Problem 1

Consider a DASH system for which there are N video versions (at N different rates and qualities) and N audio versions (at N different rates and qualities). Suppose we want to allow the player to choose at any time any of the N video versions and any of the N audio versions.

(a) If we create files so that the audio is mixed in with the video, so server sends only one media stream at given time, how many files will the server need to store (Each a different URL)?

(b) If the server instead sends the audio and video streams separately and has the client synchronize the streams, how many files will the server need to store?

Write your solution to Problem 1 in this box

a. The server will need to store $N \cdot N$ files to account for all possible versions a user can request, resulting in N^2 different possible URLs.

b. If ~~instead~~ instead, the client synchronizes the audio and ~~video~~ video themselves, the server only needs to store $N + N = 2N$ different files.

Problem 2

Suppose you have a new computer just set up. `dig` is one of the most useful DNS lookup tool. You can check out the manual of `dig` at `http://linux.die.net/man/1/dig`. A typical invocation of `dig` looks like `dig @server name type`

Suppose that on April 19, 2018 at 15:35:21, you have issued "`dig google.com A`" to get an IPv4 address for `google.com` domain from your caching resolver and got the following result:

```
; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer.
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17779
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             239     IN      A       172.217.4.142

;; AUTHORITY SECTION:
google.com.             55414   IN      NS      ns4.google.com.
google.com.             55414   IN      NS      ns2.google.com.
google.com.             55414   IN      NS      ns1.google.com.
google.com.             55414   IN      NS      ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         145521  IN      A       216.239.32.10
ns2.google.com.         215983  IN      A       216.239.34.10
ns3.google.com.         215983  IN      A       216.239.36.10
ns4.google.com.         215983  IN      A       216.239.38.10

;; Query time: 81 msec
;; SERVER: 128.97.128.1#53(128.97.128.1)
;; WHEN: Wed Apr 19 15:35:21 2017
;; MSG SIZE rcvd: 180
```

(a) What is the discovered IPv4 address of `google.com` domain?

(b) If you issue the same command 1 minute later, how would "ANSWER SECTION" look like?

(c) When would be the earliest (absolute) time the caching resolver would contact one of the `google.com` name servers again?

(d) If the client keeps issuing `dig google.com A` every second, when would be the earliest (absolute) time the caching resolver would contact one of the `.com` name servers?

Problem 2 continued on next page...

Write your solution to Problem 2 in this box

a. The discovered IPv4 address is 172.217.4.142.

b. The answer section would be as follows:

google.com 179 IN A 172.217.4.142

c. The earliest time the google.com name server would be contacted is in 239 sec.

d. The earliest time a .com name server ~~would~~ would be contacted is in 55414 sec.

Problem 3

The sender side of *rdt3.0* simply ignores (that is, takes no action on) all received packets that are either in error or have the wrong value in the acknum field of an acknowledged packet. Suppose that in such circumstances, *rdt3.0* were simply to retransmit the current data packet. Would the protocol still work? (Hint: Consider what would happen if there were only bit errors; there are no packet losses but premature timeouts can occur. Consider how many times the *nth* packet is sent, in the limit as n approaches infinity).

Write your solution to Problem 3 in this box

The protocol would still work. rdt3.0 retransmits the packet in the case where the packet appears to be lost; if this new condition were to be added, rdt3.0 would now expand to retransmit the packet in not only the case where the packet is lost but also in the case where the ~~ACK~~ packet received (the ACK) has been corrupted. The receiver doesn't distinguish between ~~the~~ a packet being lost or corrupted so retransmitting the packet won't cause any changes to the receiver side. However, ~~in this case,~~ in the case where premature timeouts can occur and only bit error had occured, extra packets would result in extra ACKs being sent and received, which in turn cause another extra packet to be sent since the packets are extraneous and thus do not have the correct acknum value. As a result, the number of times the packet is sent increases as n approaches ∞.

Problem 4

Consider a reliable data transfer protocol that uses only negative acknowledgments. Suppose the sender sends data only infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why? Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

Write your solution to Problem 4 in this box

A NAK-only protocol would not be ideal since, say, you lose a packet n. The protocol would only detect the loss of packet n after it has received packets n-1 and then n+1. The time between sending n and receiving n+1 would be a ~~long~~ long delay if data is transmitted infrequently.
In the second case, a NAK-only protocol would be more useful since an error would be detected more quickly & thus, quicken the receiving process. If errors are few, NAKs are sent infrequently. In a protocol that uses ACKs, if errors are infrequent, many ACKs would be sent, an increase in network traffic.

Problem 5

Consider the GBN protocol with a sender window size of 4 and a sequence number range of 1,024. Suppose that at time t, the next in-order packet that the receiver is expecting has a sequence number of k. Assume that the medium does not reorder messages. Answer the following questions:

(a) What are the possible sets of sequence numbers inside the senders window at time t? Justify your answer.

(b) What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.

Write your solution to Problem 5 in this box

a. The receiver expects the next packet to be k so the last four ACKs have sequence number k-1, k-2, k-3, k-4. If all of these have been received, the window of the sender would be ~~[k, k+3]~~ (k, k+1, k+2, k+3). If none have been received, the window would be (k-4, k-3, k-2, k-1). Thus, all possible sets are: (k-4, k-3, k-2, k-1), (k-3, k-2, k-1, k), (k-2, k-1, k, k+1), (k-1, k, k+1, k+2), (k, k+1, k+2, k+3).

b. The receiver is currently waiting for k, so the packets (k-4, k-3, k-2, k-1) have been sent back to the sender and ACKs for those would currently be ~~pro~~ propagating.