End-to-End Transmission Control through Inference about the Network

Keith Winstein and Hari Balakrishnan

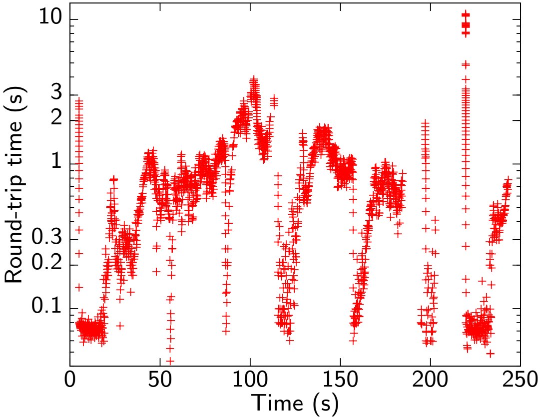
keithw@mit.edu

March 27, 2012

The problem

- Historically, network was dumb.
- Since 1980s, endpoints “share” network with TCP.
- Now network is:
  - designed for TCP
  - very smart.

The smart network is a messy one



Round-trip time on Verizon “4G” LTE in Cambridge, Mass.

Evolvability

- Network is now much smarter than endpoints.
  - Link-layer retransmit.
  - Rate adaptation.
  - Reordering.
  - Huge buffers.
- Need kludges to accommodate TCP congestion control.

TCP Congestion Control (Jacobson 88)

- Maintains and updates three variables:
  - `cwnd` = congestion window
  - `SRTT` = smoothed round-trip time
  - `RTTVAR` = round-trip time variation
- Feedback from network: delivery (with delay) or loss.

The "teleology" of TCP

- TCP achieves "minimum potential delay" throughput fairness (Kunniyur '03) if:
  - in steady state / for long flows
  - all losses due to buffer overflow
  - all RTTs equal
  - Otherwise, TCP achieves. . . ?

The "teleology" of TCP

- TCP achieves "minimum potential delay" throughput fairness (Kunniyur '03) if:
  - in steady state / for long flows
  - all losses due to buffer overflow
  - all RTTs equal
  - Otherwise, TCP achieves. . . ? **unknown / not easily stated!**

Where most congestion control methods have problems

- Data centers (DCTCP)
- Bufferbloat
  - vs. Skype
  - vs. new TCP (e.g. Web browser)
- Wireless link
  - Stochastic loss (not due to buffer overflow)
  - Rate adaptation
  - Link reorders packets to hide loss — uses RTO only
  - 10 second delays!
- Intermittent or roaming link
- WAN
  - Amazon EC2 Singapore to Virginia has fat pipe, 1% loss!
- Many short connections (e.g. Web browsers)

Point solutions are inadequate

- TCP congestion control algorithms for different situations.
  - NewReno, CUBIC good with multiplexing, low BDP.
  - Vegas / Compound good with high BDP, low multiplexing.
  - Data Center TCP for tiny RTT.
- **Mobility** makes a host's (or flow's!) regime change over time.
- Congestion control is performed by the **sender**.
- "Why is TCP $x >$ TCP y?"
  - Hard to answer, because...

What traditional TCP conflates

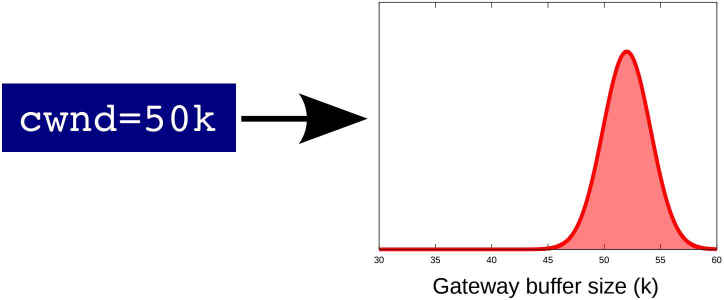
Most congestion control algorithms smush together conversation about:

1. What are the assumptions and model of the network?
2. What is the goal?
   - Per-flow throughput "fairness" for long-running flows?
   - Throughput fairness without undue delay to real-time flows (e.g. Skype)?
   - Balance between long-running flows and possible new flows?
3. Given assumptions and goal, what to do *now*?

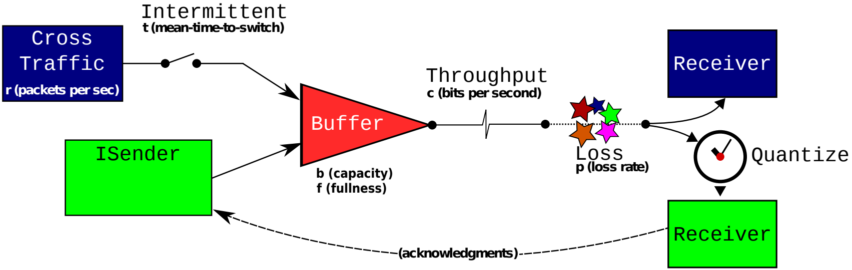
Our proposal: **unsmush**.

Proposal

- Be **smarter** and **evolvable** at the endpoints.
- Preserve **uncertainty** and optimize **utility**.



Example network model




```
typedef Series< Series< Series< CrossTraffic, Intermittent >,
                        ISender >,
                Series< Buffer,
                        Series< Series< Throughput, StochasticLoss >,
                                Diverter< Series< TimeQuantize, ReceiverObject >,
                                          Collector > > > > Channel;
```

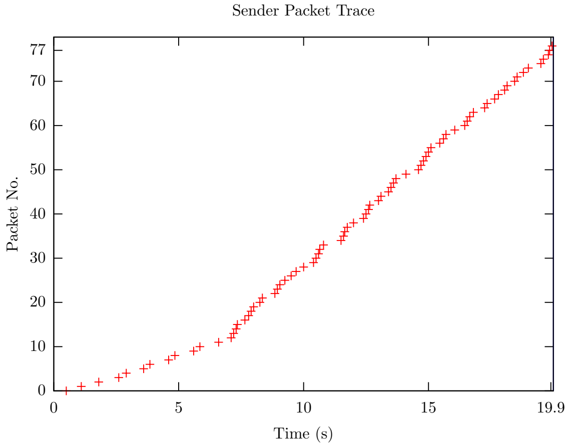
How it works

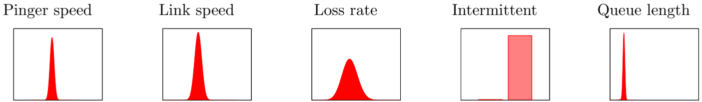
At each time step:

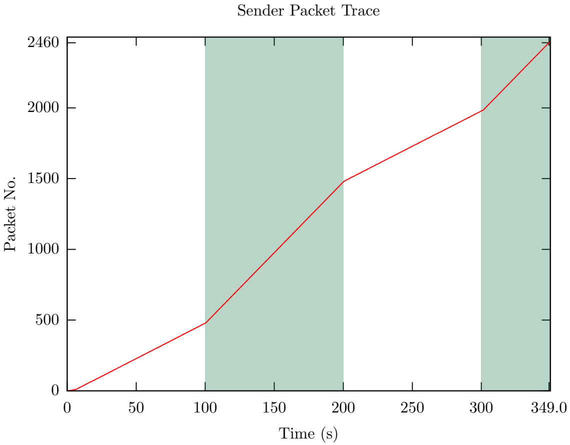
1. Update probability distribution over possible network states.
2. Take single action to maximize utility.

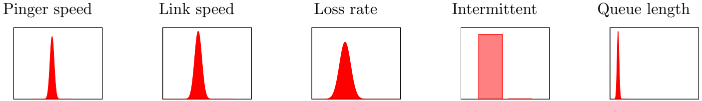
- Network is modeled as nondeterministic automaton
- Represent uncertainty as weighted set of possible states
- **Update rule:** simulate possible states, discard contradictions
  - Can be precomputed.

- Best "action" ⇒ delay for next packet that maximizes *expected* utility


Sender Packet Trace
Packet No.
Time (s)
0
10
20
30
40
50
60
70
77
0
5
10
15
19.9


Pinger speed
Link speed
Loss rate
Intermittent
Queue length


Sender Packet Trace
Packet No.
2460
2000
1500
1000
500
0
0
50
100
150
200
250
300
349.0
Time (s)


Pinger speed
Link speed
Loss rate
Intermittent
Queue length

What's next?

- Realistic return path
- Uncertain topology
- Continuous parameters
- Compare vs. TCP and router-assisted congestion control
- Stability of multiple senders

Summary

- We factored out the **utility** and the network **model assumptions** from the congestion control algorithm.
- Let's move from classical estimation to machine inference.
- TCP has assumptions too — being explicit about them will help the network evolve.

Questions: `keithw@mit.edu`