Decongestion
Control

Barath Raghavan and Alex C. Snoeren
University of California, San Diego
Network resource sharing

1. Statistical multiplexing + **buffers**
2. Admission control + **reservations**
3. Decongestion control + **coding**
### Sharing resources

<table>
<thead>
<tr>
<th></th>
<th>Implicit</th>
<th>Explicit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>End-point</strong></td>
<td>HighSpeed Vegas</td>
<td>TFRC PCP</td>
</tr>
<tr>
<td></td>
<td>FAST TCP BIC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Westwood Scalable</td>
<td></td>
</tr>
<tr>
<td><strong>In-network</strong></td>
<td>RED ECN AQM</td>
<td>WFQ RCP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Admission Control XCP</td>
</tr>
</tbody>
</table>
## Sharing resources

<table>
<thead>
<tr>
<th></th>
<th>Implicit</th>
<th>Explicit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>End-point</strong></td>
<td>HighSpeed Vegas</td>
<td>Request and set</td>
</tr>
<tr>
<td></td>
<td>FAST, TCP, BIC</td>
<td>PCP</td>
</tr>
<tr>
<td></td>
<td>Backoff</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Westwood</td>
<td>Scalable</td>
</tr>
<tr>
<td><strong>In-network</strong></td>
<td>RED</td>
<td>WFQ</td>
</tr>
<tr>
<td></td>
<td>Network hints</td>
<td>RCP</td>
</tr>
<tr>
<td></td>
<td>AQM</td>
<td>XCP</td>
</tr>
</tbody>
</table>
Ignoring packet loss

1. A simple thought experiment
2. Decongestion control
3. Design considerations
Simple greedy transport

\[ \text{S0} \] 10Mbps \[ 10Mbps \] \[ 10Mbps \] \[ \text{D0} \]

\[ \text{S0} \] \[ \text{S1} \] \[ \ldots \] \[ \text{S9} \] \[ 10Mbps \] \[ 10Mbps \] \[ 10Mbps \] \[ 10Mbps \] \[ 10Mbps \] \[ 10Mbps \] \[ 10Mbps \] \[ 10Mbps \] \[ \text{D9} \] \[ \text{D1} \] \[ \ldots \] \[ \text{D0} \]

- 90% packet loss!
- 100% utilization
Decongestion control

Persistent network congestion is ok, if:

- End-to-end goodput is high
- End-to-end delay is low
- We maintain inter-user fairness
Decongestion control

S0 → 10Mbps → D0

S0 → S1 → ... → S9

10Mbps

D0 → D1 → ... → D9
Decongestion control

• Send packets as fast as possible
  – Aligned with end-host incentives
• Erasure code the data
  – Most/all packets received will be useful
  – Dynamically change coding rate
• Drop packets fairly at routers
Challenge: transmission

Set flow rates

Wasted transmissions

Lost packets

Set coding parameters

10Mbps

2Mbps

D0

D1
Design challenges

1. Setting transmission parameters
Sending data

1a. Caravan size, $c$

1b. Coding rate

2. Flow allocation, $f$

Socket $\xrightarrow{\text{send()}}$ Buffer $\xrightarrow{\text{Data ready}}$ Caravan $\xrightarrow{\text{Erasure coding}}$ Coded Caravan $\xrightarrow{\text{Transmit}}$
Setting caravan size, $c$

- Tradeoff: coding overhead vs. latency
  - Bulk vs. interactive flows

![Flowchart showing the process of adjusting caravan size](chart.png)
Setting coding rate

• Estimate goodput with receiver feedback
  – Set coding rate accordingly

• Tradeoff: type of erasure code
  – Standard: Reed-Solomon
  – Rateless: LT codes, online codes
  – Simple: Redundancy

• Coding rate doesn’t impact other flows
  – Provides stability of traffic demands
Setting flow allocation, $f$

- End host has limited bandwidth
  - Must apportion bandwidth to its flows
  - Still wants to be greedy

```
Caravan ACK’d

Decase $f$ No $g' < g$ Yes Increase $f$
```

Old goodput, $g$
New goodput, $g'$
Challenge: long vs. short paths

S0 → 10Mbps → D0
S0 → 10Mbps → S2
S1 → 10Mbps → D1
S2 → 10Mbps → D2
Design challenges

1. Setting transmission parameters
2. Enforcing fairness at bottlenecks
Flow fairness

Idea:
- Throttle flows to their fair-share at routers

Implementation:
- Use fair *dropping* rather than fair *queueing*
  - For example, Approximate Fair Dropping (AFD)
Long paths with fair dropping
Challenge: link wastage

10Mbps

S0

S1

Dead packets

2Mbps

S2

D0

D1

D2
Design challenges

1. Setting transmission parameters
2. Enforcing fairness at bottlenecks
3. Avoiding “dead packets”
Conjecture: few dead packets
Design challenges

1. Setting transmission parameters
2. Enforcing fairness at bottlenecks
3. Avoiding “dead packets”
Potential benefits

• High total end-to-end goodput
  – Network is always delivering coded data

• Small router buffer requirements
  – Traffic is not bursty or sensitive to loss

• Incentive compatibility
  – Aligned with greedy sender behavior

• Traffic stability
  – Only flow arrival/departure affects path demand
Questions!