Traffic: Host A sends 1000-byte pkts to B using a doubling slow-start algorithm until an ACK pkt is lost.

Problem: What is the sequence number of the first pkt that will be dropped if the R1 output queue can hold 600,000 bytes?

Questions:
» Give an estimate of the sequence number of the first pkt drop.
» What will you see in a solution?

Like Lab 2 Setup
- xrcv
  » In-order accept policy
  » ACK in-order pkt
  » Silently drop all others
- xsnd
  » Send pkts over UDP
  » Internal hdr
    » Pkt sequence number: 0, 1, 2, ...
    » Timestamp
  » Doubling slow-start
    » Initial: \( \text{win} = 1, \text{nxtsn} = 0 \)
    » Rev ACK: \( \text{win}' = 2 \times \text{win} \)
      » Send next \( \text{win}' \) pkts
  » Timeout
    » \( \text{win}' = 1 \)
      » Send next pkt

The Lab Solution (600,000B Queue)
Topics
1) NPR data path
2) Packet-pair example
   • meaning of “port rate”
   • ~/onl/.topology and ~/onl/.topology.csh shell scripts
3) Filters and queues
4) Using the delay plugin
5) Traffic generation scripts

The NPR Data Path

NPR Packet Processing (1)

NPR Packet Processing (2)
NPR Packet Processing (3)

- **Rx:** Receive Pkt
  - Put pkt in DRAM; Send meta-packet
- **Mux:** Multiplex traffic (inputs, CP, plugins)
- **PLC:** Parse, Lookup and Copy
  - Implements Route Table and Filter Table lookup
  - Uses TCAM (Ternary Content-Addressable Memory)
- **QM:** Queue Manager
  - Pkt scheduler for each of 5 output ports
  - Implements port rate concept using token bucket
  - Implements bandwidth sharing concept using Weighted Deficit Round Robin (WDRR) algorithm
- **Hdr Format**
  - Create ethernet frame
- **Tx:** Transmit Ethernet Frame

Queue Table ➔ Port Rate

- Controlled by a token bucket model
  - R: long-term average link rate (RLI parameter)
  - 54 Kbps granularity
  - b: maximum bucket depth (4000 bytes)
  - R': physical link rate (1 Gbps)
- **Effect**
  - avg output rate is R
  - peak rate is R'
- **Operational definition**
  - fill token bucket at rate R
  - transmit pkt when #tokens ≥ length of pkt at head of queue
- **Link regulator** model, NOT link emulator

The Meaning of Port Rate

- **Rx:** Receive Pkt
  - Put pkt in DRAM; Send meta-packet
- **Mux:** Multiplex traffic (inputs, CP, plugins)
- **PLC:** Parse, Lookup and Copy
  - Implements Route Table and Filter Table lookup
  - Uses TCAM (Ternary Content-Addressable Memory)
- **QM:** Queue Manager
  - Pkt scheduler for each of 5 output ports
  - Implements port rate concept using token bucket
  - Implements bandwidth sharing concept using Weighted Deficit Round Robin (WDRR) algorithm

Queue Table ➔ Port Rate

- Controlled by a token bucket model
  - R: long-term average link rate (RLI parameter)
  - 54 Kbps granularity
  - b: maximum bucket depth (4000 bytes)
  - R': physical link rate (1 Gbps)
- **Effect**
  - avg output rate is R
  - peak rate is R'
- **Operational definition**
  - fill token bucket at rate R
  - transmit pkt when #tokens ≥ length of pkt at head of queue
- **Link regulator** model, NOT link emulator

The Packet-Pair Problem

- **Keshav’s idea**
- Rcvr receives the two pkts at times t0 and t1
  - What will the interpacket time t1-t0 be?
  - average interbit time?
ONL Packet-Pair Experiment (1)

link capacity 1 Gbps 12 Mbps 1 Gbps

Snrd A > R1 > R2 > B Rcvr

Ideal:

T = L/1 Gbps

T = L/12 Mbps

ONL:

T = L/11.611 Mbps

note initial burst

R1

B

R2

A

1 Gbps 12 Mbps 1 Gbps

T = L/11.611 Mbps

note initial burst

R1

B

R2

A

1 Gbps 12 Mbps 1 Gbps

ONL Packet-Pair Experiment (2)

Puts $n1p2$, etc. into Linux bash environment

rcvr: n2p1 # pkts: 7

20-byte IP hdr
8-byte UDP hdr
1470-byte payload

Filters And Queues

Directing Pkts to Port 4, Queue 64

Filter F forwards pkts to queue 64

Route table would forwards pkts to one of 64 datagram queues based on hash
Filters and Queues

Parse, Lookup, & Classify

Route Table

Highest Priority

Filter Table

Meta-packet

Packet headers

TCAM

5-tuple

match?

Key

Action

Control

do action if selected

Use filter to direct pkt to NPR queue or plugin or drop

Input Port 3: Forward to Output Queue

send matching pkts to queue 64, output port 4

0 means: qid = hash()
Filter F forwards pkts to delay plugin D which will delay pkts by 50 msec.

Filter

XScale

TCAM

Freelist Manager

Stats

Rx (2 MEs)

Mux (1 ME)

TCAM

Queue Manager

Plugin 1 (1 ME)

Plugin 2 (1 ME)

Plugin 3 (1 ME)

Plugin 4 (1 ME)

Plugin 0 (1 ME)

Any src/dst address, port and protocol

Delay plugin will delay pkts by 50 msec.

Add Filter to Direct Pkts to Delay Plugin
Sending Commands to Plugin

- npkts = 82008
- maxinq = 43
- ndrops = 0
- reset counters

Changing Delay to 25 msec

- note SPACE after delay=

Traffic Generation Scripts

- Lab 2 Scripts
  - Puts $n1p2, etc. into Linux bash environment
  - Run "xrcv ..." on hosts $n2p1, $n2p2 and $n2p3
  - Send 8000 1000-byte pkts
  - Normally, run scripts from host onlusr
Iperf UDP Traffic

Run "iperf ..." on hosts $n2p1$, $n2p2$ and $n2p3$

Send at 200 Mbps for 30 seconds

Run as UDP client sending to n2p1

1 MB receive socket buffer

10 second staggered start

See ~kenw/bin/

Iperf TCP Traffic

16 MB sender socket buffer

16 MB receiver socket buffer ≥ BDP

See ~kenw/bin/

Simple Features of an Approximate Solution

- Transmission Rate
  - \( R = 12 \text{ Mbps} = 1.5 \text{ Kpps} \) since
  - Transmission Delay = 2/3 msec

- Bandwidth Delay Product
  - \( BDP = 1.5 \text{ Kpps} \times 100 \text{ msec} = 150 \text{ pkts} \)

- Queue Capacity
  - \( Q = 600,000 \text{ bytes} = 600 \text{ pkts} \)

- Max Queueing Time
  - \( \text{Max Qtime} = 600 \text{ pkts} / 1.5 \text{ Kpps} = 400 \text{ msec} \)

- Max RTT
  - \( \text{Max RTT} = \text{Prop. Delay} + \text{Queuing Delay} = 500 \text{ msec} \)

- \( 2^k \) pkts transmitted in round \( K, K = 0, 1, 2, ... \)
Transmission Doubling

Space-Time Diagram
- Transmission proceeds in rounds
  - About 100 msec for each round (actually 102 msec)

Einstein’s Observations At R1
- In the early 100 msec rounds:
  - sees an empty queue
- Eventually, sees non-empty queue
  - pkts in queue ➔ overload in previous round
  - when input rate > transmission rate at R1
  - sustained queuing begins when (approximately):
    - \(2^K \text{ pkts} / 100 \text{ msec} > 1.5 \text{ Kpps}\)
    - \(2^K > 150 \text{ pkts}\) ➔ \(K = 8\) ➔ \(1500 \text{ pkts per sec}\)
- When the bottleneck is overloaded:
  - 2 pkts queued during each transmission period
    - Queuing rate = 1 pkt per transmission period
    - Max input rate = 2 x bottleneck transmission capacity

Einstein’s Solution
- What would we see if we rode on top of pkt 0 and all resulting pkts ???
  - pkt 0, its ACK, pkt 1, its ACK, pkt 3, its ACK, etc
- How many pkts would I see in R1’s output queue ???

Approximate Solution (R = 12 Mbps)