Improving TCP Performance with Path Error Rate Information

—

Wesley M. Eddy

Outline

- TCP Background
- The Problem with Packet Errors
- Measuring Loss Rates
- Modifying Congestion Control
- Future Work & Summary

Background

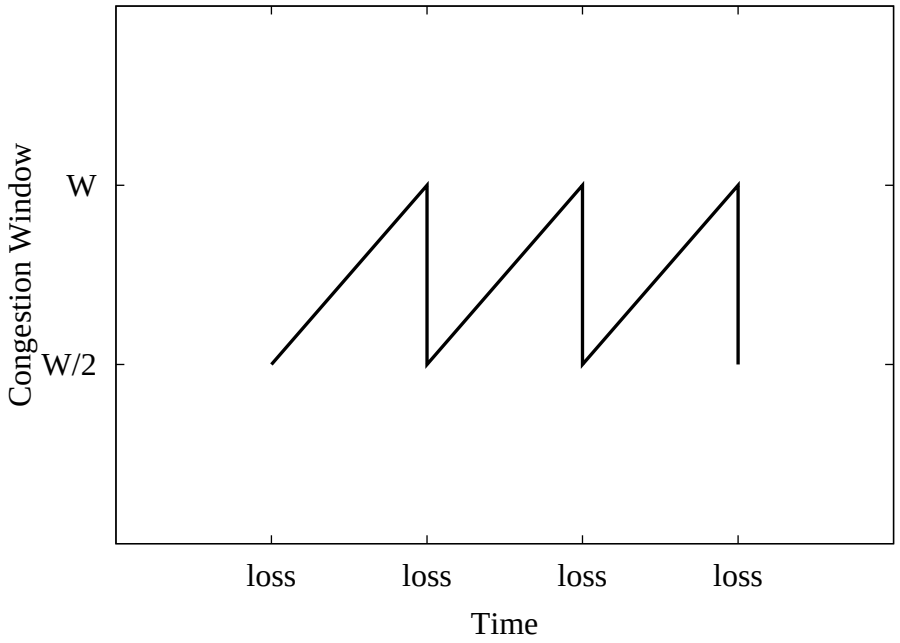
- We're going to focus on fixing TCP.
  - It's not really "broken", it just underachieves.
  - It's what (practically) everything uses.
  - Other transports use the same information and similar algorithms to TCP and so we expect our work should cover them as well (e.g., SCTP, TFRC).

Background (cont.)

- TCP worked well until the mid-80s when the Internet suffered from *congestion collapse*.
  - Think about traffic jams, lots of things moving, but really slowly
- Van Jacobson added a set of *congestion control and avoidance* techniques to TCP to combat congestion collapse.

Background (cont.)

- Steady state TCP:



Problems with CC

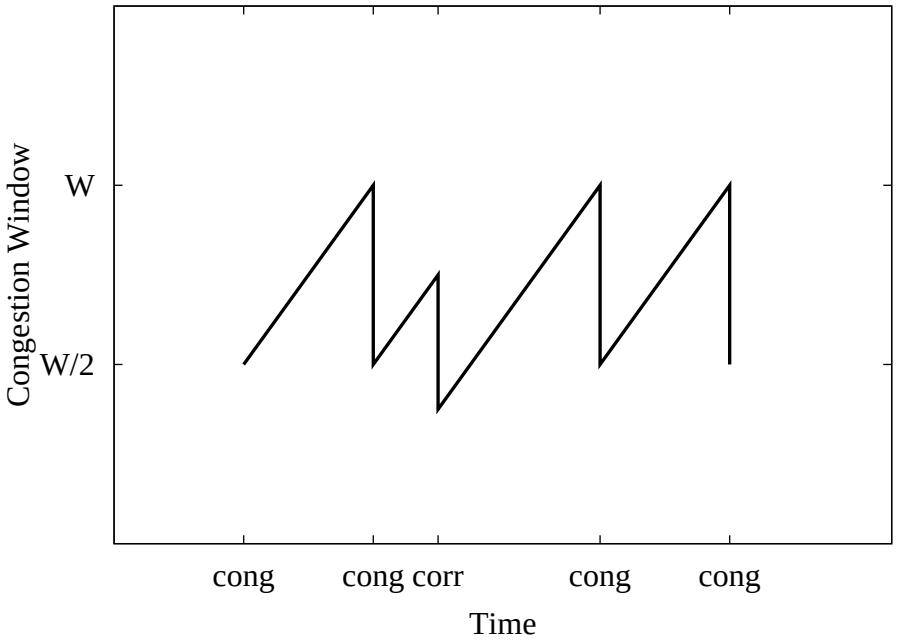
- The premise of Jacobson's work is that *nearly all* packet loss is caused by overloaded (*congested*) routers.
  - Not a bad assumption (then or now).
- But, there are exceptions:
  - Noisy wireless connections
  - lousy hardware (see Stone/Partridge, SIGCOMM 2000)?

Problems (cont.)

- If a TCP connection experiences significant losses that are not congestion-related then its performance suffers
- E.g., just because a bird flew in front of your antenna does not mean that there is any reason for TCP to reduce the sending rate.
- Fundamental Problem: TCP has no way to derive the cause of a packet loss.

Problems (cont.)

- Steady state with non-congestion-loss:



TCP Model

- An analytical model of TCP performance has been developed:

$$R = \frac{MSS}{RTT \cdot \sqrt{\frac{2bp}{3}} + \left(RTO \cdot \sqrt{\frac{3bp}{8}} \cdot p \cdot (1 + 32p^2) \right)}$$

- Developed by Mathis (CCR 1997), Padhye (SIGCOMM 1998), et. al.

- There are a few variants, but all have the same basic form.

TCP Model (cont.)

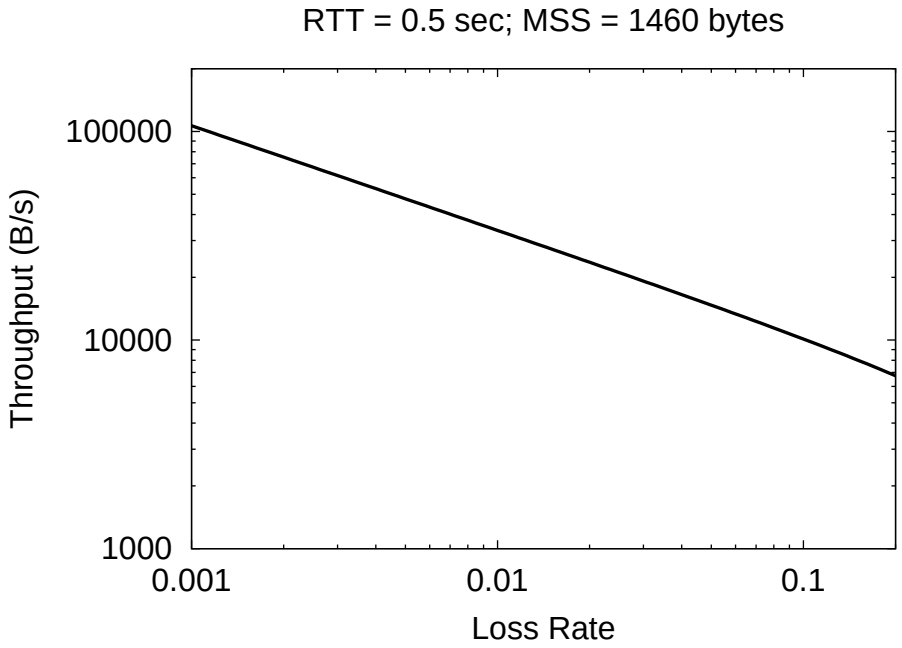
- For our purposes the model can be distilled to:

  $R \propto \frac{1}{\sqrt{p}}$

- This makes sense because the goal of congestion control is to avoid congestion collapse by adapting the sending rate.
  - So, as the loss rate increases the sending rate decreases.

TCP Model (cont.)

- Model TCP performance:



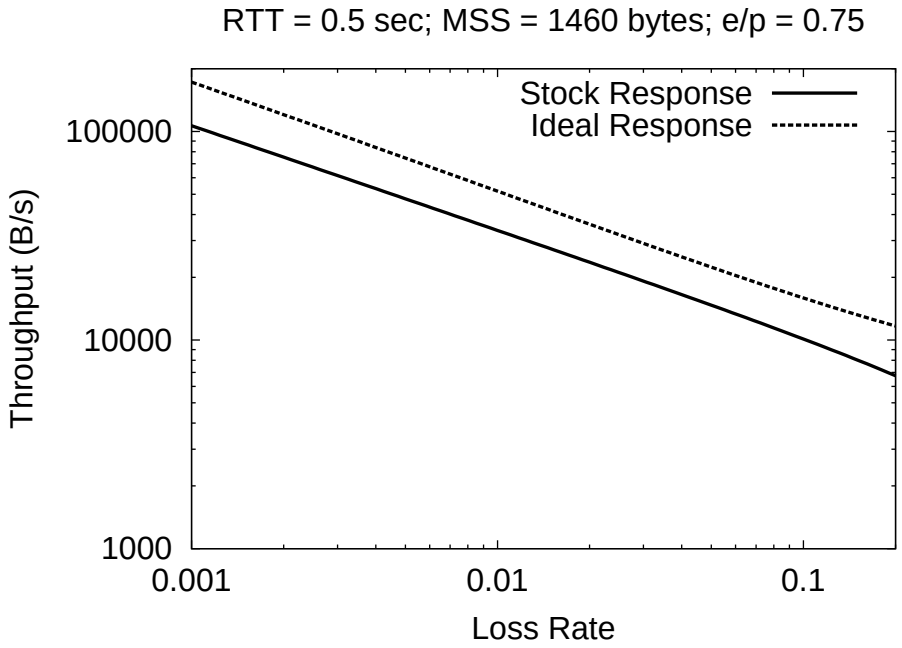
TCP Model (cont.)

- But, p is a combination of congestion-based loss (c) and corruption-based loss (e): $p = c + e$
- Ideally we'd like to change TCP's congestion response function:

  $R \propto \frac{1}{\sqrt{p}} \quad \Rightarrow \quad R \propto \frac{1}{\sqrt{c}}$

TCP Model (cont.)

- Ideal TCP performance:



CETEN

- Cumulative Explicit Transport Error Notification
  - Originally outlined by Krishnan, Sterbenz, Partridge, Allman
    - BBN tech report
  - Refined by Eddy, Ostermann, Allman in this thesis
- If TCP can obtain two of p, c or e we have the whole story about losses and can form a more intelligent congestion response.
  - Surprisingly, the TCP endpoints actually have none of these quantities.
    - We estimate p at the sender
    - We ask the network for "e"

Estimating p

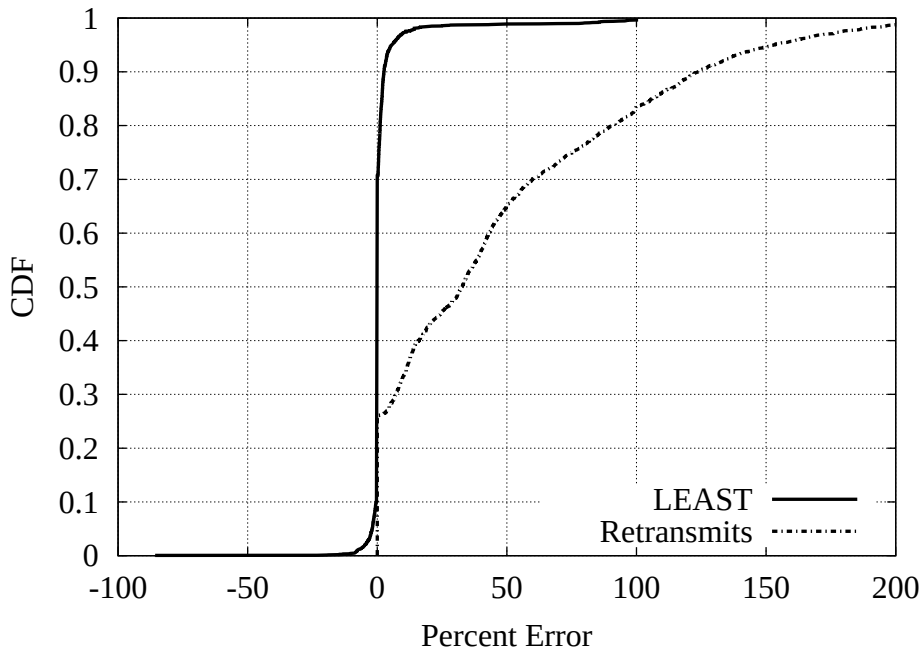
- At first glance it looks easy to determine the total loss rate of a TCP connection since it is reliable.
  - I.e., just count the retransmits
- However, depending on TCP variant the retransmission mechanism is fairly gross.
- We developed several algorithms for estimating the total loss rate based on the number of retransmits and hints coming back from the receiver as to which retransmits were not required.
  - LEAST: Loss Estimation AlgorithmS for TCP
  - Paper in Performance Evaluation Review, December 2003.

Estimating p (cont.)

- The point is - losses = *needed* retransmits
  - needed = total - unneeded
  - we can measure total exactly
  - we estimate unneeded

Estimating p (cont.)

- LEAST performance:



Estimating e

- No good way for the end hosts to determine why an intermediate node dropped a packet.
- So, we involve the routers.

Estimating e (cont.)

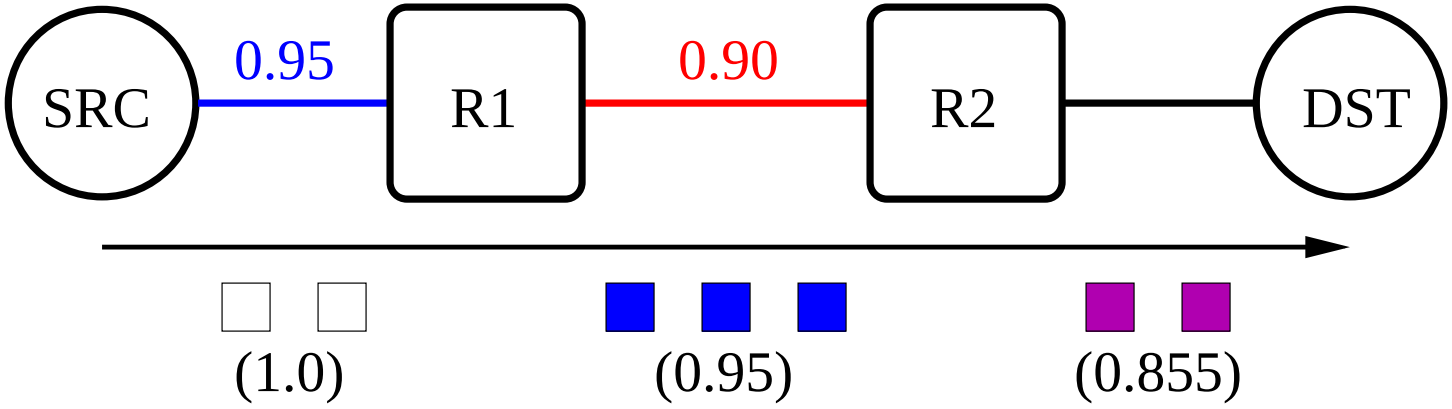
- Tag packets with a "corruption survival probability" header field.
  - Initialized to 1.0 by the sending TCP
  - Updated by each router along the path by multiplying the value in the packet with the probability of corruption survival on the incoming link.
  - When a packet arrives at the receiver the probability in the packet represents the probability of corruption survival across the entire path — this probability is echoed to the TCP sender in ACKs.

Estimating e (cont.)

- Pros: no extra control traffic, more reliable than polling or random sampling
- Cons: we have to change (or extend) the network or transport layer protocol

Estimating e (cont.)

- CETEN e collection example:



Adjusting the Response (1)

- On each loss event TCP flips a coin weighted by e/p to determine whether the congestion window is reduced or not.
- On average the long term reduction factor should be based on "c" not "e"
- Denoted "CETEN-C"

Adjusting the Response (2)

- Rather than using a static multiplicative decrease factor (MDF) of 1/2 at the TCP sender, a variable MDF is computed as:

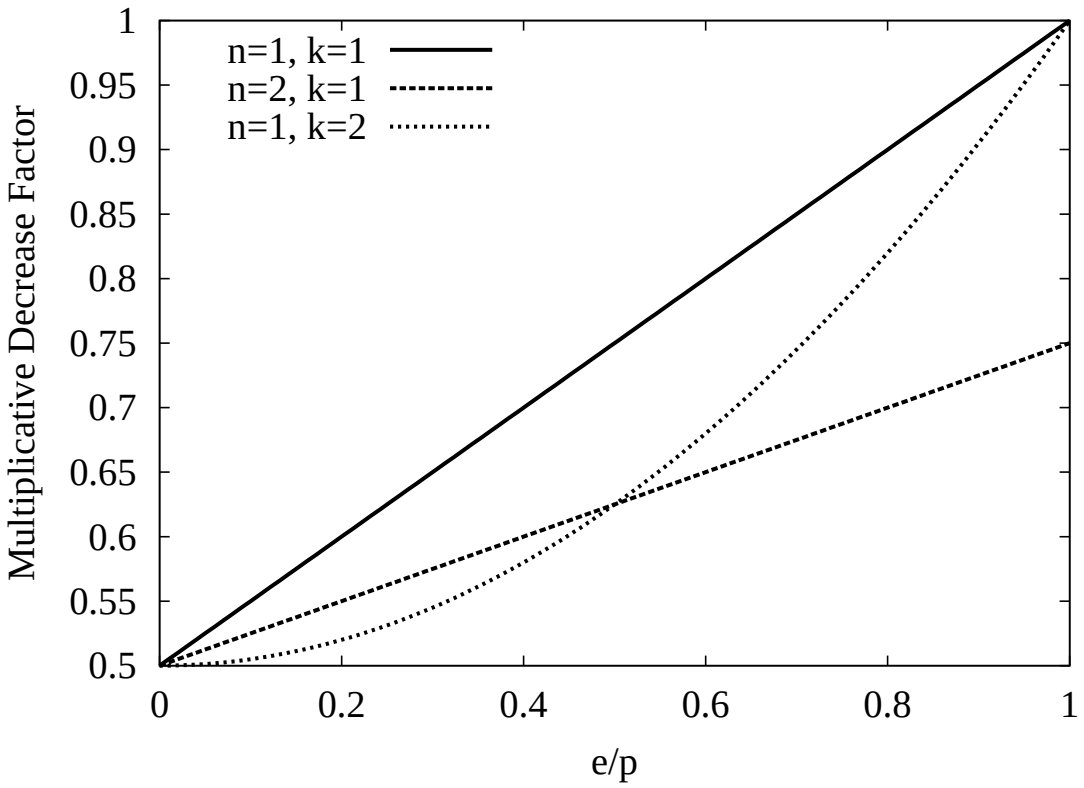
  $MDF = \frac{1+(\frac{e}{np})^k}{2}$

- Where n and k are shaping and bounding parameters.

- Denoted "CETEN-A"

Adjusting the Response (2)

- Example MDF parameter sets:



Deployment

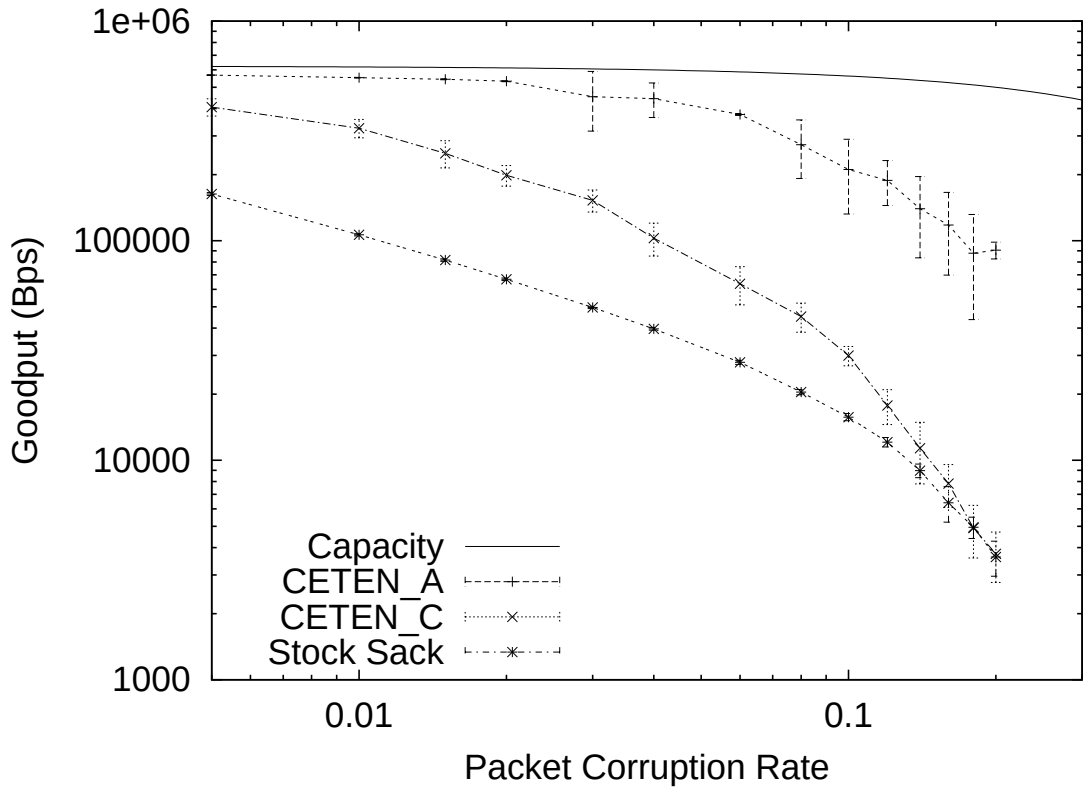
- CETEN does not require ubiquitous deployment.
- Rather, CETEN is only needed on routers/base-stations where there are non-negligible corruption rates.
  - And, *needed* is an overstatement

Preliminary Evaluation

- Implemented CETEN in the *ns* network simulator
- Dumbell topology:
  - RTT of roughly 85 msec
  - Bottleneck bandwidth of 5 Mbps
  - Drop-tail routers with 150 packets worth of queueing capacity (on order of BD-product)
- SACK TCP
  - MSS = 1460 bytes
  - with delayed ACKs
- Uniform loss model (!)

Single Flow Tests

- One end-to-end TCP flow



Single Flow Tests (cont.)

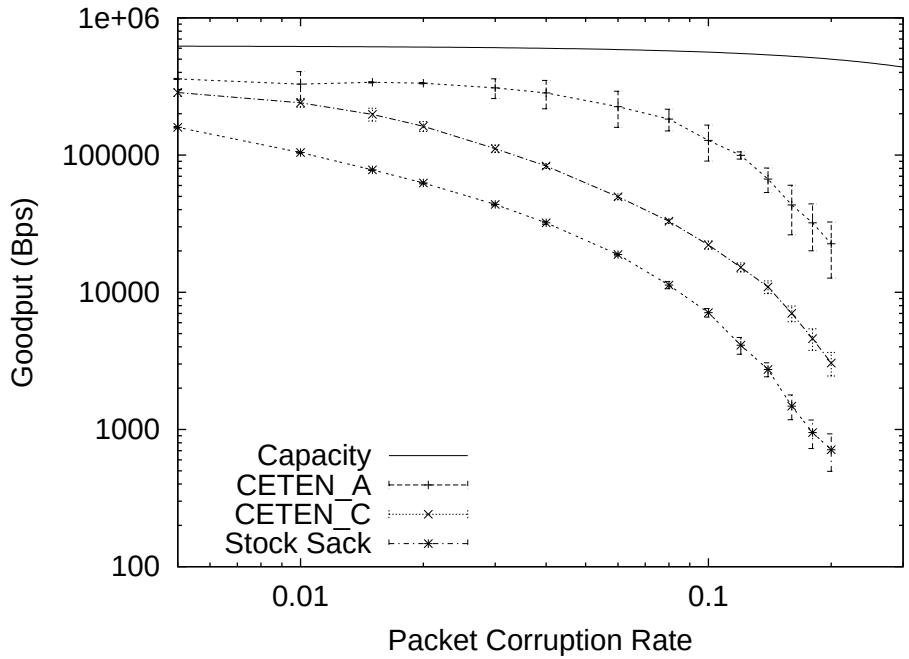
- CETEN-C is flawed in that it does not account for the change in the loss probabilities caused by its "incorrect" congestion response
  - It guesses right at least 50% of the time
  - But bad guesses can blow up the loss rate by increasing congestion

Tests with Cross Traffic

- One TCP connection in each direction
- 5 on/off CBR flows in each direction
  - Mean on time: 2.5 seconds
  - Mean off time: 10 seconds
  - When on each flow sends at 1 Mbps (one-fifth of the bottleneck bandwidth)

Tests with Cross Traffic (cont.)

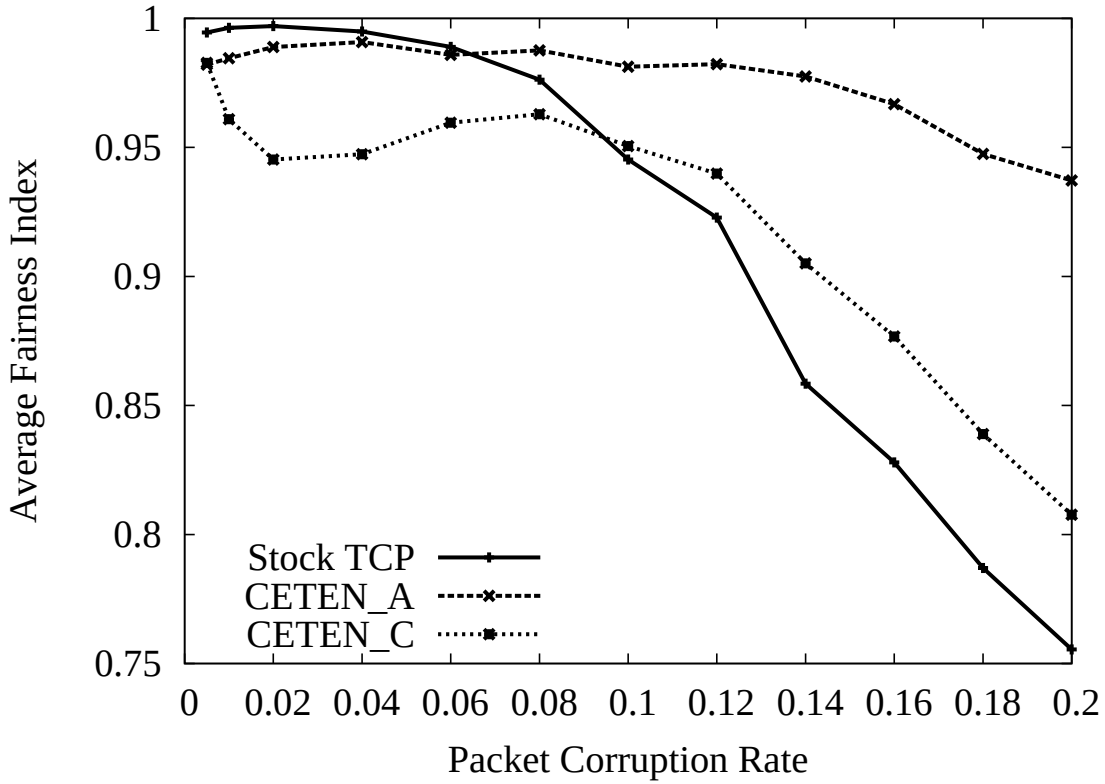
- Results from congested network:



Fairness Experiments

- TCP (mostly) shares evenly across like flows
- Does CETEN?
- Experiment
  - 20 competing flows
    - all of the same variant
  - Metric: Jain's fairness index

Fairness Experiments (cont.)



Results Summary

- Both versions of CETEN aid performance, with CETEN-A gaining better performance than CETEN-C
- CETEN-A is a promising technique
  - Offers nice performance benefits
  - Offers good fairness properties
- But, CETEN is still is a heavy-weight mechanism

Future Work

- How do routers average and report corruption rates? Over what timescales?
- Can routers manipulate packets to include e in an efficient enough way? At what speeds?
- What does CETEN performance look like under a different corruption loss model?

Future Work (cont.)

- How do we prevent lying receivers from gaming the sender's congestion control for their own benefit?
- How do we prevent DoS attacks on routers that involve making them spend more cycles on every packet than they otherwise would?
- How much information should the network be expected to provide to the end hosts?

Summary

- CETEN is an interesting and potentially useful technique for improving performance for a certain class of network traffic
  - in light of the increasing amount of wireless traffic
  - space communications
- Potential gains are large
- Deployment is "possible"
  - Especially since it's incremental

More Information

- Me: weddy@irg.cs.ohiou.edu
- Project web page:

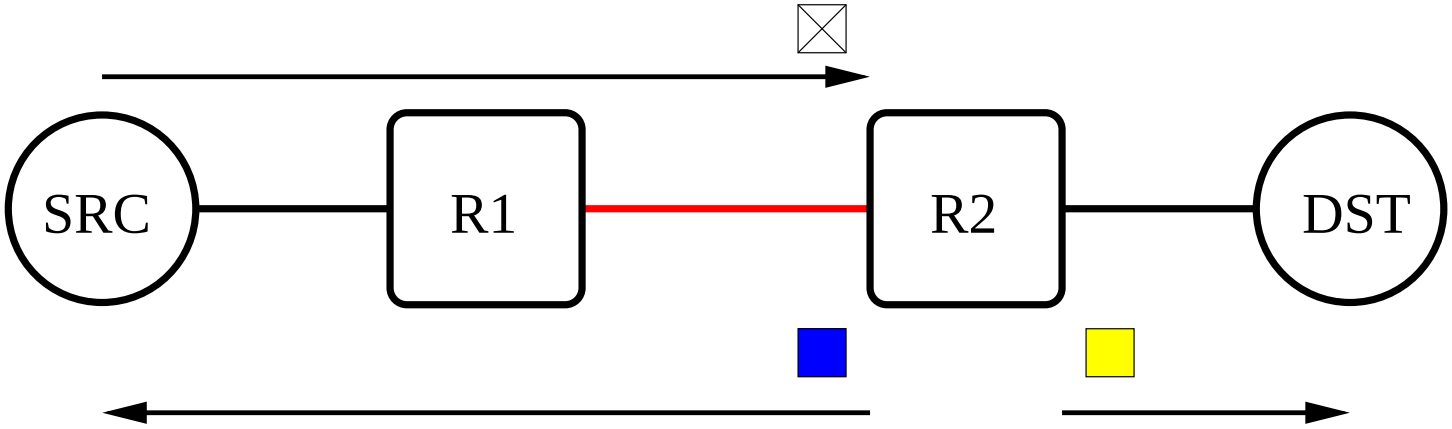
  `http://www.icir.org/mallman/research/proj-eten.html`
- Questions? Comments?

Previous Work

- The literature is filled with potential solutions to the performance problems caused by non-congestion based loss.
- Three general classes:
  - Notification schemes
  - Local repair
  - Connection splitting
    - Breaks the end-to-end nature of TCP
    - Omitting from discussion today

Notification Schemes

- When a packet is detected as corrupted by the data-link layer a notification is sent to one of the endpoints of the connection.
  - What if the addresses are corrupted?
  - What if the addresses are encrypted?

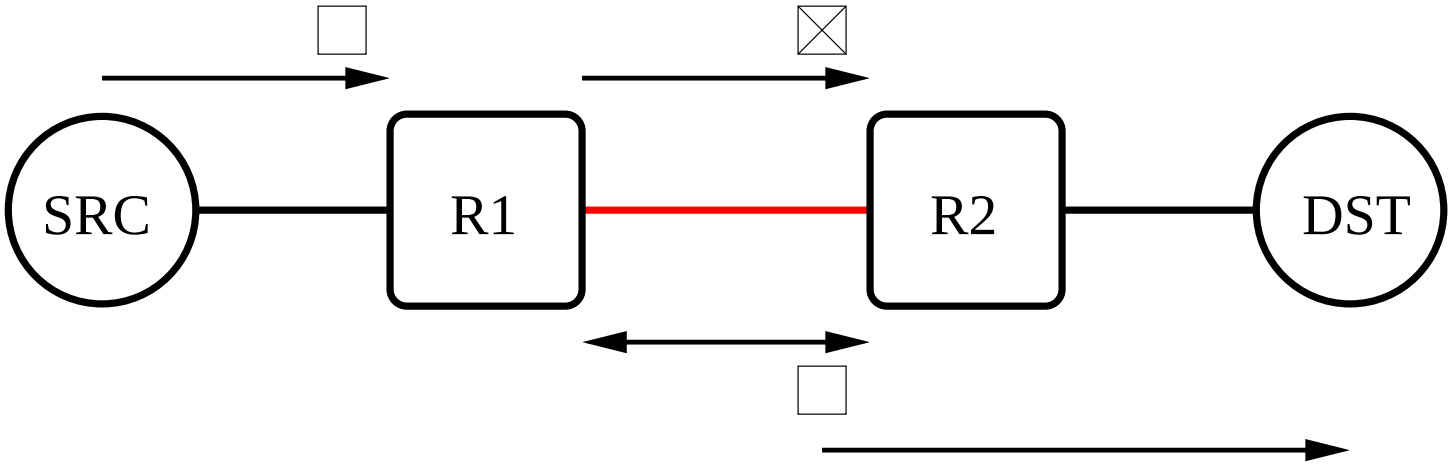


Local Repair

- Each link is responsible for presenting a "clean" (error free) transmission path
  - ARQ (layer 2), snooping (layer 4)
  - FEC (layer 2)
- Potential problems:
  - Requires time or bandwidth

Notification Schemes (cont.)

- ARQ:



- FEC: