Review

• A fundamental question to networking:
  – Multiple users share the same network. Who can send at what speed?

• Approach 1: End-to-end congestion control
  – TCP uses AIMD to probe for available bandwidth, and exponential backoff to avoid congestion
  – XCP: routers explicitly stamp feedback (increase or decrease)
    • Nice control algorithms
Why does it work? [Chiu-Jain]

- A feedback control system
- The network uses feedback $y$ to adjust users’ load $\sum x_i$
Goals of Congestion Avoidance

- Efficiency: the closeness of the total load on the resource to its knee

- Fairness:
  \[ F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)} \]
  - When all \( x_i \)'s are equal, \( F(x) = 1 \)
  - When all \( x_i \)'s are zero but \( x_j = 1 \), \( F(x) = 1/n \)

- Distributedness
  - A centralized scheme requires complete knowledge of the state of the system

- Convergence
  - The system approaches the goal state from any starting state
Model the system as a linear control system

\[ x_i(t+1) = \begin{cases} 
  a_i + b_i x_i(t) & \text{if } y(t) = 0 \Rightarrow \text{Increase,} \\
  a_D + b_D x_i(t) & \text{if } y(t) = 1 \Rightarrow \text{Decrease.} 
\end{cases} \]

- Four sample types of controls
- AIAD, AIMD, MIAD, MIMD
Phase plane

https://en.wikipedia.org/wiki/Phase_plane
Macroscopic behavior of TCP

Figure 1: TCP window evolution under periodic loss
Each cycle delivers \((\frac{W}{2})^2 + \frac{1}{2}(\frac{W}{2})^2 = \frac{1}{p}\) packets and takes \(W/2\) round trip times.
• The total data delivered for each cycle is the area under the sawtooth

\[ \left( \frac{W}{2} \right)^2 + \frac{1}{2} \left( \frac{W}{2} \right)^2 = \frac{3}{8} W^2 \]

• Each cycle also delivers $1/p$ packets
• Solve for $W$, we have

\[ W = \sqrt{\frac{8}{3p}} \]

• Substituting $W$ into the bandwidth equation:

\[ BW = \frac{\text{data per cycle}}{\text{time per cycle}} = \frac{MSS \times \frac{3}{8} W^2}{RTT \times \frac{W}{2}} = \frac{MSS/p}{RTT \sqrt{\frac{2}{3p}}} \]

• Collect the constant at one term $C$, we have

\[ BW = \frac{MSS}{RTT} \frac{C}{\sqrt{p}} \]
XCP key ideas

• Separate efficiency control from fairness control

• Using explicit router feedback to help users adjust sending rates
How does XCP Work?

- Routers send explicit bandwidth adjustment information to end users.
How does XCP Work?

- Routers send explicit bandwidth adjustment information to end users
- A downstream router overwrites the upstream feedback only if its feedback is smaller
How does XCP Work?

- Routers send explicit bandwidth adjustment information to end users

Congestion Window = Congestion Window + Feedback
XCP: how a router computes feedback

• Efficient control
  – First figure out the aggregate bandwidth to reallocate

• Fairness control
  – Then figure out how to distribute spare bandwidth or reclaim over-allocated bandwidth
  – Bandwidth shuffle to prevent unfairness when \( C = \sum_i x_i \)

• Separate for analytic tractability
How to compute the aggregate bandwidth

- \( \phi = \alpha \times d \times S - \beta \times Q \)
- \( S = C - \sum_i x_i \) (\( C \) is the output link capacity)
- Why “\(-\beta \times Q\)”
Bandwidth shuffling

• To prevent no program in terms of fairness when link is fully utilized
• Simultaneously allocate and deallocate bandwidth to achieve fairness
• $h = \max (0, r \cdot y - |\phi|)$
• If $|\phi| > r \cdot y$, then no need to shuffle. AIMD will take care of fairness
Per packet feedback

- Feedback\(_i\) = positive\(_i\) – negative\(_i\)
- If \(\phi > 0\), allocate it so that the increase in throughput of all flows is the same
  - Aggregate bandwidth to increase is
    - \((\phi + h)/d\)
  - Aggregate bandwidth to decrease is \(h\)
- If \(\phi < 0\), allocate it so that the decrement is proportional to a flow’s current throughput
  - Aggregate bandwidth to reduce is
    - \((|\phi| + h)/d\)
  - Aggregate bandwidth to increase is \(h\)
How to compute per-packet positive feedback

• Each flow’s throughput increases by a constant $a$: $x_i(t+1) = x_i(t) + a$

• The change in a flow’s $cwnd_i$ is proportional to its $rtt_i$ to keep throughput increase constant $\Delta cwnd_i \propto rtt_i$

• Next step is to translate the change into a per-packet feedback:
  – total change of $cwnd_i$ divided by $N_i$, the number of packets from flow $i$ seen in interval $d$
• Ni is proportional to cwndi divided by packet size si, inversely proportional to rtti (cwndi/si/rtti*d)

\[ p_i \propto \frac{rtt_i^2}{cwnd_i/s_i} \]

• So we have

\[ p_i = \xi_p \frac{rtt_i^2 \cdot s_i}{cwnd_i}, \] where \( \xi_p \) is a constant

• The sum of all flow’s throughput increase is

\[ \frac{h + \max(\phi, 0)}{d} = \sum_{i=1}^{L} \frac{p_i}{rtt_i}, \]
Per-packet positive feedback

• Thus

$$\xi_p = \frac{h + \max(\phi, 0)}{d \cdot \sum \frac{rtt_i \cdot s_i}{cwnd_i}}.$$ 

• Routers compute $\xi_p$ per control interval

• And compute per packet feedback on per packet arrival/departure

$$p_i = \xi_p \frac{rtt_i^2 \cdot s_i}{cwnd_i}.$$
How to compute per packet negative feedback

• The aggregate negative feedback is proportional to a flow’s sending rate (MD)
• The aggregate change of $cwnd_i$ should be proportional to the current $cwnd_i$

$$\Delta cwnd_i \propto cwnd_i$$
• Again, a router needs to divide the total change in congestion window by the number of packets received in a control interval

• Recall $N_i$ is proportional to $\frac{cwnd_i}{s_i}$ divided by packet size $s_i$, inversely proportional to $\text{rtt}_i$ ($\frac{cwnd_i}{s_i/\text{rtt}_i*d}$)

• So we have

$$ n_i = \xi_n \cdot \text{rtt}_i \cdot s_i $$
• The aggregate of all flows’ rate decrease is the sum of all per packet rate decrease:

\[
\frac{h + \max(-\phi, 0)}{d} = \sum_{i=1}^{L} \frac{n_i}{rtt_i}.
\]

• Therefore, we can compute \( \xi_n \) as

\[
\xi_n = \frac{h + \max(-\phi, 0)}{d \cdot \sum s_i},
\]
Comments

• It turned out that routers need not keep per flow state to compute exact AIMD parameters
• Per-packet computation
• A small number of state variables
• Any simplification to approximate XCP?
• Programmable router implementation?
• Incremental deployment?
Discussion

• Explicit congestion signaling + dynamic packet state
  – With congestion header, a router does not have to keep per-flow state!
  – Clever math

• Limitations of XCP
  – Security
  – Rounding errors
  – Cannot deal with link layer congestion
  – Complexity
  – Non convergence with multiple bottlenecks

• How can we robustly control resource allocation?
One more bit is enough

• Variable Structure Congestion Control Protocol

• Key idea
  – Four bits to signal regions of action
  – 01: low load MI
  – 10: high load AI
  – 11: overload MD
TCP uses **binary** congestion signals, such as loss or one-bit Explicit Congestion Notification (ECN)

- **Additive Increase (AI)** with a fixed step-size can be very **slow** for large bandwidth

![Graph showing Multiplicative Decrease (MD) and Additive Increase (AI)]
Key observation

Fairness is not critical in low-utilization region

- Use Multiplicative Increase (MI) for fast convergence onto efficiency in this region
- Handle fairness in high-utilization region
Variable structure control protocol

- Routers signal the level of congestion
- End-hosts adapt the control algorithm accordingly
VCP Properties

- Use network link load factor as the congestion signal
- Decouple efficiency and fairness controls in different load regions
- Achieve high efficiency, low loss, and small queue
- Fairness model is similar to TCP:
  - Long flows get lower bandwidth than in XCP (proportional vs. max-min fairness)
  - Fairness convergence much slower than XCP (solvable with even more, e.g., 8 bits)
Conclusion

• Review of TCP AIMD congestion control
• XCP
  – Key ideas
  – How to compute positive and negative feedback