Erlang secure RPC and SSH module

Kenji Rikitake

for Tokyo Erlang Workshop #4

26-FEB-2010

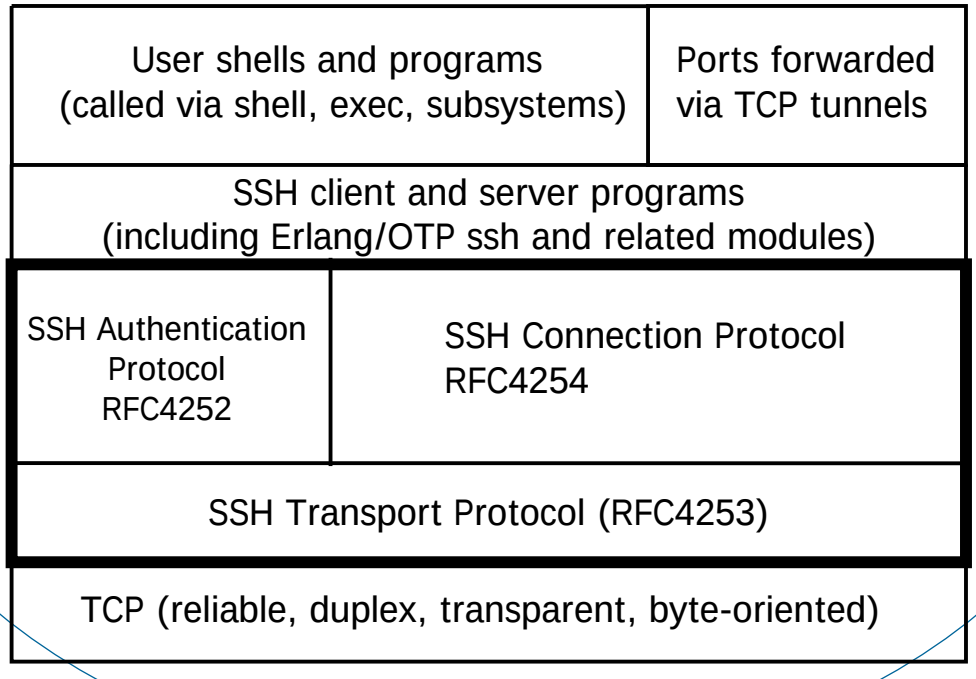
Why SSH for RPC?

- Need for secure remote procedure call
  - Distributed Erlang through hostile networks
  - current inet_ssl_dist is still experimental
    - epmd is not cryptographically protected anyway
- SSH is easier for sysadmin than SSL
  - Erlang already has full SSH capability
    - including SFTP client/server in Erlang
    - OTP ssh_channel module provided

Related works

- Jungerl SSH
  - ancestor of OTP ssh module
  - no longer maintained since 2006
  - not working on current Erlang R13B03
- RPC ideas
  - BERT-RPC: generic RPC through Erlang
    - http://www.bert-rpc.org/
  - SDIST by Dave Smith (of rebar tool)
    - multi-level authentication and security models

SSH protocol overview (as in RFC4251)



SSH Communication Protocol (RFC4254) (1)

- Handling multiple streams of:
  - pseudo ttys (termcap, window size, signals)
  - TCP tunnels (X11/port forwarding)
- Application over SSH works as:
  - shell: interactive shell
  - exec: one-time remote execution (SCP)
  - subsystem: user-named services (SFTP)
  - Erlang ssh module supports all of these

SSH Communication Protocol (RFC4254) (2)

- Maintaining send/receive window
  - Keeping buffer windows for each direction
    - after sending message, window size decreases
    - after receiving acks, window size increases
    - This will prevent flooding without acks
  - Window size is adjustable per request

SSH Transport Protocol (RFC4253)

- Server authentication (Diffie-Hellman)
- Protocol negotiation
  - Transport details
    - shared-key encryption and compression algorithms
    - HMAC for message integrity check
  - Server public-key encryption
- Binary packet format passed on to TCP
- Service requests
  - User authentication / Channel connection

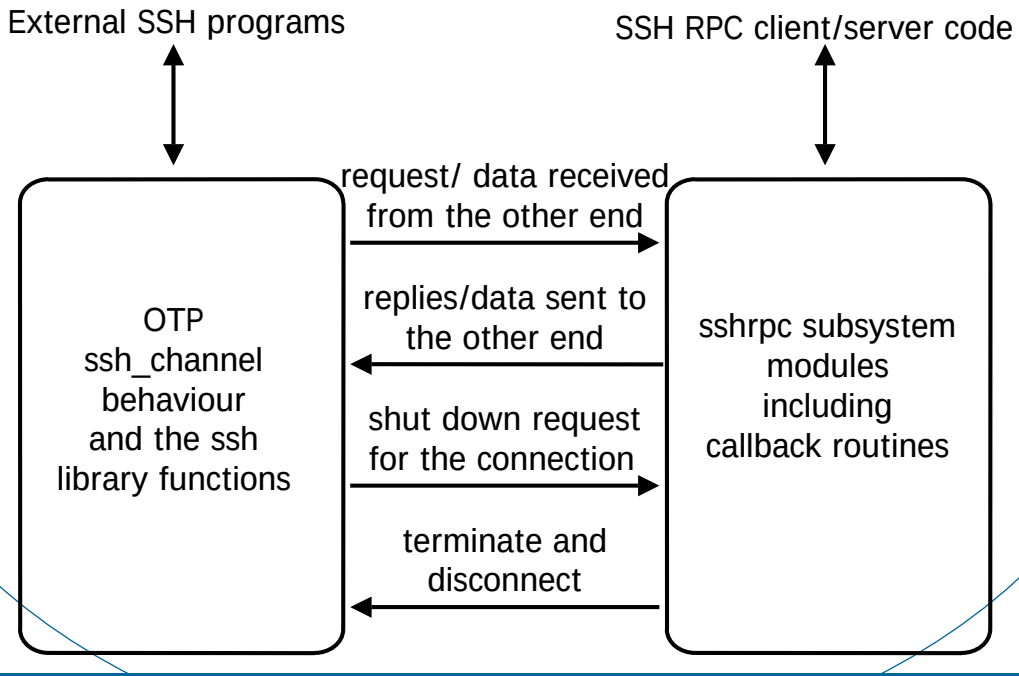
SSH Authentication Protocol (RFC4252)

- User authentication
  - after the SSH transport is established
  - available authentication methods
    - public-key: pre-distributed private and public keys
    - password: conventional password of the host
    - host-based: trusting the host auth (rlogin/rsh)
- Banner message handling

What Erlang/OTP provides

- R13B03 ssh application provides:
  - password and public-key user authentication
    - CAUTION: no password encryption for private keys
  - interactive SSH shell running on a BEAM
  - one-time SSH execution on a BEAM
    - passing a string to the shell as a command
  - frameworks for SSH subsystems
    - example of SFTP client/server code
  - ssh_channel behaviour of OTP programming

interaction between user code and ssh_channel behaviour



Goals of Erlang SSH RPC

- Remote execution of functions
  - Module:Function(Arguments)-style execution
- Asynchronous call handling
  - Exchanged data may be more than single SSH packet; subsystem-level buffering required
- Spawning a remote process
- Sending a message to a running process

Status as of 26-FEB-2009

- Basic server code complete
  - simply passing {M, F, A} to erlang:apply/3
  - multi-packet SSH message can be handled
- Client code: on development
  - Async (event-driven) OTP code required
- See my GitHub repository for the details
  - http://github.com/jj1bdx/sshrpc/