Transport Protocols for Gracefully Mobile Applications

Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan

M.I.T. CSAIL

November 15, 2012

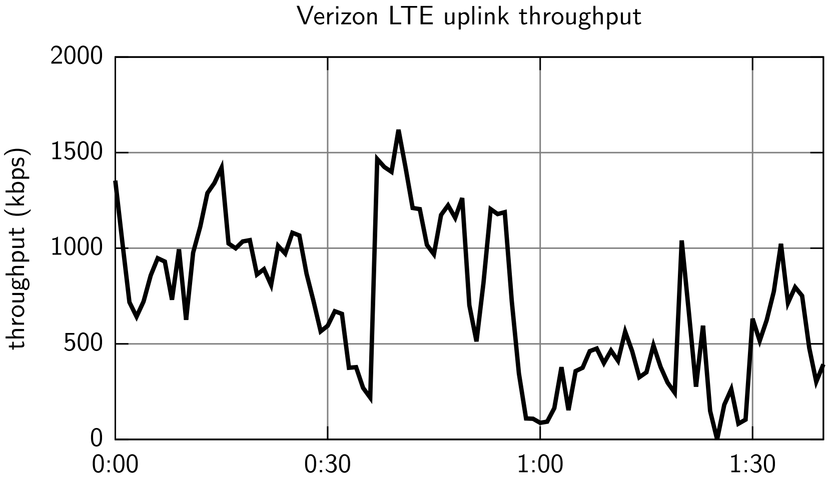
Outline

Sprout: Flow control for interactive apps

SSP: Graceful mobility

Alfalfa: video for varying networks

Mobile wireless networks are variable



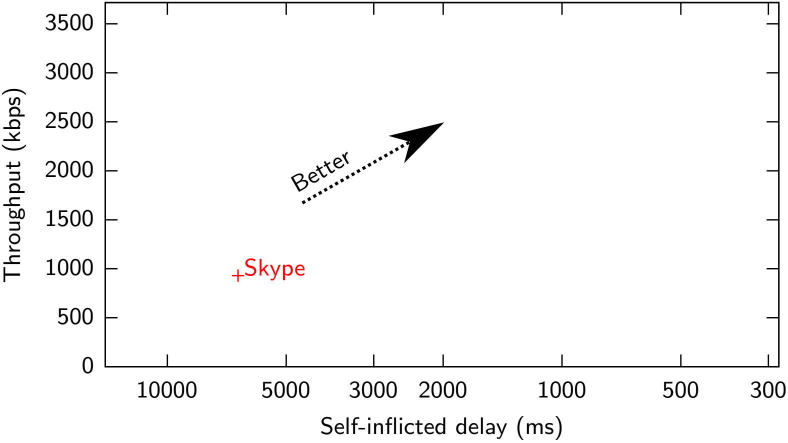
Videoconferencing systems work poorly on LTE

- 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 performance so bad?

- Exiting schemes **react** to congestion signals.
  - 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.

Performance summary



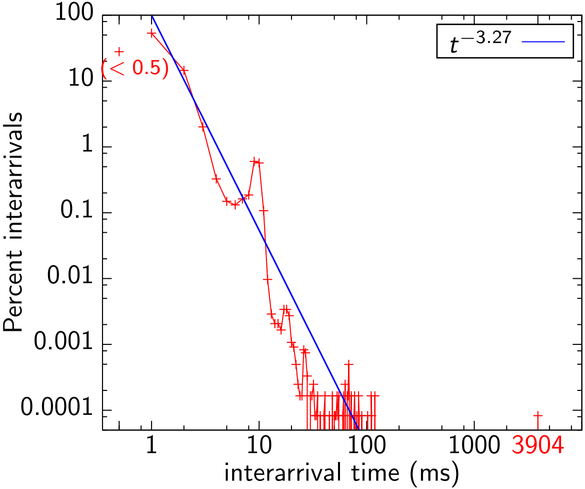
Sprout's goal

- As much throughput as possible, with
- bounded risk of delay $>$ 100 ms.

Bounded risk of delay

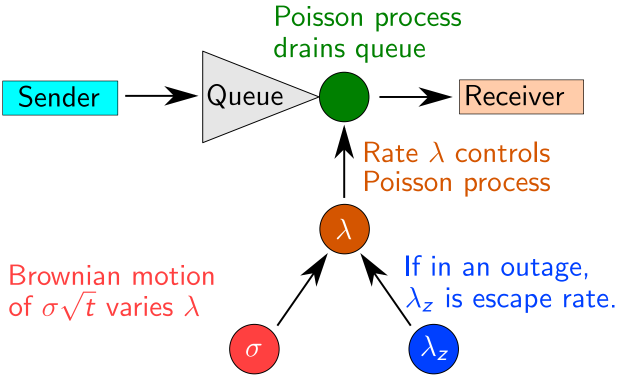
- **Infer** link speed from interarrival distribution.
- **Predict** future link speed.
  - Don't wait for congestion.
- **Control:** Send as much as possible, but require:
  - 95% probability all packets will arrive within 100 ms.

Infer link speed from interarrival distribution



Predict future link speed

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



Poisson process drains queue

Rate λ controls Poisson process

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

If in an outage, λ_z is escape rate.

The cautious forecast

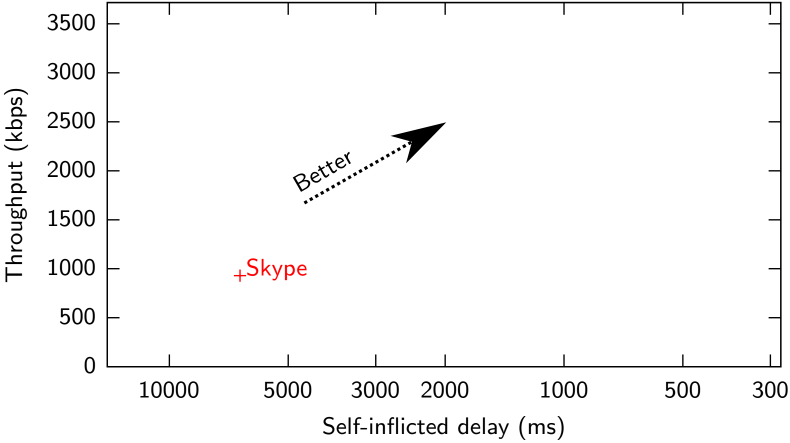
- Receiver has cloud of current link speeds
- For eight 20 ms ticks in the future:
  - Predict future link speed
  - Find 5th percentile of cumulative packets
- Send forecast to sender (piggyback)
- Most of the math is precalculated.

Limitations

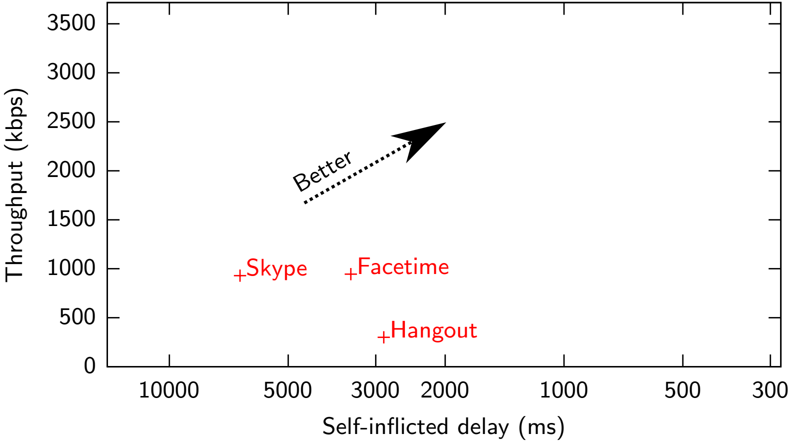
- Stochastic model has not been tuned
- Designed for cellular link with per-user queue
- If other users can cause you big delay, can't solve end-to-end

Verizon LTE uplink: head-to-head

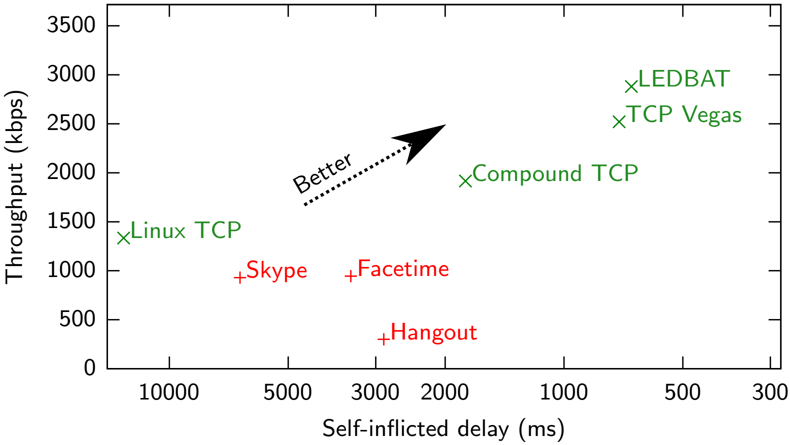
Verizon LTE uplink



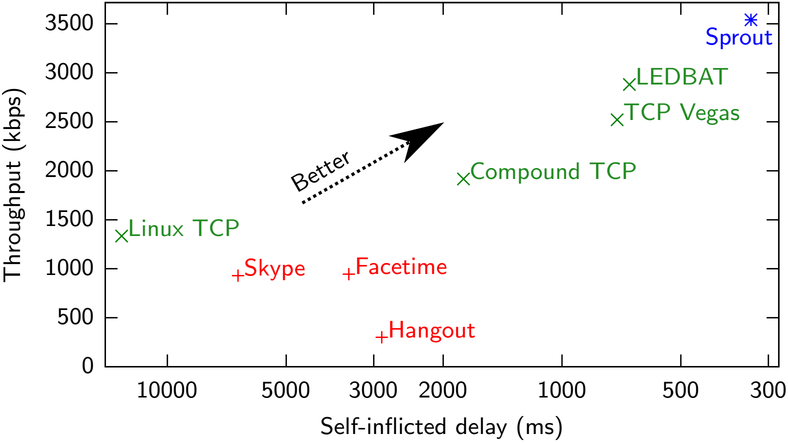
Verizon LTE uplink



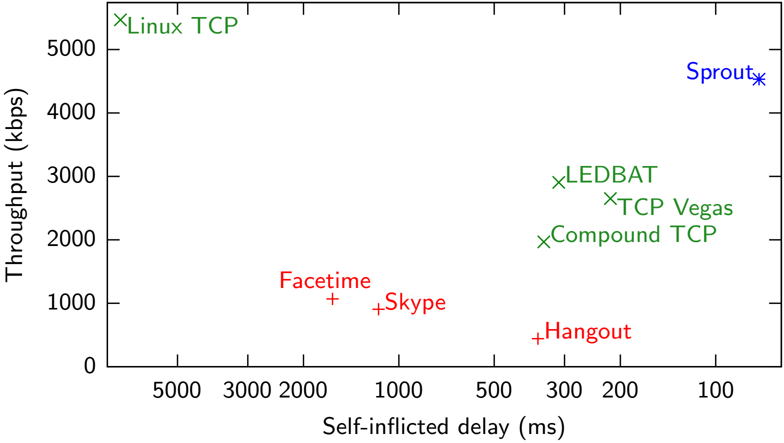
Verizon LTE uplink



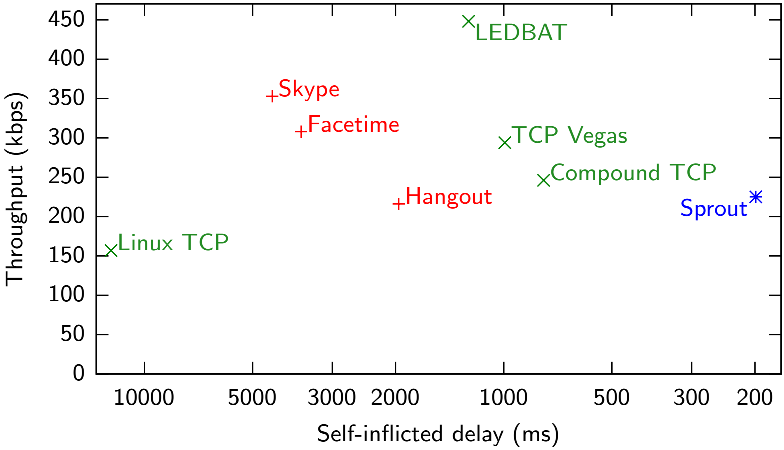
Verizon LTE uplink



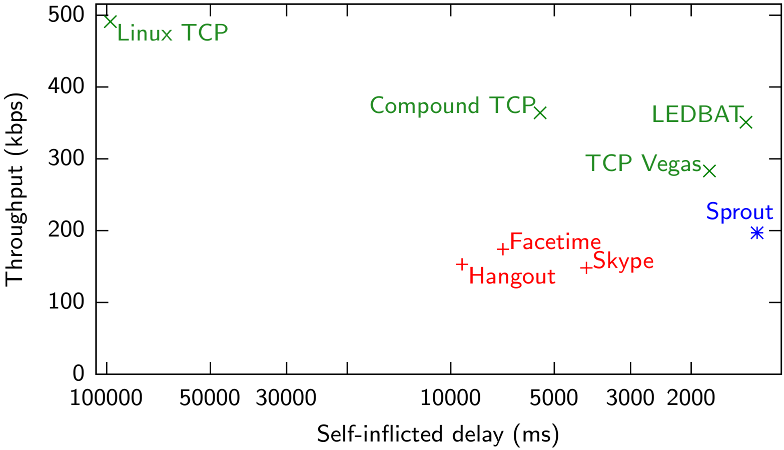
Verizon LTE downlink



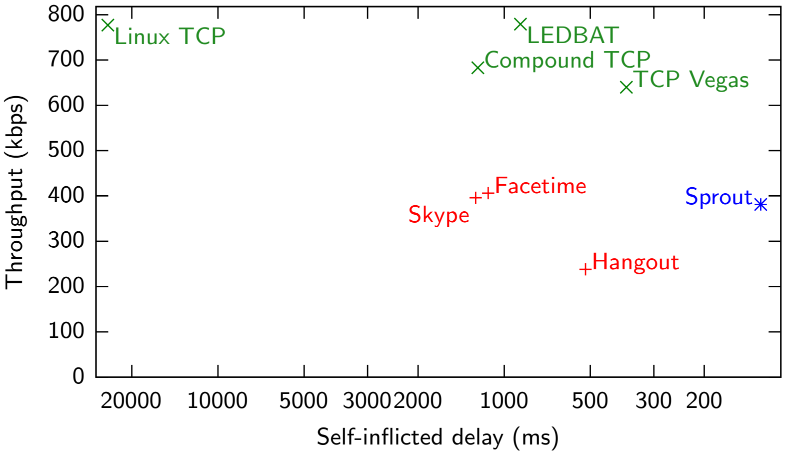
Verizon 3G (1xEV-DO) uplink



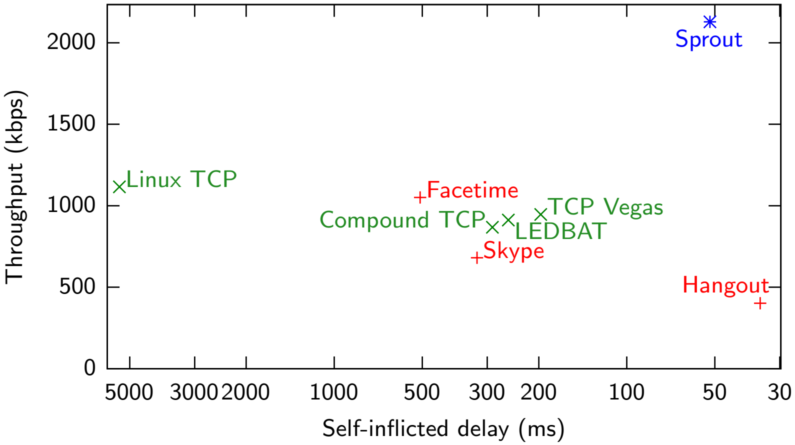
Verizon 3G (1xEV-DO) downlink



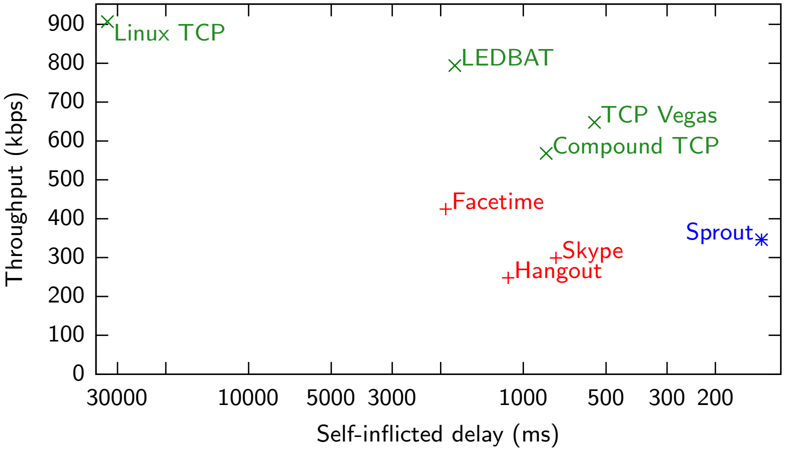
AT&T LTE uplink



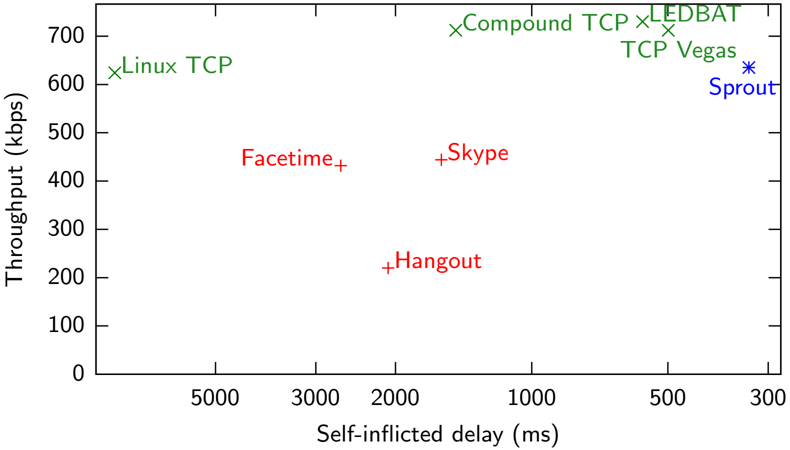
AT&T LTE downlink



T-Mobile 3G (UMTS) uplink



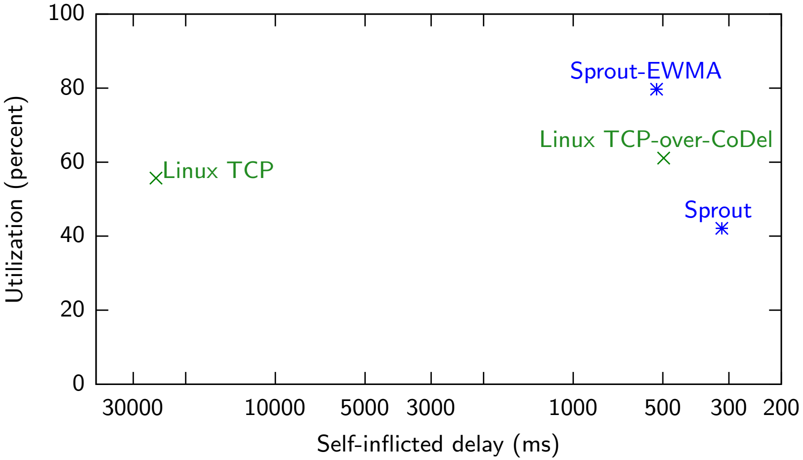
T-Mobile 3G (UMTS) downlink



Overall results

Sprout vs.	Avg. speedup	Delay reduction
Skype	2.2×	7.9×
Hangout	4.4×	7.2×
Facetime	1.9×	8.7×
Compound	1.3×	4.8×
TCP Vegas	1.1×	2.1×
LEDBAT	Same	2.8×
Linux TCP (CUBIC)	1.1×	79×

Competes with AQM even though end-to-end



Future work

- Public contest for best predictor
- Anybody will be able to build protocol using results

Sprout for controlled delay over cellular networks

- **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**

Sprout: Flow control for interactive apps

SSP: Graceful mobility

Alfalfa: video for varying networks

Motivation: frustration with SSH

- Can't roam:
  - . . . across Wi-Fi networks.
  - . . . from Wi-Fi to cell or vice versa.
- Can't sleep and wake up (usually).
  - . . . TCP disconnects if unacked data for too long.
- Responds poorly to packet loss.
- All UI requires round-trip to server.

Octet stream is wrong layer of abstraction

- Client wants *latest* screen.
- After interruption, don't want to replay megabytes.
- But SSH doesn't understand data, so must send everything.
- TCP fills buffers, so Control-C takes forever.

What we built

1. Protocol for low-latency **object synchronization**
   - with roaming
   - through suspend/resume
   - over lossy network paths
2. Supports **rolling latency compensation** on client side
3. Mobile shell application to replace SSH.

State Synchronization Protocol

- 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.

Secure quick roaming

- Protected by AES-OCB (Krovetz 2011)
  - Integrity and confidentiality with one key.
- All packets are idempotent operations.
- Unlike SSH or TLS, connection control is also authenticated.
  - Attacker cannot terminate connection with FIN or RST.
- Roaming is easy:
  - Source address of latest authentic packet from client $\Rightarrow$ server's new target
  - Client may not even **know** it has roamed.

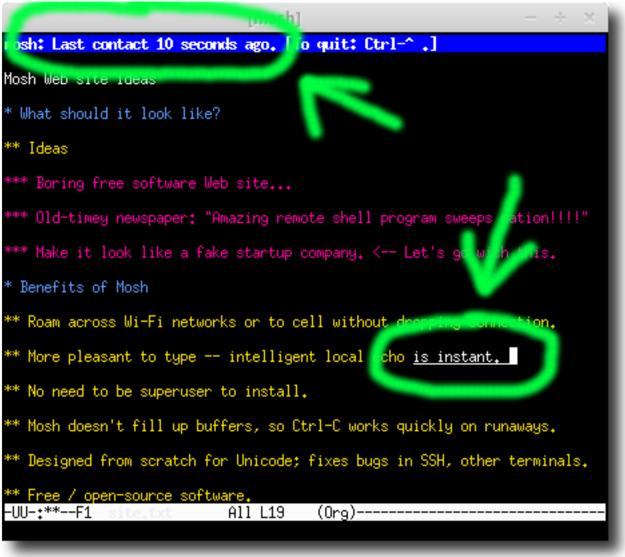
P·retransmissions trade performance for robustness.

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*.
- Better than Meteor's on/off "local mode."



```
mosh: Last contact 10 seconds ago. [To quit: Ctrl-^ .]
Mosh Web site ideas
* 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.
** Free / open-source software.
-UU-:**--F1  site.txt        All L19     (Org)------------------------------
```

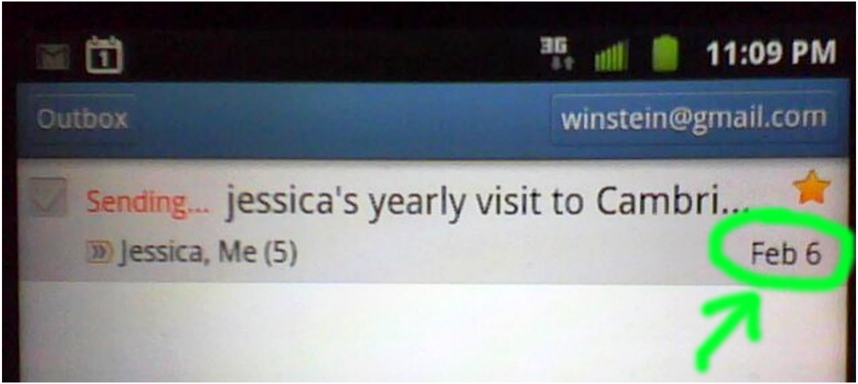
Demo

Deployment

- In Debian, Ubuntu, Fedora, Gentoo, Arch, Slackware.
- Available for Red Hat, CentOS, Oracle Linux.
- In MacPorts, Homebrew, FreeBSD ports collection.
- Works on Cygwin, Solaris, experimental port to Android.
- Cover of Linux Magazine this month (Nov. 2012).
- Top repository of the month on GitHub.
- 300,000+ page views, 75,000+ downloads, 1,500+ VCS followers.

Gmail app if user roams at the wrong time

July 5, 2012:



Outbox winstein@gmail.com

Sending... jessica's yearly visit to Cambri...

Jessica, Me (5) Feb 6

State synchronization for all

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

- Android Gmail app
- Skype
- Google Chat
- gmail.com
- quora.com
- Google Voice
- Twitter

These problems may also be expressed as state synchronization.

Sprout: Flow control for interactive apps

SSP: Graceful mobility

Alfalfa: video for varying networks

Coded video is also a state synchronization problem

- P-pictures are “diff” from previously-sent frame.
- B-pictures are “diff” from two previously-sent frames.

Mobile video conferencing

For reliable mobile video conferencing, application wants from:

- **Network**: "How much is it safe to send now?"
  - TCP and DCCP don't supply this.
- **Video encoder**: "Give me a P-picture from this frame to *this* frame with maximum length x."
  - x264 (H.264) and libvpx (VP8) don't do this.
- **Video decoder**: "Apply this P-picture to *this* predicate picture, and give me the results."
  - QuickTime, libavcodec, etc. don't allow this.

Solution: Sprout/SSP and "explicit-state" video codec API.

Video complications

- P-picture is not just based on predicate *frame*.
- Also have quantization tables, probability tables, other inter-frame state.
- Specifications don't envision this use.
- Aren't explicit about what state needs to be carried across frames.

So far, we have implemented API for MPEG-2 video.

What about Netflix / YouTube?

Current practice:

- encode multiple quality levels, each with VBV.
- player can switch, but requires I-picture and visible jump.
- often switch only available every 10 seconds!

Our view:

- VBV was designed for isochronous broadcast channels!
- Ditch VBV: the player's buffer is all that matters.
- Encode full trellis of diffs between quality levels.
- **Let the decoder choose** how it wants to plan ahead.
- Can run over HTTP.

Conclusion

- Sprout: end-to-end flow control for cellular networks that matches or outperforms in-network modifications.
- SSP: protocol for gracefully mobile state synchronization.
- Alfalfa: video for varying networks.