ELSEVIER

Computer Networks 38 (2002) 531–548

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks ☆,☆☆

Shugong Xu [a,*], Tarek Saadawi [b]

[a] *Sharp Labs of America, 5750 NW Pacific Boulevard, Camas, WA 98607, USA*
[b] *Department of Electrical Engineering, City University of New York, City College, New York, NY 10031, USA*

Received 1 September 2001; accepted 11 October 2001

Responsible Editor: I.F. Akyildiz

Abstract

The IEEE 802.11 medium access control (MAC) protocol is a standard for wireless LANs, it is also widely used in almost all test beds and simulations for the research in wireless mobile multi-hop ad hoc networks. However, this protocol was not designed for multi-hop networks. Although it can support some ad hoc network architecture, it is not intended to support the wireless mobile ad hoc network, in which multi-hop connectivity is one of the most prominent features. In this paper, we focus on the following question: can IEEE 802.11 MAC protocol function well in multi-hop networks? By presenting several serious problems encountered in transmission control protocol (TCP) connections in an IEEE 802.11 based multi-hop network, we show that the current TCP protocol does not work well above the current 802.11 MAC layer. The relevant problems include the TCP instability problem found in this kind of network, the severe unfairness problem, and the incompatibility problem. We illustrate that all these problems are rooted in the MAC layer. Furthermore, by revealing the in-depth cause of these problems, we conclude that the current version of this wireless LAN protocol does not function well in multi-hop ad hoc networks. We thus doubt whether the current WaveLAN based system is workable as a mobile multi-hop ad hoc test bed. All the results shown in this paper are based on NS2 simulations, and are compatible with the results from the OPNET simulations.

Keywords: 802.11; MAC; Multi-hop; Ad hoc networks; TCP

☆ Prepared through collaborative participation in the Advanced Telecommunications & Information Distribution Research Program (ATIRP) Consortium sponsored by the US Army Research Laboratory under the Federal Laboratory Program, Cooperative Agreement DAAL01-96-2-0002. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the US Government.

☆☆ This work was done when the first author was with the City College, CUNY, New York.

* Corresponding author. Tel.: +1-360-8348672; fax: +1-306-8348696.
E-mail address: xu001@hotmail.com (S. Xu).

1389-1286/02/$ - see front matter
PII: S1389-1286(01)00273-0

1. Introduction

Wireless Ad hoc networks are required where a fixed communication infrastructure, wired or wireless, does not exist or has been destroyed. In a multi-hop ad hoc network, nodes communicate with each other using multi-hop 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. One 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 paper we discuss another challenge in wireless multi-hop ad hoc networks—medium access control (MAC). As the media is a shared and scarce resource in a wireless network, efficiently control the access to this shared media becomes a complicated task. A great deal of effort has been applied to this task, and many MAC layer protocols have been proposed. However, few of them were designed to be used in multiple hop wireless links; furthermore, few of them have been evaluated in multi-hop networks. In this paper, we focus on the IEEE 802.11 MAC layer protocol and examine the problems caused by multi-hop 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 it is widely used in almost all test beds and simulations for wireless ad hoc network research. An important and natural question is whether the IEEE 802.11 MAC protocol works well in multi-hop ad hoc networks.

This question comes out when we evaluate the performance of transmission control protocol (TCP) in the 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 effectively, it will be impractical 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 paper, TCP traffic intensifies the problem in this MAC layer protocol when it is used in IEEE 802.11 based multi-hop ad hoc networks.

In this paper, 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 multi-layer traces produced by the simulation, we reveal the in-depth cause of these problems and offer the possible solutions. Before doing this, however, we will give an overview of the IEEE 802.11 standard in Section 2. The simulation environment and methodology we used for our simulations are reported in Section 3. Sections 4–6 provide respectively the three kinds of problems encountered in an 802.11-based wireless multi-hop network. By analyzing the multi-layer simulation traces, we will reveal the underlying causes for these problems. Discussion and related works are presented in Section 7. Finally, Section 8 provides the conclusion.

2. An overview of IEEE 802.11 standard [2]

Like any 802.x protocol, the 802.11 protocol covers the MAC and physical layers. The standard defines a single MAC which interacts with different PHYs. The MAC Layer defines two different access methods—the distributed coordination function and the point coordination function. We now describe the distributed coordination function (DCF) in detail (since the PCF can not be used in ad hoc networks it is not described here).

The basic access mechanism, called the distributed coordination function, is basically a carrier sense multiple access with collision avoidance mechanism (usually known as CSMA/CA). CSMA protocols are well known in the industry, the most popular being the Ethernet, which is a CSMA/CD protocol (CD standing for collision detection). 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 trans-

mitting 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, transmitting at the same time and 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 collision avoidance (CA) mechanism coupled with a positive acknowledge scheme, as follows:

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

In order to reduce the probability of two stations colliding due to not hearing each other—is well known as the ''hidden node problem''—the standard defines a virtual carrier sense 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 the 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 carrier sense indicator (called NAV, for network allocation vector), for the given duration, and use this information together with the physical carrier sense 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.

Besides the hidden node problem, the wireless packet networks face the exposed node problem. 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 the multi-hop wireless networks. We will discuss this in more detail in the next sections.

3. Simulation environment and methodology

Before proceeding further we need to introduce the simulation environment and methodology. The results reported in this paper are based on simulations using the NS2 network simulator from Lawrence Berkeley National Laboratory (LBNL) [4], with extensions from the MONARCH project at Carnegie Mellon [5]. 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 Refs. [4,5]. Besides of these simulations using NS2, we have also conducted similar simulations using OPNET to validate our NS2 simulation. More detail about

the OPNET simulations will be published in another paper. The results from them are pretty positive. They are compatible with the results we will present in this paper, and support our conclusion about the MAC layer issues. In the following parts of this paper, we will focus on the NS2 simulations.

For better demonstrate our results and analysis, a brief introduction about the physical and data link layer model in NS2 is necessary. The signal propagation model combines both a free space propagation model and a two-ray ground reflection model. When a transmitter is within the reference distance r of the receiver, we use the free space model where the signal attenuates as $1/r^2$. Outside of this distance, we use the ground reflection model where the signal falls off as $1/r^4$. The position of the mobile nodes are used by the radio propagation model to calculate the propagation delay from one node to another, and to determine the power level of a received signal at each mobile node.

Each mobile node has one wireless network interfaces, with all interfaces of the same type (on all mobile nodes) linked together by a single physical channel. When a network interface transmits a packet, it passes the packet to the appropriate physical channel object. This object then computes the propagation delay from the sender to every other interface on the channel and schedules a "packet reception" event for each. This event notifies the receiving interface that the first bit of a new packet has arrived. At this time, the power level at which the packet was received is compared to two different values: the carrier sense threshold and the receive threshold. If the power level falls below the carrier sense threshold, the packet is discarded as noise. If the received power level is above the carrier sense threshold but below the receive threshold, the packet is marked as a packet in error before being passed to the MAC layer. Otherwise, the packet is simply handed up to the MAC layer.

Once the MAC layer receives a packet, it checks to insure that its receive state is presently "idle." If the receiver is not idle, one of two things can happen. If the power level of the packet already being received is at least 10 dB greater than the received power level of the new packet, we assume capture, discard the new packet, and allow the receiving interface to continue with its current receive operation. Otherwise, a collision occurs and both packets are dropped. If the MAC layer is idle when an incoming packet is passed up from the network interface, it simply computes the transmission time of the packet and schedules a "packet reception complete" event for itself. When this event occurs, the MAC layer verifies that the packet is error free, performs destination address filtering, and passes the packet up the protocol stack.

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 the commercially available 802.11-based WaveLan wireless radios, which have a bandwidth of 2 Mbps and a nominal transmission radius of 250 m. The reference distance r is 100 m.

With a few exceptions, we chose to keep most of the parameters of the simulations used in Ref. [5]. 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 of multi-hop connectivity. Only a portion of the nodes in this network is 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 the wireless

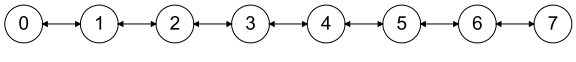


Fig. 1. String topology.

mobile ad hoc networks. It is need to be emphasized that we use this simple network scenario to show the possibility of the problem. It is clear that this is not the only one scenario in which the problems could happen. To determine when these problems will happen is beyond the scope of this paper, which need further investigation. Also, we do not intend to imply that this is the most possible scenario for those problems. We have observed these problems in many different network scenarios. The only reason we show the results from it here, is that it is the first network scenario from which we captured these TCP layer problems and MAC layer misbehaviors. Anyway, this does not, and should not harm our following analysis and our conclusions.

In this paper, 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 of data which the TCP sender always has to send out). The TCP packet size was 512 bytes unless otherwise indicated. Note that the current version NS2 software can only support a fixed size TCP packet in each simulation. That is why we can use the packet index as the TCP sequence number in the next section. Now we offer more explanation for our use of TCP in this paper.

The TCP is the prevalent reliable transport protocol used in the internet today. This is the first reason why we use TCP traffic to demonstrate the MAC layer problems of IEEE 802.11 in multi-hop ad hoc networks. To make this network function well, the TCP protocol must be supported.

Besides reliability, 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 we mentioned in the introduction above, TCP traffic enlarges the problems of the MAC layer protocol.

In the following, we present a brief introduction to TCP. It is a window-based ACK-clocked flow control protocol. (Note that here ACK means TCP layer acknowledgment from the 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 non-duplicate ACK until the resource estimate of network capacity is reached. This is the slow start (SS) phase, and the capacity estimate is called the SS 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, aims to slowly probe the network for any extra bandwidth. The window increase will stop when it reaches the maximum TCP window size, which is defined when the connection starts. Otherwise, the 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 a little difference among different TCP versions in the processing method for a loss. The source supposes that the network congested and sets its estimate of the capacity to half the current window. TCP Reno version has a fast recovery algorithm to retransmit the losses. We will use this TCP variance as an example, which is now the most popular version. For more information about TCP, please refer to Ref. [6] and the references therein.

Now, we are ready to go to the main part of this paper. We will present three problems existing in the IEEE 802.11 based multi-hop wireless ad hoc networks. They are the TCP instability problem, the unfairness problem, and the incompatibility 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 used in a multi-hop environment. After the description for each problem, we will illustrate its underlying cause by showing the multiple layer traces.

4. The transmission control protocol instability problem and analysis

In the first set 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 a network. No background traffic exists. Hence there are no network condition changes in the whole lifetime of each experiment. As we mentioned earlier in this paper, TCP can adaptively adjust its transmission rate according to the network condition. If the network condition does not vary, the TCP throughput should stay in some level. 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 a 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 is node 5. TCP packet size is 1460 Byte. Note that the current version NS2 software can only support fixed size TCP packets in each simulation. This does not hurt the universality of our conclusions.

Fig. 2 shows the related results. It includes three small figures. Each of them illustrates the measured throughput variations during the lifetime of one simulation run. The plotted values of the throughput are measured over each 1.0 second 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. 4(a) at first. In the 120 s lifetime of this connection, there are 20 instances when the throughput reaches or nears zero. In those 1.0 s interval, almost no TCP packets were successively received. That means the 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 multi-hop network. We call this "instability"of the TCP in this specific kind of network. The TCP maximum window size (*window_*) this problem. Since this is the limit of the real transmission window size in a TCP connection, the TCP instability problem can be lessened or eliminated with a smaller maximum window size. In Fig. 2(a), this parameter is set as 32. Fig. 2(b) demonstrates serious oscillation with a *window_* of 8 and a packet size of 1460. The results are better than those associated with a

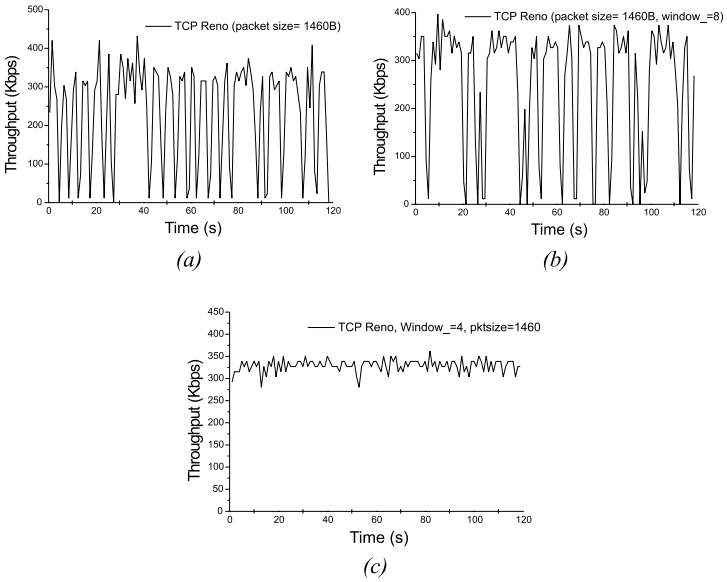


Fig. 2. Instability problem in the four hops TCP connection. (From node 1 to node 5) TCP packet size = 1460B, (a) Reno, *window_* = 32, (b) Reno, *window_* = 8 and (c) Reno, *window_* = 4.

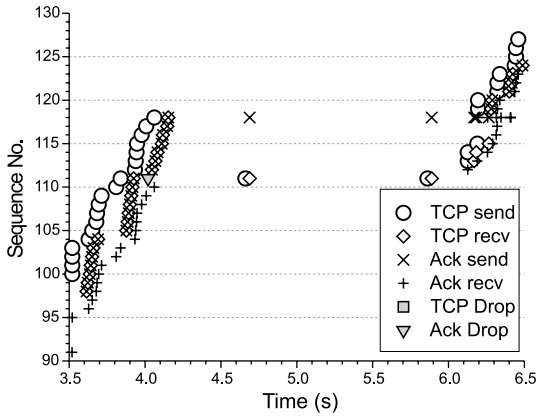


Fig. 3. Part of the packet events of TCP session shown in Fig. 2(b), Reno, *window_* = 8, packet size = 1460 byte.

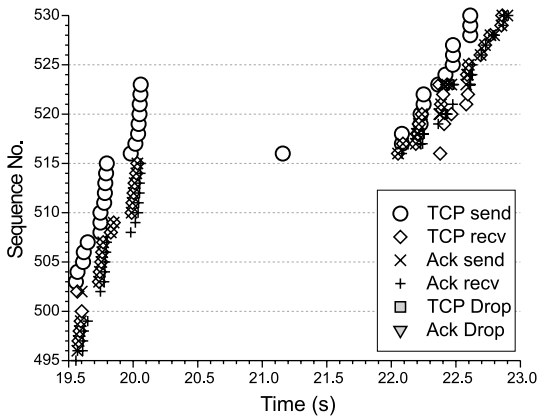


Fig. 4. Another part of the packet events of TCP session in Fig. 2(b), Reno, *window_* = 8, packet size = 1460 byte.

*window_* = 32, but the oscillation is still very serious. In the 120 s lifetime of this TCP connection, there are 16 moments when the throughput reaches or nears zero! Fig. 2(c) shows the results of *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 caused by one node failing to reach its adjacent node. This triggers a route failure. If it is an intermediated node, the node drops all queued packets (ACK in most cases) to the 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 a 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.

For example, we will use the TCP connection shown in Fig. 2(b) to demonstrate the cause of this problem. This TCP connection is from node 1 to 5. As we can see in that figure, the throughput falls to zero at around 4.0 s. Fig. 3 illustrates the packet events from this part of the simulation. Obviously, after the ACK drops at 4.02 s, no ACK packet arrives at the TCP sender until a route from node 5 to 1 becomes available again after 6.1 s. In this period, the TCP packet with sequence number 111 is retransmitted two times. Although these two packets arrive at the TCP receiver (node 5) safely, the corresponding ACK packet cannot be sent out, since no route is available. Note that in NS2, the TCP packet size is fixed and the sequence number here is counted in packets (or segments) instead of bytes. Fig. 4 illustrates another part of the trace. It shows the packet events from 19.5 to 23.0 s. Fig. 2(b) evinces another throughput degradation in this period. It is the TCP packet 516 that cannot find a route this time. Note that there is no data packet drop due to route failure. As we will show below, the route failure is due to node 1 not reaching node 2. Since node 1 is the TCP sender, the TCP packet in the IFQ of this node will not be dropped. It will trigger a route discovery immediately after the route failure is reported. Fig. 5 is a zoom of Fig. 4 around 20.0 s.

Now we will look at the cause of route failure, focusing on the case shown in Figs. 4 and 5. By analyzing the simulation trace, we find that this 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 seven retries is a parameter defined in IEEE 802.11. A part of the MAC layer packet trace is shown in Fig. 6. In this figure, *Data* means TCP packet or TCP ACK packet. In reference to the MAC layer, they are all data from upper layer. RTS means "request to send"; CTS means "clear to send". M-Ack means the MAC layer acknowledgement. (Refer to Refs. [4,5] for

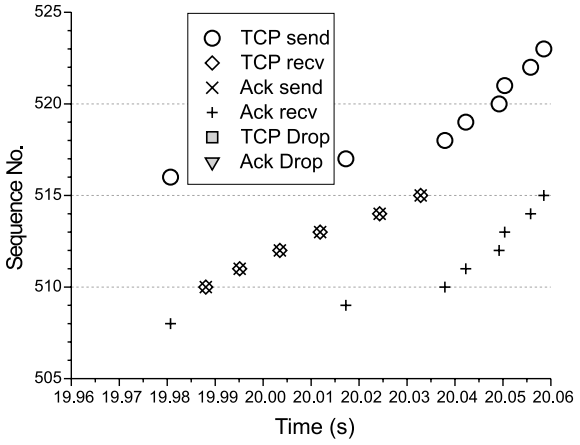


Fig. 5. Zoom of Fig. 4, for session in Fig. 2(b), Reno, *window_* = 8, packet size = 1460 byte.

more details about the implementation of IEEE 802.11 MAC layer in NS2 software.) From this figure, we can find that the major cause of node 1's failure to reach node 2 is that node 2 cannot successfully receive the RTS of node 1. Node 1 sends out seven RTS packets between 19.981 and 20.012 s. However, node 2 only successively receives three of these RTS packets. For a reason we will discuss later, node 2 does not send back a CTS to node 1 for all three RTS packets. Due to collision, the other four RTS packets are dropped at node 2. Note that there are, in total, five MAC packets dropped in this figure. Four of them occur at node 2. The other one occurs at node 1. This is because node 1 quits the delivery of the MAC layer data packet. The MAC layer data packet is dropped by node 1. At the same time, a link breakage event is reported to the upper layer. Of course node 1 does not receive any CTS from node 2. Thus, after each failure to get a reply, node 1 defers a random back-off period before transmitting again. The back-off interval is chosen using the binary exponential back-off scheme.

Let us focus on what happens to the first RTS packet dropped at 19.9826 s. After deferring a while, node 1 sends out the second RTS. That is the one sending out at 19.9826 s. At this time, node 4 is sending a data packet to node 5. Since node 1 cannot sense the transmission occurring at node 4, it sends out the RTS packet. Unfortunately, this packet experiences a collision at node 2. So, the cause of this collision must be the interference from node 4. There are, in total, four MAC packets dropped at node 2 in Fig. 6. All of them are caused by a collision with the TCP data packet from node 4. It must to be stated that, in a carrier sense 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 [7]. WaveLAN wireless systems are engineered in such a way. 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 the sensing range are more than two times the size of the communication range. This is the reason why a collision occurs at node 2 when node 1 and node 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. This is a typical "hidden node problem" in wireless packet networks. Node 4 is the hidden

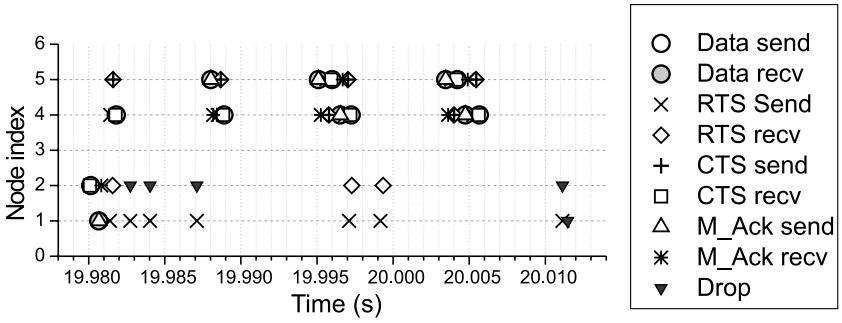


Fig. 6. Part of the MAC layer packet trace, for session in Fig. 2(b), Reno, *window_* = 8, packet size = 1460 byte.

node in this case. It is within the interfering range of the intended destination (node 2) but out of the sensing range of the sender (node 1). Since the nominal communication range is 250 m, which is smaller than the interfering range, node 1 cannot hear the CTS packet from node 4. Thus the virtual carrier sense mechanism cannot function, either in this case. Now, we can explain why node 2 still cannot send back a CTS when node 4 is sending, even if node 2 successfully receives the RTS from node 1. Note that node 2 can sense node 4. This is a typical "exposed node problem" in wireless packet networks.

Since node 4 is sending several back-to-back TCP data packets, the channel is keeping busy. After failing seven times to receive CTS from node 2, node 1 quits and reports a link breakage to its upper layer. Then a route failure event occurs. The TCP session has to pause until the route becomes available again. Since the time of recovering from a route failure is usually more than one second, as shown in Figs. 3 and 4, the TCP throughput in this period is zero. Since this time period is larger than the TCP timeout threshold, the TCP session has to restart from a window of one after the route is discovered in the TCP source node. It also needs to do retransmissions for those un-acked packets. This will further damage the TCP good-put.

Now, it is clear that the exposed station problem and collisions are preventing the intermediated node from reaching its next hop. The random backoff scheme used in the MAC layer makes this worse, since it always favors the latest successful node. As 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 backoff 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 the report of a route failure. This also explains why Reno in Fig. 2(c) does not have the "instability problem". The maximum number for possible back-to-back sending is four. This greatly reduces the chance that other node may fail to access the channel in seven tries. Thus, no route failure occurs.

Since the distance between each pair of neighboring nodes in our experiments is 200 m, and each node has a nominal transmission radius of 250 m, we did not think node 4 could interfere with node 2 before this work. This means that we did not expect the hidden node and exposed node problems to exist in such a simple multi-hop 802.11-based network. In fact, no existing work, including the latest papers on 802.11 unfairness [9–11], ever talks about these problems. However, as we have shown above, these two problems are inherent when we use 802.11-based radio in multi-hop networks. They may cause many problems and affect the performance of these networks.

From the discussion of this problem, it is clear that IEEE 802.11 based multi-hop wireless networks might suffer from serious exposed node problem and collisions. By adjusting one parameter in TCP, it is possible to lessen and eliminate the TCP instability problem. However, 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 the TCP parameter.

5. Serious unfairness and analysis

In this section, we will examine unfairness problems. Our results show that, with the IEEE 802.11 MAC layer, two simultaneous TCP traffics may suffer from unacceptable 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 due to the difference of TCP round trip time. This issue is rooted in MAC layer problems in multi-hop 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 paper, one simple case will be illustrated as an example. We call it "neighboring node one-hop unfairness". In each of the experiments presented below, we setup two TCP connections in the network shown in Fig. 1. The first one starts at 10.0 s, the second one begins 20.0 s later. We will call the first one "first session" and

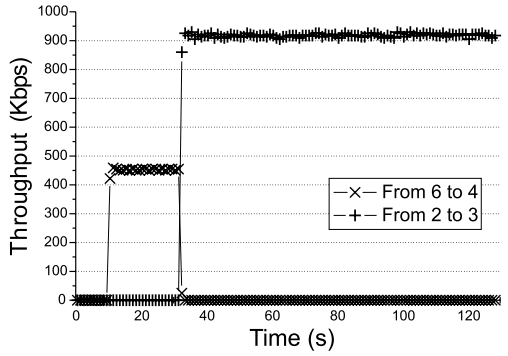


Fig. 7. Throughput of two TCP connections with different sender and receiver, *window_* = 4.

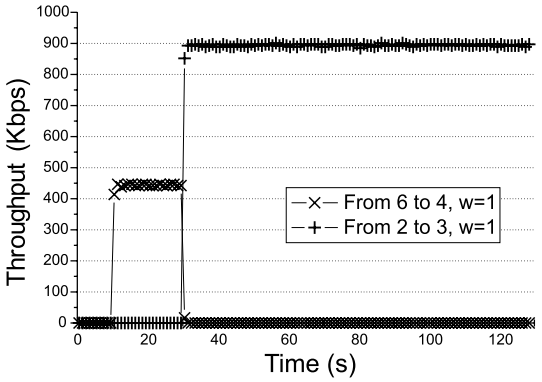


Fig. 8. Throughput of two TCP connections with different sender and receiver, *window_* = 1.

the second one "second session". The whole experiment lasts 130.0 s.

Fig. 7 shows the throughput for one run of such an experiment. In this experiment, the first session is from 6 to 4; the second one is from 2 to 3. The first session is a two hop TCP. The first session has a throughput of around 450 Kbps 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 belongs completely to the second session—around 920 Kbps in the 30.0–130.0 s lifetime. This is also serious unfairness. The loser session is completely shut down even if it starts much earlier.

Note that 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 identical to that shown in Fig. 7, excepting the value of the *window_*. The first TCP session is from node 6 to 4, the second one is from 2 to 3. Fig. 8 shows the throughput for one run of the experiment. The first session has a throughput of around 440 Kbps from 10.0 s after starting. However, it is shut down completely 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 for it to restart. The aggregate throughput of these two TCP connections is almost completely supplied by the second session—around 900 Kbps in the 30.0–130.0 s lifetime. We call this run "W1 run" in the following statement.

Now we will explain why this happens. Fig. 9 illustrates some of the TCP packet events in the "W1 run". Fig. 10 is its zoom. Obviously, after 30.07 s, no TCP packet from the first TCP session is delivered successfully from source node 6 to the receiver. The packet with sequence number 2164 never arrives at the destination (node 4), although

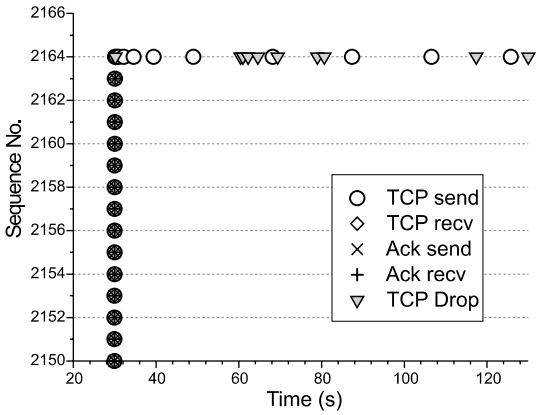


Fig. 9. Part of the packet events of the first TCP session in the "W1 run", *window_* = 1.

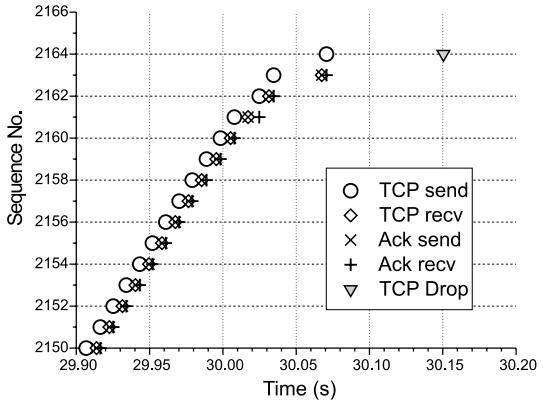


Fig. 10. Zoom of Fig. 9.

it is retransmitted ten 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 trace is shown in Table 1; only packet drops 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, SeqNo columns report the node at which the event occurred, and the TCP sequence number of the packet depicted in the event. From this table, we find 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 a little bit later.) 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 expired and dropped TCP packets. Unfortunately, the TCP packet never reaches node 4 after 30.07 s—even after a route to node 4 becomes available. That is because the route failure happens again very shortly.

Table 1
TCP packet drop events of the first TCP session in the "W1 run"

Time (s)	Node	SeqNo	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

Now, we will look at the cause of the route failure. By analyzing the multi-layer simulation trace, we find that the route failure 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 the parameter of seven retries is defined in IEEE 802.11. A part of the MAC layer packet trace is shown in Fig. 11. From this figure, we find that the cause of node 5's failure to reach 4 is that node 4 cannot successfully receive the RTS of node 5. Of course node 5 does not receive any CTS from node 4. Thus, after each failure to get a reply, node 5 defers a random back-off period before transmitting again. The back-off interval is chosen using the binary exponential back-off scheme. Node 5 sends out seven RTS packets between 30.074 and 30.15 s. However, node 4 never successively receives any of these RTS packets. Due to collision, they are dropped at node 4. Note that there are, in total, eight MAC packets dropped in this figure. Seven of them occur at node 4. The other one occurs at node 5. This is because node 5 quits the delivery of the MAC layer data packet. The MAC layer data packet is dropped by node 5. At the same time, a link breakage event is reported to the upper layer. In order to see clearly why the RTS packets from node 5 experience collision at node 4, we present Fig. 12, which is a zoom of Fig. 11.

Let us focus on what happens before the first collision at 30.0744 s. After it receives the data packet from node 6 at 30.0741 s and sends back an M_Ack to node 6, node 5 tries to forward the packet to node 4. That is why it sends out an RTS to node 4 at 30.0744 s. At this time, node 2 is sending a data packet to node 3. Since node 5 cannot sense the transmission occurring at node 2, it sends out an RTS packet immediately.

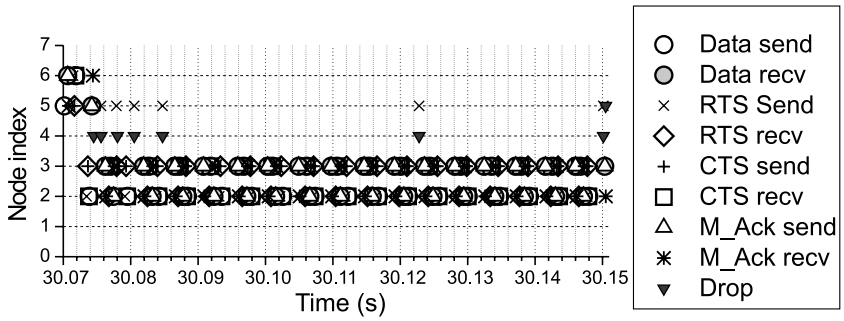


Fig. 11. Part of the MAC layer packet trace in the "W1 run", *window_* = 1.

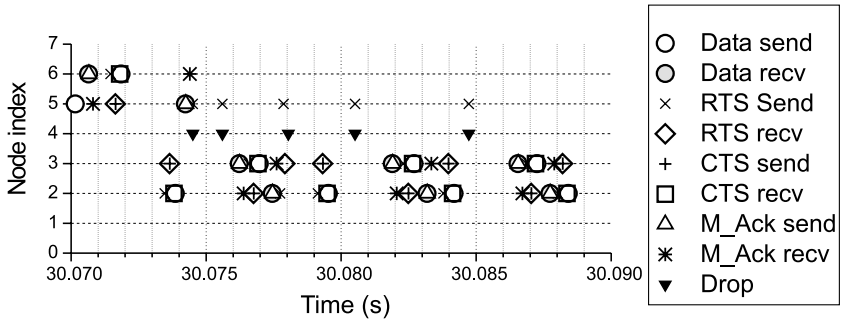


Fig. 12. Zoom of Fig. 11, Part of the MAC layer packet trace in the "W1 run", *window_* = 1.

Unfortunately, this packet experiences a collision at node 4. So, the cause of this collision must be the interference from node 2. There are, in total, five MAC packets dropped at node 4 in Fig. 12. Four of them are caused by the collision with the TCP data packet from node 2. One is caused by a collision with the RTS control packet from node 2 (the middle one of these five dropped MAC packets, at 30.078 s in Fig. 12). Since node 4 is within the interfering range of node 2, a collision occurs at node 4 when node 2 and 5 are sending at the same time—even though node 4 cannot directly communicate with node 2. This is a typical "hidden node problem"in wireless packet networks. Node 2 is the hidden node in this case. It is within the interfering range of the intended destination (node 4) but out of the sensing range of the sender (node 5). Since the nominal communication range is 250 m, which is smaller than the interfering range, node 5 cannot hear the CTS packet from node 3. Thus the virtual carrier sense mechanism cannot function in this case. Furthermore, even if node 4 successfully receives the RTS from node 5, it still cannot send back a CTS when node 2 or 3 is sending. Note that node 4 can sense node 3 and 2. This is a typical "exposed node problem" in wireless packet networks. We will provide more discussion on this in Section 7.

Now we inspect in what conditions node 5 can reach node 4 if there is a TCP session between node 2 and 3. Since node 5 can sense node 3, it has to defer when node 3 is sending. So, it can only send out an RTS when node 3 is not sending. 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 an 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 will start sending the TCP packet. 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. 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 this should 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. And also because the binary exponential back-off scheme in the MAC layer always favors the last succeeding station (node 2 in this case), node 5 hardly wins the contention. After seven attempts, it will quit and report a link breakage to its upper layer. Then a route failure event occurs.

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 in this paper. Anyway, since one hop connection is the most popular case in wireless ad hoc networks, it is really an important problem that needs to be solved.

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

6. Incompatibility problem

In this section, we will report another kind of problem—the incompatibility problem. We will show that with the IEEE 802.11 MAC layer, two simultaneous TCP traffics cannot coexist in the network at the same time. Once one session develops, the other one will be shut down. Unlike the unfairness problem in the last section, there is no definite winner between these two TCP sessions. The overturn can happen at any random time. In our following experiments, the two TCP connections have the same path length (i.e. hop number). They have the same long term average throughput; however, the short term behavior is unacceptable—the currently active TCP connection might be completely shut down at any time. We will show that this issue is also rooted in MAC layer problems in multi-hop wireless links.

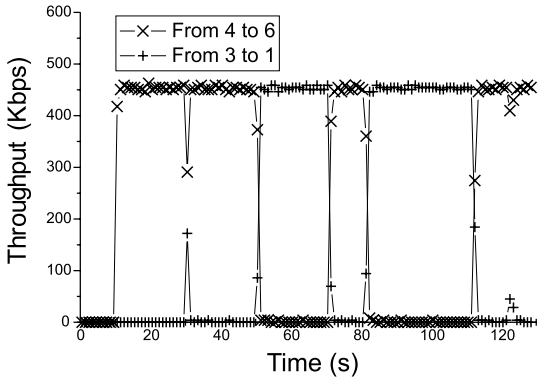


Fig. 13. Throughput of two TCP connections with same hop numbers, $window\_ = 4$.

Like we did in the last section, in each of the following experiments we setup two TCP connections in the network scenario shown in Fig. 1. The first one starts at 10.0 s, the second one begins 20.0 s later. We call the first one "first session" and the second one "second session". The experiment stops at 130.0 s.

Fig. 13 shows the throughput for one run of such an experiment. In this experiment, the first session is from 4 to 6; the second one is from 3 to 1. Both sessions have only two hops between the TCP source and the destination. The first session has a throughput of about 450 Kbps after starting from 10.0 s. Unlike the unfairness case described in the last section, it is not forced down when the second session starts at 30.0 s. Instead, it stays alive until 51.0 s. At the same time, the second session finally develops at the expense of the first session's shut down. It also has as throughput of around 450 Kbps when it is alive. However, it cannot always maintain this fortunate state. It is shut down at 71.0 s and the first session becomes alive with a throughput of around 450 Kbps. Two more turnovers occur at 82.0 and 111.0 s, respectively. In the lifetime of this experiment, this turnover happens four times. The aggregate throughput of these two TCP connections is always around 450 Kbps in the 30.0–130.0 s lifetime. However, sometimes the aggregate throughput belongs to the first session and at the other times it belongs to the second one. The two TCP sessions cannot keep alive at the same time. This is a

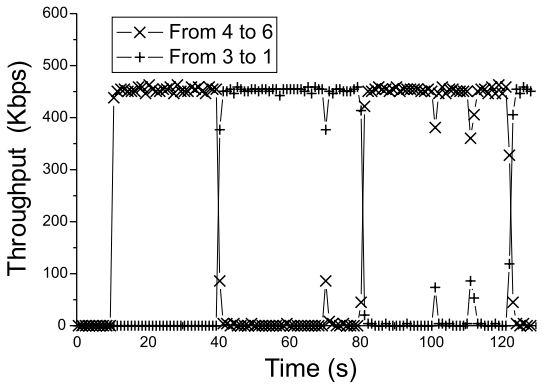


Fig. 14. Throughput of two TCP connections with same hop numbers (another run), *window_* = 4.

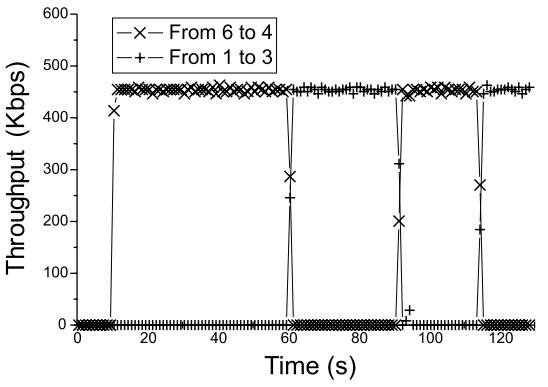


Fig. 15. Throughput of two TCP connections with same hop numbers, *window_* = 4.

serious problem. After repeating this experiment many times with different simulation seed, we find that these two TCP sessions cannot coexist in that a network. Once one session develops, the other one is shut down. The overturn can happen at any time. Fig. 14 presents the throughput for another run with the same experimental settings. In the lifetime of this experiment run, the overturn happens three times. Comparing Figs. 13 and 14, we can easily notice that the overturns happen in different time points. From a great deal of runs for this same simulation, Figs. 13 and 14 showing two of them, we find that the time points of the turnovers are totally random. This makes the problem even worse, since you cannot predict when it may happen. We call this "incompatibility problem".

In Figs. 13 and 14, the TCP sources for the two TCP connections are neighboring nodes, node 4 and 3. Since the size of TCP data packets usually are much larger than the TCP ACK, one might want to know what will happen if the TCP source are not direct neighbor. Fig. 15 illustrates the throughput for one run of such an experiment. In this experiment, the first session is from 6 to 4; the second one is from 1 to 3. The TCP sources are five hops away while the TCP receivers are neighbors. The incompatibility problem can be clearly isolated. In the lifetime of this experiment run, three turnovers occur.

Note the maximum window size (*window_*) of TCP in this experiment is set at 4. Unlike the TCP instability problem, adjusting this parameter cannot eliminate the incompatibility problem. For this purpose, we list another example with *window_* = 1. Fig. 16 illustrates the throughput for one run of this experiment. 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. To simplify the expression, we call this run "W1_2 run" in the following statement. In this experiment, the first session is from 4 to 6; the second one is from 3 to 1. The incompatibility problem can be clearly identified from this figure. In the lifetime of this experiment run, the overturn happens three times. The first turnover happens around 32.0 s; the second one happens around 57.0 s; and the third one happens around 68.0 s.

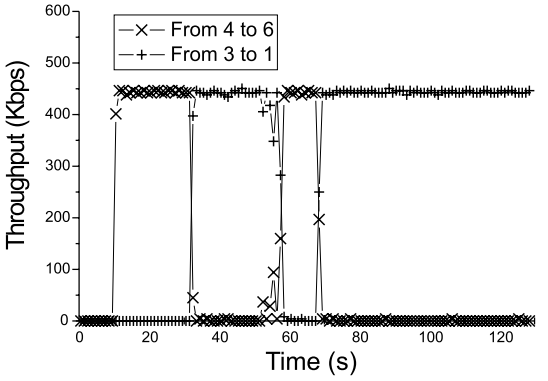


Fig. 16. Throughput of two TCP connections with same hop numbers, *window_* = 1, "W1_2 run".

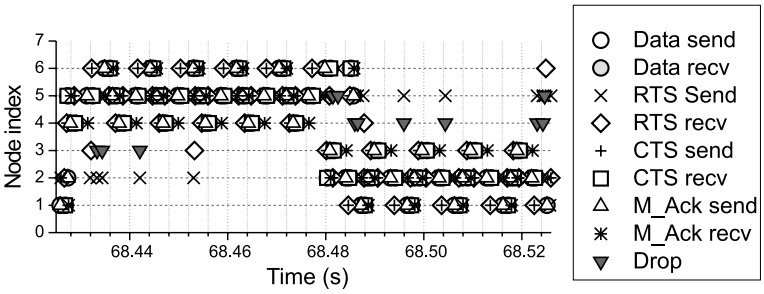


Fig. 17. Part of the MAC layer packet trace in the "W1_2 run", *window_* = 1.

After analyzing the multi-layer traces, we found that the underlying causes are the same as those in the last two sections. The link failure reported by MAC layer causes a route failure. After the route failure is reported back to the source, a route discovery is triggered. Since the TCP layer traces are pretty similar to those shown in Section 5, we do not list them. We will focus on the MAC layer traces to see what happens in the moment of turnover. In Fig. 17, we illustrate the part of the MAC layer packet trace of the third turnover in the "W1_2 run". Fig. 18 is a zoom of Fig. 17.

Let us focus on what happens before the first collision at 68.486 s. After it receives the data packet from node 6 at 68.4856 s and sends back an M_Ack to node 6, node 5 tries to forward the packet to node 4. That is why it sends out an RTS to node 4 at 68.486 s. At this time, node 2 is sending a data packet to node 1. Since node 5 cannot sense the transmission occurring at node 2, it sends out an RTS packet immediately. Unfortunately, this packet experiences a collision at node 4. So, the cause of this collision is the interference from node 2. There are, in total, six MAC packets dropped at node 4 in Fig. 17. Since node 4 is within the interfering range of node 2, a collision occurs at node 4 when node 2 and 5 are sending at the same time—even though node 4 cannot directly communicate with node 2. This is a typical "hidden node problem" of wireless packet networks. Node 2 is the hidden node in this case. It is within the interfering range of the intended destination (node 4) but out of the sensing range of the sender (node 5). Since the nominal communication range is 250 m, which is smaller than the interfering range, node 5 cannot hear the CTS packet from node 3. Thus the virtual carrier sense mechanism cannot work in this case. Furthermore, even if node 4 successfully receives the RTS from node 5, it still cannot send back a CTS when node 2 or node 3 is sending. The third RTS from node 5 is successfully received by node 4 at 68.4878 s. But node 4 cannot send a CTS back to node 5, since it

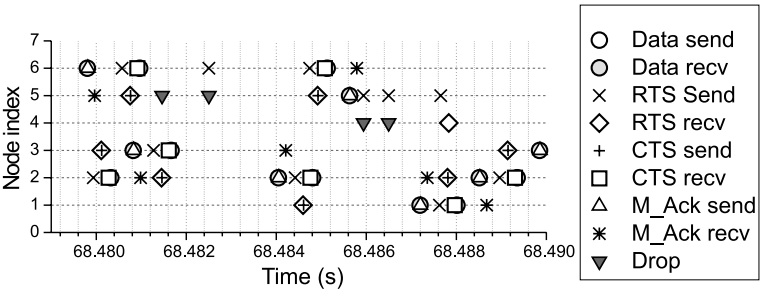


Fig. 18. Zoom of Fig. 17, part of the MAC layer packet trace in the "W1_2 run", *window_* = 1.

can sense node 3 and 2. This is a typical "exposed node problem" in wireless packet networks.

There is a turnover occurring around 68.48 s. Before that moment, the TCP session from node 4 to 6 occupies the channel. After that, it experiences difficulties accessing the channel, since the other TCP session has developed. Due to the link failure, which happens at 68.524 s, this TCP session pauses and has to wait for the route to become available again. Now let us have a look at Fig. 17 again to see what happened before the turnover. From 68.432 to 68.480 s, the session from node 3 to node 1 also suffers from the link access problem. Node 2 fails five times to reach node 1. The reasons are the same—the hidden node and exposed node problems in MAC. It is almost impossible for node 2 to reach node 1 when node 4 or 5 is sending. At 68.480 s, Node 2 tries again. Fortunately, neither node 4 nor 5 are sending. After node 1 receives the packet, it starts to send out the TCP data packet. After that moment, it is the packets from node 6 and 5 that experience collisions. This further causes the route failure in node 5. Then a turnover occurs.

We call this problem the "incompatibility problem". Two simultaneous TCP traffics cannot coexist in the network at the same time. From the above analysis, it is clear that it is rooted in the MAC layer. We believe that the resolution must come from the MAC layer or lower layer. Adjusting TCP parameters cannot solve this problem.

7. Discussion and related works

In the last three sections, we present three problems encountered in TCP sessions in an IEEE 802.11 based multi-hop wireless Network. By illustrating the 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 importantly, 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 the IEEE 802.11 is used in multi-hop networks. TCP traffic clearly shows the problems existing in the MAC. In fact, these problems always appear when the traffic load becomes large enough—even if the traffic is not from TCP.

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

- The hidden node problem still exists in multi-hop 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 multi-hop network.
- There is no scheme in this standard to deal with the exposed node problem, which will be more harmful in a multi-hop 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 the 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 multi-hop case. The larger interfering range makes the hidden node problem worse; the larger sensing range intensifies the exposed node problem.
- The binary exponential back-off scheme always favors the latest successful node. This will cause unfairness, even when this protocol is not used in multi-hop networks, like in the typical wireless LAN defined in the IEEE 802.11 standard.

There might also be another factor which complicates this matter further. That is the word "ad hoc". Note that in the document of standard IEEE 802.11, "an ad hoc architecture" is clearly declared as supported. Since wireless mobile ad hoc networks also use the same word, this makes some sort of confusion, at least at our early stage of related research. People imagine that the IEEE 802.11 MAC protocol should automatically support these kinds of networks, of which multi-hop connectivity is an important feature (if not, why would we need routing?). So, it is better to use the word "multi-hop" explicitly when we refer to wireless mobile multi-hop ad hoc networks.

Recently, several researchers have studied the performance of the MAC layer on multi-hop networks. Gerla et al. [3] and Tang and Gerla [8] investigated the impact of the MAC protocol on the 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 paper. 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 paper since they are not caused by one node capturing the channel. In Ref. [12], we evaluated several prominent TCP algorithms in 802.11-based multihop wireless ad hoc networks. We showed a method to eliminate or relieve the TCP instability problem in such a network. However, that method cannot solve the other two problems we illustrate in this paper.

In several recent published works, a so-called fairness problem in 802.11 MAC protocol is addressed [9–11]. This problem is similar to the third one we mentioned above, which is caused by the back-off scheme in 802.11. These works propose some more fair back-off schemes to replace the one defined in the standard. This surely will help to improve the fairness in wireless LANs. However, they do not address the first two problems we described above. In fact, the inherent hidden node problem and exposed node problem in the 802.11-based multi-hop networks have never been reported before. Since 802.11-based wireless multihop networks are widely used in almost all test beds and simulations in the research of MANET area, it is hard to believe that no one has explicitly indicated the potential existence of the hidden node and exposed node problems. As we have shown in this paper, the effect of these two problems is much more serious than the one which comes from the back-off scheme. So, these proposals cannot eliminate all above-mentioned problems existing in the 802.11 MAC layer when it is used in a wireless multi-hop network.

Besides changing the back-off policy, other potential resolutions for these problems might include adjusting the interfering (and sensing) range and introducing a priority mechanism. Some schemes to deal with the exposed node problem are also needed. Also, the efforts in IEEE 802.11 Wge will help to solve some of the problems.

8. Conclusions

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

By presenting several serious problems encountered in an IEEE 802.11-based multi-hop network, we show that the current TCP/IP stack has serious problems running in such a network. In fact, TCP connections can hardly work in many cases. Moreover, after revealing the underlying causes of them, we conclude that the current version of this wireless LAN protocol does not function well in multi-hop ad hoc networks. We also indicate the specific features of this protocol which cause the upper layer problems when it is

used in a multi-hop network. Based on this analysis, we point out the potential direction to resolve those problems.

So, we doubt that the WaveLAN-based system is workable as a mobile ad hoc test bed, even if they are only used to test the routing protocols. As we have shown in this paper, the MAC layer problem can cause routing protocol failing. And more efforts on the MAC layer are needed to design a usable wireless mobile network. As part of these efforts, we are developing some new features for IEEE 802.11 to support the multi-hop application based on this WLAN standard.

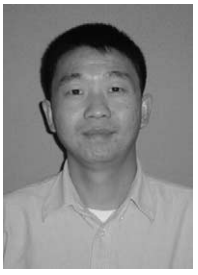
Once again, we emphasize that we use a simple network scenario in this paper to show the possibility of these problems. It is not the only one scenario in which the problems could happen. And probably it is not the most possible scenario for these problems, either. To determine when these problems will happen is beyond the scope of this paper, which need further investigation.

Acknowledgements

We would like to thank the Monarch group at CMU, VINT group at USC and Berkeley for making the NS2 simulator available. The first author thanks Guanhua Ye, Ph.D. student of CCNY, who conducted the OPNET simulations to validate our NS2 results.

References

[1] J.P. Macker, M.S. Corson, Mobile ad hoc networking and the IETF, ACM Mobile Computing and Communications Review 2 (1) (1998).
[2] IEEE, IEEE std 802.11—Wireless LAN media access control (MAC) and physical layer (PHY) specifications, 1997.
[3] M. Gerla, K. Tang, R. Bagrodia, TCP Performance in wireless multi-hop networks, IEEE WMCSA'99, New Orleans, LA, February 1999.
[4] K. Fall, K. Varadhan, ns Notes and Documentation, LBNL, August 1998, http://www-mash.cs.berkeley.edu/ns/.
[5] J. Broch, D.A. Maltz, D.B. Johnson, Y. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, ACM/IEEE Int. Conf. on Mobile Computing and Networking, October 1998, pp. 85–97.
[6] C. Barakat, E. Altman, W. Dabbous, On TCP performance in a hetergeneous network: a survey, IEEE Communications Magazine, January 2000.
[7] J.L. Sobrinho, A.S. Krishnakumar, Quality-of-service in ad hoc carrier sense multiple access wireless networks, IEEE JSAC 17 (8) (1999) 1353–1368.
[8] K. Tang, M. Gerla, Fair sharing of MAC under TCP in wireless ad hoc networks, Proceedings of IEEE MMT'99, Venice, Italy, October 1999.
[9] B. Bensaou, Y. Wang, C.C. Ko, Fair media access in 802.11 based wireless ad-hoc networks, Proceedings of Mobihoc'2000, Boston, USA, August 2000.
[10] H. Luo, S. Lu, V. Bharghavan, A new model for packet scheduling in multihop wireless networks, Proceedings of Mobcom'2000, Boston, USA, August 2000.
[11] T. Nandagopal, T. Kim, X. Gao, V. Bharghavan, Achieving MAC layer fairness in wireless packet networks, Proceedings of Mobcom'2000, Boston, USA, August 2000.
[12] S. Xu, T. Saadawi, Performance evaluation of TCP algorithms in multi-hop wireless packet networks, Journal of Wireless Communications and Mobile Computing 2 (1) (2002).



Shugong Xu (xu001@ieee.org) received a B.Sc degree from Wuhan University, China, and M.E. and Ph.D. degrees 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. Before he joined Sharp Labs of America in August 2001, he worked with CCNY as 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 thirty technical papers in these areas.

Tarek N. Saadawi received the B.Sc. and the M.Sc. from Cairo University Egypt and the Ph.D. from the University of Maryland, College Park (all in Electrical Engineering). Since 1980 he has been with the Electrical Engineering Department, The 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'', John Wiley & Sons, 1994. He is also the lead author of Egypt Telecommunications Infrastructure Master Plan covering the fiber network, IP/ATM, DSL and the wireless local loop. Dr. Saadawi is a Former Chairman of IEEE Computer Society of New York City (1986–1987). He has received IEEE Region 1 Award, 1987, and the Nippon Telegraph and Telephone (NTT) of America for research on Broadband Telecommunication Networks.