Communication Networks

MAP-TELE

2011/12

José Ruela

Network basic mechanisms

Error Control

Error Detection

Error control – goals

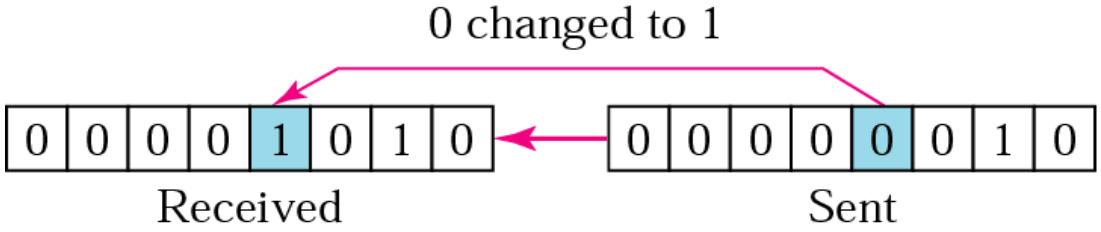
- The main goal of an error control mechanism is to provide a reliable transfer of data units between systems (that is, in-sequence delivery, without losses or duplicates)
- Error control mechanisms are usually implemented at the Data Link layer and at the Transport layer – they adopt similar principles, but there are some important differences
  - The Data Link layer deals with errors that affect the communication between adjacent systems over a physical channel – frames are not misordered by the channel (but the receiver may receive out-of-sequence frames) and there are no delays introduced by intermediate nodes
  - The Transport layer operates on an end-to-end basis and deals with errors due to the network operation – packets may be lost, misordered and misrouted and the network introduces variable delays due to queuing on network nodes
- Without loss of generality we shall describe and analyse error control from the perspective of the Data Link layer (and, at the end, we shall refer to particular aspects related with the Transport layer)
  - Error control procedures include error detection and error recovery mechanisms

Error detection

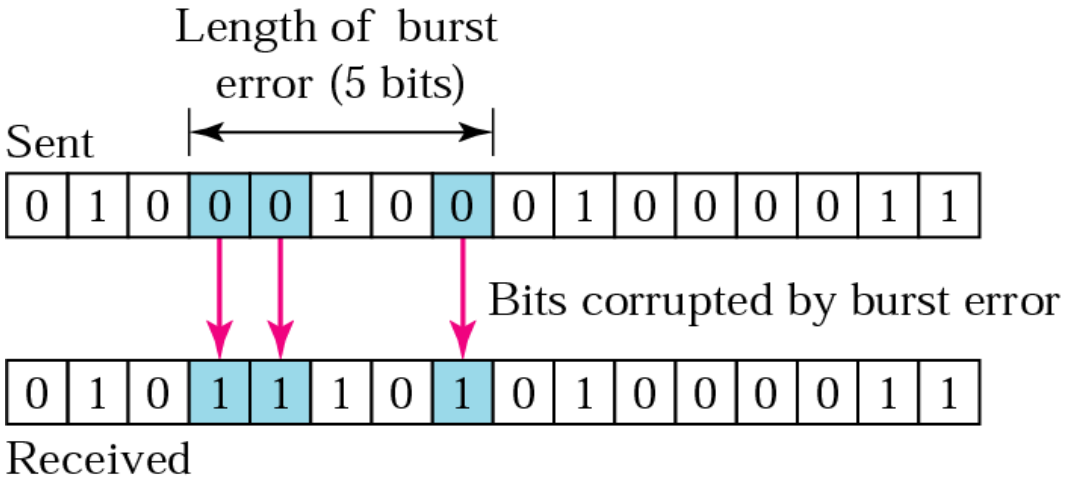
- Data frames may be corrupted on a transmission medium and a received frame may differ from the transmitted one due to a variety of reasons
  - Due to transmission impairments one or more bits may have been erroneously received – such errors may occur randomly (independent errors) or in bursts
  - Due to synchronization problems, bits may be added to or removed from the original sequence
  - A frame may be so severely corrupted that its structure is affected, e.g., one frame may be split in two or more frames (and conversely) – this could occur, for example, when bit errors create false flags or destroy real flags
- Error detection mechanisms cope with these situations and it is expected that they will catch most of these errors – the corresponding frames will be discarded and it is up to error control mechanisms to recover
- Some errors may not be detected by the error detection mechanism and thus erroneous frames are accepted for processing at the Data Link layer
  - In some cases such errors may be detected at the Data Link layer (e.g., a wrong value occurs in a header field), at a higher layer or remain undetected
  - If these errors are not detected at the Data Link layer, erroneous frames are handled by the protocol as if they were valid ones

Types of transmission errors

- Single bit errors
  - Occur in a random way and are independent



- Burst errors
  - Are not independent and affect close by bits
  - The burst length is determined by the first and last corrupted bits



Error detection principles

- Error detection techniques use the concept of redundancy, which consists in adding *k* redundant bits to *n* information bits, according to an algorithm known by the sender and the receiver
  - Different algorithms have different error detection properties
- The sender applies the algorithm to the *n* information bits to generate the *k* redundant bits and transmits a total of *n* + *k* bits
- The receiver applies the same algorithm to the *n* information bits it receives – the *k* redundant bits generated by the receiver are compared with the corresponding *k* bits really received
  - If they are different it is certain that an error has occurred (single or multiple errors in any bit position)
  - If they are equal, either there was no error or an undetected error has occurred (beyond the detection capacity of the algorithm) – it is desirable that the probability of undetected errors is low

Single parity check

- In single parity check, a parity bit is added to *n* information bits, so that the total number of ones is even (*even parity*) or odd (*odd parity*)
  - This simple algorithm allows detecting single bit errors and more generally an odd number of errors in the block of *n+1* bits
  - The algorithm cannot detect errors that occur on an even number of bits on the block
- This method is used in character oriented protocols – one example is the use of ASCII characters with 7 bits of information and one extra parity bit for a total of 8 bits per character
- Considering the following sequence of 7 bit ASCII characters

1110111 1101110 1010110 1101100 1100100

the actual bits sent will be (assuming even parity per character)

11101110 11011101 10101100 11011000 11001001

Two-dimensional parity check

- In two-dimensional parity check, a block of bits is divided into rows (each row typically represents a character with a single parity bit) and a redundant row of bits is added to the whole block
- Example (for the same set of five characters)

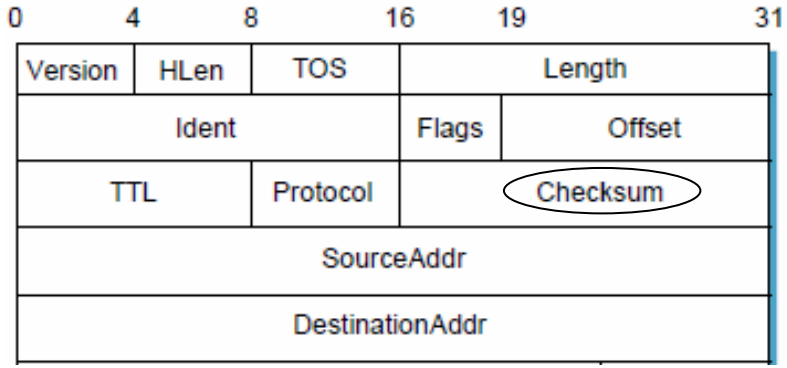
	1110111	**0**
	1101110	**1**
	1010110	**0**
	1101100	**0**
	1100100	**1**
Parity byte	**1000111**	**0**

Parity bits

- Two-dimensional parity check detects all single bit errors, two and three bit errors and most four bit errors

Checksum

- The information to be protected is organized in words (for example 16 bit words)
- All data words to be transmitted are added up and the sum (called *checksum*) is transmitted following the data words
- The receiver calculates the checksum of the received data words and compares it with the received checksum
- This method is used to protect the header of IP packets
  - The addition is performed in one's complement arithmetic over 16 bit words (the default header has ten 16 bit words)
  - The checksum is the one's complement of the result of the addition



2 and 1's complement arithmetic

- 2's complement arithmetic

Decimal	Binary
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
-1	1111 1111
-2	1111 1110
-3	1111 1101

(- x) is obtained by inverting all bits in (+ x) and adding 1

- 1's complement arithmetic

Decimal	Binary
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
-0	1111 1111
-1	1111 1110
-2	1111 1101
-3	1111 1100

(- x) is obtained by inverting all bits in (+ x)

- Example: - 3 + 5 = 2

```
  1111 1101
  0000 0101
-------------
(1) 0000 0010
```

(carry in the sum is discarded)

```
  1111 1100
  0000 0101
-------------
(1) 0000 0001
(add carry)  1
-------------
  0000 0010
```

(carry in the sum is added to LSB)

Cyclic Redundancy Check (CRC)

- Cyclic Redundancy Check (CRC) methods are based on polynomial division
- A data unit with *n* information bits is represented by a polynomial M(x) with degree *n - 1* – each bit is the coefficient of a polynomial term
- The method requires a divisor polynomial C(x) with degree *k* (the number of redundant bits to be added)
  - The error detection properties of the method depend on the choice of C(x)
- The *k* redundant bits are generated by imposing that the polynomial P(x) that represents the $n+k$ bits to be transmitted is a multiple of C(x), that is, when P(x) is divided by C(x) the remainder should be zero
- The receiver simply divides the polynomial that represents the $n+k$ received bits by C(x) and, in case the remainder is not zero, it can conclude that an error has occurred – otherwise, there was no error, or the polynomial E(x) that represents the error pattern is divisible by C(x)

Detection properties of CRC codes

- CRC codes can detect
  - All single-bit errors, provided that the x^k and x^0 terms of C(x) have non-zero coefficients
  - All double-bit errors, if C(x) contains a factor with at least three terms
  - Any odd number of errors, if C(x) contains the factor $(x+1)$
  - Any burst error with length up to *k* bits, if C(x) has degree *k*
  - Most burst errors of length larger than *k* bits, if C(x) has degree *k*

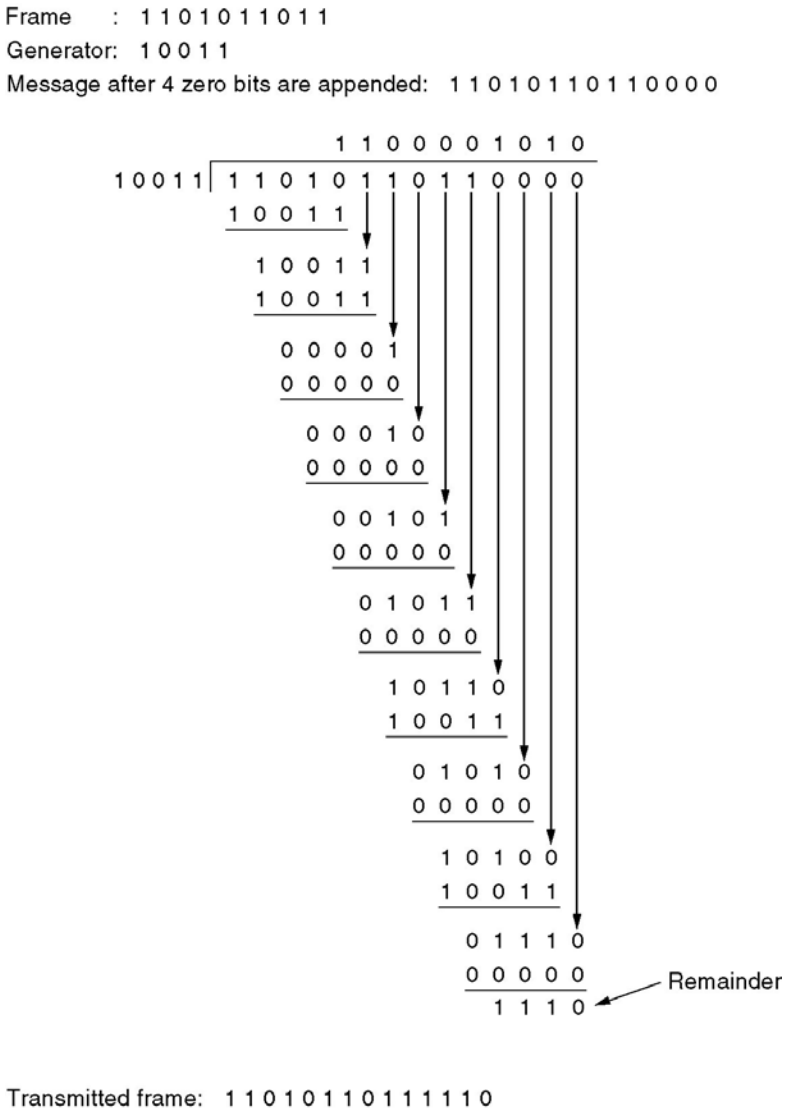
- Examples of CRC polynomials

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	**ATM header**
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	**ATM AAL**
ITU-16	$x^{16} + x^{12} + x^5 + 1$	**HDLC**
ITU-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	**LANs**

Polynomial division

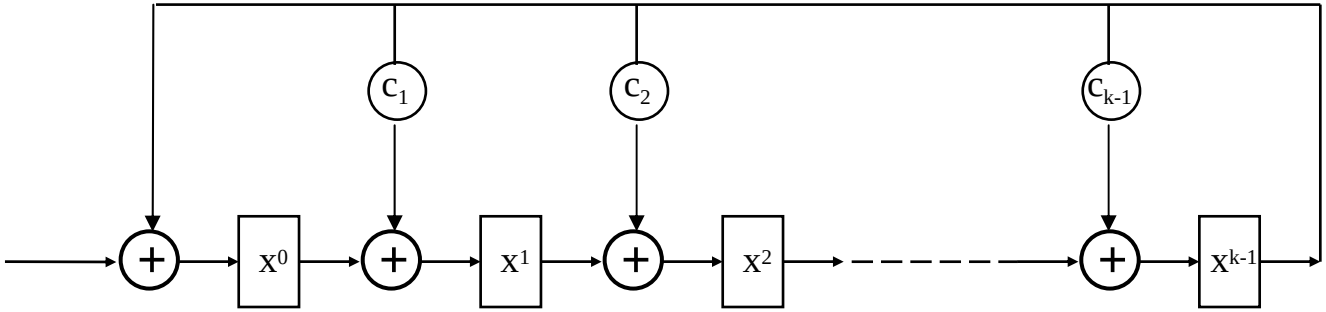
- Polynomial division is performed in modulo 2 arithmetic
  - Addition and subtraction are performed bit by bit, without carry – they are identical and equivalent to a bit by bit XOR
- In the example of CRC generation, a number of zeros equal to the degree *k* of the generator polynomial C(x) is added to the coefficients of the polynomial M(x) that represents the data bits
  - This corresponds to multiplying M(x) by x^k
- The remainder of the division has at most degree *k-1*, and thus *k* coefficients
  - The remainder of the polynomial division represents the CRC generated, which occupies, for transmission purposes, the *k* positions initially filled with zeros
- The subtractions in the successive steps of the polynomial division are performed as bit by bit XOR operations

Example of CRC generation

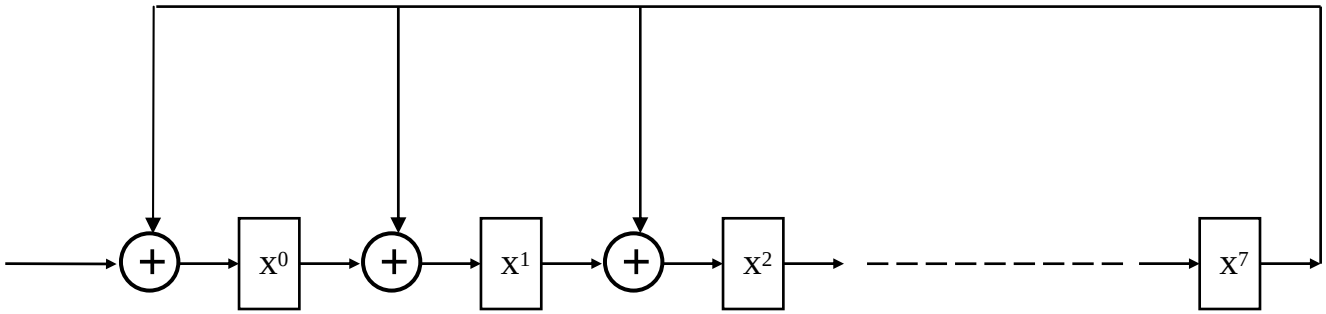


CRC generation – examples

- Simplified generic circuit, assuming that the coefficients c_k and c_0 of C(x) are not null



- Example for the generator polynomial $C(x) = x^8 + x^2 + x + 1$ (CRC-8 / ATM *header*)



Error Control

Error Recovery (ARQ)

Error control at the Data Link layer

- In general, error control performed at the Data Link layer deals with lost or discarded frames, which may have other causes than transmission errors
  - Interruption of the physical media may lead to loss of one or more frames
  - A valid frame may be discarded at the receiver due to buffer overflow
  - A valid frame may be discarded at the receiver as determined by protocol procedures (e.g., error or flow control)
- Error control mechanisms implemented at the Data Link layer are intended to provide a reliable service that recovers from frame losses – recovery actions depend on the type of the frame lost, the protocol state and the specific recovery mechanism
- Even if the Data Link layer provides a reliable service, the Network layer may loose packets (especially in Datagram networks) or even misroute packets (packets are missing at the intended destination and are wrongly delivered to another destination)
  - In these cases recovery is provided by the Transport Layer (if a reliable service is required by the Applications)

Types of Data Link frames

- Frame designations adopted by standard Data Link protocols (such as HDLC) will be used to ease the explanation
- At the Data Link layer, Information frames (I-frames) are used to carry packets on their payload
- Error control mechanisms require that the data link stations exchange Control or Supervisory frames (S-frames)
  - S-frames are also used to support flow control
- All frames are protected by an error detection code – if an error is detected, the common practice in bit and byte oriented protocols is to discard the frame without further processing
  - In character oriented protocols erroneous frames are also discarded but, in some cases, it is possible to take actions based on recognized control characters carried on the frames
- In the following examples we assume that one station acts as sender of I-frames and the other as receiver but, in general, a station plays both roles (I-frames may be sent simultaneously in both directions)

Basic strategy

- I-frames that for some reason are not accepted by the receiver must be retransmitted
  - The sender must keep a copy of each transmitted I-frame, for a possible retransmission, until it is sure that the frame was accepted by the receiver
- In the basic strategy the receiver issues positive acknowledgements (ACK) for I-frames that it correctly received and accepted
  - The sender discards the copy of each I-frame for which it received an ACK
- If an ACK is not received within a predefined time interval (*time-out*) the sender assumes that an I-frame may have not been accepted by the receiver and thus may have to be retransmitted
  - Retransmitting an I-frame after a time-out may create a duplicate, since it is possible that the frame had been correctly received but the corresponding ACK reply was lost – error control must detect and eliminate such duplicates
- It is also possible, under certain conditions, for the receiver to explicitly ask for the retransmission of I-frames
  - This may be understood as a negative acknowledgement (NAK) – a term adopted in earlier character oriented protocols
- Bit oriented protocols provide a set of S-frames to support ACKs and NAKs, and ACKs may also be piggybacked on I-frames on the reverse direction

Frame numbering

- Error control mechanisms require the identification of I-frames correctly received and accepted (so that they can be acknowledged) and of I-frames that were not received or not accepted by the receiver (so that they can be retransmitted)
- Data Link layer protocols must thus provide a numbering scheme for identifying I-frames
  - Flow control mechanisms have a similar requirement and may share the numbering scheme with error control
- I-frames carry on their header a sequence number N(s)
- S-frames carry on their header a sequence number N(r) that is used to refer to I-frames in the reverse direction
  - I-frames also carry an N(r) to support piggybacking of ACKs
- By convention N(r) indicates the next in-sequence N(s) expected by the receiver
  - For example, to acknowledge the reception of I-frame with N(s) = 3, the receiver replies with N(r) = 4 carried on an S or I-frame – in fact this also acknowledges other previous I-frames that might be outstanding (not yet acknowledged)

Numbering scheme

- The sequence numbers N(s) and N(r) are coded in the control field of Information and Supervisory frames
- When using *k* bits for this purpose, the total number of identifiers is $M = 2^k$ and their possible values are (0, 1, 2, …, M-1)
- To reduce the overhead, a small number of bits is used – typical values for *k* are 1, 3 and 7 and the corresponding values for M are 2, 8 and 128, respectively
- These identifiers have to be reused, since they must repeat cyclically
  - One immediate consequence is that it is not possible to have two different outstanding I-frames (transmitted but not yet acknowledged) bearing the same sequence number
  - The maximum number of outstanding I-frames depends on M and on the retransmission strategy (but is always less than M, as will be demonstrated)

State variables

- To keep the correct references for numbering N(s) and N(r), each station may use two state variables V(s) and V(r)
- V(s) indicates the number of the next in-sequence I-frame to be sent by a station
  - When transmitting a new I-frame, N(s) is filled with the current value of V(s)
  - V(s) is incremented after the transmission of an I-frame (preparing for the transmission of the next one)
  - When I-frames have to be retransmitted, V(s) has to be updated
- V(r) indicates the number of the next in-sequence I-frame expected by a station
  - When sending a frame that contains N(r), this field is filled with the current value of V(r)
  - V(r) is incremented after the reception and acceptance of the expected I-frame, that is, with N(s) = V(r), and before acknowledging it
- V(s) and V(r) are initially set to 0

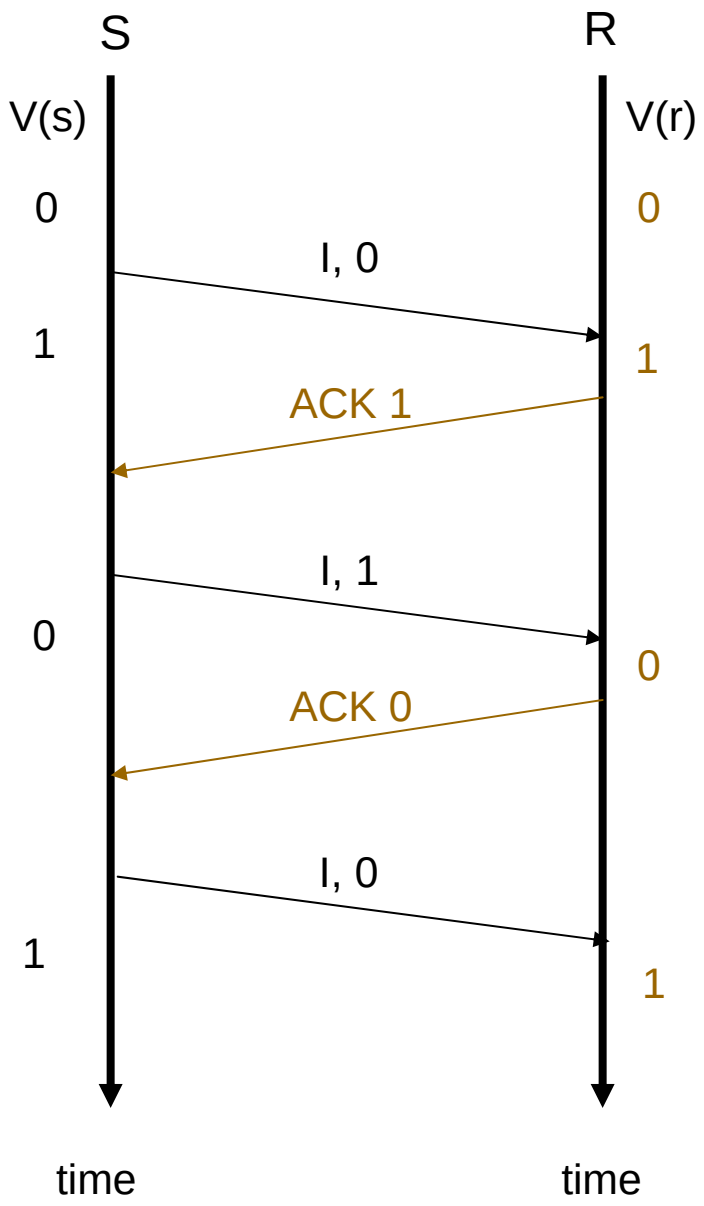
Automatic Repeat Request (ARQ)

- The most usual strategy to support reliable delivery of I-frames is called *Automatic Repeat Request* (ARQ) and is based on two basic mechanisms (acknowledgements and time-outs) that may trigger retransmissions
- There are two classes of ARQ mechanisms – *Stop and Wait* and *Sliding Window*
- *Stop and Wait* limits the maximum number of outstanding I-frames to one
- *Sliding Window* allows multiple outstanding I-frames and has two variants, *Go-Back-N* and *Selective Repeat* (also called *Selective Reject*)
  - The window size (W) imposes a limit on the maximum number of I-frames that can be outstanding
  - *Go-Back-N* and *Selective Repeat* adopt different error recovery strategies and are characterized by different maximum possible values for the window size (W_{max}) and thus the configured window size must be $W \leq W_{max}$
  - *Stop and Wait* may be characterized as having $W_{max} = 1$
- To better understand ARQ protocols, we start by describing error-free operation and then consider scenarios that require error recovery actions

Stop and Wait – error-free operation

- The basic error-free operation may be described as follows
  - After transmitting an I-frame the sender must wait for a positive acknowledgement (ACK) from the receiver before sending a new I-frame
  - The receiver must reply with ACK after receiving and accepting a valid I-frame
  - After receiving ACK, the sender is allowed to transmit a new I-frame
- Such a simple mechanism would suggest that numbering I-frames and acknowledgements would not be necessary – this is not the case, since for correct error recovery both I-frames and ACKs need to be numbered (as will be demonstrated with examples)
- It is enough to use a single bit for numbering frames and thus I-frames will be alternately numbered 0 and 1
- In character oriented protocols, the special ACK character may be used for acknowledgements – however to number such acknowledgements escape sequences are used (DLE 0/1, meaning ACK 0/1, respectively)
- In bit and byte oriented protocols acknowledgements are carried on Supervisory frames, which also perform other functions – under normal operation, acknowledgements are carried on *Receiver Ready* (RR) frames or piggybacked on I-frames

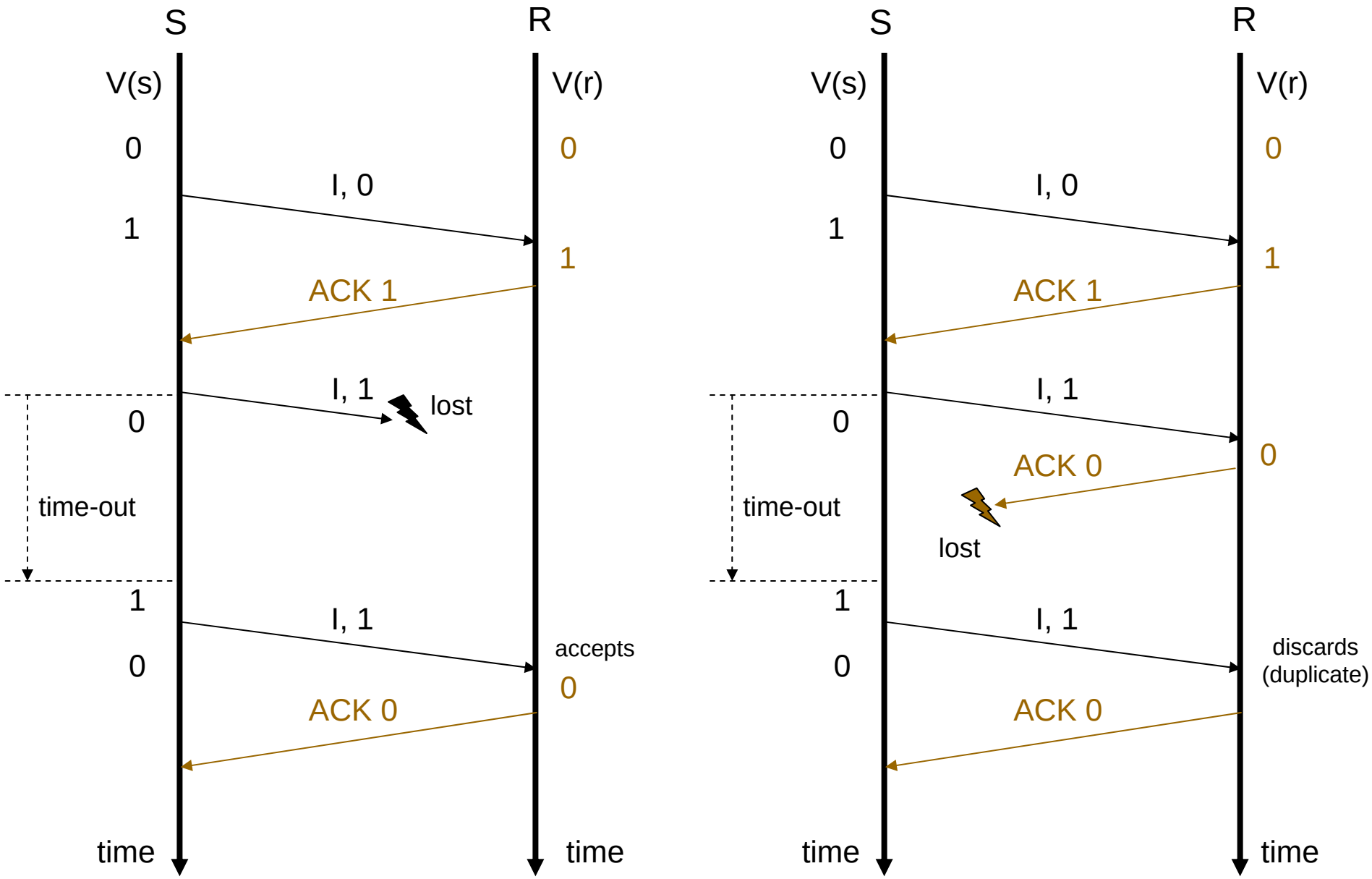
Stop and Wait – example of error-free operation



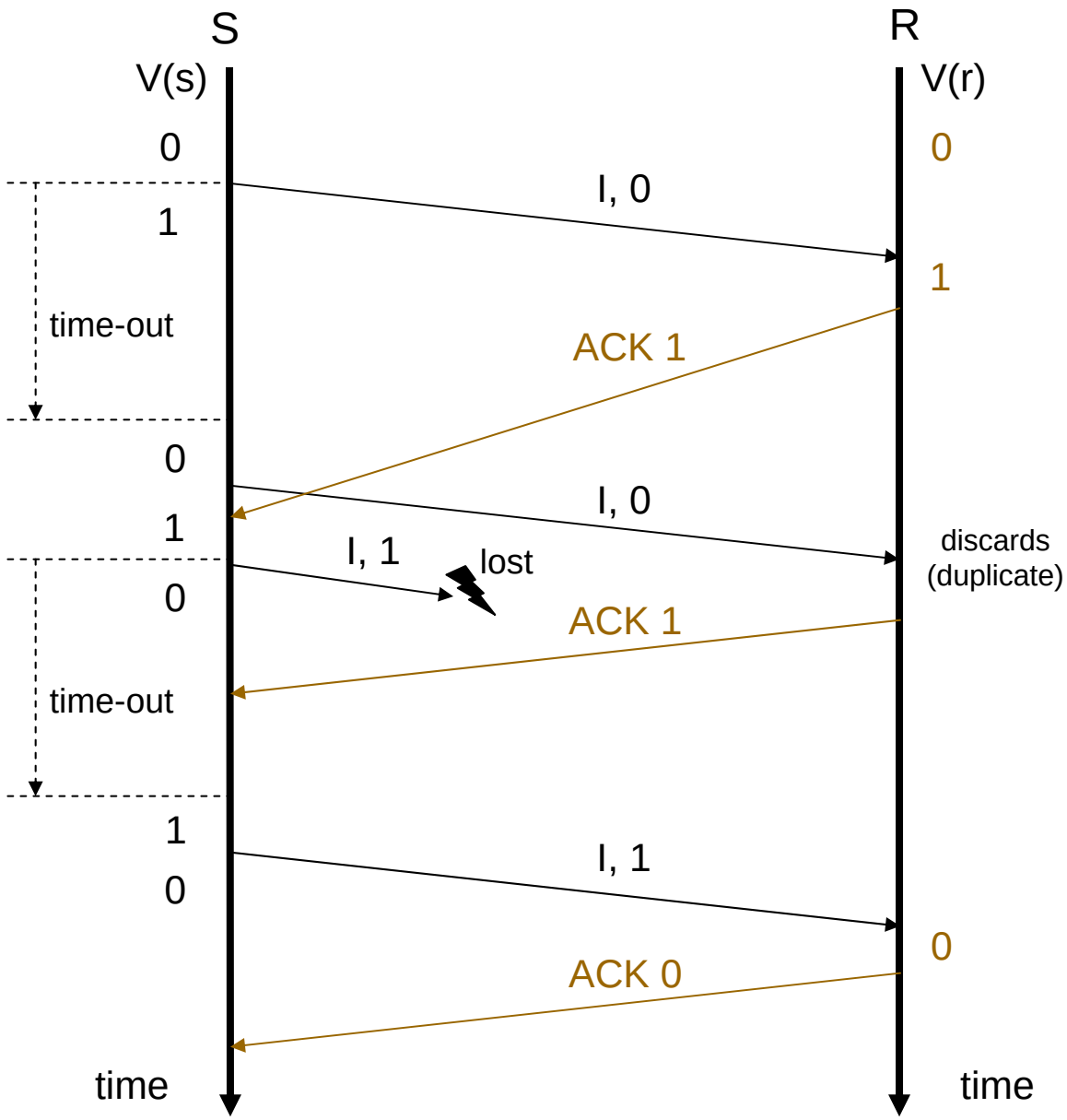
Stop and Wait – error recovery

- Due to transmission errors, one of two cases may occur
  - An I-frame transmitted by the sender is corrupted and thus not accepted by the receiver – in this case no ACK reply is generated
  - An I-frame transmitted by the sender is correctly received and accepted by the receiver, an ACK reply is generated but is corrupted
- In both cases the sender does not receive the expected ACK, but it has no way of knowing which case really happened
- After a time-out it must retransmit the I-frame – otherwise a frame loss could occur (case 1), but there is the risk of creating a duplicate (case 2)
- To allow the detection of a duplicate by the receiver, I-frames must be numbered
  - When detecting a duplicate, the receiver discards it, but generates the corresponding ACK reply (since the previous one may have been lost)
- It is also possible that ACKs are delayed beyond the time-out interval
  - ACKs must also be numbered to avoid ambiguities that will arise when an I-frame sent after receiving a delayed ACK is corrupted

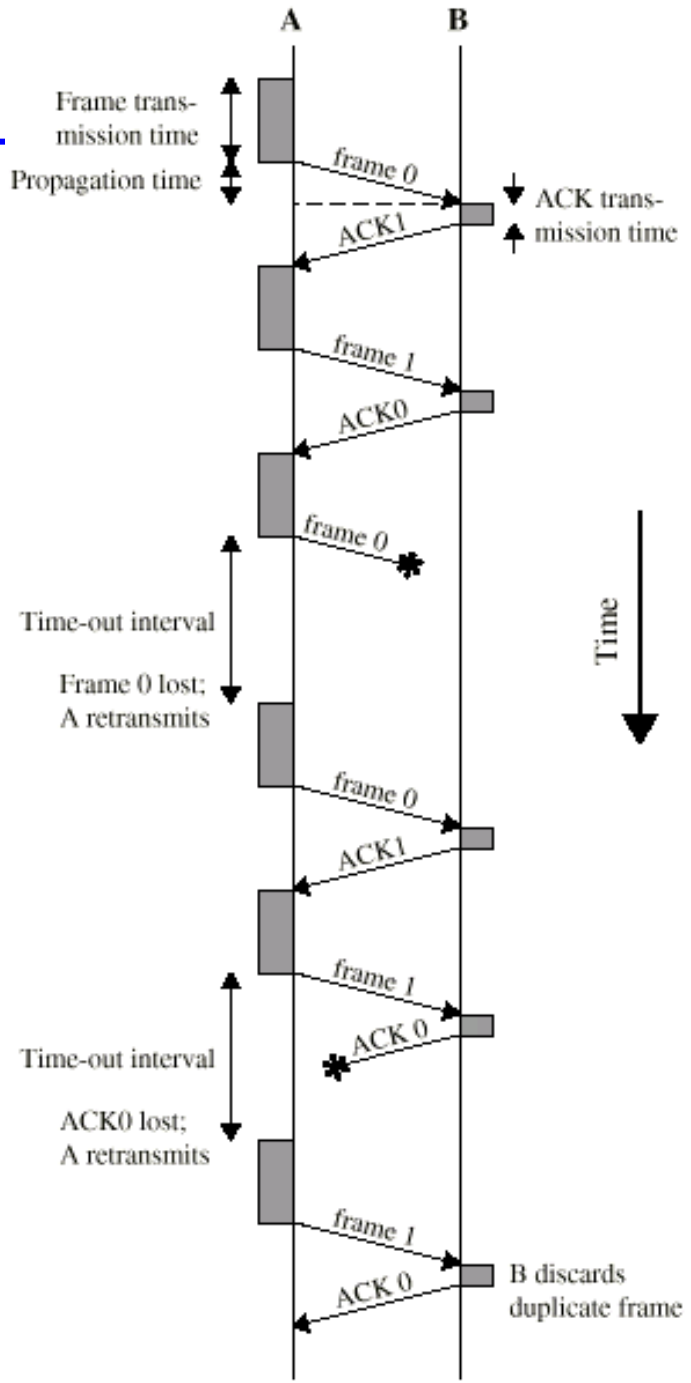
Stop and Wait recovery – lost I-frame or ACK



Stop and Wait recovery – delayed ACK and lost I-frame



Stop and Wait – example

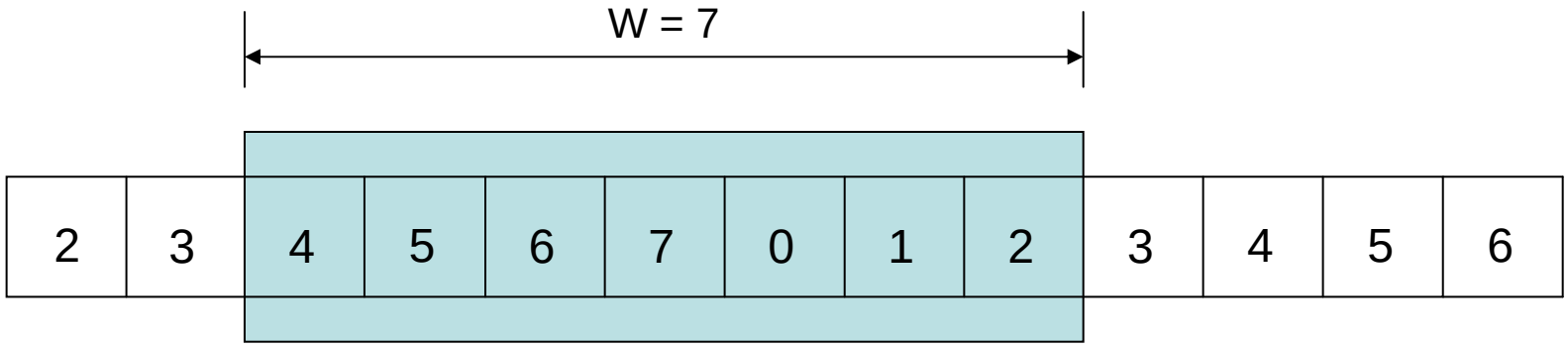


Sliding Window

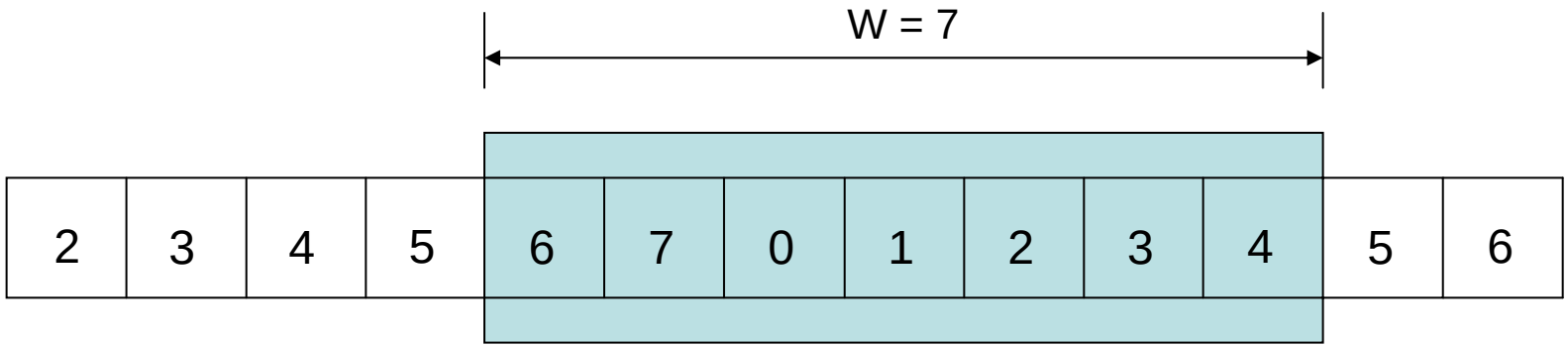
- In Sliding Window mechanisms the sender is allowed to transmit multiple I-frames before receiving ACKs for the outstanding I-frames
- The maximum number of possible outstanding I-frames is equal to the size of the window (W) in use, agreed between sender and receiver
  - When the window limit is reached, the sender must stop and wait for acknowledgements
- The receiver may acknowledge separately each received I-frame but consecutive I-frames may be acknowledged in blocks
  - This is possible since N(r) is filled with the current value of V(r), which avoids sending outdated ACKs, and is useful when an ACK is lost but a subsequent ACK is received by the sender before a time-out occurs
  - When the sender receives a frame that carries an N(r) value, all outstanding I-frames with sequence numbers up to (but excluding) N(r) are automatically acknowledged
- When an ACK is received, the sender window is rotated – both the lower and the higher edges are updated by an amount corresponding to the number of outstanding I-frames really acknowledged

Window rotation

- Before rotation



- After rotation (frames 4 and 5 acknowledged)



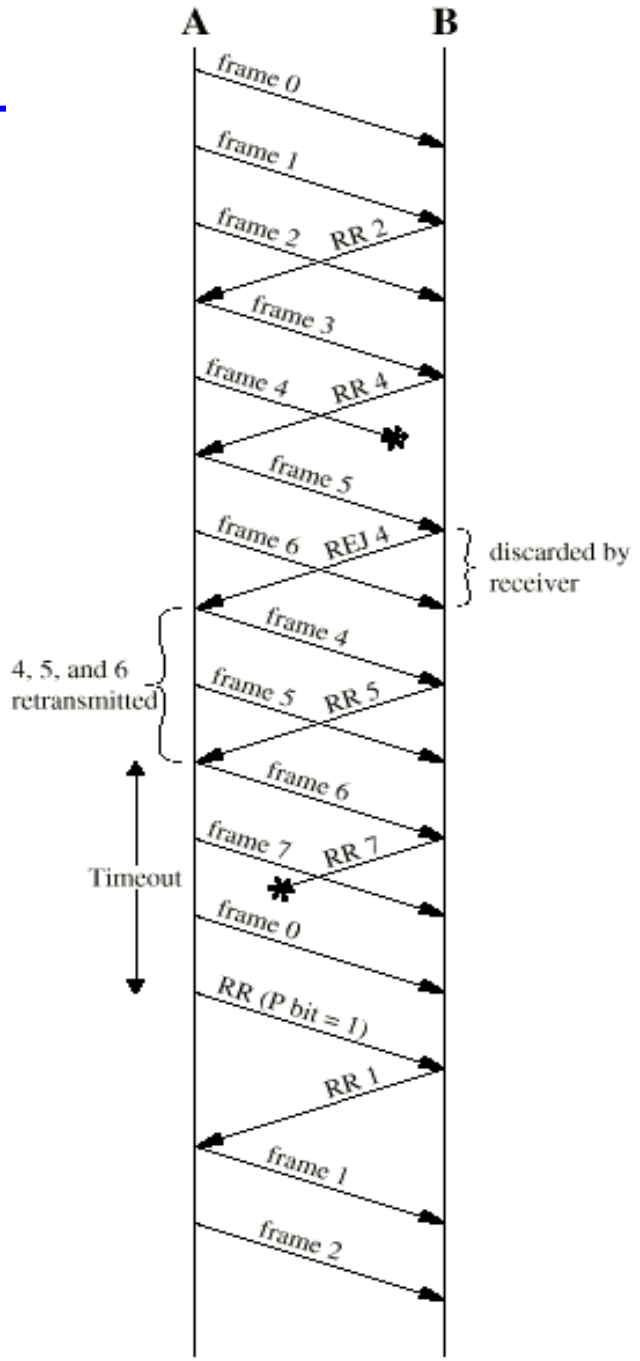
Go-Back-N

- In Go-Back-N the receiver only accepts in-sequence I-frames
  - A valid I-frame is accepted only if its sequence number N(s) is equal to the current value of V(r) kept by the receiver
  - Any valid I-frame whose sequence number N(s) is different from V(r) is simply discarded – this means that one or more I-frames have been lost and a recovery action may be initiated by the receiver
- A recovery action is initiated by the receiver only after detecting the first out of sequence I-frame (this means that one or more I-frames have been lost)
  - The receiver asks for a retransmission of the first (oldest) missing I-frame, which is indicated by the current value of V(r), and of all subsequent I-frames
  - For this purpose a *Reject* (REJ) S-frame is used – the sequence number N(r) indicates to the sender where to start retransmission in the sequence of I-frames (and simultaneously acknowledges previous outstanding frames, if any)
  - The receiver will only send one REJ for each recovery action it initiates – this exception condition is cleared as soon as the first missing frame is correctly received, after being retransmitted (this means that in-sequence reception of frames has been resumed and will continue with frames previously discarded)
- When the sender receives a REJ frame, it goes back and starts to retransmit all I-frames from the one indicated by the sequence number N(r)
  - A REJ frame plays the role of a negative acknowledgement (NAK)

Go-Back-N – maximum window size

- Since the receiver only accepts in-sequence I-frames, the receiver window size is one – it only accepts the I-frame with the expected sequence number
- On the other hand, based on the current value of V(r) and the sender window size (W), the receiver knows which are the next W frames the sender may transmit, assuming that the sender had already received the last acknowledgement corresponding to V(r)
- However, the window at the sender may have not been updated due to missing or delayed ACKs and therefore the receiver would not be able to distinguish the new (in-sequence) expected I-frame from an old I-frame being retransmitted by the sender and bearing the same sequence number
- This is the reason why the maximum sender window size cannot be equal to $M = 2^k$, but has to be reduced to $W_{max} = M - 1 = 2^k - 1$
- A similar argument applies to *Stop and Wait* ($M = 2$ and $W_{max} = 1$)

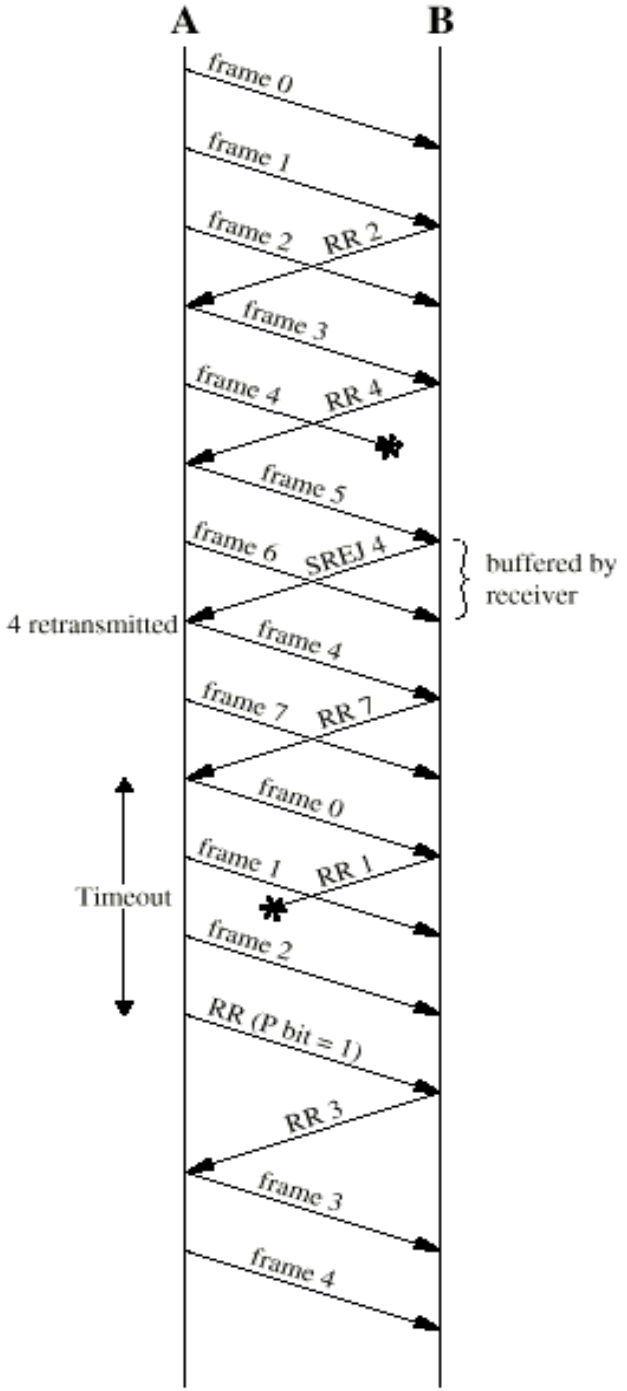
Go-Back-N ARQ – example



Selective Repeat

- In Selective Repeat the receiver accepts out of sequence I-frames
- Retransmission of out of sequence I-frames is requested by means of *Selective Reject* (SREJ) S-frames
  - The sender only retransmits the I-frame indicated by the N(r) field carried on each SREJ, thus reducing the number of retransmissions
- The receiver only acknowledges blocks of consecutive I-frames (as in Go-Back-N) and thus it has to wait for missing frames and fill the corresponding gaps before acknowledging the I-frames that had been received out of sequence and stored
- This method is more complex than Go-Back-N but is advantageous in links with a high Delay $*$ Bandwidth product (e.g., satellite links), for which large window sizes are recommended, or in links with high *Bit Error Ratio* (BER), even if small window sizes are acceptable
- To allow the receiver to distinguish new out of sequence I-frames from duplicates originated by the retransmission of I-frames already acknowledged, the maximum sender window size has to be reduced to $W_{max} = M / 2 = 2^{k-1}$ (and the maximum receiver window size has the same value)

Selective Repeat ARQ – example



ARQ performance – without errors

- An important parameter that has a strong impact on the performance of ARQ protocols (as well as in LAN access protocols) is the ratio *a* between the propagation delay (τ) on the link and the time required to transmit a frame T_f

$$a = \tau / T_f$$

- The maximum link utilization or efficiency (S) is easily derived for error-free operation and in ideal acknowledging conditions

Stop and Wait

$$S_{max} = 1 / (1 + 2a)$$

Sliding Window

$$S_{max} = 1 \qquad \text{if } W \geq 1 + 2a$$

$$S_{max} = W / (1 + 2a) \qquad \text{if } W < 1 + 2a$$

Values of *a* for different communication scenarios

- Values of *a* for P = 2500 bits and various distances and channel rates

P = 2500 bits		100 kbit/s	1 Mbit/s	10 Mbit/s	100 Mbit/s	**R**
		25 ms	2.5 ms	0.25 ms	0.025 ms	**T_f**
1 km	0.005 ms	0.0002	0.002	0.02	0.2	
10 km	0.05 ms	0.002	0.02	0.2	2	
100 km	0.5 ms	0.02	0.2	2	20	
1000 km	5 ms	0.2	2	20	200	
10000 km	50 ms	2	20	200	2000	
satellite	250 ms	10	100	1000	10000	
d	–					

- For P = 500 bits (small frame), the values in the table must be multiplied by 5
- For P = 10000 bits (large frame), the values in the table must be divided by 4

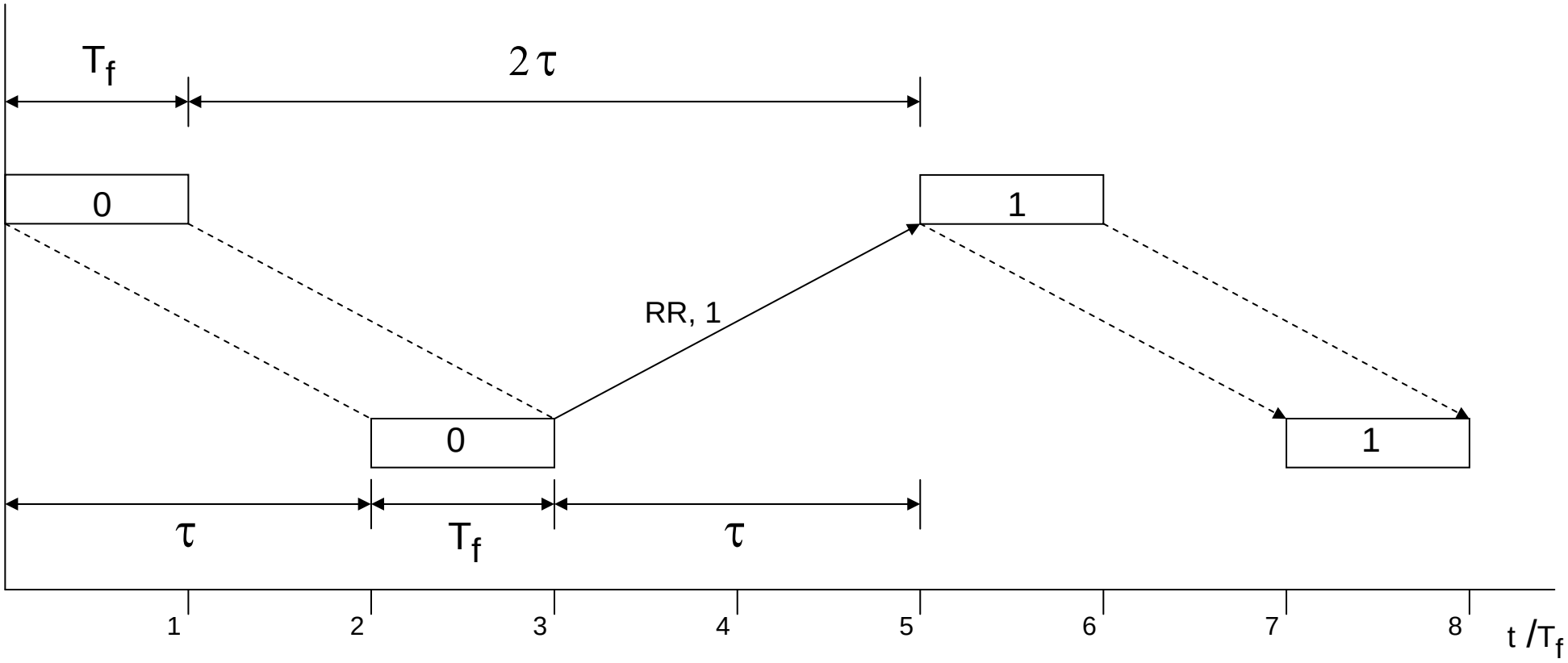
Performance of Stop and Wait

$$S = \frac{T_f}{T_f + 2\tau} = \frac{1}{1 + 2a}$$

$a = 2$

$1 + 2a = 5$

$S = 0.2$



Performance of Sliding Window

$$S = \frac{W * T_f}{T_f + 2\tau} = \frac{W}{1 + 2a} \qquad (W < 1 + 2a)$$

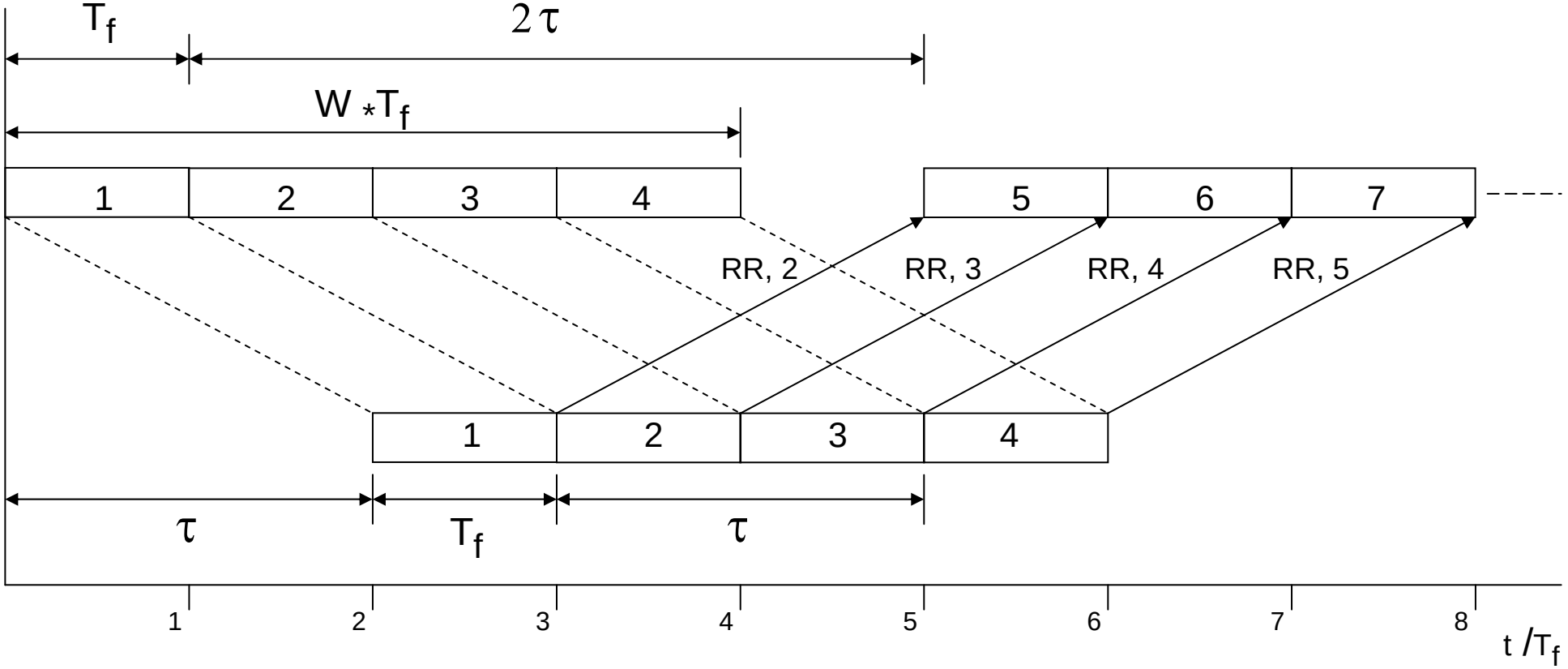
$$S = 1 \qquad (W \geq 1 + 2a)$$

$a = 2$

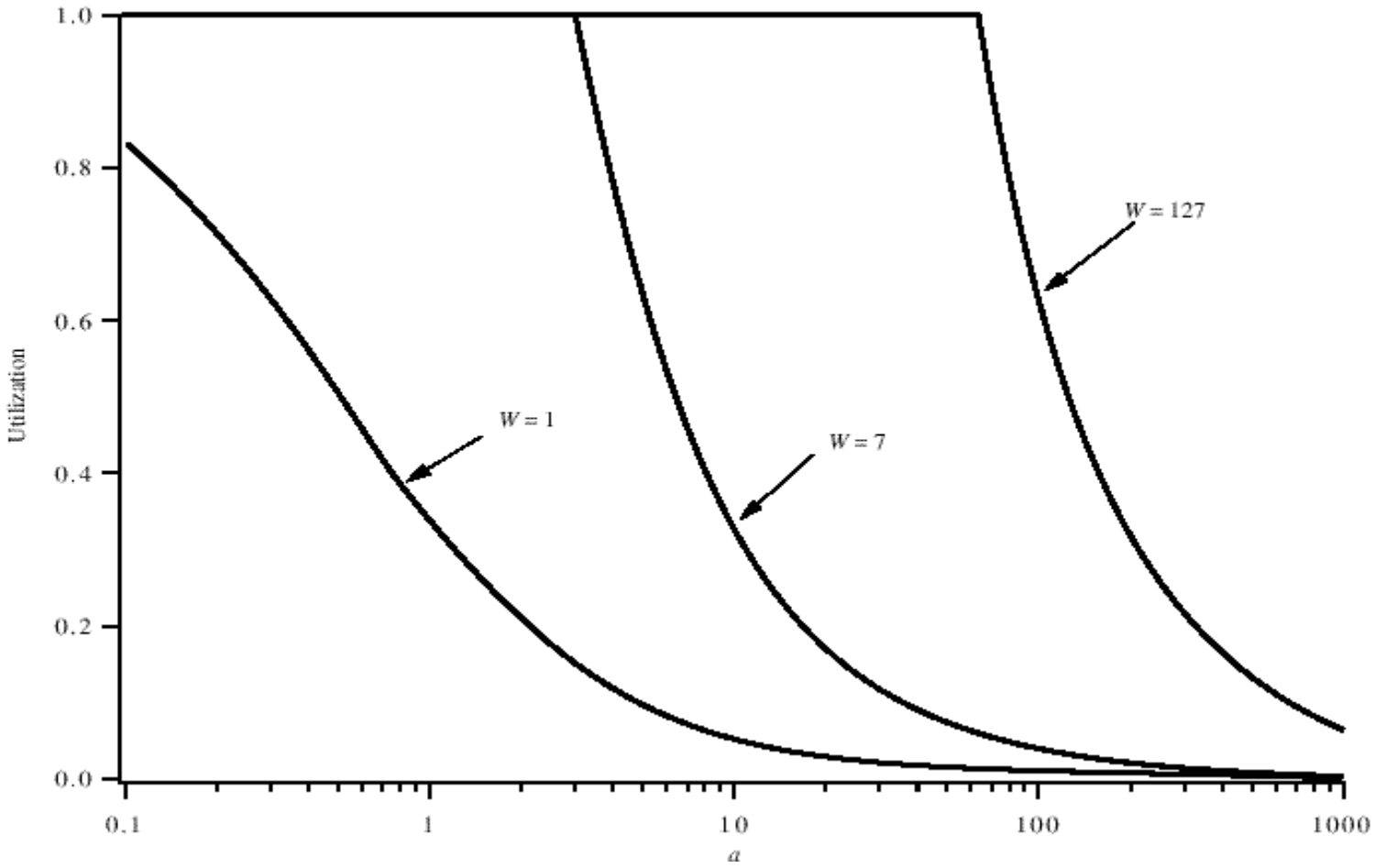
$1 + 2a = 5$

$W = 4,\ S = 0.8$

$W \geq 5,\ S = 1$



Sliding Window efficiency



ARQ performance – with errors

- The utilization of a link is reduced when I-frames have to be retransmitted due to errors
- Considering that P_e is the probability of a frame being corrupted, the maximum link utilization is now given by

Go-Back-N

$$S = \begin{cases} \dfrac{1-P_e}{1+2aP_e} & W \geq 1+2a \\ \dfrac{W(1-P_e)}{(1+2a)(1-P_e+WP_e)} & W < 1+2a \end{cases}$$

Selective Repeat

$$S = \begin{cases} 1-P_e & W \geq 1+2a \\ \dfrac{W(1-P_e)}{1+2a} & W < 1+2a \end{cases}$$

Sliding window and flow control

- In the context of data link protocols, the sliding window mechanism is an important component of the ARQ strategy
  - This includes Stop and Wait, which can be described as a sliding window mechanism with $W_{max} = 1$
- However, a sliding window used by ARQ protocols may limit the rate at which the sender is allowed to send frames
  - This is always the case of Stop and Wait (since $W_{max} < 1 + 2a$, for any value of *a*)
  - In Go-back-N and Selective Repeat this occurs in case $W < 1 + 2a$ and the number of unacknowledged frame has reached the window limit
  - In both cases there is an implicit flow control, even if the sliding window mechanism is not used with this purpose
- A sliding window mechanism may also be intentionally used for flow control purposes – for example, in TCP the window announced by the receiver is seen as a (variable) credit given to the sender
- In data link protocols, explicit flow control may be imposed by the receiver, by means of a *Receiver Not Ready* (RNR) S-frame, which temporarily stops frame transmission from the sender (*Stop and Go*)

Error control at the Transport Layer (TCP)

- The Network layer may loose and disorder packets, and thus error control at the Transport layer may be required to provide a reliable service to applications – TCP provides such a service
- A TCP receiver accepts out of order packets and reorders them, but gaps may form due to lost packets
- TCP only supports positive acknowledgements (for packet sequences without gaps) – the receiver does not ask for retransmissions
- The sender retransmits unacknowledged packets after a time-out
  - This allows filling the gaps at the receiver, thus enabling further acknowledgements of previously accepted out of order packets
  - Retransmissions may originate duplicates that must be detected and eliminated by the receiver
- The receiver window (explicit credits offered to the sender) is measured in bytes (unlike windows at the Data Link layer, which are measured in frames)
- Since the sender assumes that missing acknowledgements are due to packet losses (and thus, possibly, to network congestion), the sender window is dynamically updated as a measure to control congestion