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*
Resources

Buffering: Keshav § 8.4

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

Leaky Buckets: Keshav § 13.3.4

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

To reduce burstiness: To prevent overload downstream.

† Sometimes called “congestion”, though we’ll define congestion shortly.
Goals for buffering

• Low chance of loss from overflow
• Small buffers
  • Delay
  • Implementation cost
• Simple (e.g. prefer FIFO)
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  + classification  + cut-through forwarding 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) usually detected by CRC mismatch.

Frames with incorrect length:

- **Runts**: too small
  - Most protocols have frame overheads (e.g. 802.3: 26B, 802.11: 28B)
    - frames shorter than minimum overhead are invalid
  - Ethernet imposes minimum frame length (64B, excluding preamble) to ensure that source can hear any collision during frame transmission

- **Giants**: too large
  - Protocols limit frame length to prevent large serialisation delays during periods of contention.
    - e.g. Ethernet payload $\leq 1500$B $\Rightarrow$ frame $> 1518$B is invalid


Typical measurement of

- 9 032 273 frames
  - 4 694 (5E-4)

- 72 (8E-6)

- 2 059 (2E-4)
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 bandwidth 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
  \[ \text{delay}_{\text{min}} \text{(store-and-forward)} = \frac{\text{frame length}}{\text{transmission rate}} \]
  e.g. 1.2ms for 10Mb/s Ethernet and 1.5kB frame
  \[ \text{delay}_{\text{min}} \text{(cut-through)} = \frac{\text{DA length}}{\text{transmission rate}} + \text{classification delay} \]
  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
  - input volume 48b + 10M\(^*t\) = output volume 100M\(^*t\).
    - equal after 0.53us (at 5.33us)
    - ⇒ 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.
- Bridges
  - Learn location of nodes by observing Source Addresses.
  - Shouldn’t learn SA from erroneous frame, since it may lead to later mis-direction of frames.
  ⇒ **Cut-through switch:**
    - needs to verify CRC
    - should defer learning until after verifying CRC.
- **Hybrids**: Many switches operate in a hybrid mode:
  1. Measure frame error rates.
  2. If error rate is low
     - use cut-through: low delay, few erroneous frames propagating
  else
     - use store-and-forward: increases delay but confines erroneous frames
Variations of cut-through

Figure from CCNA course material.
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).
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
Example of a VRAM switch: IDT 77V400 chip
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) queueing:
When a FCFS queue is served by a single server & all jobs have the same service time,
the result will be FIFO
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**

  With unicast traffic distributed uniformly amongst output ports, maximum output utilisation = \(2 - \sqrt{2} = 59\%\) [Karol87].

- **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**

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)
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\textsuperscript{†} 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:
  • 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

\textsuperscript{†} aka per-connection queueing
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 deterioriates 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.

Overload isn’t detected immediately because of router buffers, round trip times, and delayed timeout.
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
- Explicit Congestion Notification – an alternative to discard

try to manage congestion
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)
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.