BEng Course B38CN2: Introduction to Communications and Networks

Chapter 3. The Data Link Layer

Sheng Tong

Xidian University
School of Telecommunications Engineering
Room: I-304, Main Building, North Campus
E-mail: ts_xd@163.com

Acknowledgement: these slides are adapted from those from Prof. Cheng-Xiang Wang.

Contents (1/2)

Contents (2/2)

3. The Data Link Layer

- This chapter deals with the algorithms for achieving reliable, efficient communication between adjacent machines at the data link layer.
- Factors that limit the reliability & efficiency of the data transfer:
  - Communication circuits make errors occasionally.
  - Communication circuits have only a finite data rate.
  - There is a nonzero propagation delay between the time a bit is sent and the time it is received.

3.1 Data Link Layer Design Issues

- Providing services to the network layer by using the services provided by the physical layer.
- Error control: dealing with transmission errors.
- Flow control: regulating the data flow so that slow receivers are not swamped by fast senders.
- Frame management: The data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission.
  - The heart of what the data link layer does (providing services, errors control, flow control).
  - Frame: a frame header, a payload field for holding the packet, a frame trailer.
- Multiple access→MAC sublayer (Chapter 4).

Packets and Frames

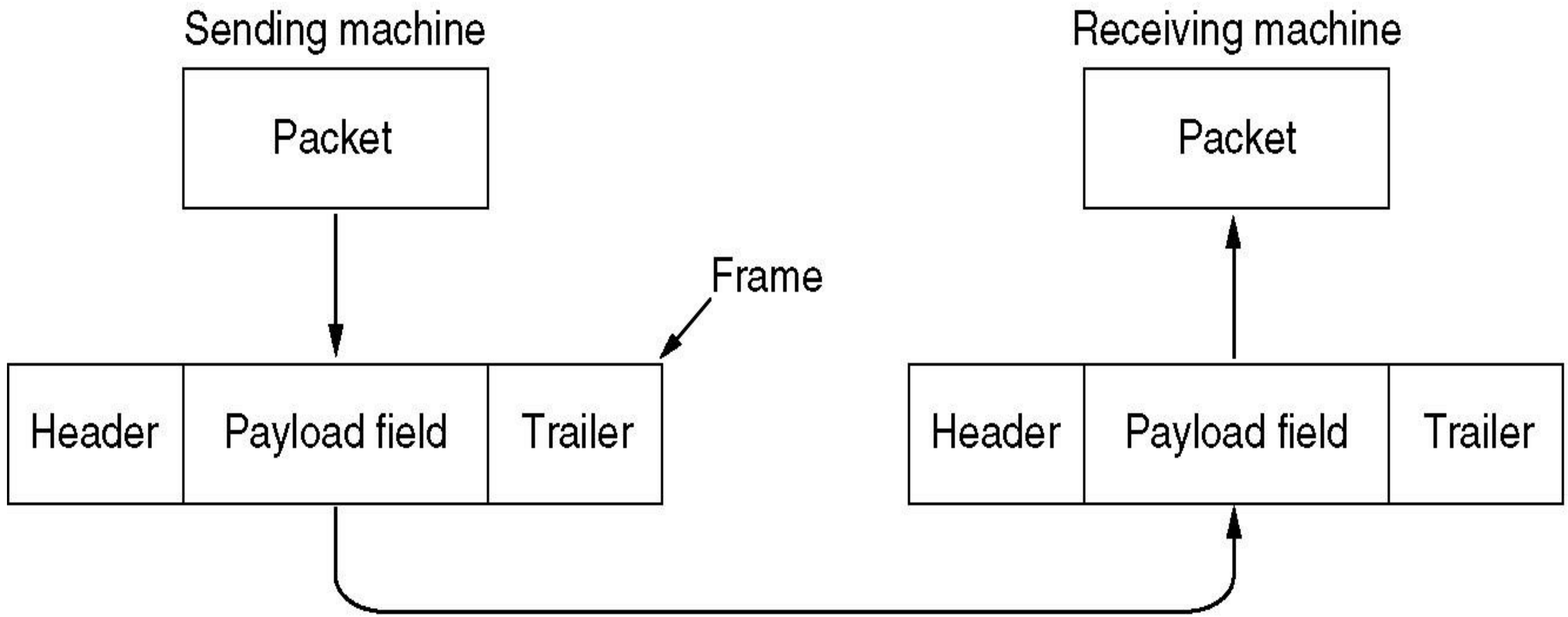


Fig. 3.1: Relationship between packets and frames.

3.1.1 Services Provided to the Network Layer

- Principal service: transfer data from the network layer on the source machine to the network layer on the destination machine.

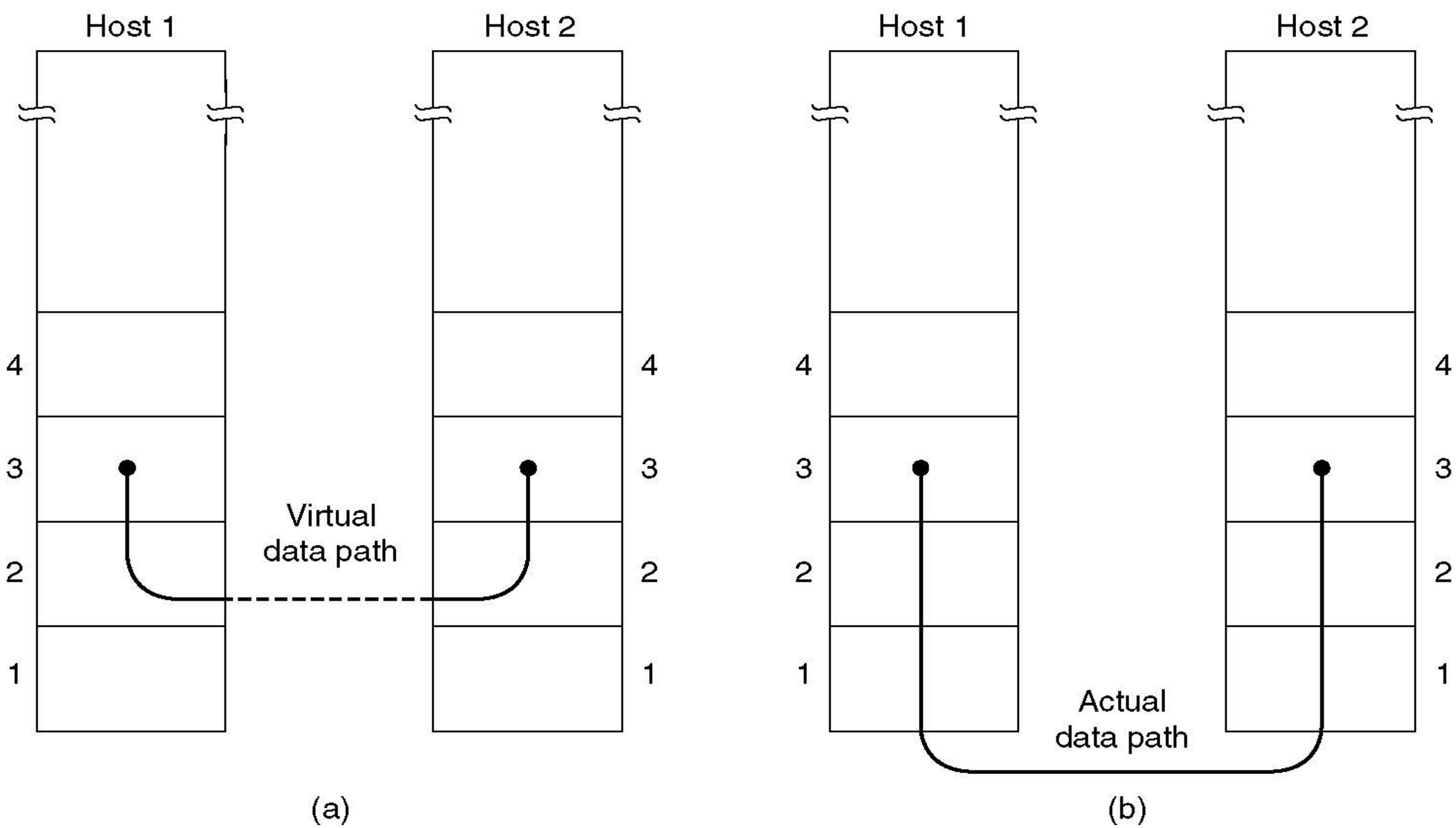


Fig.3.2: (a) Virtual communication; (b) Actual communication.

Various Services

- Unacknowledged (Unreliable) connectionless service
  - No acknowledgement.
  - No logical connection is estiablished beforehand or released afterwards.
  - No attempt to detect the loss of frames or recover in the data link layer.
  - Appropriate when the error rate is very low so that recovery is left to higher layers or for real-time traffic, such as voice. →Most **LAN**s.
- Acknowledged (Reliable) connectionless service
  - Each frame sent is individually acknowledged.
  - No logical connections.
  - If a frame has not arrived within a specified time interval, it can be sent again.
  - Useful over unreliable channels, such as **wireless sytems**.
- Acknowledged (Reliable) connection-oriented service

Acknowledged Connection-Oriented Service

- Three distinct phases:
  - Establish a connection by initializing variables and counters needed to track frames.
  - Transmit frames using the connection.
  - Release the connection by freeing up the variables, buffers, and other resources.
- Provides the network layer processes with the equivalent of a reliable bit stream.
  - Each frame is numbered, received exactly once, and received in the right order.
  - Connectionless service: a packet can be sent and received several times if the acknowledgement is lost.
- Typical example: a **WAN** subnet consisting of routers connected by point-to-point leased telephone lines.

An Example: A WAN Subnet

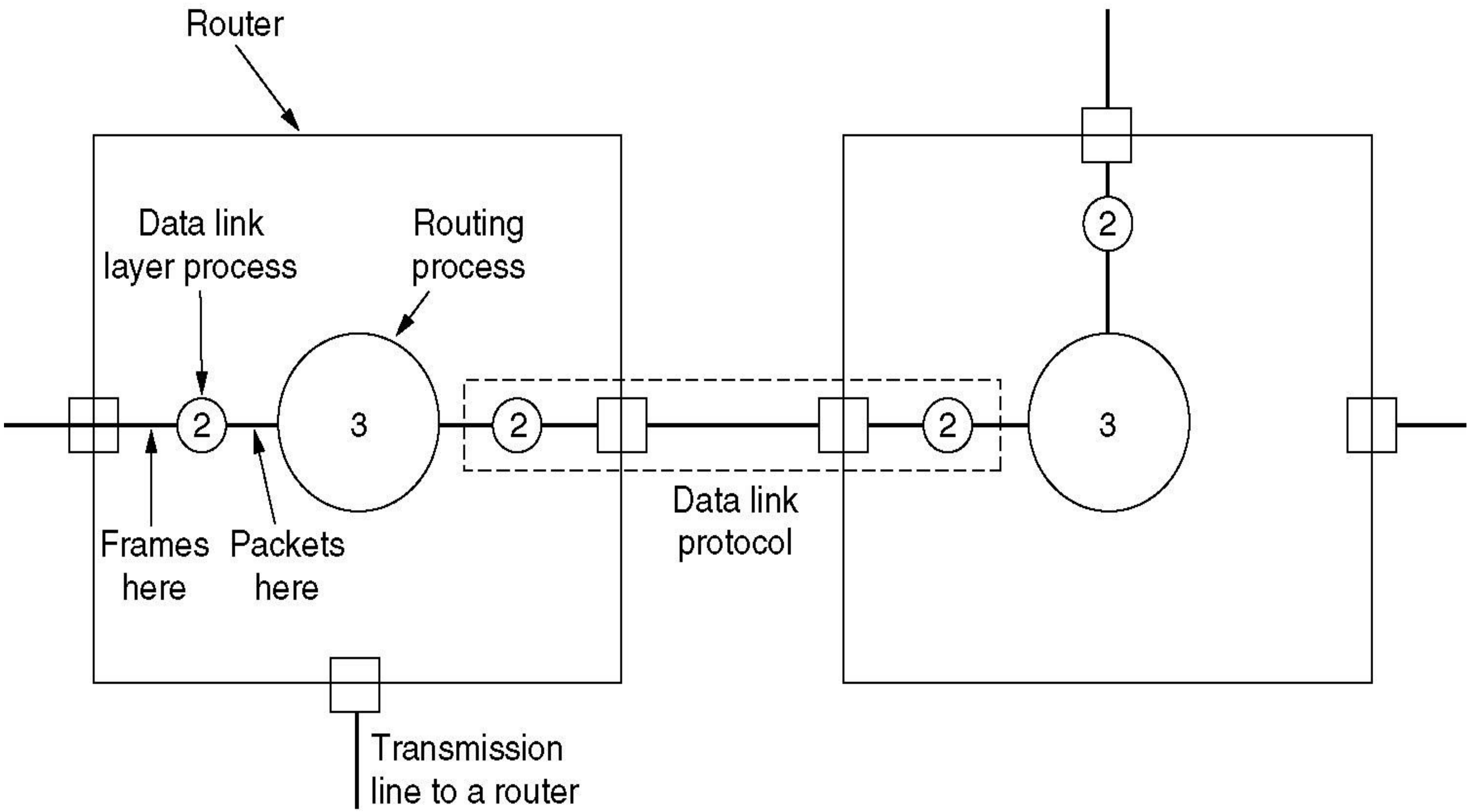


Fig. 3.3: Placement of the data link protocol.

3.1.2 Framing

- How to break up the bit stream from the physical layer into discrete frames (mark the start and the end of each frame)?

⇨ Recompute the checksum for each frame at the destination.

⇨ Four methods/algorithms:

  – Character count.
  – Flag bytes with byte stuffing.
  – Starting and ending flags, with bit stuffing.
  – Physical layer coding violations.

Character Count

- Uses a field (character count) in the header to specify the number of characters in the frame.
- The character count tells how many characters follow and where the end of the frame is.
- Problem: If the character count is garbled by a transmission error, the destination will get out of the **synchronization** and will be unable to locate the start of the next frame.
  - ⇨ No solution: Incorrect checksum or retransmission does not help.
  - ⇨ The character count method is rarely used anymore.

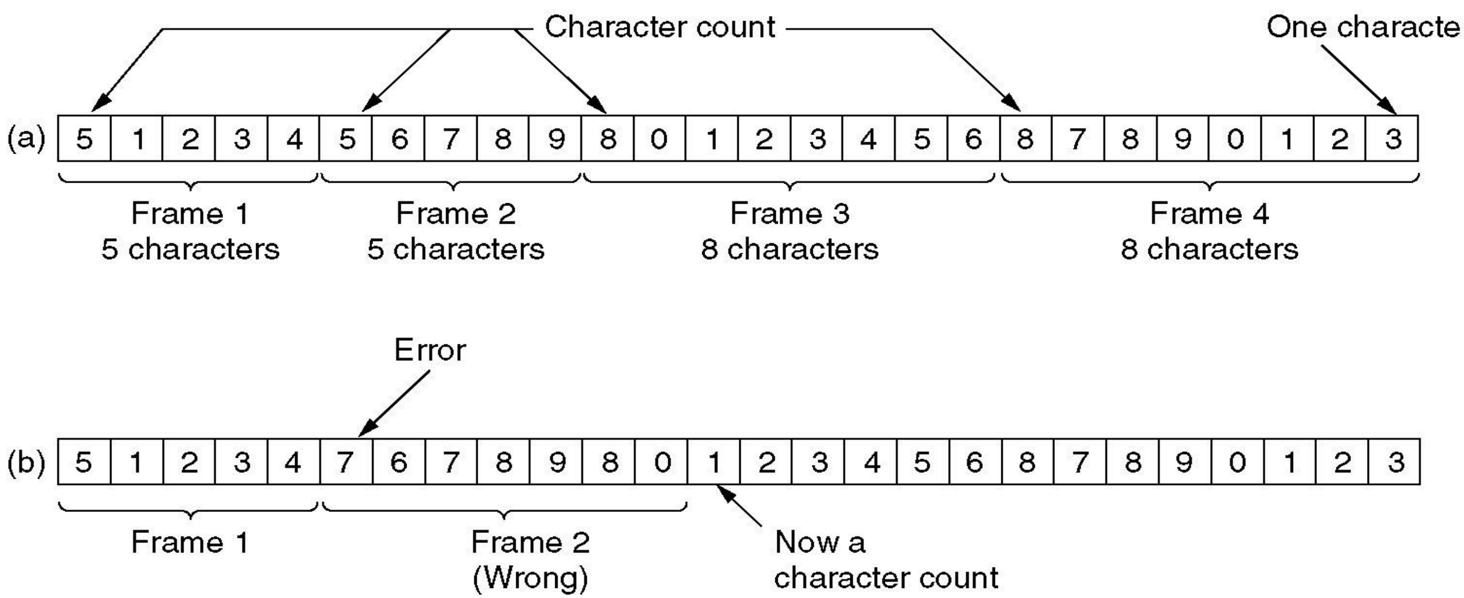


Fig. 3.4: A character stream: (a) Without errors; (b) With one error.

Flag Bytes with Byte Stuffing (1/2)

- Gets around the problem of resynchronization after an error by having each frame start and end with special bytes (flag bytes).
  - Two consecutive flag bytes indicate the end of one frame and start of the next one.

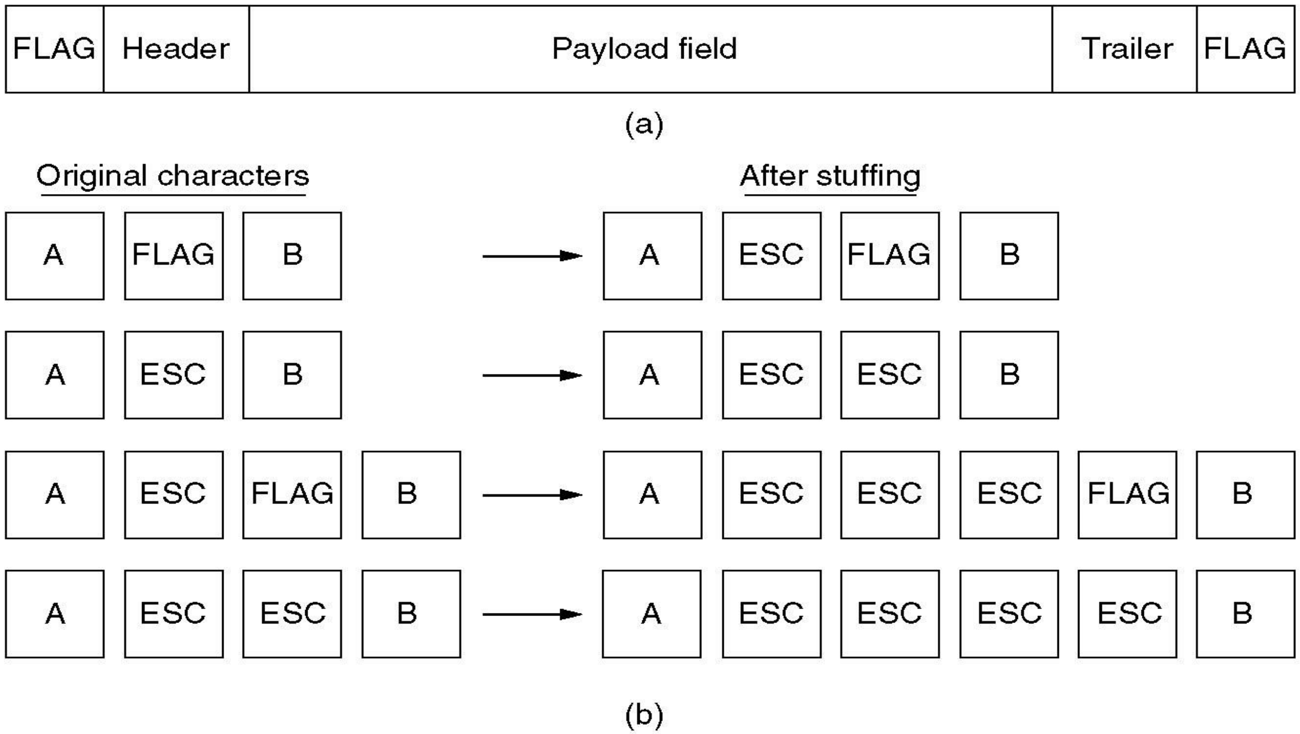


Fig. 3.5: (a) A frame delimited by flag bytes; (b) Four examples of byte sequences before and after stuffing.

Flag Bytes with Byte Stuffing (2/2)

- Problems with transmitting binary data: The flag byte's bit pattern may occur in the data.

⇨ Solution: **Byte stuffing** or character stuffing.
  - Inserts a special escape byte (ESC) before each "accidental" flag byte in the sender's data.
  - Removes the escape byte on the receiving end before the data are given to the network layer.
  - ⇨ A framing flag byte: no escape byte; a flag byte in the data: with an escape byte.

- Another problem: what happens if an escape byte occurs in the middle of the data?

⇨ Solution: Stuffing with an additional escape byte.
  - Any single escape byte is part of an escape sequence.
  - A double escape bytes indicates that a single escape occurred naturally in the data.
  - ⇨ Receiver: $2k$ or $2k+1$ ($k=0, 1, \ldots$) escape bytes ⇨Transmitter: k escape bytes in the data.

- Major disadvantage: Closely tied to the use of 8-bit characters, instead of arbitrary sized characters.

Bit Stuffing

- Allows data frames and character codes to contain an arbitrary number of bits.
- Each frame begins and ends with a special bit pattern, 01111110 (a flag byte).
- **Bit stuffing** in the sender's data: Automatically stuffs a 0 bit after five consecutive 1s.
- When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs/deletes the 0 bit.
  - ⇨The boundary between two frames is unambiguous.

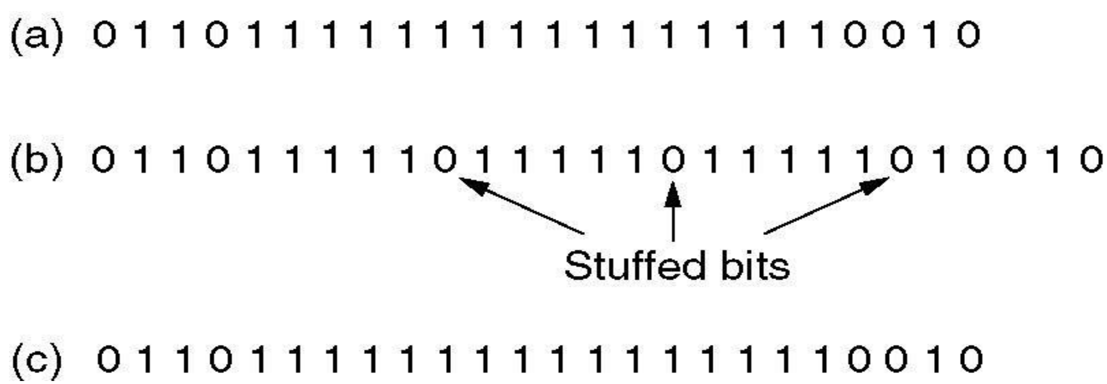


Fig. 3.6: Bit stuffing: (a) The original data; (b) The data as they appear on the line; (c) The data as they are stored in receiver's memory after destuffing.

Physical Layer Coding Violations

- Only applicable to networks in which the encoding on the physical medium contains redundancy.
- Some codes, which are not used for physical layer data codes (i.e., redundancy/violation), can be used for delimiting frames in some protocols.
- **Note**: Many data link layer protocols use a combination of a character count with one of the other methods for extra safety.

3.1.3 Error Control

- How to make sure all frames are delivered to the destination reliably and in the proper order?
- The receiver sends back special control frames bearing positive or negative acknowledgements about the incoming frames.
  - Problem: The sender will hang forever if a frame is ever lost due to, e.g., malfunctioning hardware.
- Timers: Start after transmitting a frame; Canceled if the acknowledgement gets back before the timer runs out; Go off if either the frame or acknowledgement is lost.
  - Problem: The receiver may accept the same frame several times.
- Sequence numbers: By assigning sequence numbers to outgoing frames, the receiver can distinguish retransmissions from originals.

3.1.4 Flow Control

- What to do with a sender that systematically wants to transmit frames faster than the receiver can accept them?
- Two approaches
  - Feedback-based flow control: The protocol contains well-defined rules about when a sender may transmit the next frame, i.e., frames can only be sent until the receiver gives the sender permission.
  - Rate-based flow control: The protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver. Never used in the data link layer!

3.2 Error Detection and Correction

- Transmission errors (random or burst) are unavoidable in physical layer communication media.
  - Typical BERs: systems using copper wires (10^{-6}) , optical fiber systems (10^{-9} or less), wireless systems (10^{-3} or worse).

- Two basic approaches of error control:
  - Automatic Retransmission reQuest (**ARQ**): The receiver detects only the occurrence (not positions) of errors and requests retransmissions.
    - Suitable for highly reliable channels, e.g., systems using optical fiber or copper wire.
  - Forward error correction (**FEC**): The receiver detects the occurrence of errors and tries to correct errors.
    - Appropriate when a return channel is not available or large delay is not tolerated or for channels with many errors, e.g., satellite and wireless systems.

Hamming Distance

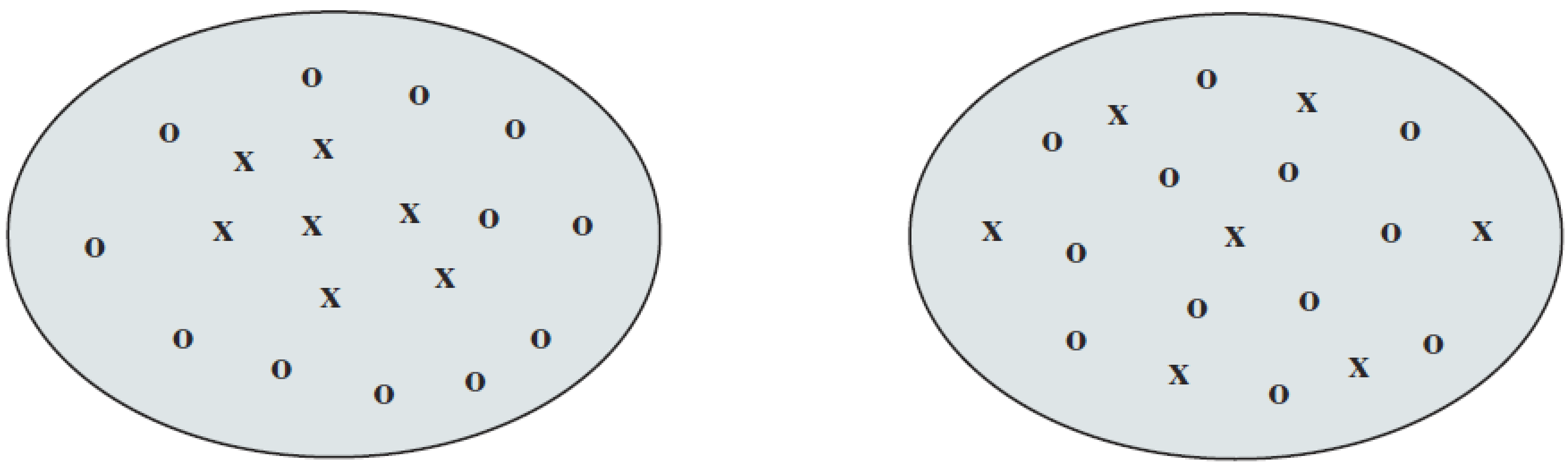
- *n*-bit codeword: *m* data bits and *r* redundant or check bits; $n=m+r$.
- Hamming distance of two codewords: the number of bit positions in which two codewords differ.
- Hamming distance of the complete code: the **minimum** Hamming distance of a complete list of all the legal codewords.
- To **detect** all *d* errors and less, we need a code with the Hamming distance of at least $d+1$.
- To **correct** all *d* errors and less, we need a code with the Hamming distance of at least $2d+1$.

Distance Properties of Codes

- Error detection/correction requires redundancy.
- Every error detection technique will fail to detect some errors; every error correction technique will fail to correct some errors.

(a) A code with poor distance properties

(b) A code with good distance properties



x codewords o non-codewords

3.2.1 Error Detection Codes

- When the error rate is very low, **error detection** and retransmission is usually more efficient than error correction.
- **Single parity check code**: Append a single parity bit to a bit string so that the total number of 1 bits in the codeword is even (or odd).

  ⇒A code with a single parity bit has a distance of at least 2.

  ⇒It can be used to detect single errors.
- Example:
  - Even parity: 1011010→1011010**0**
  - Odd parity: 1011010→1011010**1**

Polynomial Code

- Widespread use in practice, also known as Cyclic Redundancy Check (CRC).
- Treat bit strings as polynomials with coefficients of 0 and 1 only.
- A k-bit frame is regarded as the coefficient list for a polynomial with k terms, ranging from x^{k-1} (the high-order/leftmost bit) to x^0. Such a polynomial is said to be of degree k-1.
- Example:
  - A 6-bit frame 110001$\rightarrow 1\bullet x^5+1\bullet x^4+0\bullet x^3+0\bullet x^2+0\bullet x^1+1\bullet x^0=x^5+x^4+x^0$.
- **Polynomial arithmetic**: modulo 2 arithmetic; Both addition and subtraction are identical to exclusive OR.

Linear Codes and Cyclic Codes

- *Linear code:* an (n, k) binary linear code, C, is k-dimensional vector ***subspace*** of $(F_2)^n$.
- We say C is a linear code with length n and dimension k. An element of C is called a code word of C.
- *Cyclic code:* an (n, k) binary linear code C is called cyclic code if every cyclic shift a code word is also a code word in C.

$$(c_{n-1}, c_{n-2}, \cdots, c_1, c_0) \quad \overset{\substack{\text{cyclically left shift} \\ \text{by one position}}}{\Longrightarrow} \quad (c_{n-2}, \cdots, c_1, c_0, c_{n-1})$$

Cyclic Codes: (7,4) cyclic code

Input	Code word
0000	0000000
0001	0001011
0010	0010110
0011	0011101
0100	0100111
0101	0101100
0110	0110001
0111	0111010

Input	Code word
1000	1000101
1001	1001110
1010	1010011
1011	1011000
1100	1100010
1101	1101001
1110	1110100
1111	1111111

Cyclic Codes

$$(c_{n-1}, c_{n-2}, \cdots, c_1, c_0)$$

$$c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1 x + c_0$$

$$xc(x) = c_{n-1}x^{n} + c_{n-2}x^{n-1} + \cdots + c_1 x^2 + c_0 x$$

$$xc(x) \bmod x^n - 1 = c_{n-2}x^{n-1} + \cdots + c_1 x^2 + c_0 x + c_{n-1}$$

Cyclic Codes

- Let $g(x) = 1 + g_1x + ... + g_{r-1} x^{r-1} + x^r$ be the nonzero code polynomial of minimum degree in an (n, k) cyclic code C. A binary polynomial of degree n-1 or less is code polynomial if and only if it is a multiple of g(x).
- The polynomial g(x) is called *the generator polynomial* of the code.
- The generator polynomial g(x) is a factor of $x^n + 1$.
- Exercise: check $x^3+x+1 \mid x^7+1$

Polynomial Coding

- **Generator polynomial, $G(x)$, of degree r**: Essential for the polynomial coding method; Both high- and low- order bits must be 1.
- **Polynomial coding**:
  - A m-bit frame with the corresponding polynomial $M(x)$ of degree m-1, $m-1>r$.
  - Compute the checksum.
  - Append the checksum to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisible by $G(x)$.
  - Divide the checksummed frame at the receiver by $G(x)$. If there is a remainder, there has been a transmission error.

The Algorithm for Computing the Checksum

- Given: generator polynormial $G(x)$ of degree r and a m-bit frame with polynomial $M(x)$ of degree m-1, m-1>r.
- **Algorithm**:
  - Append r zero bits to the low-order end of the frame. The new frame now contains $m+r$ bits with polynormial $x^r M(x)$.
  - Divide the bit string corresponding to $G(x)$ into the bit string corresponding to $x^r M(x)$, using modulo 2 division. The remainder is always r or fewer bits.
  - Subtract (modulo 2) the remainder from the bit string corresponding to $x^r M(x)$. The result is the **checksummed frame** to be transmitted. The corresponding polynomial $T(x)$ is divisible (modulo 2) by $G(x)$.

Calculation of the Polynomial Code Checksum

Frame : 1 1 0 1 0 1 1 0 1 1

Generator: 1 0 0 1 1

Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

```
                 1 1 0 0 0 0 1 0 1 0
          ____________________________
1 0 0 1 1 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
            1 0 0 1 1
            ---------
              1 0 0 1 1
              1 0 0 1 1
              ---------
                0 0 0 0 1
                0 0 0 0 0
                ---------
                  0 0 0 1 0
                  0 0 0 0 0
                  ---------
                    0 0 1 0 1
                    0 0 0 0 0
                    ---------
                      0 1 0 1 1
                      0 0 0 0 0
                      ---------
                        1 0 1 1 0
                        1 0 0 1 1
                        ---------
                          0 1 0 1 0
                          0 0 0 0 0
                          ---------
                            1 0 1 0 0
                            1 0 0 1 1
                            ---------
                              0 1 1 1 0
                              0 0 0 0 0
                              ---------
                                1 1 1 0   <-- Remainder
```

Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Error Detection Capability of the Polynomial Code

- A single burst error: a bit string that the initial and the final bits are 1.
- A polynomial code with *r* check bits can detect all burst errors of length ≤r.
- The polynomial used in IEEE 802:
  - $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x^{1}+1.$
  - Can detect all bursts of length 32 or less.
- Polynomial coding can efficiently be implemented in hardware with a simple shift register circuit. In practice, this hardware is nearly always used, e.g., in LANs.

3.2.2 Forward Error-Correction Codes

- In FEC, the detection of transmission errors is followed by processing to determine the most likely **error locations**.
- **Classification**:
  - **Block codes**: operate on non-overlapping blocks of information.
    - **Hamming code:** can correct any single-bit error.
    - More powerful codes: BCH codes, Reed-Solomon codes etc.
  - **Convolutional codes**: operate on overlapping blocks of information.
- Correction of burst errors: FEC+Interleaving.

Hamming Code

- The bits of the codeword are numbered consecutively, starting with bit 1 at the left end, bit 2 to its immediate right, and so on.

- Every bit at position $2^p (p \geq 0)$ is used as a **parity/check bit**, which forces the parity of some collection of bits to be even (or odd). The rest are filled up with the *m* data bits.
  – Check bit at position $1=2^0$: control positions 1,3,5,7,9,... (every other bit).
  – Check bit at position $2=2^1$: control positions 2,3,6,7,10,11,... (every other block of 2 bits).
  – Check bit at position $4=2^2$: control positions 4,5,6,7,12,13,14,15,... (every other block of 4 bits).
  – Check bit at position $8=2^3$: control positions 8,9,...,15,24,25,... (every other block of 8 bits).
  – And so on.

- A bit in position *k* is checked by just those check bits occuring in its expansion as a sum of powers of 2. For example, bit 11 is checked by bits 1,2,8, i.e., 11=1+2+8.

An Example of Hamming Coding

- Parity bit position for (11, 7) Hamming code: m=7, r=4, n=m+r=11.
- Hamming coding (even parity) for the transmitted data bits: 1100001→10111001001.

	b_1	**b_2**	b_3	**b_4**	b_5	b_6	b_7	**b_8**	b_9	b_{10}	b_{11}
1	×		×		×		×		×		×
2		×	×			×	×			×	×
4				×	×	×	×				
8								×	×	×	×
	1	**0**	1	**1**	1	0	0	**1**	0	0	1

Hamming Decoding

- The receiver examines each check bit, k (k=2^p), to see if it has the correct parity.
  - If yes, the codeword is accepted as valid.
  - If no, the receiver increments a counter (initialized to be zero) by k. The value of the counter in the end gives the position of the wrong bit, which should then be inverted.

An Example of Hamming Decoding

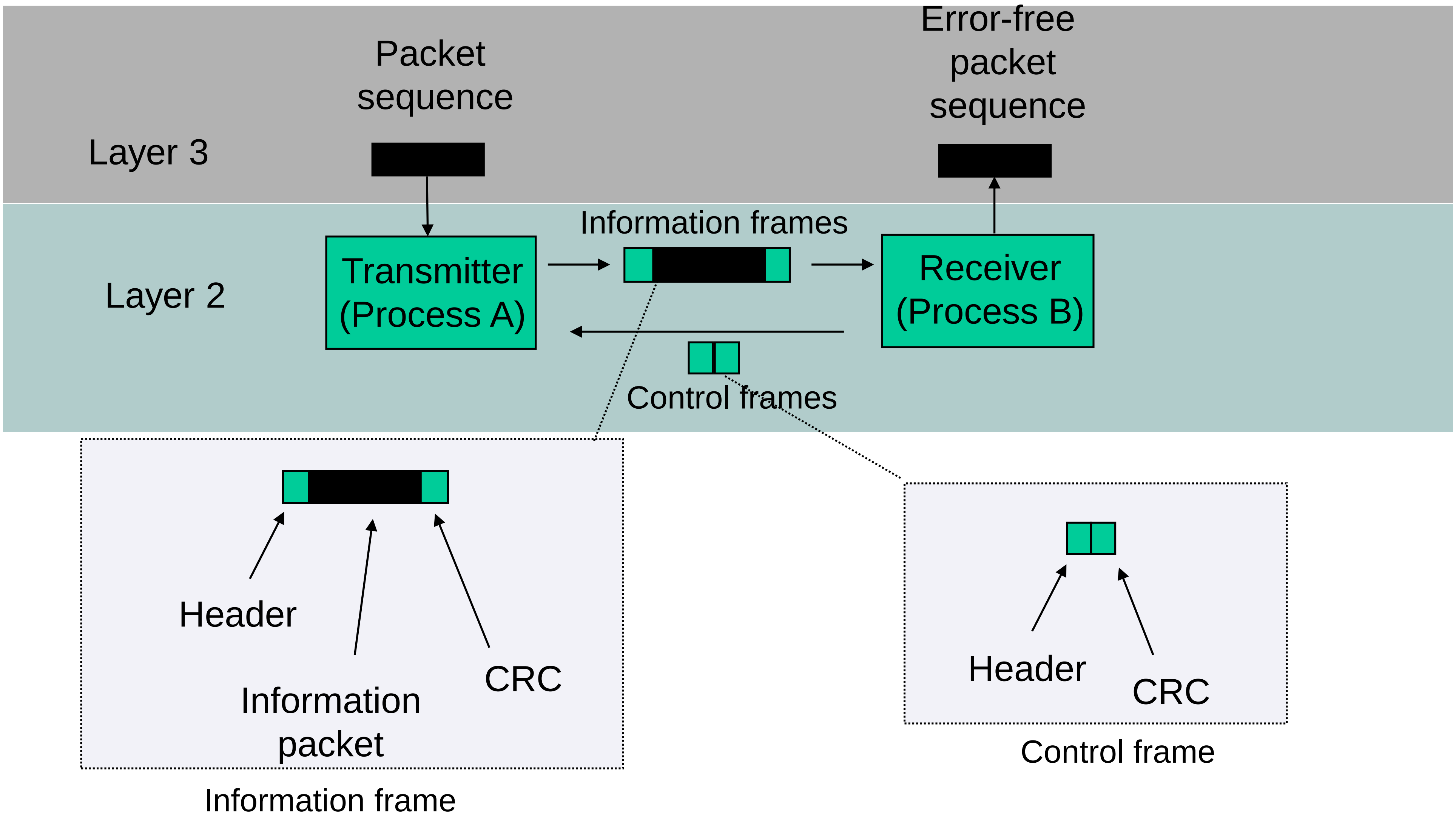
- T: Transmitted bit string.
- R: Received bit string (bit 9 is in error).
- Error Correction: check bits 1 and 8 are not satisfied ⇒ counter value=1+8=9 ⇒ bit 9 is inverted.
- F: Final bit string after correction.

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
1	×		×		×		×		×		×
2		×	×			×	×			×	×
4				×	×	×	×				
8								×	×	×	×
T	1	0	1	1	1	0	0	1	**0**	0	1
R	1	0	1	1	1	0	0	1	**1**	0	1
F	1	0	1	1	1	0	0	1	**0**	0	1

3.2.3 Automatic Repeat reQuest (ARQ)

- ARQ = error detection + retransmission.
- **Objective**: Ensure the packets are delivered error free to the destination, exactly once without duplicates and gaps, in the same order in which they are transmitted.
- **Basic elements** of ARQ:
  - Error detection code: CRC.
  - Information frames that transfer the user packets
  - Control frames: header (may include ACKs and NAKs) and CRC.
  - Time-out mechanisms
- **Three types**: stop-and-wait ARQ, go-back-n ARQ, and selective repeat ARQ.

Basic Elements of ARQ



3.2.3.1 Stop-and-Wait ARQ

- The transmitter and receiver work on the delivery of **one frame** at a time.
- After the transmitter sends an information frame to the receiver, it then **stops and waits** for an acknowledgment from the receiver.
- If no acknowledgment is received within some **time-out period**, the transmitter resends the frame, and once again **stops and waits**.

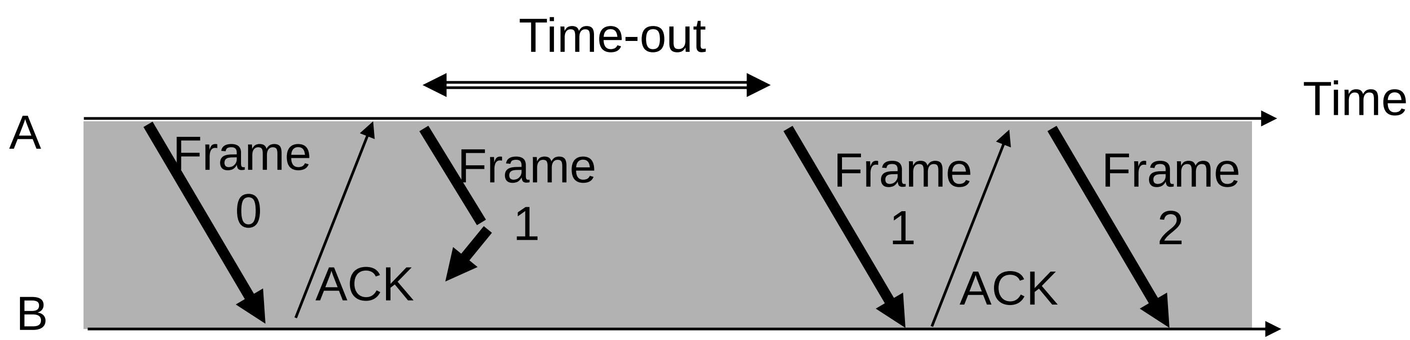


Fig. 3.7: An example of stop-and-wait ARQ: how ACK and time-outs can be used to provide recovery when Frame 1 gets lost.

The Need for Sequence Numbers (1/2)

- **Problem**: The loss of an **ACK** can result in the delivery of a duplicate packet.
- **Solution**: The ambiguity can be eliminated by including a **sequence number S_{last}** in the header of each **information frame**. S_{last} is the sequence number of the most recent transmitted frame.

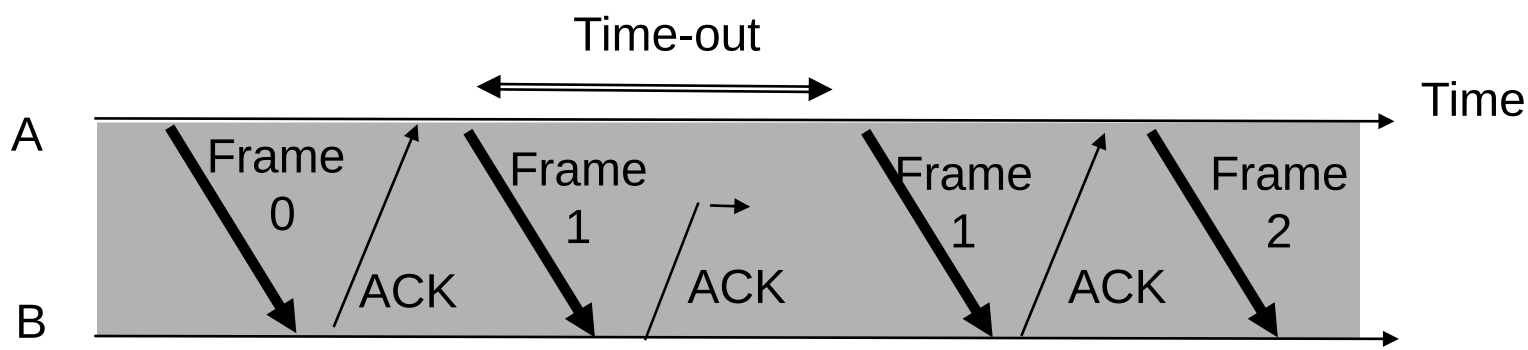


Fig. 3.8: Possible ambiguities when **frames** are **unnumbered**: Receiving process B accepts Frame 1 twice when ACK gets lost.

The Need for Sequence Numbers (2/2)

- **Problem**: Premature time-outs (or delayed ACKs) combined with loss of information frames can result in gaps in the delivered packet sequence.
- **Solution**: The ambiguity can be resolved by providing a **sequence number $\mathbf{R_{next}}$** in the header (acknowledgment) of each **control frame**. R_{next} is the sequence number of the next frame expected by the receiver. It implicitly acknowledges receipt of all prior frames.

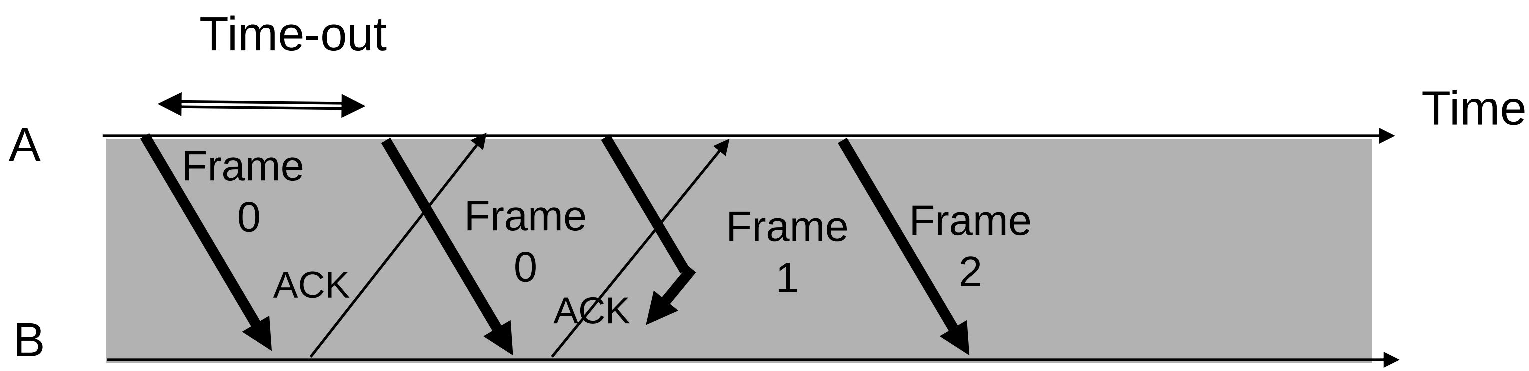


Fig. 3.9: Possible ambiguities when **ACKs** are **unnumbered**: Transmitting process A misinterprets duplicate ACKs.

One-Bit Sequence Number Suffices in Stop-and-Wait ARQ

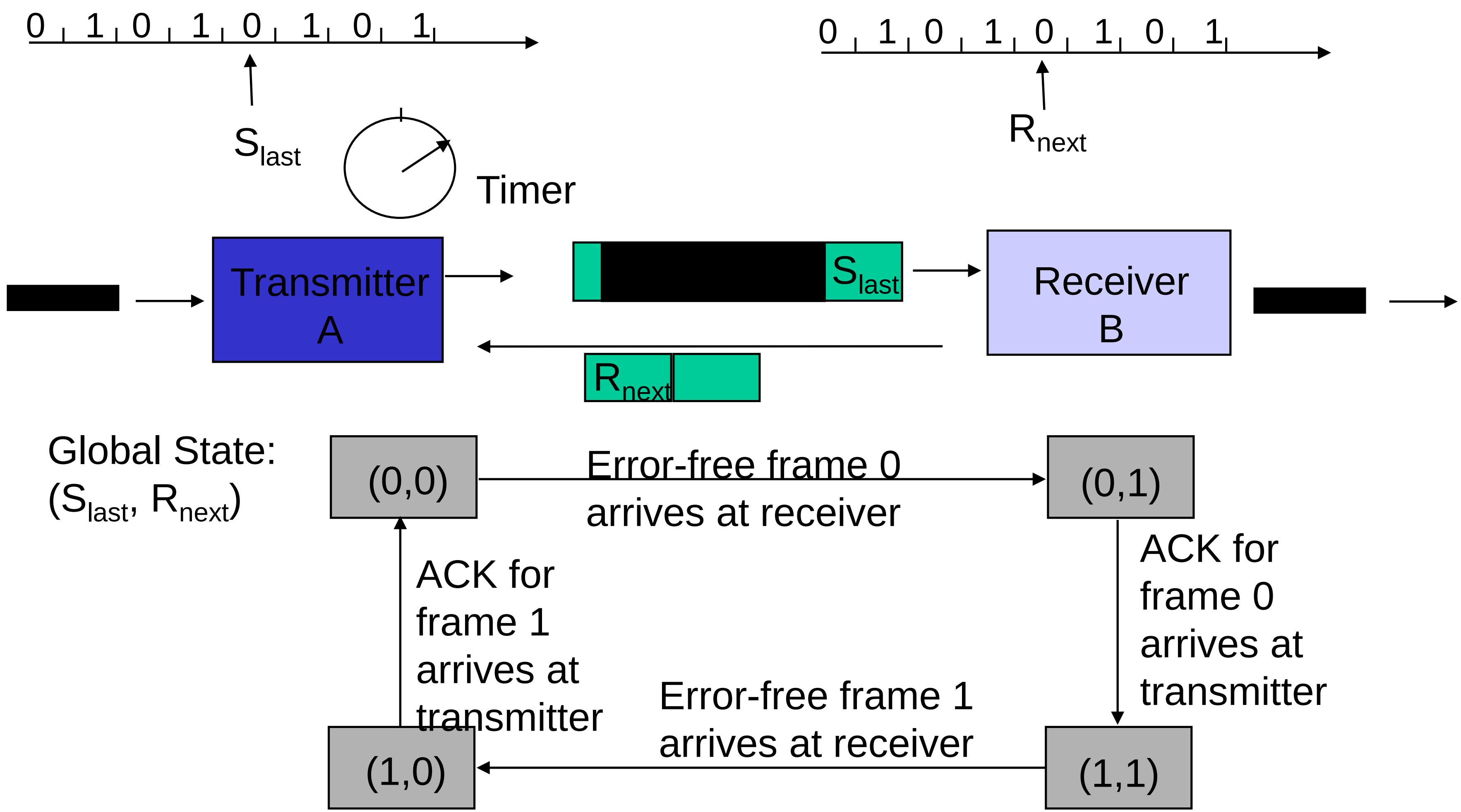


Fig. 3.10: System state information in Stop-and-Wait ARQ.

Stop-and-Wait ARQ Protocol—Transmitter

- **Ready state**:
  - Awaits a request from the higher layer for packet transfer.
  - When a request arrives, transmits a frame with updated S_{last} and CRC; starts a timer; then enters a wait state.
- **Wait state**:
  - Waits for ACK or timer to expire; blocks requests from the higher layer.
  - If the **time-out expires**, retransmits the frame; resets the timer; and stays in the wait state.
  - If **ACK is received**:
    - If the sequence number is incorrect or if errors detected: ignores ACK and stays in the wait state.
    - If the sequence number is correct, i.e., $R_{next} = (S_{last} + 1) \bmod 2$: accepts the frame, updates S_{last} to R_{next}, returnes to the ready state.

Stop-and-Wait ARQ Protocol—Receiver

- Always in **Ready State**:
  - Waits for the arrival of a new frame.
  - When a frame arrives, checks for errors.
  - If no errors are detected and the sequence number is correct ($S_{last}=R_{next}$), then accepts the frame, updates R_{next} to $(R_{next}+1) \bmod 2$, sends ACK frame with R_{next}, and delivers the packet to the higher layer.
  - If no errors are detected but the wrong sequence number, discards the frame, sends ACK frame with $R_{next.}$
  - If errors are detected, discards the frame.

Stop-and-Wait ARQ Efficiency

- Works well on channels that have low propagation delay.
- Becomes **inefficient** when the propagation delay is much **greater** than the time to transmit a frame.
- Transmission of a 10000-bit frame over a channel (1 Mbps) takes 10 ms. Waits for ACK=1 ms⇒efficiency=10/11=91%; Wait for ACK=20 ms⇒ efficiency=10/30=33%.

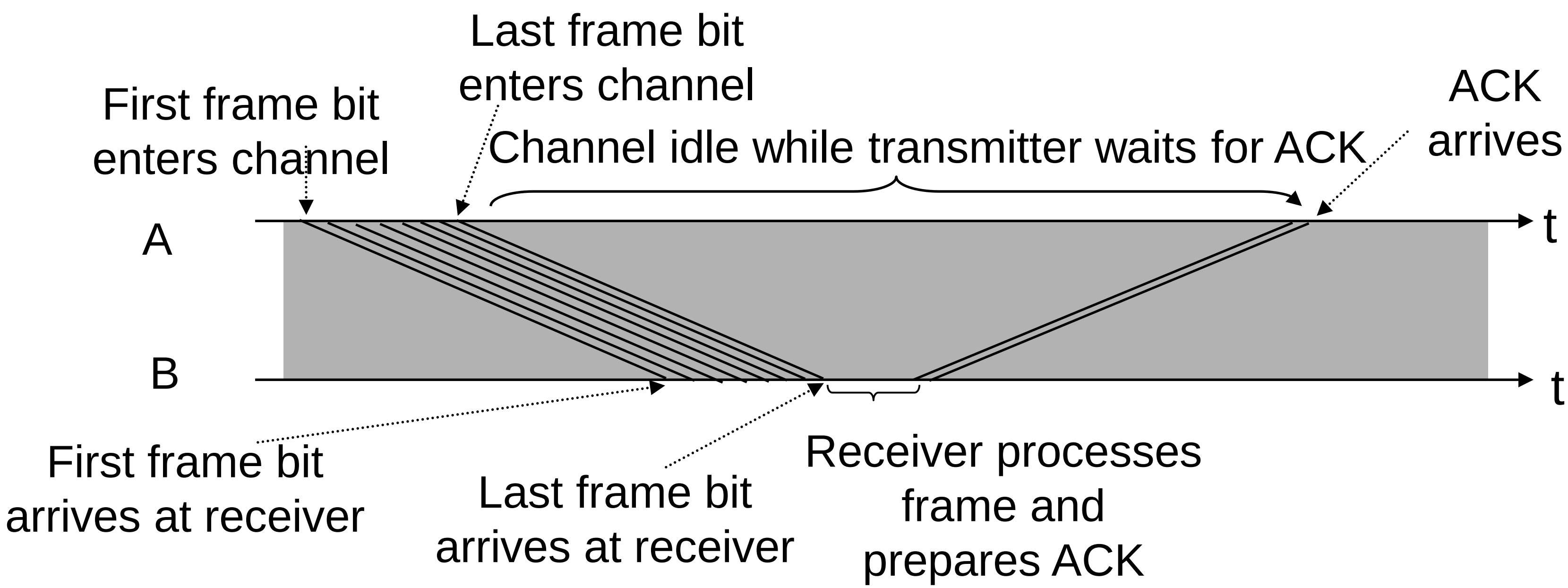


Fig. 3.11: Stop-and-wait ARQ is inefficient when the time to receive an ACK is large compared to the frame transmission time.

3.2.3.2 Go-Back-N ARQ

- **Improves Stop-and-Wait** by continuing to send frames so that the channel is kept busy while the transmitter waits for acknowledgments.
- The **transmitter** has a **limit** on the number of frames $\boldsymbol{W_s}$ that can be outstanding without acknowledgment.
- The **receiver window size**: 1 frame.
- If ACK for oldest frame arrives before window is exhausted, we can continue transmitting.
- Takes its **name** from the action that is taken when an error occurs.
- The basis for the **HDLC protocol**.

Basic Go-Back-N ARQ

- The frame with error and subsequent out-of-sequence frames are ignored.
- Transmitter is forced to go back N (N=W_s) frames when window of W_s (=4 here) is exhausted.

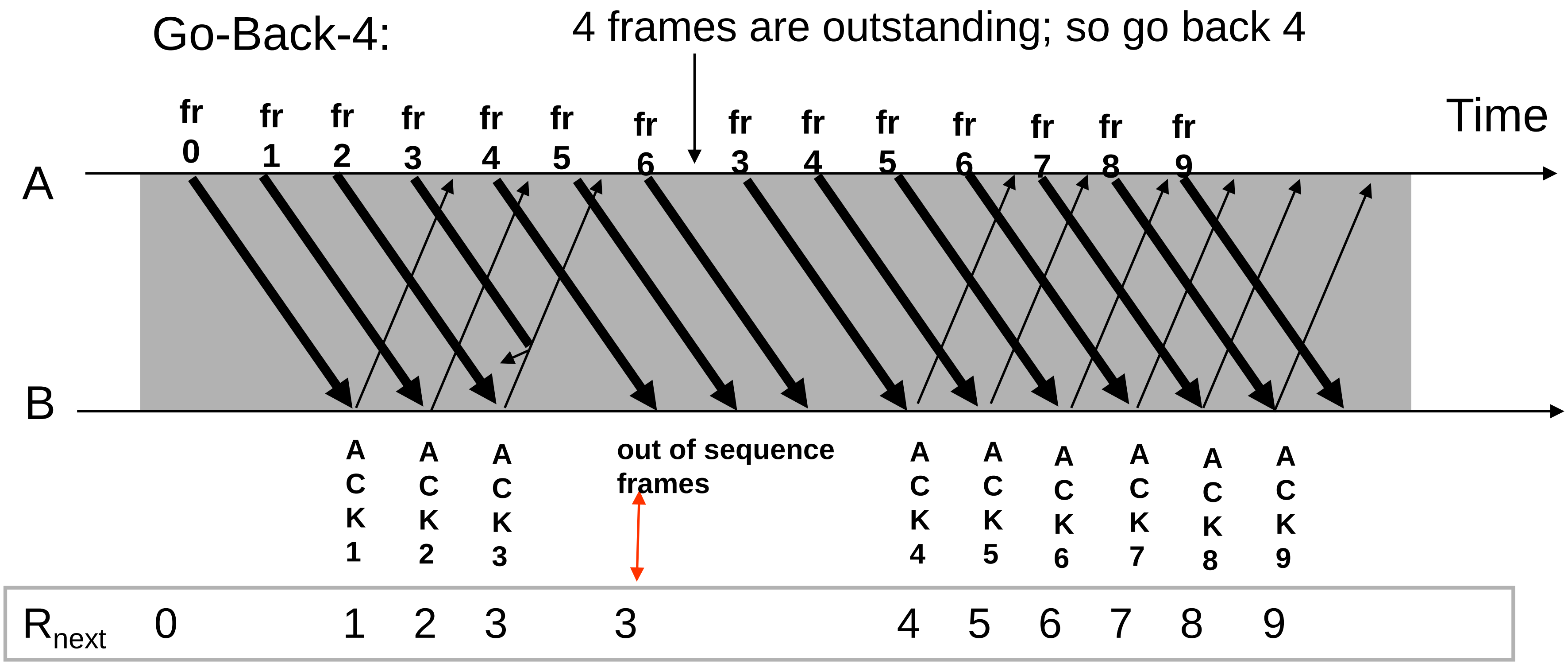
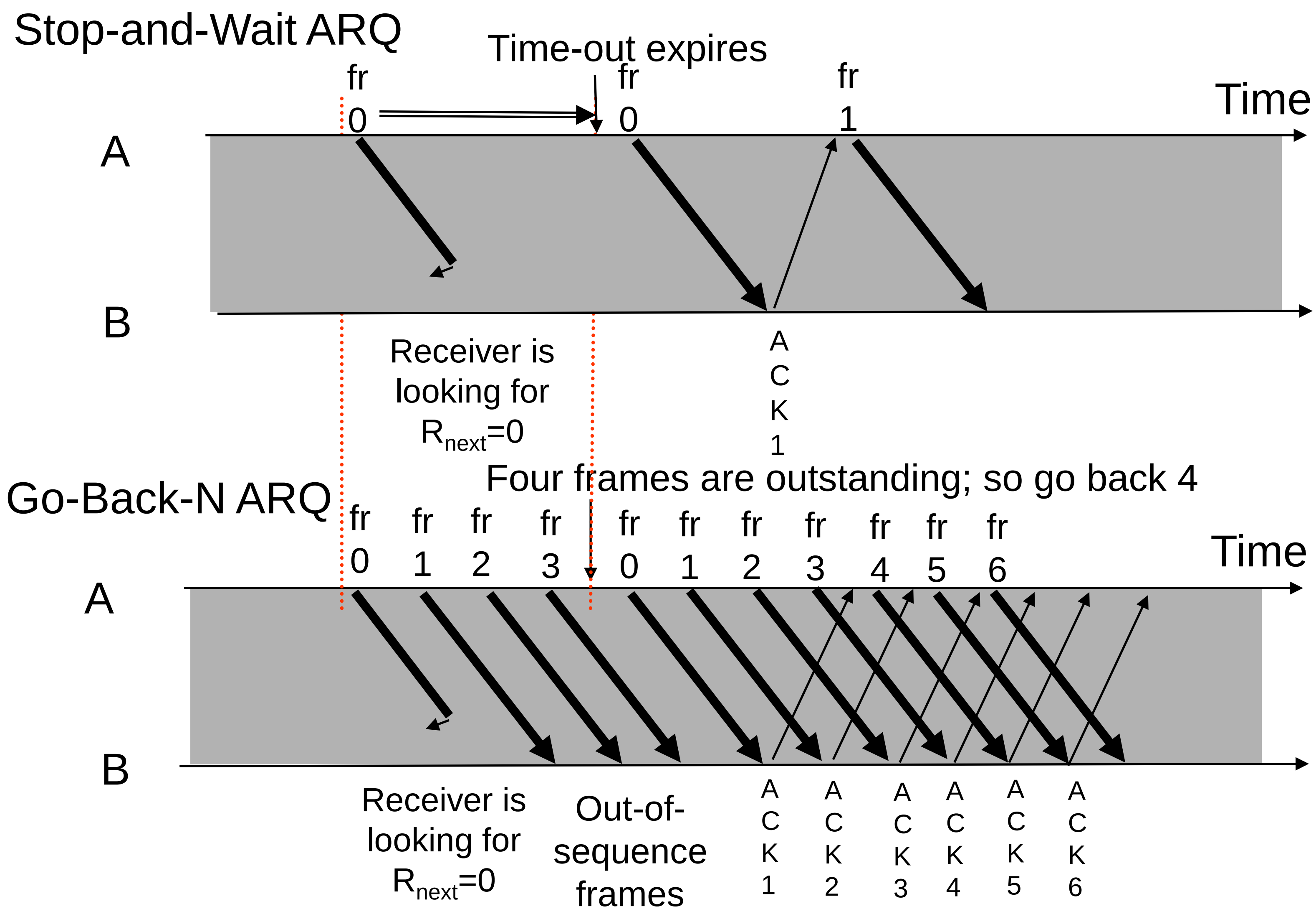


Fig. 3.12: Basic Go-Back-N ARQ.

Relationship of Stop-and-Wait ARQ and Go-Back-N ARQ

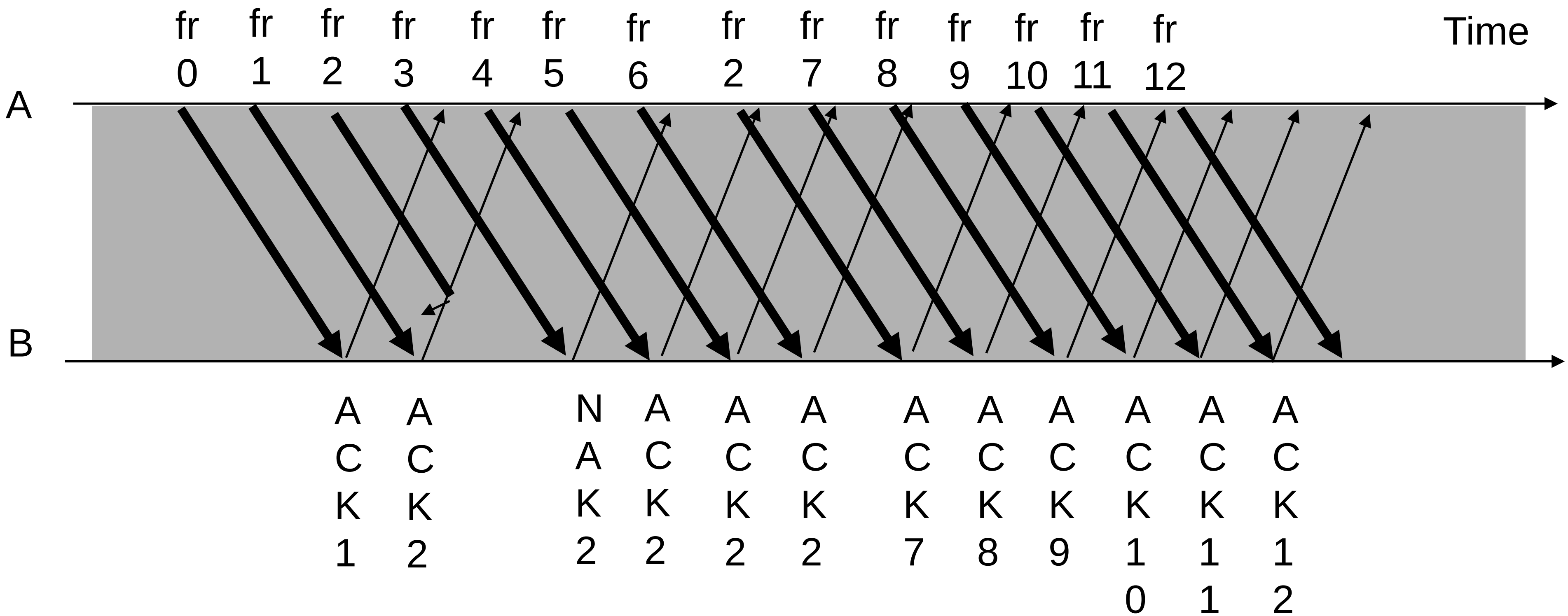


3.2.3.3 Selective Repeat ARQ

- If channels have high error rate, Go-Back-N ARQ is inefficient because *multiple* frames are resent when errors or losses occur.
- Selective Repeat ARQ is more efficient by adding two features:
  - The **receive window** is made larger than one frame so that the receiver can accept frames that are out of order but error free.
  - Only **individual** frames are retransmitted.
    - When the timer expires, only the corresponding frame is retransmitted.

Error Recovery in Selective Repeat ARQ

- Whenever an out-of-sequence frame is observed at the receiver, a NAK frame is sent with sequence number R_{next}.
- The transmitter will then only retransmit the specific frame, namely R_{next}.



3.3 Sliding Window Protocols

- How to use various elements of the **ARQ** protocols to provide other types of services, e.g., **feedback-based flow control**?
- **Sliding window protocols**:
  - **Sending window**: a window of frame sequence numbers the sender is permitted to send.
  - **Receiving window**: a window of frame sequence numbers the receiver is permitted to accept.
- By using the sliding window protocols for **flow control**, we view each acknowledgment of a frame as an issuing of a credit by the receiver that **authorizes** the transmitter to send another frame.
- The sliding window mechanism is widely used to **integrate error control and flow control** in a convenient way.

Three Bidirectional Sliding Window Protocols

- **Three categories** by the sizes of the sending window and receiving window
  - **Stop-and-wait**: Both window sizes are equal to 1.
  - **Go-back-n**: sending window size>1; receiving window size=1.
  - **Selective repeat**: Both window sizes>1.

  ⇒ Differ in terms of efficiency, complexity, and buffer requirements.

- Bidirectional (Duplex):
  - Two separate communication channels: a forward channel for data and a reverse channel for acknowledgments⇒Waste bandwidth (reverse channel).
  - The same circuit for data in both directions.
  - **Piggybacking**: The ACK is attached to the next outgoing data frame that happens to be sent from the receiver to the sender.
    - A separate ACK must be sent if no reverse traffic is available.

3.3.1 A One-Bit Sliding Window Protocol

- **Stop-and-wait** protocol: maximum window size=1.

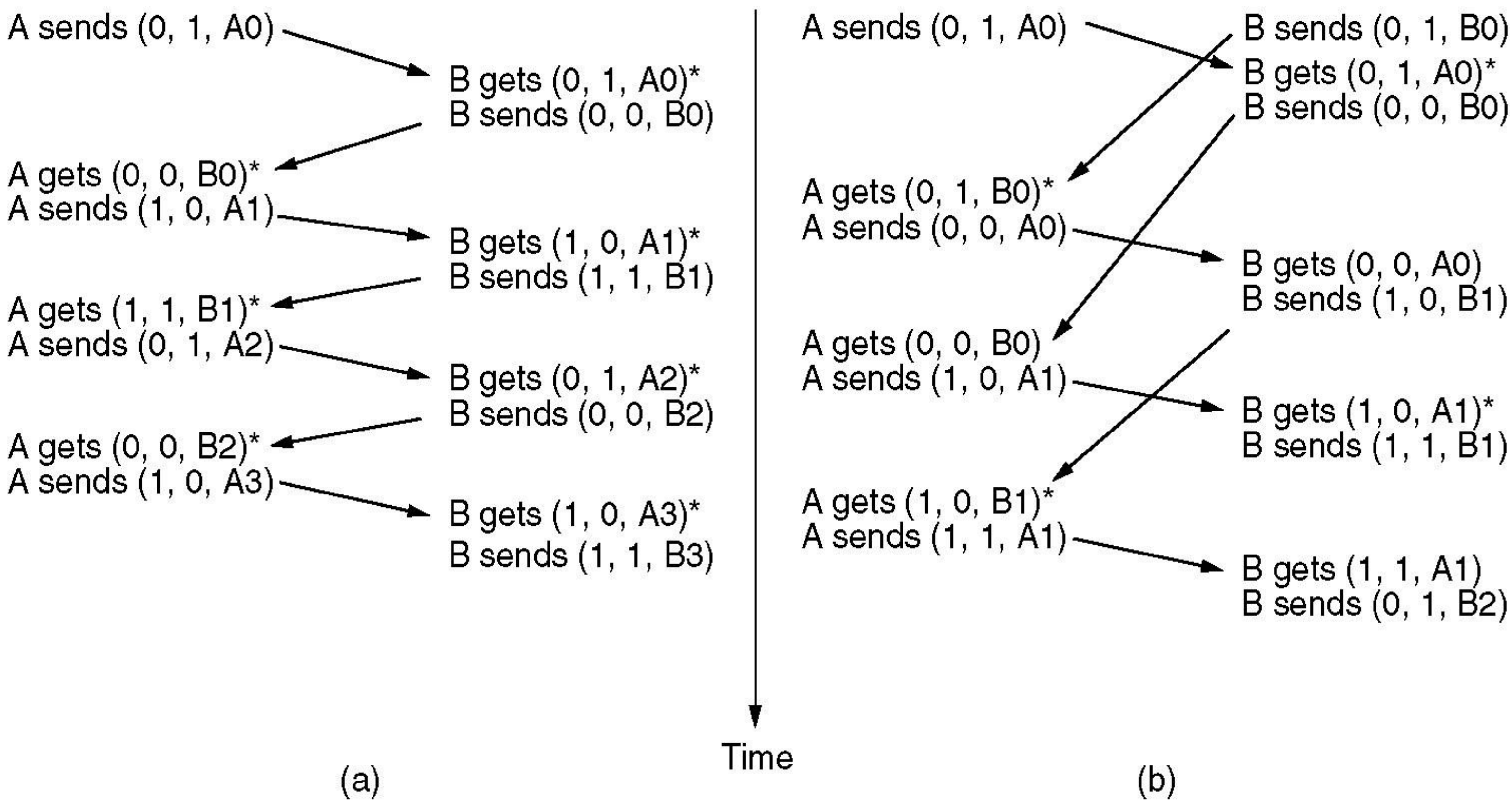


Fig. 3.13: Two scenarios: (a) Normal case; (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

3.3.2 A Protocol Using Go Back N

- Improve the efficiency of stop-and-wait by using a **large sending window size**: whenever bandwidth×round-trip delay is large.

- **Pipelining**: The processing of a new task is begun before the completion of the previous task.

- Channel capacity: b bits/sec; Frame size: l bits; Round-trip delay: R sec.
  - Stop-and-wait utilization=$(l/b)/(l/b+R)=l/(l+bR)$. If $l<bR$, utilization<50%.
  - Pipelining can keep the line busy during the round-trip delay (>0) interval.

- Two approaches dealing with errors in the presence of pipelining:
  - **Go-back-n**: receive window size=1; simply discard all subsequent frames.
  - **Selective repeat**: receive window size>1; Good frames received after the bad frame (discarded) are buffered. A NAK stimulates the retransmission of one specific frame.

Pipelining and Error Recovery

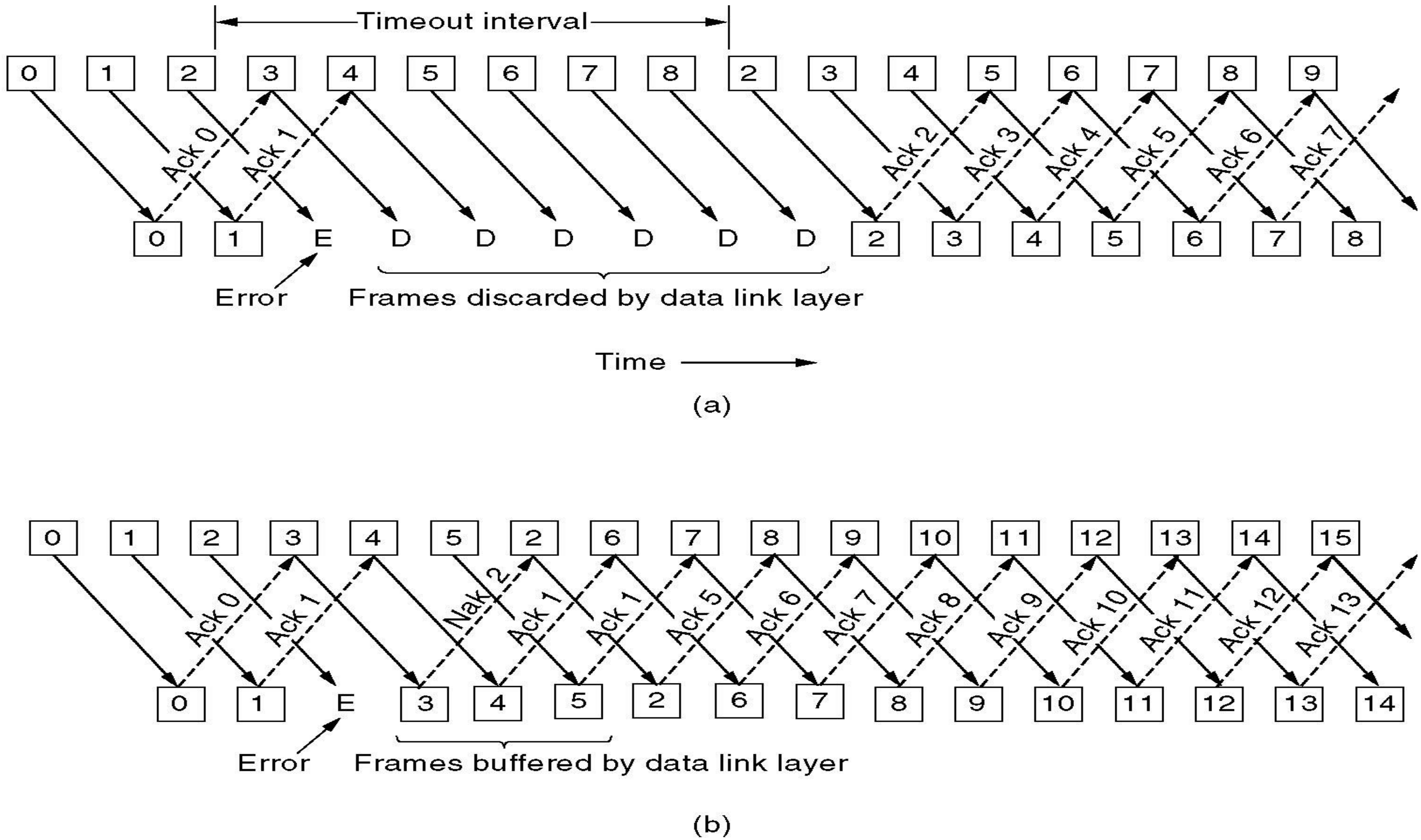


Fig. 3.14: Pipelining and error recovery. Effect on an error when (a) receiver's window size is 1; (b) receiver's window size is large.

3.3.3 A Sliding Window Protocol Using Selective Repeat

- **Problem**: Frames need not be received in order, i.e. we may have an undamaged frame #N, while still waiting for an undamaged version of #N-1.
- **Solution**: Avoid overlapping sending and receiving windows⇒The maximum window size should be at most half the range of the sequence numbers.

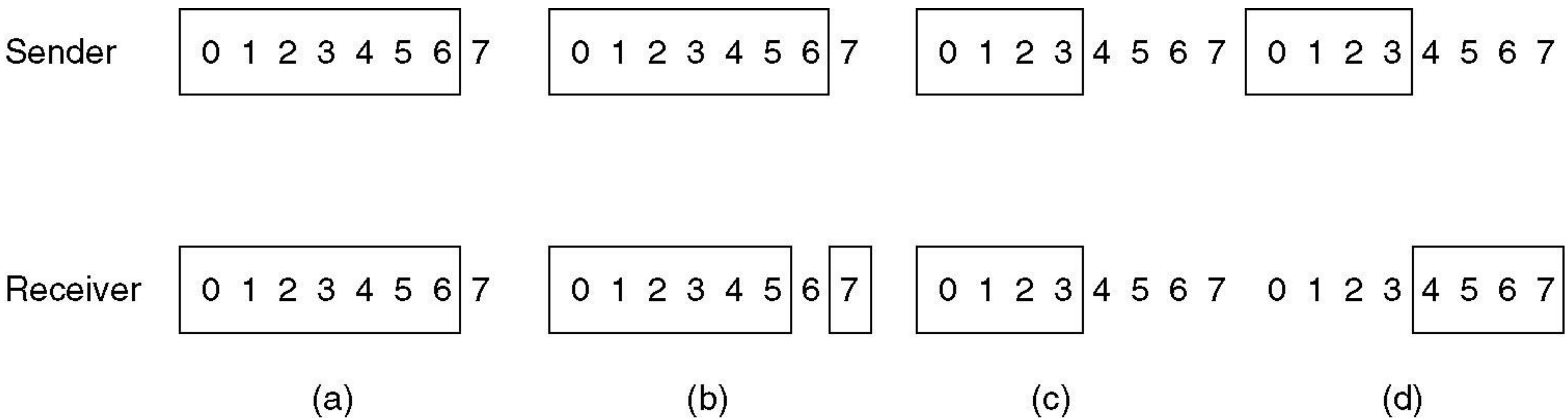


Fig. 3.15: (a) Initial situation with a window size seven; (b) After seven frames sent and received, but not acknowledged; (c) Initial situation with a window size of four; (d) After four frames sent and received, but not acknowledged.

3.4 Example Data Link Protocols

3.4.1 HDLC—High-Level Data Link Control

- A bit-oriented, pretty old but still widely used ISO standard protocol.
- Use **bit stuffing** for framing.
- Frame structure:
  - **Address field**: important with multiple terminals.
  - **Control field**: sequence numbers, acknowledgments, and others.
  - **Checksum field**: CRC.

Bits	8	8	8	≥ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0

Fig. 3.16: Frame format for bit-oriented protocols.

3.4.2 The Data Link Layer in the Internet

- Most wide area infrastructure is built up from point-to-point leased lines.
- **Point-to-point communication**: router-to-router traffic and home user-to-ISP traffic.

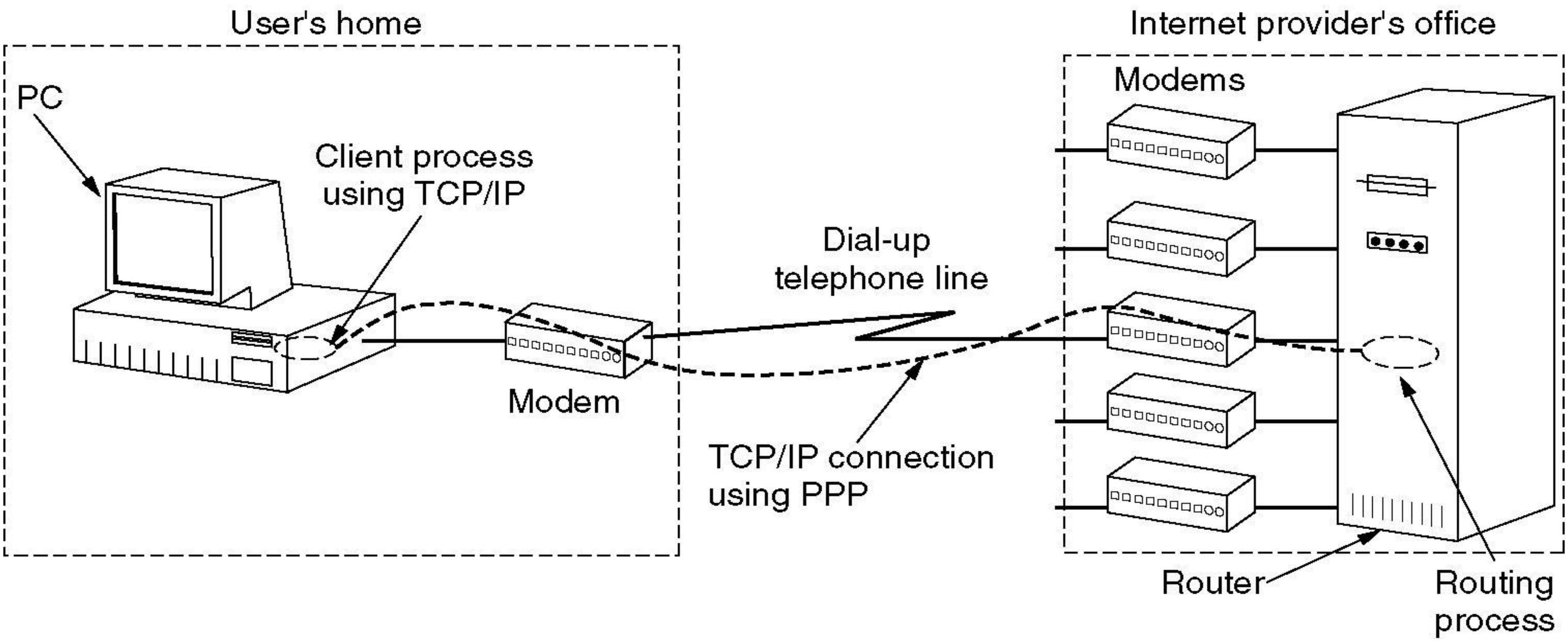


Fig. 3.17: A home personal computer acting as an Internet host.

PPP – Point to Point Protocol

- Three features:
  - A **framing** method: marks the start and the end of frames.
  - A **Link Control Protocol (LCP)**: controls lines (set-up, test, negotiating options, tear-down).
  - A **Network Control Protocol (NCP)**: negotiates network-layer options.

PPP Example—PC Connection to the Internet

- PC calls the ISP's router via a modem to establish a physical connection.
- PC & ISP router negotiate the connection parameters by exchanging LCP packets in the payload field of PPP frames.
- Once the parameters are agreed, NCP packets are sent to configure the network layer.
- To run a TCP/IP protocol stack, the PC needs an IP address.
  - As IP addresses are not unlimited, it is not possible for each PC to have one.
  - An ISP will dynamically assign a PC address to each newly attached PC.
  - The NCP for IP assigns an IP address.
- On disconnection, NCP closes the network layer connection and releases the IP address.
- LCP shuts down the data link connection.
- The PC modem shuts down the physical connection.

PPP Frame Format

- Different from HDLC:
  - **Character oriented** rather than bit oriented.
  - Use **byte stuffing** for framing so that all frames are an integer number of bytes.
- Frame format:
  - The **address field** (11111111) and **control field** (00000011) are rarely used.
  - **Protocol field**: tells what kind of packet (LCP, NCP, IP etc.) is in the Payload field.
  - **Payload field**: variable length; can use a default length of 1500 bytes.
  - **Checksum field**: normally 2 bytes, can be 4 bytes.

Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

Fig. 3.18: The PPP full frame format for unnumbered mode operation.