Rethinking Startup in Congestion Control

Dan Liu (Case), Mark Allman (ICSI), Shudong Jin (Case), Limin Wang (Bell Labs)

NEONet, March 2006

"I’ll be on the hill ’cause I can’t stop,
I’ll be on that hill with everything I got"
Fundamental Question

• At what rate should transmission begin when sending across an unknown, best-effort, packet-switched internetwork?

• We have a *workshop answer*

• OK, a *small workshop handwave*
• Slow Start

![Graph showing the relationship between Round-Trip Time and Congestion Window (segments)]
• However, slow start can be really slow
  ▶ e.g., in large bandwidth-delay product networks

• Slow start can underutilize a bunch of available capacity
• Example #1

![Graph showing congestion window over time for different RTT values.]

- RTT = 0.05 sec
- RTT = 0.1 sec
- RTT = 0.5 sec
• Example #2
• Example #2

The graph illustrates the relationship between time (in seconds) and congestion window size (in segments). The x-axis represents time, ranging from 0 to 0.7 seconds, while the y-axis represents congestion window size, ranging from 0 to 120 segments. The graph shows two lines:

- **Avail. Capacity** (red line) indicates the maximum capacity available for use, with a label **RTT=0.1 sec** indicating the round-trip time.

- **Un-used Capacity** (green dotted line) shows the unused capacity over time, which increases as time progresses.

**Used Capacity** (green line) starts from zero at time 0 and increases over time, approaching the **Avail. Capacity** line as time increases.

This visual representation helps in understanding how the congestion window adapts to network conditions, especially when RTT is considered a key factor in adjusting the window size to avoid congestion.
Proposed Solutions

• Three general classes
  1. Information sharing
     ▪ use information from previous network usage
  2. Capacity estimation
     ▪ use the first few (low-rate) packets to estimate the capacity
  3. Ask the network
     ▪ send an explicit request that each node in the path must deal with
Proposed Solutions (cont.)

- Proposed solutions trade complexity for possible performance improvement.
Proposed Solutions (cont.)

- What if we didn’t want to pay for complexity?
- Then we’d add our own proposal to the list:
  4. Blind stupidity
Jump Start

• General idea:
  ▶ send as fast as desirable when starting up
    ▪ i.e., no startup phase at all
  ▶ but, retain the remainder of congestion control (AIMD)

  ▶ (There are some mechanistic details that I am glossing over here.)
• Do we *need* a conservative startup phase?

• What does this end of the spectrum look like?

• We’re building a simulation framework
  ▶ While this seems easy enough to investigate, there are details
• We assume the worst case for Jump Start is pretty bad
  ▶ E.g., Jump Start will cause congestion
  ▶ E.g., Jump Start will decrease performance
  ▶ E.g., Jump Start will impact competing traffic

• On the other hand, with end-to-end reservations, per-flow queueing, etc. Jump Start seems like no big deal

• So, how complex would we have to make the network to support a very simple scheme like Jump Start?
• No more complexly

• We leverage the heavy-tailed nature of traffic
  ▶ Most connections cannot put much load on a network simply because they have so little to send
  ▶ The big connections that can place a burden on the network are rare
    ▼ Can we weather a rare transient?

  ▶ (Add in that end hosts place anemic advertised window limits on transmission.)
• No more complexity

• User networks are thin and they can’t create much congestion

• Server networks are fat, but policies will dictate that no one connection will be able to use massive bandwidth

• The core weathered Slammer (1 packet UDP fire and forget worm); the ends couldn’t overwhelm it even without any congestion control
• A little help

• It might be nice if the end hosts flagged segments that were part of the Jump Start phase of a connection

• Routers could then preferentially drop these segments to try to limit the impact on competing traffic
• A little help

• Use an AQM strategy that preferentially drops based on usage
• It can’t be done without lots of per-flow state and mechanism
Summary

• Explore:
  ▶ an extremely simple scheme for startup
  ▶ whether some form of startup is required for Internet congestion control
  ▶ whether some form of startup is required for reasonable performance

• Well.... what do you think? How crazy are we?