Peer-to-Peer Protocols

The HTTP protocol makes use of the services provided by the TCP protocol to transfer messages in an arrangement such as that shown in Figure 5.1, two entities or peer processes carry out a conversation at each layer. The communications between the layer n+1 entities is virtual and in fact is carried out by a service provided by layer n.

We examine the processing that takes place from when layer n+1 requests a transfer a Service Data Unit (SDU) until the SDU is delivered to the destination layer n+1. In particular, we examine how the layer n peer processes construct Protocol Data Units (PDUs) and convey control information through the headers. We show how each peer process maintains a state that dictates what actions are to be performed when certain events occur.

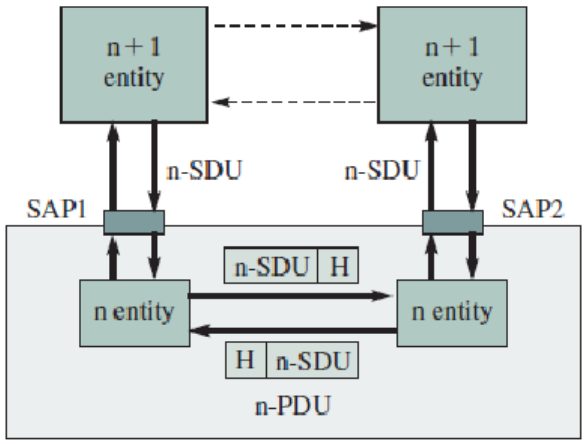


FIGURE 5.1 Layer n entities carry out a protocol to provide service to layer n + 1

PEER-TO-PEER PROTOCOLS AND SERVICE MODELS

A peer-to-peer protocol involves the interaction of two processes or entities through the exchange of messages, called protocol data units (PDUs). The purpose of a protocol is to provide a service to a higher layer. Typically the service involves the sending and receiving of information, possibly with features such as confirmation of delivery and guarantees regarding the order in which the information is delivered. Other features may include guarantees regarding the delay or jitter incurred by the PDUs.

The service provided by a protocol is described by a service model.

Peer-to-peer protocols occur in two basic settings across a single hop in the network or end to end across an entire network. These two settings can lead to different characteristics about whether PDUs arrive in order, about how long it takes for the

PDUs to arrive, or about whether they arrive at all. The design of the corresponding protocols must take these characteristics into account.

Figure 5.2 shows a peer-to-peer protocol that operates across a single hop in a network. Part (a) of the figure uses the lower two layers of the OSI reference

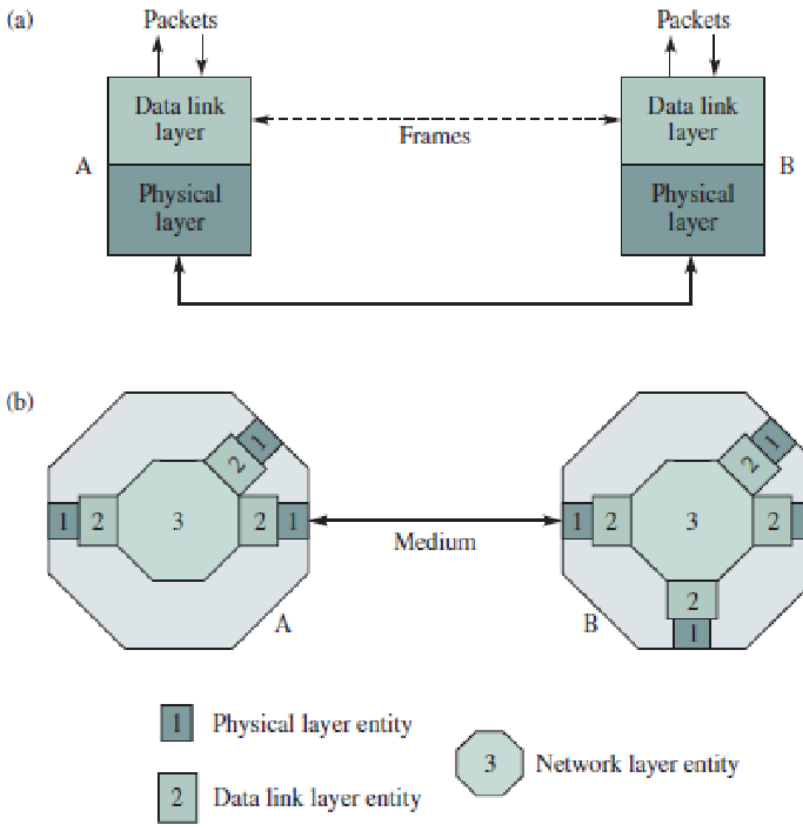


FIGURE 5.2 Peer-to-peer protocol across a single hop

Model to show how the data link layer protocol provides for the transfer of packets across a single link in the network. The data link layer takes packets from the network layer, encapsulates them in frames that it transfers across the link, and delivers them to the network layer at the other end. Figure 5.2b shows a more detailed description of the situation in part (a). Each octagon represents a switch in a packet-switching network. Each switch has a single network layer entity and several pairs of data link and physical link entities. Packets arrive from neighboring packet switches and are passed by the data link layers to the network layer entity. The network layer entity decides how each packet is to be routed and then passes the packet to the data link layer entity that connects the switch to the next packet switch in the route. In Figure 5.2b we focus on the data link that connects packet switch A and packet switch B. For example, the peer processes in this data link could be providing a reliable and sequenced transfer service between the two switches.

Figure 5.3 shows a peer-to-peer protocol that operates end to end across a network. In the figure the transport layer peer processes at the end systems accept messages from their higher layer and transfer these messages by exchanging segments end to end across the network. The exchange of segments is

accomplished by using **network layer services**.

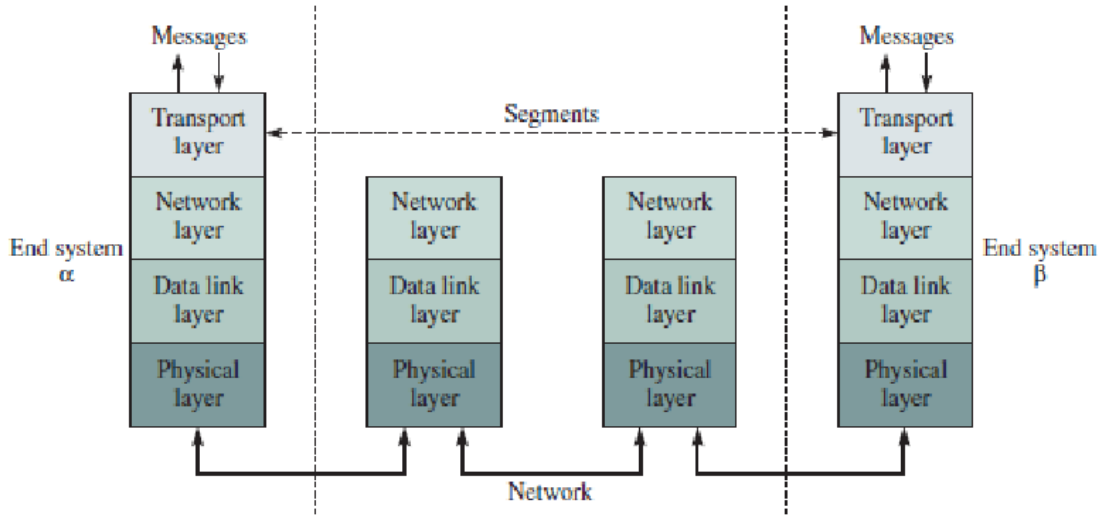


FIGURE 5.3 Peer-to-peer protocols operating end to end across a network—protocol stack view

We use Figure 5.4 to show that the task of the peer-to-peer protocols in this case can be quite complicated.

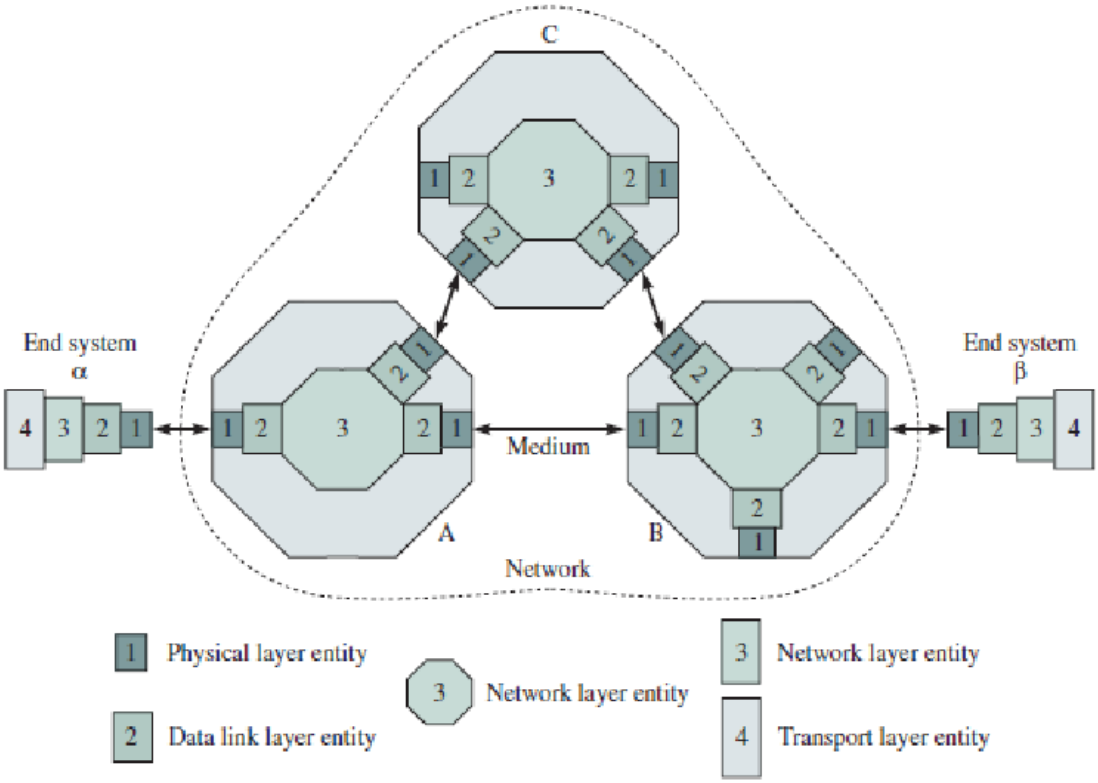


FIGURE 5.4 Peer-to-peer protocols operating end to end across a network—spatial view

Service Models

The service model of a given layer specifies the manner in which information is transmitted. There are two broad categories of service models **connection oriented and connectionless**

Parameter	Connection Less Service	Connection Oriented Service
Definition	It is the Communication System in which there is no need to establish virtual connection between sender and receiver.	It is the communication system in which virtual connection is established between sender and receiver before the communication begins.
Data Acknowledge	No data acknowledge is used, sender can not be sure about the accurate delivery of the message.	Receiver can acknowledge the data send by the sender and can re request the data if any packet fails or gets damaged.
Connection Termination	No Need of Connection termination.	Connection needs to be terminated after completion of communication.
Packet Route	Packets follow different path to reach destination and may reach in any order.	All the frames are sent through same route or path.
Delay	There is a delay in transfer of information, but once the connection is established faster delivery can be achieved.	Due to the absence of connection establishment phase, the transmission is faster.
Example	Postal System	Telephone Call

The service model can also include a **quality-of-service (QoS)** requirement that specifies a level of performance that can be expected in the transfer of information. For example, QoS may specify levels of reliability in terms of probability of errors, probability of loss, or probability of incorrect delivery. QoS may also address transfer delay. For example, a service could guarantee a fixed (nearly) constant transfer delay, or it could guarantee that the delay will not exceed some given maximum value. The variation in the transfer delay is also an important QoS measure for services such as real-time voice and video. The term best-effort service describes a service in which every effort is made to deliver information but without any guarantees.

End-to-End Requirements and Adaptation Functions

When attempting to provide a certain application across a network, the network service frequently does not meet the end-to-end requirements of the application. Figure 5.5 shows how adaptation functions can be introduced between the network and the applications to provide the required level of service.
In this section we consider the following end-to-end requirements and the possible adaptation functions that can be used to meet the level of service required

Services offered by a given layer are

- Arbitrary message size.

- Reliability and sequencing.
- Pacing and flow control.
- Timing.
- Addressing.
- Privacy, integrity, and authentication.

1. A common element of all user-to-user communication systems and indeed all communication networks is the transfer of a message from one point to another. To accommodate a wide range of services, network must be able to handle user messages of arbitrary size. In the case of e-mail, the message is a discrete, well- defined entity. In the case of computer data files, the size of the files can be very large, suggesting that the files be broken into smaller units that are sent as data blocks and reassembled at the far end to recreate the complete file.

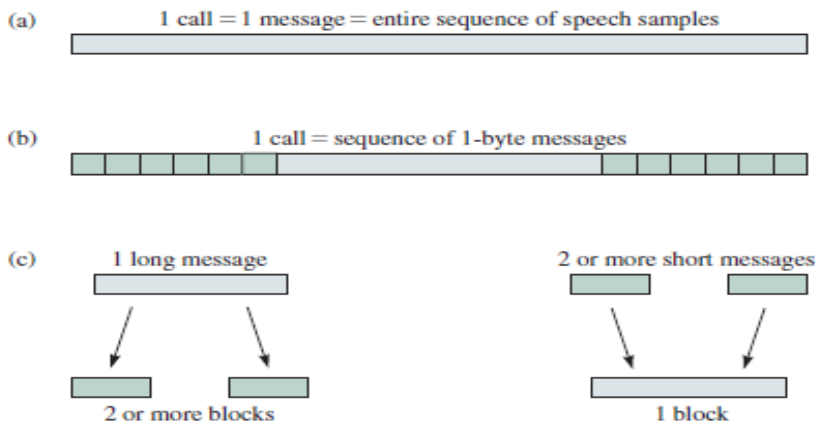


FIGURE 5.6 Message, stream, and sequence of blocks

Figure 5.6a, in telephony one could view the telephone call as generating a single message that consists of the entire sequence of speech samples, but this approach fails to take into account the real-time nature of the application.

Figure 5.6b, which motivates the view of the telephone call as consisting of a sequence of one-byte messages corresponding to each speech sample.

Figure 5.6c, long messages need to be segmented into a **sequence** of blocks. And sequences of small messages can be handled either by converting each small message into a block or by combining one or more small messages into a block.

2. **Reliable** end-to-end communication involves not only the arrival of messages to the destination but also the actual delivery of the messages (e.g., to the listener for voice or to the computer program for a data file). A problem that can arise here is that the receiving system does not have sufficient buffering available to store the arriving message. In this case the message is lost. This problem tends to arise when the transmitter can send information at a rate higher than the receiver can accept it. The *sliding-window protocols* developed in the next section can also be used to provide the receiver with

the means of **pacing or controlling** the rate at which the transmitter sends new messages.

3. For example, in the case of digital speech the receiver must know the appropriate rate at which samples should be entered into the digital-to-analog converter. **Sequence numbering and timestamps** can be used to reconstruct the necessary timing information.
4. In computer communications different processes in a host may simultaneously share a given network connection. In these situations the user messages need to include **addressing** information to allow the host to separate the messages and forward them to the appropriate process.
5. Public networks increasingly have a new type of "impairment" in the form of security threats. Imposters attempt to impersonate legitimate clients or servers. Attempts to deny service to others are made by flooding a server with requests.

End to End Versus Hop by Hop

Introducing the above adaptation functions on an end-to-end basis or on a hop-by-hop basis

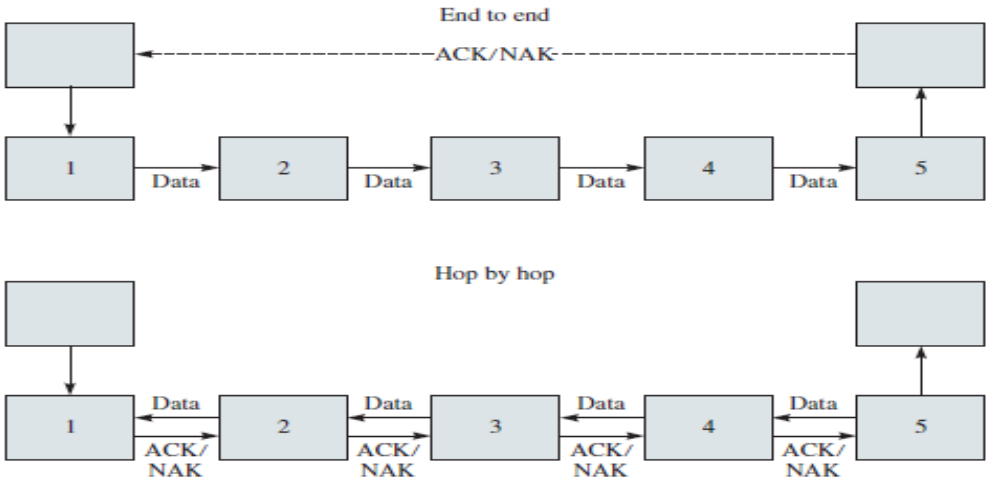


FIGURE 5.7 End-to-end versus hop-by-hop approaches

- To provide reliable communication, error-control procedures can be introduced at every hop, that is, between every pair of adjacent nodes in a path across the network.
- Every node is then required to implement a protocol that checks for errors and requests retransmission using ACK and NAK messages until a block of information is received correctly. Only then is the block forwarded along the next hop to the next node.
- An end-to-end approach, on the other hand, removes the error-recovery responsibility from the intermediate nodes.
- Instead blocks of information are forwarded across the path, and only the end systems are responsible for initiating error recovery.

Automatic Repeat Request

Automatic Repeat Request (ARQ) is a technique used to ensure that a data stream is delivered accurately to the user despite errors that occur during transmission. ARQ forms the basis for peer-to-peer protocols that provide for the reliable transfer of information.

The ARQ mechanism requires the block to contain a header with control information that is essential to proper operation, as shown in Figure 5.8. The transmitter will also append CRC check bits that cover the header and the information bits to enable the receiver to determine whether errors have occurred during transmission. We assume that the design of the CRC ensures that transmission errors can be detected with very high probability, The term frame refer to the binary block that results from the combination of the header, user information, and CRC at the data link layer. We refer to the set of rules that govern the operation of the transmitter and receiver as the ARQ protocol.

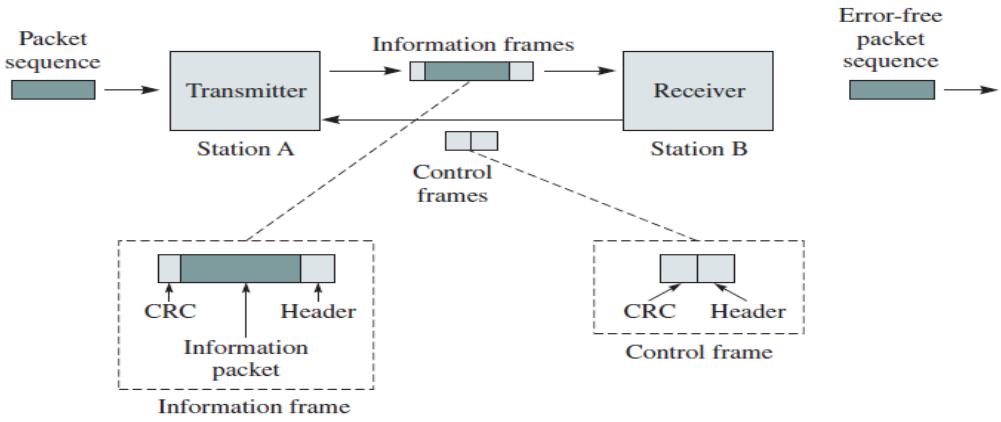


FIGURE 5.8 Basic elements of ARQ

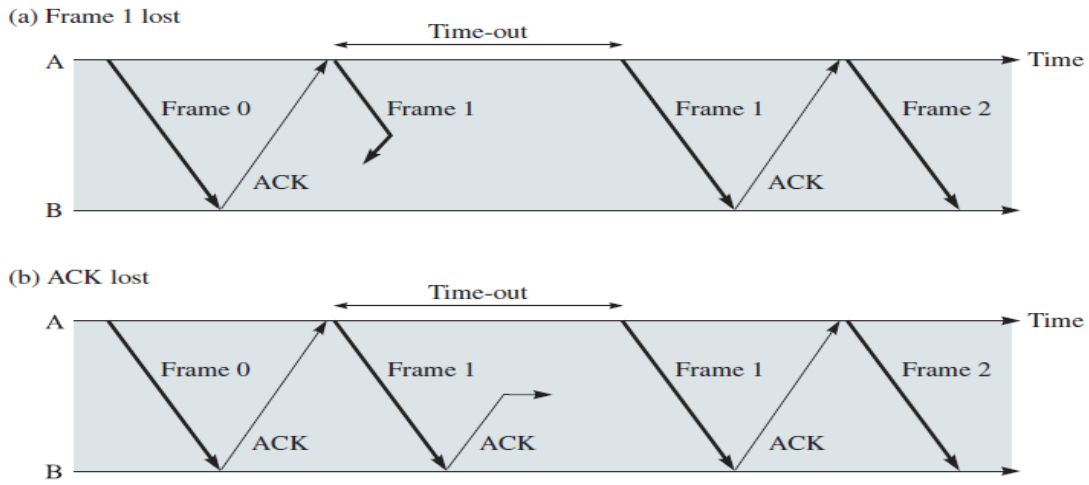
1. Stop-and-Wait ARQ

The first protocol we consider is **Stop-and-Wait ARQ** where the transmitter and receiver work on the delivery of one frame at a time through an alternation of actions. In Figure 5.9a we show how ACKs and time-outs can be used to provide recovery from transmission errors, in this case a lost frame. At the initial point in the figure, stations A and B are working on the transmission of frame 0. Note that each time station A sends an I-frame, it starts an **I-frame timer** that will expire after some time-out period. The time-out period is selected so that it is greater than the time required to receive the corresponding ACK frame. Figure 5.9a shows the following sequence of events„

1. Station A transmits frame 0 and then waits for an ACK frame from the receiver.
2. Frame 0 is transmitted without error, so station B transmits an ACK frame.
3. The ACK from station B is also received without error, so station A knows the

frame 0 has been received correctly.

4. Station A now proceeds to transmit frame 1 and then resets the timer.
5. Frame 1 undergoes errors in transmission. It is possible that station B receives frame 1 and detects the errors through the CRC check; it is also possible that frame 1 was so badly garbled that station B is unaware of the transmission.4 In either case station B does not take any action.
6. The time-out period expires, and frame 1 is retransmitted.

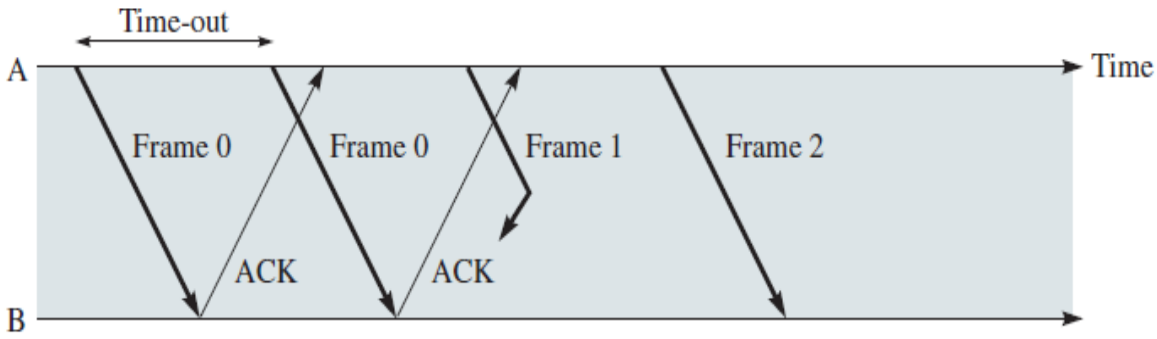


In parts (a) and (b) transmitting station A acts the same way, but part (b) receiving station B accepts frame 1 twice.

FIGURE 5.9 Possible ambiguities when frames are unnumbered

Transmission errors in the reverse channel lead to ambiguities in the Stop-and-Wait protocol that need to be corrected. Figure 5.9b shows the situation that begins as in Figure 5.9a, but where frame 1 is received correctly, and its acknowledgment undergoes errors. After receiving frame 1 station B delivers its contents to the destination. Station A does not receive the acknowledgment for frame 1, so the time-out period expires.

Note that at this point station A cannot distinguish between the sequence of events in parts (a) and (b) of Figure 5.9. Station A proceeds to retransmit the frame. If the frame is received correctly by station B, as shown in the figure, then station B will accept frame 1 as a new frame and redeliver it to the user. Thus we see that the loss of an ACK can result in the delivery of a duplicate packet. The ambiguity can be eliminated by including a sequence number in the header of each I-frame. Station B would then recognize that the second transmission of frame 1 was a duplicate, discard the frame, and resend the ACK for frame 1.



Transmitting station A misinterprets duplicate ACKs

FIGURE 5.10 Possible ambiguities when ACKs are unnumbered

A second type of ambiguity arises if the ACKs do not contain a sequence number. In Figure 5.10 frame 0 is transmitted, but the time-out expires prematurely. Frame 0 is received correctly, and the (unnumbered) ACK is returned. In the meantime station A has resent frame 0. Shortly thereafter, station A receives an ACK and assumes it is for the last frame. Station A then proceeds to send frame 1, which incurs transmission errors. In the meantime the second transmission of frame 0 has been received and acknowledged by station B. When station A receives the second ACK, the station assumes the ACK is for frame 1 and proceeds to transmit frame 2. The mechanism fails because frame 1 is not delivered. This example shows that premature time-outs (or delayed ACKs) combined with loss of I-frames can result in gaps in the delivered packet sequence. This ambiguity is resolved by providing a sequence number in the acknowledgment frames that enables the transmitter to determine which frames have been received.

The sequence numbers cannot be allowed to become arbitrarily large because only a finite number of bits are available in the frame headers. We now show that a one-bit sequence number suffices to remove the above ambiguities in the Stop- and-Wait protocol.

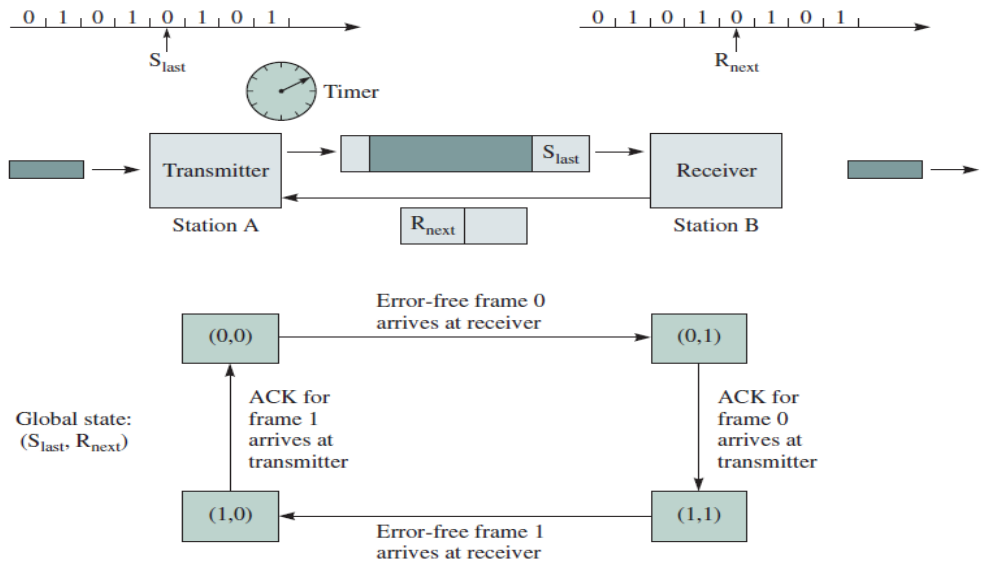


FIGURE 5.11 System state information in Stop-and-Wait ARQ

Figure 5.11 shows the information or "state" that is maintained by the transmitter and receiver. The transmitter must keep track of the sequence number S_{last} of the frame being sent, its associated timer, and the frame itself in case retransmission is required. The receiver keeps track only of the sequence number R_{next} of the next frame it is expecting to receive. Suppose that initially the transmitter and receiver are synchronized in the sense that station A is about to send a frame with S_{last} = 0 and station B is expecting R_{next}=0.

In Figure 5.11 the global state of the system is defined by the pair (S_{last}, R_{next}), so initially the system is in state (0,0).
The system state will not change until station B receives an error-free version of frame 0. That is, station A will continue resending frame 0 as dictated by the time-out mechanism. Eventually station B receives frame 0, station B changes Rnext to 1 and sends an acknowledgment to station A with R_{next} = 1 implicitly acknowledging the receipt of frame 0. At this point the state of the system is (0,1). Any subsequent received frames that have sequence number 0 are recognized as duplicates and discarded by station B, and an acknowledgment with R_{next} = 1 is resent.

Eventually station A receives an acknowledgment with R_{next} = 1 and then begins transmitting the next frame, using sequence number S_{last} = 1. The system is now in state (1,1). The transmitter and receiver are again synchronized, and they now proceed to work together on the transfer of frame 1. Therefore, a protocol that implements this mechanism that follows the well- defined sequence of states shown in Figure 5.11 can ensure the correct and orderly delivery of frames to the destination.

Checkpointing can be used for error recovery using a short control frame called Enquiry Frame (ENQ). When a timeout expires, if the frame that needs to transmitted is very long, then transmitter sends ENQ.

And receiver sends last ACK for every ENQ.

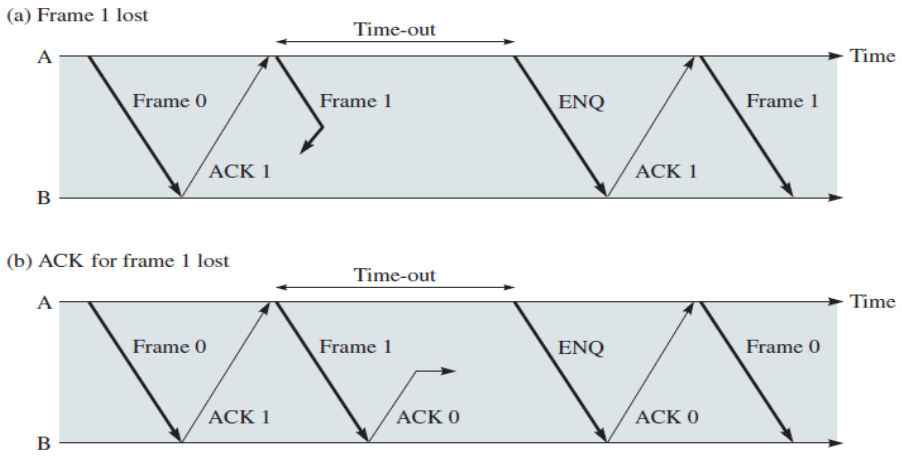


FIGURE 5.12 Stop-and-Wait ARQ enquiry frame

2. Go-Back-N ARQ

The inefficiency of Stop-and-Wait ARQ can be overcome by allowing the transmitter to continue sending enough frames so that the channel is kept busy while the transmitter waits for acknowledgments.
The idea of the Basis Go-Bask-N ARQ is as follows, Consider the transfer of a reference frame, say, frame 0. After frame 0 is sent, the transmitter sends W_S - 1 additional frames into the channel to keep the channel busy.

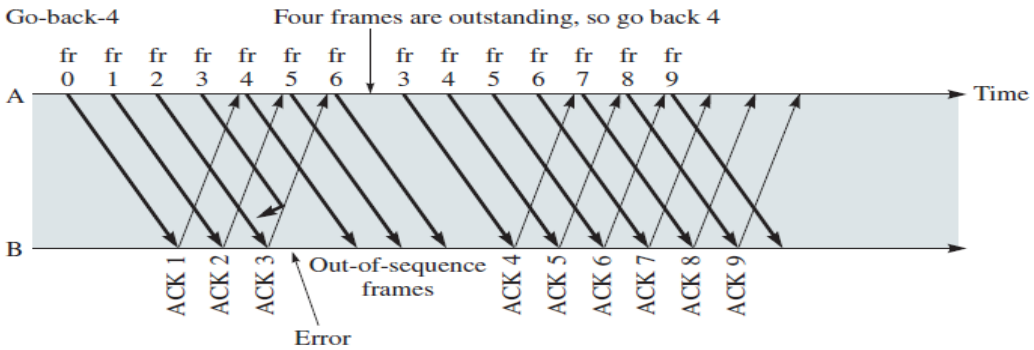


FIGURE 5.13 Basic Go-Back-N ARQ

Go-Back-N ARQ gets its name from the action that is taken when an error occurs. As shown in Figure 5.13, after frame 3 undergoes transmission errors, the receiver ignores frame 3 and all subsequent frames.
Eventually the transmitter reaches the maximum number of outstanding frames. It is then forced to "go back N" frames, where N = W_S, and begin retransmitting all packets from 3 onwards.

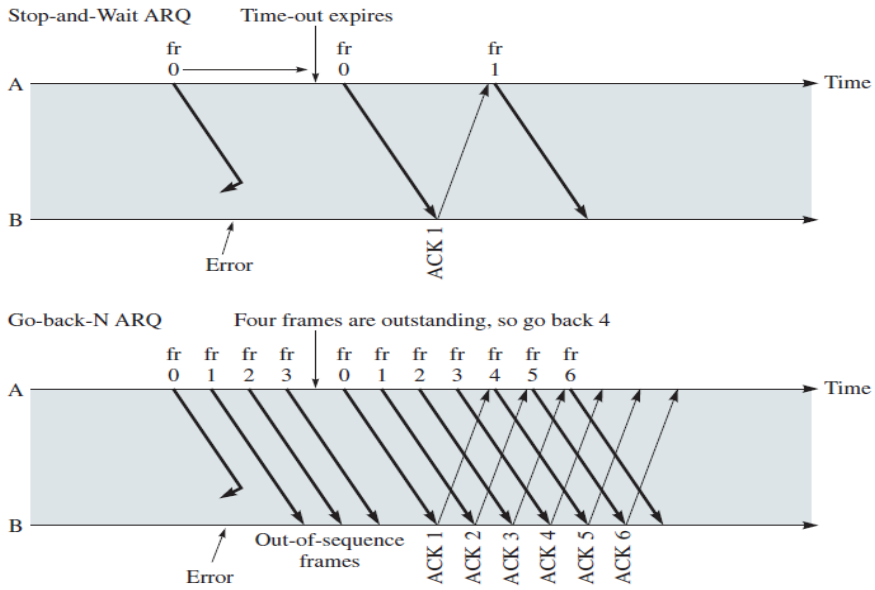


FIGURE 5.14 Relationship of Stop-and-Wait ARQ and Go-Back-N ARQ in terms of error recovery.

- The Go-Back-N ARQ as stated above depends on the transmitter exhausting its maximum number of outstanding frames to trigger the retransmission of a frame.
- Thus this protocol works correctly as long as the transmitter has an unlimited supply of packets that need to be transmitted.
- In situations where packets arrive at irregular intervals, there may not be W_S -1 subsequent transmissions.
- In this case retransmissions are not triggered, since the window is not exhausted.
- This problem is easily resolved by modifying Go-Back-N ARQ such that a timer is associated with each transmitted frame.
- The Go-Back-N protocol is an example of a sliding-window protocol.

Figure 5.15 shows how the resulting Go-Back-N ARQ protocol operates. The transmitter must now maintain a list of the frames it is processing, where S_{last} is the number of the last transmitted frame that remains unacknowledged and S_{recent} is the number of the most recently transmitted frame. The transmitter must also maintain a timer for each transmitted frame and must also buffer all frames that have been transmitted but have not yet been acknowledged. At any point in time the transmitter has a **send-window** of available sequence numbers. The lower end of the window is given by Slast, and the upper limit of the transmitter window is S_{last} + W_S 1. If S_{recent} reaches the upper limit of the window, the transmitter is not allowed to transmit further new frames until the send window slides forward with the receipt of a new acknowledgment.

The Go-Back-N protocol is an example of a **sliding-window protocol**. The receiver maintains a **receive window** of size 1 that consists of the next frame Rnext it expects to receive. If an arriving frame passes the CRC check and has the correct sequence number that is, Rnext, then it is accepted and Rnext is incremented. We

say that the receive window slides forward. The receiver then sends an acknowledgment containing the incremented sequence number Rnext, which implicitly acknowledges receipt of all frames prior to Rnext. Note how we are making use of the assumption that the channel is wirelike„ When the transmitter receives an ACK with a given value Rnext, it can assume that all prior frames have been received correctly, even if it has not received ACKs for those frames, either because they were lost or because the receiver chose not to send them. Upon receiving an ACK with a given value Rnext, the transmitter updates its value of Slast to Rnext and in so doing the send window slides forward.

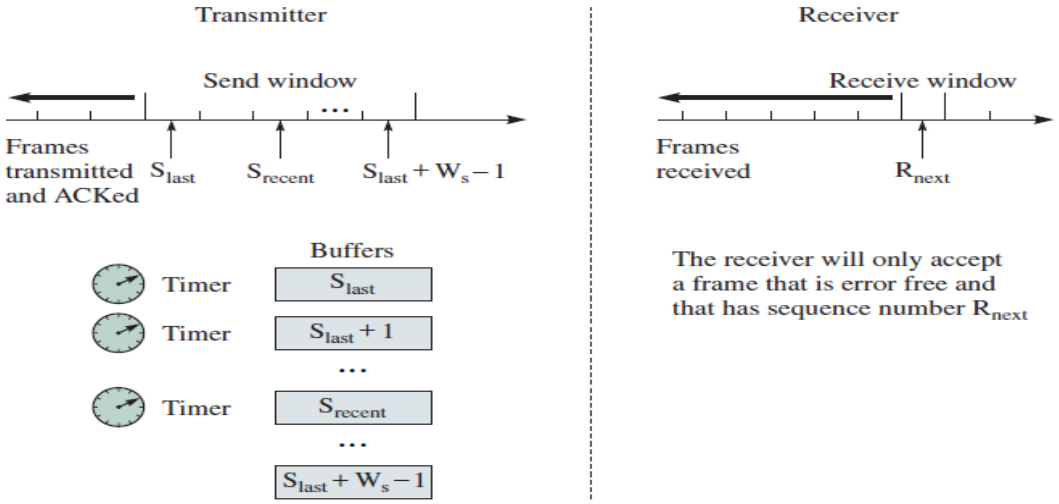


FIGURE 5.15 Go-Back-N ARQ

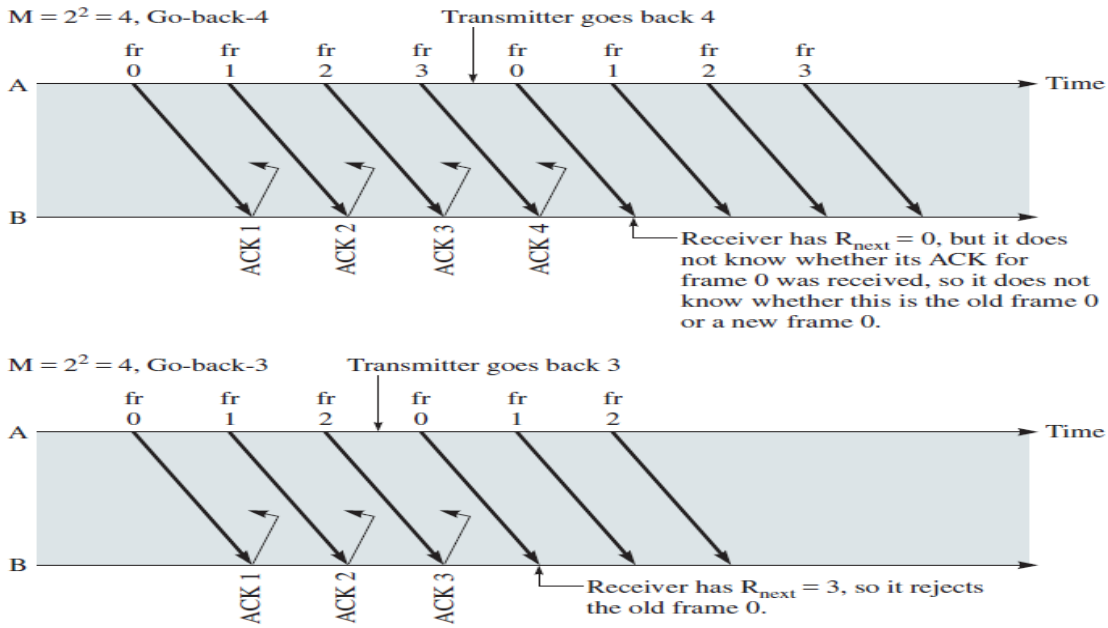


FIGURE 5.16 The window size should be less than 2^m

In figure 5.17 the performance of Go-Back-N ARQ can be improved by having the receiver send a NAK message immediately after the first out-of-sequence frame is received.
The NAK informs the transmitter that the receiver is now discarding frames that followed Rnext and instructs the transmitter to go back and retransmit Rnext and all subsequent packets.

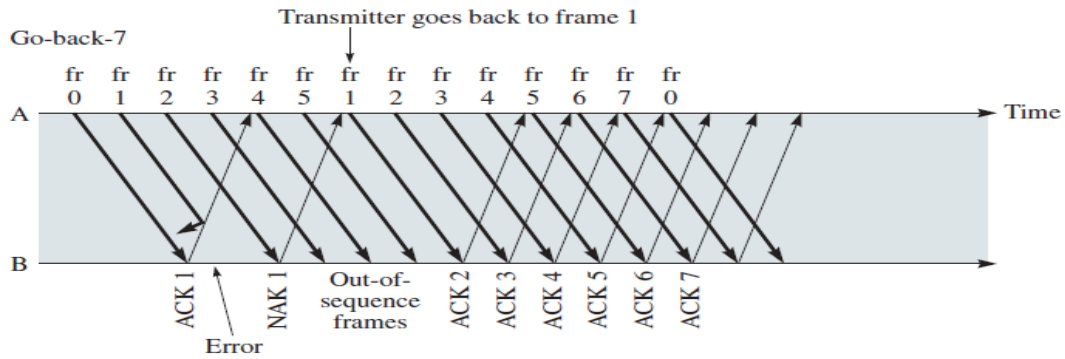


FIGURE 5.17 NAK error recovery

3. Selective Repeat ARQ

In channels that have high error rates, the Go-Back-N ARQ protocol is inefficient because of the need to retransmit the frame in error and all the subsequent frames.

A more efficient ARQ protocol can be obtained by adding two new features,

- First, the receive window is made larger than one frame so that the receiver can accept frames that are out of order but error free;

- Second, the retransmission mechanism is modified so that only individual frames are retransmitted. We refer to this protocol as Selective Repeat ARQ.

Figure 5.20 shows that the send window at the transmitter is unchanged but that the receive window now consists of a range of frame numbers spanning from R_{next} to R_{next} + Wr-1, where W_R is the maximum number of frames that the receiver is willing to accept at a given time. As before, the basic objective of the protocol is to advance the values of R_{next} and S_{last} through the delivery of the oldest outstanding frame. Thus ACH frames carry R_{next}, the oldest frame that has not yet been received. The receive window is advanced with the receipt of an error-free frame with sequence number R_{next}. Unlike the case of Go-Back-N ARQ, the receive window may be advanced by several frames. This step occurs when one or more frames that follow R_{next} have already been received correctly and are buffered in the receiver. R_{next} and the following consecutive packets are delivered to the destination at this point.

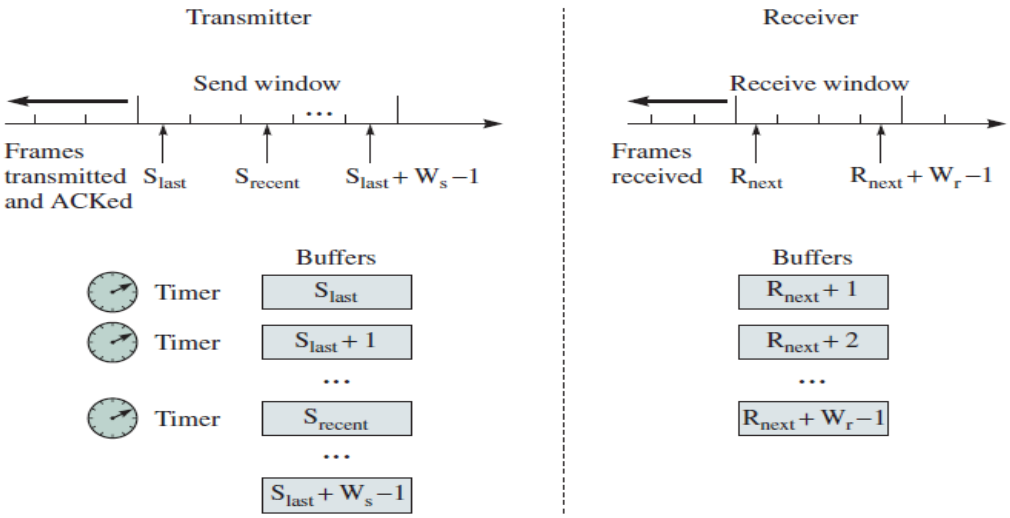


FIGURE 5.20 Selective Repeat ARQ

Now consider the retransmission mechanism in Selective Repeat ARQ. The handling of timers at the transmitter is done as follows. When the timer expires, only the corresponding frame is retransmitted. There is no longer a clear correspondence between the age of the timers and the sequence numbers. This situation results in a considerable increase in complexity. Whenever an out-of- sequence frame is observed at the receiver, a NAK frame is sent with sequence number R_{next}. When the transmitter receives such a NAK frame, it retransmits the specific frame, namely, R_{next}.

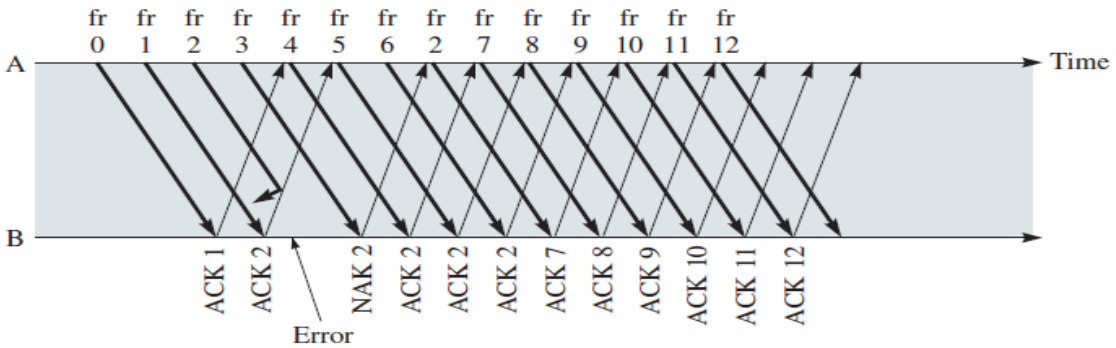


FIGURE 5.21 Error recovery in Selective Repeat ARQ

For example, in Figure 5.21 when the frame with sequence number 2 finally arrives correctly at the receiver, frames 3, 4, 5, and 6 have already been received correctly. Consequently, the receipt of frame 2 results in a sliding of the window forward by five frames.

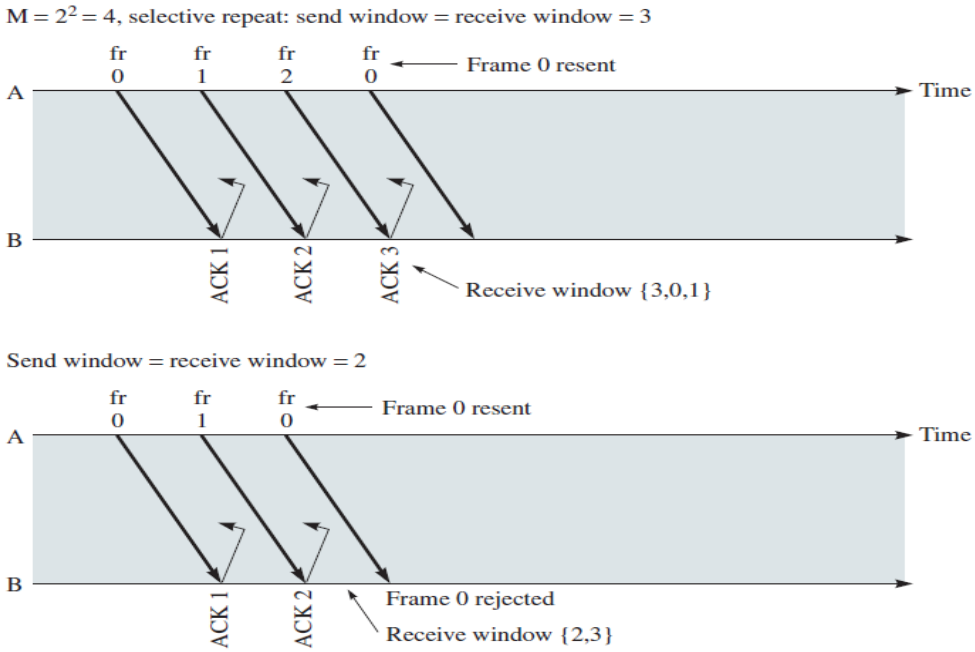


FIGURE 5.22 Maximum window size in Selective Repeat ARQ

Consider now the question of the maximum send window size that is allowed for a given sequence numbering 2^m. In Figure 5.22 we show an example in which the sequence numbering is 2^2=4 and in which the send windows and receive windows are of size 3. Initially station A transmits frames 0, 1, and 2.

All three frames arrive correctly at station B, and so the receive window is advanced to
{3, 0, 1}. Unfortunately all three acknowledgments are lost, so when the timer for frame 0 expires, frame 0 is retransmitted. The inadequacy of the window size now becomes evident. Upon receiving frame 0, station B cannot determine whether it is the old frame 0 or the new frame 0. So clearly, send and receive windows of size 2^m-1 are too large.

Transmission Efficiency of ARQ Protocols

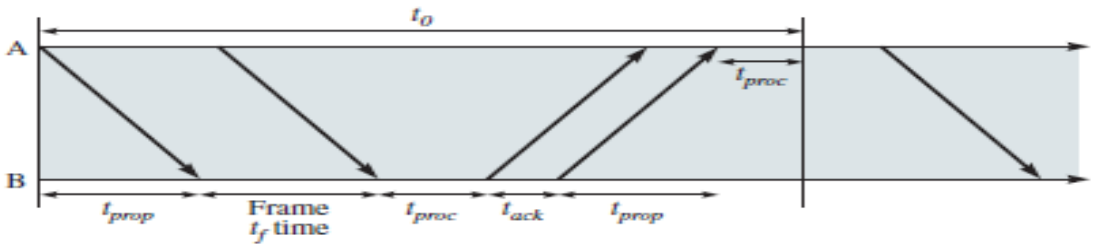


FIGURE 5.23 Delay components in Stop-and-Wait ARQ

$$t_0 = 2t_{prop} + 2t_{proc} + t_f + t_{ack} = 2t_{prop} + 2t_{proc} + \frac{n_f}{R} + \frac{n_a}{R}$$

where n_f is the number of bits in the information frame, n_a is the number of bits in the acknowledgment frame, and R is the bit rate of the transmission channel.

The effective information transmission rate of the protocol in the absence of errors is then given by

$$R_{eff}^0 = \frac{\text{number of information bits delivered to destination}}{\text{total time required to deliver the information bits}} = \frac{n_f - n_o}{t_0}$$

where n_o is the number of overhead bits in a frame and is given by the total number of bits in the header and the number of CRC check bits. The transmission efficiency of Stop-and-Wait ARQ is given by the ratio R_{eff}^0 to R:

$$\eta_0 = \frac{\frac{n_f - n_o}{t_0}}{R} = \frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}}$$

Find the efficiencies of Stop-and-Wait ARQ in a number of scenarios. First assume a frame size of 1024 bytes (8192 bits) and $n_o = n_a = 8$ bytes.

Consider three values of reaction time $t_{prop} + t_{proc}$: 5 ms, 50 ms, and 500 ms. Because the speed of light is $3 * 10^8$ meters/second, these delays correspond to distances of about 1500 km, 15,000 km, and 150,000 km links that could correspond roughly to moderate distance, intercontinental, and satellite links, respectively.

Consider four values of bit rate 30,000 bps for an ordinary telephone modem, 1.5 Mbps for a high-speed telephone line, 45 Mbps for high-speed access lines, and 2.4 Gbps for a high-speed backbone line.

		R (bps)			
		30 kbps	1.5 Mbps	45 Mbps	2.4 Gbps
(a) $n_{frame} = 8192$ $n_{overhead} = 64$, $n_{ack} = 64$					
$t_{prop} + t_{proc}$:	**0.005**	0.95	0.35	1.77E−0.2	3.39E−04
	0.050	0.72	5.14E−02	1.80E−03	3.39E−05
	0.500	0.21	5.39E−03	1.81E−04	3.39E−06
(b) $n_{frame} = 524{,}288$					
$t_{prop} + t_{proc}$:	**0.005**	0.99	0.97	0.53	2.14E−02
	0.050	0.99	0.77	1.04E−01	2.18E−03
	0.500	0.21	0.26	1.15E−02	2.18E−04

TABLE 5.1 Efficiency of Stop-and-Wait ARQ in the absence of errors

ARQ technique	Efficiency	Comments
Stop-and-Wait	$\eta = \frac{\frac{n_f - n_0}{E[t_{total}]}}{R} = (1 - P_f)\frac{1 - \frac{n_0}{n_f}}{1 + \frac{n_a}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}} = (1 - P_f)\eta_0$	Delay-bandwidth product is main factor
Go-Back-N	$\eta = \frac{\frac{n_f - n_0}{E[t_{total}]}}{R} = (1 - P_f)\frac{1 - \frac{n_0}{n_f}}{1 + (W_S - 1)P_f}$	Average wasted time: $(W_S - 1)P_f$
Selective-Reject	$\eta = (1 - P_f)\left(1 - \frac{n_0}{n_f}\right)$	Note impact of delay-bandwidth product through W_S

TABLE 5.2 Summary of performance results

OTHER ADAPTATION FUNCTIONS

In this section we show how the various elements of the ARQ protocols can be used to provide other adaptation functions.

- In particular, we show how adaptation functions can provide flow control,
- Provide synchronization and timing information.
- TCP protocol for providing reliable stream service end-to-end across a network.

1. Sliding-Window Flow Control

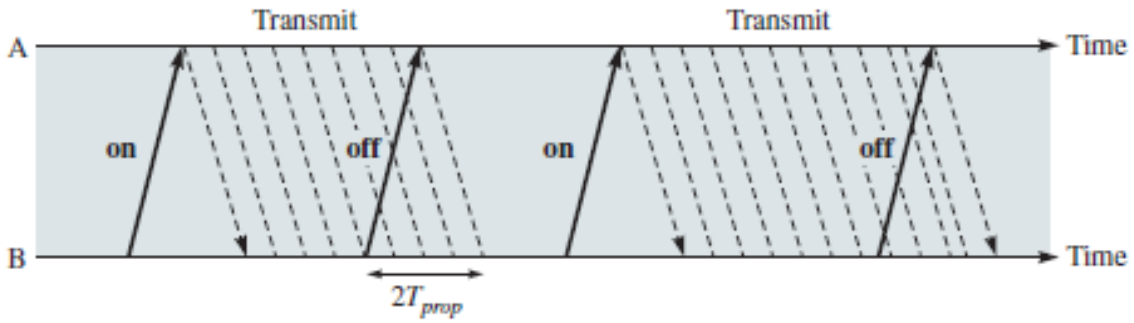


FIGURE 5.26 ON-OFF flow control

2. Timing Recovery for Synchronous Service

Ex: digital speech, video signals

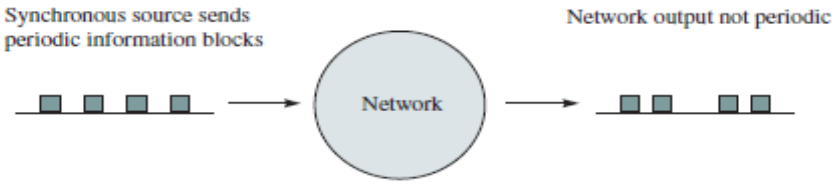


FIGURE 5.28 Timing recovery

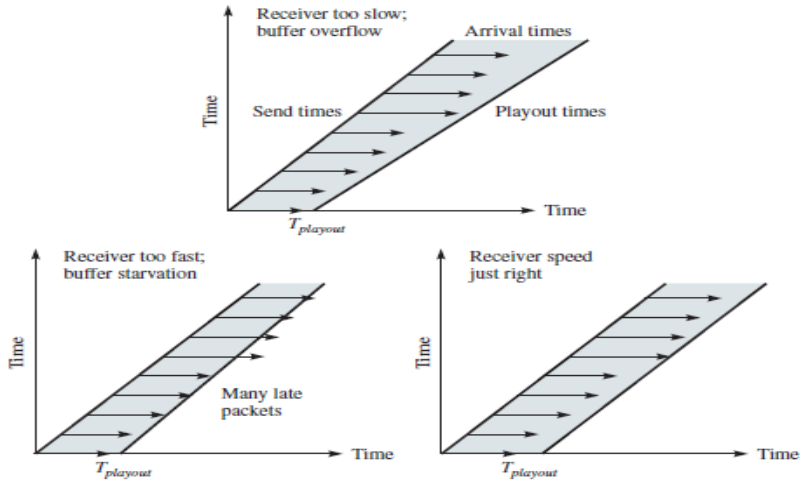


FIGURE 5.29 Clock rate differences and their effect on transmission

- The figure shows the times at which the information blocks were produced, and the arrows indicate the total delay that was encountered in the network.
- When the receiver clock is slow relative to the transmitter clock, the receiver will play its samples at a rate slower than they were produced.
- Consequently, over a period of time the receiver buffer will grow, eventually leading to loss of information due to buffer overflow.
- On the other hand, when the receiver is too fast, the receiver buffer will gradually empty and the playout procedure will be interrupted due to lack of available samples.
- These examples show why the adaptation function must also include a clock recovery procedure that attempts to also synchronize the receiver clock to the transmitter clock.

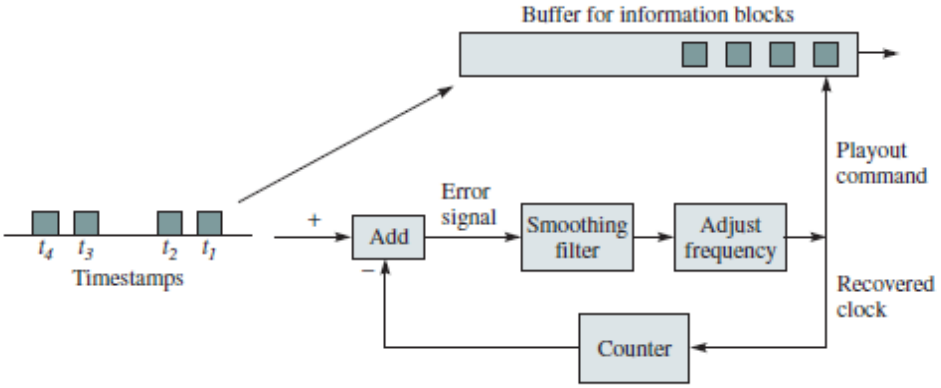


FIGURE 5.30 Adaptive clock recovery

- The sequence of timestamp values are used to perform clock recovery.
- The timestamp in the arriving blocks of information were generated by sampling a counter that is driven by the transmitter clock.
- The receiver system has a counter that attempts to synchronize to the transmitter clock.
- The sequence of timestamp values is compared to the local counter values to generate an error signal that is indicative of the difference between the transmitter and receiver clocks.
- This error signal is filtered and used to adjust the frequency of the local clock. If the difference signal indicates that the local clock is slow, then the frequency of the local clock is increased.
- If the difference indicates that the local clock is slow, then the frequency of the local clock is reduced. The recovered clock is then used to control the playout from the buffer.

Another very effective clock recovery procedure can be used when the transmitter and receiver are connected to a network in which all the elements are synchronized to a common clock

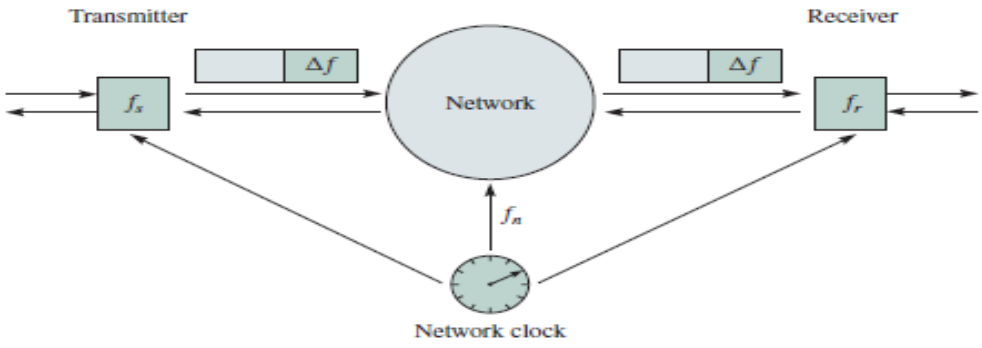


FIGURE 5.31 Clock recovery with synchronous network

The receiver then controls the playout using the frequency f_r, which is given by

$$f_r = f_n - \Delta f$$

3. Reliable Stream Service

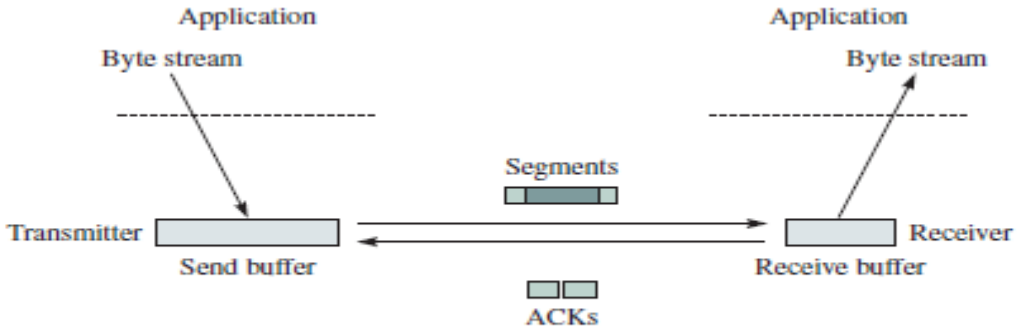


FIGURE 5.32 TCP preview

Media Access Control (MAC)

When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link. The problem of controlling the access to the medium is similar to the rules of speaking in an assembly. The procedures guarantee that the right to speak is upheld and ensure that two people do not speak at the same time, do not interrupt each other, do not monopolize the discussion, and so on. Many protocols have been devised to handle access to a shared link. All of these protocols belong to a sublayer in the data-link layer called **media access control (MAC).**

Multiple access communications situation in which a number of user stations share a transmission medium.

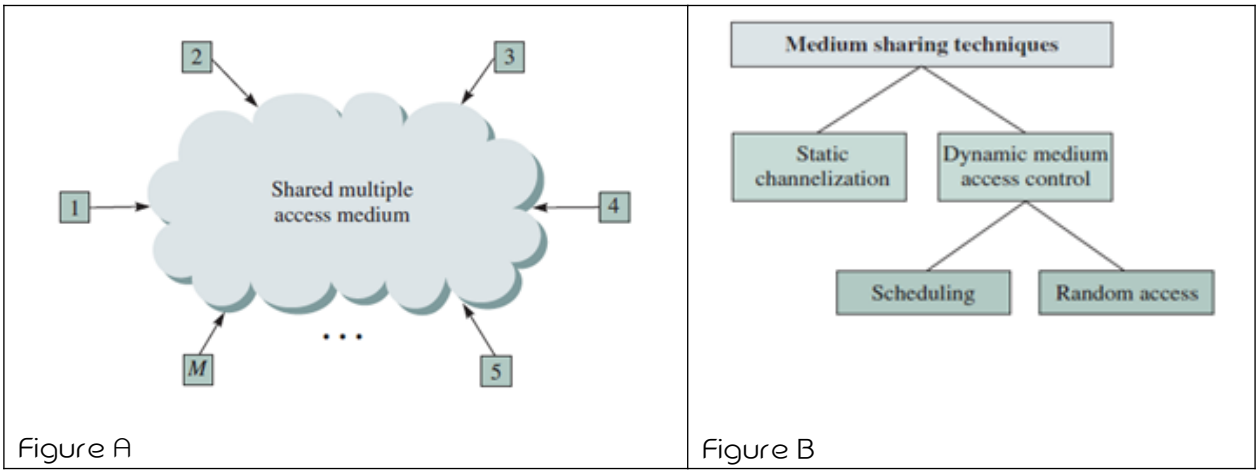


Figure A Figure B

In figure B, the first category involves a static and collision-free sharing of the medium. they involve the partitioning of the medium into separate channels that are then dedicated to particular users.

The second category involves a dynamic sharing of the medium on a per packet basis that is better matched to situations in which the user traffic is bursty.

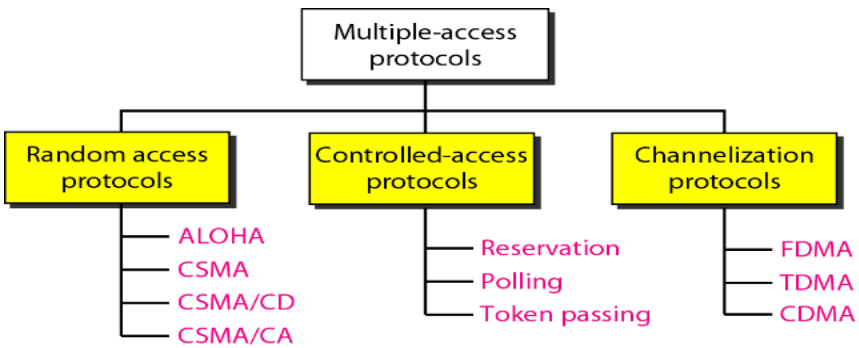


Figure 12.2 Taxonomy of multiple-access protocols

RANDOM ACCESS

In random access methods, no station is superior to another station and none is assigned the control over another. No station permits, or does not permit, another station to send. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send.

Random access is based on two features:

1. There is no scheduled time for a station to transmit.
2. Transmission is random among the stations
3.

To avoid access conflicts (collisions), a station must answer

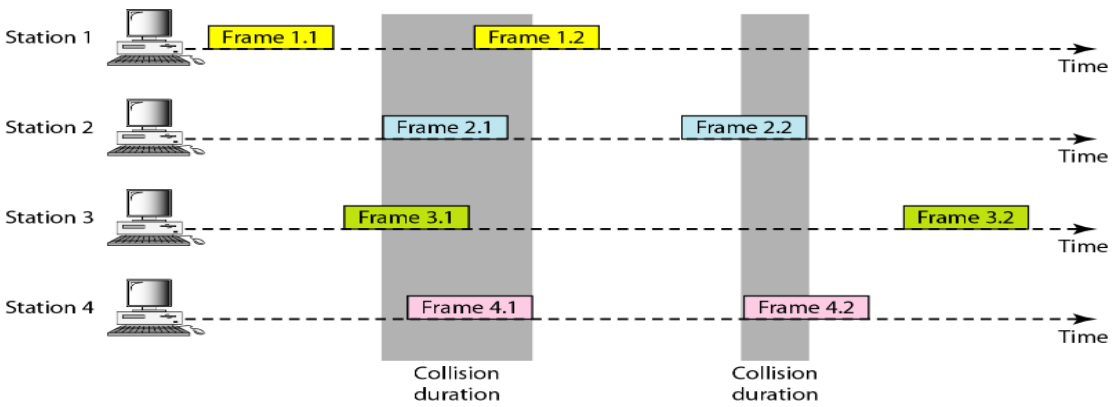
1. When can the station access the medium?
2. What can the station do if the medium is busy ?
3. How can the station determine the success or failure of the transmission?
4. What can the station do if there is an access conflict?

RANDOM ACCESS protocols

- ALOHA
- Carrier Sense Multiple Access
- Carrier Sense Multiple Access with Collision Detection
- Carrier Sense Multiple Access with Collision Avoidance

1. **ALOHA:** the earliest random access method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN, but it can be used on any shared medium (Pure ALOHA and Slotted ALOHA)

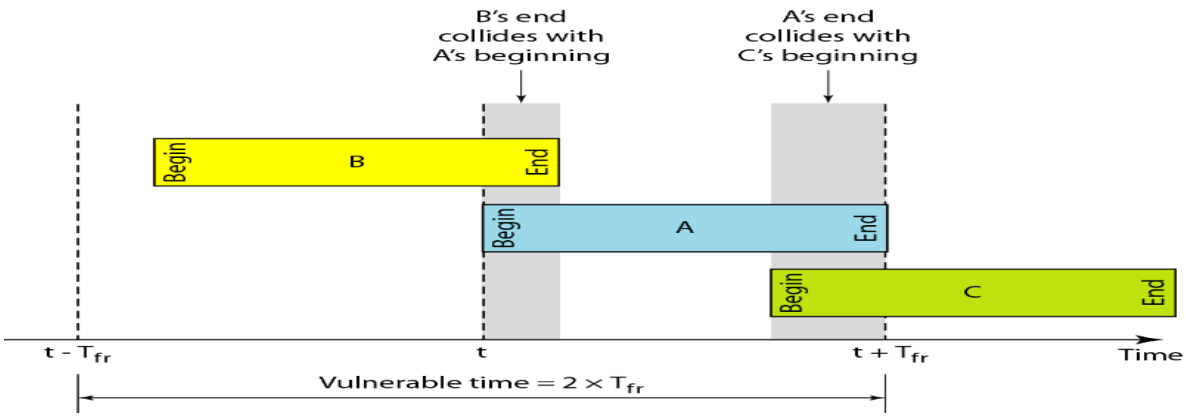
Pure ALOHA : The idea is to send a frame whenever a station has a frame to send.



Vulnerable time

Let us find the vulnerable time, the length of time in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking T_{fr} seconds to send. Figure 12.4 shows the vulnerable time for station B.

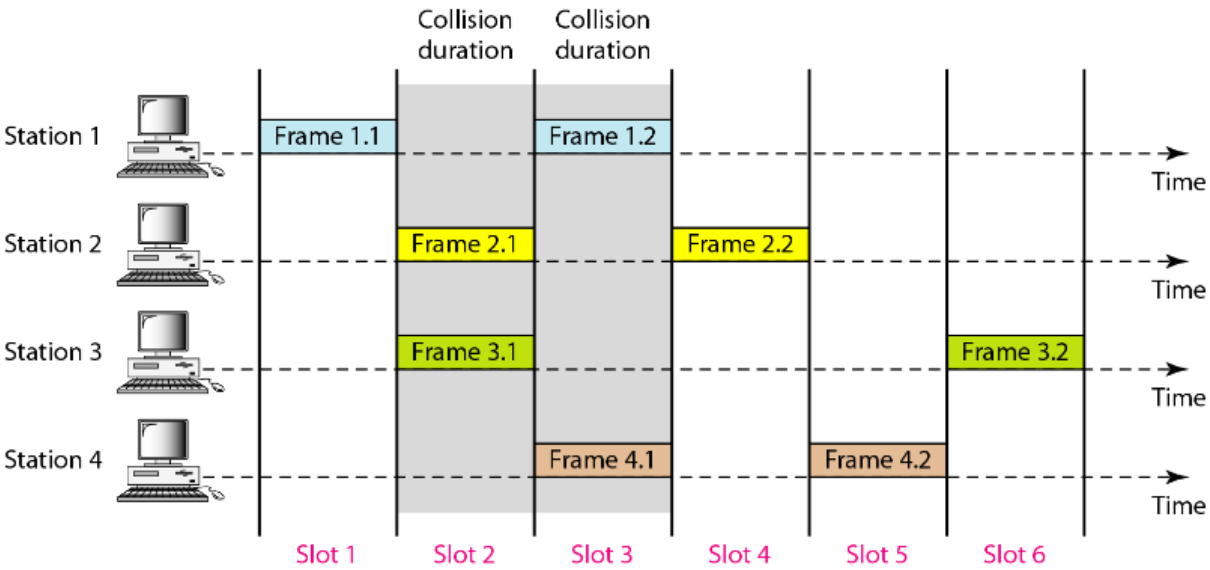
Station B starts to send a frame at time t. Now imagine station A has started to send its frame after t- T_{fr}. This leads to a collision between the frames from station B and station A. On the other hand, suppose that station C starts to send a frame before time t + T_{fr}. Here, there is also a collision between frames from station B and station C.



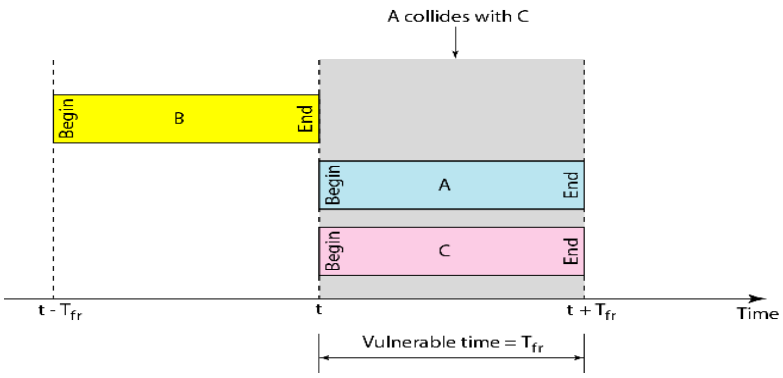
Vulnerable time for pure ALOHA : 2 X T_{fr}

Slotted ALOHA network

Divides time into slots of Tfr and forces the station to send at the beginning of each timeslot.



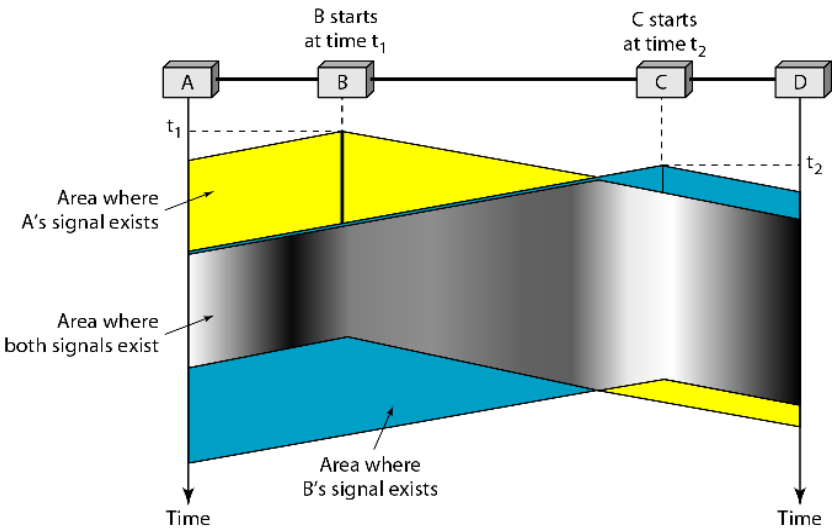
Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has already finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot. However, the vulnerable time is now reduced to one-half, equal to T_{fr}. Figure 12.6 shows the situation.



2. CSMA

To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed.
Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle "sense before transmit" or "listen before talk." CSMA can reduce the possibility of collision, but it cannot eliminate it.

Figure 12.8 Space/time model of the collision in CSMA

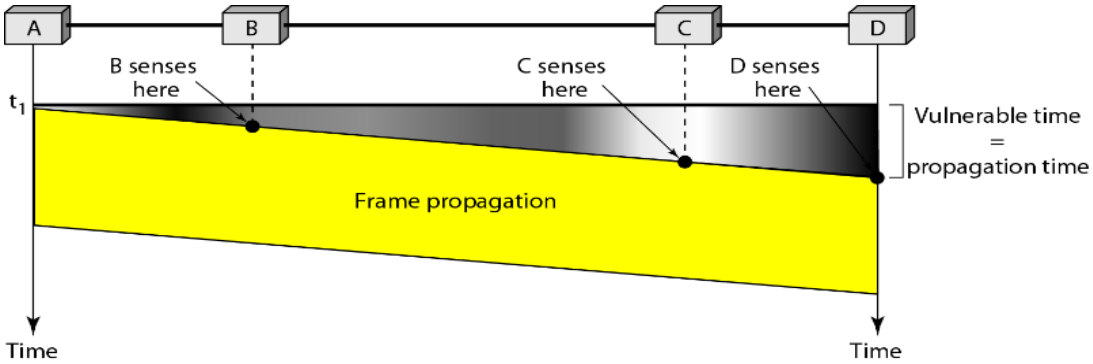


At time t1, station B senses the medium and finds it idle, so it sends a frame. At time t2 (t2 > t1), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.

Vulnerable Time

The vulnerable time for CSMA is the propagation time Tp. This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending. Figure 12.8 shows

the worst case. The leftmost station, A, sends a frame at time t1, which reaches the rightmost station, D, at time t1+ *Tp*. The gray area shows the vulnerable area in time and space.



Persistence Methods

What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions: the 1-persistent method, the non-persistent method, and the p-persistent method. Figure 12.9 shows the behavior of three persistence methods when a station finds a channel busy.

- 1-Persistent

The 1-persistent method is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately.

- Nonpersistent

In the nonpersistent method, a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. The nonpersistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously.

- p- Persistent

The p-persistent method is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The p-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

- With probability p, the station sends its frame.
- With probability q = 1 - p, the station waits for the beginning of the next time slot and checks the line again.
    a. If the line is idle, it goes to step 1.
    b. If the line is busy, it acts as though a collision has occurred and uses the

back- off procedure.

3. CSMA/CD

Carrier sense multiple access with collision detection (CSMA/CD)

In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

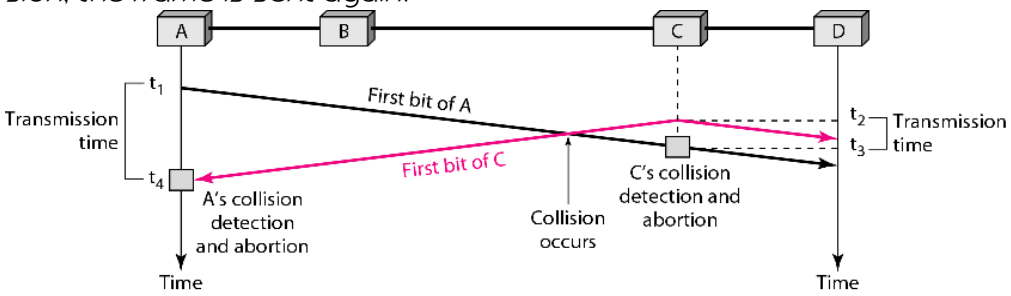


Figure 12.12 *Collision of the first bit in CSMA/CD*

At time t1, station A has executed its persistence procedure and starts sending the bits of its frame. At time t2, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right. The collision occurs sometime after time t2. Station C detects a collision at time t3 when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission. Station A detects collision at time t4 when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at the figure, we see that A transmits for the duration t4- t1; C transmits for the duration t3- t2.

4. CSMA/CA

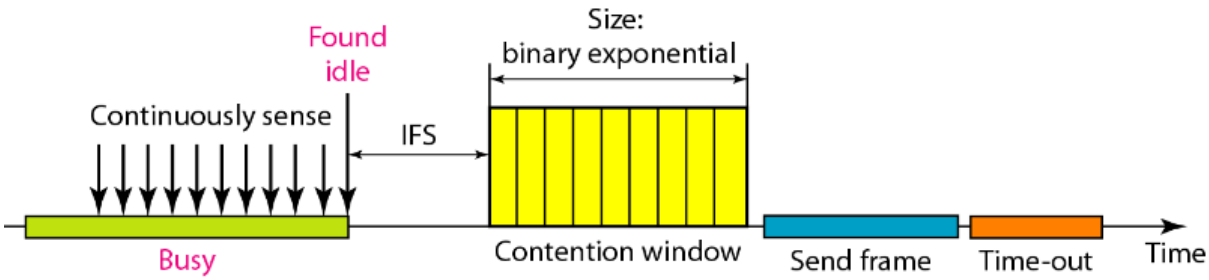
Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for wireless networks. Collisions are avoided through the use of CSMA/CA's three strategies: the interframe space, the contention window, and acknowledgments.

❑ Interframe Space (IFS). First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the interframe space or IFS.

❑ Contention Window. The contention window is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential backoff strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the p-persistent method except that a random outcome defines the

number of slots taken by the waiting station.

❑ Acknowledgment. With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

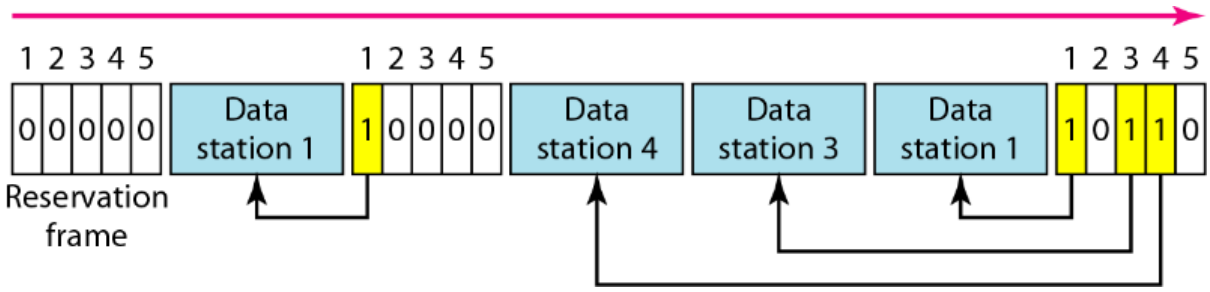


CONTROLLED ACCESS

In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We dis- cuss three controlled-access methods.

1. Reservation

In the reservation method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.
If there are N stations in the system, there are exactly N reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot. The stations that have made reservations can send their data frames after the reservation frame.
Figure 12.18 shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.



2. Polling

Polling works with topologies in which one device is designated as a primary station and the other devices are secondary stations. All data exchanges must be made

through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time. The primary device, therefore, is always the initiator of a session (see Figure 12.19). This method uses poll and select functions to prevent collisions. However, the drawback is if the primary station fails, the system goes down.

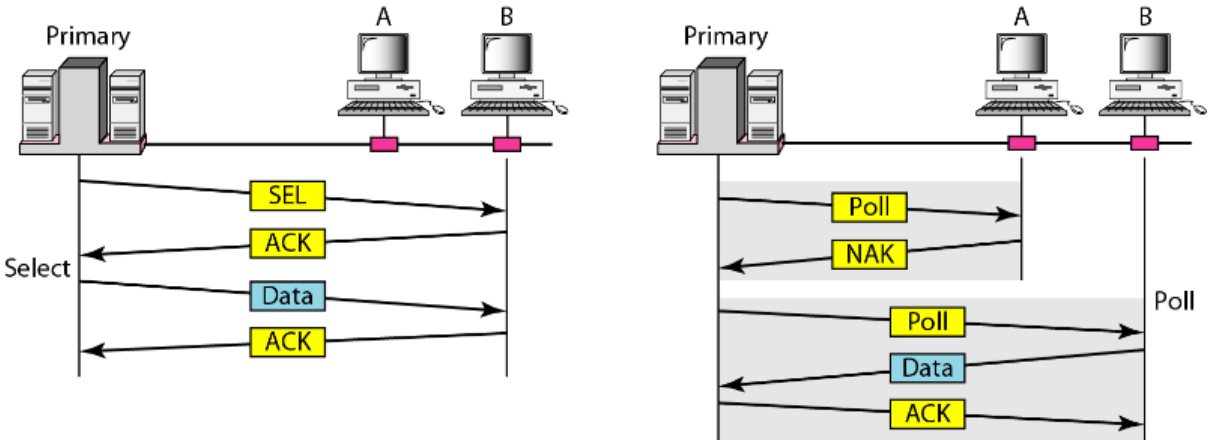


Figure 12.19 Select and poll functions in polling access method

Select

The select function is used whenever the primary device has something to send.

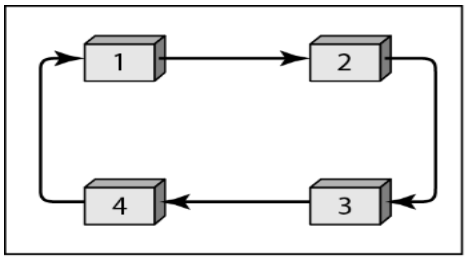
Poll

The poll function is used by the primary device to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does. If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send. When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

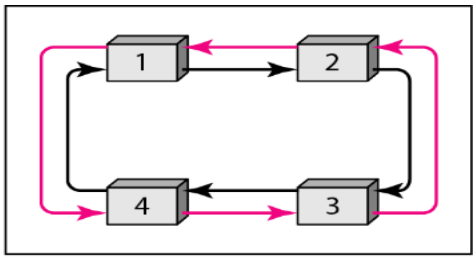
3. Token Passing

In the token-passing method, the stations in a network are organized in a logical ring. In other words, for each station, there is a predecessor and a successor. The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in the ring.

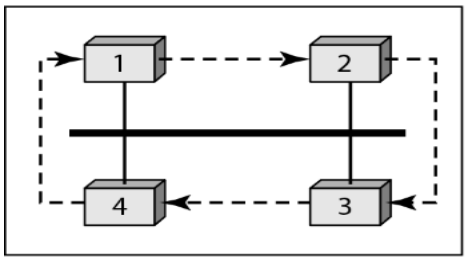
But how is the right to access the channel passed from one station to another? In this method, a special packet called a token circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring.



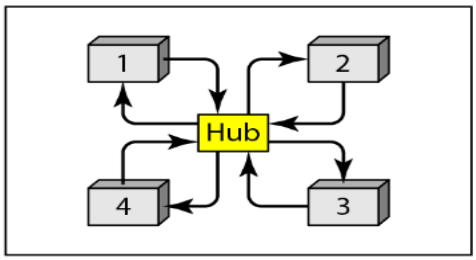
a. Physical ring



b. Dual ring



c. Bus ring



d. Star ring

CHANNELIZATION

Channelization (or channel partition, as it is sometimes called) is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, among different stations.

In this section, we discuss three channelization protocols: **FDMA, TDMA,** and **CDMA**.

1. FDMA

In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands that are separated by guard bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time.

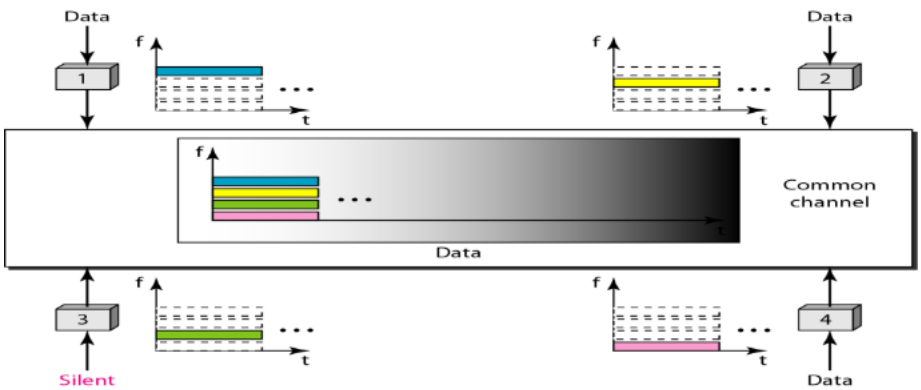
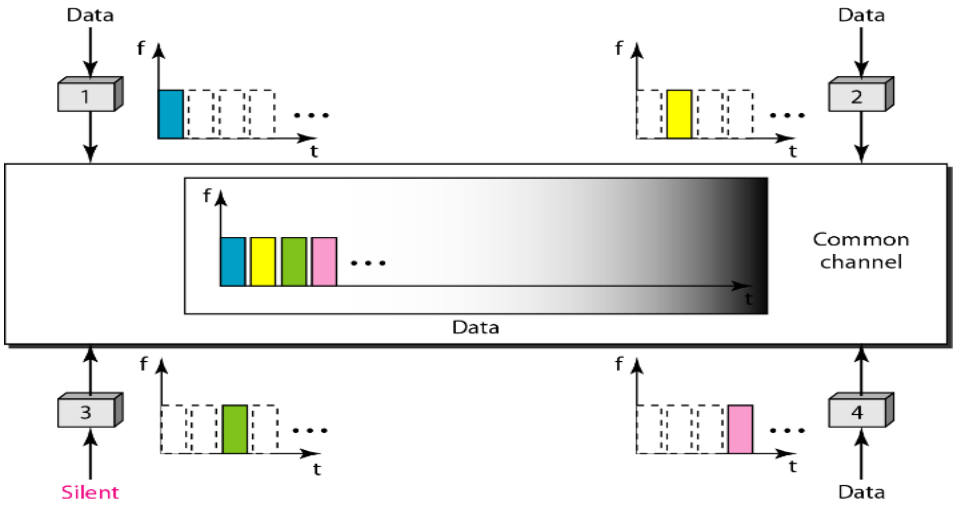


Figure 12.21 Frequency-division multiple access (FDMA)

2. TDMA

In time-division multiple access (TDMA), the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot. Figure 12.22 shows the idea behind TDMA.



The main problem with TDMA lies in achieving synchronization between the different stations. Each station needs to know the beginning of its slot and the location of its slot. This may be difficult because of propagation delays introduced in the system if the sta- tions are spread over a large area. To compensate for the delays, we can insert guard times.

In TDMA, the bandwidth is just one channel that is timeshared between different stations.

3. CDMA

Code-division multiple access (CDMA) In CDMA, one channel carries all transmissions simultaneously.

Idea

Let us assume we have four stations, 1, 2, 3, and 4, connected to the same channel. The data from station 1 are d1, from station 2 are d2, and so on. The code assigned to the first station is c1, to the second is c2, and so on. We assume that the assigned codes have two properties.

1. If we multiply each code by another, we get 0.
2. If we multiply each code by itself, we get 4 (the number of stations).

With these two properties in mind, let us see how the above four stations can send data using the same common channel, as shown in Figure 12.23.

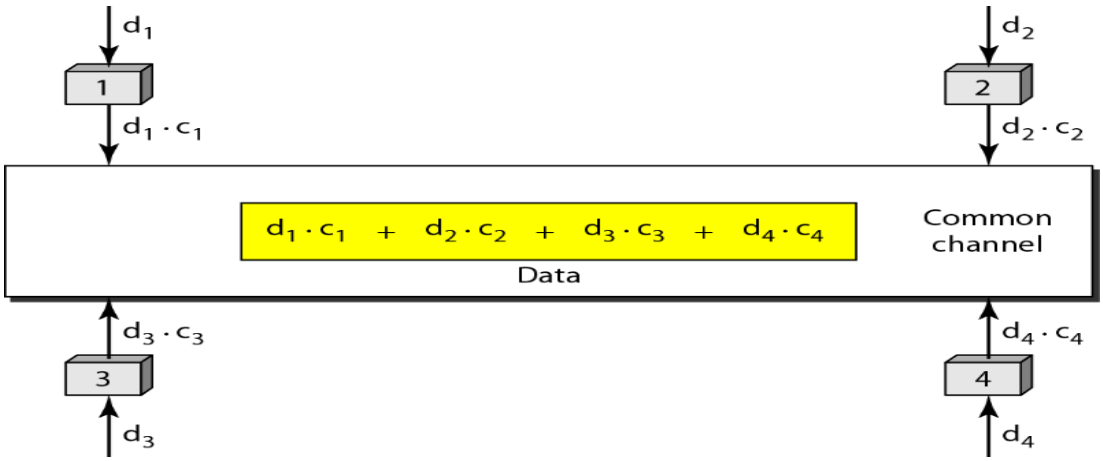


Figure 12.23 Simple idea of communication with code

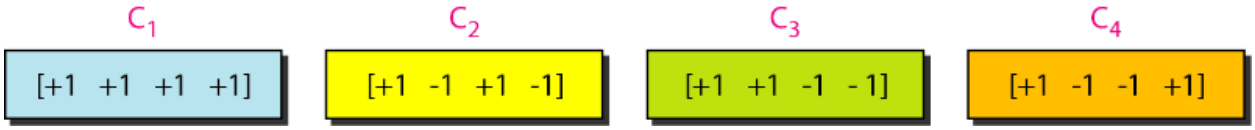


Figure 12.24 Chip sequences

Each sequence is made of N elements, where N is the number of stations.

If we multiply two equal sequences, element by element, and add the results, we get N, where N is the number of elements in each sequence. This is called the ***inner product*** of two equal sequences. For example,

$$[+1 +1 -1 -1] \cdot [+1 +1 -1 -1] = 1 + 1 + 1 + 1 = 4$$

If we multiply two different sequences, element by element, and add the results, we get 0. This is called the *inner product* of two different sequences. For example,

$$[+1 +1 -1 -1] \cdot [+1 +1 +1 +1] = 1 + 1 - 1 - 1 = 0$$

Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

$$[+1 +1 -1 -1] + [+1 +1 +1 +1] = [+2 +2 \ 0 \ 0]$$

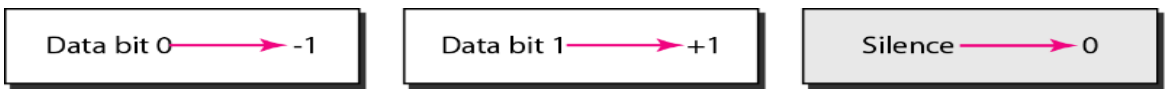


Figure 12.25 Data representation in CDMA

Encoding and Decoding

As a simple example, we show how four stations share the link during a 1-bit interval. The procedure can easily be repeated for additional intervals. We assume that stations 1 and 2 are sending a 0 bit and channel 4 is sending a 1 bit. Station 3 is silent. The data at the sender site are translated to -1, -1, 0, and +1. Each station multiplies the corresponding number by its chip (its orthogonal sequence), which is unique for each station. The result is a new sequence which is sent to the channel. For simplicity, we assume that all stations send the resulting sequences at the same time. The sequence on the channel is the sum of all four sequences as defined before. Figure 12.26 shows the situation.

Now imagine that station 3, which we said is silent, is listening to station 2. Station 3 multiplies the total data on the channel by the code for station 2, which is [+1 -1 +1 -1] to get

$$[-1\ -1\ -3\ +1] \cdot [+1\ -1\ +1\ -1] = -4/4 = -1 \rightarrow \textbf{bit 0;}$$

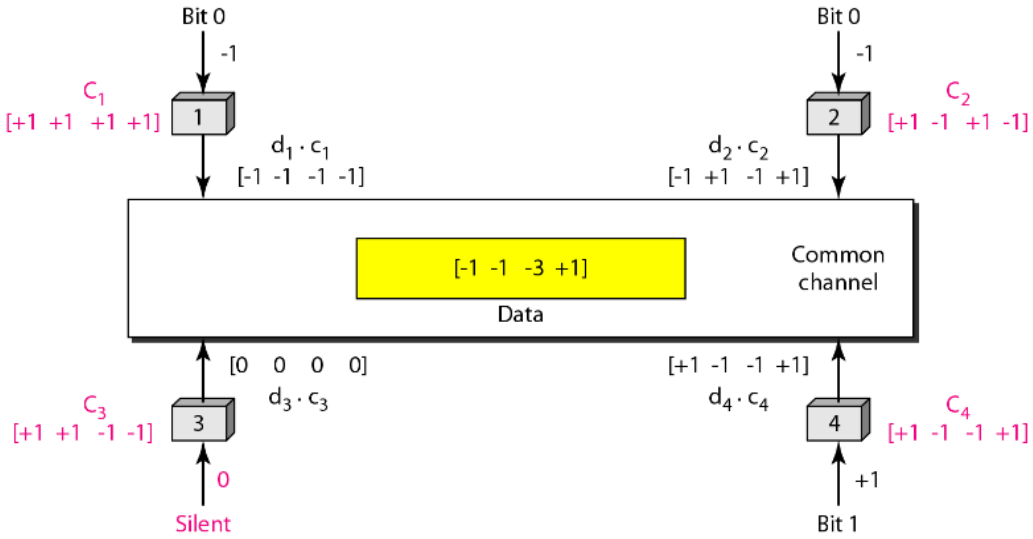


Figure 12.26 Sharing channel in CDMA

Signal Level

The process can be better understood if we show the digital signal produced by each station and the data recovered at the destination (see Figure 12.27). The figure shows the corresponding signals for each station (using NRZ-L for simplicity) and the signal that is on the common channel.

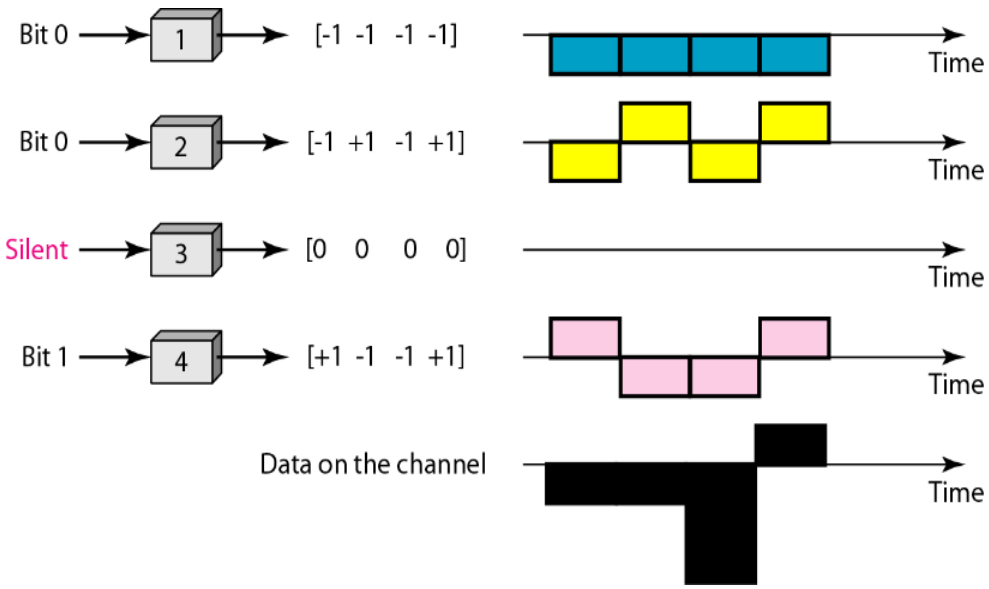


Figure 12.27 Digital signal created by four stations in CDMA

Figure 12.28 shows how station 3 can detect the data sent by station 2 by using the code for station 2. The total data on the channel are multiplied (inner product operation) by the signal representing station 2 chip code to get a new signal. The station then integrates and adds the area under the signal, to get the value 4, which is divided by 4 and interpreted as bit 0.

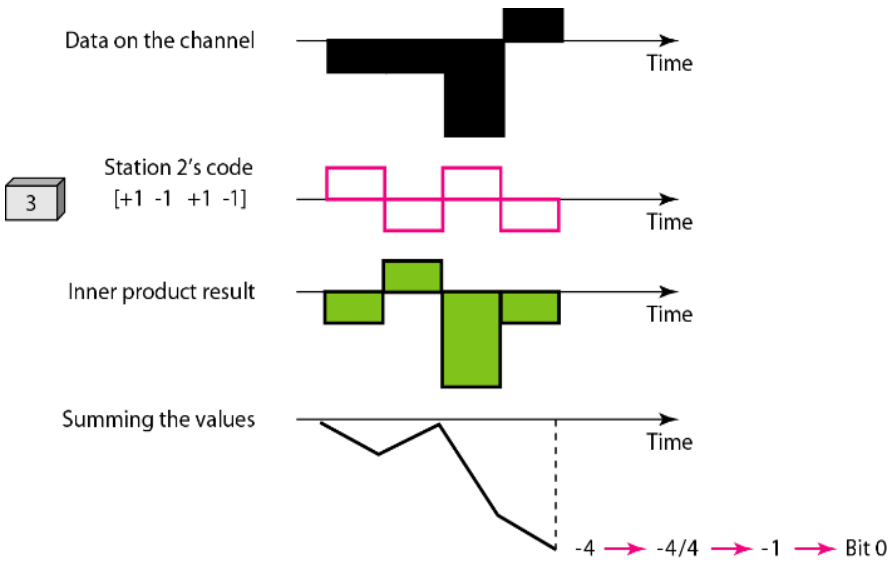


Figure 12.28 Decoding of the composite signal for one in CDMA