Buffering

Outline

Should a switch buffer?: Forwarding modes
  Store-and-forward
  Cut-through

How to buffer
  Memory technologies
  FIFOs, Where to buffer
  priority queues, per-flow queueing

What if the buffer overflows?
  Congestion – definition and TCP response
  Discard policies

Reasons for switch buffering

To gain time: hold packets while doing things such as classification
To avoid packet loss, during times of:
  • Contention: Competition for the same output. Only one packet can go out at a
time; others must be buffered.
  • Short-term overload: Input rate exceeds output rate; need to buffer some inputs
  so that they can be output later.

  Buffering can only accommodate short-term overload.
  In the long term, real buffers (of finite size) will overflow.

For reordering:
  • after mis-sequencing in the network (e.g. recirculating Banyan)
  • for TSI switches

For reduce burstiness: To prevent overload downstream.

Resources

Buffering: Keshav § 8.5
G. Appenzeller, I. Keslassy and N. McKeown: 'Sizing router buffers',
Proc. SIGCOMM, pp. 281-92, Aug. 2004

Partial Packet Discard: Keshav § 9.7.2
Random Early Detection:
  • Keshav § 9.7.3
  • RFC 2309

Explicit Congestion Notification:
  • Keshav § 13.4.9
  • RFC 3168

† Known as Explicit Forward Congestion Indication (EFCI) in ATM, Forward
ECN (FECN) in Frame Relay.
Goals for buffering

- Low chance of loss from overflow
- Small buffers
- Delay
- Implementation cost
- Simple (e.g. prefer FIFO)

Review: Statistical Multiplexing

Observations
1. The bigger the buffer, the lower the packet loss.
2. If the buffer never goes empty, the outgoing line is busy 100% of the time.

Review: TCP Congestion Control

Rule for adjusting
- If an ACK is received: $W \leftarrow W + 1/W$
- If a packet is lost: $W \leftarrow W/2$

Example

- 10Gb/s linecard
  - Rule-of-thumb: 250ms of buffering (RTT)
  - Requires 300Mbytes of buffering.
  - Read and write 40 byte packet every 32ns.
- Memory technologies
  - SRAM: require 80 devices, 1kW, $2000$
  - DRAM: require 4 devices, but too slow.
- Problem gets harder at 40Gb/s

Example

- 10Gb/s linecard
  - Rule-of-thumb: 250ms of buffering (RTT)
  - Requires 300Mbytes of buffering.
  - Read and write 40 byte packet every 32ns.
- Memory technologies
  - SRAM: require 80 devices, 1kW, $2000$
  - DRAM: require 4 devices, but too slow.
- Problem gets harder at 40Gb/s
Outline

Forwarding modes

Forwarding could start as soon as packet has been classified; DA in header+fast classification ⇒ could start forwarding before finish receiving, e.g. Ethernet frame:

<table>
<thead>
<tr>
<th>Destination Address</th>
<th>Source Address</th>
<th>Type</th>
<th>(Data)</th>
<th>(Pad)</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6B</td>
<td>6B</td>
<td>2B</td>
<td>0-1500B</td>
<td>0-46B</td>
<td>4B</td>
</tr>
</tbody>
</table>

⇒ got info for classification delay could start now
got check store-and-forward can start now

Forwarding modes:
• **Store-and-forward**: Store complete frame before forwarding. ✓ Saves transmission capacity.
• **Cut-through**: Start forwarding frame as soon as know output port and port is free. ✓ May reduce delays.

Types of erroneous frames

Frames with bit errors (lack integrity) are usually detected by CRC mismatch.
Frames with incorrect length:
• **Runt**: too small
  • Most protocols have frame overheads (e.g. 802.3: 18B, 802.11: 28B)
  ⇒ frames shorter than minimum overhead are invalid
• **Giant**: too large
  • Protocols limit frame length to prevent large serialisation delays during periods of contention, e.g. Ethernet payload ≤ 1500B ⇒ frame > 1518B is invalid

† Only MAC overheads are shown. Physical layer preambles are also overheads.

Store-and-forward

Classification:
• Can start after receipt of required info, e.g. progress in parallel with receipt of payload.
• Simpler approach: leave it until after verified frame.

Switch:
1. Waits to receive complete frame
   ✓ Delay
2. Verifies validity (integrity and length) of frame
3. Forwards frame if valid; discards if erroneous
   ✓ Confines propagation of erroneous frames, in particular giant frames or those without integrity.
   ⇒ won’t waste transmission capacity forwarding frames that will ultimately be discarded by the receiver anyhow.
Cut-through forwarding

Start forwarding before frame has been completely received

✓ Can reduce the delay across the switch
  \[ \text{delay}_{\text{cut-through}} = \frac{\text{frame length}}{\text{transmission rate}} + \frac{\text{classification delay}}{\text{transmission rate}} \]
  e.g. 4.8us for 10Mb/s Ethernet + classification delay

May become important when delays accumulate as path traverses multiple switches

Alternative technique is to reduce frame size, e.g. ATM

Note that “cut-through forwarding” is distinct from “cut-through routing” [Keshav, p. 537]
Also, “cut-through forwarding” is distinct from “buffer cut-through” in which layers pass pointers to data between themselves, rather than the data itself.

Limitations of cut-through

• May have to store frame anyhow if output port is busy
  • Subject to MAC, e.g. CSMA/CD access delays
  • Possible only when there is no queuing at the output port
• Difficult when input rate < output rate†
  e.g. frame of 1kB takes 800us @ 10Mb/s, and 80us @ 100Mb/s
  if start forwarding frame at time 4.8us will run out of data to send
  volume equal to 0.53us (at 5.3us)
  • Need to wait (720us) to receive enough data from input to sustain output
• Difficult when transmitting on multiple output ports (e.g. multicast)
  • Some ports may be busy while others aren’t
  • Output ports may have different rates
  ⇒ “Cut-through is not an alternative to store-and-forward operation, it is in addition to it.”

† Cisco terms: “synchronous switching”: forwarding packets between ports operating at same speed, and “asynchronous switching” if ports have differing speeds.

Variations of cut-through

• “Fast forward”: Cisco term for plain cut-through (to distinguish it from —)
• “Fragment-free cut-through”:
  • Ensures frame exceeds minimum length (little additional delay)
  • Cuts-through once this condition is met, preventing propagation of runt frames.
• Adaptive:
  Many switches operate in an adaptive mode:
  1. Measure frame error rates.
  2. If error rate is low (e.g. <10%†)
    use cut-through: low delay, few erroneous frames propagating
  else
    use store-and-forward: increases delay but confines erroneous frames

† Default value for Cisco Catalyst 2900 switch
Cut-through & layer of operation

“Switches can cut-through, but routers can’t. So switches are faster”

**Theory:** Layers are separate, and higher layer doesn’t start processing incoming packet until lower layer has finished with it. Interpretation of network layer (IP) fields can’t start until frame has been fully received and link layer has checked integrity.

**Practice:**
- Switches at low layers may interpret fields from higher layers. e.g. frame switch may inspect IP address to determine outgoing port.
- Violates insulation benefits of layering, but provides marketing advantage (network layer routing, without high delays).

**Summary of forwarding modes**

- **Store-and-forward:**
  - Receives whole frame before starting to forward it.
  - Checks integrity before forwarding.
  - Can prevent erroneous frames being forwarded.

- **Cut-through:**
  - Can start forwarding as soon as classified and output is free
  - Potentially low delay.
  - May have to store-and-forward if output isn’t free.
  - Checks integrity while forwarding.
  - Can’t prevent erroneous frames being forwarded
  - But can monitor error rate. Adaptive system will switch to store-and-forward if past error rate is high.

**Outline**

- Video RAM
  - CPU can randomly access DRAM
  - Access to DRAM is shared between CPU and cache(s)

  **Cache(s):**
  - access DRAM one complete row at a time
  - can be accessed independently of DRAM, e.g. to feed video raster

  **Application to network switches:**
  - ports access caches
  - achieve switching by using DRAM to transfer between caches & for buffering
First In First Out queues (FIFOs)

- The simplest type of buffer because output order = input order
- Readily handles variable-length packets
  - May be implemented:
    - using a RAM as a circular buffer
    - as a dual-port memory chip (e.g. IDT7028L: 64K x 16)
    - optical delay line
- Discard policy is limited to:
  - Drop-Tail (don’t add to FIFO) or Drop-Head (discard first out)

FIFOs are very similar to First Come First Served (FCFS) queuing.
When a FCFS queue is served by a single server & all jobs have the same service time, the result will be FIFO.

Non-FIFO buffers

- Non-FIFO is difficult:
  - Need to maintain a list showing packet service order
  - With variable-length packets: Tradeoff between wasting space and having space scattered in useless positions.
    - Solution 1: Packets occupy contiguous bytes of memory according to length. If multiple short packets leave buffer, may have enough bytes to store a new long packet, but not in contiguous positions.
      ⇒ expensive shuffling for “garbage collection”
    - Solution 2: Packets occupy fixed-length pages of memory.
      - One page size: Simple, but too large for some packets ⇒ waste space
      - Multiple page sizes: Complex & might only have many small ones free. Only need a few different sizes (e.g. short for TCP ACK or long for TCP MSS)
    - Solution 2': Packet is fragmented into pieces, each stored in a fixed-length page of memory, linked in a list.
- Easier for fixed-length packets (ATM - may waste transmission capacity, or fixed length within switch)

Locating buffering

- Input ports: Resolve contention by queueing at input and sending through fabric when output is free.
- FIFO input buffers can cause head-of-line blocking
- Output ports: FIFOs are fine. Reliance on output requires fast switch and buffers since many inputs may simultaneously flow to one output.
- May need some output buffering because of variable delay in accessing shared medium.
- Within the switch fabric

Priority queues

Separate queues (e.g. linked lists) for different packets having different delay requirements, e.g. voice: high priority, file transfer low priority.

Service disciplines:
- **Preemptive**: Empty high priority queue before serving low priority queue. New high priority arrivals preempt existing low priority jobs.
- **Weighted**: Serve high priority queue more often than low priority queue. High priority gets better service, but not exclusive service.

Per-flow queueing

We seek some isolation between flows of information (from one source to another) s.t. if one transmits excessively fast, it won’t exhaust the buffer and lock out others.

- FIFO provides no isolation: Service is proportional to arrival rate.
- Ideally switch should have a separate queue for each flow

Identifying flows is:
- simple when the switch maintains state about connections (e.g. VCs):
  - Classify packet and connection state will record whether connection has recently sent a burst.
  - Store packet only if connection has been well-behaved.
  - another advantage of the connection-oriented ATM.
- hard when the switch lacks state (e.g. IP routers): No record of past activity
  ⇒ For IP routers prefer congestion control techniques that don’t need flow identification and can use FIFO queuing

Outline

Congestion

Definition: When the network service is poor because of excessively high load.

Informally: “too many sources sending too much data too fast for the network to handle”

Symptoms:
- Packet loss (load>capacity)
- Delays (load near capacity)

Congestion collapse: Network throughput deteriorates with increasing load:
- Packet loss causes retransmission
- Packet delays cause unnecessary retransmission.
  Both magnify the load on links near sources
Transmission Control Protocol

- TCP provides reliable unicast transfer for applications like web and email
- The early Internet lacked mechanisms to deal with congestion, leading to episodes of congestion collapse in the late 1980s.
- Too hard to change routers (often implemented in hardware)
  ⇒ change TCP (implemented in software & BSD Unix was widely used)

TCP’s congestion control

- Use packet loss as a congestion indicator (inappropriate for wireless networks)
- Increase rate linearly over time when there are no indications of congestion (to approach, and eventually surpass available capacity)
  - TCP also includes a Slow Start phase to expedite the increase when well away from congestion level, consisting of limited bursts of packets.
- Reduce rate multiplicatively over time (e.g. halve it) when source observes indications of congestion.
  i.e. TCP responds to packet loss by slowing down.

TCP congestion window (cwnd) (unit=MSS)

Path capacity (MSS/RTT)

Dropping old or new packets

Some things get better with age, some get worse.
Analogy often drawn with milk (gets worse) vs wine (gets better).
- Packets that get worse with age: Multimedia: Packet may arrive too late to be played out in time.
- Packets that get better with age (within limitations): Packets involved in reliable transfer: If using go-back-N retransmission, the smaller N is, the more that will need to be retransmitted.
  i.e. switch should drop:
  - TCP packets from tail (TCP provides reliable transfer- though not necessarily go-back-N)
  - UDP packets from head (often carrying multimedia)

Outline

Discard strategies
- According to position in queue
- Loss priorities
  - Source marking
  - Network marking
- According to packet length
- Discarding when buffers aren’t full:
  Partial Packet Discard
  Early Packet Discard
  Random Early Detection
  try to manage congestion
- Explicit Congestion Notification – an alternative to discard
Loss priorities: Source tagging

Some applications exchange packets that vary in their loss sensitivity.

- e.g. streaming media can often be coded hierarchically:
  - one level of low-fidelity baseband information
  - + one or more levels of enhancement information (less loss sensitive)

- e.g. normal definition TV + HDTV supplement.

Such applications benefit from the network discarding unimportant packets in order to transfer important packets.

(Priority is only relative to other packets from this application)

- Sources have incentive to “tag” packets, indicating their loss sensitivity

Implementing tagging

ATM: Cell Loss Priority bit in cell header

IPv4: TOS

IPv6: set the Traffic Class/Differentiated Services field to indicate the required per hop behaviour (drop priority)

Network tagging of loss priority

To guarantee service: Applications negotiate a contract with the network.

- Network agrees to provide service guarantees.
- Application agrees to abide by certain traffic profile (e.g. as described by a Leaky Bucket).

Network may “police” source traffic to ensure that it conforms to the agreed profile.

- Traffic that does not conform to the profile may be summarily discarded, or may be tagged for preferential discard within the network.
- Source can “get away” with transmitting excess traffic, but only if it doesn’t degrade service to others.

... end of tagging discussion.

Outline