Congestion Control for Streaming Media

October 15, 1999
EE290T, UC Berkeley
Sally Floyd

Outline of talk:

- Why do we need end-to-end congestion control?
- Characterizing TCP congestion control
- Equation-based congestion control for unicast traffic
- Equation-based congestion control for multicast traffic
- Related issues: RED, ECN, FEC, Diff-Serv, CM (Congestion Manager), and others.
The Internet is a work in progress, with no central control or authority, many players independently making changes, and many forces of change (e.g., new technologies, new applications, new commercial forces, etc.).

The Internet is like the elephant, and each of us is like the blind man who knows only the part closest to us.

So far, the success of the Internet has rested on the IP architecture, rather decentralized and fast-changing evolution of the Internet architecture, robustness, flexibility, and ability to scale, and not on its efficiency, optimality, or fine-grained control.

There is no guarantee that this architecture has worked reasonably well to date. The Internet is like the elephant, and each of us is like the blind man who knows only the part closest to us.

The Internet is a work in progress, with no central control or authority.
Why do we need end-to-end congestion control?

E.g., minimizing loss and delay, maximizing throughput.

• As a tool for the application to better achieve its own goals:
  • To avoid congestion collapse.
  • Fairness.
What is the fairness goal? (the pragmatic answer)

- No connection/session/end-node hog should hog the network resources.

- New forms of traffic that compete with TCP as best-effort traffic in FIFO queues should not be significantly more aggressive (or significantly less aggressive).

- The current Internet is dominated by best-effort traffic and FIFO scheduling.

- TCP is the dominant transport in the Internet (90-95% of the bytes/packets).

- No connection/session/end-node hog should hog the network resources.

TCP is the dominant transport in the Internet (90-95% of the bytes/packets).
Why is fairness a concern?

Simulations showing three TCP flows and one UDP flow (without end-to-end congestion control), with a congested link using FIFO scheduling.

Simulation showing three TCP flows and one UDP flow (without end-to-end congestion control), with a congested link using FIFO scheduling.

- Solid line: TCP Goodput
- Bold line: Aggregate Goodput
- Dashed line: UDP Arrivals
- Dotted line: UDP Goodput
What is the fairness goal? (other possible answers)

• Fairness goals not dependent upon pricing:
  – Min-max fairness: On each link of the network, each entity has an equal claim to the bandwidth on each link.
  – Global fairness: Each entity has an equal claim to the scarce resources (where an entity traversing N congested links is using N times more scarce resources than an entity traversing 1 congested link).
  – Global fairness: Each entity has an equal claim to the scarce resources (where an entity traversing N congested links is using N times more scarce resources than an entity traversing 1 congested link).
  – Global fairness: Each entity has an equal claim to the scarce resources (where an entity traversing N congested links is using N times more scarce resources than an entity traversing 1 congested link).
  – Global fairness: Each entity has an equal claim to the scarce resources (where an entity traversing N congested links is using N times more scarce resources than an entity traversing 1 congested link).
  – Global fairness: Each entity has an equal claim to the scarce resources (where an entity traversing N congested links is using N times more scarce resources than an entity traversing 1 congested link).
  – Min-max fairness: On each link of the network, each entity has an equal claim to the bandwidth of that link. (e.g., Fair Queuing.)

• Fairness goals related to pricing:
  – Pricing: For some services, bandwidth is allocated to those willing to pay for it. (e.g., Internet, DiffServ.)
  – Congestion-based pricing: The "cost" of the bandwidth on each link varies as a function of the level of congestion (e.g., the packet drop rate).
Why is congestion collapse a concern?

Congestion collapse occurs when the network is increasingly busy, but little useful work is getting done.

Problem: Classical congestion collapse: Paths clogged with unnecessarily-retransmitted packets [Nagle 84].

Fix: Modern TCP retransmit timer and congestion control algorithms [Jacobson 88].
Problem:
Paths clogged with fragments of packets invalidated because another fragment (or cell) has been discarded along the path. [Kent and Mogul, 1987]

Fix: MTU discovery [Kent et al., 1988], Early Packet Discard in ATM networks [Romanow and Floyd, 1995].
Congestion collapse from undelivered packets:

**Problem:** Paths clogged with packets that are discarded before they reach the receiver [Floyd and Fall, 1999].

**Fix:** Either end-to-end congestion control or a “virtual-circuit” style of guaranteed delivery of packets that enter the network will be delivered to the receiver.

---

TCP

UDP

10 Mbps

10 Mbps

10 Mbps

1.5 Mbps

1.28 Mbps

128 Kbps

10

S1

S2

R1

R2

S3

S4

1.5 Mbps

10 Mbps

10 Mbps

10 Mbps

TCP

UDP

10 Mbps

10 Mbps

128 Kbps

1.28 Mbps
How can end-to-end congestion control be useful to an application for its own reasons?

In an environment of either per-flow scheduling or small-scale statistical multiplexing:

- The use of end-to-end congestion control can reduce unnecessary loss and delay for that flow.
- The loss and delay experienced by a flow is affected by its own sending rate.

For its own reasons:
How can end-to-end congestion control be useful to an application for its own reasons? Part 2:

Strategies and "gaming" against other users. Not individual users determining their own end-to-end congestion control are largely independent of their own sending rate (holding the congestion control behavior of all other flows fixed).

- End-to-end congestion control can be useful to a flow to avoid mechanisms that could be deployed by the network to penalize best-effort traffic (for instance, the "tragedy of the commons" is avoided in part because the "players" are not individual users determining their own end-to-end congestion control).

- The loss rate and delay experienced by a flow is largely independent of its own sending rate (holding the congestion control behavior of all other flows fixed).

- The loss rate and delay experienced by a flow is largely independent of its own sending rate (holding the congestion control behavior of all other flows fixed).

- The loss rate and delay experienced by a flow is largely independent of its own sending rate (holding the congestion control behavior of all other flows fixed).

- In an environment of FIFO scheduling and large-scale statistical multiplexing at all congestion points, all flows are large-scale statistical multiplexing at all congestion points.
Characterizing TCP congestion control

TCP uses Additive Increase Multiplicative Decrease (AIMD).

- Decrease congestion window by 1/2 after loss event.
- Increase congestion window by one packet per RTT.
- In heavy congestion, when a retransmitted packet is itself dropped, use exponential backoff of the retransmit timer.
- slow-start: start by doubling the congestion window every roundtrip time.

TCP uses Additive Increase Multiplicative Decrease (AIMD).
Why not use TCP for unicast streaming media?

Reliable delivery is not needed.

Acknowledgements are not returned for every packet, and the application would prefer a rate-based approach to a window-based approach anyway.

Cutting the sending rate in half in response to a single packet drop is undesirable.

•
Other possibilities for end-to-end congestion control?

- Equation-based congestion control: adjust the sending rate as a function of the longer-term packet drop rate.

- Use a rate-based version of TCP’s congestion control mechanisms, with:
  - AIMD with different increase/decrease constants.
  - E.g., decrease multiplicatively by 3/4, increase additively by 3/7 packets.

- Rate Adaptation Protocol (RAP) [RH99] or TCP’s ACK-clocking.

- The Rate Adaptation Protocol (RAP) [RH99].
The "steady-state model" of TCP:

- **The model:**
  - Fixed packets size in bytes.
  - Fixed round trip time in seconds, no queue.
  - A packet is dropped each time the window reaches \( W \) packets.
  - TCP's congestion window: \( W \) packets.

- **TCP's fixed round trip time:** \( R \) in seconds.

- **The maximum sending rate in packets per round trip time:** \( W \)

- **The maximum sending rate in bytes per second:** \( R/B \cdot W \)

- **The average sending rate:** \( \frac{W}{2(R/B)} \)

- **The packet drop rate:**

\[
\frac{d}{2R/B^2} = \frac{2R/B}{6B} = T
\]

The result:

\[
W \left( \frac{R}{2B} \right) = d
\]

- **The model:** Fixed packet size \( B \) in bytes.

- **The steady-state model of TCP:**
Verifying the "steady-state model" of TCP:

Drop Rate (Percent of Arriving Packets Dropped)

TCP-Friendly Arrival Rate (KBps)

Numbered lines: simulation results

Solid line: the simple equation characterizing TCP

(1.460-byte packets, 0.06 second roundtrip time)
The "steady-state model" of TCP: an improved version.

\[
(1) \quad \frac{d^2Z + 1}{dZ^2} \left( \frac{d}{dZ} \sqrt{\frac{Z}{RTT}} \right) = \frac{Z}{B} \frac{d}{dZ} \sqrt{\frac{Z}{RTT}} \cdot \frac{d}{dZ} \frac{Z}{RTT}
\]

\[d^p\] packet drop rate
\[B\] packet size in bytes
\[T_r\] sending rate in bytes/sec
Equation-based congestion control:

- Use the TCP equation characterizing TCP's steady-state sending rate as a function of the RTT and the packet drop rate.
- Use the TCP equation characterizing TCP's steady-state sending rate as a function of the measured roundtrip time and packet loss rate.
- Over longer time periods, maintain a sending rate that is a function of the measured roundtrip time and packet loss rate.

- One or more packet drops/marks in a roundtrip time.
- The justification: It is acceptable not to reduce the sending rate in half in response to a single packet drop.

The cost:

- Limited ability to make use of a sudden increase in the available bandwidth.

The benefits:

- Limited ability to make use of a sudden increase in the available bandwidth.
Given equation-based congestion control, why use the "TCP-friendly" equation?

Because best effort traffic in the current Internet is likely to compete in FIFO queues with TCP traffic.

Criteria for evaluating an equation for equation-based congestion control:

- Stability, potential for oscillations.
- Adaptive range: Range in packet drop rate needed for desired range in sending rate.
- Sending rate as a function of the roundtrip time?
  (How does this generalize to multicast?)
- Sending rate as a function of the packet size?

\[
\frac{d^{H/p}}{\beta^{3/2}} = \beta
\]
Further evaluation of equation-based congestion control:

- Transient performance.
- Long-term fairness with respect to TCP.
- Synchronization among multiple flows.
- Stability, oscillations.
Equation-based congestion control in an environment with FIFO scheduling and large-scale statistical multiplexing:

- Packet drop rate is largely independent of individual flow's sending rate.
- The flow monitors the long-term packet drop rate, and the roundtrip time.
- Benefit over TCP: Smoother changes in the sending rate in response to changes in congestion levels.
- Packet drop rate is largely independent of individual flow's sending rate.
- Benefit over TCP: Smoother changes in the sending rate in response to changes in congestion levels.
rate (if queuing delay dominates propagation delay).

Packet drop rate is in part a function of individual flow's sending rate.

Concern: The steady-state "model" assumes a fixed roundtrip time. The actual roundtrip time can vary significantly as a function of the sending rate. The roundtrip time can vary significantly as a function of the sending rate.

- Increase in sending rate → increase in packet drop rate.
- There is an upper bound on the allowed increase in the sending rate.

Equation-based congestion control in an environment with either per-flow scheduling, or small-scale statistical multiplexing:
The receiver averages the packet loss rate over the most recent several RTTs.

- The receiver reports the loss average to the sender once per RTT.
- The sender averages the roundtrip over the most recent several measurements.

- The sender averages the roundtrip over the most recent several measurements.
- The receiver averages the packet loss rate over the most recent several RTTs.
- The receiver also takes into account the K+1, K+2, and K+3 rd loss intervals, with reduced weights.
- A loss interval is a sending period ending in a loss event (e.g., one or more packet drops in a window of data).

- Joint work with Mark Handley, Jitendra Padhye, and Joerg Widmer.
Using the equation, the sender calculates its allowed sending rate.

- If the sending rate is less than one packet/RTT, decrease the sending rate by at most one packet/RTT.
- If the allowed sending rate is less than current sending rate, decrease sending rate.
- If the allowed sending rate is greater than current sending rate, increase sending rate more slowly.

Using the equation, the sender calculates its allowed sending rate.
Slow-start:
– Increase the sending rate by a factor (e.g., 2) each RTT.
– Rate increases are "smoothed out" over a RTT.
– Twice the receiver's reported receive rate is an upper bound on the sending rate.

• If two report intervals pass without receiving the expected report from the receiver, cut the sending rate in half.
Simulations with TFRC: TCP-friendly Rate Control
Simulations of TCP and TFRC flows.
Equation-based congestion control for single-sender multicast traffic: Advantages of equation-based congestion control for multicast:

- The sender responds over slightly slower time scales than does TCP.
- The sender does not have to hear about every packet drop from every receiver.

Equation-based congestion control for single-sender multicast traffic:
Single-sender multicast: simple congestion control.

If receivers did not have to measure their RTT to the sender:

- Each receiver could simply measure its packet drop rate.
- Some mechanism could be used (probabilistic feedback, tree-structured feedback).
- The sending rate would be limited by the combination of the worst-case RTT and the worst-case packet drop rate, even if these two worst-cases were not experienced by the same receiver [Whetten 98].

Drawback:

- The sending rate would be limited by the combination of the worst-case RTT and the worst-case packet drop rate, even if these two worst-cases were not experienced by the same receiver [Whetten 98].
Assume that all members of the multicast group have synchronized clocks (e.g., GPS).

The sender:

• Single-sender multicast: simple congestion control, attempt #2.

 Each receiver can determine the one-way time from the sender to each receiver.

Receiver:

• Each receiver would cause the sender to slow down.

The sender reports its current sending rate.

• Receivers know from their combined packet drop rate and RTT whether their feedback would cause the sender to slow down.

• Probabilistic or tree-structured mechanisms are used for feedback to the sender.
Single-sender multicast: more complicated congestion control:

- No assumption of synchronized clocks.

Receiver's know from their combined packet drop rate and RTT whether the sender using some mechanism:

Their feedback would cause the sender to slow down.

- Receivers with high packet drop rates have to "measure" their RTT to

Receivers with high packet drop rates have to "measure" their RTT to

their RTT whether the sender using some mechanism.

No assumption of synchronized clocks.
Other complications introduced by multicast:

- How aggressively can the sender slow-start?
- Does the sender need positive feedback to keep on sending, or do re-
  ceivers have the responsibility to unsubscribe if their congestion control
  feedback is not reaching the sender?
- What are the transient traffic dynamics when round trip times are chang-
  ing due to increased queuing delay, for example?
Other approaches to congestion control for multicast traffic:

- Layered multicast groups.
- Layered multicast with receivers subscribing and unsubscribing from layered multicast groups, with receivers subscribing and unsubscribing from.
- Intserv (Integrated Services) and some forms of differentiating services (differentiated services) to eliminate the need for end-to-end congestion control.
- Intserv (Integrated Services) and some forms of differentiated services (differentiated services) to eliminate the need for end-to-end congestion control.

Other approaches to congestion control for multicast traffic:
ECN indications:
- For TCP, TCP-level feedback from TCP receiver to TCP sender about
  - Congestion Experienced (CE) indication from router to receiver.
  - ECN-Capable Transport (ECT) indication from sender to router.
  - ECN is an experimental addition to the IP architecture [RFC 2481].

ECN is an experimental addition to the IP architecture [RFC 2481].

- Given that routers are not necessarily waiting until buffer overflow to
dropping the packet, to inform end-nodes of congestion.
- Drop a packet, routers can set an ECN bit in the packet header instead of
  drop a packet, routers can set an ECN bit in the packet header instead of
  given that routers are not necessarily waiting until buffer overflow to
  packets before buffer overflow, as an indication of congestion to end nodes.
- Routers measure the average queue size, and probabilistically drop
  - Active queue management (e.g., RED) is being incorporated into routers.

Related issues: Explicit Congestion Notification (ECN)
related issues: the congestion manager

the congestion manager: a proposal for a congestion control mechanism that would reside below the transport layer (e.g., below TCP and below UDP traffic)

A longer-term research question: congestion control provided by a collaboration between the sending and receiving node, including detection and feedback about packet drops/marks.

- The first step: congestion control provided by the sender, for flows that have end-to-end feedback about packet drops/marks (e.g., TCP, or at the application layer for UDP traffic).

- A longer-term research question: congestion control provided by a collaboration between the sending and receiving node, including detection and feedback about packet drops/marks.

36
References:


RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet, R. Rejaie, M. Handley, D. Estrin.


