Thoughts on the Evolution of TCP in the Internet (version 2)

Sally Floyd

ICIR Wednesday Lunch

March 17, 2004

www.icir.org/floyd/talks.html

Themes:

- Proposing a new mechanism is easy.
- It is harder, and more interesting, to consider:
  - the exact problem being solved;
  - the range of possible solutions;
  - the architectural tradeoffs involved in picking this particular solution.

Example: convergence times.

- How to improve convergence times with HighSpeed TCP?
- Architectural questions raised:
  - Explicit feedback from routers?
  - Flow-specific state in routers (or in packet headers)?
  - QoS?
  - What are the limitations of TCP, or of no per-flow state in routers, or of best-effort service, for high-speed flows (e.g., 10 Gbps)?

Example: DCCP's congestion control

- Question: How far can DCCP go in providing faster startup, faster recovery after idle periods, etc.?
- Architectural questions raised:
  - Limitations of window-based congestion control?
  - Of no per-flow state in routers/packet headers?
  - Of best-effort traffic?

Past history of TCP

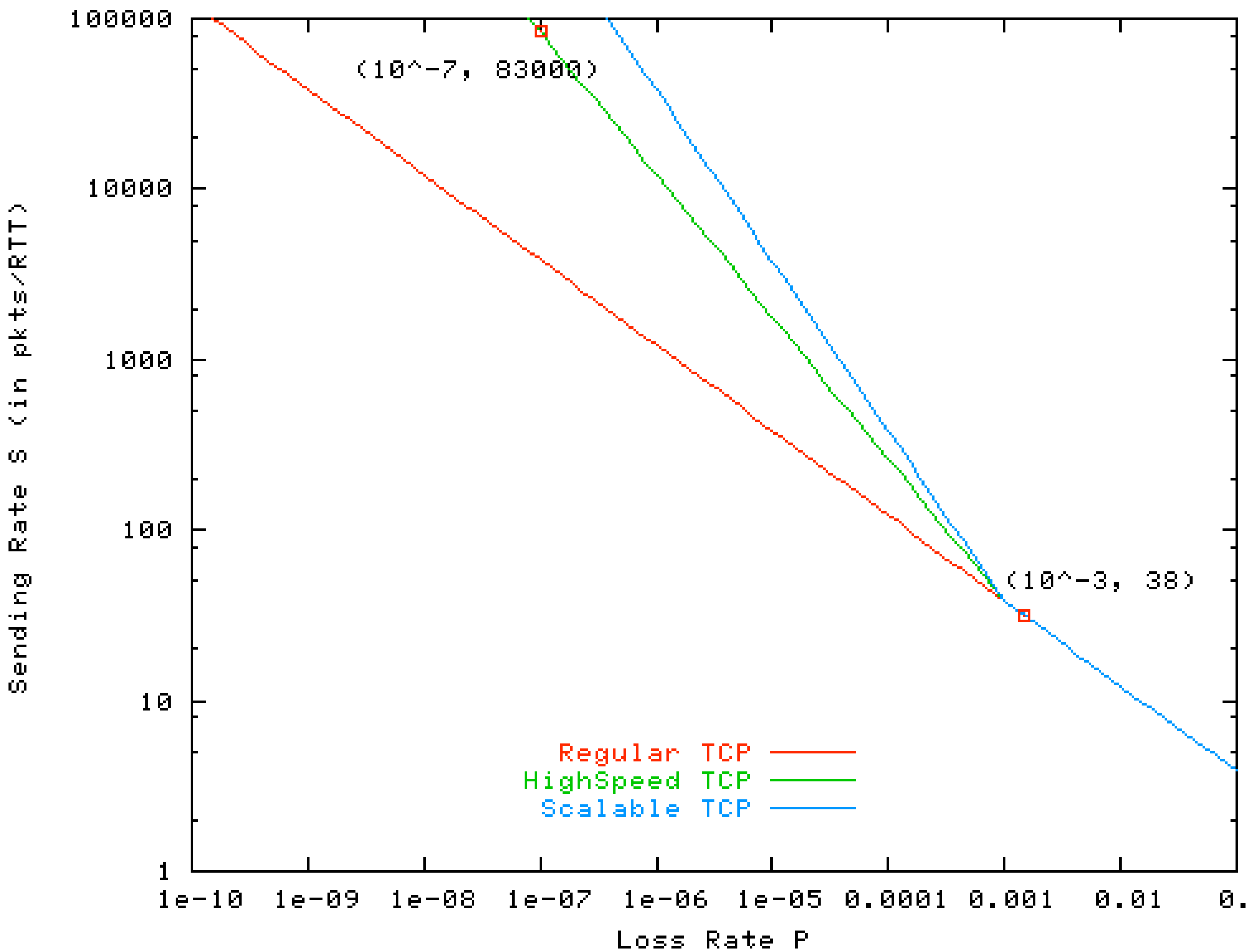
- Reno/NewReno/SACK:
  - Half of servers use SACK, many others use NewReno.
  - Almost all browsers use SACK.
  - DON'T use Reno in simulations or experiments!!!
- Delay-based congestion control:
  - Vegas, FAST
  - TCP-Nice and TCP-LowPriority use delay-based congestion control for low priority TCP .
- ECN:
  - Explicit instead of implicit notification.
  - Standardized but not deployed.

Past history of TCP

- Quality of Service:
  - Intserv, diffserv, etc.
  - Limited deployment.

- New transport protocols:
  - SCTP: multi-streaming, multihomed transport.
  - DCCP: for unreliable, congestion-controlled transport.

TCP's response function



Convergence Times for HighSpeed TCP et al:

- Different models give different results!
  - Model #1: DropTail queues with global synchronization for loss events.
  - Model #2: Drop Tail queues, some synchronization, depending on traffic mix.
  - Model #3: RED queues, some synchronization.
  - Model #4: RED queues, no synchronization

- Which model is the best fit for the current or future Internet?

Convergence times for HighSpeed TCP et al:

- What would improve convergence times?
  - TCP changes:

    Less aggressive increases after loss events?
  - Explicit feedback from routers to transport:

    Finer-grained congestion feedback?
  - Router changes:

    Flow-specific state in routers (or in packet headers)?
  - QoS:

    Something other than best-effort service.

Additional Feedback from Routers?

Examples: XCP, QuickStart.

- Explicit feedback from routers would be useful (and necessary) for faster startups.
  - Also for faster recovery after idle periods.

- Per-packet feedback (as in XCP) would give greater power, at greater cost.

Evaluating additional feedback from routers:

- Possible kinds:
  - Needed from all routers along the path (e.g., QuickStart, XCP);
  - Needed only from one router (e.g., ECN).
- Possible semantics:
  - Faster startup (e.g., QuickStart, XCP);
  - Advice/instruction to slow down, or to increase less aggressively (e.g., ECN, XCP).
  - Info from link layer (e.g., corruption, link-up).
  - …

Flow-specific state in routers?

- What are the cost/benefit tradeoffs for maintaining state in routers/packet headers for very large flows?
  - E.g., for a 10Gbps TCP flow?
- Flow-specific marking or dropping, for faster convergence?
- Flow-specific state to help use the bandwidth promptly when a short fat flow ends?
  - (short in time, fat in bytes)

How far can DCCP go in providing faster startup, fast recovery after idle periods, etc.

- Many of DCCP's target applications want:
  - Faster startup;
  - Abrupt changes in the sending rate;
  - To start up fast after idle periods;
  - To minimize changes in sending rates;
- How much can this be done with:
  - Window-based congestion control?
  - Best-effort traffic but no per-flow state in routers or in packet headers?
  - Best-effort traffic?

Fundamental limitations of window-based congestion control?

- The jostling of ACKs can lead to unnecessary burstiness?
  - Rate-based pacing could help.
  - Equation-based congestion control (e.g., TFRC) is another alternative.
- Slow start-up?
  - Explicit feedback from routers could help.
- Decreasing the window after a loss event.
  - "Decreasing" does not necessarily require "halving".
  - TFRC is another alternative.

Fundamental limitations of no per-flow state in routers/packet headers?

- For environments with high link utilization, there are limits to faster start-up, and to faster convergence.
  - E.g., a new flow starting up in a high-bandwidth environment with a small number of competing flows.
- For environments with short fat flows, there are limits to link utilization.
  - E.g., many flows wanting to use high bandwidth, each for a fraction of an RTT.

Fundamental limitations of best-effort service?

- Best-effort flows need to avoid persistent, high drop rates.
- In environments with FIFO scheduling at congested links, best-effort flows need to pay attention to per-flow fairness.
- There are no common assumptions about average or worst-case queueing delay.
- Some flows prefer better-than-best-effort delay, throughput, or loss rates.
- A best-effort flow can’t assume that bandwidth is available when the flow is ready to use it.

Extra viewgraphs:

Interactions between transport and link layers:

- Wireless links with variable delay, throughput, corruption, etc?
- Hints from transport to link layers, e.g., about robustness to reordering or to delay?
- New issues raised by optical networks, e.g., by optical burst switching?