EURECOM
Sophia Antipolis

Institut Eurécom
Department of Corporate Communications
2229, route des Crètes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-04-106

Towards a quantitative T-RAT

April 28, 2004

Siekkinen Matti, Urvoy-Keller Guillaume, Biersack Ernst

Tel : (+33) 4 93 00 26 26
Fax : (+33) 4 93 00 26 27
Email : {Matti.Siekkinen, Guillaume.Urvoy, Ernst.Biersack}@eurecom.fr

[1]Institut Eurécom's research is partially supported by its industrial members: Bouygues Téécom,Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Télécom, Hitachi, ST Microelectronics, Swisscom, Texas Instruments, Thales

Abstract

This is a technical report about the work performed and results achieved during the first three months of my Ph.D. studies. It describes work done around a tool for analyzing transmission rate limitation causes of TCP connections, namely T-RAT.

1 Introduction

The basis of this work is a paper published in 2002 Sigcomm [5]. It describes a tool, T-RAT, for analyzing TCP connections and inferring causes for limiting it's transmission rate. Since the tool was not available for public usage, a Master's student, Christian Schleippmann implemented it from scratch. Throughout the report some level of familiarity with the paper and the original tool described in it is assumed.

We debugged the implemented tool and modified it to give some quantitative results on the limiting causes. The tool was run with several different kinds of TCP dump traces that were obtained.

In this report we will explain how the tool works and what kinds of modifications were done to it with respect to the original one. In addition we'll describe the way we "calibrated" the tool and validated it's functioning. Finally we will show results and discuss some observations on applying the tool on real traces.

2 T-RAT description and modifications

T-RAT is a tool that analyzes a TCP connection and determines the cause that is limiting its transmission rate. The original tool was working on flow level and we modified it to work on connection level. This was done because later on we are planning to focus especially on analyzing long lived TCP connections.

2.1 Second test for application limitation

We added a second test for determining if a connection is application limited. This test counts the number of TCP packets sent with the push flag set. In RFC-793 it is said: " A TCP MAY implement PUSH flags on SEND calls. If PUSH flags are not implemented, then the sending TCP: (1) must not buffer data indefinitely, and (2) MUST set the PSH bit in the last buffered segment (i.e., when there is no more queued data to be sent)." If the connection is limited by the application, that is, the application is not providing the sender's buffer data to send all the time, then intuitively this test is justified.

2.2 RTT estimations and measurements

Getting good estimate of the RTT for a connection is important for the functioning of T-RAT. If it is all wrong the tool gives false results in many occasions. In the original version of T-RAT an estimate for RTT is computed in a following way: The estimator considers 27 candidates between 0.003 and 3 seconds. The packets are divided into flights according to observed inter-arrival times for each RTT candidate. The estimator then tries to identify as many of the flights as possible to belong to correspond to a certain TCP state. The RTT candidate that leads to identifying the most flights in terms of data bytes will be chosen as the correct one. Another, and generally more accurate, approach for estimating RTT that people have used is to keep track of the TCP state engine. In this way the achieved RTT estimate also takes into account the variations with time. However, this is a heavier task to perform and requires very careful analysis of out-of-sequence packets in terms of reordering, packet loss or other anomalies. An example of such approach is in [3]. The important question, though, is: Is all this necessary? For the purposes of T-RAT the current method is considered as good enough for the time being.

In identifying the flights, large inter-arrival times are considered as boundaries according to a simple algorithm. The nature of the algorithm tends to cause the chosen best RTT estimate to underestimate the true one. To cope with this, we are recomputing the RTT by simply calculating the smallest delay between successive flights. Now, this on the other hand may lead in some situations to huge overestimations of the true RTT. This occurs when the traffic is heavily application limited in the sense that it consists of small bursts with large lulls between them. Web browsing produces typically this kind of traffic. Traffic that is heavily categorized as application limited(the original test seeking for lulls) seems to exhibit this behavior. To be able to reduce this effect, if we see that as a result of re-computation we get a larger RTT than ten times the original best estimate, we will discard the recomputed value and choose the original estimate instead.

For the purposes of validating the RTT estimator part of T-RAT we are measuring the RTT also in two other ways. The first one is the well known syn based method to calculate the RTT from the delays of the first packets of the connection. In other words the RTT is got from the delays between the syn/ack packets during the handshake. Another method that we are using is to simply calculate a delay between a data packet and a corresponding acknowledgment for all data packets seen and take the minimum of these values. This is done separately for both directions of data flow and the results summed up. In this way we should be getting a justified measurement of the RTT no matter where our measurement point is located on the path. We consider the RTT measured in this way to be “robust” if we have more than five samples from both directions of data flow.

2.3 Quantitative results

Original version of T-RAT gave for each limitation cause a positive or negative test result. In other words if the connection is limited by this particular cause or not. We wanted to get more insight into the limitations and decided to define quantitative test results were applicable. Quantitative result means a number that tells how strongly a given connection is limited by this current test. This would also prove useful in calibrating the tool to classify the connections in different limitation causes in a reasonable way. This would be a necessary task to perform since not all of the original tests would be applicable as such because we moved from flow level to connection level and also because we added a new test.

We defined quantitative values for six of the ten tests. The original application limitation test is defined to search for a lull in data transfer that is longer than RTT and that is preceded by a packet smaller than maximum segment size. This is clearly an indication of a situation that the TCP sender has no more data to send at the moment. As a quantitative measure we are counting the number of these kinds of lulls that are observed during the lifetime of the connection and dividing this by the duration of the connection. We have to take into account the duration because just defining a fixed number of lulls doesn’t make sense since the connection can last from seconds to hours. Note that this is a modification due to moving from flows to connection level.

The added test for application limitation is concerned about TCP push flags. One can set this flag in a TCP packet in order to “push” it through the channel with minimal delay. This means that TCP should send it right away and not wait for more data for more efficient segmentation. Also the receiving TCP side should forward this kind of packet with minimal delay to the application. The difference with urgent flag is that urgent packets may be even put ahead in a queue whereas packet with a push flag set may not. Normally one observes a packet with a push flag set when it is the last one of a burst. So it is also a kind of an indicator of a situation that the TCP sender has temporarily no more data to send. This means that connections that are sending data in small bursts would have a high number of packets with the push flag. An extreme case is a TCP sender that is transmitting bursts consisting of a single packet. One could imagine some kind of a keep alive messaging to produce this kind of traffic. Also traffic produced by network gaming in the form of update packets from each individual player to all the other players could be an example. Note, however, that traffic such as caused by web browsing is normally not of this kind. The sent bursts corresponding to an object on a web page are normally larger than a single packet causing the relative presence of push flags to be quite low. This kind of traffic falls into the original application test category. Also worth noticing is that this push flag test should intuitively identify a subset of the other application test. This is because the lull after each small(one packet) burst should be always greater than RTT. Otherwise the sender would have data to send all the time and push flags would not be present. We are counting the percentage of packets with a push flag out of all packets of a connection and using this as a quantitative measure.

The test for TCP receiver window limitation looks for three consecutive flights with a flight size, S_i, very close to the maximum receiver advertised window, $awnd_{max}$. More specifically the following inequality must be fulfilled: $S_i \cdot MSS > awnd_{max} - 3 \cdot MSS$. We are counting the percentage of flights that have a flight size that

Main limitation	bandwidth	congestion	receiver win	sender win	opportunity	application	transport
estimated RTT(ms)	150	120.1	186.2	189.1	200.6	2774.8	147.0
measured RTT(ms)	0.023	0.01	0.023	0.02	0.024	90.5	0.019
synack RTT(ms)	200.2	200.2	200.2	200.2	200.2	200.2	200.2
estimated RTT(ms)	3.9	14.6	8.6	8.5	20.4	20.3	20.3
measured RTT(ms)	0.015	0.008	0.014	0.015	0.016	90.6	0.014
synack RTT(ms)	20.3	20.3	20.3	20.3	20.3	20.3	20.2

Table 1. Obtained RTTs for test traces. First set of results are for 200 ms RTTs and bottom ones are for 20ms.

satisfies this inequality and take that as a quantitative measure.

Testing for TCP sender window limitation requires the connection neither to be receiver window limited nor bandwidth limited. In addition two inequalities must hold. First one is $S_{F_{80}} < S_{F_{med}} + 3$ where $S_{F_{80}}$ is the 80th percentile and $S_{F_{med}}$ the median of the flight sizes. Secondly, there has to be four consecutive flights with flight sizes between $S_{F_{80}} - 2$ and $S_{F_{80}} + 1$. Again we are counting the percentage of flights that have a flight size that falls between these limits and use that as the quantitative value.

Bandwidth limitation detection is divided into two categories. The first test looks for at least three flights with retransmission. In addition it requires that the maximum and minimum flight sizes before the loss occurs do not differ by more than the MSS. For this test we count the percentage of flights with a flight size in between these maximum and minimum flight sizes before losses and take that number as the quantitative result. The second bandwidth limitation test aims to identify a connection that is sustaining a constant bandwidth on its bottleneck link. Therefore it is searching for packets that are nearly equally spaced. For this second test we have no quantitative definition.

The test for congestion limitation is based on experiencing losses. We are taking the loss rate as the quantitative measure of presence of congestion.

3 T-RAT validation

To validate the correct functioning we ran the tool with some traces that we created to resemble traffic from each kind of limiting category. Then we validated the tool by running it with real life collected traces.

We first needed to validate the RTT estimator part of T-RAT. Obviously with the traces that we created this was very easy. We had two sets of traces: one for connections with an RTT of 20ms and one for connections with an RTT of 200ms. Each trace of a set represents a single connection limited by a specific cause. An exception is the trace for congestion where two connections are present. Both sets contained examples for seven limitation causes. The push flag application and bottleneck bandwidth(second bw test) limitations are not present. More details about the traces can be found in [4]. The results are shown in table 1. One can see immediately that the measured values are totally wrong. The reason for this is that the traces contain data flowing only in one direction. Therefore, the tool can only measure one direction from the measurement point. Since in this case we are effectively sitting on the receiver side, the measured values correspond basically only the processing time of a data packet at the receiver side.

The initial idea was also to validate the RTT estimator by comparing the RTT estimations with the measured ones while analyzing real life traces recorded from the Internet. We analyzed an Internet traffic trace of approximately four hours collected at LAAS. Figure 1 contains scatter plots about these comparisons. In the up left one the measured value of the RTT is plotted against the estimate. The one in the up right corner is the same but only includes the connections for which we have a “robust” measured value for RTT (see section 2.2). One can see that

factor	measure / estimate	measure(robust) / estimate	synack / estimate	measure / synack
1.01	43	67	11	15
1.1	56	70	35	45
1.5	66	73	52	65

Table 2. Percentages of connections having given RTT values that differ by no more than a given factor.

things are getting better. Table 2 gives numbers that confirm this. It tells the percentages of connections having the given RTTs that differ from each other by maximum of the given factor. The columns corresponds to the plots in figure 1. When looking at robust measurements vs. estimates, the relatively small increase in percentages when drastically increasing the factor suggests that most of the connections have RTT measurements and estimations extremely close to each other and the rest of the connections have them differing by a value that can vary from relatively small to very large. The bottom left plot visualizes the situation when changing a measured RTT to an RTT obtained with the synack method. The results look visually quite a lot alike with the up left plot but in fact they are much worse. Again this can be seen from table 2. The bottom right plot shows the measured RTT against synack RTT and at first sight they also seem to correspond to each other pretty well. There seem to be few values very far from the line x=y but on the other hand the concentration around this line is not as sharp as in the plots at the top. Taking a closer look on the numbers reveals that the percentages are not so encouraging. This confirms the difference between comparisons of estimated against measured and estimated against synack RTTs.

When looking at the limiting causes of the connections with RTT estimates very close to the measured value, one can see that opportunity limitation is clearly dominating. It is by far the most common limiting cause also when looking at all the connections but the trend here is even stronger. The reason can be understood when thinking about behavior of TCP in the beginning of a connection. Opportunity limited connections are generally very short and this means that TCP never leaves slow start. Normally in this state the flight nature can be cleanly identified because as the congestion window is relatively small the TCP sender needs to wait for feedback every time before sending a new flight of packets. Then as the connection lives on, identifying flight boundaries becomes more difficult because a kind of balance between congestion window size and delay in getting feedback is reached. All this suggests that the RTT estimator part of T-RAT, as its functioning is based on being able to identify flights, would give less accurate RTT estimations with respect to the minimum RTT of the connection on long lived connections. However, this doesn't necessarily mean that the tool is not working well in these cases. We will get back to this in short.

In addition to just trying to validate the RTT estimator part of T-RAT by inspecting at the RTT values itself, we had a glance at how it is performing in recognizing TCP states with the best RTT candidate. By looking at how flight sizes are changing, T-RAT tries to identify the state of TCP at each flight. Possible states are slow start, congestion avoidance and unknown. For more information about identifications and transitions between these states refer to [5]. We calculated the amount of bytes carried by flights that were recognized as being in either slow start or congestion avoidance and divided that with the total amount of bytes of the connection. So, we got the fraction of recognized bytes and we plotted two histograms of them shown in the left side of figure 2. The one on the top views the situation when purely counting the number of connections and looks rather sad. It shows that for more than 60% of the connections T-RAT classifies all flights as being in unknown state. The histogram on the bottom is weighted by bytes that a connection carries and looks very different. One can see that almost 50% of the total bytes of all connections are carried by connections for which T-RAT is able to classify all flights being in either congestion avoidance or slow start. We can conclude from these that even though T-RAT recognizes basically no flight states out of great majority of connections, it is able to recognize very well the connections that are carrying significant amounts of data. These connections are generally the interesting ones and the ones that

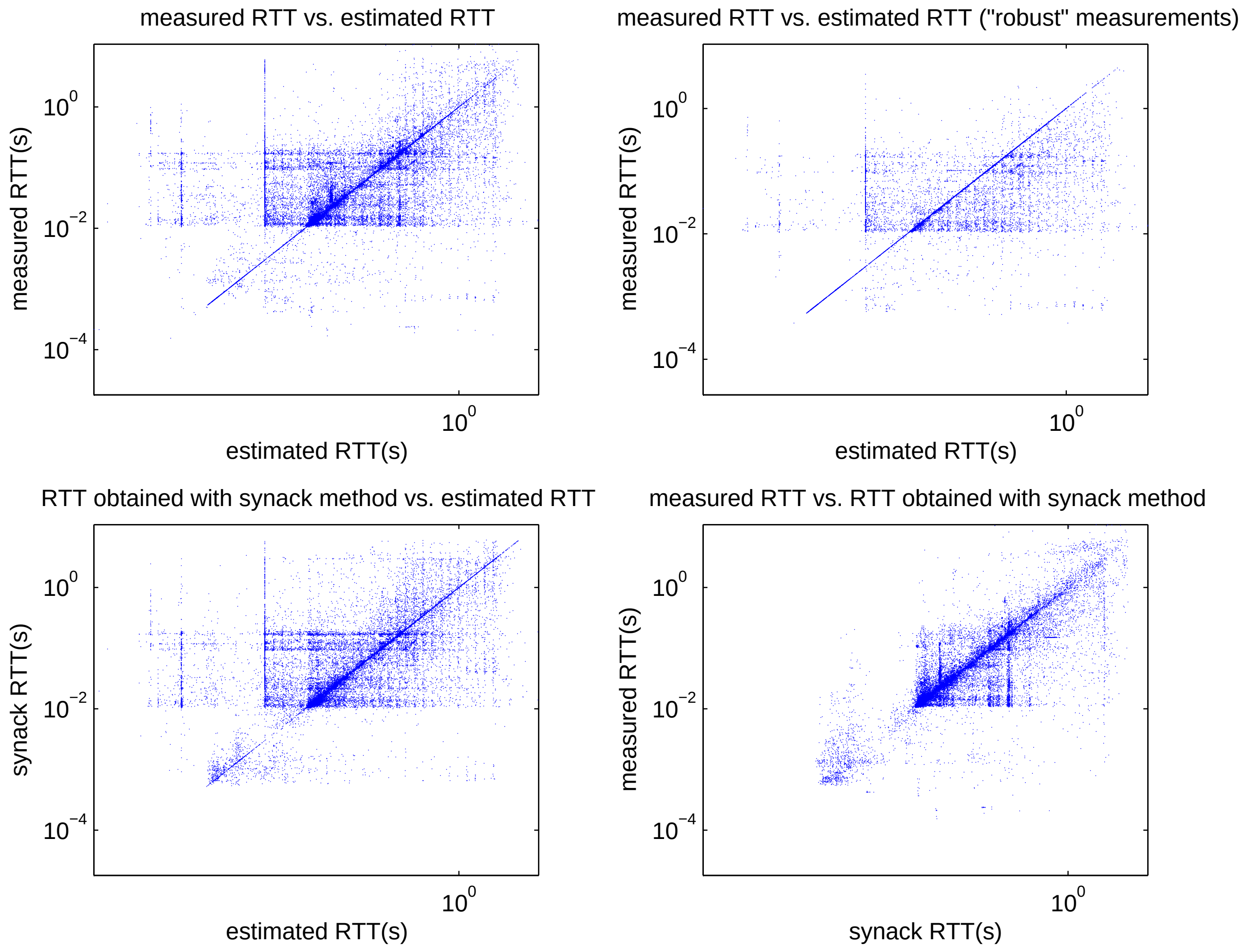
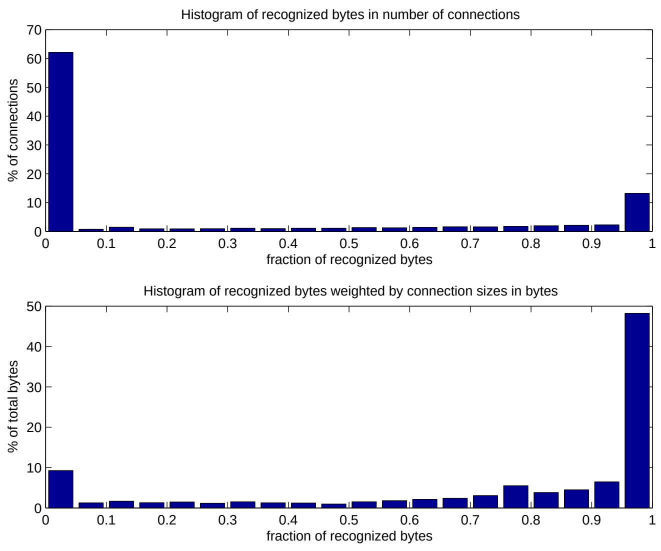


Figure 1. Scatter plots of different obtained RTTs with Internet traffic.



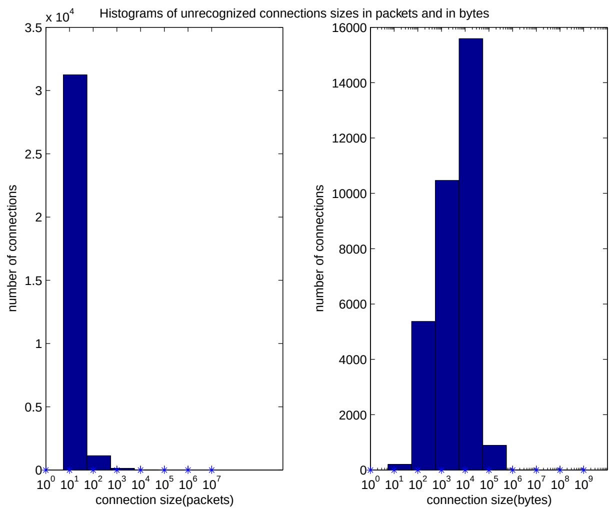


Figure 2. Recognized TCP states and size distribution of unrecognized connections.

we would focus on. This is confirmed by the right side plots in figure 2. They show the sizes of connections of which T-RAT can't recognize any flights in other states than unknown. In fact, a closer calculation reveals that 80,3% of these connections have a size smaller than 19KB, a value that equals 13*MSS when MSS is set to 1,5KB which is a typical value for it. This means that by the definition of opportunity limitation test, these 80,3% would be automatically classified as opportunity limited because of their small size and the flight categorization doesn't matter.

There is a paradox though. Just before we were claiming that with long lived connections the RTT estimates would be less accurate and the tool wouldn't function accurately. Long lived connections are typically also the ones that are carrying most of the data. This might also suggest that perhaps comparing the RTT that is used in flight classification in T-RAT with the measured RTT is not meaningful. This is because the measured RTT is representing the minimum RTT of the path and the RTT used in the tool may represent something else. Also another point is that with long lived connections the probability for the RTT to vary more during the lifetime on the connection is higher. As the estimated RTT is somewhat averaged over the behavior during the entire connection, it may differ significantly from the minimum RTT obtained with measurements but still lead to good flight identification.

4 Calibrating T-RAT

Currently T-RAT may classify a connection to be limited by more than one cause. However, with current calibration of the thresholds for the quantitative test results 86% of the connections are only classified to one category and 11% to two categories. Table 2 show the correlation coefficient matrix for the tests. Negative correlation between opportunity and all the others is explained by the fact that if a connection is opportunity limited it can't be anything else. This is defined in the tests. Different application limitation tests have some non-neglible positive correlation to each other as was expected. Remember that the push flag test should in fact identify, at least in some extent, a subset of the other one. Strongest positive correlation that we observe is between sender window and lull application limited tests. This could be explained by the fact that if we always have only a little amount of data to send between idle periods, the flight sizes remain small and this keeps the 80^{th} percentile and median small and close to each other. This causes the sender window test to easily give positive results. Besides these mentioned cases there are no neglible correlations between the tests.

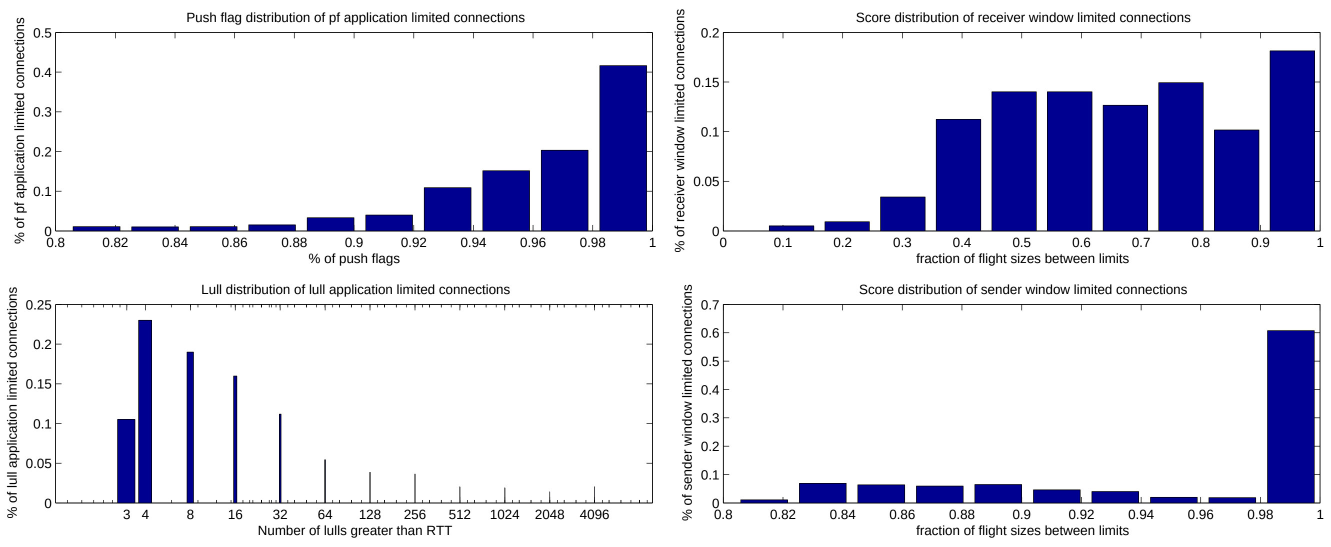


Figure 3. Score distributions for application limitation and window limitation tests.

	bw(test1)	bw(test2)	congestion	rcv win	snd win	opportunity	appl(push flag)	appl(lull)	transport
bw(test1)	1.0000	0.0064	-0.0242	-0.0019	-0.0299	-0.1020	0.1490	0.1031	-0.0040
bw(test2)	0.0064	1.0000	0.0008	-0.0087	-0.0182	-0.0620	0.0011	0.0212	-0.0024
congestion	-0.0242	0.0008	1.0000	0.0036	0.2186	-0.4000	0.1165	0.1265	0.0986
rcv win	-0.0019	-0.0087	0.0036	1.0000	-0.0692	-0.2359	-0.0127	0.0916	0.0566
snd win	-0.0299	-0.0182	0.2186	-0.0692	1.0000	-0.4939	0.2594	0.3704	0.0245
opportunity	-0.1020	-0.0620	-0.4000	-0.2359	-0.4939	1.0000	-0.3135	-0.5822	-0.0660
appl(push flag)	0.1490	0.0011	0.1165	-0.0127	0.2594	-0.3135	1.0000	0.2662	0.0008
appl(lull)	0.1031	0.0212	0.1261	0.0916	0.3704	-0.5822	0.2662	1.0000	0.0272
transport	-0.0040	-0.0024	0.0986	0.0566	0.0245	-0.0660	0.0008	0.0272	1.0000

Table 3. Correlation coefficients for the nine limitation tests. Rows from left and columns from top correspond to tests in the following order: bw(test1), bw(test2), congestion, receiver win, sender win, opportunity, application(push flag), application(lull) and transport.

Currently the thresholds for passing a limitation test are set according to table 3. Refer to section 2.3 for explanations about meanings of the numbers. The thresholds were determined by using common sense(for instance in defining the packet loss threshold) and looking at the distribution of the quantitative test results. For example when defining a threshold for push flag test or window limitations we could see a clear peak at the high end of the interval [0,1] and then set the threshold before the peak. A few examples are in figure 3.

5 Applying T-RAT on traces

We are presenting here analysis of two different traffic traces. First one was collected at LAAS. We were in fact provided with a very big one week long Internet traffic trace containing all common categories of traffic. We analysed a four hour piece of this trace with T-RAT. The second trace that we analyzed was recorded at Eurecom and is an approximately four hour trace containing solely BitTorrent traffic. We will present observations about both of these categories of traffic.

Limitation	bw(test1)	congestion	rcv win	snd win	appl(push flag)	appl(lull)
Threshold	0.8	0.01	0.4	0.8	0.9	0.1(lulls/s)

Table 4. Thresholds for passing a limitation test.

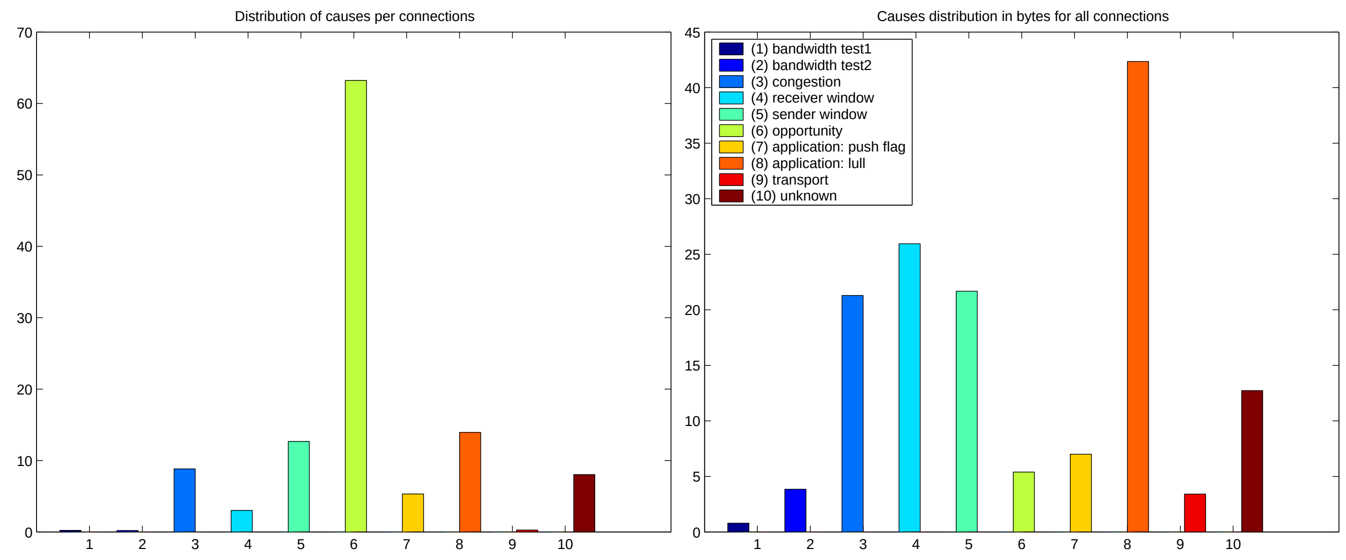


Figure 4. Distribution of limitation causes for LAAS trace.

5.1 LAAS

Figure 4 shows the distribution of the limitation causes in connections and in bytes. The results are not very surprising. Opportunity limited connections dominate in their number but don't carry much data. Application limited is clearly the dominating one and window limited and congested connections are following it when looking at bytes. The dominance of application limited connections can be explained with the fact that great majority of the traffic in the trace is web traffic and this is more application limited than any other category of traffic as will be show soon.

Table 4 describes how the limiting causes are distributed among the most popular applications. We looked at TCP port numbers and based on the knowledge of well known port numbers [2] we figured out the application. Web traffic consists of both HTTP and HTTPS traffic. P2P and others include all the dynamic and private ports(from 49152 through 65535). There are a few observations worth pointing out. Web traffic is much less opportunity limited and more application limited than the other categories. This confirms what we just saw about the dominance of application limited traffic. SMTP and POP traffic is heavily opportunity limited which is logical because normally emails are small in terms of data. For now it is difficult to say anything about the traffic from/to port 2688 since we have no idea what application this traffic represents. It seems to contribute significantly so maybe it should be clarified in the future.

Connection sizes and durations are visualized in figure 5. The plots support the well know fact that most of the connections are small and short. In this case less than 100 packets, less than 100KB and shorter than 10 seconds. Looking more closely at the durations one can see peaks occurring at 17, 60 and 120 seconds. These are most likely due to durations stretched by timers. In other words they are protocol peculiarities.

In figure 6 several characteristics of a connection with respect to different limitation causes are shown. What comes to average flight sizes, it is clear that opportunity limited connections have the smallest values. Similarly the

Application	bw(test1)	bw(test2)	congestion	rcv win	snd win	opportunity	appl(push flag)	appl(lull)	transport	unknown
ftp	0.58	0.29	0	0	4.5	92	8.1	6.4	0	0
ssh	0.45	0	0.15	0.45	5.1	93	4.8	6.4	0	0.15
smtp	0.02	0.04	0.81	0.41	0.79	95	0.27	4.2	0.02	0.11
web	0.44	0.04	8.8	4.4	13	55	5.7	20	0.30	10
pop	0	0	0.11	0.16	0.05	99	0	0.65	0	0
nntp	0	0	0	0.16	1.6	92	0	7.3	0	0.47
port 2688	0.68	0.08	6.4	0.31	13	80	4.9	11	0.08	2.4
P2P & others	0.08	0.05	5.9	1.4	9.5	73	4.7	13	0.30	4.2

Table 5. Percentages of traffic of different applications for LAAS trace.

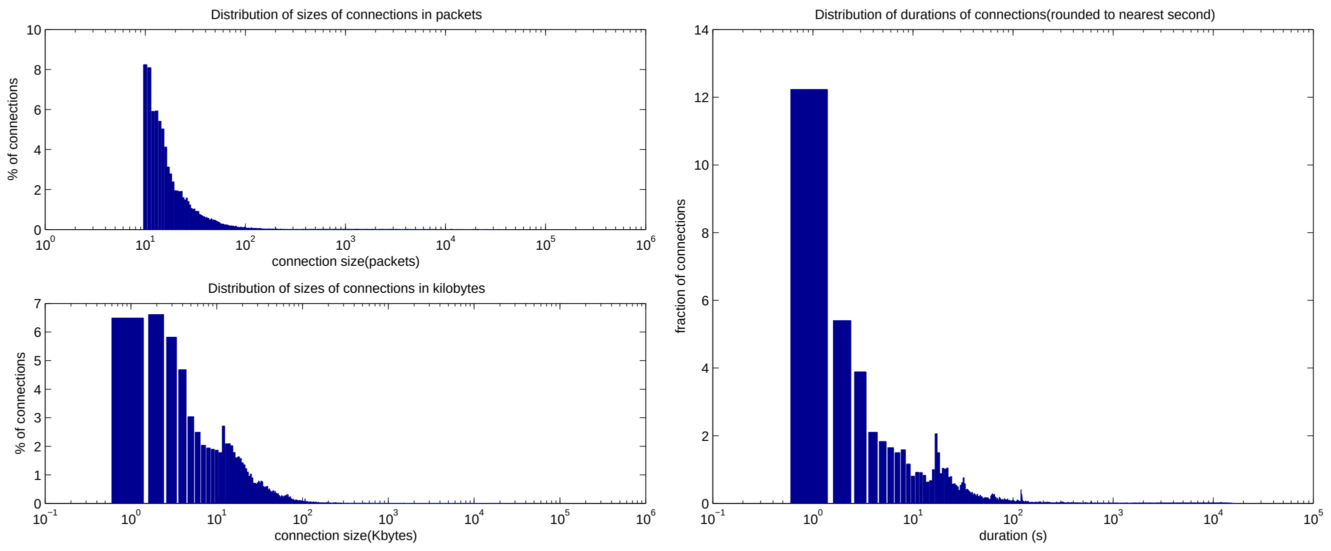


Figure 5. Sizes and durations of connections for LAAS trace.

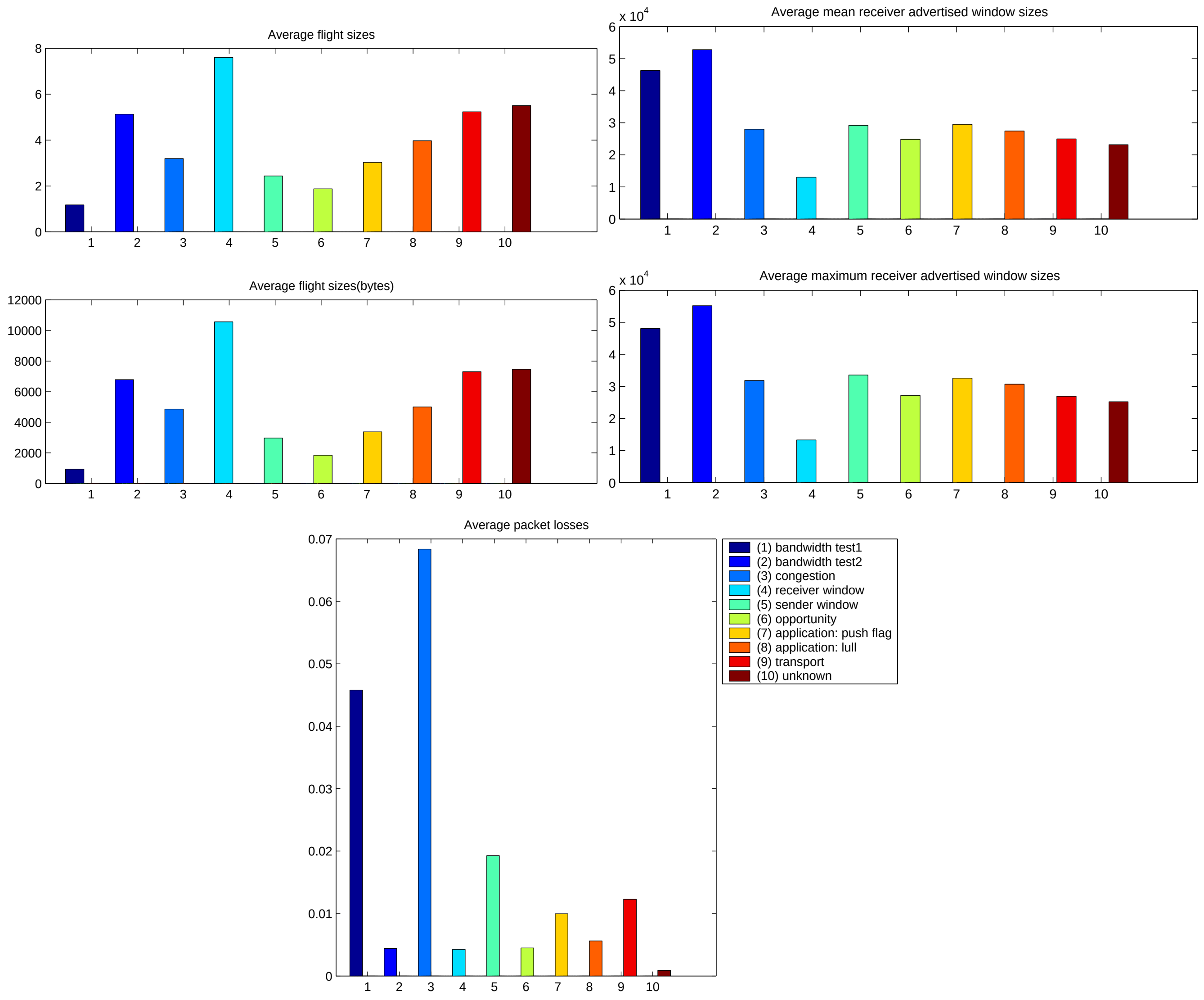


Figure 6. Different attributes of connections for LAAS trace.

fact that receiver window limited connections have large flight sizes on average is not surprising. From advertised window sizes we can see the effect of receiver window limited connections. Also the difference between maximum and mean advertised windows is small. Packet loss is significantly larger only with congested connections as it should be.

T-RAT is analyzing only connections that have at least 10 packets. The reason is that it becomes very difficult to recognize flight behavior with fewer packets. However, we are dumping some simple information about these small connections as well. The most important observation is that 73% of them are not carrying any bytes of data. So this means that they are basically syn packets for attempting to initiate a connection but failing to do so. Part of them might also represent traffic generated by port scanning.

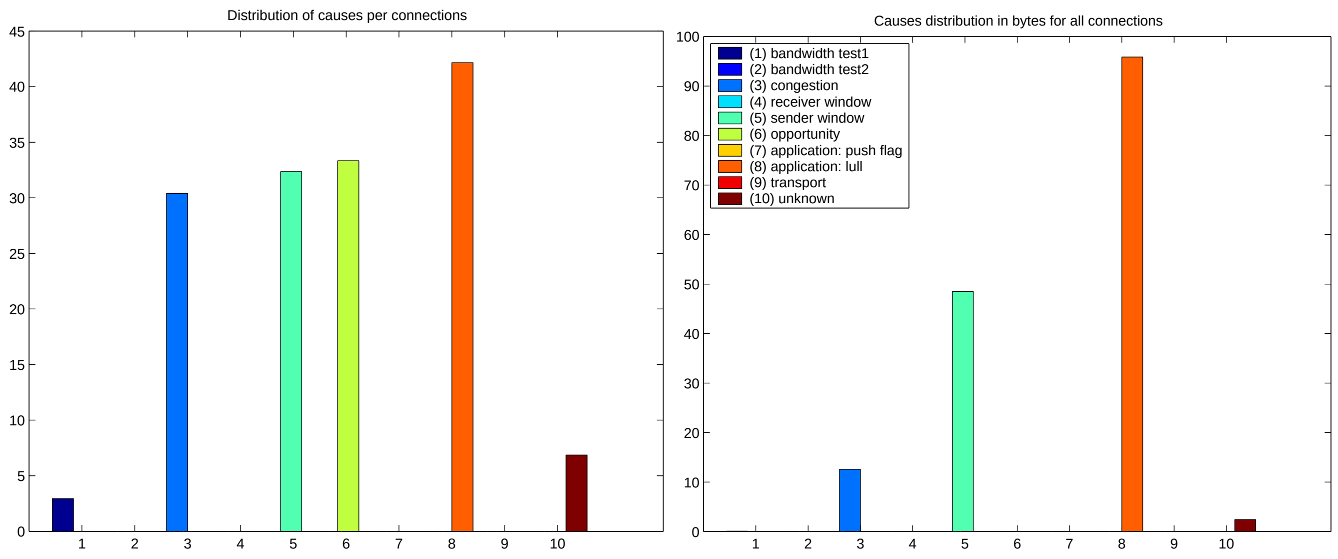


Figure 7. Distribution of limitation causes for BitTorrent trace.

5.2 BitTorrent

The second category of traffic that we analyzed was BitTorrent traffic. BitTorrent is a P2P system for sharing data. For details about its functioning and architecture refer to [1]. Figure 7 shows the distribution of the limitation causes in connections and in bytes. Differences with respect to LAAS trace analysis results are that in terms of number of connections opportunity limitation is clearly less dominating. Also some categories are missing totally. In terms of bytes opportunity limitation disappears completely from the chart and application limitation is through the roof. In fact almost 100% of the bytes in all connections are application limited. There's a logical explanation to that though. The BitTorrent protocol is designed to work as follows: When initiating a download, a client gets a list of peers that it co-operate with, in other words to whom it should start uploading data. The client is not uploading to all of the peers all the time but instead there is a way how BitTorrent cycles the active connections. This means that a TCP connection which is initiated for a BitTorrent session is switching its state from active to idle constantly. In idle state only keep alive messages are sent. This explains that BitTorrent traffic is indeed application limited. Also sender window limitation plays a rather significant role. By manually inspecting some of the sender window limited connections one can see that the life cycle of these connections can be divided into two phases. At first there is continuous data transmission and then it stops and the connection is kept alive with very small keep alive messages. It seems that in general the second phase lasts clearly longer than the first one. This affects seriously the parameters that are used in the sender window test. Remember that we are computing the 80^{th} percentile and the median of the flight sizes and comparing these in the test. The fact that there is a long basically idle part of the connection when one is only sending flights of one packet will cause these values to be very small. This causes the test to give a positive result very easily. This behavior is most likely again due to BitTorrent and not really the network.

The connection sizes are considerably larger with BitTorrent as was with regular Internet traffic in the LAAS trace as can be seen from figure 8. We plotted histograms instead of distribution because of the small number of samples. The mass of the connections are between 100KB and 1MB but there are significant amounts of also larger connections up to more than 100MB. This makes sense because the trace was collected from connections that were downloading a Linux Red Hat distribution which is a very large software in size. Also durations are visualized as a histogram because the individual values were so few and distributed in a way that nothing could be

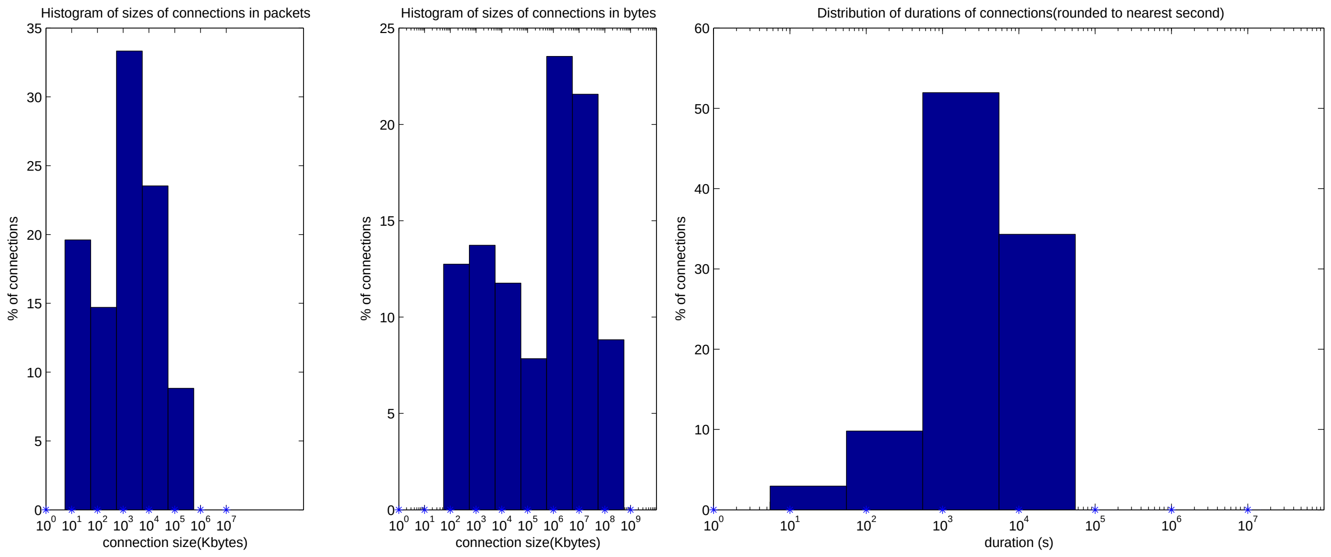


Figure 8. Sizes and durations of connections for BitTorrent trace.

said. Most of the mass is around 1000 and 10000 seconds which just tells that the connections are mostly long.

The average flight sizes that are shown in figure 9 are naturally also affected by the behavior of BitTorrent. The idle periods when the connection is only kept alive are driving down the average flight sizes.

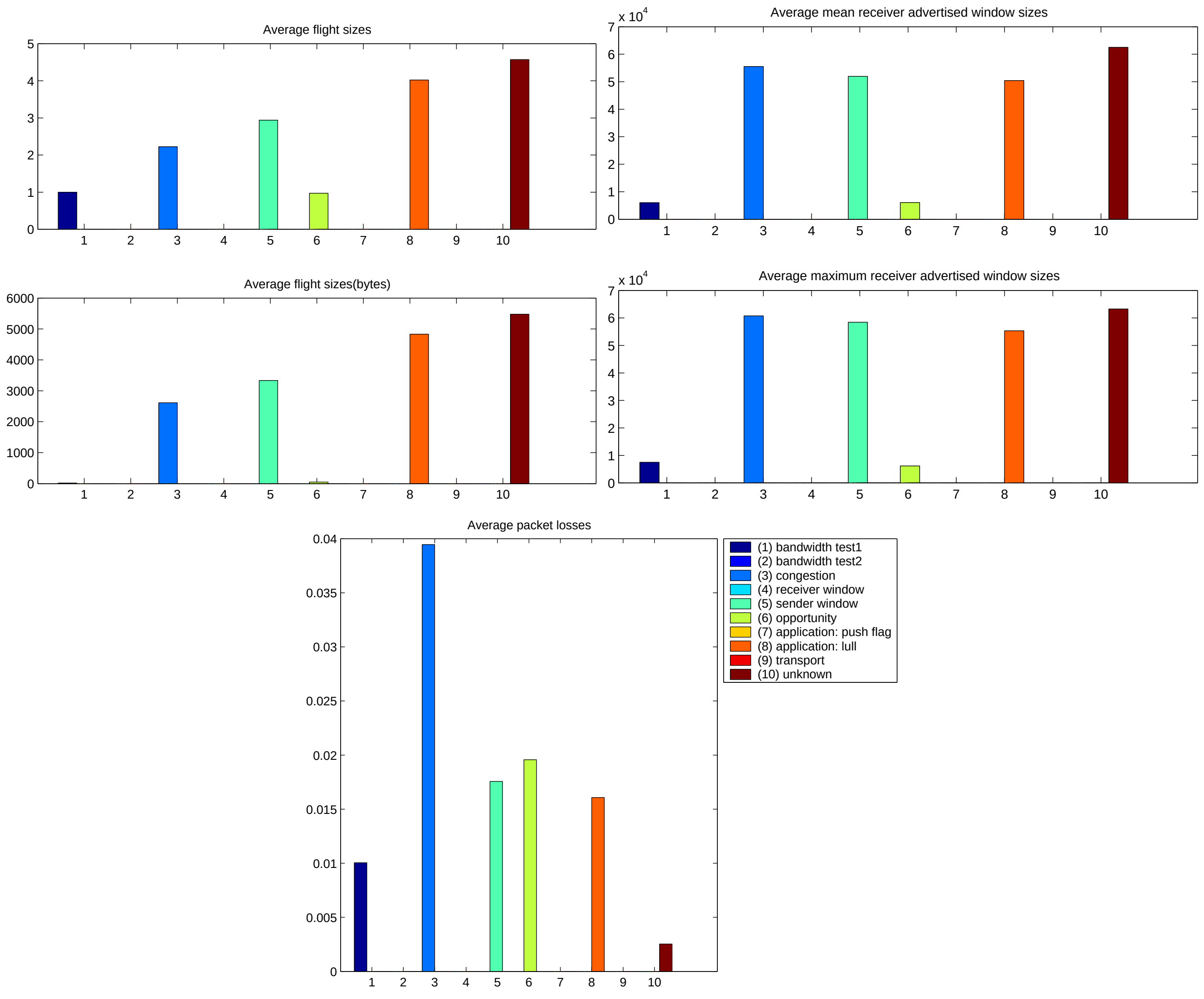


Figure 9. Different attributes of connections for BitTorrent trace.

6 Conclusions

We found out by applying T-RAT that most of regular Internet traffic seems to be mostly application limited but also window and congestion limited. We also noticed that T-RAT is able to capture some of the specific features of a peer to peer application BitTorrent from its traffic trace. However, on the way we have somewhat discovered the limitations of T-RAT as well. It seems to be easily fooled by certain kinds of traffic. Especially if the traffic is very alternating in the sense that it has clearly different phases during the lifetime. More specifically there seems to be a sort of trade-off between flow and connection level analysis and in the former one also in the length of the flow, granularity so to speak. We are able to discover different characteristics of a long connection depending on whether it's broken into flows. This suggests a some kind of smart combination of these methods. Time will tell if this kind of extension is going to be done in the future or not.

This report encloses more or less the work I've performed during the first three months as a Ph.D. candidate. It has been on and around T-RAT tool and we have discovered some interesting things with it. However, now is the time to think if it is worthwhile to pursue further with it. On one hand we have seen what it can tell us but on the other hand there's also room for improvements. The question is whether they would bring enough added value.

References

[1] The official bittorrent home page: http://bitconjurer.org/bittorrent/.

[2] The well known ports assigned by iana: http://www.iana.org/assignments/port-numbers.

[3] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, Jim Kurose, and Don Towsley. Inferring TCP connection characteristics through passive measurements. In *To appear in IEEE Infocom*, Hong Kong, 2004.

[4] Christian Schleippmann. Design and implementation of a tcp rate analysis tool. Master's thesis, TU Muenchen/Eurecom, July 2003.

[5] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. On the characteristics and origins of internet flow rates. In *Proceedings of the 2002 ACM SIGCOMM Conference*, August 2002.