Topics

• Issues in Protocol Design

• Sample Application-layer Protocols
  – TELNET
  – FTP
  – DNS
Application Protocol Design

• Think of different people/teams, working on the client and server programs.
  – Different programming languages.
  – Diverse hardware, operating systems.
• Be unambiguous, precise.
  – Consider potential error conditions.
• Allow for future extensions.
  – Leave room for additional data, meta-data.
• Do not replicate services provided by lower-layer protocols
  – e.g., checksum
In Summary

Strive for:

– Interoperability
– Precision
– Extensibility
– Efficiency
– Minimality
Learn by Example

• Many existing protocols are the result of long term collaborations.

• Look at existing Request for Comments (RFC) documents, specifying protocols:
  See http://www.rfc-editor.org/rfc.html
Knock-Knock Protocol

Server: “Knock knock!”
Client: “Who's there?”
Server: “Dexter.”
Client: “Dexter who?”
Server: “Dexter halls with boughs of holly.”
Client: “Groan.”
Java Implementation

• Client class
  – KnockKnockClient.java

• Server class
  – KnockKnockServer.java

• Protocol class
  – KnockKnockProtocol.java
Supporting multiple clients

- Main listener code
  - KKMultiServer.java

- Protocol service thread code
  - KKMultiServerThread.java
The TELNET Protocol

Reference: RFC 854
TELNET vs. telnet

• TELNET is a protocol that provides “a general, bi-directional, eight-bit byte oriented communications facility”.

• telnet is a program that supports the TELNET protocol over TCP.

• Many application protocols are built upon the TELNET protocol.
The TELNET Protocol

- TCP connection
- data and control over the same connection.
- Network Virtual Terminal
- negotiated options
Network Virtual Terminal

- intermediate representation of a generic terminal.
- provides a standard language for communication of terminal control functions.
Network Virtual Terminal

Server Process

TCP

NVT

TCP

NVT

NVT
Negotiated Options

- All NVTs support a minimal set of capabilities.
- Some terminals have more capabilities than the minimal set.
- The 2 endpoints negotiate a set of mutually acceptable options (character set, echo mode, etc).
Negotiated Options

• The protocol for requesting optional features is well defined and includes rules for eliminating possible negotiation “loops”.

• The set of options is not part of the TELNET protocol, so that new terminal features can be incorporated without changing the TELNET protocol.
Option examples

• Line mode vs. character mode
• echo modes
• character set (EBCDIC vs. ASCIIII)
Control Functions

• TELNET includes support for a series of control functions commonly supported by servers.

• This provides a uniform mechanism for communication of (the supported) control functions.
Control Functions

• Interrupt Process (IP)
  – suspend/abort process.

• Abort Output (AO)
  – process can complete, but send no more output to user’s terminal.

• Are You There (AYT)
  – check to see if system is still running.
More Control Functions

• Erase Character (EC)
  – delete last character sent
  – typically used to edit keyboard input.

• Erase Line (EL)
  – delete all input in current line.
Command Structure

• All TELNET commands and data flow through the same TCP connection.
• Commands start with a special character called the Interpret as Command escape character (IAC).
• The IAC code is 255.
• If a 255 is sent as data - it must be followed by another 255.
Looking for Commands

• Each receiver must look at each byte that arrives and look for IAC.
• If IAC is found and the next byte is IAC - a single byte is presented to the application/terminal (a 255).
• If IAC is followed by any other code - the TELNET layer interprets this as a command.
Command Codes

- IP  243  WILL  251
- AO  244  WON’T  252
- AYT 245  DO  253
- EC  246  DON’T  254
- EL  247  IAC  255
Playing with TELNET

• You can use the `telnet` program to play with the TELNET protocol.
• `telnet` is a *generic* TCP client.
  – Sends whatever you type to the TCP socket.
  – Prints whatever comes back through the TCP socket.
  – Useful for testing TCP servers (ASCII based protocols).
Some TCP Servers you can play with

- Many Unix systems have these servers running (by default):
  - echo        port 7
  - discard     port 9
  - daytime     port 13
  - chargen     port 19
telnet hostname port
> telnet rcs.rpi.edu 7
Trying 128.113.113.33...
Connected to cortez.sss.rpi.edu (128.113.113.33).
Escape character is '^[].
Hi dave
Hi dave
stop it
stop it
^[]}
telnet> quit
Connection closed
**telnet vs. TCP**

- Not all TCP servers talk TELNET (most don't)

- You can use the `telnet` program to play with these servers, but the fancy commands won't do anything.
  - type `^]`, then "help" for a list of fancy TELNET stuff you can do in `telnet`.

- See `GenericClient.java`
FTP
File Transfer Protocol

Reference:
RFC 959
FTP Objectives
(from RFC 959)

• promote sharing of files
• encourage indirect use of remote computers
• shield user from variations in file storage
• transfer data reliably and efficiently
• “FTP, although usable directly by a user at a terminal, is designed mainly for use by programs”
The FTP Model

PI: Protocol Interpreter
DTP: Data Transfer Protocol

File System
Server PI
Server DTP

User Interface
User PI
User DTP

File System

Data
Control
Control and Data Connections

• Control functions (commands) and reply codes are transferred over the control connection.
• All data transfer takes place over the data connection.
• The control connection must be “up” while data transfer takes place.
Control Connection

• The control connection is the “well known” service.
• The control connection uses the TELNET protocol.
• Commands and replies are all line oriented text (default is ASCII).
Standard Connection Model

A \rightarrow B

\text{Control}

\text{Data}

B \rightarrow A
Alternative Connection Model

Control\[\rightarrow\] A \[\rightarrow\] Control

Data

B \[\rightarrow\] \[\rightarrow\] C
Access Control Commands

USER  
specify user

PASS  
specify password

CWD   
change directory

CDUP  
change directory to parent

QUIT  
logout
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT</td>
<td><em>publish local data port</em></td>
</tr>
<tr>
<td>PASV</td>
<td><em>server should listen</em></td>
</tr>
<tr>
<td>TYPE</td>
<td><em>establish data representation</em></td>
</tr>
<tr>
<td>MODE</td>
<td><em>establish transfer mode</em></td>
</tr>
<tr>
<td>STRU</td>
<td><em>establish file structure</em></td>
</tr>
</tbody>
</table>
Service Commands

RETR  retrieve file
STOR  send file
STOU  send file and save as unique
APPE  send file and append
ABOR  abort prev. service command
PWD   print working directory
LIST  transfer list of files over data link
FTP Replies

• All replies are sent over control connection.
• Replies are a single line containing
  – 3 digit status code (sent as 3 numeric chars).
  – text message.
• The FTP spec. includes support for multiline text replies.
FTP Reply Status Code

First digit of status code indicates type of reply:

‘1’: Positive Preliminary Reply (got it, but wait).
‘2’: Positive Completion Reply (success).
‘3’: Positive Intermediate Reply (waiting for more information).
‘4’: Transient Negative Completion (error - try again).
‘5’: Permanent Negative Reply (error - can’t do).
FTP Reply Status Code

- 2nd digit indicates function groupings.
  - ‘0’: Syntax (problem with command syntax).
  - ‘1’: Information (reply to help or status cmds).
  - ‘2’: Connections (problem with a connection).
  - ‘3’: Authentication (problem with login).
  - ‘4’: Unspecified.
  - ‘5’: File system (related to file system).

- 3rd digit indicates specific problem within function group.
Data Transfer Modes

• STREAM: file is transmitted as a stream of bytes.
• BLOCK: file is transmitted as a series of blocks preceded by headers containing count and descriptor code (EOF, EOR, restart marker).
• COMPRESSED: uses a simple compression scheme - compressed blocks are transmitted.
RFC 959

• The RFC includes lots more information and many details including:
  – parameters for commands
  – lists of reply status codes
  – protocol state diagrams
  – support for a variety of file structures
  – sample sessions
Address Conversion Functions
and
The Domain Name System

Based on Notes by D. Hollinger
Refs: UNIX Network Programming, Stevens, Chapter 9
RFC 1034
RFC 1035
Also based on Java Network Programming and
Distributed Computing, Chapter 3
Hostnames

• IP Addresses are great for computers
  – IP address includes information used for routing.
• IP addresses are tough for humans to remember.
• IP addresses are impossible to guess.
  – ever guessed at the name of a WWW site?
The Domain Name System

• The *domain name system* is usually used to translate a host name into an IP address.

• Domain names comprise a hierarchy so that names are unique, yet easy to remember.
DNS Hierarchy
Host name structure

• Each host name is made up of a sequence of *labels* separated by periods.
  – Each label can be up to 63 characters
  – The total name can be at most 255 characters.

• Examples:
  – whitehouse.gov
  – barney.the.purple.dinosaur.com
  – monica.cs.rpi.edu
Domain Name

- The domain name for a host is the sequence of labels that lead from the host (leaf node in the naming tree) to the top of the worldwide naming tree.

- A domain is a subtree of the worldwide naming tree.
Top level domains

- edu, gov, com, net, org, mil, ...
- Countries each have a top level domain (2 letter domain name).
- New top level domains include:
  .aero .biz .coop .info .name .pro
DNS Organization

• Distributed Database
  – The organization that owns a domain name is responsible for running a DNS server that can provide the mapping between hostnames within the domain to IP addresses.
  – So - some machine run by RPI is responsible for everything within the rpi.edu domain.
DNS Distributed Database

- There is one primary server for a domain, and typically a number of secondary servers containing replicated databases.
DNS Clients

• A DNS client is called a resolver.

• A call to `getByName(host)` is handled by a resolver (typically part of the client).

• Most Unix workstations have the file `/etc/resolv.conf` that contains the local domain and the addresses of DNS servers for that domain.
/etc/resolv.conf

domain rpi.edu
128.113.1.5
128.113.1.3
nslookup

• nslookup is an interactive resolver that allows the user to communicate directly with a DNS server.

• nslookup is usually available on Unix workstations.
$ nslookup
Default Server: oldtotter.cs.rpi.edu
Address: 128.213.8.12

> rpi.edu
Server: oldtotter.cs.rpi.edu
Address: 128.213.8.12

Non-authoritative answer:
Name: rpi.edu
DNS Servers

• Servers handle requests for their domain directly.
• Servers handle requests for other domains by contacting remote DNS server(s).
• Servers cache external mappings.
Server - Server Communication

• If a server is asked to provide the mapping for a host outside it’s domain (and the mapping is not in the server cache):
  – The server finds a nameserver for the target domain.
  – The server asks the nameserver to provide the host name to IP translation.

• To find the right nameserver, use DNS!
DNS Data

- DNS databases contain more than just hostname-to-address records:
  - Name server records  NS
  - Hostname aliases    CNAME
  - Mail Exchangers     MX
  - Host Information    HINFO
The Root DNS Server

• The root server needs to know the address of 1st (and many 2nd) level domain nameservers.

```
edu  com  org  jp
  /    /    /    /
 rpi  albany
```
Server Operation

- If a server has no clue about where to find the address for a hostname, ask the root server.
- The root server will tell you what nameserver to contact.
- A request may get forwarded a few times.
DNS Message Format

<table>
<thead>
<tr>
<th>HEADER</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERIES</td>
</tr>
<tr>
<td><strong>Response</strong> RESOURCE RECORDS</td>
</tr>
<tr>
<td><strong>Response</strong> AUTHORITY RECORDS</td>
</tr>
<tr>
<td><strong>Response</strong> ADDITIONAL INFORMATION</td>
</tr>
</tbody>
</table>
DNS Message Header

- query identifier
- flags
- # of questions
- # of RRs
- # of authority RRs
- # of additional RRs

16 bit fields
Message Flags

- QR: Query=0, Response=1
- AA: Authoritative Answer
- TC: response truncated (> 512 bytes)
- RD: recursion desired
- RA: recursion available
- rcode: return code
Recursion

• A request can indicate that recursion is desired - this tells the server to find out the answer (possibly by contacting other servers).

• If recursion is not requested - the response may be a list of other name servers to contact.
Question Format

• Name: domain name (or IP address)

• Query type (A, NS, MX, …)

• Query class (1 for IP)
Response Resource Record

- Domain Name
- Response type
- Class (IP)
- Time to live (in seconds)
- Length of resource data
- Resource data
UDP & TCP

• Both UDP and TCP are used:
  – TCP for transfers of entire database to secondary servers (replication).
  – UDP for lookups
  – If more than 512 bytes in response - requestor resubmits request using TCP.
Lots more

- This is not a complete description!
- If interested - look at:
  - RFC 1034: DNS concepts and facilities.
  - RFC 1035: DNS implementation and protocol specification.
  - play with nslookup.
  - Look at code for BIND (DNS server code).
Internet Addresses in Java

- `java.net.InetAddress` class

- You get an address by using static methods:

  ```java
  ad = InetAddress.getByName(hostname);
  myAddress = InetAddress.getLocalHost();
  ```
Printing Internet Addresses

• You get information from an InetAddress by using methods:

ad.getHostName();
ad.getHostAddress();

• Both return Strings representing the host name, and the IP address in dotted decimal format.
Additional InetAddress methods

- `getAddress()` returns the IP address.
  - in byte array format (network byte order), with highest byte at `bytearray[0]`.

- `getAllByName(hostname)` returns an array of InetAddress instances for the given host name.
  - One host name may be mapped to multiple machines.
  - One host name can map to multiple addresses in the same machine (virtual addresses).
Additional InetAddress methods

- `isMulticastAddress()` returns a boolean representing whether it is a Class D address.

- `getAllByName(hostname)` returns an array of InetAddress instances for the given host name.
  - One host name may be mapped to multiple machines.
  - One host name can map to multiple addresses in the same machine (virtual addresses).