IN403 - Advanced Networks: Exercise Collection

Chapter 1 – TCP and Reliable Data Transfer

Exercise 1 – TCP 3-Way Handshake Scenarios

Problem: A client attempts to connect to a server. The client sends a SYN with `Seq=500`. The server responds with a SYN-ACK with `Seq=1200`.

Questions

Part A: What is the Acknowledgement Number in the server's SYN-ACK?

Part B: What is the Sequence Number and Acknowledgement Number in the client's final ACK?

Part C: If the client's final ACK is lost, what state will the server remain in, and what will the server eventually do?

Solution

Part A:

- The server's ACK number acknowledges the client's SYN
- SYN flag consumes 1 sequence number
- Client sent: `Seq=500`
- **Server's ACK = 500 + 1 = 501**

Part B:

- Client's Sequence Number: The client's SYN consumed sequence 500, so the next segment starts at **Seq=501**
- Client's Acknowledgement Number: The server sent `Seq=1200` with SYN flag (consumes 1), so **ACK=1201**

Answer: `Seq=501`, `ACK=1201`

Part C:

- After sending SYN-ACK, the server enters the **SYN_RECEIVED** state
- The server waits for the client's ACK to transition to **ESTABLISHED**
- If the ACK is lost, the server remains in **SYN_RECEIVED** state
- Eventually, the server will **timeout and retransmit the SYN-ACK**
- After multiple retransmission failures, the server will **close the connection** and return to **LISTEN** state

Exercise 2 – TCP Congestion Control (Tahoe Algorithm)

Problem: A TCP connection uses the Tahoe algorithm. Current `cwnd = 8 MSS` and `ssthresh = 16 MSS`.

Questions

Part A: After one successful RTT, what will the new cwnd be?

Part B: If a Timeout occurs when cwnd = 12 MSS, what are the new values for ssthresh and cwnd?

Solution

Part A:

- Compare cwnd to ssthresh: 8 < 16
- Since cwnd < ssthresh, we are in **Slow Start** phase
- In Slow Start, cwnd doubles each RTT (increases by 1 MSS for each ACK received)
- **New cwnd = 8 × 2 = 16 MSS**

Part B:

- On timeout, TCP Tahoe reacts as follows:
  - **ssthresh = cwnd / 2 = 12 / 2 = 6 MSS**
  - **cwnd = 1 MSS** (reset to initial value)
- The connection returns to Slow Start phase

Answer: `ssthresh = 6 MSS`, `cwnd = 1 MSS`

Exercise 3 – TCP Retransmission and Cumulative ACKs

Problem: Host A sends two segments to Host B:

- Segment 1: `Seq=92` (8 data bytes)
- Segment 2: `Seq=100` (20 data bytes)
- ACK for Segment 1 is lost
- ACK for Segment 2 (`Ack=120`) arrives before timeout for Segment 1

Question: Will Host A retransmit Segment 1? Explain using cumulative ACKs.

Solution

No, Host A will NOT retransmit Segment 1.

Explanation:

- Segment 1 contains bytes 92-99 (8 bytes), so an ACK for it would be `ACK=100`
- Segment 2 contains bytes 100-119 (20 bytes), so an ACK for it is `ACK=120`
- TCP uses **cumulative acknowledgments**

- When Host A receives `ACK=120`, this means: *"I have successfully received all bytes up to and including byte 119"*
- Since `ACK=120` acknowledges all bytes through 119, it implicitly acknowledges:
  - Bytes 92-99 (Segment 1)
  - Bytes 100-119 (Segment 2)
- Even though the explicit ACK for Segment 1 was lost, the cumulative `ACK=120` confirms its successful delivery
- Therefore, no retransmission is needed

Key Concept: Cumulative ACKs make TCP robust to ACK loss because later ACKs implicitly acknowledge earlier data.

Chapter 2 – Quality of Service (QoS)

Exercise 4 – Serialization Delay Calculation

Problem: Calculate the serialization delay for a 1500-byte packet transmitted over a 10 Mbps link.

Solution

Formula:

```
Serialization Delay = Packet Size / Link Bandwidth
```

Step 1: Convert packet size to bits

- Packet size = 1500 bytes × 8 bits/byte = **12,000 bits**

Step 2: Convert link bandwidth to bps

- Link bandwidth = 10 Mbps = 10×10^6 = **10,000,000 bps**

Step 3: Calculate delay

```
Delay = 12,000 bits / 10,000,000 bps
      = 0.0012 seconds
      = 1.2 milliseconds
```

Answer: 1.2 ms

Exercise 5 – Leaky Bucket Policing

Problem: A Leaky Bucket has:

- Bucket size `b = 20 tokens`

- Token generation rate `r = 10 tokens/second`
- Time interval `t = 5 seconds`

Question: Calculate the maximum number of packets that can be admitted.

Solution

Formula:

```
Max packets = b + (r × t)
```

Where:

- `b` = initial bucket capacity (burst size)
- `r × t` = tokens generated during the interval

Calculation:

```
Max packets = 20 + (10 × 5)
            = 20 + 50
            = 70 packets
```

Explanation:

- The bucket starts with 20 tokens (allows initial burst)
- During 5 seconds, 50 more tokens are generated (10/sec × 5 sec)
- Total tokens available = 70
- Assuming 1 token per packet, **70 packets** can be admitted

Answer: 70 packets

Exercise 6 – DiffServ DSCP Marking

Problem: A packet has DSCP decimal value of 14.

Questions

Part A: Represent this value in 6-bit binary

Part B: What is the corresponding Assured Forwarding (AF) PHB name?

Solution

Part A:

```
Decimal: 14
Binary:  001110
```

Conversion check:

- 0×32 + 0×16 + 1×8 + 1×4 + 1×2 + 0×1 = 8+4+2 = 14 ✓

Answer: 001110

Part B:

DiffServ AF encoding uses the format: `XXXYYY00` where:

- `XXX` = Class (1-4)
- `YYY` = Drop Precedence (1-3)
- Last 2 bits are always 00

For DSCP = 14 (001110):

- `XXX = 001` → Class 1
- `YYY = 110` → But wait, this is decimal 6, which doesn't fit the AF pattern

Actually, DSCP 14 (001110) corresponds to AF13:

- AF13 in binary: `001110` ✓
- Class Selector: CS1 family
- **AF13 = Assured Forwarding Class 1, High Drop Precedence**

In standard notation:

- AF13 → Class **1**, Drop Precedence **3** (highest drop probability in class 1)

Answer: AF13 (Assured Forwarding 13)

Chapter 3 – Network Function Virtualization (NFV)

Exercise 1 – VNF Placement (Resource Calculation)

A data center has:

- 4 physical servers
- Each server: 16 vCPUs, 32 GB RAM

A VNF requires:

- 4 vCPUs
- 8 GB RAM

Questions

1. How many VNF instances can be deployed per server?
2. What is the maximum total VNFs in the data center?

Solution

Per server:

- CPU limit: 16 / 4 = 4 VNFs
- RAM limit: 32 / 8 = 4 VNFs

👉 Limiting factor = both equal

Answer:

- VNFs per server = 4
- Total VNFs = 4 servers × 4 = **16 VNFs**

Exercise 2 – Scaling Decision (Threshold Logic)

Policy: If CPU > 80% for 5 minutes → scale out by 1 VNF

Current:

- 2 VNFs
- Each VNF = 4 vCPUs
- Traffic needs = 12 vCPUs

Solution

Current capacity:

- 2 × 4 = 8 vCPUs

Required:

- 12 vCPUs

CPU usage:

- 12 / 8 = 150% ❌

After scale-out:

- 3 VNFs × 4 = 12 vCPUs
- CPU usage = 100%

✅ Scaling stops

Answer:

→ VNFM triggers scale-out, final VNFs = **3**

Exercise 3 – Architecture Identification

You are given a telecom cloud with:

- OpenStack managing compute and networking
- A virtual firewall image stored in QCOW2 format
- A component that instantiates, scales, and heals the firewall
- A component that deploys a full VPN service (firewall + router)

Questions

1. Identify the VIM, VNFM, NFVO, VNF, and NFVI
2. Which component communicates with OSS/BSS?
3. Which reference point is used when scaling the firewall?

Answer

1. **VIM** = OpenStack (it manages compute/network/storage).
   VNFM = component that instantiates/scales/heals the firewall VNF.
   NFVO = component deploying full VPN service (service chain lifecycle).
   VNF = the virtual firewall software running as VM/container.
   NFVI = the commodity servers + storage + networking + hypervisor layer.
2. **NFVO** communicates with OSS/BSS (northbound interface).
3. Scaling uses **Or–Vnfm** (NFVO ↔️ VNFM).

Exercise 4 – Lifecycle Flow (Scenario)

A customer orders a 5G enterprise slice.

Questions

1. List the exact sequence of interactions starting from BSS → OSS → NFVO → VNFM → VIM
2. What happens if CPU usage exceeds 80% for 5 minutes?
3. Who decides *where* the new VNF instance is placed?

Answer

1. **BSS → OSS → NFVO → VNFM → VIM → NFVI**
   - OSS sends service request to NFVO
   - NFVO selects blueprint + orchestrates
   - VNFM instantiates VNFs
   - VIM allocates virtual resources on NFVI
2. If CPU >80% for 5 minutes → **VNFM** executes scale-out/scale-up based on policy (often triggered by NFVO policy).
3. Placement decision is made by **NFVO** using global policies, then enforced via VIM resource allocation.

Exercise 5 – Compare & Explain

Fill the table:

Feature	Traditional Network	NFV
Hardware dependency		
Scaling		
Service deployment time		
Vendor lock-in		

Answer

Feature	Traditional	NFV
Hardware dependency	Dedicated proprietary appliances	Commodity servers + VNFs
Scaling	Add physical boxes	Elastic on-demand scaling
Service deployment time	Slow/manual	Rapid via orchestration
Vendor lock-in	High	Reduced (decouple HW/SW)

Exercise 6 – Failure Analysis

A physical server fails:

Questions

1. Which component detects it?
2. Which component triggers healing?
3. Which reference point carries the fault notification?

Answer

1. Physical server failure is detected by **VIM** (infrastructure fault/performance monitoring).
2. Healing is initiated by **VNFM** (restart/rebuild/migrate VNF).
3. Fault comes from NFVI → VIM via **Nf–Vi**, then VIM can notify VNFM (**Vi–Vnfm**) and NFVO (**Or–Vi**).

Exercise 7 – ETSI Framework Roles for 5G IoT Slice

Describe the specific technical steps taken by the NFVO, VNFM, and VIM when launching a new 5G Slice for IoT.

Solution

1. NFV Orchestrator (NFVO) - The "Service Designer & Conductor"

Steps:

1. **Service Composition:** Receives the 5G IoT slice request and determines which VNFs are needed (e.g., vEPC, vIMS, IoT Gateway VNF, Firewall VNF)
2. **Resource Planning:** Calculates total resource requirements (CPU, RAM, storage, network bandwidth)
3. **NS Instantiation:** Creates a Network Service (NS) instance and generates a deployment plan
4. **Resource Orchestration:** Coordinates with VIM to check resource availability across data centers
5. **VNF Lifecycle Request:** Instructs multiple VNFMs to instantiate required VNFs
6. **Service Chaining:** Configures the virtual links between VNFs to create the end-to-end service
7. **Monitoring:** Tracks the overall service health and performance

2. VNF Manager (VNFM) - The "Application Manager"

Steps:

1. **VNF Instantiation:** Receives requests from NFVO to create specific VNF instances (e.g., one VNFM handles the vEPC)
2. **Image Selection:** Retrieves the appropriate VNF software image from the VNF catalog
3. **Configuration Generation:** Creates the initial configuration for the VNF (IP addresses, routing tables, policies)
4. **Lifecycle Coordination:** Requests VIM to create the actual VM/container
5. **VNF Configuration:** Once the VM is running, pushes the configuration to the VNF
6. **Health Monitoring:** Continuously monitors VNF health (CPU, memory, connection status)
7. **Scaling Operations:** Can scale VNF instances up/down based on IoT traffic patterns
8. **Healing:** Detects VNF failures and initiates recovery procedures

3. Virtualized Infrastructure Manager (VIM) - The "Data Center Manager"

Steps:

1. **Resource Allocation:** Receives VM/container creation requests from VNFM
2. **Compute Provisioning:** Allocates CPU cores, RAM, and storage from the physical infrastructure
3. **Hypervisor Management:** Instructs the hypervisor (e.g., KVM, VMware) to create VMs
4. **Network Configuration:** Sets up virtual switches, VLANs, and network connections
5. **Storage Attachment:** Attaches virtual disks and persistent storage volumes
6. **IP Assignment:** Assigns management and data plane IP addresses
7. **Resource Monitoring:** Reports physical resource usage (CPU %, memory %, disk I/O)
8. **Infrastructure Tracking:** Maintains an inventory of all physical and virtual resources

End-to-End Flow:

```
Service Request → NFVO (orchestrates) → VNFM (manages VNF lifecycle) → VIM (provides infrastructure) -
```

Exercise 8 – Power Grid Analogy - Appliance Specialist

Using the Power Grid analogy, identify which NFV component corresponds to the Appliance Specialist and explain their role when a Virtual Firewall fails.

Solution

The Appliance Specialist = VNFM (VNF Manager)

Analogy Mapping:

- **Power Plant** = VIM (manages raw infrastructure)
- **Power Grid Operator** = NFVO (orchestrates the whole system)
- **Appliance Specialist** = VNFM (knows how specific appliances/VNFs work)

Why VNFM is the Appliance Specialist:

Just as an appliance specialist understands how specific devices (refrigerators, air conditioners) work and can troubleshoot them, the VNFM:

- Has **deep knowledge** of specific VNF types (firewalls, load balancers, routers)
- Understands the **internal workings** and configuration of each VNF
- Knows how to **diagnose VNF-specific problems**

Critical Role When Virtual Firewall (vFW) Fails:

1. **Failure Detection:** VNFM's monitoring detects the vFW has stopped responding
2. **Diagnosis:** Determines if it's a configuration issue, software crash, or resource starvation
3. **Healing Decision:** Decides on the appropriate action:
    - **Restart:** If it's a software crash
    - **Reconfigure:** If it's a configuration error
    - **Recreate:** If the VM is corrupted
4. **Coordination with VIM:** Requests new resources if VM recreation is needed
5. **State Recovery:** Restores the vFW configuration and firewall rules
6. **Verification:** Tests that the vFW is functioning correctly before returning to service
7. **NFVO Notification:** Informs the NFVO that service has been restored

Why This Role is Critical:

- Without VNFM, the system would know "something is broken" but not "how to fix this specific VNF"
- VNFM provides the **specialized knowledge** needed for VNF-specific operations
- Generic infrastructure management (VIM) can't handle application-layer issues

Chapter 4 – Software Defined Networking (SDN)

Exercise 7 – Flow Table Decision (Match–Action Logic)

Flow table:

Priority	Match	Action
100	TCP dst port = 22	DROP
50	IP dst = 10.0.0.0/8	FORWARD port 2
10	ANY	FORWARD port 1

Packet:

- Src: 192.168.1.5
- Dst: 10.1.2.3
- TCP dst port: 22

Question: What happens to the packet?

Solution

Step 1: Check highest priority

✔️ TCP dst port = 22 → match

Step 2: Stop searching (highest priority wins)

Answer:

➡️ Packet is **DROPPED**

Exercise 8 – Longest Prefix Matching

Routing table:

Prefix	Interface
192.168.0.0/16	1
192.168.1.0/24	2
192.168.1.128/25	3

Packet destination: 192.168.1.130

Question: Which interface is used?

Solution

Check prefixes:

- /16 → match
- /24 → match
- /25 → match (most specific)

Answer:

➡️ Forward to **Interface 3**

Exercise 9 – Control vs Data Plane

Explain why SDN separates control and data planes.

Then answer:

1. Where are routing decisions made?
2. Where is packet forwarding performed?
3. Why does this reduce vendor lock-in?

Answer

- SDN separates planes so control becomes programmable + centralized while forwarding stays fast in hardware.

1. Routing decisions: **controller / control plane**
2. Packet forwarding: **switches/routers data plane**
3. Vendor lock-in reduced because control uses **open APIs** (OpenFlow + standard interfaces).

Exercise 10 – OpenFlow Match-Action

Given this rule set:

Match	Action
src=10.1.2.3	send to controller
dst=3.4..	forward port 2
dst TCP port 22	drop

Questions

1. What happens to packet 10.1.2.3 → 8.8.8.8?
2. What happens to packet 192.168.1.1 → 3.4.5.6?
3. Which rule has highest priority?

Answer

Given rules:

- src=10.1.2.3 → controller

- dst=3.4.. → forward port2
- TCP dst port 22 → drop

1. 10.1.2.3 → 8.8.8.8 → **sent to controller** (matches src rule).
2. 192.168.1.1 → 3.4.5.6 → **forward port 2**.
3. Highest priority = **most specific match** (exact src or exact TCP port beats wildcard prefix). Priorities are used to resolve overlaps.

Exercise 11 – SDN Controller Design

Design an SDN solution for a data center that requires:

- Load balancing
- Firewalling
- Traffic engineering

Questions

1. Which SDN controller architecture fits best (centralized, distributed)?
2. Which northbound APIs are needed?
3. Which southbound protocol is used?

Answer

1. Best architecture: **distributed controller** (scalability + reliability).
2. Northbound APIs needed: apps for load balancing, firewall policies, traffic engineering intents/**REST**.
3. Southbound protocol: **OpenFlow** to install flow rules.

Exercise 12 – CIDR Aggregation (Binary Math)

Networks:

- 172.16.0.0/24
- 172.16.1.0/24
- 172.16.2.0/24
- 172.16.3.0/24

Question: What single CIDR prefix can summarize them?

Solution

4 subnets → 2^2 → increase prefix by 2 bits

Original: /24

Aggregated: /22

Check: 172.16.0.0/22

Answer:

➡️ Aggregated prefix = **172.16.0.0/22**

Exercise 13 – OpenFlow Flow Table Logic

An SDN switch receives a packet with:

- `Src IP = 10.1.0.5`
- `Dst IP = 10.2.0.3`
- `TCP Dport = 22`

Flow Table:

- Entry 1 (Higher Priority): `Match: IP Dst=10.2.0.3; Action: Forward(3)`
- Entry 2 (Lower Priority): `Match: TCP Dport=22; Action: Drop`

Question: What action will the switch take?

Solution

Answer: Forward the packet out of port 3

Explanation:

OpenFlow switches process flow tables with **priority-based matching**:

1. **Priority Order:** Entry 1 has higher priority than Entry 2
2. **Matching Process:**
   - Switch checks Entry 1 first: Does `Dst IP = 10.2.0.3` ? **YES** ✓
   - Entry 1 MATCHES
3. **Action Execution:** When a match is found, the switch:
   - Executes the action: **Forward(3)**
   - **STOPS** processing (does not check lower priority entries)
4. **Entry 2 is Never Evaluated:** Even though the packet also matches Entry 2 (`TCP Dport=22`), this entry is never checked because a higher priority match was found

Key Principle: First match wins in OpenFlow. Once a flow entry matches, its action is executed and table processing stops.

Real-World Implication:

- This allows specific rules to override general rules
- Example: "Forward all traffic to 10.2.0.3" (specific) overrides "Drop all SSH traffic" (general)
- To drop SSH to 10.2.0.3, you'd need a higher-priority rule specifically matching both conditions

---

Exercise 14 – SDN Link Failure Handling - Six-Step Process

Detail the six-step process when a link fails between two SDN switches.

Solution

Step 1: Failure Detection

- **Who:** The SDN switch adjacent to the failed link
- **How:** Detects link down via:
  - Physical layer notification (no carrier signal)
  - LLDP (Link Layer Discovery Protocol) timeout
  - OpenFlow echo request/reply timeout
- **Result:** Switch recognizes the link is unavailable

Step 2: Southbound API Notification (Switch → Controller)

- **Who:** The switch
- **Protocol:** OpenFlow OFPT_PORT_STATUS message
- **Content:**
  - Port number that went down
  - New port state (LINK_DOWN)
  - Reason code (e.g., physical link failure)
- **Direction:** Southbound API (data plane → control plane)

Step 3: Controller State Update

- **Who:** SDN Controller
- **Action:** Updates its network topology database:
  - Marks the link as inactive
  - Updates the network graph representation
  - Removes affected paths from the topology
- **Result:** Controller now has accurate view of network topology

Step 4: Network Application Notification

- **Who:** Controller
- **What:** Notifies relevant network applications (e.g., routing app, monitoring app)
- **How:** Internal northbound API or event bus
- **Content:** Topology change event with affected link details

Step 5: Routing Application Recomputation

- **Who:** Routing application (runs on top of controller)
- **Action:**

    - Detects that current paths use the failed link
    - Runs path computation algorithm (e.g., Dijkstra's) with updated topology
    - Calculates new alternative paths avoiding the failed link
    - Determines which flow tables need updating
- **Result:** New routing decisions generated

Step 6: Flow Table Updates (Controller → Switches)

- **Who:** Controller
- **Protocol:** OpenFlow OFPT_FLOW_MOD messages
- **Direction:** Southbound API (control plane → data plane)
- **Actions:**
    - **Delete** old flow entries that used the failed link
    - **Install** new flow entries with alternative paths
    - Update multiple switches along the new path
- **Result:** Network traffic rerouted around the failure

Complete Flow Diagram:

```
[Physical Link Fails]
        ↓
[Switch Detects] ← Step 1
        ↓
[OpenFlow Port Status Message] ← Step 2 (Southbound API)
        ↓
[Controller Updates Topology] ← Step 3
        ↓
[Routing App Notified] ← Step 4
        ↓
[New Paths Computed] ← Step 5
        ↓
[Flow Mods Sent to Switches] ← Step 6 (Southbound API)
        ↓
[Traffic Rerouted]
```

Key Concepts Demonstrated:

- **Southbound API:** Used twice (notification up, flow mods down)
- **Controller State Management:** Centralized topology database
- **Application Layer:** Routing logic separated from controller core

Chapter 5 – BGP

Exercise 13 – BGP Route Selection

You receive the following routes to 140.20.1.0/24:

Path	LocalPref	AS_PATH	MED
AS2 AS5	200	2 hops	50
AS3 AS7 AS9	100	3 hops	10

Questions

1. Which route is chosen and why?
2. Which attribute dominates the decision?
3. What changes if LocalPref is equal?

Answer

Routes to 140.20.1.0/24:

- Path1: LocalPref 200, AS_PATH 2 hops, MED 50
- Path2: LocalPref 100, AS_PATH 3 hops, MED 10

1. Choose **Path1** because highest LocalPref wins first.
2. Dominant attribute here: **LOCAL_PREF**.
3. If LocalPref equal → choose **shortest AS_PATH** (hop count).

Example 1 – BGP Route Selection (Complete Process)

Routes learned for prefix 203.0.113.0/24 (all are valid & reachable):

Route	Next Hop	LOCAL_PREF	AS_PATH	MED	ORIGIN	IGP cost to next-hop	Router ID
A	10.0.0.2	100	64510 64520	30	IGP	20	1.1.1.1
B	10.0.0.3	200	64530 64540 64550	5	IGP	5	2.2.2.2
C	10.0.0.4	200	64560 64570	50	EGP	10	3.3.3.3
D	10.0.0.5	200	64580 64590	15	IGP	30	4.4.4.4
E	10.0.0.6	150	64511	0	INCOMPLETE	8	5.5.5.5

Step-by-step selection

Step 1: Highest LOCAL_PREF

Values: A=100, B=200, C=200, D=200, E=150

✅ Keep only highest = {B, C, D} (LOCAL_PREF=200).

Step 2: Shortest AS_PATH (hop-count)

- B: 3 AS hops
- C: 2 AS hops
- D: 2 AS hops

✅ Remove B (longer). Now candidates = {C, D}.

Step 3: MED (lower is better)

- C: MED=50
- D: MED=15

✅ Lower MED wins → choose D.

Final Answer:

✅ **Best route: D**

Reason: Highest LocalPref → shortest AS_PATH tie → lowest MED.

Example 2 – BGP Route Selection (ORIGIN + IGP cost + Router-ID)

Routes learned for prefix 198.51.100.0/24:

Route	Next Hop	LOCAL_PREF	AS_PATH	MED	ORIGIN	IGP cost	Router ID
A	10.0.1.1	150	65010 65100	20	EGP	10	2.2.2.2
B	10.0.1.2	150	65020 65100	20	IGP	40	1.1.1.1
C	10.0.1.3	150	65030 65100	20	IGP	10	3.3.3.3
D	10.0.1.4	150	65040 65100	20	IGP	10	0.9.9.9

Step-by-step selection

Step 1: LOCAL_PREF

All are 150 → keep {A,B,C,D}.

Step 2: AS_PATH length

All are 2 hops → keep {A,B,C,D}.

Step 3: MED

All MED=20 → keep {A,B,C,D}.

Step 4: ORIGIN type (IGP < EGP < INCOMPLETE)

A is EGP, B/C/D are IGP

✅ Remove A. Now candidates = {B,C,D}.

Step 5: Lowest IGP cost

- B cost = 40
- C cost = 10
- D cost = 10

✅ Remove B. Candidates = {C, D}.

Step 6: Router ID tie-break

- C router ID = 3.3.3.3
- D router ID = 0.9.9.9

✅ Lowest router ID wins → choose D.

Final Answer:

✅ **Best route: D**

Reason: everything tied until ORIGIN, then IGP cost, then Router ID.

Example 3 – Import/Export Policy (LOCAL_PREF changes result)

You receive two routes for 140.20.1.0/24:

Route	Learned from	Relationship	Default LOCAL_PREF
X	Provider P1	provider	100
Y	Customer C1	customer	200

Rule: "Prefer customer routes over provider routes" → set higher LOCAL_PREF for customer.

Result

Even if Y has a longer AS_PATH, **Y wins** because LOCAL_PREF is evaluated first.

Key Insight: Policy-based LOCAL_PREF assignment allows operators to override path optimization and enforce business relationships.

Exercise 14 – BGP Path Selection Process

A BGP router receives two advertisements for `192.168.0.0/16`:

- Path A: `Local_Pref=200`, `AS-PATH=<65001, 65002>`, `MED=50`
- Path B: `Local_Pref=100`, `AS-PATH=<65003>`, `MED=10`

Question: Which path will be selected and why?

Solution

BGP Path Selection Process (in order):

1. **Highest LOCAL_PREFERENCE** (prefer higher)
2. Shortest AS_PATH
3. Lowest ORIGIN type
4. Lowest MED
5. eBGP over iBGP
6. Lowest IGP metric to next-hop
7. Lowest BGP router ID

Analysis:

Step 1: Compare LOCAL_PREFERENCE

- Path A: LOCAL_PREF = **200**
- Path B: LOCAL_PREF = **100**
- **Path A has higher LOCAL_PREFERENCE** ✓

Since Path A wins on the **first criterion**, the decision is made.

Answer: Path A is selected

Why the Other Attributes Don't Matter:

- Path B has shorter AS_PATH (1 hop vs 2 hops) → **Irrelevant**
- Path B has lower MED (10 vs 50) → **Irrelevant**
- BGP only considers subsequent attributes if the earlier ones are tied

Key Concept:

- **LOCAL_PREFERENCE** is the **first tiebreaker** after rejecting invalid routes
- It's an **intra-AS** attribute that allows an AS to express routing policy
- Higher LOCAL_PREF = more preferred path
- LOCAL_PREF is **never sent to external peers** (stays within the AS)

Real-World Usage:

- Set higher LOCAL_PREF on preferred provider links
- Set lower LOCAL_PREF on backup links
- Override AS_PATH length to enforce policy decisions

Exercise 15 – CIDR Aggregation

An ISP owns:

- 232.71.0.0/24
- 232.71.1.0/24
- 232.71.2.0/24
- 232.71.3.0/24

Questions

1. What single CIDR prefix can summarize them?
2. Why does CIDR reduce routing table size?
3. What happens if one subnet goes down?

Answer

Subnets: 232.71.0.0/24 to 232.71.3.0/24

1. Summary = **232.71.0.0/22** (covers 4 × /24).
2. CIDR reduces table size by advertising **one aggregated prefix** instead of many.
3. If one /24 goes down, the ISP may need to announce a **more specific prefix** for remaining reachable subnets (de-aggregation) to keep correct reachability.

Exercise 17 – Policy Control

A non-transit AS:

Questions

1. Which routes should it advertise?
2. What happens if it violates export rules?
3. Explain the *customer-provider* relationship effect on routing.

Answer

1. Non-transit AS advertises **its own routes only**, not routes learned from other providers/peers.
2. If it violates export rules → becomes a "free transit" path and may carry huge traffic (customer-transit problem).
3. Relationships affect export:
    - To provider/peer: export own + customers only
    - To customer: export everything (own + learned)

Chapter 5 – OSPF

Exercise 18 – OSPF Cost Calculation

A router has:

- 100 Mbps link
- 10 Mbps link
- Reference bandwidth = 100,000,000 bps

Questions

1. Calculate OSPF cost for each link
2. Which path is preferred?
3. How can cost be manually changed?

Solution

Cost = 100,000,000 / bandwidth(bps).

- 100 Mbps = 100,000,000 bps → cost = 100,000,000 / 100,000,000 = **1**
- 10 Mbps = 10,000,000 bps → cost = 100,000,000 / 10,000,000 = **10**

Preferred path = lower total cost → **100 Mbps link**.

Manual change: `bandwidth` or `ip ospf cost` commands.

Exercise 19 – OSPF Cost Calculation (Alternative)

Reference bandwidth = 100 Mbps

Links:

- Link A = 10 Mbps

- Link B = 100 Mbps
- Link C = 1 Gbps

Solution

Formula: Cost = Reference BW / Interface BW

- A: 100 / 10 = **10**
- B: 100 / 100 = **1**
- C: 100 / 1000 = 0.1 → rounded to **1**

Answer:

- Best link: B or C
- Lowest cost = **1**

Exercise 20 – OSPF Neighbor States

List OSPF neighbor states in order from startup to full adjacency.

Which state:

- Elects DR/BDR?
- Exchanges LSAs?
- Indicates full synchronization?

Answer

Order (standard OSPF): **Down → Init → 2-Way → ExStart → Exchange → Loading → Full**.

- **DR/BDR election** happens on multi-access networks at **2-Way** stage.
- **LSA database exchange** begins in **ExStart/Exchange/Loading**.
- Fully synchronized = **Full**.

Exercise 21 – OSPF Cost Calculation with Modern Links

A router has three interfaces:

- Interface 1: 100 Mbps
- Interface 2: 1 Gbps
- Interface 3: 10 Gbps

Questions

Part A: Calculate OSPF cost for each interface using default Cisco reference bandwidth

Part B: Why do 1 Gbps and 10 Gbps interfaces have the same cost? How to fix this?

Solution

Part A: OSPF Cost Calculation

OSPF Cost Formula:

```
Cost = Reference Bandwidth / Interface Bandwidth
```

Default Cisco Reference Bandwidth = 100 Mbps = 10^8 bps

Interface 1 (100 Mbps):

```
Cost = 100 Mbps / 100 Mbps = 1
```

Interface 2 (1 Gbps = 1000 Mbps):

```
Cost = 100 Mbps / 1000 Mbps = 0.1
OSPF rounds down: Cost = 1 (minimum cost)
```

Interface 3 (10 Gbps = 10,000 Mbps):

```
Cost = 100 Mbps / 10,000 Mbps = 0.01
OSPF rounds down: Cost = 1 (minimum cost)
```

Answer:

- 100 Mbps interface: **Cost = 1**
- 1 Gbps interface: **Cost = 1**
- 10 Gbps interface: **Cost = 1**

Part B: Problem and Solution

Problem Explanation:

- The default reference bandwidth (100 Mbps) was designed when 100 Mbps was the fastest common link
- For links faster than the reference bandwidth, the cost calculation gives values < 1
- OSPF has a **minimum cost of 1**, so all fast links get the same cost
- This means OSPF cannot differentiate between 1 Gbps and 10 Gbps links

Solution: Increase the Reference Bandwidth

Configuration (Cisco):

```
router ospf 1
  auto-cost reference-bandwidth 100000
```

This sets reference bandwidth to 100,000 Mbps (100 Gbps)

New Costs:

- 100 Mbps: 100,000 / 100 = **1,000**
- 1 Gbps: 100,000 / 1,000 = **100**
- 10 Gbps: 100,000 / 10,000 = **10**

Result: Now OSPF correctly prefers the 10 Gbps path over 1 Gbps

Important: This change must be made **on all routers** in the OSPF domain for consistent routing.

Exercise 22 – DR/BDR Election

On an Ethernet LAN:

- R1 priority = 1
- R2 priority = 100
- R3 priority = 50

Questions

1. Who is DR?
2. Who is BDR?
3. What happens if DR fails?

Answer

Priorities:

- R2 = 100
- R3 = 50
- R1 = 1

1. DR = **R2** (highest priority).
2. BDR = **R3** (second highest).
3. If DR fails, **BDR becomes DR**, then a new BDR is elected.

Chapter 6 – IPSec, SSL/TLS, VPN

Exercise 23 – Transport vs Tunnel Mode (Packet Structure)

Original IP packet size = 1500 bytes

ESP adds:

- ESP header = 8 bytes
- ESP trailer + auth = 22 bytes

Questions Calculate new packet size for transport and tunnel modes

Solution

Transport Mode:

New size = 1500 + 30 = **1530 bytes**

Tunnel Mode:

Original packet encrypted + new IP header (20 bytes)

Total = 1500 + 30 + 20 = **1550 bytes**

Answer:

- Transport = **1530 bytes**
- Tunnel = **1550 bytes**

Exercise 24 – IPSec Mode Selection

Choose transport or tunnel mode and justify:

1. Host-to-host secure communication
2. Site-to-site VPN
3. Remote worker VPN

Answer

1. Host-to-host protection → **Transport mode** (protects payload, original IP header stays).
2. Site-to-site VPN → **Tunnel mode** (protects entire original packet + adds new IP header).
3. Remote worker VPN → typically **Tunnel mode** (remote-to-gateway is classic VPN).

Exercise 25 – AH vs ESP

Fill the table:

Feature	AH	ESP
Authentication		
Confidentiality		
IP header protected		
Used in VPNs		

Answer

Feature	AH	ESP
Authentication	✅ yes	✅ yes
Confidentiality	❌ no	✅ yes (encryption)
IP header protected	partially (some header fields included)	❌ no
Used in VPNs	less common	very common

Exercise 26 – IPSec Modes for Site-to-Site VPN

An organization needs to secure traffic between two branch office routers over the public Internet.

Questions

Part A: Should they use Transport Mode or Tunnel Mode?

Part B: Which protocol (AH or ESP) for confidentiality of the original IP header?

Solution

Part A: Transport Mode or Tunnel Mode?

Answer: Tunnel Mode

Reasoning:

Transport Mode:

- Encrypts only the **payload** of the IP packet
- Original IP header remains **unencrypted**
- Used for **end-to-end** communication between hosts
- **Problem:** Exposes source/destination IP addresses

Tunnel Mode:

- Encrypts the **entire original IP packet** (header + payload)
- Encapsulates it in a **new IP packet**
- New outer header has the **router IP addresses**
- Original IP addresses are **completely hidden**

Why Tunnel Mode for Router-to-Router:

1. **Site-to-Site VPN:** The routers create a secure tunnel for all traffic between sites
2. **Topology Hiding:** Internal IP addresses are not exposed to the Internet
3. **Gateway-to-Gateway:** Routers act as security gateways for entire networks
4. **Transparent to Hosts:** Internal hosts don't need IPsec configuration

Example:

```
Branch A (10.1.0.0/24) ↔ Router A (203.0.1.1) ←[IPsec Tunnel]→ Router B (203.0.2.1) ↔ Branch B (10.2

Original Packet: [IP: 10.1.0.5→10.2.0.8 | Data]
Tunnel Mode:     [IP: 203.0.1.1→203.0.2.1 | ESP | Original Packet Encrypted]
```

Part B: AH or ESP for Confidentiality?

Answer: ESP (Encapsulating Security Payload) in Tunnel Mode

Protocol Comparison:

AH (Authentication Header):

- Provides: **Authentication** and **Integrity**
- Does NOT provide: **Encryption/Confidentiality**
- Even in Tunnel Mode, the **original packet** is **not encrypted** (only authenticated)
- Cannot protect the original IP header's confidentiality

ESP (Encapsulating Security Payload):

- Provides: **Confidentiality**, **Authentication**, and **Integrity**
- **Encrypts** the payload (and in Tunnel Mode, the entire original packet)
- In Tunnel Mode: Original IP header is **encrypted** within the ESP payload

Configuration:

```
IPsec: Tunnel Mode + ESP
```

What Gets Protected:

- **Encrypted:** Original IP header, TCP/UDP header, Application data
- **Authenticated:** ESP header, encrypted payload, ESP trailer
- **Visible (outer header):** Source/Dest router IPs, ESP header

Why ESP is Required:

- AH **cannot encrypt**, only authenticate
- To hide the original IP header, **encryption is mandatory**
- ESP is the only IPsec protocol that provides encryption

Complete Answer:

- **Mode:** Tunnel Mode
- **Protocol:** ESP
- **Result:** Original IP header is encrypted and hidden within the ESP payload

---

Exercise 27 – IKE Role

Explain:

1. Why IPSec needs IKE
2. What happens when sequence number reaches maximum
3. How replay attacks are prevented

Answer

1. IPSec needs IKE to create **Security Associations (SAs)** and keys in SADB.
2. When sequence number reaches max → old SA removed + **new SA established**.
3. Replay protection uses **sequence numbers + sliding window** (default 64).

Chapter 7 – HTTP/3 & QUIC

Exercise 28 – Connection Establishment Time (RTT Math)

RTT = 50 ms

Protocol	RTTs needed
TCP + TLS	3 RTT
QUIC (first time)	1 RTT
QUIC (0-RTT)	0 RTT

Question: Calculate connection time for each protocol.

Solution

- TCP + TLS = 3 × 50 = **150 ms**
- QUIC = **50 ms**
- QUIC 0-RTT = **0 ms**

Answer:

➡️ QUIC saves **100–150 ms**

Exercise 29 – Multi-object Page Load

A page needs 12 objects (same server), each object is small (ignore transmission time). RTT = 50 ms.

Assume:

- HTTP/1.1 over TCP+TLS: setup = 4 RTT to first byte, then each extra object costs 1 RTT
- HTTP/2 over TCP+TLS: setup = 4 RTT, then all objects in 1 RTT

- HTTP/3 over QUIC: setup = 2 RTT, then all objects in 1 RTT (multiplexed streams)

Question: Compute total time for each protocol.

Solution

A) HTTP/1.1:

- Setup: 4 RTT
- Objects: 12 RTT
- Total RTT = 16 RTT
- Time = 16 × 50 = **800 ms**

B) HTTP/2:

- Setup: 4 RTT
- All objects: 1 RTT
- Total RTT = 5 RTT
- Time = 5 × 50 = **250 ms**

C) HTTP/3:

- Setup: 2 RTT
- All objects: 1 RTT
- Total RTT = 3 RTT
- Time = 3 × 50 = **150 ms**

✅ **Answer:** HTTP/1.1 800 ms, HTTP/2 250 ms, HTTP/3 150 ms

Exercise 30 – Protocol Comparison

Explain why HTTP/3 is faster than HTTP/2 even with packet loss.

Mention:

- Transport protocol
- Head-of-line blocking
- Multiplexing
- Connection establishment

Answer

Why HTTP/3 Faster than HTTP/2 (with loss):

- HTTP/2 runs over **TCP**, so one lost packet can block others (**transport HOL blocking**).
- HTTP/3 runs over **QUIC (UDP-based)** with **independent streams**, so loss in one stream doesn't stall others.

Exercise 31 – QUIC 0-RTT

Questions

1. What information is reused?
2. What security risk exists with 0-RTT?
3. Why is UDP required?

Answer

1. Client reuses **session ticket / saved transport parameters** from previous connection.
2. Security risk: **replay attacks** are the classic concern with 0-RTT early data.
3. **UDP** is used because QUIC implements its own reliability + congestion + encryption on top, without TCP handshake overhead.

Exercise 32 – Network Switch Scenario

A mobile user switches from Wi-Fi to 4G during a video call.

Questions

1. What happens in TCP?
2. What happens in QUIC?
3. Why is QUIC better here?

Answer

1. **TCP:** changing IP usually breaks connection → needs new handshake.
2. **QUIC:** supports **connection migration** via Connection IDs → session continues.
3. QUIC is better because it maintains **session continuity** and reduces latency.

Exercise 33 – HTTP/3 vs HTTP/2 Packet Loss Handling

Explain how HTTP/3 over QUIC handles packet loss in multiplexed streams compared to HTTP/2 over TCP.

Scenario: A client is downloading 3 resources simultaneously (image, CSS, JavaScript) and one packet from the CSS file is lost.

Solution

HTTP/2 over TCP Behavior:

Problem: Head-of-Line (HOL) Blocking

1. **Single TCP Connection:** All three resources share one TCP connection
2. **Packet Loss Detected:** TCP's congestion control detects the lost CSS packet (via missing ACK or duplicate ACKs)
3. **TCP Behavior:**
    - TCP's **receive buffer** has a gap (missing segment)

- TCP **cannot deliver any data** beyond the gap to the application layer
- This is due to TCP's **in-order delivery guarantee**

4. **HTTP/2 Blocked:**
    - Even though image and JavaScript packets arrive successfully
    - HTTP/2 **cannot access them** until TCP fills the gap
    - **ALL streams are blocked** waiting for one lost packet
5. **Retransmission:**
    - Lost CSS packet is retransmitted
    - Only after it arrives can TCP release all buffered data
6. **Result:** One lost packet blocks **all multiplexed streams**

Diagram:

```
Timeline:
t0: [Image OK] [CSS LOST] [JS OK] → Arrive at receiver
t1: TCP buffer: [Image] [GAP] [JS] → Gap prevents delivery
t2: Application layer receives: NOTHING (blocked by TCP)
t3: CSS retransmitted arrives, fills gap
t4: Application layer receives: [Image][CSS][JS] all at once
```

HTTP/3 over QUIC Behavior:

Solution: Stream-Level Independence

1. **QUIC Stream Multiplexing:** Each resource is a separate QUIC stream with independent sequence numbers
2. **Packet Loss Detected:** Lost CSS packet detected via QUIC's acknowledgment mechanism
3. **QUIC Behavior:**
    - Only the **CSS stream** is affected
    - QUIC's receive buffer has gaps **per stream**, not globally
    - Image and JavaScript streams are **independent**
4. **HTTP/3 Continues:**
    - Image stream: **Delivered immediately** to application
    - JavaScript stream: **Delivered immediately** to application
    - CSS stream: **Blocked** waiting for retransmission
5. **Retransmission:**
    - Only the lost CSS packet is retransmitted
    - Uses **UDP** (no connection state to maintain)
6. **Result:** Loss affects **only the CSS stream**; other streams proceed normally

---

Diagram:

```
Timeline:
t0: [Image OK] [CSS LOST] [JS OK] → Arrive at receiver
t1: QUIC streams:
    Stream 1 (Image): [Complete] → Delivered to app
    Stream 2 (CSS):   [GAP]     → Blocked
    Stream 3 (JS):    [Complete] → Delivered to app
t2: Application renders: Image + JS (partial page load)
t3: CSS retransmitted arrives
t4: Application receives: CSS → Full page rendered
```

Key Differences Summary:

Aspect	HTTP/2 over TCP	HTTP/3 over QUIC
Transport	Single TCP connection	Multiple QUIC streams over UDP
Loss Detection	TCP sequence numbers	QUIC packet numbers (per stream)
Ordering	Global in-order delivery	Per-stream in-order delivery
Blocking	All streams blocked	Only affected stream blocked
Retransmission	TCP-level (connection-wide)	QUIC-level (stream-specific)
User Experience	Entire page stalls	Partial page loads progressively

Technical Explanation:

Why TCP Blocks:

- TCP guarantees **byte-stream in-order delivery**
- Sequence numbers are **connection-wide**, not per-stream
- A gap at byte position 1000 prevents delivery of byte 1001+, even if they belong to different HTTP/2 streams

Why QUIC Doesn't Block:

- QUIC has **stream-level sequence spaces**
- Stream 1's packet 5 is independent of Stream 2's packet 5
- Loss in one stream doesn't create gaps in other streams
- QUIC implements **per-stream receive buffers**

Additional Benefits:

- **0-RTT Connection Resumption:** QUIC can resume connections without handshake
- **Connection Migration:** Can survive IP address changes (mobile networks)
- **Improved Congestion Control:** Per-stream loss recovery

Conclusion: HTTP/3's use of QUIC eliminates the head-of-line blocking problem that plagued HTTP/2, making it significantly more performant on lossy networks (mobile, Wi-Fi).

Exercise 34 – BDP + Window-Limited Transfer

Bandwidth = 20 Mbps

RTT = 100 ms

Receiver window (or congestion window cap) = 250 KB

File size = 2 MB

Assume sender can send at most 250 KB per RTT (window-limited).

Question: How many RTTs to deliver the full file? Ignore handshake.

Solution

1. Convert file size: 2 MB = 2048 KB
2. Per RTT deliverable = 250 KB
3. RTTs needed = ceil(2048 / 250)

Compute:

- 250×8 = 2000 KB
- Remaining = 48 KB
- So RTTs = 8 + 1 = **9 RTT**

Time = 9 × 100 ms = **900 ms**

✅ **Answer:** 9 RTT = 900 ms

End of Exercise Collection