8 Networking

The Telephone Network

- Telephone network.
  - Circuit switching: single physical path between sender and receiver is dedicated for duration of communication.
  - Has worked for 100 years.
  - Advantage: real-time communication.

Internet

- Global communication network containing millions of computers.
- Computer networks communicate using TCP/IP protocol.
- Provides access to services: email, chat, world wide web, KaZaa.
- Started by military around 1969 as ARPANET: survivability, robustness, efficiency.
- Operating system and hardware independent.

Everybody but you grew up without it!
Internet protocol (IP)
- Rules for moving bits from A to B.
- Divide data into packets (header bits to say where to go, data bits) and transmit each individually, possibly along different paths.
- No guarantee packets arrive in order, or even arrive.

Transmission control protocol (TCP)
- Rules to provide reliable communication between two computers.
- Packets arrive, and they arrive in order.
  - resend over IP until recipient acknowledges.

SMTP
- Simple Mail Transfer Protocol (SMTP).
- Set of rules for sending email.
- Server smtp.princeton.edu waits for connection requests on port 25.
- User specifies sender, recipient, subject, message body, etc.

Transmission control protocol (TCP)
- Rules to provide reliable communication between two computers.
- Packets arrive, and they arrive in order.
  - resend over IP until recipient acknowledges.

TCP/IP
- Internet protocol (IP)
  - Rules for moving bits from A to B.
  - Divide data into packets (header bits to say where to go, data bits) and transmit each individually, possibly along different paths.
  - No guarantee packets arrive in order, or even arrive.

TCP
- Transmission control protocol (TCP)
  - Rules to provide reliable communication between two computers.
  - Packets arrive, and they arrive in order.

SMTP
- Simple Mail Transfer Protocol (SMTP).
- Set of rules for sending email.
- Server smtp.princeton.edu waits for connection requests on port 25.
- User specifies sender, recipient, subject, message body, etc.

Java Client: SMTP

A Java program that uses SMTP.

```java
import java.net.Socket;

public static void main(String[] args) throws Exception {
    Socket socket = new Socket("smtp.princeton.edu", 25); // open socket
    In in = new In(socket); // acquire input
    Out out = new Out(socket); // acquire output
    server = new Socket(); // open server socket
    socket.connect(); // connect server socket
    socket.setBlocking(true); // block on socket
    socket.send(); // send message
    socket.close(); // close socket
}
```

Protocols
- Many higher layer application protocols use TCP/IP.
  - Used by http, smtp, telnet, ftp.

Ex 1: HyperText Transfer Protocol (HTTP)
- Set of rules for transferring files (text, graphics, video).
- Server java.sun.com waits for connection requests.
- Browser issues GET and POST commands.
- Server responds with header and body.

Ex 2: Simple Mail Transfer Protocol (SMTP)
- Set of rules for sending email.
- Server smtp.princeton.edu waits for connection requests on port 25.
- User specifies sender, recipient, subject, message body, etc.

Open relay. smtp server accessible from anywhere.

Warning: do not spoof without recipient's consent.
**Echo Server**

Echo server: use `ServerSocket` to listen for connection requests; connect with a client; read text that client sends; and send it back.

```
public class EchoServer {
    public static void main(String[] args) throws Exception {
        ServerSocket serverSocket = new ServerSocket(4444);
        while (true) {
            Socket socket = serverSocket.accept();
            In in = new In(socket);
            Out out = new Out(socket);
            String s;
            while ((s = in.readLine()) != null) {
                out.println(s);
                System.out.println(in.readLine());
            }
            out.close();
            in.close();
            socket.close();
        }
    }
}
```

**Remote Control Server**

Remote control server.
- Client sends text commands.
- Server launches: { TiVo recorder, coffee maker, burglar alarm }.
- Server acknowledges task completion.

Trivial modification to Java programs.
- Detail: must be running Java coffee maker on Internet.
Chatroom client.
- Each client requests a connection to server.
- Client sends message to server.
- Server broadcasts message to ALL connected clients.

Echo + a few details.
- Graphical user interface (event-based programming).
- Server must process several simultaneous connections (threads).

```
\java ChatClient alice bicycle.cs.princeton.edu
```

Multi-Threaded Chatroom Application

Graphical user interface.
- Interact with computer using mouse and keyboard.
- Buttons, menus, scrollbars, toolbars, file choosers.
- Ex: Mac, Windows, KDE.

Event-based programming.
- Program remains in infinite loop, waiting for event.
- Upon event, program executes some code.

Callback.
- Programmer registers an event to listen for.
- Programmer writes code to process event.

```
public class ChatClient extends JFrame implements ActionListener {

    // listener for user entering input
    private JTextArea enteredText = new JTextArea(10, 32);
    private JTextField typedText = new JTextField(32);

    public ChatClient (String screenName, String hostName) {
        // layout placement of typed text
    }

    public ChatClient (String screenName, String hostName) {
        // register event
        Container c = getContentPane();
        c.add(enteredText, BorderLayout.CENTER);
        c.add(typedText, BorderLayout.SOUTH);
        pack();
        show();
    }

    public void actionPerformed(ActionEvent e) {
        // callback
        out.println(typedText.getText());
        typedText.setText('');
    }
}
```

Multi-Threaded Server

Threads of control.
- "Illusion" that several things are happening at once in a program.
  - similar idea used by OS to execute several programs at once
  - parallel computers make illusion a reality
- Timesharing: CPU processes each thread one for a fraction of a second before moving on to the next one.

Multi-threaded server.
- Server listens for connection requests in one thread.
- Server handles communication with each client in a separate thread.
- Makes server scalable - can accept requests, independent of speed in handling them.

Java has built-in support for threads.
Java Multi-Threaded Server: ChatServer

Chat server:
- The listener thread broadcasts messages to clients.
- Each client thread communicates with one client.

```java
import java.net.Socket;
import java.net.ServerSocket;
import java.util.ArrayList;

public class ChatServer {
    public static void main(String[] args) throws Exception {
        ArrayList clients = new ArrayList();
        ServerSocket serverSocket = new ServerSocket(4444);
        ClientListener listener = new ClientListener(clients);
        listener.start(); ← start thread to broadcast message
        while (true) {
            Socket socket = serverSocket.accept(); ← wait for client
            Client client = new Client(socket); ← connection request
            clients.add(client);
            client.start();
        }
    }
}
```

Listener Thread: Broadcast Messages to Clients

Server constantly monitors messages sent from clients.

```
public class ClientListener extends Thread {
    private ArrayList clients;
    ...
    public void run() { ← method invoked when thread is started
        while (true) {
            Client ith = (Client) clients.get(i);
            if (!ith.isAlive()) clients.remove(i);
            String message = ith.getMessage(); ← check client
            if (message != null) {
                for (int j = 0; j < clients.size(); j++) {
                    Client jth = (Client) clients.get(j);
                    jth.println(message);
                }
                broadcast to everyone
            }
        }
    }
    sleep(100);
    }
}
```

Synchronization

Gotcha: two threads simultaneously accessing the same object.
- ith connection thread puts next message into client ith’s buffer.
- listener thread gets next message from client ith’s buffer.

Dining philosophers problem.
- To eat, philosopher needs both adjacent chopsticks.
- If each philosopher grabs chopstick to right, then waits for chopstick to left, then everyone starves.
- Deadlocking = bane of programming with threads.

Java has built-in support for synchronization.
- Indicate blocks of code that can’t be simultaneously accessed.
- No need unless you are using more than one thread.

Synchronization in Java

Server maintains a separate thread for each client connection.

```
public class Client extends Thread {
    private String buffer; ← message received from client
    ...
    public Client(Socket socket) {...
    
    public void run() { ← only this thread gets hold up if no incoming message
        String s;
        while ((s = in.readLine()) != null) { setMessage(s); }
    }

    public synchronized String getMessage() {
        if (buffer == null) return null;
        String temp = buffer;
        buffer = null;
        notifyAll(); ← tell setMessage to stop waiting
        return temp;
    }

    public synchronized void setMessage(String s) {
        if (buffer != null)
            wait(); ← if current message not yet broadcast,
            then wait to receive next message.
        buffer = s;
    }
}
```

Server maintains a separate thread for each client connection.
Peer-to-Peer Networking

All clients communicate and share resources as equals.

Ex: Napster, Kazaa, ICQ, Gnutella, Morpheus, Limewire, Gnucleus, Skype, Jabber, eMule, BitTorrent, SoulSeek, ...

Centralized (e.g., original Napster)
- Central server keeps index of all files and where they are.
- User queries server to find file, the connects directly with host.
- No central bottleneck.
- Scalable.

Decentralized (e.g., Gnutella)
- User queries neighbors to find file, neighbors query their neighbors.
- No central point of control or failure.
- Massively scalable.

Circuit switching vs. packet switching.
- Circuit switching for real-time communication.
- Packet switching is cheaper per bit.
- Never underestimate the bandwidth of a 747 filled with DVDs!

Client-server vs peer-to-peer.
- Client-server for browsing web.
- Peer-to-peer for file-sharing.

Where to learn more? COS 318, COS 461.

Reference: http://digital5.ece.tntech.edu/690603/lectures.htm