Lecture #11: In-network resource management
continuing to share a network, this time with help from switches

CAIDA's IPv4 AS Core, February 2017 (https://www.caida.org/research/topology/as_core_network/2017/)

application the things that actually generate traffic
transport sharing the network, reliability (or not)
examples: TCP, UDP
network naming, addressing, routing
examples: IP
link communication between two directly-connected nodes
examples: ethernet, bluetooth, 802.11 (wifi)

question: TCP congestion control doesn't react to congestion until after it's a problem; could we get senders to react before queues are full?
CAIDA's IPv4 AS Core,
February 2017
(https://www.caida.org/research/topology/as_core_network/2017/)

**question:** TCP congestion control doesn't react to congestion until after it's a problem; could we get senders to react before queues are full?
you can fill out this chart as you watch the lecture. use it in whatever way is helpful, and I’ll show you my answers at the end

<table>
<thead>
<tr>
<th>type of management</th>
<th>what does this type of management allow a switch to do</th>
<th>example protocols</th>
<th>how the protocol works</th>
<th>pros/cons?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Management</td>
<td></td>
<td>DropTail</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RED, ECN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delay-based Scheduling</td>
<td></td>
<td>Priority Queueing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth-based Scheduling</td>
<td></td>
<td>Round-robin</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Weighted Round-robin</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deficit Round-robin</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
queue management: given a queue, when should it drop (or mark) packets?

**droptail:** drop packets only when the queue is full. simple, but leads to high delays and synchronizes flows.

**RED (drops) / ECN (marks):** drop (or mark) packets before the queue is full: with increasing probability as the queue grows. prevents queue lengths from oscillating, decreases delay, flows don’t synchronize. but complex and hard to pick parameters

as long as our switches are taking a more active role, let's see what else they can do (we'll return to queue management later in the lecture)
**delay-based scheduling:** can we give latency guarantees for some types of traffic?

**priority queueing:** put latency-sensitive traffic in its own queue and serve that queue first. does not prevent the latency-sensitive traffic from “starving out” the other traffic (in other queues).

as long as our switches are taking a more active role, let’s see what else they can do

(we’ll return to priority queueing later in the lecture)
**bandwidth-based scheduling:** can we allocate specific amounts of bandwidth to some traffic?

**round robin:** can’t handle variable packet sizes (and in its most basic form doesn’t allow us to weight traffic differently)

**weighted round robin:** can set weights and deal with variable packet sizes, but needs to know mean packet size

In each round:

For each queue q:

\[ q.n\text{\_norm} = \frac{q.weight}{q.mean\_packet\_size} \]

\[ \text{min} = \min \text{ of } q.n\text{\_norm’s over all flows} \]

For each queue q:

\[ q.n\_packets = \frac{q.n\text{\_norm}}{\text{min}} \]

Send \( q.n\_packets \) from queue q
bandwidth-based scheduling: can we allocate specific amounts of bandwidth to some traffic?

round robin: can’t handle variable packet sizes (and in its most basic form doesn’t allow us to weight traffic differently)

weighted round robin: can set weights and deal with variable packet sizes, but needs to know mean packet size

deficit round robin: doesn’t need mean packet sizes, near-perfect fairness and low packet processing overhead

in each round:

for each queue q:

q.credit += q.quantum

while q.credit >= size of next packet p:
    q.credit -= size of p
    send p

now let’s start revisiting some of our previous strategies
**delay-based scheduling:** can we give latency guarantees for some types of traffic?

**priority queueing:** put latency-sensitive traffic in its own queue and serve that queue first. does *not* prevent the latency-sensitive traffic from “starving out” the other traffic (in other queues).

can solve this problem by doing something similar to bandwidth-based scheduling across the two queues.
## In-network Resource Management

<table>
<thead>
<tr>
<th>Type of Management</th>
<th>What Does This Type of Management Allow a Switch to Do</th>
<th>Example Protocols</th>
<th>How the Protocol Works</th>
<th>Pros/Cons?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Queue Management</strong></td>
<td>Signal congestion, potentially before queues are full</td>
<td>DropTail, RED, ECN</td>
<td>Drop packets when the queue is full</td>
<td>Simple, but queues get full (among other problems)</td>
</tr>
<tr>
<td><strong>Delay-based Scheduling</strong></td>
<td>Prioritize latency-sensitive traffic</td>
<td>Priority Queueing</td>
<td>Serve some queues before others</td>
<td>Prioritized queues can starve out the others</td>
</tr>
<tr>
<td><strong>Bandwidth-based Scheduling</strong></td>
<td>Enforce (weighted) fairness among different types of traffic</td>
<td>Round-robin, Weighted Round-robin, Deficit Round-robin</td>
<td>Try to give each type of traffic an equal share of bandwidth, round robin, but incorporate average packet size, round robin, but do a better job with packet sizes</td>
<td>Can't handle variable packet sizes, average packet size hard to get, honestly pretty good</td>
</tr>
</tbody>
</table>

### DropTail

- **How the Protocol Works**: Drop packets when the queue is full.
- **Pros/Cons**: Simple, but queues get full (among other problems).

### RED, ECN

- **How the Protocol Works**: Drop or mark packets before the queue is full.
- **Pros/Cons**: Can keep queues from filling up, but complicated.

### Priority Queueing

- **How the Protocol Works**: Serve some queues before others.
- **Pros/Cons**: Prioritized queues can starve out the others.

### Round-robin

- **How the Protocol Works**: Try to give each type of traffic an equal share of bandwidth.
- **Pros/Cons**: Can't handle variable packet sizes.

### Weighted Round-robin

- **How the Protocol Works**: Round robin, but incorporate average packet size.
- **Pros/Cons**: Average packet size hard to get.

### Deficit Round-robin

- **How the Protocol Works**: Round robin, but do a better job with packet sizes.
- **Pros/Cons**: Honestly pretty good.

---

**Is in-network resource management a good idea on the Internet?**

If you wrote something different, that's okay! There are other pros/cons for these protocols, and other ways to describe how they work.
1970s: ARPAnet  
1978: flexibility and layering  
early 80s: growth → change  
late 80s: growth → problems  
1993: commercialization

<table>
<thead>
<tr>
<th>hosts.txt</th>
<th>distance-vector routing</th>
<th>TCP, UDP</th>
<th>OSPF, EGP, DNS</th>
<th>congestion collapse</th>
<th>policy routing</th>
<th>CIDR</th>
</tr>
</thead>
</table>

1978: flexibility and layering: the things that actually generate traffic sharing the network, reliability (or not)

Examples: TCP, UDP

network: naming, addressing, routing

Examples: IP

link: communication between two directly-connected nodes

Examples: ethernet, bluetooth, 802.11 (wifi)

question: TCP congestion control doesn’t react to congestion until after it’s a problem; could we get senders to react before queues are full? yes, if switches take a more active role

CAIDA’s IPv4 AS Core, February 2017
(https://www.caida.org/research/topology/as_core_network/2017/)

Katrina LaCurts | lacurts@mit.edu | 6.033 2021