing to accept a packet from the same transmitter [6]. WaveLAN wireless systems are engineered in this way. This is the reason a collision occurs at node 2 when nodes 1 and 4 are sending at the same time, even though node 4 cannot directly communicate with node 2. Node 2 is within the interfering range of node 4.

Now, it is clear that the exposed station problem and collisions prevent the intermediated node from reaching its next hop. The random backoff scheme used in the MAC layer makes this worse. Since bigger data packet sizes and sending back-to-back packets both increase the chance of the intermediated node failing to obtain the channel, the node has to back off a random time and try again. This will increase the delay of ACKs if it finally succeeds. If it still fails after several tries, a link breakage will be declared. The result is a report of route failure. This explains why Reno in Fig. 2c does not have the instability problem. The maximum number for possible back-to-back sending is four. This greatly reduces the chance that other nodes might fail to access the channel in seven tries. Thus, no route failure occurs.

From the discussion of this problem, it is clear that IEEE 802.11-based multihop wireless networks might suffer from the serious exposed node problem and collisions. By adjusting one parameter in TCP, it is possible to lessen and eliminate the TCP instability problem. However, the in-depth problems still exist in the MAC layer. We will show another serious problem in this network in the next section, which cannot be eliminated by adjusting TCP parameters.

Serious Unfairness and Analysis

In this section we will examine the unfairness problems. Our results show that with the IEEE 802.11 MAC layer, simultaneous TCP traffic may suffer from severe unfairness, even between connections with the same number of hops. This problem is not the same as the former reported TCP unfairness problem, which is caused by the difference of TCP round-trip time. The present issue is rooted in MAC layer problems in multihop wireless links. In our experiment, one TCP connection might be completely shut down even if it starts much earlier than the competing TCP traffic.

We have found several kinds of unfairness problems. In this article, one simple case is illustrated as an example. We call it *neighboring node one-hop unfairness*. In each of the experiments presented below, we set up two TCP connections in the network shown in Fig 1. The first starts at 10.0 s, the second 20.0 s later. We will call them *first session* and *second session* in the following parts of this article. The whole experiment stops at 130.0 s.

Figure 4 shows the throughput for one run of such an experiment. In this experiment, the first session is from 6 to 4, the second from 2 to 3. The first session is a two-hop TCP. The first session has a throughput of around 450 kb/s after starting from 10.0 s. However, it is completely forced down after the second session starts at 30.0 s. In most of its lifetime after 30.0 s, the throughput of the first session is zero. There is not even a chance for it to restart. The aggregate throughput of these two TCP connections completely belongs to the second session — around 920 kb/s in the lifetime from 30.0 s to 130.0 s. This is also serious

Time (s)	Node	Seq. no.	Drop reason
30.1504	5	2164	NRTE
60.4217	6	2164	TOUT
61.0105	6	2164	TOUT
62.2259	6	2164	TOUT
64.5989	6	2164	TOUT
69.3867	6	2164	TOUT
78.9893	6	2164	TOUT
80.6576	5	2164	NRTE
117.3958	6	2164	TOUT
130.0000	6	2164	END
130.0000	6	2164	END

Table 1. *TCP packet drop events of the first TCP session in the WT run.*

unfairness. The loser session is completely shut down even if it starts much earlier.

Note the maximum window size (*window_*) of TCP in this experiment is set at 4. Unlike the TCP instability problem, the unfairness problem cannot be eliminated by adjusting this parameter. Since the TCP traffic in the stop-and-go case (*window_* = 1) is very simple, it can be used to demonstrate the cause of the unfairness problems. For this purpose, we list another example with *window_* = 1.

The experiment setting is almost completely the same as that shown in Fig 4, except the value of *window_*. The first TCP session is from node 6 to 4, the second from 2 to 3. Figure 5 shows the throughput for one run of the experiment. The first session has a throughput of around 440 kb/s after starting from 10.0 s. However, it is forced completely down after the second session starts. In most of its lifetime after 30.0 s, the throughput of the first session is zero. There is no chance even for it to restart. The aggregate throughput of these two TCP connections is almost completely supplied by the second session — around 900 kb/s in the lifetime from 30.0 s to 130.0 s. To simplify the expression, we call this the *W1 run* in the following statement.

Now we will explain why this happens. Figure 6 illustrates some of the TCP packet events in the W1 run. Figure 7 is its zoom. Obviously, after 30.07 s, no TCP packet of the first TCP session is delivered successively from source node 6 to the receiver. The packet with sequence number 2164 never arrives at the destination (node 4), although it is retransmitted 10 times. Note that in NS2, the TCP packet size is fixed in one connection, and the sequence number here is counted in packets (or segments) instead of bytes. A condensed version of the simulation packet traces is shown in Table 1; only drop events are listed. In this table, the drop reason column lists the reason why the packet is dropped — NRTE means no route available, TOUT means packet expired, and END means the simulation finished. The node and sequence number columns report the node at which the event occurred and the TCP sequence number of the packet depicted in the event. This table shows that the reason for the first TCP packet drop is the route failure in node 5. Since no node moves in our simulation, the route failure seems very strange. (We will explain why this happens below.) After the route failure is reported back to the source (node 6), a route discovery is triggered. Before a route to node 4 is found again, the TCP source retransmits the TCP packet after timeout. They are queued in the IFQ of node 6, waiting for forwarding. That is why we see several TCP packets expired and dropped. Unfortunately, the TCP packet never reaches node 4 after 30.07 s, even after a route to node 4 becomes available. This is because a route failure happens again very soon.

Now, we will look at the cause of route failure. By analyzing the simulation trace, we find that this problem is rooted in the MAC layer. Node 5 cannot reach node 4. After node 5 tries to contact node 4 and fails seven times, the MAC layer reports a link breakage. Note that retrying seven times is a parameter defined in IEEE 802.11. A part of the MAC layer packet trace is shown in Fig. 8. It is clear that the major cause of node 5 failing to reach 4 is the collision. Since node 5 can sense node 3, it has to defer when node 3 is sending, so it can only send out a RTS when node 3 is not sending. The result is that node 5 cannot send back CTS even if it receives the RTS correctly. However, the TCP connection from node 2 to 3 is only one hop. After node 2 receives the data packet (here it is a TCP ACK) from 3, it sends out a RTS to request the channel, preparing to send out another TCP packet. Once node 3 receives this RTS and replies with a CTS, node 2 starts sending the TCP packets. Normally, the size of this data packet is much larger than the control packets. If node 5 sends out an RTS for the channel to node 4, this control packet will experience a collision at node 4. There are altogether five

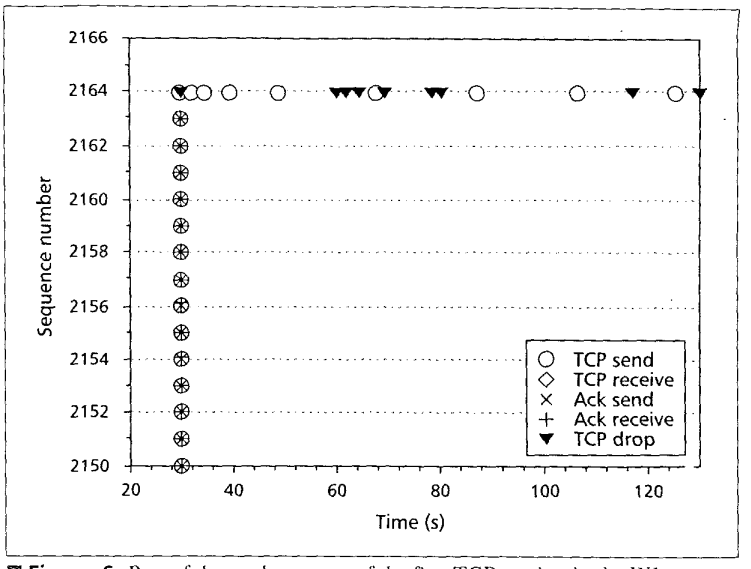

Figure 6. *Part of the packet events of the first TCP session in the W1 run, window_ = 1.*

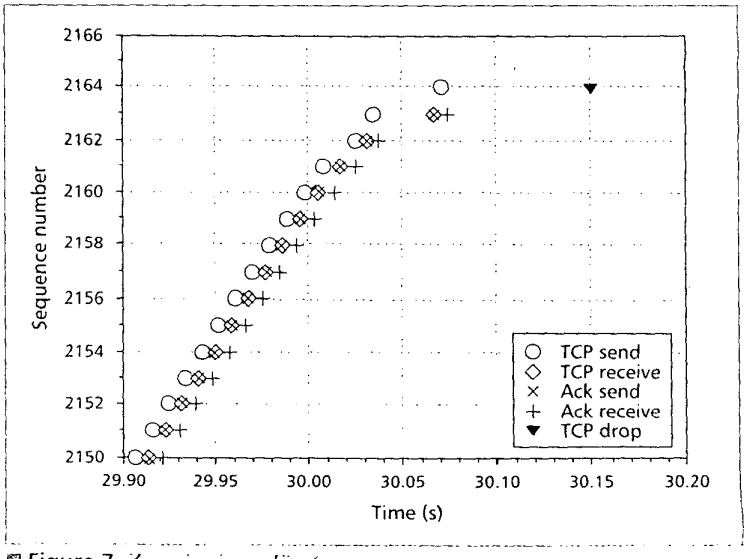

Figure 7. *Zooming in on Fig. 6.*

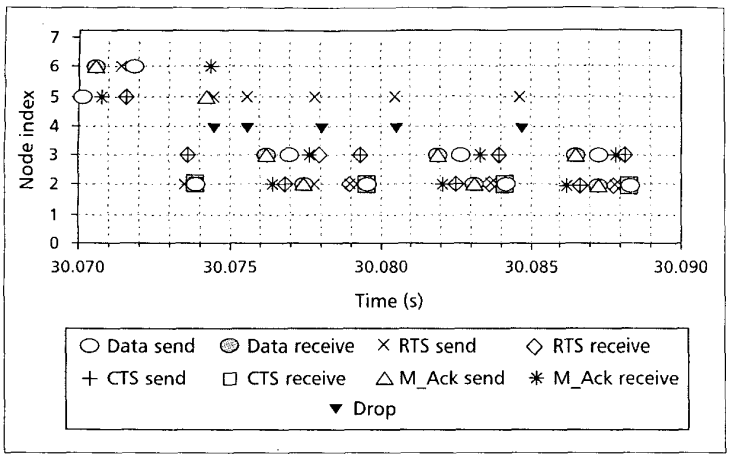

Figure 8. *Part of the MAC layer packet trace in the W1 run; window_ = 1.*

MAC packets dropped at node 4 in Fig. 8. Four of them are caused by collisions with the TCP data packet from node 2. One is caused by collision with the RTS control packet from node 2 (the middle of those five dropped MAC packets at 30.078 s in Fig. 8.) So the only chance for node 5 to access the channel to node 4 is by sending out an RTS before node 2 sends out an RTS. Note that this must be after node 3 finishes sending back the data packet (TCP ACK). The time window opening for node 5 to access the channel is very small. Also, because the binary exponential backoff scheme in the MAC layer always favors the last succeeding station (node 2 in this case), node 5 hardly wins the contention. After seven failures, it will quit and report a link breakage to its upper layer. Then a route failure event occurs.

As indicated in the last section, in a CS wireless network the interfering range (and sensing range) is typically larger than the communication range [7]. This is the reason a collision occurs at node 4 when node 2 and node 5 are sending at the same time, even though node 4 cannot directly communicate with node 2. Node 4 is within the interfering range of node 2.

We call this kind of unfairness *one-hop unfairness*. Note that the distance between each pair of neighboring nodes is the same (200 m) in our simulation. If the distances are not equal, the situation will be much more complicated. Due to space limitation, we do not discuss such a situation here. Anyway, since one-hop connection is the most popular case in a wireless ad hoc LAN, it is really an important problem that needs to be solved.

Besides one-hop unfairness, there are also other kinds of serious unfairness. The cause of them all is the same: the MAC layer does not function well in multihop wireless links.

Discussion and Related Works

In the last two sections we present two problems encountered in TCP sessions in an IEEE 802.11-based multihop wireless network. By illustrating multiple-layer packet traces, we conclude that the MAC layer is the cause of these problems. Since the TCP sets up a two-way connection to maintain reliability, and, more important, it can use as much bandwidth as possible in the network, it enlarges and intensifies the problems in the MAC layer. In other words, even if we do not use TCP, the problems still exist in the MAC layer when IEEE 802.11 is used in multihop networks. TCP traffic shows the problems existing in the MAC layer very clearly. In fact, these problems appear when the traffic load becomes large enough, even if the traffic is not from TCP.

More specifically, when it is used in a multihop network, the current IEEE 802.11 MAC protocol has the following problems:

- The hidden node problem still exists in multihop networks, although the standard has paid much attention to this problem. The protocol has defined several schemes to deal with this, such as physical carrier sensing and the RTS/CTS handshake. These schemes work well to prevent the hidden node problem in a wireless LAN where all nodes can sense each other's transmissions. The sufficient condition for not having hidden nodes is: any station that can possibly interfere with the reception of a packet from node A to B is within the sensing range of A. This might be true in an 802.11 basic service set; obviously, however, this condition cannot be true in a multihop network.
- There is no scheme in this standard to deal with the exposed node problem, which will be more harmful in a multihop network.
- The 802.11 MAC is based on carrier sensing, including the physical layer sensing function (CCA). As we know, carrier sense wireless networks are usually engineered in such a way that the sensing range (and interfering range) is typically larger than the communication range [12]. According to the IEEE 802.11 protocol implementation in the NS2 simulation software, which is modeled after the WaveLAN wireless radio, the interfering range and sensing range are more than two times the size of the communication range. The larger sensing and interfering ranges will degrade the network performance severely in the multihop case. The larger interfering range makes the hidden node problem worse; the larger sensing range intensifies the exposed node problem.
- The binary exponential backoff scheme always favors the latest successful node. This will cause unfairness, even when this protocol is not used in multihop networks, as in the typical wireless LAN defined in the IEEE 802.11 standard.

There might also be another factor that complicates this matter further. That is the term *ad hoc*. Note that in the document of standard IEEE 802.11 [2], "an ad hoc architecture" is clearly declared as supported. However, "ad hoc network" is defined as "a network composed solely of stations within mutual communication range of each other via the wireless media." The term *ad hoc* is often used as slang in [2] to refer to an independent basic service set.

Since wireless mobile ad hoc networks also use the same word, this leads to some confusion, at least at our early stage of related research. People imagine that the IEEE 802.11 MAC pro-

tocol should automatically support these kinds of networks, of which multihop connectivity is an important feature (if not, why would we need routing?). Thus, it is better to use the word *multihop* explicitly when we refer to wireless mobile ad hoc networks.

Recently, several researchers have studied the performance of the MAC layer on multihop networks. Gerla *et al.* [7, 8] investigated the impact of the MAC protocol on performance of TCP in multihop networks. They found that the interaction between the TCP and MAC layer backoff timers causes severe unfairness and capture conditions. The reported unfairness in these two papers is slight compared to that in our article. A yield time scheme is proposed to address the unfairness problem in 802.11; however, this will cause the aggregated throughput to degrade badly. Moreover, we do not think this scheme can solve the unfairness problem reported in this article, since it is not caused by one node capturing the channel.

In several state-of-the-art published works, a so-called fairness problem in 802.11 MAC protocol is addressed [9, 10]. This problem is similar to the third one mentioned above, which is caused by the backoff scheme in 802.11. These papers propose some fairer backoff schemes to replace that defined in the standard. This surely will help to improve fairness in wireless LANs. However, they do not address the first two problems we described above. So these proposals cannot eliminate all above mentioned problems existing in 802.11 MAC layer when used in a wireless multihop network.

Besides changing the backoff policy, other potential resolutions for these problems might include adjusting the interfering (and sensing) range. Some schemes to deal with the exposed node problem are also helpful. Note that the latest efforts in the IEEE 802.11 WG include some quality of service schemes. They will be helpful to address these problems, but they cannot eliminate all of them since they still merely focus on the IBSS or ESS scenario.

Conclusions

In this article we focus on the following question: Can the IEEE 802.11 MAC protocol function well in multihop networks? The IEEE 802.11 MAC protocol is the standard for wireless LANs, and, more important, is widely used in almost all testbeds and simulations for research on wireless multihop ad hoc networks. However, this protocol was not designed for multihop networks. Although it can support some kind of ad hoc network architecture, which only means a distributed network as opposed to a centralized one, it is not intended to support the wireless mobile ad hoc network, in which multihop connectivity is one of the most prominent features.

By presenting several serious problems encountered in an IEEE 802.11-based multihop network and revealing the underlying causes of them, we conclude that the current version of this wireless LAN protocol does not function well in multihop ad hoc networks. We also indicate the specific problems existing in this protocol when used in a multihop network. Based on this analysis, we point out the potential direction to resolve those problems.

So we doubt whether the WaveLAN-based systems are workable as a mobile ad hoc testbed, even if they are only used to test the routing protocols. As shown in this article, the MAC layer problem can cause the routing protocol to fail. And more efforts on the MAC layer are needed to design a usable wireless mobile network.

Acknowledgments

We would like to thank the Monarch group at CMU, and the VINT group at USC and Berkeley for making the NS2 simulator available.

References

[1] J. P. Macker and M. S. Corson, "Mobile Ad Hoc Networking and the IETF," *ACM Mobile Comp. and Commun. Rev.*, vol. 2, no. 1, Jan. 1998.

[2] IEEE Std. 802.11, "Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications," 1999.

[3] K. Fall and K. Varadhan, notes and documentation, LBNL, Aug. 1998: http://www_ mash.cs.berkeley.edu/ns

[4] J. Broch *et al.*, "A Performance Comparison of Multihop Wireless Ad-hoc Network Routing Protocols," *ACM/IEEE Int'l. Conf. Mobile Comp. and Networking*, Oct. 1998, pp. 85–97.

[5] C. Barakat, E. Altman, and W. Dabbous, "On TCP Performance in a Hetergeneous Network: A Survey," *IEEE Commun. Mag.*, Jan. 2000.

[6] J. L. Sobrinho and A. S. Krishnakumar, "Quality-of-Service in Ad Hoc Carrier Sense Multiple Access Wireless Networks," *JSAC*, 1999, vol. 17, no. 8, pp. 1353–68.

[7] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multihop Networks," *IEEE WMCSA '99*, New Orleans, LA, Feb. 1999.

[8] K. Tang and M. Gerla, "Fair Sharing of MAC under TCP in Wireless Ad-hoc Networks," *Proc. IEEE MMT '99*, Venice, Italy, Oct. 1999.

[9] B. Bensaou, Y. Wang, and C. C. Ko, "Fair Media Access in 802.11 Based Wireless Ad-hoc Networks," *Proc. Mobihoc 2000*, Boston, MA, Aug. 2000.

[10] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN," *Proc. Mobcom 2000*, Boston, MA, Aug. 2000.

Biographies

SHUGONG XU (xu001@ieee.org) received a B.Sc. degree from Wuhan University, China, and his M.E. and Ph.D. from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1990, 1993, and 1996, respectively. Between 1997 and 1999 he was with Tsinghua University, China, and Michigan State University. In 1999 he joined the Information Networking and Telecommunications group at City College of New York (CCNY), where he is a research scientist. His current research interest lies in wireless LAN, wireless mobile ad hoc networks, Internet QoS, and multimedia communication. He has published over two dozen technical papers in these areas.

TAREK N. SAADAWI received a B.Sc. and an M.Sc. from Cairo University, Egypt, and a Ph.D. from the University of Maryland, College Park (all in electrical engineering). Since 1980 he has been with the Electrical Engineering Department, City University of New York, City College, where he currently directs the Information Networking and Telecommunications group at CCNY. His current research interests are telecommunications networks, high-speed networks, multimedia networks, ad hoc networks, and packet radio networks. He has published extensively in the area of telecommunications networks. He is a co-author of the book *Fundamentals of Telecommunication Networks* (Wiley, 1994). He is also the lead author of the Egypt Telecommunications Infrastructure Master Plan covering the fiber network, IP/ATM, DSL, and the wireless local loop. He is a former Chair of IEEE Computer Society of New York City (1986–1987). He has received the IEEE Region 1 Award, 1987, and the Nippon Telegraph and Telephone (NTT) of America award for research on broadband telecommunication networks.

By presenting several serious problems encountered in an IEEE 802.11-based multihop network and revealing the underlying causes of them, we conclude that the current version of this wireless LAN protocol does not function well in multihop ad hoc networks.