COMPUTER NETWORKS-PT 2

ANSWER KEY

19)

1️⃣Purpose of Sequence Number in Go-Back-N Protocol
Sequence numbers uniquely identify each packet in transmission.

They help the receiver detect:

*Out-of-order packets

*Duplicate packets

*Missing packets

In Go-Back-N, the sender can transmit multiple packets (up to window size) without waiting for individual ACKs, but uses sequence numbers to track which packets are acknowledged and which need retransmission.

2️⃣Stop-and-Wait Protocol Construction + Prediction

🔧 Stop-and-Wait Protocol (for packet 3)

Send packet 0 → receive ACK 1
Send packet 1 → receive ACK 2
Send packet 2 → receive ACK 3
Send packet 3 → lost → timeout → retransmit → receive ACK 4
Send packet 4 → lost → timeout → retransmit → receive ACK 5
Send packet 5 → receive ACK 6

- Sender sends packet 3 and waits for ACK.
- If ACK is received → send packet 4.

- If packet 3 is lost or ACK not received → sender times out and retransmits packet 3.

🔮 **Prediction in Go-Back-N (if packet 3 is lost and ackNo = 3 not received)**

- Receiver only accepts packets in order.
- Packets 4 and 5 are discarded (even if received) because packet 3 is missing.
- Sender times out and **retransmits packet 3 and all subsequent packets** (Go-Back-N behavior).
- Once packet 3 is acknowledged, window slides forward and transmission resumes.

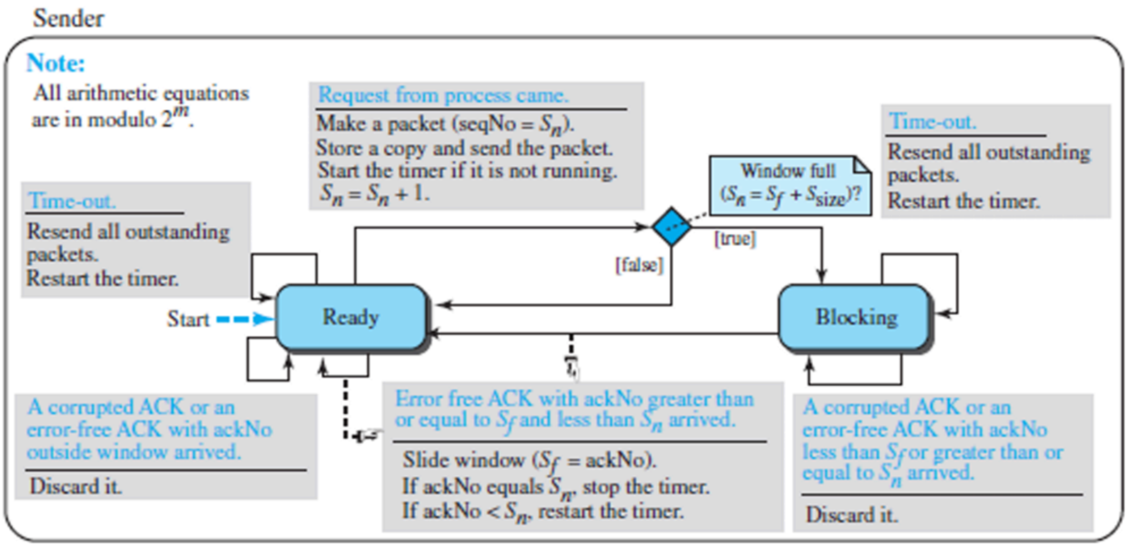
3️⃣ Handling Packet Loss in Go-Back-N Protocol

✅ **Mechanism to Handle Loss:**

- **Timeout-based retransmission**: If ACK not received within timeout, sender retransmits all unacknowledged packets starting from the lost one.
- **Cumulative ACKs**: Receiver sends ACK for the last correctly received in-order packet.
- **Sliding window**: Once ACK is received, sender slides window forward and sends new packets.

20)

i)



ii) 🔌 Connection Interrupted Between Component 1 and 2

- **Component 1**: Application generating data
- **Component 2**: GBN Sender transmitting packets

Impact of interruption:

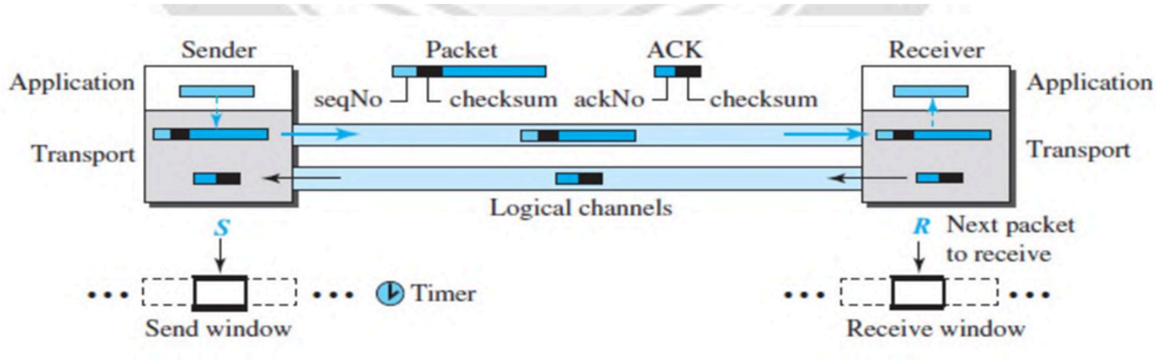
- No new packets passed to the sender
- Sender becomes idle (no fresh data to send)
- Only retransmits outstanding packets (if any) on timeout
- Data transmission halts until connection is restored

iii. 🛑 Handling corrupted ACK outside the window

- **Corrupted ACK** or **ACK with ackNo outside sender window** is considered invalid.
- Sender **discards the ACK** — no action taken.
- Timer continues running for unacknowledged packets.
- If no valid ACK arrives before timeout → **retransmit all outstanding packets**.

21)

i)



ii) Statement: *Stop-and-Wait sends multiple messages before accepting ACK.*

- **Answer: ❌ False**
- **Justification:** In Stop-and-Wait, the sender transmits **only one packet** and waits for its ACK before sending the next. Sending multiple without waiting is a feature of pipelined protocols like Go-Back-N or Selective Repeat, not Stop-and-Wait.

iii) Purpose of checksum in Packet 9 and 10:

- ✅ **Checksum is for error detection** in both **data packets** (Packet 9) and **ACKs** (Packet 10).
- It ensures integrity of transmitted information:
    - **Packet 9:** Receiver checks the checksum to confirm data is not corrupted. If error → discard packet → sender retransmits.

- ○ **Packet 10 (ACK):** Sender checks the checksum to verify ACK correctness. If error → ACK discarded → sender relies on timeout to retransmit.

- Thus, checksum guarantees **reliable communication** over noisy channels by detecting errors early.

—--

22)

i. Sequence Number of the 18th Packet
Formula:

SeqNum(n)=(n−1)/mod 16

Calculation:

SeqNum(18)=(18−1)mod 16

=17 mod 16

=1

Answer: 1

---

ii. Sequence Number Pattern
Sequence Numbers:

0,1,2,…,15,0,1,2,…

Formula:

$\text{SeqNum}(n) = (n-1) \bmod 16$

Go-Back-N Transmission (Window Size = W)

Steps:

1. Sender transmits **W packets** using sequence numbers modulo 16.
2. Receiver sends **ACK for last in-order packet**.
3. If a packet is lost, sender **goes back and retransmits from the lost packet onward**.

Example (W = 4):

- Sent: 0, 1, 2, 3
- If packet 2 is lost → sender retransmits: 2, 3, 4, 5

23)

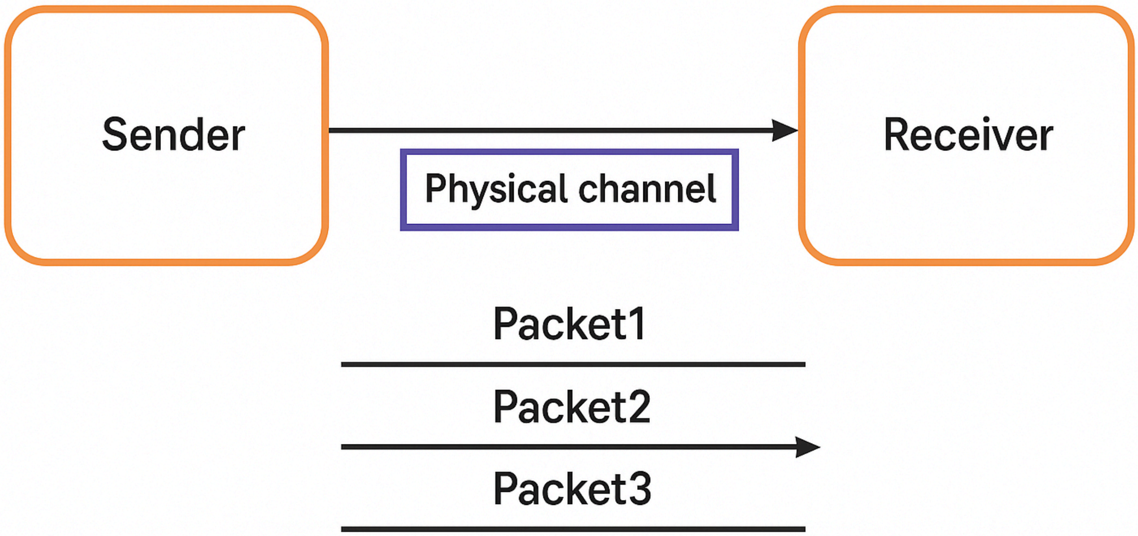


Figure: Connectionless Protocol

24)

- Distance = 4000 km = 4×10^6 m
- Propagation speed = 3×10^6 m/s
- Bandwidth = 300×10^6 bps
- Packet size = 10^5 bits

Step 1: Propagation delay

$$t_p = \frac{Distance}{Speed} = \frac{4 \times 10^6}{3 \times 10^6} = 1.33 \text{ sec}$$

Step 2: Round Trip Time (RTT)

$$RTT = 2 \times t_p = 2 \times 1.33 = 2.66 \text{ sec}$$

Step 3: Bandwidth–Delay Product

$$= Bandwidth \times RTT = 300 \times 10^6 \times 2.66 = 800 \times 10^6 \text{ bits}$$

Step 4: Number of packets in transit

$$\frac{800 \times 10^6}{10^5} = 8000 \text{ packets}$$

Step 5: For Go-Back-N → $2^k - 1 \geq 8000$

$$2^{13} = 8192 \geq 8000$$

✅ Minimum sequence number field = 13 bits

25)

i. DHCP Message Functions

1. **DHCPDISCOVER – Client searches for DHCP servers**
2. **DHCPOFFER – Server offers IP configuration**
3. **DHCPREQUEST – Client requests offered IP**
4. **DHCPACK – Server confirms IP lease**
5. **DHCPDECLINE – Client rejects offered IP**
6. **DHCPRELEASE – Client releases IP**

7. **DHCPINFORM – Client requests config without IP**
8. **DHCPNAK – Server denies request**

ii. Numbering of Options in DHCP Message Pattern

Value	Message Type
1	DHCPDISCOVER
2	DHCPOFFER
3	DHCPREQUEST
4	DHCPDECLINE
5	DHCPACK
6	DHCPNAK
7	DHCPRELEASE
8	DHCPINFORM

Each option uses: `[Tag] [Length] [Value]` format.
Magic cookie: `99.130.83.99` marks start of options.

26)

i. Discuss how the webpage looks without DNS.

Without DNS, webpages cannot load, resulting in errors like "Server not found." Manual IP address entry is required, but this is impractical due to complexity and frequent changes, rendering the internet unusable and disrupting services like email.

(or)

Without DNS, users must enter numerical IP addresses (like 192.0.0.1) to access websites instead of easy-to-remember domain names. This makes browsing difficult, confusing, and impractical.

ii. Reconstruct the figure for the given DNS (www.bitsathy.com).

```
 • (Root)
 |
com
 |
bitsathy
 |
Www
```

The root node (●) is at the top.

It connects down to the "com" node.

From "com", it goes down to "bitsathy".

From "bitsathy", it finally goes to the "www" node.

—---

27.

i. Polling every 5 seconds can slow network performance due to increased traffic and processing load, especially when many devices are queried simultaneously.

ii. Increasing the polling interval to 20 seconds reduces overhead and bandwidth usage, improving network efficiency with minimal impact on monitoring accuracy

---

28.

1. Total Number of Addresses in ISP Block

- **Given**: `16.12.64.0/10`
- **Calculation**: (2^{32 − 10} = 2^{22} = 4,194,304) addresses

---

2. Allocation for 4 Organizations

- **Each organization needs**: 16 addresses → (2^4)
- **Subnet size**: `/28` (since (32 − 4 = 28))
- **Total used**: (4 × 16 = 64) addresses

---

3. Address Range of ISP Block

- **Starting IP**: `16.12.64.0`
- **Ending IP**: `16.76.63.255`
- **Range**: `16.12.64.0` → `16.76.63.255`

✅ Final Answer

- **Total addresses**: 4,194,304
- **Each organization gets**: `/28` (16 addresses)

- **Overall ISP range**: `16.12.64.0` – `16.76.63.255`

29.

i. Two main parts of the DNS message

1. **Query** – The request sent by a client to a DNS server asking for information about a domain.

2. **Response** – The reply from the DNS server containing the requested information or an error.

ii. Arrangement of DNS Message Components

🔹 **Query Part:**

- Header
- Question Records

🔹 **Response Part:**

- Header
- Question Records
- Answer Records
- Authoritative Records
- Additional Records

—---

30.

Step 1: Convert Binary → Decimal (Number)

1. **Get Request:** `10100001` → $1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
   = 128 + 32 + 1 = **161**
2. **Set Request:** `10100010` → 128 + 32 + 2 = **162**
3. **Inform Request:** `10100000` → 128 + 32 + 0 = **160**
4. **Report:** `10101001` → 128 + 32 + 8 + 1 = **169**

Step 2: Convert Binary → Hexadecimal

1. **Get Request:** `10100001` → `A1` (Hex)
2. **Set Request:** `10100010` → `A2`
3. **Inform Request:** `10100000` → `A0`
4. **Report:** `10101001` → `A9`

✅ Final Table

Data	Number	Whole Tag (Binary)	Whole Tag (Hexa)
Get Request	161	10100001	A1
Set Request	162	10100010	A2
Inform Request	160	10100000	A0
Report	169	10101001	A9