# Snoop Behaviour in Multihop Wireless Networks

Prasad Nambiar
School of Computer Science
University of Hertfordshire
Hatfield, Hertfordshire
AL10 9AB, UK
P.Nambiar@herts.ac.uk

Hannan Xiao
School of Computer Science
University of Hertfordshire
Hatfield, Hertfordshire
AL10 9AB, UK
H.Xiao@herts.ac.uk

James A Malcolm
School of Computer Science
University of Hertfordshire
Hatfield, Hertfordshire
AL10 9AB, UK
J.A.Malcolm@herts.ac.uk

## ABSTRACT

The Snoop protocol is one proposal for improving TCP throughput in wireless networks. We investigated the application of this protocol in wireless ad hoc networks and observed that a single hop in the ad hoc network experienced large variations in round trip time in a very short period. Without changes to the Snoop protocol to accommodate these dramatic RTT variations, Snoop was performing badly compared with regular TCP even when there were no packet losses or errors. The main cause for this is premature retransmissions performed by Snoop. We have modified the Snoop protocol to avoid these unnecessary retransmissions by having a higher local retransmission timeout. The results show us that Snoop benefits from this approach which has made a significant performance improvement over regular TCP in multihop wireless networks.

## Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication—*TCP Performance*

## General Terms

Performance measurement

## Keywords

TCP performance; Snoop protocol; Ad hoc network

## 1. INTRODUCTION

The performance of TCP (Transmission Control Protocol) [1] over a wireless link has been a research topic for a long time. Many different proposals have been put forward in relation to improving the TCP performance [2] and the Snoop protocol is one among the proposals for improving its performance over a wireless link. This protocol [3] was originally proposed by Balakrishnan *et al.* in 1995. Over a wired-cum-wireless link such as illustrated in Figure 1 the overall TCP throughput may be affected by the erroneous wireless link mainly because a packet drop occurring over the wireless medium may be seen as congestion by the TCP sender. But a packet drop may happen on the wireless medium for various reasons like collision, mobility, channel errors, etc. [4] Assuming that the packet drop is due to congestion, the sender may invoke the congestion control mechanism which is designed to prevent the source from overloading the network. Traditional TCP congestion control involves slow start, congestion avoidance, fast retransmit and fast recovery [5] and basically reduces the overall throughput. With the Snoop protocol the base station would cache packets locally enabling it to retransmit the packets if they had not reached the destination. The protocol suggests validations for duplicate acknowledgments and timeouts before performing retransmission to prevent overloading the wireless network with retransmissions.

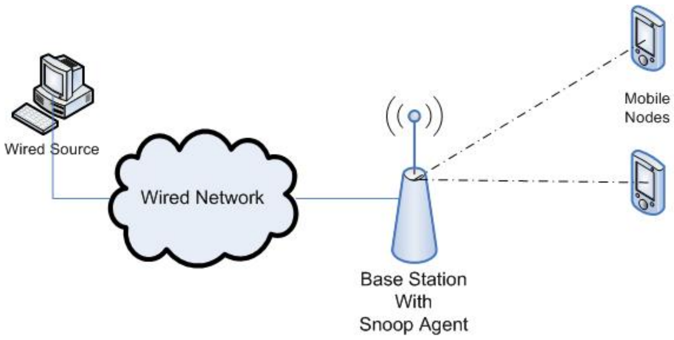
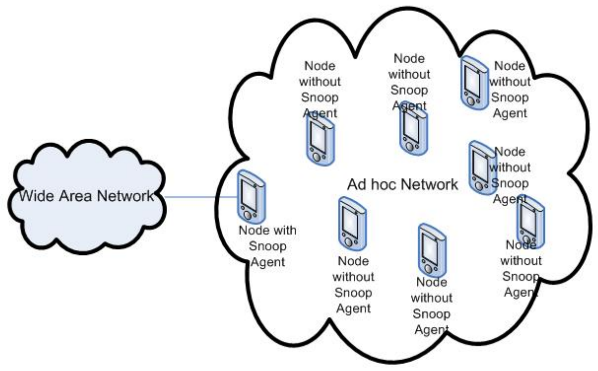


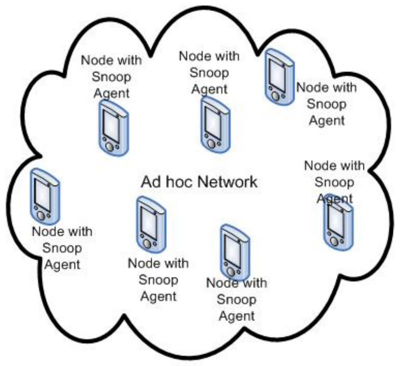
**Figure 1: Snoop in wired-cum-wireless network**

The Snoop protocol was originally applied in a wired-cum-wireless network. For such a network it is assumed that the wired link is more reliable than the wireless link, so applying Snoop in the base station tries to hide wireless link losses from the sender thereby stabilizing the overall throughput.

Multihop wireless networks are networks which use two or more wireless nodes to convey information from a source to a destination. Though Snoop was originally proposed for a single hop wireless link, in principle the Snoop protocol could be applied in multihop wireless links like mobile ad hoc networks or static sensor networks. In these networks Snoop could be applied in two ways. For an ad hoc network connected to a wide area network (WAN) Snoop could be positioned at the entry point of the ad hoc network so any packet loss within the ad hoc network may be hidden from the WAN (Figure 2a). As well Snoop could be applied to all the nodes within the ad hoc network (Figure 2b). This would especially be beneficial when there are many error prone links in the networks. For both the cases Snoop should be able to improve the throughput by avoiding unnecessary congestion control measures at the sender because of some error prone links.



(a) Snoop at the ad hoc network entry point



(b) Snoop in every ad hoc network node

**Figure 2: Application of Snoop in ad hoc networks**

According to the design, when Snoop is applied to nodes in a multihop wireless network (Figure 3) with error-prone links the throughput is expected to improve. But our simulations below showed that when Snoop is applied in this configuration, the throughput is actually reduced. In fact Snoop shows slightly reduced throughput even when there are no errors. In this paper we consider the problem of reduced throughput when Snoop is applied in a multihop topology. The following sections explain why Snoop is showing this behavior and the steps we have taken to make the protocol perform better in this situation.

## 2. SIMULATION OF SNOOP

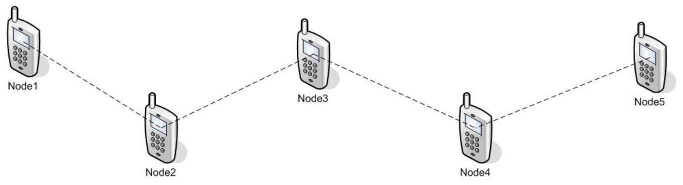


**Figure 3: Multihop wireless topology**

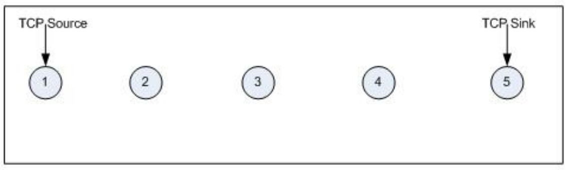
### 2.1 Snoop Implementation in NS2

NS2 is a simulation tool widely used by people in the network research community. NS2 provides substantial support for TCP, routing and other protocols over different mediums including wired and wireless networks [6]. The Snoop protocol is one among the NS2's supported protocols allowing one to simulate and study the protocol behavior. But the current implementation of Snoop in NS2 does not allow to simulate a desirable topology where the Snoop protocol is highly applicable. The Snoop protocol was originally applied in a wired-cum-wireless network. For a typical wired-cum-wireless link it is assumed that the wired link is more reliable compared with the wireless link. The NS2 Snoop implementation allows a wired erroneous network simulation to simulate the effect of the Snoop protocol. This simulation is achieved using wired nodes and a LAN with high bit error rate. With this topology the high error rate LAN is seen as the equivalent of a high error rate wireless link between the base station and the mobile node. But the implementation did not allow simulating a real wireless link. So using the Snoop module, NS2 fails to run a wired-cum-wireless simulation. We have analyzed the Snoop source code and made some modifications to allow simulation of Snoop on wireless links. These changes allowed us to perform more realistic simulation of Snoop applied topology. Before these changes simulations were using a full duplex wired link and so it would not be showing actual wireless link characteristics. For example the long RTT variation we discuss below would not happen if an error prone full duplex link is used to simulate the wireless link.

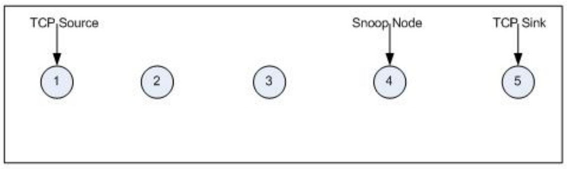
### 2.2 Simulation Environment

We used NS2 to perform the simulation of a multihop scenario that consist of 5 nodes in a chained fashion. The nodes are arranged in a linear format so that they take fixed long route as shown in Figure 3. We applied commonly used wireless node properties for the simulation. The parameters used were two way propagation model, 802.11 as the MAC, omni directional antenna and drop tail interface queue with length of 50 for every node. We used Ad hoc On demand Distance Vector (AODV) as the routing algorithm and the TCPNewReno version of the TCP. FTP traffic is established from node 1 to node 5 with packet size of 1040 bytes and a default window size of 20 packets. The maximum throughput on the simulated wireless link is 1Mbps. The simulation is run for around 200 seconds. The routing protocol is considered less important in the scope of this paper since all of the nodes in this simulation are static.

A set of simulations were performed to get the throughput of TCP without applying Snoop (Figure 4a). Another set of tests were performed on the same topology but node 4 having Snoop applied (Figure 4b). For all these simulations



(a) Simulation topology without Snoop



(b) Simulation topology with Snoop

**Figure 4: Simulation Topologies**

there are no artificially induced errors, only errors occurring at wireless medium may be present such as channel errors. The simulation recorded average throughput based on the actual number of packets received at the sink node.

## 2.3 Simulation Results

Table 1 shows the simulation results with and without using the Snoop protocol. It shows the number of packets received by the destination with and without applying Snoop. In the simulation where Snoop is involved Snoop is applied to node 4. Figure 5 highlights the changes in sequence number increase. When Snoop is applied the TCP sequence number increase is slower than that with regular TCP which causes the throughput to come down. The table and figure clearly show the reduction in throughput when Snoop is applied.

| Simulation Type | Throughput | Number of Packets Received |
|---|---|---|
| Without Snoop | 173 Kbps | 4083 |
| With Snoop | 167 Kbps | 3934 |

**Table 1: Simulation results with and without using Snoop**

## 3. ANALYSIS OF SNOOP

From Table 1 it is clear that the application of Snoop is having a negative impact on the TCP performance. In an error free scenario Snoop is expected have no obvious effect on the performance. In this section we discuss the simulations and the investigations we have performed to understand and explain the problem.

## 3.1 Simulation Environment

In order to look at the effect of Snoop in these simulations we had to carefully analyze the link between the Snoop node and the destination node which is the traffic between node 4 and node 5 as shown in Figure 4b. This is the link mainly getting influenced by the presence of Snoop. For the rest of this section all measurements are taken in the context of node 4 and node 5 only. We have applied all other parameters for this simulation as described in the section 2. The simulation recorded the packet processing times at various layers of the Snoop node and the receiving node, to look at possible packet queuing at the MAC layer.

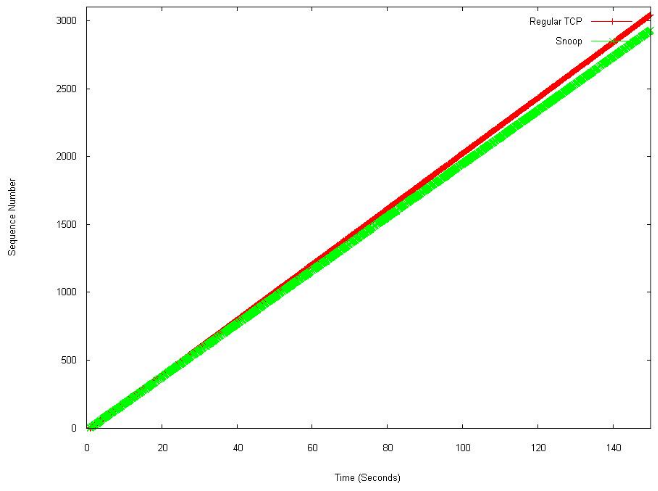


**Figure 5: TCP sequence number increase with and without using Snoop**

## 3.2 Snoop Retransmissions

Snoop protocol aims to reduce the effect of non-congestion-related losses on TCP performance over wireless links. Snoop tries to solve this problem by hiding the errors on the link from the TCP sender so the source will not be aware about the data loss and, hence, it will not reduce its transmission rate. Snoop does this by performing local retransmission of the lost packets and by suppressing duplicate acknowledgments going to the TCP sender. The Snoop protocol is applied at the link layer of TCP/IP. Snoop monitors every packet that passes through the TCP connection and maintains a cache of TCP segments sent across the link that have not yet been acknowledged by the receiver. If a packet is lost because of errors en route, Snoop can know about this loss by receiving duplicate acknowledgment. Moreover, Snoop maintains a timer for each packet and when the timer expires, Snoop knows the packet is probably lost. Snoop can then do local retransmission of the lost packet. Also, Snoop will suppress the duplicate acknowledgment and prevent them from reaching the sender. This will prevent TCP sender from noticing the loss and hence preventing the sender from invoking the congestion avoidance algorithms and reducing its transmission rate.

In the Snoop implementation whenever Snoop buffers a packet it assigns a timer to it. This is done so that after the timeout occurs the packet will be retransmitted if it has not been acknowledged. Snoop calculates this local retransmission timeout value based on the previous round trip time (RTT) values. For Snoop the round trip time is the time to receive the acknowledgment from the destination for a packet after the packet leaves Snoop agent. On each new acknowledgment, Snoop calculates the smoothed round trip timeout (SRTT) which is running average of the RTT values. Snoop applies a similar algorithm to that of TCP to calculate the SRTT. The following algorithm is adopted by Snoop.

```
rtt = currentTime - packet sentTime;
srtt = g * srtt + (1-g)*rtt;
```

*(1)*

*(where g is a smoothing factor between 0 and 1, current time is the time of receiving acknowledgment)*

The smoothed round trip time calculated by Snoop varies based on the time taken by an ACK to arrive the Snoop agent. Snoop applies an external parameter to configure the minimum retransmission granularity by defining the minimum RTO value. This parameter is exposed in NS2-Snoop as SnoopTick. SnoopTick is a static parameter and can be set explicitly in the simulation script. The SnoopTick parameter is used while computing the retransmission timeout. Snoop calculates the its retransmission timeout as the maximum of either srtt + 4* rttvar or the SnoopTick value.

$$\textit{retransmission timeout} = \textit{maximum}(\ \textit{srtt} + 4{*}\textit{rttvar},\ \textit{SnoopTick}) \quad (2)$$

*(where rttvar is the variation of rtt )*

Because Snoop retransmission timeout is primarily dependent on the SRTT, any variation of SRTT would vary the retransmission timeout directly. If the SnoopTick value is also smaller, then a packet arriving at Snoop from the source at a lower SRTT period might have lower timeout set. Due to the low timeout value these packet are likely to timeout before the ACK arrives and Snoop would retransmit them. This retransmission is premature as Snoop did not give the destination sufficient time to respond with an acknowledgment. These unnecessary retransmissions cause increased medium contention and as a result cause the throughput to come down. If the timeout calculation had not considered the shortest SRTT values, the timeout would have been large enough to avoid this unnecessary retransmission.

The value Snoop uses for retransmission timer is thus very important. If the timer is set too low, Snoop might start retransmitting a packet that was actually received by the destination, because Snoop did not wait long enough for the acknowledgment of that packet to arrive. This applies in the low error case. Instead, if Snoop set the timer too high, Snoop wastes time waiting for an acknowledgment that will never arrive and the sender may timeout before Snoop can retransmit the packet and this applies in the high error case [7].

## 3.3 Effect of Queuing Delay

From the Snoop simulation traces it is clear that the Snoop module is performing unnecessary retransmissions. One of the reasons for retransmission by Snoop is the packet's retransmission timer timeout. The packets retransmission timer is primarily depend on the Snoop's calculated SRTT. So the simulation looks to record RTT and SRTT values calculated at the Snoop node.

We analyzed the simulation results for one run to see the reason for reduction in performance when Snoop module is applied. During the simulations the Snoop module retransmitted 427 packets out of the 3934 packets transmitted. In this case all these packets were actually received by the destination, so it ignores the duplicate packet as an out-of-order delivery. But these additional packet transmission take time and can cause extra contention on the medium.

The Snoop round trip time is the time from when a packet leaves the Snoop node to when the Snoop node receives the acknowledgment from the destination node. We analyzed the RTT calculated by the Snoop node at each ACK arrival. Also we have analyzed the queuing delay of a TCP packet at the Snoop node and the queuing delay of the packet at the destination node. Queuing delay here is defined as the duration between a packet leaving the TCP agent and being sent out on the medium.

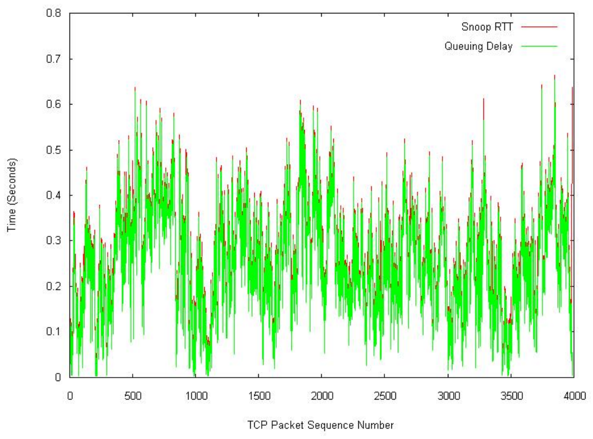
In a two node topology typically the RTT should be the amount of queuing delay at both nodes and the transmission time of TCP packet and ACK packet. With static nodes the transmission time of packet is fairly fixed. We noticed that the RTT for each packet calculated at Snoop node is hugely influenced by the combined queuing delay of that packet at Snoop node and of its acknowledgment at destination. Figure 6a shows the variations of RTT and queuing delay during the simulation. The comparison of RTT and queuing delay is represented in the Figure 6b. In general the RTT is proportional to the queuing delay, but the plots appearing out of proportions are of the pre-mature retransmissions.

In a wireless node the main factor for high queuing delay is the medium contention. Medium contention increases as more and more packets are waiting to be sent out i.e. when packets are queued. We have recorded the combined queue length of Snoop node and destination node to confirm its effect on the RTT. Figure 6c shows the comparison between RTT and interface queue length. The RTT varies for the same combined queue length. Average RTT increases as the combined queue size of the TCP sink and source increases and this is probably due to medium contention. This problem is more relevant to the wireless medium because of the medium contention.

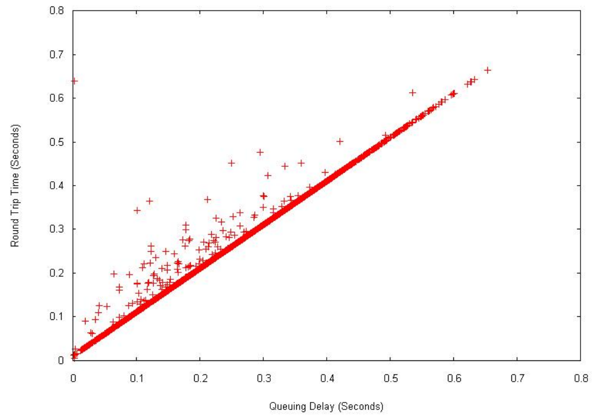
## 4. SUPPRESSING RETRANSMISSIONS

We have noted that the principal reason for Snoop performing poorly in a multihop wireless network is the unnecessary retransmissions. We performed several experiments with the Snoop protocol and compared the resulting performance with regular TCP. In this section we explain the impact of suppressing premature retransmissions on Snoop performance.

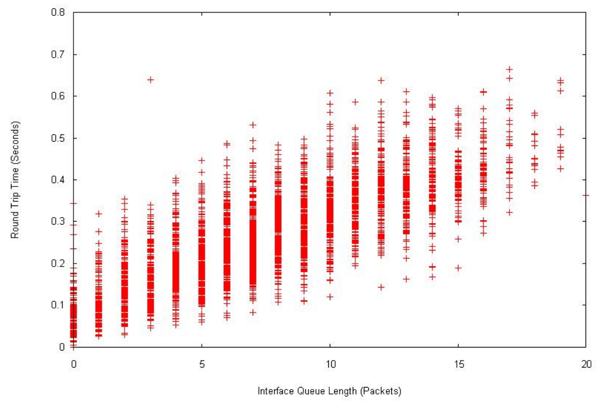
The problem observed here is because Snoop is timing out before the acknowledgments are received from the destination. Snoop implements two validations to detect packet loss. It detects packet loss when a duplicate acknowledgment is received or if the local retransmission timer expires. So if the local retransmission timer is not used or not expired Snoop will still detect the packet loss when the next duplicate ACK is received. Except the last packet for all other packets this will be true. We recommend suppressing these pre-mature retransmissions by way of setting a higher retransmission timer at Snoop node which could cover the



(a) RTT and Queuing Delay



(b) Variation of RTT against queuing delay



(c) Variation of RTT against interface queue Length

**Figure 6: Effect of queuing delay on RTT**

problem of variation in RTT. But when the RTT is much smaller to that of the retransmission timer and if the packet is dropped, the Snoop node will not be retransmitting the packet until the larger timeout occurs. This is in contrast to the original Snoop proposal.

To the simulation topology with one Snoop node (Figure 4b) we applied a higher SnoopTick value of 1 second. In the Snoop implementation this made sure that the value of retransmission timeout will be 1 second or higher. The simulation was setup as explained in section 2 along with the higher SnoopTick value. Then the simulation was run with various error rate and for each error rate it was repeated ten times to get average throughput. The error was artificially introduced by dropping packets corresponding to the applied bit error rate. Analyzing the results it was apparent that the higher Snoop retransmission timeout value has suppressed the pre mature retransmissions. This has lead to Snoop performing better than in the without Snoop case. Table 2 shows the comparison of average throughput of an FTP connection using the Snoop protocol with that of an FTP connection using the regular TCP implementation. The table also lists the results for various bit error rates on a log scale.

| Bit Error Rate | Average Throughput in Kbps | |
|---|---|---|
| (1 in every x bits) | (Without Snoop) | (With Snoop) |
| 64Kb | 117.8 | 174.9 |
| 128Kb | 150.7 | 173.2 |
| 256Kb | 161.5 | 172.0 |
| 512Kb | 166.3 | 171.1 |
| 1Mb | 168.5 | 170.8 |
| 2Mb | 169.1 | 170.0 |
| 4Mb | 169.8 | 170.0 |
| 8Mb | 169.9 | 169.9 |
| No Errors | 170.0 | 170.0 |

**Table 2: Result when Snoop on single node with higher retransmission timer**

Figure 7 compares the average throughputs and its variation of the above simulation. The vertical bars in the plots show the standard deviation of average TCP throughput. The figure shows that from an error rate equivalent of 1 in 1Mbits (close to the point 5 on the X-axis of the graph), Snoop is performing much better than the regular TCP. At an error rate of 1 in 64Kbits Snoop improved the average throughput by around 50 percent.

In an ad hoc network there may not be any central node or a fixed route. From the TCP perspective every ad hoc network node is likely to be identical and so it is not fair to use or apply Snoop to one node. We have applied Snoop to all nodes to make the simulation more realistic (Figure 8).

We again performed the simulation with higher SnoopTick value of 1 second. In these simulations the only difference is that Snoop is applied to all the nodes. Along with the application of Snoop we also introduced errors into each of the nodes on its outgoing packets. The results obtained show that Snoop is able to make significant improvement to the performance of TCP in a multihop wireless network. Table 3 shows the average TCP performance with and with out

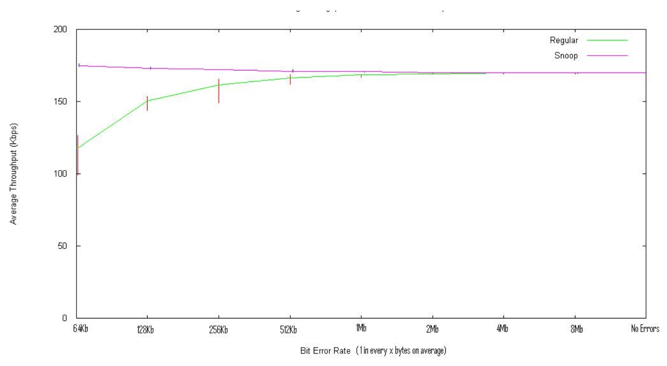


**Figure 7: Average TCP throughput with high retransmission timer at different error rates**

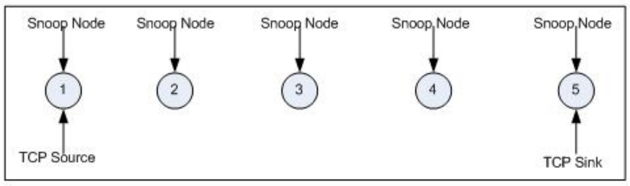


**Figure 8: Simulation topology with Snoop on all nodes**

Snoop. The average throughput at higher error rates are very low because in this case artificial error is introduced at each node.

| Bit Error Rate | Average Throughput in Kbps | |
|---|---|---|
| (1 in every x bits) | (Without Snoop) | (With Snoop) |
| 64Kb | 2.2 | 8 |
| 128Kb | 34.4 | 60.3 |
| 256Kb | 114 | 150.4 |
| 512Kb | 153.7 | 162.1 |
| 1Mb | 163.7 | 166.6 |
| 2Mb | 167.8 | 166.9 |
| 4Mb | 168.7 | 167.9 |
| 8Mb | 169.5 | 166.9 |
| No Errors | 170 | 170 |

**Table 3: Result when Snoop on all nodes results with higher retransmission timer**

Figure 9 compares the performance of regular TCP with Snoop and shows the throughput and it deviation at different error rates. At error rates above 1 in 1Mbits (the left hand side of the graph) Snoop has significant improvement on the TCP performance.

Figure 9 is comparable to that in the original Snoop implementation paper [3]. In both the cases the pattern of Snoop behavior is very much similar. The original paper demonstrated that the Snoop protocol made significant improvements when the error rates are over 1 error bit in 2Mbits. On lower error rates Snoop behaves as if it is not present and this ensures no degradation in the performance.

## 5. OTHER APPROACHES

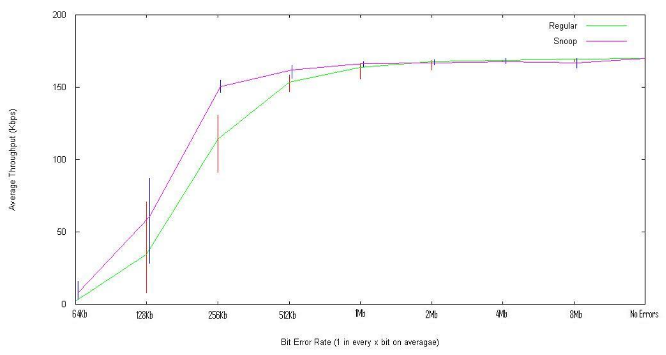


**Figure 9: Average TCP throughput with high retransmission timer at different error rates when Snoop is on all nodes**

In the following sections we discuss some of the potential ways to improve the problem with Snoop in multihop wireless links. We also detail the impact of these fixes.

### 5.1 Reduce Interface Queue Size

A reduction in queue size at the mobile node would avoid the condition of having high number of packets queued and thus would provide a less variable RTT. But when more packets are sent by the TCP source they might be dropped if any of the node's queue is full. The overall throughput would be reduced this time because of the packet drop at nodes due to insufficient queue size instead of the retransmissions at Snoop node. Our simulations also included the different TCP window size along with the the queue size applied. We analyzed these scenarios and the results are presented in Table 4. From our simulation results it is clear that reducing queue size may not be the best solution for supporting Snoop to perform better in multihop wireless networks.

| IF Queue Size | TCP Window Size | Throughput(Kbps) |
|---|---|---|
| 50 | 20 | 169.6 |
| 25 | 20 | 169.6 |
| 10 | 20 | 170.6 |
| 5 | 20 | 168.6 |
| 2 | 20 | 167.3 |
| 50 | 5 | 167.4 |
| 25 | 5 | 167.4 |
| 10 | 5 | 167.4 |
| 5 | 5 | 168.6 |
| 2 | 5 | 166.8 |

**Table 4: TCP throughput with Snoop on node 4 when queue size and window size vary (See Figure 4b)**

### 5.2 Modify Snoop's RTO Calculation Logic

The SRTT calculation within Snoop could be modified to adapt to the variation of RTT. The RTO calculation could include algorithms to dynamically cater for the large variation of RTT. The purpose is that the RTO values can be higher enough to acomodate the above average RTT value and so in most of the higher RTT cases it would prevent Snoop from retransmitting packet pre maturely. From our simulation results we observed that at the Snoop node the

RTT value varies between 0.1 second to over 0.5 seconds. This is a huge variation, roughly 400 percent. But the existing Snoop RTO calculation logic is only accommodating a much smaller variation. The regular TCP retransmission timeout calculation mentioned in [1] was also amended few times to accommodate the RTT variance [9]. In the original implementation [1] TCP assumed that the variance in RTT would be small constant. Jacobson's algorithm recommended incorporating the measured RTT variance is especially important on a low-speed link [8]. Karn's algorithm for selecting RTT measurements ensures that ambiguous round-trip times will not corrupt the calculation of the smoothed round-trip time in TCP [10]. Currently calculation of retransmission timeout in Snoop is based on smoothed round trip time, which incorporates simple variance. The amendments applied to regular TCP to improve the situation are not fully applied to the current Snoop implementation.

The calculation of the optimal value can be further complicated because of the fact that in a multihop wireless network Snoop could be applied to all nodes. In this case the RTO calculation might need to depend on the number of hops from that specific node to the destination.

## 6. FUTURE WORK

We have applied the higher minimum retransmission timer to the Snoop protocol and performed some simulations. In real world solutions involving Snoop it is likely that Snoop shall be applied to all the nodes in a network because unlike a wired-cum-wireless scenario there are no base station or central node in an ad hoc network. When a real congestion occurs in the ad hoc network it is possible that many of the Snoop agents could trigger retransmissions. There is work to be done to analyze the effect of Snoop retransmissions in an already congested network.

Although the higher retransmission timer is helping to boost Snoop performance compared to its original approach, the throughput could be further improved if a mechanism for optimal timeout value could be implemented. Also so far our simulations concentrated only on static ad hoc networks. Since in a typical ad hoc network there may not be any fixed routes, a packet could travel via different routes at various times. Because of these potential route changes it is required to monitor the Snoop behaviour and performance in such networks. We are planning to work on the various applications of Snoop in a multihop dynamic ad hoc wireless networks.

## 7. CONCLUSION

The Snoop protocol has been proposed to improve TCP throughput in a wired-cum-wireless environment. It can also improve TCP performance quite well in wireless links but has a problem; when the retransmission timeout is set to very low, the Snoop protocol performs premature retransmissions. This results in the degradation of TCP throughput and so creates a negative effect on a wireless network. We have applied a higher retransmission timer and have proved that it helps Snoop to improve its performance in a multihop wireless network. Our simulations show that a wireless network is more complicated than just a point to point link with a higher error rate.

## 8. REFERENCES

[1] Transmission Control Protocol, *RFC 793*, Sep. 1981.

[2] Ahmad Al Hanbali, Eitan Altman and Philippe Nain, *A Survey of TCP over Mobile Ad Hoc Networks*, IEEE Communications Surveys and Tutorials, 2005, volume 7, pages 22-36.

[3] Hari Balakrishnan, Srinivasan Seshan, Elan Amir and Randy H. Katz, *Improving TCP/IP Performance over Wireless Networks*, ACM International Conf. on Mobile Computing and Networking (Mobicom), 1995.

[4] Xiang Chen, Hongqiang Zhai, Jianfeng Wang and Yuguang Fang, *TCP Performance over Mobile Ad Hoc Networks*, Canadian Journal of Electrical and Computer Engineering (CJECE) (Special Issue on Advances in Wireless Communications and Networking), 2004.

[5] M. Alman, V. Paxson, W. Stevens, *RFC 2581- TCP Congestion Control*, Network Working Group, 1999.

[6] VINT project, *Network Simulator (NS) Manual*, The Network Simulator ns-2: Documentation, http://www.isi.edu/nsnam/ns/ns-documentation.html.

[7] Mohammed Alnuem, James A.Malcolm, *Enhancing the Snoop Protocol on Wireless LANs with high error rates TR 497*, University of Hertfordshire.

[8] Requirements for Internet Hosts – Communication Layers, *RFC 1122*, Oct. 1989.

[9] Computing TCP's Retransmission Timer, *RFC 2988*, Nov. 2000.

[10] Karn, P. and C. Partridge, *Improving Round-Trip Time Estimates in Reliable Transport Protocols*, SIGCOMM, 1987.