What we know so far

- TCP Congestion control based on AIMD window adjustment [Jac88]
  - Saved Internet from congestion collapse
- All Internet apps need to use E2E congestion control [FF99]
  - Avoid future congestion collapse
- Today
  - Congestion Control for multimedia apps
  - Congestion Control for LFNs

Requirements of multimedia apps

- Don’t require (full) reliability
- Require stable sending rate
- TCP is not appropriate protocol
  - Full reliability through ACKs + timeouts
  - Data delivered in order
  - Sending rate is halved in reaction to congestion
- We need a TCP alternative
  - TCP-Compatible: Does not use more bw than TCP in steady state

TCP Friendly

- Any other scheme needs to co-exist with TCP in FIFO queues
  - Scheme must have same long-term throughput as a function of packet loss and delay
  - Scheme must not affect TCPs negatively in short time scale
- TCP Reno Model
  \[ T = \frac{RTT}{3} + 2 * T_{slo} * (3 \frac{RTT}{3} + p * (1 + 32p^2) \]
General Idea

- Use TCP model to calculate TCP throughput as a function of loss rate and RTT
  - If sending rate is higher than model, reduce the sending rate to the model's rate
  - Otherwise increase the sending rate
- We need to measure the parameters
  - How to measure loss rate?
  - How to measure RTT?

Loss Rate Measurement

- Average loss interval (ALI) is the weighted average of last N intervals between losses
- Loss event probability = 1/ALI

Example of Loss Measurement

- How many intervals to keep?
  - More intervals → smooth
  - Less intervals → higher responsiveness
  - If loss process is known then # intervals can be computed
  - In practice use 8 intervals
- Results
  - Increase 0.14 packets/RTT
  - When losses occur, it takes 4RTTs to reduce sending rate to half
Improving Stability

- If EWMA is small, TFRC does not react strongly to congestion.
- If EWMA is high, then TFRC reduces rate strongly.
- Solution: Space out consecutive packets.

Slowstart

- How do you start sending when no RTT and loss measurements?
- Receiver reports receiving rate.
- If no loss,$\ T_{actual,i+1} = \min(2T_{actual,i}, 2T_{received,i})$.
- Otherwise, use TFRC.

Experimental Evaluation

- Simulate how TFRC performs against TCP.
  - Different link speeds.
  - Different number of flows.
- Mean sending rate:
  - Different from short and medium-term rate.

Smoothness
High Delay High Bandwidth Challenges: Slow Start
- TCP throughput controlled by congestion window (cwnd) size
- In slow start, window increases exponentially, but may not be enough
- Example: 10Gb/s, 200ms RTT, 1460B payload, assume no loss
  - Time to fill pipe: 18 round trips = 3.6 seconds
  - Data transferred until then: 382MB
  - Throughput at that time: 382MB / 3.6s = 105Mb/s
  - 8.5% utilization → not very good
- Loose only one packet → drop out of slow start into AIMD (even worse)

High Delay High Bandwidth Challenges: AIMD
- In AIMD, cwnd increases by 1 packet/RTT
- Available bandwidth could be large
  - E.g., 2 flows share a 10Gb/s link, one flow finishes → available bandwidth is 5Gb/s
  - E.g., suffer loss during slow start → drop into AIMD at probably much less than 10Gb/s
- Time to reach 100% utilization is proportional to available bandwidth
  - E.g., 5Gb/s available, 200ms RTT, 1460B payload → 17,000s

AQM doesn’t help
- Shown analytically in [Low01] and via simulations

Proposed Solution
- Example: In TCP, Additive-Increase Multiplicative-Decrease (AIMD) controls both
- How does decoupling solve the problem?
  - To control congestion: use MIMD which shows fast response
  - To control fairness: use AIMD which converges to fairness
Characteristics of Solution

- Improved Congestion Control (in high bandwidth-delay & conventional environments):
  - Small queues
  - Almost no drops
- Improved Fairness
- Scalable (no per-flow state)
- Flexible bandwidth allocation: min-max fairness, proportional fairness, differential bandwidth allocation, ...

XCP: An eXplicit Control Protocol

1. Congestion Controller
2. Fairness Controller

How does XCP Work?

Round Trip Time
Congestion Window
Feedback = + 0.1 packet
Congestion Header

Round Trip Time
Congestion Window
Feedback = - 0.3 packet
### How does XCP Work?

Routers compute feedback without any per-flow state.

Congestion Window = Congestion Window + Feedback

### How Does an XCP Router Compute the Feedback?

#### Congestion Controller
- **Goal:** Matches input traffic to link capacity & drains the queue
- **Looks at aggregate traffic & queue**
- **Algorithm:**
  - Aggregate traffic changes by \( \Delta \)
  - \( \Delta \sim \) Spare Bandwidth
  - \( \Delta \sim \) Queue Size
  - So, \( \Delta = \alpha \frac{Cwnd_{avg}}{Spare} - \beta \text{Queue} \)

#### Fairness Controller
- **Goal:** Divides \( \Delta \) between flows to converge to fairness
- **Looks at a flow’s state in Congestion**
- **Algorithm:**
  - If \( \Delta > 0 \) ⇒ Divide \( \Delta \) equally between flows
  - If \( \Delta < 0 \) ⇒ Divide \( \Delta \) between flows proportionally to their current rates

### Details

**Congestion Controller**
- \( \Delta = \alpha \frac{Cwnd_{avg}}{Spare} - \beta \text{Queue} \)
- System converges to optimal utilization (i.e., stable) for any link bandwidth, delay, number of sources if:
  - \( 0 < \alpha < \frac{\pi}{4\sqrt{2}} \) and \( \beta = \alpha \sqrt{2} \)
- No Parameter Tuning

**Fairness Controller**
- **Algorithm:**
  - If \( \Delta > 0 \) ⇒ Divide \( \Delta \) equally between flows
  - If \( \Delta < 0 \) ⇒ Divide \( \Delta \) between flows proportionally to their current rates
- Need to estimate number of flows \( N \)
  - \[ N = \sum_{p \in \mathcal{P}} \frac{1}{RTT_{pkts}} \times \frac{Cwnd_{avg}}{RTT_{pkts}} \]
- No Per-Flow State

### Subset of Results

Similar behavior over:
XCP Remains Efficient as Bandwidth or Delay Increases

Utilization as a function of Bandwidth

![Graph showing utilization vs. bandwidth](image)

Utilization as a function of Delay

![Graph showing utilization vs. delay](image)

XCP increases proportionally to spare bandwidth

\[ \alpha \text{ and } \beta \text{ chosen to make XCP robust to delay} \]

Bottleneck Bandwidth (Mb/s)

Round Trip Delay (sec)

XCP Shows Faster Response than TCP

![Graph comparing XCP and TCP](image)

XCP shows fast response

XCP Summary

- XCP
  - Outperforms TCP
  - Efficient for any bandwidth
  - Efficient for any delay
  - Scalable (no per flow states)

- Benefits of Decoupling
  - Use MIMD for congestion control which can grab/release large bandwidth quickly
  - Use AIMD for fairness which converges to fair bandwidth allocation

XCP is Fairer than TCP

![Graph comparing XCP and TCP](image)

Same RTT

Different RTT

Avg. Throughput

Flow ID

(RTT is 40 ms → 330 ms)