Rethinking Data Transport for a Truly Mobile World

Keith Winstead, Anirudh Sivaraman, Hari Balakrishnan

Wireless@MIT
M.I.T. Computer Science and Artificial Intelligence Lab

January 2013
http://wireless.csail.mit.edu
The Era of *Truly Mobile* Computing

- The current (and future) Internet is mobile *first*
- Mobile & wireless networks are very different from the traditional Internet
- Current protocols and applications perform poorly: Should we fix the end points or fix the network?
5.5B mobile phones (> 1B with “broadband” access)
Exceeds people with access to shoes, electricity, toothbrushes
In Q2, 2012:
- Population growth: 0.25 million
- Smartphone growth: 1.7 million
Many Problems

- Can’t roam across IP addresses (WiFi ↔ cellular)
- Can’t suspend/resume
- Highly variable rates
- Highly variable and often high delays
Highly Variable Rates

Verizon LTE uplink throughput

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan
Rethinking Data Transport for a Truly Mobile World
Awful Delays: The *Too-Reliable* Network

Verizon LTE TCP RTT (Cambridge, MA)

AT&T Wireless on iPhone 3G
- \( \mu: 1697.2 \text{ ms} \)
- \( \text{stddev}: 2346.5 \text{ ms} \)
- \( \text{min}: 155.6 \text{ ms} \)
- \( \text{max: } 12126.6 \text{ ms} \)

On the Acela Amtrak train (BOS \( \rightarrow \) NY)
New transport protocols to adapt applications to the vagaries of modern wireless and mobile networks

Mosh: Graceful mobility

Sprout: Flow control for cellular networks
Motivation for Mosh: Frustration with SSH

- Cannot roam
  - ... across Wi-Fi networks
  - ... from Wi-Fi to cellular or vice versa
- Can’t sleep and wake up (usually): home ↔ work fails
  - ... TCP disconnects if unacked data for “a while”
  - ....... or if peer away for “too long”
- TCP (and SSH) responds poorly to packet loss
- All UI requires round-trip to server
Octet Stream: Wrong Abstraction

- Client wants **latest screen**
- After interruption, don’t want to replay megabytes of “old stuff”
- But SSH runs over TCP, which sends everything in order
- TCP fills buffers, so Control-C takes forever
What We Built

1. Transport protocol for low-latency **object synchronization**
   - with **fast, secure roaming**
   - and its own flow control
   - through suspend/resume
   - over lossy network paths

2. Supports **rolling latency compensation** on client side

3. **Mobile shell application** to replace SSH:
   runs over object sync protocol with predictive local echo

**Mosh project:** [http://mosh.mit.edu](http://mosh.mit.edu)
K. Wintein, H. Balakrishnan, USENIX Annual Conf., June 2012
—, article in *;login:*, August 2012
State Synchronization Protocol (SSP)

- Runs over UDP
- Instead of sending *octet streams*, synchronize *objects*
- Object must support:
  - `diff`: make vector from state $A \rightarrow B$
  - `patch`: apply vector to $A$ to make $B$
- Object implementation, **not protocol**, defines synchronization semantics.
- Flow control: two object operations per RTT (one in flight, another in process)
Secure Quick Roaming

- Protected by Offset Codebook (OCB) mode of AES-128 (Krovetz 2011; Rogoway’s scheme)
  - Integrity and confidentiality with one key
- Each packet is an idempotent operation with incrementing sequence number
- Roaming is easy:
  - Source address of latest authentic packet from client ⇒ server’s new target
  - Client does not even know it has roamed
  - Server can’t move
Loss Resilience: Prophylactic Retransmissions

SSP has options in choosing which diff to send:

1. Last ack was for state #3. Then state changes to #4.
2. Host sends diff from 3 → 4.
3. Object changes to state #5.
4. If no timeout yet, make next diff as 4 → 5.
5. **Also** make diff from 3 → 5: the *prophylactic retransmission*.
6. If P·Retransmission is shorter or not much longer, send instead.
Rolling Predictions

▶ Client runs predictive model of server UI.
▶ Make predictions in *epochs*.
▶ If any from epoch $n$ is confirmed, show whole epoch.
▶ If prediction from epoch $n$ is wrong, hide that epoch.
▶ If user does something difficult to handle, become tentative: *increment epoch*. 
Demo

* What should it look like?
** Ideas
*** Boring free software Web site...
*** Old-timey newspaper: "Amazing remote shell program sweeps nation!!!!"
*** Make it look like a fake startup company. <-- Let's go with this.
* Benefits of Mosh
** Roam across Wi-Fi networks or to cell without dropping connection.
** More pleasant to type -- intelligent local echo is instant.
** No need to be superuser to install.
** Mosh doesn't fill up buffers, so Ctrl-C works quickly on runaways.
** Designed from scratch for Unicode; fixes bugs in SSH, other terminals.

---

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan

Rethinking Data Transport for a Truly Mobile World
Deployment

- Mac OS X: in MacPorts, Homebrew, FreeBSD ports
- In most Linux distros (Nov. 2012 cover of Linux Magazine)
  - Debian, Ubuntu, Fedora, Gentoo, Arch, Slackware
  - Red Hat, CentOS, Oracle Linux
- iOS (iSSH), Android (expt)
- Works on Windows Cygwin, Solaris
- 300,000+ page views, ≈100,000 known downloads, 1,600+ VCS followers
- Top repo of the month on GitHub
Performance

Keystroke response times CDF over Sprint EV-DO (3G) network

- **Mosh**
  - Median: 5 ms
  - Mean: 173 ms

- **SSH**
  - Median: 503 ms
  - Mean: 515 ms
Reviews

- USENIX accepted the paper, grudgingly...
  - “Not a research paper”
  - “ISO 2022 locking escape sequences oh flying spaghetti monster please kill me now”
- The Twitterati are far kinder...
  - “After 48 hours ... I can say it rocks!” — paulrouget
  - “Mosh is the only thing that makes internet on Greyhound useful whatsoever.” — standaloneSA
  - “I’m on a train, tethering. SSH keeps failing hard. Installed Mosh. Instant win.” — maciejmalecki
  - “MOSH: mobile shell that’s suitable for real-world connectivity; example, accessing my Euro boxes from India” — danparsons
  - “Mosh is world-changing.” — mftb
  - “Mosh gives me faith in humanity.” — runa
  - “Go get mosh: NOW” — echorand
SSP for all Mobile Apps?

Many Web and native apps have trouble with roaming and intermittent connectivity.

Screenshot from July 5!

- Gmail app, Skype, Google Chat, quora.com and other web sites, Google Voice, Twitter, ...
- Problems may be expressed as state sync problems and handled well using SSP
Mosh: Graceful mobility

Sprout: Flow control for cellular networks
Mobile Wireless Networks are Variable

Verizon LTE uplink throughput

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan

Rethinking Data Transport for a Truly Mobile World
Videoconferencing over LTE

- We measured cellular networks while driving:
  - Verizon LTE
  - Verizon 3G (1xEV-DO)
  - AT&T LTE
  - T-Mobile 3G (UMTS)

- Then ran apps across emulated network:
  - Skype (Windows 7)
  - Google Hangout (Chrome on Windows 7)
  - Apple Facetime (OS X)
Why is Wireless Videoconferencing So Bad?

- Applications/transport protocols **react** to congestion signals
- Today, these are
  - Packet loss
  - Increase in round-trip time
- This feedback comes too late to help
- The killer: **self-inflicted queueing delay**
- Any overshoot means a queue filling up with packets
- No network model, no prediction ⇒ Poor adaptation
Characterizing Performance

![Graph showing throughput vs. self-inflicted delay]

Better

+Skype

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan

Rethinking Data Transport for a Truly Mobile World
Sprout’s Goal

- As much throughput as possible, with
- **bounded risk of delay** \( > D \) (e.g., \( D = 100 \text{ ms} \)).

**Alfalfa project** (http://alfalfa.mit.edu)
“How to Beat Skype, Hangout, and Facetime”
K. Winstein, A. Sivaraman, H. Balakrishnan, NSDI 2013
Bounded Risk of Delay

- **Infer** rate from *interarrival distribution*
- **Predict** future link rate and convey prediction to sender
  - Don’t wait for congestion
- **Control:** Send as fast as possible, but require:
  - 95% probability all packets will arrive within 100 ms
Infer Rate from Interarrival Distribution

A flicker noise process

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan
Predict Future Link Rate

- Count packets in every 20 ms tick
- Use Bayesian updating to infer (uncertain) link speed
- Make a cautious forecast
Network Model

Sender → Queue → Receiver

Poisson process drains queue

Rate $\lambda$ controls Poisson process

Brownian motion of $\sigma \sqrt{t}$ varies $\lambda$

If in an outage, $\lambda_z$ is escape rate.
Bayesian Update

- Discrete set of possible rates, $\lambda$ (e.g., 0 to 1000 packets/s)
- Initially, each $\lambda$ is equi-probable
- Each tick ($\tau$ seconds), if we receive $k$ bytes, run update step:

$$
\mathbb{P}_{new}(\lambda = x) \leftarrow \frac{\mathbb{P}_{old}(\lambda = x) \cdot \frac{(x\tau)^k}{k!} e^{-(x\tau)}}{Z},
$$

where $Z$ ensures that the probabilities sum to 1.
The Cautious Forecast

- Receiver has “cloud” of current link speeds
- For eight ticks in the future:
  - Predict future link rate by simulating Brownian motion of rates
  - Find 5th percentile of cumulative packets
- Send forecast to sender on ACKs
- Most of the math is pre-computed
Limitations

- Stochastic model has not been tuned
- Designed for cellular link with per-user queue
- If other users can cause you big delay, can’t solve end-to-end
Verizon LTE uplink

Throughput (kbps) vs. Self-inflicted delay (ms)

- Better
- Skype

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan

Rethinking Data Transport for a Truly Mobile World
Verizon LTE uplink

Throughput (kbps) vs. Self-inflicted delay (ms)

- Skype
- Facetime
- Hangout

Better direction for improved performance.
Verizon LTE uplink

- Throughput (kbps)
- Self-inflicted delay (ms)

- Linux TCP
- TCP Vegas
- LEDBAT
- Compound TCP
- Skype
- Facetime
- Hangout

Better
Verizon LTE uplink

Throughput (kbps) vs. Self-inflicted delay (ms)

- Mosh: Graceful mobility
- Sprout: Flow control for cellular networks
- Linux TCP
- Skype
- Facetime
- Hangout
- LEGBAT
- TCP Vegas
- Compound TCP
- Better

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan
Rethinking Data Transport for a Truly Mobile World
Verizon LTE downlink

Throughput (kbps) vs. Self-inflicted delay (ms)

- Linux TCP
- Sprout
- LEDBAT
- TCP Vegas
- Compound TCP
- Facetime
- Skype
- Hangout
AT&T LTE uplink

Throughput (kbps) vs. Self-inflicted delay (ms)

- Linux TCP
- LEDBAT
- Compound TCP
- TCP Vegas
- Skype
- Hangout
- Facetime
- Sprout

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan
Wireless@MIT M.I.T. Computer Science and Artificial Intelligence Lab
Rethinking Data Transport for a Truly Mobile World
Mosh: Graceful mobility
Sprout: Flow control for cellular networks

AT&T LTE downlink

<table>
<thead>
<tr>
<th>Throughput (kbps)</th>
<th>Self-inflicted delay (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Linux TCP
- Facetime
- Skype
- Hangout
- Compound TCP
- LEDBAT
- TCP Vegas
- Sprout

Keith Winstein, Anirudh Sivaraman, Hari Balakrishnan
Wireless@MIT M.I.T. Computer Science and Artificial Intelligence Lab
Rethinking Data Transport for a Truly Mobile World
## Overall Results from Four Networks

<table>
<thead>
<tr>
<th>Sprout vs.</th>
<th>Avg. speedup</th>
<th>Delay reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skype</td>
<td>2.2×</td>
<td>7.9×</td>
</tr>
<tr>
<td>Hangout</td>
<td>4.4×</td>
<td>7.2×</td>
</tr>
<tr>
<td>Facetime</td>
<td>1.9×</td>
<td>8.7×</td>
</tr>
<tr>
<td>Compound</td>
<td>1.3×</td>
<td>4.8×</td>
</tr>
<tr>
<td>TCP Vegas</td>
<td>1.1×</td>
<td>2.1×</td>
</tr>
<tr>
<td>LEDBAT</td>
<td>Same</td>
<td>2.8×</td>
</tr>
<tr>
<td>Linux TCP (CUBIC)</td>
<td>1.1×</td>
<td>79×</td>
</tr>
</tbody>
</table>
Competes with AQM even though end-to-end

![Graph showing utilization versus self-inflicted delay for different protocols: Linux TCP, Linux TCP-over-CoDel, Sprout, and Sprout-EWMA. The graph compares the performance of these protocols in terms of their ability to manage network congestion and optimize data transport.]
On-going & Future Work

- Sprout relay for easy deployment
- Alfalfa: videoconferencing utility using Sprout
- Public contest for best predictor
  - Incorporate winning ideas into Sprout
  - Open courseware → Open research
Sprout Summary: High Throughput, Tolerable Delay

- **Infer** link speed from interarrival distribution
- **Predict** future link speed
- **Control** risk of large delay
- Yields 2–4× throughput of Skype, Facetime, Hangout
- Achieves 7–9× reduction in self-inflicted delay
- Matches active queue management *without router changes*
Conclusion

- The current (and future) Internet is mobile *first*
- Mobile & wireless networks are very different from the traditional Internet
- Current protocols and applications perform poorly
- Mosh and Sprout show how to handle
  - Changing addresses, suspend/resume connectivity, high losses
  - High rate variability
  - Delay variability
- Contact: hari@mit.edu & http://wireless.csail.mit.edu
  Also http://mosh.mit.edu & http://alfalfa.mit.edu