Computer Programming – CS 6011
Lecture 3: Basic Network Architecture

Fall 2023

Topics

- Recap
  - ArrayList
  - File Input/Output
- Clients and Servers
- Protocols

Recap

- `public / private`
  - Every member variable / method should be individually declared public / private.
  - If you do not specify one, it will default to `package` (Similar to public but not the same).
- `Shorter println`
  - `import static java.lang.System.out;`
  - `out.println( “Hello World” );`

ArrayList<T>

- This a Java “Template” – known as a Generic

```
ArrayList<String> names = new ArrayList<String>();
```

- Different ways to iterate through an ArrayList:
  - `for( String name : names ) {}`

  -
    ```
    for( int i = 0; i < names.size(); i++ ) {
          out.println( names.get( i ) );
      }
    ```
  - Why do you prefer one over the other?
  - Common mistake using set(idx,) on element that does not exist vs add

Copying ArrayList<T> tips:

- Approach 1 (Not a copy / just an Alias):

```
ArrayList<String> names = new ArrayList<>();
ArrayList<String> namesAlias = names;
```

- Approach 2 (Shallow copy):

```
ArrayList<String> names = new ArrayList<>();
ArrayList<String> copyOfNames = new ArrayList<>( names );
```

Approach 3 (Deep copy):

File Input / Output

- Different methods:
  - We will use the Scanner class

- Read from a file

```
File file = new File( "test.txt" ); // Open file
Scanner sc   = new Scanner( file );
int i = sc.nextInt(); // Reads an integer from the file.
String s = sc.nextLine();  // Reads a line from the file.
```

- Write to a file

```
FileWriter myWriter = new FileWriter( “filename.txt” );
myWriter.write( “Put this text into a file.” );
myWriter.close();
```

Clients and Servers

Protocols

Clients and Servers

- Most networking applications involve “clients” and “servers”
- Servers are long running programs waiting for clients to connect to them.
- Clients are shorter-lived programs that connect to servers.
    - Opens a connection, asks for data, receives data, closes connection. (Then renders the info.)
- Clients and servers often run on different machines
- Networked applications where programs could be client and server at the same time are often called “peer to peer” applications.

Examples of Client / Server Systems

- Web Browsing
  - Web browser (Client)
  - Web server (Server)
- Email
  - Email client like outlook (Client)
  - Email server like Microsoft Exchange Server (Server)
- Zoom
  - Zoom web application / desktop application (Client)
  - Somewhere on the cloud (Server)
- Games (Networked)
- Many Social Media Apps
- Git
  - Git push (Client)
  - Git pull (github is the server)

How does the client connect to the server?

- **Server**
  - Waiting for client requests
- **Client**
  - Get the Server location
    - **IP address** or host name
  - Decide which program in the Server machine to connect to
    - **Port Number**
  - Connect to the server
  - Perform Send/Receive
  - Close the connection

IP Address – Specifies the devices

- A unique* machine identifier
- IPv4
  - 32-bit integer broken up as: W.X.Y.Z
    - Typical (local) address looks like: 192.168.1.15
    - Each piece has up to 256 values.
    - W.X.Y is the prefix and routes the packet to the local network.
    - Z says which machine on that local network.
  - How many addresses are there?
    - ~4.3 billion
- IPv6
  - 128-bit – every atom in the universe can have an IP address.

Ports

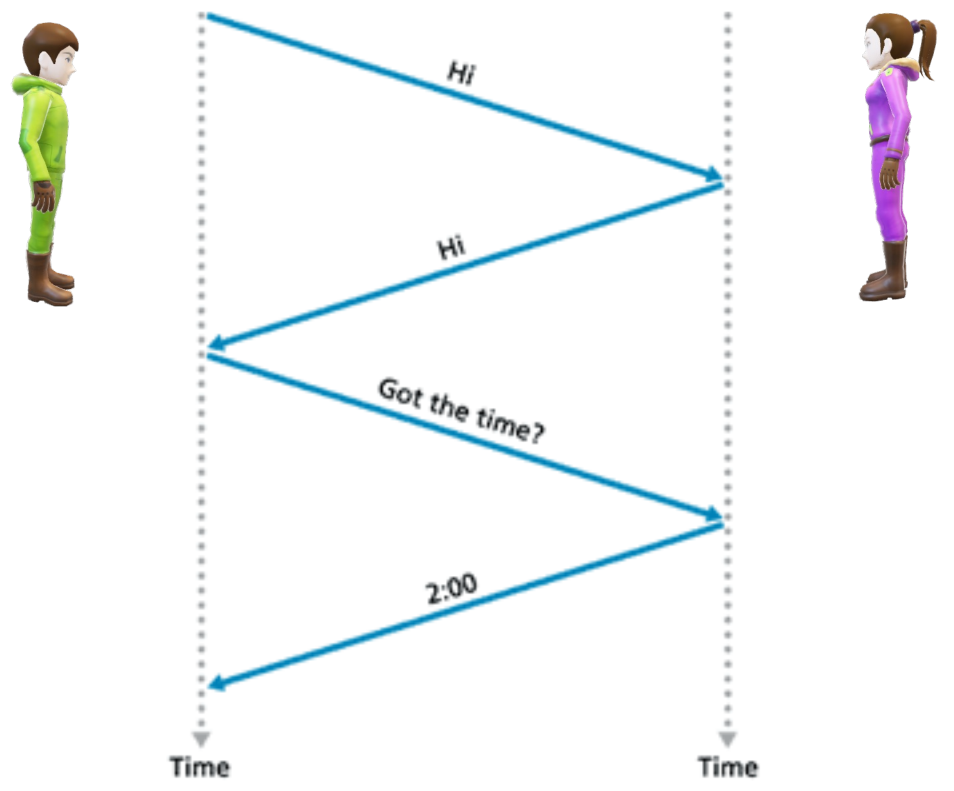
- Specifies which program on the machine gets the data.
  - The “program id”
- We call this the *port number*.
- Many common applications have fixed port numbers assigned to them:
  - HTTP – port 80
  - HTTPS – port 443
  - SSH – port 21
  - List of TCP and UDP port numbers - Wikipedia
- As an application programmer, we don’t need to know the details of these protocols, just enough to specify how to get our data to the remote machine.

Socket

- A process on the internet communicates with another process over a **connection**
- Clients and servers communicate by sending **streams of bytes** over connections
- **Socket**:
  - Endpoint of the connection link between two processes
  - Identified by the combination of both client and server IP address and a port Number

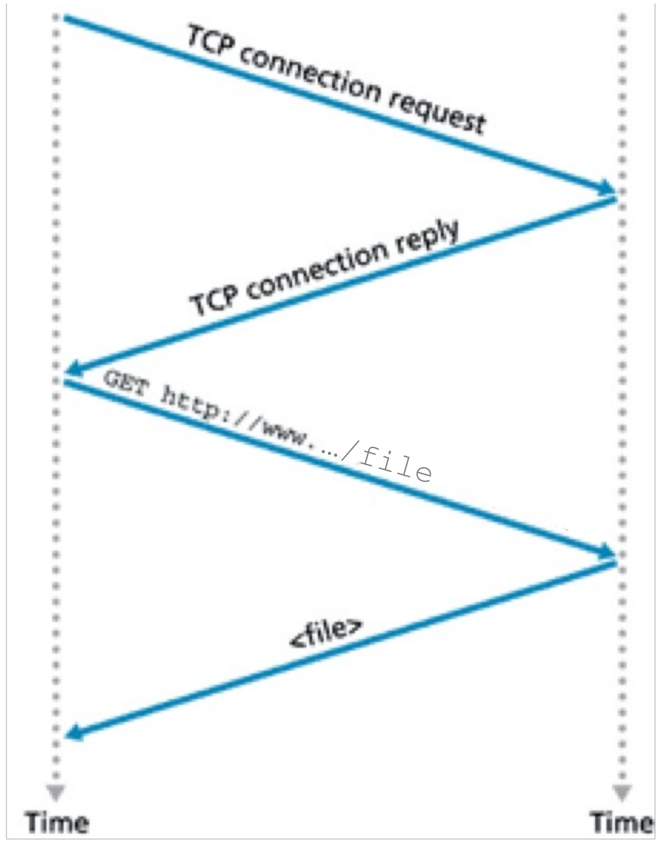
How humans communicate?

- Some Protocol



Network Protocol

Client



Server

Protocols

- Describes rule of how two programs communicate.
- They can be “state-ful” or “state-less”.
- Many protocols exist for many different types of applications.

Protocol Examples

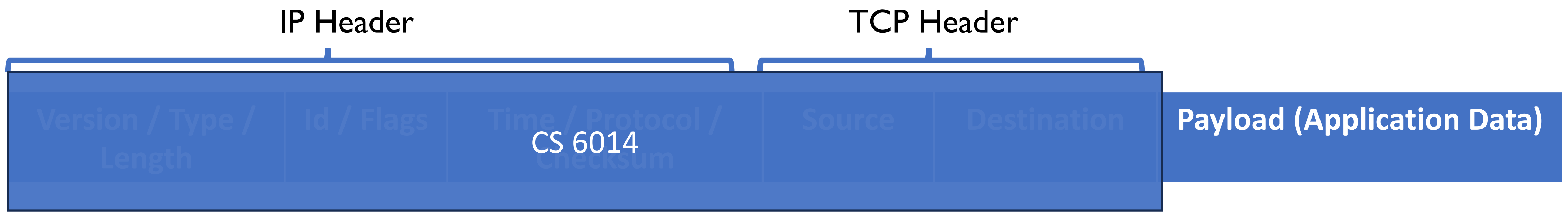
- HTTP – for web pages
- SSH – secure-shell
- SMTP – simple mail transfer protocol
- DNS – Domain Name System
  - Translates human readable computer (network) names into unique IP addresses (192.168.1.1).

Connections: A Programmer's View

- Client/servers connect and can read/write data to a socket.
- At the lowest level for programmers, usually we can think of network connections as a "stream of bytes".
- The functions for reading from / writing to a network stream look just like those for reading /writing a file.
- We use the TCP protocol which has nice guarantees, and in most operating systems the libraries refer to connections as *sockets*.

Sending Data

- TCP/IP model divides the message into network packet

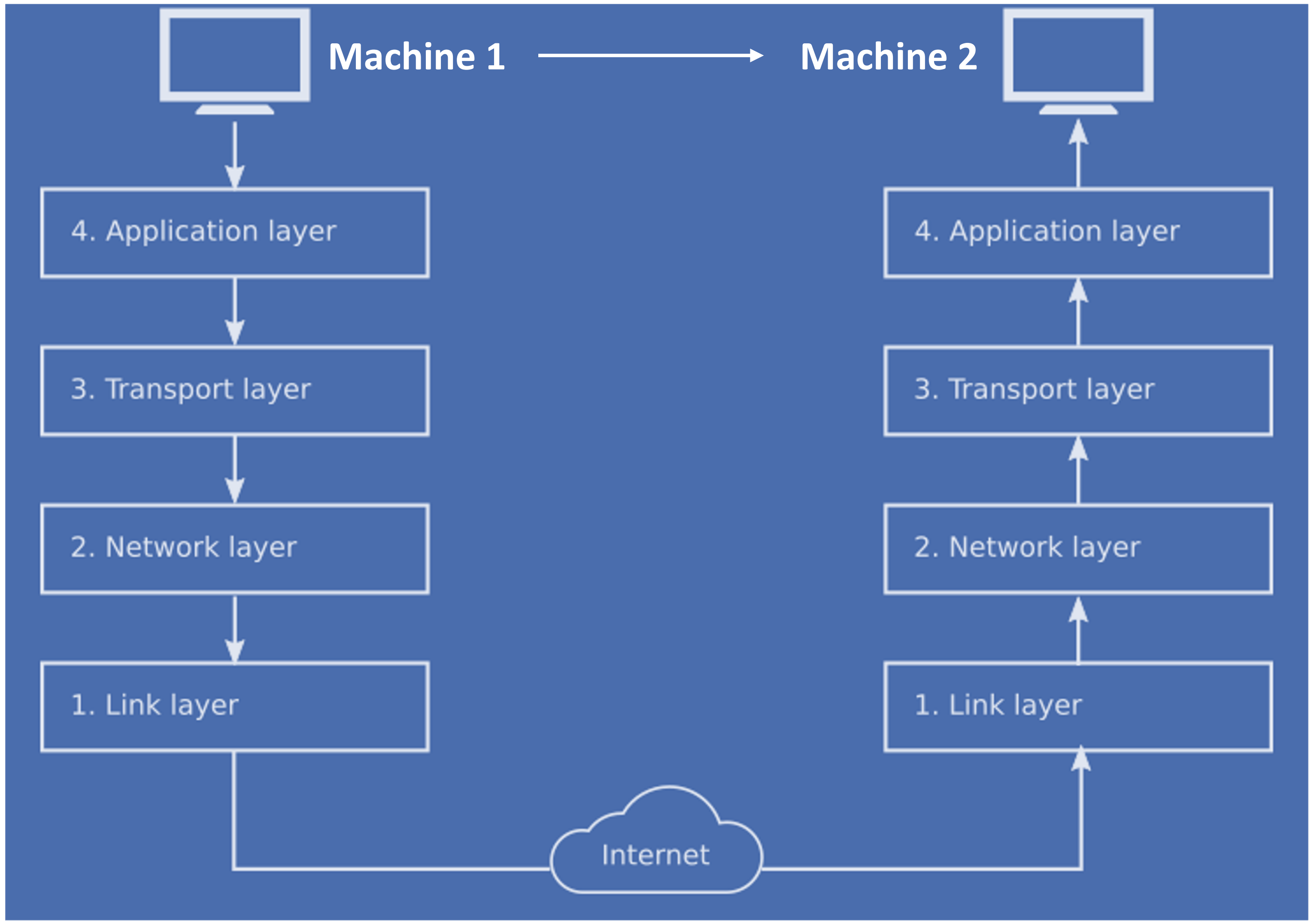


- From the programmer's view, looks like:
  - `Create Server/Client Sockets`
  - `byte[] read();`
  - `void write( byte[] );`

Wireshark

- Record and analyzing network traffic.
- Filter Bar
- In Class Example.

Network Protocol Layers



Machine 1 communicating with Machine 2

Wednesday Assignment(s)

- Lab – Snooping with Wireshark
- HW – Continue Rainfall
- Code Review - Fractions