CSCI 120 Introduction to Computation
Computer Networks (draft)

Saad Mneimneh
Visiting Professor
Hunter College of CUNY

1 Introduction

Primitive forms of data networks have a long history. After all, it is all about communication. Early societies used smoke signal to communicate information. In the 19th century, telegraphy was used as means of communicating messages. Messages were manually encoded into strings of symbols and then manually transmitted and received. When necessary, the messages were manually relayed at intermediate points. Example of such manual encoding and transmission is the Morse code, created by Samuel Morse in the 1830s. Morse code used standardized sequences of short and long marks or pulses (commonly known as dots and dashes) to encode letters, numerals, and punctuation marks. For instance the Morse code for the letter V is **...**— [1]. Reflecting on the knowledge that we have accumulated in this course so far, this is equivalent to bits! A dot corresponds to a 0 and a dash corresponds to a 1. Therefore, we can say that telegraphy used **binary symbols**.

In the 1050s and 1960s, we have seen that central computers were connected to terminals by links to form time-sharing real-time processing systems (see Lecture 11). This constitute the first form of modern computer networks that we are familiar with today.

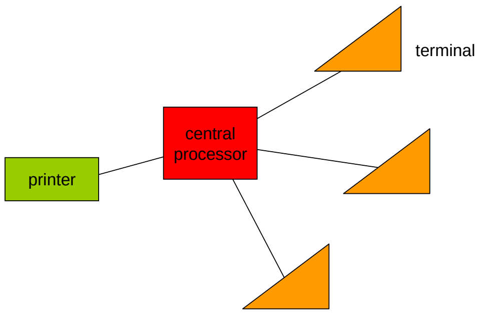

Figure 1: Central computer network

[1]The 4 opening notes of Beethoven's fifth symphony, named Victory, represent the Morse code for the letter V: three short notes followed by a long note. This is just a coincidence. Beethoven died in 1827 before Morse invented his code.

Later on, multiplexers were used to collect all the traffic from a set of peripherals in the same geographic area and send it on a single link to the central processor. To free the processor from the burden of handling communication, a special processor, called front end, was also developed to control communication.

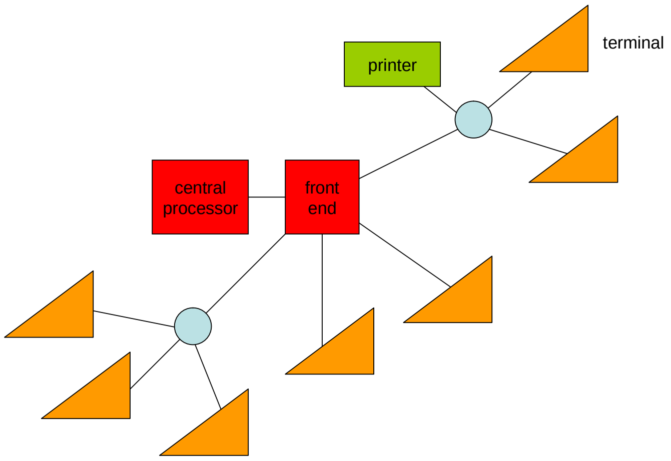

Figure 2: Multiplexers and front end added

In these networks, and in contrast to telegraphy, communication is automated. However, while such a system can be referred to as a computer network, it is simpler to view as one computer with remote peripherals. Real networks emerged in the 1970s when ARPANET and SYMNET were introduced. These were the first large scale general purpose computer networks connecting geographically distributed computer systems, users, peripherals, etc...

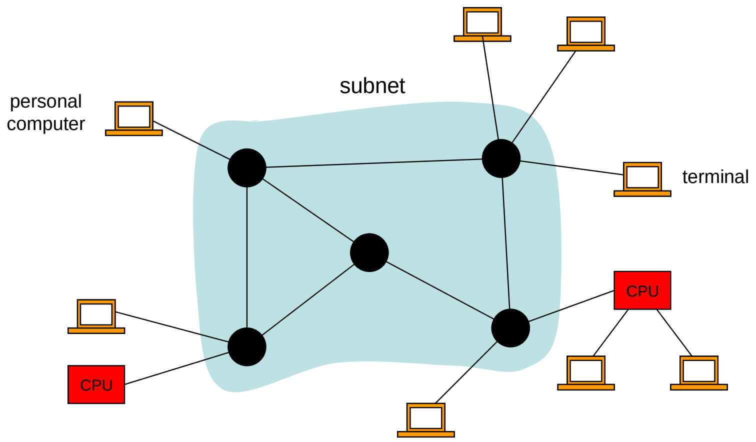

Figure 3: Computer network

Therefore, instead of having a computer as the center of the network, the subnet (communication part of the network) becomes central. Inside the subnet we have a set of nodes (these are usually computers in their own right), various pairs of which are connected by communication links. Outside the subnet, various computers, databases, terminals, and other devices, are connected via the subnet. A message originates at an external device (the sender), passes through the subnet from one node to another on the links, and goes out to the external receiver.

The subnet contains a somewhat arbitrary placement of links between nodes. This placement, often called **topology**, is typical of wide-area networks **WANs**, i.e. networks covering more than a metropolitan area. In contrast, local-area networks **LANs**, cover few square kilometers or less, and usually have more restricted topologies, including ring, bus, and star. Today, bus topology, having been standardized as Ethernet, is probably most popular.

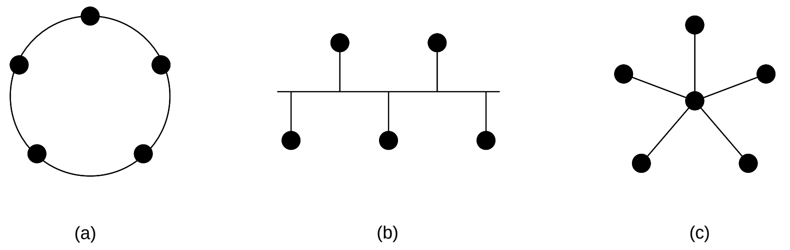

Figure 4: LAN topologies: (a) ring, (b) bus, (c) star

Since the 1970s, there have been an explosive growth in the number of WANs and LANs including ARPANET, TYMNET as WANs and Ethernet and token ring networks as LANs. All these networks needed to connect and communicate. In the 1980s many networks started to connect to each other as in the figure below:

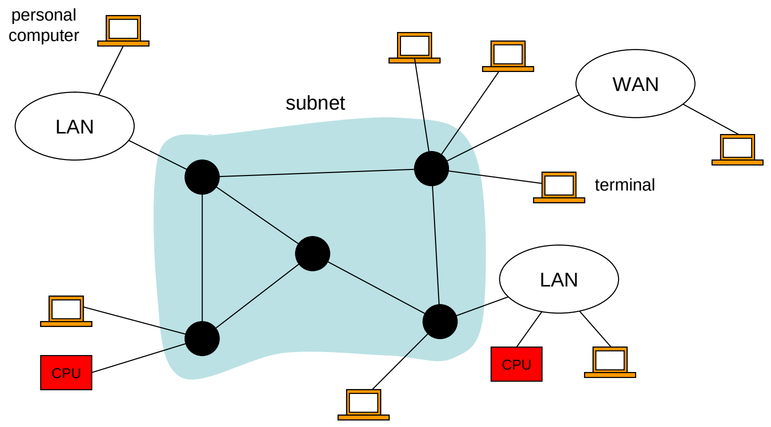

Figure 5: WANs and LANs

Conceptually speaking, one could regard a network of networks as a bigger network. Practically speaking, however, a network of networks is more complicated then just a network of nodes. Because different networks evolve differently and have different conventions and control algorithms for handling data. Therefore, **bridges** and **gateways** (also called routers) were used to handle the inhomogeneity. Bridges are used to connect LANs, while gateways are used to connect WANs. This distinction has to do with routing and we will see it when we talk about the network layered architecture.

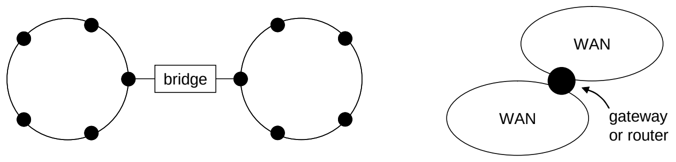

Figure 6: Bridges and gateways

2 A flavor of network protocols

For a network to function properly, it is important to establish rules by which network activities are conducted. Such rules are known as protocols. For instance, consider the problem of coordinating transmission of messages among computers in a network. If all computers insist on transmitting their own messages at the same time, without helping in relaying other messages, no message will be delivered. We will describe two network protocols that address this issue in LANs: Token ring and Ethernet.

2.1 Token ring

IBM developed in 1970 the token ring protocol. In this protocol, computers are on a ring topology, and the messages are transmitted and relayed in only one common direction (either clockwise or counterclockwise). Each message has information about the sender and the intended receiver. When a message reaches its destination, the receiver machine keeps a copy of it, but continues the act the of relaying the message. Eventually, the message will reach back to the originating machine, at which point, the machine knows that the message must have been delivered and stops relaying it.

This scheme depends on the machines cooperation. If a machine insists on constantly transmitting messages of its own rather than relaying those of other machines, nothing will be accomplished. This is where the most important part of the protocol comes in. A unique bit pattern (to distinguish it from any other message), called a **token**, is passed around the ring. Possession of this token gives the machine the right to transmit its own message. Normally, machines relay the token in the same way they relay messages around the ring. But when a machine wants to transmit a message, it grabs the token, and sends its own message instead. Then this machine will wait until it sees its own message again before releasing the token and relaying it to the next machine on the ring. Therefore, since no other machine can posses the token during that time, eventually the message will reach back to its sender (all machines are just relaying). We can argue that eventually every message will be delivered because every machine will have a chance to grab the token.

2.2 Ethernet

Ethernet was developed for a network with bus topology. In Ethernet, the right to transmit messages is controlled by a collection of protocols known as Carrier Sense and Multiple Access with Collision Detection CSMA/CD. With this protocol, each message is broadcast on the bus to all machines. But every machine keeps those messages that are only addressed to it. To transmit a message, each machine “sense the bus”, and waits until it is silent, then beings transmission. If another machine begins transmitting, both will "detect a clash" and pause for a brief **random** (why?) period of time, and then retry.

3 Messages, packets, and sessions

So far we have deliberately used the term "messages" to signify units of communication. But what constitute a unit of communication? The answer is: "it depends". From the standpoint of a network user, a message is a single unit of communication. For that user, receiving part of the message would be worthless. However, in the subnet or computer, the message is just a sequence of bits. Moreover, in most networks, messages are broken into smaller chunks called **packets**. Therefore, one should distinguish between a message and its representation. A message carries a specific information that does not change; however, that message may undergo several transformations as it travels from sender to receiver, e.g. broken into packets, compressed, encrypted, etc...

Messages between two users occur as sequence in a larger transaction, such sequence or messages or transaction is called a **session**, e.g. a user sending messages to update a database. Typically, a setup procedure is required to initiate the session, in this case the session is called a **connection**. In other networks, no such setup is required, and each message is treated independently; this is called a **connectionless** service. Therefore, it is important to look at how messages (or packets) are transmitted during a session. We have two main approaches to transmit packets in a session: circuit switching and store-and-forward switching.

4 Circuit switching

When a session is initiated (connection) it is allocated a given rate in bits per second (bps). A path is then chosen from source to destination, and each link on this path allocates the desired portion of its capacity to the session.

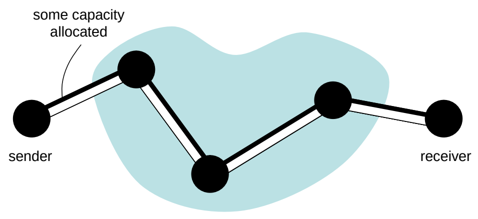

Figure 7: Circuit switching

Circuit switching, however, is rarely used in computer networks because typical sessions tend to have a short burst of high activity, followed by long inactive periods. Therefore, circuit switching would waste the allocated rate during the time in which the session is inactive and not sending any packets.

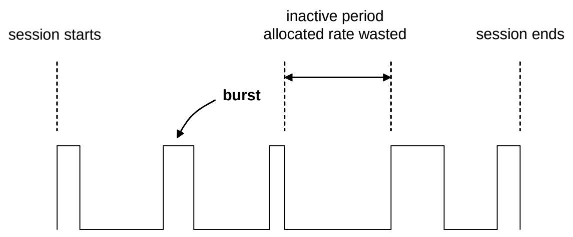

Figure 8: Bursts and inactive periods

5 Store-and-forward switching

In store-and-forward, one packet is transmitted at a time using the full capacity of the link. The links are still shared between different sessions, but the sharing is done on a need basis, rather than by a fixed allocation of rates. Therefore, if a packet needs to use a link, and that link is not available because another packet is being transmitted, then the packet must wait. For this reason, queues or **buffers** are used at every node. A packet waits in the buffer until the link is available (hence the name store-and-forward).

Although queuing delays are hard to manage and control, it can be shown that using communication link on a need basis often reduces delay in networks relative to circuit switching. One of the major issues in buffered networks is that buffers may become overloaded. Controlling the buffers might involve the overloaded node to send to the offending inputs (those who are sending a lot of packets) some control information telling them to slow down (this is one technique for **congestion control**). But even then, a considerable number of packets may already be in the subnet on its way to the node. When reaching the buffer, those packets may be dropped as a result of buffer overflow.

Store-and-forward can be either connectionless, or can establish a connection. If connectionless, there is an issue of packet ordering: Packets travel individually on different paths from the source to the destination and may arrive in the wrong order. Since there is no established path, another issue with connectionless store-and-forward is routing, which is to determine how packets should travel from the source to the destination (we are not going to study routing issues, but we will mention IP addresses later on).

If on the other hand a connection is established with store-and-forward switching, it is called **virtual circuit switching**. This is a combination of both approached (circuit switching and store-and-forward switching). In virtual circuit switching, a particular path is set up when the session in initiated and is maintained during the life of the session. But the capacities of the links on that path are shared by sessions on a need basis, rather than by fixed allocation of rates. This is why is it called *virtual circuit* switching; the path is not really reserved.

6 Network Architecture

When designing complex systems, such as a network, a common engineering approach is to use the concepts of modules and modularity. In this approach, the design of the system evolves by breaking the big task into smaller tasks. Each module is responsible for a specific task and provides services to the other modules to accomplish their tasks. We can interact with a module as a black box that provides certain functionality without knowing the details of how it works. We only need to know how to interface with the module. Someone can remove the module and update it with a newer one, and we would still be able to continue our work in the same way. Moreover, modularity is important to simplify tasks (divide and conquer). For instance, in a network, reliability of message delivery and routing of messages can be treated separately by different modules. Changing one would not impact the other. If a better routing procedure is employed, it only affects the module responsible for it. Certainly, we do not desire a change in routing to affect our ability to reliably deliver messages.

Modules often interact in a hierarchy. A network is designed as a hierarchical or layered architecture in which every module or layer provides services to the upper layer. Users, sitting at the top layer of the network, communicate as if there is a *virtual link* between them, and need not be aware of the details of the network. The following figure illustrates the standardized 7 layers of a network. A description of each one follows.

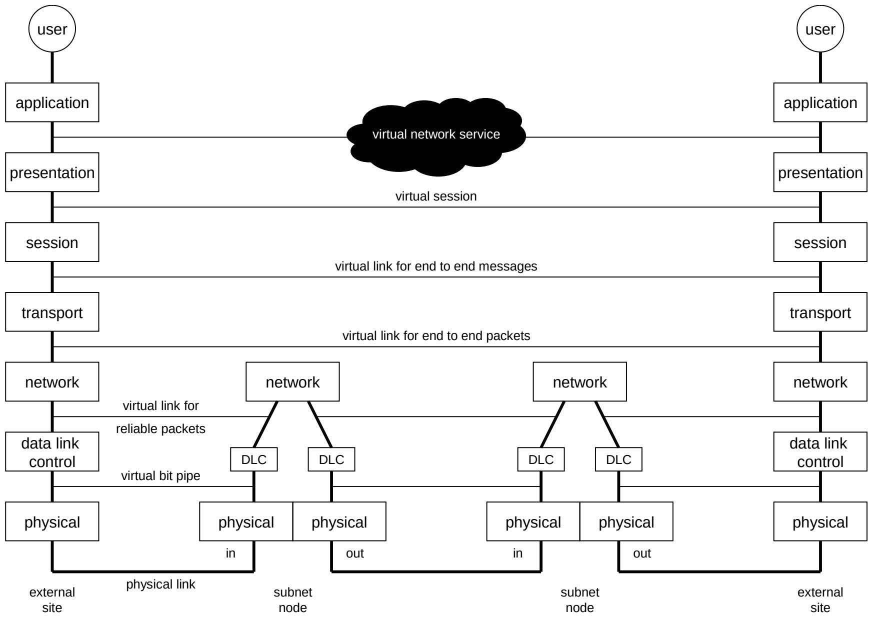

Figure 9: Network architecture: the 7 layers

6.1 Physical layer

The physical layer handles the actual communication using the physical link, e.g. a modem or a network card. Therefore, the physical layer provides to the layer above it a virtual link which could be considered as a simple bit pipe carrying a flow of bits.

6.2 Data link control layer

The data link control, DLC, converts the unreliable bit pipe into a reliable error-free link for sending packets in both directions. The goal is to ensure that every packet is delivered once, only once, without errors, and in order. To accomplish this task, the DLC inserts its own control information as header and trailer to the packet received by the upper layer, i.e. the network layer. This is an example of the message undergoing some transformations as it travels from sender to receiver. The thus modified packet is called a *frame*. At the other end, the DLC strips out the header and trailer to recover the original packet and delivers it to the network layer.

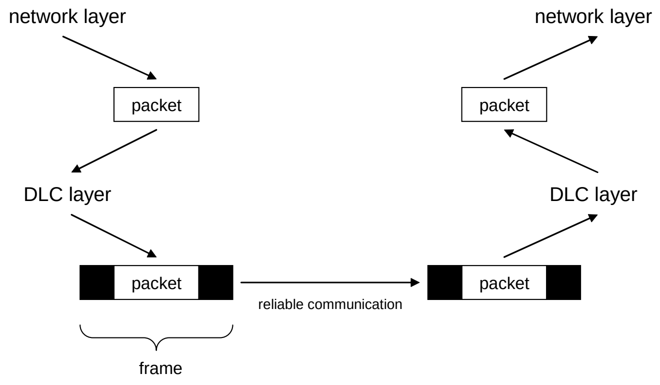

Figure 10: Frames: DLC inserts control information

In addition to other information, the DLC layers communicate sequence numbers in the frame's header and trailer to ensure delivery of packets in order. This forms the basis for the virtual reliable packet link provided to the network layer: once the recipient DLC ensures that it received a packet with the correct sequence number, it delivers that packet to the network layer.

This technique of inserting headers and trailers to packets received the upper layer (and stripping them out when the packet is delivered to the upper layer) is performed by every layer in order to insert its own control information needed to accomplish its task.

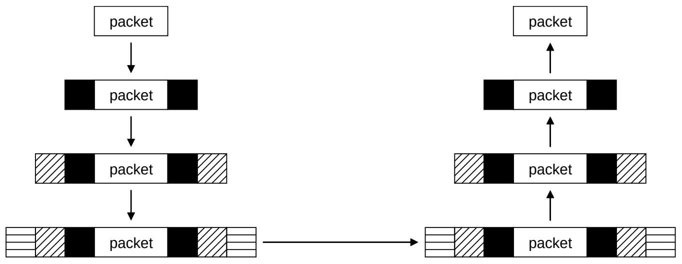

Figure 11: Layers inserting/stripping their control information

The MAC (medium access control) is considered a sublayer of the DLC and handles the task of coordinate multiple access so that multiple nodes can use the link. Since routing in LANs is done at this layer (e.g. token ring or CSMA/CD), the bridge that connects two LANs usually talks to the MAC.

6.3 Network layer

The network layer uses its own packet header (and trailer) to accomplish routing and flow control functions. For instance, the source/destination information can be stored in the network layer's header.

For networks using virtual circuit routing (i.e. the route per session is fixed over the lifetime of the session and store-and-forward is used):

- The route must be selected. This is usually carried in a distributed way by all the nodes in the subnet. The standard X.25 accomplished this task using the concept of virtual channels (we will skip the details here).
- Each packet must be ensured to follow the assigned route. This can be done by placing enough information in the header, e.g. (1) previous node and (2) virtual channel number (we will skip the details here again).

For other networks that do not require a connection phase, usually called datagram networks, each packet is routed individually. Although this appears to be a very natural and simple approach (for instance, it does not require extra information to be stored in the packet header), the dynamic nature of the traffic and the lack of timely knowledge about traffic patterns makes it much more difficult that one would think. For one thing, order of packets becomes an issue because packet follow different routes. Therefore, most WANs use virtual circuit switching.

The IP (Internet Protocol) standard accomplishes the routing of packets in datagram networks. With IP, since no route information is provided, the source and destination of a packet must be universally recognizable throughout the internet. This is done by using IP addresses. Each node in the network has an IP address which is a 32 bit address that uniquely identifies it. The IP address consists of four 8 bit numbers (a total of 32 bits), e.g. 146.95.2.168.

The immediate user of the network layer, i.e. the transport layer is definitely concerned with the type of *service* provided by the network layer. Based on the above description, we have two kinds of services.

- virtual circuit service: all packets are delivered once, only once, and in order, but connection is required.
- datagram service: packets can be delivered out of order and may occasionally fail to be delivered, but connectionless.

In addition, the network layer is responsible for congestion control. Unlike LANs (e.g. token ring and CDMA/CD), since each WAN implements its own flow and congestion control algorithms, it is not easy to simply connect two networks together. The gateway or router is used to connect two WANs. The gateway is actually considered to operate in a sublayer of the network layer called the internet sublayer and is part of both networks, called subnets in that context. A gateway connecting two subnets interfaces with each subnet using the appropriate network layer compatible with that subnet. Compare this to the bridge which works at the DLC level for LANs.

6.4 Transport layer

The transport layer is responsible for the following:

- Breaking messages into packets for the network layer (and re-assembling them).
- Multiplexing several low rate sessions into one, thus reducing overhead (the network layer sees one session).

- Splitting a high rate session into multiple ones which is useful when the network layer cannot handle high rate session.
- Achieving reliable end-to-end communication if the network is unreliable, e.g. IP datagram (even if the network is reliable, failures are possible).

A famous example for a transport layer standard is TCP (Transmission Control Protocol). When combined with IP at the network layer, we refer to the whole thing as TCP/IP.

6.5 Session layer

The session layer mainly deals with access rights in setting up sessions, e.g. who has access to particular network services, etc...

6.6 Presentation layer

The major functions of the presentation layer are data encryption/decryption and , data compression/decompression.

6.7 Application layer

The application layer is simply what is left over. Each application performs something specific to the user needs, e.g. browsing the internet, transferring files, sending text messages, etc...