Communication Scheme Makes Popular Applications 'Gracefully Mobile'

The Secure Shell, or SSH, is a popular program that lets computer users log onto remote machines. Software developers use it for large collaborative projects, students use it to work from university servers, customers of commercial cloud-computing services use it access their accounts, and system administrators use it to manage computers on their networks.

First released in 1995, SSH was designed for an Internet consisting of stationary machines, and it hasn't evolved with the mobile Internet. Among other problems, it can't handle roaming: If you close your laptop at the office and reopen it at home, your SSH session will have died; the same goes for an SSH session on a tablet computer that switches from a Wi-Fi connection to the cellular network.

At the Usenix Annual Technical Conference in Boston this month, researchers at MIT's Computer Science and Artificial Intelligence Laboratory presented a paper describing a new remote-login program called Mosh, for mobile shell, which solves many of SSH's problems. The researchers also believe that the communication scheme underlying Mosh could improve the performance of a host of other mobile applications.

Even before they presented the paper, they made Mosh freely available on a number of different websites; it's now been downloaded at least 70,000 times. "That's from the ones that we're able to track," says Keith Winstein, a graduate student in MIT's Department of Electrical Engineering and Computer Science and lead developer of Mosh.

Echo location

Besides roaming, another of the problems that Mosh addresses is the delayed "echoing" of keystrokes in SSH. During a standard SSH session, when a user strikes a key on the keyboard, nothing appears onscreen until information about the keystroke travels to the remote machine, which performs a computation and sends back the result. That's because, in many applications commonly run through SSH, keystrokes don't necessarily correspond directly to displayed symbols: In an email program, for instance, the "N" key might call up the next email; similarly, when a user enters a password, it shouldn't appear onscreen.

Mosh has an algorithm running in the background that deduces when keystrokes should be displayed and when they shouldn't. Until the remote computer confirms Mosh's predictions, the characters onscreen are underlined. "I have never seen it display anything wrong," says Hari Balakrishnan, a professor in the Department of Electrical Engineering and Computer Science and Winstein's coauthor on the Usenix

paper.

The reason Mosh handles roaming so much better than SSH does is that it abandons the Transmission Control Protocol, or TCP — the framework that governs almost all the traffic in today's Internet.

"TCP has some wonderful ideas embedded in it — congestion control, ways of doing reliability and so forth," Balakrishnan says. "But it has this one big, big problem: It provides a reliable, in-order byte-stream abstraction between two fixed endpoints. If you were to pick the worst possible abstraction for the mobile world, it would be that."

With mobile applications, Balakrishnan explains, it's not as crucial that every byte of information be displayed in exactly the order in which it was sent. If you've lost connectivity while using the map application on a smartphone, for instance, when the network comes back up, you probably want an accurate map of your immediate surroundings; you don't want to wait while the phone reloads data about where you were when the network went down.

State currency

Winstein and Balakrishnan developed their own communications protocol, which they call SSP, for state synchronization protocol. SSP, Balakrishnan says, works more like the protocols that govern videoconferencing, where getting timely data about the most recent state of the application is more important than getting exhaustive data about previous states.

Mosh is already proving itself useful: At his computer in his office, Balakrishnan pulls up the connection log for one of the servers in MIT's Athena network; a third of the people logged into it are using Mosh. But in ongoing research, Winstein and Balakrishnan are investigating how SSP can be improved and extended so that other applications can use it as well.

"We have sort of a broader agenda here," Winstein says. "Mosh is a gracefully mobile application. But there's a lot of even more popular network applications that have the same problems, like Gmail, or Google Chat, or Skype. None of these programs gracefully handle mobility, even though they're intended for mobile devices."

"Mosh is a fine bit of engineering, focused on the precise requirements of the application at hand," says Jon Howell, a researcher at Microsoft Research who specializes in web applications. Winstein, Howell says, "spent a lot of effort on the gritty practical details of correct terminal behavior."

Howell points out that there's a good deal of existing research on state synchronization, the technique underlying SSP, and he's not sure that SSP offers any clear advantages over its predecessors. "If [Winstein] pushes the SSP idea farther, I suspect he'll find himself covering some of the [same] territory," Howell says. "Perhaps he'll discover a new strategy or a new class of applications for which

synchronization is particularly useful."

Article Provided by MIT News [1]

Posted by Sara Cohen, Editorial Intern

July 2, 2012

Links:
[1] http://web.mit.edu/newsoffice/2012/mobile-secure-shell-0628.html