Improving Performance of Internet Protocols Over Wireless Networks

Mark Allman
ICSI Center for Internet Research
mallman@icir.org

Kent State University
November 5, 2003

"Here come Johnny with the power and the glory; Backbeat the talkin’ blues"
Acknowledgments

• Collaborators
  ▶ Wesley Eddy (Ohio Univ.)
  ▶ Shawn Ostermann (Ohio Univ.)

• Support
  ▶ BBN Technologies
  ▶ NASA Glenn Research Center
Outline

• Background
• The problem
• Previously tried mitigations
• New technique: CETEN
• Preliminary evaluation
• Future work
• Summary

• Hopefully there is a little something in here for everyone:
  ▶ theory, practice, math, measurements, simulations, plots, architecture, color pictures & hyperbole

• Please ask questions as they come up.
The transport layer of the network stack is charged with delivering data between applications on end systems.

TCP is the most heavily used transport protocol on the Internet.

- Other transports follow TCP’s basic controls and so we expect our work to cover them as well (e.g., SCTP).

TCP happens to also provide reliable in-order delivery of data bytes.

TCP is a sliding window protocol that originally used a static sized window so the receiver could control its resources.
TCP worked well until the mid-80s when the Internet suffered from congestion collapse.

- The state when the network is highly utilized carrying a ton of traffic, but very little useful work is getting accomplished.

- Van Jacobson added a set of congestion control and avoidance techniques to TCP to combat congestion collapse.

- The key observation is that packet loss is a pretty good implicit signal that congestion is occurring somewhere in the network path.
Background (cont.)

![Diagram of a router with traffic flow](image-url)
The solution: when signals of congestion arrive (packet loss, or later explicit signals of congestion) TCP reduces the sending rate (by half).

In the absence of a congestion signal TCP increases the sending rate (linearly) in an effort to detect newly available capacity.

Additive Increase Multiplicative Decrease (AIMD)

We control the sending rate with a congestion window.
• Steady state TCP:
Outline

• Background
• The problem
• Previously tried mitigations
• New technique: CETEN
• Preliminary evaluation
• Future work
• Summary
Problems with CC

• What could be wrong with TCP’s AIMD-based congestion control?
  ▶ well ...

• The premise of Jacobson’s work is that nearly all packet loss is caused by resource contention in routers.
  ▶ Which was true.
  ▶ And, is still likely true.

• 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.

• Fundamental Problem: TCP has no way to derive the cause of a packet loss.
Problems with CC (cont.)

• Steady state with non-congestion-loss:

![Graph showing congestion window changes over time]

- Congestion Window
- Time
- W
- W/2

States:
- cong
- cong corr
- cong
- cong
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.
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.
• Model TCP performance:

RTT = 0.5 sec; MSS = 1460 bytes
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:

\[
\begin{align*}
\text{Throughput (B/s)} & \quad \text{Loss Rate} \\
100000 & \quad 0.001 \\
10000 & \quad 0.01 \\
1000 & \quad 0.1 \\
\end{align*}
\]

RTT = 0.5 sec; MSS = 1460 bytes; e/p = 0.75

Stock Response  
Ideal Response
Outline

- Background
- The problem
- Previously tried mitigations
- New technique: CETEN
- Preliminary evaluation
- Future work
- Summary
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

Allman  19
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
Local Repair (cont.)

- ARQ:

- FEC:
Outline

- Background
- The problem
- Previously tried mitigations
- **New technique: CETEN**
- Preliminary evaluation
- Future work
- Summary
CETEN

- Cumulative Explicit Transport Error Notification
  - Originally outlined by Krishnan, Sterbenz, Partridge, Allman
    - BBN tech report
  - Refined by Eddy, Ostermann, Allman
    - In progress

- 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 under submission.
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:

![Graph showing CDF and percent error for 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.

• Mechanism 1:
  ▶ The TCP sender polls the router (with a TTL-limited request) for the current error rate on their connected link.
  ▶ Pros: no on-the-wire protocol changes
  ▶ Cons: extra network traffic, extra control messages for firewalls to nuke, unreliable
• Mechanism 2:
  ▶ The router probabilistically sends an "e"-report to the packet source for a given random packet that is being forwarded.

  ▶ Pros: no on-the-wire protocol changes

  ▶ Cons: extra network traffic (but more controlled than mechanism 1), extra control messages, unreliable
Estimating "e" (cont.)

- Mechanism 3:
  - 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 --- this probability is echoed to the TCP sender in ACKs.
  - Pros: no extra control traffic, more reliable
  - Cons: we have to change (or extend) the network or transport layer protocol
• We chose mechanism 3.

• 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:

\[ MDF = \frac{1 + \left( \frac{e}{n_p} \right)^k}{2} \]

• Where \( n \) and \( k \) are shaping and bounding parameters.

• Denoted "CETEN-A"
• Example MDF parameter sets:

![Graph showing multiplicative decrease factor vs. e/p for different n and k values.]

- $n=1, k=1$
- $n=2, k=1$
- $n=1, k=2$
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
Outline

• Background
• The problem
• Previously tried mitigations
• New technique: CETEN
• Preliminary 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 (!)
• One end-to-end TCP flow
• 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
  - 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
Outline

• Background
• The problem
• Previously tried mitigations
• New technique: CETEN
• Preliminary evaluation
• Future work
• Summary
• 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 friendly is CETEN?

• 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.)

• The bigger picture:
  ▶ How much information should the network be expected to provide to the end hosts?

    ▪ e.g., for CETEN?
    ▪ e.g., for Quick Start?
    ▪ e.g., for XCP?

    ▪ e.g., ??? (queueing delay, reordering, etc.)

• When does the network become "too smart"?
• When does the amount of information requested by the end hosts become too much of a burden?
• Or, does it?
Outline

• Background
• The problem
• Previously tried mitigations
• New technique: CETEN
• Preliminary evaluation
• Future work
• Summary
Summary

• CETEN is an interesting and potentially useful technique for improving performance for a certain class of network traffic

  ▶ E.g., the increasing amount of wireless traffic

• There are many issues that still need to be worked out. This is still very much research.

  ▶ (much grist for the grad student energy mill!)
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?