**CS422 Computer Networks (Fall 2025): Homework 2**

**Juan Florez 039614388**

1. (5 points) Web browsing and HTTP
2. (2 points) Alice uses Chrome on her laptop to visit www.amazon.com and finds that she is already logged in. How does this happen?

Chrome shows Alice already logged in because the browser stored a session cookie, when Alice revisits, the browser automatically sends that cookie with the HTTP(S) request; the server recognizes the cookie and restores her authenticated session.

1. (3 points) Alice used Chrome on her laptop to visit www.lego.com yesterday. Later, she used Chrome on her laptop to visit a new website, say A (which has been never visited before), she saw an LEGO advertisement in the webpage displayed in Chrome. After a while, she used Chrome on her laptop to visit another new website, say B (which has been never visited before), she saw the same LEGO advertisement in the webpage displayed in Chrome. How does it happen?

Alice sees LEGO ads on unrelated sites because of third-party tracking/advertising. LEGO, placed a third party cookie or used a tracking pixel on lego.com. Later, other sites include the same ad network’s script that script reads the tracker and serves LEGO ads tailored to Alice. Ad networks and cross-site trackers enable this targeted ad delivery.

1. (8 points) Domain Name System (DNS) and nslookup
2. (4 points) Suppose Alice’s laptop does NOT know the IP address of “www.lego.com” at the start. And the cache in the local DNS server is empty. Assume the local DNS server uses recursive inquiry. Please explain the procedure of DNS inquiry with main steps.

First Alice’s laptop asks its local resolver for lego.com then the local resolver, having empty cache, starts a recursive query:

Query the root server for the IP lego.com, and root returns the authoritative servers for .com., also query a .com TLD server; it returns the authoritative name servers for lego.com.,finally query one of lego.com’s authoritative servers; it returns the A record for lego.com.

To finish the local resolver returns the answer to Alice’s laptop and caches the result for the TTL.

1. (2 points) Suppose Alice wants to get the IP address of the mail server for gmail.com. How could she use nslookup to achieve it? Please list the commands (choose one mail server is enough)

nslookup

set type=mx

gmail.com

1. (2 points) Explain why DNS messages are not human-readable, and provide one example of a human-readable application-layer protocol.

DNS messages are binary-encoded and optimized for size, so they are not human readable. An example of a human-readable application layer protocol is **HTTP** (textual requests and responses like GET /index.html HTTP/1.1).

1. (6 points) UDP uses checksum to detect error.
2. (2 points) Suppose you have the following 2 bytes: 01011100 and 01100101. What is the checksum of these 2 bytes?

Bytes: 01011100 (0x5C) and 01100101 (0x65).  
16-bit word = 0x5C65. One’s-complement sum of single word is itself;

checksum = one’s-complement of sum = ~0x5C65 = 0xA39A. So checksum = **0xA39A** (binary 1010001110011010).

1. (2 points) Suppose you have the following 2 bytes: 11011010 and 01100101. What is the checksum of these 2 bytes?

Bytes: 11011010 (0xDA) and 01100101 (0x65).  
16-bit word = 0xDA65. Checksum = ~0xDA65 = 0x259A.

So checksum = **0x259A** (binary 0010010110011010).

1. (2 points) Given the bytes in part (a), given an example where the bit errors occur and yet the checksum doesn’t change.

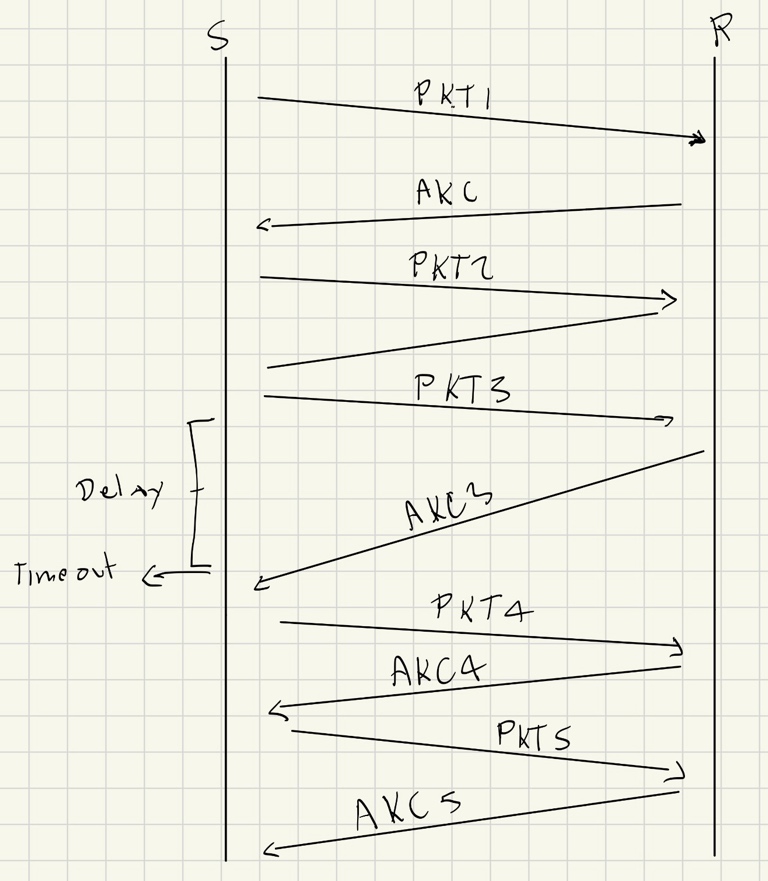
Example of undetected bit errors for a: Suppose both bytes flip the same bit position in opposite directions so the 16-bit sum stays the same. For instance, flip bit 4 of the first byte from 1 to 0 and flip bit 4 of the second byte from 0 to 1 such that the 16-bit word changes by +16 and -16 which cancel out. The checksum computed over the corrupted word equals the original checksum, error undetected.

1. (6 points) In RDT2.0, ACK and NAK both are used; In RDT2.2, only ACK is used. Can we get rid ofNAK in RDT2.0? If no, please explain why. What will RDT2.2 do if the packet has error?

In RDT2.0 NAKs are necessary because the sender cannot reliably distinguish a corrupted ACK from a corrupted data packet or lost ACK; NAK explicitly signals a detected corruption to trigger immediate retransmission. If you remove NAKs in RDT2.0 and rely only on ACKs, the sender may not learn quickly that the receiver observed corruption, which can stall progress.

RDT2.2 avoids explicit NAKs by using **sequence numbers** and ACK/duplicate-ACK semantics: the receiver sends an ACK with the expected sequence number (or repeats an ACK) to indicate the last successfully received in-order packet. If a packet arrives with errors, the receiver re-ACKs the last good packet, and the sender retransmits after timeout or upon duplicate ACKs.

1. (10 points) Please follow the examples given in the class and plot the diagram of RDT3.0 in action. The sender uses RDT3.0 to send 5 packets to the receiver. The third packet experiences a super long delay but its ack is received just after the timeout. Afterwards, there are no errors, losses, or long delays. That is, all other packets and their acks are received correctly without triggering any timeout.
2. (3 points) Please use a diagram to plot how these 5 packets are sent out. How many packets in total are sent by the sender?



1. (3 points) By default, RDT3.0 ignores duplicate acks (check FSM). Suppose instead that RDT3.0 behaves like RDT2.2 (the sender resends a packet upon receiving a duplicate ack). Please use a diagram to plot how these 5 packets are sent out. How many packets in total are sent by the sender?

7 packets.

1. (4 points) Does RDT3.0 have a fatal flaw? If yes, please give an example. If no, please explain why not.

**No fatal flaw** for stop-and-wait RDT3.0 in theory: it guarantees reliable, in-order delivery despite losses and corruptions by using timeouts and retransmissions and sequence numbers (0/1). The protocol can be inefficient but it is correct: timeouts + retransmissions ensure progress. If timers are set incorrectly or sequence numbers insufficient for higher order pipelining, practical issues appear, but conceptually RDT3.0 is not fatally flawed.

1. (5 points) In Selective Repeat, suppose the window size is 4.
2. (3 points) List all the possible ranges for the receiver’s window when the sender’s window is currently [501, 502, 503, 504].

[501, 502, 503, 504]

[502, 503, 504, 505]

[503, 504, 505, 506]

[504, 505, 506, 507]

1. (2 points) List all the possible ranges for the sender’s window when the receiver’s window is currently [501, 502, 503, 504].

[498, 499, 500, 501]

[499, 500, 501, 502]

[500, 501, 502, 503]

[501, 502, 503, 504]