****

ATM Forum Document Number: ATM_Forum/96-0518

****

Title: Performance of TCP over UBR and buffer requirements

****

Abstract:

We study TCP throughput and fairness over UBR for several buffer and maximum window sizes. TCP performs best when there is no loss. The performance is severely degraded under loss (regardless of UBR or ABR). Therefore, we study the switch buffer requirements for zero loss. The required buffers are found to be the function of number of VCs and their round-trip times.

****

Source:

Raj Jain, Rohit Goyal, Shiv Kalyanaraman, and Sonia Fahmy.
The Ohio State University

Raj Jain is now at Washington University in Saint Louis, jain@cse.wustl.edu http://www.cse.wustl.edu/~jain/

The presentation of this contribution at the ATM Forum is sponsored by NASA.

****

Date: April 1996

****

Distribution: ATM Forum Technical Working Group Members
(Traffic Management)

****

****

INTRODUCTION

TCP congestion control mechanisms effectively recover from segment loss and also avoid congestion collapse. It is useful to study the performance of TCP over the Unspecified Bit Rate (UBR) service provided by ATM networks. UBR provides no congestion control mechanisms. Enhancements have been proposed to UBR that propose intelligent cell drop policies at the switches.

The purpose of this and our subsequent contributions on this topic is to characterize the performance of TCP over UBR and its various enhancements. This particular contribution concentrates on the effect TCP receiver window, switch buffer size, and cell drop policy on TCP performance.

In our other contributions [8,10] and papers [9], we have studied the performance of TCP/IP over ABR. In [10], we characterize ABR buffer requirements with TCP sources.

SOME OBSERVATIONS ABOUT TCP

TCP congestion control mechanisms consist of a "slow start" and "congestion avoidance". Slow start, or exponential rise phase is triggered whenever a TCP source detects network congestion indicated by the triggering of the retransmission timout. Slow start results in the reduction of the TCP congestion window to one segment. The congestion window then doubles every round trip until the "congestion avoidance" or linear rise phase is reached. During this phase, the TCP source increases the congestion window by one segment every round trip. As a result, if the TCP congestion window is allowed to increase, TCP can potentially use all available network capacity.

The TCP congestion control mechanisms successfully avoid congestion collapse. Slow start empties the TCP pipe every time it detects loss. Although slow start limits the packet loss, it loses considerable time whenever there is loss. It takes several roundtrips for TCP to return to its optimal operating point. As a result TCP throughput decreases considerably when segments are dropped. TCP performance is optimal when there is zero segment loss. Even a single segment loss can decrease throughput considerably.

TCP Reno includes the Fast Retransmit and Fast Recovery algorithms that improve TCP performance when a single segment is lost. However, in high bandwidth links, network congestion results in several dropped segments. In this case, fast retransmit and recovery are not able to recover from the loss and slow start is triggered. Fast retransmit and recovery are effective in single packet losses typically due to error. In our experiments, all losses are due to congestion and result in multple segments being dropped. Therefore, we first study TCP without fast retransmit and recovery running on UBR.

TCP begins slow start whenever the retransmission timer is triggered. The accuracy of this timout depends on the timer granularity used in the TCP implementation. Most current TCP implementations measure round-trip delays using a granularity of 100 or 500 ms. Larger granularity means that the sources have to wait longer before timing out and recovering from a lost packet. All our simulations use a timer granularity of 100 ms.

PARAMETERS:

TCP MAX.WIN

TCP maximum window size plays an important role in its congestion dynamics. The default maximum congestion window is 65536 bytes. However, in high delay links, this window is too small to achieve full throughput. On the other hand, having maximum window above the RTT is not fruitful, since, at any time, only 1 RTT worth of segments can be in the TCP pipe. The maximum window size determines the amount of data that can be present in the TCP pipe. As a result, this parameter determines the storage capacity needed in the network. We experiment with various values of the TCP maximum window size.

SWITCH BUFFER

UBR provides no explicit feedback to its sources. Multiple TCP sources may dump a window of segment each on any UBR switch. As a result, the switch may need to store the cells from segments of all the TCP windows. Based on the MAX.WIN parameter used, we use various buffer sizes in our simulations.

The following variations of MAX.WIN and switch buffer sizes were simulated:

```
Set 1: These consist of Switch buffers smaller than 1 RTT and
       relatively large TCP maximum windows.
       a. TCP.MAX.WIN = 600000 bytes, Switch Buffer = 4096 cells
       b. TCP.MAX.WIN = 600000 bytes, Switch Buffer = 8192 cells

Set 2: These consist of Switch Buffers = sum of all TCP maximum
       windows for five sources. The window sizes correspond to
       being less than, equal to and greater than 1 RTT.
       a. TCP.MAX.WIN = 65536  bytes, Switch Buffer = 330000 cells
       b. TCP.MAX.WIN = 120000 bytes, Switch Buffer = 12000  cells
       c. TCP.MAX.WIN = 360000 bytes, Switch Buffer = 36000  cells

Set 3: Full factorial of:
       TCP.MAX.WIN = 600000, 18000000 bytes
       Switch Buffer = 12000, 36000 cells
```

Here the switch buffers are some multiple (less than the number of sources) of the TCP maximum windows.

CELL DROP POLICY

In its simplest form, a switch could implement a tail drop policy in which all cells arriving after the buffers are full are dropped. This is expected to result in excessive wasted bandwidth. If cells are dropped from multiple TCP packets, then all these packets need to be retransmitted by the source TCP. Early Packet Discard (EPD) has been suggested to remedy the problems caused by tail drop. EPD tries to discard cells from the same TCP packet during congestion. A threshold is set at the switches, and when the switch queue length exceeds this threshold, cells from any new packets are dropped. The EPD algorithm is the one suggested by [3,7]. However, EPD makes no attempt at achieving fairness among different VC's. It is also not known how to choose the threshold for the switch buffer. The choice of EPD threshold value may effect performance. We experiment with different buffer sizes and EPD thresholds.

CONFIGURATION

Our simulations use a five source configuration as illustrated below.

```
S1----|                         |---->D1
      |                         |
S2----|                         |---->D2
```

```
      |                         |
S3----|-->Sw1---------->Sw2---->|---->D3
      |                         |
S4----|                         |---->D4
      |                         |
S5----|                         |---->D5
```

* All link delays are 5 milliseconds. Thus, the Round Trip Time
  due to the propagation delay is 30 ms.

* All link bandwidths are 155.52 Mbps

* PCR = 155.52 Mbps

* All sources are infinite TCP sources. TCP layer always has a
packet to send as long as permitted by the TCP window. The actual
traffic as seen by the ATM layer is bursty.

* The traffic is unidirectional. Only the sources send data. The
  destination sends only acks.

* TCP Fast Retransmit and Recovery are not implemented.

* TCP segment size = 512 bytes.

* TCP timer granularity = 100ms.

* TCP maximum window size: parameter.

* TCP delay ack timer is NOT set. Segments are acked as soon are
they are
  received.

* Duration of simulation runs: 2 seconds.

METRICS:

We measure performance based on the throughput as follows:

TCP throughput: This is measured at the destination TCP layer.
The destination measures throughput of all the segments received
in sequence. If a segment is received out of sequence, (due to
error/loss of previous segments) it is not included in the
throughput measurement until all the missing segments are
received.

We can also plot TCP sequence numbers measured at the source TCP
layer.

SIMULATION RESULTS

The simulation results are summarized in the table below.

Set 1:

Buffer	MAX.WIN	Throughput Mbps					
cells	bytes	D1	D2	D3	D4	D5	Total
4096	600000	5.56823	35.6997	2.62913	4.49441	4.13301	52.52448

8192	600000	6.21502	4.28641	5.84917	3.2953	7.06766	26.71356

Set 2:

Buffer	MAX.WIN	Throughput Mbps					
cells	bytes	D1	D2	D3	D4	D5	Total
330000	65536	15.475	15.4554	15.3671	15.4028	15.3712	77.0715
12000	120000	22.1366	22.089	21.6766	21.9468	21.743	109.592
36000	360000	22.00	22.2327	21.7957	22.0239	21.8398	109.8921

Set 3:

Buffer	MAX.WIN	Throughput Mbps					
cells	bytes	D1	D2	D3	D4	D5	Total
12000	600000	13.1048	13.4268	5.91446	19.417	6.55147	58.41453
36000	600000	17.159	18.238	4.0895	16.0899	11.1238	66.7002
12000	1800000	13.1048	13.4268	5.91446	1.9417	6.55147	58.41453
36000	1800000	15.00	9.98533	3.46452	14.1859	9.22931	51.86506

From the above results, we observe the following:

Set 1 uses switch buffers less than 1 RTT*Bandwidth (11040 cells). We observe repeated loss. The table shows that the case with switch buffers = 8192 cells performs worse (aggregrate throughput is less) than when the buffer = 4096 cells. Hence, when there is cell loss, increasing UBR switch buffers does not necessarily mean increased TCP throughput.

Set 2 shows simulations where there is no loss.
The first simulation in this set shows that the maximum throughput is less than optimal. This is because the TCP maximum window size is less than 1 RTT (approx. 120000 bytes). As a result, the TCP source could not fill the available pipe capacity and hence the low throughput resulted. For a single TCP source to achieve maximum possible throughput, its maximum window size must be greater than 1 RTT*bandwidth. In the second and third simulations, the total throughput is close to the optimal (minus overhead due to the TCP/IP/ATM headers) value. In these cases, all sources obtained an equal amount of bandwidth.

In the cases covered by Set 2, the switch buffers were at least n*TCP.MAX.WIN. Therefore, to ensure zero cell loss, either TCP.MAX.WIN must be controlled to 1/nth of the smallest switch buffer (where n is the number of sources), or the UBR switches must have buffers as much as the sum the maximum TCP windows of all the sources. Since every source needs to have a maximum window size greater than 1 RTT*bandwidth, it is the switch which must buffer the sum the maximum TCP windows of all the sources. Since the required buffering depends upon the number of sources, zero loss UBR may be expensive for networks with large number of sources.

Set 3 shows more simulations varying the TCP MAX.WIN and switch buffer, such that the switch buffers are less than n*TCP.MAX.WIN. There is more than 50% drop in total throughput. There is some unfairness in the bandwidths obtained. However, the unfairness is not consistent, in the sense that, a single source is not given preferential treatment throughout the simulation. There were periods when any source obtained higher throughput than any other source. Similarly, no single source is starved either.

It should be noted that these experiments of UBR did not use special policies for dropping cells. We see that the simple tail drop policies result in unfairness among the TCP connections and substantial wasted bandwidth. Fairness and throughput can be improved by using better buffer allocation and drop policies. Drop policies are more critical for UBR than ABR to improve throughput. We show in [10] that the switch scheme is more important than the drop policies for ABR. We will present further results with UBR using drop policies at the ATM Forum meeting.

CONCLUSIONS

* Switch queues may be as high as the sum of the TCP windows. Zero cell loss for TCP requires that the buffers equal at least the sum of the TCP maximum windows. The required buffers thus depends on the number of TCP sources.

* TCP receiver window must be greater than 1 RTT for full throughput with one source.

* Cell loss results in unfairness in many cases. However, no single TCP source is given preferential treatment and no TCP sources are starved.

* Cell loss results in lower TCP throughput. This decrease in throughput shows up as increased file transfer times or a lower network capacity (but not generally as broken connections).

* Fairness may be improved by proper buffer allocation and drop policies.

* UBR may be a viable low cost solution for LANs without VBR and with a limited number of sources.

REFERENCES:

[1] Chien Fang, Arthur Lin: "On TCP Performance of UBR with EPD and UBR-EPD with a Fair Buffer Allocation Scheme," AF-TM 95-1645, December 1995.

[2] Chien Fang, Arthur Lin, "A Simulation Study of ABR Robustness with Binary-Mode Switches: Part II," AF-TM 95-1328R1, October 1995.

[3] Hongqing Li, Kai-Yeung Siu, and Hong-Ti Tzeng, "TCP over ATM with ABR service versus UBR+EPD service," AF-TM 95-0718, June 1995.

[4] AF-TM 95-0503, Hongqing Li, Kai-Yeung Siu, Hong-Yi Tzeng, Brian Hang, Wai Yang, "Issues in TCP over ATM," April 1995.

[5] Tim Dwight, "Guidelines for the Simulation of TCP/IP over ATM," AF-TM 95-0077r1, March 1995.

[6] Stephen Keung, Kai-Yeung Siu, "Degradation in TCP Performance

under Cell Loss," AF-TM 94-0490, April 1994.

[7] Allyn Romanov, Sally Floyd, "Dynamics of TCP Traffic over ATM Networks."

[8] Raj Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, F. Lu, and S. Srinidhi, "TBE and TCP/IP traffic," AF-TM 96-0177, February 1996.

[9] S. Kalyanaraman, Raj Jain, S. Fahmy, R. Goyal, F. Lu, and S. Srinidhi, "``Performance of TCP/IP over ABR,'' Submitted to Globecom'96.

[10] Raj Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, F. Lu, and S. Srinidhi, "Buffer requirements for TCP over ABR" AF-TM 96-0517, April 1996.

All our past ATM forum contributions/presentations, and recent papers can be obtained on-line:
   http://www.cse.wustl.edu/~jain/