Socket Programming and UNIX Networking Tools

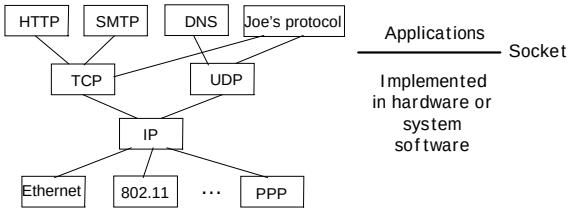
Kai Shen

Dept. of Computer Science, University of Rochester

Course Outline

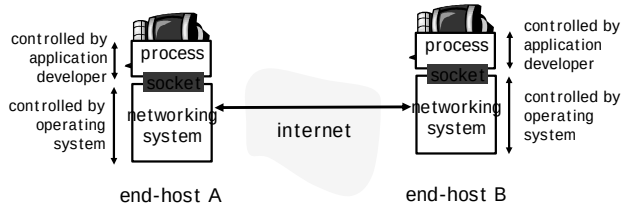
- Traditional materials about computer networks
  - Data links: Ethernet etc.
  - Network: IP addressing and routing.
  - Transport: TCP/UDP.
  - Application: HTTP, FTP, SMTP, and DNS.
- Advanced topics in computer networks
  - Multimedia networking (quality of service), computer security, wireless networks, overlay networks.
- Case studies of emerging network systems/technologies
  - HTTP switching (Foundry networks).
  - Network caching (Inktomi).
  - Content distribution (Akamai).
  - Peer-to-peer systems (Gnutella).

Socket Programming



Socket: the interface between application processes and end-to-end transport protocol (TCP or UDP)

Socket Programming (cont.)



Our goal: learn how to build network applications that communicate using sockets.

Socket API

socket

a host-local, application-created, OS-controlled interface (a "door") into which application process can send and receive data to/from another remote application process

- Introduced in BSD 4.1 UNIX, 1981
- Client-server paradigm
- Two types of transport services via socket API:
  - connection-oriented byte stream (TCP)
  - connectionless datagram (UDP)

Port Numbers

- Multiple sockets might exist in each host.
- A port number identifies each such socket in each host.
- Each port number is a 16-bit number, ranging from 0 to 65535.
- Port numbers ranging from 0 to 1023 are called well-known port numbers and are restricted.

Socket Programming with TCP

At the server:

- server must have created socket (door) that welcomes client's contact
- server process must first be running

Client contacts server by:

- creating client-local TCP socket
- specifying IP address, port number of the server socket
- When client creates socket: client TCP establishes connection to server TCP

- When contacted by client, server TCP creates a new connection socket for communicating with the client
  - allows server to talk with multiple clients

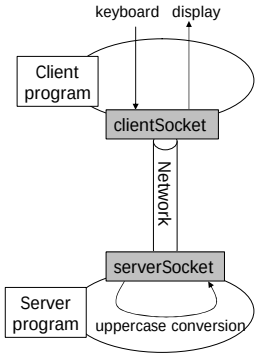
application viewpoint

TCP provides connection-oriented byte stream between client and server

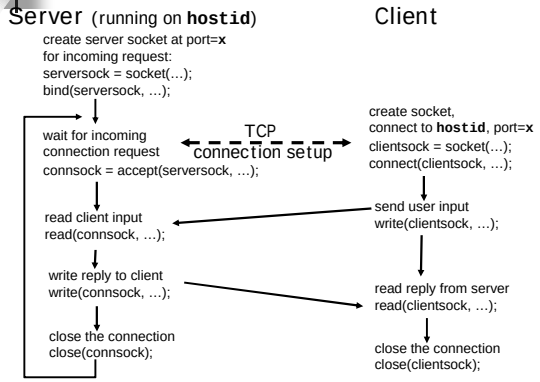
Client/Server TCP Socket Interaction: An Example

Example client-server app:

1) client reads line from standard input, sends to server via socket
2) server reads line from socket
3) server converts line to uppercase and sends it back to client
4) client reads and prints modified line from socket



Client/Server TCP Socket Interaction: Pseudo-code



Socket Programming with UDP

UDP: connectionless

- no handshaking
- sender explicitly attaches IP address and port of destination to each packet

UDP: datagram-oriented

- clear boundaries between groups of bytes
- no ordering among datagrams

application viewpoint

UDP provides transfer of groups of bytes ("datagrams") between client and server

TCP versus UDP

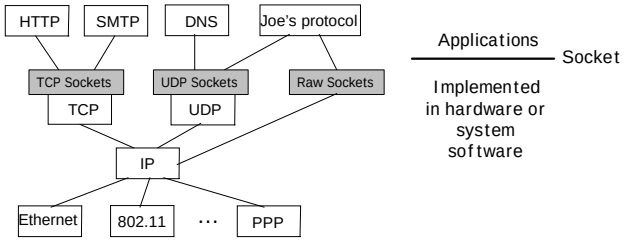
- Connection-oriented vs. connectionless
- Stream-oriented vs. datagram-oriented

- reliable data transfer
- flow control and congestion control

Basic Socket API in C

- Create a socket
  int socket(int domain, int type, int protocol);
- Passively open a socket connection (server-side)
  int bind(int socket, struct sockaddr *addr, int addr_len);
  int listen(int socket, int backlog);
  int accept(int socket, struct sockaddr *addr, int addr_len);
- Actively open a socket connection (client-side)
  int connect(int socket, struct sockaddr *addr, int addr_len);
- Send/receive messages
  int send(int socket, char *msg, int mlen, int flags);
  int recv(int socket, char *buf, int blen, int flags);
- Close a socket connection
  int close(int socket);

Raw Sockets



Socket: the interface between application process and the network system software.

References/Tutorials on Socket Programming

C-language:

- Reference: UNIX manual pages.
- Reference and tutorial: Stevens, "Unix Network Programming, Volume I", 2nd edition, 1998.

Java-language:

- Reference: http://java.sun.com/products/jdk/1.2/docs/api/
- Tutorial: http://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.html

Network Performance Metrics

- Bandwidth (throughput)
  - amount of data transmitted per time unit – Mbps
- Latency (delay)
  - time for a zero-sized message to travel from host A to host B
  - roundtrip latency
- Transmission delay for a size-k message
  - latency + K/bandwidth
- Latency is important for small messages while bandwidth is more important for large messages.

Assignment #0 – Warm-up on Socket Programming

- Will not be graded. You are advised to finish it by this Sunday, when assignment #1 will be given.
- Write simple TCP/UDP programs in C or Java.
- The server reads an input text string from any client, modifies it slightly and returns back to the client.
- The client reads an input string from keyboard, sends it to the server. After the server responds, it displays the response.

Assignment #0 – Advanced Work

- What is the biggest sized UDP datagram you can send out?
- Measure the round-trip transmission delay of a UDP datagram between two machines.
  - Provide measurement results for the following UDP datagram sizes: 1-byte, 10-byte, 100-byte, 1,000-byte, 10,000-byte.
- Can you also come up the link bandwidth with the data on transmission delay?
- Repeat the measurements for TCP.

UNIX Networking Tools

- ping – telling if a server is alive
  ping tallinn.cs.rochester.edu
- traceroute – find intermediate nodes to a destination
  traceroute booboo.cs.ucsb.edu
- nslookup – query domain name servers (DNS); find the IP address of a machine
  nslookup tallinn.cs.rochester.edu
- netstat – show various network status of a host; find active TCP/UDP sockets and their states
  netstat -a

UNIX Networking Tools (cont.)

Find the number of active sockets for a particular process through /proc filesystem in Linux:

- find the process ID of the process you are interested in (through ps), e.g. 20417
- "cd /proc/20417/fd"
- "ls –l"
- "/proc/20417/" contains runtime information about this process

Disclaimer

- Parts of the lecture slides contain original work of James Kurose, Larry Peterson, and Keith Ross. The slides are intended for the sole purpose of instruction of computer networks at the University of Rochester. All copyrighted materials belong to their original owner(s).