David Boggs, Co-Inventor of Ethernet, Dies at 71

Thanks to the invention he helped create in the 1970s, people can send email over an office network or visit a website through a coffee shop hot spot.

Before becoming the dominant networking protocol, Ethernet was challenged by several other technologies. In the early 1980s, Mr. Metcalfe said, when Mr. Boggs took the stage at a California computing conference, at the San Jose Convention Center, to discuss the future of networking, a rival technologist questioned the mathematical theory behind Ethernet, telling Mr. Boggs that it would never work with large numbers of machines.

His response was unequivocal. “Seems Ethernet does not work in theory,” he said, “only in practice.”

Lecture #12: In-network resource management
continuing to share a network, this time with help from switches
1970s: ARPAnet  
1978: flexibility and layering  
early 80s: growth → change  
late 80s: growth → problems  
1993: commercialization

hosts.txt  distance-vector routina  TCP, UDP  OSPF, EGP, DNS  congestion collapse  policy routing  CIDR

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, January 2020
(https://www.caida.org/projects/cartography/as-core/2020/)
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?
**queue management**: given a queue, when should it drop packets?

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

- **RED**: drop packets before the queue is full, with increasing probability as the queue grows. Prevents queue lengths from oscillating, decreases delay, flows don’t synchronize.
queue management: given a queue, when should it drop packets?

**droptail**: drop packets only when the queue is full.
queue management: given a queue, when should it drop packets?

**droptail**: drop packets only when the queue is full. simple, but leads to high delays and synchronizes flows.
queue management: given a queue, when should it drop packets?

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

**RED:** drop packets before the queue is full, with increasing probability as the queue grows.
queue management: given a queue, when should it drop packets?

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

**RED**: drop packets before the queue is full, with increasing probability as the queue grows.
**queue management**: given a queue, when should it drop packets?

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

**RED**: drop packets before the queue is full, with increasing probability as the queue grows.
**queue management**: given a queue, when should it drop packets?

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

**RED**: drop packets before the queue is full, with increasing probability as the queue grows.
queue management: given a queue, when should it drop packets?

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

**RED:** drop packets before the queue is full, with increasing probability as the queue grows.
queue management: given a queue, when should it drop packets?

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

**RED**: drop packets before the queue is full, with increasing probability as the queue grows.
queue management: given a queue, when should it drop packets?

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

**RED**: drop packets before the queue is full, with 
increasing probability as the queue grows.
queue management: given a queue, when should it drop packets?

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

**RED:** drop packets before the queue is full, with increasing probability as the queue grows.
**queue management**: given a queue, when should it drop packets?

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

**RED**: drop packets before the queue is full, with increasing probability as the queue grows. Prevents queue lengths from oscillating, decreases delay, flows don't synchronize.
queue management: given a queue, when should it drop packets?

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

**RED:** drop packets before the queue is full, with increasing probability as the queue grows. prevents queue lengths from oscillating, decreases delay, flows don't synchronize.
**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.

![Graph showing the relationship between average queue size and drop probability](image)
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.

![Graph showing the behavior of drop probability as a function of average queue size](image-url)
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?
**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 (can extend this idea to multiple queues/types of traffic)
**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).
**delay-based scheduling**: can we give latency guarantees for some types of traffic?

**priority queuing**: 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 queuing later in the lecture)
**bandwidth-based scheduling:** can we allocate specific amounts of bandwidth to some traffic?

- weighted round robin can set weights and deal with variable packet sizes, but needs to know mean packet size.
**bandwidth-based scheduling:** can we allocate specific amounts of bandwidth to some traffic?

round robin
**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)
**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
**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

---

**in each round**:

- for each queue \( q \):
  - \( q_n = q_{\text{mean packet size}} / q_{\text{weight}} \)
  - \( q_n \) packets from queue \( q \)

- min = min of \( q_n \)’s over all flows
**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

---

**In each round:**

for each queue q:

\[
q.\text{norm} = \frac{q.\text{weight}}{q.\text{mean\_packet\_size}}
\]
**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

**in each round**:

- for each queue q:
  - \( q.\text{norm} = q.\text{weight} / q.\text{mean\_packet\_size} \)
- \( \text{min} = \text{min of } q.\text{norm}'s \text{ over all flows} \)
**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

---

**in each round:**

for each queue $q$:

- $q.norm = q.weight / q.mean\_packet\_size$

min = min of $q.norm$’s over all flows

for each queue $q$:

- $q.n\_packets = q.norm / min$

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

---

**in each round:**

```plaintext
for each queue q:
    q.norm = q.weight / q.mean_packet_size

min = min of q.norm’s over all flows

for each queue q:
    q.n_packets = q.norm / 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**
**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**

---

**in each round:**

**for each queue q:**

- for each 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**

---

**in each round**:  

for each queue q:  

q.credit += q.quantum
**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**

---

**in each round:**

**for each queue q:**

\[
q.\text{credit} \; += \; q.\text{quantum}
\]

while \(q.\text{credit} \geq \text{size of next packet } p\):

\[
q.\text{credit} \; -= \; \text{size of } p
\]

send \(p\)
**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:**

\[
q.\text{credit} \, += \, q.\text{quantum}
\]

while \( q.\text{credit} \, >= \, \text{size of next packet } p \):

\[
q.\text{credit} \, -= \, \text{size of } p
\]

send \( p \)
**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:

- \( q.\text{credit} += q.\text{quantum} \)

while \( q.\text{credit} \geq \text{size of next packet } p \):

- \( q.\text{credit} -= \text{size of } p \)
- 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).
**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.
<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>signal congestion, potentially before queues are full</td>
<td>DropTail</td>
<td>drop packets when the queue is full</td>
<td>simple, but queues get full (among other problems)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RED, ECN</td>
<td>drop or mark packets before the queue is full</td>
<td>can keep queues from filling up, but complicated</td>
</tr>
<tr>
<td>Delay-based Scheduling</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>Bandwidth-based Scheduling</td>
<td>enforce (weighted) fairness among different types of traffic</td>
<td>Round-robin</td>
<td>try to give each type of traffic an equal share of bandwidth</td>
<td>can't handle variable packet sizes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Weighted Round-robin</td>
<td>round robin, but incorporate average packet size</td>
<td>average packet size hard to get</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deficit Round-robin</td>
<td>round robin, but do a better job with packet sizes</td>
<td>honestly pretty good</td>
</tr>
</tbody>
</table>
# 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</td>
<td>drop packets when the queue is full</td>
<td>simple, but queues get full (among other problems)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RED, ECN</td>
<td>drop or mark packets before the queue is full</td>
<td>can keep queues from filling up, but complicated</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</td>
<td>try to give each type of traffic an equal share of bandwidth</td>
<td>can't handle variable packet sizes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Weighted Round-robin</td>
<td>round robin, but incorporate average packet size</td>
<td>average packet size hard to get</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deficit Round-robin</td>
<td>round robin, but do a better job with packet sizes</td>
<td>honestly pretty good</td>
</tr>
</tbody>
</table>

Is in-network resource management a good idea on the Internet?
<table>
<thead>
<tr>
<th>1970s: ARPAnet</th>
<th>1978: flexibility and layering</th>
<th>early 80s: growth (\rightarrow) change</th>
<th>late 80s: growth (\rightarrow) problems</th>
<th>1993: commercialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts.txt</td>
<td>distance-vector routina</td>
<td>TCP, UDP</td>
<td>OSPF, EGP, DNS</td>
<td>congestion collapse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(which led to congestion control)</td>
<td>policy routing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CIDR</td>
</tr>
</tbody>
</table>

**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**