Mosh

An Interactive Remote Shell for Mobile Clients

Keith Winstein and Hari Balakrishnan

keithw@mit.edu

June 14, 2012

What we built

1. Protocol for low-latency **object synchronization**
    - with roaming
    - through suspend/resume
    - over marginal networks
2. Mobile shell application to replace SSH, telnet.
    - with "predictive" instant keystroke feedback

Remote terminals

- 1969: TELNET (RFC 15)
- 1977: SUPDUP
- 1991: BSD rlogin
- 1995: SSH

Secure Shell, 1995

- Uses TCP.
  - Connection named by IP:port endpoints.
- Sends:
  - user keystrokes → server
  - octet stream (coded screen updates) → client terminal
- All UI from server.

Problems with SSH

- Can't roam:
  - . . . across Wi-Fi networks.
  - . . . from Wi-Fi to cell or vice versa.
- Times out if data unacknowledged after *n* minutes.
  - . . . if laptop goes to sleep.
- Responds poorly to packet loss.

More problems with SSH

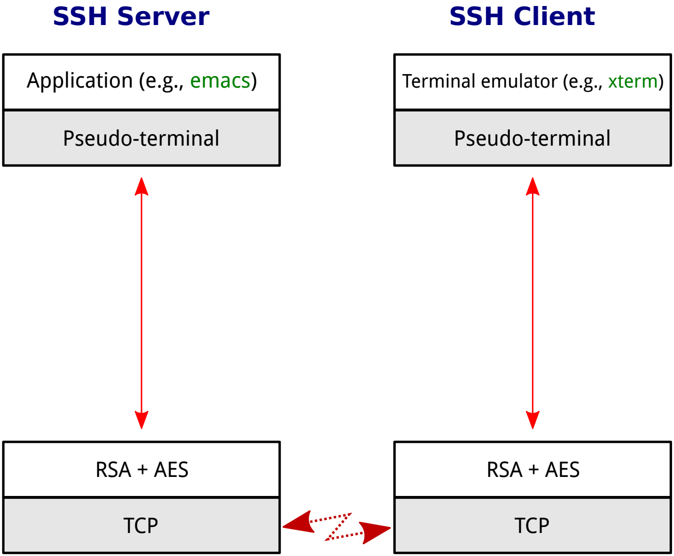
- Byte stream is wrong layer of abstraction.
    - Client wants *latest* screen.
    - Don't want to replay megabytes in between.
    - SSH doesn't understand data, so must send everything.
    - TCP fills buffers, so Control-C takes forever.
- Typing and editing on high-latency path is frustrating.
    - Cellular wireless (100 ms to 500 ms)
    - Intercontinental (250 ms)
    - Loaded "4G LTE" (5,000 to 40,000 ms!)

State Synchronization Protocol

- Runs over UDP.
- Instead of synchronizing *octet streams*, synchronize *objects*.
- Object interface:
  - `diff`: make vector from state $A \rightarrow B$
  - `patch`: apply vector to A to make B
- Object implementation, **not protocol**, defines synchronization semantics.

State Synchronization Protocol (cont.)

- Protected by AES-OCB (Krovetz 2011)
  - Integrity and confidentiality with one key.
- Key exchange happens out of band.
  - Uses SSH to bootstrap.
  - Runs `mosh-server` on remote side.
  - No privileged code, no daemons.
- Roaming is easy:
  - Source address of latest authentic packet from client ⇒ server's new target



SSH Server

Application (e.g., emacs)

Pseudo-terminal

RSA + AES

TCP

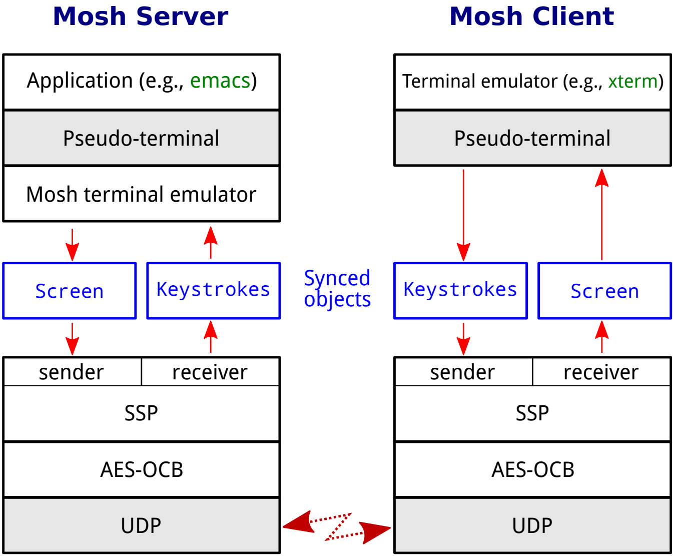
SSH Client

Terminal emulator (e.g., xterm)

Pseudo-terminal

RSA + AES

TCP



Mosh Server

Application (e.g., emacs)

Pseudo-terminal

Mosh terminal emulator

Screen

Keystrokes

sender

receiver

SSP

AES-OCB

UDP

Synced objects

Mosh Client

Terminal emulator (e.g., xterm)

Pseudo-terminal

Keystrokes

Screen

sender

receiver

SSP

AES-OCB

UDP

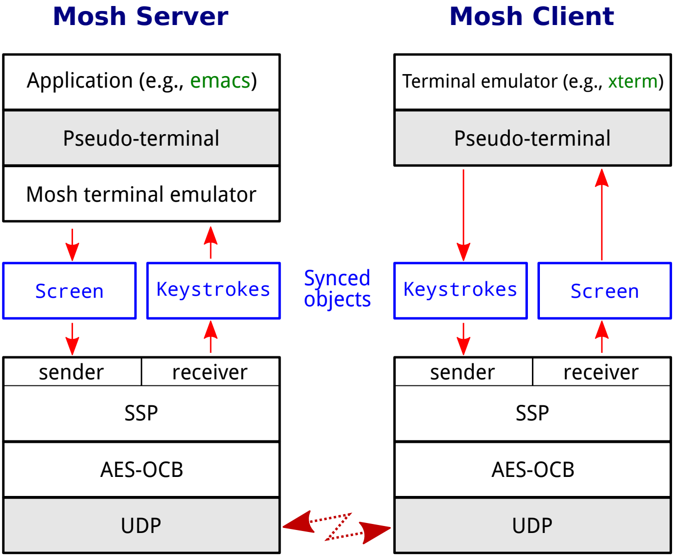
State Synchronization Protocol (cont.)

- **Flow control**: adapt frame rate to network conditions.
- Minimum interval between frames: smoothed RTT/2.
- Can skip over states.
- “P·retransmissions” (post-paper improvement).

P·retransmissions

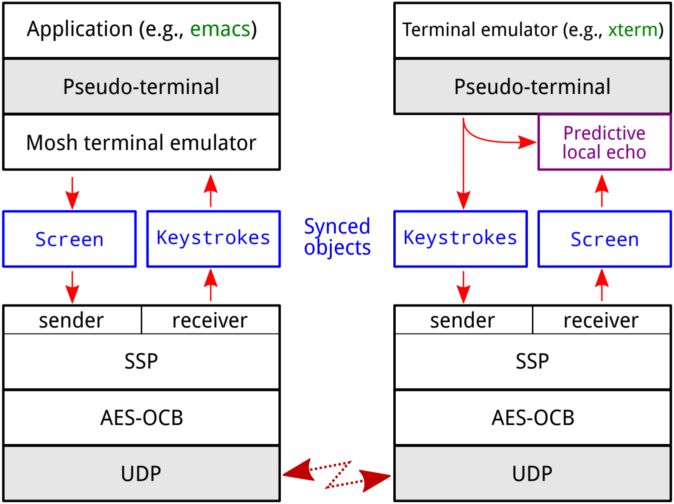
“Prophylactic” retransmission reduces latency in presence of loss.

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 *p·retransmission*.
6. If retransmission is shorter or not much longer, send it instead.


Mosh Server
Mosh Client
Application (e.g., emacs)
Pseudo-terminal
Mosh terminal emulator
Screen
Keystrokes
Synced objects
Terminal emulator (e.g., xterm)
Pseudo-terminal
Keystrokes
Screen
sender
receiver
SSP
AES-OCB
UDP
sender
receiver
SSP
AES-OCB
UDP

Mosh Server

Mosh Client



Application (e.g., emacs)

Pseudo-terminal

Mosh terminal emulator

Screen

Keystrokes

sender

receiver

SSP

AES-OCB

UDP

Synced objects

Terminal emulator (e.g., xterm)

Pseudo-terminal

Predictive local echo

Keystrokes

Screen

sender

receiver

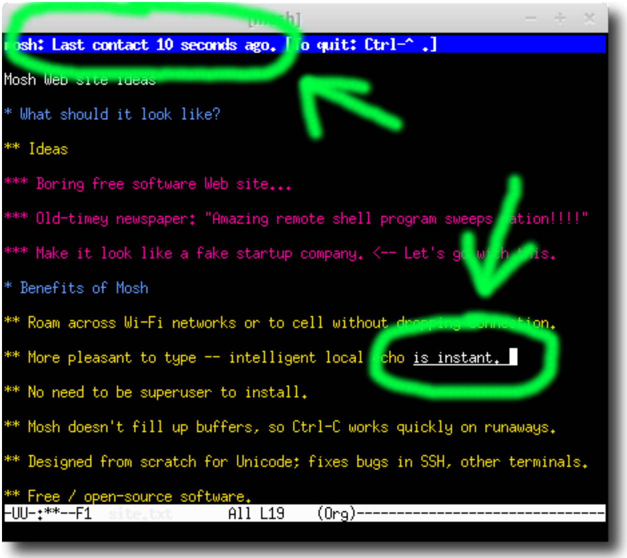
SSP

AES-OCB

UDP

Speculative Local Echo and Editing

- Client anticipates server response.
- Runs predictive model in the background.
- Make predictions in *epochs*.
- If any from epoch *n* is confirmed, show whole epoch.
- If user does something difficult to handle, become tentative: *increment epoch*.
  - Carriage return
  - Escape
  - Up/down arrow
  - Control char



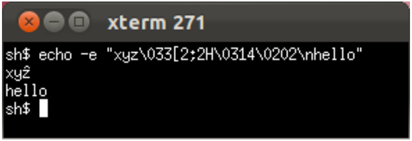
```
[mosh]
mosh: Last contact 10 seconds ago. [To quit: Ctrl-^ .]

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

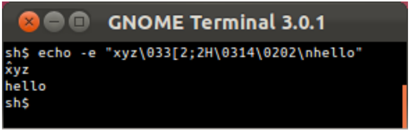
Benefits

- Roaming and suspend/resume:
  - Sleep and wake up later.
  - Change networks at will (Wi-Fi, cellular, wired, VPN).
- Helpful warnings: won't hang without notice.
- Performance:
  - Works on marginal links.
  - Good interactivity even when RTT is $>$ 100 ms.
  - Semantically appropriate flow control (won't fill up queues, Ctrl-C works quickly, no beeping fits).
- Security
  - Uses SSH to bootstrap: no privileged code, no daemons.
- Correctness
  - Unicode

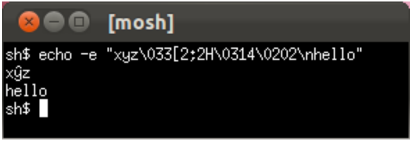
Unicode admits varying interpretations.



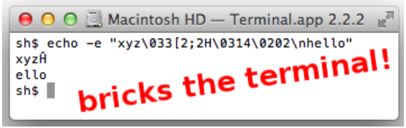
```
sh$ echo -e "xyz\033[2;2H\0314\0202\nhello"
xyẑ
hello
sh$ █
```



```
sh$ echo -e "xyz\033[2;2H\0314\0202\nhello"
x̂yz
hello
sh$
```



```
sh$ echo -e "xyz\033[2;2H\0314\0202\nhello"
xŷz
hello
sh$ █
```



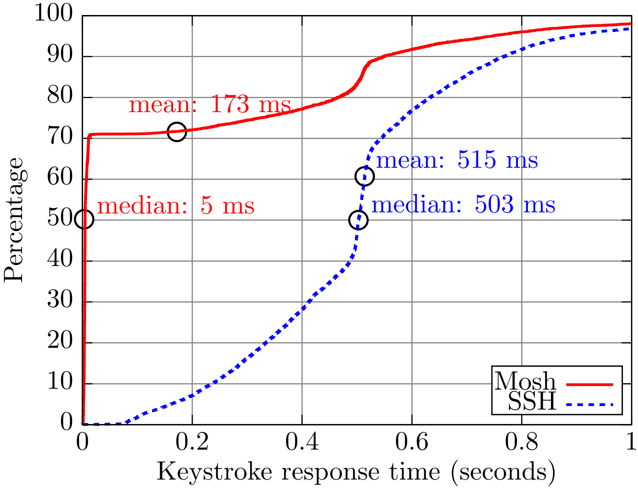
```
sh$ echo -e "xyz\033[2;2H\0314\0202\nhello"
xyzĤ
ello
sh$ █
```

bricks the terminal!

Evaluation

- Collected 40 hours of terminal usage from six users.
- Covers 10,000 keystrokes using shell, e-mail, text editor (emacs and vi), chat, Web browser.
- Replayed over:
  1. Sprint 1xEV-DO (3G)
  2. Verizon LTE (4G)
  3. MIT-Singapore
  4. 50% loss path
- Result: 70% of keystrokes predicted instantly.
- Prediction errors $< 1\%$

Sprint 1xEV-DO cumulative keystroke response distribution



Evaluation (cont.)

Verizon LTE service in Cambridge, Mass., running one concurrent TCP download:

	Median latency	Mean	σ
SSH	5.36 s	5.03 s	2.14 s
Mosh	$<$ 0.005 s	1.70 s	2.60 s

Deployment

- Distributed in Debian, Ubuntu, Fedora, Gentoo, Arch, Slackware versions of GNU/Linux.
- Available via EPEL for Red Hat, CentOS, Oracle Linux.
- Included in MacPorts, Homebrew, FreeBSD ports collections.
- Works on Cygwin and Solaris, (very raw) on Android and iOS.
- News stories in April on Hacker News, Reddit, The Register, Twitter, Slashdot, Barrapunto.
- Top repository of the month on GitHub.
- 200,000+ page views, 70,000+ downloads, 1,200+ followers of version control repo.

Reception

@xlfe: "one of those times you don't realize something is broken until you see it fixed"

@adamhjk: "the user experience really is dreamy."

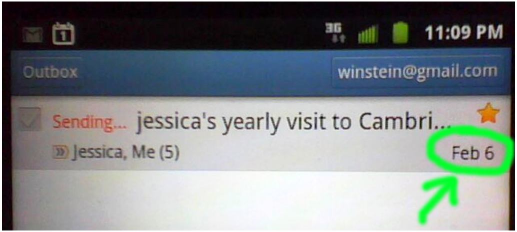
@esmolanka: "mosh is awesome. Tested it for two weeks and it really made my life easier: faster feedback and no more reconnects(!)"

@andyd: "Using mosh on the train rather than plain ssh, and it does actually make a huge difference!"

USENIX review: "ISO 2022 locking escape sequences oh flying spaghetti monster please kill me now."

State Sync Protocol for all?

- We believe SSP may be appropriate for many network problems.
- Android Gmail, Google Chat, Skype cannot roam without failure.
- **June 13, 2012**:



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

Jessica, Me (5) Feb 6

- Neither can Gmail (Web site).
- These problems can be expressed as state synchronization.

Next Steps

- Essay to appear in *;login:* magazine.
- Mosh software under development by a team of contributors.
- We are working to apply SSP to mobile videoconferencing.
- We hope to show quantitative improvement on standard metrics (latency, quality), plus features like roaming.

Summary

- SSP is a secure datagram protocol that synchronizes abstract objects across a roaming IP connection.
- Mosh uses SSP to synchronize a terminal emulator with predictive local echo.
- In evaluations with 10,000 real-world keystrokes from six users, Mosh markedly reduced user-visible latency across several Internet paths.
- We think SSP will be useful for other applications as well.
- `http://mosh.mit.edu`