UNIT 3 PART 1

Quality of Service: Requirements, Techniques for Achieving Good Quality of Service

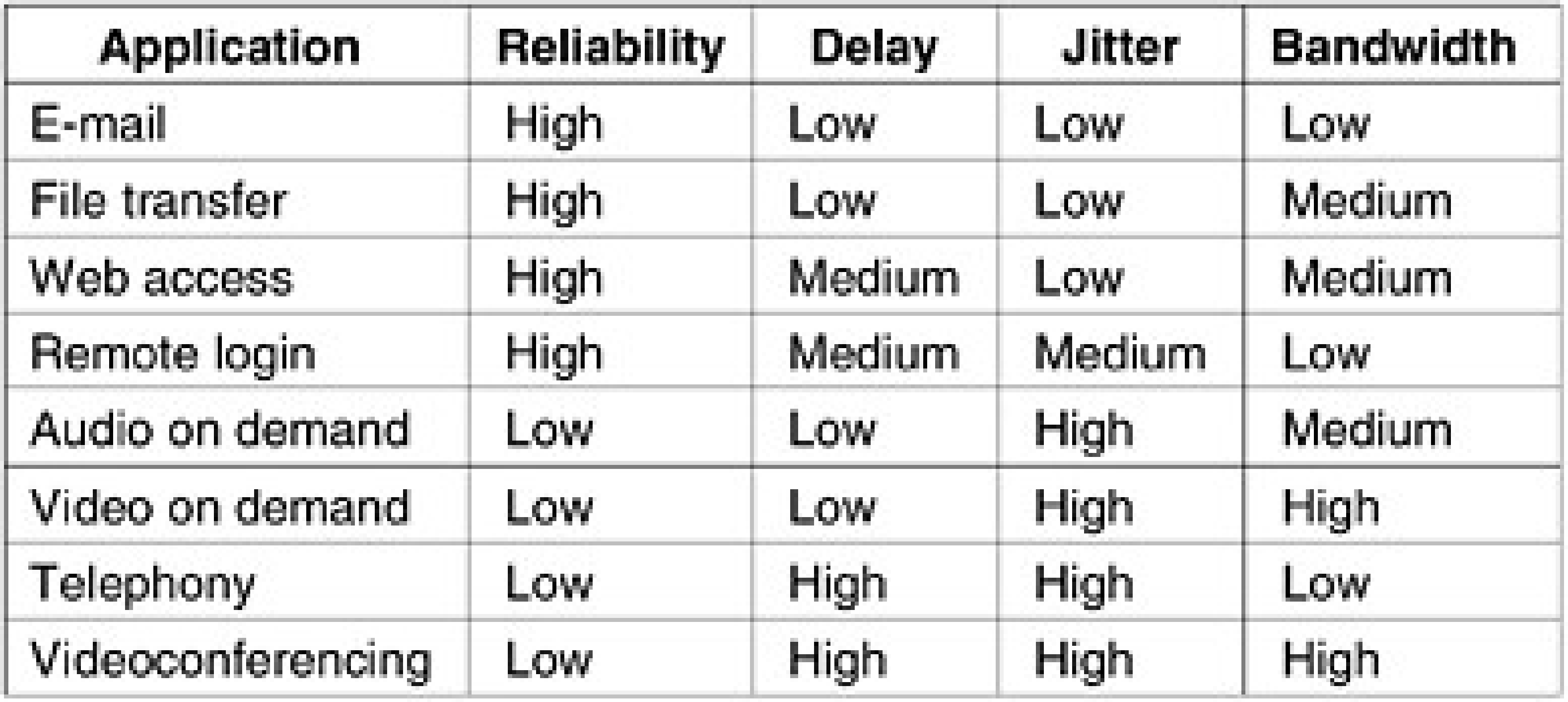
Quality of Service

- The techniques we looked at in the previous sections are designed to **reduce congestion** and **improve network performance.**
- However, with the growth of multimedia networking, often these ad hoc measures are not enough.
- Serious attempts at guaranteeing quality of service through network and protocol design are needed.
- In the following sections we will continue our study of network performance, but now with a sharper focus on ways to provide a quality of service matched to application needs.
- It should be stated at the start, however, that many of these ideas are in flux and are subject to change.

Requirements

- A stream of packets from a source to a destination is called a **flow**.
- In a connection-oriented network, all the packets belonging to a flow follow the same route; in a connectionless network, they may follow different routes.
- The needs of each flow can be characterized by **four** primary parameters: **reliability, delay, jitter, and bandwidth**.
- Together these determine the QoS (Quality of Service) the flow requires.
- Several common applications and the stringency of their requirements are listed in below figure.

How stringent the quality-of-service requirements are?



Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Requirements

- The first four applications have stringent requirements on **reliability**.
- No bits may be delivered incorrectly.
- This goal is usually achieved by checksumming each packet and verifying the checksum at the destination.
- If a packet is damaged in transit, it is not acknowledged and will be retransmitted eventually.
- This strategy gives high reliability.
- The four final (audio/video) applications can tolerate errors, so no checksums are computed or verified.

Requirements

- File transfer applications, including e-mail and video, are not **delay** sensitive. If all packets are delayed uniformly by a few seconds, no harm is done.
- Interactive applications, such as Web surfing and remote login, are more delay sensitive.
- Real-time applications, such as telephony and videoconferencing have strict delay requirements. If all the words in a telephone call are each delayed by exactly 2.000 seconds, the users will find the connection unacceptable.
- On the other hand, playing audio or video files from a server does not require low delay.

Requirements

- The first three applications are not sensitive to the packets arriving with irregular time intervals between them.
- Remote login is somewhat sensitive to that, since characters on the screen will appear in little bursts if the connection suffers much **jitter**.
- Video and especially audio are extremely sensitive to jitter.
- If a user is watching a video over the network and the frames are all delayed by exactly 2.000 seconds, no harm is done.
- But if the transmission time varies randomly between 1 and 2 seconds, the result will be terrible.
- For audio, a jitter of even a few milliseconds is clearly audible.

Requirements

- Finally, the applications differ in their **bandwidth** needs, with e-mail and remote login not needing much, but video in all forms needing a great deal.
- ATM networks classify flows in **four** broad categories with respect to their QoS demands as follows:

1. Constant bit rate (e.g., telephony).
2. Real-time variable bit rate (e.g., compressed videoconferencing).
3. Non-real-time variable bit rate (e.g., watching a movie over the Internet).
4. Available bit rate (e.g., file transfer).

Requirements

- These categories are also useful for other purposes and other networks.
- Constant bit rate is an attempt to simulate a wire by providing a uniform bandwidth and a uniform delay.
- Variable bit rate occurs when video is compressed, some frames compressing more than others.
- Thus, sending a frame with a lot of detail in it may require sending many bits whereas sending a shot of a white wall may compress extremely well.
- Available bit rate is for applications, such as e-mail, that are not sensitive to delay or jitter.

Techniques for Achieving Good Quality of Service

- No single technique provides efficient, dependable QoS in an optimum way.
- Instead, a variety of techniques have been developed, with practical solutions often combining multiple techniques.
- We will now examine some of the techniques system designers use to achieve QoS.

- ➢ Overprovisioning
- ➢ Buffering
- ➢ Traffic Shaping
  - ✓ The Leaky Bucket Algorithm
  - ✓ The Token Bucket Algorithm
- ➢ Resource Reservation
- ➢ Admission Control
- ➢ Proportional Routing
- ➢ Packet Scheduling
  - ✓ Fair Queuing
  - ✓ Weighted Fair Queuing

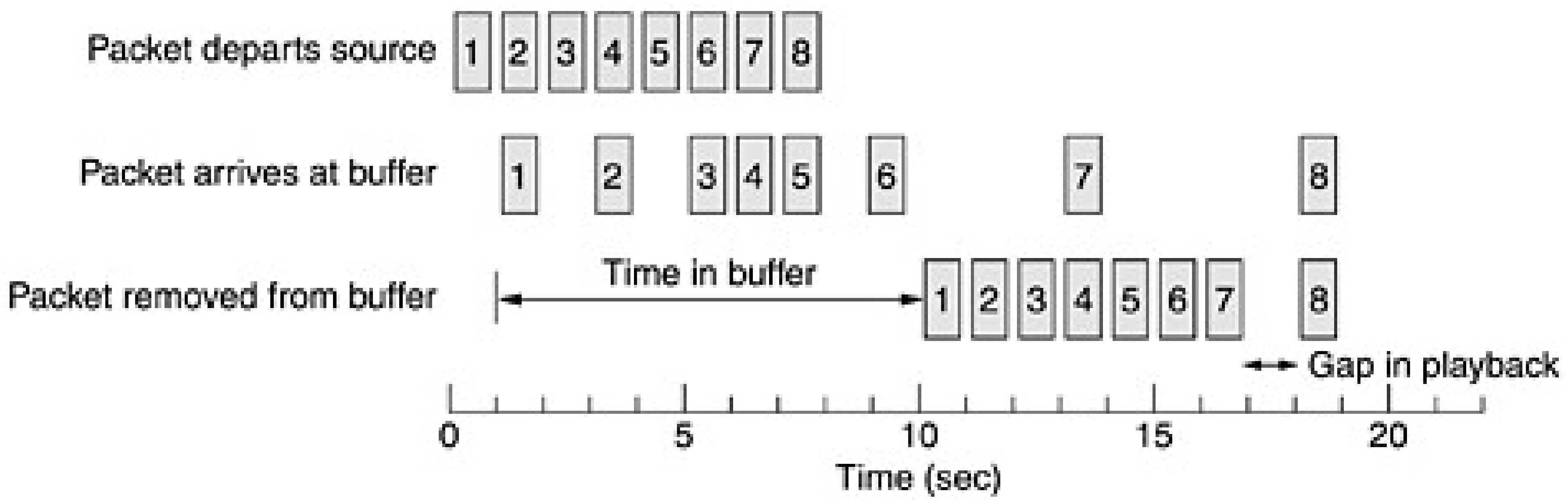
Overprovisioning

- An easy solution is to provide so much router capacity, buffer space, and bandwidth that the packets just fly through easily.
- The trouble with this solution is that it is expensive.

Buffering

- Flows can be buffered on the receiving side before being delivered. Buffering them does not affect the reliability or bandwidth, and increases the delay, but it smooths out the jitter.
- For audio and video on demand, jitter is the main problem, so this technique helps a lot.

Smoothing the output stream by buffering packets



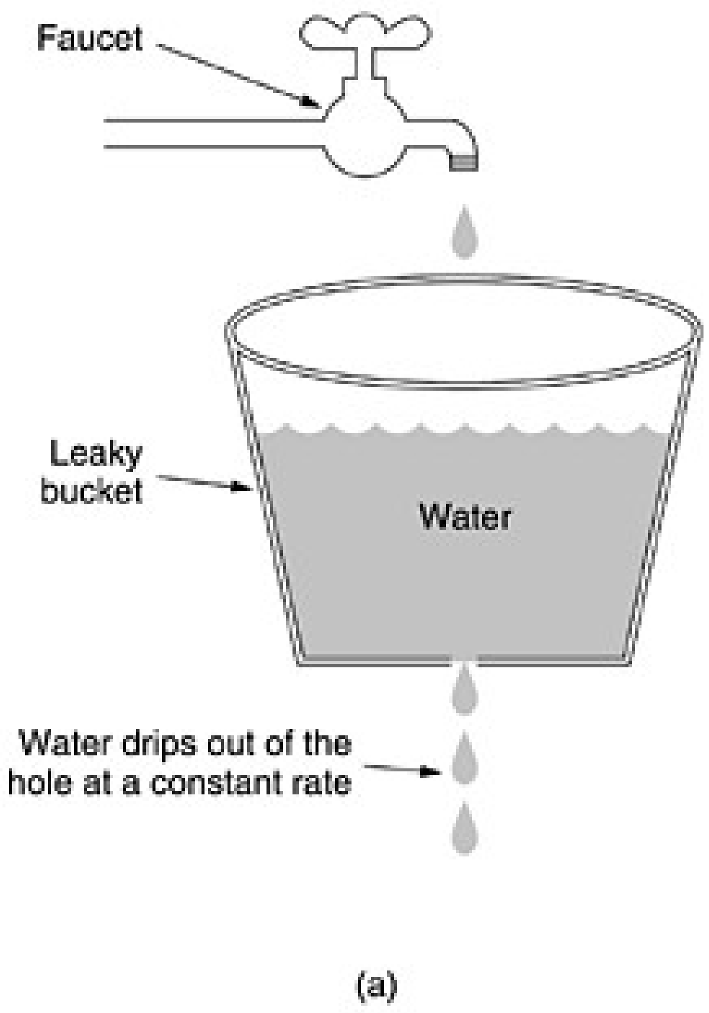
Traffic Shaping

- Traffic shaping is about regulating the average rate (and burstiness) of data transmission.
- Traffic shaping smooths out the traffic on the server side, rather than on the client side.
- When a connection is set up, the user and the subnet (i.e., the customer and the carrier) agree on a certain traffic pattern (i.e., shape) for that circuit. Sometimes this is called a **service level agreement.**
- As long as the customer fulfills her part of the bargain and only sends packets according to the agreed-on contract, the carrier promises to deliver them all in a timely fashion.
- Traffic shaping reduces congestion and thus helps the carrier live up to its promise.
- Such agreements are not so important for file transfers but are of great importance for real-time data, such as audio and video connections, which have stringent quality-of-service requirements.
- Two techniques can shape traffic:
  - Leaky bucket
  - Token bucket.

The Leaky Bucket Algorithm

- If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket.
- The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty.
- The input rate can vary, but the output rate remains constant.
- Similarly, in networking, a technique called **leaky bucket** can smooth out bursty traffic.
- Bursty chunks are stored in the bucket and sent out at an average rate.
- Below figure shows a leaky bucket and its effects.

The Leaky Bucket Algorithm



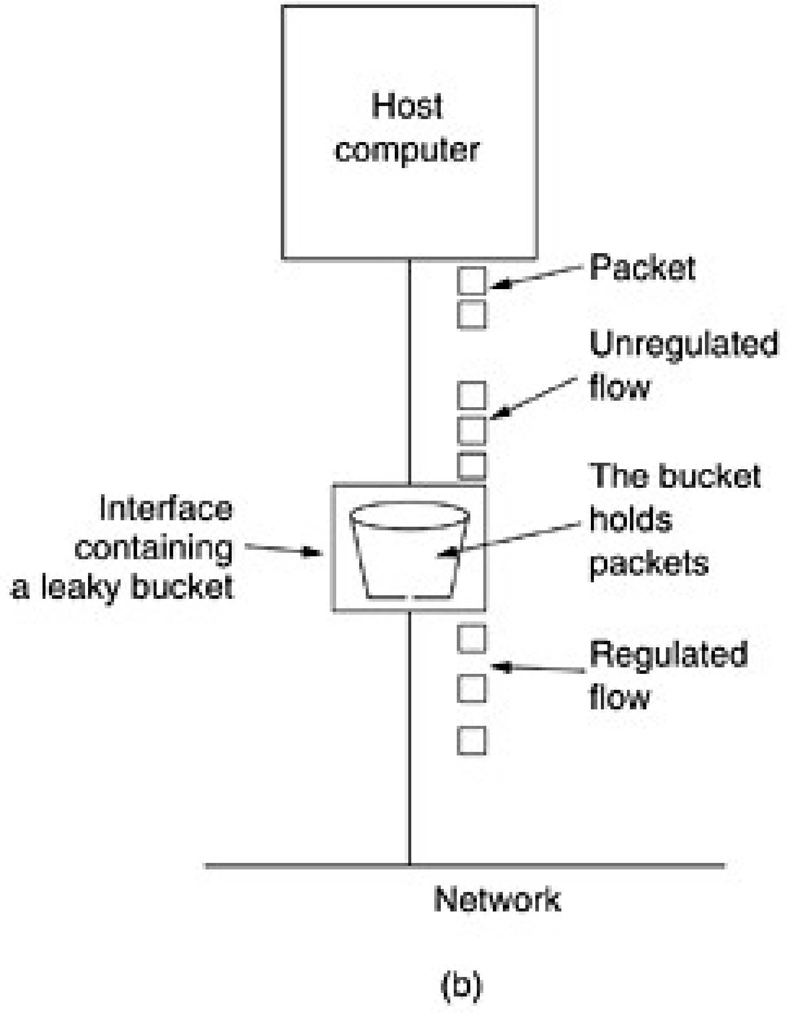
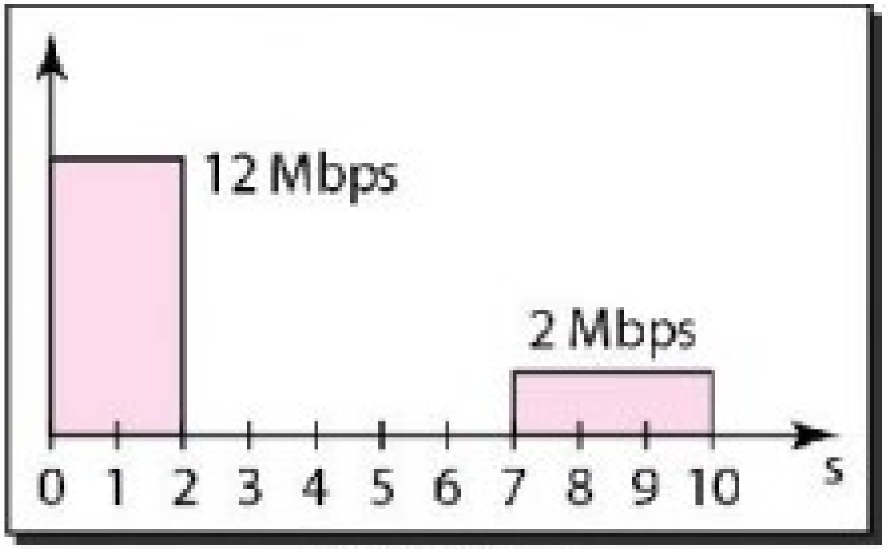


Figure: (a) A leaky bucket with water. (b) A leaky bucket with packets.

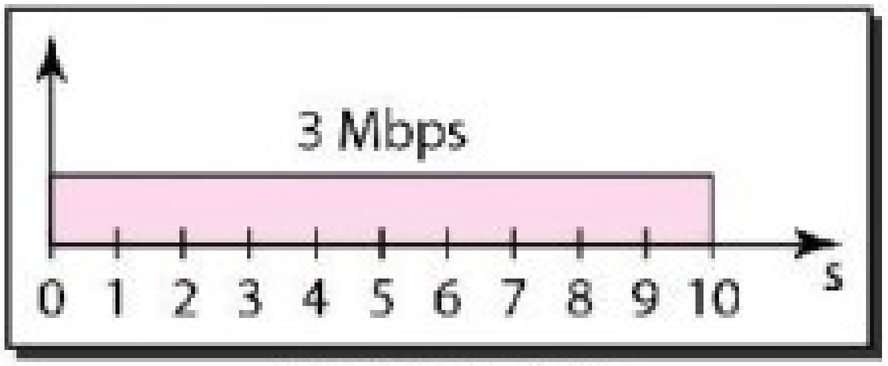
The Leaky Bucket Algorithm

- In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host.
- The use of the leaky bucket shapes the input traffic to make it conform to this commitment.
- In below figure the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data.
- The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data.
- In all, the host has sent 30 Mbits of data in 10 s.
- The leaky bucket smooth’s the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

The Leaky Bucket Algorithm



Bursty data

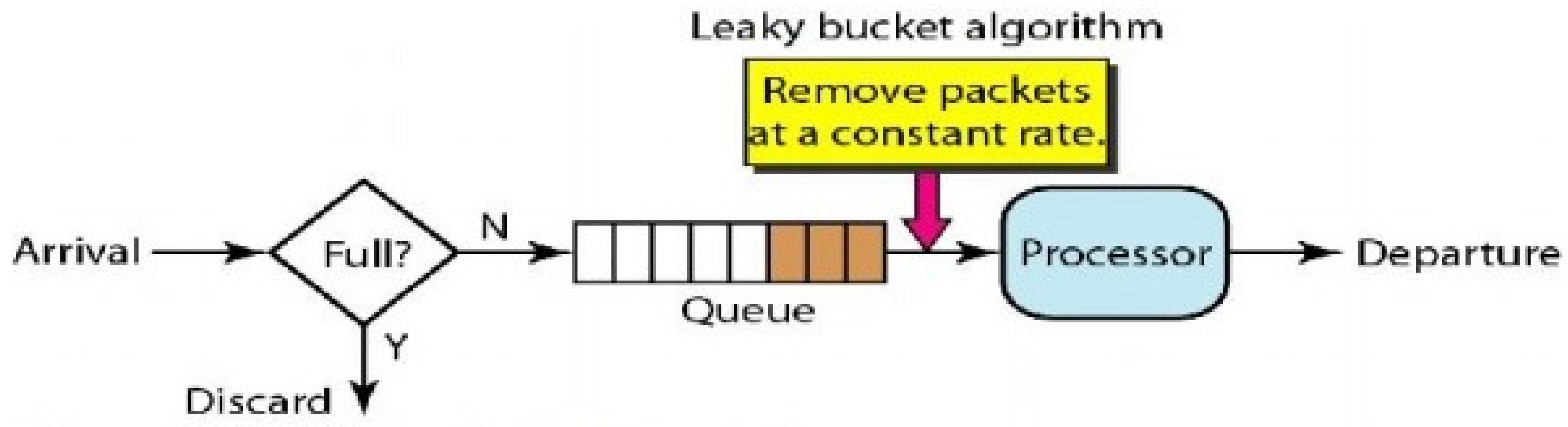


Fixed-rate data

Leaky Bucket Implementation

- A simple leaky bucket implementation is shown in below figure.
- A FIFO queue holds the packets. If the traffic consists of fixed-size packets (e.g., cells in ATM networks), the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.
- The following is an algorithm for variable-length packets:
  - Initialize a counter to n at the tick of the clock.
  - If n is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until n is smaller than the packet size.
  - Reset the counter and go to step 1.
- **A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.**

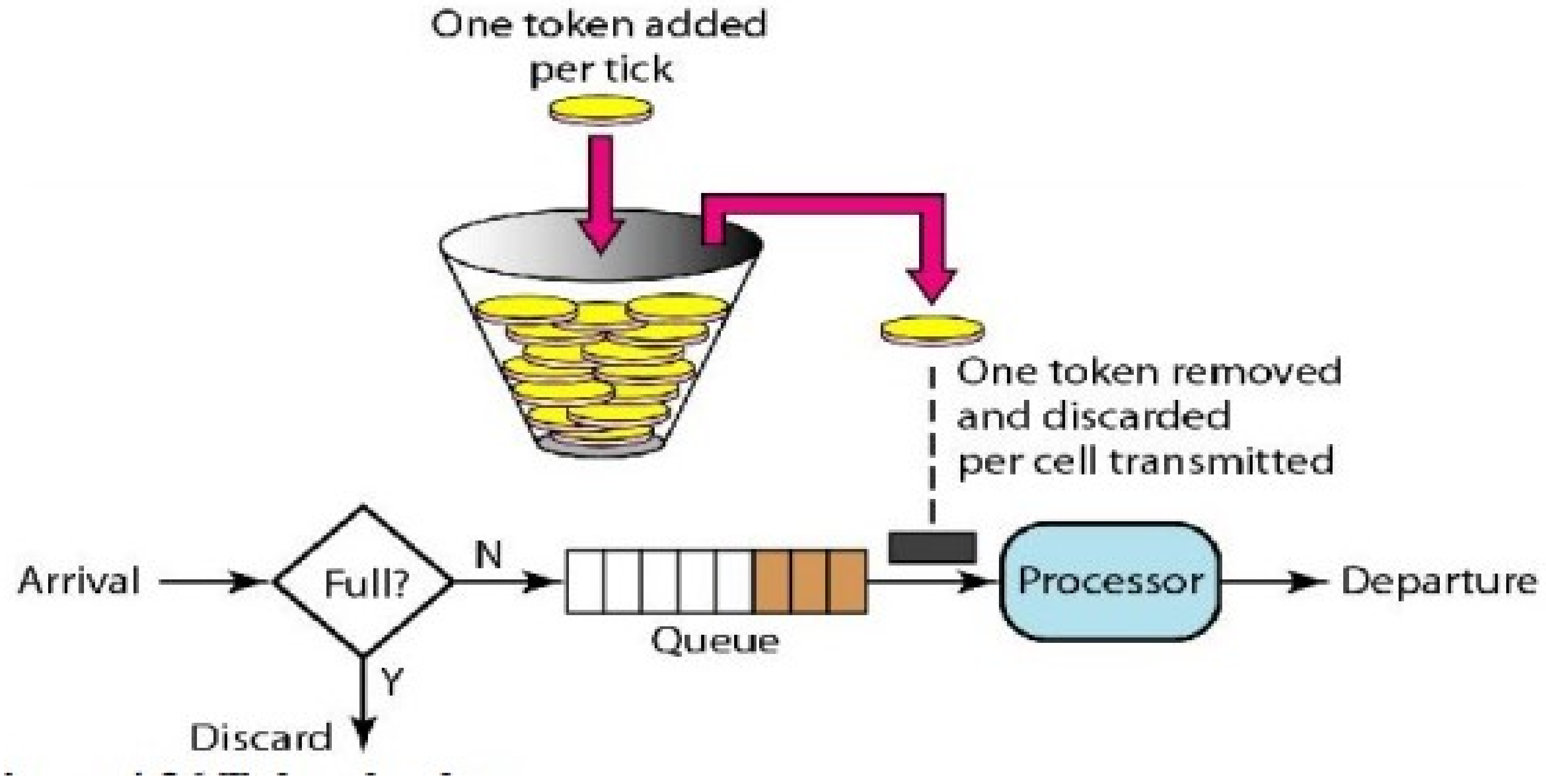
Leaky Bucket Implementation



The Token Bucket Algorithm

- The leaky bucket is very restrictive. It does not credit an idle host.
- For example, if a host is not sending for a while, its bucket becomes empty.
- Now if the host has bursty data, the leaky bucket allows only an average rate.
- The time when the host was idle is not taken into account.
- The **token bucket algorithm** allows idle hosts to accumulate credit for the future in the form of tokens.
- For each tick of the clock, the system sends n tokens to the bucket.
- The system removes one token for every cell (or byte) of data sent.
- For example, if n is 100 and the host is idle for 100 ticks, the bucket collects 10,000 tokens.

The Token Bucket Algorithm



The Token Bucket Algorithm

- The token bucket can easily be implemented with a counter.
  - The token is initialized to zero. Each time a token is added, the counter is incremented by 1.
  - Each time a unit of data is sent, the counter is decremented by 1.
  - When the counter is zero, the host cannot send data.
- **The token bucket allows bursty traffic at a regulated maximum rate.**

Combining Token Bucket and Leaky Bucket

- The two techniques can be combined to credit an idle host and at the same time regulate the traffic.
- The leaky bucket is applied after the token bucket; the rate of the leaky bucket needs to be higher than the rate of tokens dropped in the bucket.

Resource Reservation

- A flow of data needs resources such as a buffer, bandwidth, CPU time, and so on. The quality of service is improved if these resources are reserved beforehand.

Admission Control

- Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called **flow specifications.**
- Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

Admission Control

An example of flow specification.

Parameter	Unit
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

Proportional Routing

- Most routing algorithms try to find the best path for each destination and send all traffic to that destination over the best path.
- A different approach that has been proposed to provide a higher quality of service is to split the traffic for each destination over multiple paths.
- Since routers generally do not have a complete overview of network-wide traffic, the only feasible way to split traffic over multiple routes is to use locally-available information.
- A simple method is to divide the traffic equally or in proportion to the capacity of the outgoing links.

Packet Scheduling

- If a router is handling multiple flows, there is a danger that one flow will hog too much of its capacity and starve all the other flows.
- Processing packets in the order of their arrival means that an aggressive sender can capture most of the capacity of the routers its packets traverse, reducing the quality of service for others.
- To thwart such attempts, various packet scheduling algorithms have been devised.
  - ✓ Fair Queuing
  - ✓ Weighted Fair Queuing

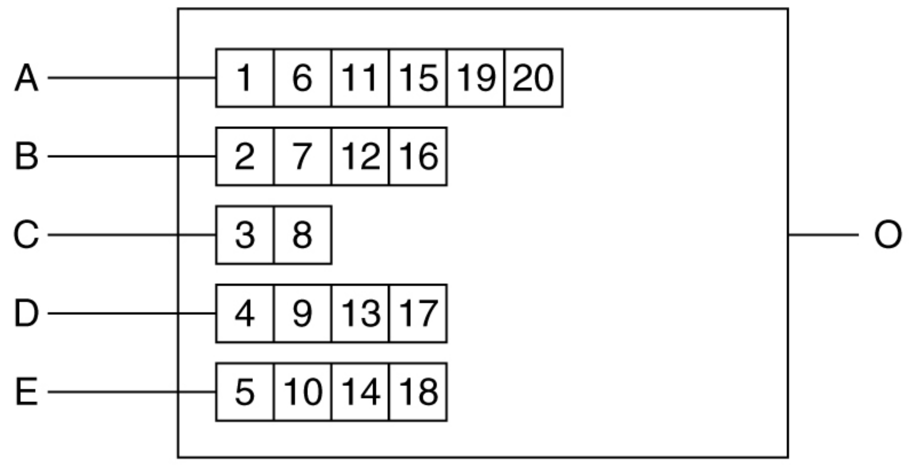
Fair Queuing

- The essence of the algorithm is that routers have separate queues for each output line, one for each flow.
- When a line becomes idle, the router scans the queues round robin, taking the first packet on the next queue.
- In this way, with n hosts competing for a given output line, each host gets to send one out of every n packets.
- Sending more packets will not improve this fraction.

Fair Queuing

- Although a start, the algorithm has a problem: it gives more bandwidth to hosts that use large packets than to hosts that use small packets.
- Demers et al. (1990) suggested an improvement in which the round robin is done in such a way as to simulate a byte-by-byte round robin, instead of a packet-by-packet round robin.
- In effect, it scans the queues repeatedly, byte-for-byte, until it finds the tick on which each packet will be finished.
- The packets are then sorted in order of their finishing and sent in that order.
- The algorithm is illustrated in below figure.

Fair Queuing



(a)

Packet	Finishing time
C	8
B	16
D	17
E	18
A	20

(b)

Fig (a) A router with five packets queued for line O. (b) Finishing times for the five packets.

In Fig(a) we see packets of length 2 to 6 bytes. At (virtual) clock tick 1, the first byte of the packet on line A is sent. Then goes the first byte of the packet on line B, and so on. The first packet to finish is C, after eight ticks. The sorted order is given in Fig(b). In the absence of new arrivals, the packets will be sent in the order listed, from C to A.

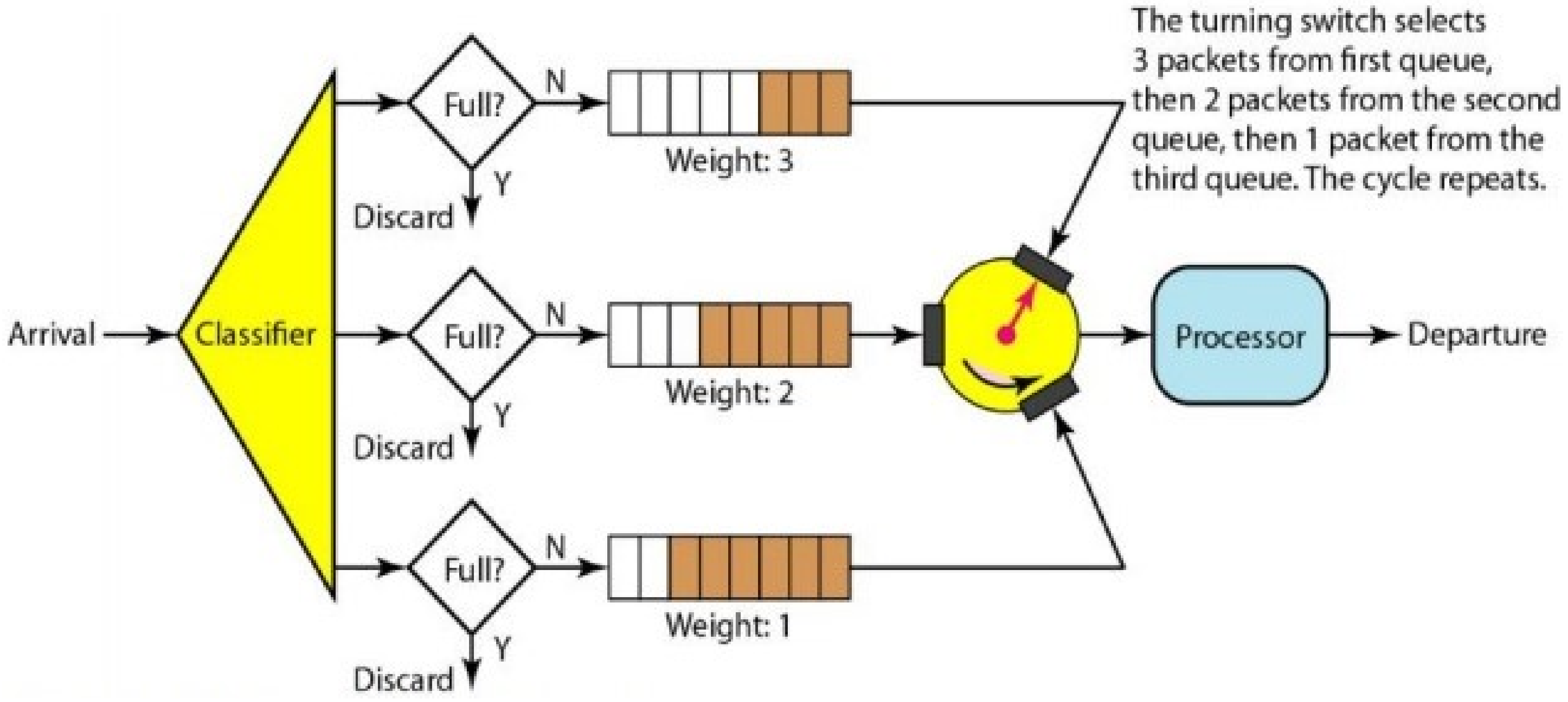
Weighted Fair Queuing

- One **problem with fair queuing algorithm** is that it gives all hosts the same priority.
- In many situations, it is desirable to give video servers more bandwidth than regular file servers so that they can be given two or more bytes per tick.
- This modified algorithm is called **weighted fair queuing** and is widely used.

Weighted Fair Queuing

- In this technique, the packets are still assigned to different classes and admitted to different queues.
- The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight.
- The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight.
- For example, if the weights are 3, 2, and 1, three packets are processed from the first queue, two from the second queue, and one from the third queue.
- If the system does not impose priority on the classes, all weights can be equal.
- In this way, we have fair queuing with priority.
- Below figure shows the technique with three classes.

Weighted Fair Queuing



The turning switch selects 3 packets from first queue, then 2 packets from the second queue, then 1 packet from the third queue. The cycle repeats.

UNIT 3 PART 2

The Network Layer in the Internet: The IP Protocol, IP Addresses, Internet Control Protocols.

The Network Layer in the Internet

- Principles of the network layer in the internet are as follows:

1. **Make sure it works –** The design of network layer must not finalized until there is a goof communication between multiple prototypes.
2. **Keep it simple –** The design must be made as simple as possible.
3. **Make clear choices -** If there are several ways of doing the same thing, choose one. Having two or more ways to do the same thing is looking for trouble.
4. **Exploit modularity -** This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones.

Cont..

The Network Layer in the Internet

5. **Expect heterogeneity -** Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.
6. **Avoid static options and parameters -** If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value than defining fixed choices.
7. **Look for a good design, It need not be perfect -** The designers must choose a good design that has the capability of handling the complex situations.

Cont..

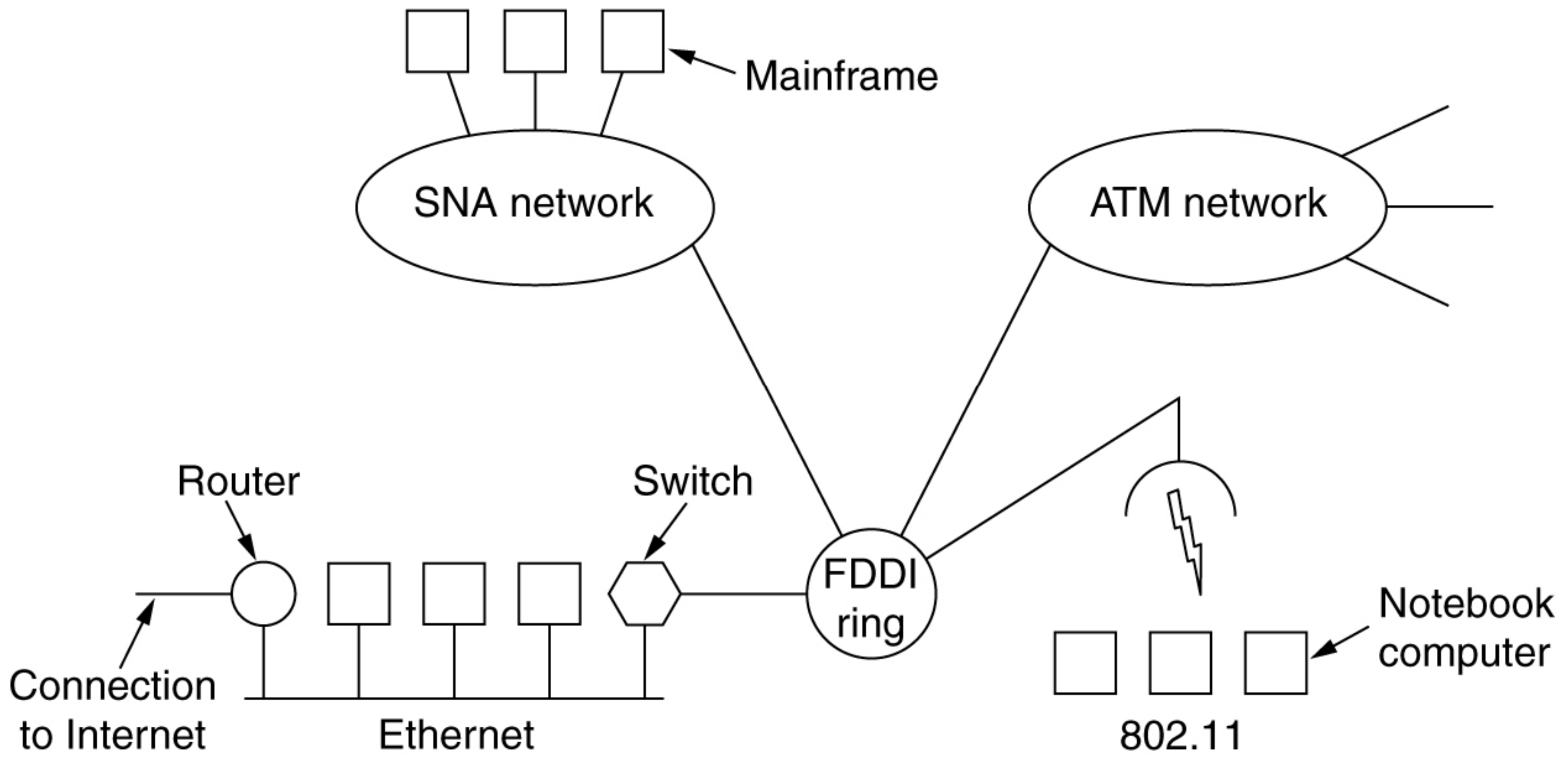
The Network Layer in the Internet

8. **Be strict when sending and tolerant when receiving** - Send packets that rigorously comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.
9. **Think about scalability** - If the system is to handle millions of hosts and billions of users effectively, no centralized databases of any kind are tolerable and load must be spread as evenly as possible over the available resources.
10. **Consider performance and cost** – The network that gives good performance or cost less must be designed.

The Network Layer in the Internet

- At the network layer, the Internet can be viewed as a collection of **subnetworks** or **Autonomous Systems (ASes)** that are interconnected.
- There is no real structure, but several major backbones exist.
- These are constructed from high-bandwidth lines and fast routers.
- Attached to the backbones are regional (midlevel) networks, and attached to these regional networks are the LANs at many universities, companies, and Internet service providers.
- A sketch of this quasi-hierarchical organization is given in below figure.

Connecting Networks



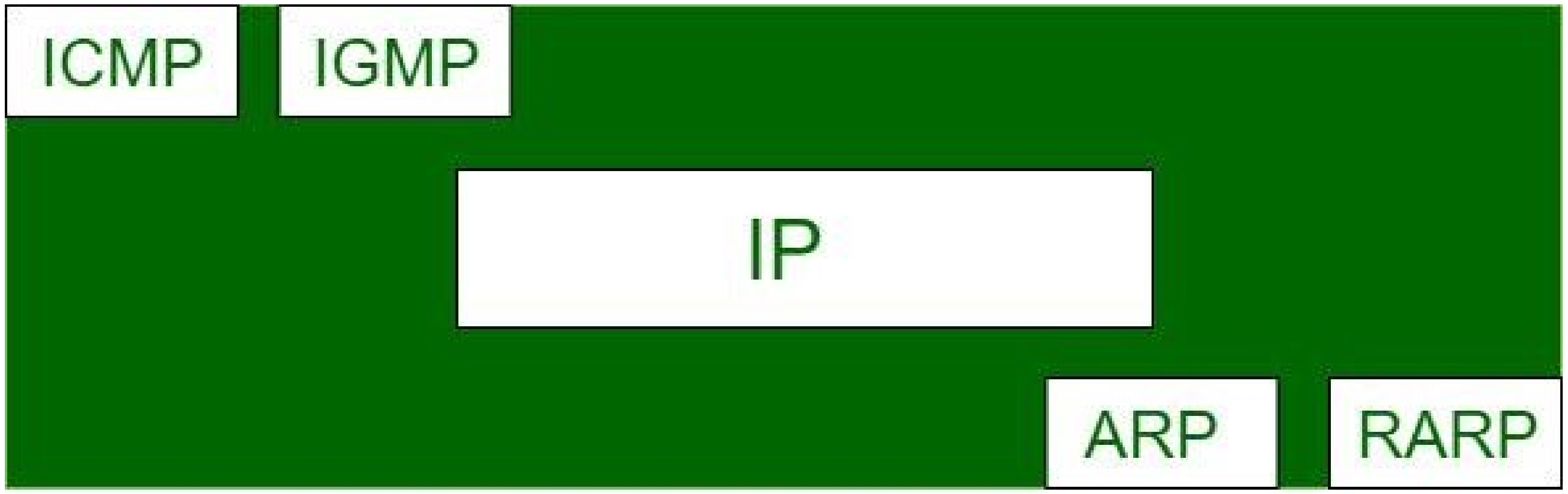
The Network Layer in the Internet

- The glue that holds the whole Internet together is the network layer protocol, IP (Internet Protocol).
- Unlike most older network layer protocols, it was designed from the beginning with internetworking in mind.
- A good way to think of the network layer is this.
- Its job is to provide a best-efforts (i.e., not guaranteed) way to transport datagrams from source to destination, without regard to whether these machines are on the same network or whether there are other networks in between them.

The Network Layer in the Internet

- Communication in the Internet works as follows -
  - The transport layer takes data streams and breaks them up into datagrams.
  - In theory, datagrams can be up to 64 Kbytes each, but in practice they are usually not more than 1500 bytes (so they fit in one Ethernet frame).
  - Each datagram is transmitted through the Internet, possibly being fragmented into smaller units as it goes.
  - When all the pieces finally get to the destination machine, they are reassembled by the network layer into the original datagram.
  - This datagram is then handed to the transport layer, which inserts it into the receiving process' input stream.
  - As can be seen from above figure, a packet originating at host 1 has to traverse six networks to get to host 2.
  - In practice, it is often much more than six.

Network Layer Protocols



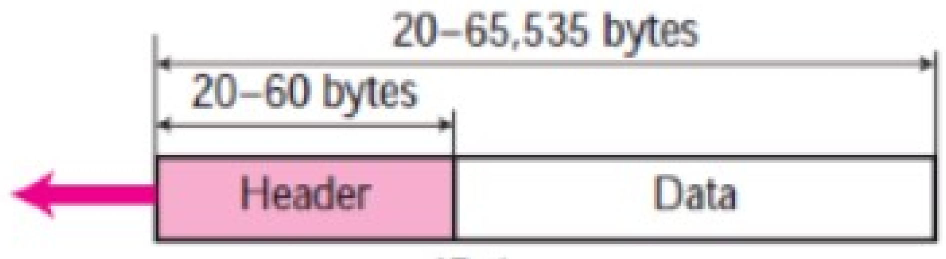
DLL

- Hdlc
- ppp

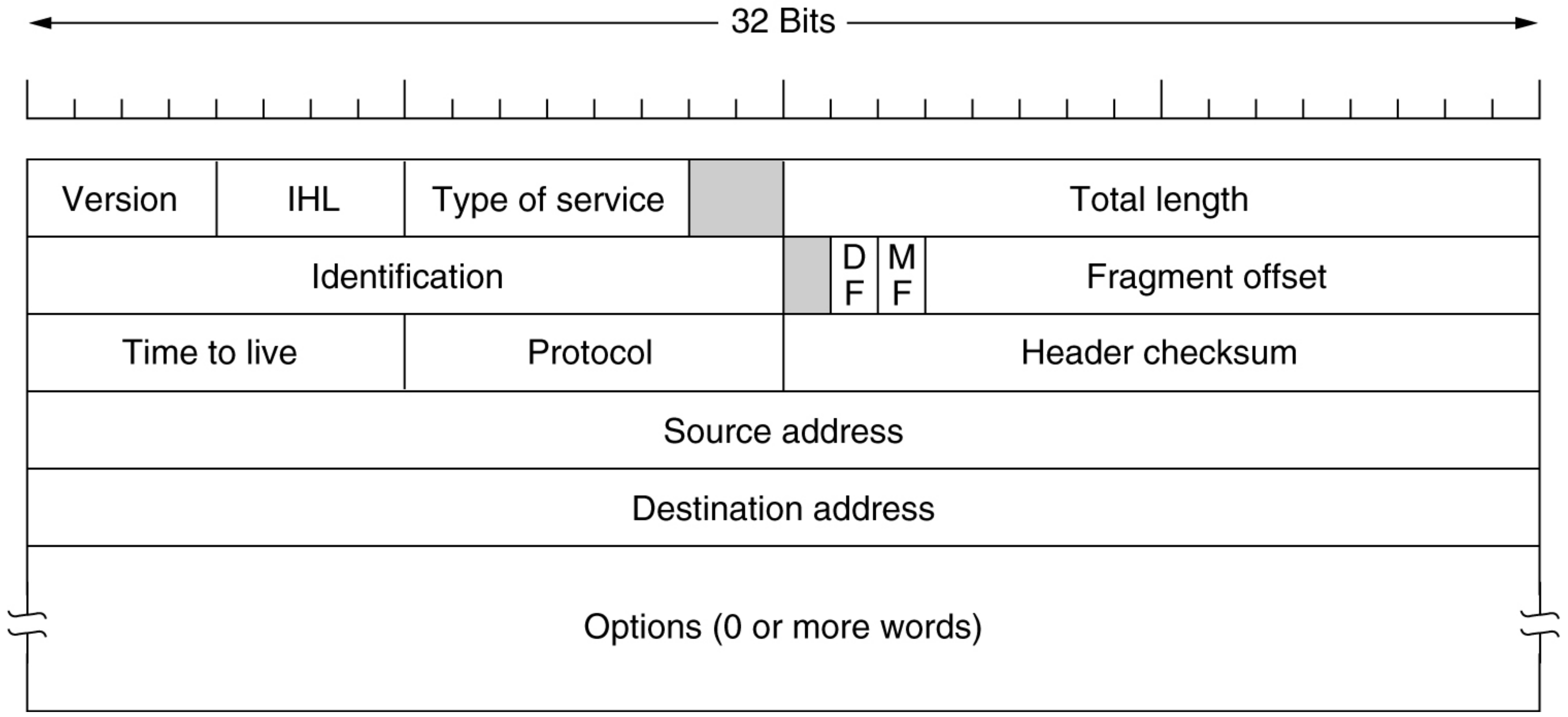
The IP Protocol

- Packets in IP layer are called **datagrams**.
- An IP datagram is a variable length with two parts: the header and data shown in below figure(a).
- The header is 20 to 60 bytes in length and contains the information essential for routing and delivery.
- The other part of the datagram is the data field which is of variable length.
- The header format is shown in below figure(b).

The IPv4 (Internet Protocol) header



Fig(a): IP Datagram Format



Fig(b): IP header format

The IP Protocol

- **Version** – This 4-bit field defines which version of the protocol the datagram belongs to. Current version of IP is IPV4 and the latest version of IP is IPv6.
- **IHL** – This 4-bit defines the length of the datagram header in 32-bit words. This field is needed because the length of the header is variable (20 & 60 Bytes).
- When there is no options, the header length is 20 bytes and the value of this field is 5 (5*4=20).
- When option field is at its maximum size, the value of this field is 15 (15*4=60).

The IP Protocol

- **Type of services** – This 8-bit field allows the host to tell the subnet what kind of service it wants.
- Various combinations of reliability and speed are possible.
  - For digitized voice, fast delivery beats accurate delivery.
  - For file transfer, error-free transmission is more important than fast transmission.
- This field contains a **3-bit** Precedence field, **3 flags**, D, T, and R & **2 unused bits**.
- The **3-bit** Precedence field defines the priority of the datagram in case of congestion. Priority range from 0 (normal) to 7 (network control packet).
- Next **3-Flag** bits allow the host to specify what it cared most about from the set {Delay, Throughput, Reliability}.

The IP Protocol

- **Total length** – This 16-bit field defines the total length of the IPv4 datagram (both header and data) in bytes. the maximum length of this field is 65,535 (2^16-1) bytes of which 20 to 60 bytes are the header and the remaining are data from the upper layer.

The IP Protocol

- **Identification** – This 16-bit field is needed to allow the destination host to determine which datagram a newly arrived fragment belongs to. All the fragments of a datagram contain the same Identification value.
- Next comes an **unused bit** and then **two 1-bit** fields.
- **DF (Don't Fragment)** – This 1-bit field is set to 1, means the routers must not fragment the datagram because the destination is incapable of putting the pieces back together again.

The IP Protocol

- **MF (More Fragments)** – This 1-bit field is set to 1, means the datagram is not the last fragment, there are more fragments after this one.
- **Fragment Offset** – This 13-bit field tells where in the current datagram this fragment belongs to.
- **Time to live** – It is 8-bit field. A datagram has a limited lifetime in its travel through an internet. It is supposed to count time in seconds, allowing a maximum lifetime of 255 sec. It must be decremented on each hop. When it hits zero, the datagram is discarded and a warning packet is sent back to the source host.

The IP Protocol

- **Protocol** – It is 8-bit field, specify to which transport layer protocol (TCP/UDP) the datagram is to be given.
- **Header checksum** – This 16-bit field verifies the header only. Such a checksum is useful for detecting errors generated by bad memory words inside a router.
  - **Note** - The Header checksum must be recomputed at each hop because at least one field always changes (Time to live field).

The IP Protocol

- **Source address** – This 32-bit field defines the IPv4 address of the source. This field remain unchanged during the time the IPv4 datagram travels from the source host to destination host.
- **Destination address** – This 32-bit field define the IPv4 address of the destination.

The IP Protocol

- options are not required for every datagram. They are used for **network testing and debugging**.
- IP provides several optional features, allowing a packets sender to set requirements on the path it takes through the network (source routing), trace the route a packet takes (record route), and label packets with security features.

The IP Protocol

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Fig: Some of the IP options

The IP Protocol

- The **Security** option tells how secret the information is. In theory, a military router might use this field to specify not to route through certain countries the military considers to be "bad guys." In practice, all routers ignore it, so its only practical function is to help spies find the good stuff more easily.

The IP Protocol

- The **Strict source routing** option gives the complete path from source to destination as a sequence of IP addresses. The datagram is required to follow that exact route.
- The **Loose source routing** option requires the packet to traverse the list of routers specified, and in the order specified, but it is allowed to pass through other routers on the way. Normally, this option would only provide a few routers, to force a particular path.

The IP Protocol

- The **Record route** option tells the routers along the path to append their IP address to the option field.
- The **Timestamp** option is like the Record route option, except that in addition to recording its 32-bit IP address, each router also records a 32-bit timestamp. This option is mostly for debugging routing algorithms.

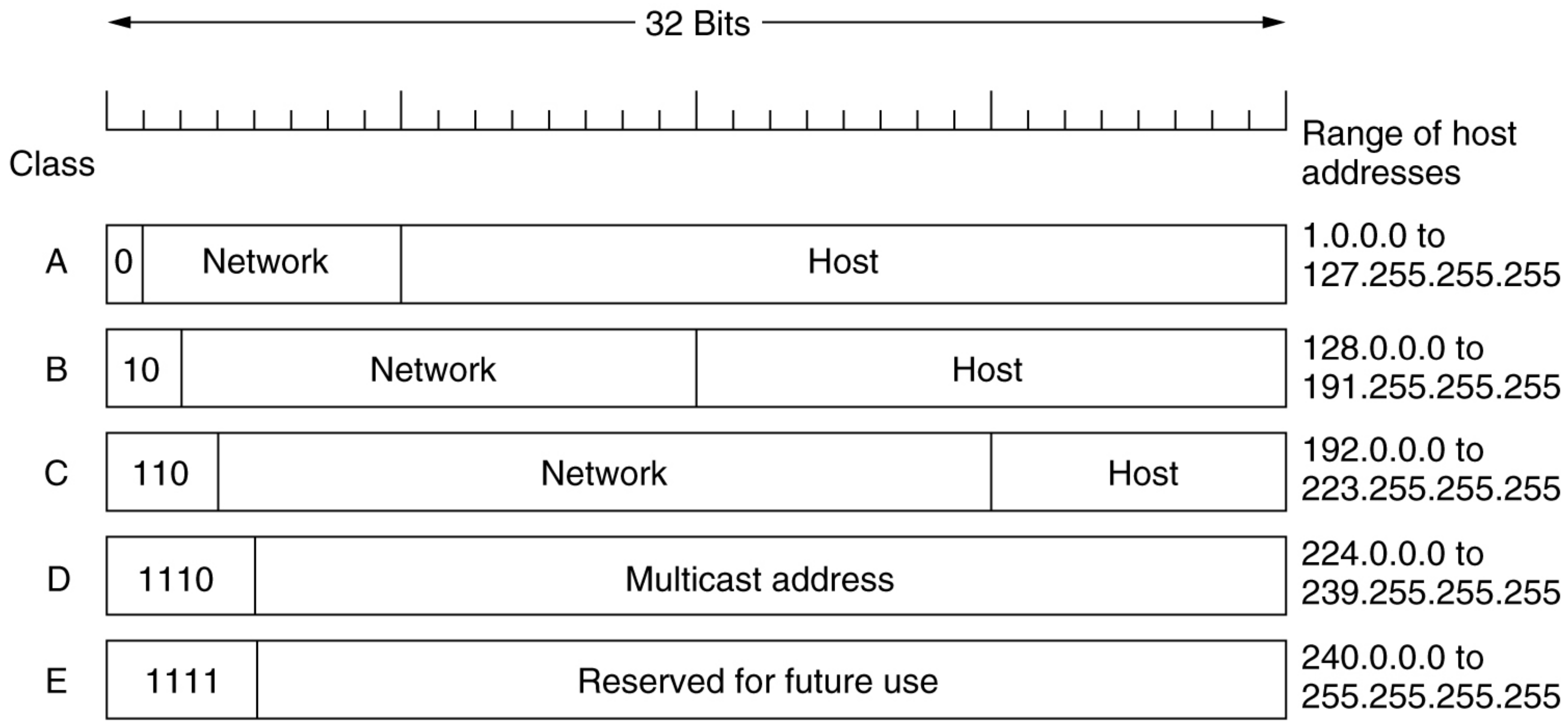
IP Addresses

- Every host and router on the Internet has an IP address, which encodes its network number and host number.
- The combination is unique: in principle, no two machines on the Internet have the same IP address.
- All IP addresses are 32 bits long and are used in the Source address and Destination address fields of IP packets.
- It is important to note that an IP address does not actually refer to a host.
- It really refers to a network interface, so if a host is on two networks, it must have two IP addresses.
- However, in practice, most hosts are on one network and thus have one IP address.

IP Addresses

- For several decades, IP addresses were divided into the five categories listed in below figure.
- This allocation has come to be called **classful addressing.**
- It is no longer used, but references to it in the literature are still common.

IP Addresses



32 Bits
Class
Range of host addresses
A
0
Network
Host
1.0.0.0 to 127.255.255.255
B
10
Network
Host
128.0.0.0 to 191.255.255.255
C
110
Network
Host
192.0.0.0 to 223.255.255.255
D
1110
Multicast address
224.0.0.0 to 239.255.255.255
E
1111
Reserved for future use
240.0.0.0 to 255.255.255.255

IP address formats

IP Addresses

- The class A, B, C, and D formats allow for up to 128 networks with 16 million hosts each, 16,384 networks with up to 64K hosts, and 2 million networks (e.g., LANs) with up to 256 hosts each (although a few of these are special).
- Also supported is multicast, in which a datagram is directed to multiple hosts.
- Addresses beginning with 1111 are reserved for future use.
- Over 500,000 networks are now connected to the Internet, and the number grows every year.
- Network numbers are managed by a nonprofit corporation called **ICANN (Internet Corporation for Assigned Names and Numbers)** to avoid conflicts.
- In turn, ICANN has delegated parts of the address space to various regional authorities, which then dole out IP addresses to ISPs and other companies.

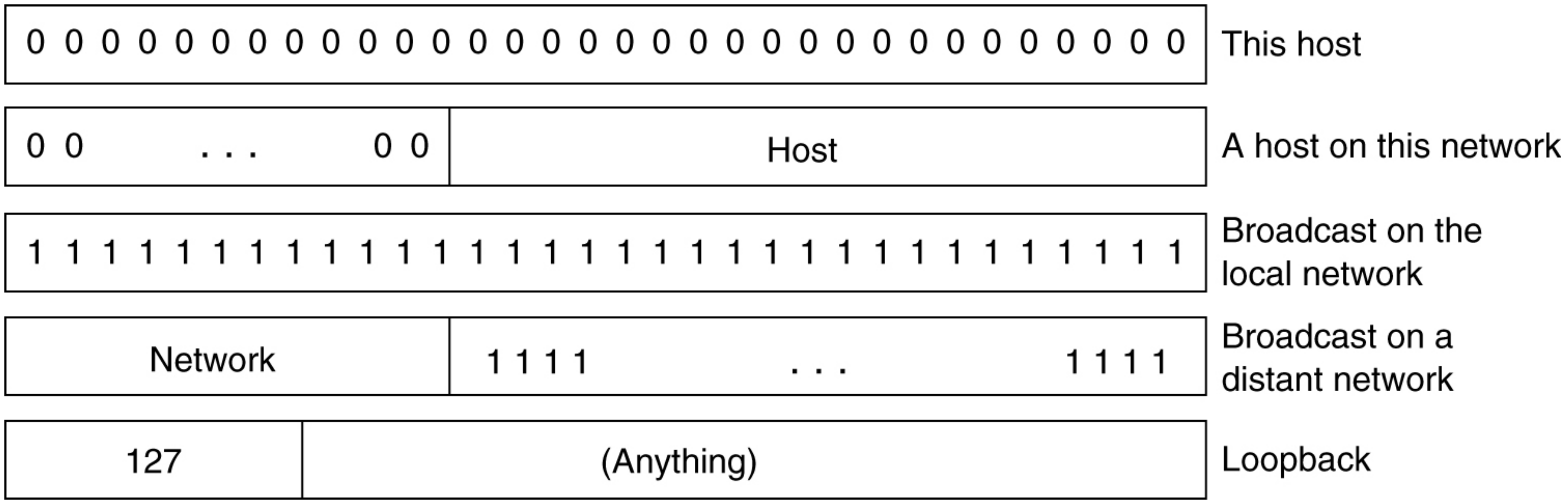
Network Addresses

- Network addresses, which are 32-bit numbers, are usually written in dotted decimal notation.
- In this format, each of the 4 bytes is written in decimal, from 0 to 255.
- For example, the 32-bit hexadecimal address C0290614 is written as 192.41.6.20.
- The lowest IP address is 0.0.0.0 and the highest is 255.255.255.255.

Special IP Addresses

- The values 0 and -1 (all 1s) have special meanings, as shown in below figure.
- The value 0 means this network or this host.
- The value of -1 is used as a broadcast address to mean all hosts on the indicated network.
- The IP address 0.0.0.0 is used by hosts when they are being booted.
- IP addresses with 0 as network number refer to the current network. These addresses allow machines to refer to their own network without knowing its number (but they have to know its class to know how many 0s to include).

IP Addresses



Special IP addresses

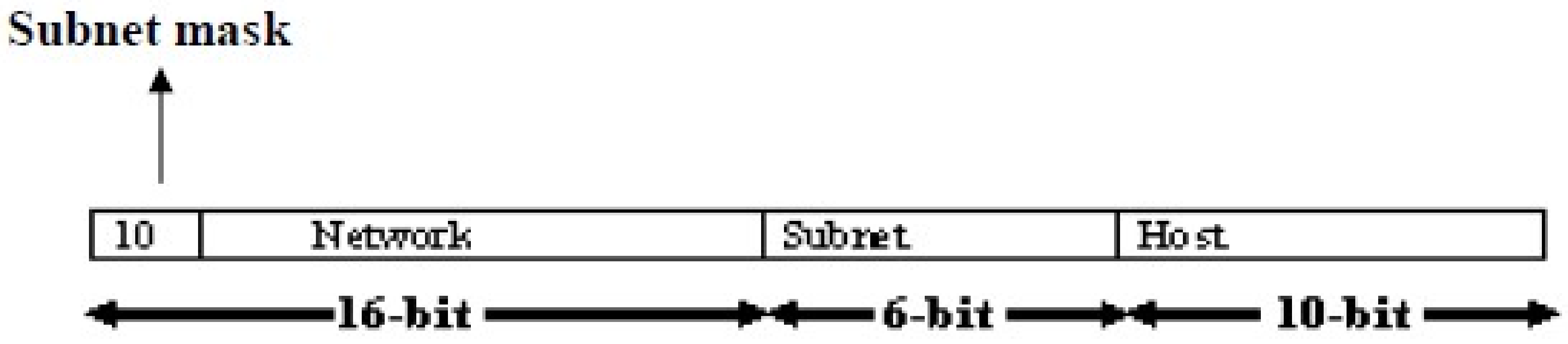
Special IP Addresses

- The address consisting of all 1s allows broadcasting on the local network, typically a LAN.
- The addresses with a proper network number and all 1s in the host field allow machines to send broadcast packets to distant LANs anywhere in the Internet (although many network administrators disable this feature).
- Finally, all addresses of the form 127.xx.yy.zz are reserved for loopback testing.
- Packets sent to that address are not put out onto the wire; they are processed locally and treated as incoming packets. This allows packets to be sent to the local network without the sender knowing its number.

Subnets

- A network is allowed to split into several parts for internal use but still act like a single network to the outside world. In the internal literature, these parts are called **SUBNETS.**
- **Eg :** A company started up with a class B address and had grown as time passed by which require a second LAN .
- Then, 16-bit host number is splitted up into a 6-bit subnet number and a 10-bit host number.
- This split allows 62 LANs (0 and –1 are reserved), each with up to 1022 hosts.

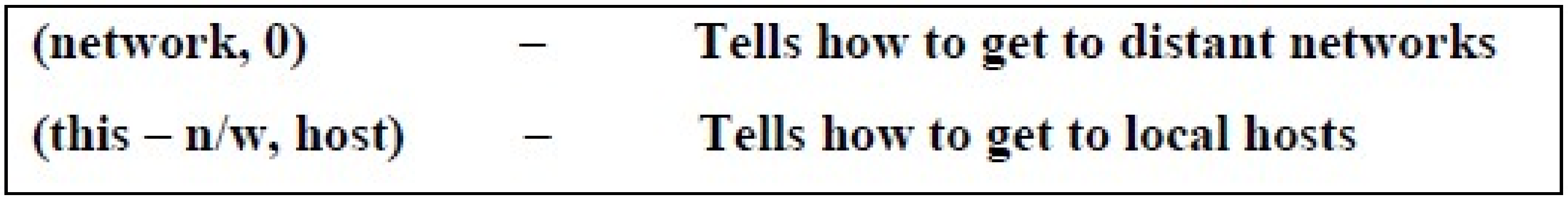
Subnets



- In this example, the subnet might use IP address starting at 130.50.4.1, the second subnet might start at 130.50.8.1, and so on.

Subnet Working

- Each router has a table listing some number of (network, 0) IP addresses and some number of (this- n/w, host) IP addresses.



(network, 0)	–	Tells how to get to distant networks
(this – n/w, host)	–	Tells how to get to local hosts

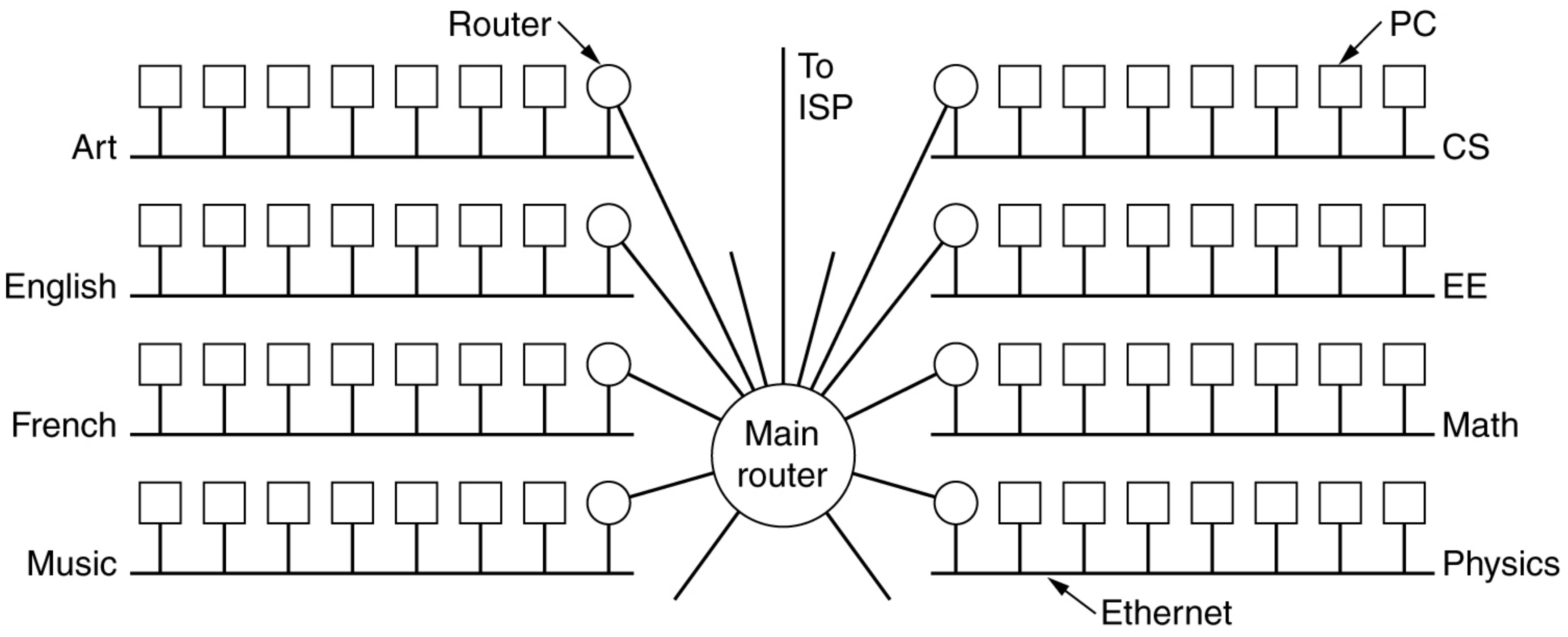
Subnet Working

- Associated with each table is the network interface to use to reach the destination, and certain other information.
- When an IP Packet arrives, its destination addresses is looked up in the routing table.
- If the packet is for a distant network, it is forwarded to the next router on interface given in the table.
- If it is a local host, it is sent directly to the destination.
- If the network is not present, the packet is forwarded to default router.

Subnet Working

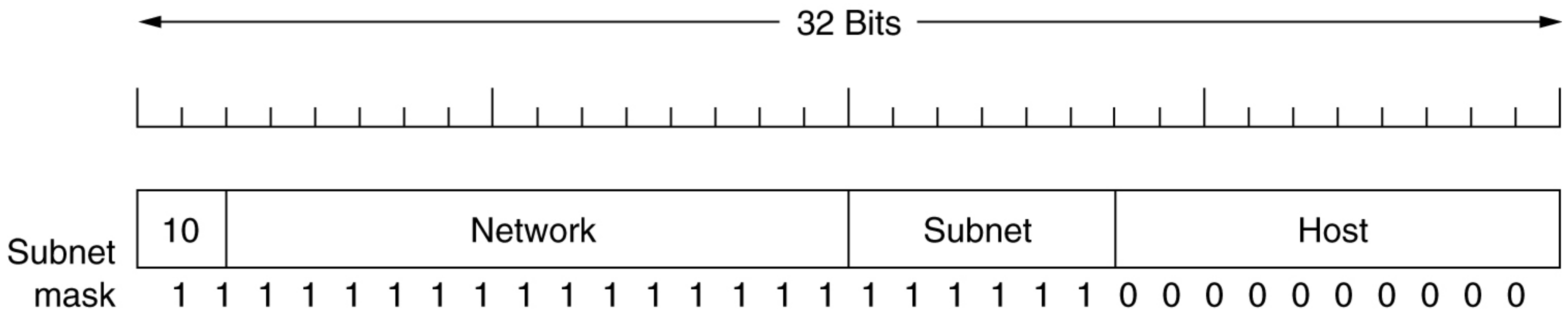
- When subnet is introduced, the routing tables are changed, adding entries of the form (this – n/w, subnet, 0) and (this – n/w, this subnet, host).
- Thus, a router on a subnet ‘k’ knows how to get all other subnets and also how to get to all the hosts on subnet ‘k’.
- Each router performs a Boolean AND with network’s subnet mask to get rid of host number and looks up the resulting address in its tables.

Subnets



A campus network consisting of LANs for various departments

Subnets



A class B network subnetted into 64 subnets

CIDR – Classless InterDomain Routing

- Dividing the IP address space into A, B and C classes turned out to be inflexible. IP is rapidly becoming a victim of its own popularity, it is running out of address.
- In 1993, the classful address space restriction was lifted. An arbitrary prefix length to indicate the network number, known as **Classless InterDomain Routing (CIDR)**, was adopted in place of the classful scheme.

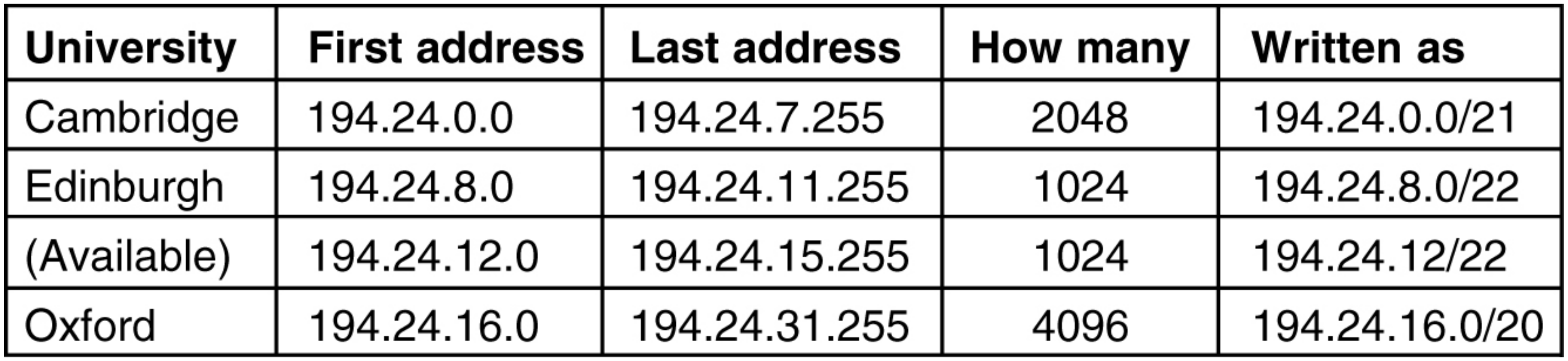
CIDR – Classless InterDomain Routing

- Using a CIDR notation, a prefix 205.100.0.0 of length 22 is written as 205.100.0.0/22.
- The corresponding prefix range runs from 205.100.0.0 through 205.100.3.0.
- The /22 notation indicates that the network mask is 22 bits or 255.255.252.0. CIDR route packets according to the higher order bits of the IP address.
- The entries in a CIDR routing table contain a 32-bit IP address and a 32-bit mask. CIDR uses a technique called **supernetting** so that a single routing entry covers a block of classful addresses.

CIDR – Classless InterDomain Routing

- For example, address of class C i.e. 205.100.0.0, 205.100.1.0, 205.100.2.0, 205.100.3.0, CIDR allows a single entry 205.100.16.0/22. The use of variable length prefixed requires that the routing tables be searched to find the longest prefix match.
- For example, a routing table may contain entries for the above supernet 205.100.0.0/22 as well as for 205.100.0.0/20. This situation may arise when a large number of destinations have been aggregated into the block 205.100.0.0/20, but packets destined to 205.100.16.0/22 are to be routed differently.
- A packet with destination address 205.100.1.1 will match both of these entries, so the algorithm must select the match with the longest prefix.

CIDR – Classless InterDomain Routing



University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

A set of IP address assignments

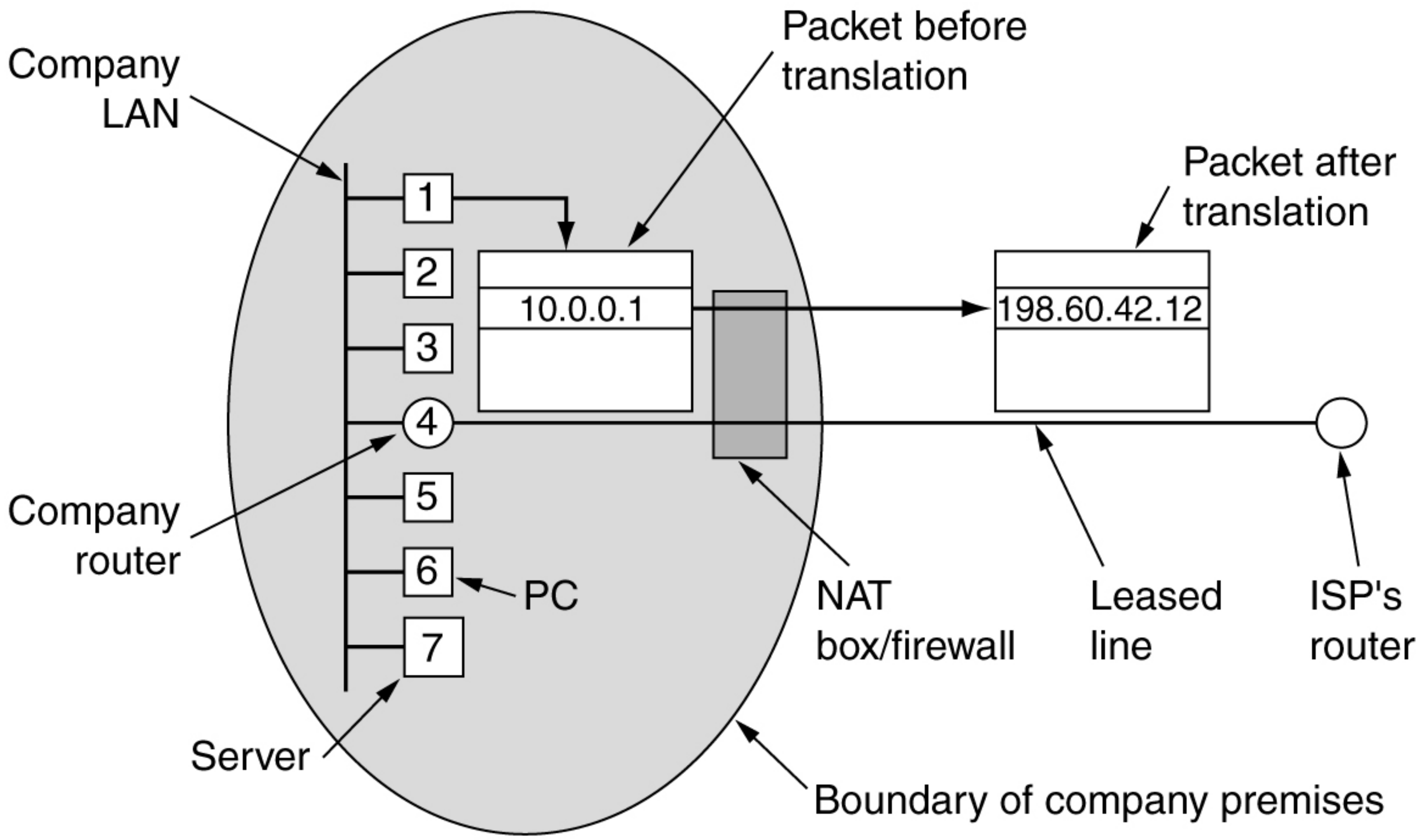
NAT – Network Address Translation

- With in the company, every machine has a unique address of the form 10.X.Y.Z.
- When a packet leaves the company premises, it passes through the NAT box that convert the internal IP source address 10.0.0.1.
- NAT box is often combined in a single device with a firewall. It is also possible to integrate the NAT box into the company router.

NAT – Network Address Translation

- Whenever an outgoing packet enters the NAT box, the 10.X.Y.Z Source address is replaced by the company true IP address.
- In addition, TCP source port field is replaced by an index into the NAT box 65536 entry translation table.
- This table entry contains the original IP address and original source port. Finally both the IP and TCP header checksums are recomputed and inserted into the packet.
- Below figure shows the placement of NAT box.

NAT – Network Address Translation



Placement and operation of a NAT box

NAT – Network Address Translation

- When process want to establish a TCP connection with a remote process, it attached itself to an unused TCP port on its own machine.
- This is called a source port and tells the TCP code where to send incoming packets belonging to this connection.
- The process also supplies a destination port to tell who to give the packet to on the remote side.

Internet Control Protocols

- In addition to IP, which is used for data transfer, the Internet has several control protocols used in the network layer, including **ICMP, ARP, RARP, BOOTP, and DHCP**.

The Internet Control Message Protocol

- The operation of the Internet is monitored closely by the routers.
- When something unexpected occurs, the event is reported by the ICMP (Internet Control Message Protocol), which is also used to test the Internet.
- About a dozen types of ICMP messages are defined.
- The most important ones are listed in below figure.
- Each ICMP message type is encapsulated in an IP packet.

Internet Control Message Protocol

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo request	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Figure: The principal ICMP message types

The Internet Control Message Protocol

- The **DESTINATION UNREACHABLE** message is used when the subnet or a router cannot locate the destination or when a packet with the DF bit cannot be delivered because a "small-packet" network stands in the way.
- The **TIME EXCEEDED** message is sent when a packet is dropped because its counter has reached zero. This event is a symptom that packets are looping, that there is enormous congestion, or that the timer values are being set too low.

The Internet Control Message Protocol

- The **PARAMETER PROBLEM** message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host's IP software or possibly in the software of a router transited.
- The **SOURCE QUENCH** message was formerly used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down. It is rarely used any more because when congestion occurs, these packets tend to add more fuel to the fire. Congestion control in the Internet is now done largely in the transport layer.

The Internet Control Message Protocol

- The **REDIRECT** message is used when a router notices that a packet seems to be routed wrong. It is used by the router to tell the sending host about the probable error.
- The **ECHO** and **ECHO REPLY** messages are used to see if a given destination is reachable and alive. Upon receiving the **ECHO** message, the destination is expected to send an **ECHO REPLY** message back.
- The **TIMESTAMP REQUEST** and **TIMESTAMP REPLY** messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility is used to measure network performance.

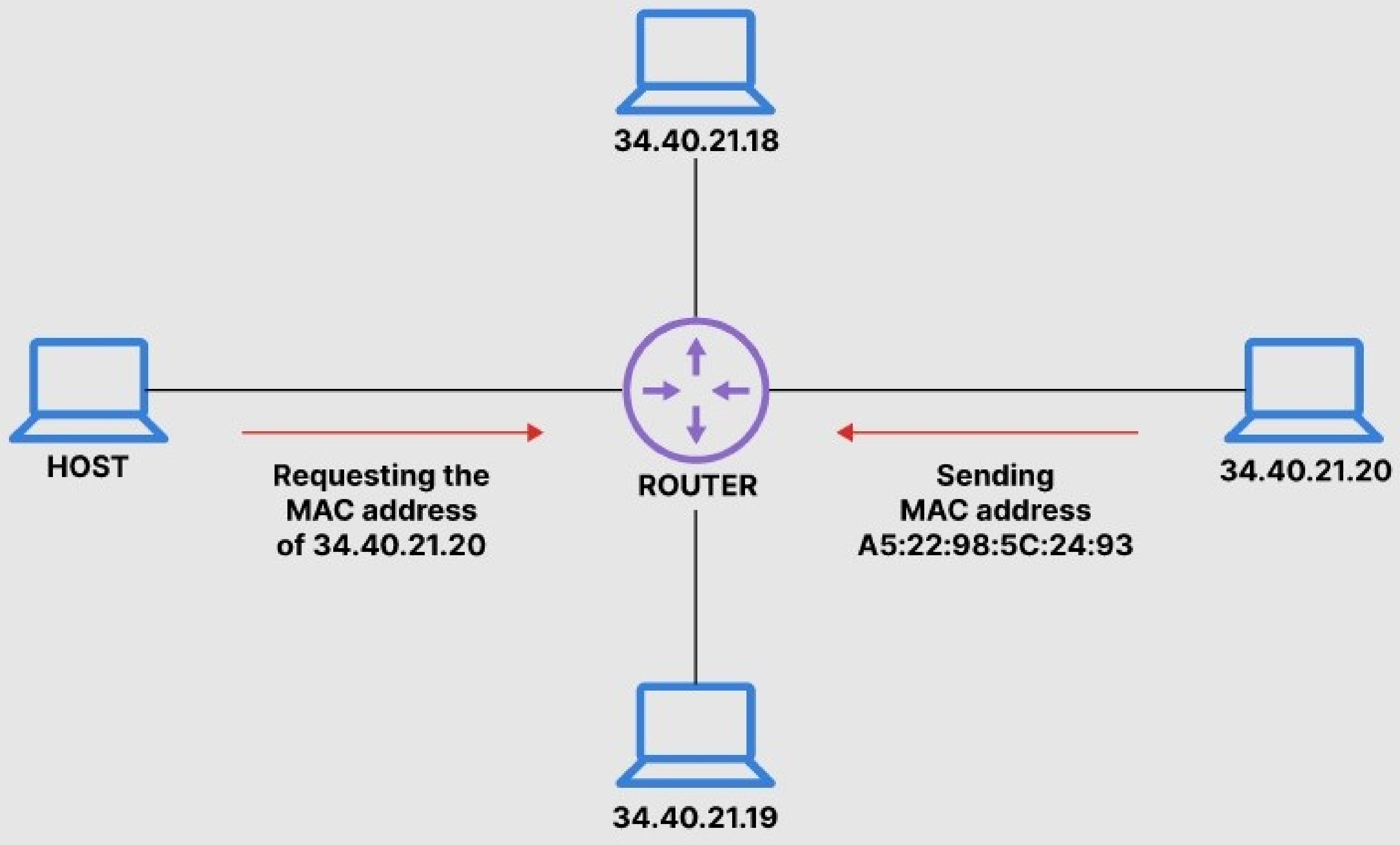
ARP

- Address Resolution Protocol (ARP) is a protocol or procedure that connects an ever-changing Internet Protocol (IP) address to a fixed physical machine address, also known as a media access control (MAC) address, in a local-area network (LAN).
- This mapping procedure is important because the lengths of the IP and MAC addresses differ, and a translation is needed so that the systems can recognize one another. The most used IP today is IP version 4 (IPv4). An IP address is 32 bits long. However, MAC addresses are 48 bits long. ARP translates the 32-bit address to 48 and vice versa.

ARP

- There is a networking model known as the Open Systems Interconnection (OSI) model. First developed in the late 1970s, the OSI model uses layers to give IT teams a visualization of what is going on with a particular networking system. This can be helpful in determining which layer affects which application, device, or software installed on the network, and further, which IT or engineering professional is responsible for managing that layer.
- The MAC address is also known as the data link layer, which establishes and terminates a connection between two physically connected devices so that data transfer can take place. The IP address is also referred to as the network layer or the layer responsible for forwarding packets of data through different routers. ARP works between these layers.

How Address Resolution Protocol (ARP) Works


34.40.21.18
HOST
Requesting the MAC address of 34.40.21.20
ROUTER
Sending MAC address A5:22:98:5C:24:93
34.40.21.20
34.40.21.19

What Does ARP Do and How Does It Work?

- When a new computer joins a local area network (LAN), it will receive a unique IP address to use for identification and communication.
- Packets of data arrive at a gateway, destined for a particular host machine. The gateway, or the piece of hardware on a network that allows data to flow from one network to another, asks the ARP program to find a MAC address that matches the IP address. The ARP cache keeps a list of each IP address and its matching MAC address. The ARP cache is dynamic, but users on a network can also configure a static ARP table containing IP addresses and MAC addresses.

What Does ARP Do and How Does It Work?

- ARP caches are kept on all operating systems in an IPv4 Ethernet network. Every time a device requests a MAC address to send data to another device connected to the LAN, the device verifies its ARP cache to see if the IP-to-MAC-address connection has already been completed. If it exists, then a new request is unnecessary. However, if the translation has not yet been carried out, then the request for network addresses is sent, and ARP is performed.
- An ARP cache size is limited by design, and addresses tend to stay in the cache for only a few minutes. It is purged regularly to free up space. This design is also intended for privacy and security to prevent IP addresses from being stolen or spoofed by cyber attackers. While MAC addresses are fixed, IP addresses are constantly updated.

RARP

- ARP solves the problem of finding out which Ethernet address corresponds to a given IP address.
- Sometimes the reverse problem has to be solved: **Given an Ethernet address, what is the corresponding IP address?** In particular, this problem occurs when a diskless workstation is booted.
- Such a machine will normally get the binary image of its operating system from a remote file server. But how does it learn its IP address?

RARP

- The first solution devised was to use **RARP** (Reverse Address Resolution Protocol).
- This protocol allows a newly-booted workstation to broadcast its Ethernet address and say: My 48-bit Ethernet address is 14.04.05.18.01.25. Does anyone out there know my IP address? The RARP server sees this request, looks up the Ethernet address in its configuration files, and sends back the corresponding IP address.
- Using RARP is better than embedding an IP address in the memory image because it allows the same image to be used on all machines. If the IP address were buried inside the image, each workstation would need its own image.

BOOTP

- A disadvantage of RARP is that it uses a destination address of all 1s (limited broadcasting) to reach the RARP server.
- However, such broadcasts are not forwarded by routers, so a RARP server is needed on each network.
- To get around this problem, an alternative **bootstrap protocol** called **BOOTP** was invented.
- Unlike RARP, BOOTP uses UDP messages, which are forwarded over routers.
- It also provides a diskless workstation with additional information, including the IP address of the file server holding the memory image, the IP address of the default router, and the subnet mask to use.

DHCP

- A serious problem with BOOTP is that it requires manual configuration of tables mapping IP address to Ethernet address.
- When a new host is added to a LAN, it cannot use BOOTP until an administrator has assigned it an IP address and entered its (Ethernet address, IP address) into the BOOTP configuration tables by hand.
- To eliminate this error-prone step, BOOTP was extended and given a new name: **DHCP (Dynamic Host Configuration Protocol)**.
- DHCP allows both manual IP address assignment and automatic assignment.
- In most systems, it has largely replaced RARP and BOOTP.

DHCP

- Like RARP and BOOTP, DHCP is based on the idea of a special server that assigns IP addresses to hosts asking for one.
- This server need not be on the same LAN as the requesting host.
- Since the DHCP server may not be reachable by broadcasting, a DHCP relay agent is needed on each LAN, as shown in below figure.

DHCP

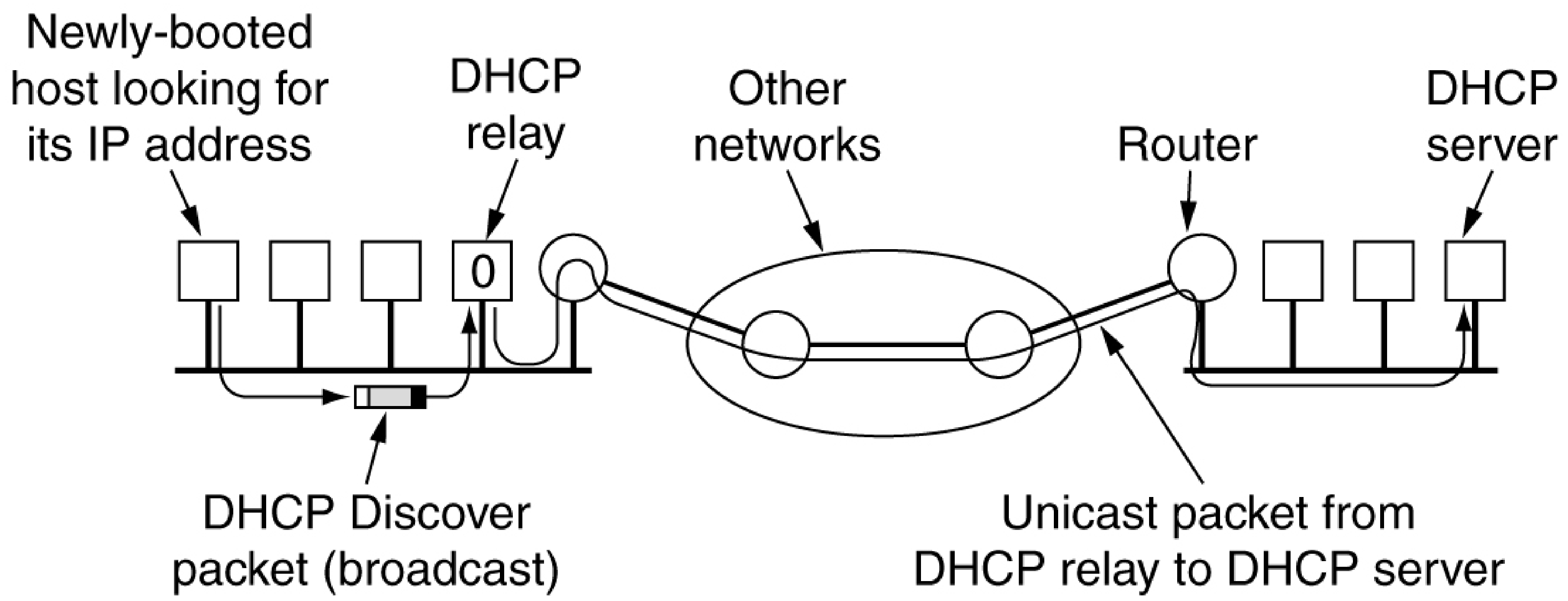

Newly-booted host looking for its IP address
DHCP relay
0
Other networks
Router
DHCP server
DHCP Discover packet (broadcast)
Unicast packet from DHCP relay to DHCP server

Fig: Operation of DHCP

DHCP

- To find its IP address, a newly-booted machine broadcasts a DHCP DISCOVER packet.
- The DHCP relay agent on its LAN intercepts all DHCP broadcasts.
- When it finds a DHCP DISCOVER packet, it sends the packet as a unicast packet to the DHCP server, possibly on a distant network.
- The only piece of information the relay agent needs is the IP address of the DHCP server.

DHCP

- An issue that arises with automatic assignment of IP addresses from a pool is how long an IP address should be allocated.
- If a host leaves the network and does not return its IP address to the DHCP server, that address will be permanently lost.
- After a period of time, many addresses may be lost.
- To prevent that from happening, IP address assignment may be for a fixed period of time, a technique called **leasing**.
- Just before the lease expires, the host must ask the DHCP for a renewal.
- If it fails to make a request or the request is denied, the host may no longer use the IP address it was given earlier.

UNIT 3 PART 3

The Transport Service: Services Provided to the Upper Layers, Transport Service Primitives.

Transport Layer

- The task of the transport layer is to provide reliable, cost-effective data transport from the source machine to the destination machine, independent of physical network or networks currently in use.

Transport Layer services

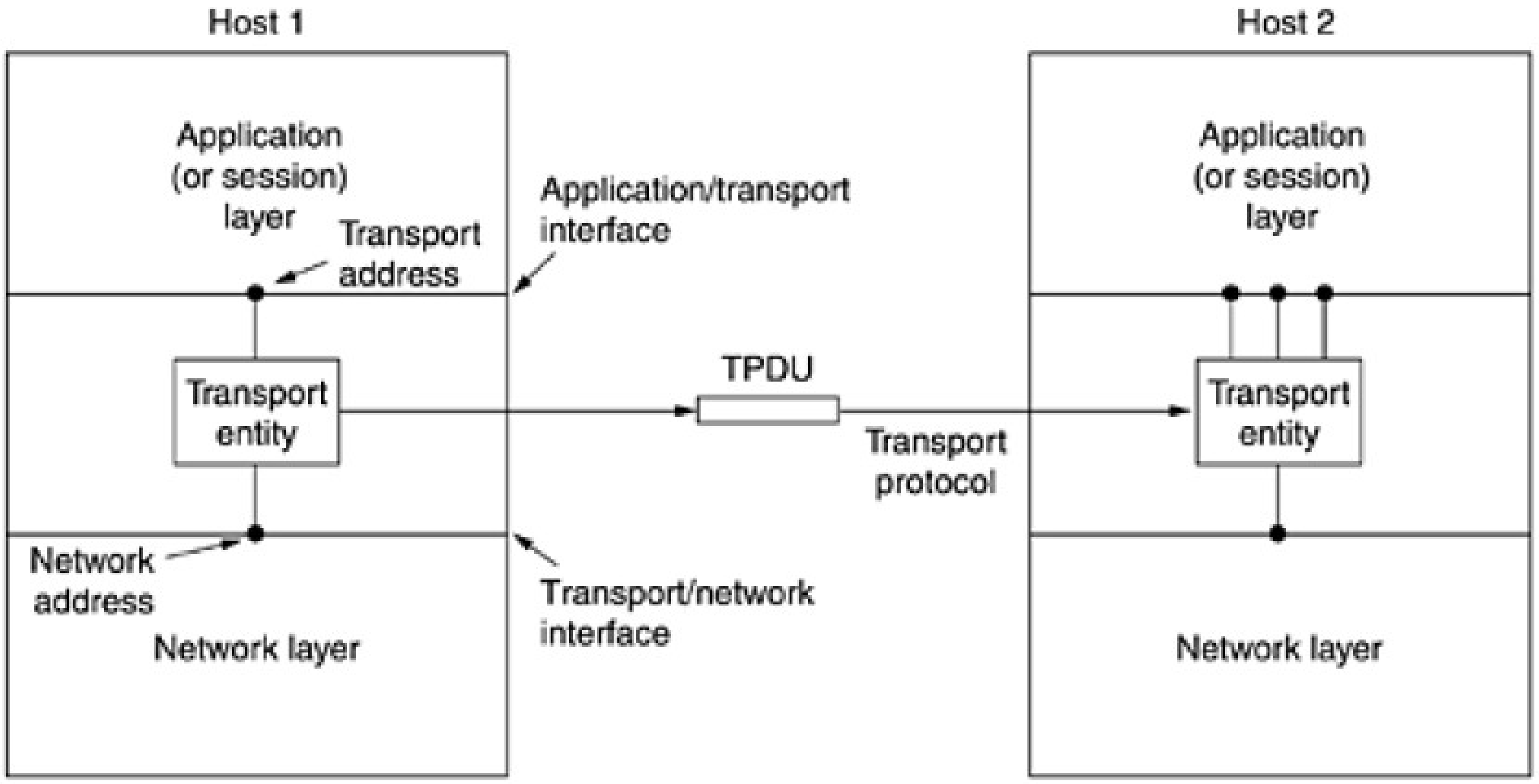
- We will deal with different types of services

1. The type of services provided by the transport layer to the upper layer (Application layer).
2. The type of service it is accepting from the lower layer
3. Quality of services provided by the transport layer to its users.
4. Different service primitives.

Services provided to the upper layers

- The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective service to its users, normally processes in the application layer.
- To achieve this goal, the transport layer makes use of the services provided by the network layer.
- The hardware and/or software within the transport layer that does the work is called the **transport entity.**
- The transport entity can be located in the operating system kernel, in a separate user process, in a library package bound into network applications, or conceivably on the network interface card.
- The (logical) relationship of the network, transport, and application layers is illustrated in below figure.

Services provided to the upper layers



Services provided to the upper layers

- Just as there are two types of network service, **connection-oriented** and **connectionless**, there are also two types of transport service.
- The connection-oriented transport service is similar to the connection-oriented network service in many ways.
- In both cases, connections have three phases: establishment, data transfer, and release. **Addressing and flow control** are also similar in both layers.
- Furthermore, the connectionless transport service is also very similar to the connectionless network service.

Services provided to the upper layers

- The obvious question is then this: If the transport layer service is so similar to the network layer service, why are there two distinct layers? Why is one layer not adequate?
- The transport code runs entirely on the users' machines, but the network layer mostly runs on the **routers**, which are operated by the carrier (at least for a wide area network).
- What happens if the network layer offers inadequate service? Suppose that it frequently loses packets? What happens if routers crash from time to time?

Services provided to the upper layers

- Problems occur, that's what. The users have no real control over the network layer, so they cannot solve the problem of poor service by using better routers or putting more error handling in the data link layer.
- The only possibility is to put on top of the network layer another layer that improves the quality of the service.
- If, in a connection-oriented subnet, a transport entity is informed halfway through a long transmission that its network connection has been abruptly terminated, with no indication of what has happened to the data currently in transit, it can set up a new network connection to the remote transport entity.
- Using this new network connection, it can send a query to its peer asking which data arrived and which did not, and then pick up from where it left off.

Transport Service Primitives

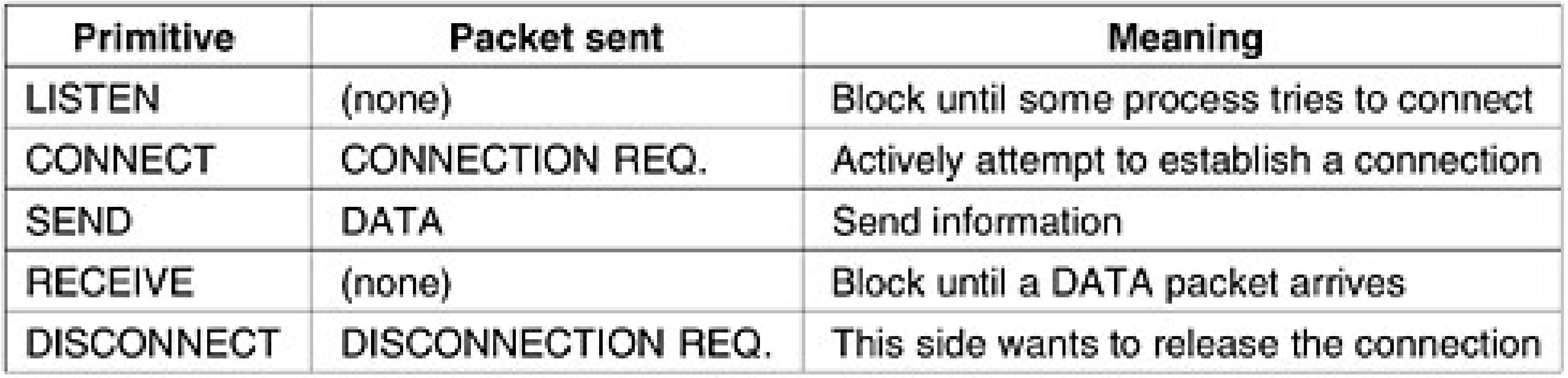
- To allow users to access the transport service, the transport layer must provide some operations to application programs i.e. a transport service interface.
- Each transport service has its own interface.

Transport Service Primitives

- The main difference is that the network service is intended to model the service offered by real networks, warts and all.
- Real networks can lose packets, so the network service is generally unreliable.
- The (connection-oriented) transport service, in contrast, is reliable.
- Of course, real networks are not error-free, but that is precisely the purpose of the transport layer—to provide a reliable service on to of an unreliable network.

Transport Service Primitives

- To get an idea of what a transport service might be like, consider the five primitives listed in below figure.
- This transport interface is truly bare bones, but it gives the essential flavor of what a connection-oriented transport interface has to do.
- It allows application programs to establish, use, and then release connections, which is sufficient for many applications.



Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

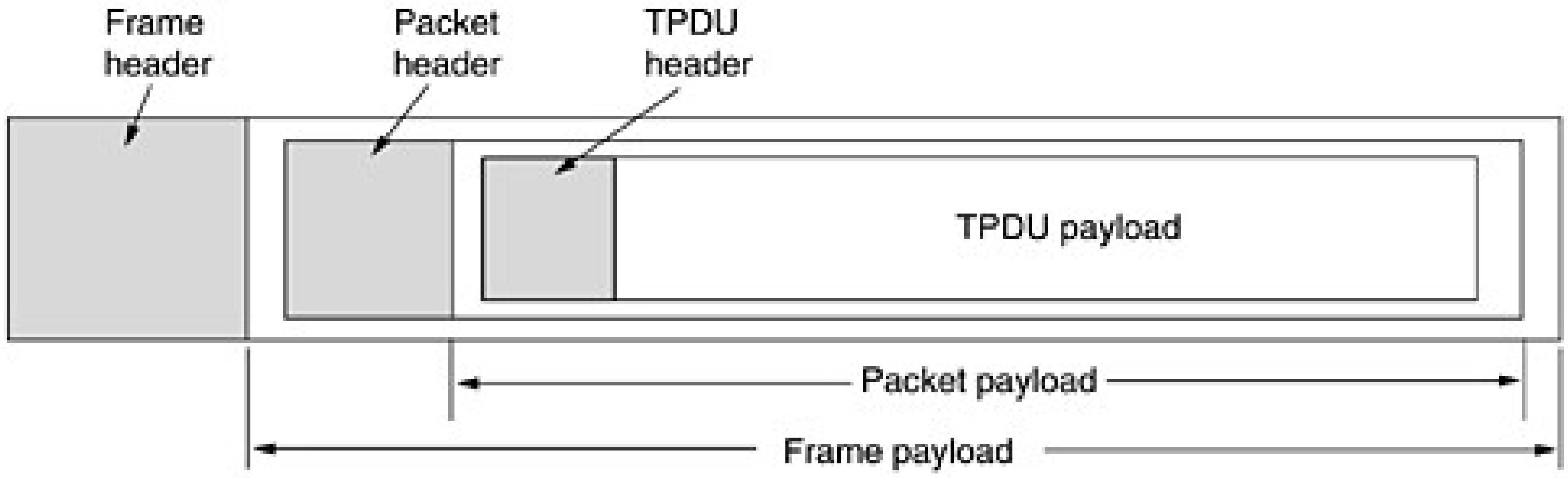
Figure: The primitives for a simple transport service

Transport Service Primitives

- For lack of a better term, we will reluctantly use the somewhat ungainly acronym **TPDU** (Transport Protocol Data Unit) for messages sent from transport entity to transport entity.
- Thus, TPDUs (exchanged by the transport layer) are contained packets (exchanged by the network layer).
- In turn, packets are contained in frames (exchanged by data link layer).
- When a frame arrives, the data link layer processes the frame header and passes contents of the frame payload field up to the network entity.

Transport Service Primitives

- The network entity processes the packet header and passes the contents of the packet payload up to the transport entity.
- This nesting is illustrated in below figure.



Transport Service Primitives

- To see how these primitives might be used, consider an application with a server and a number of remote clients.

1. The server executes a "**LISTEN**" primitive by calling a library procedure that makes a system call to block the server until a client turns up.
2. When a client wants to talk to the server, it executes a "**CONNECT**" primitive, with "**CONNECTION REQUEST**" TPDU sent to the server.
3. When it arrives, the TE unblocks the server and sends a "**CONNECTION ACCEPTED**" TPDU back to the client.

Transport Service Primitives

4. When it arrives, the client is unblocked and the connection is established. Data can now be exchanged using “**SEND**” and “**RECEIVE**” primitives.
5. When a connection is no longer needed, it must be released to free up table space with in the 2 transport entries, which is done with “**DISCONNECT**” primitive by sending “**DISCONNECTION REQUEST**” TPDU.
   – This **disconnection** can be done either by **symmetric variant** or by **asymmetric variant**. **Symmetric disconnection** is closed separately independent of other one. In this connection is released when both sides have done a **DISCONNECT**. In **Asymmetric disconnection** , this connection is released when any of the sides have done a **DISCONNECT**.

Transport Service Primitives

- A state diagram for connection establishment and release for these simple primitives is given in below figure.
- Each transition is triggered by some event, either a primitive executed by the local transport user or an incoming packet.
- For simplicity, we assume here that each TPDU is separately acknowledged. We also assume that a symmetric disconnection model is used, with the client going first.

Transport Service Primitives

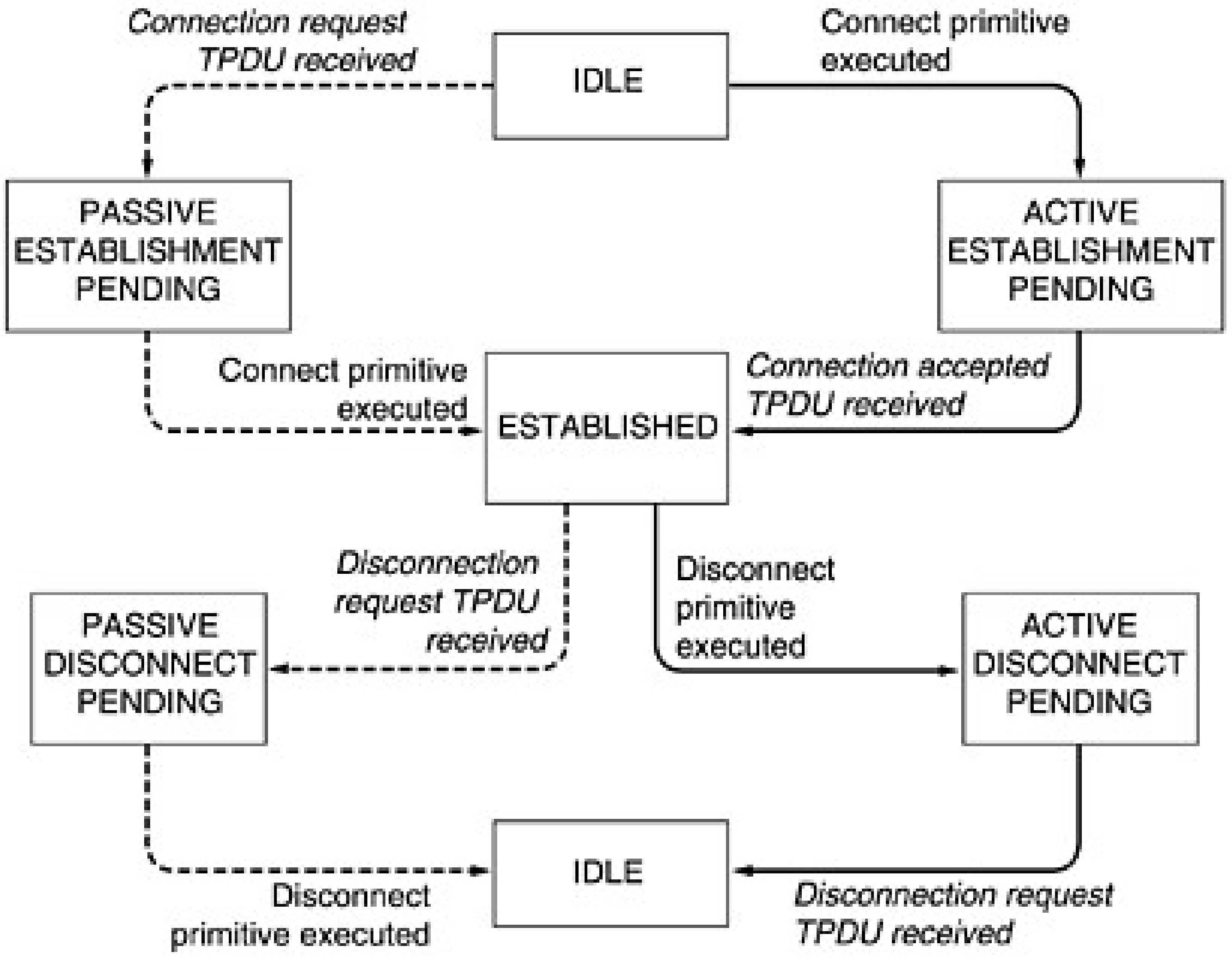

Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

Figure: A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

UNIT 3 PART 4

Elements of Transport Protocols: Addressing, Connection Establishment, Connection Release, Flow Control and Buffering, Multiplexing, Crash Recovery.

Elements of Transport Protocols

- The transport service is implemented by a transport protocol used between the two transport entities. Below figure shows the environment of Data Link layer and Transport layer.

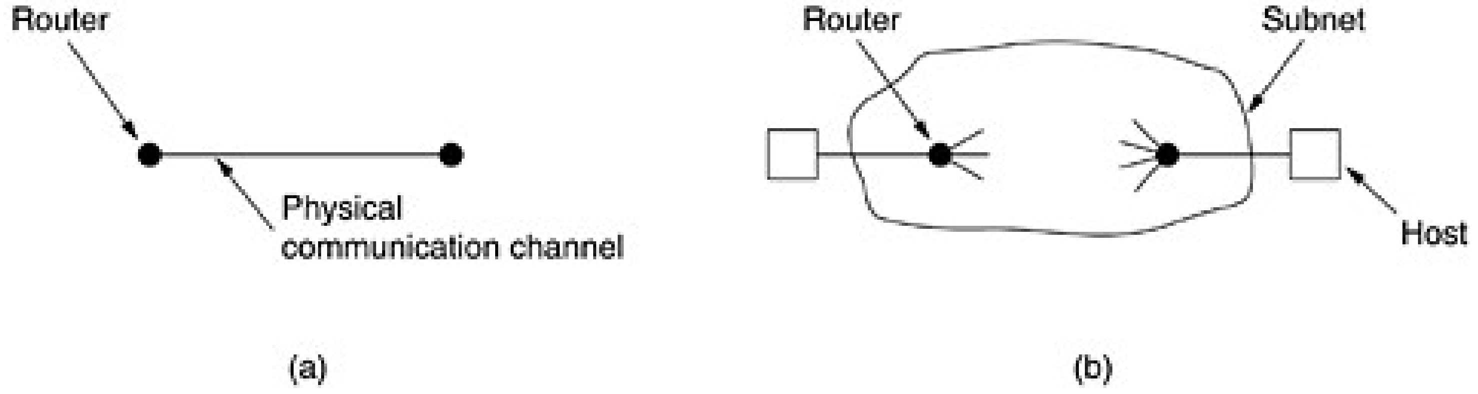


Figure: (a) Environment of the data link layer. (b) Environment of the transport layer.

Elements of Transport Protocols

- At the data link layer, two routers communicate directly via a physical channel, whereas at the transport layer, this physical channel is replaced by the entire subnet.
- For router in the data link layer, it is not necessary for a router to specify which router it wants to talk to—each outgoing line uniquely specifies a particular router.
- In the transport layer, explicit addressing of destinations is required.
- The process of establishing a connection over the wire is simple at Data Link Layer.
- In the transport layer, initial **connection establishment** is more complicated.
- **Buffering** and **flow control** are needed in both layers, but the presence of a large and dynamically varying number of connections in the transport layer may require a different approach than we used in the data link layer.

Addressing

- When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to.
- The method normally used is to define transport addresses to which processes can listen for connection requests.
- In the Internet, these end points are called **ports**. We will use the generic term **TSAP, (Transport Service Access Point)**.
- The analogous end points in the network layer (i.e., network layer addresses) are then called **NSAPs**.
- IP addresses are examples of NSAPs.

Addressing

- Below figure illustrates the relationship between the NSAP, TSAP and transport connection.
- Application processes, both clients and servers, can attach themselves to a TSAP to establish a connection to a remote TSAP.
- These connections run through NSAPs on each host, as shown.
- The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport end points that share that NSAP.

Addressing

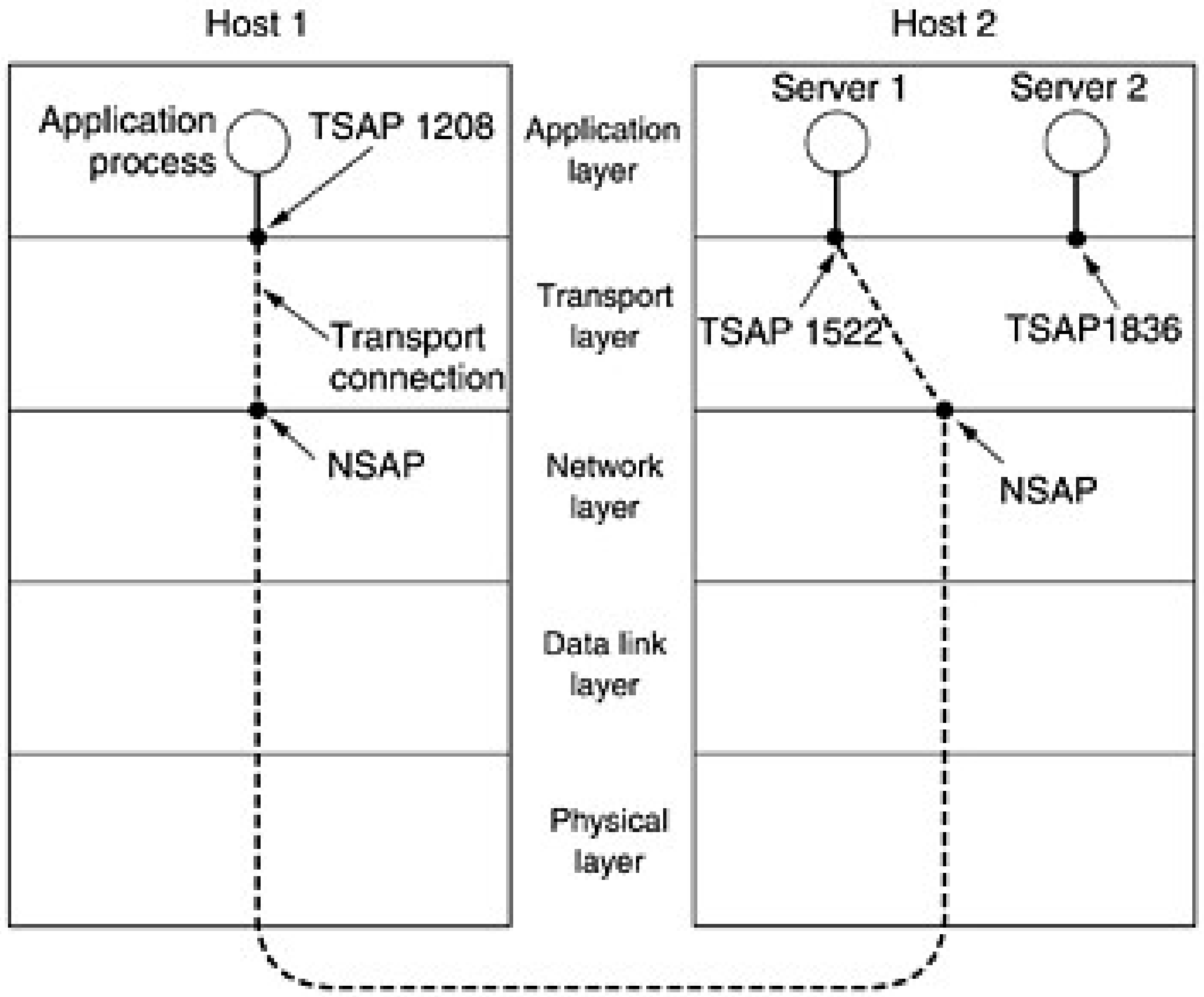


Figure: TSAPs, NSAPs, and transport connections

Addressing

- A possible scenario for a transport connection is as follows.

1. A time of day server process on host 2 attaches itself to TSAP 1522 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to find out the time-of-day, so it issues a CONNECT request specifying TSAP 1208 as the source and TSAP 1522 as the destination. This action ultimately results in a transport connection being established between the application process on host 1 and server 1 on host 2.
3. The application process then sends over a request for the time.
4. The time server process responds with the current time.
5. The transport connection is then released.

Connection Establishment

- Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST TPDU to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, store, and duplicate packets.
- Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted subnet design.
2. Putting a hop counter in each packet.
3. Timestamping each packet.

Connection Establishment

- To solve this problem, Tomlinson (1975) introduced the three-way handshake.
- This establishment protocol does not require both sides to begin sending with the same sequence number, so it can be used with synchronization methods other than the global clock method.
- The normal setup procedure when host 1 initiates is shown in below Figure (a).
- Host 1 chooses a sequence number, x, and sends a CONNECTION REQUEST TPDU containing it to Host 2.
- Host 2 replies with an ACK TPDU acknowledging x and announcing its own initial sequence number, y.
- Finally, Host 1 acknowledges host 2's choice of an initial sequence number in the first data TPDU that it sends.

Connection Establishment

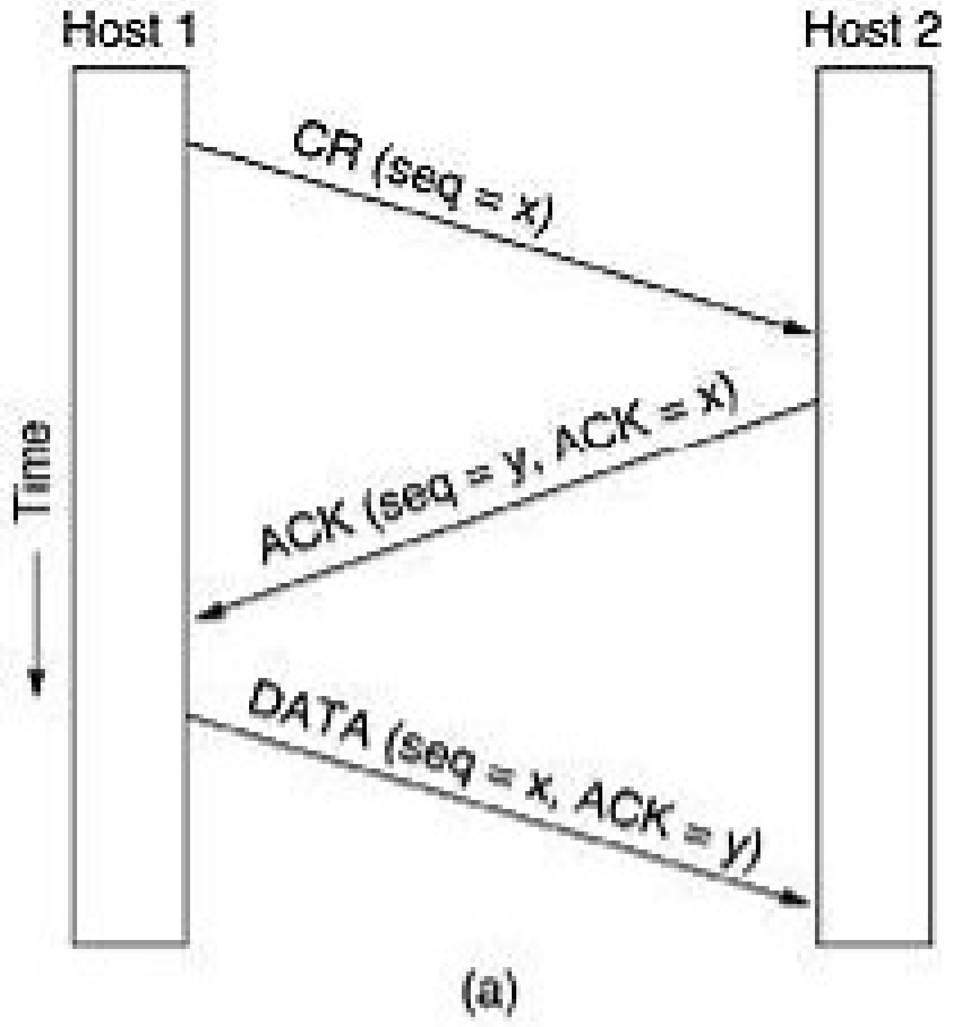


Figure: Normal operation

Connection Establishment

- Now let us see how the three-way handshake works in the presence of delayed duplicate control TPDUs.
- In figure(b), the first TPDU is a delayed duplicate CONNECTION REQUEST from an old connection. This TPDU arrives at Host 2 without Host 1's knowledge.
- Host 2 reacts to this TPDU by sending Host 1 an ACK TPDU, in effect asking for verification that Host 1 was indeed trying to set up a new connection.
- When Host 1 rejects Host 2's attempt to establish a connection, Host 2 realizes that it was tricked by a delayed duplicate and abandons the connection.
- In this way, a delayed duplicate does no damage.

Connection Establishment

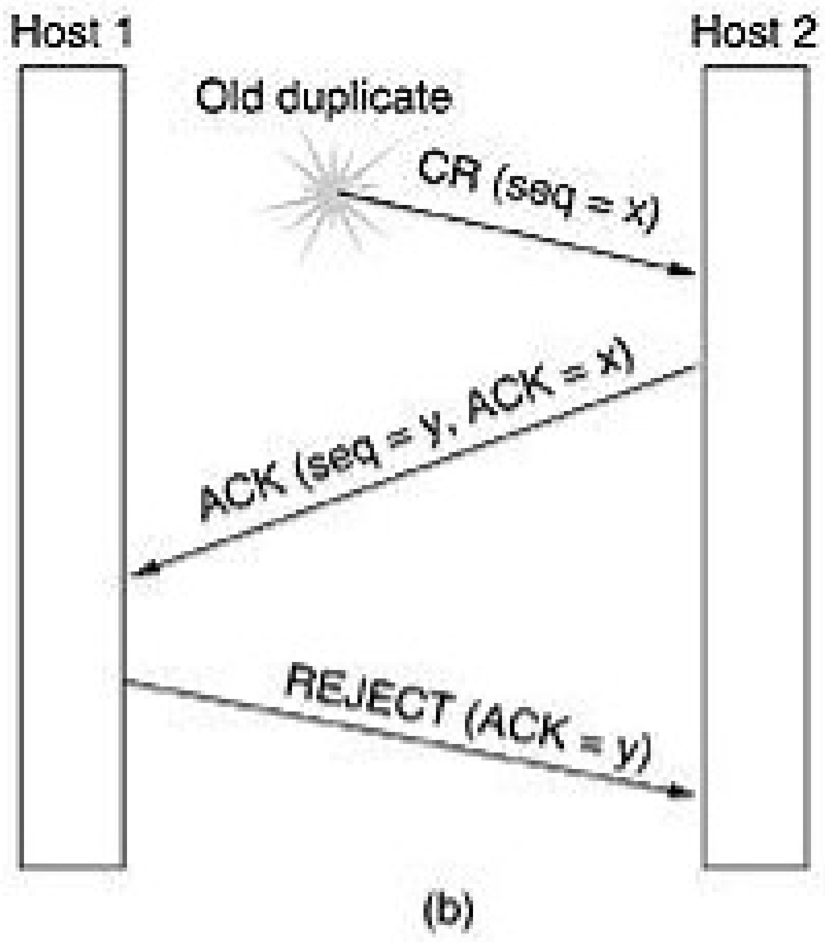
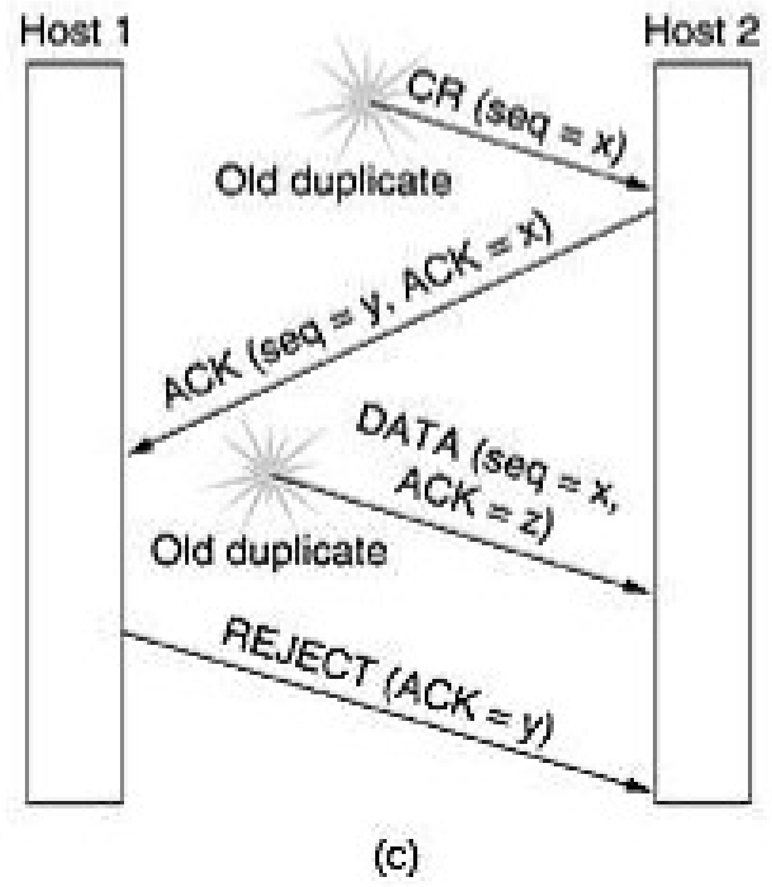


Figure: Old duplicate CONNECTION REQUEST appearing out of nowhere

Connection Establishment

- The worst case is when both a delayed CONNECTION REQUEST and an ACK are floating around in the subnet.
- This case is shown in below figure(c).
- As in the previous example, Host 2 gets a delayed CONNECTION REQUEST and replies to it.
- At this point it is crucial to realize that Host 2 has proposed using y as the initial sequence number for Host 2 to Host 1 traffic, knowing that no TPDUs containing sequence number y or acknowledgements to y are still in existence.
- When the second delayed TPDU arrives at Host 2, the fact that z has been acknowledged rather than y tells Host 2 that this, too, is an old duplicate.

Connection Establishment



Figure(c): Duplicate CONNECTION REQUEST and duplicate ACK

Connection Release

- Releasing a connection is easier than establishing one.
- There are **two** styles of terminating a connection: **asymmetric release** and **symmetric release**.
- **Asymmetric release** is the way the telephone system works: when one party hangs up, the connection is broken.
- **Symmetric release** treats the connection as two separate unidirectional connections and requires each one to be released separately.

Connection Release

- Asymmetric release is abrupt and may result in data loss.
- Consider the scenario of below figure.
- After the connection is established, host 1 sends a TPDU that arrives properly at host 2. Then host 1 sends another TPDU.
- Unfortunately, host 2 issues a DISCONNECT before the second TPDU arrives. The result is that the connection is released and data are lost.
- Clearly, a more sophisticated release protocol is needed to avoid data loss. One way is to use symmetric release, in which each direction is released independently of the other one. Here, a host can continue to receive data even after it has sent a DISCONNECT TPDU.

Connection Release

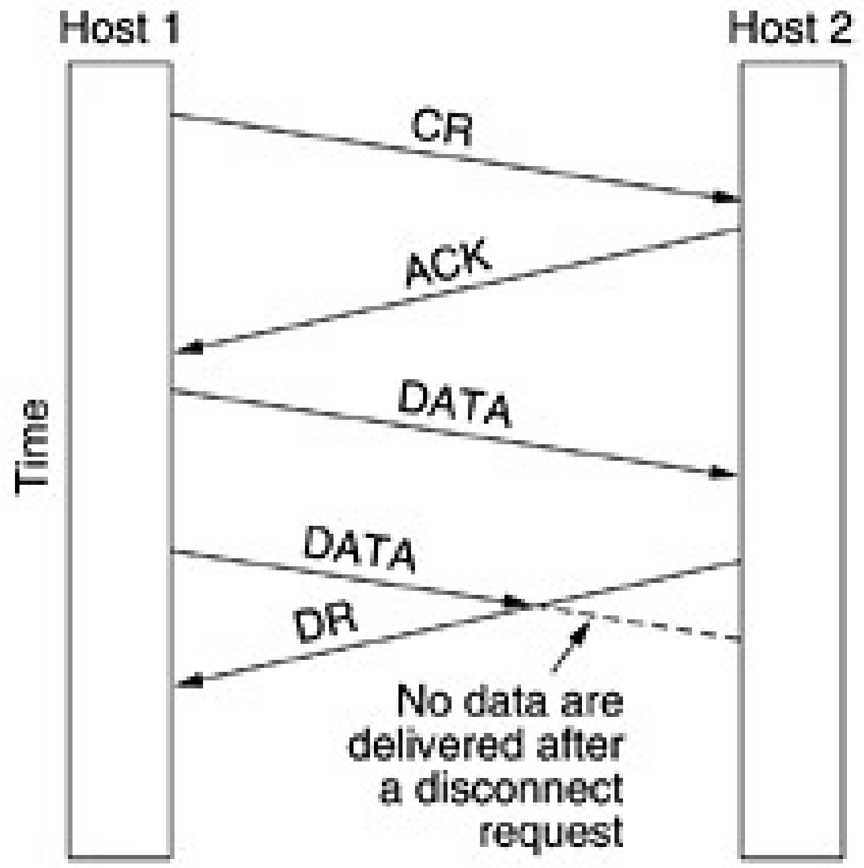


Figure: Abrupt disconnection with loss of data

Connection Release

- Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it.
- In other situations, determining that all the work has been done and the connection should be terminated is not so obvious.
- One can envision a protocol in which host 1 says: I am done. Are you done too?

  If host 2 responds: I am done too. Goodbye, the connection can be safely released.

Two-army problem

- Unfortunately, this protocol does not always work.
- There is a famous problem that illustrates this issue. It is called the **two-army problem**. Imagine that a white army is encamped in a valley, as shown in below figure.
- On both of the surrounding hillsides are blue armies.
- The white army is larger than either of the blue armies alone, but together the blue armies are larger than the white army.
- If either blue army attacks by itself, it will be defeated, but if the two blue armies attack simultaneously, they will be victorious.

Two-army problem

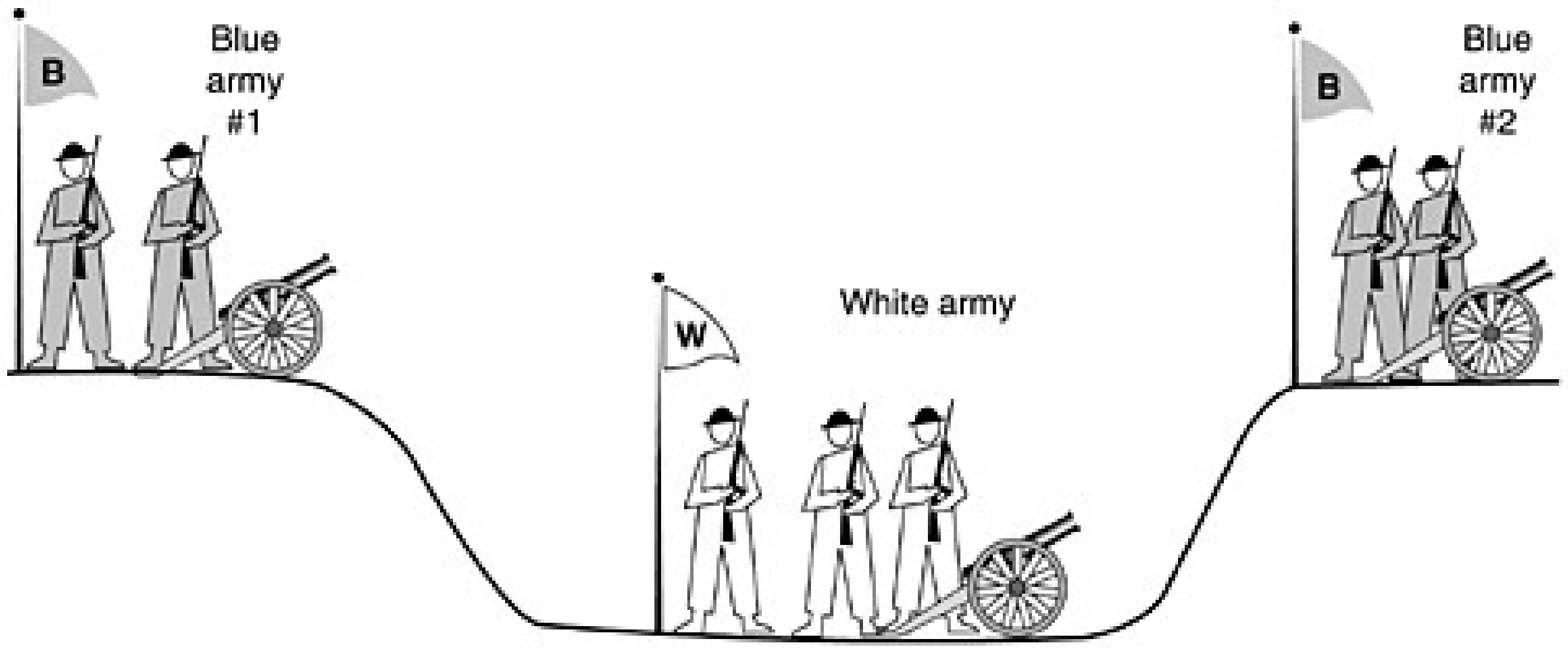


Figure: The two-army problem

Two-army problem

- The blue armies want to synchronize their attacks.
- However, their only communication medium is to send messengers on foot down into the valley, where they might be captured and the message lost (i.e., they have to use an unreliable communication channel).
- The question is: Does a protocol exist that allows the blue armies to win?
- The answer is there is NO such protocol exists.

Two-army problem

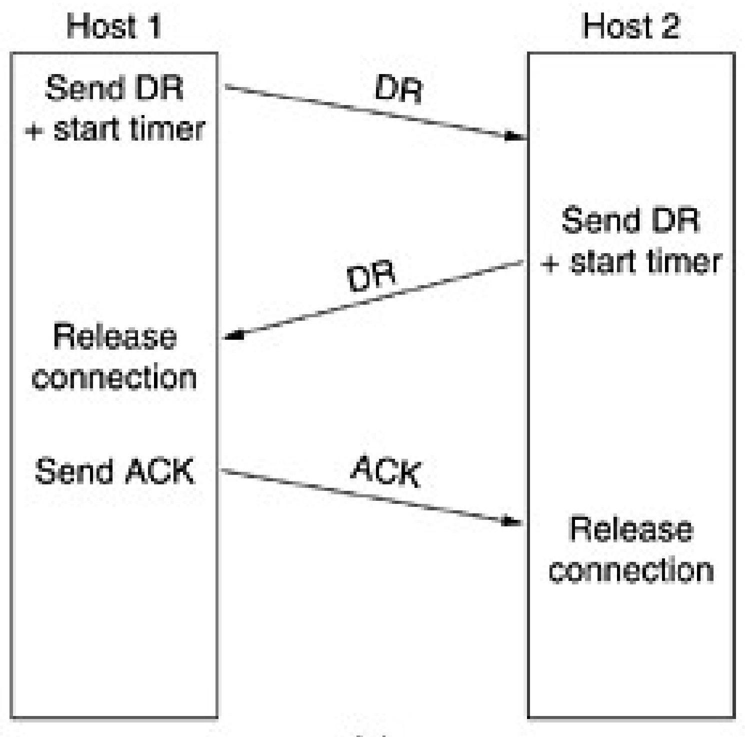
- To see the relevance of the two-army problem to releasing connections, just substitute ''disconnect'' for ''attack.'' If neither side is prepared to disconnect until it is convinced that the other side is prepared to disconnect too, the disconnection will never happen.
- In practice, one is usually prepared to take more risks when releasing connections than when attacking white armies, so the situation is not entirely hopeless.
- Below figures illustrates four scenarios of releasing using a three-way handshake. While this protocol is not infallible, it is usually adequate.

a. Normal case of three-way handshake

- In fig(a), we see the normal case in which one of the users sends a DR (DISCONNECTION REQUEST) TPDU to initiate the connection release.
- When it arrives, the recipient sends back a DR TPDU, too, and starts a timer, just in case its DR is lost.
- When this DR arrives, the original sender sends back an ACK TPDU and releases the connection.
- Finally, when the ACK TPDU arrives, the receiver also releases the connection.

a. Normal case of three-way handshake

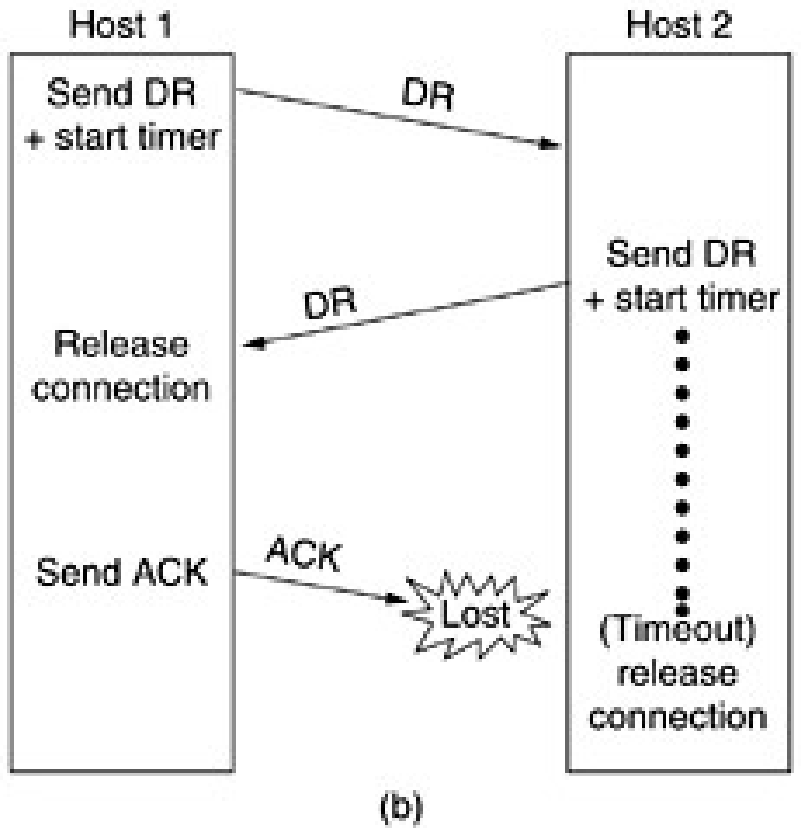
- Releasing a connection means that the transport entity removes the information about the connection from its table of currently open connections and signals the connection's owner (the transport user) somehow.
- This action is different from a transport user issuing a DISCONNECT primitive.



(a)

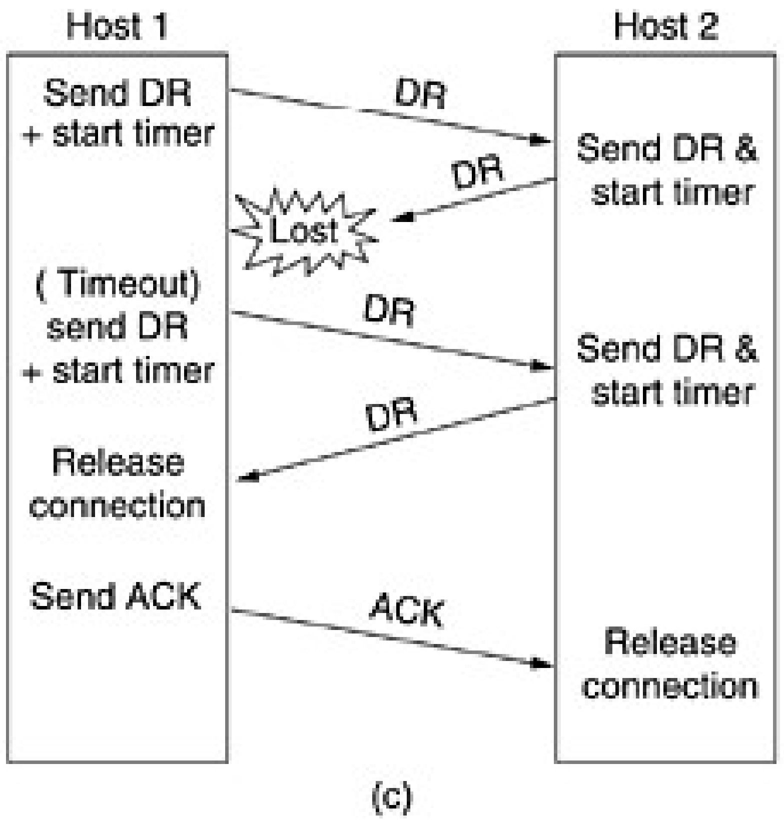
b. Final ACK lost

- If the final ACK TPDU is lost, as shown in below figure, the situation is saved by the timer. When the timer expires, the connection is released anyway.



c. Response lost

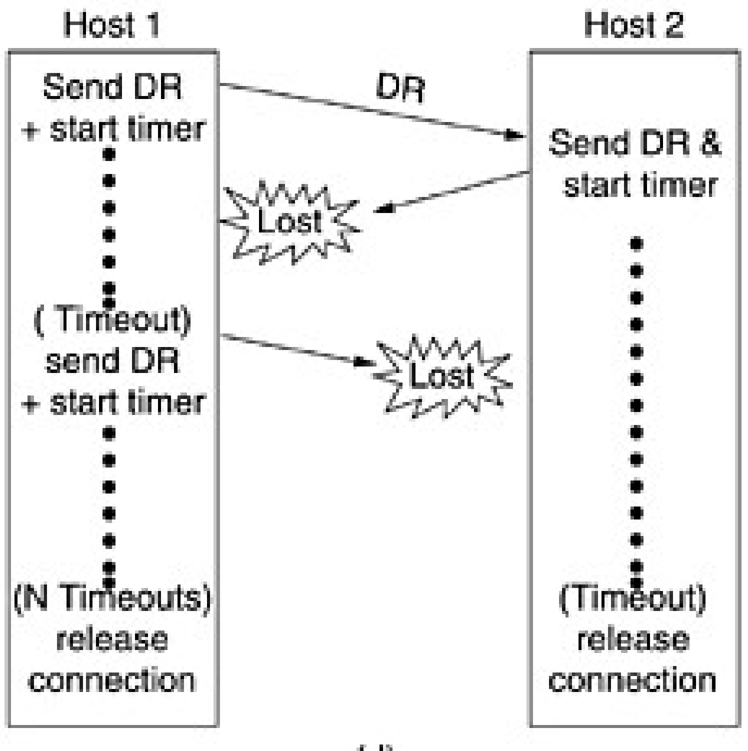
- If the second DR being lost then the user initiating the disconnection will not receive the expected response, will time out, and will start all over again.
- In figure(c) we see how this works, assuming that the second time no TPDUs are lost and all TPDUs are delivered correctly and on time.



(c)

d. Response lost and subsequent DRs lost

- Last scenario, figure(d), is the same as figure(c) except that now we assume all the repeated attempts to retransmit the DR also fail due to lost TPDUs.
- After N retries, the sender just gives up and releases the connection. Meanwhile, the receiver times out and also exits.


Host 1
Host 2
Send DR + start timer
DR
Send DR & start timer
Lost
(Timeout) send DR + start timer
Lost
(N Timeouts) release connection
(Timeout) release connection
(d)

Flow Control and Buffering

- Flow control is implemented using modified form of sliding window protocol.
- The window size is variable and is controlled by the receiver.
- The receiver sends a credit allocation to the sender.
- The credit allocation indicates how many TPDU the receiver is ready to receive if the network service is unreliable, the sender must buffer all TPDU's sent.

Flow Control and Buffering

- With reliable network service, if the sender knows that the receiver always has buffer space, it need not retain copies of the TPDU it sends.
- If the receiver cannot guarantee that every incoming TPDU will be accepted, the sender will have to buffer anyway.
- Even if the receiver has agreed to do the buffering, then the problem comes with buffer size.

Flow Control and Buffering

- If most TPDUs are nearly the same size, it is natural to organize the buffers as a pool of identical size buffers, with one TPDU per buffer.
- If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short TPDU arrives.
- If the buffer size is less than the maximum TPDU size, multiple buffers will be needed for long TPDUs with the attendant complexity.

Flow Control and Buffering

- How connections are managed while they are in use?
- For flow control, a sliding window is needed on each connection to keep a fast transmitter from overrunning a slow receiver.
- Since a host may have numerous connections, it is impractical to implement the same data link buffering strategy.
- The sender should always buffer outgoing TPDUs until they are acknowledged.
- The receiver may not dedicate specific buffers to specific connections. Instead, a single buffer pool may be maintained for all connections. When a TPDU comes in, if there is a free buffer available, the TPDU is accepted, otherwise it is discarded.

Flow Control and Buffering

- However, for high-bandwidth traffic (e.g. file transfers), it is better if the receiver dedicate a full window of buffers, to allow the data to flow at maximum speed. How large the buffer size should be?
- If most TPDUs are nearly the same size, it is natural to organize the buffers as a pool of identically-sized buffers, with one TPDU per buffer, as in figure(a).

Flow Control and Buffering

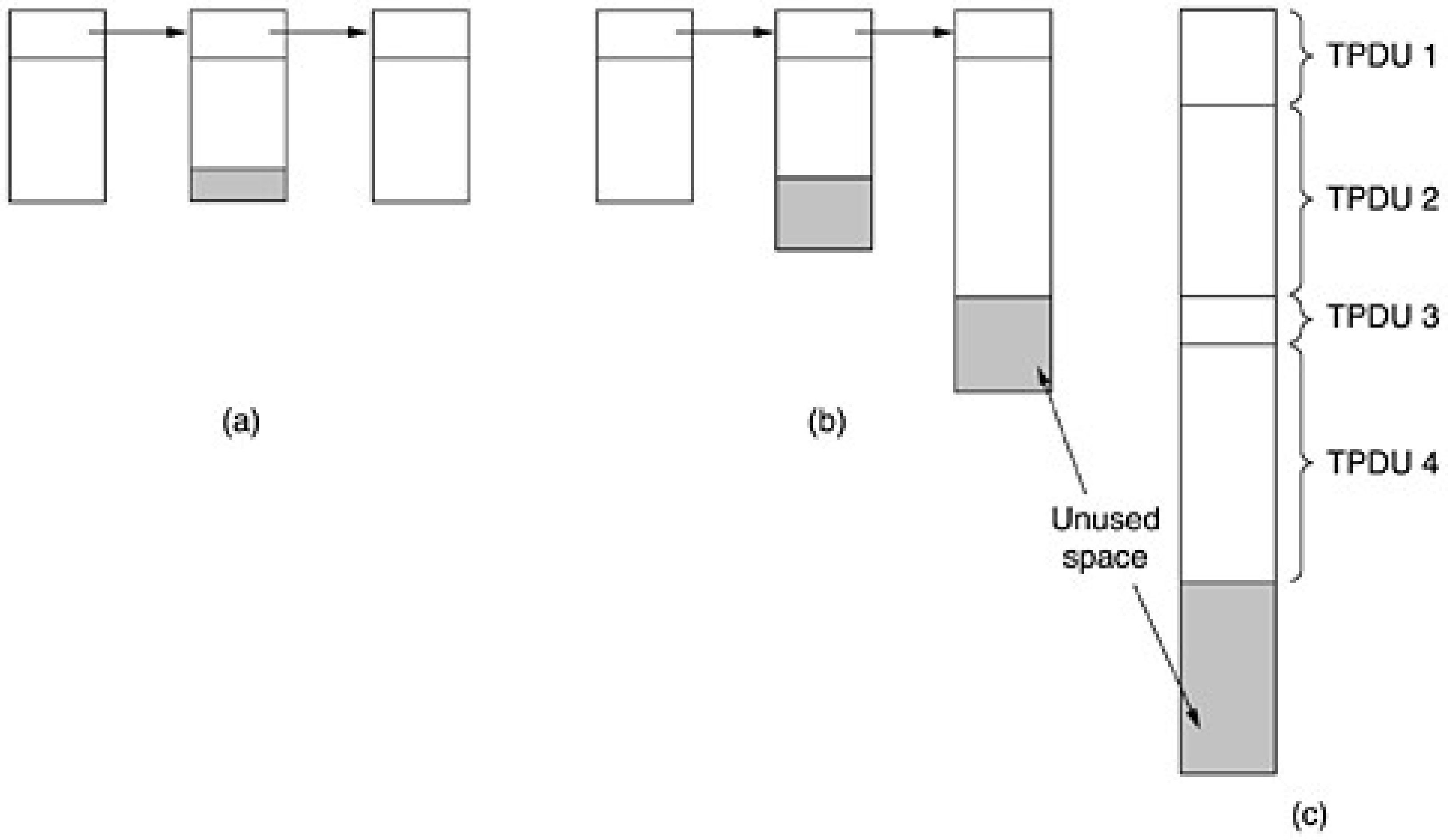


Figure (a): Chained fixed-size buffers (b) Chained variable-sized buffers (c) One large circular buffer per connection

Flow Control and Buffering

- If there is wide variation in TPDU size, from a few characters typed at a terminal to thousands of characters from file transfers, a pool of fixed-sized buffers presents problems.
  - If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short TPDU arrives.
  - If the buffer size is chosen less than the maximum TPDU size, multiple buffers will be needed for long TPDUs, with the attendant complexity.

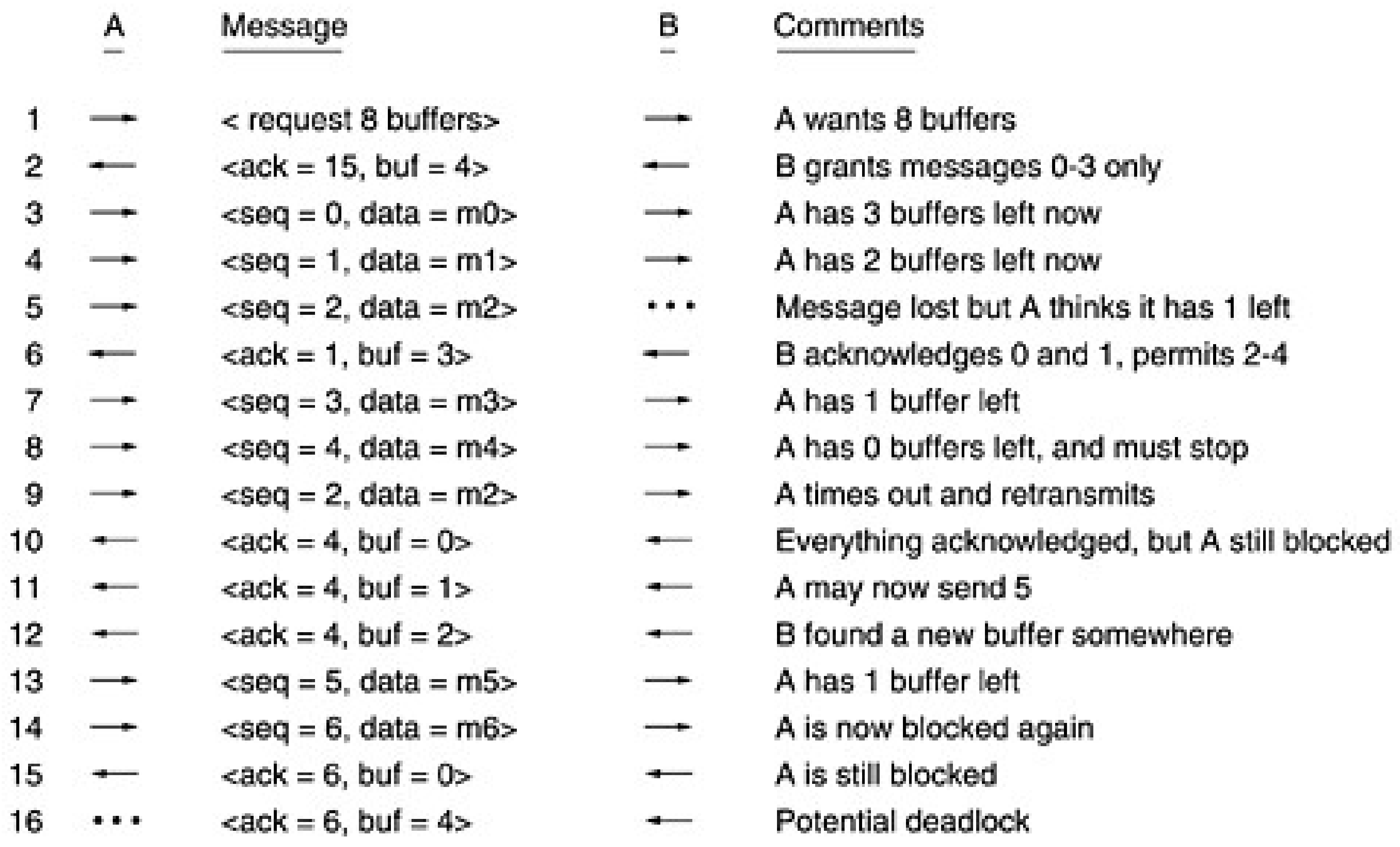
Flow Control and Buffering

- Another approach to the buffer size problem is to use variable-sized buffers, as in figure(b).
  - The advantage here is better memory utilization, at the price of more complicated buffer management.
- A third possibility is to dedicate a single large circular buffer per connection, as in figure(c).
  - This system also makes good use of memory, provided that all connections are heavily loaded, but is poor if some connections are lightly loaded.
- For low bandwidth burstly traffic, it is better to buffer at the sender, and for high bandwidth smooth traffic, it is better to buffer at the receiver.

Flow Control and Buffering

- Dynamic buffer management means allocating buffer with variable-sized window.
- Initially, the sender requests a certain number of buffers, based on its perceived needs.
- The receiver then grants as many of these as it can afford.
- Every time the sender transmits a TPDU, it must decrement its allocation, stopping altogether when the allocation reaches zero.
- The receiver then separately piggybacks both acknowledgements and buffer allocations onto the reverse traffic.

Flow Control and Buffering



	A	Message	B	Comments
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	• • •	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has 1 buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	• • •	<ack = 6, buf = 4>	←	Potential deadlock

Figure: Dynamic buffer allocation. The arrows show the direction of transmission. An ellipsis (...) indicates a lost TPDU.

Multiplexing

- Multiplexing several conversations onto connections, virtual circuits, and physical links plays a role in several layers of the network architecture.
- In the transport layer the need for multiplexing can arise in a number of ways.
- For example, if only one network address is available on a host, all transport connections on that machine have to use it.
- When a TPDU comes in, some way is needed to tell which process to give it to. This situation, called **upward multiplexing**, is shown in below figure(a).
- In this figure, four distinct transport connections all use the same network connection (e.g., IP address) to the remote host.

Multiplexing

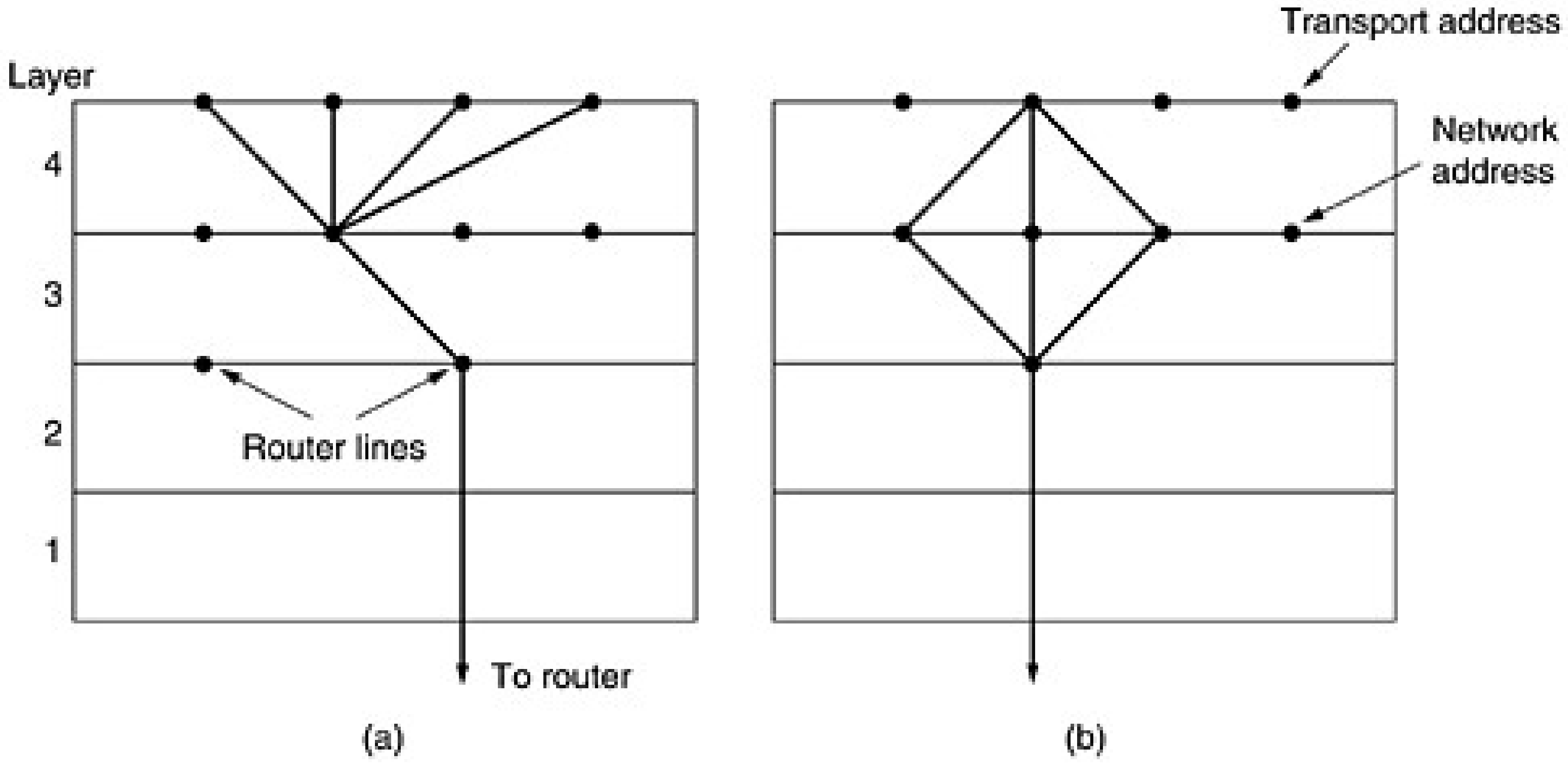


Figure (a) Upward multiplexing (b) Downward multiplexing

Multiplexing

- Multiplexing can also be useful in the transport layer for another reason.
- Suppose, for example, that a subnet uses virtual circuits internally and imposes a maximum data rate on each one.
- If a user needs more bandwidth than one virtual circuit can provide, a way out is to open multiple network connections and distribute the traffic among them on a round-robin basis, as indicated in below figure(b).
- This modus operand is called **downward multiplexing**.

Crash Recovery

- If hosts and routers are subject to crashes, recovery from these crashes becomes an issue.
- If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward.
- If the network layer provides **datagram service**, the transport entities expect lost TPDUs all the time and know how to cope with them.
- If the network layer provides **connection-oriented service**, then loss of a virtual circuit is handled by establishing a new one and then probing the remote transport entity to ask it which TPDUs it has received and which ones it has not received. The latter ones can be retransmitted.

Crash Recovery

- A more troublesome problem is how to recover from host crashes.
- In particular, it may be desirable for clients to be able to continue working when servers crash and then quickly reboot.
- To illustrate the difficulty, let us assume that one host, the client, is sending a long file to another host, the file server, using a simple stop-and-wait protocol.
- The transport layer on the server simply passes the incoming TPDUs to the transport user, one by one. Partway through the transmission, the server crashes. When it comes back up, its tables are reinitialized, so it no longer knows precisely where it was.

Crash Recovery

- In an attempt to recover its previous status, the server might send a broadcast TPDU to all other hosts, announcing that it had just crashed and requesting that its clients inform it of the status of all open connections.
- Each client can be in one of two states: one TPDU outstanding , S1, or no TPDUs outstanding, S0.
- Based on only this state information, the client must decide whether to retransmit the most recent TPDU.

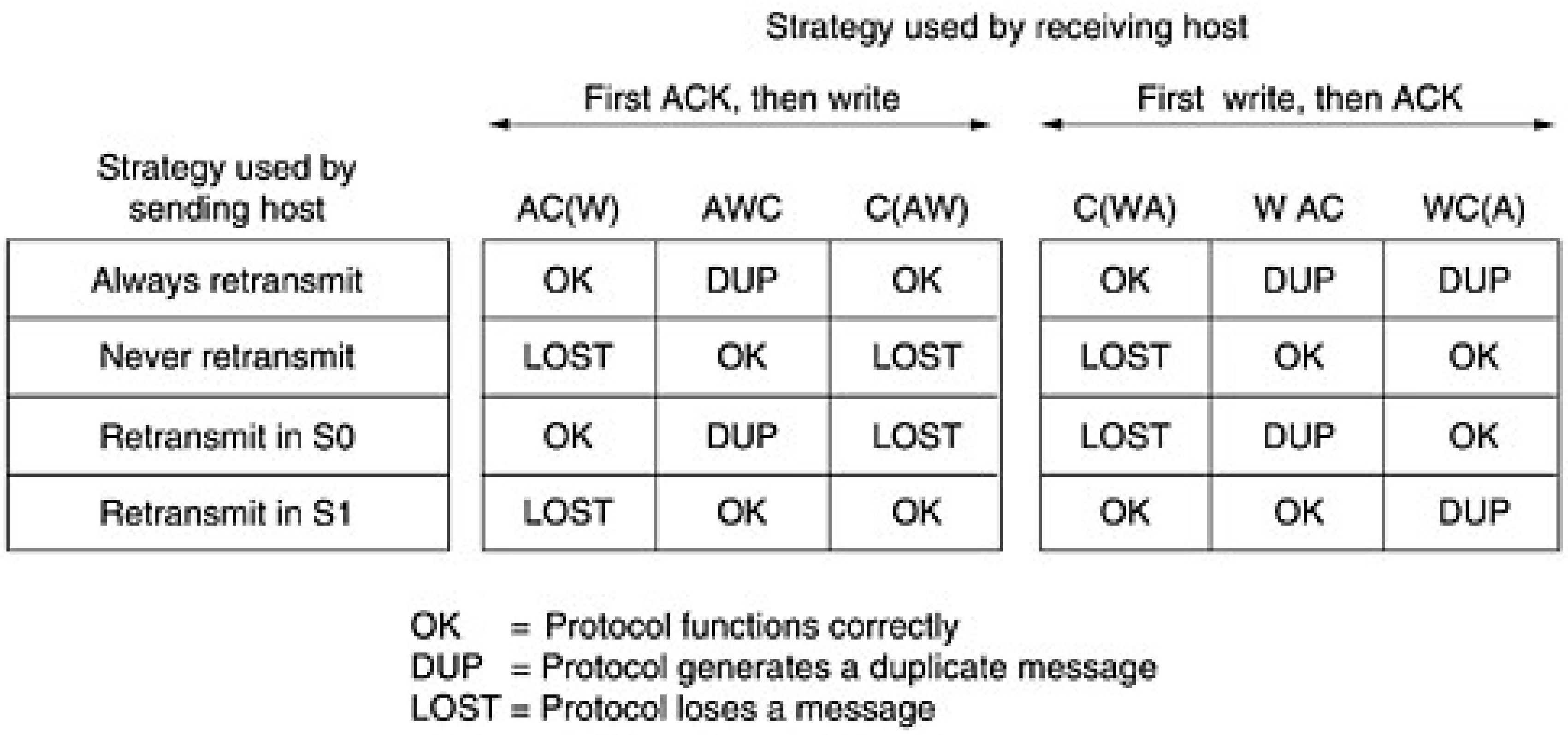
Crash Recovery

- To avoid the duplicate of data the client should retransmit only if and only if it has an unacknowledged TPDU outstanding when it learns of the crash.
- There are many situation for crash recovery.
- If the server send acknowledge and crash the server before writing the data. The writing (saving data) data and sending acknowledgement, both are different process. So the server and client programmed in any way, the lost of data and duplicate of data may occurs.

Crash Recovery

- No matter how the client and server are programmed, there are always situations where the protocol fails to recover properly.
- The server can be programmed in one of two ways: acknowledge first or write first.
- The client can be programmed in one of four ways: always retransmit the last TPDU, never retransmit the last TPDU, retransmit only in state S0, or retransmit only in state S1.
- This gives eight combinations, but as we shall see, for each combination there is some set of events that makes the protocol fail.

Crash Recovery



	Strategy used by receiving host					
	First ACK, then write			First write, then ACK		
Strategy used by sending host	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

OK	= Protocol functions correctly
DUP	= Protocol generates a duplicate message
LOST	= Protocol loses a message

Figure: Different combinations of client and server strategy