UNIVERSITY OF HERTFORDSHIRE

Faculty of Information Sciences

MASTER OF SCIENCE IN DISTRIBUTED SYSTEMS AND NETWORKS

Project Report

Selected extracts of an old report for the guidance of current students

Please note: numerous chapters, sections and sub-sections have been omitted from this copy, as have the bibliography and appendices.

In the real report, each chapter started on a new page (use insert break to achieve this, not multiple new-lines).

Please remember that while this is a very good report, it is not perfect: there is a considerable number of errors here.

The Performance of Snoop Protocol over Wireless Networks

September 2005

Abstract

Many solutions have been proposed to overcome the problem of TCP bad performance over wireless networks. one of the promising solutions is the Snoop protocol. This is a link layer protocol that resides in the base station between the sender and receiver and tries to hide the errors on the wireless link from the TCP sender. it does that by suppressing the duplicate acknowledgments from the TCP sender and by doing local retransmission of lost packets based on duplicate acknowledgment and local timer.

We investigated how the increase of the bit error rate on the wireless link affects the performance of Snoop protocol. Results show that Snoop performance degrades gradually with the increase in the bit error rate and it becomes worse than TCP at some point.

We have studied this behaviour and propose some solutions to improve Snoop performance over networks suffering from high bit error rates. We found that, under high bit error rates it is better to make Snoop to pass the duplicate acknowledgments to the TCP sender in order to prevent it from having time out. Another solution we propose is to increase the local retransmission rate of Snoop since this will increase the probability of receiving the sent packet by the TCP receiver.

These enhancements have been applied to the Snoop under ns2 simulator and the tests showed big improvement on Snoop protocol performance under high bit error rates.

Finally, we combined the new enhancements with the original design of Snoop to produce an adaptive protocol that works well both at high and low bit error rates.

Acknowledgments

I wish to express my gratitude .....

Table of Contents

Chapter 1

Introduction

In this chapter I give an overview of the problem dealt with in this project, then present the project goals and research question. Finally I give a brief about the report chapters.

1.1 Overview

Mobile computing has become an important part of today's life especially with the gradual advances in wireless communication technologies. Usually, mobile computing devices can move freely over a mobile network, where a mobile network physical topology is a combination of two kinds of networks: wired network as a backbone, and a wireless network (elaarag, 2002).

Transmission Control Protocol, widely known as TCP (Postel, 1981), is the standard transport protocol in the internet (elaarag, 2002). This is because of the wide range of network and internet applications that use TCP in their communications. As a result, it is absolutely essential that TCP & IP can be used in mobile networks so that they can integrate easily with already existing systems and networks.

However, in order to achieve acceptable mobile internetworking, the mobile user should be able to access the same level of service as provided to a fixed user. Unfortunately, TCP is found to perform badly over wireless networks. The problem appeared because wireless links has different characteristics than wired links. one difference is that wireless links suffer from higher rates of bit errors and also suffer from frequent disconnections. TCP was not designed to deal with this type of errors and disconnections; as a result, this has caused big degradation in TCP performance. This happens because TCP confuses the data loss because of the errors in the wireless link and the loss because of congestion in the link. Hence, it reduces its transmission rate to avoid having permanent congestion in the link.

Many solutions have been proposed to overcome this problem and to improve TCP performance over wireless network. one of these solutions is the Snoop protocol (Balakrishnan *et al.*, 1995). Snoop tries to solve the problem by hiding the errors on the link from the TCP sender so it 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 from the TCP sender. This strategy gave good results when used under low bit error rates.

In my work I investigate the behavior of Snoop protocol when operate on wireless networks suffering from high bit error rates. The results show that Snoop performance falls dramatically with the increase in the bit error rate and the Snoop performance gave even worse results than TCP at some points. The main objective at this point was to understand the cause of this dramatic fall in the Snoop performance and to try to advice solutions for the problem.

The solutions we have suggested include the use of the duplicate acknowledgments to prevent the TCP sender from having timeout and to increase the number of packet local retransmission in order to face the increase in packet loss. These enhancements has been applied to Snoop and the results show a big improvement in Snoop performance over wireless networks suffering from high bit error rates.

The final step is to combine the original Snoop design, that works well under low bit error rates, with the new enhancements, which work well under high bit error rates, to produce one protocol which we call Adaptive Snoop that give good performance under both high and low bit error rates.

1.2 Project Goals

The issue we deal with in this work is TCP performance problems over wireless networks and Snoop performance problems. This issue is part of a wider and active research area of wireless network performance.

In this section I present the project objectives and research question which were developed at the start of the project.

The research question I address in this project is:

- How do high bit error rates in a wireless network affect the performance of the Snoop protocol (Balakrishnan *et al.*, 1995) and how do these high bit error rates affect the number of duplicate packets sent.

[objectives listed here]

In order to achieve these objectives I divided my objectives into tasks and I made a time plan to do these tasks. The project plan can be found in Appendix i. The evaluation of the project plan and a discussion of the extent to which we have achieved the objectives we had set, can be found at the end of this report in chapter 6.

1.3 Outline of the Report

The research work presented in the report is developed within six chapters including the present one.

Chapter two discusses the issue of TCP performance over wireless networks. An overview of TCP and how it works is reviewed, then a brief explanation of the congestion control and avoidance algorithms is given in order to understand what causes TCP to decrease its performance. We give then an evaluation of TCP performance over wireless networks by running TCP over a simulated wired-wireless network using ns2 simulator. Finally the chapter presents a brief of related work by presenting solutions proposed to improve TCP performance over wireless networks.

Chapter three describes how Snoop protocol works and explains the main modules involved in the design of Snoop protocol.

In chapter four, we do a set of experiments to evaluate the performance of Snoop protocol under low bit error rates and under high bit error rates and we show how the increase in the bit error rate has affected Snoop performance dramatically.

As a result of what we have got in chapter four, in chapter five we discuss several solutions to the problem and test them. We found that, under high bit error rates, passing Duplicate Acknowledgment to the TCP sender will prevent the TCP sender from having timeout. Moreover, we found that under high bit error rates, increasing the Snoop local retransmission rate will increase the probability that the packet will reach the receiver. Finally, to combine the benefits of the original Snoop, which works well under low bit error rates, with our solutions, we have integrated them into one protocol that gives high performance under both low and high bit error rates; we call it Adaptive Snoop.

Chapter six give a conclusion and evaluation of the results we have got and evaluation of the project plan we have applied in this project along with discussion of the objectives we have achieved in this work. Also it gives some recommendations for future work.

Finally, the appendices contain details of the experiments we did during this project, the Project Plan we applied, explanation of the congestion avoidance algorithms, and the changes we have made to Snoop implementation.

Chapter 2

TCP Performance over Wireless Networks

In this chapter we discuss the issue of TCP performance over wireless networks. We start by giving an overview of TCP and how it works since a good understanding of TCP will help us to understand the problem it may face in wireless networks. Then we discuss TCP congestion control and avoidance mechanisms and why they cause TCP to reduce its performance. Then we evaluate TCP performance over wireless networks by running TCP over a simulated wired-wireless network using ns2 simulator. Finally in this chapter, we give a brief of related work by presenting proposed solutions to improve TCP performance over wireless networks.

2.1 Overview of TCP

2.2 TCP Congestion Control

2.3 TCP Performance over Wireless Networks

initially the experiment was performed with no error rate so TCP will give full performance. After that and in order to see how TCP behaves when errors are present in the link, I run the experiment four times each with different error rate. First, I add an error rate of 10% (10% of the packets are dropped) then 20%, 30%, 40% were added to the link. This will show if the increase in the error rate will affects the TCP throughput.

2.3.3 Adding error rate to the wireless link in ns2

The ns2 manual (ns Notes and Documentation, 2002) provides a full documentation about how to add error to wired and wireless links. The explanation can be found in Chapter 13 under the title error Model (ns Notes and Documentation, 2002).

Here I will give brief explanation of the TCL code I have added in order to introduce errors to the wireless link:

```
$ns_ node-config -incomingerrProc Wirelesserr

proc Wirelesserr {} {
set wl-err [new errorModel]
$wl-err unit packet
$wl-err set rate_ 0.3
return $wl-err }
```

From above TCL code we can see that errors can be added to the mobile node using the node-config command. in the node-config we specify the error model or the procedure that implement that error model. For wireless nodes we can't add the error model directly to the node, instead, we have to specify a procedure and supply the mobile node with the procedure name via `node-config`.

2.4 Solutions for TCP performance over wireless links

Several techniques were proposed to overcome this problem (*i.e.* TCP bad performance over mobile networks). in general, solutions can be classified into three categories: end-to-end protocols, link layer protocols and split connection protocols (elaarag, 2002; Balakrishnan *et al.*, 1996). These categories divide the solutions into classes "*based on their fundamental philosophy*" (Balakrishnan *et al.*, 1996).

The end-to-end protocols try to improve TCP performance by making the TCP sender distinguish between losses because of congestions and those because of link failure by using a mechanism called explicit Loss Notification (eLN) (Balakrishnan *et al.*, 1996). eLN will mark lost packets because of wireless link error and will identify them as non-congestion loss. This way, TCP will not invoke the congestion control procedure and so no need to reduce the transmission window (Balakrishnan *et al.*, 1996).

Another proposed end-to-end solution suggests using of Selective Acknowledgments of received packets. Hence, allowing the sender to retransmit packet based on selective acknowledgments sent by the receiver and so the sender can create a bit mask of successfully received packets (Balakrishnan *et al.*, 1996).

The main advantage of the end-to-end solutions is that they do not require any changes to be made to the base station or to the intermediate routers and they limit the changes to the communicating parties only, the fixed host and the mobile host. Hence, this approach will work on any wireless networks without the need to change any of the network components. However, this approach requires changing the TCP implementation in both sides and this requires recompilation of existing systems, which not desirable because it will violate the backward compatibility with existing systems (elaarag, 2002).

Link layer protocols try to solve the problem in the link layer rather than the transport layer. They do so by increasing the quality of the wireless link in away that allows TCP to get a reliable but slow connection (Parsa & Garcia-Luna-Aceves, 2000). The link layer protocol will prevent TCP from dealing with lost packets, instead, it will monitor every packet and detect lost packet using a timeout and duplicate acknowledgments. The protocol then will retransmit the lost packet on behalf of the TCP. This way TCP will not know about lost packet and, hence, it will work like in a reliable wired link. (Balakrishnan *et al.*, 1996). examples of link layer protocols are Snoop protocol developed by Balakrishnan *et al.* (1995), TULiP developed by Parsa & Garcia-Luna-Aceves (2000) and AiRMAiL protocol developed by Ayanoglu *et al.* (1995). The main advantage of this approach is that it maintains the end-to-end semantic of TCP protocol since it does not break the connection (Balakrishnan *et al.*, 1996). The main disadvantage of this approach is that some times it can not completely hide the link errors from the TCP sender

Finally, the split connection approach works by splitting the TCP connection into two connections: wired TCP connection between the fixed host and the base station and the other is wireless connection between the base station and the mobile host, the wireless connection can use specialized protocol that increase the performance over wireless links (elaarag, 2002; Balakrishnan *et al.*, 1996). An example of this kind is the indirect TCP Protocol developed by Bakre & Badrinath (1997). The main advantage of this approach is that it hides the TCP sender (*i.e.* fixed host) from the wireless link. Moreover, no need to recompile existing systems at the fixed host, so it provides compatibility with existing systems (elaarag, 2002). on the other hand, it violates the end-to-ends semantic of TCP since the TCP connection should pass through the base station instead of aiming directly to the destination as in regular TCP connection (Balakrishnan *et al.*, 1996).

Chapter 3

The Snoop Protocol

In this chapter we describe how Snoop protocol works and explain the main modules involved in the design of Snoop protocol.

Chapter 4

Performance evaluation of Snoop Protocol

4.1 Overview

We have discussed in the previous chapter that Snoop improves the performance of TCP over wireless networks by hiding the errors in the wireless link form the TCP sender and also by doing fast local retransmission of the lost packets so the TCP sender will not need to resend it again. However, some authors like Parsa & Garcia-Luna-Aceves (2000) indicated that under higher bit error rates Snoop performance will not hold up.

In this chapter we will do a set of experiments to, first, show how Snoop improves the performance of TCP under low bit error rates. After that a set of experiments will show how Snoop behaves under higher bit error rates and if it is still improving TCP performance.

4.2 Simulation Methodology

In our experiment, a complete network was simulated using ns2, the network topology and the experiment settings are explained below.

4.3 Performance of Snoop under low bit error rates

Chapter 5

Improving Snoop Performance

The results we have found from the previous experiments shows that Snoop performance decreases with the increase in the error rate. This let us to think if there is a solution that can increase Snoop performance. in This chapter, I will test four proposed solutions to improve Snoop performance over wireless networks. each solution has been tested and the results have been discussed.

5.1 Changing the packet size

5.2 Changing Buffer size

5.3 Enhancing the Snoop Protocol

In the previous experiment we have noticed that with high error rates, like 30% and 40%, we need to reduce the buffer size in order to increase the performance see Appendix F for the experiment details and see Figure 5.4.

For example, in Figure 5.5 the optimal buffer size is reached at 4KB for error rate 30% and when the buffer size increased, the Snoop performance dropped down (in Figure 5.5 we show the buffer size on KB on each point). This behavior is strange because with bigger buffer size Snoop will be able to cache more packets and, hence, will be able to retransmit more lost packet without the knowledge of TCP sender so this suppose to improve the performance. But what happens here is the opposite. After reaching 4KB, the increase in the Snoop buffer size has decreased the performance.

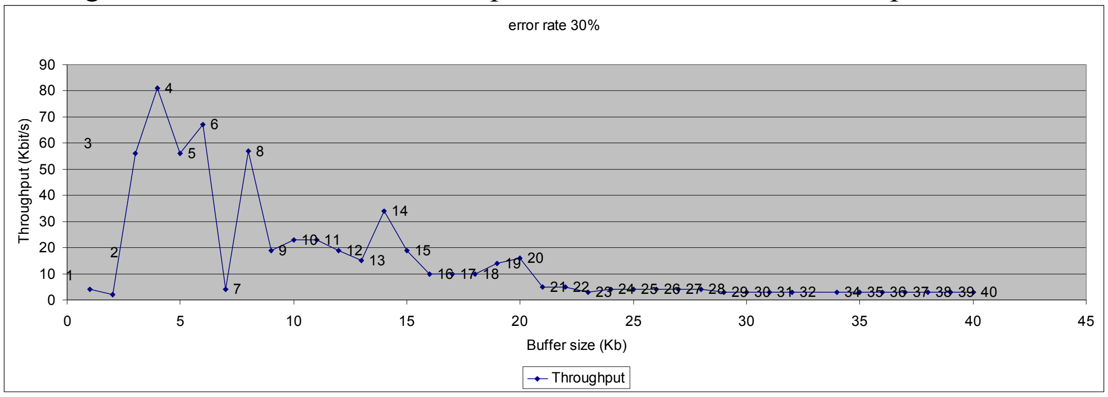

Figure 5.5: Buffer size *vs.* Throughput for error rate 30%.

5.3.1 Building new trace files

In order to understand why Snoop performance decreases with the increase in the buffer size, I have made detailed search through the ns2 trace files. This search included looking inside ns2 slandered trace files which give information about the connection on timely bases.

However, in order to trace Snoop behavior we used more information than what is provided by standard ns2 trace files. For example we used following information:

- Knowledge of when TCP sender timeout.
- Snoop buffer status at each moment, what are the packets in the Snoop buffer at each time and how many packets has been queued in the buffer and how many of them are delivered successfully to the TCP receiver.
- Which packets has been retransmitted by Snoop and which has not

- Which acknowledgment or duplicate acknowledgment was suppressed from TCP sender and which was passed to the TCP sender.

Some of previous points can be traced through ns2 standard trace files but with large effort. Also ns2 trace files do not give the throughout or the average throughout of a transmission session. Fore these reasons we have to build our trace files that represent the state of Snoop at each point of time.

This made us to look at the TCP and Snoop implementation and try to understand the execution path in order to debug through the code and to understand what happen in each step so we can create flags through the code. For example, in order to understand the behavior of Snoop with different buffer sizes, I made a new trace output where the buffer is visualized as a series of packets sequence numbers which are currently in the buffer. The trace file also show what happen until next update of the buffer like having acknowledgment or having packet retransmitted. Moreover, when an acknowledgment received, the type of acknowledgment (new or duplicate acknowledgment) is noted. Finally when the TCP sender times out, a "`TCP Sender timeout`" comment will appears in the trace file, this was done by finding the relevant module in the TCP-Reno C++ implementation and add this output code to it.

In fact this gave me a good opportunity to look at TCP implementation and more deeply at Snoop implementation and allow me to trace what happen at each step when Snoop operate under high bit error rates. In Appendix J we provide more discussion about why we needed this new trace files.

Following we present a snapshot from the new trace file. As we can see this new trace format provide information about the packets in the Snoop buffer at each point of time.

5.3.2 TCP Sender Timeout

In order to answer the question we raised before of why having smaller Snoop buffer size has increased the performance under high error rates we have to look again at the experiments we did in chapter 4 and compare the Snoop performance under low bit error rates with Snoop performance under high bit error rates and try to find the major changes between the two cases.

5.4 Passing Duplicate acknowledgment to TCP sender

5.5 Multi-retransmission of lost packets

Chapter 6

Conclusions

6.1 Results Evaluation

During the course of this project we have conducted a set of experiments using ns2 to show how TCP performance decreases over wireless networks. We have explained that the reason for that degradation in performance is that the TCP sender mistakenly thinks that the losses in the wireless link are because of congestion.

We have shown how Snoop improves the performance of TCP by hiding the errors in the wireless link from the TCP sender and by doing local retransmission of the lost packets from the base station. The resulted show that Snoop dramatically improved the performance of TCP when the error rates are 1% to 10% of packet loss.

However, when the error rate reached 25%, Snoop failed to improve the performance of TCP and even it gave worse performance than TCP alone.

one important objective of this work is to show by experiment how Snoop performance degrade with the increase on the error rate and to show how it can not improve the performance of TCP under high bit error rates.

Another objective is to try to understand why Snoop performance degrades dramatically under high bit error rate and to try to find solutions for the problem.

In order to find a solution we have tested different hypotheses. We started by changing packet size to see if smaller packet sizes will improve the performance by allowing more acknowledgment to reach the TCP sender. The results showed that this idea did not improve the performance and with smaller packet sizes we got less throughput.

Then we tried to think about Snoop buffer and if the increase in the Snoop buffer will increase the performance since bigger buffer will make it more likely to find the lost packet in the buffer and hence prevent TCP sender from noticing the loss and then reducing its performance. The results show that the Snoop buffer size was not the problem because we found that the buffer size that gives highest performance is much less than the default Snoop buffer size in current ns2 implementation. Hence, the buffer size is not the cause of the performance degradation.

At this point and from the results we have got in the previous experiments we created good idea about the problem the Snoop suffer from under high bit error rates. We found the problem is that TCP timeout because Snoop unable to deliver the lost packet before TCP sender timer expires. This make us to look in the Snoop design specially the Snoop_ack model and try to enhance it.

The first enhancement we did is to allow Snoop to pass the duplicate acknowledgments to the TCP sender in order to prevent it from timeout. The experiment we did on Snoop after this enhancement, show a noticeable improvement of Snoop performance under high bit error rates.

The second enhancement is to increase the Snoop local retransmission rate with the increase in the error rare because a higher number of sent packets will make it more likely to reach the destination. The experiments on this enhancement show a big improvement in Snoop performance.

Applying previous enhancements on Snoop has increased the performance dramatically over high bit error rates 10% to 40%. However, Snoop original design doing well under lower bit error rates from 1% to 10%. So, in order to make one protocol that gives good performance under low and high bit error rates we have integrated our enhancements with the original Snoop design and we made one protocol called Adaptive Snoop. Adaptive Snoop works by monitoring the status of the link and by switching between the original design and the enhanced as necessary.

In figure 6.1, we present a comparison between the enhancements we made and the original Snoop. We compare following: increasing retransmission rate, passing duplicates acknowledgments, increasing retransmission rate and passing duplicates acknowledgments together and the original Snoop. As we can see increasing the retransmission rate gave the best performance in most cases. This is because with high retransmission rate the packet will be delivers before TCP sender timeout and also without the need to pass duplicate acknowledgments which will decrease the TCP performance. in case of Dup ack we pass the duplicate ack to prevent TCP sender from timeout and to give Snoop more time to resend the packet (by local timer for example)

In figure 6.2, we compare the performance of Adaptive Snoop, original Snoop and TCP alone. We can see that since adaptive Snoop apply both the original Snoop and the enhanced one; it gives good performance In both low and high bit error rates

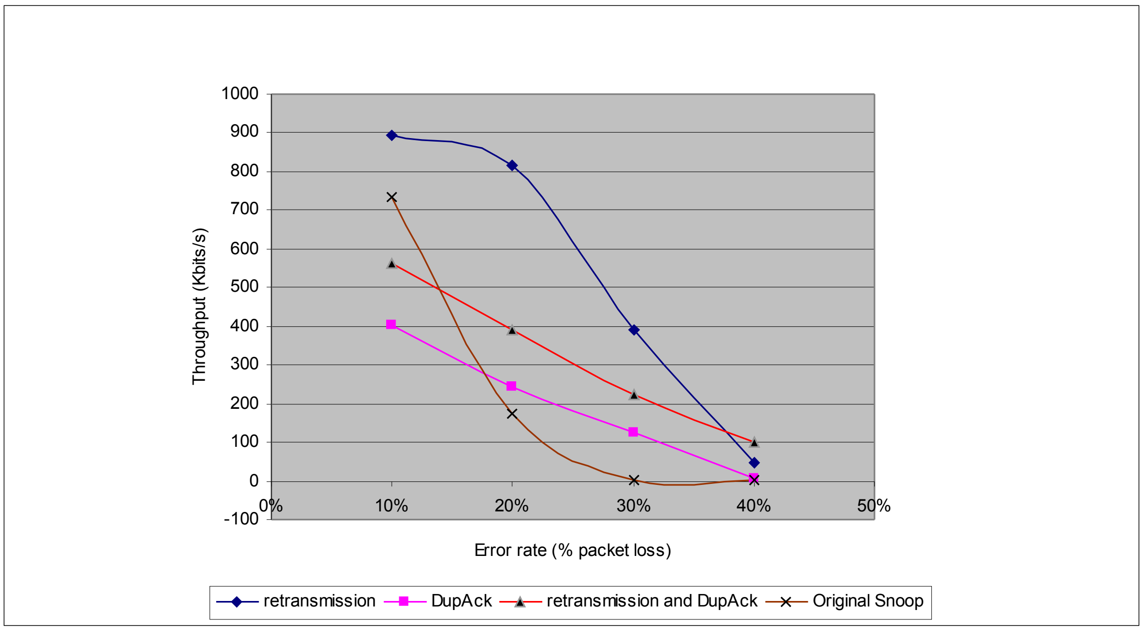

Figure 6.1: Comparison of different enhancements of Snoop

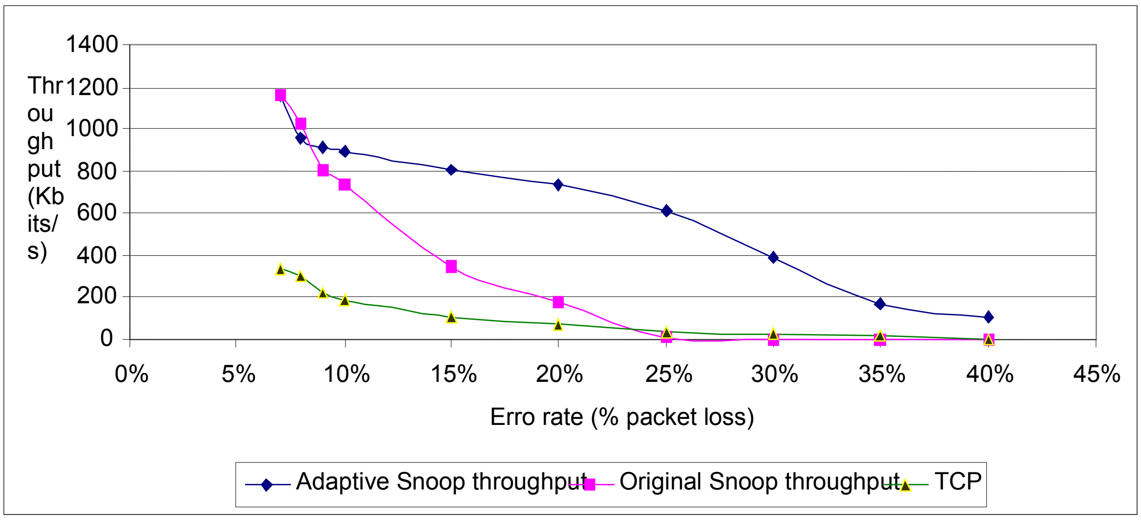

Figure 6.2: Comparison between Adaptive Snoop, original Snoop and TCP without Snoop

The methodology we used to test our hypotheses is to conduct a set of experiments. The experiments have been done using simulation software called *ns2* (The Network Simulator-ns2,2002).

Many authors have used the simulation technique to measure the performance of a specific network or a network protocol. For example Holland and Vaidya (2002) has used ns2 simulator to analyze the performance of TCP over Mobile Ad-hoc Networks also Hung-Yun and Raghupathy (2001) have used ns2 to compare the performance of a Multi-hop wireless networks.

The results of all experiments are presented throughout the report chapters and detailed numerical results of some of them were shown in the Appendix.

The results of this work will be of interest to the research community of wireless networks and wireless network performance since it focus on the performance of an important protocol that supposed to give high performance over wireless links and also tries to propose and apply some solutions to problems Snoop faces under high bit error rates.

6.2 Evaluation of the Project Plan

The project plan I have conducted is presented in Appendix i. I have tried to design the plan in such away it reflects the objectives we have set at the beginning of this report. I did this by making a set of tasks that we have to perform to achieve our objectives.

In order to achieve the core objectives we have set the following tasks:

1. Literature search
2. Critical Literature review
3. Study the ns2 and produce first wireless model
4. Model the Snoop protocol
5. introduce the bit error and run the simulation

The explanation of each task and what it involves can be found in Appendix i.

I have worked to achieve the first and second task by making a literature review of the TCP performance over wireless network in chapter 2. This involved a relatively deep look into TCP functionality and TCP congestion avoidance algorithms as they important to this issue. Then I discussed how the congestion avoidance algorithm made TCP to degrade its performance over wireless networks and then I supported this argument by making an experiment using ns2 to show how TCP performance degrade over wireless network and by doing this I have achieved task 3.

In order to achieve task 4 and 5, I have discussed the design and functionality of Snoop protocol in chapter 3 and then in chapter 4, I did a set of experiments to, first, show how Snoop improves the performance of TCP under low bit error rates. After that a set of experiments to show how Snoop behaves under higher bit error rates and find out if it still improves TCP performance.

In task 4, we were intending to add Snoop to the link layer of the base station but we decided for simplicity to simulate a wireless network using a wired network suffering from errors. Later we find this a good decision because there are problems in ns2 if we add Snoop to the link layer of base station; we discussed this issue in chapter 4 (look under simulation methodology). Task 5 was performed and the results were presented in chapter 4.

For the advanced objectives we have set following tasks:

6. Changing Snoop buffer size.
7. Changing the packet size.
8. Passing the duplicate acknowledgments to the TCP sender.
9. increasing Snoop local retransmission rate.
10. Test Snoop in Ad-hoc wireless network.

The explanation of each task and what it involves can be found in Appendix i.

The tasks 6 and 7 have been done in chapter 5. The tasks 8 and 9 were not in the original plan but they were added later as a result of the outcome of the previous tasks. We did these tasks (*i.e.* tasks 6, 7, 8 and 9) in chapter 5 where we present how we did them along with the experiments and their results.

Task 10 was not performed because the outcome of the previous tasks has given us good results and I decided to continue on the direction of the tasks 6, 7, 8 and 9.

The remaining tasks are:

11. Analyze the results
12. Writing the report
13. Finalize the project

In task 12 I analyse the results from each experiment and try to explain what caused the problem. Usually I do this right after the experiment itself. This method has lead me to good solution and logical sequence of work since each results lead me to other solutions and hence to new experiments. Another way is to keep the discussion at the end of the report but because I have more than one experiment and some of them has lead me to the next I thought it is better to discuss the results of each experiment right after doing it.

The result of task 13 and 14 is the production of this report.

As I said before, these tasks were done in order to achieve the project objectives and hence to answer the research question stated at the beginning of this report: How do high bit error rates in a wireless network affect the performance of the Snoop protocol and how do these high bit error rates affect the number of duplicate packets sent. The results of my work show how high bit error rates affected the Snoop performance and then proposed solutions for this problem like increasing the number of duplicate packet sent.

6.3 Future work

There are several things that can be considered for future work:

- **Snoop Documentation in ns2:** one important task is to build a proper documentation for Snoop protocol in ns2. According to the authors of current version of Snoop manual Fall & Varadhan (2005) there is no documentation for Snoop in ns2, see under Undocumented Facilities. This task will include understanding the ns2 documentation format and create a section that cover all aspects of running Snoop under ns2 including examples of TCL codes and explanation of the building blocks of the C++ implementation of Snoop (like class hierarchy for example).

- **Implementation of enhancements in real network:** Another area for future work is to implements the enhancement we proposed on a real network since what we did was a simulation on ns2.

- **TCP connection from the mobile host:** our work discusses the design enhancements on Snoop when the data is traveling from the fixed host to the mobile host. A future work would be to investigate the case when TCP connection is initiated from the mobile host to the fixed host.

6.4 Conclusion

The main contribution of this project is that it has shown how to adopt the Snoop protocol to work well over a wide range of error rates. This was done by applying enhancements to the Snoop design and then integrating them with Snoop in one protocol. The enhancements were added to the ns2 implementation of Snoop and we hope they can be added to the new release of ns2.

The results will be of interest to the research community of wireless networks and wireless networks performance, especially people concerned with TCP performance over wireless networks and link layer solutions like the Snoop protocol.