Performance evaluation of ECN-based TCP error discriminator over wireless networks

M. Alnuem, J. Mellor, R. Fretwell

Mobile Computing and Networks Research Group
School of Informatics
University of Bradford
Bradford, UK

Abstract

Some of the promising End-To-End solutions to improve TCP performance over networks suffering from wireless transmission errors are error discriminators. An error discriminator is any technique that tries to differentiate between error types and deal with each type differently. By distinguishing between error types, TCP can avoid unnecessary cuts of congestion window in the case of wireless errors where retransmission of lost packets is enough and there is no need to slow down. One of the techniques for error discrimination is the use of Explicit Congestion Notification (ECN). ECN is a technique to mark packets with a flag whenever congestion is expected. An active queue management (AQM) mechanism that anticipates congestion is located in the middle routers and allowed to mark packets with an ECN flag whenever a congestion is about to happen. In this work we evaluate the performance of a proposal to improve TCP performance over wireless network by using an ECN-based error discriminator at TCP sender and a RED queuing mechanism that sends congestion notifications. The ECN-based error discriminator is used to differentiate between error types based on an ECN feedback from a Random Early Detection AQM placed in the congested node. When an error occurs, if this error was preceded by ECN the error is considered to be a congestion error, and hence TCP reduces it's transmission rate. However, if the error occurs while there is no ECN notification then the error is considered as a wireless error and TCP does not need to reduce it's throughput. A set of experiments were conducted to validate this solution. The results show that ECN is a good predictor of imminent congestion and can be used actively in an error discriminator over networks that suffer from low and medium congestions.

Keywords: Performance evaluation, error discrimination, wireless networks, Explicit Congestion Notification, TCP

1 Introduction

Explicit congestion notification [1] was first introduced to help TCP avoiding congestion-collapses by allowing intermediate nodes to set a congestion notification bit in the IP header whenever a congestion is expected. The TCP sender will respond to this notification by reducing it's transmission rate. An Active Queue management mechanism (e.g. RED[2]) is placed at the congested nodes and become responsible for marking packets when a congestion is expected (in case of RED the packets will be marked when the queue reaches a particular threshold). Using ECN requires changes on the TCP sender and receiver. Also it requires the use of AQM at the congested nodes. However, using ECN does not requires changing the TCP congestion mechanisms since TCP required to respond to ECN in the same way as it respond to a packet drop.

However, in wireless links, errors caused by link failure are high. In case of a wireless error, TCP does not need to reduce it's transmission rate because there is no actual congestion. Since TCP respond to all errors by reducing it's transmission rate, distinguishing between transmission errors and congestion errors is expected to improve TCP performance over error prone links like wireless links.

Dawkins et al. in [3] has proposed the use ECN to improve TCP performance over wireless links by modifying the way TCP responds to ECN. Biaz [4] explains the technique as following: If a drop is detected by receiving duplicate acknowledgments, then we look if we have received an ECN in the near past. If ECN is received before the error happened then this is a strong indication that this error is caused by congestion. This is based on the understanding that, in ECN-capable connections, ECN should always happen before congestion drops. So, if the ECN preceded the drop then TCP consider this drop as congestion drop and act by reducing sender's window size in order to solve the congestion. However, if the drop happens while not

preceded by ECN, it then considered as wireless error and so no need to reduce the sender's window size [4]. However, we still need to retransmit the lost packet. The authors in [3] argue that this approach will improve TCP performance over wireless networks specially those suffers from high wireless error rates.

However, Biaz in [4] studied the visibility of using ECN to distinguish between error types. He argues that this approach is not an accurate method to differentiate between congestion and wireless errors. He showed that the probability that ECN marked packets will come before a congestion error or a wireless errors is approximately the same. He computed the probability based on measuring the time between the occurring of ECN and the occurring of a congestion or wireless error. He found this time interval is the same in average. His conclusion is that, because ECN happens before congestion and wireless errors at the same rate, ECN can not be used to differentiate between error types and hence using ECN as described by Dawkins et al. in [3] will not improve TCP performance.

However, in this following section we discuss Biaz [4] argument and propose extension to it.

Do we need to discriminate between errors under high congestion rates?

As we said before, Biaz in [4] argue that we can not use ECN to improve TCP performance over wireless networks to discriminate between error types and act differently according to each type.

However, let us discuss this argument by making some questions and try to answer them. First, when the probability of having ECN before congestion and wireless errors is the same, does this means that the congestion error is high?. To answer this question let us look first how RED mark the packets with ECN.

RED uses average queue size and a threshold to mark packets. If the average queue size in between the minimum and maximum threshold then RED mark the packets according to probability P. However, if the average queue size exceeded the maximum threshold, RED marks all packets. Following summary of the procedure as presented in Floyd[2]:

For each received packet:

If min-thresh<=avg< max-thresh then mark packet according to probability P

Else

If avg >= max-thresh then mark the packet.

Avg is the average queue size, min-thresh and max-thresh are the minimum and maximum thresholds respectively.

We can see from above algorithm that congestion drops will occur when the average size exceeded the max-threshold. Hence all packets will be marked during this period. If this persists then the only solution is to slow down TCP's transmission rate. We have notice from experimental observation that with the increase in the congestion error rates the number of ECN notification increases; hence the probability of having ECN before wireless errors as well will increase. This will lead us to the next question: Do we need to discriminate between errors if congestion rate is high?. We believe that under high congestion rates there is no point from discriminating between errors since the right action to take in this case is to reduce the transmutation rate in order to solve the congestion.

However, when the congestion rate is low then the probability of having ECN before wireless error will reduce gradually specially with the increase in the wireless error rates. Hence, if we use error discriminator based on ECN it will give positive results especially with the increase in the error rates.

We can conclude the following: We mainly have two cases; one case is when the wireless link operates on a network with light or no congestion. In this case we believe that using ECN to discriminate error types will improve the performance since the number of wireless errors will be much higher than generated ECNs. In this case mistaking wireless error to a congestion error will be rare and hence the unnecessary cuts of TCP window will be small.

The other case is when wireless links operates with a heavily congested wired network. We believe in this case there is no point of discriminating errors (using ECN or any other method) because even if we could discriminate between errors, the priority is to solve the congestion and this can only be done by reducing TCP transmission rate. So we are going to cut the sender's window any way.

In this work we will design a sender based TCP error discriminator that use ECN to distinguish error types. We are going to test the performance of the new protocol in presence of different wireless error rate and in network with light or no congestion. Testing the new protocol in heavily congested networks will be left for future work.

Following we will give a brief description of RED[2] since it perform a basic role the proposed solution. Then we will explain the design of the proposed solution. Finally will present the experiments results and discussion.

2 Related Work

TCP-Error discriminator is a name for all methods that try to understand the cause of the error and make TCP to act differently to each type of error.

Some error discriminators deal with the network as a black box and do not need any feed back from the network in order to discriminate errors. Other type of error discriminators uses help from intermediate networks in order to understand the cause of the error. Following I will talk about both types and I will start with error discriminators that depends on the network to help distinguishing errors and we will call it Network dependent error discriminators.

Network dependent error discriminators: this kind is based at the end-point of the connection but uses help from the intermediate nodes. For example *TCP-Casablanca* and *TCP-Ifrane* by Biaz et al. [5]. Also Biaz et al. in [6] has used information provided by intermediate nods, like information provided by Snoop agents[7], to help error discriminator at the end-point to distinguish errors types see [6]. Another example is the use of congestion notifications provided by the congested nodes (ECN) to discriminate between errors as described in [3].

Network dependent error discriminators: this kind is based at the end host (or one of them) and does not require help from the intermediate nodes. Some of these solutions are based on using congestion predictors like [8-10].The discriminator works by taking input from the congestion predictor about the congestion status. If a loss occurs then if the congestion predictor was predicting a coming congestion then the drop is considered as congestion loss. However if the predictor was suggesting increasing the sending rate because it does not predicts any congestion in the near future then the drop is considered to be caused by link error [11]. Also we must notice that as [11] indicated, designing accurate error predictors is important since mistakes of distinguishing transmission errors from congestion errors could case unnecessary congestion which is usually avoidable by using normal congestion control algorithms [11]. For example if a congestion error is mistaken by the congestion predictor to be as transmission error then TCP will not decrease the window size and this will make the current congestion much worse.

3 TCP Error Discriminator Using ECN

Here we will describe the design and testing of a sender based TCP error discriminator that uses ECN to discriminate between error types. The Idea we propose is to give TCP the ability to discriminate between wireless and congestion errors by using feedback from Active Queue management mechanism (AQM) at the middle nodes that can mark packets when congestion is happening or about to happen. When error occur, an error discriminator at TCP sender will decide if the error is congestion error or wireless error by looking at the feedback received from the AQM mechanism, RED in our case. If RED indicates that a congestion is taking place then the error is considered a congestion error and TCP reduces it's transmission rate. On the other hand, if RED did not mark packets because there is no congestion then the error considered to be wireless error and the sender resend the lost packet without the need to reduce it's transmission rate.

The motivation behind this mechanism (i.e. TCP error discriminator + RED) is that it preserves the end-to-end semantic of TCP because the connection between the sender and receiver is not broken at any point of the path. Also, it uses already used and tested AQM mechanism (i.e. RED) and with the increase deployment of RED in computer networks, this solution will not need any change in the network (changes will be needed only at the TCP sender). For this reason this solutions will need minimum changes on the existing systems and it gives a noticeable improvement of TCP performance over wireless networks as the experiment results show in the next section. Moreover, it is able also to deal with congestions, just like normal TCP, and reduces TCP transmission rate if necessary.

Following we will describe the work of the error discriminator but first we will put some assumptions:

1- In order for this mechanism to work there should be no congestion drops having explicit notification sent to TCP.
2- In case of RED there will be no dropping, instead RED will send ECN whenever it expect that a congestion is about to occur (Usually RED drop packets to avoid congestion, in our case RED will mark the packets instead of dropping them).
3- If the RED queue is full then ECN should be sent to TCP sender (Usually RED mark all packets if the queue average size exceeded the maximum threshold).
4- If there are congestion errors as well as wireless errors then TCP should solve the congestion first by dropping transmission rate (Congestion is given higher priority).
5- RED is required at the bottleneck only.

The TCP implementation used is TCP-Reno. We modify TCP-Reno so it can discriminate between congestion and wireless errors based on the ECN feedback from RED. The changes are done to TCP sender only. It works as following:

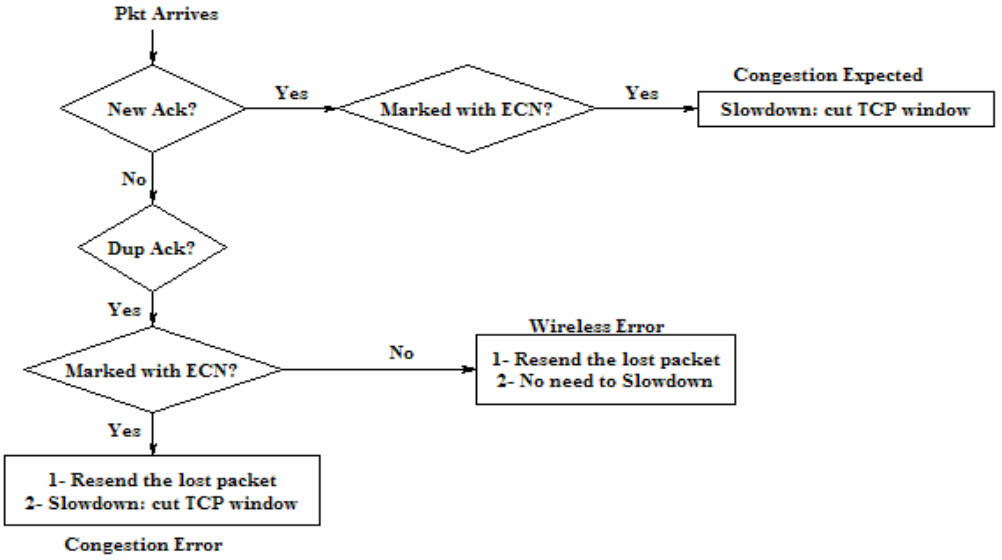

Figure 1: TCP-sender Error Discriminator

- If a packet is received by TCP sender, the discrimination module will read it first. If the received packet is a duplicate acknowledgment then we see: if the packet is marked by RED with an ECN or if we have received a marked packet recently (normally tow previous marked packets is enough) then we consider it to be an indication of a congestion error. In this case we pass the control to TCP-Reno congestion control mechanism.
- If we received a duplicate acknowledgment but with no marked packets in the recent past, then it is considered as a wireless error. In this case we do not call congestion control, instead, we do retransmission of the lost packet and reset the retransmission timeout (RTO).
- In case of timeout we pass the control to the TCP-Reno congestion control.
- If a new acknowledgment is received with an ECN mark, then TCP reduces it's transmission rate to prevent expected congestion.

Figure 1 shows the TCP-sender error discriminator.

4 Multiple losses from the same Window

In our work we have implemented and applied a simple version of the Idea suggested by Floyd et al. in [12] to recover from multiple packet drops. To do that TCP remain in the fast recovery until a new acknowledgment that acknowledge all outstanding packets is received. During that we increases it's congestion window with every duplicate acknowledgment it receives. Also TCP resend the last acknowledge packet +1. Using this approach we have noticed a big improvement in TCP performance particularly in high bit error rates.

In order to limit the congestion created by increasing retransmission we limited the resent of lost packets to one. This way each lost packet will be resent only once before a timeout happens. However the problem of unnecessary retransmission remains for further research.

Figure 2 shows the TCP-sender after adding the multiple packet drop support to the error discriminator.

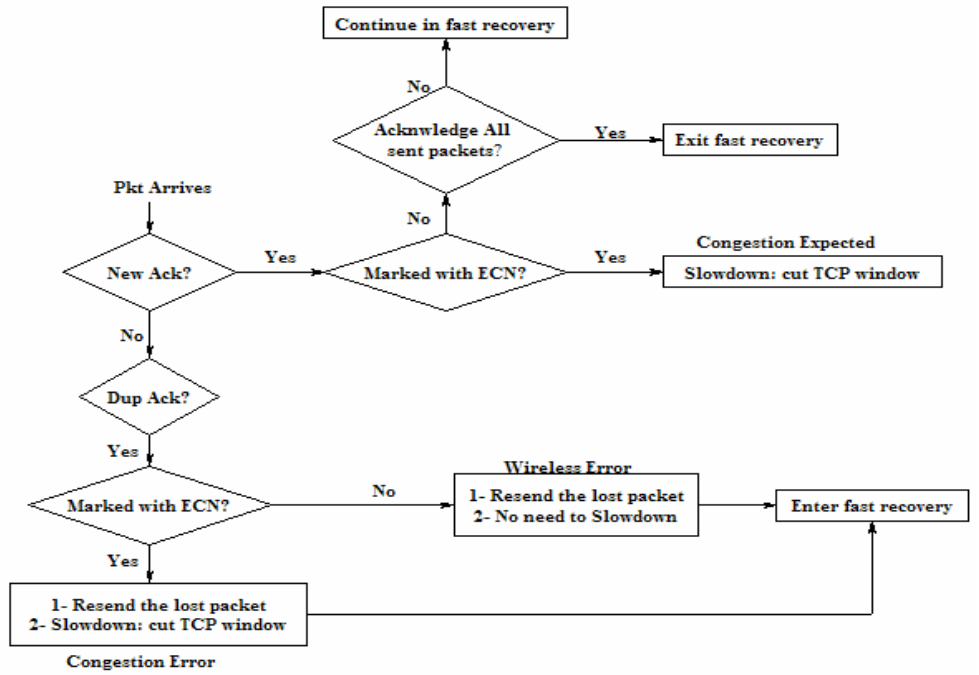

Figure 2: TCP Error Discriminator with multiple packet drop

5 End-To-End Performance of TCP with error discriminator

When a TCP sender receives a new packet, then it is one of three possibilities: New Acknowledgment, Duplicate Acknowledgment or a New/Dup Acknowledgment packet with an ECN (The ECN is send over Acknowledgments). Table 3.2 list all the possible cases for an acknowledgment packet and it's interpretation.

NewAck	DupAck	ECN	Interpretation
0	1	0	Wireless Error
0	1	1	Congestion Error
1	0	0	Normal Ack
1	0	1	Congestion Error Expected

Table 1: All possible cases for an Acknowledgment

The above table (Table 1) can be simplified as following:

1- If ECN flag is 0 then either there is a wireless error or it is a normal acknowledgment and in either case we don't slowdown using Reno slowdown. Instead we just resend the lost packet in case of wireless error.

2- If ECN is 1 then there is congestion or an expected congestion and in either case we have to slowdown.

Following tables shows the actions TCP-Reno and the modified TCP takes when Congestion or/and Wireless errors occur.

TCP-Reno:

	Congestion error	Wireless error	Action
Case1	0	0	Do nothing
Case2	0	1	Slowdown+Resend
Case3	1	0	Slowdown+Resend
Case4	1	1	Slowdown+Resend

Table 2: TCP-Reno reaction to errors

Modified TCP:

	Congestion error	Wireless error	Action
Case1	0	0	Do nothing
Case2	0	1	Resend
Case3	1	0	Slowdown+Resend
Case4	1	1	Slowdown+Resend

Table 3: Modified TCP reaction to errors

As we can notice from the above tables, the only change is when wireless error occur (Case2).

6 End-To-End Performance with No Congestion

Here we will test the performance of the improved TCP in case of small or non congestion. The wireless error rate will vary from low to high error rates during the experiment.

Topology

The experiment was done using ns2[13] and the topology presented in figure 3.

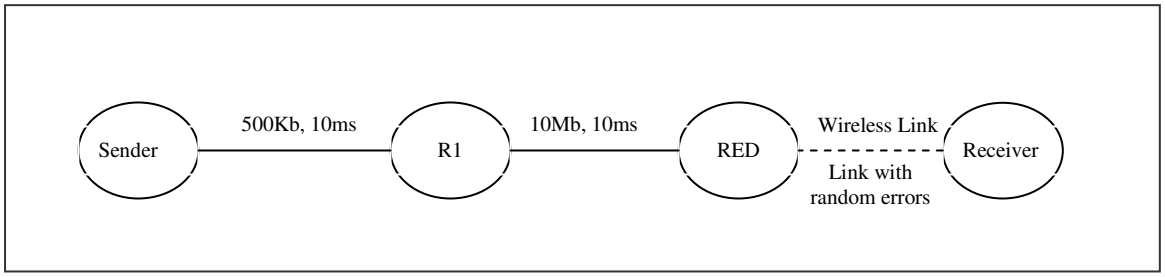

Figure 3: Network Topology

The bandwidth is increasing in the way from the sender to receiver so we insure that only wireless errors occur and no congestion errors. We need that to show how the new improvements increase the performance of TCP in presence of wireless errors.

The Errors were added to the link using the Default *ErrorModel* class provided by ns2. The ns2 documentation has given a full explanation of how to add errors to wired and wireless links. For more information see chapter 13 of ns2 documentation edited by Fall and Varadhan[14]. We have used the TCL script provided in the ns documentation to add errors in our topology.

Following we will present the experiment results and discussion.

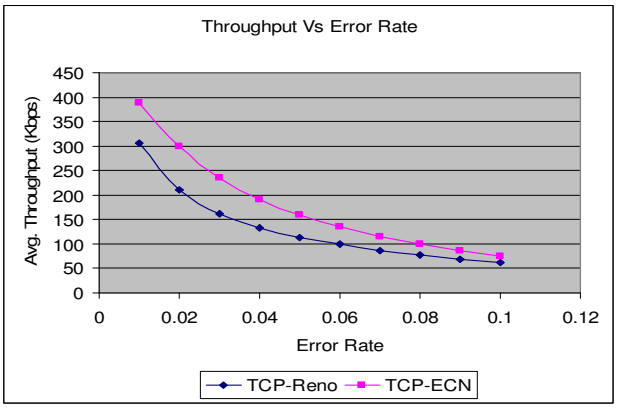

Figure 4: TCP-Reno VS TCP-ECN – Low error

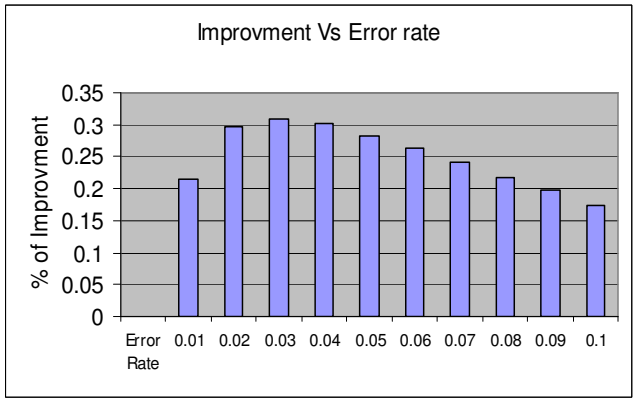

Figure 5: % Improvement –TCP-Reno VS TCP-ECN

7 Results and Discussion

I we will refer to the improved TCP as TCP-ECN. An experiment has been done using the above topology in ns2. The results shows following:

Low wireless error rates

We can see that the performance of TCP-ECN is higher that TCP for error rates from 1% to 10%. As we can see also, the improvement reaches a peak and then starts to slow down. The initial observation indicates

that the reason of the degradation in the improvement with the increase of the bit error is the RTO (Retransmission timeout). With the increase in the error rate more packets are lost and hence more probability to have a timeout. Moreover, several timeouts in a raw will lead to more waiting time before resuming transmission since TCP double the timeout after each time the RTO clock expires. Table 5, figure 4 and figure 5 show the improvement in the performance at each error rate

Error Rate	TCP-Reno Avg. Throughput	TCP-ECN - Avg. Throughput	improvement
0.01	305.5831	389.02816	0.214496
0.02	210.4549	299.50502	0.297324
0.03	162.6884	235.11761	0.308055
0.04	133.4648	191.05589	0.301436
0.05	113.876	158.76625	0.282745
0.06	99.25327	134.59256	0.262565
0.07	87.51342	115.29034	0.24093
0.08	77.88032	99.516184	0.21741
0.09	69.35052	86.4562	0.197854
0.1	61.68463	74.757437	0.17487

Table 5: % Improvement – TCP-ECN Vs TCP-Reno

Each value is the mean of throughput received per second. We run T-test on the samples and the mean lies in the 90% confidence interval.

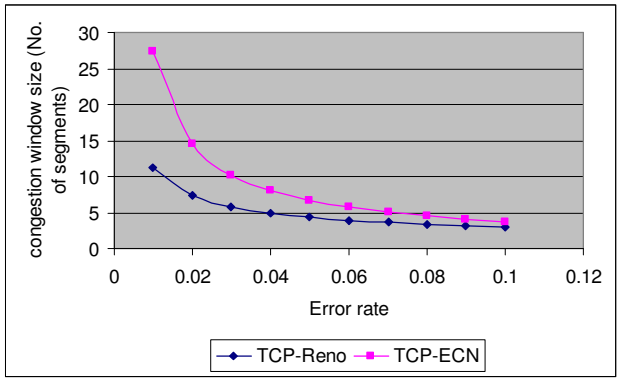

Figure 6: Congestion window size for TCP-Reno and TCP-ECN

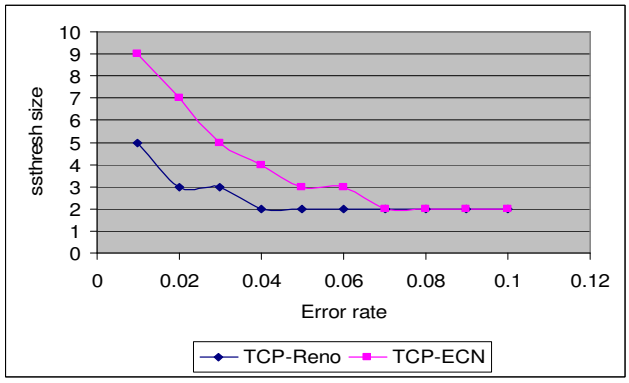

Figure 7: ssthresh size for TCP-Reno and TCP-ECN

A suggestion that we will investigate is to make TCP-ECN to increase the RTO with the increase in the BER hence preventing TCP from having timeout.

The average congestion window and ssthresh has improved, see figure 6. and figure 7 respectively.

High wireless error rates

Under higher bit error rates the improvement decreases until it reaches 0% under 40% error rate see figure8. We can explain this behaviour under high error rates as following: TCP-ECN prevent unnecessary slowdown in case of wireless error by preventing unnecessary cuts of congestion window because of receiving duplicate acknowledgments caused by wireless error. However, if a timeout occur, cutting the congestion window is unavoidable because timeout happens only if there is no feedback (i.e. Acknowledgments) received from the receiver. With no acknowledgments being received, we can't discriminate between errors using ECN since ECN comes over acknowledgment. A suggestion to over come this problem is to allow RED to send backward ECN. However, if these backward ECN will be sent over acknowledgments also, the problem will remain. So, we suggest sending backward ECN using dedicated packets for this reason. We will study the visibility of this suggestion and it's impact on the network overhead during the future work.

8 End-To-End Performance with Congestion

Here we repeated the previous experiment after adding more sources to create moderate congestion. Figure 6 shows the used topology.

Low wireless error rates

The results show that the TCP-ECN improves the performance also with presence of moderate congestion, see figure 11 and figure 12. However, unlike where there is no congestion, with presence of congestion we can notice that the improvement increases with the increase in the wireless error rate. We can explain that as following: with the increase in the error rate number of wireless losses that coexists with ECN notification becomes lower. This will lead to reduce the number of cases where the error discriminator will confuses a wireless error as a congestion error and hence the discrimination process will give better results.

High wireless error rates

When working under error rates higher that 10%, the improvement decreases dramatically until it reaches 0 % under 40% error rate. Figure 13 show these results. These results are very similar with the results we have got in case of no congestion. The reason for that is with high error rates major factor that affect the performance in both cases is the timeout.

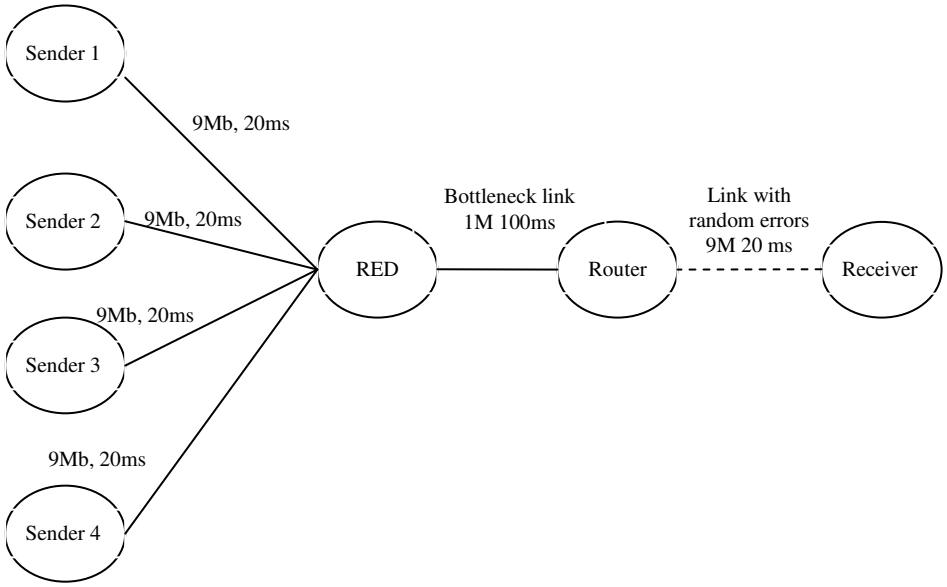

Figure 10: Network Topology, moderate congestion with wireless errors.

9 Validation of the simulation results

We applied an analytical module provided by Padhye et al. in [15] in order to validate the experimental results,. This mathematical module computes TCP throughput in presence of errors taking into account the factors that affects the performance. The module contains two parts one to compute the effect of congestion window cuts on the performance and another to compute the timeout effect on the performance. We use the module to compute the expected results and compare them with what we have got. Biaz et al. in[4]gave a general explanation for how to use the module to compute the performance after adding wireless errors.

The formula as presented by Padhye et al. in [15] is :

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1,3\sqrt{\frac{3bp}{8}}\right) p\left(1+32p^2\right)} \quad (1)$$

In this part the effect of congestion window cuts in the performance is calculated. The *RTT* is the average round trip time, *b* is number of packets acknowledged per each acknowledgment and *p* is the error rate.
The second part considers the effect of timeout on TCP-Reno performance:

$$T_0 \min\left(1,3\sqrt{\frac{3bp}{8}}\right) p\left(1+32p^2\right) \quad (3)$$

In formula (3) T_0 is the first timeout occurred during the connection. We use formula (1) to compute the performance of TCP after introducing the wireless and congestion errors. Biaz et al. [4] explain how to introduce both error types as following : we will consider wireless error to be p_w and the congestion error to be p_c. Then the error rate *p* in formula (1) is computed as following $p = p_w + p_c$.

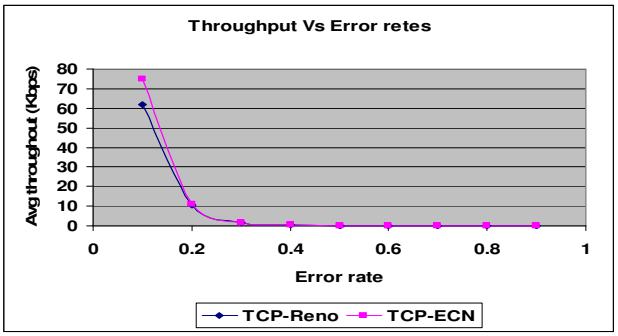

Figure 8: TCP-Reno Vs TCP-ECN – High error rates

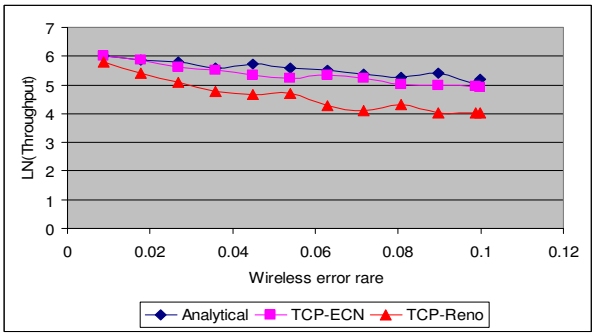

Figure 9: comparison of expected improvement and improvement by TCP-ECN

However, TCP-ECN prevent window cuts in case of wireless errors so in formula (2) the only error rate affect the window size is p_c, p_w has no effect on formula (2). The formula (2) will be as following:

$$RTT\sqrt{\frac{2bp_c}{3}} \quad (4)$$

However, both wireless errors and congestion errors can cause timeout in TCP-ECN so we will include both error types in formula (3):

$$T_0 \min\left(1,3\sqrt{\frac{3b(p_c+p_w)}{8}}\right)(p_c+p_w)\left(1+32(p_c+p_w)^2\right) \quad (5)$$

Finally the formula to compute the expected performance of TCP-ECN is:

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2bp_c}{3}} + T_0 \min\left(1,3\sqrt{\frac{3b(p_c+p_w)}{8}}\right)(p_c+p_w)\left(1+32(p_c+p_w)^2\right)} \quad (6)$$

The formula (6) were used to compute the expected throughput of a TCP-ECN . We then use formula (1) to compute the expected throughput of TCP-Reno. The improvement as a result of applying this formula found very close to the improvement we got from using TCP-ECN in low or no congestion. In figure 3.5 we show the logarithmic throughput in packets/second for both the TCP-ECN and the expected throughput calculated using formula (6) (named Analytical in the chart) and as we can see they are very close. All these experiments and analytical calculation were done assuming a little congestion rate of 0.0008 which is equivalent to 1 dropped packet every 1250. The validation for the simulation results in case of medium congestion is left for future work.

10 Conclusion and Future work

In this work we have tested the performance of a proposal to improve TCP performance over wireless networks [3]. The proposed solution suggests that TCP uses feedback from congested nodes to discriminate between congestion and wireless errors and act differently in response to each type of errors.

In our work we showed that explicit congestion notification can be used to improve TCP performance over network suffering from different rates of wireless errors. A new version of TCP, TCP-ECN, has been designed and tested using network simulator ns2[13]. TCP-ECN use an error discrimination module to discriminate between error types based on the feedback from the congested nodes. The results showed that TCP-ECN outperforms TCP-Reno in networks suffering from different wireless error rates and low or moderate congestion. The initial observations showed that discriminating between error types in case of high congestion is not helpful because in case of high congestion the priority is to resolve the congestion by reducing the sender transmission rate.

As a future work we will give RED more study to see how to improve RED to give better performance when interact with wireless environment. One proposal is to make automatic update of the minimum and maximum threshold with different number of flows. For example we want to investigate if with the increase in number of flows increasing the minimum threshold will increase the performance because less ECNs will be generated and hence less cuts to the congestion window. Another proposal is to test if we can improve

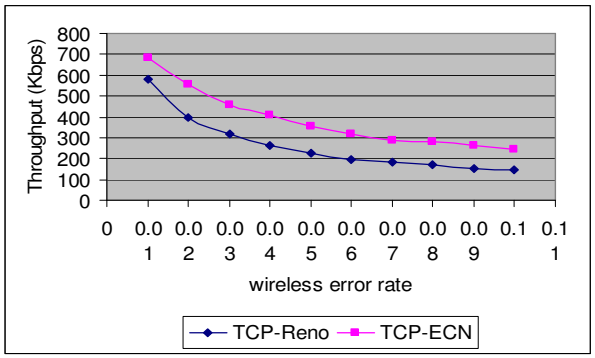

Figure 11: TCP-Reno Vs TCP-ECN – Low error rates

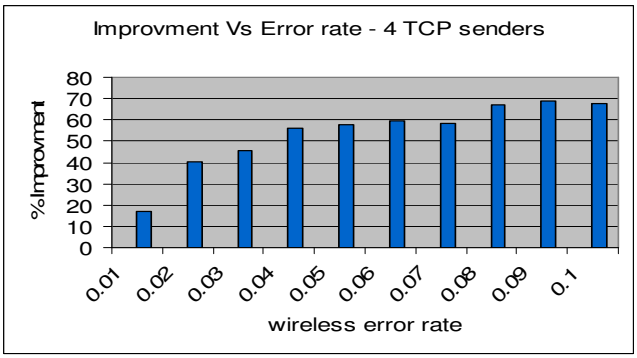

Figure 12: : % Improvement – TCP-ECN Vs TCP-Reno

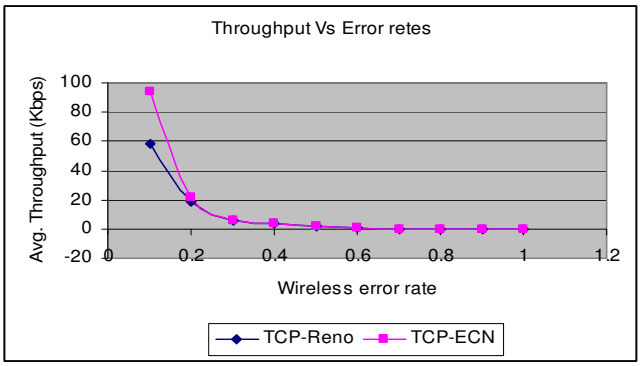

Figure 13 :TCP-Reno Vs TCP-ECN – High error rates

RED by making it to monitor the direction of the queue and to generate ECN based on the direction. For example ECN will be generated if the queue direction is "up" meaning that the queue is building up. However, if the queue is decreasing, even if the average queue size is more than the min-threshold, then less, or none, ECN notifications are generated. Considering the queue direction will give TCP sender a clearer image about the congestion and how it is going.

References

[1] Floyd, S., *TCP and Explicit Congestion Notification.* ACM Computer Communication Review, 1994. **24**(5): p. 10-23.

[2] Floyd, S. and V. Jacobson, *Random early detection gateways for congestion avoidance.* IEEE/ACM Trans. Netw., 1993. **1**(4): p. 397-413.

[3] Dawkins, S., et al., *Ongoing TCP Research Related to Satellites.* 2000.

[4] Biaz, S. and X. Wang, *Can ECN Be Used to Differentiate Congestion Losses from Wireless Losses?* 2004.

[5] Biaz, S. and N.H. Vaidya, *"De-randomizing" congestion losses to improve TCP performance over wired-wireless networks.* IEEE/ACM Trans. Netw., 2005. **13**(3): p. 596-608.

[6] Biaz, S. and N.H. Vaidya, *Heterogeneous data networks: congestion or corruption?* 1999. p. 124.

[7] Balakrishnan, H., et al., *Improving TCP/IP Performance Over Wireless Networks.* In Proc. 1st ACM int'l Conf. on Mobile Computing and Networking (Mobicom), 1995.

[8] Wang, Z. and J. Crowcroft, *A new congestion control scheme: slow start and search (Tri-S).* SIGCOMM Comput. Commun. Rev., 1991. **21**(1): p. 32-43.

[9] Jain, R., *A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks.* SIGCOMM Comput. Commun. Rev., 1989. **19**(5): p. 56-71.

[10] Brakmo, L.S., S.W. O'Malley, and L.L. Peterson. *TCP Vegas: new techniques for congestion detection and avoidance*. in *Proceedings of the conference on Communications architectures, protocols and applications*. 1994. London, United Kingdom: ACM Press.

[11] Biaz, S. and N.H. Vaidya. *Distinguishing congestion losses from wireless transmission losses: a negative result*. 1998.

[12] Floyd, S. and T. Henderson, *The NewReno Modification to TCP's Fast Recovery Algorithm.* RFC2582, 1999.

[13] ns2, *The Network Simulator.* 2006. http://www.isi.edu/nsnam/ns.

[14] Fall, K. and K. Varadhan, *The ns Manual (formerly ns Notes and Documentation).* The VINT Project,Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC., 2006. http://www.isi.edu/nsnam/ns.

[15] Padhye, J., et al., *Modeling TCP Reno performance: a simple model and its empirical validation.* IEEE/ACM Trans. Netw., 2000. **8**(2): p. 133-145.