Re-routing Instability in IEEE 802.11 Multi-hop Ad-hoc Networks*

Ping Chung Ng and Soung Chang Liew
Department of Information Engineering
The Chinese University of Hong Kong
{pcng3, soung}@ie.cuhk.edu.hk

Abstract

TCP throughput instability is a well-known phenomenon in IEEE 802.11 multi-hop ad-hoc networks. However, we find that this problem is not restricted to TCP traffic only, but also occurs in UDP traffic. The associated throughput oscillations are not acceptable for real-time applications such as video conferencing and voice over IP. This paper re-defines this throughput fluctuation as a "re-routing instability problem" since it is caused by the triggering of the re-routing function. In particular, we show that the throughput instability is mainly induced by re-routing, not the binary exponential back-off of the IEEE 802.11 MAC protocol. Turning off the re-routing function, for example, eliminates the problem. We believe that this is the first paper in the literature to study this phenomenon in the context of re-routing instability. We propose to modify the ad-hoc routing protocols with a "don't-break-before-you-can-make" strategy. The scheme does not require modifications of the IEEE 802.11 standard, making it readily deployable using existing commercial Wireless LAN (WLAN) products. Simulations show that the proposed scheme can significantly reduce the throughput variation in a traffic flow by 50-70% and improve the average throughput by up to 11%.

1. Introduction

The performance of wireless ad-hoc networks based on IEEE 802.11 has been extensively studied. Much of the previous work attempts to solve the one-hop performance problems [1][2]. In the multi-hop scenario, most of investigations focused on TCP performance [3][4]. Besides traditional TCP applications like file transfer and e-mail, the demands for real-time applications like multi-media streaming and voice services are also increasing. These real-time services are usually transported on UDP rather than TCP. In this paper, we investigate a common phenomenon that leads to throughput degradations and oscillations for both TCP and UDP traffic in multi-hop networks: the re-routing instability problem.

Previous studies [5][6][7] showed that the so-called "TCP instability problem" exists in a multi-hop flow. References [5][6] provided a solution to solve TCP instability by limiting the traffic at the transport layer. The solution assumes TCP Vegas and limits the TCP window size to at most 4. This limit bounds the number of packets in the path to prevent individual nodes from capturing the channel for a sustained period of time. Two observations are as follows. First, it is not clear that the solution is effective when there are multiple TCP flows along the same path, or when TCP flows on adjacent paths may interfere with the flow. Second, perhaps more importantly, the instability problem is caused by false declaration of link failures which is rooted at the link layer. In other words, this problem is not a phenomenon for TCP traffic only, but also for other types of traffic. The declaration of link failures in turn triggers the re-routing function, which exacerbates the situation. We believe that the problem should be properly defined as a "re-routing instability problem", and a more general approach should be used to solve the problem by eliminating its root cause directly.

Reference [7] reconfirmed the TCP throughput instability and proposed a modification of the IEEE 802.11 back-off algorithm such that only two back-off window sizes could be used. The main idea is to adopt the larger window for the next packet after a successful transmission. This allows other nodes using the smaller window to transmit with less chance of collisions. However, the decision for the choice of the value of these two back-off window sizes is based on the assumption that the packet payload is fixed at 1460bytes. We believe this assumption is not valid in real wireless LAN networks. When packets could be of different size, this scheme may fail to work properly.

The rest of this paper is organized as follows. Section 2 gives details of the simulation set-up in this paper. In Section 3, we review the throughput instability problem. Section 4 introduces the existing ad-hoc routing protocols and describes how they handle link-layer failures. In Section 5, we suggest a solution to deal with re-routing instability and show how our solution can be applied to the AODV routing protocol to eliminate instability.

* This work was sponsored by the Areas of Excellence scheme established under the University Grant Committee of the Hong Kong Special Administrative Region, China (Project Number AoE/E-01/99).

Section 6 provides simulation results quantifying the improvements that can be obtained by our proposed scheme. Section 7 analyzes factors that cause the triggering of re-routing. Finally, in Section 8, we investigate the link-layer penalty in a scenario with multiple flows interfering with each other.

2. Simulation Set-up

The simulations in this paper were conducted using NS2 [8]. All nodes communicate using identical, half-duplex wireless radio based on the IEEE 802.11 Distributed Coordination Function (DCF), with data and basic rates set at 11Mbps. The RTS/CTS mechanism is turned off. Nodes are stationary. The transmission range is 250m, the carrier-sensing range is 550m, and the capture threshold, *CPThreshold*, is set to 10dB. Each node has a drop-tail FIFO queue which holds up to 500 packets. This large link-layer buffer size eliminates the chance of throughput oscillations caused by packet losses due to buffer overflow. The Ad-hoc On-Demand Distance Vector (AODV) routing protocol and the two-ray propagation model are used. Unless otherwise indicated, all traffic streams use fixed packet size of 1460bytes. The TCP Reno algorithm is used since it is the most widely deployed TCP version. The advertised window (window_) of TCP is set to a large value to prevent the TCP traffic from being limited by the receiver. The throughputs plotted in this paper are obtained by averaging over one-second intervals.

3. Re-routing Instability

In this section, we use a 7-node string multi-hop network as an example to illustrate the "re-routing instability" problem. In Fig. 1, node 1 sends a UDP or TCP traffic stream to node 7. For UDP, the traffic is generated at node 1 in a saturated manner, in which as soon as a packet is transmitted to node 2, another is waiting in line. The traffic at later nodes all originates from node 1 and the later nodes are not saturated. For TCP, the traffic injected into the network by node 1 is restricted by the congestion control algorithm of the TCP Reno.

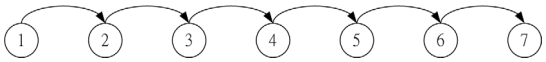


Figure 1. UDP/TCP traffic flow with node 1 as the source and node 7 as the destination in a 7-node multi-hop network

Figure 2 shows that the UDP throughput tends to oscillate widely over time. The throughput oscillations are caused by triggering of the re-routing function. In the multi-hop path, nodes 1 and 2 sense fewer interfering stations than later nodes. As a result, they pump more traffic into the network than can be supported. This results in high contention rates at later nodes.

When one of the later nodes fails to transmit a packet after a number of retries, it declares the link as being broken. The routing agent is then invoked to look for a new route. Before a new route is discovered, no packet can be transmitted, and this causes the throughput to drop drastically.

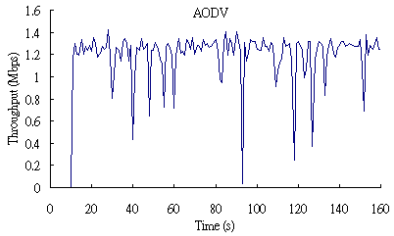


Figure 2. UDP end-to-end throughput in a 7-node flow using the routing agent in the original AODV

In this string network topology, there is only one route from node 1 to node 7, so the routing agent will eventually "re-discover" the same route again. The breaking and rediscovery of the path results in the drastic throughput oscillations observed. For a general network with multiple possible paths from source to destination, the same throughput oscillations will still be expected. This is because the declaration of the link failure is caused by self-interference of traffic of the same flow.

3.1 Hidden-Terminal Problem

The hidden-terminal problem can increase the chance of link-failure declarations significantly. Consider the illustration in Fig. 3. When node 6 sends a packet to node 7, node 4 senses the channel to be busy while node 3 senses the channel to be idle, since node 6 is inside the carrier-sensing range of node 4 but outside that of node 3. Once node 3 senses the channel as idle, it may count down its back-off contention window until zero and transmit a packet to node 4.

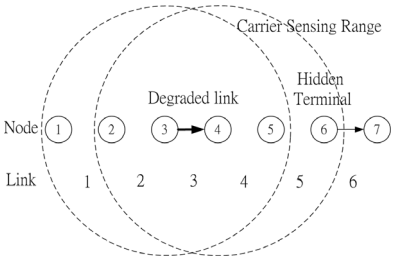


Figure 3. Node 6 as a hidden terminal to node 3

If the transmission from node 6 is still in progress, node 4 will continue to sense the channel as busy, and it will not receive the packet from node 3. As a result, node 4 will not return an ACK to node 3. Node 3 may then time out and double the contention window size for retransmission later.

Meanwhile, node 6 transmits the packet successfully and is not aware of the interference at node 4. When transmitting the next packet, node 6 will use the minimum contention window size. The hidden-terminal scenario favors node 6, and the chance of collision at node 4 can not be reduced even though node 3 backs off for a duration of time before the next retry. The hidden-terminal problem increases the chance of multiple retries by node 3.

Note that the negative effect of a hidden terminal is much more than that of a contending terminal within the carrier-sensing range. This is because the carrier-sensing capability in the CSMA protocol is lost whenever there is a hidden terminal. The lack of carrier sensing with respect to the hidden-terminal causes the MAC protocol to behave much like an Aloha protocol.

3.2 Ineffectiveness of Solving Hidden-Terminal Problem with RTS/CTS

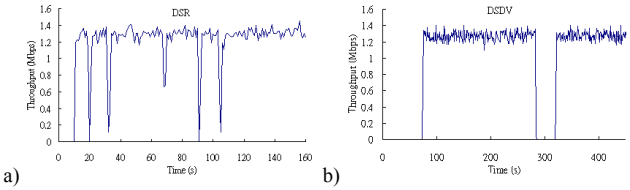
The RTS/CTS mechanism in IEEE 802.11 is designed to solve the hidden terminal problem. However, using RTS/CTS in multi-hop networks does not eliminate the hidden terminal problem. The effectiveness of the RTS/CTS mechanism is based on the assumption that transmissions by mutually hidden terminals are to a common receiver, and this common receiver may forewarn the other terminals while the transmission of a hidden terminal is in progress. This assumption may not hold in a multi-hop network.

Consider the scenario in Fig. 3 again. The RTS transmitted by node 6 will cause a CTS to be returned by node 7. However, this CTS cannot be received by node 3. Therefore, node 3 may still transmit a packet to node 4 while the transmission of node 6 is in progress. The hidden-terminal effect as described in the previous subsection cannot be eliminated. For more details, the interested reader is referred to [9], in which it was argued that when the carrier-sensing range is larger than two times of the transmission range, RTS/CTS is no longer needed. In this paper, we assume the use of the basic access mode without RTS/CTS.

4. Ad-hoc routing protocols

Strictly speaking, in the scenario in Section 3, the link has not failed, although it is congested and the attempt to look for a new path is definitely warranted. However, before a new route can be discovered, one should continue to use the old route. That is, a "don't-break-before-you-can- make" strategy should be adopted.

Numerous ad-hoc routing protocols have been proposed in the literature. They can be categorized into two approaches: 1) proactive / table-driven; or 2) reactive / on-demand-driven [10]. The proactive approach protocols (e.g., Destination Sequenced Distance Vector (DSDV)), attempt to preserve consistent and up-to-date routing information from each node to every other node in the entire network. Each node maintains its own routing table and propagates route updates throughout the network to notify other nodes of changes in the network topology. In reactive approach protocols (e.g. Ad-hoc On-demand Distance Vector (AODV) and Dynamic Source Routing (DSR)), route discoveries are initiated only when desired by the source nodes. A node keeps using the created route until that route becomes inaccessible or the route is no longer needed.



a) b)

Figure 4. UDP end-to-end throughput in a 7-node flow using a) DSR and b) DSDV

The "re-routing instability problem" is a common performance problem suffered by various ad-hoc routing protocols. Figures 2 and 4 show that AODV, DSR (reactive) and DSDV (proactive) all experience throughput oscillations. Although the severity of the oscillations may vary, they are caused by the same reason, the triggering of the re-routing function. These routing protocols treat the link-failure notification as an indication of the loss of the link to next hop. In IEEE 802.11, this link-failure notification can be induced by the hidden-terminal problem as well as the real-break case. Obviously, simply discarding the route after receiving a link-failure notification is not appropriate for IEEE 802.11 multi-hop networks.

5. Proposed scheme

A possible solution is to modify the routing algorithm so that the routing agent continues to use the previous route for transmissions before a new route can be found. In practice, this means computers equipped with wireless LAN devices only need to install slightly modified routing agent software. In this paper, we choose the AODV routing protocol for implementation of this strategy, mainly because details of AODV have been published in an IETF RFC [11]. There is no reason why this approach can not be applied in other ad-hoc routing protocols.

5.1 Original AODV

We quote the following excerpt from the IETF RCF 3561 on AODV [11]: "Any suitable link layer notification, such as those provided by IEEE 802.11, can be used to determine connectivity, each time a packet is transmitted to an active next hop. For example, absence of a link layer ACK or failure to get a CTS after sending RTS, even after the maximum number of retransmission attempts, indicates loss of the link to this active next hop."

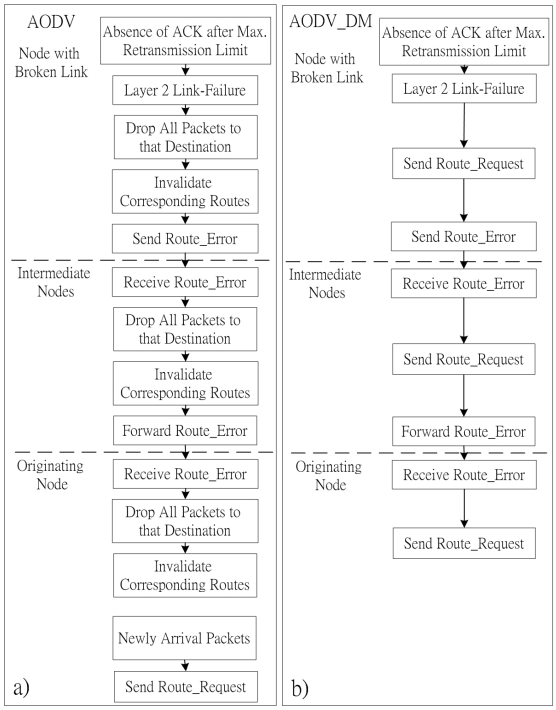


Figure 5. Procedures in handling link-failure in a) original AODV and b) our proposed scheme (AODV_DM)

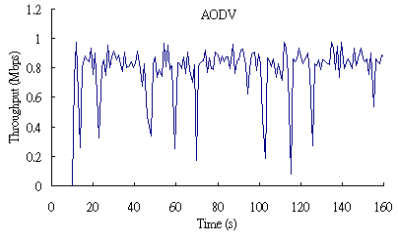


Figure 6. TCP end-to-end throughput in a 7-node flow using original AODV

Figure 5a shows the procedures for handling link-failure in the original AODV. When a node fails to receive the link-layer ACK from the next hop after the retransmission limit, its link layer reports the link failure to the routing agent. The AODV protocol then generates a list of unreachable destinations that use the unreachable neighbor as the next hop. It drops all packets destined to that hop and invalidates the corresponding routes in its routing table. Then the node with the broken link propagates the route error (RERR) message to its upstream neighbors until the source node is reached. When the source and intermediate nodes receive the RERR message, they also drop all packets that utilize the broken route for forwarding and are destined to the nodes in the unreachable destination list attached with the RERR message. The nodes then remove the corresponding routes form their routing tables. After that, a newly arrival packet targeted for these unreachable destinations will trigger the route discovery process, and the transmissions of packets to that destination will be resumed after the new route is generated.

5.2 AODV with Proposed Scheme

In our proposed solution as shown in Fig. 5b, the link layer notifies the routing agent of the "link failure" after the maximum retransmission attempts. The AODV routing agent then broadcasts a route request (RREQ) message immediately. Unlike the original AODV, our routing agent does not drop packets and invalidate the corresponding routes. However, it continues to propagate the RERR message to its upstream neighbors. When an intermediate node receives the RERR message, it broadcasts another RREQ message and forwards the RERR message to upstream nodes until the source node is reached. During this process, no packets will be dropped and all nodes continue to use the previous routes. After sending RREQ messages, the nodes wait for the route reply (RREP) message returned by the destination node or an intermediate node with an up-to-date route (i.e., the destination sequence number stored in the node's routing table is greater than that in the RREQ message [11]). After a new route is created, all nodes discard the previous route and switch to the new one for transmissions.

In the following sections, we will show simulation results of AODV modified with "don't-break-before-you-can- make" strategy (AODV_DM) in two scenarios: 1) a single flow in a single chain of nodes; and 2) a real-break case.

5.2.1 A Single Flow in a Single Chain of Nodes

Figures 2 and 6 show the existence of "re-routing instability" of UDP and TCP traffic in a 7-node chain using the original AODV. As shown in Fig. 7, the AODV_DM scheme eliminates these oscillations. With the AODV_DM scheme, no packets are dropped and nodes continue to use the old route, while the new route discovery process is ongoing. For our scenario of a single-chain network, when the node with the broken link receives the responded RREP message or the Hello message broadcasted periodically by the next hop, it notices that the next hop is still active and the routing agent will re-discover the same route for transmissions.

5.2.2 Real-break Case

Figure 8 shows a scenario with two alternative routes from node 1 to node 7. Both of them are accessible in the first 70 seconds. At the 70th second, node 4 is switched off and this breaks the upper route. Figures 9 and 10 show the simulation results. In the first 70 seconds, both the original AODV and AODV_DM choose the upper route since this path requires fewer number of hops. After the 70th second, they switch to the lower route for

transmissions. Since the number of hops in the lower route is more than that of the upper route, the average throughputs are slightly reduced. Our proposed scheme keeps the route discovery property of original AODV and switch to a new route if the existing one is broken. At the same time, AODV_DM eliminates the "re-routing instability problem" experienced by the original AODV.

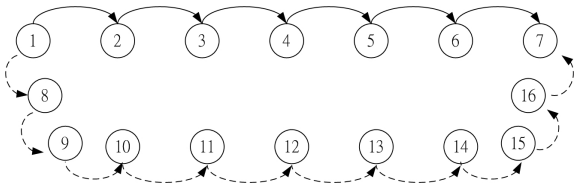
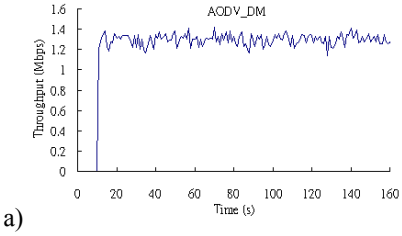
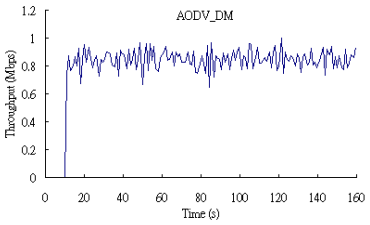


Figure 8. Two alternative routes for UDP/TCP traffic flow with node 1 as the source and node 7 as the destination in a multi-hop network

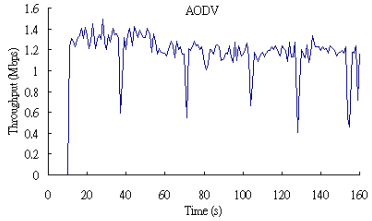


a)

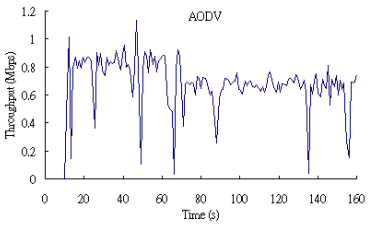


b)

Figure 7. a) UDP and b) TCP end-to-end throughput in a 7-node flow using AODV_DM

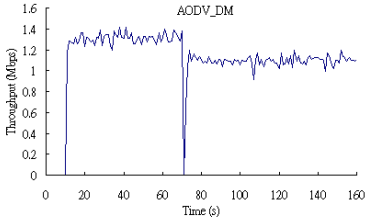


a)

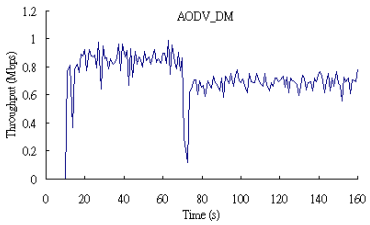


b)

Figure 9. a) UDP and b) TCP end-to-end throughput in a real-break case using original AODV

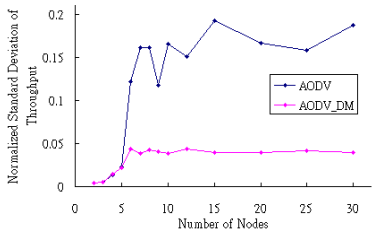


a)

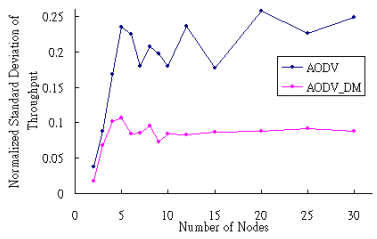


b)

Figure 10. a) UDP and b) TCP end-to-end throughput in a real-break case using AODV_DM



a)



b)

Figure 11. Normalized standard deviation of a) UDP and b) TCP end-to-end throughput versus the number of nodes in a string multi-hop network

6. Improvements

Simulations show that whenever re-routing occurs, the throughput drops severely for the duration of 1 to 3 seconds. For real-time applications like video conferencing or voice over IP (VoIP), this may not be acceptable. Compared with the original AODV, our proposed solution reduces the throughput variations by 70% for UDP and 50% for TCP as shown in Fig. 11. Also, from Table II, the minimum throughputs of the original AODV are near zero when there are more than five nodes in the UDP flow; and when there are more than three nodes in the TCP flow. Using AODV_DM, the minimum throughputs are only slightly less than the average values. As shown in Fig. 12, another improvement of our proposed scheme is to boost the average throughput up to 11% for both TCP and UDP in a long chain of nodes (i.e., more than 12 nodes).

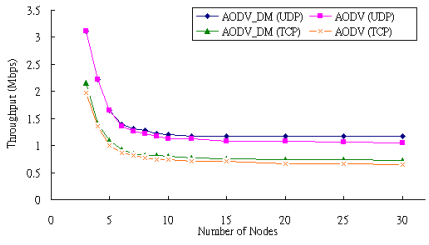


Figure 12. UDP and TCP end-to-end throughput versus number of nodes in a string topology

Table II. a) UDP and b) TCP throughput result (Mbps) with various number of nodes in a string multi-hop network using AODV and AODV_DM in a 500-second simulation run

a)

Num. of Nodes	AODV			AODV_DM		
	Mean	Max	Min	Mean	Max	Min
2	6.304	6.389	6.237	6.303	6.366	6.225
3	3.120	3.165	3.084	3.118	3.154	3.084
4	2.213	2.301	2.114	2.213	2.336	2.102
5	1.646	1.775	1.565	1.646	1.764	1.553
6	1.354	1.542	***0.350***	1.391	1.530	1.226
8	1.211	1.448	***0.245***	1.276	1.448	1.110
10	1.131	1.320	***0.199***	1.197	1.320	1.040
15	1.074	1.261	***0.070***	1.170	1.332	1.016
20	1.080	1.261	***0.070***	1.166	1.285	0.958
30	1.049	1.238	***0.093***	1.171	1.296	0.993

b)

Num. of Nodes	AODV			AODV_DM		
	Mean	Max	Min	Mean	Max	Min
2	4.231	4.659	3.746	4.341	4.560	4.117
3	1.969	2.405	1.521	2.155	2.571	1.758
4	1.359	1.946	***0.194***	1.403	1.994	0.999
5	1.002	1.457	***0.000***	1.087	1.525	0.712
6	0.867	1.101	***0.000***	0.933	1.151	0.652
8	0.766	1.098	***0.000***	0.819	1.030	0.486
10	0.742	1.029	***0.000***	0.799	1.012	0.578
15	0.710	0.976	***0.025***	0.762	0.968	0.544
20	0.671	0.952	***0.000***	0.742	0.931	0.539
30	0.649	0.811	***0.000***	0.720	0.989	0.534

7. Impacts of Data Transmission Rate and Payload Size

This section shows the effects of the data transmission rate and payload size on the re-routing instability problem. We first show the condition for the occurrence of hidden-terminal collisions. Then we introduce a quantitative approach to analyze the impact of various data transmission rates and payload sizes.

7.1 Signal Capture

Consider Fig. 3 again, both nodes 3 and 6 have a packet to transmit. This may cause the aforementioned hidden-terminal collision. However, the signal capturing property may still allow a packet from node 3 to be received successfully, provided it transmits before node 6.

More specifically, suppose that node 3 transmits first and the signal power of the transmission received at node 4 is P_3. Node 6 then transmits a packet with power P_6 received at node 4. If $P_3 > P_6 + CPThreshold$, where *CPThreshold* is the capture threshold, then no collision occurs, and node 4 can still receive the packet from node 3 successfully.

On the other hand, if node 6 transmits first, node 4 senses the signal from node 6 and declares the channel to be busy. In that case, a newly arriving packet from node 3 can not be received even if $P_3 > P_6 + CPThreshold$. Effectively, the packet from node 3 to node 4 experiences a collision.

In our simulation, *CPThreshold* is set to be 10dB. Let d be the fixed distance between nodes. In this scenario, node 3 and node 6 are separated by a distance larger than the carrier sensing range. Thus, node 3 and node 6 can send packets at the same time. From [12], in a two ray propagation model, the signal-to-noise ratio at node 4 is

$$SNR = P_3 / P_6 = (2d/d)^4 = 2^4 = 16 > CPThreshold \quad (1)$$

This means that the power level of the packet transmitted by node 3 and received at node 4 is always more than *CPThreshold* higher than the power level of the received signal from node 6.

7.2 Vulnerable region

In the analysis of the effect of the hidden-terminal problem, the key is to identify the vulnerable region during which if the node transmits, it may collide with the transmission of a hidden node. This is illustrated in Fig. 13. Note that a hidden-node collision only occurs if the transmissions of nodes 3 and 6 overlap and that the transmission of node 6 precedes that of node 3. Let $PACKET_i$ be the time to transmit packet *i*.

$$PACKET_i = PHY + (MAC + Payload)/TxRate \quad (2)$$

where *PHY* is the time to transmit the physical header, *MAC* is the size of the MAC header, *Payload* is the size of the packet payload, and *TxRate* is the data transmission rate. Let T_i be the time of the transmission cycle of packet i at node 6. As illustrated in Fig. 16, T_i includes the back-off period, the packet transmission time, the idle period, I_i ,when node 6 does not have a packet to transmit, and the busy periods used by other nodes within its carrier sensing range for their transmissions, B_i. We have

$$T_i = I_i + DIFS + W_{avg} + PACKET_i + SIFS + ACK + B_i \quad (3)$$

Let ρ be the fraction of the time corresponding to the vulnerable region induced by node 6. We have

$$\rho = \lim_{K\to\infty} \frac{\sum_{i=1}^{K} PACKET_i}{\sum_{i=1}^{K} T_i} \quad (4)$$

where *ACK* is the transmission time for an acknowledgement, *SIFS* is the time duration of short

interframe space, *DIFS* is the time duration of distributed interframe space, and W_{avg} is the average contention window size. Thus, ρ varies with different data transmission rates and payload sizes. With lower data transmission rate or larger payload size, the fraction of the time that belongs to vulnerable region in each transmission cycle becomes larger. As a result, a higher chance of hidden-terminal collisions is expected. In other words, the link-failure re-routing occurs more frequently which further deteriorates the instability problem. As shown in Fig. 14 and 15, using lower data transmission rate or larger payload size increases the number of severe drops of throughputs.

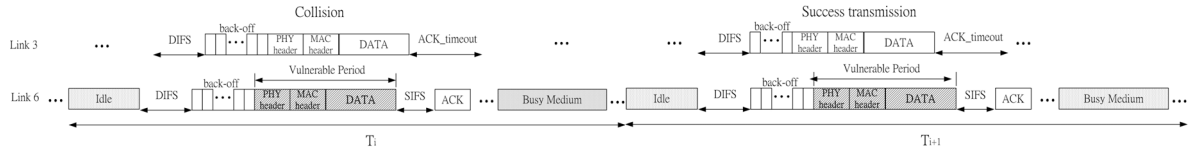


Figure 13. Collision occurs when the transmission of node 3 begins inside the vulnerable period

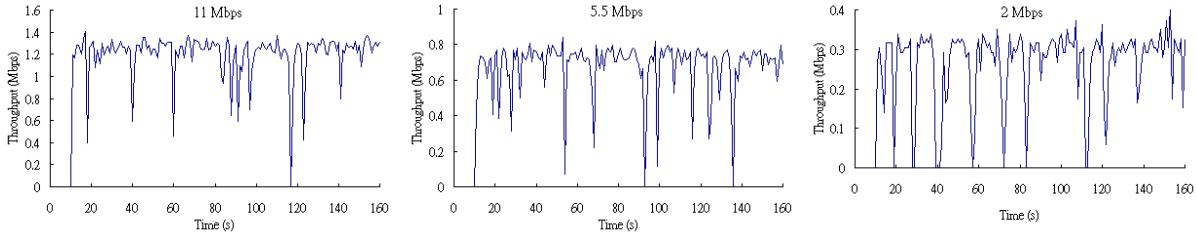


Figure 14. UDP end-to-end throughput in a 7-node flow using original AODV with various data transmission rates

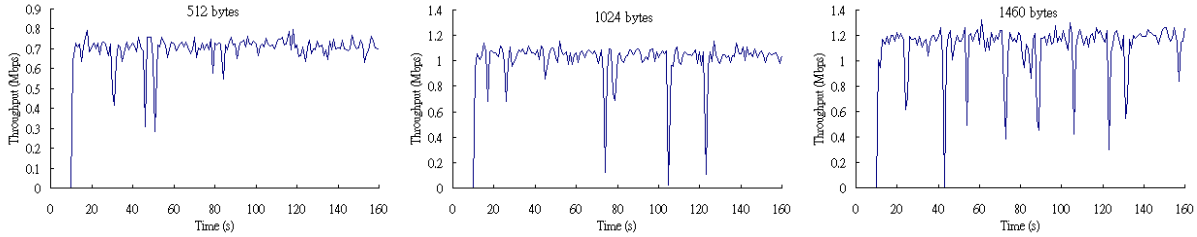


Figure 15. UDP end-to-end throughput in a 7-node flow using original AODV with various payload sizes

8. Performance Enhancements in Multiple Flows

In previous sections, we have focused on the performance degradations induced by self interference of the traffic of a single-flow in a single chain of nodes. In this section, we consider the interferences between multiple flows. We show that our proposed scheme can sufficiently increase the average throughput of a flow suffering from the hidden terminal problem.

Figure 16. Two 1-hop saturated UDP flows

Figure 16 shows a scenario with five nodes and two 1-hop saturated UDP traffic flows. As mentioned in Section 3.1, the transmissions of flow 1 may collide with the transmissions of flow 2 at node 2 due to the hidden-terminal problem. This severely deteriorates the throughput of flow 1, while flow 2 continues to achieve a much higher throughput as demonstrated in Fig. 17a. In addition, the throughput of flow 1 drops to zero from 70th to 110th second due to the successive collisions of RREQ sent out by node 1 with the transmissions of flow 2. Node 4 does not notice that node 2 is suffering from hidden-terminal collisions and attempts to transmit at the maximum sustainable rate. Once the link at node 1 is declared as failure, node 1 sends out RREQ and waits for RREP. However, this RREQ message easily collides with the aggressive transmissions of flow 2. In this way, no RREP is responded by node 2, and node 1 times out and retransmit another RREQ. No packet can be transmitted for a long period of time after a number of failed RREQ transmissions.

Previous work in the literature [13] also reported that the throughput can degrade severely in similar scenarios. They attribute this degradation to the binary exponential back-off for retransmissions caused by hidden nodes. However, we believe it is only part of the cause. Once a node fails to receive the link-layer ACK after the retry

limit, it triggers the re-routing function of the routing agent. Before a new route or the previous route is discovered, no packets can be transmitted. This "re-routing instability problem" and the "binary exponential back-off" should be treated and solved separately.

Our proposed scheme addresses the first issue. The average throughput of flow 1 is doubled as show in Fig. 17b. The "binary exponential back-off" does degrade the throughput, resulting in average throughput of flow 1 slightly less than that of flow 2. However, its influence is much smaller than that of "re-routing instability problem". To limit the scope of this paper, we refer interested readers to [13], in which MAC layer solutions were proposed to address the degradations caused binary exponential back-off.

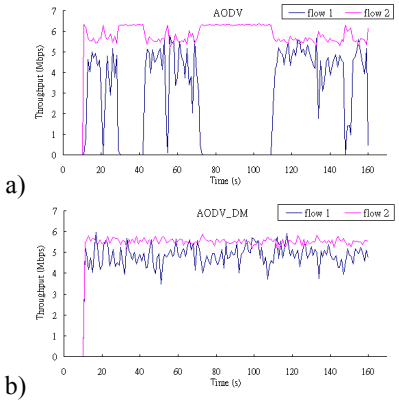


Figure 17. UDP throughputs of two 1-hop flows using a) original AODV and b) AODV_DM

9. Conclusion

This paper is an attempt to solve a throughput instability problem in IEEE 802.11 multi-hop ad-hoc networks. Existing ad-hoc routing protocols simply inherit the method for link-failure handling from the routing protocols used in wired networks, and treat the link-failure notification as an indication of the loss of the link to the next hop. This is not appropriate for wireless networks with hidden-terminal problems such as IEEE 802.11. The triggering of the re-routing function may be induced by consecutive hidden-terminal collisions rather than real link failures.

This paper has four major contributions. **First**, we have argued that the throughput instability problem should properly be re-defined as a "re-routing instability problem", since it is caused by the triggering of the re-routing function and is not specific to TCP traffic alone.

Second, we have proposed to adopt a "don't-break-before-you-can-make" modification to the existing ad-hoc routing protocols. In this strategy, the old route will continue to be used until a new one can be established. We have implemented this scheme with AODV as an example, and have shown that the instability problem can be eliminated. The modified routing agent can still switch to a new route successfully in a real-break case.

Third, we have analyzed the hidden-terminal problem by considering the "vulnerable regions: the time windows during which transmissions may collide with transmission of hidden node". We have established the impact of data transmission rate and payload size on the severity of hidden-node collisions. In particular, we have shown that lower data transmission rates and/or larger payload sizes will incur more frequent throughput oscillations.

Finally, this paper has also investigated a multiple-flow scenario. The throughput degradation induced by "re-routing instability" is much larger than that induced by "binary exponential back-off", as has been demonstrated by the restoration of UDP throughput when our "don't-break- before-you-can-make' ad-hoc routing protocol is used. We believe that this is the first paper in the literature to report this phenomenon.

References

[1] B. Bensaou, Y. Wang, C. C. Ko, "Fair media access in 802.11 based wireless ad-hoc networks", *ACM MobiHoc'00*, pp. 99 – 106, 2000.

[2] K. Brown, S. Singh, "M-TCP: TCP for mobile cellular networks", *ACM Computer Communication Review*, 27(5), Oct. 1997.

[3] G. Holland, N. Vaidya, "Analysisi of TCP Performance over Mobile Ad Hoc Networks", *ACM MobiCom'02*, pp.219-230, Seatle, USA, 2002

[4] R. Jiang, V. Gupta, C. V. Ravishankar, "Interactions between TCP and the IEEE 802.11 MAC protocol", *IEEE DISCEX'03*, Vol. 1, pp.273 – 282, April 2003.

[5] S. Xu, T. Saadawi, "On TCP over Wireless Multi-hop Networks", *IEEE MILCOM'01*, Vol.1, pp.282-288, Oct. 2001.

[6] S. Xu, T. Saadawi, "Revealing and solving the TCP instability problem in 802.11 based multi-hop mobile ad hoc networks", *IEEE VTC'01 Fall*, Vol. 1, pp.257-261, 2001

[7] Kanth K., Ansari S., Melikri M.H., "Performance enhancement of TCP on multihop ad hoc wireless networks", *IEEE ICPWC'02*, pp.90-94, Dec. 2002.

[8]"The Network Simulator–ns2", http://www.isi.edu/nsnam/ns.

[9] K. Xu, M. Gerla, S. Bae, "How Effective is the IEEE 802.11 RTS/CTS Handshake in Ad Hoc Networks?", *IEEE GLOBECOM '02*, Vol. 1 , pp.17-21, Nov. 2002.

[10] C. K. Toh, "Ad hoc mobile wireless networks: protocols and systems", Prentice Hall, New Jersey, 2002.

[11] "IETF RFC 3561 AODV Routing", http://www.ietf.org/rfc/rfc3561.txt

[12] T. Rappaport, "Wireless Communications: Principles and Practice", Prentice Hall, New Jersey, 2002.

[13] X. Huang, B. Bensaou, "On Max-min Fairness and Scheduling in Wireless Ad-Hoc Networks: Analytical Framework and Implementation", *ACM MobiHoc'02*, Long Beach, USA, Oct. 2001.