Distinguishing Congestion and Corruption Losses: A Negative Result *

(Preliminary Version Under Revision)

Saad Biaz Nitin H. Vaidya

Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
E-mail: {saadb,vaidya}@cs.tamu.edu
Web: http://www.cs.tamu.edu/faculty/vaidya/mobile.html

Technical Report 97-***

August 9, 1997

Abstract

On wireless links, the rate of corruption losses can be significant, leading to poor TCP performance. The performance gets worse when these losses are mistaken for congestion losses, unduly triggering the TCP congestion control algorithms. To avoid this, techniques to distinguish between corruption and congestion losses without any explicit information from the network (routers or switches) are of interest.

In the past, several proposals require TCP sender to reduce its window size when congestion is detected. These schemes use heuristics to detect congestion by using some simple statistics on round-trip delays and/or throughput.

If the heuristics developed in the past are good (i.e., accurate much of the time), then one possible mechanism for distinguishing between errors and congestion are as follows: (a) Use a good heuristic that asks TCP sender to reduce window size when congestion is

*Research reported is supported in part by Texas Advanced Technology Program grant 009741-052-C, National Science Foundation grant CDA-9529442 and the Fulbright Program.

detected. (b) If a packet loss occurs, see what the heuristic said just before the packet was sent. (c) If the heuristic had said reduce window (because congestion was detected), assume that the packet loss is due to congestion, otherwise assume that packet loss is due to transmission error. Take appropriate action depending on the nature of packet loss.

The above scheme will work well, if the heuristic is very accurate. Unfortunately, our preliminary measurements suggest that three such heuristics proposed previously do not perform well in practice. The reason, essentially, is that to a well-behaved TCP connection, packet losses seem to appear almost random, without much correlation to the window size or round-trip delays. This is true (and intuitive) when an individual connection represents only a small fraction of load at a router on the path.

Key Words: TCP – Congestion Avoidance – Packet losses

1 Introduction

The TCP protocol can adapt to very different network bandwidth and conditions. But there still exists the problem of recovering from losses other than congestion. All losses are all assumed, by TCP, to be congestion losses [9] and TCP reacts to all losses with congestion control mechanisms. On wired networks, this assumption is good. But on wireless links, losses occur more frequently due to corruption or for reasons other than congestion [3, 4]. This can lead to a very poor performance. *Fast Recovery* and *Fast Retransmit* [10, 14] alleviate the problem but do not solve it completely. It is then natural to try to distinguish congestion from corruption losses. This can be done by receiving explicit messages from the routers when congestion occurs or is imminent [12, 8]. This approach has two main drawbacks : first, there may be an additional message overhead, and second, the explicit scheme is protocol-dependent. The second approach would be to consider the network as a black box and try to collect implicit information on the state of the network. There is one fundamental difference between corruption losses and the congestion losses: a user, at the transport layer, does not have and cannot have any influence on the losses due to corruption. In other words, a user may be able to provoke congestion losses, but not corruption losses.

This second approach which considers the network as a black box was introduced by Jain in the excellent paper [11]. Jain observes the changes in the round trip delay resulting from changes in the congestion window size. Jain used this approach to develop a congestion avoidance scheme. But there is a natural extension of this approach to develop a tool to distinguish a corruption from a congestion loss. The task seems easier in the sense that you do not have to exactly predict congestion. You have *only* to diagnose the source of a loss. The idea is to look at what a congestion avoidance scheme would have done just before the loss : if the appropriate action was "decrease the window size" then the loss should be with a high probability a congestion loss. Wang [16] and Brakmo (Vegas) [6] look at changes in the throughput to detect congestion. We study the criteria developed by these researchers. While they have pointed out limitations of their schemes under real network conditions, there does not appear to be any work (that we are aware of) which shows how good or how bad these criteria behave on a real network.

We measured the changes of congestion losses rate versus congestion window variations and also evaluated Jain's, Wang's and Vegas criteria through measurements on TCP connections. In Section 2, we present the tools and techniques to perform our measurements and the parameters measured. The experiments performed are described in Section 3.The results on the congestion predictors are presented in Section 4 Section 5 is dedicated to the results obtained. We interpret these results in Section 6.

2 Measurements

The measurements were done using Free BSD. We modified *tcp_debug* to collect the information. This data was processed with a modified version of *trpt* [14].

2.1 Modifications to tcp_debug and trpt

The main function in *tcp_debug* is the function *tcp_trace*.

This function can be called from many other tcp procedures (e.g tcp_input, tcp_output, tcp_usrreq,..). When called, *tcp_trace* records the time (in ms), the kind of call (user, input, output, drop..) and the tcp control block in the array *tcp_debug*. Since the tcp control block takes too much space, we log it for all events except when sending. When a packet is sent, we log a smaller structure which mainly contains the time of transmission (in ms), the packet sequence number and the congestion window. At the end of a connection, we have in the memory all information which can then be processed or just stored on disk.

We modified the *trpt* program to process the information and extract some parameters.

2.2 Parameters measured

- V_m, RTT_m : For each window sent, a packet P_m is monitored. We log the time P_m is sent and the volume V_m (in bytes) sent until P_m is acknowledged. When packet P_m is acked, we compute the round trip time RTT_m for P_m. V_m is most of the time close to the congestion window. The difference comes from the fact that V_m is always a multiple of the *mss* (maximum segment size) while the congestion window is not.

- $Thgt_m$: Throughput $Thgt_m$ is computed as $\frac{V_m}{RTT_m}$. Of course, if the monitored packet P_m has to be retransmitted, then we do not log the measurements for it. But, we log the fact that a packet is lost.

- Information log : For every monitored packet, we have then its round trip time RTT_m, the congestion window size $Cwnd_m$ and the throughput $Thgt_m$. When a loss occurs, we log the window size used to send the last unacknowledged packets.

- N_i and L_i : Since the acks in TCP are cumulative, when a timeout occurs or some dupacks are received, we do not know exactly the number of packets lost. We know only that *at least one packet is lost.* With the data collected, we can count the number of windows N_i which were sent with a given window of size i. We also know the number of losses L_i which occurred when sending with a given window size i.

- P_i and q_i : For each window size i, we can compute the probability P_i that at least one packet was lost. P_i is equal to $\frac{L_i}{N_i}$. If q_i is the probability of loosing a given packet at window size i, then $P_i = 1 - (1 - q_i)^i$, assuming independent packet losses. Clearly, packet losses are not independent of each other. However, q_i does provide a measure to compare the rate of packet losses for a given window size.

- FDL_i : To investigated how congestion loss probability correlates with congestion window size, we calculated the Frequency Distribution of Losses FDL_i, defined as the fraction of packet losses occurring at congestion window size of i MSS (maximum segment size) – in fact, the congestion window size $Cwnd$ is not always an integral multiple of the mss (Maximum Segment Size). For the purpose of calculating FDL_i, we consider a window of size $Cwnd$ to be of size $\lfloor \frac{Cwnd}{mss} \rfloor$ MSS.

- We also calculated the Frequency Distribution of congestion Window size FDW_i defined as the fraction of windows sent with a size of i MSS.

The FDL and FDW measurements should reveal if losses occur more frequently at a given window size or for windows larger than some threshold. Besides these measurements, it is natural to think that when a loss occurs, we can use some criterion developed for congestion avoidance to *diagnose* the source of the loss. If these criteria are predicting congestion, we could then identify the loss as a congestion loss, otherwise, we identify it as a corruption loss.

Jain's [11], Wang and Crowcroft [16] and Vegas [6] criteria were devised for congestion avoidance : these schemes try to optimize the network usage while avoiding congestion. These schemes are based on the idea that there will be some response from the network to a congestion window size change. The response is measured as a function of round-trip times and/or throughput. Jain bases his approach on changes in the round trip delay as a response to congestion window variations. Wang and Crowcroft base their approach on the changes in throughput as response to congestion window variations.

The idea of monitoring round trip time changes due to window variations was suggested by Jain [11]. The idea stems from the relation between the load on the network and the observed throughput. If the load is low, throughput will be low. Suppose that you put only one packet at a time on the network. In this case, you send a packet and wait for the acknowledgement. You will spend much of your time waiting. Hence, throughput will be low.

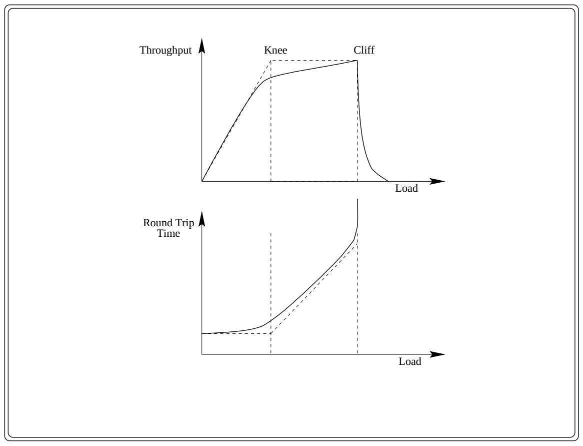


Figure 1: Throughput and RTT vs Network load

On the other hand, if you put many packets in the network, you could almost spend all of your time sending without any wait. The throughput will be then high. But, if you put too many packets, you can overflow some intermediary buffer node, resulting in packet loss. In this case, you have to retransmit the lost packets. Thus, your throughput decreases .

So, in some interval, if you increase the load, you will get a significant increase in your throughput. But, at some point, you reach the limit of the network capability (buffers, computation power of the switches ...). You reach what is called the knee. From this point, if you increase the load, your gain in throughput is smaller than in the previous phase. If you continue the load increase, the throughput reaches the cliff where you experience a collapse of the performance ("severe" congestion).

With the information collected, Jain's criterion (J), the Wang and Crowcroft's Normalized Throughput Gradient (NTG) and the Vegas criteria are computed easily.

Jain's criterion is as follows [11]:

$$J = (W_i - W_{i-1}) * (RTT_n - RTT_{n-1})$$

where W_i and W_{i-1} are respectively the current and the previous congestion window sizes. RTT_i and RTT_{i-1} are respectively the current and the previous round trip times. J's sign is used to increase or decrease the window size : if $J > 0$, the congestion window size is decreased, otherwise it is increased. Jain proved that this approach is valid for deterministic network, i.e., for networks where service time in the switches is not random. This assumption is not true for real networks. We did not find any work which evaluates Jain's criteria under real network conditions. So it was not clear as to what extent Jain's criteria is good/bad under real network conditions.

Wang's Normalized Throughput Gradient NTG is defined as [16]:

$$NTG = \frac{T(W_i) - T(W_{i-1})}{T(W_1)}$$

where $T(W_i) = W_i/RTT_i$ is the throughput of the connection with the i^{th} window, whose size is denoted as W_i. Hence, $T(W_1)$ is the throughput with the first window of size one packet. The congestion avoidance scheme is based on evaluating how much throughput increase is obtained from a congestion window increase. With this technique, the TCP connection tries to detect the flattening of throughput increase. Every RTT, the congestion window is increased by one packet. If the difference in the throughput is less than $\frac{T(W_1)}{2}$, then the congestion window is decreased. Using NTG, if $NTG < \frac{1}{2}$ then decrease the window size.

In Vegas [6], Brakmo *et al.* maintain a variable $BaseRTT$ which is the minimum of RTT's measured during the connection. $BaseRTT$ allows the authors to compute the *Expected Throughput* which is given by :

$$Expected\ throughput = \frac{Current\ Window\ Size}{BaseRTT}$$

The *Expected throughput* changes only if a packet experiences a smaller RTT than the current $BaseRTT$. Every window, Vegas computes the *Actual Throughput* which is equal to $\frac{CurrentWindowSize}{CurrentRTT}$ (one packet per window is monitored). The difference $Diff = Expected - Actual$ is then compared to two thresholds α and β with $\alpha < \beta$. If $Diff < \alpha$ then Vegas increases the congestion window. If $Diff > \beta$, Vegas decreases the congestion window size. Otherwise, if $\alpha < Diff < \beta$ then the congestion window size is left unchanged.

Measurements

To use above criteria as congestion "predictors", whenever the criterion results in the conclusion "decrease the congestion window size", this means that congestion will occur if you continue to increase the congestion window size. We measured two parameters for each criteria.

- Accuracy of Prediction AP computed by dividing the number of times the predictor said "decrease the congestion window" just before a loss occurred by the total number of losses.

- Frequency of Congestion Prediction FCP computed by dividing the number of times the predictor said "decrease the congestion window" by the total number of predictions.

One expects a good predictor to have significantly greater AP, as compared to FCP. That is, a large accuracy (AP) is not necessarily good, if the predictor always says "decrease the window size" (i.e., also large FCP).

3 Experiments

We establish two simultaneous connections from two different hosts ravel.cs.tamu.edu and verdi.cs.tamu.edu to the same discard server (TCP port 9). We run one set of tests with the host "Daedalus.crosslink.net" as a destination and another set of tests with the host "all-purpose-gunk.near.net" as a destination. The discard servers on these two machines have a receive window limited to 32 Kbytes. "Daedalus.crosslink.net" is at 13 hops and "all-purpose-gunk.near.net" is at 19 hops.

On ravel.cs.tamu.edu, we run TCP-Reno. On verdi.cs.tamu.edu, we run TCP-Reno with a slight modification in the congestion avoidance algorithm. When TCP-Reno leaves the slow-start phase, it increases the congestion window size by $\frac{mss^2}{cw}$ after each acknowledgement where mss is the maximum segment size and cw is the current congestion window size. On verdi.cs.tamu.edu, we modify the rate of increase: we increase by $\alpha * \frac{mss^2}{cw}$ with α taking the values from 0.5, 1, 2, 3, 5 and 10 in different experiments. With the variation of α, we want to investigate how a more aggressive increase policy affects congestion losses.

For each set of measurements, we establish two simultaneous connections from ravel and verdi to the same destination and send the same amount of data (1.5 MBytes or 5 MBytes). Each set of measurements consists of 25 connections from "ravel.cs.tamu.edu" and "verdi.cs.tamu.edu" to the same destination. When the destination is "daedalus.cs.tamu.edu", the connections are initiated every two minutes: on the average, a transfer of 5 MBytes takes less than 1 minute (on the average 40 s). When the destination is "all-purpose-gunk.near.net", the connections were initiated every 5 minutes: a transfer of 1.5 MBytes takes less than 5 minutes (on average 3 minutes).

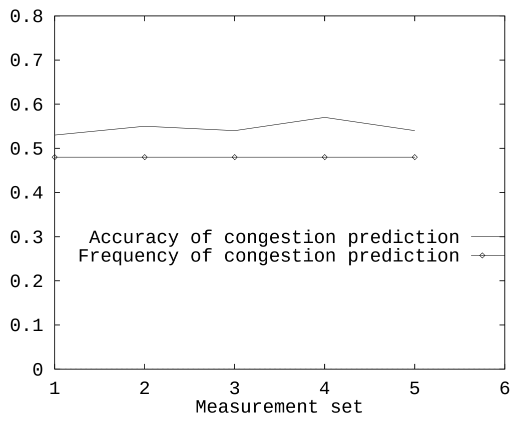
4 Results for the congestion predictors

In this section, we present two sets of measurements with "all-purpose-gunk.near.net" and "Daedalus.cross.link" as the destinations, respectively. We present the Accuracy of Prediction (AP) and the Frequency of Congestion Prediction (FCP) for Jain's, Wang's and Vegas criteria. The results presented here are averaged over all runs. Recall that ravel is running normal TCP-Reno during all connections, while verdi had a different increment of the congestion window size depending on the parameter α taking the values 0.5, 1, 2, 3, 5 and 10.

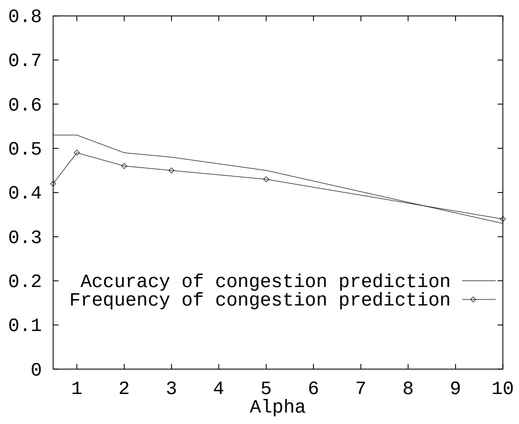
We observe that the Accuracy of Prediction (AP) is typically a little higher than the Frequency of Congestion Prediction (FCP). But, both curves have the same patterns. This means that higher the frequency of congestion prediction, higher is the accuracy of prediction.

Note that a random predictor ("coin tossing") with frequency of congestion prediction of x will give an accuracy of prediction of x. For the three predictors we evaluated, AP is only marginally better than FCP. Thus, these predictors do not perform much better than a random predictor.

In Figures 2 through 7 the plot on left(a) corresponds to host ravel, and plot on right (b) corresponds to host verdi. Figures 2, 3 and 4 present results for Jain, Wang and Vegas, respectively, with the destination being "all-purpose-gunk.near.net". Figures 5, 6 and 7 present results for Jain, Wang and Vegas, respectively, with the destination being "Daedalus.crosslink.net". For Daedalus, the measurements are averaged overs runs that use α values 1, 2, 3, 5 and 10.

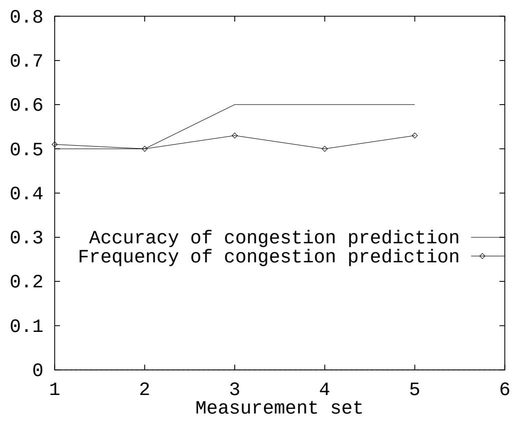


(a) Jain's predictor on Ravel

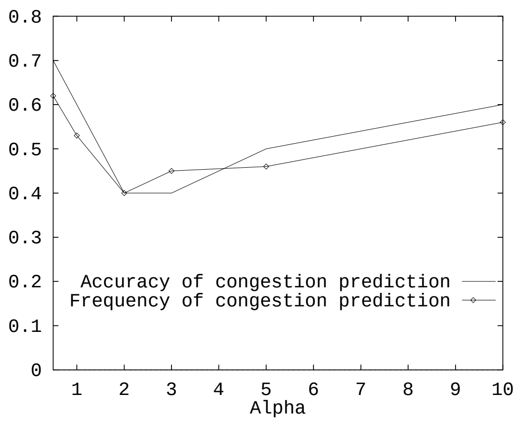


(b) Jain's predictor on Verdi

Figure 2: Jain's predictor with destination : all-purpose-gunk.near.net

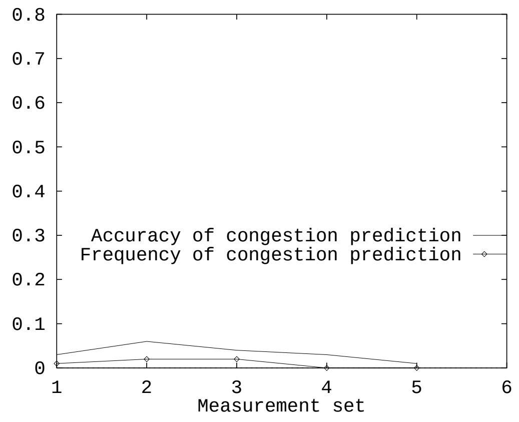


(a) Wang's predictor on Ravel

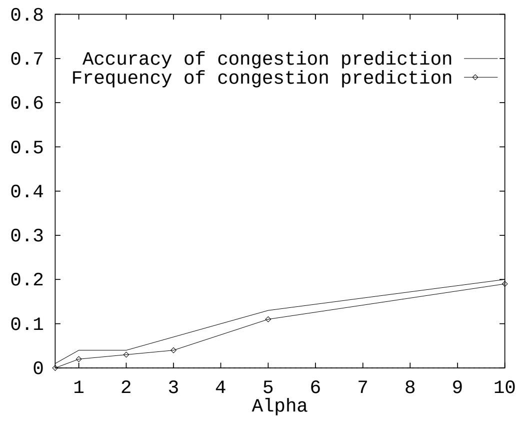


(b) Wang's predictor on Verdi

Figure 3: Wang's predictor with destination : all-purpose-gunk.near.net

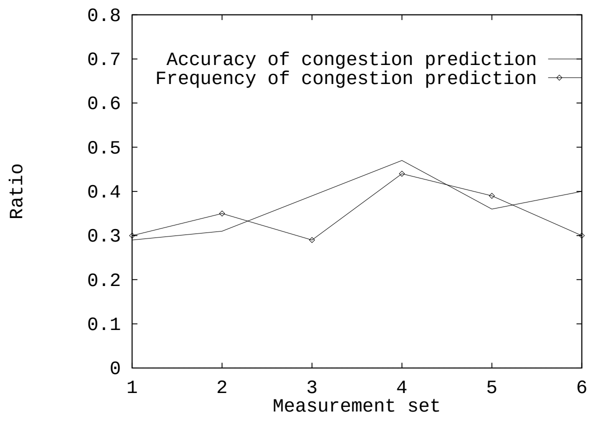


(a) Vegas predictor on Ravel

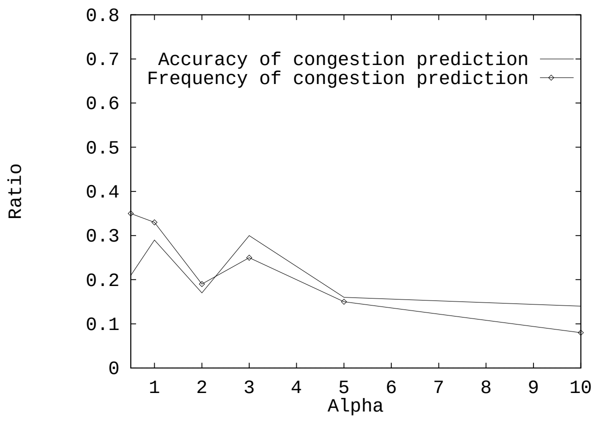


(b) Vegas predictor on Verdi

Figure 4: Vegas predictor with destination : all-purpose-gunk.near.net

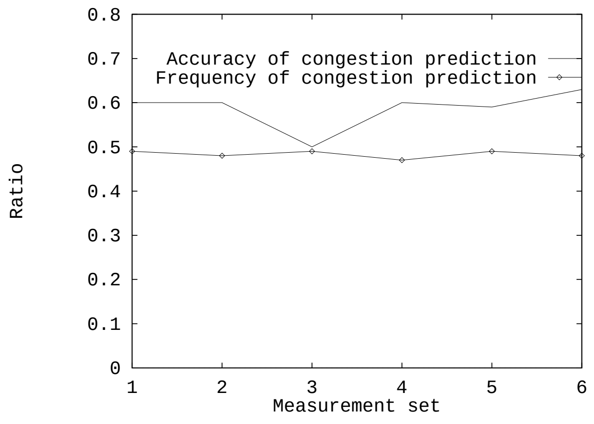


(a) Jain's predictor on Ravel

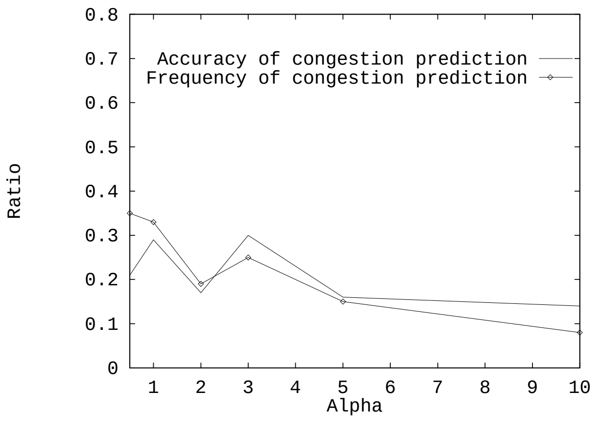


(b) Jain's predictor on Verdi

Figure 5: Jain's predictor with destination : Daedalus.crosslink.net

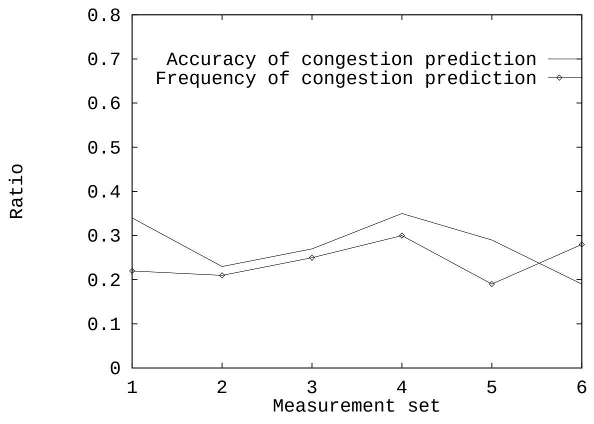


(a) Wang's predictor on Ravel

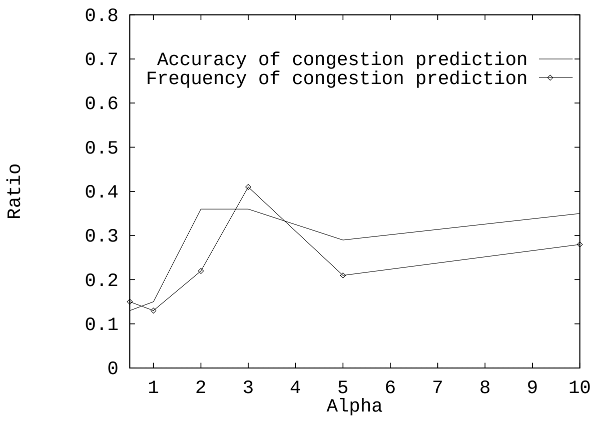


(b) Wang's predictor on Verdi

Figure 6: Wang's predictor with destination : Daedalus.crosslink.net



(a) Vegas predictor on Ravel



(b) Vegas predictor on Verdi

Figure 7: Vegas predictor with destination : Daedalus.crosslink.net

5 Results: losses versus window size

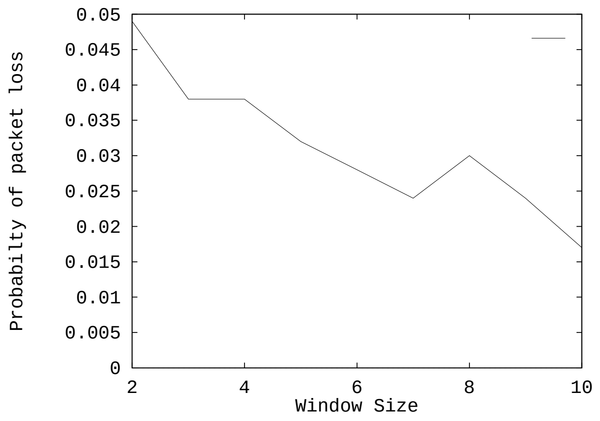
Congestion was so low from our hosts to "Daedalus.crosslink.net" (4 losses on the average for a 5 MBytes transfer) that the frequency distribution of losses can not be used with much confidence. We present here results only for the connection to "all-purpose-gunk.near.net. However, the measurements for "daedalus.crosslink.net" also follow the same general pattern as the measurements for "all-purpose-gunk.near.net".

Since the connections from ravel.cs.tamu.edu are all using normal TCP-Reno, we averaged all the runs (6 x 25 connections) and present a synthesis of the results in one subsection. It must be noted that the 6 sets of measurements were taken at different times of the day. Despite this difference, the results with respect the frequency distribution losses FDL or of the windows size FDW, or probability of losses versus window size are similar.

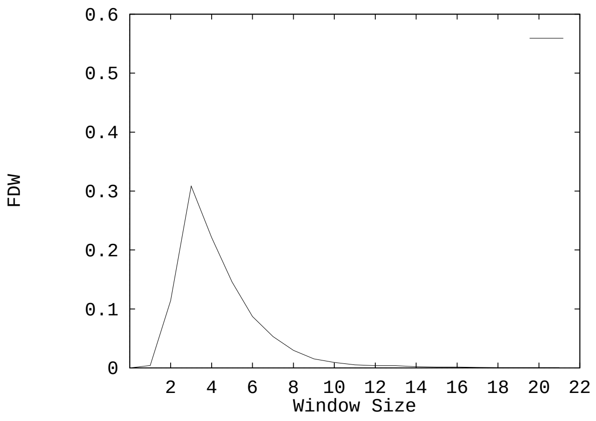
For "verdi.cs.tamu.edu", we present the results for six values of α in a different subsections. Recall that α is the rate of congestion window increase in the congestion avoidance phase on verdi.cs.tamu.edu.

5.1 Ravel

Figure 8 presents (a) the probabilty of packet loss q_i for congestion window of size i and, (b) frequency distribution of window size FDW.



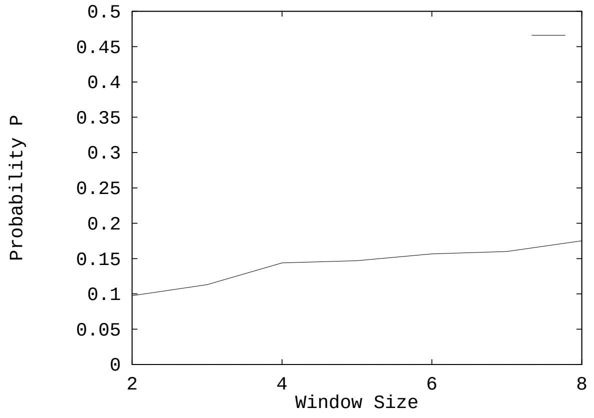
(a) Probability of packet loss



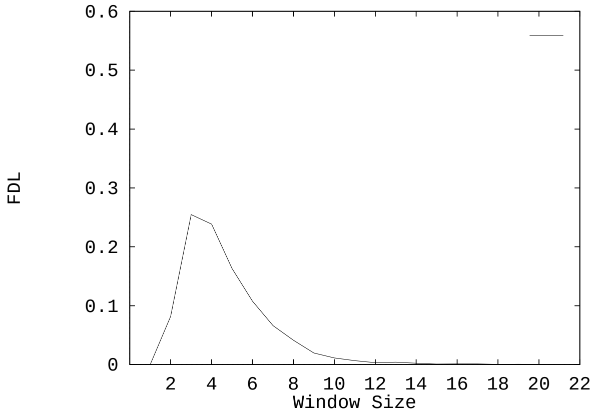
(b) Frequency distribution of Window sizes (FDW)

Figure 8: q_i and FDW

Observe that, for these connections, probability of packet loss is slightly decreasing when the congestion window size increases.



(a) Probability P_i



(b) Frequency distribution of losses(FDL)

Figure 9: P_i and FDL

In Figure 9, we compute then the probability $P_i = \frac{L_i}{N_i}$, where N_i is the number of windows sent with size i and L_i the number of losses detected for windows of size i. We computed this probability only when FDW_i is greater then 0.02. When FDW_i is too small, the proportion of losses is not significative because there is not enough data.

5.2 Verdi : $\alpha = 0.5$

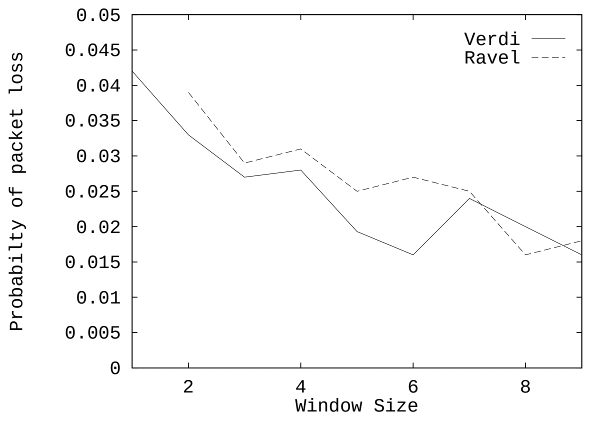
With $\alpha = 0.5$, verdi host increases the congestion window size slower than ravel host. These measurements can reveal how a less agressive policy of congestion window increase affects congestion losses. The performance of ravel was similar to that of verdi, in terms of losses and overall throughput.

5.3 Verdi : $\alpha = 1$

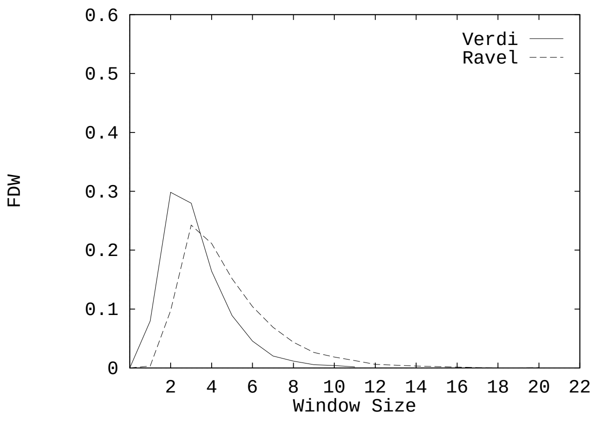
With $\alpha = 1$, verdi host runs TCP-Reno, same as the ravel host. Notice that the verdi curves are similar to those for ravel, as may be expected.

5.4 Verdi : $\alpha = 2$

With $\alpha = 2$, verdi host increases the congestion window size faster than ravel host. The performance of ravel remains similar to that of verdi, in terms of losses and overall throughput. We start to notice valleys on the frequency distribution curves (see Figure 14). This is due to the fact the congestion window is increased by larger increments. Hence, some values for the congestion window are unlikely to happen. We will see that this phenomenon is amplified

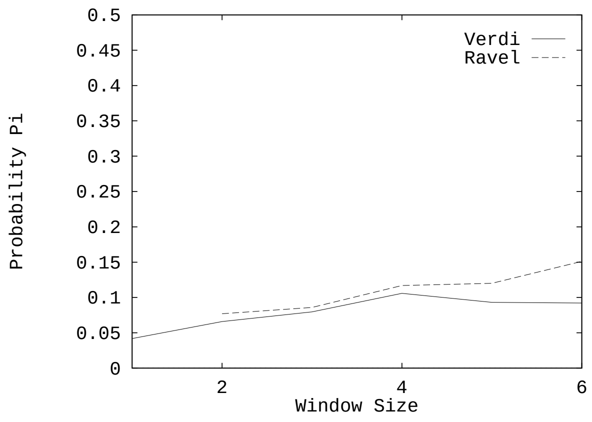


(a) Probability of packet loss

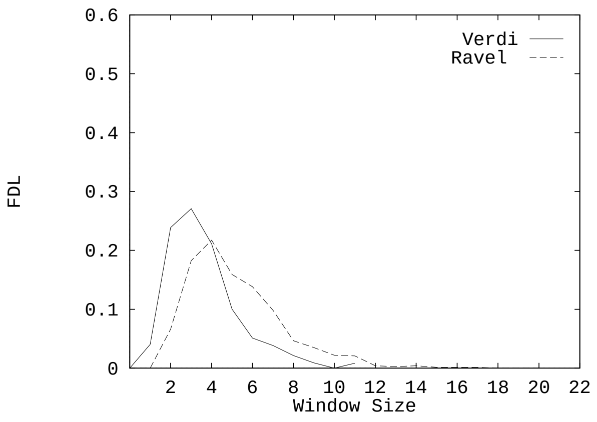


(b) Frequency distribution of window sizes(FDW)

Figure 10: q_i and FDW: $\alpha = 0.5$

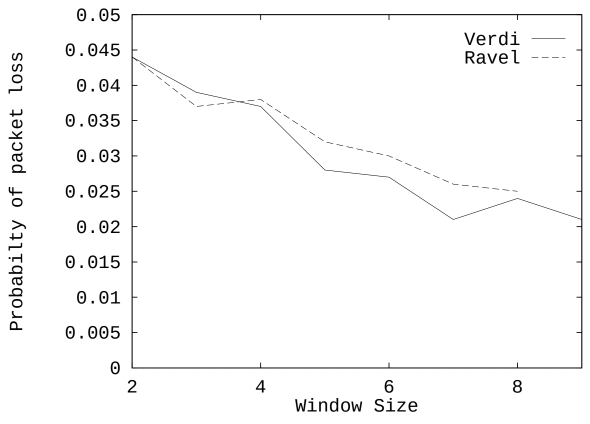


(a) Probability P_i

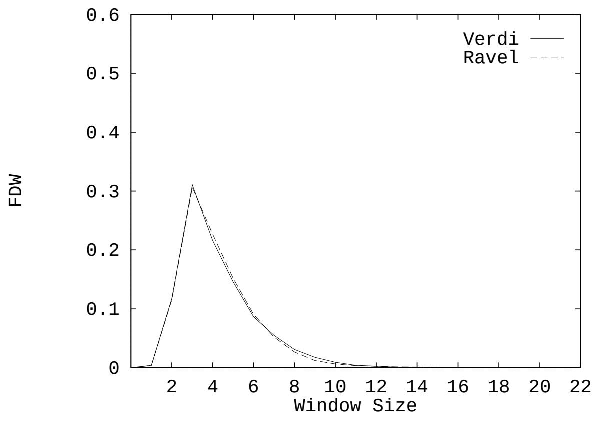


(b) Frequency distribution of losses(FDL)

Figure 11: P_i and FDL: $\alpha = 0.5$

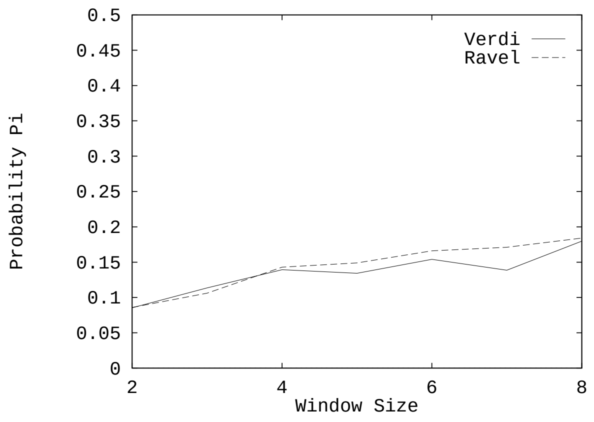


(a) Probability of packet loss

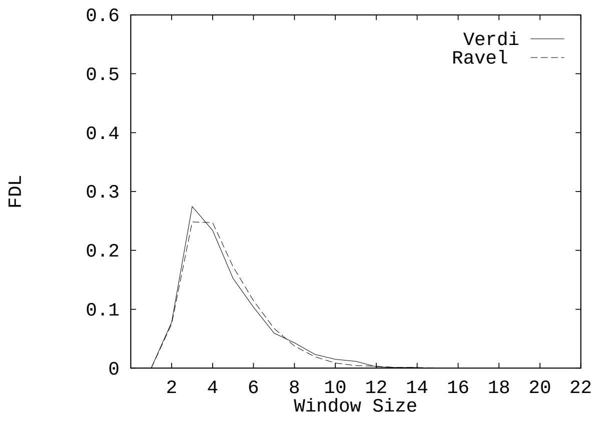


(b) Frequency distribution of window sizes(FDW)

Figure 12: q_i and FDW: $\alpha = 1$



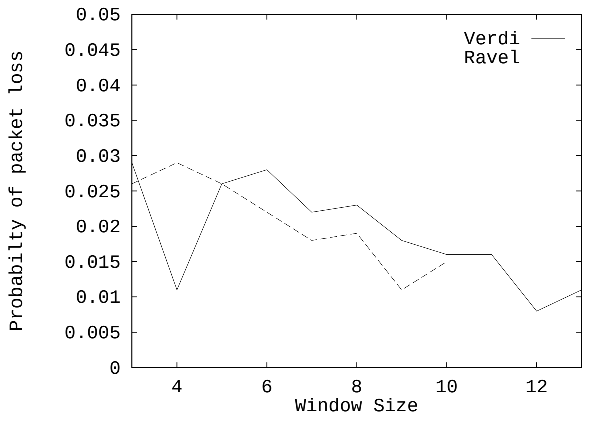
(a) Probability P_i



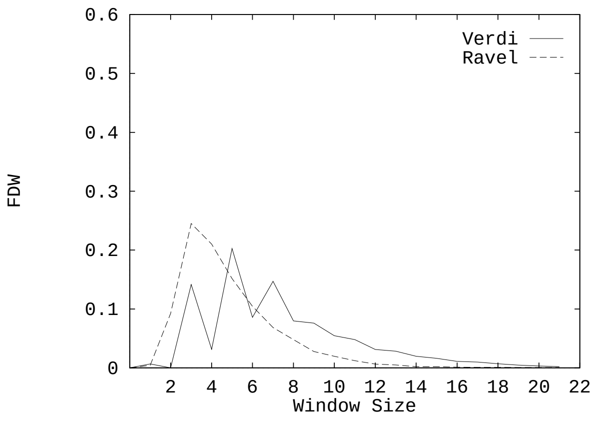
(b) Frequency distribution of losses(FDL)

Figure 13: P_i and FDL: $\alpha = 1$

when α increases. We must also notice that coincidentaly, we have "irregularities" on the other curves corresponding with these valleys : we do not have enough losses or windows at a given size.

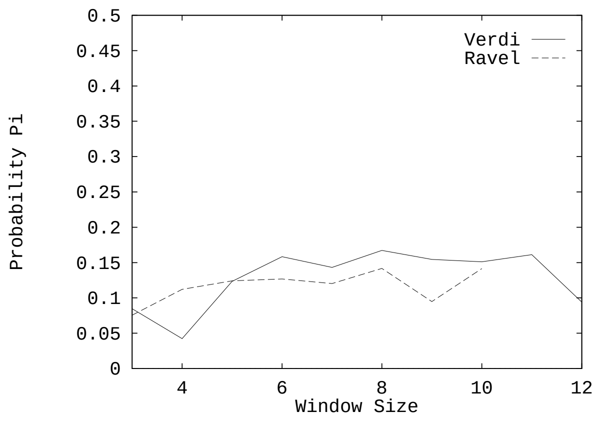


(a) Probability of packet loss

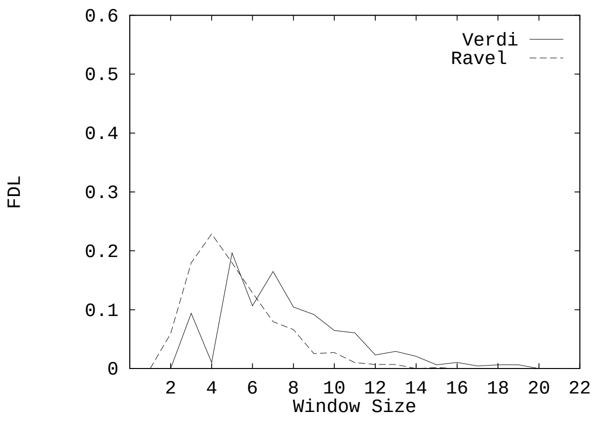


(b) Frequency distribution of window sizes(FDW)

Figure 14: q_i and FDW: $\alpha = 2$



(a) Probability P_i

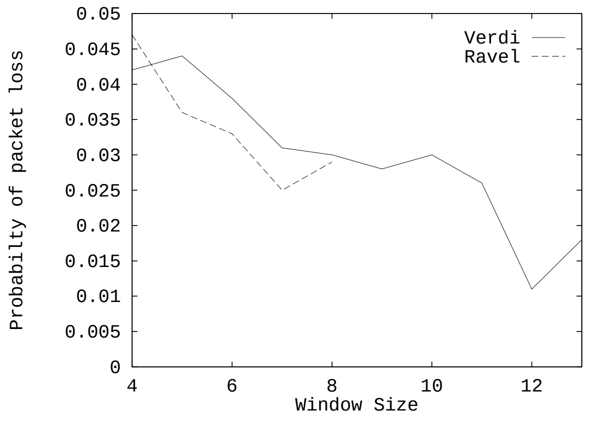


(b) Frequency distribution of losses(FDL)

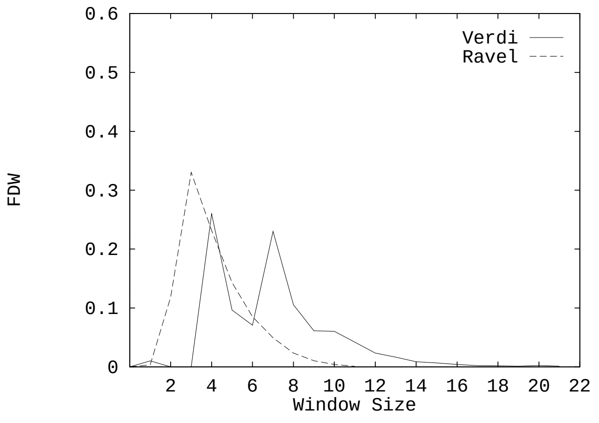
Figure 15: P_i and FDL: $\alpha = 2$

5.5 Verdi : $\alpha = 3$

With $\alpha = 3$, verdi host increases the congestion window size much faster than ravel host. The simulataneous performance of ravel remains similar to that of verdi, in terms of losses and overall throughput. We can observe that the valley at window size 6 in Figure 16b is amplified, and interestingly enough, the distribution of losses in Figure 17 follows the same pattern.

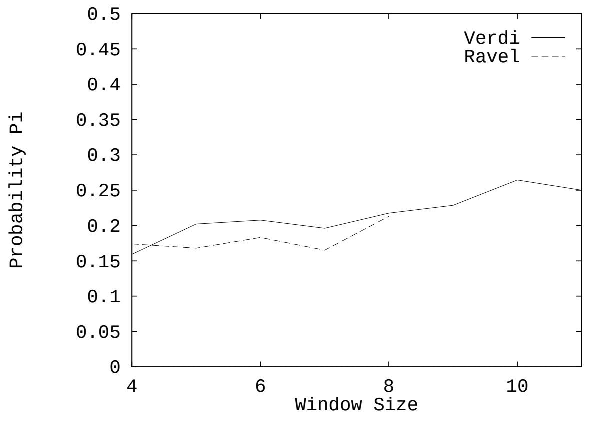


(a) Probability of packet loss

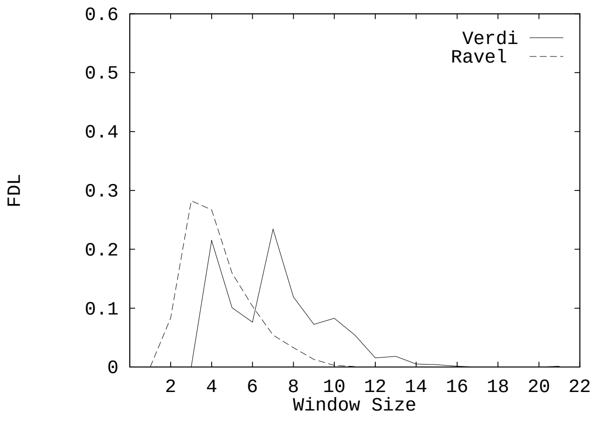


(b) Frequency distribution of window sizes(FDW)

Figure 16: q_i and FDW: $\alpha = 3$



(a) Probability P_i

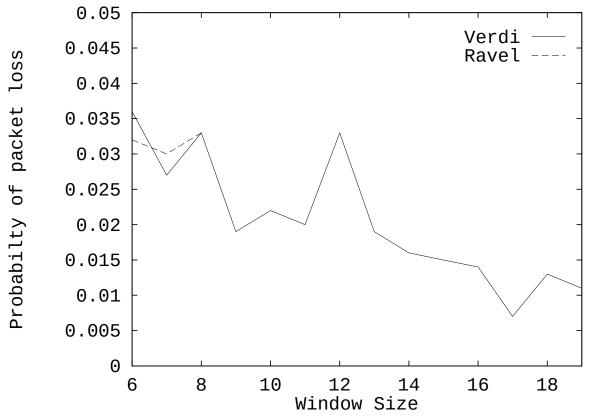


(b) Frequency distribution of losses(FDL)

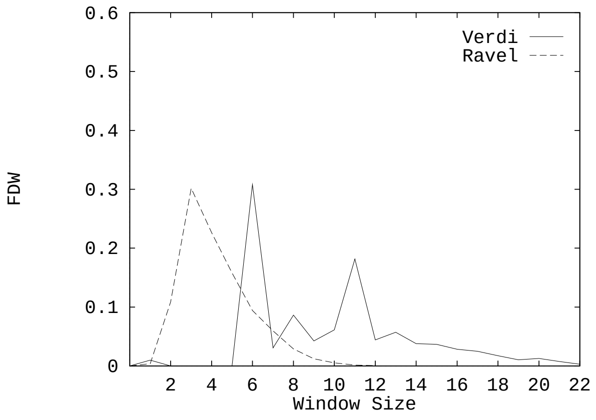
Figure 17: P_i and FDL: $\alpha = 3$

5.6 Verdi : $\alpha = 5$

With $\alpha = 5$, verdi host increases the congestion window size much faster than ravel host. The phenomenon of valleys is more amplified. Irregularities on the probability and probabilities curves coincide with the valleys.

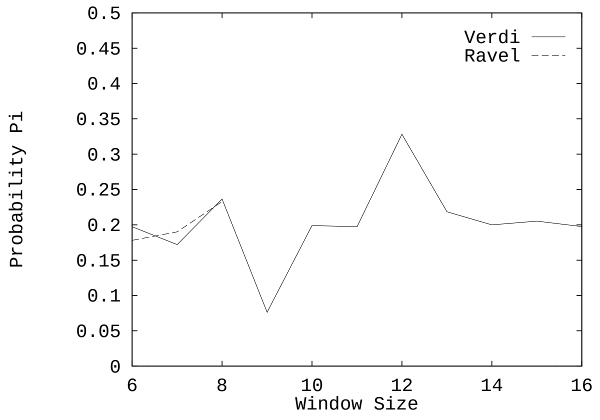


(a) Probability of packet loss

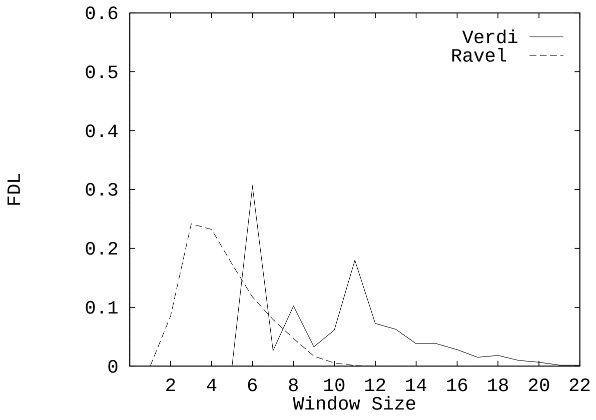


(b) Frequency distribution of window sizes(FDW)

Figure 18: q_i and FDW



(a) Probability P_i

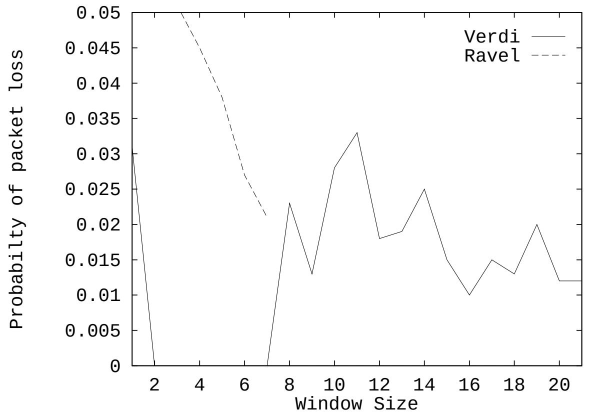


(b) Frequency distribution of losses(FDL)

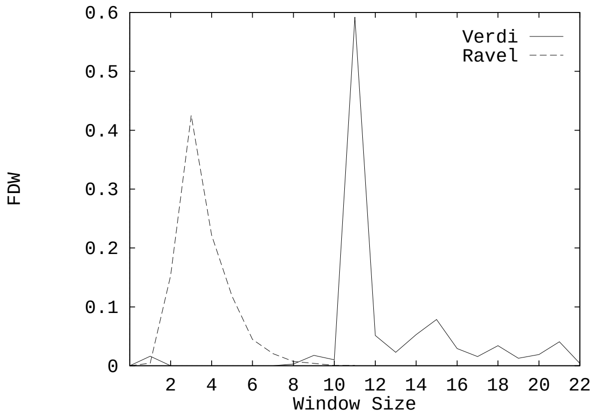
Figure 19: P_i and FDL: $\alpha = 5$

5.7 Verdi : $\alpha = 10$

With $\alpha = 10$, verdi host increases the congestion window size much faster than ravel host. The window size jumps rapidly around size 12 packets without intermediate size. We can still notice the loss pattern follows the window size distribution.

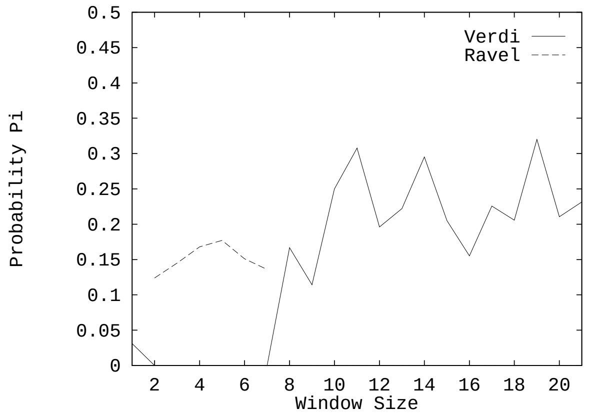


(a) Probability of packet loss

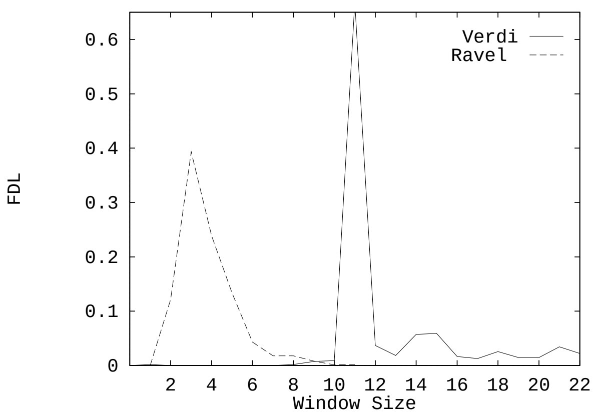


(b) Frequency distribution of window sizes(FDW)

Figure 20: q_i and FDW: $\alpha = 10$



(a) Probability P_i



(b) Frequency distribution of losses(FDL)

Figure 21: P_i and FDL: $\alpha = 10$

6 Interpretation

(This section needs revision.)

We must provide an explanation of the results above that the probability of packet loss slightly decreases with window size, and that the congestion predictors in the literature do not seem much better than a random predictor.

Let us assume, first, that the congestion losses are random and independent from ONE user's action, and , second, that the probability of loosing one packet increases with the time spent by the outstanding packets on the network. If we accept these two facts then our results would be justified. We will try to justify these two assumptions and then discuss our results.

Congestion Predictors

For the three congestion predictors, it is easy to see that they are based on functions of the form $f(RTT_i - RTT_{i-1})$ or $g(\frac{1}{RTT_i} - \frac{1}{RTT_{i-1}})$. The idea is then to detect the change in the round trip time RTT resulting from a change in the congestion window size. But, as the measurements show, the action of one user is infinitesimally small in comparison to the result of the actions of all other users. Hence, the change in RTT does not depend on one user's action. In this case, the inference from the RTT cannot be but random. For Wang's NTG, if you change the threshold from $\frac{1}{2}$ to some other number, you will get a different Frequency Congestion Prediction with an almost equal Accuracy of Prediction. For Vegas, you have only to change the thresholds α and β. It is not surprising that there are no implementations based on Jain's or Wang's criterion. For Vegas, which is a TCP implementation, we were puzzled that the authors [6] claim a throughput enhancement of 40 to 70% and a different team [2] confirmed an enhancement of at least 20% in throughput over TCP-Reno. But Vegas contains other enhancements (retransmission schemes) than the congestion avoidance scheme. The authors do not seem to have evaluated the contribution of the new congestion avoidance schemes to the performance improvement.

Packet Loss Probability and Window Size

The congestion losses occur as the result of the traffic generated by many users. Let us consider that the bottleneck stays at the same node during one connection. We can easily argue that :

- the set of users with who we share the bottleneck varies every RTT
- the overall load variation is almost independent of ONE user's actions
- ONE user's load variation is negligible in regard to the overall load variation

Since there is no synchronization of the users, congestion losses will occur randomly. In [5], Bolot reported random losses on his probe packets. Even if we know the exact queue size at the bottleneck at the time we want to send the packets, the randomness and specifically the burstiness of the traffic makes it difficult to adjust the window size with some guarantee to avoid congestion losses. Given a period of time t, there is a non null probability that a congestion loss will occur on the path. And this probability will increase with duration t of "observation".

Now, we can argue that the time for a window of $n > 1$ packets takes more time to traverse a network than a single packet. Hence, the probability to loose at least one packet is larger. But the probability P_i increases very lightly with the window size increase due to the fact that the time to traverse the network for n back to back packets is not proportional with n. This is due to the pipelining effect.

But why is the probability q_i of loosing ONE packet slightly decreasing with window size ? This comes from the relation between q_i and P_i : $q_i = 1 - (1 - P_i)^{\frac{1}{i}}$ and from the fact that P_i increases with the window size very slowly. Suppose that 10% of the windows experience congestion. But, since the packets are staggered by the network, we do not loose all of the packets of a window which experiences congestion. Now, suppose we have 100 packets to send. In a first scenario, we send windows of size 2. In this case, we would loose **at least** 5 packets on the average. Now, if we send windows of size 10, only one window would experience losses on the average. 10 back to back packets do not take five times longer to traverse the network than 2 back to back packets. And since, the packets are staggered, there will be a low probability that I loose a burst of 5 packets (under the assumption of moderate congestion). Hence, the probability of loosing one packet is small with windows of size 10 than with size 2.

One can argue also that if the congestion window reaches high values, this means that, temporarily, the conditions on the network are better (less congestion losses).

7 Conclusion

We investigated the impact of congestion window sizes on the congestion loss rate. For our connections, we found that the losses almost do not depend on the congestion window size. Also, the congestion loss rate decreased slightly when the congestion window size increased. The congestion losses seem independent of ONE user's actions. Indeed, the traffic generated by a user is negligible in regard with the total traffic induced by all the users who have to share the network.

For the congestion predictors, Jain's criterion, Wang's criterion and Vegas's criterion seem to be inadequate congestion predictors. They cannot be used to "diagnose" a loss.

These results need to be verified with a more exhaustive set of measurements. We are trying to show analytically the same results. In the meantime, we plan to implement on top of Reno, the TCP Vegas congestion avoidance scheme alone to verify the impact on the performance without the retransmissions schemes introduced by Vegas.

8 Acknowledgments

We want to thank John Hawkinson and Michael Shileds who allowed us to use their discard servers on “All-purpose-gunk.near.net” and “daedalus.crosslink.net”.

References

[1] Ab-Hamid, Waters, A.G., *Kalman Prediction Method for Congestion Avoidance in ISDN Frame-Relay Networks*, IEEE, 1991. pp 565-571.

[2] Ahn,J.S, P.B Danzig, Z.Liu and L. Yan, *Experience with TCP Vegas : Emulation and experiment*, Proc. SIGCOMM'95 Symp. Aug. 1995.

[3] B. Bakshi, P. Krishna, N. H.Vaidya, D. K. Pradhan, *Improving Performance of TCP over Wireless Networks*. In Proceedings of 17th Int. Conf. Distributed Computing Systems, Baltimore, May 1997.

[4] Balakrishnan,H., V. N. Padmanabhan, Seshan, S., Katz, R., *A Comparison of Mechanisms for Improving TCP Performance over Wireless Links*, ACM SIGCOMM'96, CA, August 1996.

[5] Bolot, J.C. *Characterizing End-to-End Packet Delay and Loss in the Internet*, Journal of High-Speed Networks, vol. 2, no 3, pp. 289-298, September 1993.

[6] L.S. Brakmo, S. O'Malley, *TCP-Vegas : New Techniques for Congestion Detection and Avoidance*, SIGCOMM '94 Conference on Communications Architectures and Protocols, London, U.K, pp 24-35, Oct 1994.

[7] Cinlar, E., *Introduction to Stochastic Processes*, Prentice-Hall, pp 88-89 1975.

[8] Floyd, S., *TCP and Explicit Congestion Notification*, ACM Computer Communication Review, V. 24, No 5, October 1994.

[9] V. Jacobson, *Congestion Avoidance and Control*, ACM SIGCOMM'88, pp 314-329, Aug. 1988. An update version is available via ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

[10] V. Jacobson. *modified TCP congestion avoidance algorithm*, mailing list, end2end-interest, April 30, 1990.

[11] Jain, R., *A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks*, Digital Equipment Corporation, Technical Report DEC-TR-566, April 11, 1989.

[12] Ramakrishnan, K.K., Jain,R., *A Binary Feedback scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer*, In Proceedings of the ACM SIGCOMM '88, pp 303-313, August 1988.

[13] J. Postel, *Transmission Control Protocol*, RFC 793, 85 pages, Sept 1981.

[14] W.R. Stevens, *TCP/IP Illustrated, Volume 1, The Protocols*, Addison Wesley, 1994.

[15] A. Tanenbaum, *Computer Networks*. Second edition.

[16] Z. Wang and J. Crowcroft, *A New Congestion Control Scheme : Slow Start and Search (Tri-S)*, ACM Computer Communication Review, vol 21, pp 32-43, January 1991.