Computer Networks
Introduction

Saad Mneimneh
Computer Science
Hunter College of CUNY
New York

"dit dit dit dah"
Beethoven's 5^{th} symphony

1 Historical overview

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 19^{th} 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. A statue of Samuel Morse is placed at Inventor's Gate on the east side of Central Park on 72^{nd} street in New York. Morse code used standardized sequences of short and long marks or pulses [commonly known as dots (dits) and dashes (dahs)] to encode letters, numerals, and punctuation marks. For instance the Morse code for the letter V is ...- [1].

A	.-	**M**	--	**Y**	-.--	**6**	-....
B	-...	**N**	-.	**Z**	--..	**7**	--...
C	-.-.	**O**	---	**Ä**	.-.-	**8**	---..
D	-..	**P**	.--.	**Ö**	---.	**9**	----.
E	.	**Q**	--.-	**Ü**	..--	.	.-.-.-
F	..-.	**R**	.-.	**Ch**	----	,	--..--
G	--.	**S**	...	**0**	-----	?	..--..
H	....	**T**	-	**1**	.----	!	·-·-·
I	..	**U**	..-	**2**	..---	:	---...
J	.---	**V**	...-	**3**	...--	"	.-..-.
K	-.-	**W**	.--	**4**	....-	'	.----.
L	.-..	**X**	-..-	**5**	.....	=	-...-

Figure 1: Morse code

[1] The four opening notes of Beethoven's fifth symphony, named Victory, represent the Morse code for the letter V (accidentally, also 5 in Roman numerals): three short notes followed by a long note. This is just a coincidence. Beethoven died in 1827 before Morse invented his code.

In modern terminology, the dots and dashes are equivalent to bits! For instance, 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, central computers were connected to terminals by links to form time-sharing real-time processing systems. This constitute the first form of modern computer networks that we are familiar with today.

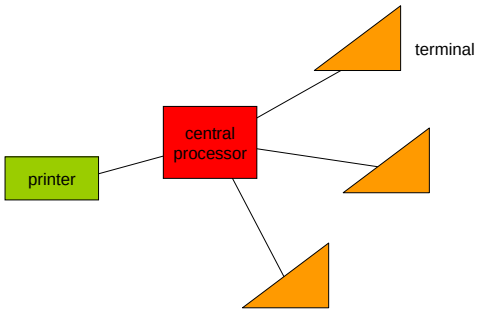


Figure 2: A time-sharing system with a central processor. Tasks from the terminals are rotated in and out of execution.

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.

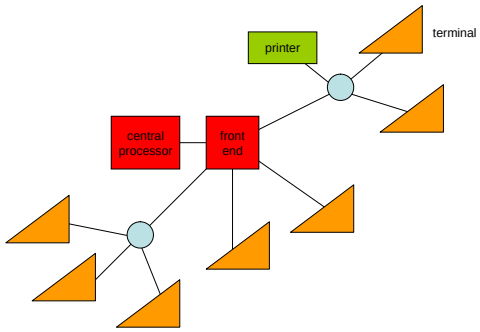


Figure 3: Multiplexers and front ends added

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...

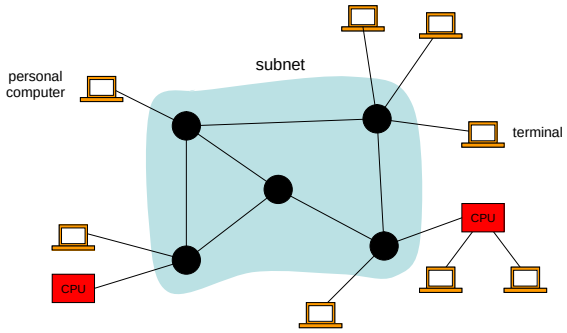


Figure 4: A computer network

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 e.g. building or department, and usually have more restricted topologies, including ring, bus, and star. Today, bus topology, having been standardized as Ethernet, is probably most popular.

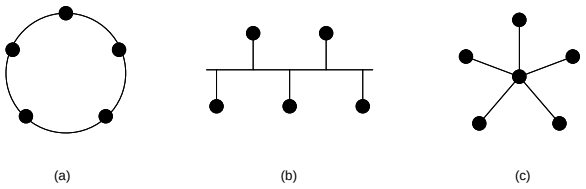


Figure 5: LAN topologies: (a) ring, (b) bus, (c) star

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 Figure 6.

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, one of the major problems for any network protocol is to coordinate the transmission of messages

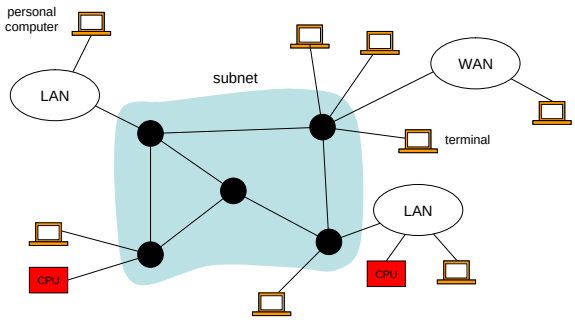


Figure 6: WANs and LANs

among computers in the 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 briefly describe two network protocols that address this issue in LANs: Token ring and Ethernet. We will study both of these protocols later on in much more detail.

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 Networks in modern terminology

With LANs and WANs joining together, network connectivity occurs at many different levels. Therefore, we can define a network recursively:

- two or more nodes connected by a physical link
  - point-to-point
  - multiple access (like Token Ring or Ethernet)
- two or more networks (clouds) connected by a node (gateway/router)

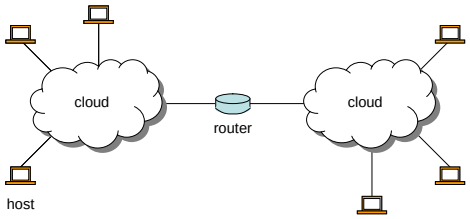


Figure 7: Networks as clouds

Nodes inside a cloud implement the network function e.g. switching as described in the next section.

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. Not all networks are created equal :) and, therefore, networks evolve differently. They have different conventions and control algorithms for handling data (later we will talk about packet formats, message forwarding mechanisms, and routing). 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 the level at which the networks are being joined together (we will see this later on).

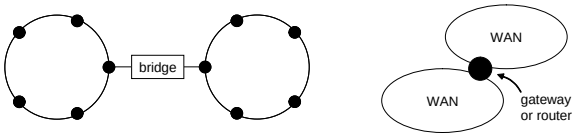


Figure 8: Bridges and gateways

4 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, on the communication link, 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 of messages or transaction is called a **session**. We can think of a session as a logical channel over which application level processes are communicating, e.g. a user sending (through some database client application) messages to update a database (database server application).

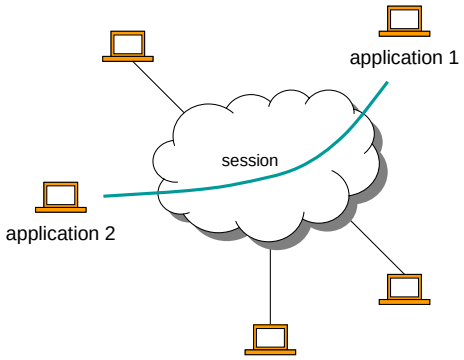


Figure 9: Session

At this application level, one important question is what service should the session provide? The service provided must of course depend on the application needs. For instance, should we allow messages to be lost? Should we receive messages in order? Regardless of the different guarantees, we may put sessions in two categories.

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** session. Moreover, properties of the session are also affected by how messages (or packets) are forwarded (**switching technique**) in the network. We have two main switching techniques:

- circuit switching
- store-and-forward switching
  - datagram
  - virtual circuit

5 Circuit switching

When a session is initiated (connection) it is allocated a given rate λ in bits per second (bps). A path is then chosen from source to destination, and each link on this path allocates the desired portion r of its capacity C to the session. If capacity is used, future sessions are blocked, e.g. phone network.

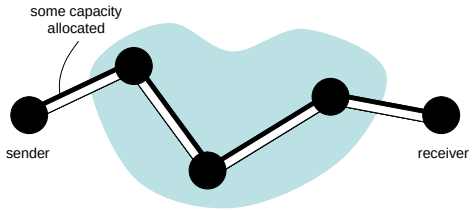


Figure 10: Circuit switching

Due to this resource dedication, circuit switching has the advantage of guaranteed delivery of messages with fixed delays. 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. Such sessions are called bursty or low factor sessions. Therefore, circuit switching would waste the allocated rate during the time in which the session is inactive and not sending any packets.

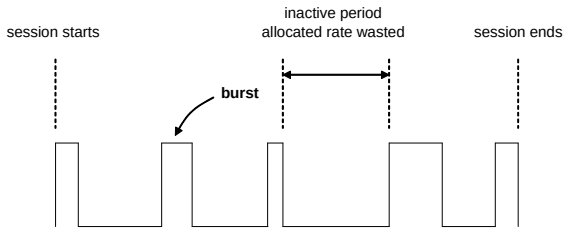


Figure 11: Bursts and inactive periods

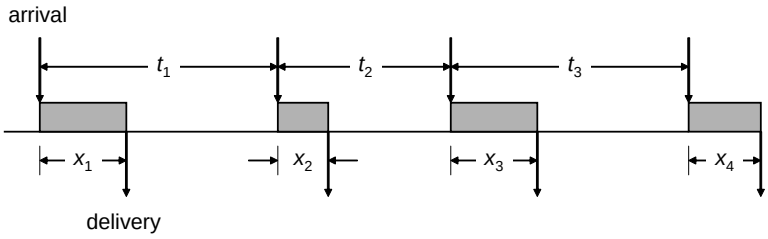


Figure 12: Example session activity

Let λ be the message arrival rate, i.e. $1/\lambda = E[t_i]$, r be the allocated rate, and L be the expected message length. Then $L/r = E[X_i]$ is the expected delay over a

link. If the allowed delay is T, then we must have $L/r < T$ (at least need to travel one link). But if $T << 1/\lambda$, then $L/r << 1/\lambda$ i.e.

$$\frac{L\lambda}{r} << 1$$

Therefore, the link is idle most of the time. Here's an example:
$L = 1000$ bytes ($= 8000$ bits)
$\lambda = 1$ message/sec
$L/r < 0.1$ (delay requirement, so $r > 80000$ bps)
$L\lambda/r < 0.1 = 10\%$

6 Store-and-forward switching

In store-and-forward, aka packet switching, 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).

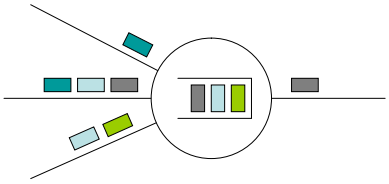


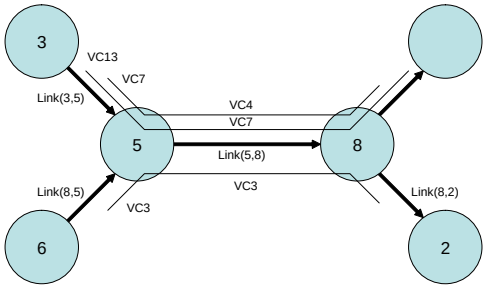
Figure 13: Buffer in packet switching

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. Connectionless store-and-forward is known as **datagram**. In datagram, routes are chosen on a packet by packet basis. Therefore, 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, e.g. IP (Internet Protocol). 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 will see this when we study routing issues in the Internet). For instance, since no path is established in IP, the source and destination addresses (IP addresses) must be included in each packet, i.e. a $2\lceil \log n \rceil$ bit overhead (where n is the number of nodes in the network).

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 approaches (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.

In circuit switching, each link can be visualized as being shared by a set of "virtual channels" VCs. When the session is set up, a path is established by assigning, on each link of the path, one unused VC. Each node maintains a mapping (a table) of VCs.



Node 5 table
(3,5) VC13 → (5,8) VC7
(3,5) VC7 → (5,8) VC4
(6,5) VC3 → (5,8) VC3

Figure 14: Virtual channels VCs

Therefore, global addresses are needed to establish the virtual circuit when the session start. But once established, local VC numbers can be used for routing. In the worst case we have $n(n-1)$ sessions, and a

$$\lceil \log n(n-1) \rceil < \lceil \log n^2 \rceil = \lceil 2 \log n \rceil \leq 2 \lceil \log n \rceil$$

bit overhead is needed to store VC numbers; however, generally virtual circuit requires less overhead. Examples are X.25 and ATM.

The following figure provides a summary of the switching techniques with their advantages/disadvantages.

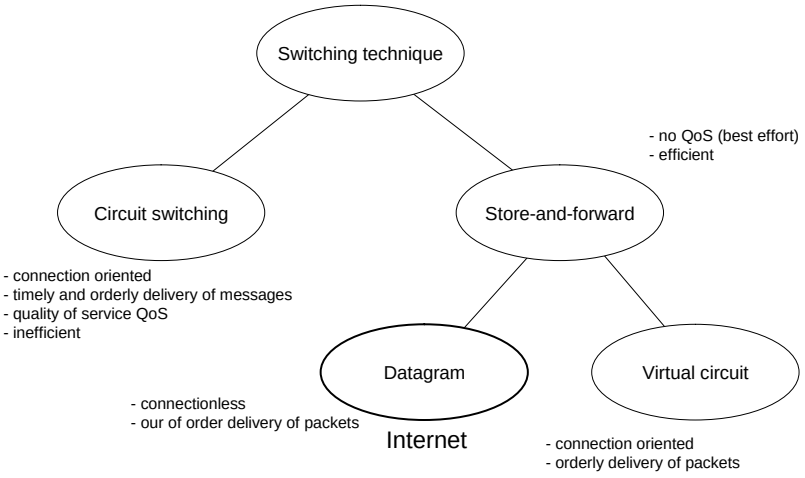


Figure 15: Switching summary

References

Dimitri Bertsekas and Robert Gallager, Data Networks
Larry Peterson and Bruce Davie, Computer Networks: A Systems Approach