F.Y.M.Sc.(Computer Applications)
Semester - II
CSA4204
Networking Concepts
Unit – IV
The Data Link Layer
Notes

Unit IV The Data Link Layer

- Framing – Concept, Methods – Character count, Flag Bytes with Byte stuffing, Starting and Ending flags with bit stuffing and physical layer coding violations
- Error Control: Hamming Code and CRC
- Elementary Data Link protocols: Simplex, stop and wait protocol, Simplex protocol for noisy channel
- Sliding window protocols: 1-bit Sliding window protocols, Pipelining, Go-Back-N, Selective repeat

Framing

- To provide service to network layer, DLL must use the service provided by physical layer.
- Physical layer provides services as raw bits without guarantee of error free.
- To make it an error free, data link layer break the bit stream up into discrete frame and compute checksum for each frame.
- When frame arrives at the destination, the checksum is recomputed. If new one is same then its error free otherwise it is not.

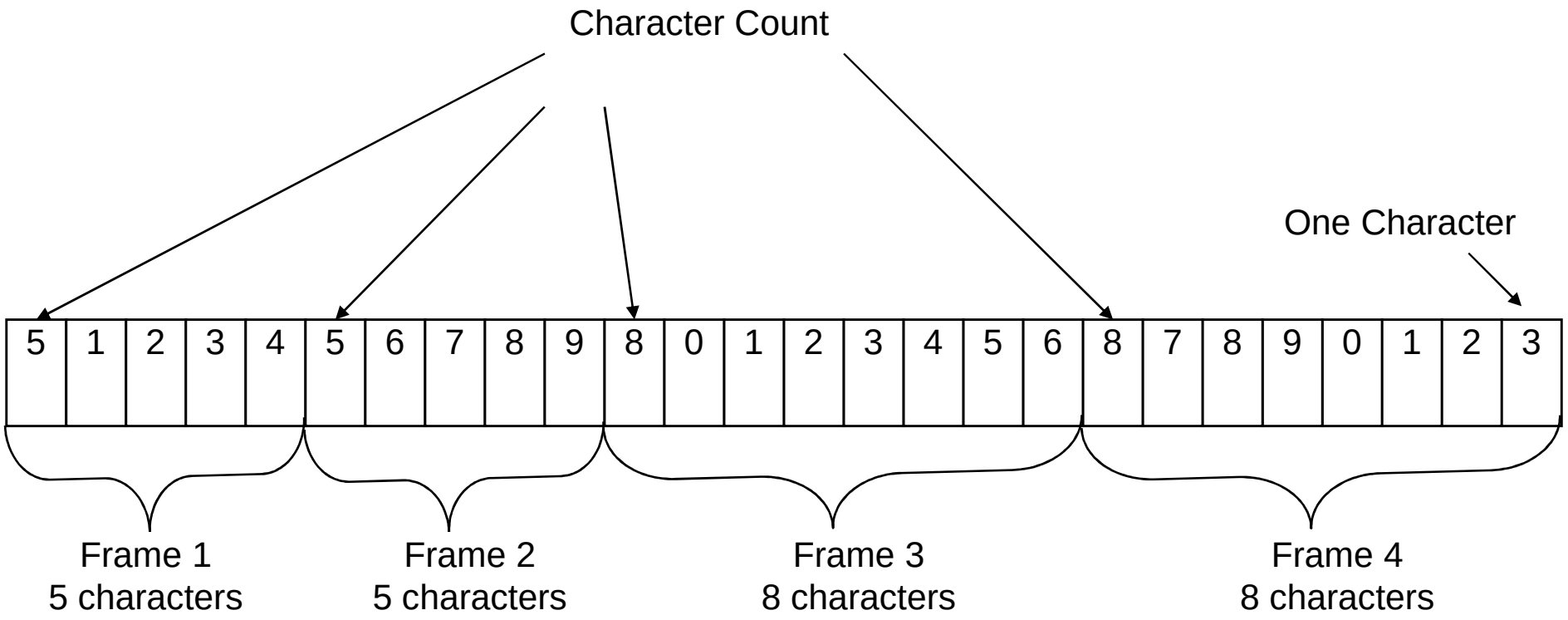
Framing

- Breaking up the bit stream into frames is more difficult than it at first appears.
- One way to achieve this framing is to insert time gaps between frames, much like the spaces between words in ordinary text.
- Since it is too risky to count on timing to mark the start and end of each frame, other methods have been devised. We look for four methods
  - Character count
  - Flag bytes with byte stuffing
  - Starting and ending flags, with bit stuffing
  - Physical layer coding violations

Character Count

- Uses a field in the header to specify the number of characters in the frame.
- When data link layer at the destination sees the character count it knows how many characters follow and hence where the end of frame is.
- Following figure shows 4 frames with character count 5, 5, 8, and 8 respectively.

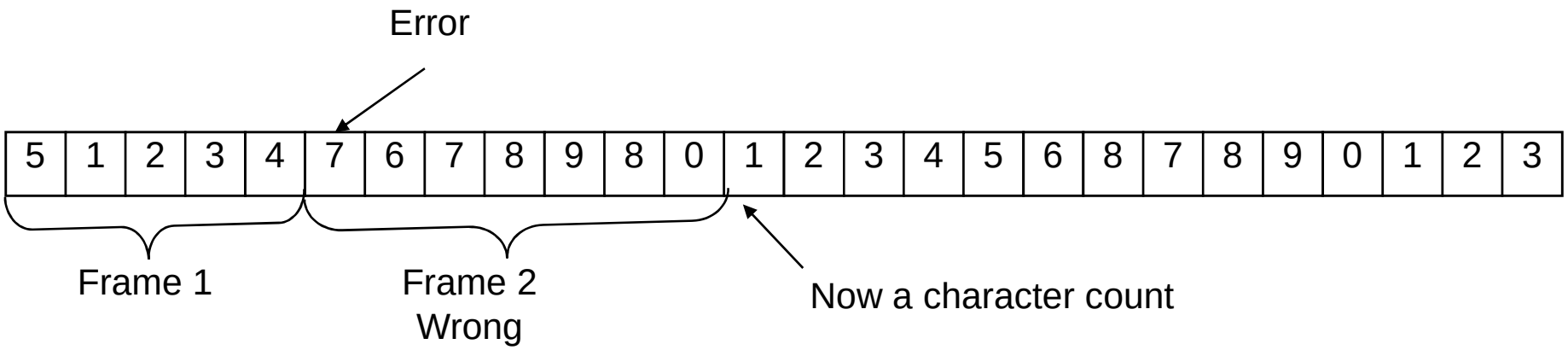
Character Count



Character Count

- Trouble is that the count can be garbled by transmission error.
- If the character count of 5 in second frame becomes 7, the destination will get out of synchronization and will be unable to locate the start of next frame.
- Sending a frame back to the source asking for retransmission does not help as destination do not know how many characters to skip over to get to the start of the retransmission.

Character Count

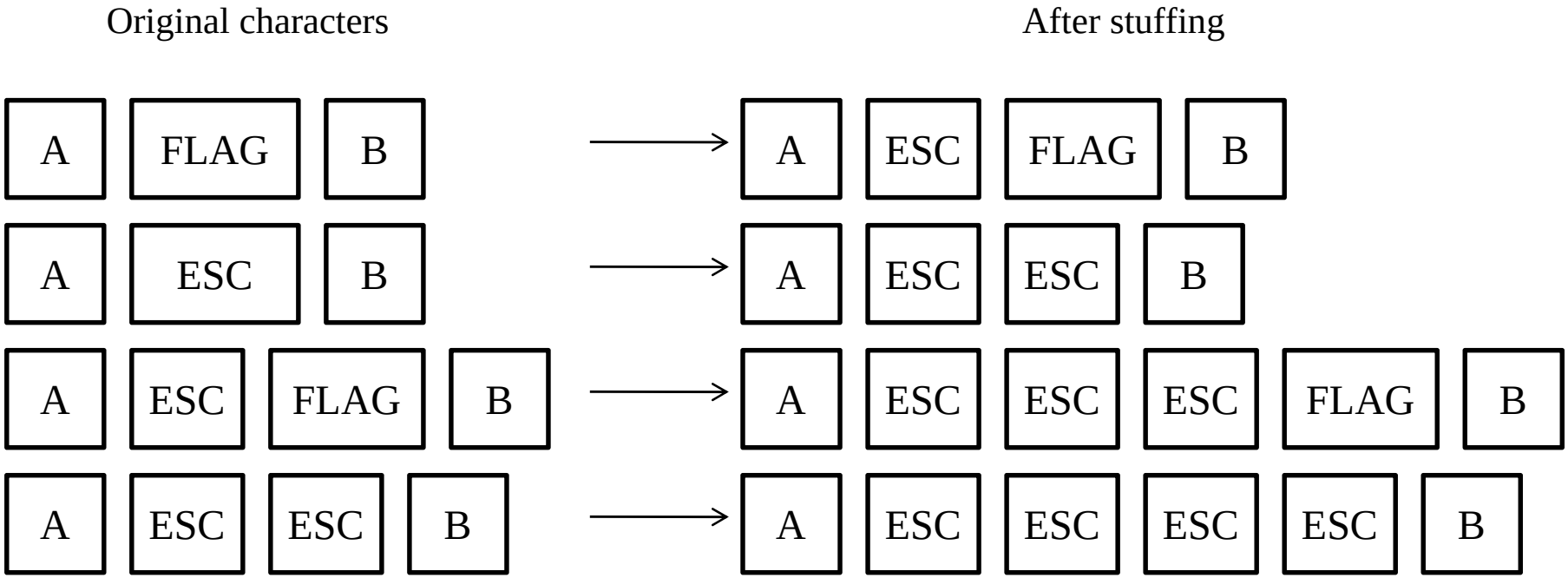


Flag Bytes with Byte Stuffing

- Method gets around the problem of resynchronization after an error by having each frame start and end with special bytes.
- In recent years most protocols have used the same byte, called flag byte, as both the starting and ending delimiter as FLAG as showed next.
- In this way if the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame.
- Two consecutive flag bytes indicate the end of one frame and start of the next one.

Flag Bytes with Byte Stuffing

(a) Frame delimited by flag bytes



(b) Four examples of byte sequences before and after byte stuffing

Flag Bytes with Byte Stuffing

- Serious problem occurs with this method when binary data, such as object programs or floating point numbers are being transmitted.
- It may easily happen that the flag byte's bit pattern occurs in data. Situation will usually interfere with the framing.
- One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each "accidental" flag byte in the data.
- DLL on the receiver end removes the escape byte before the data are given to the network layer.
- Technique is called byte stuffing or character stuffing.

Flag Bytes with Byte Stuffing

- What happens if an escape byte occurs in the middle of the data?
- The answer is that it, too, is stuffed with an escape byte.
- Any single escape byte is part of an escape sequence, whereas a doubled one indicates that a single escape occurred naturally in the data.

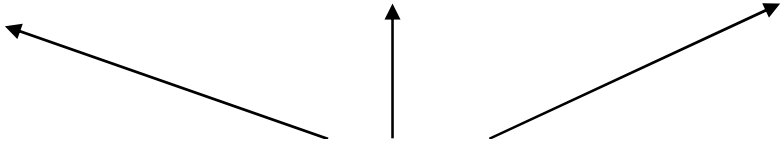
Starting and Ending flags with bit Stuffing

- Allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character.
- Each frame begins and ends with special bit pattern, 01111110 called flag byte.
- Whenever sender's DLL counts 5 consecutive 1s in the data, it stuffs a 0 bit into the outgoing bit stream. It is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data.
- When receiver sees 5 consecutive 1s followed by 0 it destuffs (i.e. deletes) 0 bit.

Starting and Ending flags with bit Stuffing

a 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

b 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0



c 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Physical layer coding violations

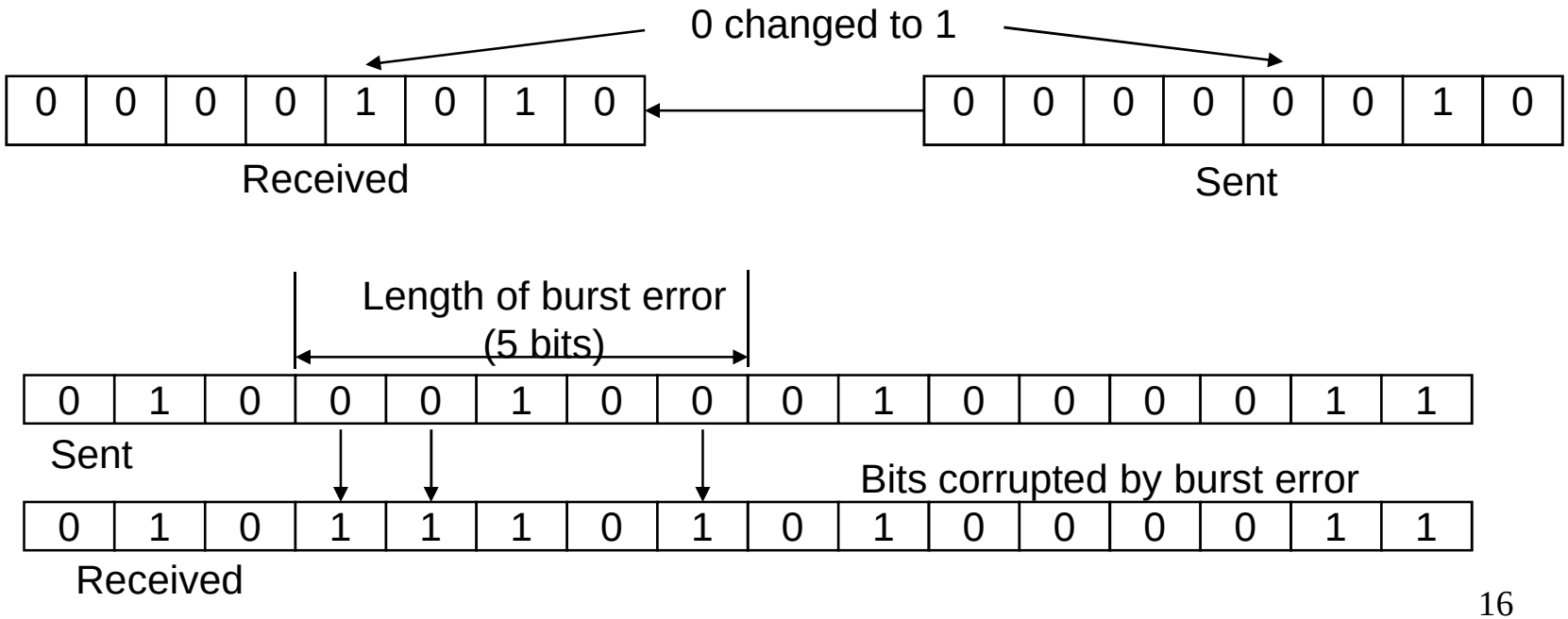
- To use a shortcut from the physical layer.
- In physical layer, the encoding of bits as signals often includes redundancy to help the receiver.
- This redundancy means that some signals will not occur in regular data.
- We can use some reserved signals to indicate the start and end of the frames.
- In effect, we are using “coding violations” to delimit frames.
- The beauty of this scheme is that, because they are reserved signals, it is easy to find start and end of frames and there is no need to stuff the data.

Error Control

- Types of Errors

  Single Bit Error - The term single bit means that only one bit of given data unit is changed from 1 to 0 or from 0 to 1.

  Burst Error - The term burst error means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.



Error Control

- Network designers have developed two basic strategies for dealing with errors.
- One way is to include enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been.
- The other way is to include only enough redundancy to allow the receiver to deduce that an error occurred but not which error, and have it request a retransmission.
- The former strategy uses error-correcting codes and the later uses error-detecting codes.
- The use of error-correcting codes is often referred to as forward error correction.

Error Control

- Hamming Code
  - Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is technique developed by R.W. Hamming for error correction.

Error Control

- Hamming Code
  - Redundant bits –
    - Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of redundant bits can be calculated using the formula: $2^r \geq m + r + 1$ where, r = redundant bit, m = data bit
    - Suppose the number of data bits is 7, then the number of redundant bits can be calculated as: $2^4 \geq 7 + 4 + 1$ Thus, the number of redundant bits= 4

Error Control

- Hamming Code
  - Parity bits–
    - A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1’s in the data are even or odd. Parity bits are used for error detection.
    - There are two types of parity bits: Even parity bit and Odd parity bit

Error Control

- Hamming Code
  - Parity bits – Even Parity Bit
    - In the case of even parity, for a given set of bits, the number of 1’s are counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1’s an even number. If the total number of 1’s in a given set of bits is already even, the parity bit’s value is 0.

Error Control

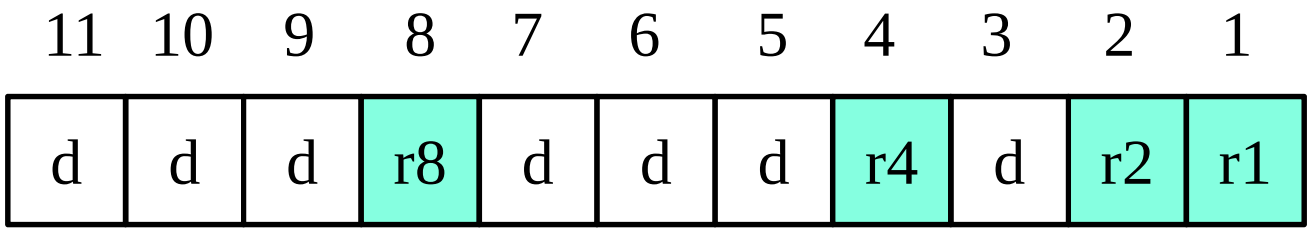
- Hamming Code
  - Parity bits – Odd Parity Bit
    - In the case of odd parity, for a given set of bits, the number of 1’s are counted. If that count is even, the parity bit value is set to 1, making the total count of occurrences of 1’s an odd number. If the total number of 1’s in a given set of bits is already odd, the parity bit’s value is 0.

Error Control

- Hamming Code
  - Can be applied to data units of any length and uses the relationship between data and redundancy bits.
  - For example, a 7-bit ASCII code requires 4 redundancy bits that can be added to the end of the data unit or interspersed with the original data bits.
  - In the figure given next, these bits are placed in positions 1,2,4 and 8 (the positions in an 11-bit sequence that are powers of 2). We refer these bits as r1, r2, r4 and r8.

Error Control

- Hamming Code



  - In Hamming code, each r bit is the parity bit for one combination of data bits, as shown below

    r1: bits 1,3,5,7,9,11

    r2: bits 2,3,6,7,10,11

    r4: bits 4,5,6,7

    r8: bits 8,9,10,11

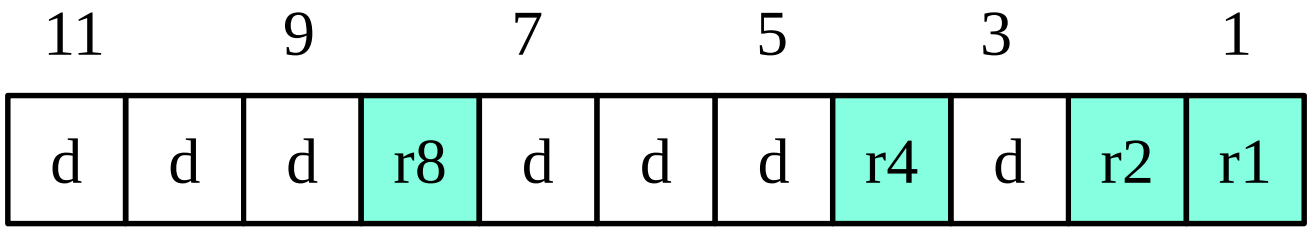
Error Control

- Hamming Code
  - Each data bit may be included in more than one calculation. In the sequences above, for example, each of the original data bits is included in at least two sets, while the r bits are included in only one.

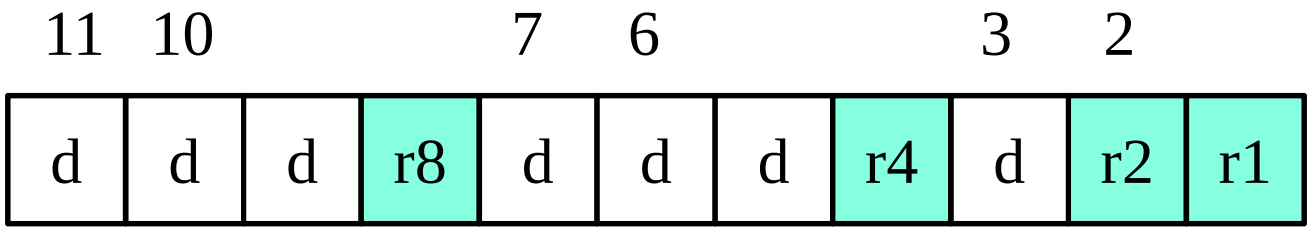
Error Control

- Hamming Code

r1 will take care of these bits



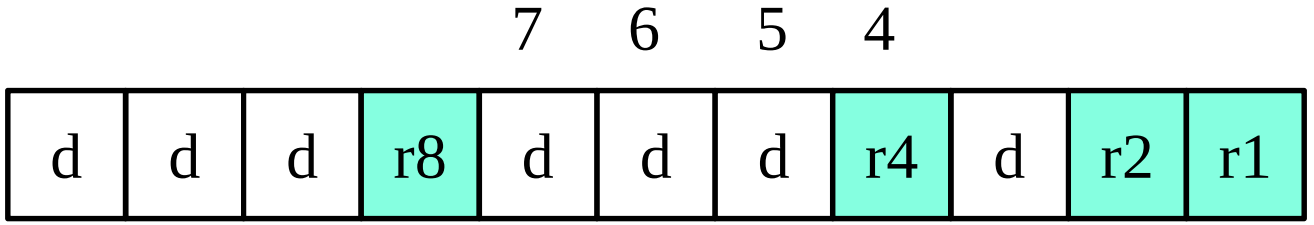
r2 will take care of these bits



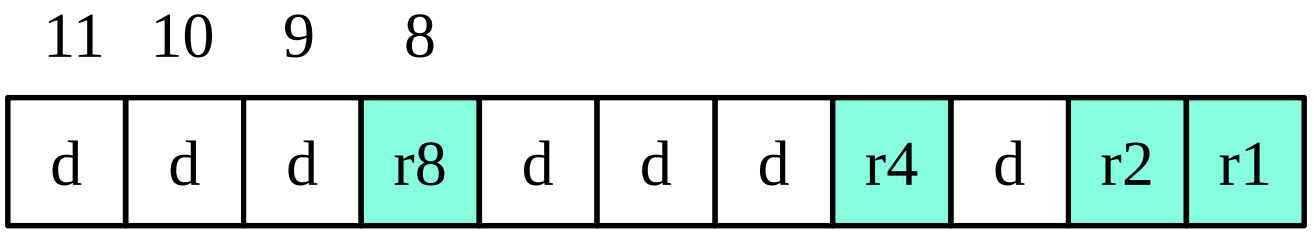
Error Control

- Hamming Code

r4 will take care of these bits



r8 will take care of these bits



Error Control

- Hamming Code – Calculating r values
  - In the first step, we place each bit of the original character in its appropriate position in the 11 bit unit.
  - In the subsequent steps, we calculate the even parities for the various bit combination.
  - The parity value for each combination is the value of the corresponding r bit.

Error Control

- Hamming Code – Error Detection and Correction
  - Imagine that by the time the above transmission is received , the number 7 bit has been changed from 1 to 0.
  - The receiver takes the transmission and recalculates 4 new parity bits, using the same sets of bits used by the sender plus the relevant parity r bit for each set.
  - Then it assembles the new parity values into a binary number in order of r position (r8, r4, r2, r1).

Error Control- CRC

- Also called as ‘Polynomial Code Method’.
- Codes are based on treating bit strings as representations of polynomials with coefficients of 0 and 1.
- A ‘*k*’ bit frame is regarded as the coefficient list for polynomial with ‘*k*’ terms, ranging from X^{k-1} to X^{0}. Polynomial is said to be degree of k-1.
- Example $X^{5}+X^{4}+X^{0}$ = 1, 1, 0, 0, 0 and 1.
- Polynomial arithmetic is done modulo 2. There are no carries for addition or borrows for subtraction.
- Addition and subtraction are identical to EXCLUSIVE OR (EXOR).

Cyclic Redundancy Code

- When polynomial code method is employed, the sender and receiver must agree upon generator polynomial G(x).
- High and low order bits of the generator must be 1.
- To compute the checksum for some frame with ‘m’ bits, corresponding to the polynomial M(x), the frame must be longer than the G(x).
- Idea is to append checksum to the end of the frame in such way that polynomial represented by the checksummed frame is divisible by G(x).
- Receiver also tries dividing it by G(x). If there is remainder, there has been a transmission error.

Cyclic Redundancy Code

- Algorithm to compute the checksum is
  - Let r be the degree of G(x). Append r zero bits to the low order end of the frame, so it now contains m+r bits and corresponds to the polynomial $X^rM(x)$.
  - Divide the bit string corresponding to G(x) into the bit string corresponding to $X^rM(x)$ using modulo 2 division.
  - Subtract the remainder from the bit string corresponding to $X^rM(x)$ using modulo 2 subtraction. The result is the checksummed frame to be transmitted. Call its polynomial T(x).

Elementary Data Link Layer Protocols

Noiseless Channel

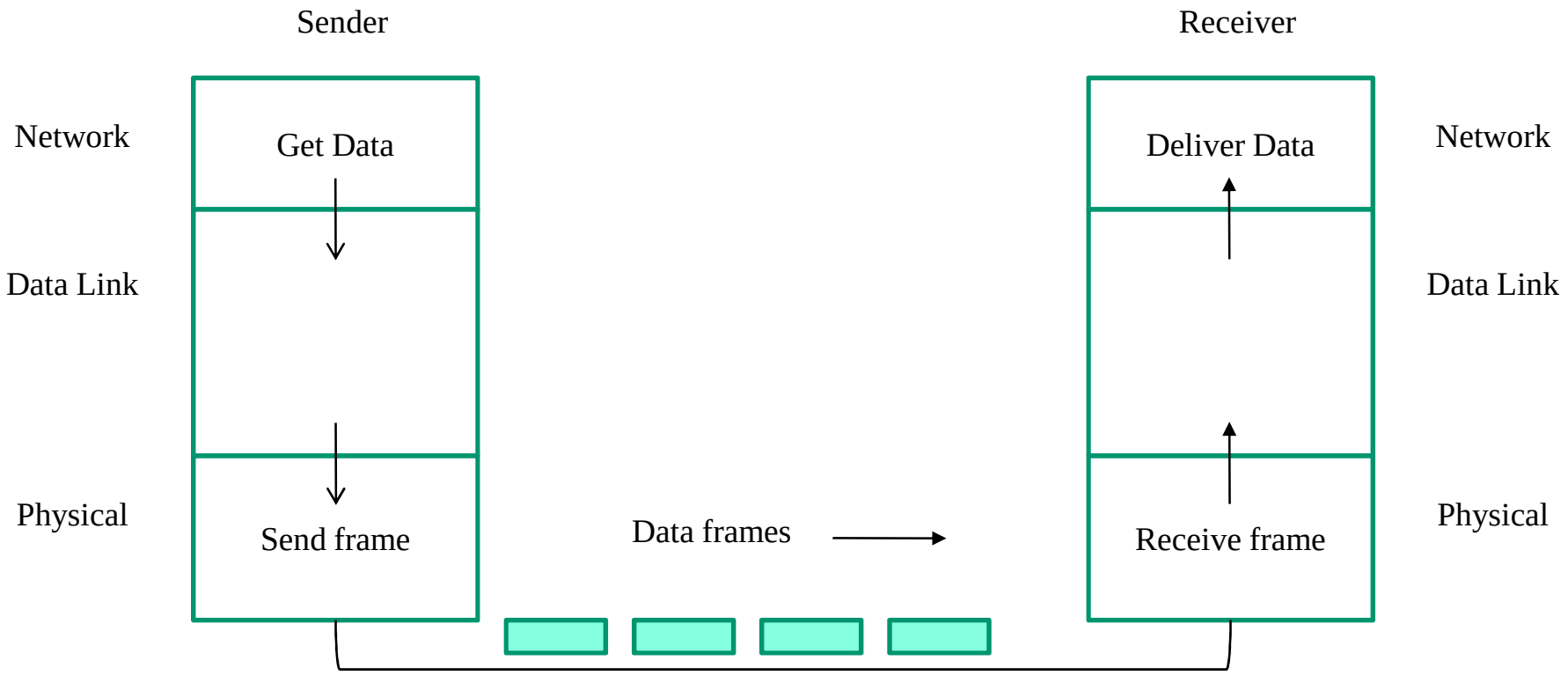
- We assume that we have ideal channel in which no frames are lost, duplicated or corrupted.
- Two types of protocols are applicable for this type of channel.
  - Does not use flow control
  - Use flow control
- Both of them neither has error control because channel is perfect noiseless channel

Noiseless Channel

- Simplest Protocol
  - Has no flow or error control.
  - It is a unidirectional protocol in which data frames are traveling in only one direction – from the sender to receiver.
  - We assume that the receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible.
  - DLL of the receiver immediately removes the header from the frame and hands the data packet to its network layer, which can also accept immediately.

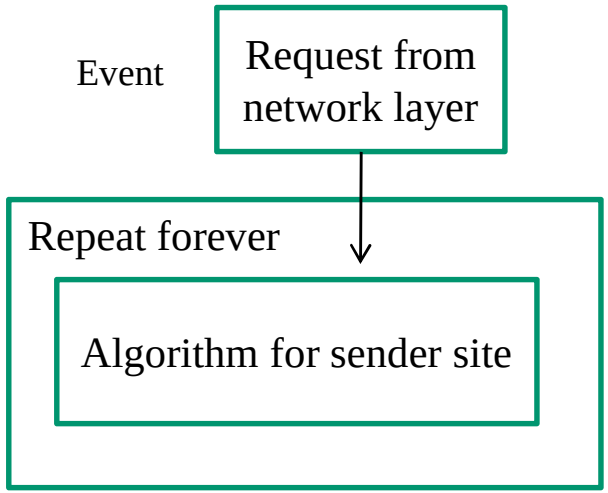
Noiseless Channel

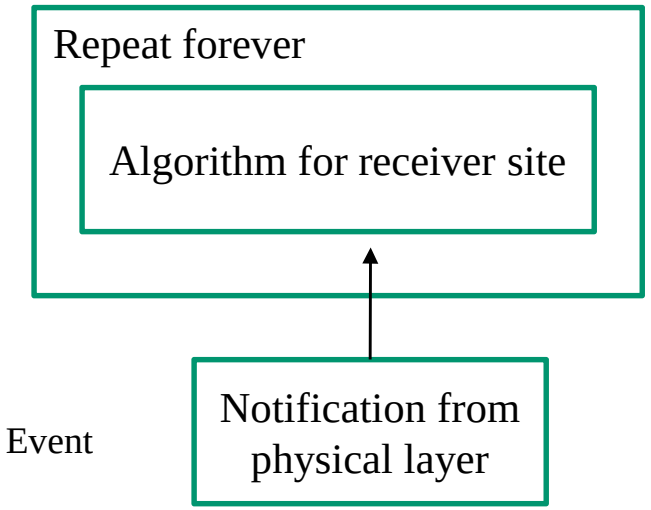
- Simplest Protocol - Design



Noiseless Channel

- Simplest Protocol - Design





Noiseless Channel

- Simplest Protocol – Design
  - No need for flow control. DLL at the sender site gets data from its network layer, makes a frame out of the data, and sends it.
  - DLL at receiver site receives a frame from its physical layer, extracts data from the frame and delivers the data to its network layer.
  - DLL of the sender and receiver provide transmission services for their network layers.
  - DLL use the services provided by their physical layers for the physical transmission of bits.

Noiseless Channel

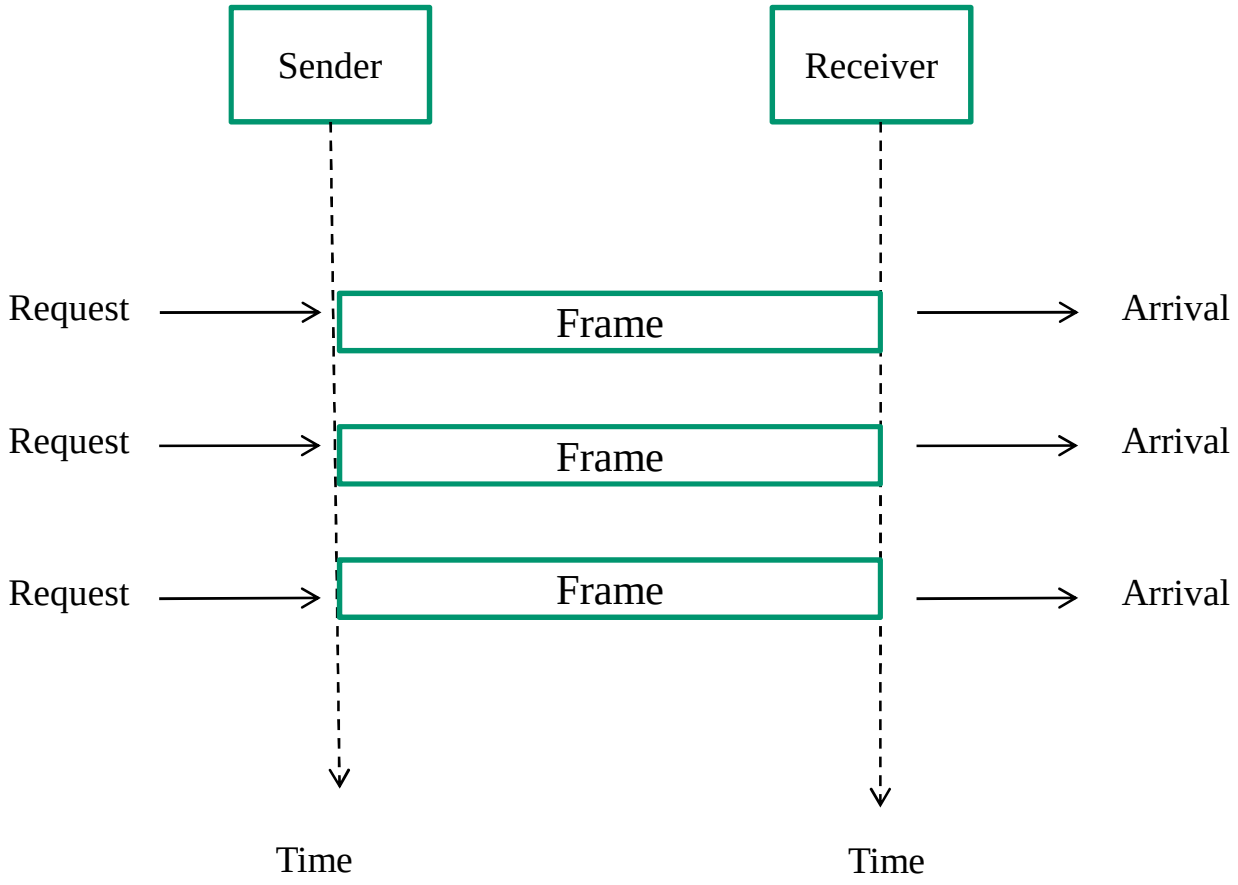
- Simplest Protocol – Design – Procedure Used by both data link layers
  - The sender site cannot send a frame until its network layer has a data packet to send.
  - The receiver site cannot deliver a data packet to its network layer until a frame arrives.
  - If the protocol is implemented as a procedure, we need to introduce the idea of **events** in the protocol.

Noiseless Channel

- Simplest Protocol – Design – Procedure Used by both data link layers
  - The procedure at the sender site is constantly running; there is no action until there is a request from the network layer.
  - The procedure at the receiver site is also constantly running, but no action until notification from the physical layer arrives.
  - Both procedures are constantly running since they do not know when the corresponding events will occur.

Noiseless Channel

- Simplest Protocol – Flow Diagram



Noiseless Channel

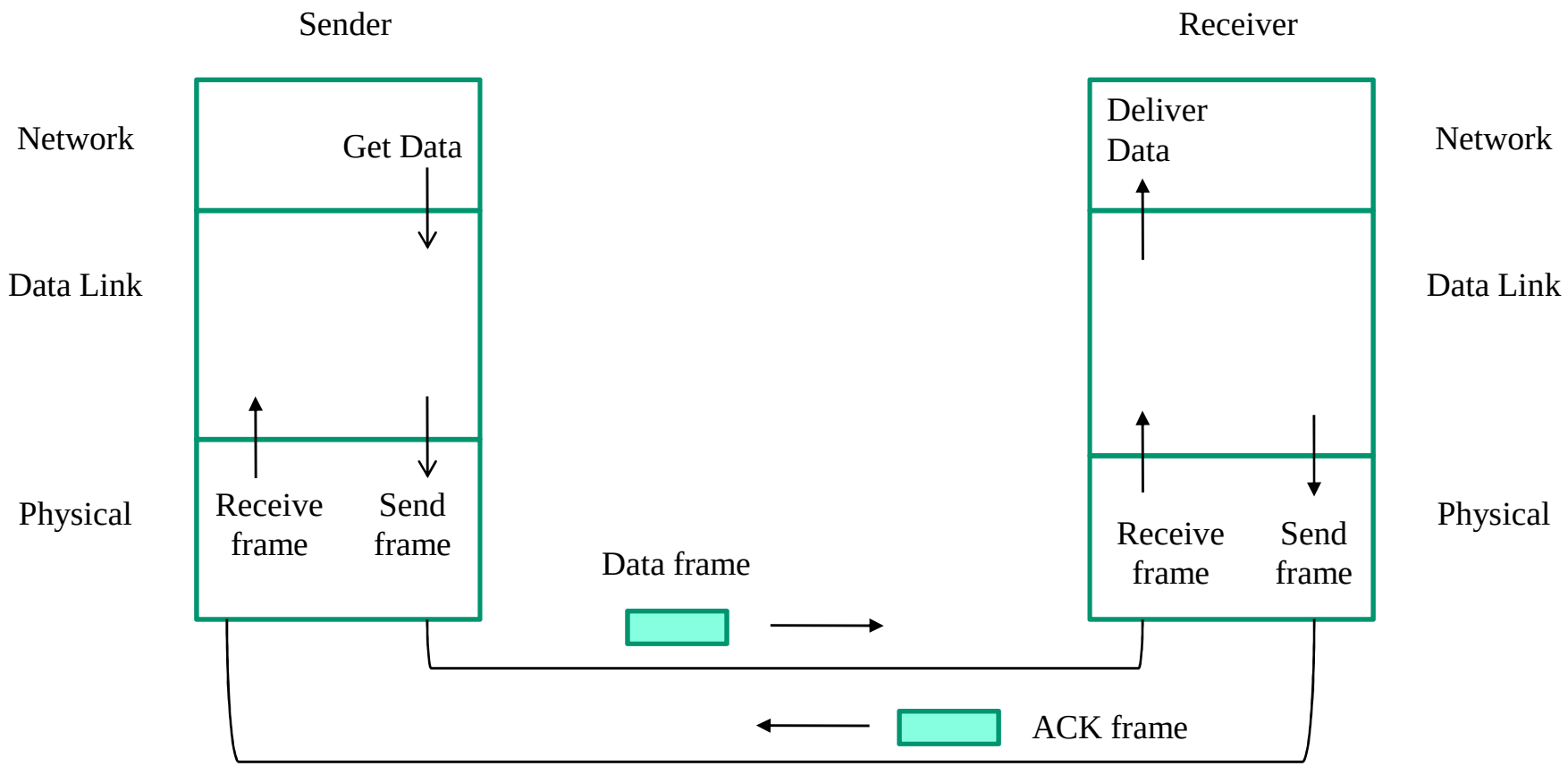
- Stop and wait protocol
  - If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use.
  - Normally, the receiver does not have enough storage space, which may result in either the discarding of frames or denial of service.
  - To prevent the receiver from becoming overwhelmed with frames, we somehow need to tell the sender to slow down.
  - There must be feedback from the receiver to the sender.

Noiseless Channel

- Stop and wait protocol
  - Protocol is call stop-and wait because the sender sends one frame, stops until it receives confirmation from the receiver and then sends the next frame.
  - Communication is still unidirectional for data frames, but auxiliary ACK frames travel from the other direction.
  - We add flow control to previous protocol.

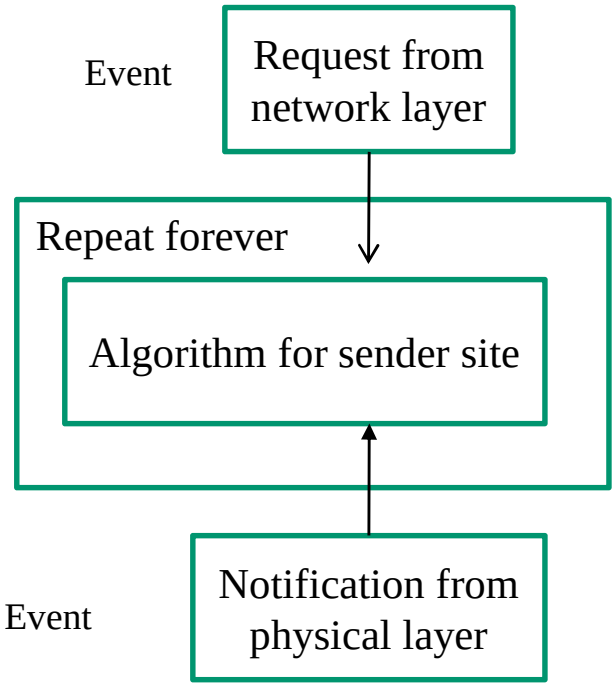
Noiseless Channel

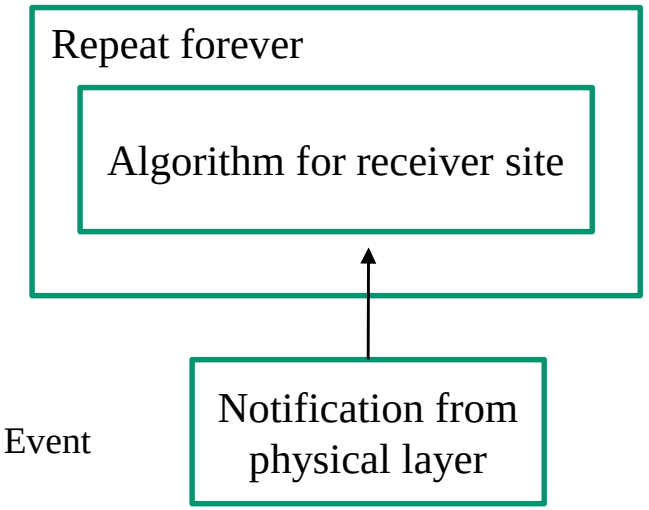
- Stop and wait protocol - Design



Noiseless Channel

- Stop and wait protocol - Design



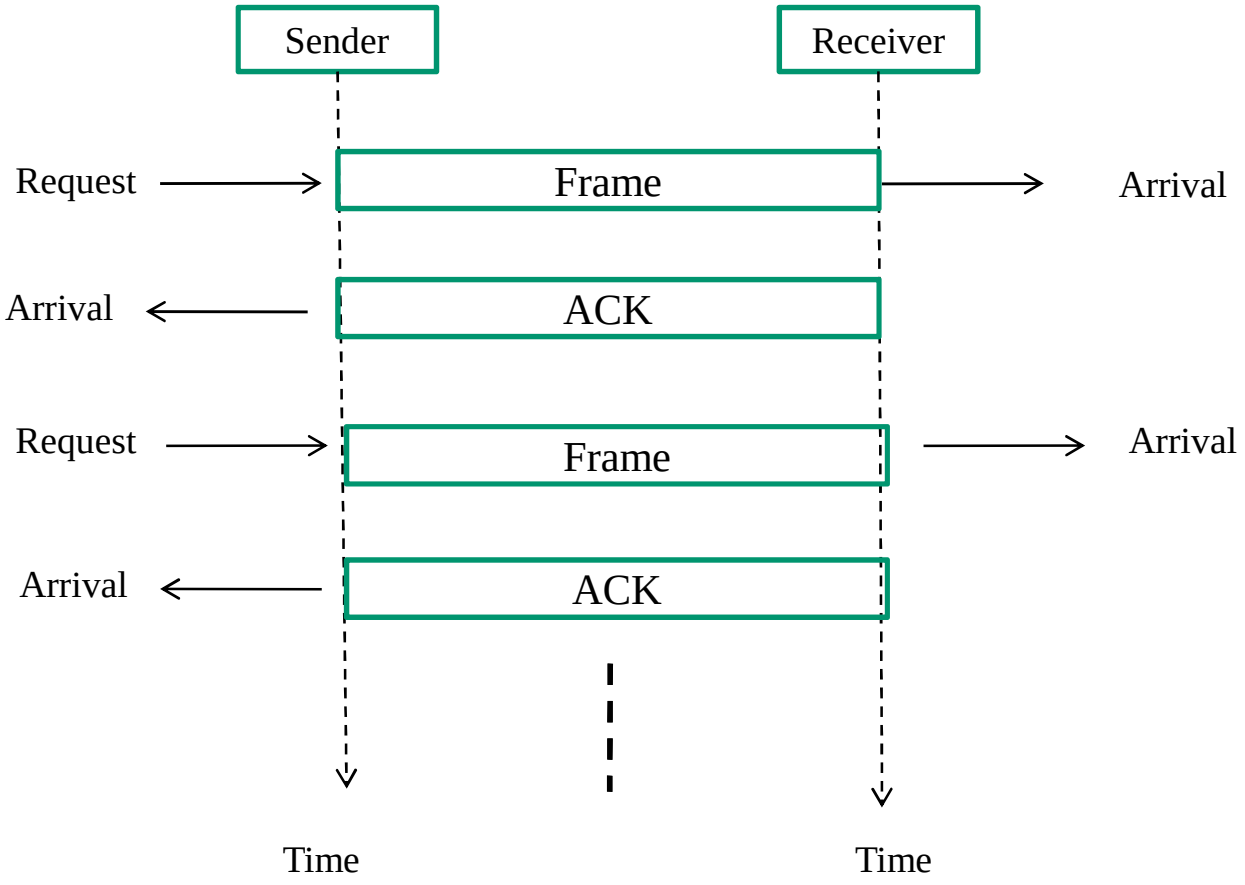


Noiseless Channel

- Stop and wait protocol - Design
  - At any time, there is either one data frame on the forward channel or one ACK frame on the reverse channel.
  - We need half duplex link.

Noiseless Channel

- Stop and wait protocol – Flow Diagram



Noisy Channel

- Stop and wait Automatic Repeat Request
  - Adds a simple error control mechanism to the Stop-and-wait protocol.
  - To detect and correct corrupted frames, we need to add redundancy bits to our data frame.
  - When the frame arrives at the receiver site, it is checked and if it is corrupted, it is discarded.
  - Detection of errors in this protocol is manifested by the silence of the receiver.

Noisy Channel

- Stop and wait Automatic Repeat Request
  - Lost frames are more difficult to handle than corrupted ones. In previous protocols, there was no way to identify a frame.
  - The received frame could be the correct one or a duplicate; or a frame out of order. Solution is to number the frames.
  - When the receiver receives a data frame that is out of order, means that frames were either lost or corrupted.
  - The corrupted and lost frames need to be resent here.

Noisy Channel

- Stop and wait Automatic Repeat Request
  - If the receiver does not respond when there is an error, how can the sender know which frame to resend? Solution: sender keeps the copy of the sent frame, also it starts the timer.
  - If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held and the timer is restarted.
  - Since protocol uses stop-and-wait, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.
  - An ACK frame can also be corrupted or lost, it too needs redundancy bits and a sequence number. ACK frame has a sequence number field.

Noisy Channel

- Stop and wait Automatic Repeat Request
  - Sequence Numbers
    - Protocol specifies, frames need to be numbered. Done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame.
    - One important consideration is range of sequence numbers. Requirement is minimize the frame size, so we look for smallest range that provides unambiguous communication.
    - The sequence numbers of course can wrap around. If we decide that the field is *m* bits long, the sequence numbers starts from 0, go to 2^m-1 and then are repeated.
    - Let us reason out the range of sequence numbers we need,

Noisy Channel

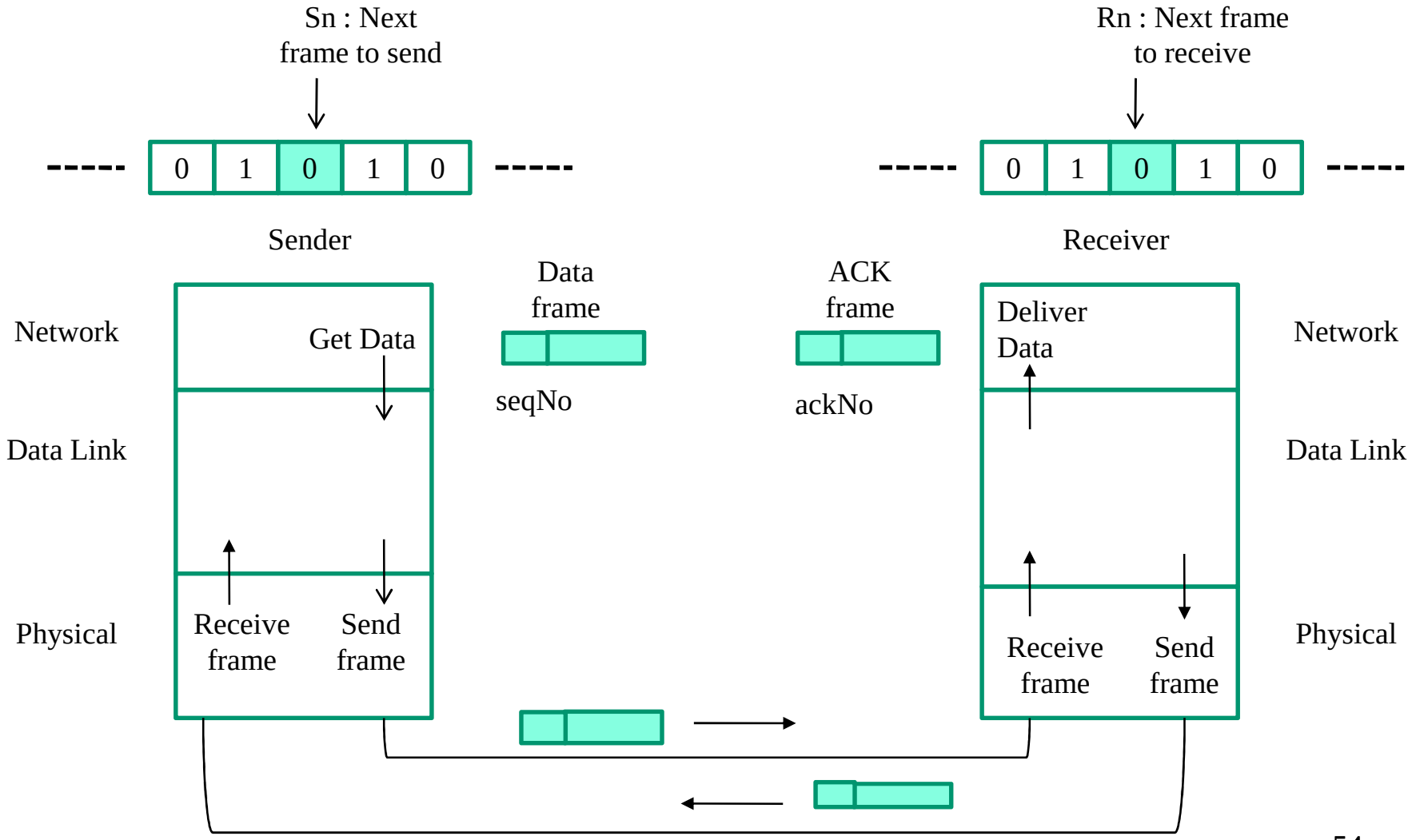
- Stop and wait Automatic Repeat Request
  - Sequence Numbers
    - The sender has sent the frame numbered x. Three things happens
      - The frame arrives safe and sound at the receiver site; the receiver sends an acknowledgement. The acknowledgement arrives at sender, causing sender to send the next frames numbered x + 1.
      - The frame arrives safe and sound at the receiver site; the receiver sends an acknowledgement, but it is corrupted or lost. The sender resends the frame (numbered x) after time-out. The frame here is duplicate. The receiver can recognize this fact because it expects frame x + 1 but frame x was received.
      - The frame is corrupted or never arrives at the receiver site, the sender resends the frame (numbered x) after the time out.

Noisy Channel

- Stop and wait Automatic Repeat Request
  - Acknowledgement Numbers
    - The sequence numbers must be suitable for both data frames and ACK frames, we use this convention: The acknowledgement numbers always announce the sequence number of the next frame expected by the receiver.
    - For example, if frame 0 has arrived safely, the receiver sends an ACK frame with acknowledgment 1 (means frame 1 is expected next). If frame 1 has arrived safe, the receiver sends an ACK frame with acknowledgement 0 (meaning frame 0 is expected).

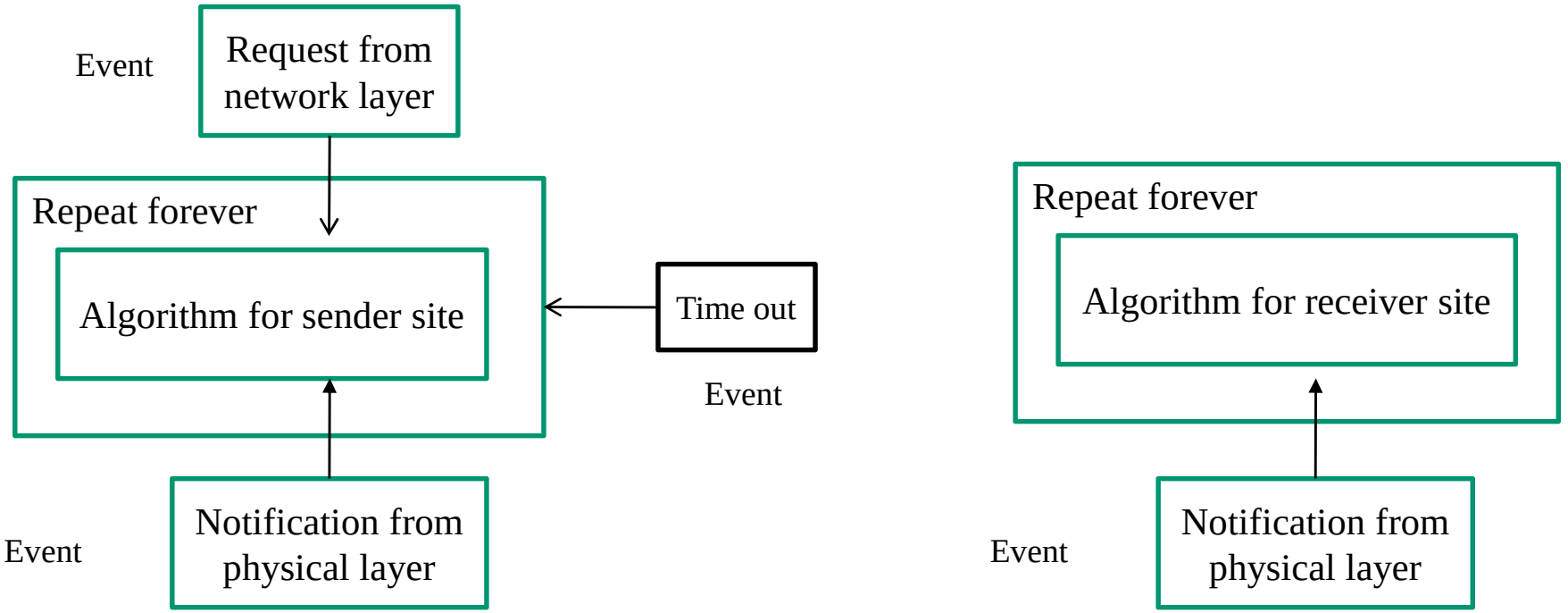
Noisy Channel

- Stop and wait Automatic Repeat Request – Design



Noisy Channel

- Stop and wait Automatic Repeat Request – Design



Noisy Channel

- Stop and wait Automatic Repeat Request – Design
  - The sending device keeps a copy of the last frame transmitted until it receives an acknowledgement for that frame.
  - A data frame uses a seqNo (sequence number); an Ack frame uses ackNo (acknowledgement number).
  - Sender has a control variable, which we call Sn (sender, next frame to send), that holds the sequence number for the next frame to be sent (0 or 1).

Noisy Channel

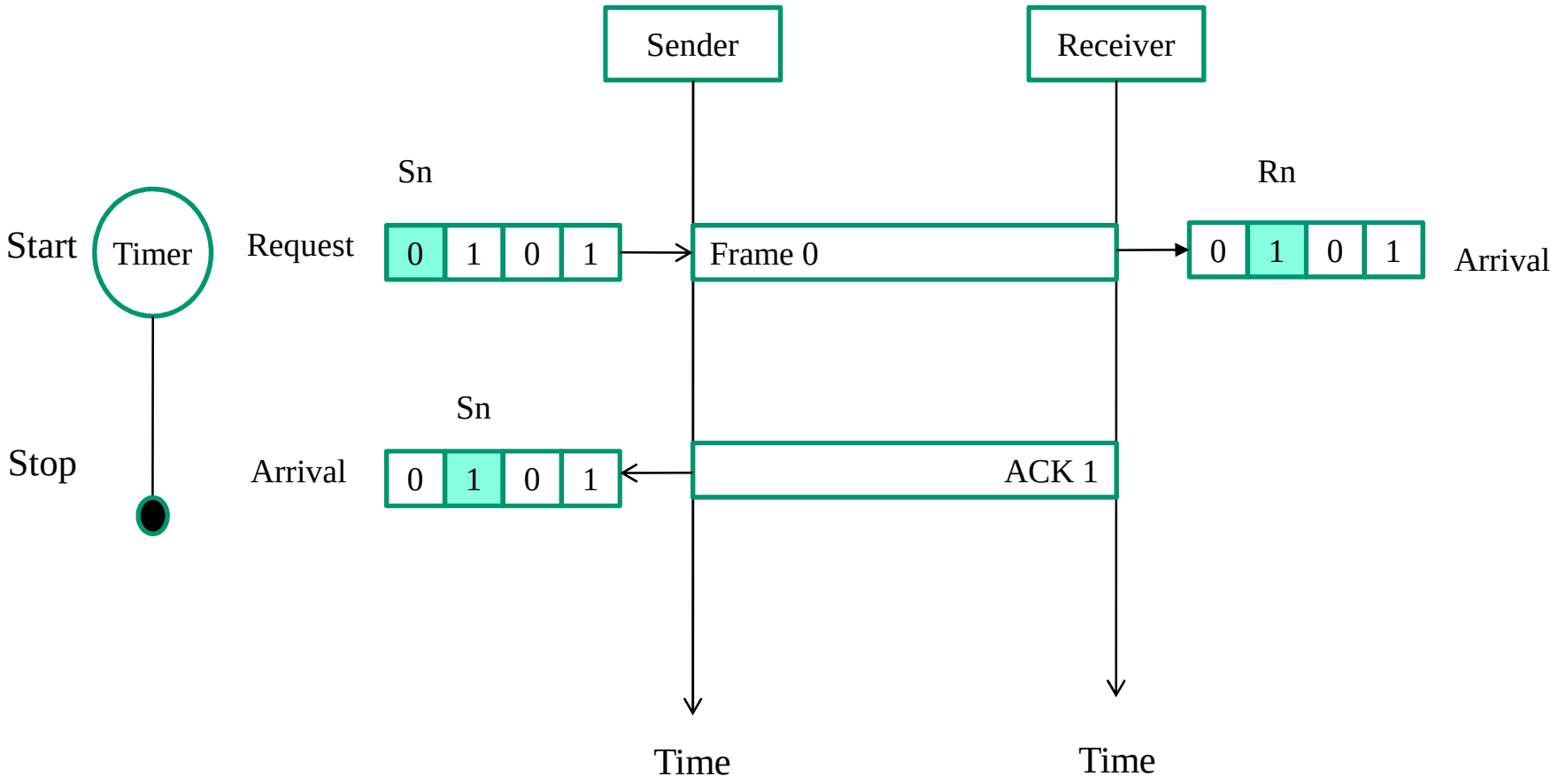
- Stop and wait Automatic Repeat Request – Design
  - The receiver has a control variable, call Rn (receiver, next frame expected) that holds the number of the next frame expected.
  - When frame is sent, the value of Sn is incremented (modulo 2), which means if it is 0, it becomes 1 and vice versa.
  - When frame is received, the value of Rn is incremented (modulo 2), which means if it is 0, it becomes 1 and vice versa.

Noisy Channel

- Stop and wait Automatic Repeat Request – Design
  - Three events can happen at sender site;one event occurs at the receiver site.
  - Variable Sn points to the slot that matches the sequence number of the frame that has been sent, but not acknowledged; Rn points to the slot that matches the sequence number of the expected frame.

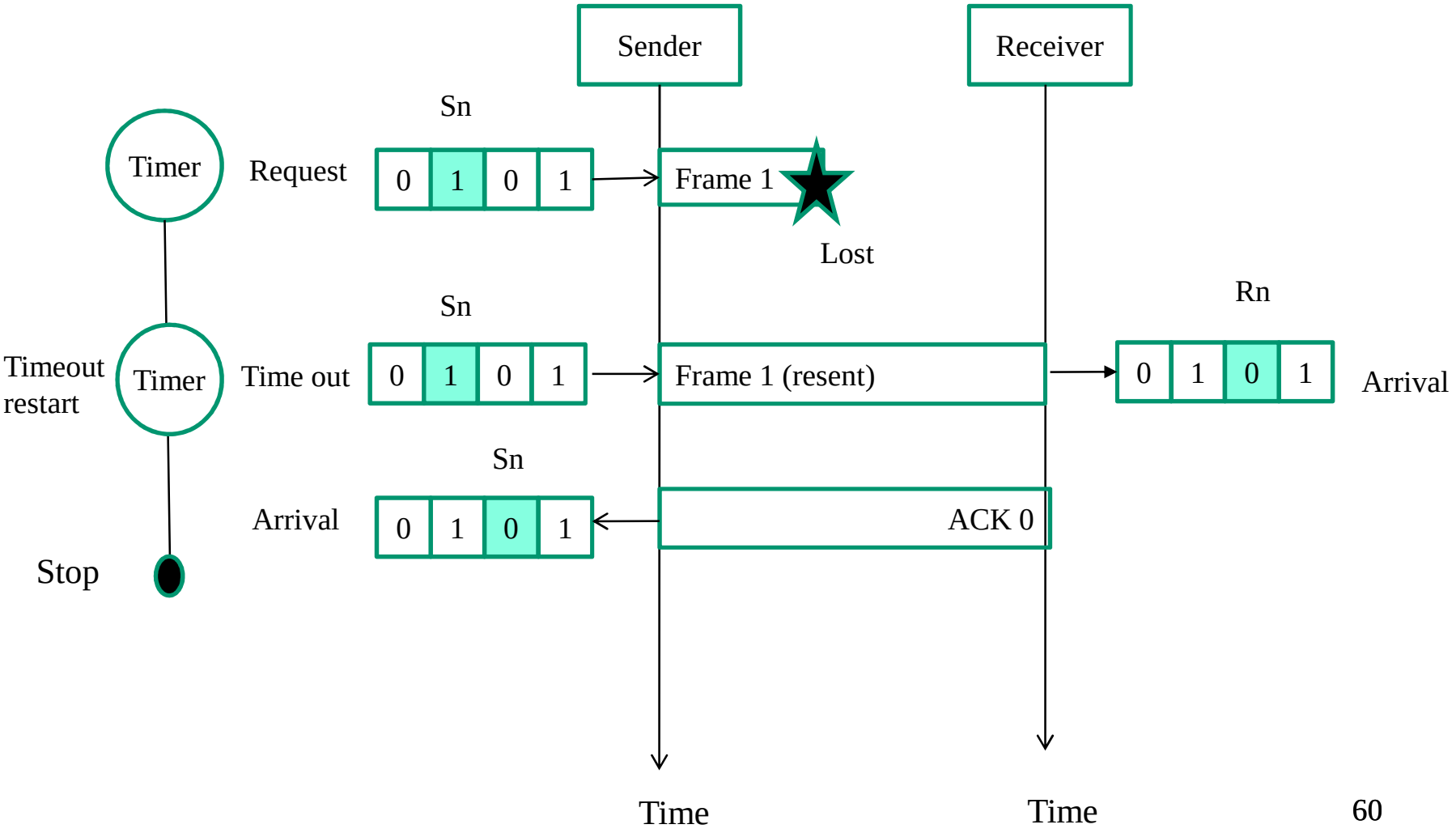
Noisy Channel

- Stop and wait Automatic Repeat Request – Flow Diagram



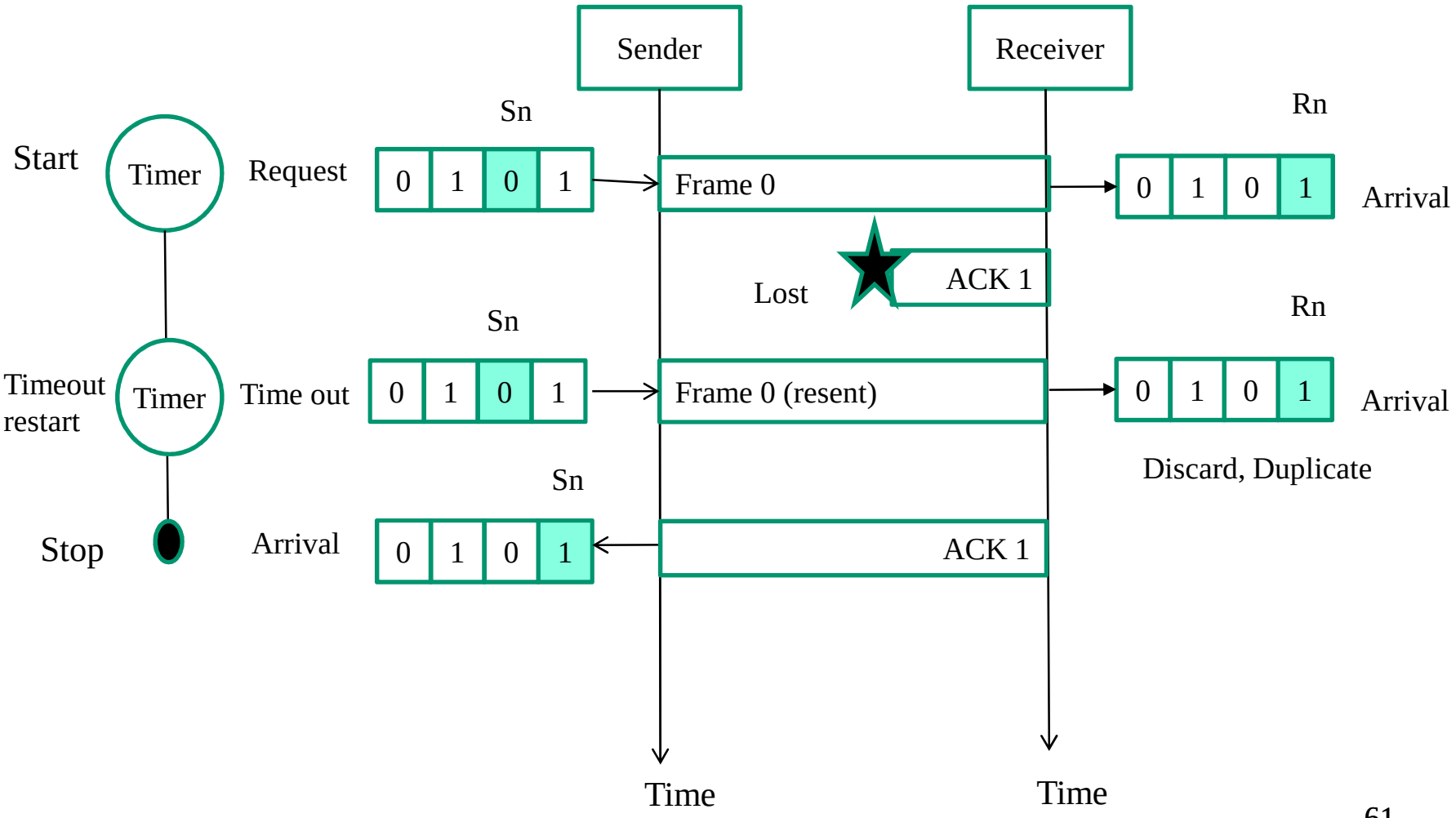
Noisy Channel

- Stop and wait Automatic Repeat Request – Flow Diagram



Noisy Channel

- Stop and wait Automatic Repeat Request – Flow Diagram



Noisy Channel

- Pipelining
  - In networking and in other areas, a task is often begun before the previous task has ended. Known as pipelining.
  - No pipelining in Stop-and-wait ARQ because we need to wait for a frame to reach the destination and be acknowledged before the next frame can be sent.
  - Pipelining does apply to our next two protocols because several frames can be sent before we receive news about the previous frames.
  - Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay product.

Noisy Channel

- Pipelining
  - To improve efficiency of transmission (filling in pipe), multiple frames must be in transition while waiting for acknowledgement.
  - We need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgement.
  - Go Back N Automatic Repeat Request protocol can achieve this goal.

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Here we can send several frames before receiving acknowledgements; we keep all of these frames until the acknowledgements arrive.
  - Sequence Numbers
    - Frames from 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 bits for the sequence number, the sequence numbers range from 0 to 2^m-1.
    - Example: if the m=3 the only sequence numbers are 0 through 7 inclusive.
    - We can repeat the sequence. So we have 0,1,2,3,4,5,6,7,0,1, 2,3,4,5,6,7

Noisy Channel

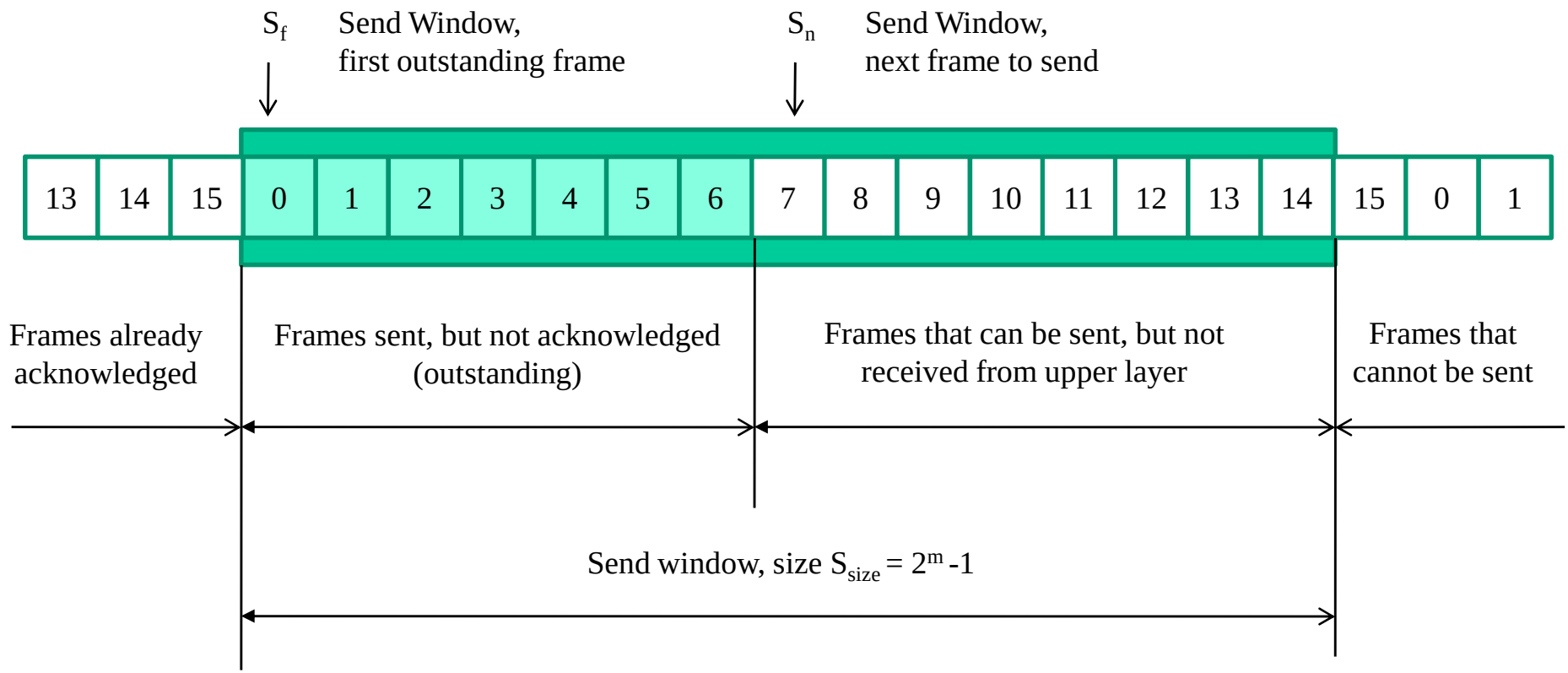
- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window
    - The sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver
    - The sender and receiver need to deal with only part of the possible sequence numbers.
    - The range which is the concern of the sender is called the send sliding window
    - The range that is the concern of the receiver is called the receive sliding window

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Send Sliding Window
    - An imaginary box covering the sequence numbers of the data frames which can be in transit.
    - In each window position, some of these sequence numbers defines the frames that have been sent; others define those that can be sent.
    - The maximum size of the window is 2^m-1.

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Send Sliding Window



a. Send window before sliding

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Send Sliding Window
    - The window at any time divides the possible sequence numbers into four regions.
    - First region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them.
    - The Second colored region defines the range of the sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. Referred as outstanding frames.

Noisy Channel

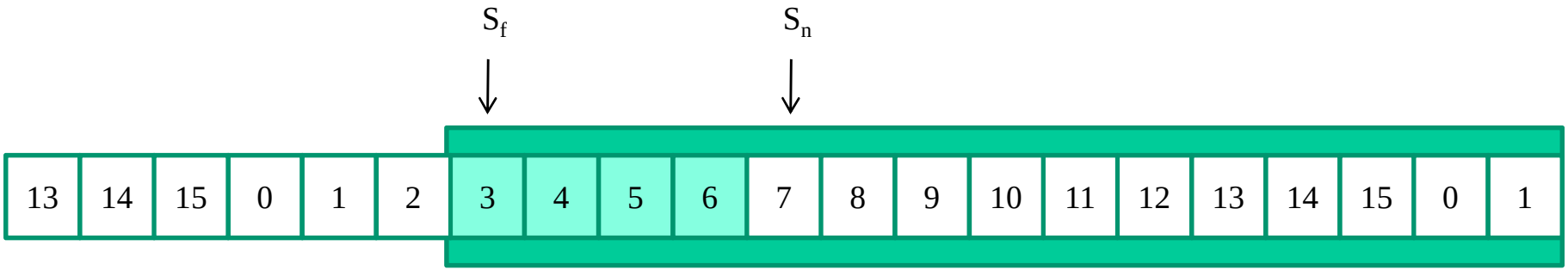
- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Send Sliding Window
    - The third region white in the figure, defines the range of sequence numbers for frames that can be sent; however the corresponding data packets have not yet been received from the network layer.
    - The fourth region defines sequence numbers that cannot be used until the window slides.

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Send Sliding Window
    - The window itself is an abstraction; three variables define its size and location at any time.
    - Variables are:
      - S_f (send window, the first outstanding frame)
      - S_n (send window, the next frame to be sent)
      - S_{size} (send window, size)

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Send Sliding Window



b. Send window after sliding

Noisy Channel

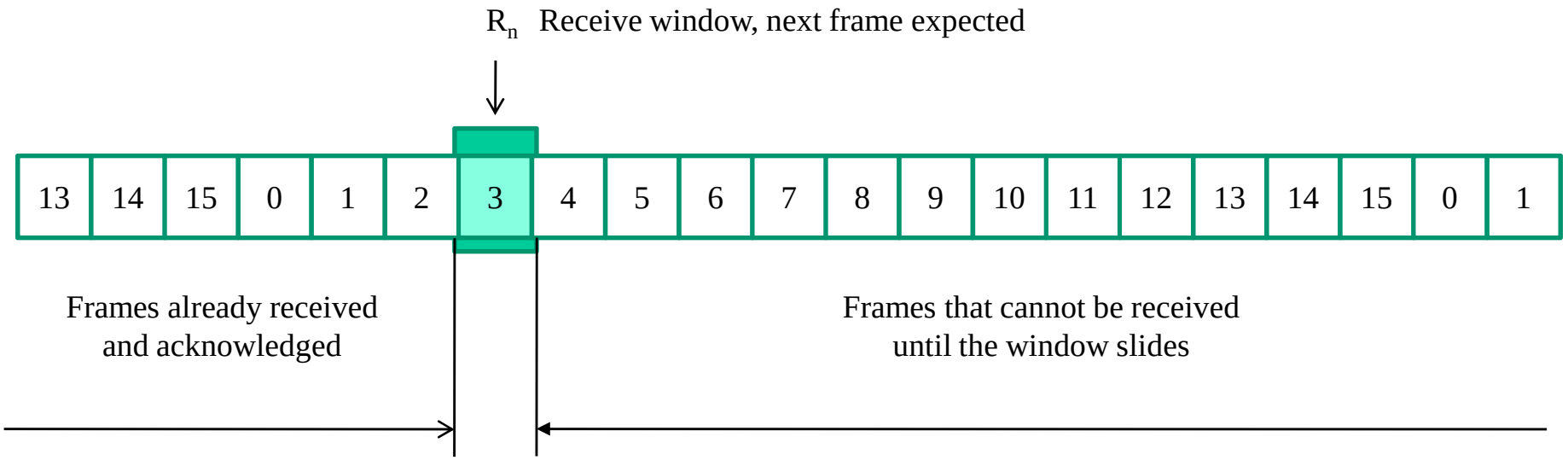
- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Send Sliding Window
    - How a send window can slide one or more slots to right when an acknowledgement arrives from the other end.
    - The acknowledgements in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame.
    - Here frames 0,1 and 2 are acknowledged, so the window has slide to the right three slots.
    - The value of S_f is 3 because frame 3 is now the first outstanding frame.

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Receive Sliding Window
    - Makes sure that the correct data frames are received and that the correct acknowledgements are sent.
    - The size of the receive window is always 1. The receiver is always looking for the arrival of a specific frame.
    - Any frame arriving out of order is discarded and needs to be resent.

Noisy Channel

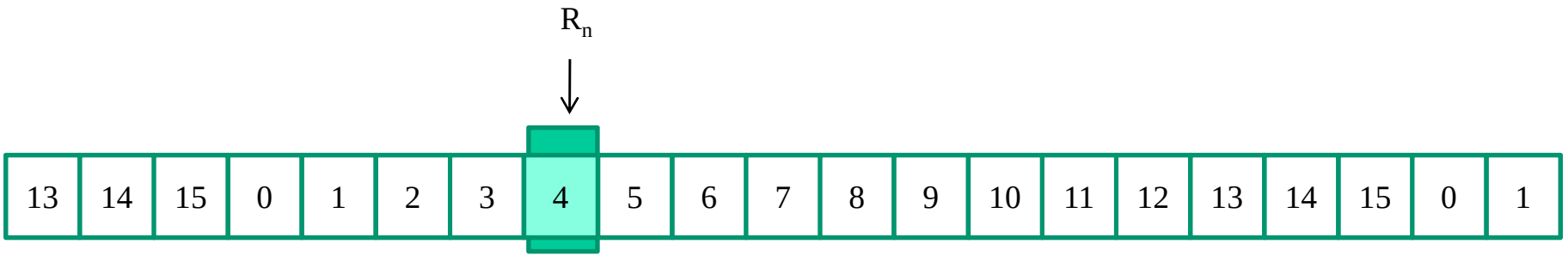
- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Receive Sliding Window



a. Receive Window

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Receive Sliding Window



b. Window after sliding

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Receive Sliding Window
    - We need only one variable R_n (receive window, next frame expected) to define this abstraction.
    - The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received.
    - Any received frame with a sequence number in these two regions is discarded.
    - Only a frame with a sequence number matching the value of R_n is accepted and acknowledged.
    - Receive window also slides but one slot at a time. When a correct frame is received, the window slides.

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Timers
    - Although there can be timer for each frame that is sent, in our protocol we use only one.
    - The reason that the timer for the first outstanding frame always expires first; we send all outstanding frames when this timer expires

Noisy Channel

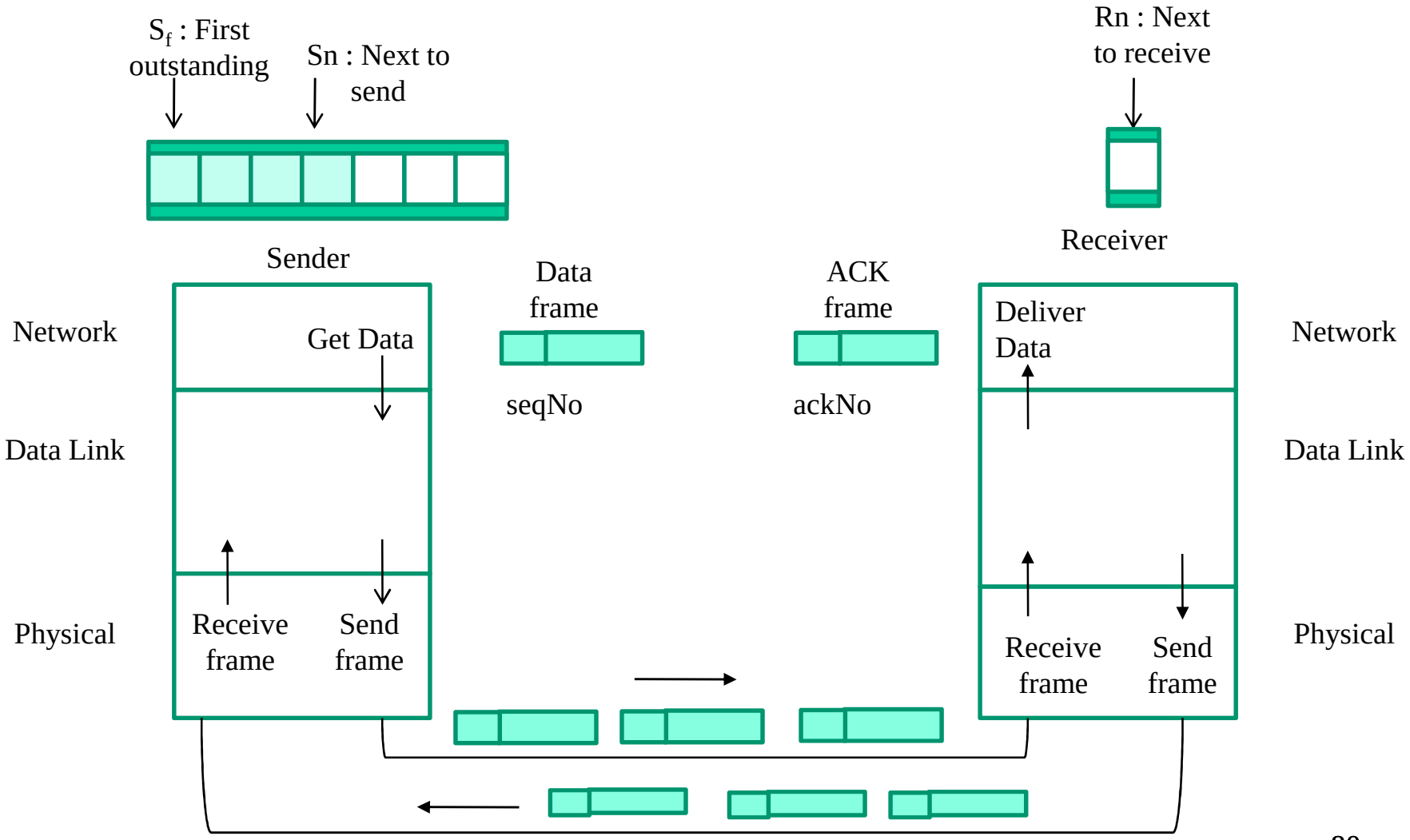
- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Acknowledgements
    - The receiver sends a positive acknowledgement if a frame has arrived safe and sound and in order.
    - If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting.
    - The silence of the receiver causes the timer of the unacknowledged frame at the sender site to expire.
    - 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

Noisy Channel

- Pipelining – Go-Back-N Automatic Repeat Request
  - Sliding Window – Resending a Frame
    - When the timer expires, the sender resends all outstanding frame.
    - Example: suppose sender has already sent frame 6, but the timer for frame 3 expires. Means that frame 3 has not been acknowledged; the sender goes back and sends frames 3,4,5, and 6 again.
    - Hence the protocol is called GO-Back-N ARQ.

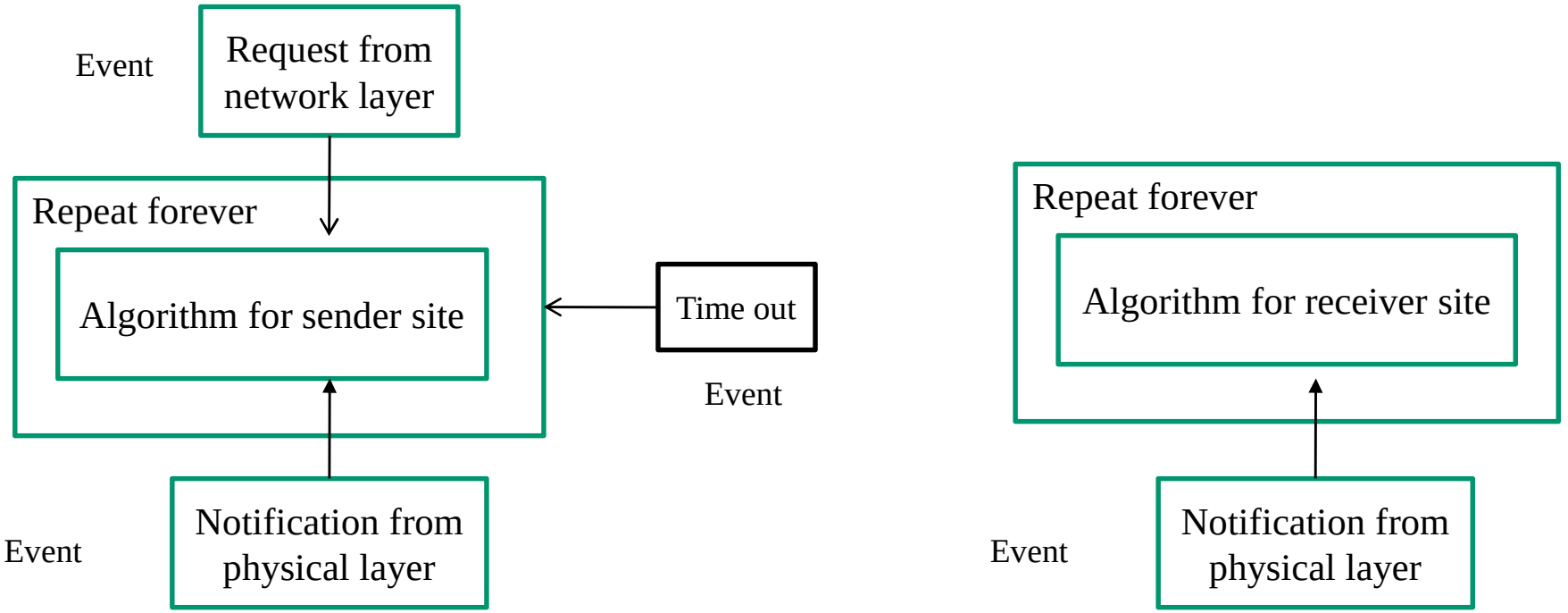
Noisy Channel

- Pipelining– Go-Back-N Automatic Repeat Request - Design



Noisy Channel

- Pipelining– Go-Back-N Automatic Repeat Request - Design



Noisy Channel

- Pipelining– Go-Back-N Automatic Repeat Request
  - Design
    - Multiple frames can be in transit in the forward direction, and multiple acknowledgements in the reverse direction.
    - Idea similar to Stop-and-Wait ARQ; the difference is that the send window allows us to have as many frames in transition as there are slots in the send window.

Noisy Channel

- Pipelining– Selective Repeat Automatic Repeat Request
  - Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded.
  - 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. Slows down the transmission.

Noisy Channel

- Pipelining– Selective Repeat Automatic Repeat Request
  - Another protocol works for noisy channel, which doesn't resend N frames when just one frame is damaged; only the damaged frame is resent.
  - Mechanism is called Selective Repeat ARQ.
  - More efficient for noisy links, but the processing at the receiver is more complex.

Noisy Channel

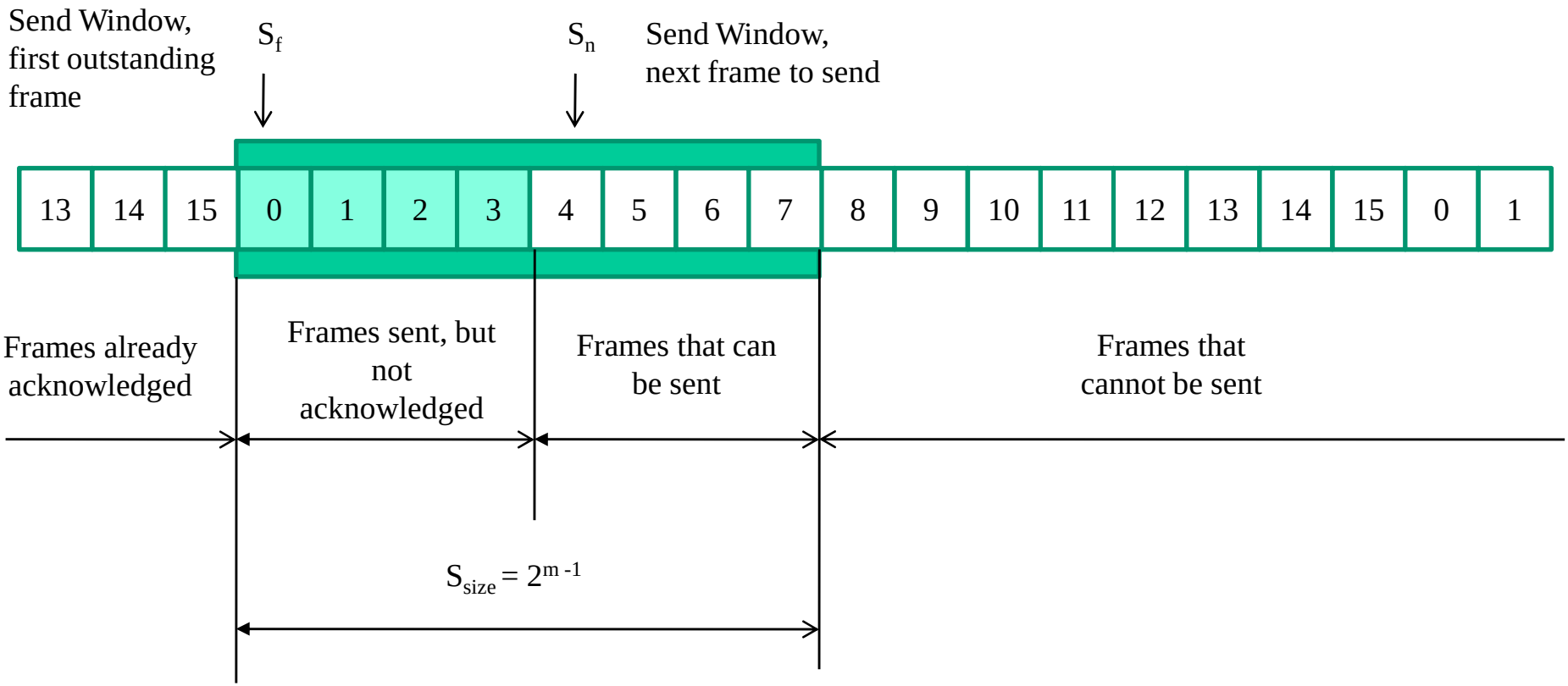
- Pipelining– Selective Repeat Automatic Repeat Request - Windows
  - Uses two windows: a send window and a receive window
  - There are differences between the windows in this protocol and the ones in GO-Back-N.
    - First – the size of the send window is much smaller, it is 2^{m-1}.
    - Second – the receive window is the same size as the send window.

Noisy Channel

- Pipelining– Selective Repeat Automatic Repeat Request – Windows – Send Window
  - The send window maximum size can be 2^{m-1}.
  - The smaller window size means less efficiency in filling the pipe, but the fact that there are fewer duplicate frames can compensate for this.
  - Protocol uses the same variables as for GO-Back-N.

Noisy Channel

- Pipelining– Selective Repeat Automatic Repeat Request – Windows – Send Window



Noisy Channel

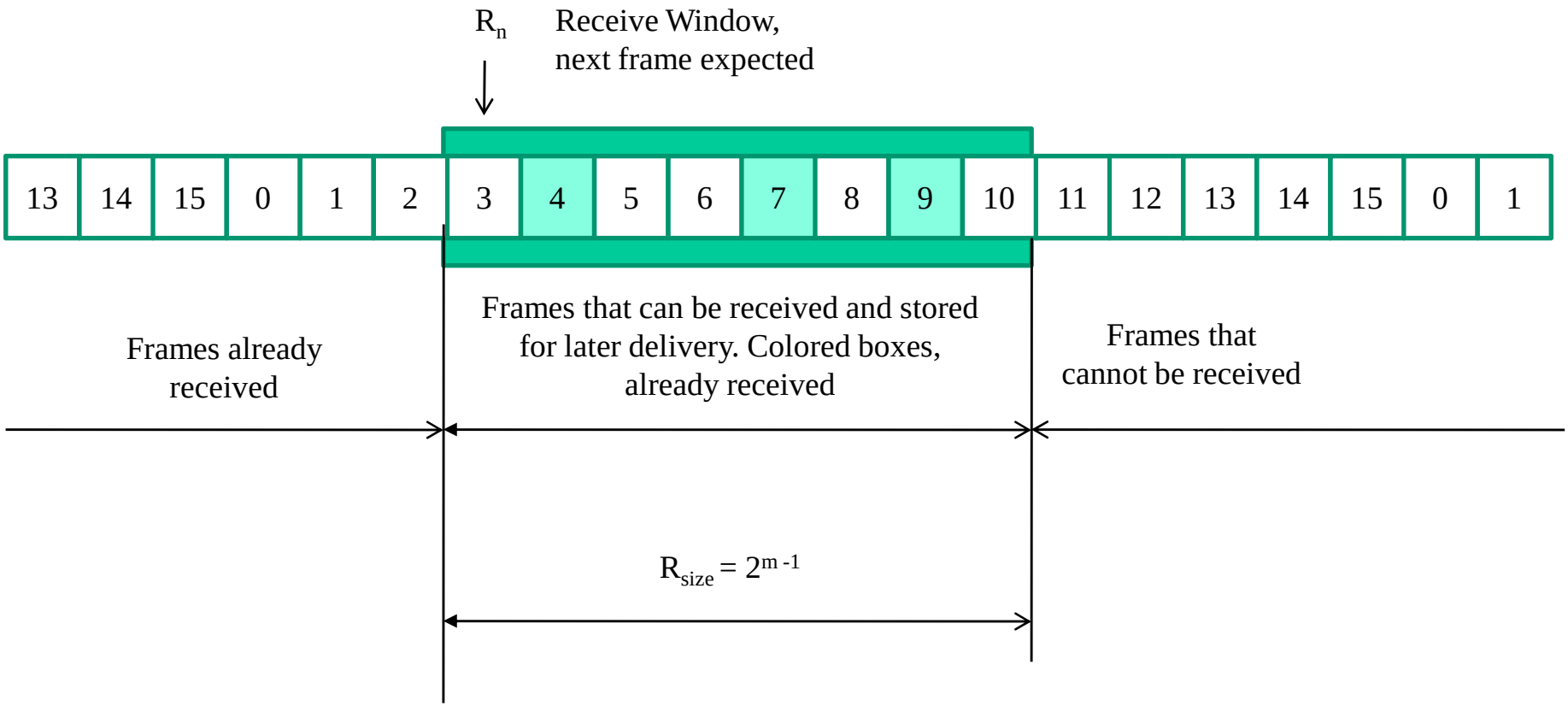
- Pipelining– Selective Repeat Automatic Repeat Request – Windows – Receive Window
  - Different from the one in Go-Back-N.
  - First, the size of the receive window is the same as the size of send window (2^{m-1}).
  - Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer.
  - Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered.

Noisy Channel

- Pipelining– Selective Repeat Automatic Repeat Request – Windows – Receive Window
  - The receiver never delivers packets out of order to the network layer.
  - Slots inside the window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

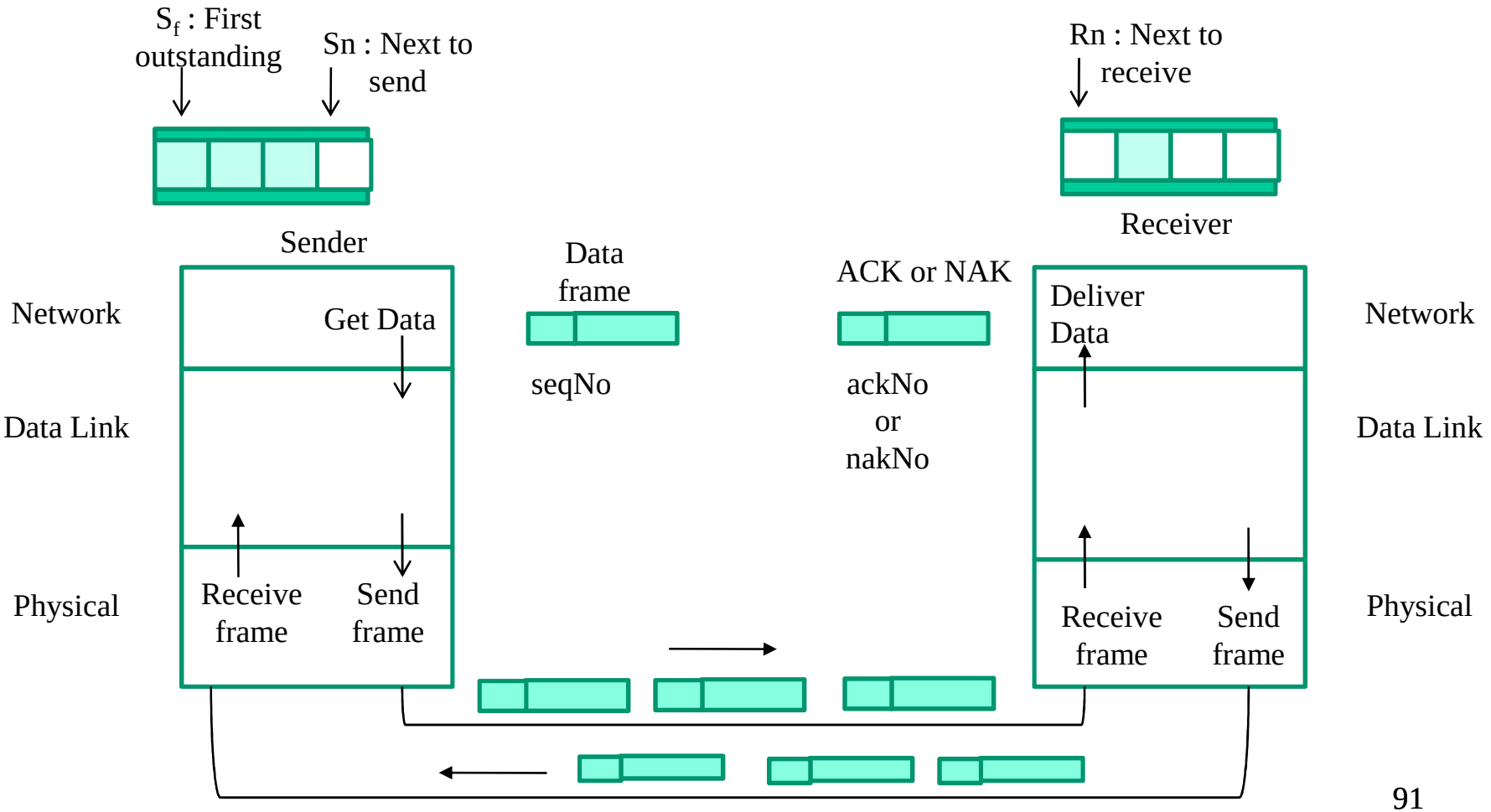
Noisy Channel

- Pipelining– Selective Repeat Automatic Repeat Request – Windows – Receive Window



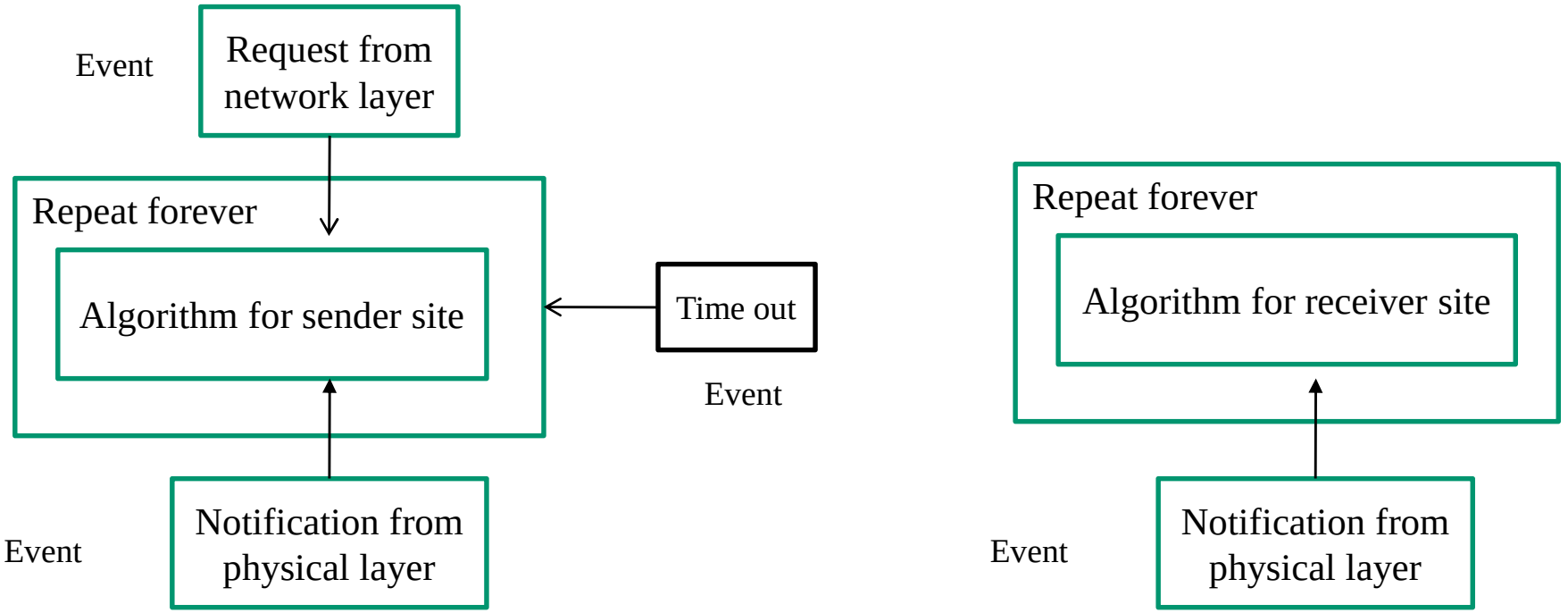
Noisy Channel

- Pipelining– Selective Repeat Automatic Repeat Request – Design



Noisy Channel

- Pipelining– Selective Repeat Automatic Repeat Request – Design



Noisy Channel

- Piggybacking
  - When a data frame arrives, instead of immediately sending a separate control frame, the receiver restrains itself and waits until the network layer passes it the next packet.
  - The acknowledgement is attached to the outgoing data frame. The acknowledgement gets a free ride on the next outgoing data frame.
  - The technique is temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as piggybacking.

Noisy Channel

- Piggybacking - Advantage
  - A better use of the available channel bandwidth.
  - The ack field in the frame header costs only a few bits, whereas a separate frame would need a header, the acknowledgement and a checksum.

Noisy Channel

- Piggybacking - Disadvantage
  - How long should the DLL wait for a packet onto which to piggyback the acknowledgement?
  - If the DLL waits longer than the sender's timeout period, the frame will be retransmitted, defeating the whole purpose of having acknowledgements.

References

- Andrew S. Tanenbaum, Computer Networks, Pearson Education. [4th Edition]
- Andrew S. Tanenbaum, David J. Wetherall, Computer Networks, Pearson. [5th Edition]
- Behrouz A. Forouzan, Data Communications and Networking, Tata McGraw Hill. [3rd Edition]
- Behrouz A. Forouzan, Data Communications and Networking, Tata McGraw Hill. [4th Edition]