Networking at the Speed of Light

Brighten Godfrey • UIUC

Wireless @ MIT Seminar
November 6, 2014
“How long until our pigeon system rivals those of the continental powers?”

– British periodical *The Nineteenth Century*, 1899
Can you perceive…

1 second?

500 milliseconds?

100 milliseconds?

10 milliseconds?

spoken syllable
~150 - 200 ms
Can you perceive…

1 second?

500 milliseconds?

100 milliseconds?

10 milliseconds?

Hiromi Uehara
“Kung Fu World Champion”
Can you perceive…

1 second?

500 milliseconds?

100 milliseconds?

10 milliseconds?

spoken syllable
~150 - 200 ms

Hiromi Uehara
88 ms per note
<table>
<thead>
<tr>
<th></th>
<th>Delay</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>+100ms</td>
<td>-1% revenue</td>
</tr>
<tr>
<td>Bing</td>
<td>+500ms</td>
<td>-1.2% revenue</td>
</tr>
<tr>
<td>Google</td>
<td>+400ms</td>
<td>-0.7% searches</td>
</tr>
</tbody>
</table>
Every millisecond matters.
A toy experiment

$ wget theoatmeal.com
  • 430ms (fetch size 20KB)

$ ping 208.70.160.53
  • 55ms

Champaign-Portland roundtrip distance
  • 5700 kilometers
  • 19ms at the speed of light in vacuum

23x inflation over c-latency!
The Internet is too slow

CDF

Inflation over c-latency
The Internet is too slow

CDF

Inflation over c-latency

34x
A speed-of-light Internet

a Grand Challenge for networking
A Qualitative Change

New interactive applications
Hit it!!
A Qualitative Change

New interactive applications

Instant response
The limits of human visual perception
A Qualitative Change

New interactive applications

Instant response

Superlinear community growth
How would a 10x decrease in latency affect community size?
How would a 10x decrease in latency affect community size?
A speed-of-light Internet

*a Grand Challenge for networking*

Is it just a pipe dream?
Why are we so far from the speed of light?
# XFINITY from Comcast vs. Verizon FiOS

<table>
<thead>
<tr>
<th></th>
<th>XFINITY</th>
<th>FiOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fastest Internet Speed</td>
<td><strong>505 Mbps</strong></td>
<td><strong>500 Mbps</strong></td>
</tr>
</tbody>
</table>

![Comcast Speed Test](comcast.com)

![Verizon Speed Test](verizon.com)

![Speedtest](speedtest.net)
Measuring latency

- 400+ PlanetLab nodes as clients
- Fetch (using cURL) thousands of Web pages each
  - 32 KB of the HTML of the landing pages, with latency broken down into components
  - 30 pings and a traceroute to each Web server
  - Geolocation of source, routers, and destination
- End-user Akamai measurements in the paper
Latency inflation breakdown

CDF

Inflation over c-latency to the Web server
Latency inflation breakdown

![Graph showing CDF of inflation over c-latency to the Web server with a point at 5.4x.]
Latency inflation breakdown

![Graph showing latency inflation breakdown with Router-path and DNS regions. The graph indicates an inflation factor of 2.3x over c-latency to the Web server.](image-url)
Latency inflation breakdown

![Graph showing CDF of inflation over c-latency to the Web server with Router-path, Min ping, and DNS highlighted. The graph indicates a 3.2x inflation at specific points.]
Router path vs. ping
Router path vs. ping
Router path vs. ping
Internet2 Combined Infrastructure Topology
Portfolio of network infrastructure and services across the Internet2 footprint

Inflation in fiber distances

Ground truth data on fiber lengths!
Inflation in fiber distances
Inflation in fiber distances

CDF

Inflation over geo-distance

Road
Inflation in fiber distances

Inflation over geo-distance

CDF

Inflation over geo-distance

Road

Fiber
Inflation in fiber distances

**Internet 2**

**GÉANT**

**ESnet***

* Dhruv Diddi (UIUC) helped process ESnet data
Latency inflation breakdown

(See bufferbloat discussion in the paper)
Latency inflation breakdown

![Graph showing Latency inflation breakdown]

- **CDF**
- **Inflation over c-latency to the Web server**
- **Router-path**
- **Min ping**
- **TCP handshake**
- **DNS**
- **TCP transfer**

- **8.7x**

The graph illustrates the breakdown of latency inflation over c-latency to the Web server, with specific components highlighted:

- **Router-path**
- **Min ping**
- **TCP handshake**
- **DNS**
- **TCP transfer**
Latency inflation breakdown

CDF

Inflation over c-latency

Router-path  Min ping  TCP handshake  DNS  TCP transfer

Total time

24.6x
Latency inflation breakdown

Total time (24.6x) = DNS resolution (5.4x) + TCP handshake (3.2x) + TCP transfer (8.7x)
Latency inflation breakdown

Total time (24.6x) = DNS resolution (5.4x) + TCP handshake (3.2x) + Request (3.2x) + TCP transfer (8.7x)

Inflation at lower layers affects everything!
Absent inflation in the lower layers ...

\[
\text{Total time (24.6x)} = \text{DNS resolution (5.4x)} + \text{TCP handshake (3.2x)} + \text{Request (3.2x)} + \text{TCP transfer (8.7x)}
\]

(Normalized by the 3.2x inflation in minimum ping time)
Research directions

Measurement

Parallel low-latency Internet infrastructure

Attack transport layer inflation
Research directions

Measurement

Parallel low-latency Internet infrastructure

Attack transport layer inflation
Unstable, RTT Unfair, Bufferbloat, Crash on Changing Networks, .......

Point Solutions
+
Performance
Far from Optimal
TCP **fails** to achieve consistent high performance

Why it is so hard?
## What is TCP

<table>
<thead>
<tr>
<th>Event</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reno</td>
<td>1 pkt loss       cwnd/2</td>
</tr>
<tr>
<td>CUBIC</td>
<td>Time pass 1ms    cwnd+1</td>
</tr>
<tr>
<td>FAST</td>
<td>RTT increase x%  Reduce cwnd to f(x)%</td>
</tr>
<tr>
<td>Scalable</td>
<td>ACK             cwnd+1</td>
</tr>
<tr>
<td>HTCP</td>
<td>100 ACK         cwnd+f(cwnd)/cwnd</td>
</tr>
</tbody>
</table>
TCP is Hardwired Mapping

<table>
<thead>
<tr>
<th>Event</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reno 1 pkt loss</td>
<td>cwnd/2</td>
</tr>
<tr>
<td>CUBIC Time pass 1ms</td>
<td>cwnd+1</td>
</tr>
<tr>
<td>FAST RTT increase x%</td>
<td>Reduce cwnd to f(x)%</td>
</tr>
<tr>
<td>Scalable ACK</td>
<td>cwnd+1</td>
</tr>
<tr>
<td>HTCP 100 ACK</td>
<td>cwnd+f(cwnd)/cwnd</td>
</tr>
</tbody>
</table>

Packet Level Events:

Action 1 → Network → Hardwired → Action 2
Hardwired Mapping is Brittle

Action 1 → Real Network → Assumed Network → Hardwired → Action 2

Packet Level Events

Performance
Flow f sends at R

Packet Loss
Flow f sends at R

Network Conditions

- f causes most congestion
- shallow buffer overflow
- other high rate flow causing congestion
- loss is random

Event-control Mapping

- Dec R a lot
- Dec R a little
- Maintain R
- Increase R

Packet Loss
Flow f sends at R

Network Conditions:
- f causes most congestion
- shallow buffer overflow
- other high rate flow causing congestion
- loss is random

Event-control Mapping:
- Dec R a lot
- Dec R a little
- Maintain R
- Increase R

Packet Loss
Flow f sends at R

Packet Loss

Network Conditions
- f causes most congestion
- shallow buffer overflow
- other high rate flow causing congestion
- loss is random

Event-control Mapping
- Dec R a lot
- Dec R a little
- Maintain R
- Increase R
Flow f sends at R

Network Conditions
- f causes most congestion
- shallow buffer overflow
- other high rate flow causing congestion
- loss is random

Packet Loss

Event-control Mapping
- Dec R a lot
- Dec R a little
- Maintain R
- Increase R
Flow f sends at R

Network Conditions

- f causes most congestion
- shallow buffer overflow
- other high rate flow causing congestion
- loss is random

Event-control Mapping

- Dec R a lot
- Dec R a little
- Maintain R
- Increase R

Packet Loss
Flow f sends at R

Network Conditions:
- f causes most congestion
- shallow buffer overflow
- other high rate flow causing congestion
- loss is random

Event-control Mapping:
- Dec R a lot
- Dec R a little
- Maintain R
- Increase R

Packet Loss
Flow f sends at R

Network Conditions
- f causes most congestion
- shallow buffer overflow
- other high rate flow causing congestion
- loss is random

Event-control Mapping
- Dec R a lot
- Dec R a little
- Maintain R
- Increase R

Packet Loss

No event-control mapping is optimal for all network scenarios
Once hardwired, TCP ignores its actions’ impact on real performance and “jumps off the cliff”, repeatedly.
What is the right rate to send?
What is the right rate to send?
What is the right rate to send?

rate → result
What is the right rate to send?

rate \( r \) \rightarrow \text{utility} \; u

U = f(tpt, loss rate, latency, etc.)

\text{e.g.} \; U = tpt \times (1 - \text{loss rate})
What is the right rate to send?

rate \( r_1 \) \( \rightarrow \) utility \( u_1 \)

\[ U = f(tpt, \text{loss rate, latency, etc.}) \]

\[ \text{e.g. } U = \text{tpt} \times (1 - \text{loss rate}) \]

No matter how complex the network, rate \( r \) \( \rightarrow \) utility \( u \)
$u_1 > u_2$?

move to $r_1$

move to $r_2$
Observe real performance

Control based on empirical evidence

*yields*

Consistent high performance
This flow causing congestion

\[ u_1 > u_2? \]

move to 98 Mbps

move to 102 Mbps
Bandwidth 50Mbit/s
RTT 30ms
BufferSize 75KB
LossRate 0
protocol cubic
stop
Software Components

Sender

- Performance Oriented Control Module
  - (Sending Rate, Utility)

- Utility Function

Sending Rate Control

Sending Module

Packet Monitoring

Monitor Module

Performance Metrics
  - tpt., loss rate, RTT

Internet

Data

SACK

Receiver
Online Learning Control

observed utility

rate
Online learning control

- observed utility
- randomized controlled trials
- rate
Consistent High Performance

Global Commercial Internet

InterDC

Satellite Networks

Lossy Networks

RTT Unfairness

Shallow Network Buffer

Rapidly Changing Networks

IntraDC

Incast
Consistent High Performance

Global Commercial Internet
Consistent High Performance

Global Commercial Internet

1.48X Median
4X less loss

44%, 10X
Consistent High Performance

Satellite Network

WINDS System

Throughput (Mbps) vs. Bottleneck Buffersize (KB)

- PCC
- TCP Hybla
- TCP Illinois
- TCP CUBIC
- TCP New Reno

17X improvement
Consistent High Performance

Mitigate Incast

![Graph showing goodput vs. number of senders for different data sizes in PCC and TCP protocols.](image-url)
Consistent High Performance

Shallow Network Buffer

100Mbps, 30ms Link

Throughput (Mbps)

Buffer Size (KB)

90 pkts, 90% tpt

6 pkts, 90% tpt
Consistent High Performance

Rapidly Changing Networks

BW: 10-100Mbps; RTT: 10-100ms; Loss Rate: 0-1%
Change every 5 seconds
Consistent High Performance

Inter Datacenter and Dedicated High Speed Network

<table>
<thead>
<tr>
<th>Transmission Pair</th>
<th>RTT</th>
<th>PCC</th>
<th>SABUL</th>
<th>CUBIC</th>
<th>Illinois</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPO → NYSERNet</td>
<td>12.1</td>
<td>818</td>
<td>563</td>
<td>129</td>
<td>326</td>
</tr>
<tr>
<td>GPO → Missouri</td>
<td>46.5</td>
<td>624</td>
<td>531</td>
<td>80.7</td>
<td>90.1</td>
</tr>
<tr>
<td>GPO → Illinois</td>
<td>35.4</td>
<td>766</td>
<td>664</td>
<td>84.5</td>
<td>102</td>
</tr>
<tr>
<td>NYSERNet → Missouri</td>
<td>47.4</td>
<td>816</td>
<td>662</td>
<td>108</td>
<td>109</td>
</tr>
<tr>
<td>Wisconsin → Illinois</td>
<td>9.01</td>
<td>801</td>
<td>700</td>
<td>547</td>
<td>562</td>
</tr>
<tr>
<td>GPO → Wisc.</td>
<td>38.0</td>
<td>783</td>
<td>487</td>
<td>79.3</td>
<td>120</td>
</tr>
<tr>
<td>NYSERNet → Wisc.</td>
<td>38.3</td>
<td>791</td>
<td>673</td>
<td>134</td>
<td>134</td>
</tr>
<tr>
<td>Missouri → Wisc.</td>
<td>20.9</td>
<td>807</td>
<td>698</td>
<td>259</td>
<td>262</td>
</tr>
<tr>
<td>NYSERNet → Illinois</td>
<td>36.1</td>
<td>808</td>
<td>674</td>
<td>141</td>
<td>141</td>
</tr>
</tbody>
</table>
Consistent High Performance

Lossy Networks

100Mbps, 30ms, varying loss rate
TCP’s throughput collapse 10X with 0.5% loss
Consistent High Performance

RTT Unfairness

100Mbps, 10ms short RTT flow
Varying RTT of long RTT flow
Selfish, but Fair

Selfish != Melt down your network
Selfish != Send as fast as it can
Selfish, but Fair

Competing Senders form a non-cooperative game
Nash Equilibrium depends on Utility Function

We want the utility function to:

- Express a generic data transmission objective
- Unique and efficient Nash equilibrium

\[ u_i(x) = T_i(x) \times \text{Sigmoid}(L(x) - 0.05) - x_i \times L(x) \]

\[ L(x) = \max\{0, 1 - \frac{C}{\sum_j x_j}\} \text{ is the loss rate,} \]
\[ T_i(x) = x_i(1 - L(x)) \text{ is sender } i\text{'s throughput, and} \]
Different Utility Functions

Tpt/Latency as the goal?
Excessive Loss Resilience?

FQ environment?
FIFO environment?

Some Results
Significant Area of Future Work
Fig. 5. Additive Increase/Multiplicative Decrease converges to the optimal point.
Dynamic Behavior and Fairness

TCP

PCC
PCC has better reactivity-stability trade-off than TCP
Rate control based on empirically observed performance yields

- Consistent high performance
- Better stability-reactiveness tradeoff than TCP
- Flexible performance objectives
- Relatively easy and safe to deploy
LIGHTSPEED IS TOO SLOW. WE’LL HAVE TO GO RIGHT TO LUDICROUS SPEED.
Backup Slides
Deployment

• Easy to Deploy
  No hardwire support, packet header, protocol change needed

• Deploy Scenarios
  • Isolated from TCP flows
    CDN backbone, Inter-data center, dedicated scientific nw
  • Deploy without isolation
    more TCP friendly than common selfish practices
  • Enable flexibility with per-flow FQ
Deployability: TCP Friendliness

PCC is not more unfriendly than common selfish practices

Diagram showing the relative unfriendliness ratio of PCC compared to TCP-Selfish for different network conditions.
Deployability: Short Flow FCT

*PCC does not fundamentally harm FCT*