Precise-feedback Congestion Control

Dina Katabi, Mark Handley

Problem: Congestion Control on Fast Links

- **The initial problem we set out to solve was that of congestion control for high delay-bandwidth product links.**

- **TCP works well when the mean window size ranges from about 1 to (say) hundreds of packets.**
  - For very slow links, the loss rate needs to be very high.
  - For very fast links, the loss rate needs to be extremely low.

Problem: Congestion Control on Fast Links

- **Assume we have 800Mb/s available per flow, 200ms mean RTT, 1KB packets.**
  - Mean window is 20 MBytes.
  - Assume it varies periodically from 13.3MB to 26.7MB under AIMD.
  - It takes 13333 RTTs to linearly increase from 13.3MB to 26.7MB.
  - 45 minutes per sawtooth (ie per loss).
  - One lost packet every 2x10^12 bits.
- **Uncorrected bit error rates are much higher than this!**

Problem: Congestion Control on Fast Links

- **How long does slowstart take?**
- **Assume we have 800Mb/s available per flow, 200ms mean RTT, 1KB packets.**
  - Assume slowstart ends when we reach mean window of 20MB.
  - Approx 40MB will have been sent.
  - Approx 25 RTTs, or 5 seconds elapsed.
- **Many (most?) flows will terminate before finishing slowstart.**
- **Long flows waste several seconds getting started.**

And 800Mb/s isn't really a very fast pipe!

- **Slow links won't go away.**
  - Wireless
  - Third world
  - AOL

- **How do we increase the dynamic range of congestion control?**

Assumptions and Goals

- **We don’t (initially) care about legacy networks.**
- **We want to work well on very slow links.**
- **We want to very rapidly converge on efficient use of the bottleneck link.**
- **We want to fairly rapidly converge on a fair share.**
- **We want routers to keep no per-flow state.**
- **End-system based congestion control is OK (we won’t worry about cheating for now).**

Router-mediated Congestion Control

- **What if the end-systems put their current congestion control parameters (window, RTT) in every packet, plus their desired window, and the routers can indicate how to change the window?**

- **Simple solution: AIMD, but let the routers specify the additive increase and multiplicative decrease constants according to their per-flow capacity.**
  - Each router along the path can reduce the constants to make the flow more conservative.
  - The receiver feeds the result back to the sender, which uses the new constants.

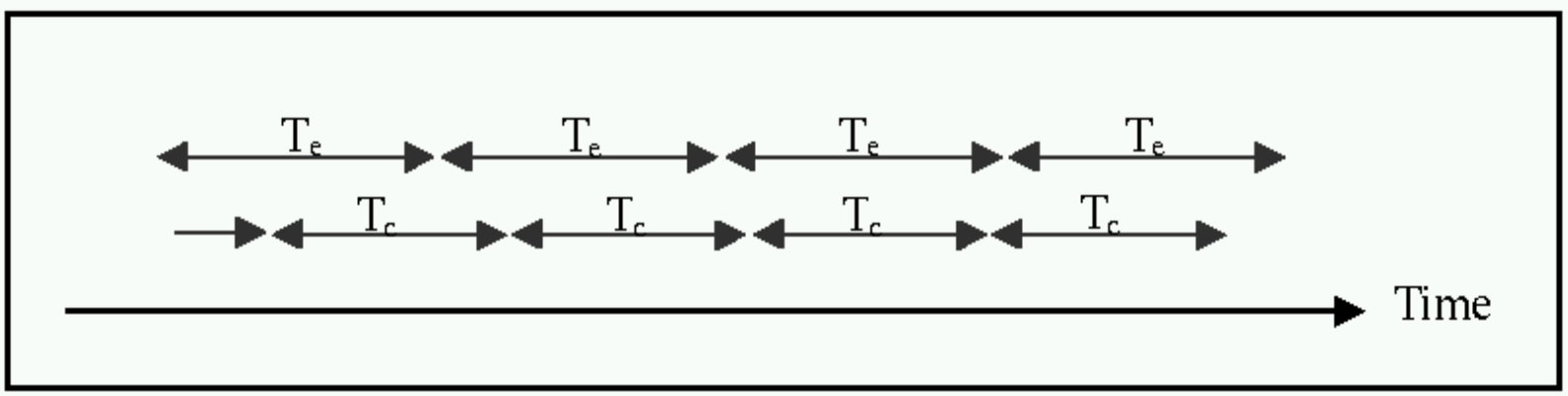
- **This gives the general idea, but isn't the actual solution we chose.**

Outline of solution

- **Separate Efficiency Control (how to fill the bottleneck) from Fairness Control (how to divide up the bandwidth).**
  - We can do this because (unlike TCP) every packet contains sufficient information.
- **Parts of the problem:**
  - What the routers can estimate?
  - How to do Efficiency Control.
  - How to add a Fairness Controller.

Estimation and Control

- **There are many ways to estimate link and traffic properties.**
- **We choose the simple mechanism of dividing time up into estimation intervals, and sample over those intervals.**
  - T_e is set to the average RTT of the flows on the link.
  - We want it to be large enough to capture the dynamics of the traffic, but short enough to be responsive.
- **We then apply the results of estimation to control the traffic during a control interval.**
  - T_c is set to the long-term (ie one month) average of the RTTs.



Estimated Parameters

- **Given the information in the packets, a router can accurately estimate the mean RTT, mean Cwnd of the flows, and number of flows.**

- **Eg. estimating the number of flows, N:**

$$N = \frac{\sum_{packet\ in\ T_e} (H_rtt / H_cwnd) \times 1}{T_e}$$

- **It can also measure**
  - the input traffic rate to the link queue
  - the spare bandwidth: SBW = link capacity - input rate

- **The objective of efficiency control is to make SBW zero.**

Efficiency Controller (v1)

- **During each T_c:**
  - BTA = Bits to Allocate
  - BTF = Bits to Free
  - SBW = Spare BandWidth

- **The router's going to dole out these to packets as they pass by to change the aggregate traffic load.**

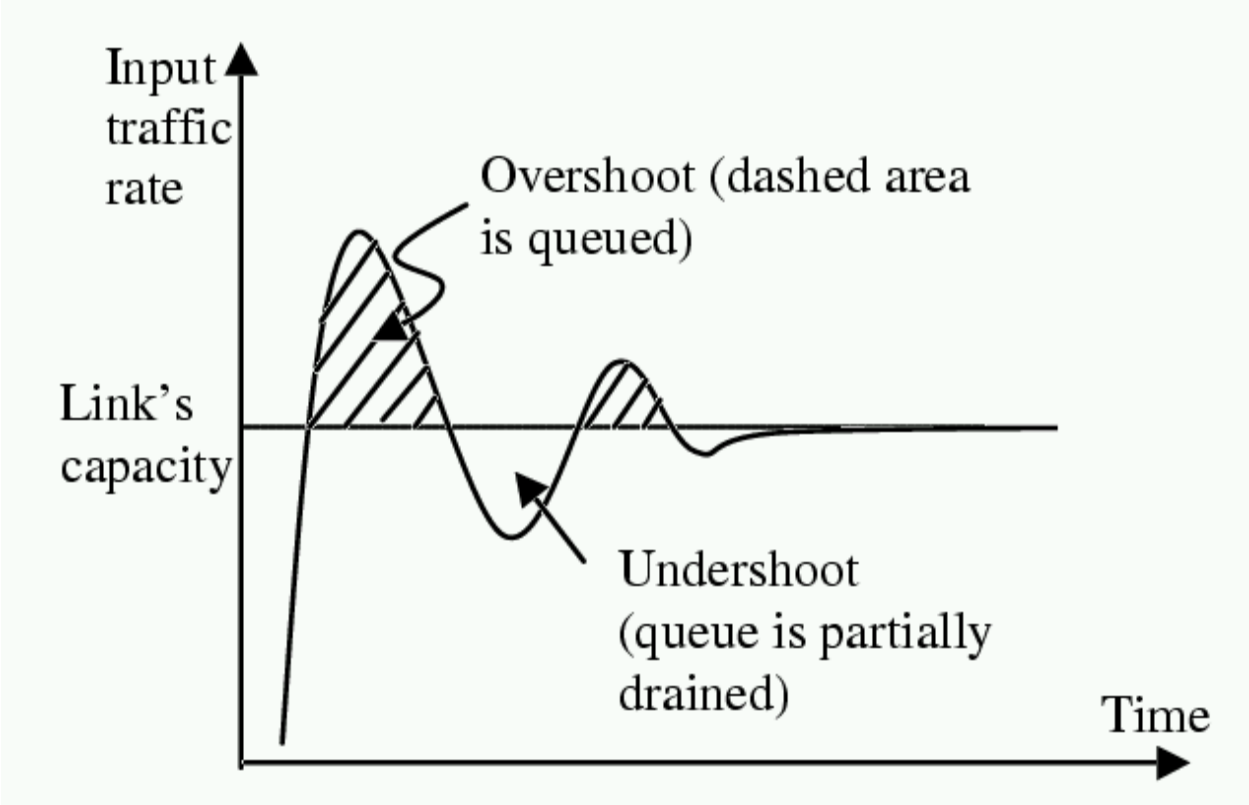
Obvious solution:

$$BTA = \max\left(0,\ k_1 \times SBW \times T_c\right);$$

$$BTF = \max\left(0,\ -k_1 \times SBW \times T_c\right); \qquad 0 < k_1 \leq 1$$

Efficiency Controller (v1)

- **Problem is that damped overshoots cause a persistent queue to build:**



Efficiency Controller (cont)

Modify the controller to drain the queue:

$$BTA = \max\left(0,\ k_1 \times SBW \times T_c - k_2 \times queue\right);$$

$$BTF = \max\left(0,\ -k_1 \times SBW \times T_c + k_2 \times queue\right),$$

queue is the queue size at T_c timeout.

But the traffic is bursty within an RTT (due to ACK compression, etc), so we don't want to aim to drain all the queue - we need to accomodate this transient queuing but reduce the persistent queue.

Finally:

$$BTA = \max\left(0,\ k_1 \times SBW \times T_c - k_2 \times \min_queue\right);$$

$$BTF = \max\left(0,\ -k_1 \times SBW \times T_c + k_2 \times \min_queue\right),$$

Efficiency Controller (cont)

- **All the Efficient Control requires is that the total traffic increases by BTA per T_c or decreases by BTF per T_c.**
  - How we divide these up between the packets (and hence between the flows) only affects fairness, and so is the job of the fairness controller.

- **Note: we've only dealt with one bottleneck router so far - we'll come back to the multiple bottlenecks case later and show that it only affects convergence time, not convergence itself.**

Fairness Controller

- **There are many ways we could do fairness control; we chose the fairly obvious:**
  - Allocate BTA to flows proportionally to 1/cwnd
  - Allocate BTF to flows proportionally to cwnd

- **We calculate positive and negative feedback separately, although we only need to feedback one value.**

FC: Positive Feedback

- **Allocate BTA to flows proportionally to 1/cwnd**
- **Thus per-packet positive feedback propostional to 1/(cwnd^2)**
  - (#pkts/RTT is proportional to cwnd)
- **This would allow flows to converge to the same cwnd. We'd prefer they converge to the same throughput.**
  - Throughput = cwnd/RTT, so the per-packet throughput needs to be scaled by the RTT too.
- **The sum of all positive feedback in T_c should be BTA, so:**

$$positive_feedback = \eta \frac{BTA}{T_c} \left(\frac{H_rtt}{H_cwnd} \right)^2;$$

  - n is a scaling factor calculated every T_e

FC: Negative Feedback

- **Allocate BTF to flows proportionally to cwnd**
  - Per-RTT negative feedback is proportional to cwnd
  - Thus per-packet negative feedback is independent of cwnd

- **We don’t need to for RTT as this is a multiplicative decrease.**

- **The sum of all negative feedback in T_c should be BTF so:**

$$negative_feedback = \frac{BTF}{N_p}$$

  - N_p is the number of data packets in a control interval.

Ensuring Convergence

- **So far, this doles out BTA and BTF in a way that ensures convergence towards fairness.**
  - But, BTA and BTF can both be zero (or small), in which case we don't converge (or converge very slowly).

- **To guarantee fast converges, we use Bandwidth Shuffling - the simultaneous allocation and deallocation of bandwidth in a way that doesn't change the total traffic rate, but that moves bandwidth between flows.**

Ensuring Convergence

- **Every T_c we shuffle 10% of the bandwidth**
  - Bits to Shuffle = BTS

$$BTS = k_3 \times (capacity - SBW) \times T_c; \quad k_3 = 0.1.$$

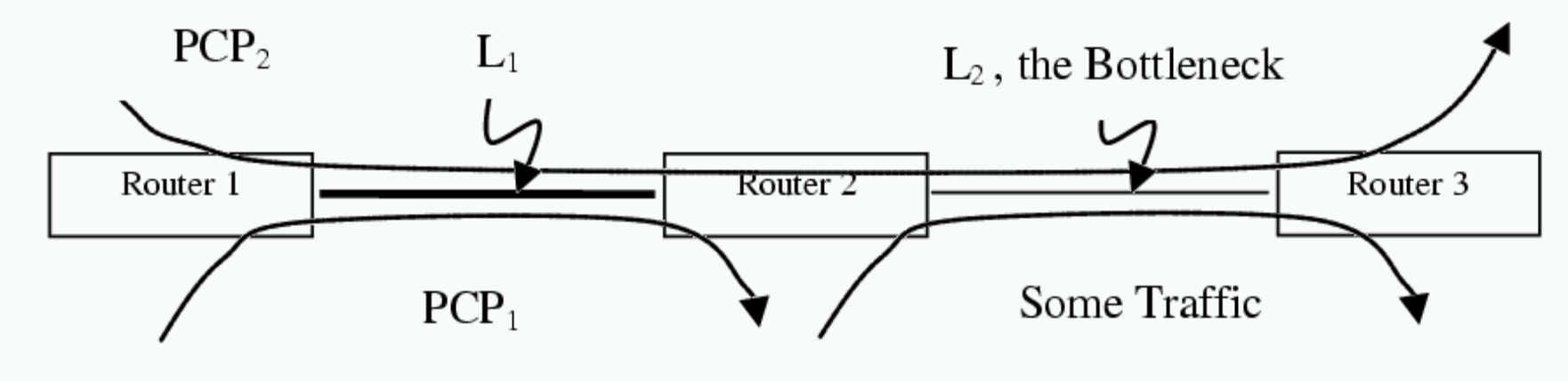
- **The positive and negative feedback then become:**

$$positive_feedback = \eta \frac{BTA + BTS}{T_c} \left(\frac{H_rtt}{H_cwnd} \right)^2$$

$$negative_feedback = \frac{BTF + BTS}{N_p},$$

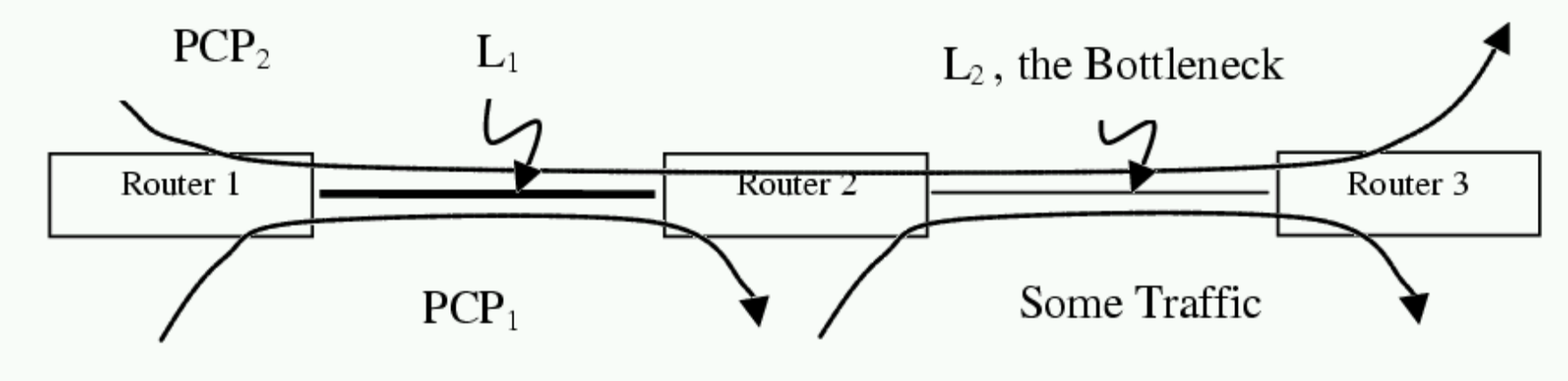
Note that we never hand out more than BTA+BTS, or BTF+BTS in any T_c - if we run out the remaining packets get no change.

Multi-hop Paths



- **R1 allocates bandwidth to PCP1 and PCP2. However, PCP2 gets the allocated bandwidth removed again by R2.**
- **As PCP1's throughput gets much higher than PCP2's, convergence to good itilization decreases, as R1 gives almost of of BTA to PCP2.**

Multi-hop Paths



- **To prevent this, we have a header bit that indicates a flow traverses a link with less that 10% SBW.**
  - Bit is set in a data packet, echoed by the receiver, and set again in the subsequent data packets.
- **BTA is only give to flows without the "bottleneck bit" set.**
- **BTS ensures final convergence.**

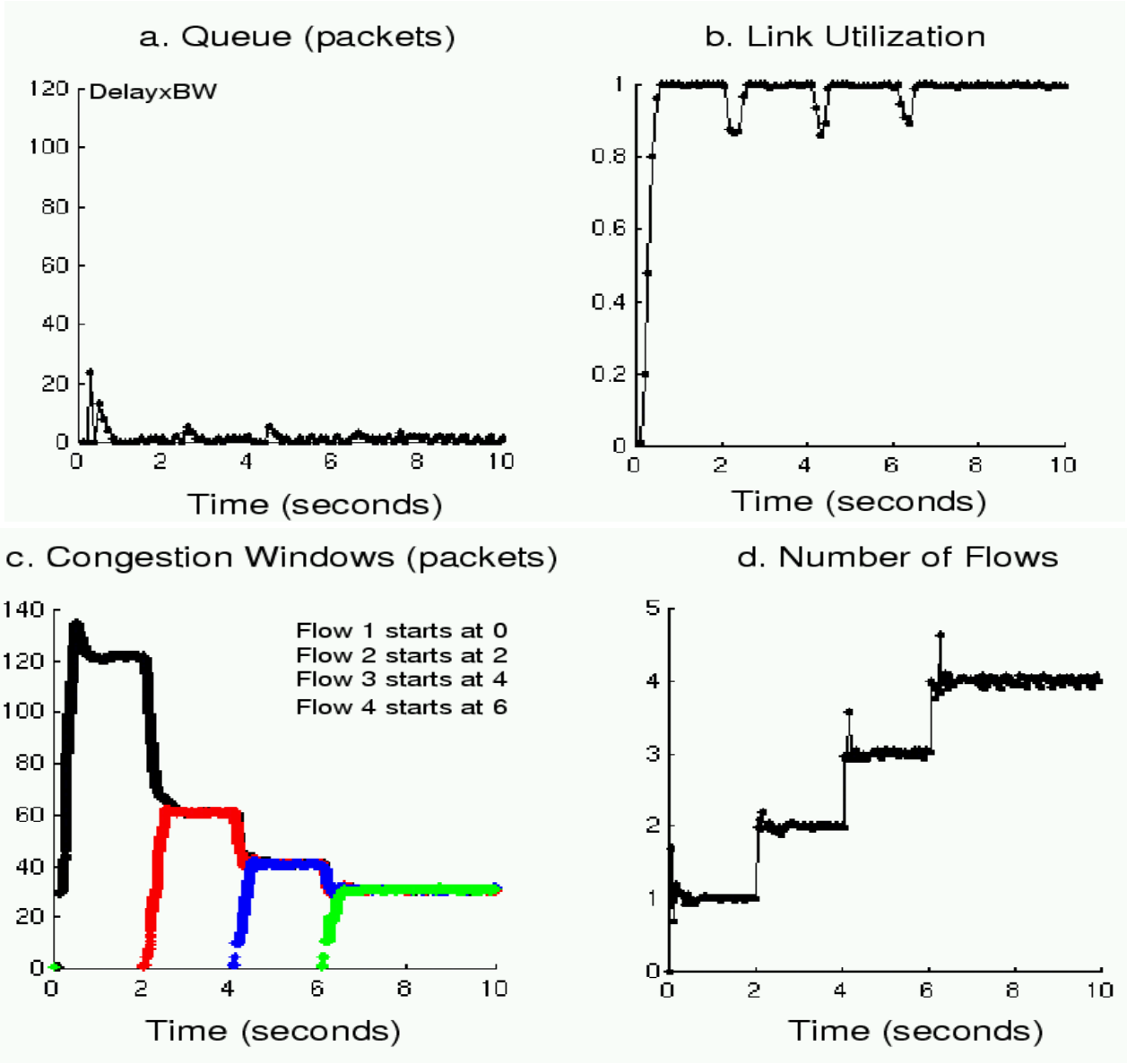
Co-existence with TCP

- **PCP can co-exist with TCP and achieve a fair share.**

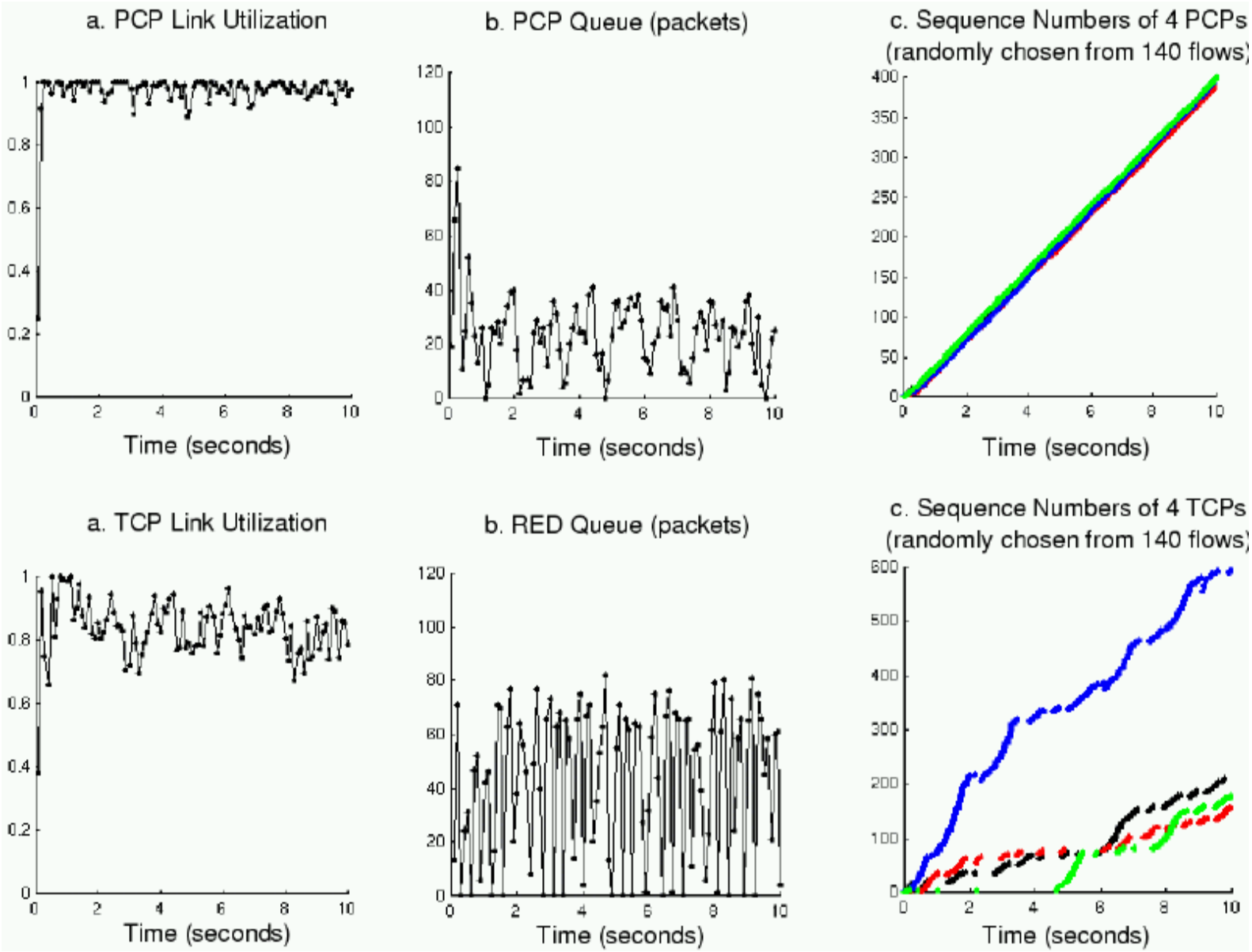
- **Each PCP router manages two queues; one for best effort, one for PCP.**
  - It knows the mean cwnd of PCP flows (from its estimators).
  - It knows the loss rate of best effort flows at that queue.
  - It can estimate the cwnd of TCP flows from the loss rate using the TCP throughput equation.

- **Given this information, it can change the weighting between the two flows in such a way as to equalize the cwnd.**
  - Need to use a modified SBW that reflects PCP's usage of its fair share of the bandwidth to prevent TCP driving SBW negative and starving the PCP flows.

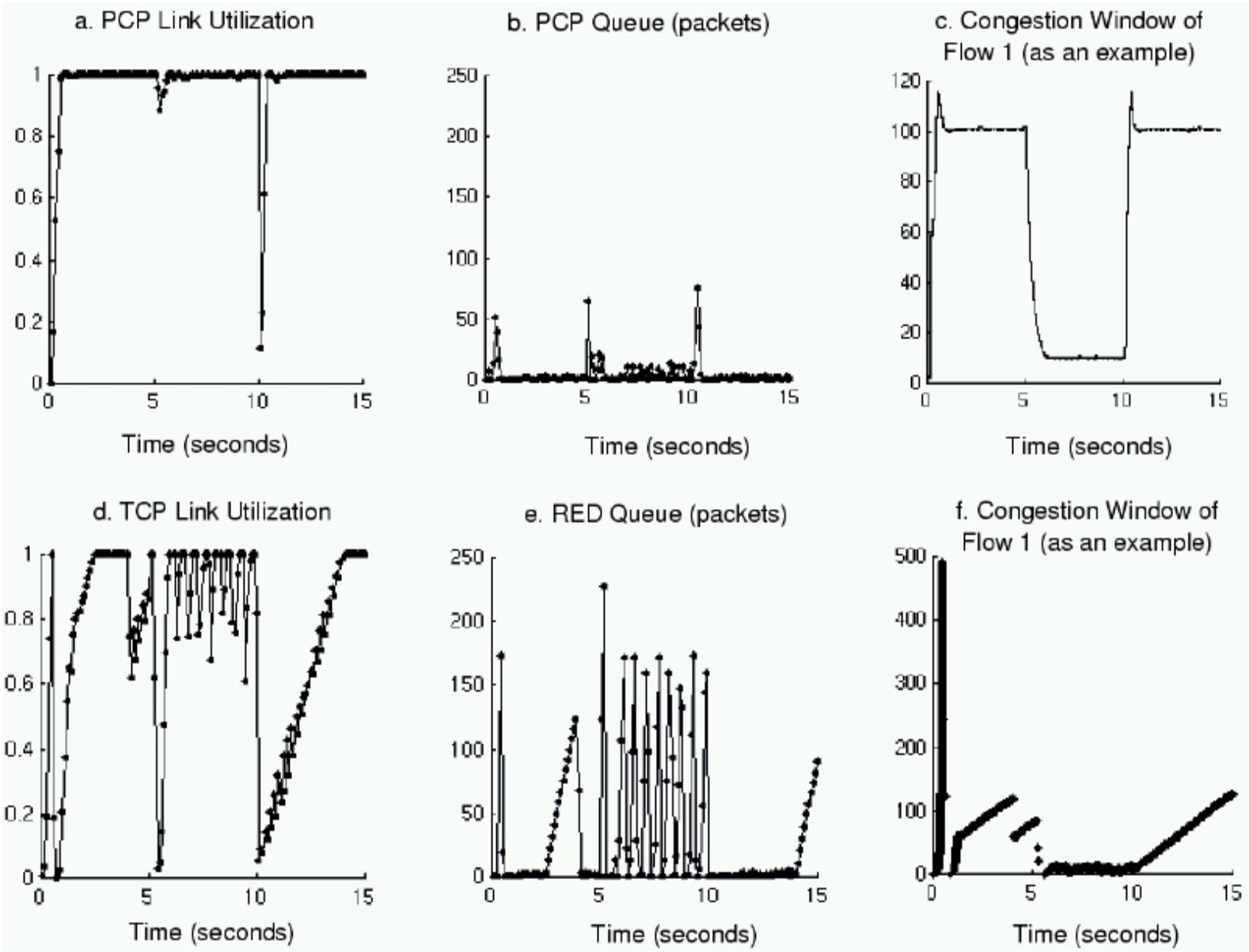
Performance



Performance



Performance



Conclusions

- **We set out to design a congestion control algorithm that improved the dynamic range of TCP.**
- **We ended up with one that provides precise control over traffic with no per-flow state in routers.**
  - Low loss rate
  - Good utilization
  - Fast convergence
- **We still don't know how to examine it on very high speed links because of the limits of the simulators.**
- **We have limited experience of it with realistic traffic.**
- **We have ignored malicious routers and end systems, and how badly they can harm us.**
- **We have ignored how expensive this is to implement in routers.**