Sizing and Management of Router Buffers

H. T. Kung
Harvard University

May 1998

with Graduate Students:
D. Lin, R. Morris and S. Y. Wang

Motivations

- **When building or configuring routers, one needs to size the packet buffer memory.**
- **However, sizing router buffers has mostly been a black art. It is hard to get a straight answer for the simple question: “how big should the buffer be?”.**
- **This does not provide much comfort for router developers and service operators who must make hard decisions on the size of router buffers.**

Choices for Buffer Size /1

C1: Small Buf_Sz

- But Buf_Sz must reflect RTT*Link_BW to accommodate flow control delay, and also to keep the link to the next hop busy.

C2: Buf_Sz = RTT*Link_BW

- But each of the active flows will need RTT*Link_BW.

C3: Buf_Sz = N*RTT*Link_BW

- But for N = 1000 flows and for RTT = 100ms and Link_BW = 10Gbps, total Buf_Sz will be about N*100MByte or 100GByte.
- Even assuming that the router can afford this large buffer, a large delay of 100s (=N*.1s*10/10) inside the network is not acceptable.

Choices for Buffer Size /2

C4: Buf_Sz = 100ms*link_BW or any seemingly reasonable compromise

- But, how reliable are such guesses? Will #flows matter?
- Where is the science?

C5: Use provably correct Buf_Sz

- This presentation will argue that this is feasible, and that some good reasoning can back up choices of Buf_Sz.

How to Think About the Problem of Sizing Buffers?

We need to decide on the goal of sizing and managing router buffers.

That is, establish the optimization criteria.

TCP Retransmission Time-outs (RTOs) Have Been a Problem

- Interactive web traffic, for which network performance is most critical (as perceived by many users), all uses TCP.
- Interactive web users often experience TCP RTOs due to congestion-induced packet loss These TCP RTOs typically last seconds or (much) longer.

Goal: Minimize Unnecessary TCP RTOs /1

- Interactive web users can typically live with small BW as low as 20kbps, provided that they don’t experience TCP RTOs. (A user takes seconds to read a page and click a link anyway.)
- Are these TCP RTOs necessary? No. For example, a backbone link of bandwidth 622mbps can support as many as 30,000 flows each at 20kbps. Today’s backbone links rarely carry this many flows.

Goal: Minimize Unnecessary TCP RTOs /2

- Most interactive timeouts could be avoided.
- Thus, an important goal for sizing and managing router buffers should be to minimize this “unnecessary TCP RTOs”.

First, We Introduce a Notion: “Small vs. Large TCP”

Definition:

A “small” or “large” TCP connection is, respectively, a TCP connection with a small or large window.

Causes for Unnecessary TCP RTOs /1

(1) A small TCP is fragile in the sense that any packet loss will likely trigger a TCP RTO.

- A window smaller than 4 or 5 packets will not allow fast retransmission/recovery under any single packet loss.
- A window smaller than 10 packets will likely not allow fast retransmission/recovery under two or more packet losses.

Causes for Unnecessary TCP RTOs /2

(2) Router buffer management is generally unfair in the sense that certain TCP connections will occupy the buffer much more than the others. Moreover a TCP that is already large tends to occupy more buffer over time, and thus to increase its window faster, than a small TCP. (This is true, for example, for conventional FIFO buffers that use the “drop-tail” packet-discard policy.)

Causes for Unnecessary TCP RTOs /3

Since web sessions typically involve a small number of packets (e.g., tens of packets), they rarely ramp up their TCP windows beyond eight packets. Thus, these are small TCPs for which unnecessary TCP RTOs can happen easily.

Fixes for Unnecessary TCP RTOs /1

Approach: Cooperative “TCP sender algorithm” and “router buffer management algorithm” to minimize unnecessary TCP RTOs.

(1) TCP Sender Algorithm: A TCP sender will make sure that the TCP connection will not time out, as long as it has at least one packet alive on the network. (Lin and Kung: INFOCOM’98)

Fixes for Unnecessary TCP RTOs /2

(2) Router Buffer Management Algorithm: A router will keep at least one packet alive for each active TCP connection. That is, it will not drop all packets in the current TCP window.
This is feasible provided that the router buffer can hold a total of N packets, where N is the number of active TCP connections. (Lin and Morris: SIGCOMM’97)

Sizing Router Buffers

Let N = #TCP connections sharing a router buffer

- Assume ideal TCP sender and router buffer management algorithms.
  Then Buf_Sz = N packets.
- Assume approximations to ideal TCP sender and router buffer management algorithms.
  Then Buf_Sz = k*N packets, where parameter k decreases to 1 for high-quality approximation.

Buffer Management Algorithms for FIFO Buffer /1

- Drop Tail
  - When drop occurs, drop probability for a packet is the same between small and large TCP
- Random Drop
  - Drop probability for a packet is the same between small and large TCP

Buffer Management Algorithms for FIFO Buffer /2

- **RED**
  - **Drop probability for an arriving packet is the same between small and large TCP**
  - **Avoid synchronization and burst-arrival problems**
- **FRED**
  - **Drop probability for an arriving packet from a large TCP is higher than that from a small TCP**
  - **Favor small TCP**

Ideal TCP Sender Algorithms

When there is only one or a few in-flight packets, TCP sender will inject a packet into the network when receiving an ACK, independent of current congestion window. (Packet conservation)

This will make sure that the TCP connection will not time out, as long as it has at least one packet alive on

Conclusions and Ongoing Research /1

- Sizing and management of router buffers should be aimed at minimizing unnecessary TCP RTOs for small TCP connections, e.g., interactive web sessions.
- Buffer size should be k*N packets, where N is the expected number of TCP connections sharing the buffer, and parameter k reflects quality of TCP sender and router buffer management algorithms.

Conclusions and Ongoing Research /2

- Harvard traces show the number of TCP flows increases as network bandwidth increases.
- Ongoing Research: Use of “TCP trunks” to reduce the number of flows on backbone, and to provide traffic separation. Multiple TCP trunks can dynamically share the same queue without flow identiy. Simulation results have demonstrated the effectiveness of TCP trunking.