Syllabus

Module II (10 Hours) Syllabus

- Transport Layer Protocols: Introduction to transport layer, Multiplexing and de-multiplexing, Principles of Reliable data transfer - Stop-and-wait and Go-back-N design and evaluation, Connection oriented transport TCP, Connectionless transport UDP, Principles of congestion control -efficiency and fairness

- Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education, 1 st Edition (2011).
- James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

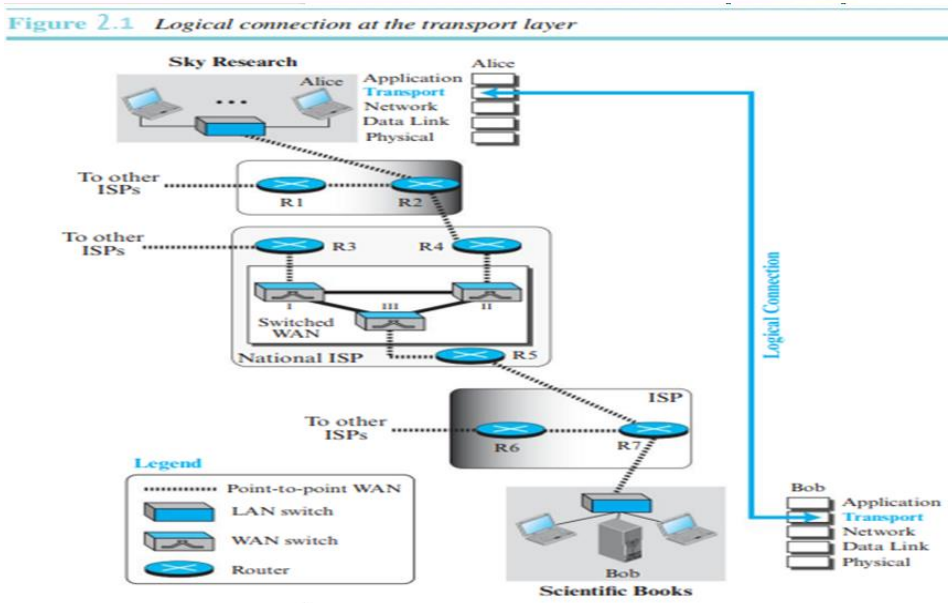
Transport Layer

- The transport layer in the TCP/IP suite is located between the application layer and the network layer.
- It provides services to the application layer and receives services from the network layer.
- The transport layer is the heart of the TCP/IP protocol suite; it is the end-to-end logical vehicle for transferring data from one point to another in the Internet.

Transport Layer - INTRODUCTION

- The transport layer is located between the application layer and the network layer.
- It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host.

Transport Layer - INTRODUCTION



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Transport-Layer Services

Process-to-Process Communication

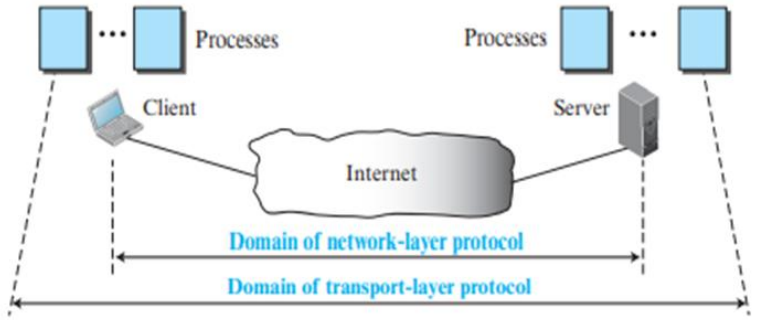
- The first duty of a transport-layer protocol is to provide process-to-process communication.
- A process is an application-layer entity (running program) that uses the services of the transport layer.
- We need to understand the difference between host-to-host communication and process-to-process communication.
- The network layer is responsible for communication at the computer level (host-to-host communication). A network-layer protocol can deliver the message only to the destination computer. However, this is an incomplete delivery.
- The message still needs to be handed to the correct process. This is where a transport-layer protocol takes over.
- A transport-layer protocol is responsible for delivery of the message to the appropriate process.

Transport-Layer Services

Process-to-Process Communication

- Figure 2.2 shows the domains of a network layer and a transport layer.

Figure 2.2 *Network layer versus transport layer*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Transport-Layer Services

Addressing: Port Numbers

- The local host and the remote host are defined using IP addresses.
- To define the processes, we need second identifiers, called port numbers.
- In the TCP/IP protocol suite, the port numbers are integers between 0 and 65,535 (16 bits).
- The client program defines itself with a port number, called the ephemeral port number.
- The word ephemeral means short-lived and is used because the life of a client is normally short.

Transport-Layer Services

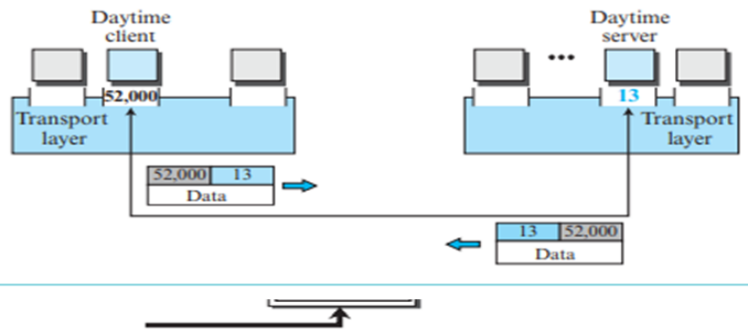
Addressing: Port Numbers

- An ephemeral port number is recommended to be greater than 1,023 for some client/server programs to work properly.
- The server process must also define itself with a port number.
- TCP/IP has decided to use universal port numbers for servers; these are called well-known port numbers.
- Every client process knows the well-known port number of the corresponding server process.

Transport-Layer Services

Addressing: Port Numbers

Figure 2.3 *Port numbers*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

- The client process can use an ephemeral (temporary) port number, 52,000, to identify itself, the server process must use the well-known (permanent) port number 13.

Transport-Layer Services

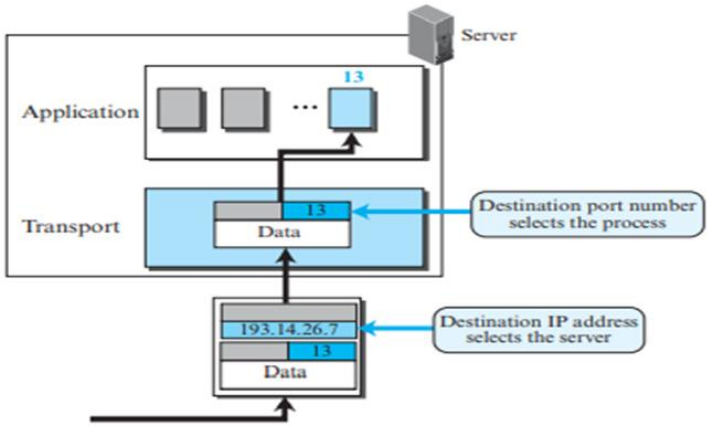
Addressing: Port Numbers

- IP addresses and port numbers play different roles in selecting the final destination of data.
- The destination IP address defines the host among the different hosts in the world.
- After the host has been selected, the port number defines one of the processes on this particular host

Transport-Layer Services

Addressing: Port Numbers

Figure 2.4 *IP addresses versus port numbers*



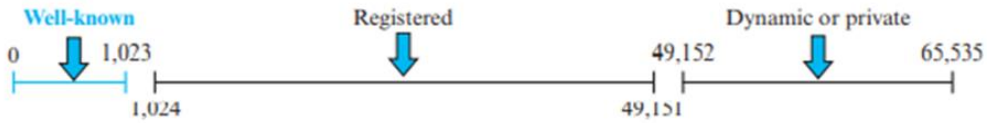
[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Transport-Layer Services

Addressing: Port Numbers

- Port numbers into three ranges: well-known, registered, and dynamic (or private)
- Well-known ports: The ports ranging from 0 to 1,023 are assigned and controlled by ICANN.
- Registered ports: The ports ranging from 1,024 to 49,151 are not assigned or controlled by ICANN.
- Dynamic ports: The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used as temporary or private port numbers.

Figure 2.5 *ICANN ranges* The Internet Corporation for Assigned Names and Numbers



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Transport-Layer Services

Example 2.1

- In UNIX, the well-known ports are stored in a file called /etc/services.
- Each line in this file gives the name of the server and the well-known port number.
- We can use the grep utility to extract the line corresponding to the desired application.
- The following shows the port for TFTP.
- Note that TFTP can use port 69 on either UDP or TCP.
  - $grep tftp/etc/services
  - tftp 69/tcp
  - tftp 69/udp

Transport-Layer Services

Example 2.1

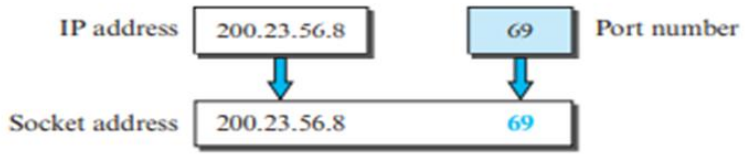
- SNMP uses two port numbers (161 and 162), each for a different purpose.
  - $grep tftp/etc/services
  - snmp161/tcp#Simple Net Mgmt Proto
  - snmp161/udp#Simple Net Mgmt Proto
  - snmptrap162/udp#Traps for SNMP

Transport-Layer Services

Socket Addresses

- A transport-layer protocol in the TCP suite needs both the IP address and the port number, at each end, to make a connection. The combination of an IP address and a port number is called a socket address.
- The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely (see Figure 2.6).

Figure 2.6 *Socket address*

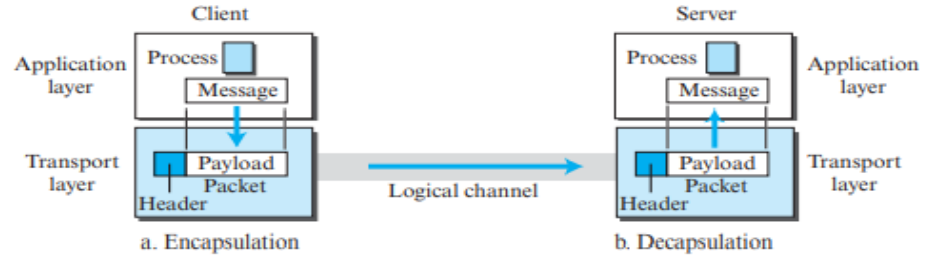


Transport-Layer Services

Encapsulation and Decapsulation

- To send a message from one process to another, the transport-layer protocol encapsulates and decapsulates.
- The packets at the transport layers in the Internet are called user datagrams, segments, or packets, depending on what transport-layer protocol we use.

Figure 2.7 *Encapsulation and decapsulation*



Transport-Layer Services

Multiplexing and Demultiplexing

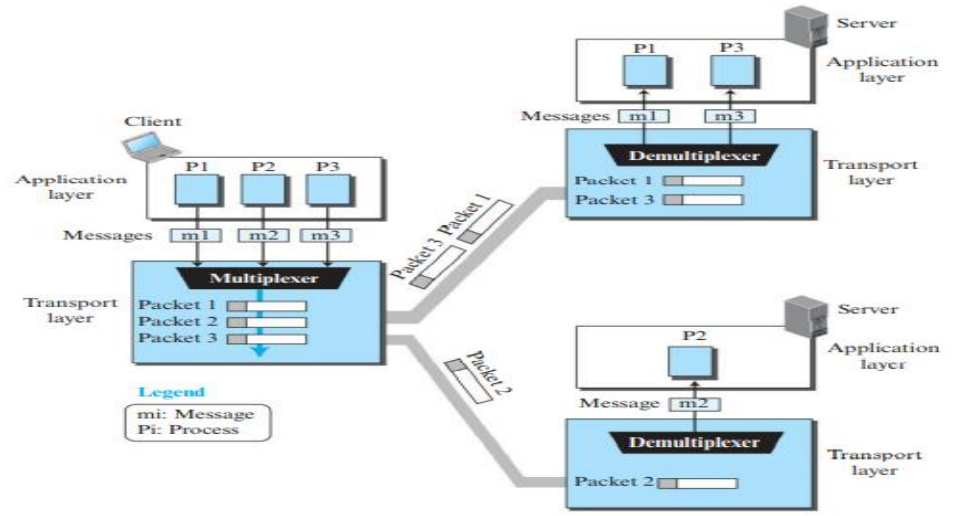
- Whenever an entity accepts items from more than one source, this is referred to as multiplexing (many to one); whenever an entity delivers items to more than one source, this is referred to as demultiplexing (one to many).

- The transport layer at the source performs multiplexing; the transport layer at the destination performs demultiplexing (Figure 2.8).

Transport-Layer Services

Multiplexing and Demultiplexing

Figure 2.8 *Multiplexing and demultiplexing*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Transport-Layer Services

Flow Control at Transport Layer

- In communication at the transport layer, we are dealing with four entities: sender process, sender transport layer, receiver transport layer, and receiver process.

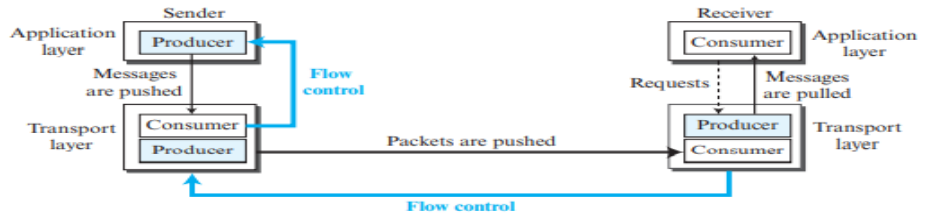
- The sending process at the application layer is only a producer. It produces message chunks and pushes them to the transport layer.
- The sending transport layer has a double role: it is both a consumer and a producer. It consumes the messages pushed by the producer. It encapsulates the messages in packets and pushes them to the receiving transport layer.

Transport-Layer Services

Flow Control at Transport Layer

- The receiving transport layer also has a double role, it is the consumer for the packets received from the sender and the producer that decapsulates the messages and delivers them to the application layer.
- The last delivery is normally a pulling delivery; the transport layer waits until the application-layer process asks for messages.

Figure 2.9 *Flow control at the transport layer*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Transport-Layer Services

Buffers

- Although flow control can be implemented in several ways, one of the solutions is normally to use two buffers: one at the sending transport layer and the other at the receiving transport layer.
- A buffer is a set of memory locations that can hold packets at the sender and receiver.
- When the buffer of the sending transport layer is full, it informs the application layer to stop passing chunks of messages; when there are some vacancies, it informs the application layer that it can pass message chunks again.
- When the buffer of the receiving transport layer is full, it informs the sending transport layer to stop sending packets. When there are some vacancies, it informs the sending transport layer that it can send packets again.

Transport-Layer Services

Error Control

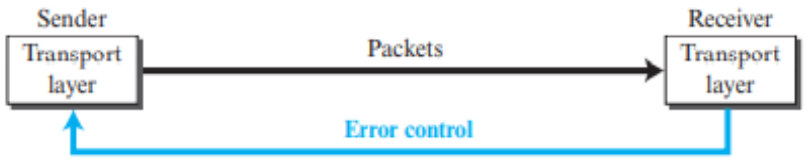
- In the Internet, since the underlying network layer (IP) is unreliable, we need to make the transport layer reliable if the application requires reliability.
- Reliability can be achieved to add error control services to the transport layer.
- Error control at the transport layer is responsible for
  1. Detecting and discarding corrupted packets.
  2. Keeping track of lost and discarded packets and resending them.
  3. Recognizing duplicate packets and discarding them.
  4. Buffering out-of-order packets until the missing packets arrive.

Transport-Layer Services

Error Control

- Error control, unlike flow control, involves only the sending and receiving transport layers.
- We are assuming that the message chunks exchanged between the application and transport layers are error free.
- Figure 2.10 shows the error control between the sending and receiving transport layer.

Figure 2.10 *Error control at the transport layer*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Transport-Layer Services

Sequence Numbers

- Error control requires that the sending transport layer knows which packet is to be resent and the receiving transport layer knows which packet is a duplicate, or which packet has arrived out of order. This can be done if the packets are numbered.
- We can add a field to the transport-layer packet to hold the sequence number of the packet.
- When a packet is corrupted or lost, the receiving transport layer can inform the sending transport layer to resend that packet using the sequence number.
- The receiving transport layer can also detect duplicate packets if two received packets have the same sequence number.
- The out-of-order packets can be recognized by observing gaps in the sequence numbers.

Transport-Layer Services

Sequence Numbers

•Packets are numbered sequentially.

•However, because we need to include the sequence number of each packet in the header, we need to set a limit.

•If the header of the packet allows m bits for the sequence number, the sequence numbers range from 0 to $2^m - 1$.

•For example, if m is 4, the only sequence numbers are 0 through 15, inclusive.

•However, we can wrap around the sequence. The sequence numbers are modulo 2^m.

•For error control, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

Transport-Layer Services

Acknowledgment

- We can use both positive and negative signals as error control, but we discuss only positive signals, which are more common at the transport layer.
- The receiver side can send an acknowledgment (ACK) for each of a collection of packets that have arrived safe and sound.
- The receiver can simply discard the corrupted packets. The sender can detect lost packets if it uses a timer. When a packet is sent, the sender starts a timer.
- If an ACK does not arrive before the timer expires, the sender resends the packet.

Transport-Layer Services

Sliding Window

•Since the sequence numbers used modulo 2^m, a circle can represent the sequence numbers from 0 to $2^m - 1$.

•The buffer is represented as a set of slices, called the sliding window, that occupies part of the circle at any time.

•At the sender site, when a packet is sent, the corresponding slice is marked.

•When all the slices are marked, it means that the buffer is full and no further messages can be accepted from the application layer.

•When an acknowledgment arrives, the corresponding slice is unmarked.

Transport-Layer Services

Sliding Window

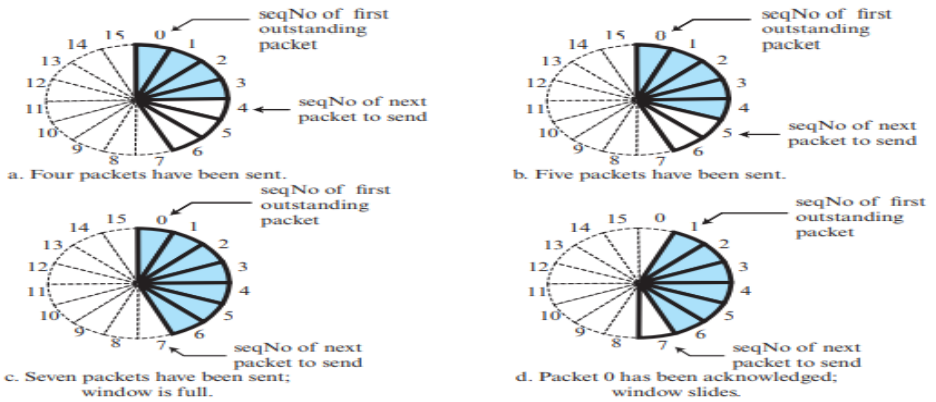
- If some consecutive slices from the beginning of the window are unmarked, the window slides over the range of the corresponding sequence numbers to allow more free slices at the end of the window.

Transport-Layer Services

Sliding Window

- Figure 2.11 shows the sliding window at the sender. The sequence numbers are in modulo 16 (m = 4) and the size of the window is 7.
- Note that the sliding window is just an abstraction.

Figure 2.11 *Sliding window in circular format*

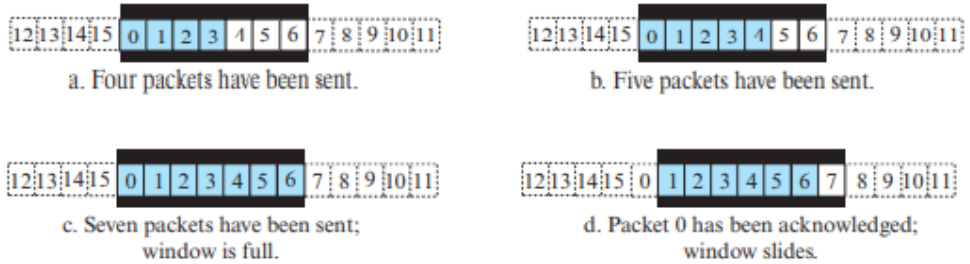


Transport-Layer Services

Sliding Window

- Most protocols show the sliding window using linear representation.

Figure 2.12 *Sliding window in linear format*



Transport-Layer Services

Congestion Control

- An important issue in a packet-switched network, such as the Internet, is congestion.
- Congestion in a network may occur if the load on the network—the number of packets sent to the network is greater than the capacity of the network—the number of packets a network can handle.
- Congestion control refers to the mechanisms and techniques that control the congestion and keep the load below the capacity.

Transport-Layer Services

Connectionless and Connection-Oriented Services

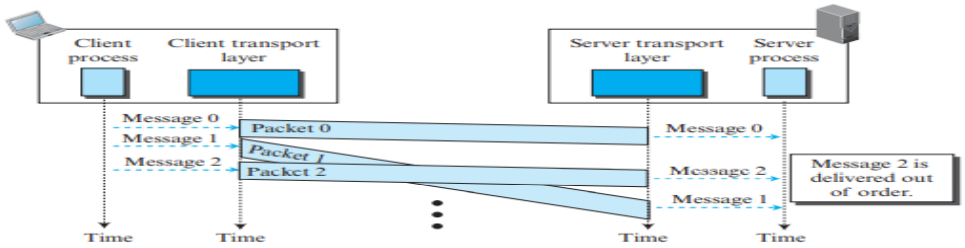
- A transport-layer protocol can provide two types of services: connectionless and connection-oriented.
- At the transport layer, we are not concerned about the physical paths of packets (we assume a logical connection between two transport layers).
- Connectionless service at the transport layer means independency between packets; connection-oriented means dependency.

Transport-Layer Services

Connectionless Service

- Assume that a client process has three chunks of messages to send to a server process.
- The chunks are handed over to the connectionless transport protocol in order.
- However, since there is no dependency between the packets at the transport layer, the packets may arrive out of order at the destination and will be delivered out of order to the server process

Figure 2.13 *Connectionless service*



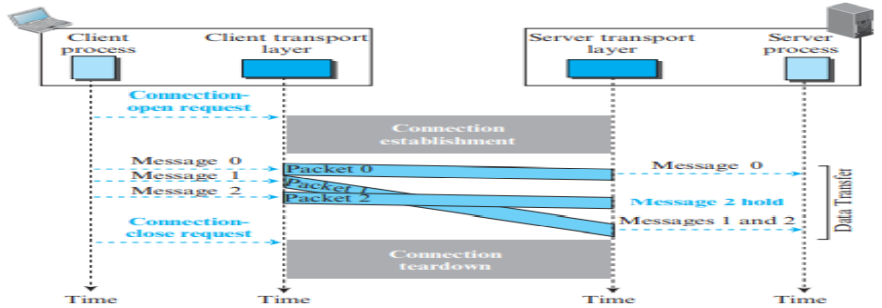
[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Transport-Layer Services

Connection-Oriented Service

- In a connection-oriented service, the client and the server first need to establish a logical connection between themselves.
- The data exchange can only happen after the connection establishment.
- After data exchange, the connection needs to be torn down (Figure 2.14).
- We can implement flow control, error control, and congestion control in a connection oriented protocol.

Figure 2.14 *Connection-oriented service*



[Behrouz A Forouzan, Firouz Mosharraf, "Computer Networks: A top down Approach", McGraw Hill Education]

Multiplexing and Demultiplexing

- How a receiving host directs an incoming transport-layer segment to the appropriate socket.
- Each transport-layer segment has a set of fields in the segment for this purpose.
- At the receiving end, the transport layer examines these fields to identify the receiving socket and then directs the segment to that socket.
- This job of delivering the data in a transport-layer segment to the correct socket is called demultiplexing.
- The job of gathering data chunks at the source host from different sockets, encapsulating each data chunk with header information (that will later be used in demultiplexing) to create segments, and passing the segments to the network layer is called multiplexing.

Multiplexing and Demultiplexing

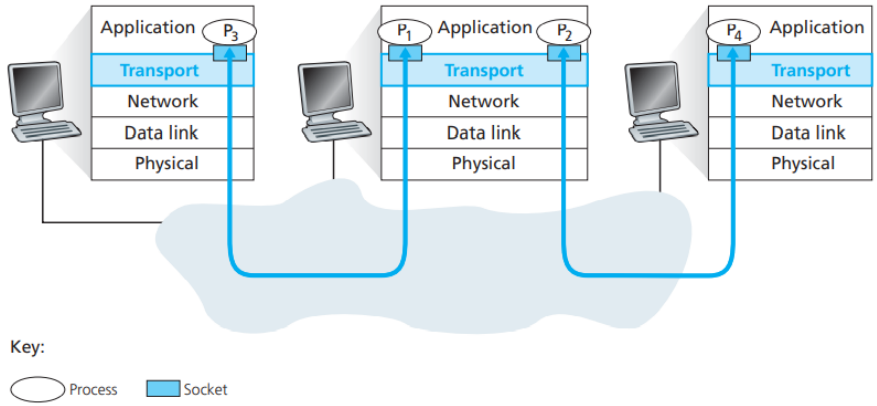


Figure 2.15 ♦ Transport-layer multiplexing and demultiplexing

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Multiplexing and Demultiplexing

- Transport-layer multiplexing requires (1) that sockets have unique identifiers, and (2) that each segment have special fields that indicate the socket to which the segment is to be delivered.
- These special fields, illustrated in Figure 2.16, are the source port number field and the destination port number field.

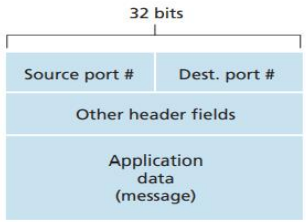


Figure 2.16 ♦ Source and destination port-number fields in a transport-layer segment

Multiplexing and Demultiplexing

- Each port number is a 16-bit number, ranging from 0 to 65535.
- The port numbers ranging from 0 to 1023 are called well-known port numbers and are restricted, which means that they are reserved for use by well-known application protocols such as HTTP (which uses port number 80) and FTP (which uses port number 21).
- The list of well-known port numbers is given in RFC 1700 and is updated at http://www.iana.org [RFC 3232].

- Connectionless Multiplexing and Demultiplexing – UDP [REFER]
- Connection-Oriented Multiplexing and Demultiplexing – TCP [REFER]

Principles of Reliable Data Transfer

- With a reliable channel, no transferred data bits are corrupted (flipped from 0 to 1, or vice versa) or lost, and all are delivered in the order in which they were sent.
- This is precisely the service model offered by TCP to the Internet applications that invoke it.
- It is the responsibility of a reliable data transfer protocol to implement this service abstraction.
- This task is made difficult by the fact that the layer below the reliable data transfer protocol may be unreliable.

Principles of Reliable Data Transfer

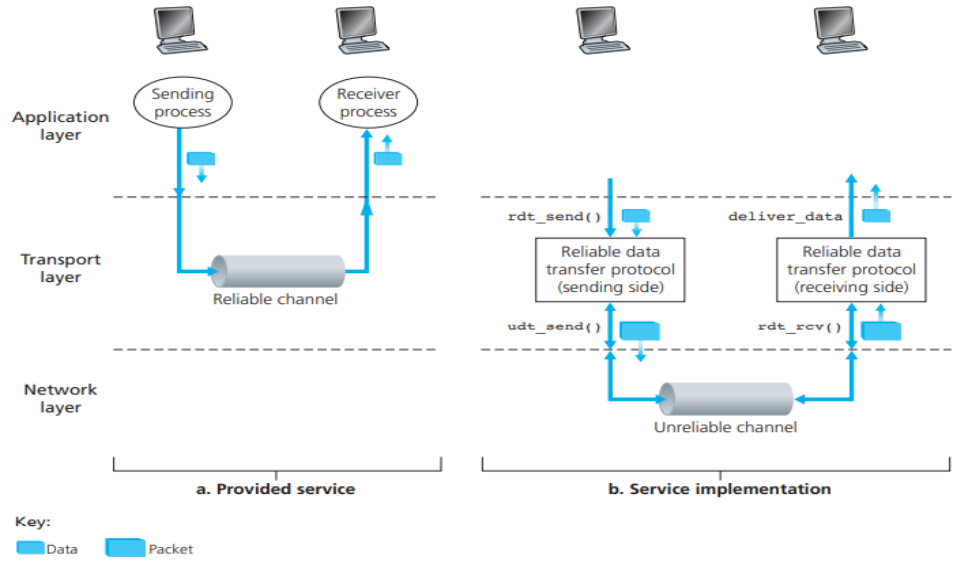


Figure 2.17 ♦ Reliable data transfer: Service model and service implementation

Principles of Reliable Data Transfer

Building a Reliable Data Transfer Protocol

- We can go through a series of protocols, each one becoming more complex, arriving at a flawless, reliable data transfer protocol.

Reliable Data Transfer over a Perfectly Reliable Channel: rdt1.0

- We first consider the simplest case, in which the underlying channel is completely reliable.
- The finite-state machine (FSM) definitions for the rdt1.0 sender and receiver are shown in Figure 2.18.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Perfectly Reliable Channel: rdt1.0

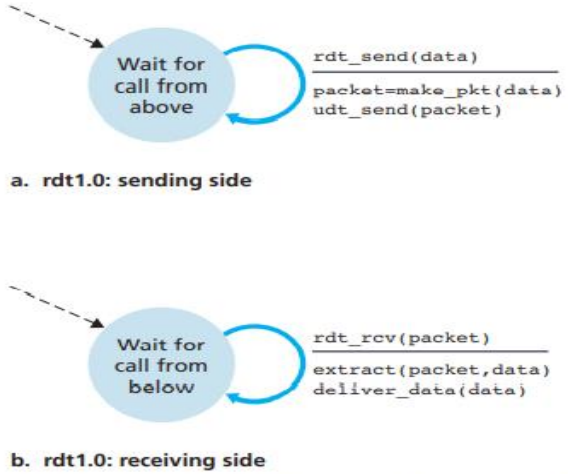


Figure 2.18♦ rdt1.0 – A protocol for a completely reliable channel

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.0

- A more realistic model of the underlying channel is one in which bits in a packet may be corrupted.
- Such bit errors typically occur in the physical components of a network as a packet is transmitted, propagates, or is buffered.
- The message-dictation protocol uses both positive acknowledgments ("OK") and negative acknowledgments ("Please repeat that.").
- These control messages allow the receiver to let the sender know what has been received correctly, and what has been received in error and thus requires repeating.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.0

- In a computer network setting, reliable data transfer protocols based on retransmission are known as ARQ (Automatic Repeat reQuest) protocols.
- Fundamentally, three additional protocol capabilities are required in ARQ protocols to handle the presence of bit errors:
  - Error detection
  - Receiver feedback
  - Retransmission
- Figure 2.19 shows the FSM representation of rdt2.0, a data transfer protocol employing error detection, positive acknowledgments, and negative acknowledgments.

Principles of Reliable Data Transfer

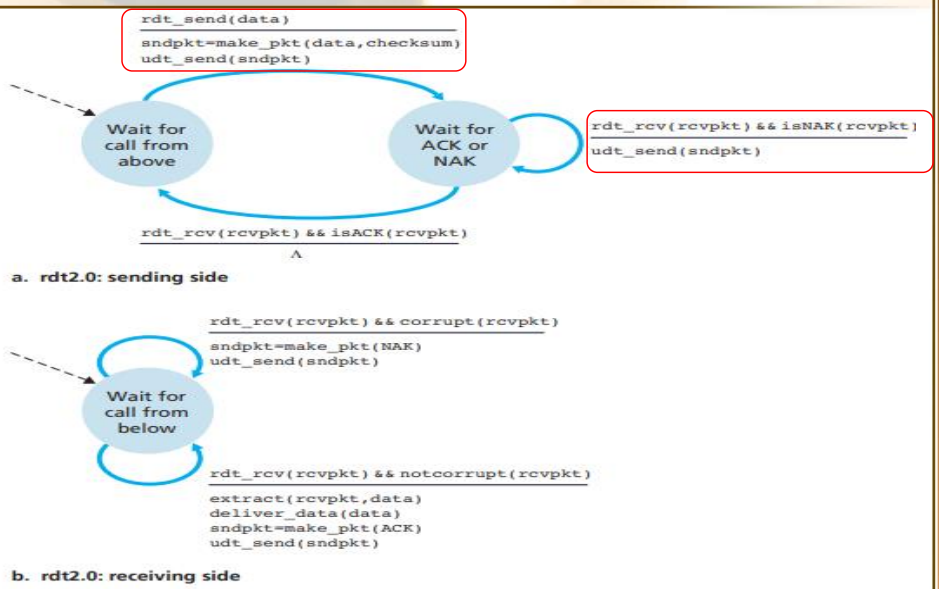


Figure 2.19 ♦ rdt2.0–A protocol for a channel with bit errors

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.0

- The send side of rdt2.0 has two states.
- In the leftmost state, the send-side protocol is waiting for data to be passed down from the upper layer. When the rdt_send(data) event occurs, the sender will create a packet (sndpkt) containing the data to be sent, along with a packet checksum and then send the packet via the udt_send(sndpkt) operation.
- In the rightmost state, the sender protocol is waiting for an ACK or a NAK packet from the receiver.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.0

- It is important to note that when the sender is in the wait-for-ACK-or-NAK state, it cannot get more data from the upper layer; that is, the rdt_send() event can not occur; that will happen only after the sender receives an ACK and leaves this state.
- Thus, the sender will not send a new piece of data until it is sure that the receiver has correctly received the current packet.
- Because of this behavior, protocols such as rdt2.0 are known as stop-and-wait protocols.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.0

- The receiver-side FSM for rdt2.0 still has a single state.
- On packet arrival, the receiver replies with either an ACK or a NAK, depending on whether or not the received packet is corrupted.
- In Figure 2.19, the notation rdt_rcv(rcvpkt) && corrupt(rcvpkt) corresponds to the event in which a packet is received and is found to be in error.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.0

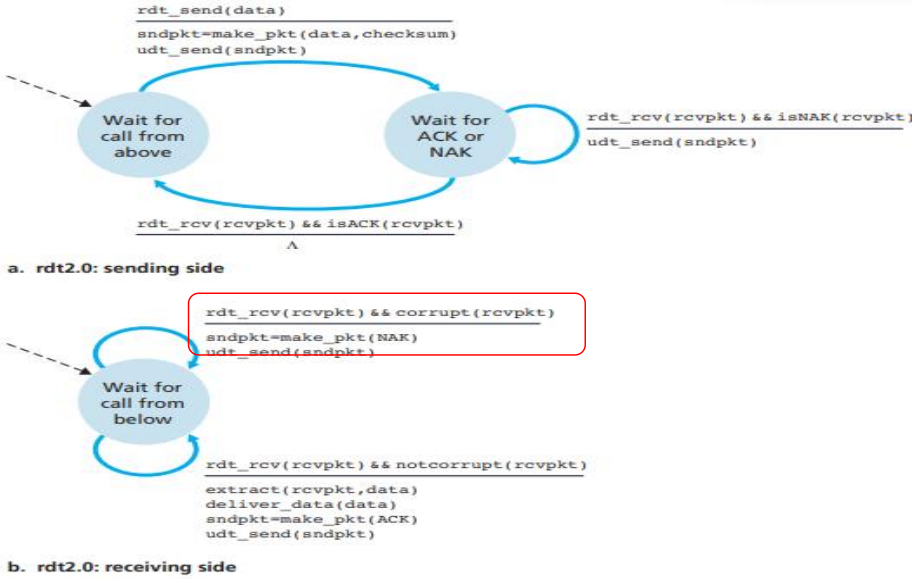


Figure 2.19 ♦ rdt2.0–A protocol for a channel with bit errors

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.0

- Protocol rdt2.0 has a fatal flaw. We haven't accounted for the possibility that the ACK or NAK packet could be corrupted! AND whether an arriving packet contains new data or is a retransmission!
- A simple solution to this new problem (and one adopted in almost all existing data transfer protocols, including TCP) is to add a new field to the data packet and have the sender number its data packets by putting a sequence number into this field.
- The receiver then need only check this sequence number to determine whether or not the received packet is a retransmission.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.0

- For this simple case of a stop-and wait protocol, a 1-bit sequence number will suffice, since it will allow the receiver to know whether the sender is resending the previously transmitted packet (the sequence number of the received packet has the same sequence number as the most recently received packet) or a new packet (the sequence number changes, moving "forward" in modulo-2 arithmetic).
- Protocol rdt2.1 uses both positive and negative acknowledgments from the receiver to the sender.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.1

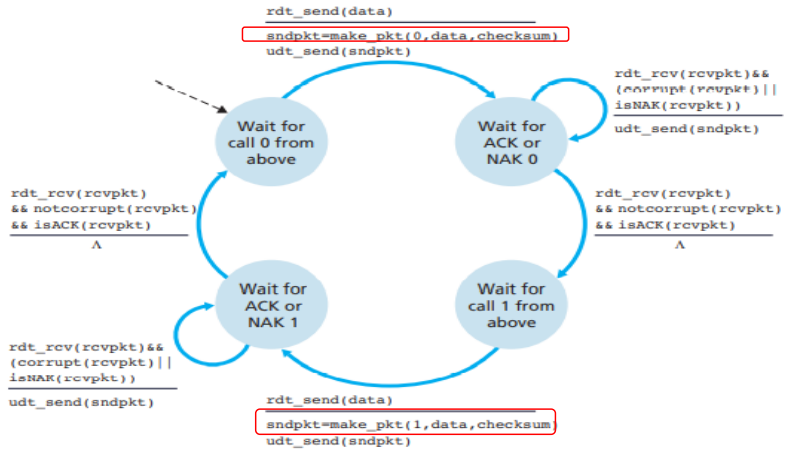


Figure 2.20 ♦ rdt2.1 sender

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.1

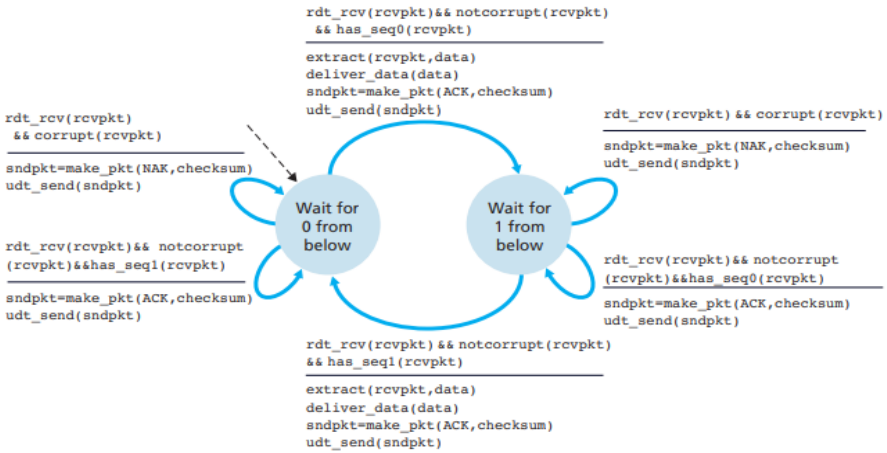


Figure 2.21 ♦ rdt2.1 receiver

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.2

- One subtle change between rtdt2.1 and rdt2.2 is that the receiver must now include the sequence number of the packet being acknowledged by an ACK message (this is done by including the ACK0 or ACK1 argument in make_pkt() in the receiver FSM), and the sender must now check the sequence number of the packet being acknowledged by a received ACK message (this is done by including the 0 or 1 argument in isACK() in the sender FSM).

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.2

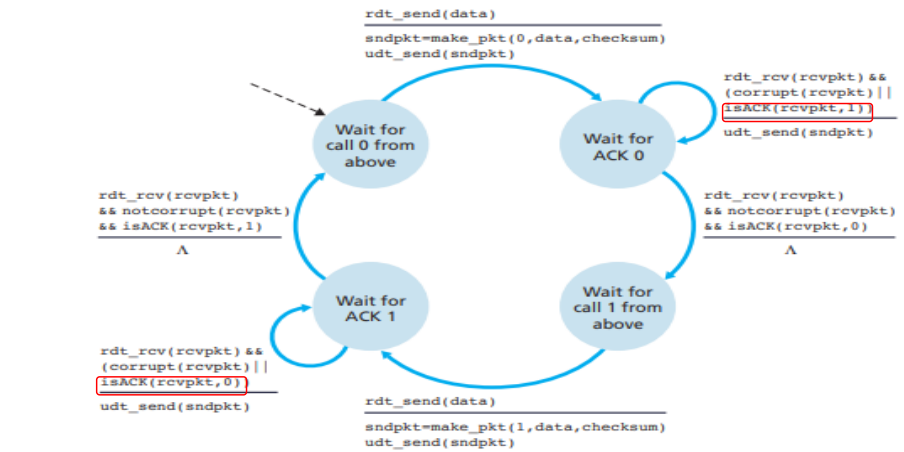


Figure 2.22 ♦ rdt2.2 sender

Principles of Reliable Data Transfer

Reliable Data Transfer over a Channel with Bit Errors: rdt2.2

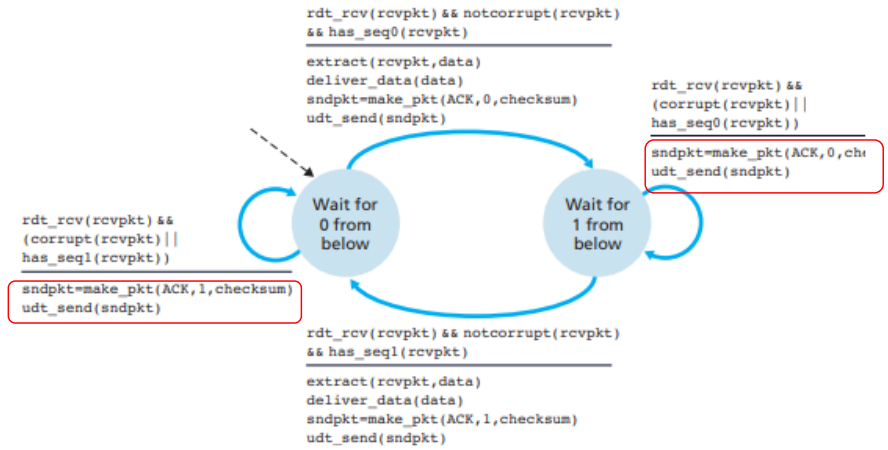


Figure 2.23 ♦ rdt2.2 receiver

Principles of Reliable Data Transfer

Reliable Data Transfer over a Lossy Channel with Bit Errors: rdt3.0

- Suppose now that in addition to corrupting bits, the underlying channel can lose packets as well, a not-uncommon event in today's computer networks (including the Internet).
- Two additional concerns must now be addressed by the protocol: how to detect packet loss and what to do when packet loss occurs.
- The use of checksumming, sequence numbers, ACK packets, and retransmissions—the techniques already developed in rdt2.2—will allow us to answer the latter concern.
- Handling the first concern will require adding a new protocol mechanism.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Lossy Channel with Bit Errors: rdt3.0

- Implementing a time-based retransmission mechanism requires a countdown timer that can interrupt the sender after a given amount of time has expired. The sender will thus need to be able to (1) start the timer each time a packet (either a first-time packet or a retransmission) is sent, (2) respond to a timer interrupt (taking appropriate actions), and (3) stop the timer.
- Because packet sequence numbers alternate between 0 and 1, protocol rdt3.0 is sometimes known as the alternating-bit protocol.

Principles of Reliable Data Transfer

Reliable Data Transfer over a Lossy Channel with Bit Errors: rdt3.0

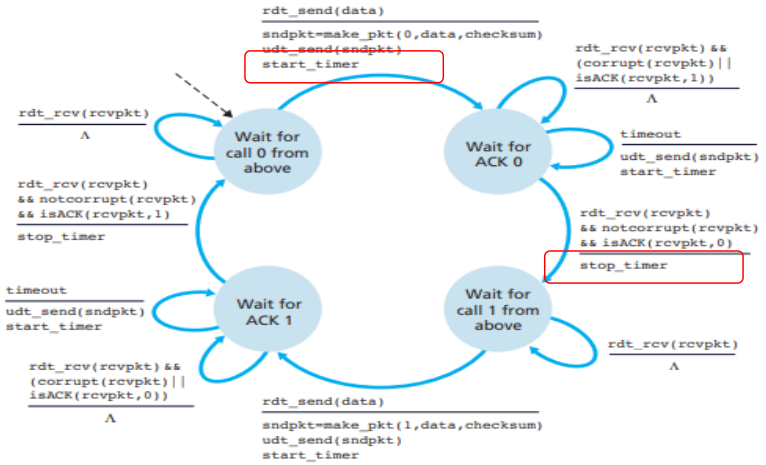


Figure 2.24 ♦ rdt3.0 sender

Principles of Reliable Data Transfer

Reliable Data Transfer over a Lossy Channel with Bit Errors: rdt3.0

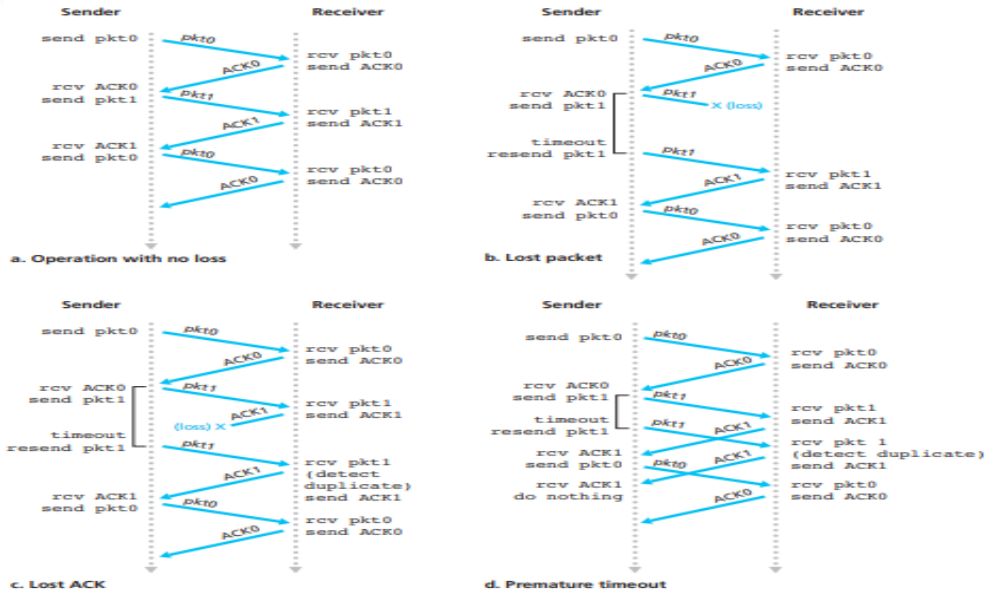


Figure 2.25 ♦ Operation of rdt3.0, the alternating-bit protocol

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

- Protocol rdt3.0 is a functionally correct protocol, but it is unlikely that anyone would be happy with its performance, particularly in today's high-speed networks.
- At the heart of rdt3.0's performance problem is the fact that it is a stop-and-wait protocol.

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

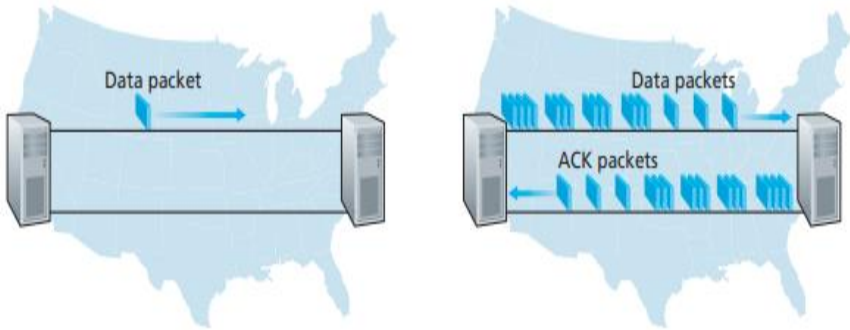


Figure 2.26 ♦ Stop-and-wait versus pipelined protocol

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Stop-and-wait protocol

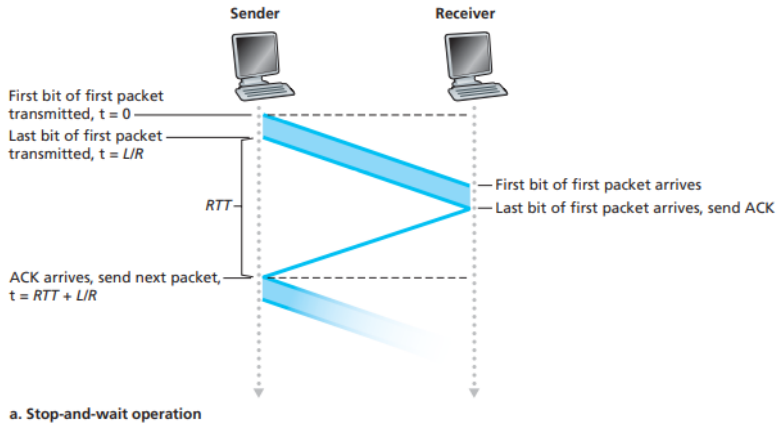
- The speed-of-light round-trip propagation delay between these two end systems, RTT, is approximately 30 milliseconds.
- Suppose that they are connected by a channel with a transmission rate, R, of 1 Gbps (10^9 bits per second). With a packet size, L, of 1,000 bytes (8,000 bits) per packet, including both header fields and data, the time needed to actually transmit the packet into the 1 Gbps link is

$$d_{trans} = \frac{L}{R} = \frac{8000 \text{ bits/packet}}{10^9 \text{ bits/sec}} = 8 \text{ microseconds}$$

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Stop-and-wait protocol



James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Stop-and-wait protocol

- The packet then makes its 15-msec cross-country journey, with the last bit of the packet emerging at the receiver at t = RTT/2 + L/R = 15.008 msec.
- The receiver can send an ACK as soon as the last bit of a data packet is received, the ACK emerges back at the sender at t = RTT + L/R = 30.008 msec. (Ignore ACK transmission time)
- At this point, the sender can now transmit the next message. Thus, in 30.008 msec, the sender was sending for only 0.008 msec.
- If we define the utilization of the sender (or the channel) as the fraction of time the sender is actually busy sending bits into the channel, the stop-and-wait protocol has a sender utilization, U_{sender}, of

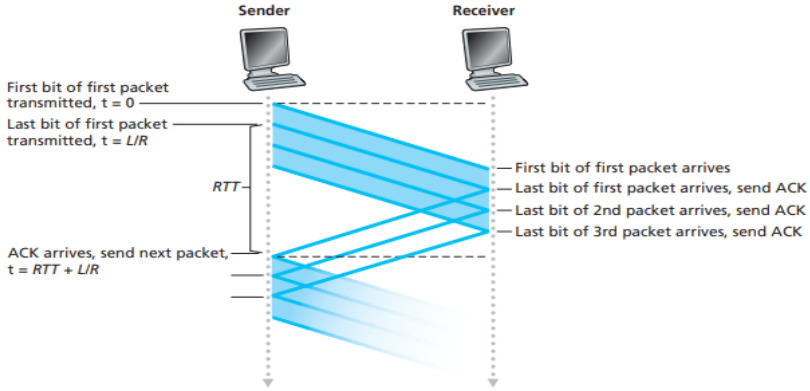
$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

That is, the sender was busy only 2.7 hundredths of one percent of the time!

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

- The solution to this particular performance problem is simple: Rather than operate in a stop-and-wait manner, the sender is allowed to send multiple packets without waiting for acknowledgment.



b. Pipelined operation

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

- The range of sequence numbers must be increased, since each in-transit packet (not counting retransmissions) must have a unique sequence number and there may be multiple, in-transit, unacknowledged packets.
- The sender and receiver sides of the protocols may have to buffer more than one packet.
- Two basic approaches toward pipelined error recovery can be identified: Go-Back-N and selective repeat.

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

- In a Go-Back-N (GBN) protocol, the sender is allowed to transmit multiple packets (when available) without waiting for an acknowledgment, but is constrained to have no more than some maximum allowable number, N, of unacknowledged packets in the pipeline.

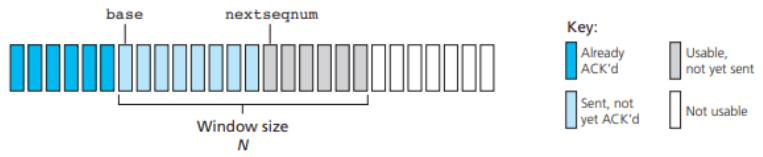


Figure 2.27 ♦ Sender's view of sequence numbers in Go-Back-N

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

- The range of permissible sequence numbers for transmitted but not yet acknowledged packets can be viewed as a window of size N over the range of sequence numbers.
- As the protocol operates, this window slides forward over the sequence number space.
- For this reason, N is often referred to as the window size and the GBN protocol itself as a sliding-window protocol.

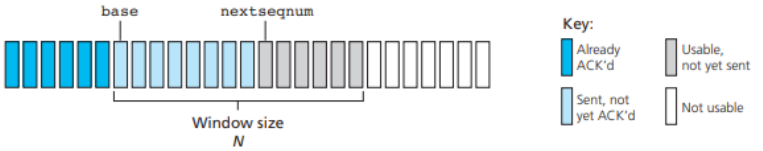


Figure 2.27 ♦ Sender's view of sequence numbers in Go-Back-N

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

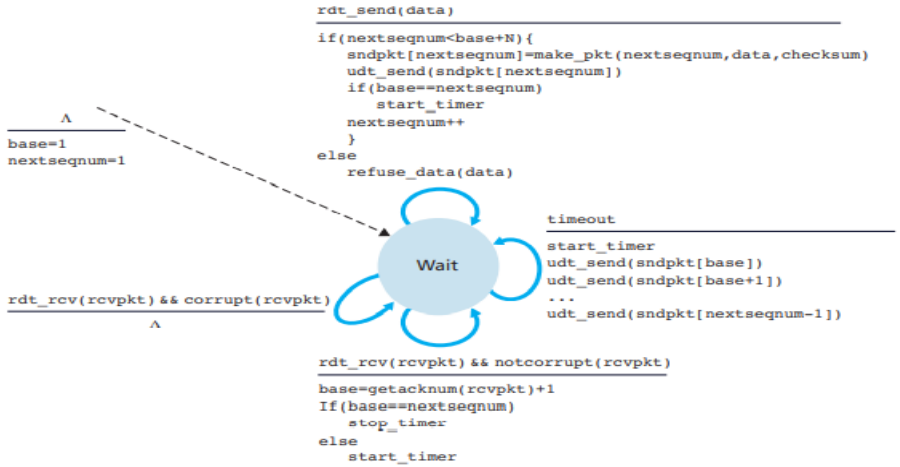


Figure 2.28 ♦ Extended FSM description of GBN sender

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

The GBN sender must respond to three types of events:

1. Invocation from above:

- When rdt_send() is called from above, the sender first checks to see if the window is full, that is, whether there are N outstanding, unacknowledged packets.
- If the window is not full, a packet is created and sent, and variables are appropriately updated.
- If the window is full, the sender simply returns the data back to the upper layer, an implicit indication that the window is full. The upper layer would presumably then have to try again later.
- In a real implementation, the sender would more likely have either buffered (but not immediately sent) this data, or would have a synchronization mechanism (for example, a semaphore or a flag) that would allow the upper layer to call rdt_send() only when the window is not full.

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

The GBN sender must respond to three types of events:

2. Receipt of an ACK:

- In our GBN protocol, an acknowledgment for a packet with sequence number n will be taken to be a cumulative acknowledgment, indicating that all packets with a sequence number up to and including n have been correctly received at the receiver.

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

The GBN sender must respond to three types of events:

3. A timeout event:

- The protocol's name, "Go-Back-N," is derived from the sender's behavior in the presence of lost or overly delayed packets.
- As in the stop-and-wait protocol, a timer will again be used to recover from lost data or acknowledgment packets. If a timeout occurs, the sender resends all packets that have been previously sent but that have not yet been acknowledged.
- Our sender in Figure 2.28 uses only a single timer, which can be thought of as a timer for the oldest transmitted but not yet acknowledged packet.
- If an ACK is received but there are still additional transmitted but not yet acknowledged packets, the timer is restarted.
- If there are no outstanding, unacknowledged packets, the timer is stopped.

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

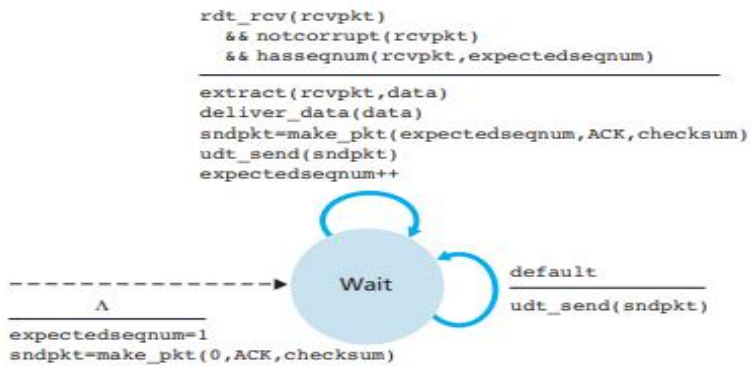


Figure 2.29 ♦ Extended FSM description of GBN receiver

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

The receiver's actions in GBN are also simple.

- If a packet with sequence number n is received correctly and is in order (that is, the data last delivered to the upper layer came from a packet with sequence number n – 1), the receiver sends an ACK for packet n and delivers the data portion of the packet to the upper layer.
- In all other cases, the receiver discards the packet and resends an ACK for the most recently received in-order packet.
- Since packets are delivered one at a time to the upper layer, if packet k has been received and delivered, then all packets with a sequence number lower than k have also been delivered.
- Thus, the use of cumulative acknowledgments is a natural choice for GBN.

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

The receiver's actions in GBN are also simple.

- In our GBN protocol, the receiver discards out-of-order packets.
- The advantage of this approach is the simplicity of receiver buffering—the receiver need not buffer any out-of-order packets.
- Thus, while the sender must maintain the upper and lower bounds of its window and the position of *nextseqnum* within this window, the only piece of information the receiver need maintain is the sequence number of the next in-order packet. This value is held in the variable *expectedseqnum*, shown in the receiver FSM in Figure 2.29.
- Of course, the disadvantage of throwing away a correctly received packet is that the subsequent retransmission of that packet might be lost or garbled and thus even more retransmissions would be required.

Principles of Reliable Data Transfer

Pipelined Reliable Data Transfer Protocols Protocol rdt3.0

Go-Back-N (GBN)

- Figure 2.30 shows the operation of the GBN protocol for the case of a window size of four packets.

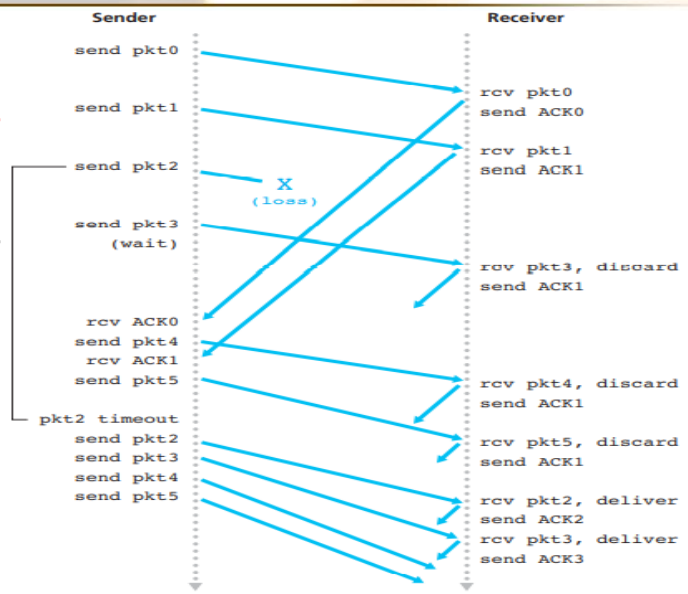
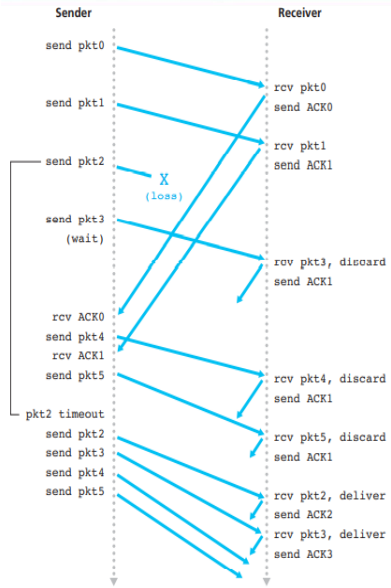


Figure 2.30 ♦ Go-Back-N in operation

Principles of Reliable Data Transfer

Go-Back-N (GBN)

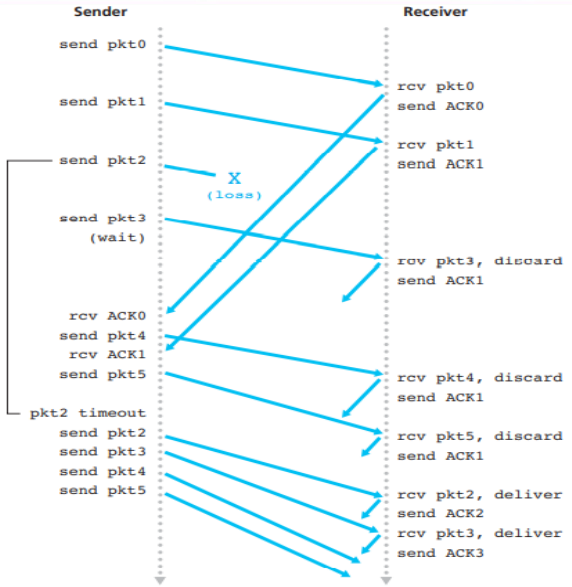


- Because of the window size limitation, the sender sends packets 0 through 3 but then must wait for one or more of these packets to be acknowledged before proceeding.
- As each successive ACK (for example, ACK0 and ACK1) is received, the window slides forward and the sender can transmit one new packet (pkt4 and pkt5, respectively).
- On the receiver side, packet 2 is lost and thus packets 3, 4, and 5 are found to be out of order and are discarded.

Figure 2.30 ♦ Go-Back-N in operation

Principles of Reliable Data Transfer

Go-Back-N (GBN)



- GBN protocol incorporates almost all of the techniques for the reliable data transfer components of TCP.
- These techniques include the use of sequence numbers, cumulative acknowledgments, checksums, and a timeout/retransmit operation

Figure 2.30 ♦ Go-Back-N in operation

Principles of Reliable Data Transfer

Selective Repeat (SR)

- As the name suggests, selective-repeat protocols avoid unnecessary retransmissions by having the sender retransmit only those packets that it suspects were received in error (that is, were lost or corrupted) at the receiver.
- This individual retransmission will require that the receiver individually acknowledge correctly received packets.

Principles of Reliable Data Transfer

Selective Repeat (SR)

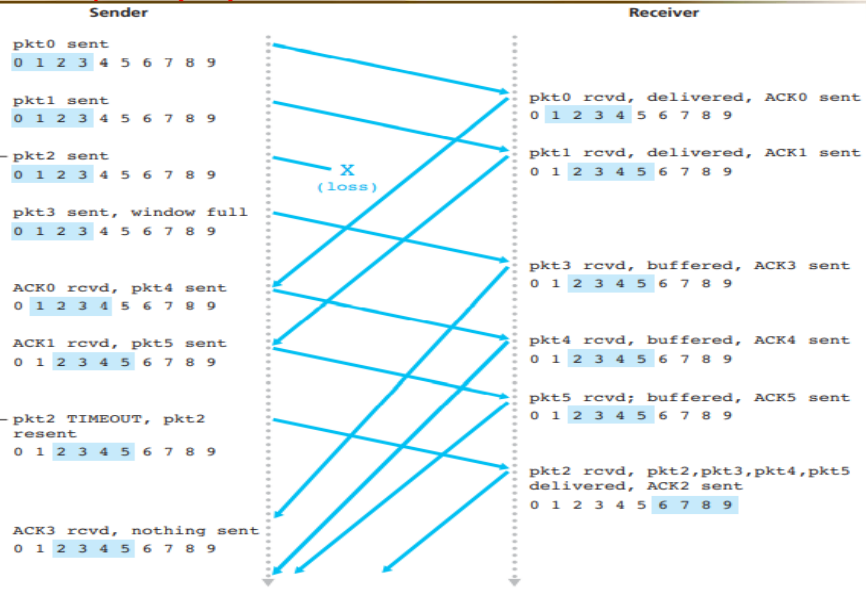


Figure 2.31 ♦ SR operation

Principles of Reliable Data Transfer

Selective Repeat (SR)

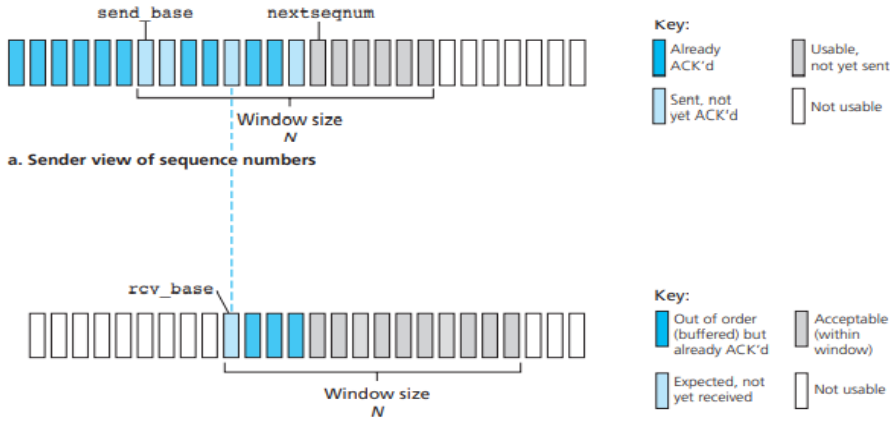


Figure 2.32 ♦ Selective-repeat (SR) sender and receiver views of sequence-number space

James F Kurose and Keith W Ross, "Computer Networking: A Top - Down Approach", Pearson Education; 6 th Edition (2017)