TCP/IP Application Overview

DECUS Europe, Fall 1994
Friday, September 16 1994

Vladimir Sukonnik
Process Software Corporation
sukonnik@process.com

Part IV

Application Level Services and Protocols

Client–Server Paradigm

- Conceptual basis for many distributed applications
- Based on premise that a program makes a request to which another responds
- Related to remote procedure call mechanism

Definitions

- Client
  - Any application program
  - Makes request
  - Awaits response
- Server
  - Specialized program (process)
  - Awaits request
  - Issues response

Illustration Of The Client–Server Paradigm

Client Sends Request

Server Sends Response

Client Side

- Any application program can be a client
- Must know server's Internet address and protocol port number
- Usually easy to build

Server Side

- Can be implemented with application program or in operating system
- Starts execution before requests arrive
- Must ensure client is authorized
- Must uphold protection rules
- Must handle multiple concurrent requests
- Usually complex to design and build

Server Algorithm (Master Copy)

- Open well–known port
- Wait for next client request
- Negotiate new port for this request
- Create slave to handle request
- Continue with wait step

Examples of Standard Servers

- Date and time of day server
- Domain name server
- Electronic mail server
- Remote login server
- File server
- ICMP echo
- UDP echo

Widely Used Application Level Internet Services (the big three)

- Remote login
- Electronic mail
- File transfer

Remote Login

- Allows a user to log onto a remote interactive system
- Follows client–server model

Remote Login (client side)

- User invokes client program on local machine
- Client program
  - Forms connection to server
  - Passes keystrokes to server
  - Displays output from server on terminal

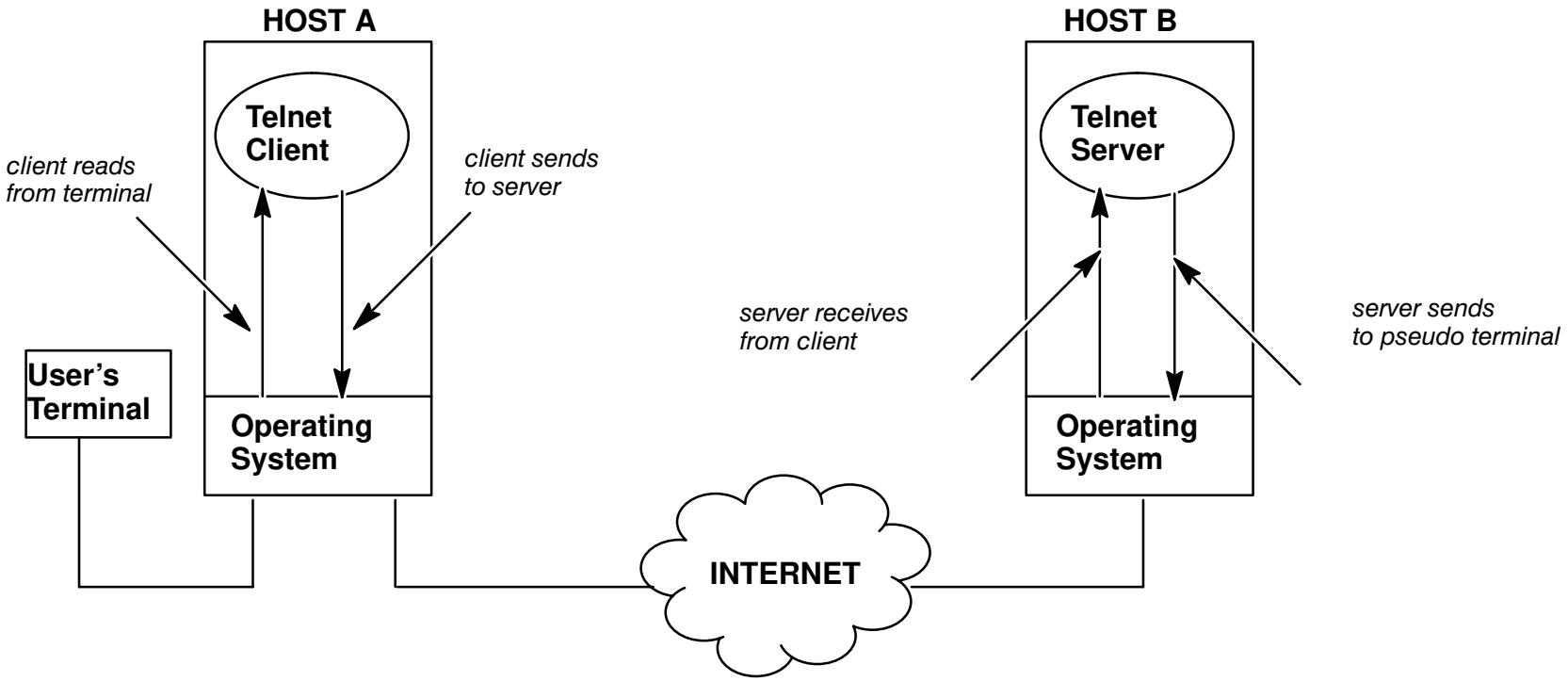
Remote Login (server side)

- Requires operating system primitives that supplies ”psudo terminal” to running program
- Master server invoked at system startup
- Master creates slaves to handle each new connection
- Slave relays characters between connection and psudo terminal

TELNET: Internet Standard Remote Terminal Login Protocol

- Basic facility
- Uses TCP connection to pass data
- Supports options:
  - Character set translation
  - Line terminator
  - Many others

Illustration of TELNET



Rlogin (Unix remote login)

- From 4BSD UNIX
- Passes environment information (e.g. terminal type) to remote machine
- Understands equivalence of login ids
- Preserves UNIX standard input, standard output, standard error
- Nonstandard

Rsh (UNIX Remote Shell)

- From 4BSD UNIX
- Allows remote command execution
- Honors equivalence of login ids
- Preserves UNIX standard input, standard output, standard error
- Nonstandard

Example of Rsh

```
rsh  sigma.process.com ps
```

- Specifies machine sigma in domain process.com
- Executes command ps
- Displays output on user's terminal

File Transfer

- Bulk data transfer
- Two standard protocols exist
  - FTP for large transfer
  - TFTP for simpler tasks
- Follows client–server model

File Transfer Protocol (FTP)

- Defined early in history of Internet
- Heavily used
- Uses TCP for transport
- Provides ability to list directories as well as transfer files

FTP (continued)

- Client can send or receive files
- Third party transfer supported
- Character translation supported
- Basic file formats defined

FTP Client Side

- Makes a TCP connection to well–known port for control
- Obtains authorization by sending a login id and password
- Sends requests to server over control connection

FTP Server Side

- Waits at a well–known port
- Uses TELNET for control connection
- Creates slave to handle each request
- Makes new TCP connection for each transfer
- Enforces file access protections

FTP Example

```
sigma.process.com> ftp lambda
Connected to lambda.process.com.
220 lambda.process.com (192.42.95.75) FTP-OpenVMS FTPD V3.1-5 (c) 1993 Process Software Corporation
Name (lambda:volz): user
331 Password required.
Password:
230 User logged in, proceed.
```

FTP Example (continued)

```
ftp> ls
200 PORT command okay.
150 Opening data connection for *.*.
a1.dir
dates.exe
dates.c
dates.txt
nfs.dir
test.dir
wpsi.dir
226 Closing data connection.
ftp> get dates.c
200 PORT command okay.
150 Opening data connection for SYS$SYSDEVICE:[USER]DATES.C;1.
226 Closing data connection.
ftp> put discardd.c
200 PORT command okay.
150 Opening data connection for SYS$SYSDEVICE:[USER]DISCARDD.C;1.
226 Closing data connection.
ftp> quit
221 Service closing connection.
sigma.process.com>
```

Network File System (NFS)

- Internet standard
- Defined by SUN Microsystems
- Allows one machine to mount files from another in its name-space
- Application program cannot distinguish local and remote files
- Uses UDP or TCP to transfer messages between NFS client and server

Electronic Mail

- Memos delivered from one user to another
- Spooling system accepts outgoing mail and keeps a copy until delivery possible
- Periodic retry (automated)
- Separate standards for mail message formats and mail transfer
- High degree of interoperability

Mail Format (RFC 822)

- Internet standard
- Used by other groups
- Blank line separates header and body
- Specifies headers of the form KEYWORD:information

Mail Format (continued)

- Only a few keywords are important
  - To:
  - From:
- Other keywords are optional
  - Received–by:
  - Reply–to:
- Others are allowed but not interpreted
  - X–VMS–From
  - Message–ID:

Mail Transfer

- Internet standard
- Specifies transfer from machine to machine
- Uses TCP for stream connection

Electronic Mail Example

- User MIKE at machine APOLLO.COM sends a single mail message to three recipients
  - JIM@PROCESS.COM
  - CATHY@PROCESS.COM
  - GEORGE@PROCESS.COM

Electronic Mail Example (continued)

- Mike's mail system stores message in spool area and initiates daemon to transfer message in background
- Daemon uses domain name system to map address PROCESS.COM into Internet address
- Daemon (client) contacts SMTP server at well–known port on machine PROCESS.COM using TCP

Electronic Mail Example (continued)

```
S:220 SIRIUS.PROCESS.COM Simple Mail Transfer Service Ready
C:HELO APOLLO.COM
S:250 sirius.process.com ...
C: MAIL FROM:<MIKE@PROCESS.COM>
S:250 OK
C:RCPT TO:<JIM@SIGMA.PROCESS.COM>
S:250 OK
C:RCPT TO:<CATHY@SIGMA.PROCESS.COM>
S:550 NO SUCH USER
C:RCPT TO:<GEORGE@SIGMA.PROCESS.COM>
S:250 OK
C:DATA
S:354 Start mail input, end with <crlf>.<crlf>
C:This is some of the data
This is some more of the data

.

S:250 OK
C:QUIT
221 sirius.process.com closing connection
```

Electronic Mail Example (continued)

- Client and server both close TCP connection
- Server delivers message to Jim's and George's mailboxes
- Client (daemon) removes messages from spooling area on APOLLO.COM

Internet Management

- Difficult with current Internet
- Implies policy–based routing

Current Internet Management Protocols

- Simple Network Management Protocol (SNMP)
  - Standard application–level program
  - Software exists
  - Allows manager to ask controlling entity about network status and control some parameters
  - Defines how messages pass between manager and controlling entity
  - Does not specify exactly what to monitor

Internet Management Protocols (continued)

- CMIP over TCP (CMOT)
  - Standard OSI network management protocol
  - Currently co–recommended with SNMP (vendor should support one or the other)

Internet Management Protocols (continued)

- Management Information Base (MIB)
  - Companion to SNMP/CMOT
  - Specifies items to be monitored
  - Provides a scheme to name items

Summary

- Most applications use client–server model
- Client may be any application program
- Server is specialized, complex program
- Internet has many application protocols
- Most frequently used applications are electronic mail, remote login, file transfer

Summary (continued)

- Network management protocols reside at application level
- Both SNMP and CMOT network management protocols are corecommended
- Both SNMP and CMOT share a common Management Information Base MIB