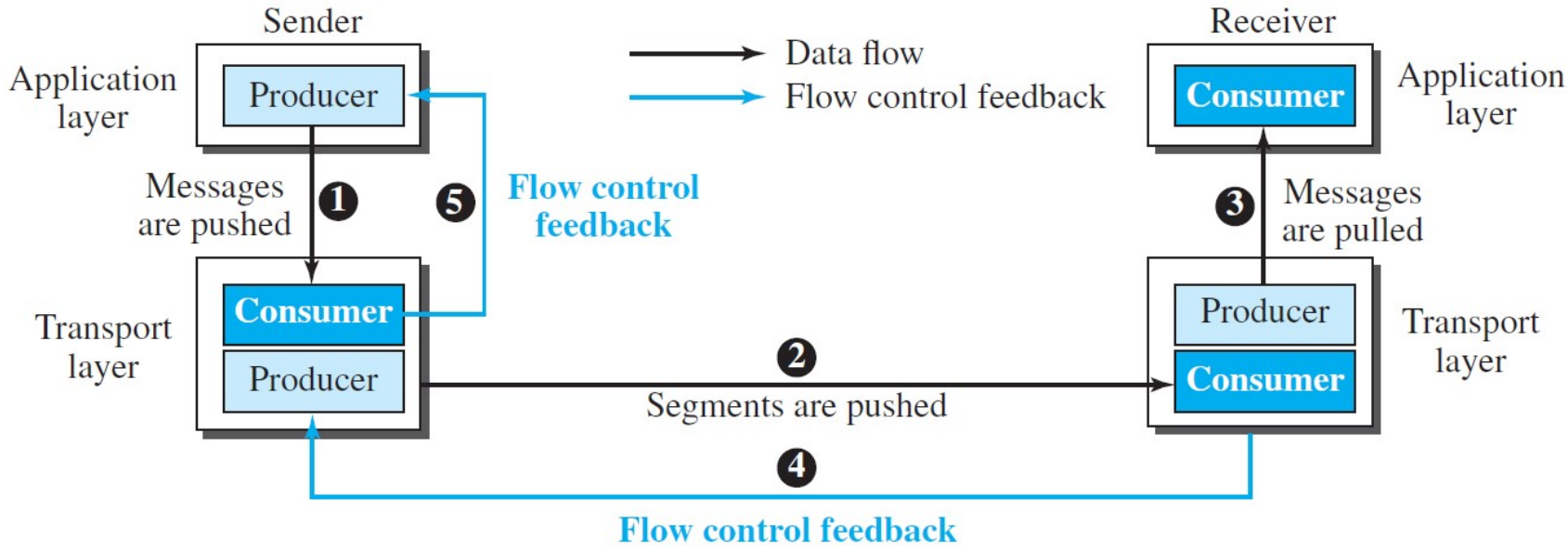
Flow Control

Flow Control

- Flow Control: Flow control balances the rate a producer creates data with the rate a consumer can use the data.

Flow Control

- Flow Control: Flow control balances the rate a producer creates data with the rate a consumer can use the data.



Flow Control

- In most of the protocols, flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the receiver.
- Flow Control coordinates the amount of data that can be sent before receiving acknowledgement. (i.e) the flow of data must not be overwhelm the receiver.
- The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily.

Flow Control

- Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data.
- Incoming data must be checked and processed before they can be used. The rate of such processing is often slower than the rate of transmission.
- For this reason, each receiving device has a block of memory, called buffer, reserved for storing incoming data until they are processed.
- If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

Flow Control

- The basic flow control mechanism are:
  - Stop and Wait ARQ
  - Go-Back N ARQ
  - Selective Repeat ARQ

Flow Control

- Stop and Wait ARQ is a connection-oriented protocol which provides both flow and error control.

Flow Control

- Stop and Wait ARQ is a connection-oriented protocol which provides both flow and error control.
- Sender and the receiver use a sliding window of size 1.
  That is, the sender sends one packet at a time and waits for an acknowledgment before sending the next one.

Flow Control

- Stop and Wait ARQ is a connection-oriented protocol which provides both flow and error control.
- Sender and the receiver use a sliding window of size 1.
  That is, the sender sends one packet at a time and waits for an acknowledgment before sending the next one.
- To detect corrupted packets, it add a checksum to each data packet.
  When a packet arrives at the receiver site, it is checked. If its checksum is incorrect, the packet is corrupted and silently discarded.

Flow Control

- Stop and Wait ARQ is a connection-oriented protocol which provides both flow and error control.
- Sender and the receiver use a sliding window of size 1.
  That is, the sender sends one packet at a time and waits for an acknowledgment before sending the next one.
- To detect corrupted packets, it add a checksum to each data packet.
  When a packet arrives at the receiver site, it is checked. If its checksum is incorrect, the packet is corrupted and silently discarded.
- Every time the sender sends a packet, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next packet (if it has any data to send).

Flow Control

- Stop and Wait ARQ is a connection-oriented protocol which provides both flow and error control.
- Sender and the receiver use a sliding window of size 1.
  That is, the sender sends one packet at a time and waits for an acknowledgment before sending the next one.
- To detect corrupted packets, it add a checksum to each data packet.
  When a packet arrives at the receiver site, it is checked. If its checksum is incorrect, the packet is corrupted and silently discarded.
- Every time the sender sends a packet, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next packet (if it has any data to send).
- If the timer expires, the sender resends the previous packet, assuming that the packet was either lost or corrupted.
  This means that the sender needs to keep a copy of the packet until its acknowledgment arrives.

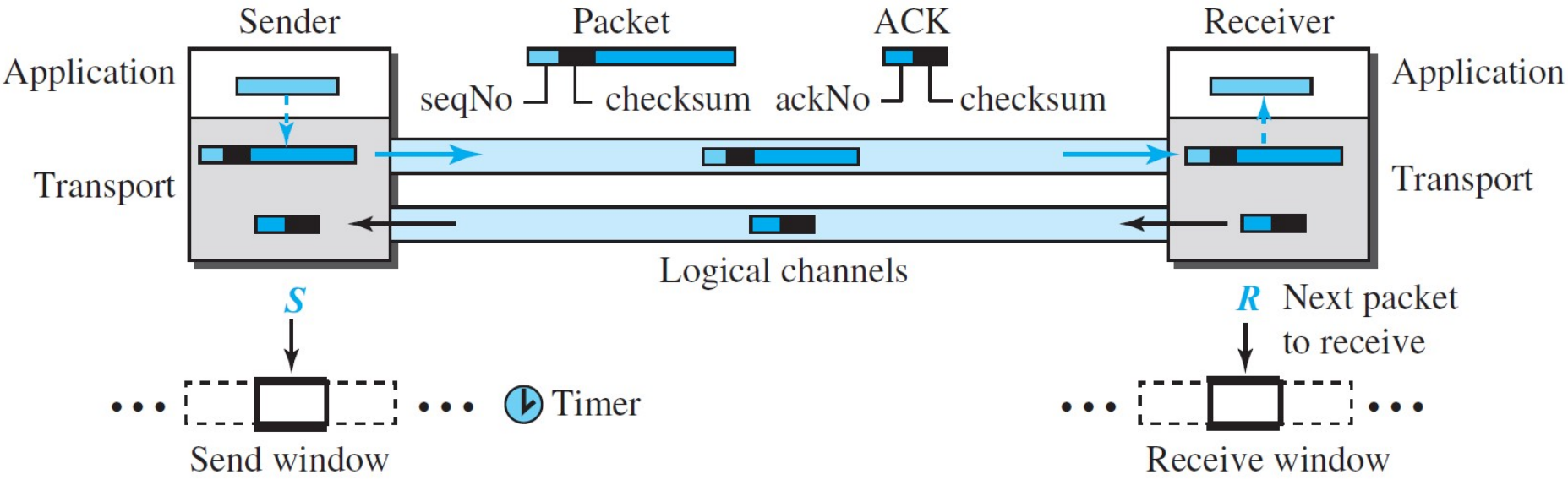


Figure 1 : Stop-and-Wait Protocol

- Sequence Numbers:
  - To prevent duplicate packets, the protocol uses sequence numbers and acknowledgment numbers.
  - A field is added to the packet header to hold the sequence number of that packet.

- Sequence Numbers:
  - To prevent duplicate packets, the protocol uses sequence numbers and acknowledgment numbers.
  - A field is added to the packet header to hold the sequence number of that packet.
- Assume that the sender has sent the packet with sequence number x. Three things can happen;
  - The packet arrives safe at the receiver site; the receiver sends an acknowledgment.
    The acknowledgment arrives at the sender site, causing the sender to send the next packet numbered $x + 1$.

- Sequence Numbers:
  - To prevent duplicate packets, the protocol uses sequence numbers and acknowledgment numbers.
  - A field is added to the packet header to hold the sequence number of that packet.
- Assume that the sender has sent the packet with sequence number x. Three things can happen;
  - The packet arrives safe at the receiver site; the receiver sends an acknowledgment.
    The acknowledgment arrives at the sender site, causing the sender to send the next packet numbered $x + 1$.
  - The packet is corrupted or never arrives at the receiver site; the sender resends the packet (numbered x) after the time-out.
    The receiver returns an acknowledgment.

- Sequence Numbers:
  - To prevent duplicate packets, the protocol uses sequence numbers and acknowledgment numbers.
  - A field is added to the packet header to hold the sequence number of that packet.
- Assume that the sender has sent the packet with sequence number x. Three things can happen;
  - The packet arrives safe at the receiver site; the receiver sends an acknowledgment.
    The acknowledgment arrives at the sender site, causing the sender to send the next packet numbered $x + 1$.
  - The packet is corrupted or never arrives at the receiver site; the sender resends the packet (numbered x) after the time-out.
    The receiver returns an acknowledgment.
  - The packet arrives safe at the receiver site; the receiver sends an acknowledgment, but the acknowledgment is corrupted or lost.
    The sender resends the packet (numbered x) after the time-out.
    Note that the packet here is a duplicate. The receiver can recognize this fact because it expects packet $x + 1$ but packet x was received.

- Acknowledgment Numbers: The acknowledgment numbers always announce the sequence number of the next packet expected by the receiver.
- For example, if packet 0 has arrived safe, the receiver sends an ACK with acknowledgment 1 (meaning packet 1 is expected next).
  If packet 1 has arrived safe, the receiver sends an ACK with acknowledgment 0 (meaning packet 0 is expected).
- In the Stop-and-Wait protocol, the acknowledgment number always announces, in modulo-2 arithmetic, the sequence number of the next packet expected.
- The sender has a control variable, S (sender), that points to the only slot in the send window.
  The receiver has a control variable, R (receiver), that points to the only slot in the receive window.

- Finite State Machine for Stop and Wait Protocol:
  Since the protocol is a connection-oriented protocol, both ends should be in the established state before exchanging data packets.
  The states are actually nested in the established state.
- Sender: The sender is initially in the ready state, but it can move between the ready and blocking state. The variable S is initialized to 0.
  - Ready state
  - Blocking state

- Finite State Machine for Stop and Wait Protocol:
  Since the protocol is a connection-oriented protocol, both ends should be in the established state before exchanging data packets.
  The states are actually nested in the established state.
- Sender: The sender is initially in the ready state, but it can move between the ready and blocking state. The variable S is initialized to 0.
  - Ready state
  - Blocking state
- Ready state
  - When the sender is in this state, it is only waiting for one event to occur.
  - If a request comes from the application layer, the sender creates a packet with the sequence number set to S.
  - A copy of the packet is stored, and the packet is sent.
  - The sender then starts the timer.
  - The sender then moves to the blocking state.

- Blocking state: When the sender is in this state, three events can occur;
  - If an error-free ACK arrives with the ackNo related to the next packet to be sent, which means ackNo $= (S + 1)$ modulo 2, then the timer is stopped.
    The window slides, $S = (S + 1) modulo 2$.
    Finally, the sender moves to the ready state.
  - If a corrupted ACK or an error-free ACK with the $ackNo \neq (S + 1) modulo 2$ arrives, the ACK is discarded.
  - If a time-out occurs, the sender resends the only outstanding packet and restarts the timer.

- Blocking state: When the sender is in this state, three events can occur;
  - If an error-free ACK arrives with the ackNo related to the next packet to be sent, which means ackNo $= (S + 1)$ modulo 2, then the timer is stopped.
    The window slides, $S = (S + 1) modulo 2$.
    Finally, the sender moves to the ready state.
  - If a corrupted ACK or an error-free ACK with the $ackNo \neq (S + 1) modulo 2$ arrives, the ACK is discarded.
  - If a time-out occurs, the sender resends the only outstanding packet and restarts the timer.
- Receiver: The receiver is always in the ready state. Three events may occur;
  - If an error-free packet with $seqNo = R$ arrives, the message in the packet is delivered to the application layer.
    The window then slides, $R = (R + 1) modulo 2$.
    Finally an ACK with $ackNo = R$ is sent.
  - If an error-free packet with $seqNo \neq R$ arrives, the packet is discarded, but an ACK with $ackNo = R$ is sent.
  - If a corrupted packet arrives, the packet is discarded

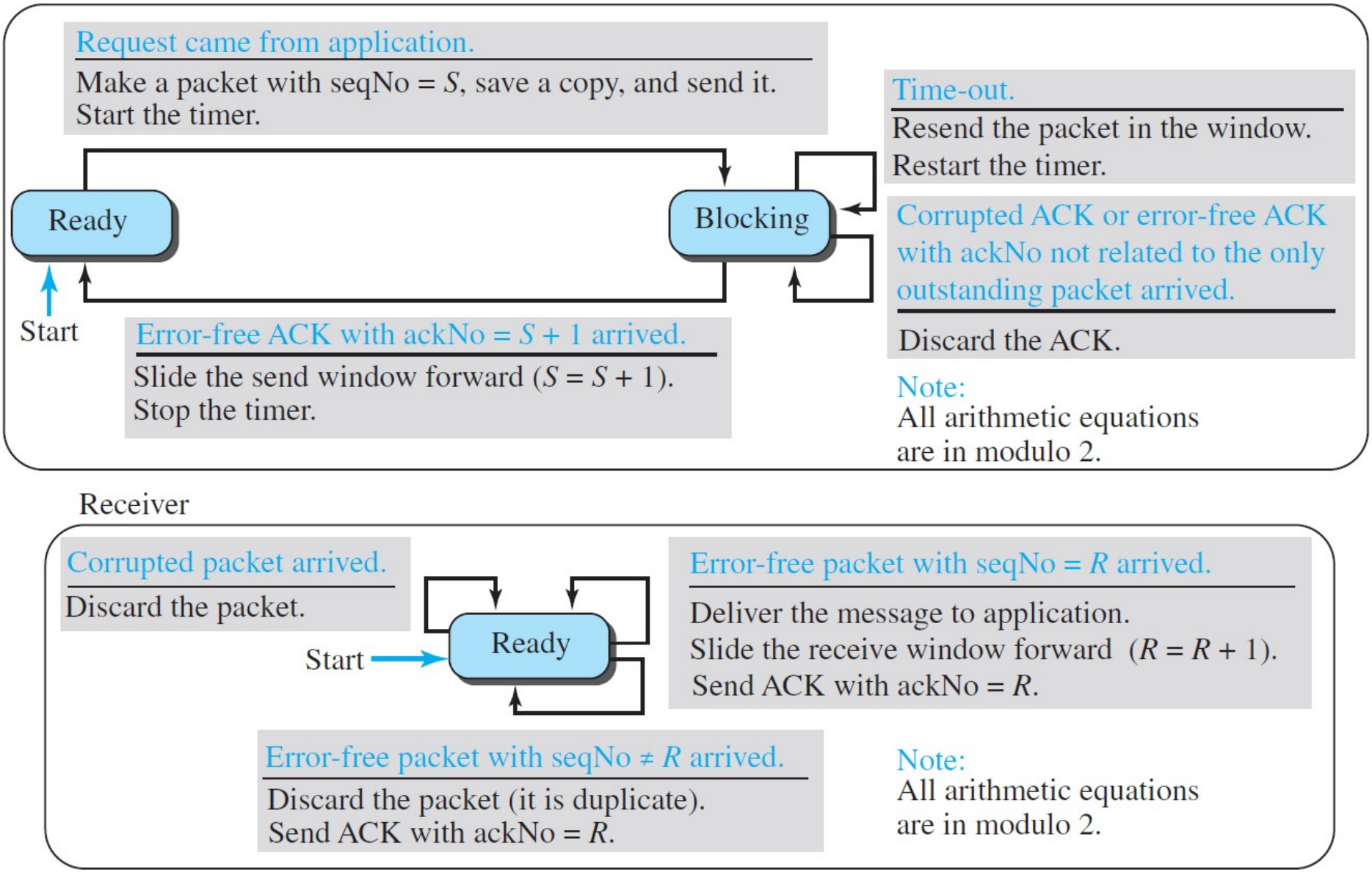
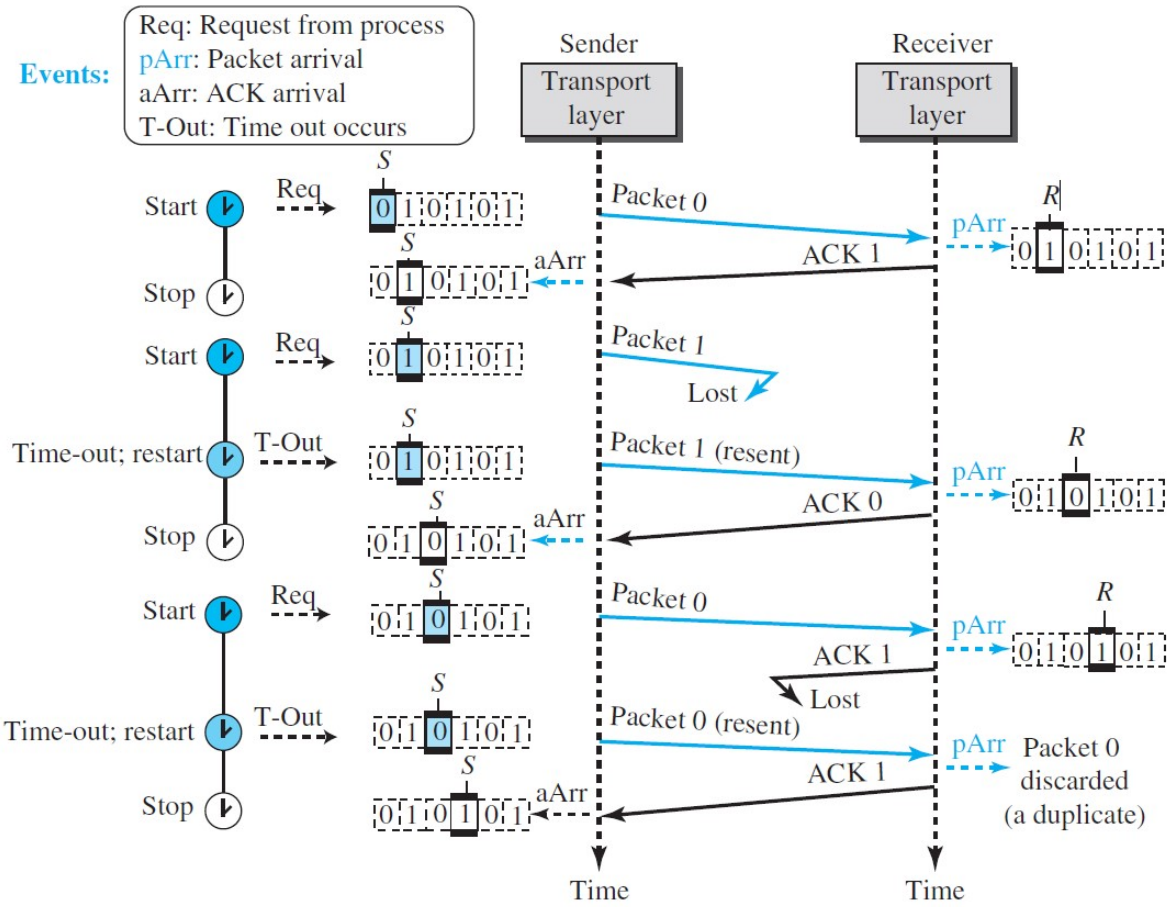


Figure 2 : Finite State Machine for Stop and Wait Protocol

- Example1: In stop and wait protocol, Packet 0 is sent and acknowledged; Packet 1 is lost and resent after the time-out; The resent packet 1 is acknowledged and the timer stops; Packet 0 is sent and acknowledged, but the acknowledgment is lost; The sender has no idea if the packet or the acknowledgment is lost, so after the time-out, it resends packet 0, which is acknowledged.



Flow Control

- Efficiency: The Stop-and-Wait protocol is very inefficient if the channel is thick and long. (i.e, with large bandwidth and long round trip delay).
  The product of these two is called the bandwidth delay product.
- Assume the channel as a pipe. Then, the bandwidth-delay product is the volume of the pipe in bits.
  The pipe is always there. It is not efficient if it is not used.
- The bandwidth-delay product is a measure of the number of bits a sender can transmit through the system while waiting for an acknowledgment from the receiver.

Flow Control

- Exercise1: Assume that, in a Stop-and-Wait system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 milliseconds to make a round trip. What is the bandwidth-delay product? If the system data packets are 1,000 bits in length, what is the utilization percentage of the link?

Flow Control

- Exercise1: Assume that, in a Stop-and-Wait system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 milliseconds to make a round trip. What is the bandwidth-delay product? If the system data packets are 1,000 bits in length, what is the utilization percentage of the link?
- Solution:
  - The bandwidth-delay product $= (1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000$ bits
    The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and the acknowledgment to come back.
  - However, the system sends only 1,000 bits,
    so the link utilization $= \frac{1000}{20000} = 5\%$
  - For this reason, in a link with a high bandwidth or long delay, the use of Stop-and-Wait wastes the capacity of the link.

Go Back N ARQ

Flow Control

- Pipelining: In networking, a task is often begun before the previous task has ended. This is known as pipelining.
- There is no pipelining in the Stop-and-Wait protocol because a sender must wait for a packet to reach the destination and be acknowledged before the next packet can be sent.
- Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidthdelay product.
- Pipelining does apply to Go-Back-N and Selective repeat protocol because several packets can be sent before a sender receives feedback about the previous packets.

Flow Control

- Go Back N ARQ: In Stop and Wait ARQ, there is only one frame, the outstanding frame, that is sent and waiting to be acknowledged.
  This is not a good use of the transmission medium.

Flow Control

- Go Back N ARQ: In Stop and Wait ARQ, there is only one frame, the outstanding frame, that is sent and waiting to be acknowledged.
  This is not a good use of the transmission medium.
- To improve the efficiency, multiple frames should be in transition while waiting for acknowledgement.
  (i.e) more than one frame be outstanding.

Flow Control

- Go Back N ARQ: In Stop and Wait ARQ, there is only one frame, the outstanding frame, that is sent and waiting to be acknowledged.
  This is not a good use of the transmission medium.
- To improve the efficiency, multiple frames should be in transition while waiting for acknowledgement.
  (i.e) more than one frame be outstanding.
- Two protocols use this concept:
  (1) Go Back N ARQ
  (2) Selective Repeat ARQ

Flow Control

- In Go Back N ARQ, we can send upto W frames before worrying about acknowledgements, but we can keep a copy of these frames until the acknowledgement arrive.

Flow Control

- In Go Back N ARQ, we can send upto W frames before worrying about acknowledgements, but we can keep a copy of these frames until the acknowledgement arrive.
- This procedure requires additional features to be added to Stop and Wait ARQ
  - Sequence Numbers
  - Acknowledgement number
  - Sender Sliding Window
  - Receiver Sliding Window
  - Control variables
  - Timers
  - Resending frame

Flow Control

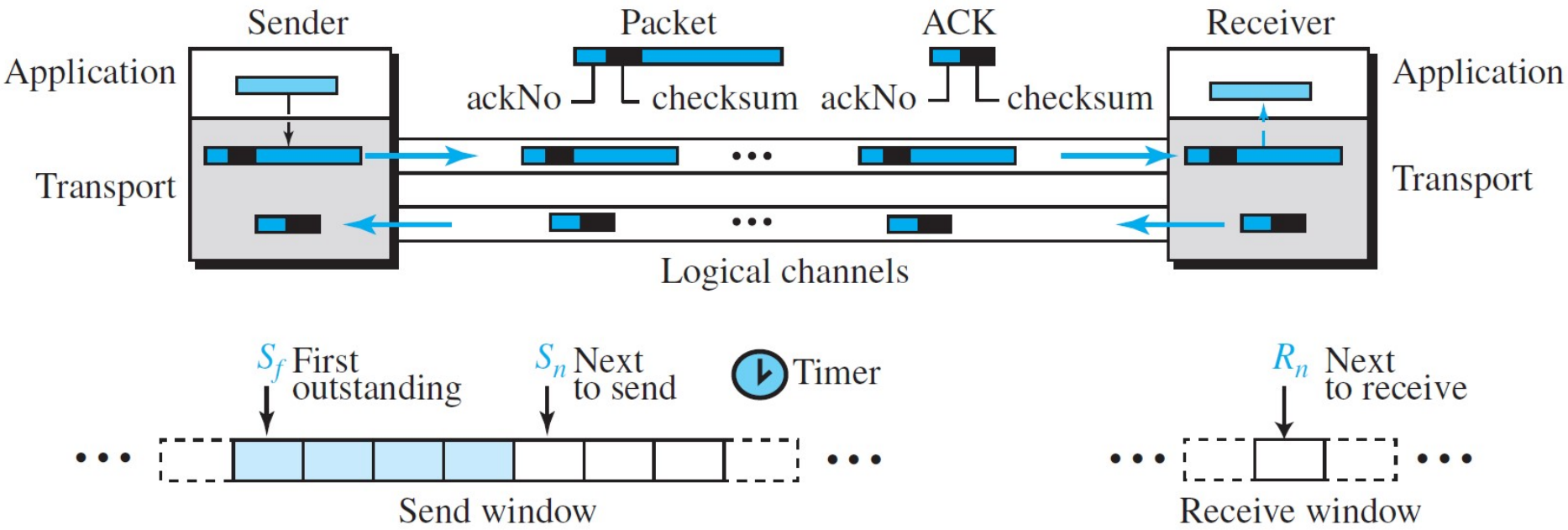


Figure 3 : Go Back N Protocol

Flow Control

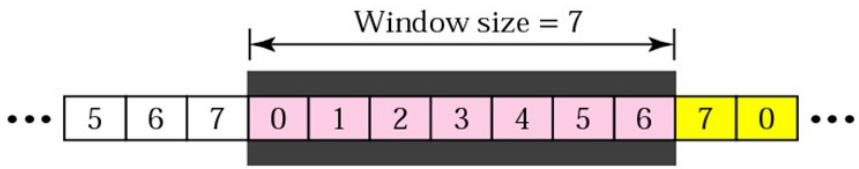
- Sequence Numbers: Frames from a sending station are numbered sequentially. We need to include the sequence number of each frame in the header, we need to set a limit.
- If the header of the frame allows m bit for the sequence number, the sequence number range from 0 to $2^m - 1$.
  (i.e) if m is 3, the only sequence numbers are 0 through 7. However, we can repeat the sequence. So the sequence numbers are
  0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1,...

Flow Control

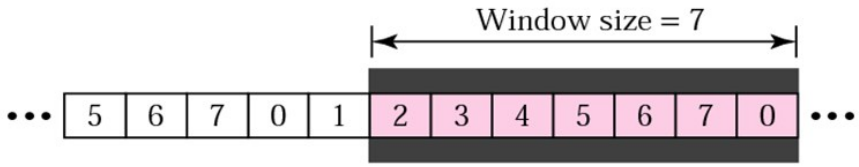
- Acknowledgement Numbers: An acknowledgment number in this protocol is cumulative and defines the sequence number of the next packet expected.
- For example, if the acknowledgment number (ackNo) is 7, it means all packets with sequence number up to 6 have arrived safe, and the receiver is expecting the packet with sequence number 7.

Flow Control

- Sender Sliding Window: At the sender side, to hold the outstanding frames until they are acknowledged, it use a concept of a window. The outstanding frames are enclosed in a window.
- The frames to the left of the window are those that have been already acknowledged;
  frames those to the right of the window cannot be sent until the window slides over them.
- The maximum size of the window is $2^m - 1$.

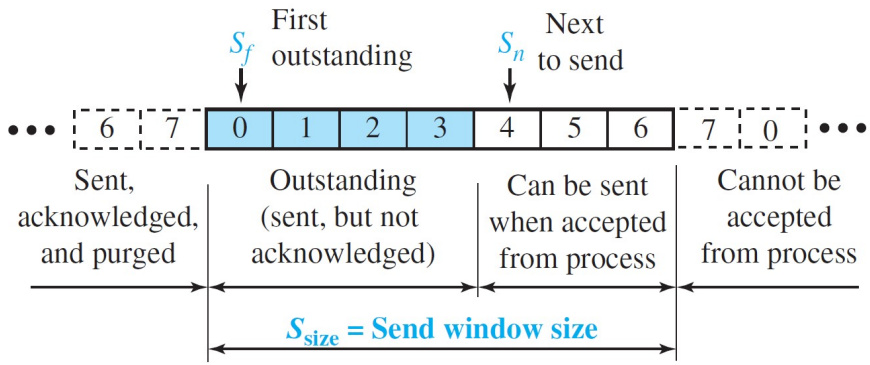


a. Before sliding

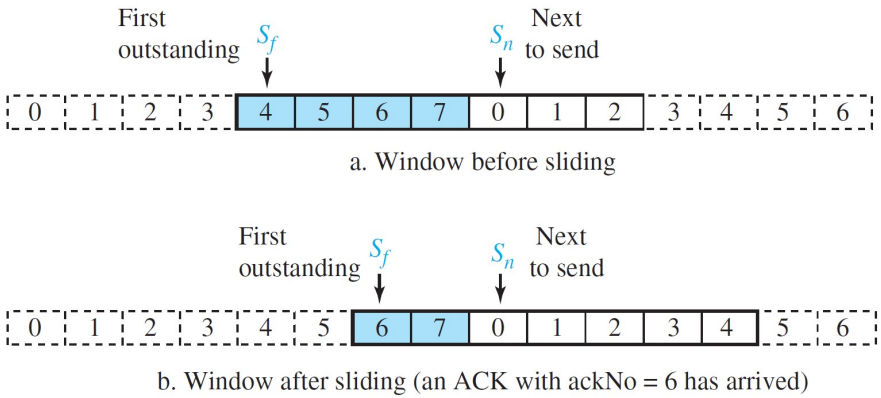


b. After sliding two frames

- The send window at any time divides the possible sequence numbers into four regions.
  - The first region, left of the window, defines the sequence numbers belonging to packets that are already acknowledged.
    The sender does not worry about these packets and keeps no copies of them.
  - The second region, colored, defines the range of sequence numbers belonging to the packets that have been sent, but have an unknown status.
    The sender needs to wait to find out if these packets have been received or were lost. We call these outstanding packets.
  - The third range, white in the figure, defines the range of sequence numbers for packets that can be sent; however, the corresponding data have not yet been received from the application layer.
  - Finally, the fourth region, right of the window, defines sequence numbers that cannot be used until the window slides.



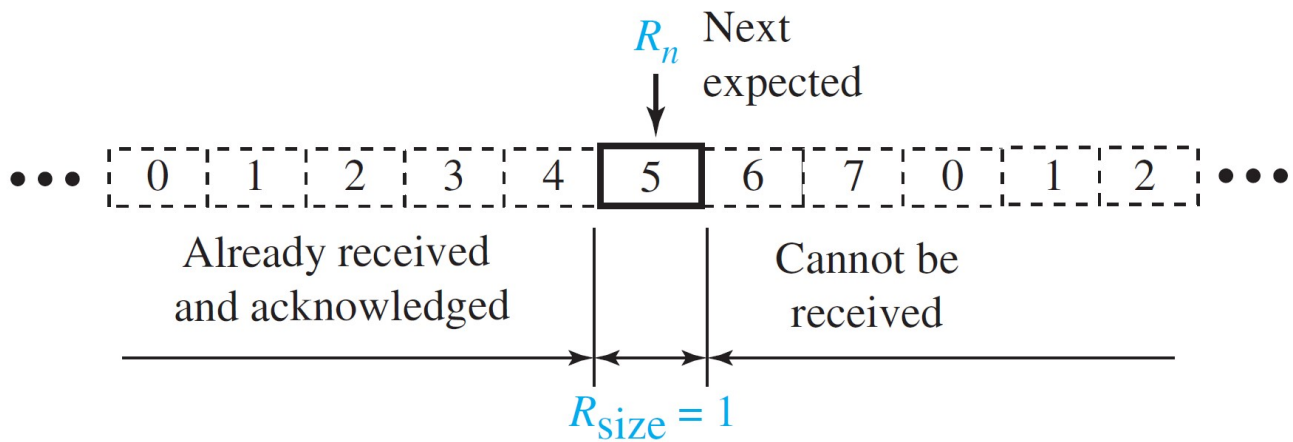
- Control Variables: Three variables define its size and location at any time;
  (1) S_f (send window, the first outstanding packet),
  (2) S_n (send window, the next packet to be sent)
  (3) S_{size} (send window, size).
  - The variable S_f defines the sequence number of the first (oldest) outstanding packet.
  - The variable S_n holds the sequence number that will be assigned to the next packet to be sent.
  - The variable S_{size} defines the size of the window, which is fixed in our protocol.



- The send window can slide one or more slots when an error-free ACK with ackNo greater than or equal to S_f and less than S_n (in modular arithmetic) arrives.

Flow Control

- Receiver Sliding Window: The receive window makes sure that the correct data packets are received and that the correct acknowledgments are sent.
- In Go-Back-N, the size of the receive window is always 1.
  The receiver is always looking for the arrival of a specific packet.
- Any packet arriving out of order is discarded and needs to be resent.
  When a correct packet is received, the window slides, $R_n = (R_n + 1) modulo 2^m$.



Flow Control

- Timers: The sender sets a timer for each frame sent. The receiver has no timers.
- Acknowledgemet: The receiver sends a positive acknowledgements if a frame has arrived and in order.
- If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting.
- The silence of the receiver causes the timer of the unacknowledged frame to expire.
  This, in turn, causes the sender to go back and resend all frames, beginning with the one with the expired timer.
- The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgement for several frames.

Flow Control

- Resending Frame: When a frame is damaged, the sender goes back and sends a set of frames starting from the damaged one up to the last one sent.
- Suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged, so the sender goes back and send frames 3,4,5,6 again.

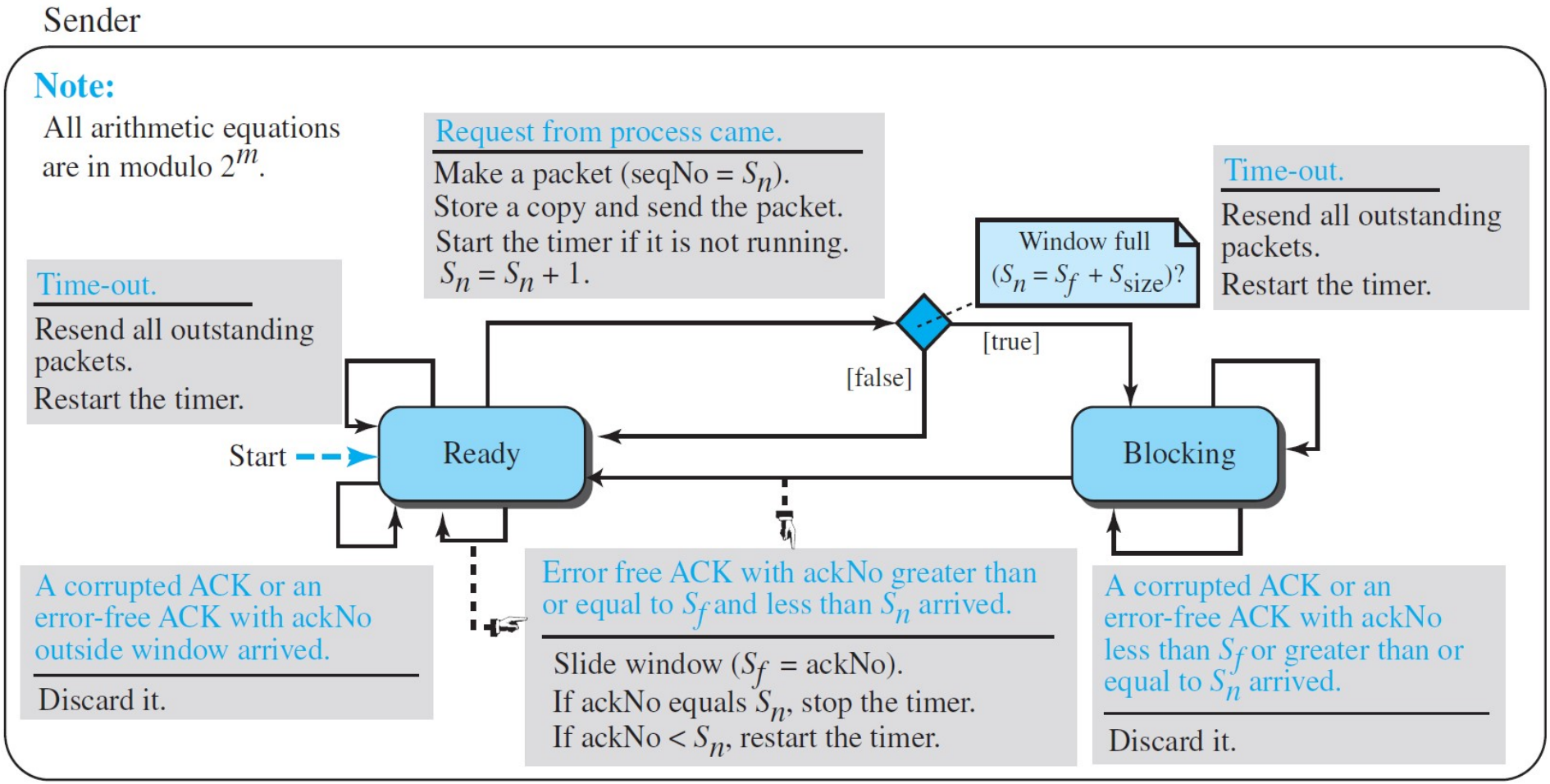


Figure 4 : FSM for Go-BACK-N Sender

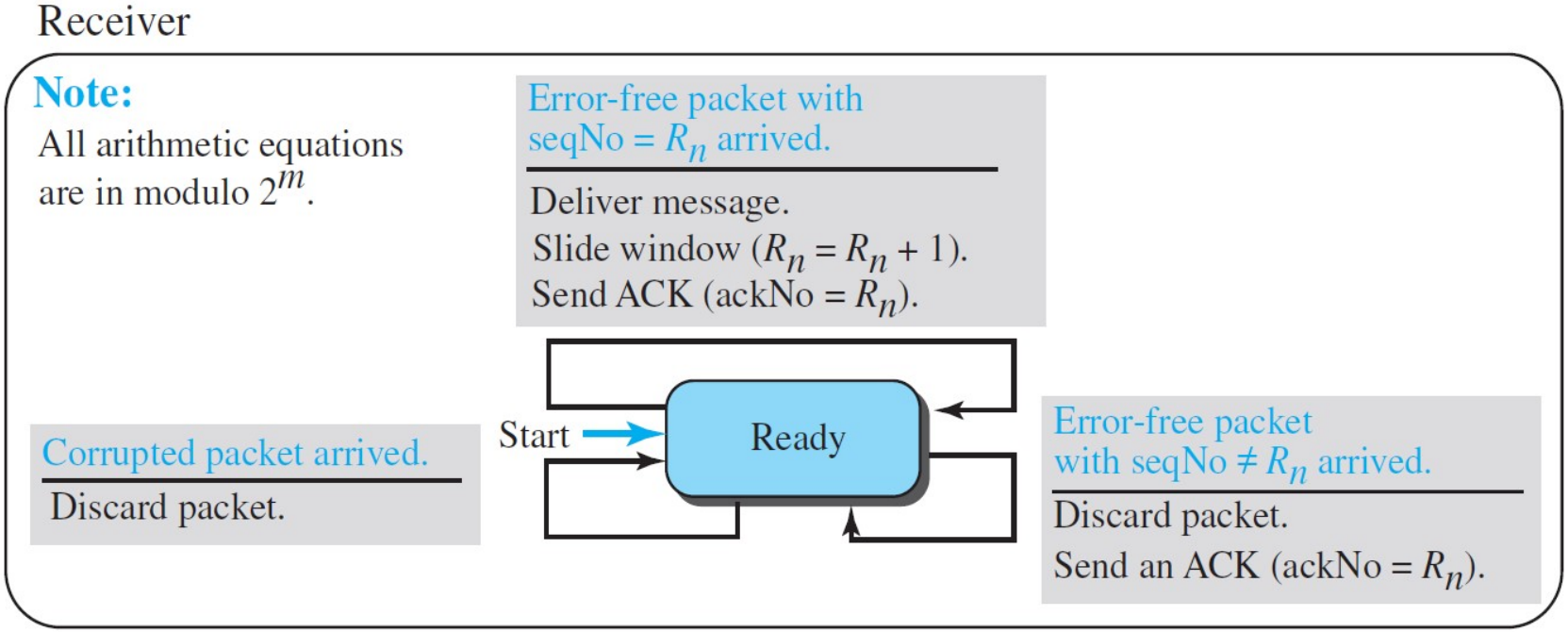


Figure 5 : FSM for Go-BACK-N Receiver

Flow Control

- Sender: The sender starts in the ready state, but thereafter it can be in one of the two states: ready or blocking.
  The two variables are normally initialized to 0 ($S_f = S_n = 0$)
- Ready State: Four events may occur when the sender is in ready state.
  - If a request comes from the application layer, the sender creates a packet with the sequence number set to S_n.
    A copy of the packet is stored, and the packet is sent. The sender also starts the only timer if it is not running. The value of S_n is now incremented, $(S_n = S_n + 1) modulo 2^m$.
    If the window is full, $S_n = (S_f + S_{size}) modulo 2^m$, the sender goes to the blocking state.
  - If an error-free ACK arrives with ackNo related to one of the outstanding packets, the sender slides the window (set S_f = ackNo), and if all outstanding packets are acknowledged ($ackNo = S_n$), then the timer is stopped.
    If all outstanding packets are not acknowledged, the timer is restarted.
  - If a corrupted ACK or an error-free ACK with ackNo not related to the outstanding packet arrives, it is discarded.
  - If a time-out occurs, the sender resends all outstanding packets and restarts the timer.

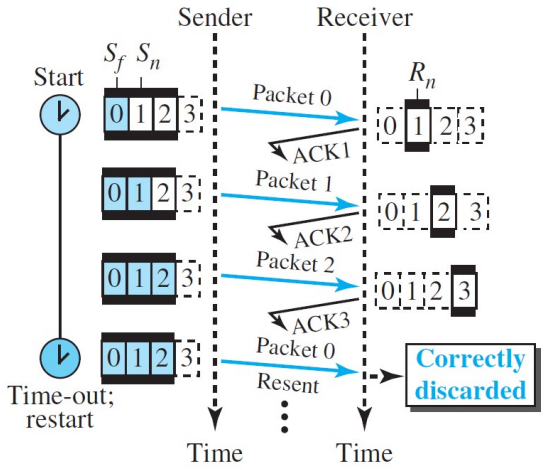
Flow Control

- Blocking State: Three events may occur in this case;
  - If an error-free ACK arrives with ackNo related to one of the outstanding packets, the sender slides the window ($setS_f = ackNo$) and if all outstanding packets are acknowledged ($ackNo = S_n$), then the timer is stopped.
    If all outstanding packets are not acknowledged, the timer is restarted.
    The sender then moves to the ready state.
  - If a corrupted ACK or an error-free ACK with the ackNo not related to the outstanding packets arrives, the ACK is discarded.
  - If a time-out occurs, the sender sends all outstanding packets and restarts the timer.

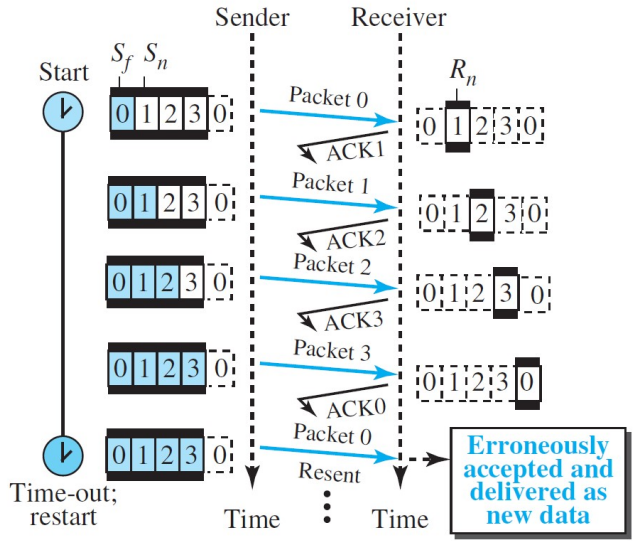
Flow Control

- Receiver: The receiver is always in the ready state. The only variable, R_n, is initialized to 0. Three events may occur;
  - If an error-free packet with $seqNo = R_n$ arrives, the message in the packet is delivered to the application layer.
    The window then slides, $R_n = (R_n + 1) modulo 2^m$. Finally an ACK is sent with $ackNo = R_n$.
  - If an error-free packet with seqNo outside the window arrives, the packet is discarded, but an ACK with $ackNo = R_n$ is sent.
  - If a corrupted packet arrives, it is discarded.

- In the Go-Back-N protocol, the size of the send window must be less than 2^m;
- Proof: Let us choose $m = 2$, which means the size of the window can be $(2^m - 1) = 3$. Let us compares a window size of 3 against a window size of 4.
  - If the size of the window is 3 (less than 2^m) and all three acknowledgments are lost, the only timer expires and all three packets are resent. The receiver is now expecting packet 3, not packet 0, so the duplicate packet is correctly discarded.
  - On the other hand, if the size of the window is 4 (equal to 2^2) and all acknowledgments are lost, the sender will send a duplicate of packet 0.
    However, this time the window of the receiver expects to receive packet 0 (in the next cycle), so it accepts packet 0, not as a duplicate, but as the first packet in the next cycle. This is an error. This shows that the size of the send window must be less than 2^m.

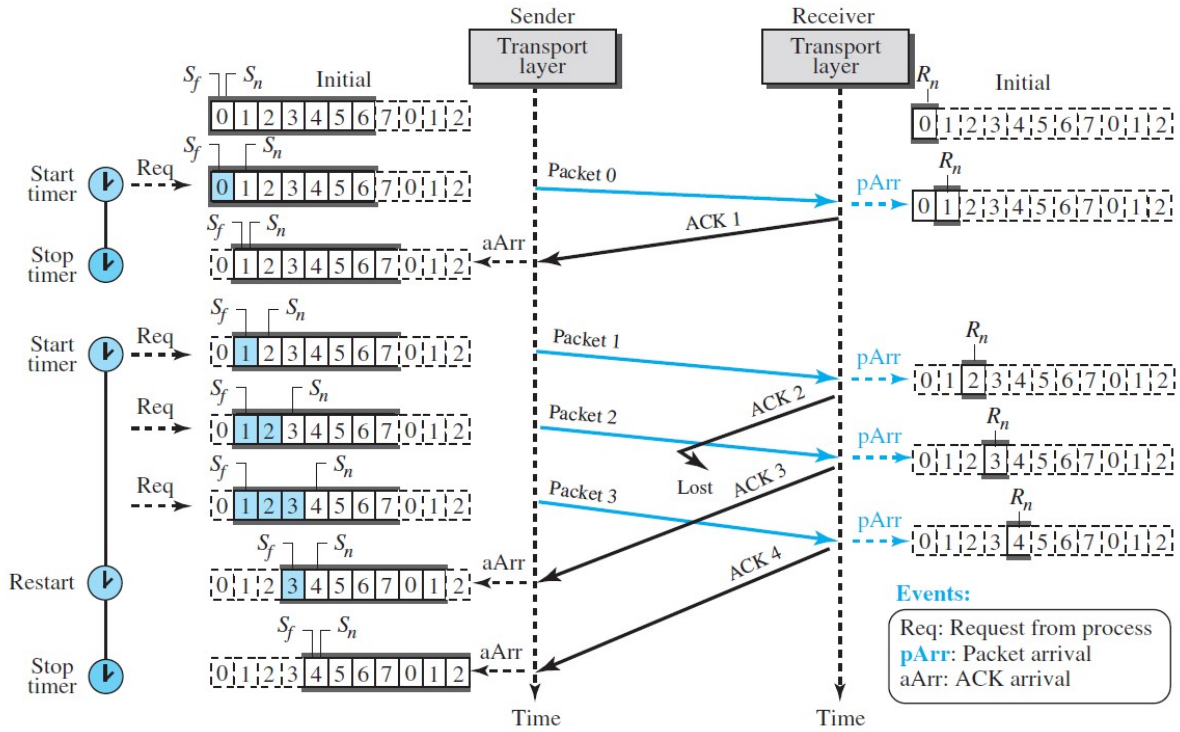


a. Send window of size $< 2^m$

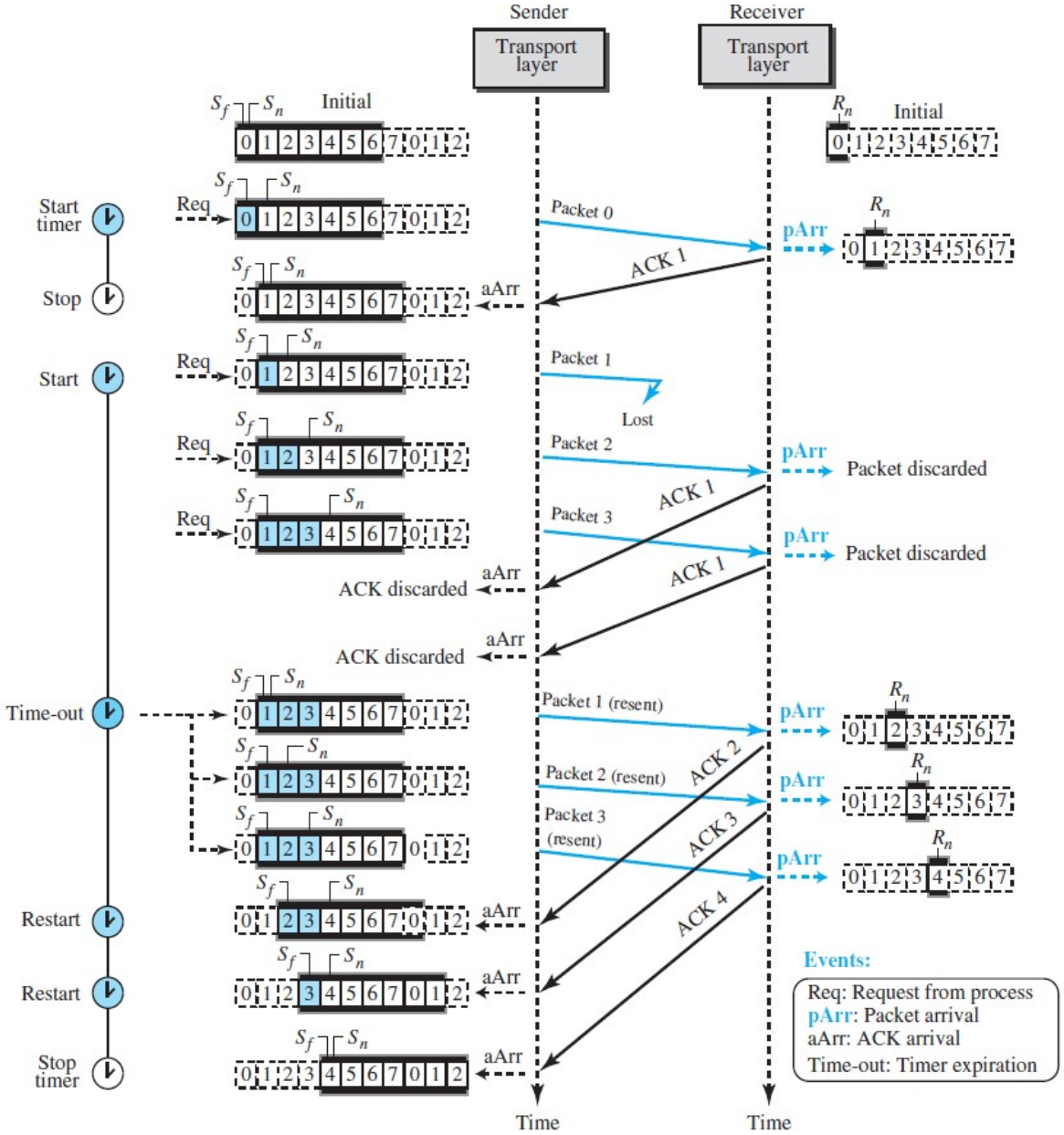


b. Send window of size = 2^m

- Example1: Go-Back-N example of a case where the forward channel is reliable, but the reverse is not. No data packets are lost, but some ACKs are delayed and one is lost. This example shows how cumulative acknowledgments can help if acknowledgments are delayed or lost.



- Example2: In Go-Back-N what happens when a packet is lost. Packets 0, 1, 2, and 3 are sent. However, packet 1 is lost. The receiver receives packets 2 and 3, but they are discarded because they are received out of order (packet 1 is expected). When the receiver receives packets 2 and 3, it sends ACK1 to show that it expects to receive packet 1. However, these ACKs are not useful for the sender because the ackNo is equal to S_f , not greater than S_f. So the sender discards them. When the time-out occurs, the sender resends packets 1, 2, and 3, which are acknowledged.


Sender
Transport layer
Receiver
Transport layer
Initial
Start timer
Stop
Start
Time-out
Restart
Restart
Stop timer
Req
Packet 0
ACK 1
pArr
aArr
Packet 1
Lost
Packet 2
Packet discarded
Packet 3
ACK discarded
Packet 1 (resent)
Packet 2 (resent)
Packet 3 (resent)
ACK 2
ACK 3
ACK 4
Events:
Req: Request from process
pArr: Packet arrival
aArr: ACK arrival
Time-out: Timer expiration
Time
Time

Selective Repeat ARQ

Selective Repeat ARQ

- Go Back N ARQ simplifies the process at the receiver side. The receiver keeps track of only one variable; there is no need to buffer out-of-order frames, they are simply discarded.
  However, this protocol is very inefficient for a noisy link.
- In a noisy link, a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission.
- For noisy links, there is another mechanism that does not resend N frames when just one frame is damaged; only the damaged frame is resent.
  This mechanism is called Selective Repeat ARQ.

Selective Repeat ARQ

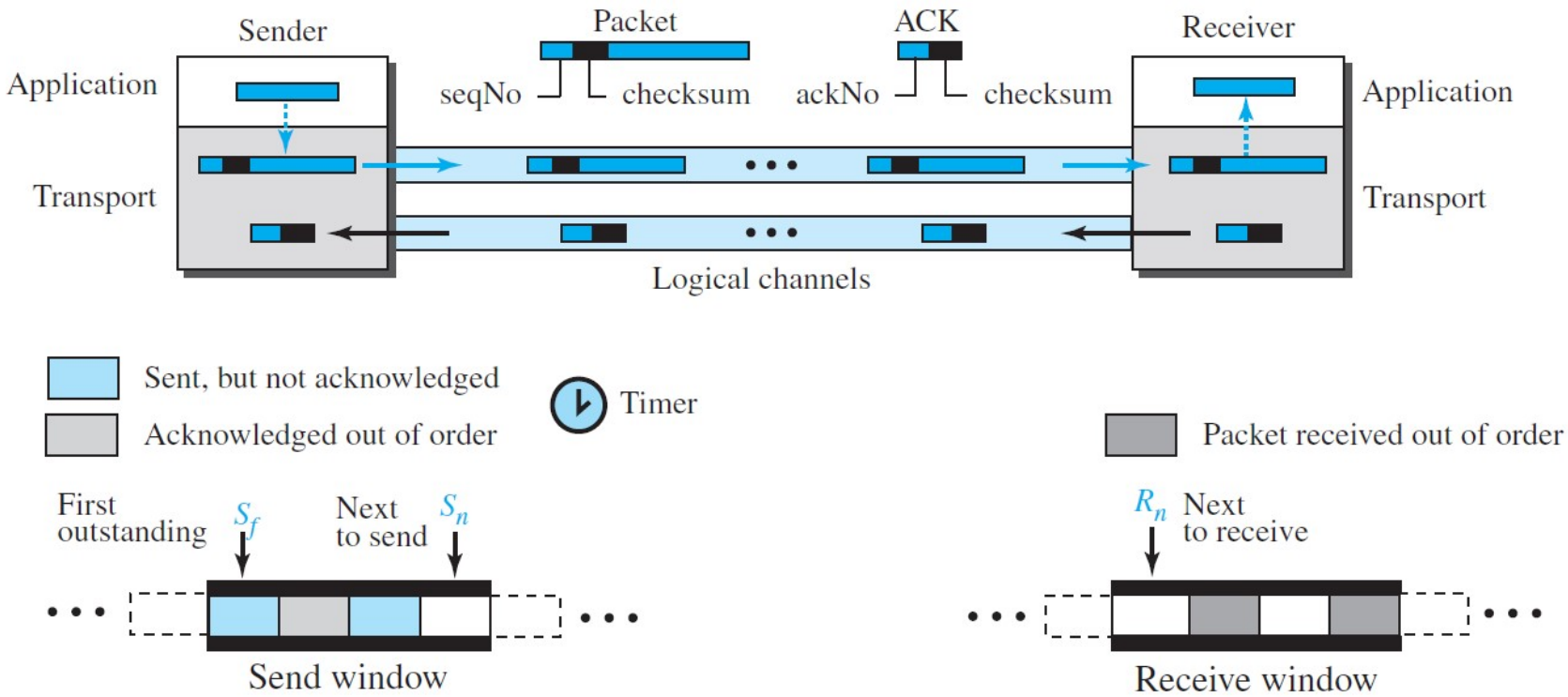


Figure 6 : Selective Repeat Protocol

Selective Repeat ARQ

- Sender and Receiver window: The Selective-Repeat protocol also uses two windows
  - send window
  - receive window
- The maximum size of the send window is much smaller; it is 2^{m-1}
- The receive window is the same size as the send window.
- For example, if $m = 4$, the sequence numbers go from 0 to 15, but the maximum size of the window is just 8.
  But, it is 15 in the Go-Back-N Protocol.

Selective Repeat ARQ

- The Selective-Repeat protocol allows as many packets as the size of the receive window to arrive out of order and be kept until there is a set of consecutive packets to be delivered to the application layer.
- Because the sizes of the send window and receive window are the same, all the packets in the send packet can arrive out of order and be stored until they can be delivered.
  the receiver never delivers packets out of order to the application layer

Selective Repeat ARQ

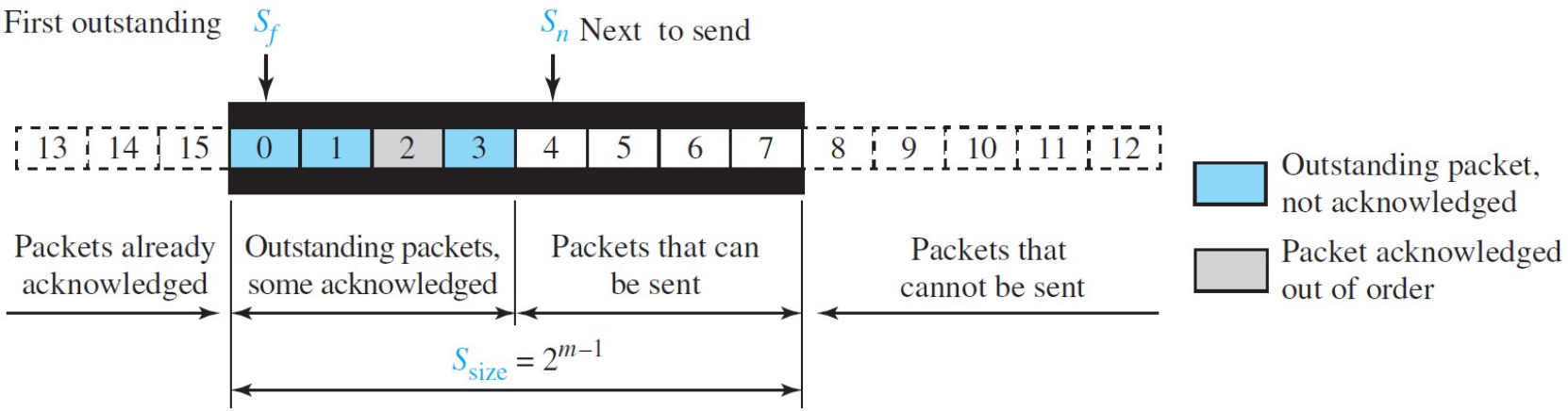


Figure 7 : Send window for Selective Repeat Protocol

Selective Repeat ARQ

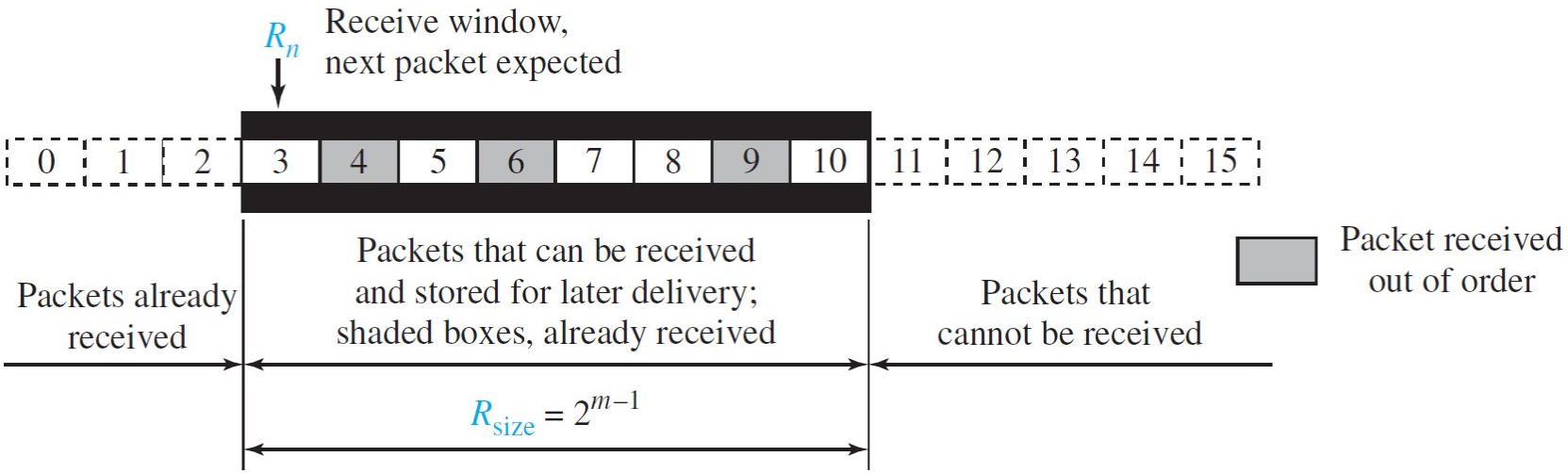


Figure 8 : Receive window for Selective Repeat Protocol

Selective Repeat ARQ

- Timer: Selective-Repeat uses one timer for each outstanding packet.
- When a timer expires, only the corresponding packet is resent.
  In other words, Go-Back-N treats outstanding packets as a group; Selective Repeat treats them individually.

Selective Repeat ARQ

- Acknowledgments: In Go-Back-N, an *ackNo* is cumulative; it defines the sequence number of the next packet expected, confirming that all previous packets have been received safe and sound.
- In Selective Repeat, an *ackNo* defines the sequence number of a single packet that is received safe and sound; there is no feedback for any other.

Selective Repeat ARQ

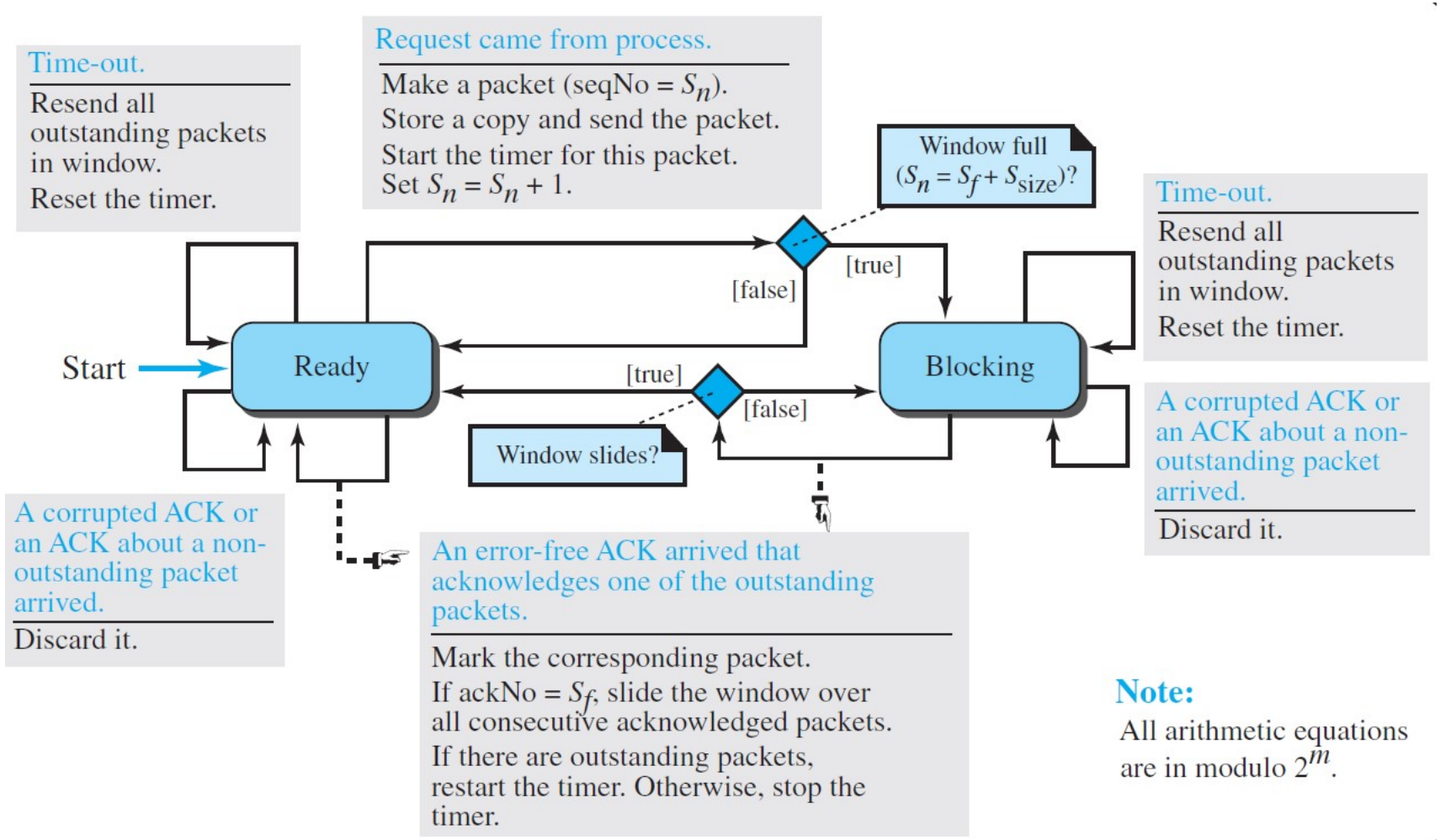


Figure 9 : Selective Repeat FSM for Sender

Selective Repeat ARQ

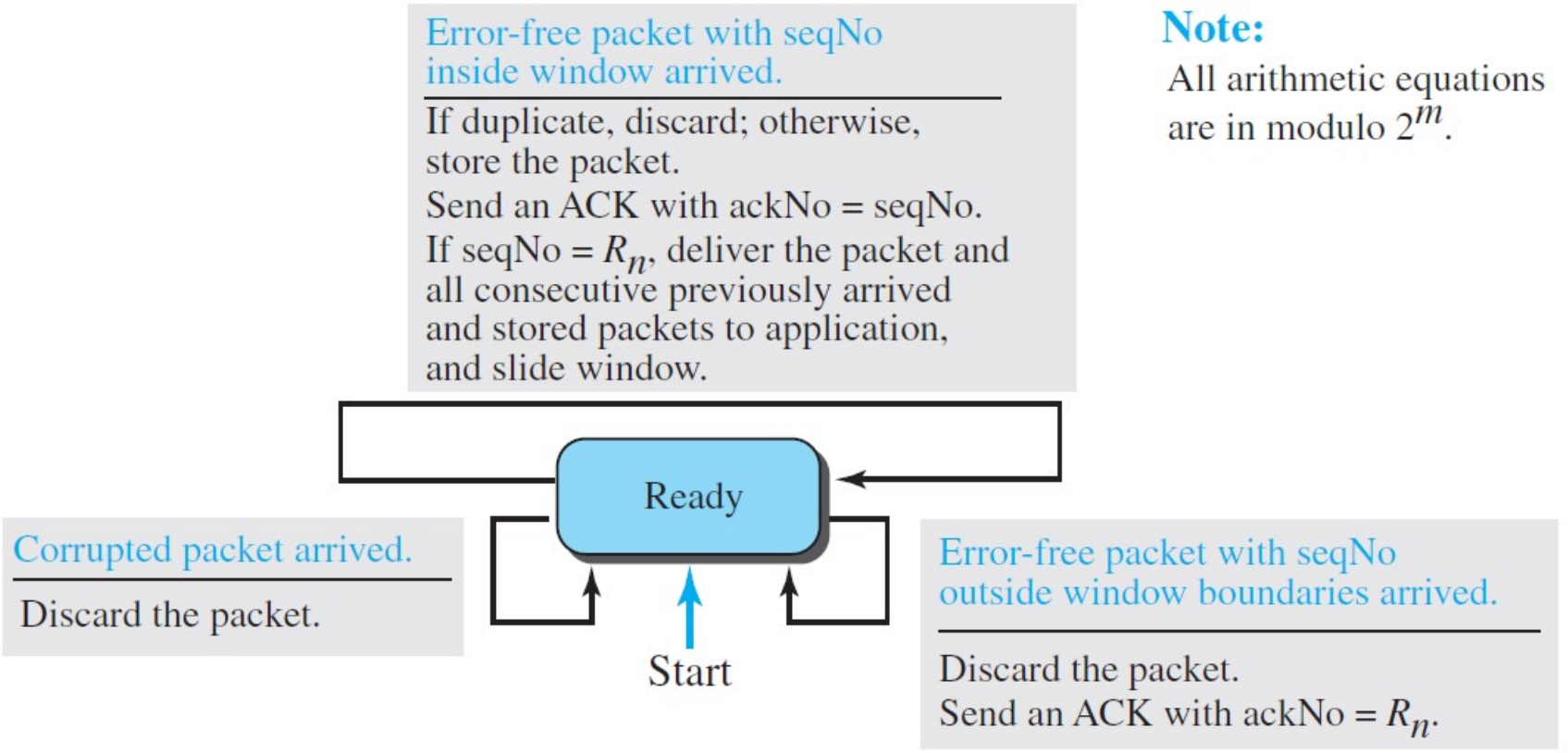


Figure 10 : Selective Repeat FSM for receiver

- Sender: The sender starts in the ready state, but later it can be in one of the two states: ready or blocking. The following shows the events and the corresponding actions in each state.
- Ready state: Four events may occur in this case;
  - If a request comes from the application layer, the sender creates a packet with the sequence number set to S_n. A copy of the packet is stored, and the packet is sent. If the timer is not running, the sender starts the timer. The value of S_n is now incremented, $S_n = (S_n + 1) modulo 2^m$. If the window is full, $S_n = (S_f + S_{size}) modulo 2^m$, the sender goes to the blocking state.
  - If an error-free ACK arrives with *ackNo* related to one of the outstanding packets, that packet is marked as acknowledged. If the $ackNo = S_f$, the window slides to the right until the S_f points to the first unacknowledged packet (all consecutive acknowledged packets are now outside the window). If there are outstanding packets, the timer is restarted; otherwise, the timer is stopped.
  - If a corrupted ACK or an error-free ACK with *ackNo* not related to an outstanding packet arrives, it is discarded.
  - If a time-out occurs, the sender resends all unacknowledged packets in the window and restarts the timer.

- Blocking state: Three events may occur in this case;

  a. If an error-free ACK arrives with ackNo related to one of the outstanding packets, that packet is marked as acknowledged. In addition, if the ackNo = S_f, the window is slid to the right until the S_f points to the first unacknowledged packet (all consecutive acknowledged packets are now outside the window). If the window has slid, the sender moves to the ready state.

  b. If a corrupted ACK or an error-free ACK with the ackNo not related to outstanding packets arrives, the ACK is discarded.

  c. If a time-out occurs, the sender resends all unacknowledged packets in the window and restarts the timer.

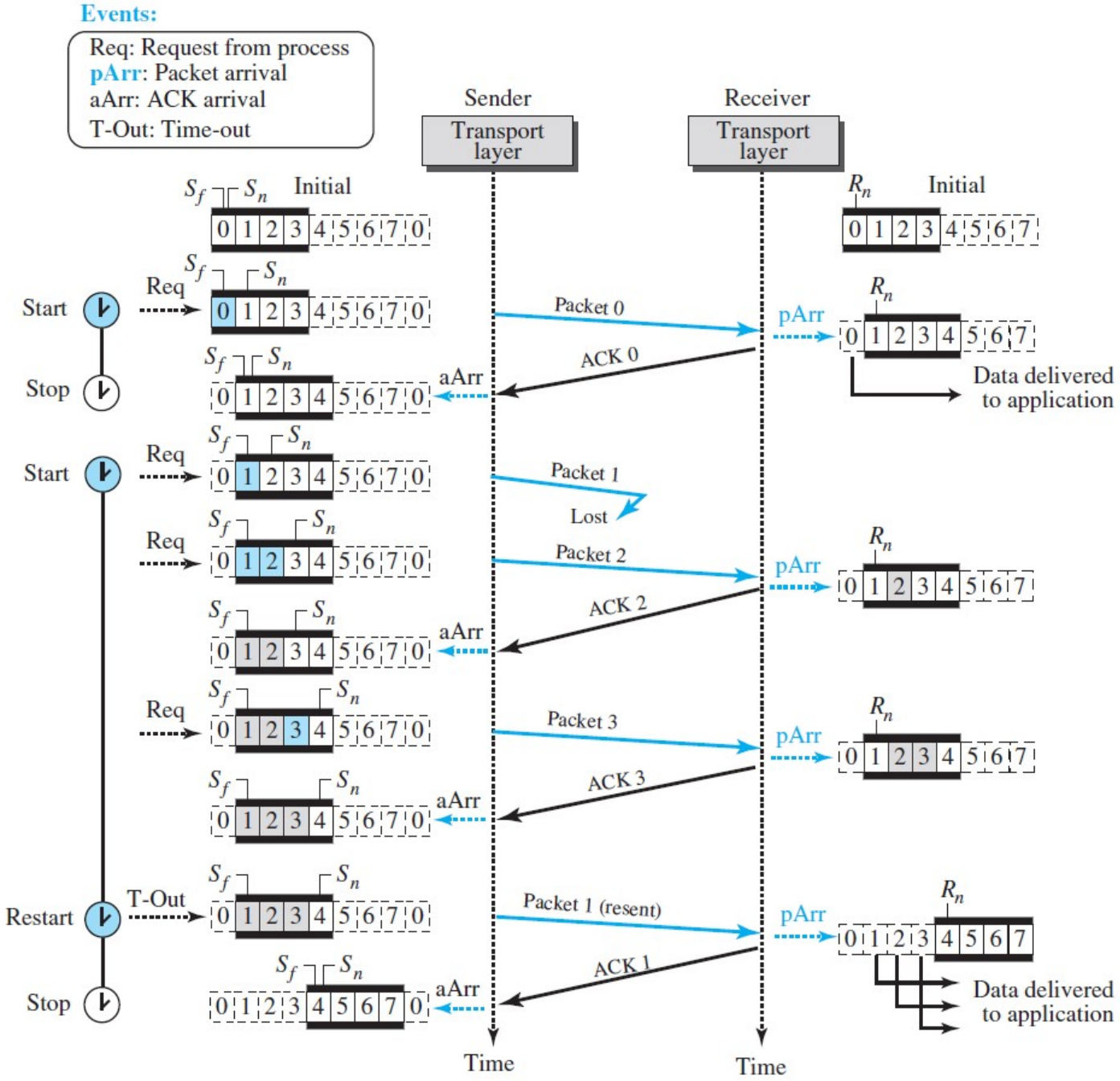
- Receiver: The receiver is always in the ready state. Three events may occur;

  a. If an error-free packet with seqNo in the window arrives, the packet is stored and an ACK with ackNo = seqNo is sent. In addition, if the seqNo = R_n, then the packet and all previously arrived consecutive packets are delivered to the application layer and the window slides so that the R_n points to the first empty slot.

  b. If an error-free packet with seqNo outside the window arrives, the packet is discarded, but an ACK with ackNo = R_n is returned to the sender. This is needed to let the sender slide its window if some ACKs related to packets with seqNo < R_n were lost.

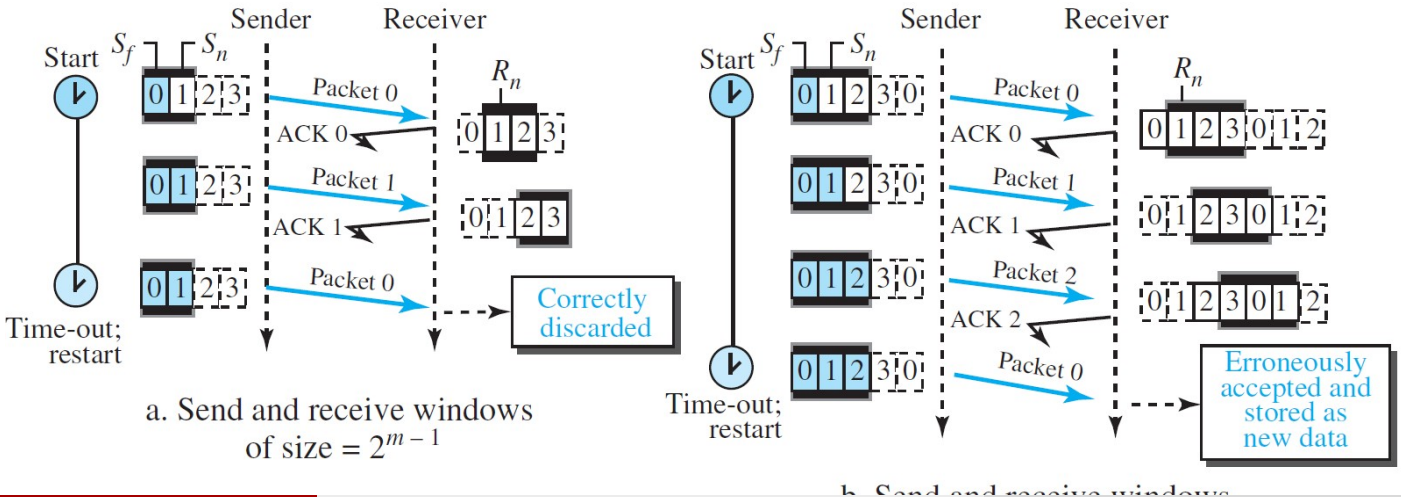
  c. If a corrupted packet arrives, the packet is discarded.

- Example1: At the sender, packet 0 is transmitted and acknowledged. Packet 1 is lost. Packets 2 and 3 arrive out of order and are acknowledged. When the timer times out, packet 1 (the only unacknowledged packet) is resent and is acknowledged. The send window then slides. At the second arrival, packet 2 arrives and is stored and marked (shaded slot), but it cannot be delivered because packet 1 is missing. At the next arrival, packet 3 arrives and is marked and stored, but still none of the packets can be delivered. Only at the last arrival, when finally a copy of packet 1 arrives, can packets 1, 2, and 3 be delivered to the application layer.


Events:
Req: Request from process
pArr: Packet arrival
aArr: ACK arrival
T-Out: Time-out
Sender
Transport layer
Receiver
Transport layer
Initial
Start
Stop
Req
Packet 0
ACK 0
pArr
aArr
Data delivered to application
Packet 1
Lost
Packet 2
ACK 2
Packet 3
ACK 3
T-Out
Restart
Packet 1 (resent)
ACK 1
Time
Time

- In Selective-Repeat, the size of the sender and receiver window can be at most one-half of 2^m.
- For example, we choose $m = 2$, which means the size of the window is $\frac{2^m}{2}$ or $2^{(m-1)} = 2$.

  Window size of 2 with a window size of 3 is compared in the Figure.
- If the size of the window is 2 and all acknowledgments are lost, the timer for packet 0 expires and packet 0 is resent. However, the window of the receiver is now expecting packet 2, not packet 0, so this duplicate packet is correctly discarded (the sequence number 0 is not in the window).
- When the size of the window is 3 and all acknowledgments are lost, the sender sends a duplicate of packet 0. However, this time, the window of the receiver expects to receive packet 0 (0 is part of the window), so it accepts packet 0, not as a duplicate, but as a packet in the next cycle. This is clearly an error.



a. Send and receive windows of size = 2^{m-1}

Access Control Protocols

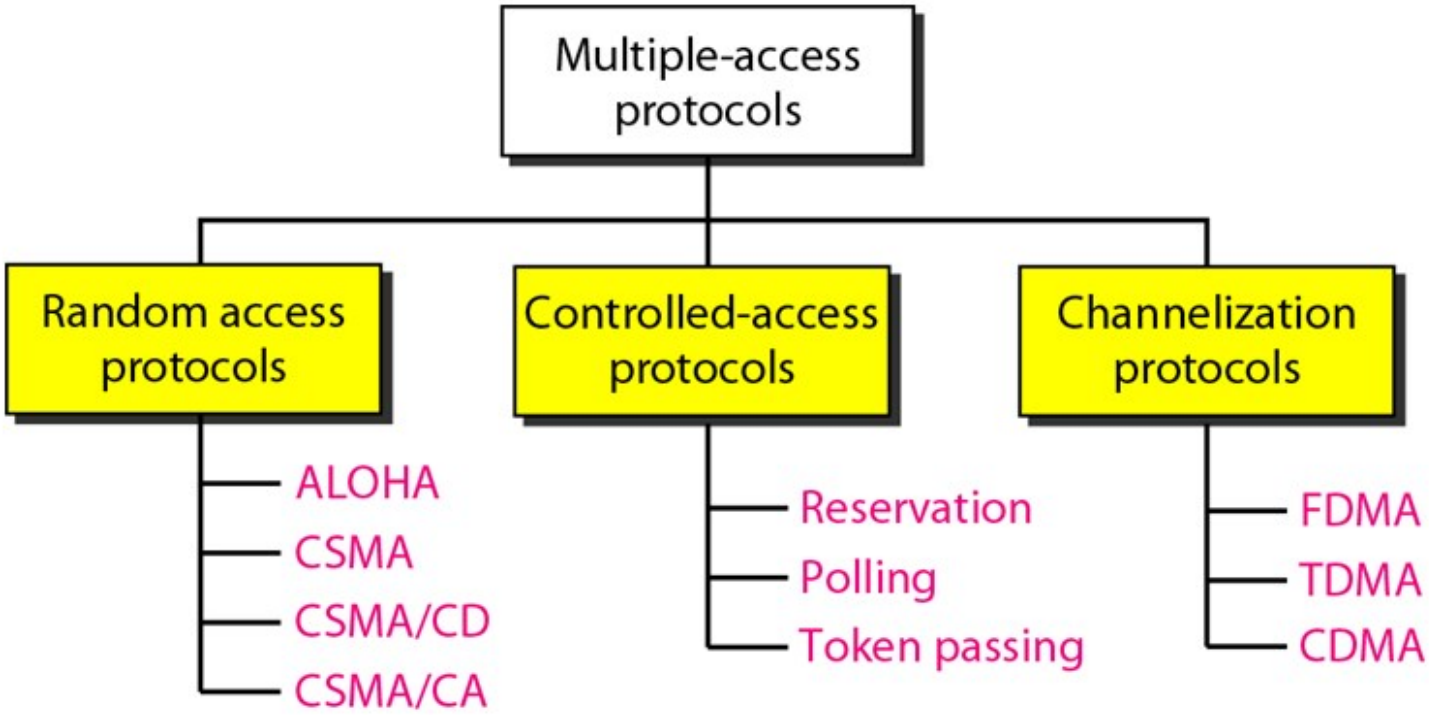


Figure 12 : Taxonomy of multiple-access protocols

Access Control Protocols

- Random Access: In a random access method, each station has the right to the medium without being controlled by any other station.
- However, if more than one station tries to send, there is an access conflict (collision) and the frames will be either destroyed or modified.
- To avoid access conflict, we need a procedure that answers the following;
  (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 transmission?
  (4) What can the station do if there is any access conflict?

Access Control Protocols

- ALOHA, the earliest random access method, was developed at the University of Hawaii in the early 1970s.
- Here,a base station is a central controller. Every station that needs to send a frame to another station first sends it to the base station.
- The base station receives the frame and relays it to the intented destination. (i.e) the base station act as a hop.
  Figure13 shows the basic idea behind an ALOHA network.

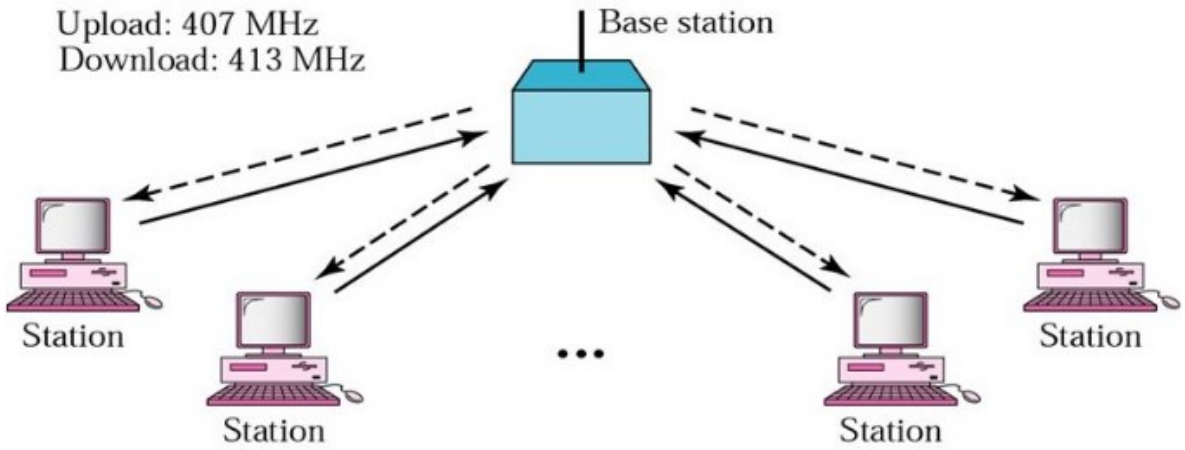


Figure 13 : ALOHA Network

Access Control Protocols

- The uploading transmission(from station to base station) uses modulation with a carrier frequency of 407MHz and downloading transmission(from base station to any station) uses a carrier frequency of 413MHz.
  It is obvious that there are potential collisions in this arrangement.
- When a station sends data at frequency 407MHz to base station, another station may attempt to do so at the same time. So, the data from the two stations collide.

Access Control Protocols

- The ALOHA protocol is very simple and it based on the following rules:
  (1) Multiple Access: Any station sends a frame when it has a frame to send.
  (2) Acknowledgement: After sending a frame, the station waits for an acknowledgemnt. If it does not receive an ACK during the allotted time, which is 2 times the maximum propagation delay, it assumes that the frame is lost; it tries sending again after a random amount of time.

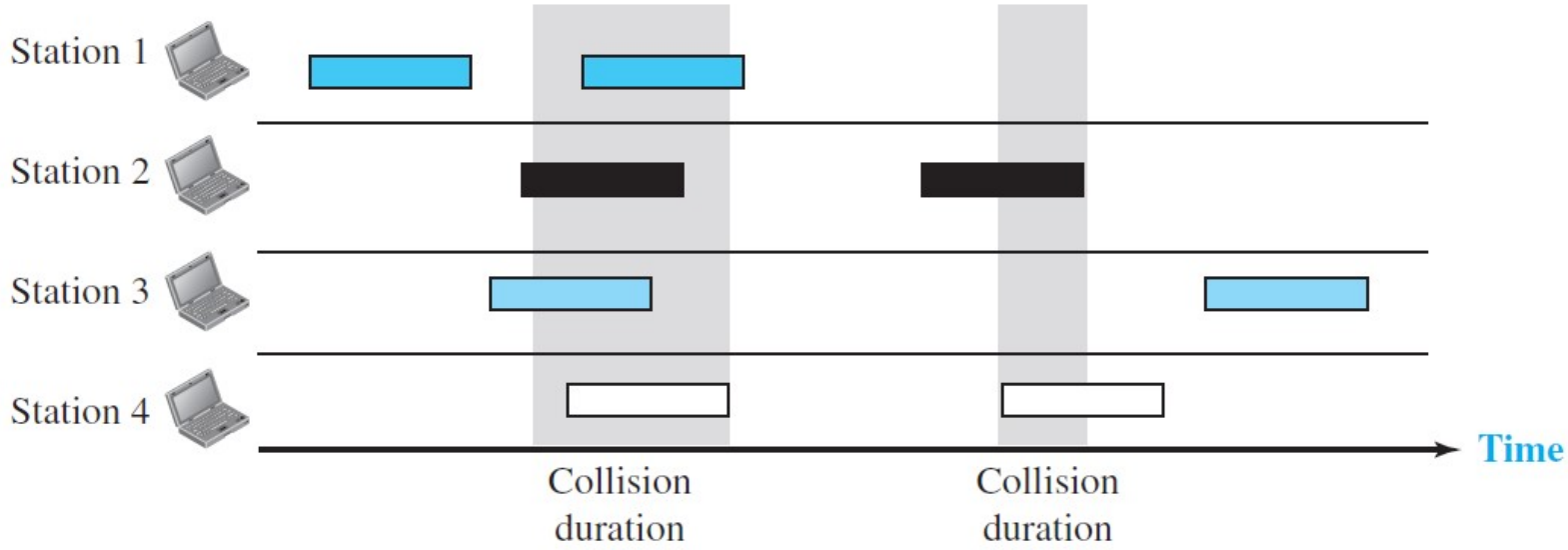


Figure 14 : Frames in a pure ALOHA network

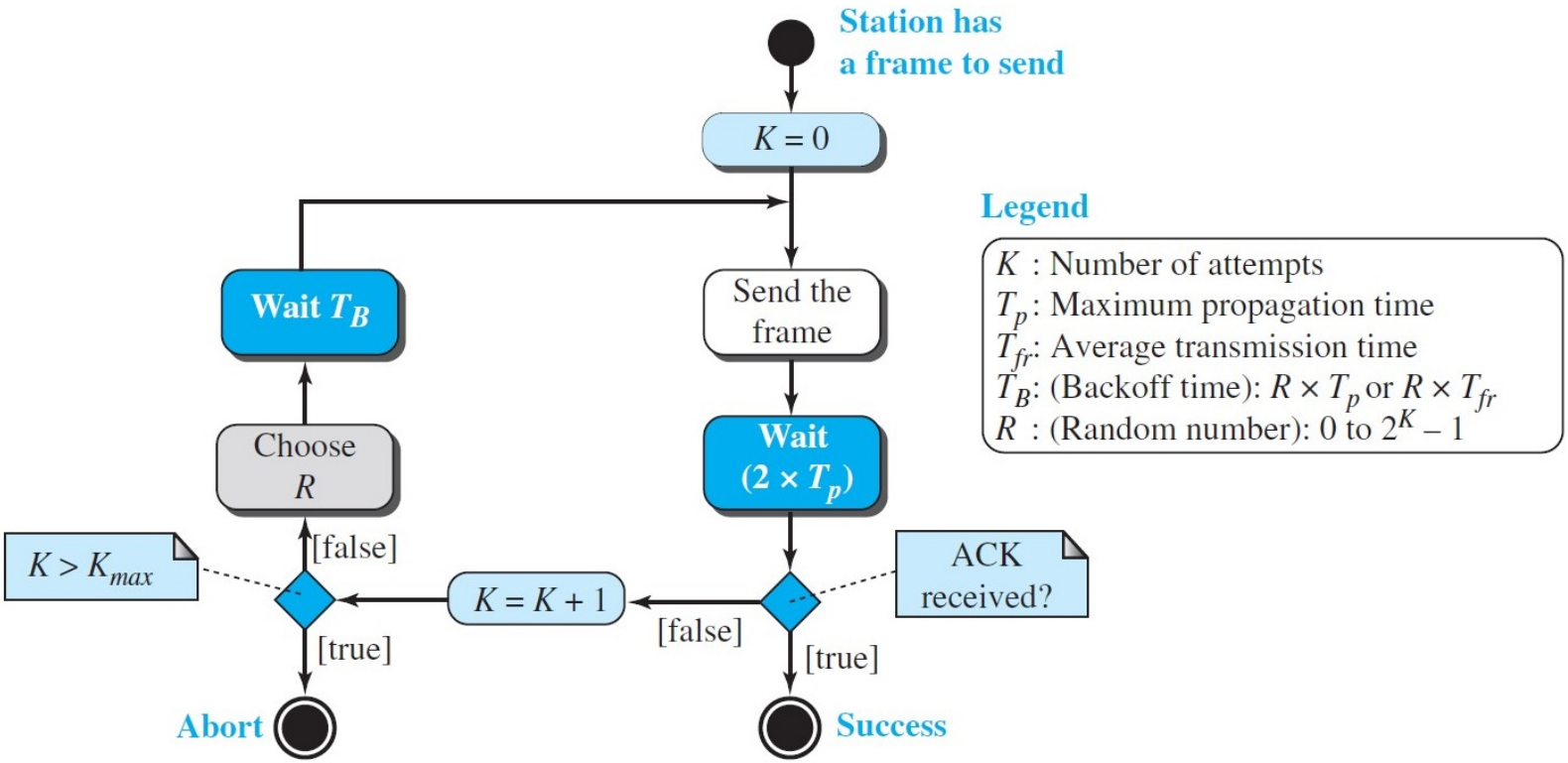


Figure 15 : Procedure for pure ALOHA protocol

Access Control Protocols

- Flowchart of pure ALOHA is shown in Figure15.
- A station that has a frame to send sends it. It then waits for a period of time, which is 2 times the maximum propagation delay.
- If it receives an ACK, the transmission is successful, otherwise the station uses a backoff strategy and sends the packet again.
- After several tries, if there is no ACK, the station gives up.

- Vulnerable time: The length of time in which there is a possibility of collision.
- Assume that the stations send fixed-length frames with each frame taking T_{fr} seconds to send.
- Figure shows the vulnerable time for station B.

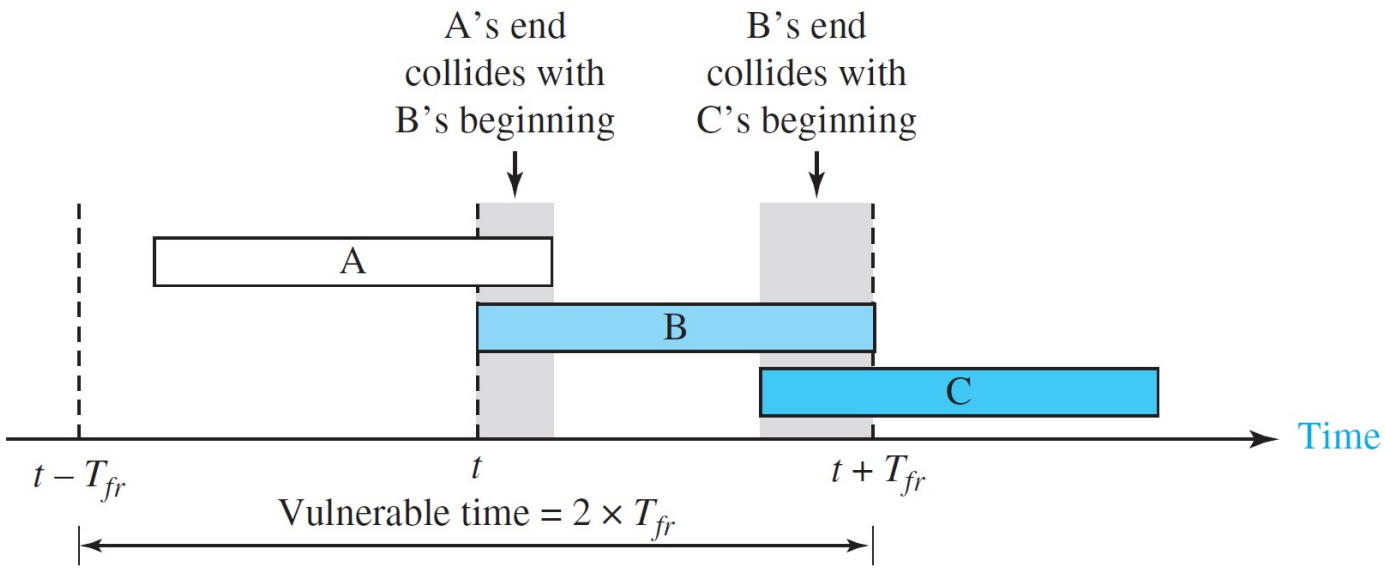


Figure 16 : Vulnerable time for pure ALOHA protocol

- 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 also, there is a collision between frames from station B and station C.
- The vulnerable time during which a collision may occur in pure ALOHA is 2 times the frame transmission time.

$$Pure\ ALOHA\ vulnerable\ time = 2 \times T_{fr}$$

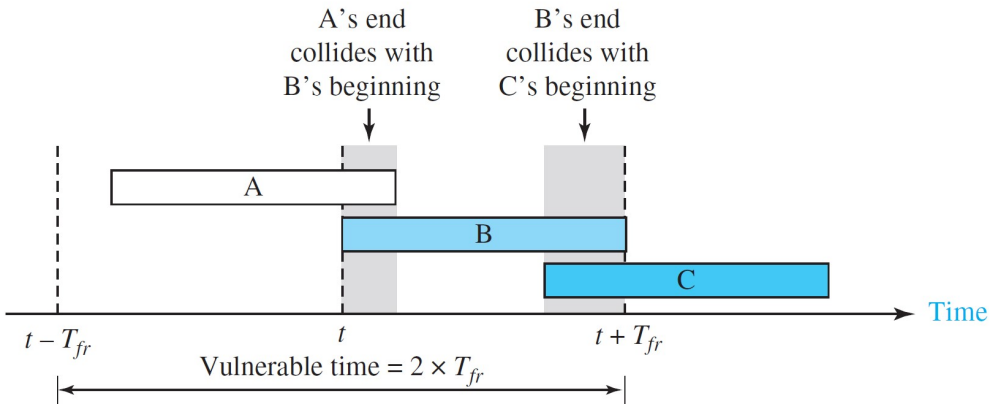


Figure 17 : Vulnerable time for pure ALOHA protocol

- Throughput: Let G be the average number of frames generated by the system during one frame transmission time.
- Then, the average number of successfully transmitted frames for pure ALOHA is

$$S = G \times e^{-2G}$$

  The maximum throughput S_{max} is 0.184 for $G = \frac{1}{2}$
- In other words, if one-half a frame is generated during one frame transmission time (one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully.
- Therefore, if a station generates only one frame in this vulnerable time (and no other stations generate a frame during this time), the frame will reach its destination successfully.
- We expect $G = \frac{1}{2}$ to produce the maximum throughput because the vulnerable time is 2 times the frame transmission time.

The throughput for pure ALOHA is $S = G \times e^{-2G}$.
The maximum throughput $S_{max} = 1/(2e) = 0.184$ when $G = (1/2)$.

- Exercise1:

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the throughput if the system (all stations together) produces

a. 1000 frames per second?

b. 500 frames per second?

c. 250 frames per second?

- Exercise1:

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the throughput if the system (all stations together) produces

a. 1000 frames per second?

b. 500 frames per second?

c. 250 frames per second?

- Solution

The frame transmission time is 200/200 kbps or 1 ms.

a. If the system creates 1000 frames per second, or 1 frame per millisecond, then $G = 1$. In this case $S = G \times e^{-2G} = 0.135$ (13.5 percent). This means that the throughput is $1000 \times 0.135 = 135$ frames. Only 135 frames out of 1000 will probably survive.

b. If the system creates 500 frames per second, or 1/2 frames per millisecond, then $G = 1/2$. In this case $S = G \times e^{-2G} = 0.184$ (18.4 percent). This means that the throughput is $500 \times 0.184 = 92$ and that only 92 frames out of 500 will probably survive. Note that this is the *maximum* throughput case, percentagewise.

c. If the system creates 250 frames per second, or 1/4 frames per millisecond, then $G = 1/4$. In this case $S = G \times e^{-2G} = 0.152$ (15.2 percent). This means that the throughput is $250 \times 0.152 = 38$. Only 38 frames out of 250 will probably survive.

Access Control Protocols

- Slotted ALOHA: Pure ALOHA has a vulnerable time of $2 \times T_{fr}$.
  This is because there is no rule that defines when the station can send.
  A station may send soon after another station has started or just before another station has finished.
- Slotted ALOHA was invented to improve the efficiency of pure ALOHA.
- In slotted ALOHA we divide the time into slots of T_{fr} seconds and force the station to send only at the beginning of the time slot.

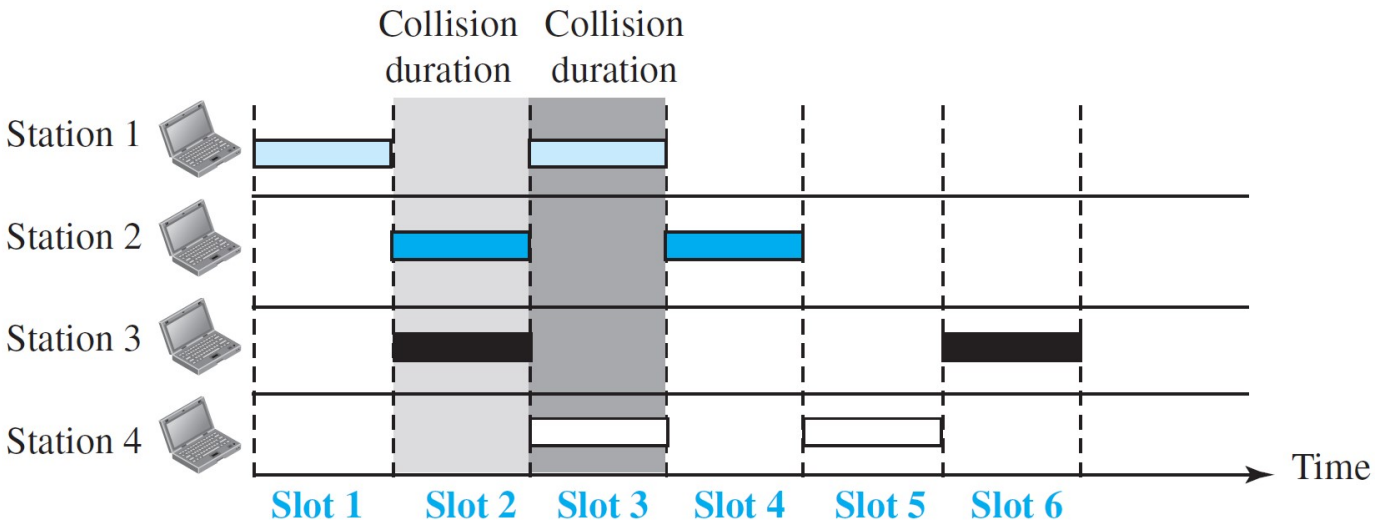


Figure 18 : Frames in a slotted ALOHA network

- 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}

$$\textit{Slotted ALOHA vulnerable time} = T_{fr}$$

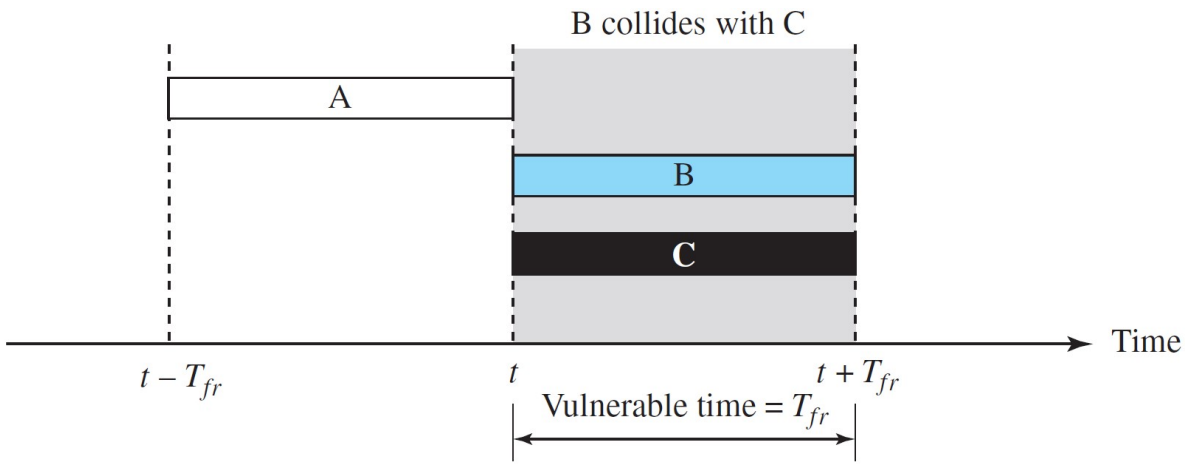


Figure 19 : Vulnerable time for slotted ALOHA protocol

Access Control Protocols

- Throughput: The average number of successful transmissions for slotted ALOHA is $S = G \times e^{-G}$. The maximum throughput S_{max} is 0.368, when $G = 1$.
- In other words, if one frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully.
- Therefore, if a station generates only one frame in this vulnerable time (and no other station generates a frame during this time), the frame will reach its destination successfully.
- We expect $G = 1$ to produce maximum throughput because the vulnerable time is equal to the frame transmission time.

The throughput for slotted ALOHA is $S = G \times e^{-G}$.
The maximum throughput $S_{max} = 0.368$ when $G = 1$.

- Exercise2:

A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200-kbps bandwidth. Find the throughput if the system (all stations together) produces

a. 1000 frames per second.

b. 500 frames per second.

c. 250 frames per second.

- Exercise2:

A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200-kbps bandwidth. Find the throughput if the system (all stations together) produces

a. 1000 frames per second.

b. 500 frames per second.

c. 250 frames per second.

- Solution

The frame transmission time is 200/200 kbps or 1 ms.

a. In this case G is 1. So $S = G \times e^{-G} = 0.368$ (36.8 percent). This means that the throughput is $1000 \times 0.0368 = 368$ frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.

b. Here G is 1/2. In this case $S = G \times e^{-G} = 0.303$ (30.3 percent). This means that the throughput is $500 \times 0.0303 = 151$. Only 151 frames out of 500 will probably survive.

c. Now G is 1/4. In this case $S = G \times e^{-G} = 0.195$ (19.5 percent). This means that the throughput is $250 \times 0.195 = 49$. Only 49 frames out of 250 will probably survive.

Carrier Sense Multiple Access(CSMA)

Access Control Protocols

- To minimize the chance of collision and increase the performance, the Carrier Sense Multiple Access(CSMA) method was developed.
- The chance of collision can be reduced if a station senses the medium before trying to use it.
- CSMA requires that each station first listen to the medium before sending.
- CSMA is evolved into
  - CSMA/CA
  - CSMA/CD
- CSMA/CD defines procedures to be followed if a collision is detected while CSMA/CA defines procedures to avoid collision.

Access Control Protocols

- CSMA can reduce the possibility of collision, but it cannot eliminate it. The possibility of collision still exists due to the propagation delay. (i.e) when a station sends a frame, it takes a while for the first bit to reach every station and for every station to sense it.

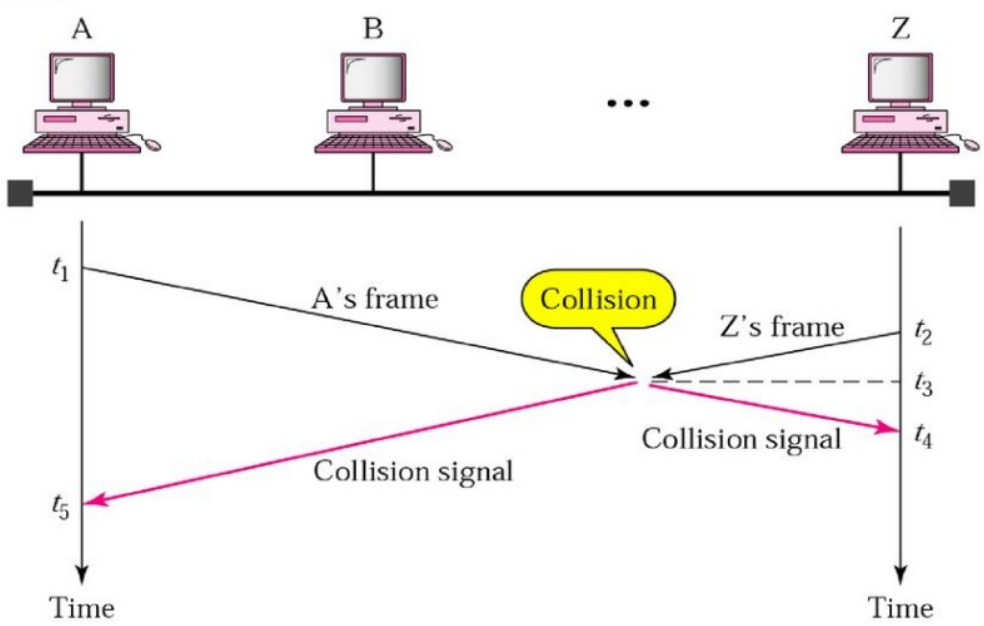


Figure 20 : Collision in CSMA

- At time t1, station A sense the medium. Since the medium is idle, it sends a frame.
- At time t2, station Z senses the medium and finds it idle because, at this time propagation from station A has not reached station Z. So, station Z also sends a frame.
- The two signals collide at time t3. The result of the collision, which is a garbled signal, will also propagate now in both directions.
- It reaches station Z at time t4 and station A at time t5.

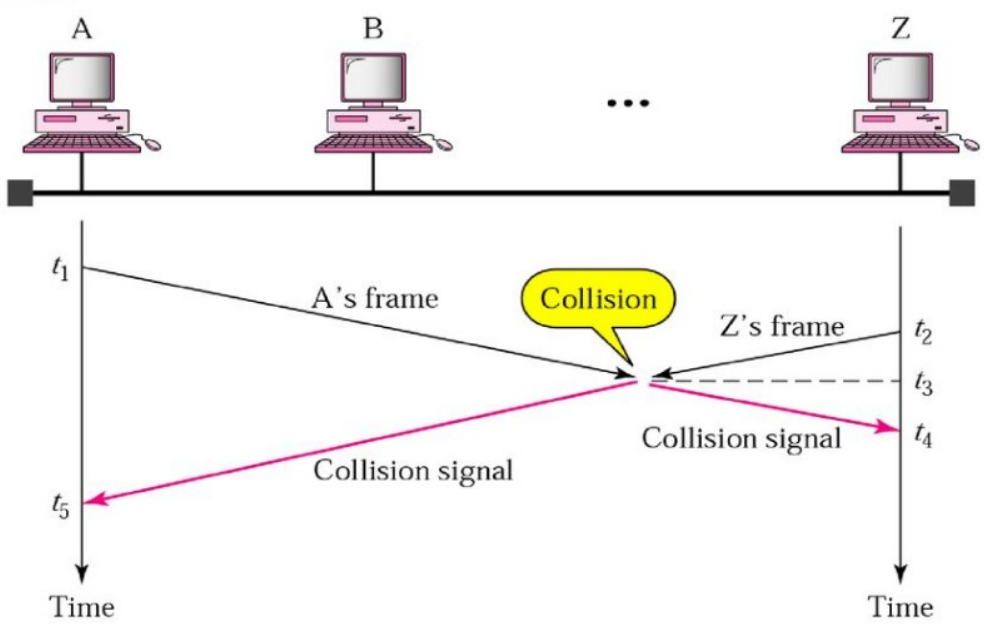


Figure 21 : Collision in CSMA

Access Control Protocols

- Persistence Strategy defines the procedures for a station that senses a busy medium.
- Two substrategies have been devised;
  (1) non persistent
  (2) persistent

Access Control Protocols

- In a nonpersistent strategy a station has a frame to send senses the line.
  - If the line is idle, the station sends immediately.
  - If the line is not idle, the station waits a random period of time and then senses the line again.

Access Control Protocols

- In a persistent strategy, a station senses the line. If the line is idle, the station sends a frame. This method has two variations
  (a) 1-persistent
  (b) p-persistent
- In the 1-persistent method, if the station finds the line idle, the station sends its frame immediately with a probability of 1.
  This method increases the chance of collision because two or more stations may send their frames after finding the line idle.
- In the p-persistent method, if the station finds the line idle, the station may or may not send. It sends with probability p and refrains from sending with probability $1 - p$
  For example, if p is 0.2, it means that each station after sensing an idle line, sends with a probability of 0.2(20 percent of the time) and refrains from sending with a probability of 0.8(80 percent of the time).

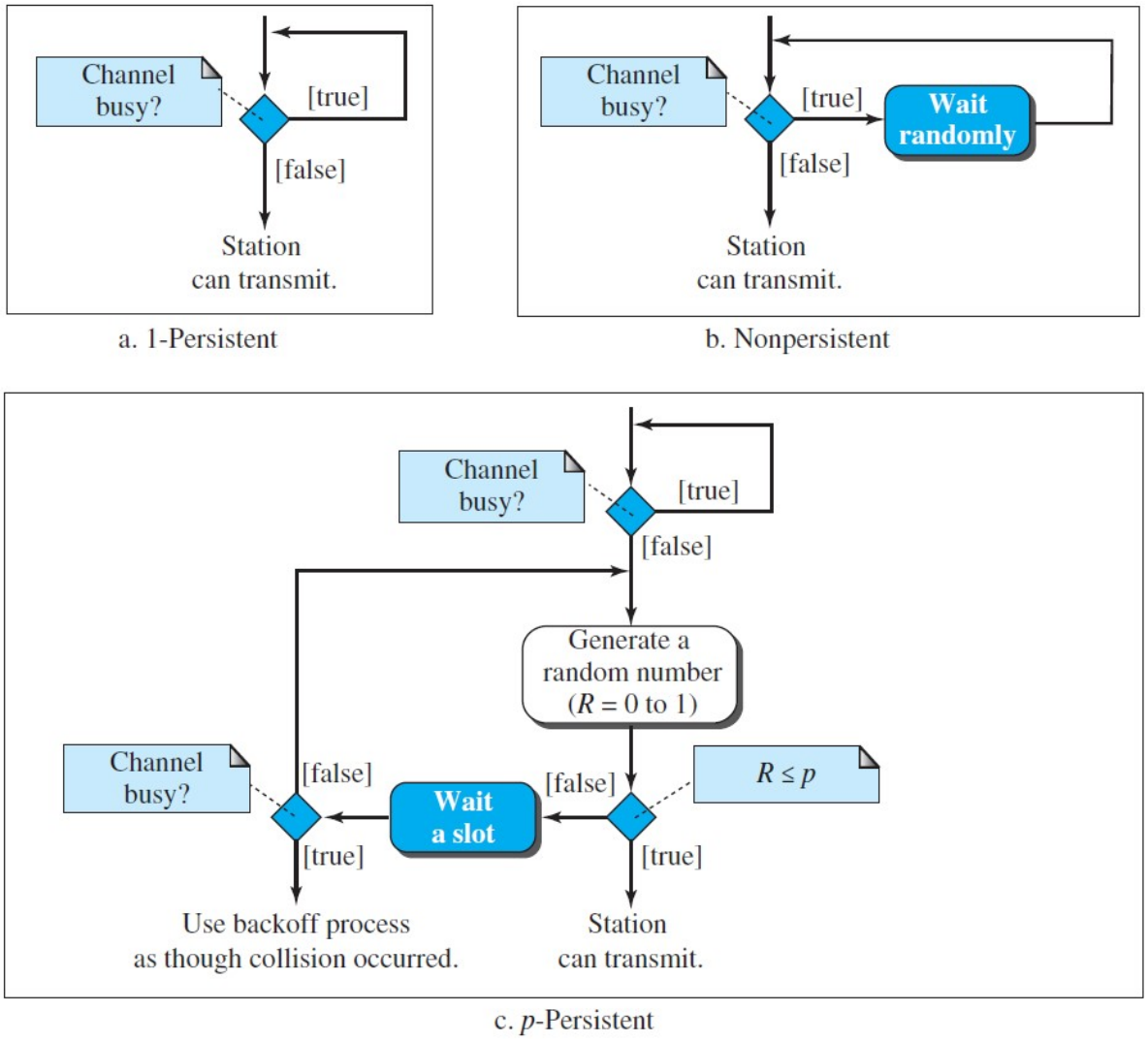


Figure 22 : Flow diagram for three persistence methods

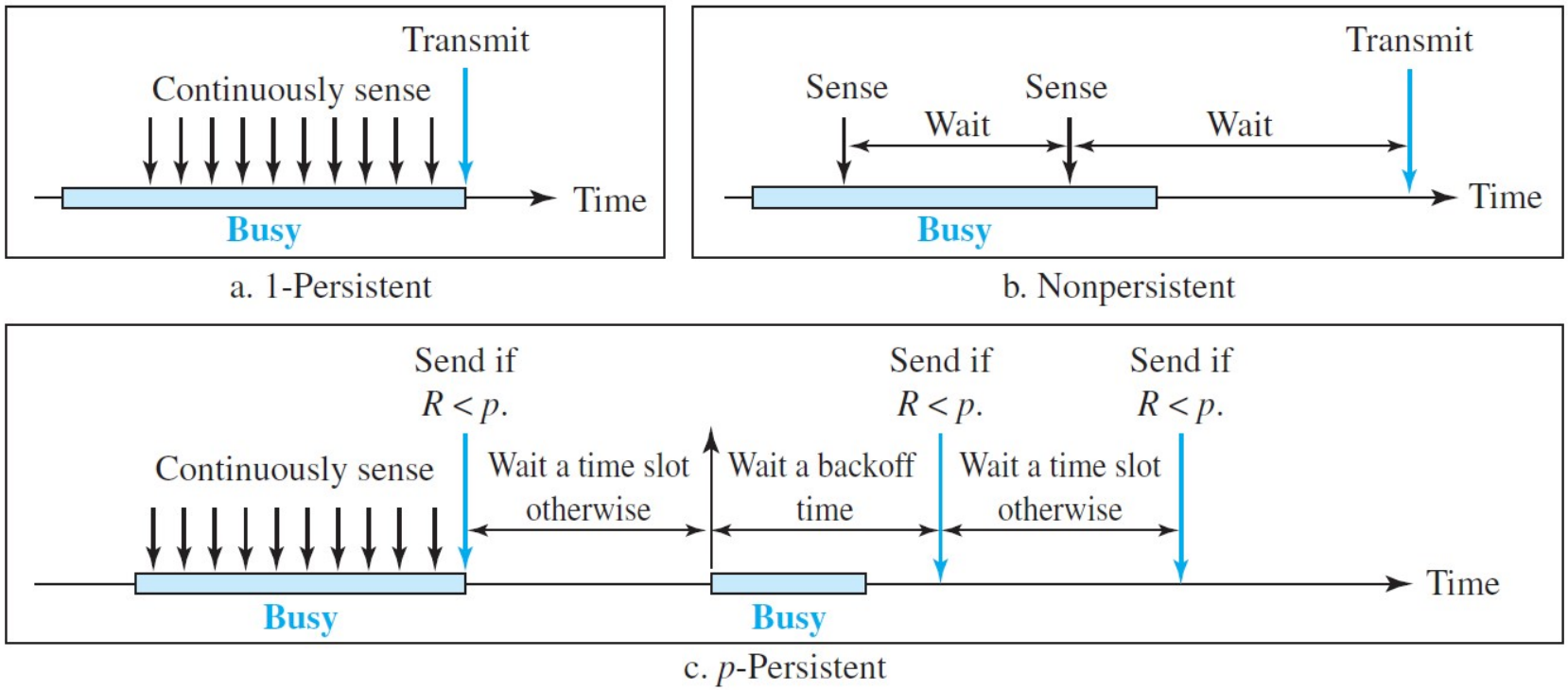


Figure 23 : Behavior of three persistence methods

Access Control Protocols

- CSMA/CD: In this method, any station can send a frame. The station then monitors the medium to see if transmission was successful. If so, the station is finished.
  If there was a collision, the frame need to be sent again.
- To reduce the probability of collision, the second time the station waits, it needs to back off.
  The question is, How much?
- It is reasonable that the station waits a little the first time, more if a collision occurs, much more if it happenns a third time and so on.
- In the exponential backoff method, the station waits an amount of time between 0 and $2^N \times$ (maximum propagation time)
  where N is the number of attempted transmissions.
  That is, it waits between 0 and $2 \times$ (maximum propagation time) for the first time, between 0 and $2^2 \times$ (maximum propagation time) for the second time and so on.

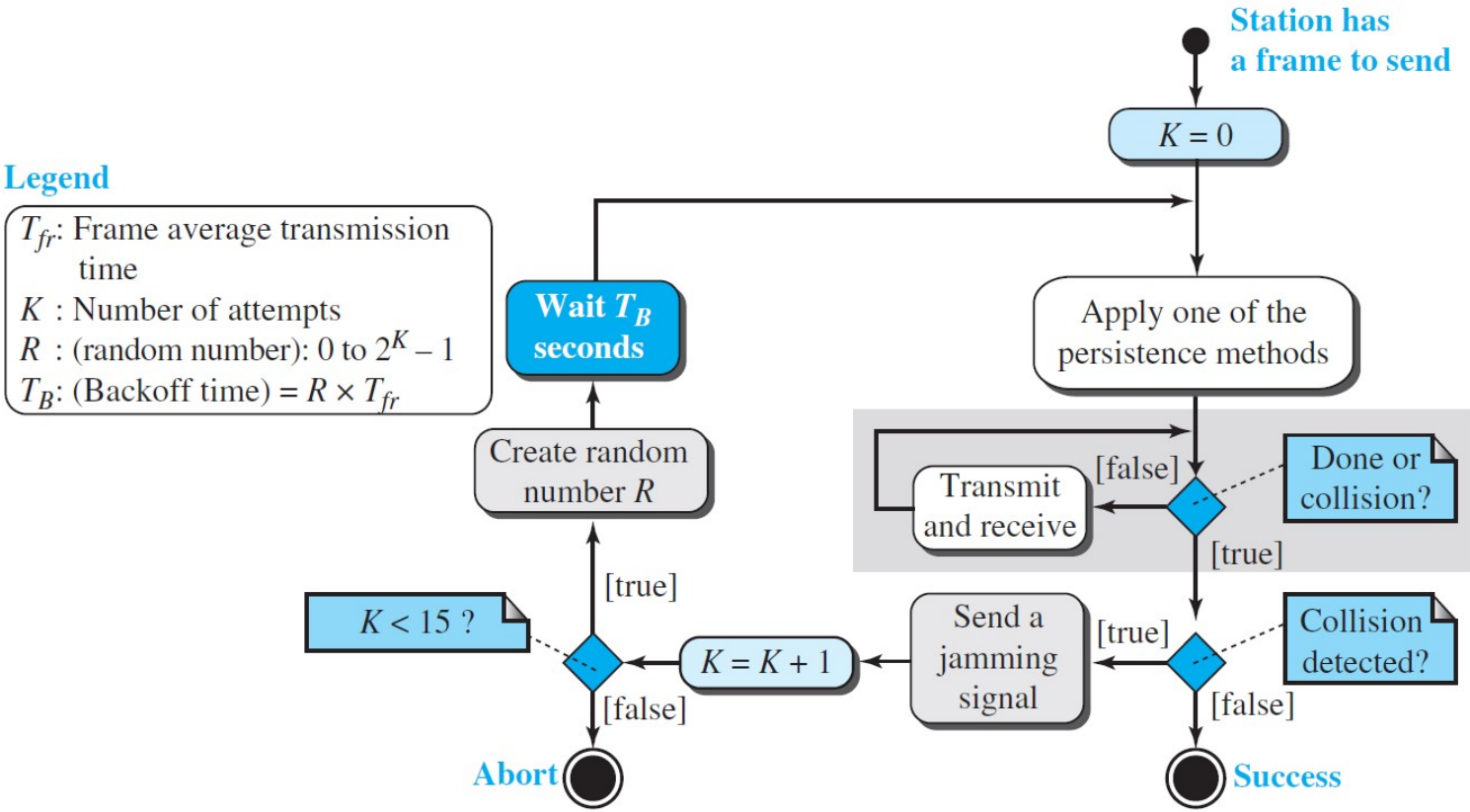


Figure 24 : Flow diagram for the CSMA/CD

- Energy Level: The energy level in the channel have three values;
  - Zero, when the channel is idle
  - Normal, when the station sending its frame
  - Abnormal, when there is a collision in the channel
- A station before sending its frame needs to monitor the energy level of the channel to determine if the channel is idle, busy or in collision mode.

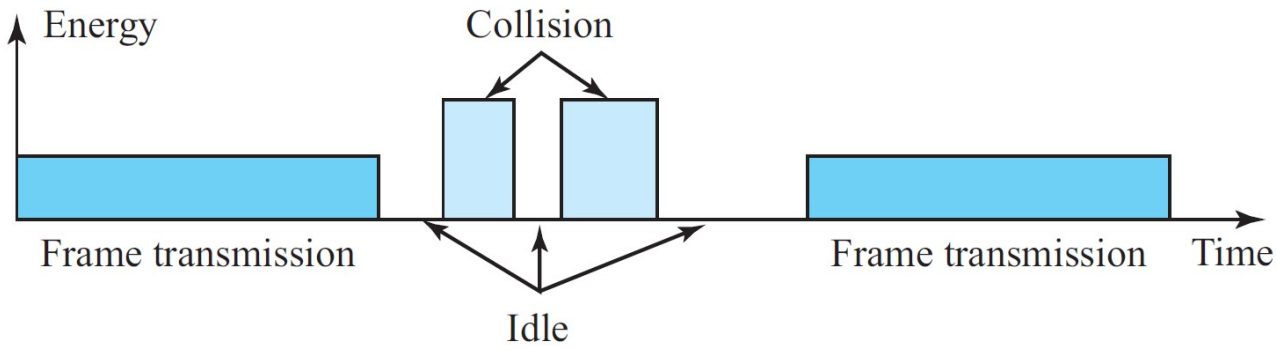


Figure 25 : Energy level during transmission, idleness or collision

Access Control Protocols

- CSMA/CA differs from the CSMA/CD procedures that avoids collision.
- Collisions are avoided through the use of three strategies;
  - the interframe space(IFS)
  - the contention window
  - acknowledgments

Access Control Protocols

- 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.
  - After waiting an IFS time, if the channel is still idle, the station can send, but it still needs to wait a time equal to the contention window.

Access Control Protocols

- 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.
    (i.e) This time window doubles with each collision and corresponds to the binary exponential backoff (BEB) that is familiar from CSMA/CD

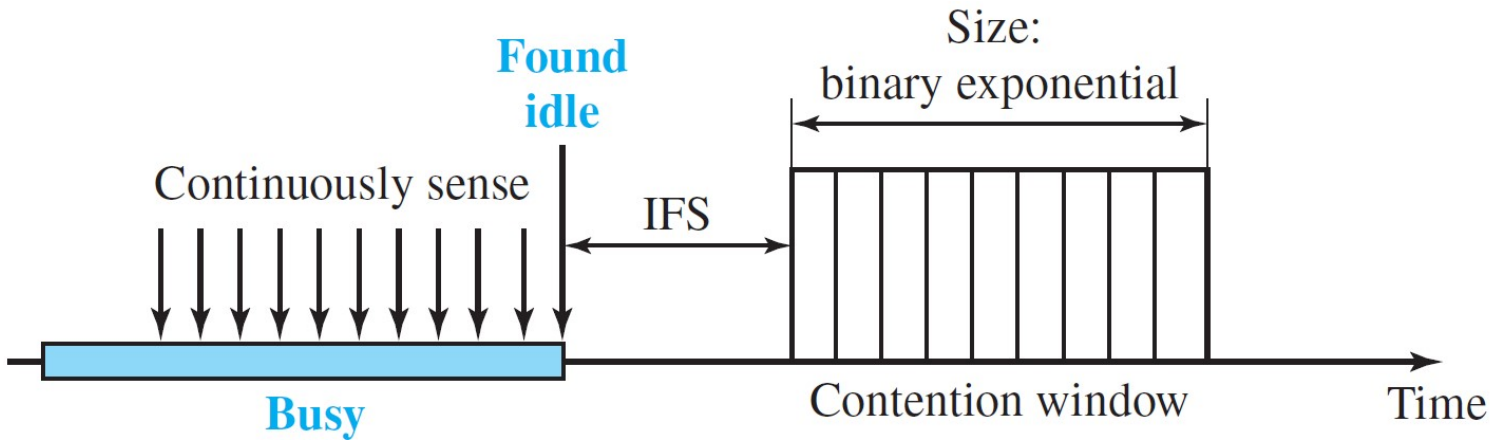


Figure 26 : Contention Window in CSMA/CA

Access Control Protocols

- Acknowledgment: With all these precautions, still there is a collision resulting in destroyed data.
- The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

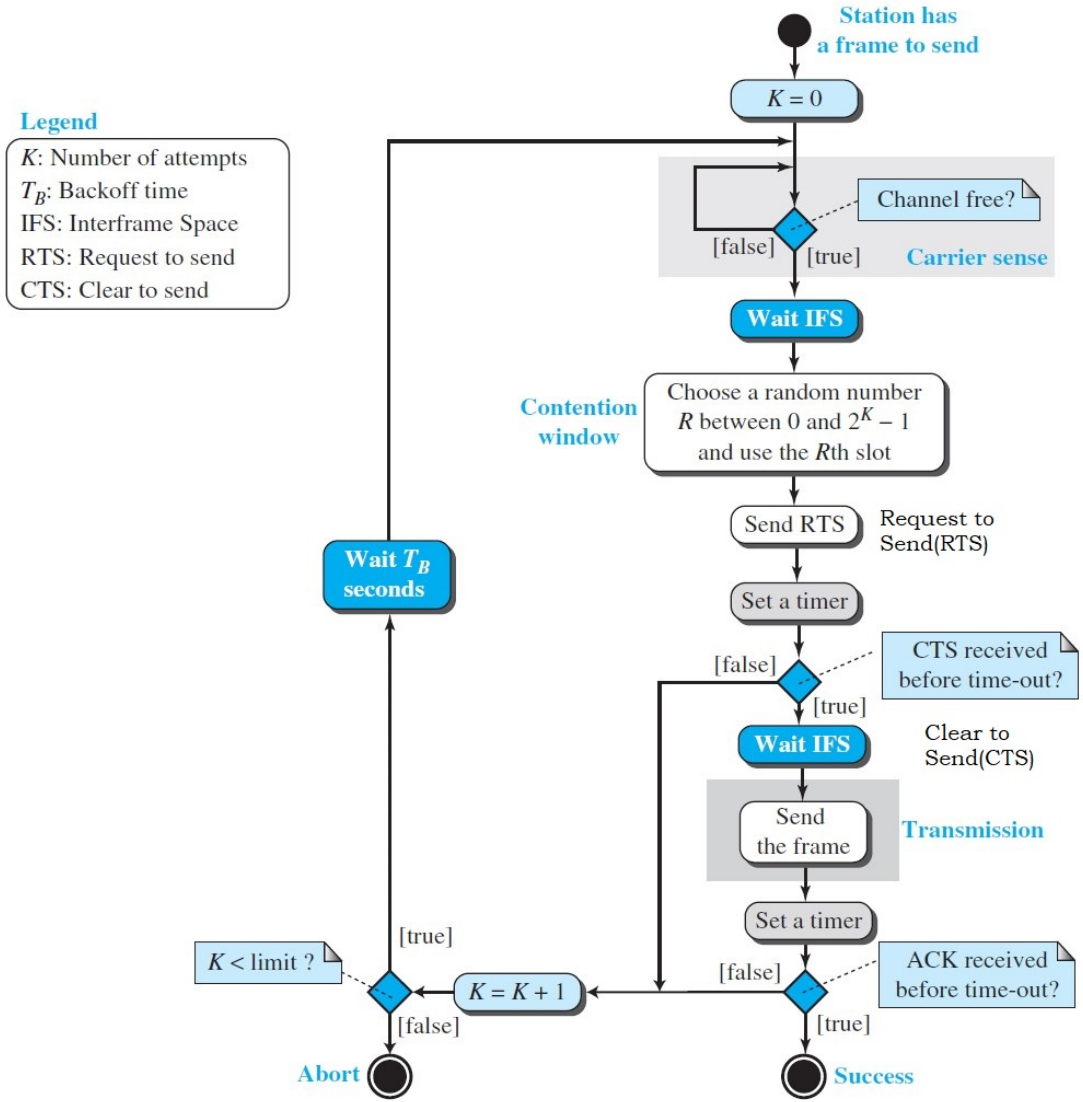


Figure 27 : Flow diagram for the CSMA/CA

Access Control Protocols

- Advantages of CSMA/CA:
  - CSMA/CA reduces the frequency of collisions, which helps ensure that devices can share a communication channel efficiently.
  - The RTS/CTS extension can help save on excessive data transmission.
  - CSMA/CA reduces the odds of collisions when data is large.

IEEE Standards

IEEE 802.11 Architecture

- Architecture of an infrastructure network includes;
  - Station: terminal with access mechanisms to the wireless medium and radio contact to the access point.
  - Basic Service Set(BSS): group of stations using the same radio frequency
  - Access Points: station integrarted into the wireless LAN and the distribution system
  - Portal:bridge to other(wired)networks
  - Distributed System: interconnection networks to form one logical network Extended Service Set(ESS) based on several BSS.

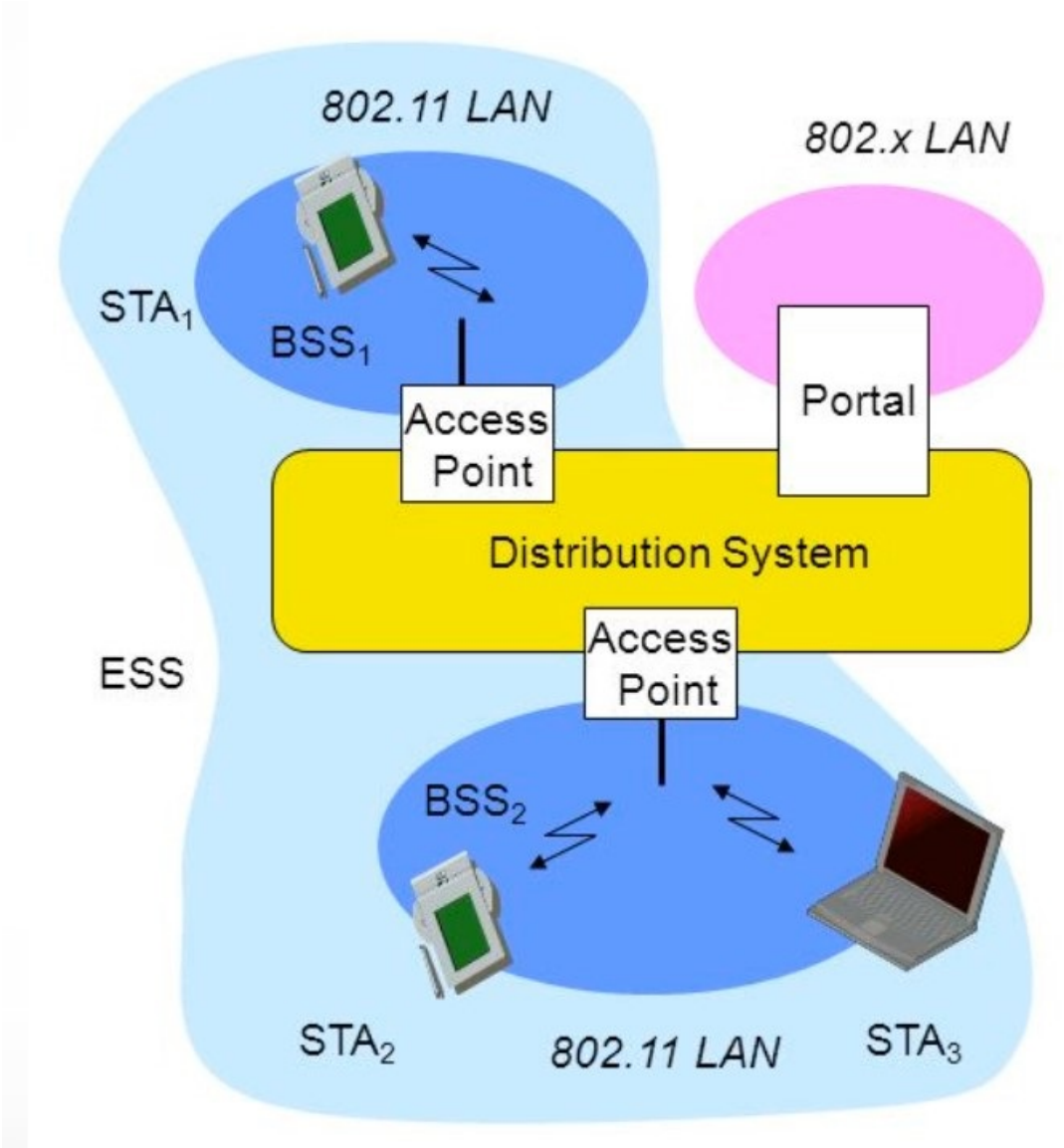


Figure 28 : IEEE 802.11 Architecture

IEEE 802.11 Architecture

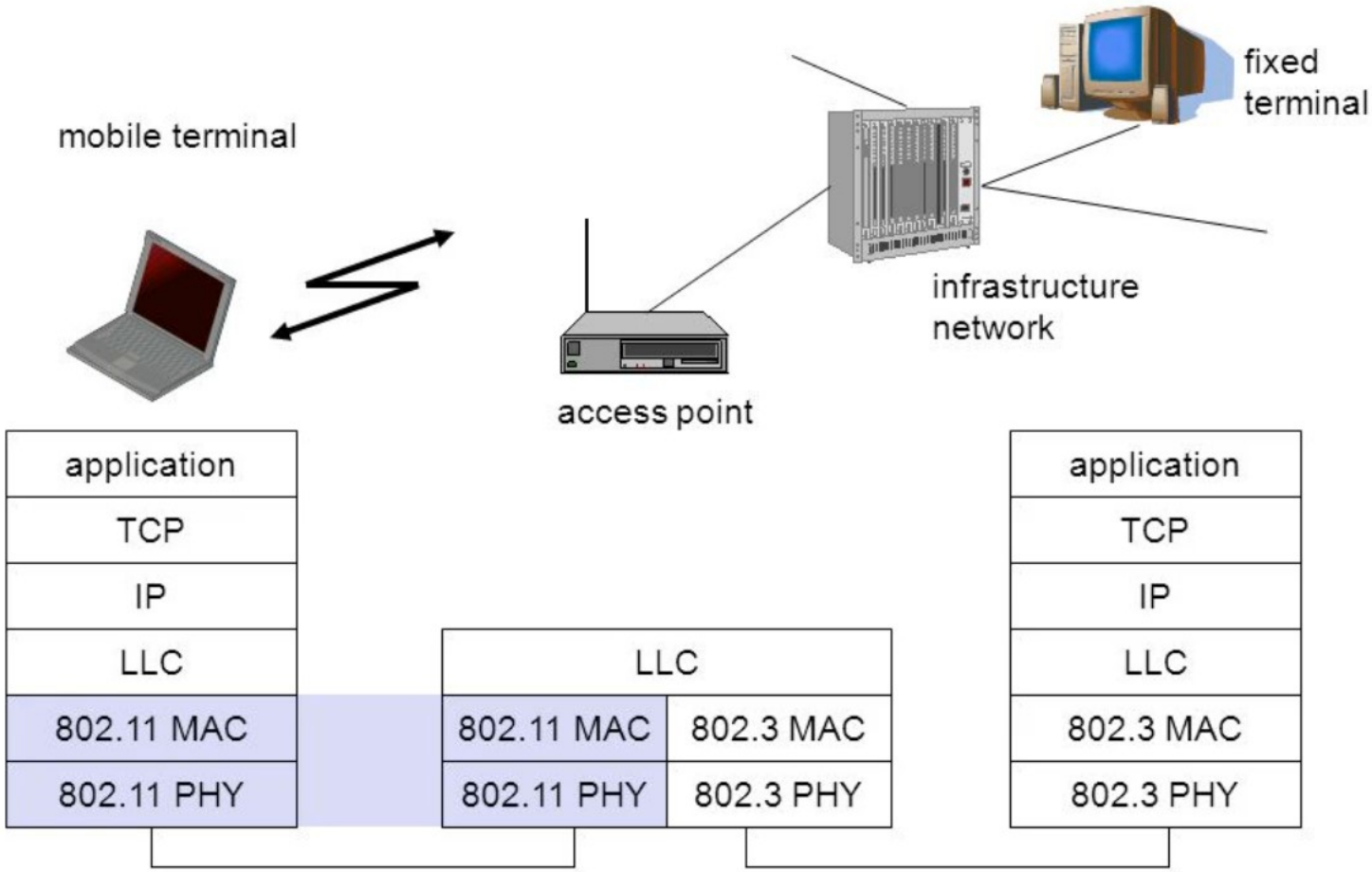


Figure 29 : IEEE standards 802.11 Layers

Wireless LAN 802.11

- IEEE has defined the specifications for a wireless LAN, called IEEE 802.11, which covers the physical and data link layers.
- IEEE 802.11 defines three types of stations based on their mobility in a wireless LAN;
  (1) No Transition
  (2) BSS Transition
  (3) ESS Transition
- No Transition Mobility: A station with no-transition mobility is either stationary or moving only inside a BSS.
- BSS Transition Mobility: A station with BSS-transition mobility can move from one BSS to another, but the movement is confirmed inside one ESS.
- ESS Transition Mobility: A station with ESS-transition mobility can move from one ESS to another.

Wireless LAN 802.11

- A BSS without an AP is called as an adhoc network; a BSS with an AP is called as an infrastructure network.

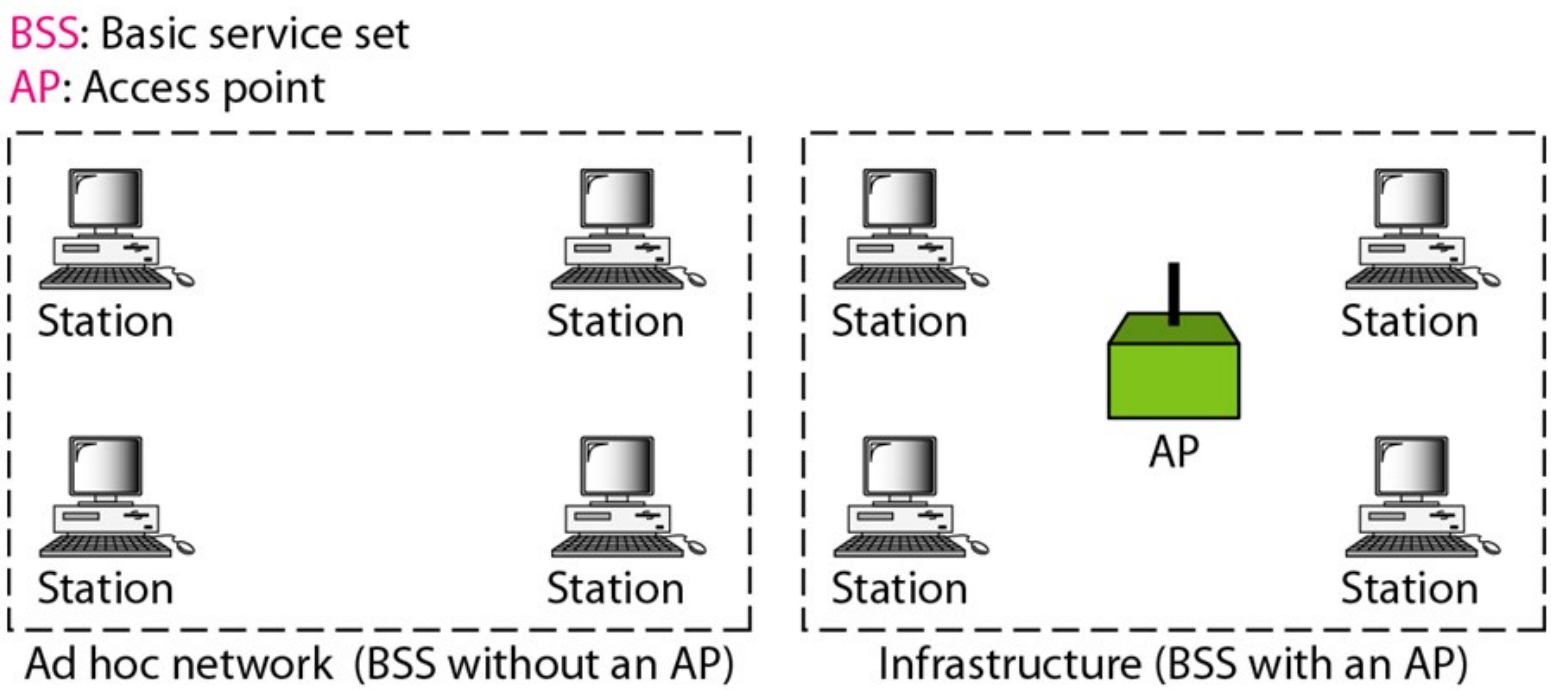


Figure 30 : Basic Service Set(BSS)

Wireless LAN 802.11

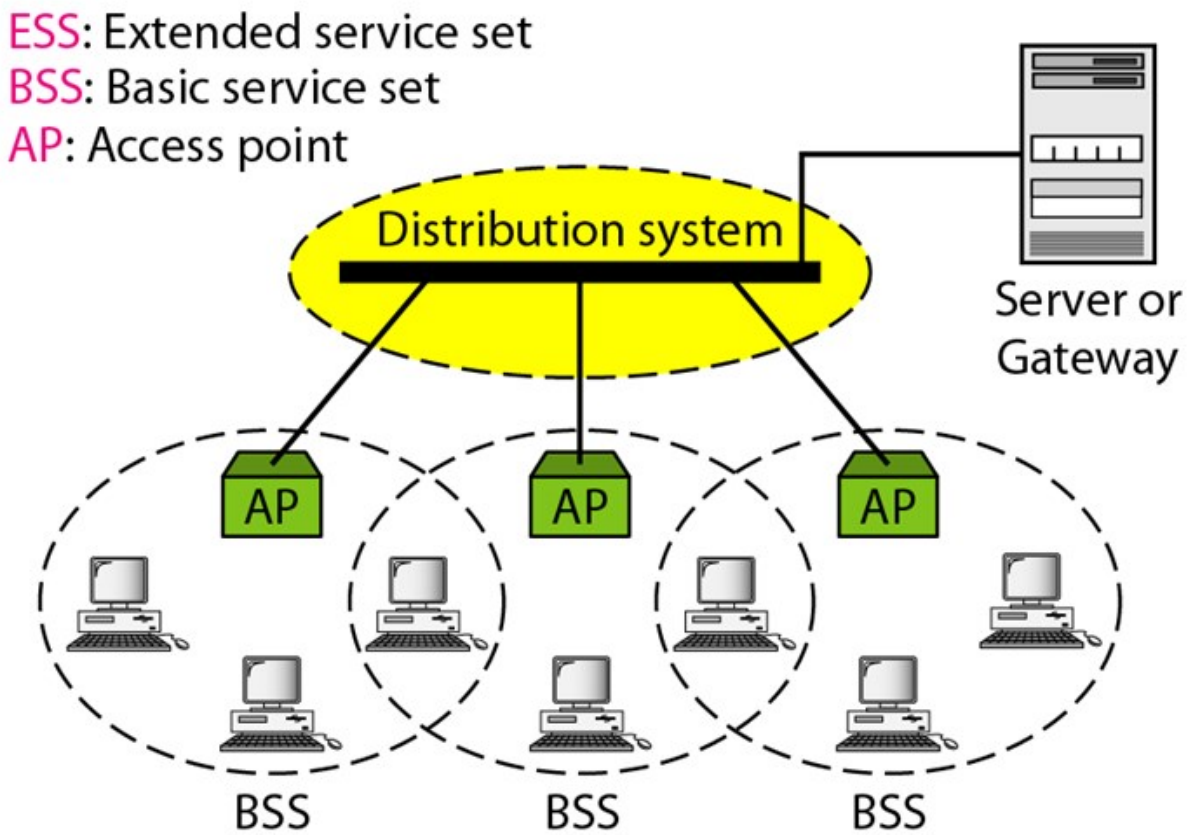


Figure 31 : Extended Basic Service Set(ESS)

Wireless LAN 802.11

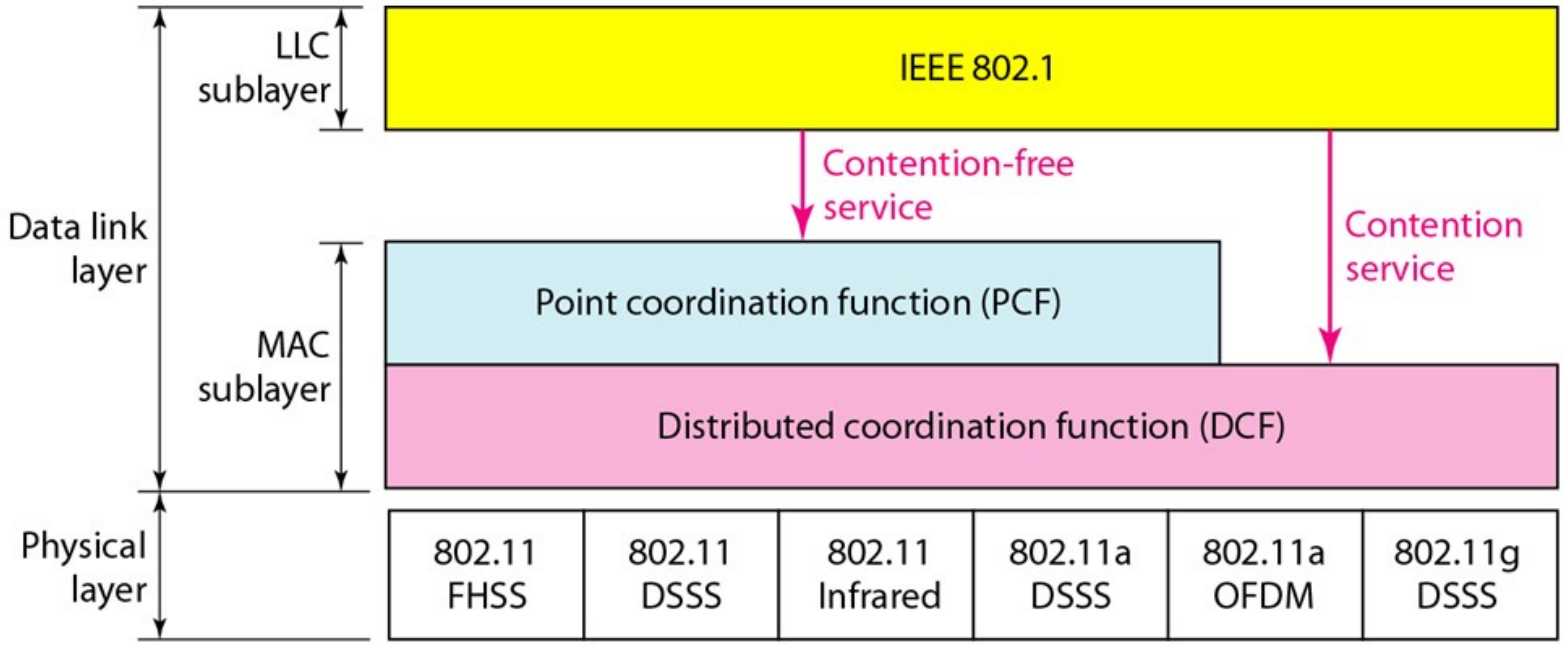


Figure 32 : MAC layers in IEEE 802.11 standard

Thank you

&

Queries