From Terminals to Mosh

COS 316: Principles of Computer System Design

Amit Levy & Wyatt Lloyd

Layering and Its Discontents

- How layering can *fail* and why abstraction can sometimes be *bad*
  - Poor choices of abstraction
  - Too much abstraction
- When to “pop open the hood”?
- Important, ubiquitous systems sometimes an accident of history

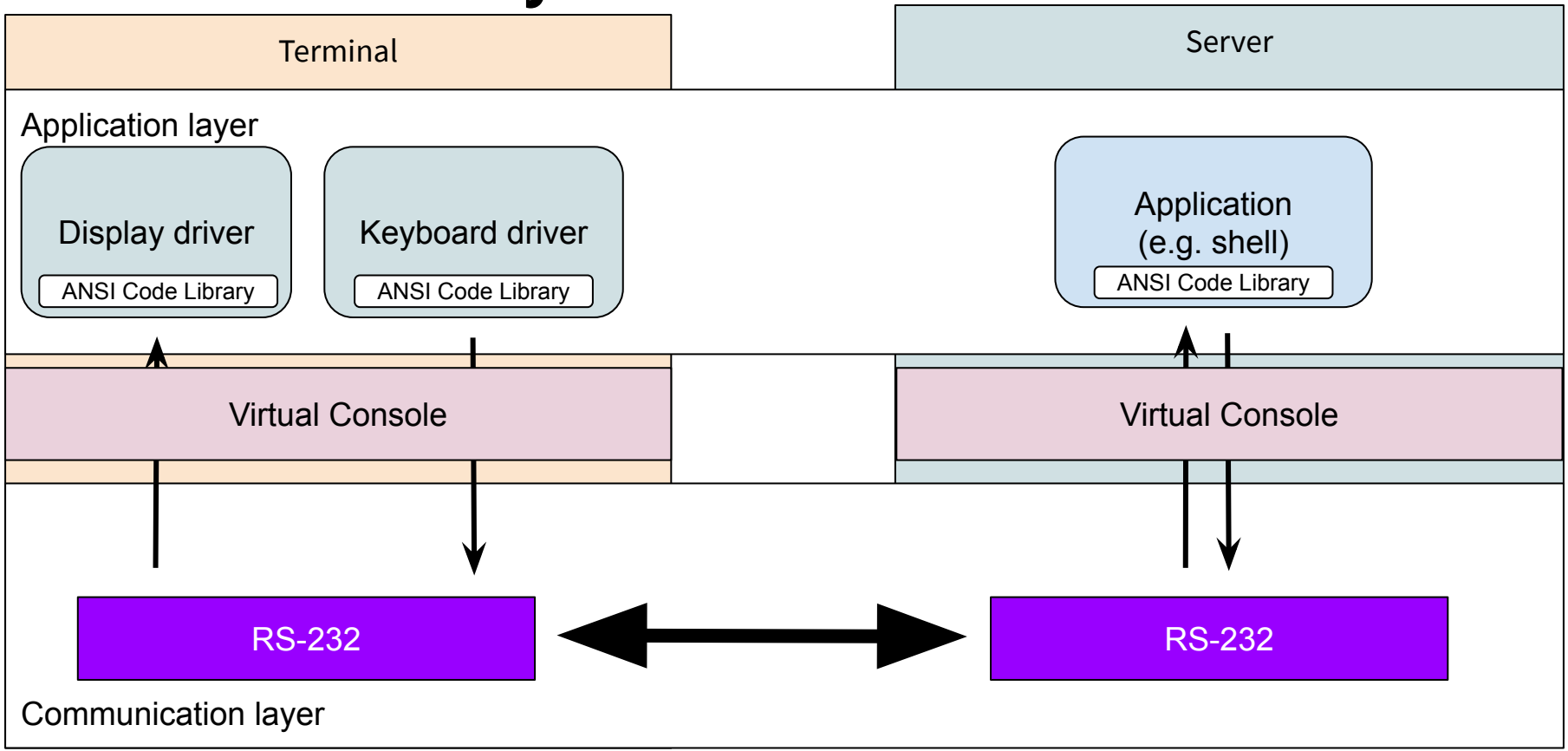
A VT100 Terminal Connected to a PDP-11


digital pdp-11

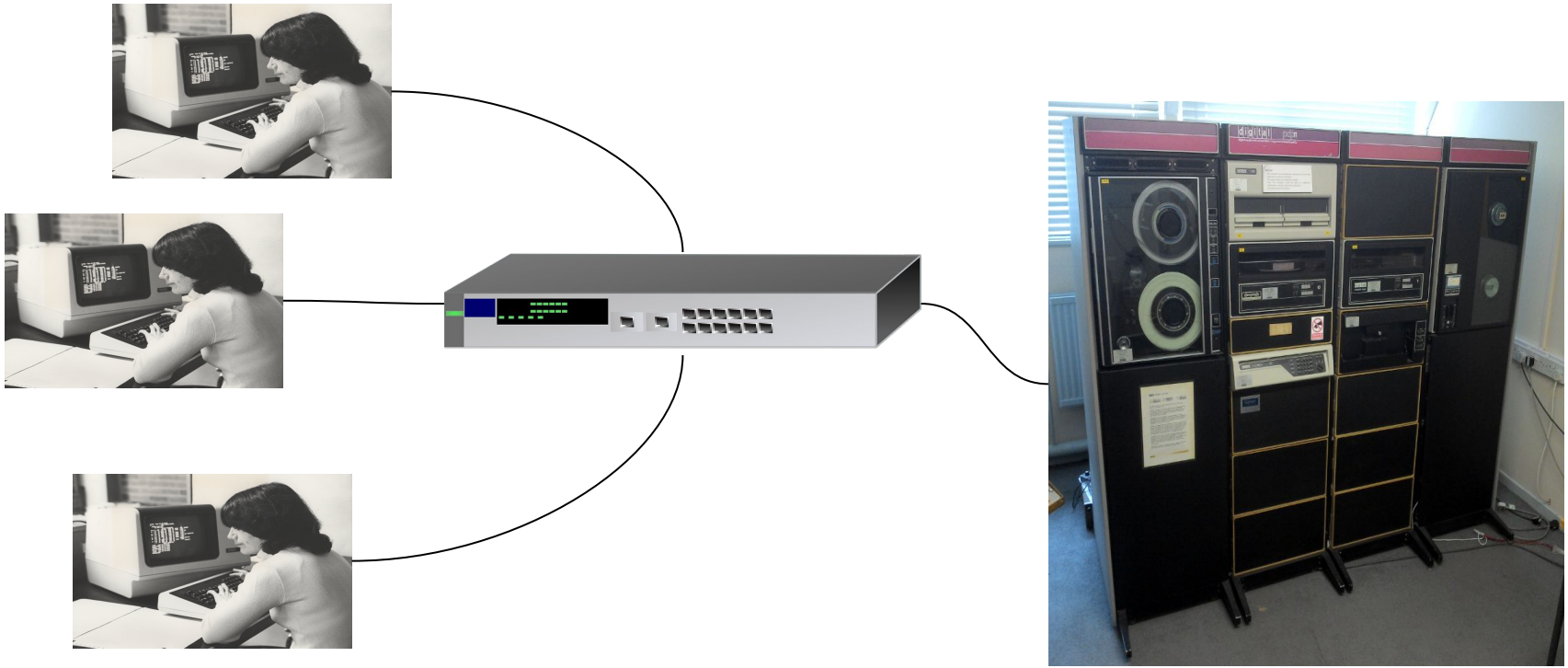
How do terminals work?

- Terminal & server connected via serial line (typically RS-232)
  - Reliable stream of bytes in both directions
- Keystrokes transmitted from terminal to mainframe
- Output transmitted from mainframe to terminal
- Common protocol: ANSI Escape Codes, E.g.:
  - ASCII characters
  - ESC [1 ~ means “Home button”
  - ESC [38 ; 5 ; 12 means “Set foreground color to blue”
- Programs written to interpret incoming byte-streams as keystroke escape-codes
- Terminal interprets escape codes and modifies display accordingly

Terminals in Layers



Remote terminals over a Network

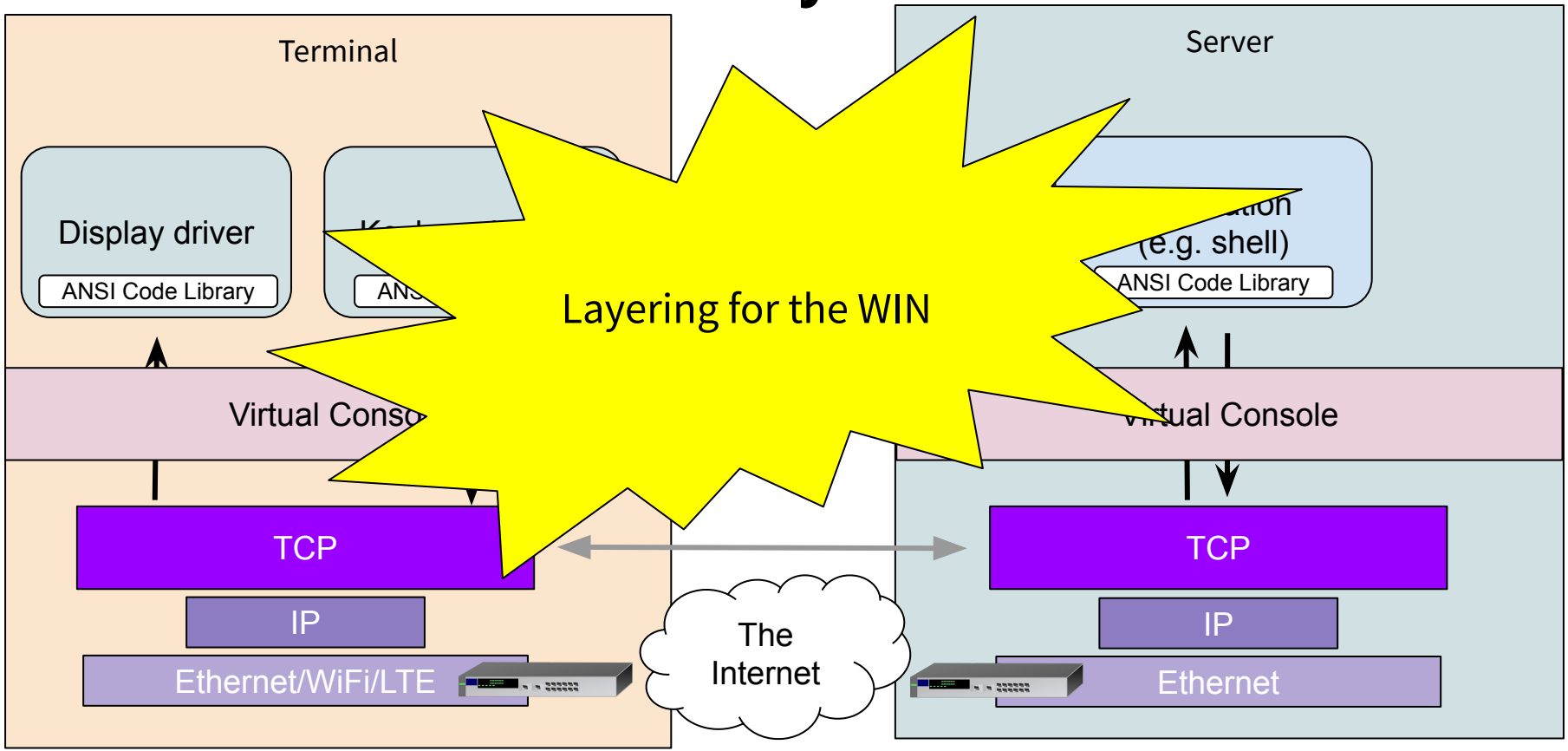


Remote terminals over a Network

How can we support remote terminals over an IP network?

TCP Provides a reliable stream of bytes!

Remote Terminals in Layers



Remote Terminals: Telnet & SSH

Telnet

- Just a TCP server
- Input/output of TCP connection piped to input/output of a shell
- *Don’t every use telnet!*

SSH

- Adds end-to-end encryption
- Allows special kinds of user authentication (e.g. using public/private SSH keys)
- Otherwise more or less the same
- *Use this!*

SSH (Secure Shell) Interlude

- First introduced 1995 as replacement for rlogin & TELNET
- Most server administration happens using SSH
- Basic building block:
  - Git uses SSH (as one common option) to communicate with remote repositories
  - Rsync & SCP file transfer
  - Duplicity backup
  - X11 forwarding
  - Network tunneling (a-la VPN)
- Learn it, love it, use it...

Problems with Layering for Terminal

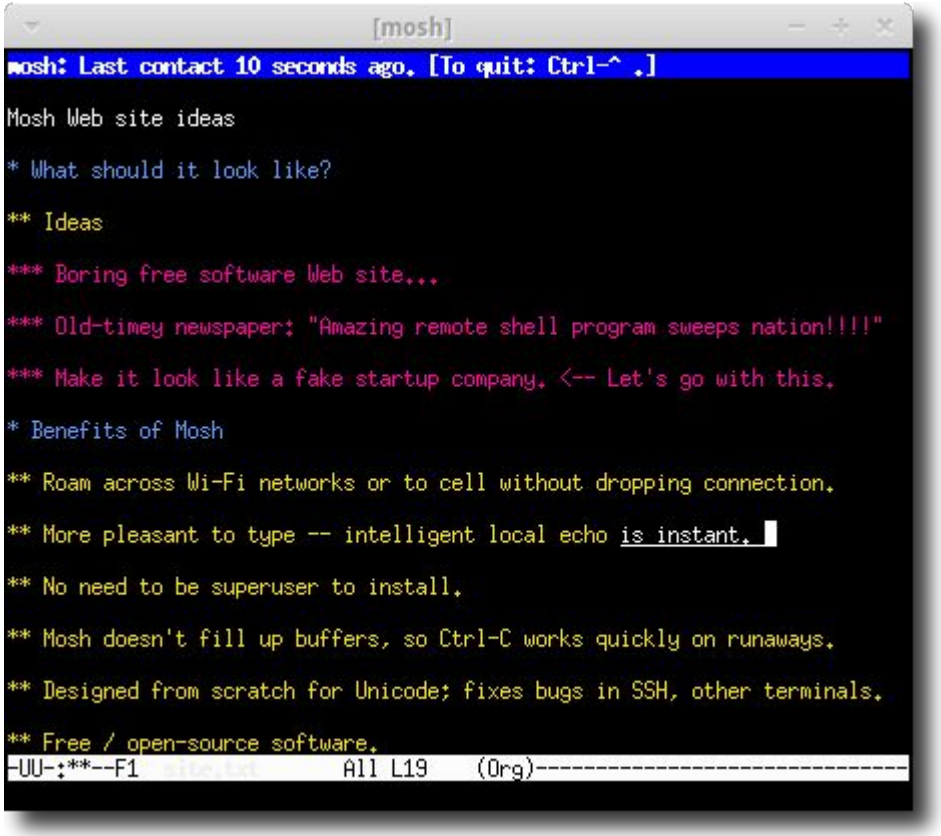
1. IP Roaming
    - TCP connections tied to particular source & destination IP addresses
    - What if I move from LTE on the train to WiFi at home?
    - What if I put my laptop to sleep before a flight and wake it up when I land?
2. Cellular networks can have latencies in the seconds
    - Even dial-up modems had/have latencies of at most 100s of ms!
    - Type a character, then wait ***seconds*** to see it on the screen
    - Control-C doesn't stop characters flooding the terminal screen

A problem of layers' abstractions:

- TCP is connection oriented and reliable
- Stream-of-bytes doesn't know which bytes matter and which don't

Mosh (mobile shell)

- Supports roaming across IP addresses
- Intermittent connections
- Interactive even when connection is very slow



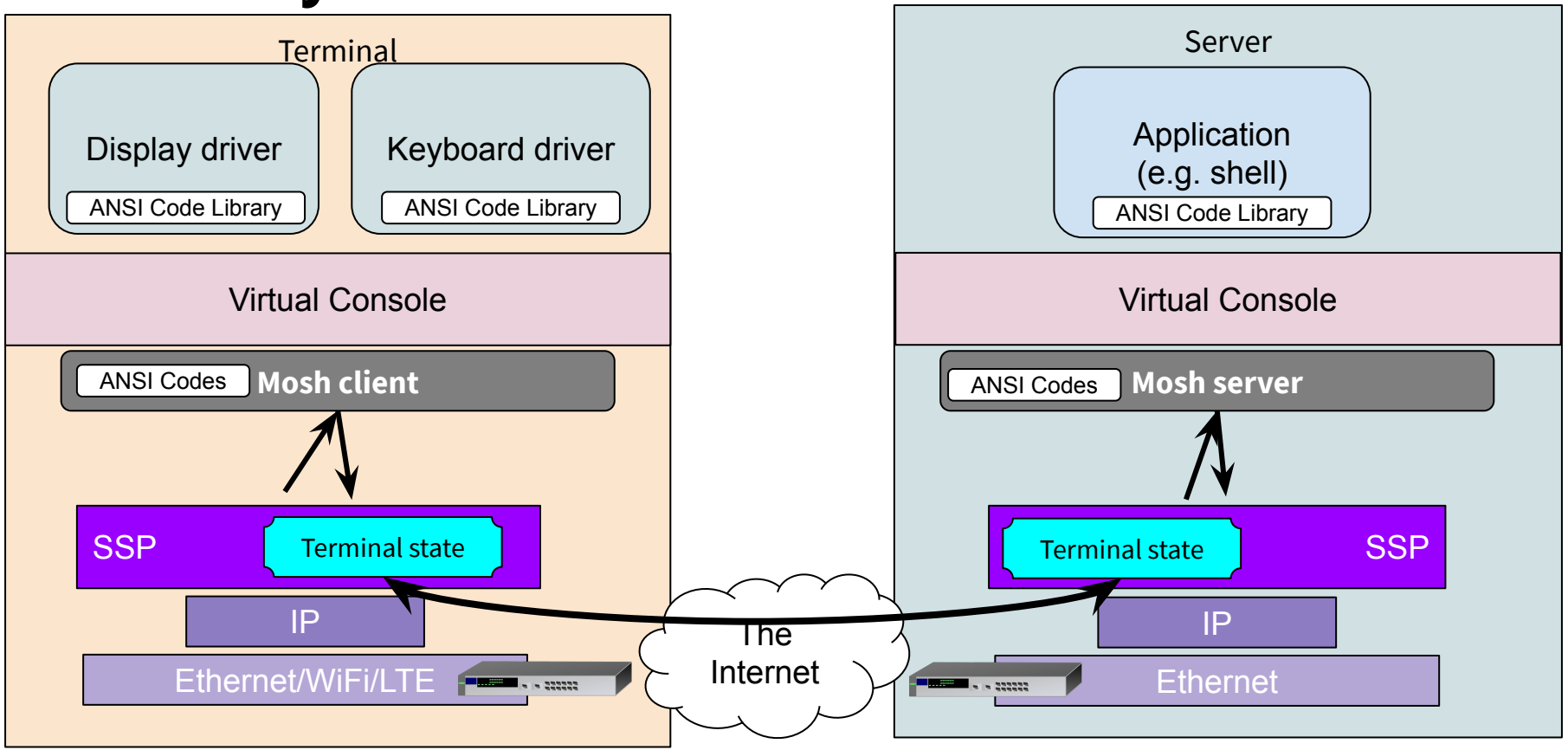
Mosh (mobile shell)

- By Keith Winstein & Hari Balakrishnan
- “Side-project” during Keith’s PhD
- Not as ubiquitous as SSH, but surprisingly popular in a short time
  - E.g. available in every major Linux distribution, OS X, Windows, iOS, Android, etc

Mosh key ideas

- New UDP-based network protocol: State Synchronization Protocol
  - Object synchronization instead of reliable byte-stream
  - Can pick up after arbitrary loss of communication
- Client program has its own terminal emulator
  - Reify object differences to terminal by itself
  - Speculatively executes effects of keystrokes before echo from server

Mosh Layers


Terminal
Display driver
ANSI Code Library
Keyboard driver
ANSI Code Library
Virtual Console
ANSI Codes
Mosh client
SSP
Terminal state
IP
Ethernet/WiFi/LTE
The Internet
Server
Application (e.g. shell)
ANSI Code Library
Virtual Console
ANSI Codes
Mosh server
Terminal state
SSP
IP
Ethernet

SSP Datagram Layer

- Goal is object (state) synchronization
  - E.g. the “object” is the current terminal view
- UDP based protocol
- Maintains a *roaming connection*
  - A packet from *any* IP address that is correctly encrypted & authenticated
  - Just looks at sequence number for ordering (last sequence number wins)
  - IP roaming by replacing the client’s IP with whatever IP was in the last *correct* packet
- Datagrams are *idempotent*
  - Each encodes a *diff* between a numbered source state and target state of object
  - Reordered or repeated packets aren’t handled by construction (because ordering doesn’t matter!)

Terminal Emulator Speculative Local Echo

Maintaining screen state at client & server allows client to speculatively apply local echo

- When user types on client, mosh client “echos” characters to local terminal
  - Mosh uses an underline font-decorator to indicate to the user the output is only speculative
- Server might “acknowledge” the echo by modifying the “real” terminal state
  - Mosh simply removes the underline font-decorator
- Server might not acknowledge (e.g., perhaps we’re in a program that doesn’t echo)
  - Mosh undoes changes to the local terminal object

Mosh and Layers

- Maintaining layering was a WIN for SSH
  - No need to replace most of the software stack
  - Applications do not need to be aware of remote connection
  - Security as an additional layer
  - TCP’s abstraction mirrors a serial connection, but over a packet-switched network
- It doesn’t work under new constraints
  - A reliable byte stream is the *wrong* abstraction
  - Leaving terminal emulation to applications is the *wrong* abstraction
- Major benefit to replacing layers with tuned alternatives