

## Computer/Human Interaction

### Lecture 18

#### Overview:

- Basic networking
- Network applications
- Java sockets

#### References:

- JTUT

## Basic networking

- Hosts have domain names (e.g., `csserver.evansville.edu`) and IP addresses (e.g., `10.10.0.9` – internal; `192.195.228.35` – external)
- Communication between hosts is through a port. Low-numbered ports are reserved for and assigned to well-known services. E.g., `sshd` listens on port 22. (List is in `/etc/services` on Unixes.)

## Network applications

- Network applications usually are structured in a client-server architecture. Often multiple clients talk to single server.
- Commonly used protocol is TCP/IP which is a reliable, connection-oriented protocol.
- Servers listen on (assigned) ports.
- Client uses ephemeral or short-lived ports, usually assigned automatically by TCP.

## Purple Pizza Parlor, v3

- Download from web or copy files  
`/home/hwang/cs350/lecture17/*.*`
- The application is a network client of a server that receives order data and writes it to the server's filesystem.
- Client interface is same as Version 1 with an added field for the port number of server. Automatically tries to connect .
- Server consists of main server class and a handler thread class.

## Sockets

- A socket is an (IP address, port) pair that uniquely identifies an endpoint of a connection.
- A server typically will create a **listening socket** and wait for client connections. When a connection is received and accepted, the server will fork off a thread to handle the connection via a **connected socket**.

## Java Sockets

- Two socket classes
  - `ServerSocket`: a listening socket; constructor receives port number, value of 0 has system choose a free port number
  - `Socket`: a connected socket; constructor receives host domain name and port number; handler socket created by calling `accept` method of `ServerSocket`

```
ServerSocket listenSocket = new ServerSocket(0);
Socket connectSocket = listenSocket.accept();

Socket clientSocket = new Socket(host, port);
```

## Server Threads

- PPP server creates a `PizzaServerThread` object to handle each connection.
- `PizzaServerThread` is a class that extends the `Thread` class. Must implement `run()` method that is called automatically when the thread is created.

## Socket I/O

- Wrap socket streams into buffered streams to read and write data using `readLine` and `println`.

```
BufferedReader in;  
PrintWriter out;  
in = new BufferedReader(  
    new InputStreamReader(  
        socket.getInputStream()));  
out = new PrintWriter(  
    socket.getOutputStream(), true);  
  
String s = in.readLine();  
out.println(s);
```

## Message Protocol

- Client and server must agree on the format of messages between them.
- PPP transmits/receives data in lines in order of the form. Address field may have more than one line and may have more than one topping, so transmission of those data have extra end data markers.

## Notes

- PPP client-server code is not as robust as it could be. It is possible to get into situations where the client locks up waiting for something.
- Handler threads automatically exit when the client disconnects, but the server process will execute indefinitely if put in the background.
- This example is given primarily for completeness and is not required. In particular, applets cannot write to the local machine, so they must talk to a server to save information.