**1. Network Layers and Layering**

**Explanation**: Networking uses layered models (e.g., OSI or Internet/TCP/IP) to divide tasks. Each layer provides services to the one above and uses the one below. Key layers: Application (e.g., HTTP), Transport (e.g., TCP/UDP), Network (e.g., IP), Link (e.g., Ethernet), Physical. Layering allows modularity; encapsulation wraps data with headers (e.g., transport adds H\_t to app message M, becoming H\_t|M).

**Short Example**: Sending an email: Application layer creates message M. Transport encapsulates as H\_t|M (adds ports). Network adds H\_n|H\_t|M (adds IP addresses). Link adds H\_l|H\_n|H\_t|M (adds MAC). Physical transmits bits.

ASCII Diagram (Encapsulation):

Source: App -> [M]

Transport: [H\_t | M]

Network: [H\_n | H\_t | M]

Link: [H\_l | H\_n | H\_t | M]

Physical: Bits over wire

Destination: Reverse process (decapsulation)

**2. Access vs. Core Networks**

**Explanation**: Access networks connect end-users/devices to the edge (e.g., WiFi, cellular, Ethernet). Core networks (routers/switches) handle packet forwarding/routing across the backbone. Access is last-mile; core interconnects providers.

**Short Example**: Purdue WiFi access points are access (user connection); campus routers are core (route to internet).

**3. Delays in Networks**

**Explanation**: Four types: Transmission (time to push bits onto link = packet size / rate), Propagation (time for signal to travel = distance / speed), Processing (router handling), Queuing (wait in buffers). RTT = 2 \* (propagation + processing + transmission/2). Total delay = sum of all.

**Short Example**: 500-byte packet, 2 Mbps link, 1000m distance, speed 10^6 m/s. Transmission delay = (500*8)/2e6 = 0.002s. Propagation = 1000/1e6 = 0.001s. RTT ≈ 2*0.001 = 0.002s (ignore others if negligible).

**4. HTTP and Web Browsing**

**Explanation**: HTTP is application-layer protocol for web. Requests (e.g., GET) and responses (e.g., 200 OK). Non-persistent: New TCP per object (high delay). Persistent: Reuse TCP (with/without pipelining). Cookies track sessions; third-party for ads.

**Short Example**: Fetch base HTML + 10 objects, RTT=1s, rate=100Mbps, files=1KB. Non-persistent delay: 2RTT (base) + 10*2RTT = 22s. Persistent without pipelining: 2RTT + 10*RTT = 12s.

**5. DNS (Domain Name System)**

**Explanation**: Translates names (e.g., [www.example.com](http://www.example.com/)) to IPs. Recursive: Local server queries root -> TLD -> authoritative. Iterative: Client handles chain. Uses UDP. nslookup queries types (e.g., MX for mail).

**Short Example**: Resolve [www.lego.com](http://www.lego.com/) (empty cache): Local -> Root (get .com TLD) -> .com (get lego.com NS) -> Authoritative (get IP). Command: nslookup > set type=A > [www.lego.com](http://www.lego.com/).

**6. UDP and Checksum**

**Explanation**: UDP is connectionless transport; adds ports, length, checksum (no reliability). Checksum: 1's complement sum of 16-bit words (including pseudo-header); receiver verifies sum + checksum = all 1s. Detects errors but may miss some (e.g., compensating flips).

**Short Example**: Bytes 0x5C65. Sum=0x5C65, checksum=~0x5C65=0xA39A. Error undetected if flips cancel (e.g., +1 in one byte, -1 in another).

**7. Reliable Data Transfer (RDT)**

**Explanation**: Builds reliability over unreliable channels. RDT2.0: ACK/NAK for errors. RDT2.2: ACKs with seq# (no NAK). RDT3.0: Timers for loss. Handles errors, losses via retransmit on timeout/duplicate ACK.

**Short Example**: RDT3.0 sending 3 packets, 2nd lost.

ASCII Diagram (Packet Loss and Retransmission):

Sender Channel Receiver

Pkt1 ------------> <------------ ACK1

Pkt2 ------------> (lost)

(timeout)

Pkt2 ------------> <------------ ACK2

Pkt3 ------------> <------------ ACK3

**8. TCP ACKs and Packet Sending**

**Explanation**: TCP is reliable transport: Connection-oriented, seq#, ACKs (cumulative, next expected byte), timeouts/retransmits on loss. Flow/congestion control. ACK loss: Sender timeouts/retransmits. Packet loss: Duplicate ACKs trigger fast retransmit.

**Short Example**: Sender sends segments 1-3, 2 lost. Receiver ACKs 1 (cumul), then duplicate ACK1 on 3. Sender retransmits 2 on timeout/3 dup ACKs.

ASCII Diagram (ACK Lost):

Sender Channel Receiver

Pkt1 ------------> <------------ ACK1 (lost)

(timeout)

Pkt1 ------------> <------------ ACK1

**9. Selective Repeat and Go-Back-N**

**Explanation**: Pipelined RDT. Go-Back-N: Sender window, retransmit all on error/loss. Selective Repeat: Individual ACKs, retransmit only lost (window size <= seq#/2 to avoid ambiguity).

**Short Example**: Window=4, sender [501-504]. Possible receiver windows: [501-504] (in sync), [502-505] (advanced by 1), etc. On loss of 502: Retransmit only 502.

ASCII Diagram (Packet Repeated/Lost in Selective Repeat):

Sender Window [1,2,3,4]

Send 1,2,3,4

1 OK, 2 lost, 3 OK, 4 OK

Receiver: ACK1, NAK2 (or dup ACK1), ACK3, ACK4

Sender: Retransmit 2

**10. Round Trip Time (RTT) and Timeout Calculations**

**Explanation**: RTT measured via timestamps. Timeout = EstimatedRTT + 4\*DevRTT (adaptive). Used for retransmissions. Total time for N packets: Depends on protocol (e.g., with routers, bottleneck link affects).

**Short Example**: 3 packets, links A-B 2Mbps, B-C 2Mbps, C-D 1Mbps, prop=1ms each. First packet to D: Trans A-B (0.002s) + prop (0.001) + trans B-C (0.002) + prop (0.001) + trans C-D (0.004) + prop (0.001) = 0.011s.

 **Question**: Explain encapsulation in network layers with a diagram. **Answer**: Data wrapped with headers per layer. ASCII Diagram:

App: [M]

Trans: [H\_t | M]

Net: [H\_n | H\_t | M]

Link: [H\_l | H\_n | H\_t | M]

 **Question**: What are access networks? Give examples and distinguish from core.

**Answer**: Connect users (e.g., WiFi, cellular, Ethernet). Core routes (routers forward packets).

 **Question**: Calculate transmission and propagation delay for 1000-byte packet on 1Mbps link, 2000km, speed 2x10^8 m/s.

**Answer**: Trans = (1000\*8)/1e6 = 0.008s. Prop = 2000000/2e8 = 0.01s.

 **Question**: For HTTP/1.1 non-persistent, delay for base + 5 objects, RTT=0.5s.

**Answer**: 2RTT (base) + 5\*2RTT = 6s (ignore file trans if small).

 **Question**: Describe DNS recursive query for [www.example.com](http://www.example.com/) (empty cache).

**Answer**: Local -> Root -> .com TLD -> example.com NS -> IP.

 **Question**: Compute UDP checksum for bytes 0xABCD.

**Answer**: Sum=0xABCD, checksum=~0xABCD=0x5432.

 **Question**: Why use NAK in RDT2.0 but not RDT2.2?

**Answer**: RDT2.0 needs explicit error signal; RDT2.2 uses seq# and dup ACKs.

 **Question**: Diagram RDT3.0 with 3 packets, 2nd lost, timeout retransmit.

**Answer**:

Sender Channel Receiver

Pkt1 --------> <-------- ACK1

Pkt2 --------> (lost)

(timeout)

Pkt2 --------> <-------- ACK2

Pkt3 --------> <-------- ACK3

 **Question**: In TCP, what happens on packet loss? Diagram with ACKs.

**Answer**: Dup ACKs or timeout triggers retransmit.

Sender: Seg1 -> OK, ACK2

Seg2 -> lost

Seg3 -> Dup ACK2

(3 dup ACKs) Retrans Seg2

 **Question**: Calculate time for first packet over 3 links: A-B 1Mbps (prop 1ms), B-C 2Mbps (1ms), C-D 0.5Mbps (1ms). Packet=500B.

**Answer**: Trans A-B=(500\*8)/1e6=0.004s +1ms; B-C=0.002s+1ms; C-D=0.008s+1ms. Total≈0.018s.

 **Question**: Diagram ACK lost in RDT3.0 for packet 1.

**Answer**:

Sender Channel Receiver

Pkt1 --------> <-------- ACK1 (lost)

(timeout)

Pkt1 --------> <-------- ACK1

 **Question**: In Selective Repeat (window=3), list possible receiver windows if sender is [10,11,12].

**Answer**: [10,11,12], [11,12,13], [12,13,14].

 **Question**: Explain third-party cookies for ads.

**Answer**: Site A sets cookie via ad network; network reads on site B, shows targeted ad.

 **Question**: Does RDT3.0 have a fatal flaw? Give example if yes.

**Answer**: No, but inefficient; e.g., long delay without loss causes unnecessary timeout/retransmit.

 **Question**: Compute RTT-based delay for persistent HTTP with pipelining: Base + 4 objects, RTT=1s.

**Answer**: 2RTT (connect + all requests/responses in pipeline) ≈2s.