

(◡‿◡) 1. Review of basic concepts from Data Communications

Nothing will be put in for section one... so we can place instructions here.

Can click on headings to the side to navigate topics faster

We can even put in smaller headings later for each topic.

So we can click to each, find corresponding slides and PDF contained in it.

[illegible]

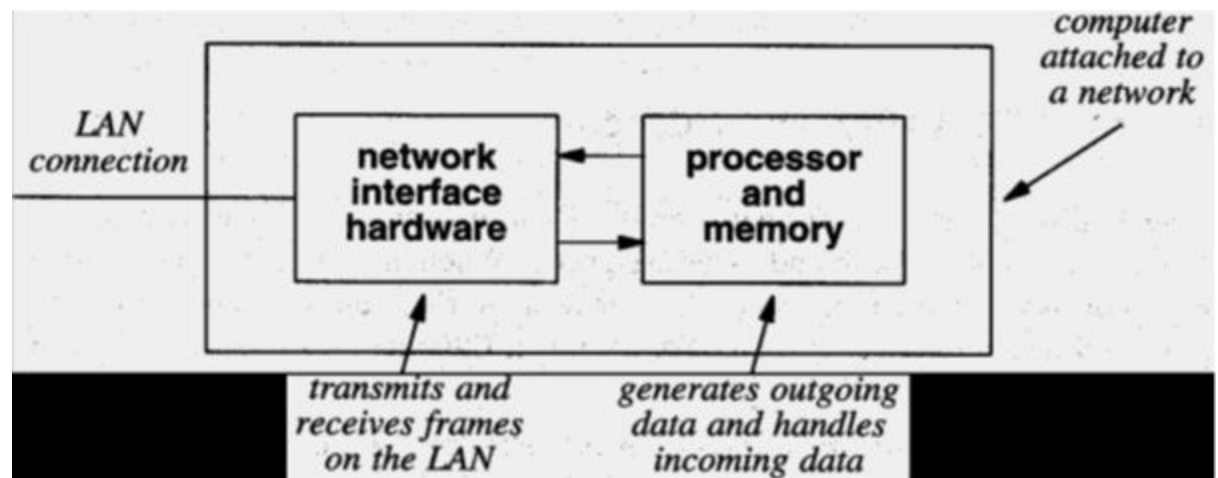
## 2. Local Area Networks

- Extending LANs using Repeaters and Bridges (Lecture 4, All Slides)
  - Repeaters:
    - Repeaters just read and repeat a connection allowing for connection to a more distant LAN.
    - Used to interconnect identical LANs i.e. LANs using the same MAC protocols (e.g. conforming to IEEE 802.3 or 802.5 etc.).
    - Repeaters do not process frames.
  - Bridges:
    - It's kinda like a switch but only exists to connect 2 similar lans, without switch functionality.
    - Used to interconnect LANs that use similar or different MAC protocols (e.g. IEEE 802.3 and/or 802.5 etc.).
    - Bridges do process frames.
  -
- MAC/Hardware Addressing (Lecture 3, All Slides)
  - LAN technologies facilitate sharing of information between all locally connected stations
  - Direct communication between specific pairs of stations is achieved using an addressing scheme
  - Each station is assigned a UNIQUE address:
    - Called a Hardware Address or a MAC Address
  - An example of an Ethernet MAC address is:
    - 00:40:05:1c:0e:9f
    - Ethernet MAC addresses are 48-bit long and are usually represented in Hex format; 12 Hex digits, each digit representing 4 bits.

Different addressing schemes are employed on different LAN topologies.

Each station's address must be unique on the LAN to which it connects.

- Address allocation falls into three categories:
  - a. – **Static**. The h/w manufacturer sets the address
  - b. – **Dynamic**. Stations sets their address at boot-up
  - c. – **Configurable**. The admin. sets the address
- MAC frame formats
  - The header within each transmitted frame contains the addresses of the sender and receiver
  - This allows the LAN interface card (Network Interface Card aka NIC) to filter out frames without conferring with the CPU:



- Bridges – operation, functionality, routing (frame filtering) and routing table
  - Bridges facilitate the interconnection of small LANs to create one large LAN
  - Reliability: The effects of a fault can be contained and restricted to only a few stations.
  - Performance: Smaller LANs provide better performance to locally attached devices. This ties in with the Principal of Locality of Reference:
    - The majority of traffic is often between locally connected stations.

#### Bridge Functionality

- Bridges that understand only one frame format are sometimes called MAC Relay Bridges. These provide the following functionality:
- Store and Forward:
  - Operating in promiscuous mode a Bridge reads all frames transmitted on one LAN.
  - It retransmits frames to an outgoing port to which another LAN is connected only if the destination station is on that LAN.

- The retransmission is done without modification to the frame i.e. bit-by-bit.
- This function is performed in both directions.
- Routing and Addressing:
  - Not all frames are copied. Only those relevant to a particular LAN segment are copied. This implies a routing capability.

#### Bridge Routing

- The use of a bridge does not affect how stations communicate with each other:
  - MAC addresses are used for routing frames between stations connected to the same bridged network.
- The routing decision that a Bridge uses to decide if a frame should be forwarded onto another LAN depends on the routing strategy employed.
- There are two routing strategies to consider:
- Fixed routing.
  - Fixed routing:
    - For each pair of source-destination station a route is created in a routing table stored on the bridge.
    - Based on the destination address in a received MAC frame the bridge performs a lookup of the routing table to determine if the frame is to be forwarded.
  - Advantages/Disadvantages of fixed routing:
    - Simplicity. Requires minimal processing overhead. However, this can become very complicated if multiple bridges are used.
    - Requires a lot of manual intervention when more stations/bridges are added or removed.
- Address Learning
  - Address Learning is an alternative approach to routing.
  - Here the Bridge can learn the location of each station
  - automatically because:
    - Each incoming MAC frame contains a source address field.
    - Each LAN attaches to one port only.
  - Using both of these pieces of information (source address and port number) the bridge constructs a routing table by itself i.e. without manual intervention

### 3. Internetworks

#### - Universal service provision and associated issues - Lecture 5 – slides 4-6

Universal Service – people being able to transmit data to other people without requiring complex computer knowledge

Issues: incompatible system technologies

The provision of universal services requires you to address the problem of having multiple network technologies.

Routers and protocol software is the solution (internetworking technology).

#### - Routers and protocol software

Lecture 5 – slides 9-11

Router: contains cpu, ram, i/o interfaces. Can connect to many LANs. Must physically pass data to provide universal service using a network interface controller.

Protocol software: MAC addresses needs to be unique & needs summary routing. Has a unique framing format understood by all hosts.

#### - IP Addressing (classful, classless, netid/hostid etc.)

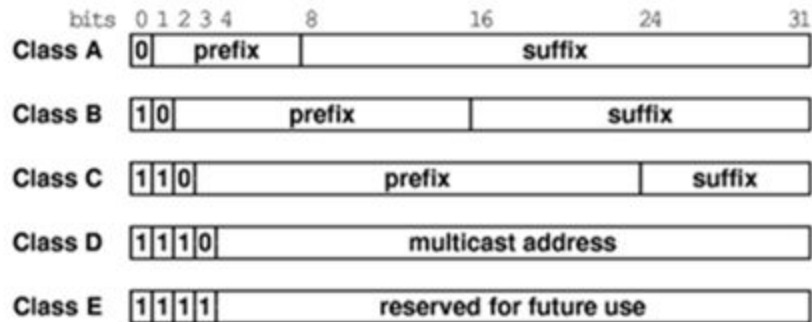
Addressing is a critical part of the internet. IP address is independent of the underlying physical address scheme.

Lecture 6- slides 5,6 mainly, bit in the rest of lecture 6

Classful: IP divided into 5 classes, each with a different prefix and suffix for different sized networks. First four bits identify class, a,b,c are primary, d is used for multicasting, e is for future use

# *Classful* IP Addressing Scheme

---



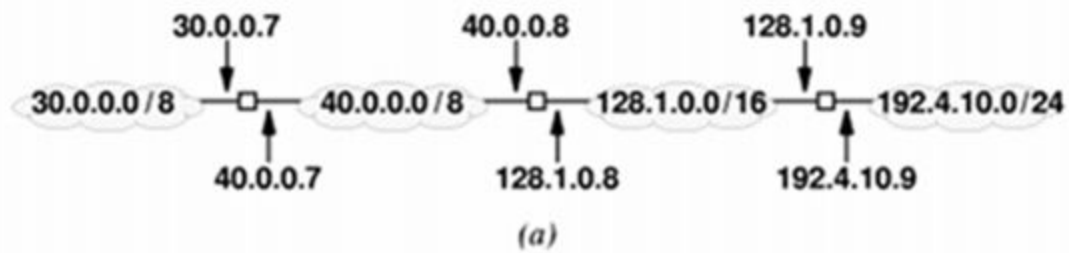
Classless: lecture 7

Too many ip addresses needed, and we don't have enough, so we add a mask at the end of the address, written in CIDR notation.

Classless addressing allows the network address to be subdivided using network masks. Each subnet has the same mask size but the network prefix is different.

- IP Address Allocation and Address Tables lecture 7

Classless address table:



Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

9

- IP Datagram Format

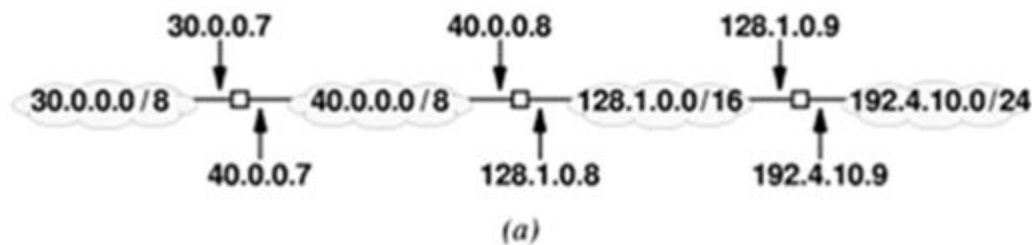
Lecture 8: slide 2

0	4	8	16	19	24	31
VERS	H. LEN	SERVICE TYPE	TOTAL LENGTH			
IDENTIFICATION			FLAGS	FRAGMENT OFFSET		
TIME TO LIVE		TYPE	HEADER CHECKSUM			
SOURCE IP ADDRESS						
DESTINATION IP ADDRESS						
IP OPTIONS (MAY BE OMITTED)					PADDING	
BEGINNING OF DATA						
⋮						

- Forwarding IP datagrams, routing tables etc.- Lecture 8: slide 3-6

- Routing table: where routing information is stored

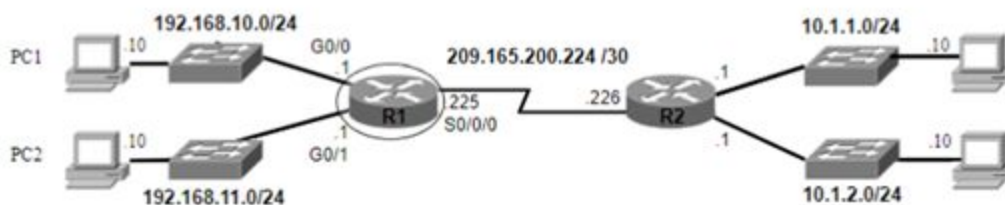
## Example IP Routing Table



Destination	Mask	Next Hop
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	deliver direct
128.1.0.0	255.255.0.0	deliver direct
192.4.10.0	255.255.255.0	128.1.0.9

4

## Example IP Routing Table



- Encapsulation - Lecture 8: slide 7-9

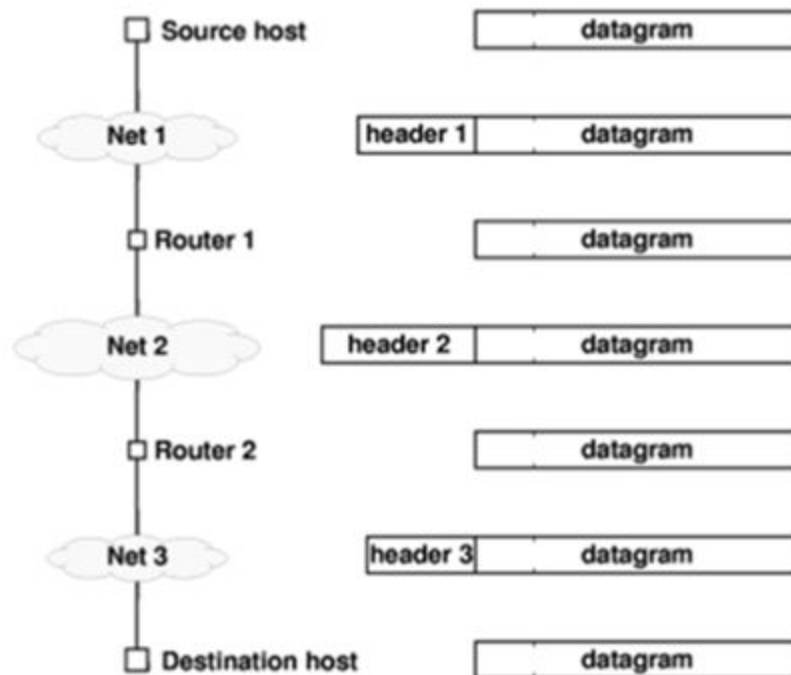
Networks don't understand datagram, so it's instead stored in the data area of a hardware frame. This is done on each leg of the transmission, with the datagram being stored without the additional frame header info.



Size differs depending on network technology.

# Encapsulation at work

---



- Address Resolution Protocol (ARP)

Lecture 9.

## 4. Protocols, Protocol Architecture and Reference Models

**ALL CONTENT IS PDF 11 + 13.**

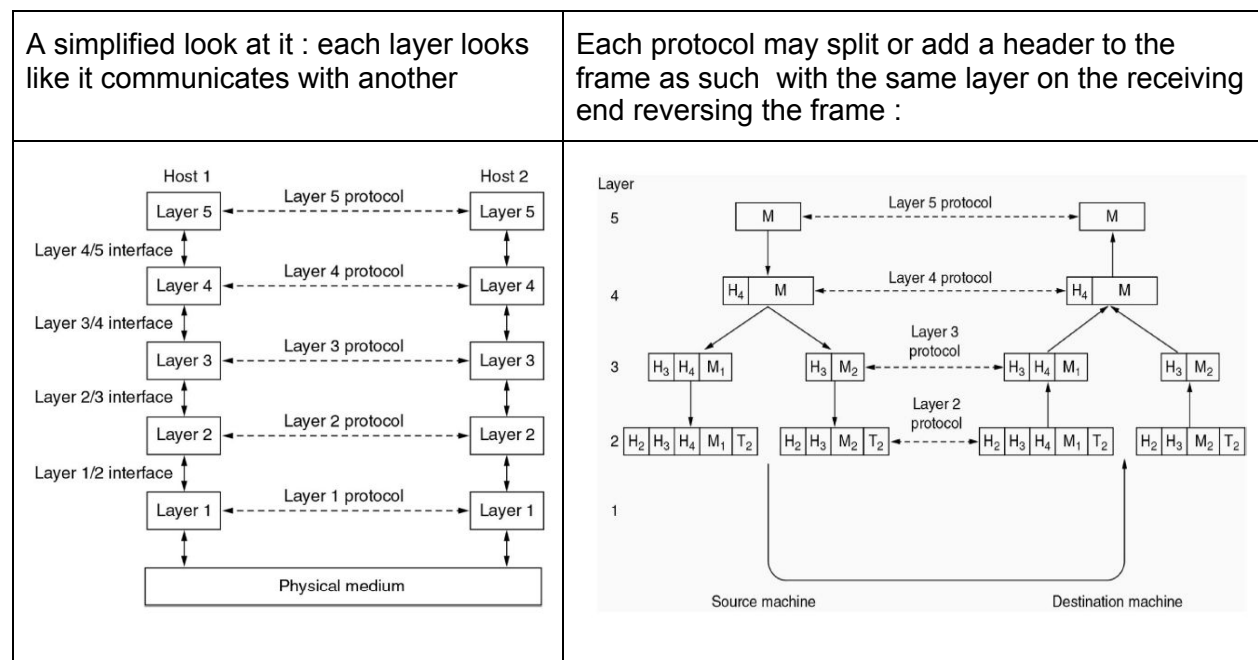
- What is a protocol? [PDF 11 slide 1](#)

A **network protocol** is an established set of rules that determine how data is transmitted between different devices in the same **network**. Essentially, it allows connected devices to communicate with each other, regardless of any differences in their internal processes, structure or design.

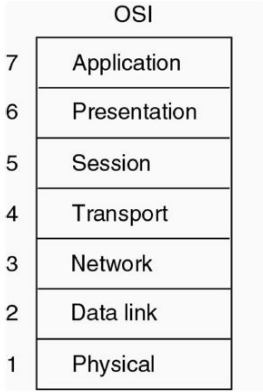
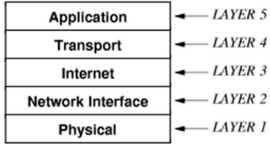
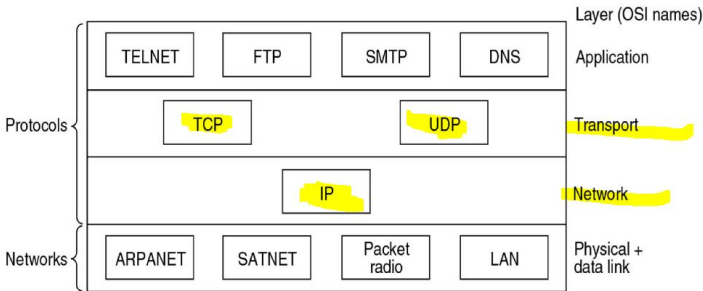
- Difference between a service and a protocol [PDF 11 slide 1 + PDF 13 slide 3](#)

A **protocol** is a set of rules for communication within a layer. A **service** is what the layer provides to the layer above it through an interface. **Protocols** at one layer are unaware of issues at another layer. ... Each layer is able to communicate with the corresponding layer of a receiving station.

- Protocol Architectures layer-by-layer [PDF 11 slide 2-9](#)



- ISO-OSI 7-Layer model and TCP 5 Layer Model [PDF 11 slide 10 - 18](#)

<p>OSI Layer 7 is not an actual protocol stack just a reference model for what layers should do :</p>	
	<ul style="list-style-type: none"> <li>- <b>Application</b> = stuff like HTTP, FTP email</li> <li>- <b>Presentation</b> = stuff like translate communication between big endian and little endian computers</li> <li>- <b>Session</b> = keeps track of connections and the type of connection.</li> <li>- <b>Transport</b> = splits data into deliverable chunks or builds received chunks</li> <li>- <b>Network</b> = deals with routing the chunks/ packets, handles addressing formats.</li> <li>- <b>Data link</b> = Breaks up data into data frames</li> <li>- <b>Physical</b> = literally just ensures physical integrity</li> </ul>
<p>TCP/IP a protocol stack. :</p>	<p>Relationship between TCP UDP and IP :</p>
	

TCP vs UDP - TCP makes sure that all data is delivered without errors, UDP just sends data to the receiver from a request and doesn't wait for confirmation. UDP is faster but no error checks are made in the protocol itself. **Can be googled fast for more details**

– TCP (Transmission Control Protocol): This is a reliable, connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error to any other machine in the internet.

– UDP (User Datagram Protocol): This is an unreliable, connectionless protocol for applications that provide their own sequencing and flow control functionality

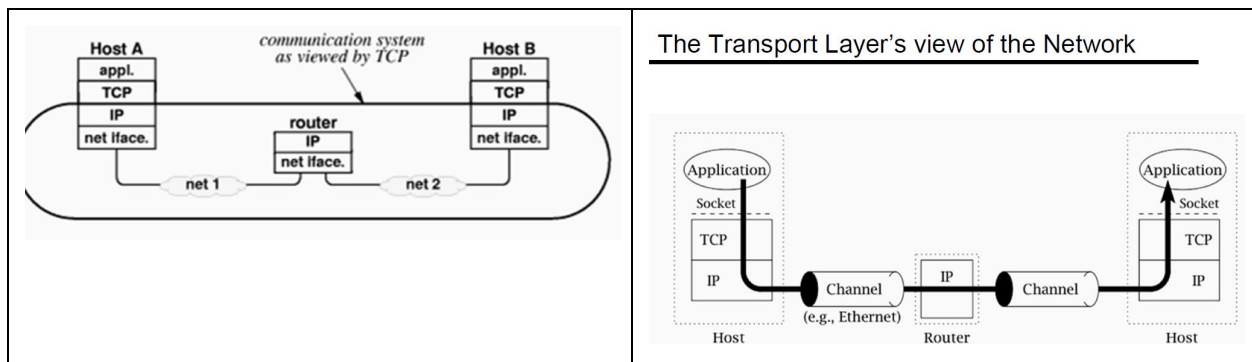
- There are a number of key design issues common to a several layers:
  - *Addressing* – Each layer needs to be able to identify *senders* and *receivers*. Some form of *addressing* is required
  - *Error control* - The *receiver* must be able to tell the *sender* which messages have been correctly received and which have not
  - *Sequencing* - The protocol software on the the receiver must be able to resequence incoming messages
  - *Flow Control* – The receiver must be able to control the flow of information from the sender

- Encapsulation at work (Wireshark)

- haven't used enough wireshark to throw a comment but you should be able to see layering?

- The transport services to upper layers [PDF 13 slide 3 ish](#)

Sits below the application layer providing a transport system to messages between applications that want to communicate across the network.



- The transport entity [PDF 13 slide 12 \(Theres honestly alotta shite on this, check PDF 13 slide 12 -20\)](#)

The tcp software within the transport layer which implements the service is referred to as the Transport entity. The service is facilitating internetwork communications.

It can be located within, the OS kernel, a separate user process, a library package bound to a network application, or on the network interface card.



## 5. Client-Servers and C-S Interaction [ALL CONTENT IS PDF 12](#)

### - Characteristics of clients and servers [PDF 12 Slides 1 - 7](#)

Server Characteristics :

A **Server** is specifically an application which is run on a server class machine in its definition, although it is also commonly used to refer to dedicated computers for running server applications.

In general :

- It's passively waiting / listening for client contact to respond to.
- It's a program which provides one specific service.
- Usually runs on dedicated/stronger bigger hardware to be able to serve multiple clients at once. - *damian refers to this as a server-class machine.*
- Usually the program starts on computer boot, since usually the computer it runs on is only there to host the service.

Client Characteristics :

A **client** refers to an application which uses a server / contacts to a server to receive Information from it .

In general it (exceptions apply ) :

- Provides general purpose computational functionality.
- Is used by a user directly.
- Runs off of a general computer system.
- Can use multiple services but can only contact one server at a time. (aka a browser accesses a website at a time, discord's also a client, contacting a chat server).

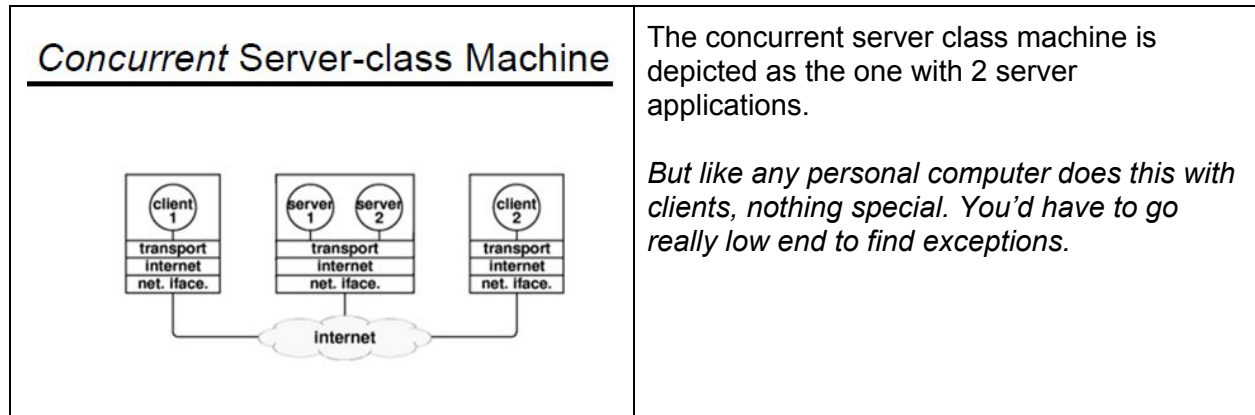
### - Multiple services on one server-class machine [PDF 12 Slides 8 - 10](#)

Server class machines can run multiple clients and servers simultaneously.  
(basically are built for it since they're strong computers).

These computers have operating systems that allow multiple application programs to execute concurrently eg UNIX windows.

For each service a server-class machine provides, there's an associated server program.

*Eg. a computer may run a file server and a world wide web server and these two are different services*



How do clients identify specific servers for communication?

Some form of addressing is acquired.

*(pretty sure this is what ports are)*

- Each server is assigned a unique identifier, clients and servers use this identifier in all interactions.

Communications paradigm is as follows

- Server application starts execution first by registering its identifier with the local protocol software, then waits for contact for clients.  
*(pretty sure it's it declaring it's port or a similar thing in not IP addressing).*
- Client and server exchange messages and terminate communication.

## 6. Berkeley Sockets (the socket primitives)

- Example client and server application (daytimeClient/Server and echoClient/Server)

(daytimeClient/Server - PDF 14 - slides 2-3)

Operating of the Daytime application:

- Client calls CONNECT to connect to the server,
- Server calls ACCEPT to accept the connection request,
- Server returns a formatted string using the SEND primitive
- Server application calls CLOSE to close the connection,
- Client calls RECV to retrieve the data from the connection,
- Client closes the connection using the CLOSE primitive,
- The Client application terminates using exit(0);

Key points for Daytime server code example:

- Server address structure,
- Calls to bind(), listen() and accept().

(echoClient/Server - PDF 14 - Slide 4)

- The Client application sends (send() ) a string (from the command-line) to the server across an open connection,
- The Server application reads (recv() ) the string and returns it to the Client application (send() ) exactly as it came in,
- Both applications call for the connection to be closed (close() ).



## - The primitives explained

(recv() - PDF 14 - slides 4 - 7)

```
while ((numBytes = recv(sock, recvbuffer, BUFSIZE - 1, 0)) > 0)
{
    recvbuffer[numBytes] = '\0';
    fputs(recvbuffer, stdout);
}
```

The while loop is necessary as the data may not arrive across the connection in a single transfer:

- Recall that data arriving from the remote socket is stored in the RECV-Q buffer within the local TCP entity,
- Repeated calls to `recv()` are needed to transfer this data from TCP (the Transport layer) into the application (the Application layer).

`numBytes` is the return value from `recv()`:

- It represents the number of bytes read from the socket,
- It returns one of three values:
  - <1 represents an error condition,
  - 0 represents a closed connection i.e. remote application has called `close()`,
  - >1 represents an open connection with potentially more data to be received.

## Breaking from *recv()*

---

- If the `recv()` primitive is used as above in the **Echo Client** and **Server** applications it will cause problems:
  - Either or both applications will remain inside the loop,
  - Refer to the solution discussed in class.

## - Sockets, Listening and Connected Sockets

???

???

???

???

???

????

??????

???????

??????????

## Old Stuff

### 5. Client-Servers and C-S Interaction

#### - Characteristics of clients and servers

(Characteristics of clients - PDF 12\_ClientServer - Slide 6)

- It provides general purpose computational functionality to a user but on occasion it becomes a client application
- It is invoked directly by a user and executes for one session
- It runs locally on a user's personal computer
- It actively initiates contact with a server
- It can access multiple services but can only contact one server at a time
- It does not require specialised hardware or a sophisticated operating system

(Characteristics of Servers - PDF 12\_ClientServer - Slide 7)

- It is a special-purpose, privileged program dedicated to providing one service
- It can handle multiple remote clients simultaneously
- It invoked automatically upon boot-up, and executes through many sessions
- It runs on a shared computer
- It passively waits for and accepts contact from arbitrary remote clients
- It requires powerful hardware and a sophisticated operating system

#### - Multiple services on one server-class machine

(Multiple services on one server-class machine - PDF 12\_ClientServer - Slides 8-10)

- Some server-class machines run multiple clients and servers simultaneously
- have an operating system that allows multiple application programs to execute concurrently.
  - e.g. UNIX or Windows
- For each service offered there must be an associated server program executing
  - e.g. a single computer might run a file server and a World Wide Web server

### Multiple Services On One Computer

- With *multiple* servers running, how can a client identify a particular server *unambiguously*?
- Some form of addressing is required:
  - Each server is assigned a unique *identifier*
  - *Clients* and *servers* use this identifier in all interactions
- The *communications paradigm* is as follows:
  - The *server* application starts execution first:
    - It registers its identifier with the local protocol software
    - It then waits for contact from clients
  - *Clients* contact *servers* by specifying the server's *location* and *unique identifier*
  - The client and server exchange messages and terminate communication