A(nother) Technique for Improving Transport Protocol Performance in Lossy Networks

Mark Allman
mallman@icir.org

ICIR Wednesday Lunch

April 28, 2004

"And he don't want to change what don't need to change"

Acknowledgments

- Joint work with:
  - Wesley Eddy (Ohio Univ.)
  - Rajesh Krishnan (BBN)
  - Shawn Ostermann (Ohio Univ.)
  - Craig Partridge (BBN)
  - James Sterbenz (UMass)
- Funding from NASA GRC and NSF

Outline

- Problem statement
- Related work
- New technique: CETEN
- Evaluation
- Future work
- Summary

Problems with CC

- The premise of TCP congestion control is that *nearly all* packet loss is caused by resource contention.
  - likely true
  - conservative
- But, not universally true
  - e.g., what if your connection is via RF?
  - e.g., what if you happen to sit behind lousy hardware (see Stone/Partridge, SIGCOMM 2000)?

Problems with CC (cont.)

- If a TCP connection experiences non-negligible amounts of loss that are not congestion-related then the performance of the connection will suffer.

- 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.

- Fundemental Problem: TCP has no way to derive the cause of a packet 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 Floyd, Mathis, Padhye, etc.

- 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.)

- 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:

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

Outline

- Problem statement
- Related work
- New technique: CETEN
- Evaluation
- Future work
- Summary

Related Work

- Local solutions
  - layer 2 (FEC & ARQ) and 4 (ARQ; snooping)

- Notification solutions
  - explicit loss notification

- Performance Enhancing Proxies
  - "spoofing"

Related Work (cont.)

- Taxonomy
  - Rajesh Krishnan, James Sterbenz, Wesley Eddy, Craig Partridge, Mark Allman. *Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks.* Computer Networks. To appear.

Outline

- Problem statement
- Related work
- New technique: CETEN
- Evaluation
- Future work
- Summary

CETEN

- *Cumulative Explicit Transport Error Notification*

- 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.

Estimating "p" (cont.)

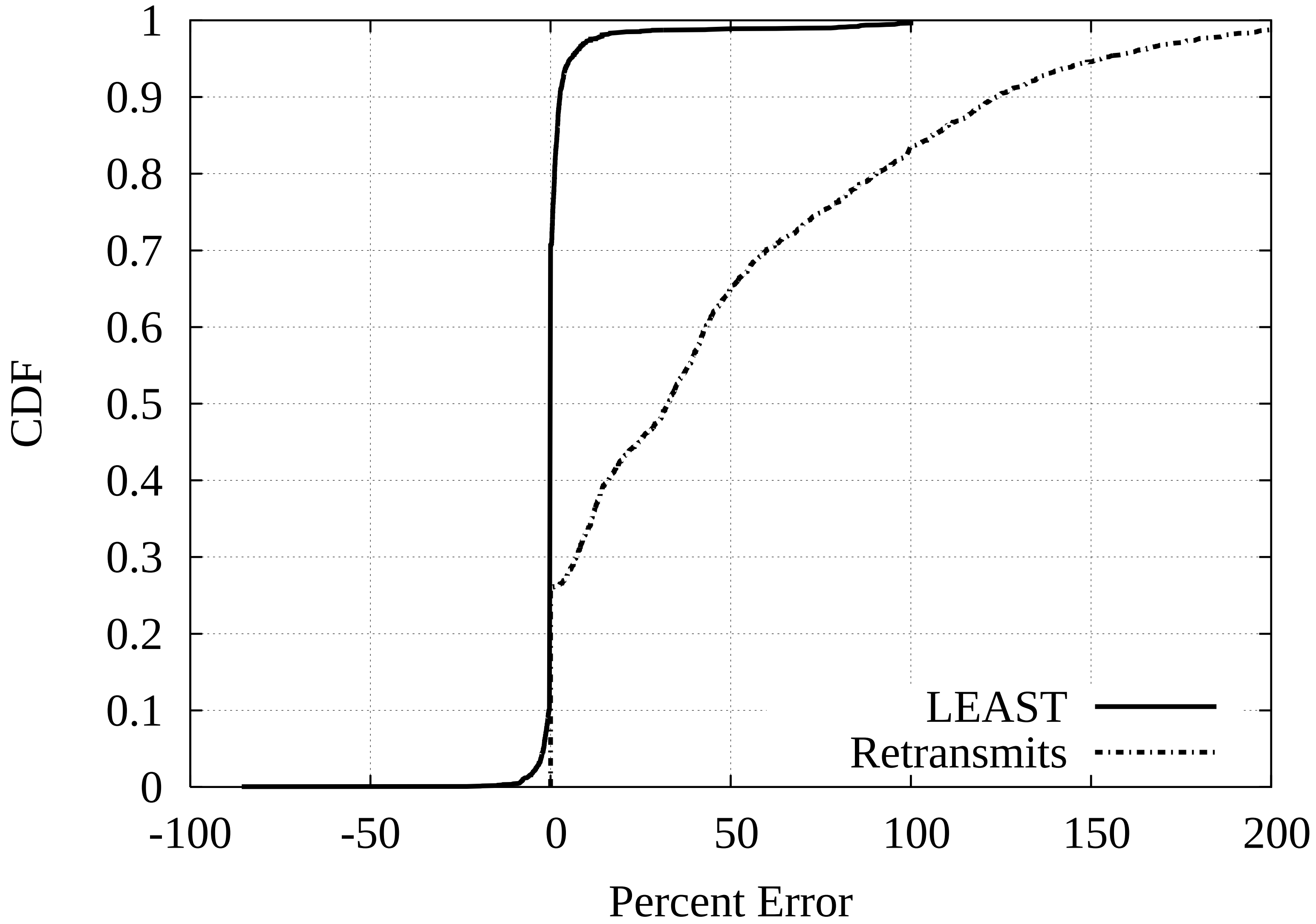
- We developed several algorithms for estimating the total loss rate based on the number of retransmits and hints coming back from the receiver as to which retransmits were not required.
  - LEAST: Loss Estimation AlgorithmS for TCP
  - Mark Allman, Wesley Eddy, Shawn Ostermann. *Estimating Loss Rates With TCP.* ACM Performance Evaluation Review, 31(3), December 2003.

Estimating "p" (cont.)

- LEAST experiments:
  - 2600+ transfers (5000 packets each)
  - NIMI mesh (20-ish hosts)
  - cap utility (Allman, IMW/2001)
  - tracing on sender and receiver
    - can accurately assess the actual loss rate
    - also, estimate using LEAST on the sender

Estimating "p" (cont.)

- LEAST performance:



Estimating "e"

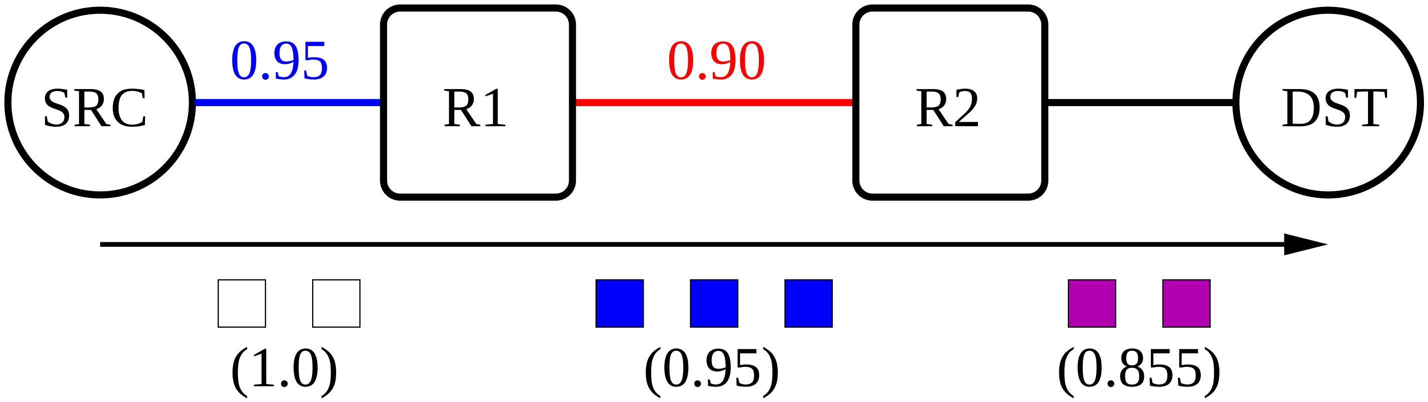
- No good way for the end hosts to determine why an intermediate node dropped a packet.
  - or, this whole problem would go away!
- So, we involve the intermediate nodes
  - routers, base stations, access points, etc.
  - the nodes that observe corruption

Estimating "e" (cont.)

- A packet is tagged 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
    - echo probability back to the TCP sender in ACKs
- There are alternate methods for getting the information ...

Estimating "e" (cont.)

- CETEN "e" collection example:



Adjusting the Response (1)

- On each loss event flip 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:

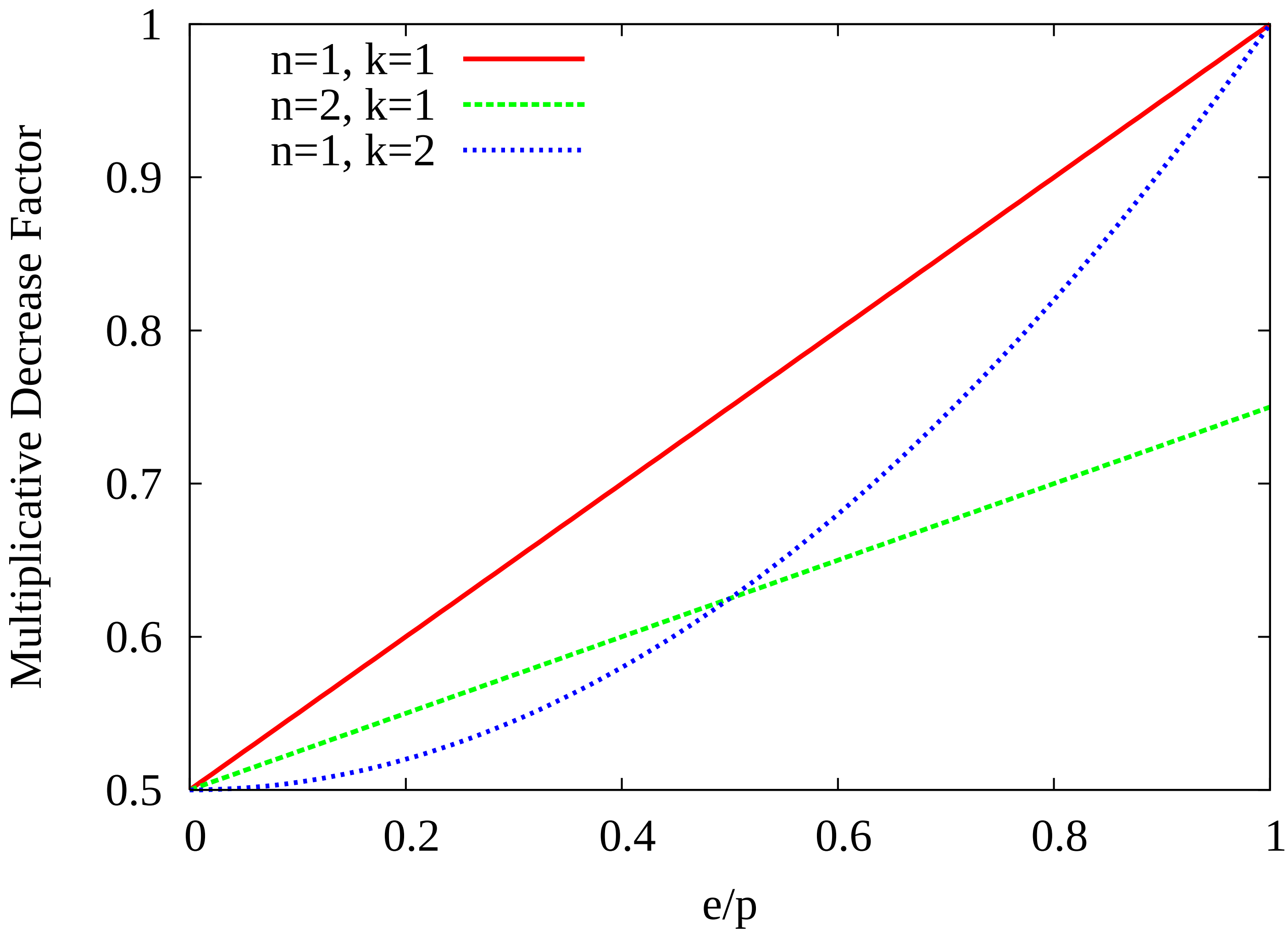
$$MDF = \frac{1+\left(\frac{e}{np}\right)^k}{2}$$

- Where n and k are shaping and bounding parameters.

- Denoted "CETEN-A"

Adjusting the Response (2) (cont.)

- Example MDF parameter sets:



Outline

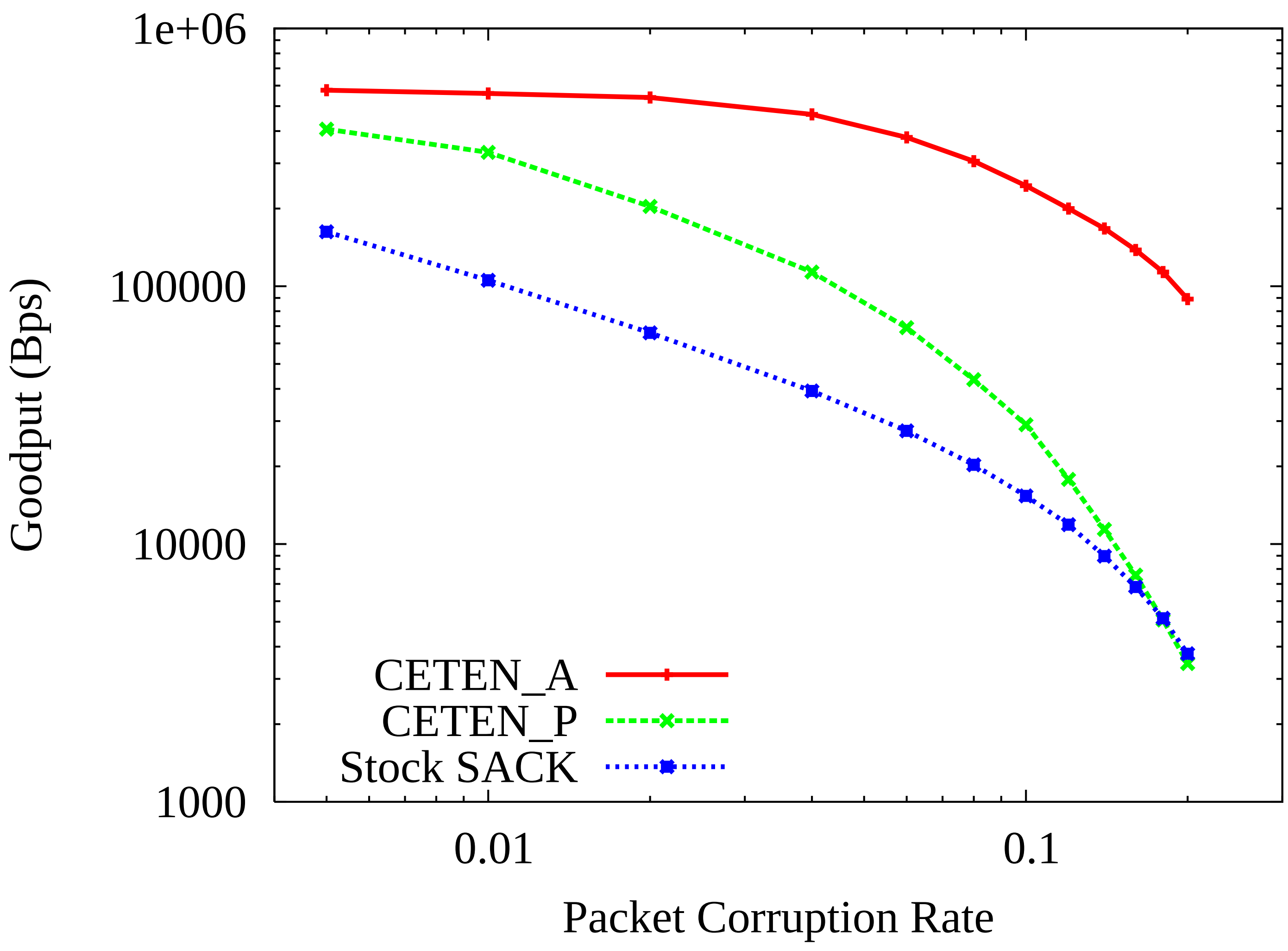
- Problem statement
- Related work
- New technique: CETEN
- Evaluation
- Future work
- Summary

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
- SACK TCP
  - MSS = 1460 bytes
  - with delayed ACKs
- Uniform loss model (!)

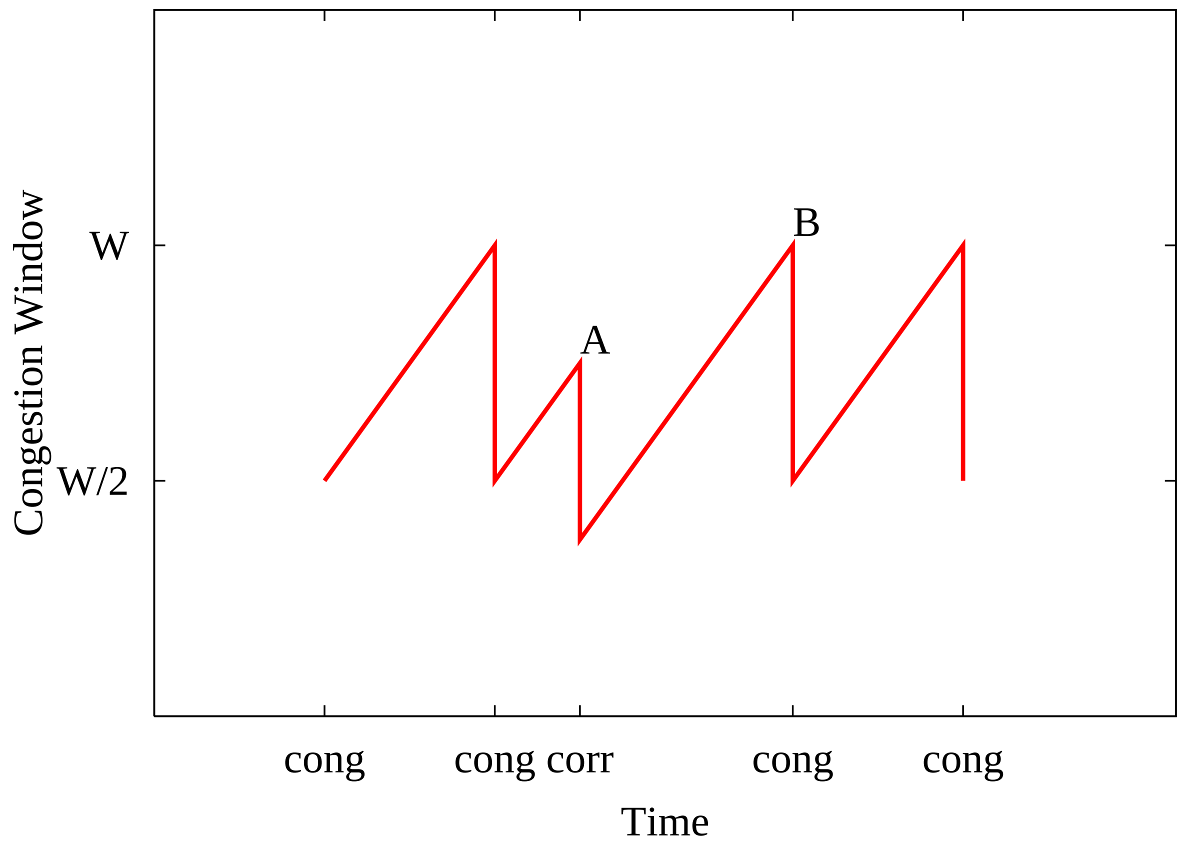
Single Flow Tests

- 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 congestion response

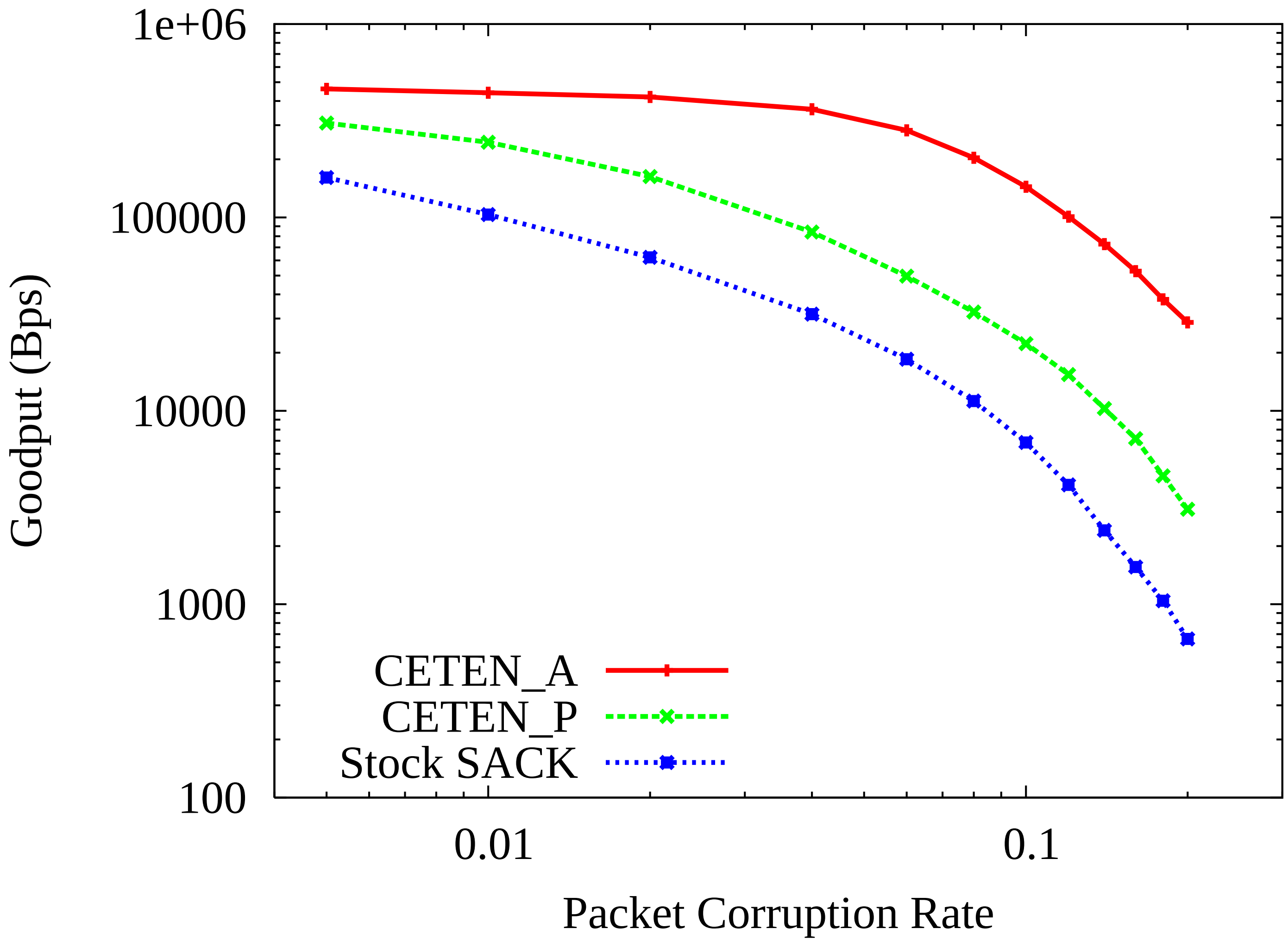


Tests with Cross Traffic

- 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.)

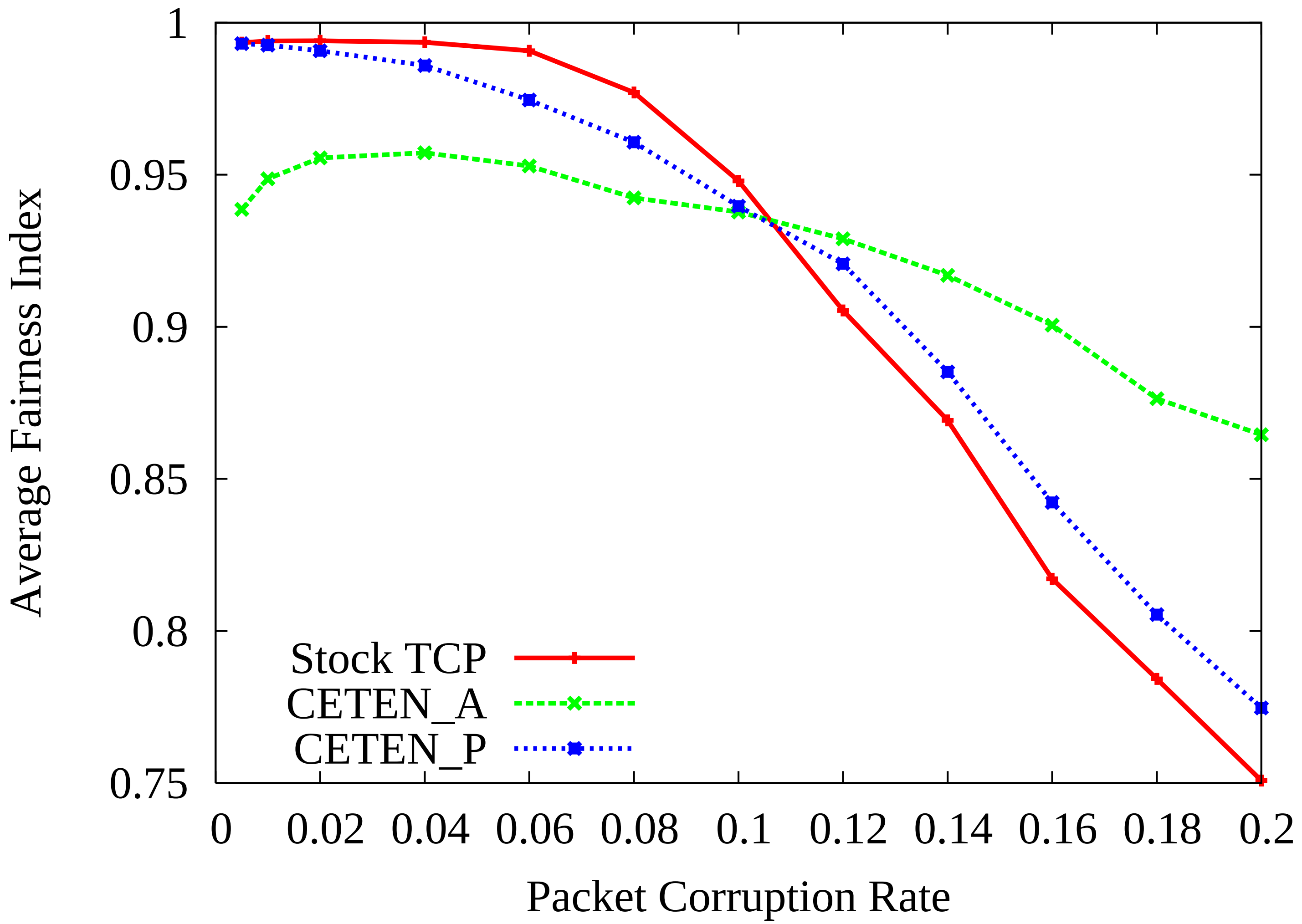
- Results from congested network:



Fairness Experiments

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

Fairness Experiments (cont.)



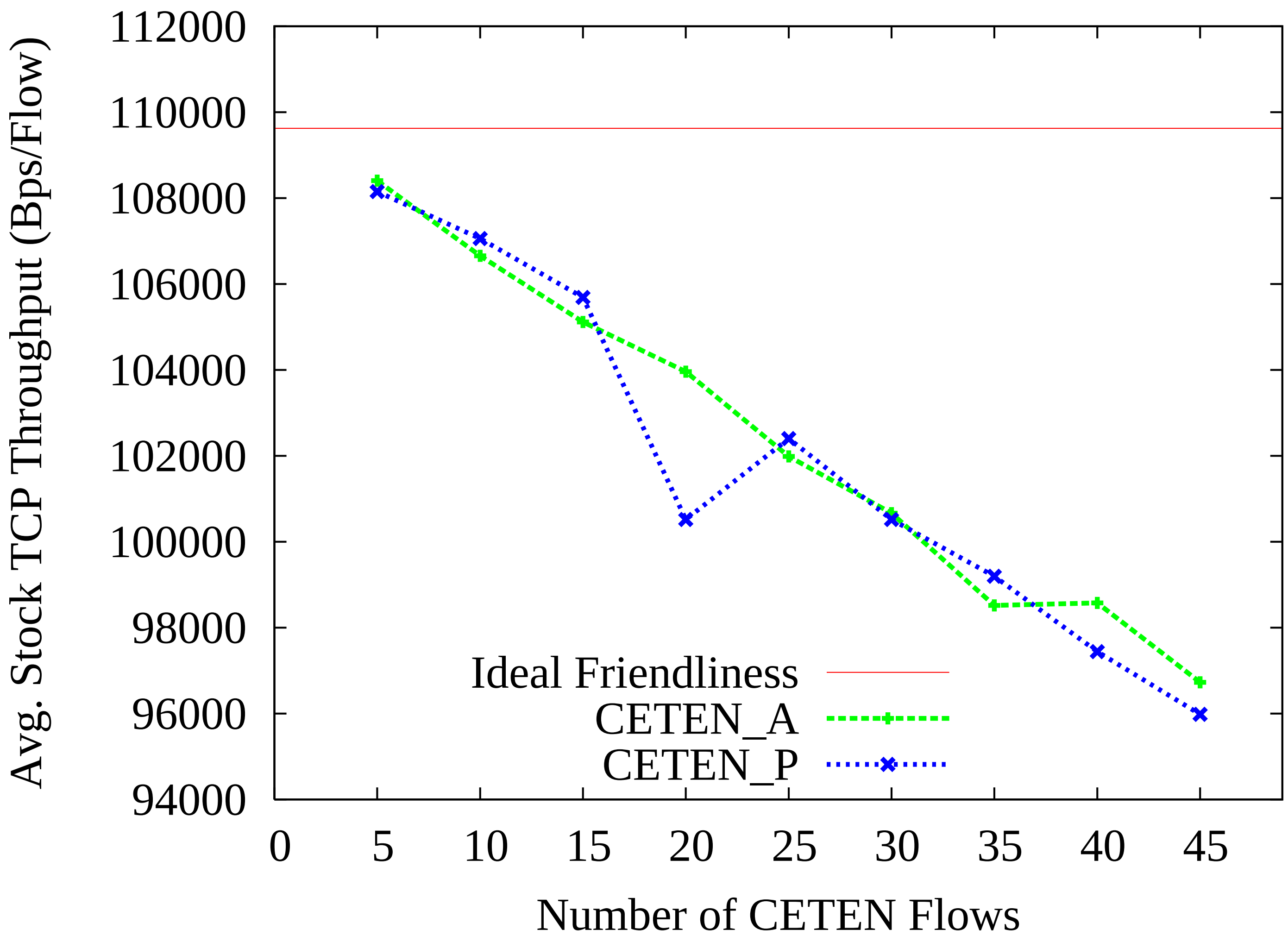
Friendliness Experiments

- How does CETEN enhanced TCP compete with stock TCP?

- Experiment
  - 50 connections
  - some stock TCP, some CETEN
    - vary the mix
  - 1% corruption loss

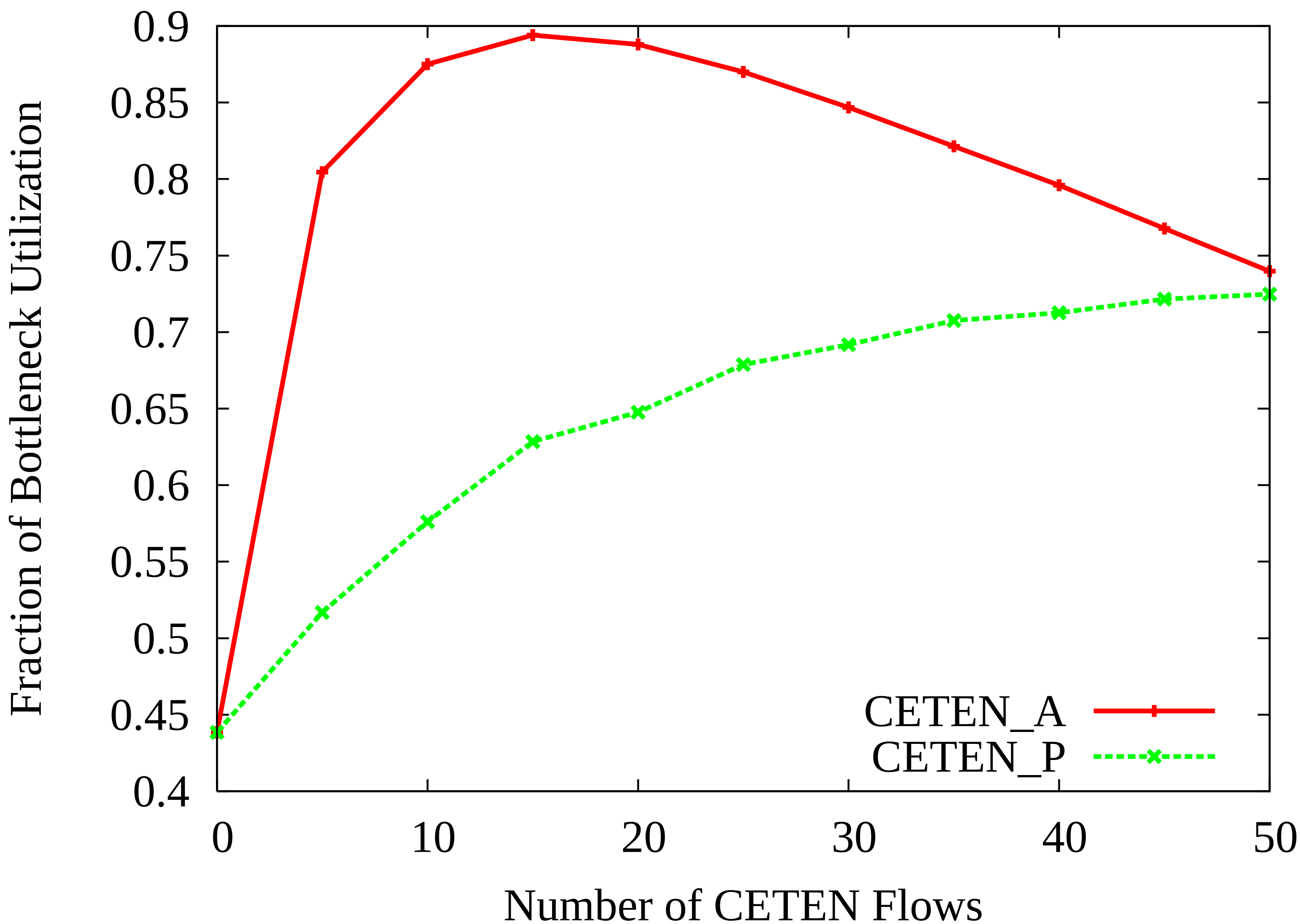
Friendliness Experiments (cont.)

- Underutilized network:



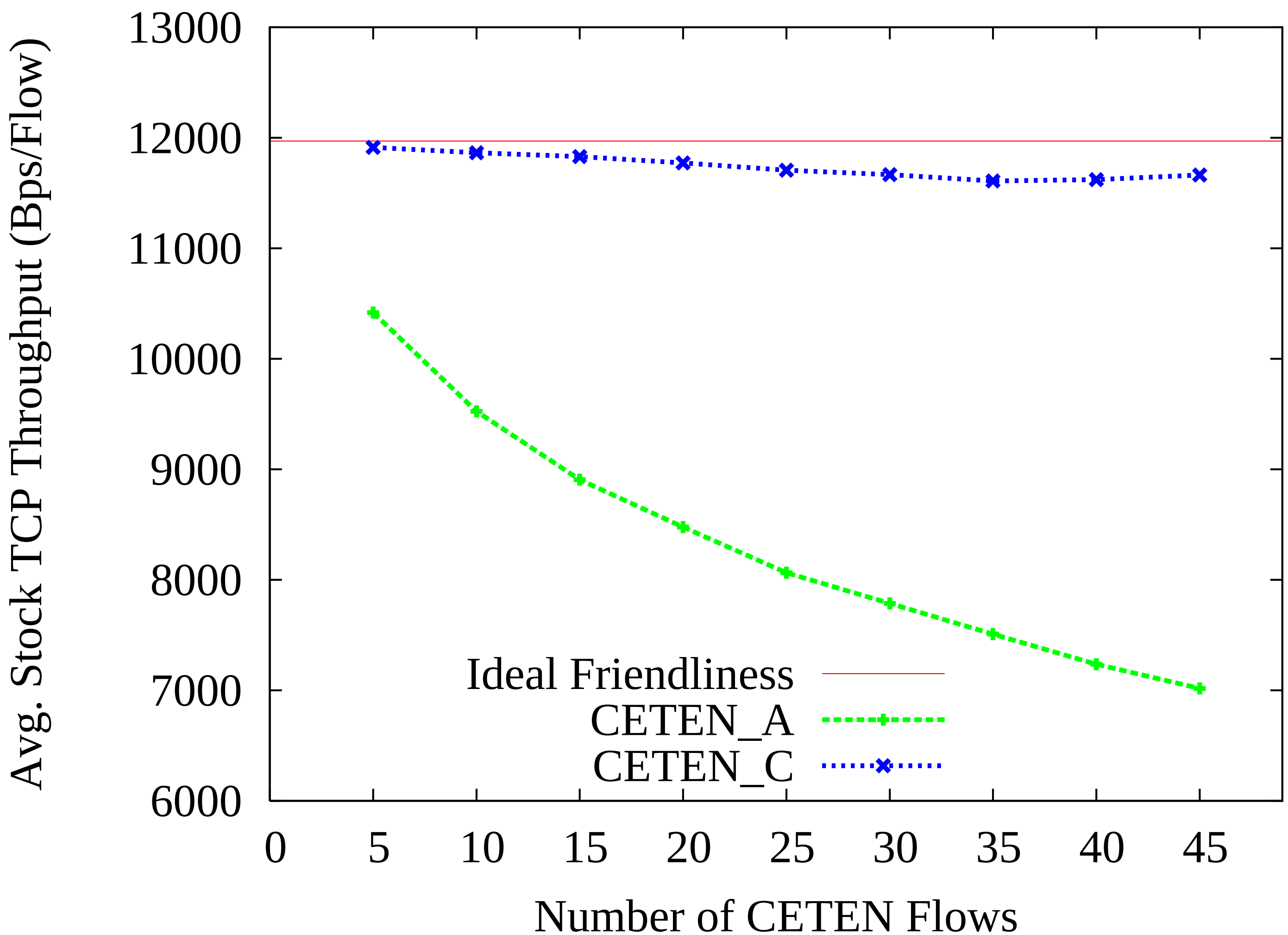
Friendliness Experiments (cont.)

- Utilization:



Friendliness Experiments (cont.)

- Saturated network:



Results Summary

- Both versions of CETEN aid performance, with ...
  - CETEN-A gaining better performance than CETEN-C
  - CETEN-C showing better friendliness properties than CETEN-A

- But, CETEN is still is a heavy-weight mechanism

Outline

- Problem statement
- Related work
- New technique: CETEN
- Evaluation
- Future work
- Summary

Future Work

- While CETEN looks and sounds promising there are a whole raft of practical issues that need to be solved.
- E.g., How do routers average corruption rates? Over what timescales?
- E.g., Should the end host average the "e" reports?
- E.g., How often should the end host request an "e" report?
- E.g., Can routers manipulate packets to include "e" in an efficient enough way? At what speeds?

Future Work (cont.)

- E.g., How do we encode these probabilities? Where?
- E.g., What does CETEN performance look like under a more realistic corruption loss model?
- E.g., How do we prevent lying receivers from gaming the sender's congestion control for their own benefit?
- E.g., How do we prevent DoS attacks on routers that involve making them spend more cycles on every packet than they otherwise would?

Future Work (cont.)

- More speculatively:
  - what else could we use the corruption sample for?
  - routing?

- Broader themes:
  - how do we communicate among layers?
  - what information should be communicated to the end point?

More Information

- Me:
  `mallman@icir.org`
  `http://www.icir.org/mallman/`

- Project web page:
  `http://www.icir.org/mallman/research/proj-eten.html`

- Questions? Comments? Concerns?