CSCI 142
Computer Science II
Networking
Today

- Networking
  - HTTP and Protocols
  - DoD 4-Layer and OSI 7-Layer Models
  - TCP/IP Sockets & Servers
  - UDP
Networking

- A network comprises the hardware and software that provides two or more computers with the ability to communicate with each other.
- Computer networks are the **big new thing** in technology.
  - They have been for decades.
- With the rapid, explosive growth of the internet, computer networks are becoming of crucial importance to all of modern society.
The World Wide Web

- It was designed to allow people to work together by combining their knowledge in a web of hypertext documents.
  - A hypertext document is a document that contains links to other documents.
- Beginning in 1990, Berners Lee wrote the first web server and the first wysiwyg hypertext browser/editor.
  - The initial version became available on the internet in the summer of 1991.

In the early 90's, Berners lee encouraged and incorporated feedback into his initial specifications of URLs, HTTP, and HTML.
The World Wide Web

A web server stores resources (e.g. HTML documents).

When a client requests a resource using a URL, the server locates the resource and sends it back.

A client uses a browser to request resources identified by a uniform resource locator (URL).

The browser renders and displays the information in a graphical user interface.

GET http://www.rit.edu/index.html

<HTML><BODY>...</HTML>
The World Wide Web

● The Uniform Resource Locator (URL) is the basis of the World Wide Web.
  ○ Think of a URL as an address for a resource stored on any computer anywhere in the world.

● A postal address:
  The White House
  1600 Pennsylvania Avenue NW
  Washington, DC 20500
  USA

● When attempting to locate an address, does one start at the top or the bottom?
  ○ It makes more sense to start first from the country, then locate the city, then the street, the building number, and finally the recipient.

● It is for this reason that URLs are constructed *backwards*.
The World Wide Web

- A URL defines the location of a resource on the web in the following way:
  - service:host:port/file and resource details
- For example:

  http://cs.rit.edu:80/~csci142/Lectures/08/Threads-java.pdf
The World Wide Web

- A URL defines the location of a resource on the web in the following way:
  - service:host:port/file and resource details

- For example:

```
http://cs.rit.edu:80/~csci142/Lectures/08/Threads-java.pdf
```

- **The service or scheme.** In this case, HTTP (hypertext transfer protocol).
- **The hostname of the machine that provides the resource.**
- **The port number to connect to.** Computers may listen for requests on many different ports numbered 0-65,535.
- **The name and location of the resource being requested from the server.**
Hostnames vs. IP Addresses

- When establishing a connection between two computers, it is common to use a *hostname*.
  - The hostname is a human readable, text-based name for a specific computer.
  - www.google.com
  - www.youtube.com
  - www.microsoft.com
  - etc.

- This is convenient for humans, but the *real* address of a computer is a series of numbers that identifies the network that the computer is connected to as well as the address of the specific computer of interest.
  - This is typically a “dotted quad” as is used by IPv4.
  - We’ll talk in detail about these shortly.

To make things even trickier, the specific, numeric address of the computer may (and often does) change.

Dynamic Host Configuration Protocol (DHCP), for example, gives a computer a short term *lease* on a specific address and the address may change each time the lease is renewed.

Q: How, then, does one computer find another computer using only the hostname?

A: A hierarchical series of *Domain Name Servers* (DNS).
Domain Name Server

- A Domain Name Server (DNS) maintains a map of hostnames to machine addresses.
  - Just like any other map: the hostname is the key and the current IP address of the specific machine is the value.
- When a computer attempts to create a connection to a specific computer using the hostname, the DNS is asked for the IP address to which that hostname corresponds.
- This works really well most of the time.
  - If the real address of the host has changed recently, the information in the DNS will be out of date.
  - It sometimes takes a few days for the new information to propagate over the entire internet.

DNSs are arranged into a hierarchy, with the root servers at the top and individual machines at the bottom.
The World Wide Web

- HTTP is not the only service that is supported on the web.
  - ftp - the file transfer protocol
  - news - for newsnet groups
  - mailto - for email
  - telnet - for telnet
  - etc.

- URLs are used in many different situations.
- Each provides specifications for the address that follows.
- Modern browsers and web servers are built with certain assumptions.
  - If no service is specified, assume http.
  - If no port is specified, assume 80
  - If a specific file name is not specified, assume index.html (or similar).

- This is why you use URLs like [www.google.com](http://www.google.com) to access [http://www.google.com:80/index.html](http://www.google.com:80/index.html)
Establishing a Connection

- To fetch a web page, the browser on the client computer first establishes a network connection with the remote host.
  - This means that the browser uses the computer’s network capabilities to send a connection request to the web server named in the web page URL.
- If the server is prepared to accept the connection, it responds with a message to the client indicating that the connection has been accepted.
  - We find the server using DNS.
- Once the connection has been established, the client and server may transfer data.

I’d like to connect to port 80.

That is acceptable.
HTTP

- As mentioned previously, HTTP is short for *hypertext transfer protocol*.
  - It defines a series of simple operations that allow a client and server to transfer hypertext documents back and forth.

- Requests are sent from the browser to the server in plain ASCII text.
  - GET - retrieve data from the server
  - POST - send data to the server, e.g. form data

- For example:
  - GET /index.html

- The server first tries to locate a resource with the name index.html. If it exists and is accessible, the contents are returned to the browser.
The Internet

- Internetworking is the practice of linking technologically different and independently operated networks together
  - A network of networks.
  - Allows users to communicate using electronic mail, store and retrieve data in databases, and to access distant computers.

- The Internet is a global internetwork.
  - The “core” of the internet includes the National Science Foundation’s NSFNET, The Department of Energy’s Energy Science Network (ESnet), the NASA Science Internet (NSI), and the Department of Defense’s ARPANET and Terrestrial Wideband Network.

The internet also includes a continually expanding collection of interconnected regional networks inside and outside of the US.

Each network is independently operated. There is no central control of the internet.
An Intranet

● The phrase “intranet” refers to a local area network (i.e. a private network) that may or may not be connected to the internet, but that has similar functions.

● Computers within the intranet can see each other, communicate, and share resources.
  ○ Some of those resources may not be available outside of the intranet, even if the network is connected to the internet.

● Some organizations have web servers that implement protocols such as HTTP for private use within the intranet.
  ○ The computers inside the intranet are typically shielded from the internet by a firewall.
Protocols

● A protocol is a specification of the format of the messages and the rules governing those messages that two or more machines must follow in order to communicate.
  ○ In order to communicate, two computers must speak the same language.
● Protocols usually exist in two forms:
  ○ A textual specification meant for humans to read and understand. This is used by developers to implement the protocol.
  ○ A coded implementation created by humans for computers to use and understand.
● Both forms should specify the precise interpretation of every bit of every message and how to respond.

If two machines don’t speak the same language, communication is not possible.
Protocols

- Protocols exist at every point where logical program flow crosses between different hosts on the network.
  - We need protocols any time we would like to do something on another computer.
  - For example, we need protocols to print, download a file, fetch a web page, store data on a database server, etc.

- Usually multiple protocols will be in use simultaneously.
  - Computers are usually doing several things at once.
  - One operation may involve several protocols.
  - For example, sending messages using your phone may alternate between SMS (for plain text) and MMS (to send photos or audio).
Protocol Layers

- Protocol layering is a common technique to simplify network designs by dividing them into functional layers.
  - For example, it is common to separate the functions of data delivery and connection management into separate layers.
  - Each layer has its own protocol(s).
- Each layer performs a specific purpose, and doesn’t need to know about the other layers or how to perform their purposes.
- There are two major layered protocol designs in use today.
  - The DoD 4-Layer model
  - The OSI 7-Layer model.
Protocol Layers

The Department of Defense (DoD) 4-Layer model. Also called the TCP/IP Model.

The Open Systems Interconnection (OSI) 7-Layer Model.

The DoD 4-Layer protocol model was developed for DARPA in the 1970s and eventually became the Internet.

The core internet protocols adhere to this model (TCP/IP).

The OSI 7-Layer model was developed in the 1980s, and is generally preferred.
OSI 7-Layer Model

- **The Physical Layer** describes the physical properties of the various communications media.
  - Also describes electrical properties and interpretation of the exchanged signals.
- **The Data Link Layer** describes the logical organization of data bits transmitted on a particular medium.
  - e.g. framing, addressing, and checksum.
- **The Network Layer** describes how a series of exchanges over various data links can deliver data between two nodes.
  - e.g. The addressing and routing structure of the Internet.
- **The Transport Layer** describes the quality and nature of data delivery.
  - If and how transmissions are used to insure data delivery.
- **The Session Layer** describes the organization of data sequences larger than the packets handled by lower layers.
  - Describes how request and response packets are paired in an RPC call.
- **The Presentation Layer** describes the syntax of data being transferred.
  - e.g. how floating point numbers can be exchanged between hosts that use different math formats.
- **The Application Layer** describes how real work is done.
TCP/IP

- Point-to-Point Protocol (PPP) is used to establish a direct connection between two nodes.
  - It is part of the Data Link Layer of the OSI model.
  - It replaces an older protocol, Serial Line Internet Protocol (SLIP) that was designed to work over serial ports and modems.
- The Internet Protocol (IP) is the principal communications protocol in the internet protocol suite.
  - It is part of the **Network Layer** in the OSI model.
- The Transmission Control Protocol (TCP) manages the packaging of data into packets that get routed on different paths over the internet and reassembled at their destination.
- TCP/IP is the two layer protocol comprised of TCP and IP that each Internet *point of presence* or PPP/SLIP user must use to communicate over the internet.
- TCP/IP can be used on many **Data Link Layers**.
  - It supports many different network hardware implementations.
TCP/IP

- TCP/IP are the two most important in a suite of internet protocols.
- These protocols often run in the application layer of the OSI 7-Layer Model, on top of TCP/IP.
  - HTTP - Hypertext Transfer Protocol
  - FTP - File Transfer Protocol
  - TFTP - Trivial File Transfer Protocol
  - Telnet - Text-oriented virtual terminal
  - Gopher - Distributing, searching, retrieving documents
  - SMTP - Simple Mail Transfer Protocol
  - UDP - User Datagram Protocol

TCP/IP is normally considered to be a four layer system.

Because of this it maps well onto the DoD 4-Layer Model (which is not an accident).

It can also be mapped onto the OSI 7-Layer model...
TCP/IP Protocol Suite

- The lowest layer is the **Network Access Layer**
  - Sometimes also called the **Link Layer**.
- This layer may include several hardware and software network configurations.
  - Ethernet
  - Token Ring
  - Frame Relay
  - ATM
- It also includes the operating system and the network device driver.
TCP/IP Protocol Suite

- The next layer is the **Internet Layer**
  - Sometimes also called the **Network Layer**, though this can be easily confused with the Network Access Layer.
- The Internet Layer is where the Internet Protocol (IP) runs.
  - Handles the movement of packets around a network.
  - Routing of packets takes place here.
- IP uses two protocols to help manage the internet layer.
  - Internet Control Message Protocol (ICMP) - used by network devices (like switches and routers) to send messages.
  - Internet Group Management Protocol (IGMP) - used by adjacent routers on the internet to establish multicast group memberships.
TCP/IP Protocol Suite

- The third layer is the **Transport Layer**
- The Transfer Control Protocol (TCP) and User Datagram Protocol (UDP) run here.
- The Transport Layer provides a flow of data between two hosts for the application layer.
- The function of the TCP protocol is to provide a communication service with the following attributes:
  - **reliability** - all data is delivered correctly
  - **connection-oriented** - the protocol provides procedures for establishing interprocess communications.
  - **byte-stream** - no visible packetization so far as the application is concerned.
  - **end-to-end** - guarantees delivery from one host to the other, even if there is no direct connection available.
The final and topmost layer is the **Application Layer**.
The application layer handles the details of the particular application.
  - Applications do not worry about the specifics of establishing connections or reliably transmitting data.
TCP/IP Protocol Suite

Application
Presentation
Session
Transport
Network
Data Link
Physical

Application
Transport
Internet
Network Access

TCP
IP
Ethernet
Token Ring
Frame Relay
ATM

Telnet
FTP
SMTP
DNS
RIP
SNMP

TCP
UDP
Internet Protocol Addresses

- Every Internet-connected system must have a unique Internet host address, or an IP address.
- This is a 32-bit (4-byte) binary number, typically represented as a “dotted quad,” e.g. A.B.C.D
  - A, B, C, and D are decimal values ranging from 0-255.
  - e.g. 129.21.37.18
- An Internet-connected system may also have a host name.
  - e.g. glados.cs.rit.edu
  - The host name is purely symbolic, and is used to find the Internet address of a machine.
  - The hostname is resolved into an IP address using DNS.
IP Address Classes

The first bit in the first octet of a Class A address is always 0.

A 7-bit unique netid. This identifies a specific network.

The decimal range of the first octet is therefore 1-127 (but 127 is a reserved address).

This means that there are only 126 Class A addresses available.

The remaining 24-bits are used to specify the id of a specific host on the network.

A Class A network has 16,777,214 unique host addresses available.
IP Address Classes

The first two bits in the first octet of a Class B address are always 10.

A 14-bit unique netid. This identifies a specific network.

The remaining 16-bits are used to specify the id of a specific host on the network.

There are 16,382 Class B addresses.

There are 65,534 usable host addresses on a Class B network.
IP Address Classes

The first three bits in the first octet of a Class C address are always 110.

A 21-bit unique netid. This identifies a specific network.

There are 2,097,150 Class C addresses.

The remaining 8-bits are used to specify the id of a specific host on the network.

There are 254 usable host addresses on a Class C network.
IP Address Classes

The first four bits in the first octet of a Class D address are always 1110.

A 28-bit multicast group ID.

Class D addresses are reserved for multicasting.
IP Address Classes

The first five bits in the first octet of a Class E address are always 11110.

The remaining 27-bits are reserved for future use.
Data Link Addresses

- An IP address is used by TCP/IP, but IP is implemented in the Network Layer and TCP in the Transport layer.
  - Below the network layer, IP addresses do not make sense.
- Hardware and software running in the data link layer uses different addressing schemes.
- For example, Ethernet uses a 6-byte hexadecimal number.
  - 3-byte vendor ID - each Ethernet manufacturer is assigned a unique 3-byte ID.
  - 3-byte vendor-defined field. Vendors must use these 3 bytes to insure that all devices on the network have a unique ID.
**Ethernet Frames**

- Ethernet encapsulates the data that it sends inside an *Ethernet frame*.
  - The encapsulated data must be between 46 and 1500 bytes.

<table>
<thead>
<tr>
<th>Ethernet Header</th>
<th>IP Header</th>
<th>TCP Header</th>
<th>application header</th>
<th>application data</th>
<th>Ethernet Trailer</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 Bytes</td>
<td>20 Bytes</td>
<td>20 Bytes</td>
<td></td>
<td></td>
<td>4 Bytes</td>
</tr>
</tbody>
</table>

46 to 1500 Bytes

---

This is an illustration of a TCP/IP packet encapsulated in an Ethernet frame.
TCP Ports

- TCP uses port numbers to identify the ultimate destination of data.
  - Generally, ports may be numbered 0-65,535
  - Some of these ports are reserved and not available.

- How does the client determine the port to try to establish a connection to?
  - Well known ports.
    - 80 - HTTP
    - 443 - HTTPS
    - 23 - Telnet
    - 20/21 - FTP
    - 25 - SMTP
    - 22 - SSH
    - etc.

- A list of well known ports can be found here.
Sockets

- Finally!
- Sockets!
- A socket is one endpoint of a two-way communication link between two programs running on a network.
- The socket classes represent the connection between a client and a server. There are two:
  - `java.net.ServerSocket` represents a server waiting for incoming connection requests on a specific port.
  - `java.net.Socket` represents each end of an established connection.
- Sockets can be used to send and receive messages over a network.
- Socket operations resemble file operations.
  - Data transfer uses input and output streams.
  - Sockets must be closed when communications are finished.
The `java.net` package defines many of the classes used for network communication in Java. This includes many classes that support TCP and UDP communication.

For TCP:
- The `URL` and `URLConnection` classes.
- The `Socket` and `ServerSocket` classes.

For UDP:
- The `DatagramSocket`, `DatagramPacket`, and `MulticastSocket` classes.
InetAddress

- The `java.net.InetAddress` class is a class that represents internet addresses.

- Recall that an IP address can be represented by a dotted quad.
  - Four numbers, each of which is a value from 0 to 255.
  - A dotted quad can be represented as a sequence of bytes.

- The `java.net.InetAddress` class provides several `static` methods that translate hostnames or byte arrays into an `InetAddress`.
  - `getByName(String host)`
  - `getAllByName(String host)` - a hostname may have multiple addresses associated with it
  - `getByAddress(byte[] addr)`

- The `InetAddress` class also provides a method to get the address of the machine on which the program is running.
  - `getLocalHost()`
InetAddress

```java
import java.net.InetAddress;
import java.net.UnknownHostException;

public class WhoAmI {
    public static void main(String[] args) throws UnknownHostException {
        // one of many static methods to retrieve an address
        InetAddress local = InetAddress.getLocalHost();

        System.out.println(local.getHostAddress());
        System.out.println(local.getHostName());
    }
}
```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class WhoAmI {
    public static void main(String[] args) throws UnknownHostException {
        // one of many static methods to retrieve an address
        InetAddress local = InetAddress.getLocalHost();
        System.out.println(local.getHostAddress());
        System.out.println(local.getHostName());
    }
}

InetAddress

The InetAddress class provides several static methods for creating an InetAddress.

The getLocalHost() method returns an InetAddress for the local host.

An InetAddress can be interrogated for hostname, IP address, etc.

$ java WhoAmI
129.21.11.200
GypsyDanger
public static void main(String[] args) throws IOException {
    ServerSocket server = new ServerSocket(5678);
    Socket client = server.accept();
    InputStream input = client.getInputStream();
    OutputStream output = client.getOutputStream();
    Scanner scanner = new Scanner(input);
    PrintStream printer = new PrintStream(output);

    while(scanner.hasNextLine()) {
        String line = scanner.nextLine();
        if(line.equals("quit")) {
            break;
        }
        System.out.println("Read line at server: " + line);
        printer.println(line);
    }

    client.shutdownInput();
    client.shutdownOutput();
    server.close();
}
public static void main(String[] args) throws IOException {
    ServerSocket server = new ServerSocket(5678);
    Socket client = server.accept();
    InputStream input = client.getInputStream();
    OutputStream output = client.getOutputStream();
    Scanner scanner = new Scanner(input);
    PrintStream printer = new PrintStream(output);
    while (scanner.hasNextLine()) {
        String line = scanner.nextLine();
        if (line.equals("quit")) {
            break;
        }
        System.out.println("Read line at server: " + line);
        printer.println(line);
    }
    client.shutdownInput();
    client.shutdownOutput();
    server.close();
}

A ServerSocket is constructed to listen for connections on a specific port, e.g. 5678.

Some port numbers are reserved and others may be in use. Trying to create a server socket on such a port will cause a Bind Exception.

The accept() method is used to begin listening for a connection on the requested port.

One server socket can be used to accept many connections from clients. Each time the accept() method is called it will block until a connection is established.
public static void main(String[] args) throws IOException {
    ServerSocket server = new ServerSocket(5678);
    Socket client = server.accept();
    InputStream input = client.getInputStream();
    OutputStream output = client.getOutputStream();
    Scanner scanner = new Scanner(input);
    PrintStream printer = new PrintStream(output);
    while (scanner.hasNextLine()) {
        String line = scanner.nextLine();
        if (line.equals("quit")) {
            break;
        }
        System.out.println("Read line at server: " + line);
        printer.println(line);
    }
    client.shutdownInput();
    client.shutdownOutput();
    server.close();
}

An Echo Server

Once a connection has been established, the accept() method returns a Socket that can be used to communicate with the client that opened the connection.

Every Socket has an InputStream that can be used to read bytes of data sent from the client to the server...

...and an OutputStream that can be used to send bytes of data from the server to the client.

If the data being sent back and forth is plain text, it can be helpful to use wrapper classes like Scanner and PrintStream that make reading/writing text easier.
An Echo Server

The main loop here reads one line of text at a time and, as long as it is not the word "quit", echoes it back to the client.

As with other IO classes that we’ve used, it is important to close Sockets and ServerSockets when they are no longer in use to avoid wasting/locking resources.

The shutdownOutput() method on the Socket class safely closes the socket output stream so that any data that is written is sent to the client before the connection is closed.

Similarly, shutdownInput() method safely advances the socket’s input stream to the end of the stream.
public static void main(String[] args) throws IOException {

    Socket socket = new Socket("localhost", 5678);

    OutputStream output = socket.getOutputStream();
    PrintStream printer = new PrintStream(output);

    InputStream input = socket.getInputStream();
    Scanner scanner = new Scanner(input);

    Scanner prompt = new Scanner(System.in);

    while(true) {
        System.out.print("Enter some text (or 'quit'): ");
        String text = prompt.nextLine();
        printer.println(text);
        if(text.equals("quit")) {
            break;
        }
        String response = scanner.nextLine();
        System.out.println(response);
    }

    socket.shutdownOutput();
    socket.shutdownInput();
}
public static void main(String[] args) throws IOException {

    Socket socket = new Socket("localhost", 5678);
    OutputStream output = socket.getOutputStream();
    PrintStream printer = new PrintStream(output);
    InputStream input = socket.getInputStream();
    Scanner scanner = new Scanner(input);
    Scanner prompt = new Scanner(System.in);

    while (true) {
        System.out.print("Enter some text (or 'quit'): ");
        String text = prompt.nextLine();
        printer.println(text);
        if (text.equals("quit")) {
            break;
        }
        String response = scanner.nextLine();
        System.out.println(response);
    }
    socket.shutdownOutput();
    socket.shutdownInput();
}

Perhaps not surprisingly, the client side of the code looks very similar. The major difference is that a ServerSocket is not used.

Instead, a new Socket is created using a constructor that takes a hostname (or IP address) and a port as arguments.

Other than that, the socket’s OutputStream (or a wrapper) is used to send data to the computer on the other end of the connection...

...and its InputStream (or a wrapper) is used to read data send from the computer on the other end of the connection.
An Echo Client

```java
public static void main(String[] args) throws IOException {
    Socket socket = new Socket("localhost", 5678);

    OutputStream output = socket.getOutputStream();
    PrintStream printer = new PrintStream(output);

    InputStream input = socket.getInputStream();
    Scanner scanner = new Scanner(input);

    Scanner prompt = new Scanner(System.in);

    while(true) {
        System.out.print("Enter some text (or 'quit'): ");
        String text = prompt.nextLine();
        printer.println(text);
        if(text.equals("quit")) {
            break;
        }
        String response = scanner.nextLine();
        System.out.println(response);
    }

    socket.shutdownOutput();
    socket.shutdownInput();
}
```

The main loop uses a Scanner to prompt the user to enter text, which is sent to the server. If the user enters "quit" the loop ends.

As with the server, it is important to safely clean up the socket resources when they are no longer in use.
Threads and Networking

- It is often the case that a server will accept multiple, concurrent connections from clients.
- This means that the server will need to make use of threads!
  - One thread (main) that waits for incoming connections using a ServerSocket.
  - Typically one thread per connected client to handle communication with that client.
- Let’s take a look at a simple example!

Writing a multi-client server is the culmination of everything that you have learned in CS2 so far this semester.

You must write classes and implement interfaces, handle exceptions, read and write input and output, and finally, use threads!

Exciting!
public class ClientHandler implements Runnable {

    private final Socket socket;
    private final Scanner scanner;
    private final PrintStream printer;

    public ClientHandler(Socket socket) throws IOException {
        this.socket = socket;
        scanner = new Scanner(socket.getInputStream());
        printer = new PrintStream(socket.getOutputStream());
    }

    public void run() {
        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            if (line.equals("quit")) {
                break;
            }
            printer.println(line);
        }
        close();
    }

    public void close() {
        socket.shutdownOutput();
        socket.shutdownInput();
    }
}
public class ClientHandler implements Runnable {

    private final Socket socket;
    private final Scanner scanner;
    private final PrintStream printer;

    public ClientHandler(Socket socket) throws IOException {
        this.socket = socket;
        scanner = new Scanner(socket.getInputStream());
        printer = new PrintStream(socket.getOutputStream());
    }

    public void run() {
        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            if (line.equals("quit")) {
                break;
            }
            printer.println(line);
        }
        close();
    }

    public void close() {
        socket.shutdownOutput();
        socket.shutdownInput();
    }
}

In order to handle multiple, concurrent clients, the server will need to create a thread for each client that connects.

This is a thread that implements the “echo server” from the previous example.

Note that the `shutdownOutput()` and `shutdownInput()` methods may throw an exception that will need to be handled (not shown here for space).
public class ThreadedServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(5678);

        while (true) {
            Socket client = server.accept();

            ClientHandler handler = new ClientHandler(client);
            new Thread(handler).start();
        }
    }
}
A Multi-Client Server

```java
public class ThreadedServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(5678);
        while (true) {
            Socket client = server.accept();
            ClientHandler handler = new ClientHandler(client);
            new Thread(handler).start();
        }
    }
}
```

The main program for the server accepts incoming connections in a loop.

Each time a connection is established, the server spins up a thread to handle communicating with that client and then goes back to waiting for the next connection.

Note that, from the client’s perspective, nothing has changed. The same client will work with the old, single-threaded server or the new, multi-threaded version.
URLs Redux

- A URL defines the location of a resource on the web in the following way:
  - service:host:port/file and resource details
- For example:

```
http://cs.rit.edu:80/~csci142/Lectures/08/Threads-java.pdf
```

- The service or scheme. In this case, HTTP (hypertext transfer protocol).
- The hostname of the machine that provides the resource.
- The port number to connect to. Computers may listen for requests on many different ports numbered 0-65,535.
- The name and location of the resource being requested from the server.
URLConnection

- The service part of the URL also defines the protocol that is needed to communicate using that service.
- The protocol determines the format of the messages that are transmitted over the network between the client and server using the service.
- Calling the openConnection() method on a URL will return an instance of the URLConnection class.
  - Each protocol is different.
  - New protocols are being added all the time.
  - Therefore, no one URLConnection can handle all possible protocols.
- URLConnection is an abstract class.
  - URL returns a different subclass depending on the URL service/protocol.
  - If the JVM doesn’t understand the protocol/service an exception is thrown.
public class GetWithURLs {
    public static void main(String[] args)
        throws MalformedURLException, IOException {

        //new thing to deal with: MalformedURLException
        URL url = new URL("http://" + host + ":" + port + "/");

        // open the connection
        URLConnection connection = url.openConnection();
        // open the input stream
        try (InputStream in = connection.getInputStream()) {
            byte[] buffer = new byte[10240];
            int n;
            while ((n = in.read(buffer)) > 0) {
                System.out.println(new String(buffer, 0, n));
            }
        }
    }
The URL class can be used to establish connections to remote servers that communicate using a specific protocol without directly using sockets.

In this case a URL is created using the HTTP protocol.
public class GetWithURLs {
    public static void main(String[] args) throws MalformedURLException, IOException {
        // new thing to deal with: MalformedURLException
        URL url = new URL("http://" + host + ":" + port + "/");

        // open the connection
        URLConnection connection = url.openConnection();

        // open the input stream
        try( InputStream in = connection.getInputStream()) {
            byte[] buffer = new byte[10240];
            int n;
            while((n = in.read(buffer)) > 0) {
                System.out.println(new String(buffer, 0, n));
            }
        }
    }
}

The `openConnection()` method on the URL returns an instance of the abstract `URLConnection` class.

The specific class of the connection is determined by the URL, e.g., `HTTPURLConnection`.

Once opened, the `URLConnection` can be used to create an `InputStream` that reads from the URL resource using the appropriate protocol.

A `URLConnection` can also be used to create an `OutputStream` that writes to the URL resource using the same protocol.
Datagram Socket

- A *datagram socket* is the sending or receiving point for a packet delivery service.
- A datagram socket is *connectionless*, which means that an end-to-end pipe is not established between the client and server.
- Instead, each packet sent or received on a datagram socket is individually addressed and routed.
  - Multiple packets sent from one machine to another may be routed differently.

Datagrams are connectionless, and so packets may arrive out of order or in duplicates.

In fact, delivery of individual packets is not guaranteed at all.

But establishing a connection and guaranteeing the order and delivery of packets is expensive and slow.

Datagrams sacrifice reliability and predictability for low overhead and speed.

Datagrams are often used when it’s “good enough” to get some of the data but not all (i.e. voice over IP, gaming).