FAST TCP:
design, implementation, experiments

Cheng Jin

http://netlab.caltech.edu

Brief History of FAST TCP

- Congestion control as an optimization problem
- Primal-dual framework to study TCP congestion control
- Modeling existing TCP implementations
- Theoretical analysis on FAST TCP
- FAST TCP Implememtation

Optimization Model

- Network bandwidth allocation as utility maximization
- Optimization problem

$$\max_{x_s \geq 0} \quad \sum_s U_s(x_s)$$
$$\text{subject to} \quad y_l \leq c_l, \quad \forall l \in L$$

- Primal-dual components

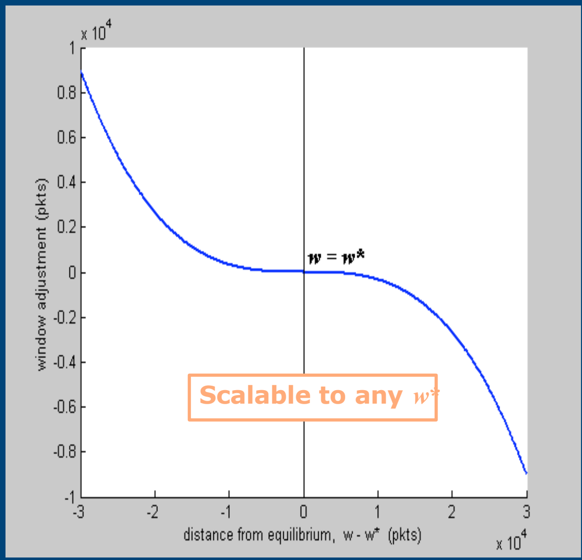
x(t+1) = F(q(t), x(t)) Source

p(t+1) = G(y(t), p(t)) Link

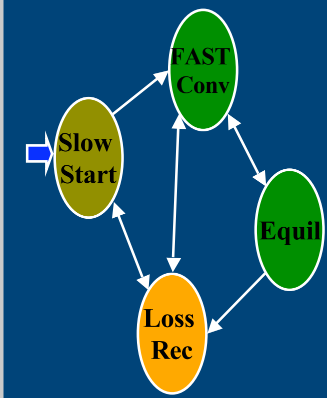
Use of Queueing Delay in FAST

- Each FAST TCP flow has a target number of packets to maintain in network buffers in equilibrium
- Queueing delay allows FAST to estimate the number of packets currently buffered and estimate its distance from the target

Solution: estimate target



- FAST



FAST and Other DCAs

- FAST is an implementation within the primal-dual framework
- Queueing delay is one example of the price from the network
- FAST does not use queueing delay to predict or avoid packet losses
- FAST may use other forms of price in the future when they become available

Packet Level

- **Reno**
  AIMD(1, 0.5)

  ACK: $W \leftarrow W + 1/W$
  Loss: $W \leftarrow W - 0.5\,W$

- **HSTCP**
  AIMD(a(w), b(w))

  ACK: $W \leftarrow W + a(w)/W$
  Loss: $W \leftarrow W - b(w)\,W$

- **STCP**
  MIMD(a, b)

  ACK: $W \leftarrow W + 0.01$
  Loss: $W \leftarrow W - 0.125\,W$

- **FAST**

  $$\text{RTT}: W \leftarrow W \cdot \frac{\text{baseRTT}}{\text{RTT}} + \alpha$$

Architecture

Each component
- designed independently
- upgraded asynchronously

Data Control | **Window Control** | **Burstiness Control**

Estimation

TCP Protocol Processing

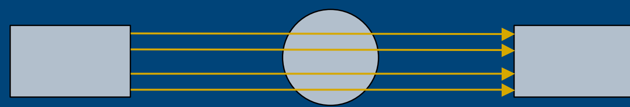
Known Issues

- Network latency estimation
  - route changes, dynamic sharing
  - does not upset stability
- Small network buffer
  - at least like TCP Reno
  - adapt α on slow timescale, but how?
- TCP-friendliness
  - friendly at least at small window
  - how to dynamically tune friendliness?
- Reverse path congestion

Experiments

- In-house dummynet testbed
- PlanetLab Internet experiments
- Internet2 backbone experiments
- ns-2 simulations

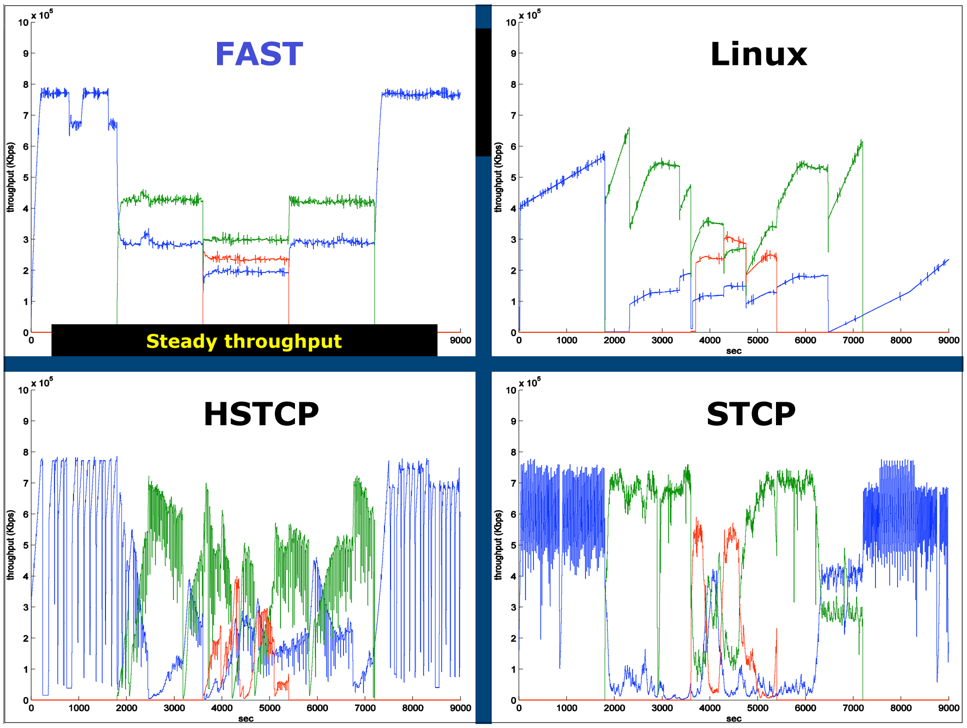
Dummynet Setup



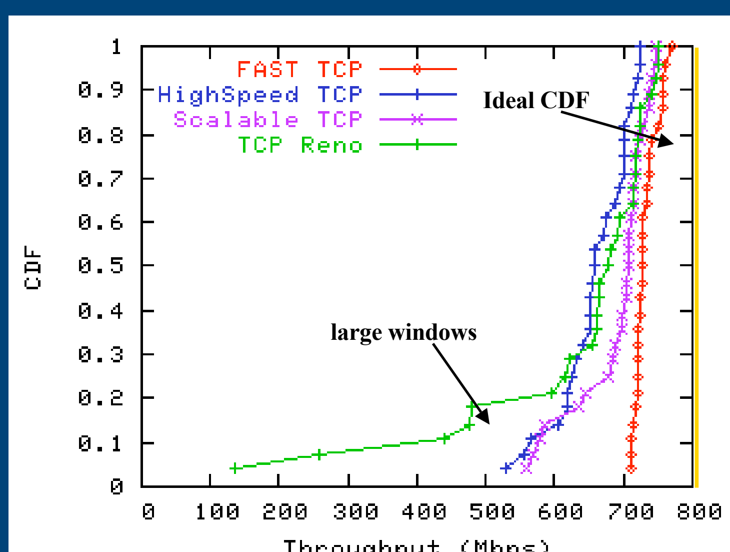
- Single bottleneck link, multiple path latencies
- Iperf for memory-to-memory transfers
- Intra-protocol testings
- Dynamic network scenarios
- Instrumentation on the sender and the router

What Have We Learnt?

- FAST is reasonable under normal network conditions
- Well-known scenarios where FAST doesn’t perform well
- Network behavior is important
- Dynamic scenarios are important
- Host implementation (Linux) also important

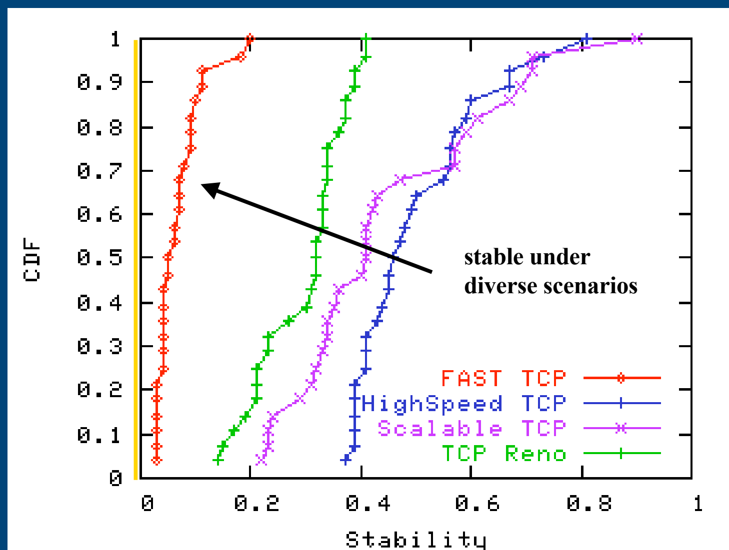


Aggregate Throughput



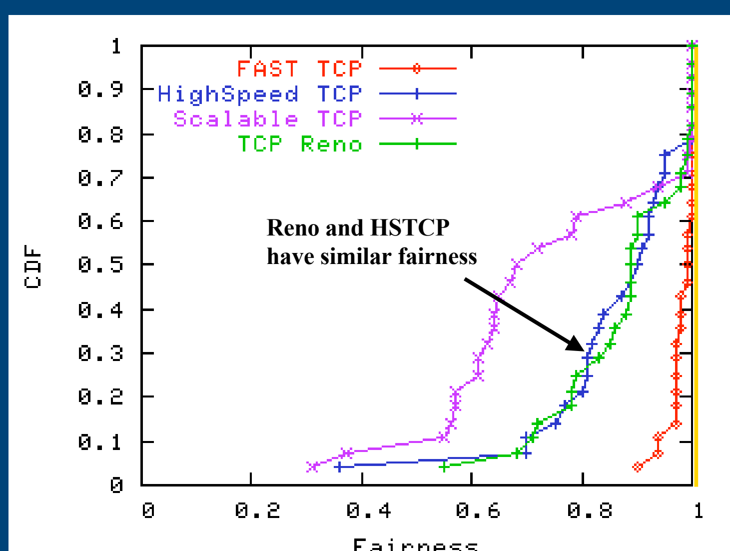
Dummynet: cap = 800Mbps; delay = 50-200ms; #flows = 1-14; 29 expts

Stability

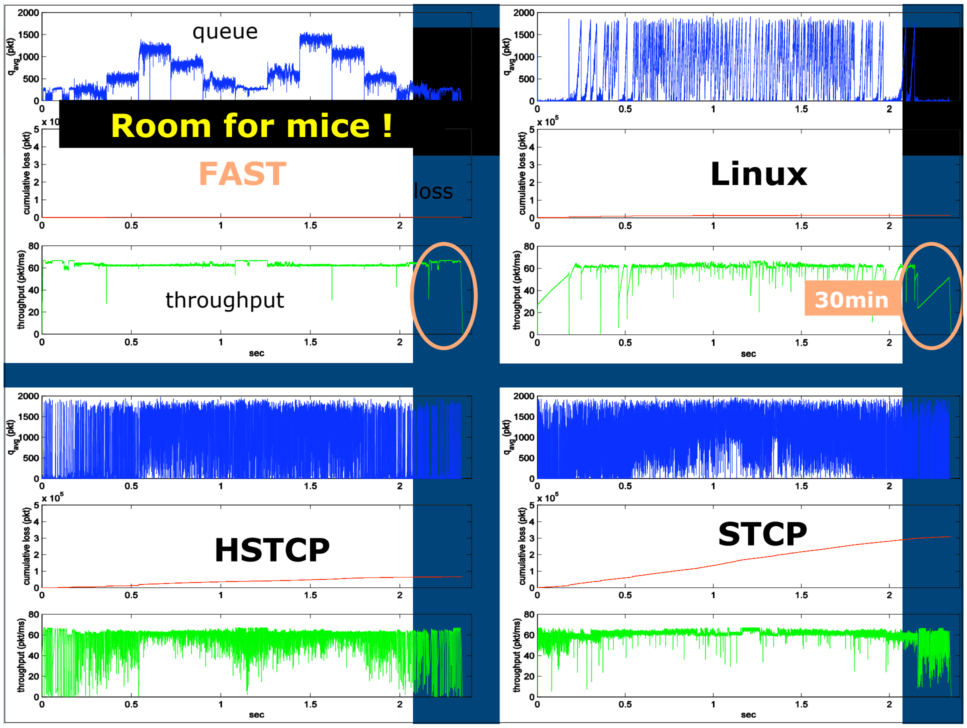


Dummynet: cap = 800Mbps; delay = 50-200ms; #flows = 1-14; 29 expts

Fairness

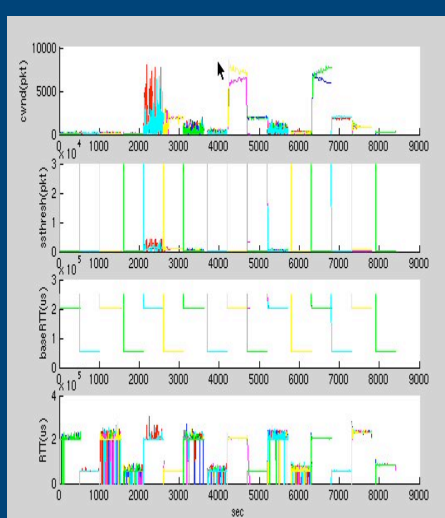


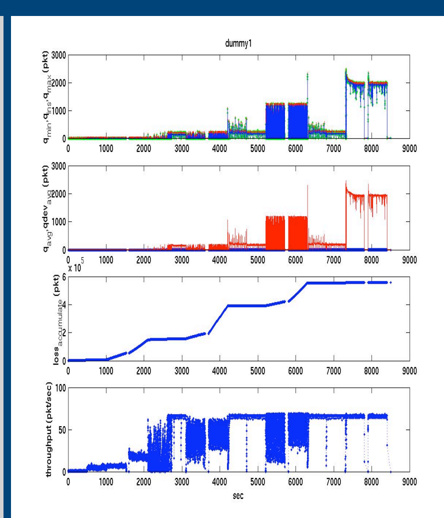
Dummynet: cap = 800Mbps; delay = 50-200ms; #flows = 1-14; 29 expts



FAST TCP v.s. Buffer Size

Sanjay Hegde & David Wei





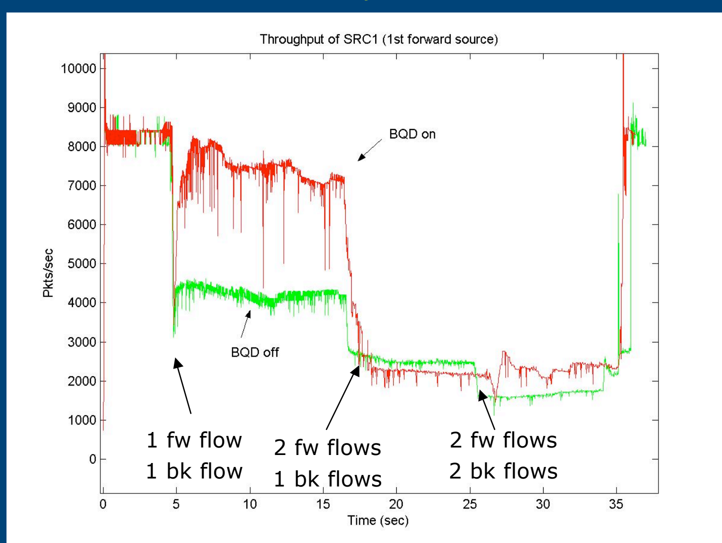
Backward Queueing Delay I

Bartek Wydrowski

- Use timestamp option on both sender and receiver
- Precision limited by sender clock
- Not requiring synchronization, same-resolution clocks, or receiver modification

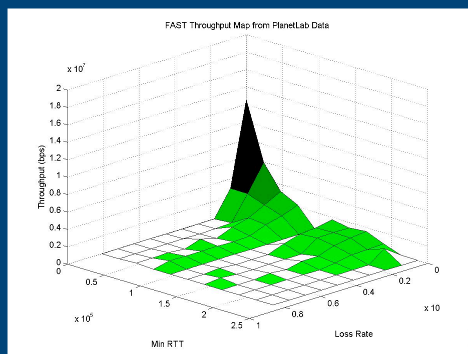
Backward Queueing Delay II

Bartek Wydrowski

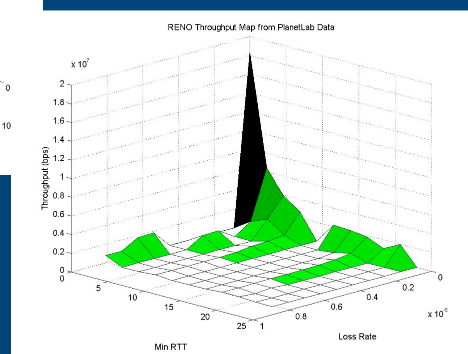


PlanetLab Internet Experiment

Jayaraman & Wydrowski



Throughput v.s. loss and delay
qualitatively similar results



FAST saw higher loss due
to large alpha value

Linux Related Issues

- Complicated state transition
  - Linux TCP kernel documentation
- Netdev implementation and NAPI
  - frequent delays between dev and TCP layers
- Linux loss recovery
  - too many acks during fast recovery
  - high CPU overhead per SACK
  - very long recovery times
  - Scalable TCP and H-TCP offer enhancements

Acknowledgments

- Caltech
  - Bunn, Choe, Doyle, Newman, Ravot, Singh, J. Wang
- UCLA
  - Paganini, Z. Wang
- CERN
  - Martin
- SLAC
  - Cottrell
- Internet2
  - Almes, Shalunov
- Cisco
  - Aiken, Doraiswami, Yip
- Level(3)
  - Fernes
- LANL
  - Wu

http://netlab.caltech.edu/FAST

- FAST TCP: motivation, architecture, algorithms, performance.

  IEEE Infocom 2004

- Code reorganization, ready for integration with web100.

- β-release: summer 2004

  Inquiry: fast-support@cs.caltech.edu

The End

Implementation Strategy

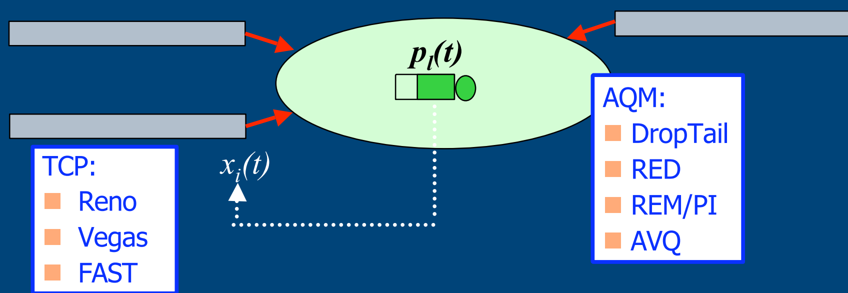
- **Common** flow level dynamics

$$\dot{w}_i(t) \quad = \quad \kappa(t) \quad \cdot \quad \left(1 - \frac{q_i(t)}{U_i'(t)}\right)$$

window adjustment = **control gain** **flow level goal**

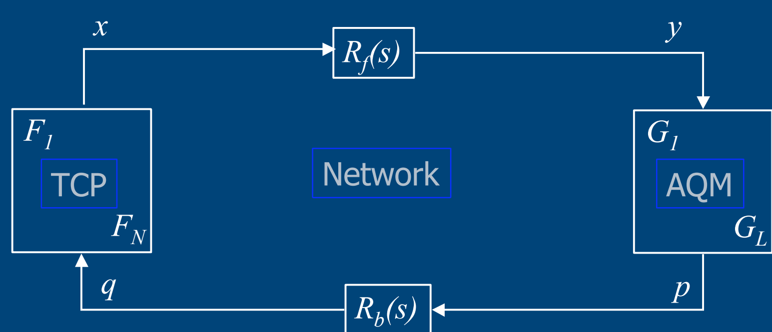
- Small adjustment when close, large far away
  - Need to estimate how far current state is from target
  - Scalable
- Queueing delay easier to estimate compared with extremely small loss probability

TCP/AQM



- Congestion control has two components
  - TCP: adjusts rate according to congestion
  - AQM: feeds back congestion based on utilization
- Distributed feed-back system
  - equilibrium and stability properties determine system performance

Network Model



- Components: TCP and AQM algorithms, and routing matrices
- Each TCP source sees an aggregate price, *q*
- Each link sees an aggregate incoming rate

FAST TCP

- Flow level
  - Understood and Synthesized first
- Packet level
  - Designed and implemented later

- Design flow level equilibrium & stability
- Implement flow level goals at packet level