TCP/IP Programming

- Joel Snyder, Opus1
- Geoff Bryant, Process Software
Course Roadmap

- NM055 (11:00-12:00) Important Terms and Concepts
  - TCP/IP and Client/Server Model
  - Sockets and TLI
  - Client/Server in TCP/IP
- NM056 (1:00-2:00) Socket Routines
- NM057 (2:00-3:00) Library Routines
- NM058 (3:00-4:00) Sample Client/Server
- NM059 (4:00-5:00) VMS specifics (QIOs)
- NM067 (6:00-7:00) Clinic - Q&A
TCP/IP Programming

Slides and Source Code available via anonymous FTP:

Host:: ftp.process.com
Directory: [pub.decus]
Slides: DECUS_F96_PROG.PS
Examples: DECUS_F96_PROG_EXAMPLES.TXT

Host: ftp.opus1.com
Slides: DECUS_F96_PROG.PS
Examples: DECUS_F96_PROG_EXAMPLES.TXT
Session NM055

TCP/IP Programming
Terms and Concepts

Joel Snyder
Opus1
Terms and Concepts Overview

- What is the TCP/IP model?
- What is the client/server model?
- What are sockets and TLI?
- What is network byte order?
- What is encapsulation? Multiplexing? Demultiplexing? Fragmentation?
- What are addresses?
Networks are layered to simplify construction

---

The Famous OSI 7-Story Apartment Building with attached Parking Garage
## The OSI Model (similar to the Holy Grail)

<table>
<thead>
<tr>
<th>OSI layer</th>
<th>Function provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Network applications such as file transfer and terminal emulation</td>
</tr>
<tr>
<td>Presentation</td>
<td>Data formatting and encryption</td>
</tr>
<tr>
<td>Session</td>
<td>Establishment and maintenance of sessions</td>
</tr>
<tr>
<td>Transport</td>
<td>Provision for end-to-end reliable and unreliable delivery</td>
</tr>
<tr>
<td>Network</td>
<td>Delivery of packets of information, which includes routing</td>
</tr>
<tr>
<td>Data Link</td>
<td>Transfer of units of information, framing, and error checking</td>
</tr>
<tr>
<td>Physical</td>
<td>Transmission of binary data of a medium</td>
</tr>
</tbody>
</table>
TCP/IP is (usually) sliced up into four layers.

There are actually more layers to TCP/IP than this set of four, but from the programming point of view, this will do just fine.
We care most about the two center layers

Application

Transport

TCP (Transmission Control Protocol)

UDP (User Datagram Protocol)

Network

IP (Internet Protocol)

Physical/Dat a Link

hardware interface
TCP

- Transmission Control Protocol is defined by RFC-793
- TCP provides connection-oriented transport service
- End-to-end transparent byte-stream
UDP

- User Datagram Protocol is defined by RFC-768
- UDP provides datagram service
- Connectionless
Client/Server is application-to-application

- TCP/IP and DECnet are client/server networks
- A client/server application has two parts
  - One which runs on one side of the network
  - One which runs on the other side of the network
- Differentiate this from a terminal-based network or most Netware applications
Clients and Servers use IPC to talk to each other

- Inter-Process Communication mechanisms let two cooperating applications communicate
- There are **LOTS** of IPC mechanisms for local communications
- The two popular TCP/IP based IPCs are Sockets (Berkeley Unix) and TLI (AT&T Unix System V)
Sockets is the standard API for TCP/IP IPC

- Normally, you’d have an operating system specific routine set to talk to the network
- In the world of standardized APIs, you would have an operating system independent set of routines
- Sockets takes it one step further: it makes the network look much like a file system
TCP/IP is a file system?

Yes, sockets treats TCP/IP as if it were a file system (with some added complexity in the open and close routines)

- Routines to open up a file
- Routines to open up a connection
- Routines to close an open file
- Routines to close an open connection

Read

Write
It looks like a file, but it’s really a network interface

Application

write()

read()

Application

read()

write()
Life was easy when machines had 8-bit words

- Welcome to the concept of “network byte order”
- Remember that “network byte order” is not the same as “host byte order”
- You have to convert to network byte order!
There are two ways to store multi-byte integers

**big endian**

- high order byte
- low order byte

Address n  Address n+1

**little endian**

- high order byte
- low order byte

Address n+1  Address n
VMS systems are little endian architectures

<table>
<thead>
<tr>
<th>Big-endian architectures</th>
<th>Little-endian architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorola 68xxx</td>
<td>Intel 80x86</td>
</tr>
<tr>
<td>IBM 370</td>
<td>VAX &amp; Alpha (VMS)</td>
</tr>
<tr>
<td></td>
<td>PDP-11 (sort of)</td>
</tr>
</tbody>
</table>
Relax: you could be doing Ethernet and Token Ring

Bits within bytes are transmitted left to right.
Encapsulation adds control information to data

<table>
<thead>
<tr>
<th>Network headers</th>
<th>Data field</th>
<th>Network trailers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data from user program such as an application program or session control information</td>
<td></td>
</tr>
</tbody>
</table>

Request

- Headers
- Request block of data
- Trailers

Response

- Headers
- Now is the time for all good men to come to the aid of their country
- Trailers
- men to come to the aid of
- Headers

Response

- Headers
- of their country
- Trailers

File server

Client workstation

Now is the time for all good men to come to the aid of their country
Each layer encapsulates the layer below it.

OSI model:

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

Network packet:

- Data link header
- Network layer header
- Transport layer header
- Session layer header
- Presentation and Application layer header
- Data link trailers
Things can be added on and peeled off at the ends

Look at your average WWW query
Make sure you memorize all of this encapsulation info

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ethernet V2.0</strong></td>
<td>6 bytes Destination address</td>
</tr>
<tr>
<td><strong>IEEE 802.3</strong></td>
<td>6 bytes Destination address</td>
</tr>
<tr>
<td><strong>IEEE 802.3 with IEEE 802.2</strong></td>
<td>6 bytes Destination address</td>
</tr>
<tr>
<td><strong>IEEE 802.3 SNAP</strong></td>
<td>6 bytes Destination address</td>
</tr>
<tr>
<td><strong>Novell proprietary</strong></td>
<td>6 bytes Destination address</td>
</tr>
<tr>
<td><strong>Token Ring</strong></td>
<td>6 bytes Destination address</td>
</tr>
<tr>
<td><strong>FDDI</strong></td>
<td>6 bytes Destination address</td>
</tr>
</tbody>
</table>
Multiplexing and Demultiplexing

- Application Process
- TCP
- Ethernet interface

- Application Process
- UDP
- Ethernet interface

- Application Process
- IP

- IPX

- DECnet
Segmentation and Reassembly

Application Data Message

TCP

TCP Segments

IP

TCP

IP Packets
Addressing identifies entities in the network

- Each layer in a TCP/IP stack has an address made up of two parts
  - The address of everything below it
  - The address of itself
- In some cases, lower-level addresses are implied and do not have to be stated
  - For example, Ethernet MAC addresses are tied directly to IP addresses, so they can be omitted
Three addresses are important: IP, Protocol, Port

- Application-level addresses
- Transport-level addresses
- Network-level addresses
IP addresses are 32-bit integers

IP Address: 192.245.12.2

Application-level addresses
Transport-level addresses
Network-level addresses
Transport addresses are 8-bit protocol numbers

- Protocol Number:
  - TCP = 6
  - UDP = 17
Port Numbers are 16-bit integers

Port Number: SMTP = 25
The triplet (IP, Proto, Port) identifies a process.

- **Protocol number** (transport address)
- **IP number** (network address)
- **Port number** (application address)
Two triplets identify a connection

- Now we have a socket!
- (sometimes called a “socket pair”)

Application number 1117
on protocol number 6 (TCP)
at IP address 192.245.12.2

Application number 25
on protocol number 6 (TCP)
at IP address 128.196.128.233
A socket maps one application to another.
Addresses are assigned by different entities

- IP addresses are assigned by the InterNIC and are managed by the local network manager.
- Protocol numbers are assigned by the IANA (Internet Assigned Number Authority) and are fixed across the Internet (RFC 1700).
- Application port numbers come from two sources:
  - Servers are assigned by IANA (RFC 1700).
  - Clients are handled by the network kernel.
IP Addressing

- IP is the Internet Protocol, currently version 4
- IPNG is the Next Generation (also known as IPv6)
- IP is defined by RFC-791
- IP uses four octet (8-bit byte) addresses
- IP takes care of getting packets to destination
Client and Server Port Numbers are coordinated

- **NORMALLY**, Server port numbers are low numbers in the range 1-1023
  - Defined in RFC 1700
- **NORMALLY**, Client port numbers are high numbers starting at 1024
  - These are called “ephemeral ports”
You only pick server ports

- A **server** running on a “well-known port” lets the operating system know what port it wants to listen on.
- A **client** simply lets the operating system pick a new port (ephemeral) that isn’t already in use.
Well-Known-Servers

- Public services (e.g. and email server) are assigned a particular number by IANA
- These numbers are stored in the Internet Assigned Numbers RFC (changing number, latest is 1797)
- These are called Well-Known-Servers
- Examples of these include TELNET (23), SMTP (25), FINGER (79), HTTP (80), RLOGIN (512) and others
Important Terms
Key Concepts

- TCP/IP networks are layered
- TCP/IP uses a client/server paradigm
- Addressing triples (IP, protocol, port) identify applications in TCP/IP
- A pair of triples beats a full house
Client/Server in TCP/IP
Client/Server in TCP/IP
Overview

- What is the server process (INETD)?
- What are ephemeral ports?
- How can you have multiple connections?
- How are processes identified?
- What is Connection-oriented? Connection-less?
Making the initial connection to a server

- When a connection enters a TCP/IP system, someone has to handle it
  - Either a running daemon is waiting (your application)
  - or a running daemon is waiting (InetD)
- Trade off efficiency for performance
  - Choose whichever model you want based on individual application characteristics
  - Seldom used? Inetd
  - Constantly used? True Daemon
Your application can wait for an incoming connection

- This is called the “Daemon” (or Demon) approach

If anything should come in for Port 25 on TCP, give it to me. I will be waiting right here.

OK.
Or you can have the InetD do it for you

- In Multinet, InetD is known as the Master Server
- In TCPware, InetD is part of NETCP
- In UCX, InetD is the Auxiliary Server

Port 25? I will start a new process running Your Application and hand it this incoming call.
Servers low, Clients high

Servers: ports 1-1023

Clients: “ephemeral ports” above 1023
A Server may connect to multiple clients

- Each connection is a pair of triples (IP, protocol, port)
- The server side may remain the same across multiple clients
- Of course, the server programmer has to keep it all straight
Socket-pairs leave no ambiguity between 2 clients

Client

1117

TCP

IP

Ethernet interface

192.245.12.2

Client

5555

TCP

IP

Ethernet interface

195.240.3.1

SERVER

25

TCP

IP

Ethernet interface

128.196.128.233
Applications are known by their port numbers

- A process is identified by its 16-bit port number
- If a server uses both TCP and UDP, it will often use the same port number for both protocols
  - Protocol number means no ambiguity
- The concept of low-numbered ports as “privileged” disappeared with PCs
  - Don’t base your security on port numbers!
- Some services use a meet-me approach
TCP/IP supports CL and CO at transport layer

- CO = Connection Oriented
  - TCP
  - “Stream”
- CL = Connectionless
  - UDP
  - “Datagram”
- CL can be provided by pure IP
  - Unusual
  - “Raw”
Connection oriented is like a phone call

- Connection oriented data communications arrived before connectionless
  - Used primarily over noisy serial lines in original intention.
    - not necessarily an issue with TCP
  - Two stations must establish a connection before data is transmitted.
  - Connection is strictly maintained using sequence numbers, acknowledgments, retries and so on.
Connect, Transfer, Disconnect

Note: This is almost like how it's done in TCP. Take the internals course for more info.
Connectionless is like a postcard

- Connectionless allows data to be transmitted without a pre-established connection between two stations.
  - This type of service flourished with the proliferation of LANs.
    - LANs tend to have a very low error rate and a connection need not be established to ensure the integrity of the data.
  - This type of service does not provide error recovery, flow or congestion control.
    - Upper layer network protocols can accomplish this.
  - It requires less overhead and is implicitly faster.
Nike style data communications

Data
Data
Data
Data

Note: This is exactly like how it’s done in UDP. Take the internals course for more info.
Client/Server in TCP/IP

Key Concepts

- Servers are handled in one of two ways: resident daemons or InetD
- Server ports are low-numbers; client ports are high-numbered ephemeral ports
- A server can talk to many clients if the programmer can keep them straight
- TCP is CO; UDP is CL