Mosh

An Interactive Remote Shell for Mobile Clients

Keith Winstein and Hari Balakrishnan

M.I.T. CSAIL

June 14, 2012

http://mosh.mit.edu

Secure Shell, 1995

- Uses TCP.
- Sends:
  - user keystrokes → server
  - octet stream (coded screen updates) → client terminal
- All UI comes from server.
  - . . . including keystroke echoes.

Problems with SSH

- Can’t roam:
  - . . . across Wi-Fi networks.
  - . . . from Wi-Fi to cell or vice versa.
- Can’t sleep and wake up (usually).
- Responds poorly to packet loss.

More problems with SSH

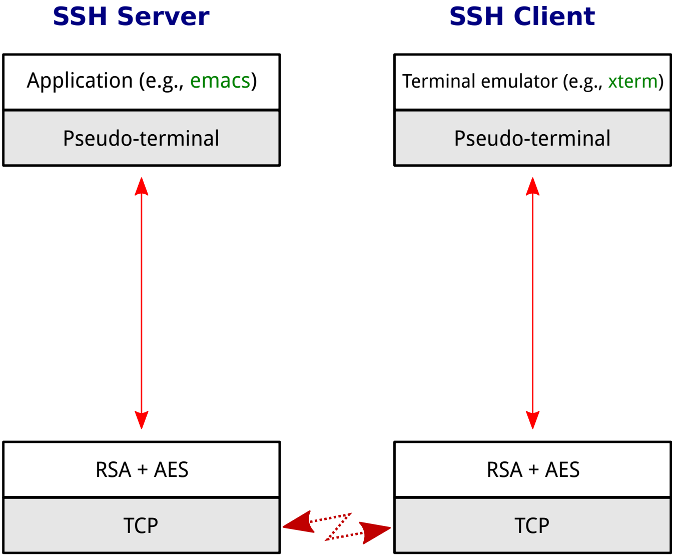
- Octet stream is wrong layer of abstraction.
  - Client wants *latest* screen.
  - After interruption, don't want to replay megabytes.
  - But SSH doesn't understand data, so must send everything.
  - TCP fills buffers, so Control-C takes forever.
- Typing and editing on high-latency path is frustrating.
  - Unloaded cellular wireless (50 ms to 500 ms)
  - Intercontinental (250 ms)
  - Loaded "4G LTE" (5,000 to 40,000 ms!)

What we built

1. Protocol for low-latency **object synchronization**
   - with roaming
   - through suspend/resume
   - over lossy network paths
2. Mobile shell application to replace SSH
   - with "predictive" local echo

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.



SSH Server

Application (e.g., emacs)

Pseudo-terminal

RSA + AES

TCP

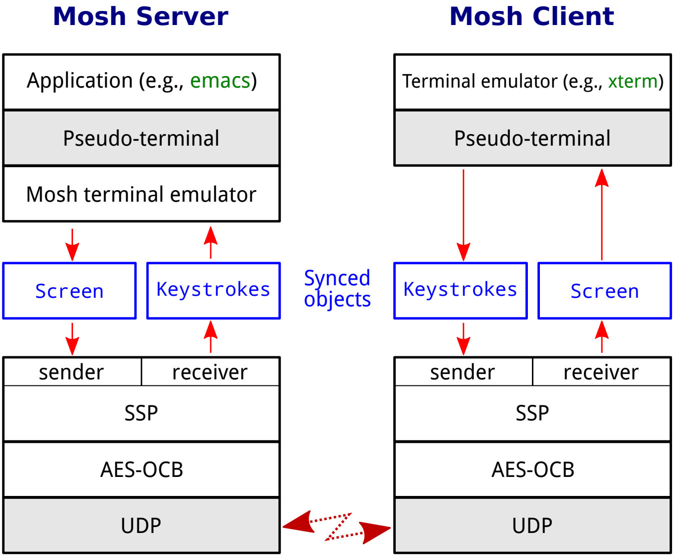
SSH Client

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

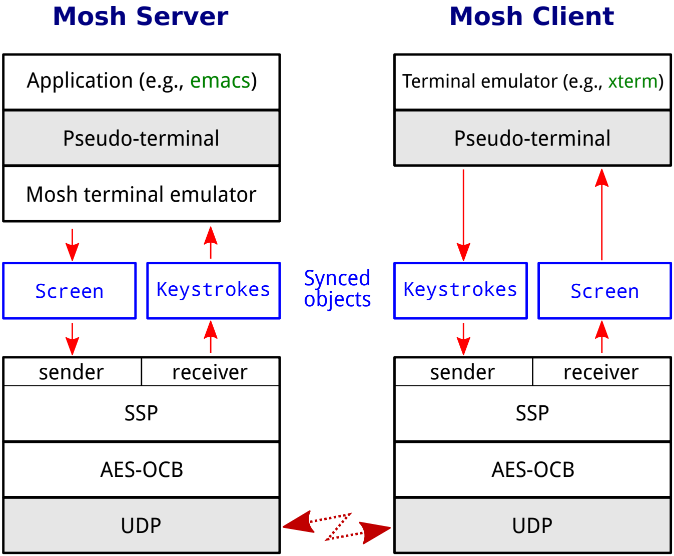
State Synchronization Protocol (cont.)

- 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
  - Client may not even **know** it has roamed.

State Synchronization Protocol (cont.)

- **Flow control**: adapt frame rate to network conditions.
- Don't fill up buffers!
- Can skip over states.
- Tricks to balance robustness vs. throughput.

Predictive Local Echo and Editing



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

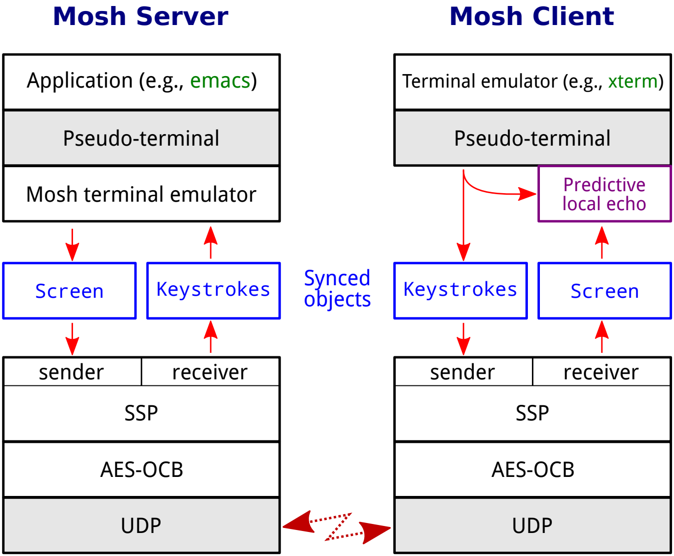
sender

receiver

SSP

AES-OCB

UDP



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

Predictive local echo

Keystrokes

Screen

sender

receiver

SSP

AES-OCB

UDP

Predictive Local Echo and Editing

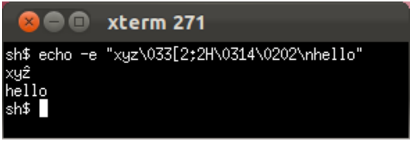
- Client anticipates server response.
- Runs predictive model in the background.
- Make predictions in *epochs*.
- If any from epoch n is confirmed, show whole epoch.
- If user does something difficult to handle, become tentative: *increment epoch*.
  - Carriage return
  - Escape
  - Up/down arrow
 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

Demo

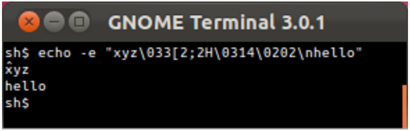
Evaluation

- Tested Mosh with 10,000 keystrokes collected from six users.
- 70% of user keystrokes displayed instantly.
- Good performance on lossy links vs. SSH.
- Full results in paper.

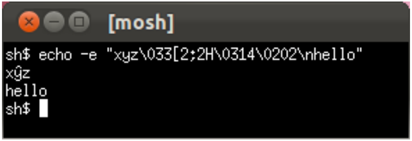
Unicode on Unix is still full of bugs.



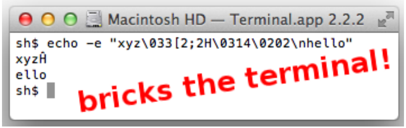
```
sh$ echo -e "xyz\033[2;2H\0314\0202\nhello"
xyẑ
hello
sh$ █
```



```
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!

Deployment

- In Debian, Ubuntu, Fedora, Gentoo, Arch, Slackware.
- Available for Red Hat, CentOS, Oracle Linux.
- In MacPorts, Homebrew, FreeBSD ports collection.
- Works on Cygwin and Solaris, (very raw) on Android.
- 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.”

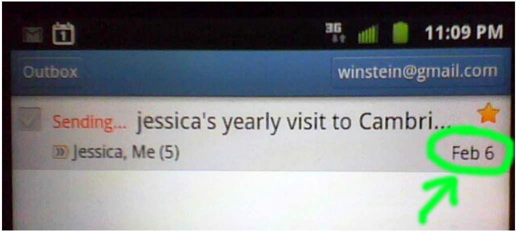
@esmolanka: “mosh is awesome. Tested it for two weeks and it really made my life easier: faster feedback and no more reconnects(!)”

@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?

- SSP may be appropriate for many network problems.
- Android Gmail, Google Chat, Skype cannot roam.
- **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.

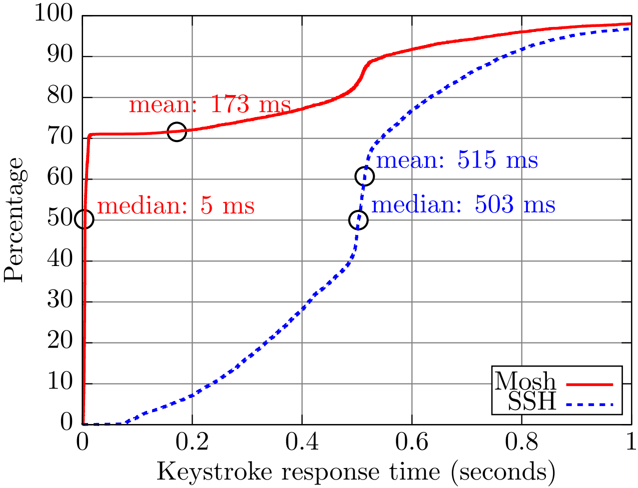
Summary

- 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.
- We think SSP will be useful for other applications as well.
- http://mosh.mit.edu

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

MIT-Singapore Internet path (to Amazon EC2 data center):

	Median latency	Mean	σ
SSH	273 ms	272 ms	9 ms
Mosh	$<$ 5 ms	86 ms	132 ms

SSP with high packet loss

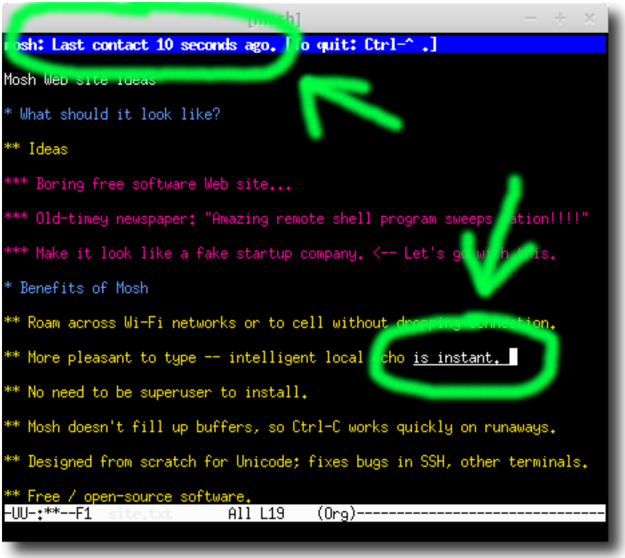
Synthetic link with 100 ms RTT, 50% round-trip i.i.d. packet loss:

	Median	Mean	σ
SSH	0.416 s	16.8 s	52.2 s

P·retransmissions shield against possible future loss.

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 it instead.



```
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)------------------------------
```