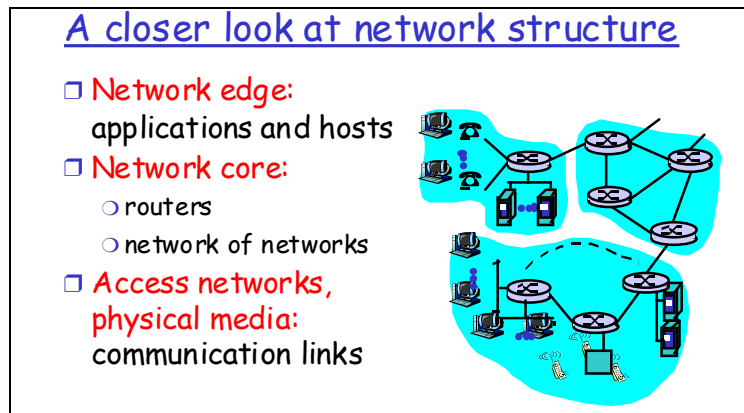


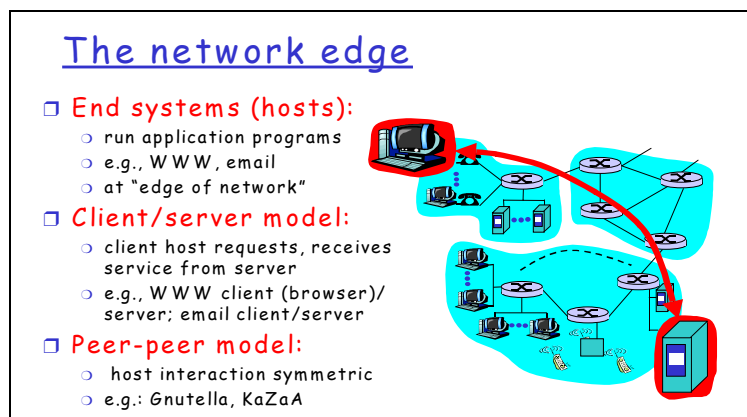
The Network Edge

Visual 9



In the previous sections we presented a high-level description of the Internet and networking protocols. We are now going to delve a bit more deeply into the components of the Internet. We begin in this section at the edge of network and look at the components with which we are most familiar – the computers (for example, PCs and workstations) that we use on a daily basis. In the next section we will move from the network edge to the network core and examine switching and routing in computer networks. Then we will discuss the actual **physical links** that carry the signals sent between the computers and the switches.

Visual 10



End Systems, Clients, and Servers

In computer networking jargon, the computers that we use on a daily basis are often referred to as hosts or end systems. They are referred to as hosts because they host (run) application-level programs such as a Web browser or server program, or an e-mail program. They are also referred to as end systems because they sit at the edge of the Internet, as shown in Visual 10. Throughout this module we will use the terms hosts and end systems interchangeably; that is, host = end system.

End-System Interaction

Hosts are sometimes further divided into two categories: clients and servers. Informally, clients often tend to be desktop PCs or workstations, whereas servers are more powerful

machines. But there is a more precise meaning of a client and a server in computer networking.

In the so-called **client/server model**, a client program running on one end system requests and receives information from a server running on another end system. This client/server model is undoubtedly the most prevalent structure for Internet applications.

The Web, e-mail, file transfer, remote login (for example, Telnet), newsgroups, and many other popular applications adopt the client/server model. Since a client typically runs on one computer and the server runs on another computer, client/server Internet applications are, by definition, distributed applications. The client and the server interact with each other by communicating (that is, sending each other message) over the Internet.

At this level of abstraction, the routers, links and other ‘pieces’ of the Internet serve as a ‘black box’ that transfers messages between the distributed, communicating components of an Internet application.

Computers (for example, a PC or a workstation), operating as clients and servers, are the most prevalent type of end system. However, an increasing number of alternative devices, such as so-called network computers and thin clients, Web TVs and set top boxes, digital cameras, etc. are being attached to the Internet as end systems.

Connectionless and Connection-Oriented Services

We have seen that end systems exchange messages with each other according to an application-level protocol in order to accomplish some task. The links, routers, and other pieces of the Internet provide the means to transport these messages between the end-system applications. But what are the characteristics of the communication services that are provided? The Internet, and more generally TCP/IP networks, provides two types of services to its applications: **connectionless service** and **connection-oriented service**.

A developer creating an Internet application (for example, an e-mail application, a file transfer application, a Web application, or an Internet phone application) must program the application to use one of these two services.

Visual 11

Network edge: connection-oriented service

<p><u>Goal:</u> data transfer between end sys.</p> <ul style="list-style-type: none"> □ <u>Handshaking:</u> setup (prepare for) data transfer ahead of time <ul style="list-style-type: none"> ○ Hello, hello back human protocol ○ <i>set up "state"</i> in two communicating hosts □ TCP - Transmission Control Protocol <ul style="list-style-type: none"> ○ Internet's connection-oriented service 	<p><u>TCP service</u> [RFC 793]</p> <ul style="list-style-type: none"> □ <u>Reliable, in-order</u> byte-stream data transfer <ul style="list-style-type: none"> ○ loss: acknowledgements and retransmissions □ <u>Flow control:</u> <ul style="list-style-type: none"> ○ sender won't overwhelm receiver □ <u>Congestion control:</u> <ul style="list-style-type: none"> ○ senders "slow down sending rate" when network congested
--	---

Connection-Oriented Service

When an application uses the connection-oriented service, the client and the server (residing in different end systems) send control packets to each other before sending packets with real data (such as e-mail messages). This so-called handshaking procedure alerts the client and server, allowing them to prepare for an onslaught of packets. It is interesting to note that this initial handshaking procedure is similar to the protocol used in human interaction.

The exchange of “Hi’s” we saw is an example of a human ‘handshaking protocol’ (even though handshaking is not literally taking place between the two people). The two TCP messages that are exchanged as part of the WWW interaction detailed in the above visual are two of the three messages exchanged when TCP sets up a connection between a sender and receiver. The third TCP message (not shown) that forms the final part of the TCP three-way handshake is contained in the get message shown in the above diagram.

Once the handshaking procedure is finished, a connection is said to be established between the two end systems. But the two end systems are connected in a very loose manner, hence the terminology connection-oriented. In particular, only the end systems themselves are aware of this connection; the packet switches (that is, routers) within the Internet are completely oblivious to the connection. This is because a TCP connection consists of nothing more than allocated resources (buffers) and state variables in the end systems. The packet switches do not maintain any connection-state information.

The Internet’s connection-oriented service comes bundled with several other services, including reliable data transfer, flow control, and congestion control. By reliable data transfer, we mean that an application can rely on the connection to deliver all of its data without error and in the proper order.

Reliability in the Internet is achieved through the use of acknowledgements and retransmissions. To get a preliminary idea about how the Internet implements the reliable transport service, consider an application that has established a connection between end systems A and B:

- When end system B receives a packet from A, it sends an acknowledgement; when end system A receives the acknowledgement, it knows that the corresponding packet has definitely been received.
- When end system A doesn’t receive an acknowledgement, it assumes that the packet it sent was not received by B; it therefore retransmits the packet.

Flow control makes sure that neither side of a connection overwhelms the other side by sending too many packets too fast. Indeed, the application at one side of the connection may not be able to process information as quickly as it receives the information. Therefore, there is a risk of overwhelming either side of an application. The flow-control service forces the sending end system to reduce its rate whenever there is such a risk. We shall see that the Internet implements the flow control service by using sender and receiver buffers in the communicating end systems. The Internet’s congestion-control service helps prevent the Internet from entering a state of gridlock.

When a router becomes congested, its buffers can overflow and packet loss can occur. In such circumstances, if every pair of communicating end systems continues to pump

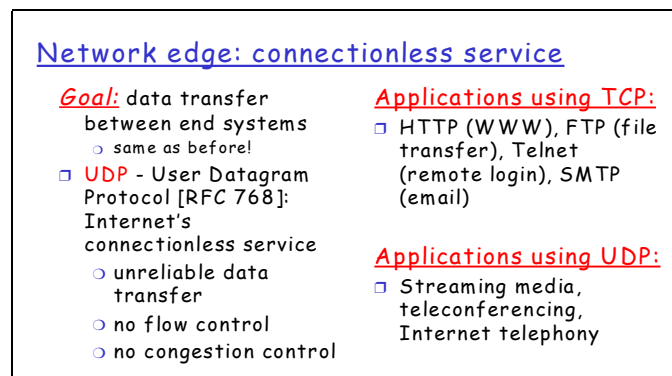
packets into the network as fast as they can, gridlock sets in and few packets are delivered to their destinations. The Internet avoids this problem by forcing end systems to decrease the rate at which they send packets into the network during periods of congestion. End systems are alerted to the existence of severe congestion when they stop receiving acknowledgements for the packets they have sent.

We emphasize here that although the Internet's connection-oriented service comes bundled with reliable data transfer, flow control, and congestion control, these three features are by no means essential components of a connection-oriented service. A different type of computer network may provide a connection-oriented service to its applications without bundling in one or more of these features. Indeed, any protocol that performs handshaking between the communicating entities before transferring data is a connection-oriented service.

The Internet's connection-oriented service has a name – TCP (Transmission Control Protocol); the initial version of the TCP protocol is defined in the Internet Request for Comments RFC 793 [RFC 793]. The services that TCP provides to an application include reliable transport, flow control, and congestion control. It is important to note that an application need only care about the services that are provided; it need not worry about how TCP actually implements reliability, flow control, or congestion control.

Connectionless Service

Visual 12



There is no handshaking with the Internet's connectionless service. When one side of an application wants to send packets to another side of an application, the sending application simply sends the packets. Since there is no handshaking procedure prior to the transmission of the packets, data can be delivered faster. But there are no acknowledgements either, so a source never knows for sure which packets arrive at the destination. Moreover, the service makes no provision for flow control or congestion control. The Internet's connectionless service is provided by User Datagram Protocol (UDP); UDP is defined in the Internet RFC 768.

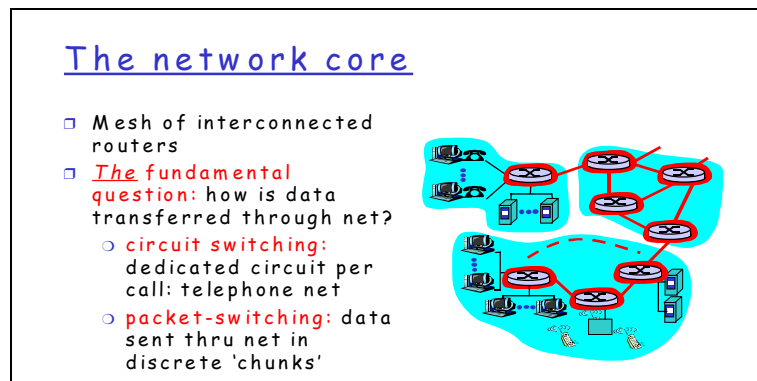
Most of the more familiar Internet applications use TCP, the Internet's connection-oriented service. These applications include Telnet (remote login), SMTP (for electronic mail), FTP (for file transfer), and HTTP (for the Web). Nevertheless, UDP, the Internet's connectionless service, is used by many applications, including many of the emerging multimedia applications, such as Internet phone, audio-on-demand, and video conferencing.

The Network Core

Circuit Switching, Packet Switching, and Message Switching

There are two fundamental approaches towards building a network core: **circuit switching** and **packet switching**. In circuit-switched networks, the resources needed along a path (buffers, link bandwidth) to provide for communication between the end systems are reserved for the duration of the session. In packet-switched networks, these resources are not reserved; a session's messages use the resource on demand, and as a consequence, may have to wait (that is, queue) for access to a communication link.

Visual 13



As a simple analogy, consider two restaurants – one that requires reservations and another that neither requires reservations nor accepts them. For the restaurant that requires reservations, we have to go through the hassle of first calling before we leave home. But when we arrive at the restaurant we can, in principle, immediately communicate with the waiter and order our meal. For the restaurant that does not require reservations, we don't need to bother to reserve a table. But when we arrive at the restaurant, we may have to wait for a table before we can communicate with the waiter.

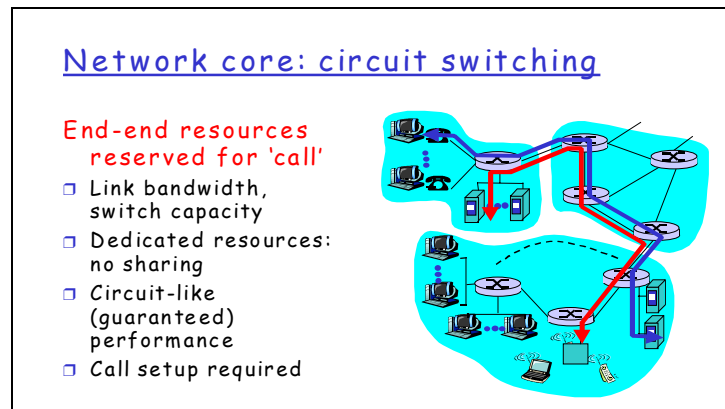
Telephone networks are examples of circuit-switched networks. Consider what happens when one person wants to send information (voice or facsimile) to another over a telephone network. Before the sender can send the information, the network must first establish a connection between the sender and the receiver. In contrast with the TCP connection that we discussed in the previous section, this is a bona fide connection for which the switches on the path between the sender and receiver maintain connection state for that connection. In the jargon of telephony, this connection is called a circuit. When the network establishes the circuit, it also reserves a constant transmission rate in the network's links for the duration of the connection. This reservation allows the sender to transfer the data to the receiver at the guaranteed constant rate.

Today's Internet is a quintessential packet-switched network. Consider what happens when one host wants to send a packet to another host over a packet-switched network. As with circuit switching, the packet is transmitted over a series of communication links. But with packet switching, the packet is sent into the network without reserving any bandwidth whatsoever. If one of the links is congested because other packets need to be transmitted over the link at the same time, then our packet will have to wait in a buffer at the sending side of the transmission line, and suffer a delay. The Internet makes its best effort to deliver the data in a timely manner, but it does not make any guarantees.

Not all telecommunication networks can be neatly classified as pure circuit-switched networks or pure packet-switched networks. For example, for networks based on the ATM technology, a connection can make a reservation and yet its messages may still wait for congested resources! Nevertheless, this fundamental classification into packet-switched and circuit-switched networks is an excellent starting point in understanding telecommunication network technology.

Circuit Switching

Visual 14

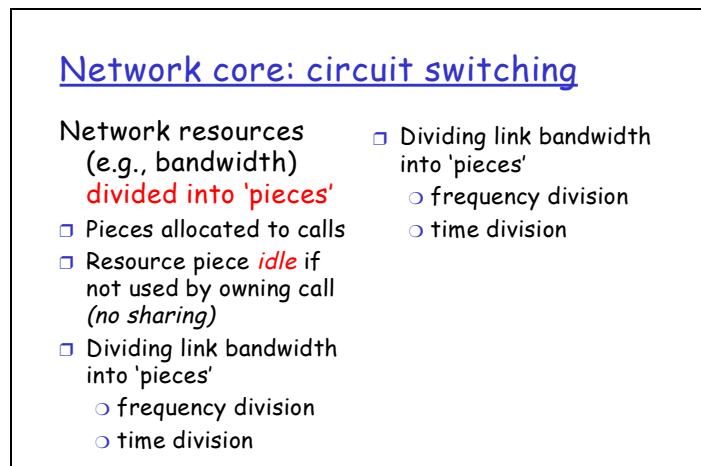


This course is about computer networks, the Internet, and packet switching, not about telephone networks and circuit switching. Nevertheless, it is important to understand why the Internet and other computer networks use packet switching rather than the more traditional circuit-switching technology used in the telephone networks. For this reason, we now give a brief overview of circuit switching.

In this network, the three circuit switches are interconnected by two links; each of these links has n circuits, so that each link can support n simultaneous connections. The end systems (for example, PCs and workstations) are each directly connected to one of the switches. (Ordinary telephones are also connected to the switches, but they are not shown in the diagram.) Notice that some of the hosts have analogue access to the switches, whereas others have direct digital access.

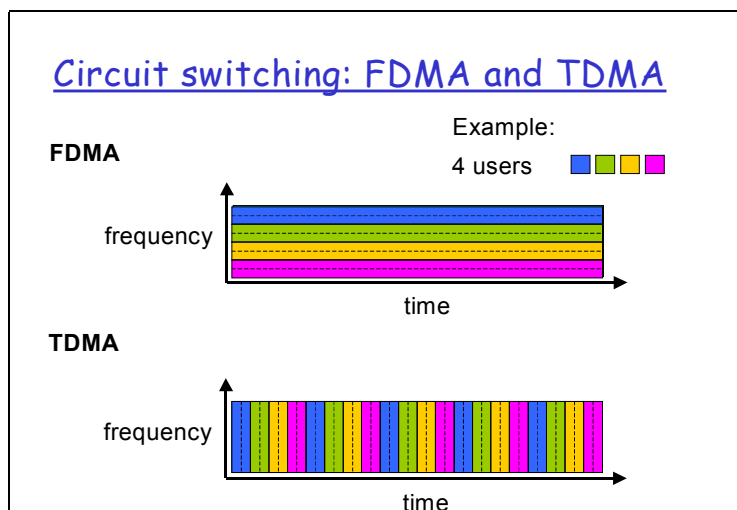
For analogue access, a modem is required. When two hosts desire to communicate, the network establishes a dedicated end-to-end circuit between two hosts. (Conference calls between more than two devices are, of course, also possible. But to keep things simple, let us suppose for now that there are only two hosts for each connection.)

Thus, in order for host A to send messages to host B, the network must first reserve one circuit on each of the two links. Each link has n circuits; each end-to-end circuit over a link gets the fraction $1/n$ of the link's bandwidth for the duration of the circuit.

Visual 15

Most types of telephone networks are examples of circuit-switched networks. Consider what happens when one person wants to send information (voice or facsimile) to another over a telephone network. Before the sender can send the information, the network must first establish a connection between the sender and the receiver.

In contrast with the TCP connection that we discussed in the previous section, this is a bona fide connection for which the switches on the path between the sender and receiver maintain connection state for that connection. In the jargon of telephony, this connection is called a circuit. When the network establishes the circuit, it also reserves a constant transmission rate in the network's links for the duration of the connection. This reservation allows the sender to transfer the data to the receiver at the guaranteed constant rate.

Visual 16

A circuit in a link is implemented with either frequency-division multiplexing (FDM) or time-division multiplexing (TDM). With FDM, the frequency spectrum of a link is shared among the connections established across the link. Specifically, the link dedicates a frequency band to each connection for the duration of the connection. In telephone networks, this frequency band typically has a width of 4 kHz (that is, 4,000 Hertz or 4,000 cycles per second). The width of the band is called, not surprisingly, the bandwidth. FM radio stations also use FDM to share the microwave frequency spectrum.

The trend in modern telephony is to replace FDM with TDM. Most links in most telephone systems in the United States and in other developed countries currently employ TDM. For a TDM link, time is divided into frames of fixed duration, and each frame is divided into a fixed number of time slots. When the network establishes a connection across a link, the network dedicates one time slot in every frame to the connection. These slots are dedicated for the sole use of that connection, with a time slot available for use (in every frame) to transmit the connection's data.

The previous diagram (Visual 16) illustrates FDM and TDM for a specific network link. For FDM, the frequency domain is segmented into a number of circuits, each of bandwidth 4 KHz. For TDM, the time domain is segmented into four circuits; each circuit is assigned the same dedicated slot in the revolving TDM frames. The transmission rate of each circuit is equal to the frame rate multiplied by the number of bits in a slot. For example, if the link transmits 8,000 frames per second and each slot consists of 8 bits, then the circuit transmission rate is 64 Kbps.

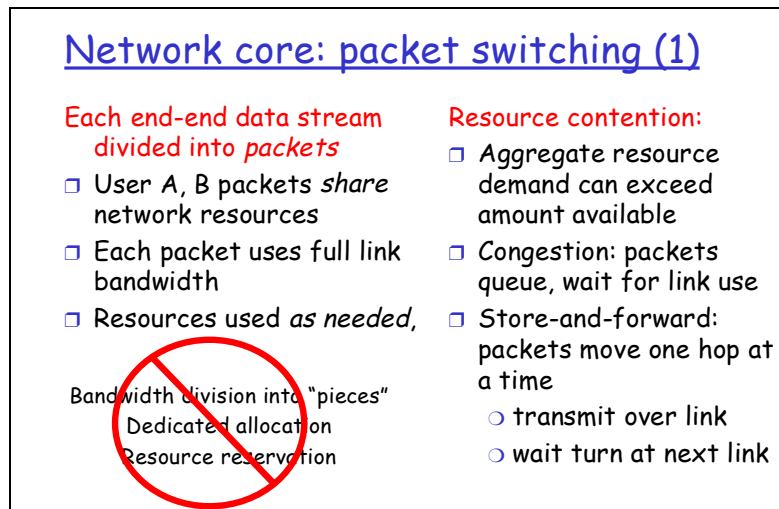
With FDM, each circuit continuously gets a fraction of the bandwidth. With TDM, each circuit gets all of the bandwidth periodically during brief intervals of time (that is, during slots).

Proponents of packet switching have always argued that circuit switching is wasteful because the dedicated circuits are idle during silent periods. For example, when one of the participants in a telephone call stops talking, the idle network resources (frequency bands or slots in the links along the connection's route) cannot be used by other ongoing connections. As another example of how these resources can be under utilized, consider a radiologist who uses a circuit-switched network to remotely access a series of x-rays. The radiologist sets up a connection, requests an image, contemplates the image, and then requests a new image. Network resources are wasted during the radiologist's contemplation periods. Proponents of packet switching also enjoy pointing out that establishing end-to-end circuits and reserving end-to-end bandwidth is complicated and requires complex signalling software to coordinate the operation of the switches along the end-to-end path.

Before we finish our discussion of circuit switching, let us work through a numerical example that should shed further insight on the matter. Let us consider how long it takes to send a file of 640 Kbits from host A to host B over a circuit-switched network. Suppose that all links in the network use TDM with 24 slots and have a bit rate of 1.536 Mbps. Also suppose that it takes 500 msec to establish an end-to-end circuit before A can begin to transmit the file. How long does it take to send the file? Each circuit has a transmission rate of $(1.536 \text{ Mbps})/24 = 64 \text{ Kbps}$, so it takes $(640 \text{ Kbits})/(64 \text{ Kbps}) = 10$ seconds to transmit the file. To this 10 seconds we add the circuit establishment time, giving 10.5 seconds to send the file. Note that the transmission time is independent of the number of links. The transmission time would be 10 seconds if the end-to-end circuit passes through one link or one hundred links.

Packet Switching

Visual 17

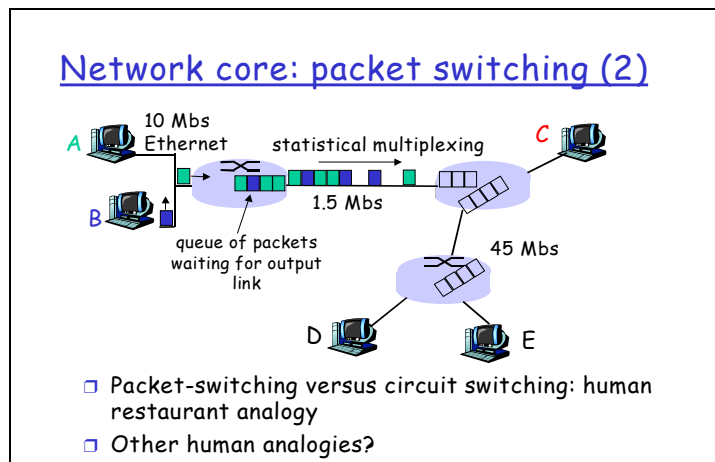


We saw that application-level protocols exchange messages in accomplishing their task. Messages can contain anything the protocol designer desires. Messages may perform a control function (for example, the “Hi” messages in our handshaking example) or can contain data, such as an ASCII file, a Postscript file, a Web page, or a digital audio file. In modern packet-switched networks, the source breaks long messages into smaller packets. Between source and destination, each of these packets traverses communication links and packet switches (also known as routers).

Packets are transmitted over each communication link at a rate equal to the full transmission rate of the link. Most packet switches use store-and-forward transmission at the inputs to the links. Store-and-forward transmission means that the switch must receive the entire packet before it can begin to transmit the first bit of the packet onto the outbound link. Thus store-and-forward packet switches introduce a store-and-forward delay at the input to each link along the packet’s route. This delay is proportional to the packet’s length in bits. In particular, if a packet consists of L bits, and the packet is to be forwarded onto an outbound link of R bps, then the store-and-forward delay at the switch is L/R seconds.

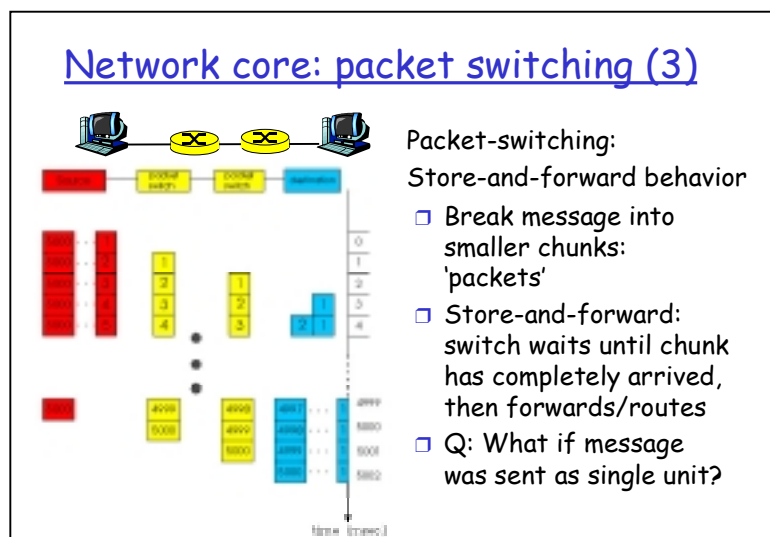
Within each router there are multiple buffers (also called queues), with each link having an input buffer (to store packets that have just arrived to that link) and an output buffer. The output buffers play a key role in packet switching. If an arriving packet needs to be transmitted across a link but finds the link busy with the transmission of another packet, the arriving packet must wait in the output buffer. Thus, in addition to the store-and-forward delays, packets suffer output buffer queuing delays. These delays are variable and depend on the level of congestion in the network. Since the amount of buffer space is finite, an arriving packet may find that the buffer is completely filled with other packets waiting for transmission. In this case, packet loss will occur – either the arriving packet or one of the already-queued packets will be dropped. Returning to our restaurant analogy from earlier in this section, the queuing delay is analogous to the amount of time one spends waiting for a table. Packet loss is analogous to being told by the waiter that you must leave the premises because there are already too many other people waiting at the bar for a table.

Visual 18



This diagram illustrates a simple packet-switched network. Suppose Hosts A and B are sending packets to Host E. Hosts A and B first send their packets along the 10 Mbps link to the first packet switch. The packet switch directs these packets to the 1.544 Mbps link. If there is congestion at this link, the packets queue in the link's output buffer before they can be transmitted onto the link. Consider now how Host A and Host B packets are transmitted onto this link. As shown in the above diagram, the sequence of A and B packets does not follow any periodic ordering; the ordering is random or statistical because packets are sent whenever they happen to be present at the link. For this reason, we often say that packet switching employs statistical multiplexing. Statistical multiplexing sharply contrasts with time-division multiplexing (TDM), for which each host gets the same slot in a revolving TDM frame.

Visual 19

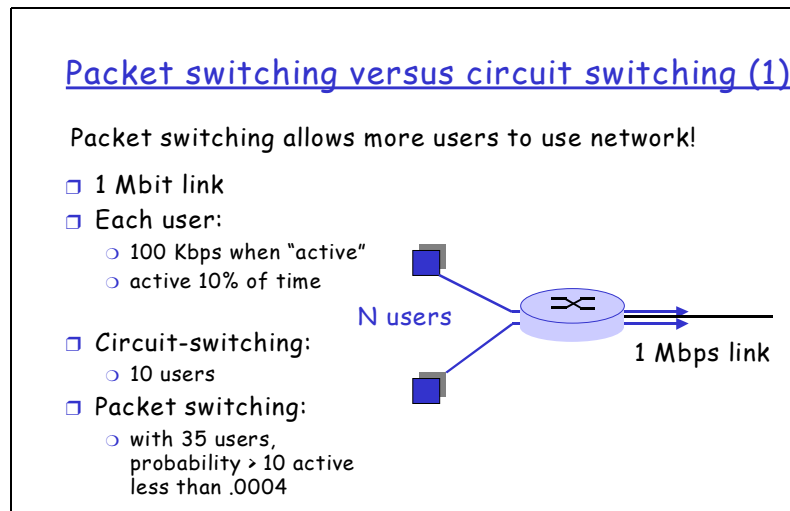


Application level protocol exchange messages in accomplishing the task. In modern packet switched networks the source breaks long messages in smaller packets. Most packet switches use **store-and-forward transmission** at the input to the links.

Store-and-forward transmission means that the switch must receive the entire packet before it can transmit the first bit of the packet on to the outbound link. Thus the store-and-forward packet switches induce a store-and-forward delay at the input link along the packet's route.

Packet Switching versus Circuit Switching

Visual 20



Having described circuit switching and packet switching, let us compare the two. Opponents of packet switching have often argued that packet switching is not suitable for real-time services (for example, telephone calls and video conference calls) because of its variable and unpredictable delays.

Proponents of packet switching argue that (1) it offers better sharing of bandwidth than circuit switching and (2) it is simpler, more efficient, and less costly to implement than circuit switching. Generally speaking, people who do not like to hassle with restaurant reservations prefer packet switching to circuit switching.

Why is packet switching more efficient? Let us look at a simple example. Suppose users share a 1 Mbps link. Also suppose that each user alternates between periods of activity (when it generates data at a constant rate of 100 Kbps) and periods of inactivity (when it generates no data). Suppose further that a user is active only 10 percent of the time (and is idle drinking coffee during the remaining 90 percent of the time). With circuit switching, 100 Kbps must be reserved for each user at all times. Thus, the link can support only 10 simultaneous users. With packet switching, if there are 35 users, the probability that there are more than 10 simultaneously active users is approximately 0.0004.

If there are 10 or fewer simultaneously active users (which happens with probability 0.9996), the aggregate arrival rate of data is less than or equal to 1 Mbps (the output rate of the link). Thus, users' packets flow through the link essentially without delay, as is the case with circuit switching. When there are more than 10 simultaneously active users, then the aggregate arrival rate of packets will exceed the output capacity of the link.

Therefore the output queue will begin to grow (until the aggregate input rate falls back below 1 Mbps, at which point the queue will begin to diminish in length). Because the probability of having 10 or more simultaneously active users is extremely small, packet-switching almost always has the same delay performance as circuit switching, but does so while allowing for more than three times the number of users.

Although packet switching and circuit switching are both very prevalent in today's telecommunication networks, the trend is certainly in the direction of packet switching. Even many of today's circuit-switched telephone networks are slowly migrating towards packet switching. In particular, telephone networks often convert to packet switching for the expensive overseas portion of a telephone call.

Packet switching has yet another important advantage over message switching. As we will discuss later in this course, bit errors can be introduced into packets as they transit the network. When a switch detects an error in a packet, it typically discards the entire packet. So, if the entire message is a packet and one bit in the message gets corrupted, the entire message is discarded. If, on the other hand, the message is segmented into many packets and one bit in one of the packets is corrupted, then only that one packet is discarded.

Visual 21

Packet switching versus circuit switching (2)

Is packet switching a 'slam dunk winner'?

- Great for bursty data
 - resource sharing
 - no call setup
- Excessive congestion: packet delay and loss
 - protocols needed for reliable data transfer, congestion control
- Q: How to provide circuit-like behaviour?
 - bandwidth guarantees needed for audio/video apps
 - still an unsolved problem (chapter 6)

Packet switching is not without its disadvantages, however.

We will see that each packet or message must carry, in addition to the data being sent from the sending application to the receiving application, an amount of control information. This information, which is carried in the packet or message header, might include the identity of the sender and receiver and a packet or message identifier (for example, number). Since the amount of header information would be approximately the same for a message or a packet, the amount of header overhead per byte of data is higher for packet switching than for message switching.

Routing

Visual 22

Packet-switched networks: routing

- **Goal:** move packets among routers from source to destination
 - we'll study several path selection algorithms (chapter 4)
- **Datagram network:**
 - *destination address* determines next hop
 - routes may change during session
 - analogy: driving, asking directions
- **Virtual circuit network:**
 - each packet carries tag (virtual circuit ID), tag determines next hop
 - fixed path determined at *call setup time*, remains fixed through call
 - routers maintain per-call state

There are two broad classes of packet-switched networks: **datagram networks** and **virtual circuit networks**. They differ according to whether they route packets according to host destination addresses or according to virtual circuit numbers.

We shall call any network that routes packets according to host destination addresses a datagram network. The IP protocol of the Internet routes packets according to the destination addresses; hence the Internet is a datagram network.

We shall call any network that routes packets according to virtual circuit numbers a virtual circuit network. Examples of packet-switching technologies that use virtual circuits include X.25, frame relay, and Asynchronous Transfer Mode (ATM).

Virtual Circuit Networks

A virtual circuit (VC) consists of (1) a path (that is, a series of links and packet switches) between the source and destination hosts, (2) virtual circuit numbers, one number for each link along the path, and (3) entries in VC-number translation tables in each packet switch along the path. Once a VC is established between source and destination, packets can be sent with the appropriate VC numbers. Because a VC has a different VC number on each link, an intermediate packet switch must replace the VC number of each traversing packet with a new one. The new VC number is obtained from the VC-number translation table.

To illustrate the concept, consider the network shown in Figure 1.12. Suppose host A requests that the network establish a VC between itself and host B. Suppose that the network chooses the path A-PS1-PS2-B and assigns VC numbers 12, 22, 32 to the three links in this path. Then, when a packet as part of this VC leaves host A, the value in the VC-number field is 12; when it leaves PS1, the value is 22; and when it leaves PS2, the value is 32. The numbers next to the links of PS1 are the interface numbers.

Datagram Networks

Datagram networks are analogous in many respects to the postal services. When a sender sends a letter to a destination, the sender wraps the letter in an envelope and writes the destination address on the envelope. This destination address has a hierarchical structure. For example, letters sent to a location in the United Kingdom include the country (England), the county (for example, Lancashire), the city (for example, Manchester), the street (for example, Caroline Street) and the number of the house on the street (for example, 306). The postal services use the address on the envelope to route the letter to its destination. For example, if the letter is sent from United Arab Emirates, then a postal office in United Arab Emirates will first direct the letter to a postal centre in the United Kingdom. This postal centre in the United Kingdom will then send the letter to a postal centre in Manchester. Finally, a mail person working in Manchester will deliver the letter to its ultimate destination.

In a datagram network, each packet that traverses the network contains in its header the address of the destination. As with postal addresses, this address has a hierarchical structure. When a packet arrives at a packet switch in the network, the packet switch examines a portion of the packet's destination address and forwards the packet to an adjacent switch. More specifically, each packet switch has a routing table that maps destination addresses (or portions of the destination addresses) to an outbound link. When a packet arrives at a switch, the switch examines the address and indexes its table with this address to find the appropriate outbound link. The switch then sends the packet into this outbound link.

The whole routing process is also analogous to the car driver who does not use maps but instead prefers to ask for directions.

For example, suppose Joe is driving from Philadelphia to 156 Lakeside Drive in Orlando, Florida. Joe first drives to his neighbourhood gas station and asks how to get to 156 Lakeside Drive in Orlando, Florida. The gas station attendant extracts the Florida portion of the address and tells Joe that he needs to get onto the interstate highway I-95 South, which has an entrance just next to the gas station. He also tells Joe that once he enters Florida he should ask someone else there. Joe then takes I-95 South until he gets to Jacksonville, Florida, at which point he asks another gas station attendant for directions. The attendant extracts the Orlando portion of the address and tells Joe that he should continue on I-95 to Daytona Beach and then ask someone else. In Daytona Beach another gas station attendant also extracts the Orlando portion of the address and tells Joe that he should take I-4 directly to Orlando. Joe takes I-4 and gets off at the Orlando exit. Joe goes to another gas station attendant, and this time the attendant extracts the Lakeside Drive portion of the address and tells Joe the road he must follow to get to Lakeside Drive. Once Joe reaches Lakeside Drive he asks a kid on a bicycle how to get to his destination. The kid extracts the 156 portion of the address and points to the house. Joe finally reaches his ultimate destination.

How would you like to actually see the route that packets take in the Internet? We now invite you to get your hands dirty by interacting with the **Tracert** program (Windows) or **traceroute** (Linux)