Problem Set 1

15-440/15-640 Distributed Systems Spring 2026

Assigned: Thursday, January 22, 2026

Due: Thursday, January 29, 2026 at 11.59 pm

Submission procedure:

- Create a **.pdf** of your answers and upload to `Gradescope`. All enrolled students should have Gradescope accounts. Ask **well in advance of the deadline** to be added to Gradescope if you don't have access; it is unacceptable to ask to be added immediately before the deadline.
- Here are some tips to make your submission easy to read and to grade. Remember, the easier you make this, the less likely we are to make grading errors. Following these guidelines will help us to focus on the technical content of your answers rather than trying to understand what you have written.
    - Don't handwrite your answers. Use LaTeX or Google Docs or some similar input mechanism. If you use LaTeX, a template can be found on the course web page.
    - Put the answer to each question on a separate page.
    - Carefully tag your pdf pages to each question on `gradescope`. You can use the SHIFT key to select multiple pages and associate them with a single question.
- Assume SI notation
    - 1 KB = 10^3 bytes, 1 MB = 10^6 bytes, 1 GB = 10^9 bytes
    - 1 Kbps = 10^3 bits per second (bps), 1 Mbps = 10^6 bps, 1 Gbps = 10^9 bps
    - but a byte is still 8 bits (not 10 bits) :-)
- **Remember that you have a limit of 2 grace days totaled over all four problem sets.** You can use at most one of those grace days per problem set. Although Gradescope does not track grace days, we will. Exceeding your grace days will result in a zero grade for that problem set.

Question 1 (20 points)

You are a systems engineer at RetroNet, a startup dedicated to bringing online multiplayer features to retro 1990s gaming consoles. Your team is building RPC infrastructure to allow these older 32-bit consoles to communicate with a modern, cloud-based 64-bit backend. The architecture is as follows:

- Client (Console): 32-bit architecture, Little Endian.
- Server (Cloud): 64-bit architecture, Little Endian.
- Network: The Internet (assumed Big Endian / Network Byte Order).

A junior engineer, Alex, has implemented the initial "Save Score" RPC. To save development time, Alex decided to simply cast the struct memory to a buffer and send it over the wire. The C structure used for the RPC request is defined as follows:

```
struct SaveScoreRequest {
    uint8_t  game_id;    // 1 byte
    uint32_t user_id;    // 4 bytes
    long     timestamp;  // Time since epoch
    uint16_t checksum;   // 2 bytes
};
```

The client-side code is as follow:

```
struct SaveScoreRequest req;
// ... fill req with data ...
send(socket, &req, sizeof(req), 0);
```

The server-side code is as follows:

```
struct SaveScoreRequest req;
recv(socket, &req, sizeof(req), 0);
// ... process req …
```

Clearly stating any assumptions you make, answer the following:

A. Draw a picture of the memory layout of the SaveScoreRequest structure on the 32-bit and 64-bit machines. You may assume alignment as you were taught in 15-213/513. Clearly label each element of the struct in the layout. How many bytes of space does this struct require on the two types of machines?

B. Alex tests the code by running the Client and the Server on his own development laptop. It works perfectly. However, when deployed to the real environment (32-bit Console Client and 64-bit Cloud Server), the system breaks. The server receives data, but the timestamp values are nonsensical, and the checksum verification consistently fails.

   In a sentence or two, explain one possible reason why this implementation fails in the production environment.

C. To fix the issues above, Alex decides to implement the SaveScoreRequest struct in a "packed" format where no padding bytes are transmitted, and all integers are normalized to fixed-width types.

   Draw a clearly labeled picture of the resulting packet, and calculate its length. Also calculate the percentage reduction in bytes required to send SaveScoreReport from client to server when using the packed format instead of the original. Interpret "percent reduction" as (old - new)/old expressed as a percentage.

D. The system in C has been running successfully without issues for many weeks. Alex now wishes to modify the SaveScoreRequest structure to send the player's username along with the score. He updates the struct as follows:

```
struct SaveScoreRequest {
    // ... previous fields ...
    char *username;
};
```

   Alex updates the packing code to write the integer fields (id, timestamp) into the buffer, extends the buffer to accommodate the `username` string, and then does `memcpy()` of `strlen(username)` bytes into it.

   The server receives the request, and unpacks the integer fields correctly. But when it attempts to print `username`, the application crashes with a Segmentation Fault (SIGSEGV). What is the likely reason for the crash? Propose a fix for this bug.

Question 2 (15 points)

Maelle just got back from a spring break trip with her friends. She wants to upload the photos she took from her phone to a shared online album hosted on a server. She has *N* photo files to upload, and the size of each file is exactly 10**B* bytes, where *B* is a large constant. The server has negligible processing delay. After receiving the file, it sends an ACK back if and only if a photo has the "ACK-me" bit set in its header. She can use one of two protocols:

- **Cautious:** Send each photo to the server separately, with the "ACK-me" bit set for each photo. She waits for the server's ACK before sending the next. If Maelle doesn't get an ACK after waiting a fixed amount of time, then she will resend the same file in its entirety.
- **Reckless:** Send all photos to the server in one go without any waiting. The "ACK-me" bit is set **only** for the last photo sent.

For questions A through C (but not for D), you may assume that there is zero packet loss, reordering, corruption, etc. In other words, the network is perfect in all respects except latency and bandwidth. Assume ACKs are of negligible size.

A. Suppose the network has a one-way latency of *L* ms, and a bandwidth of *B* bps. For each of the two protocols, how long would it take until Maelle is confident that all her photos have been received?

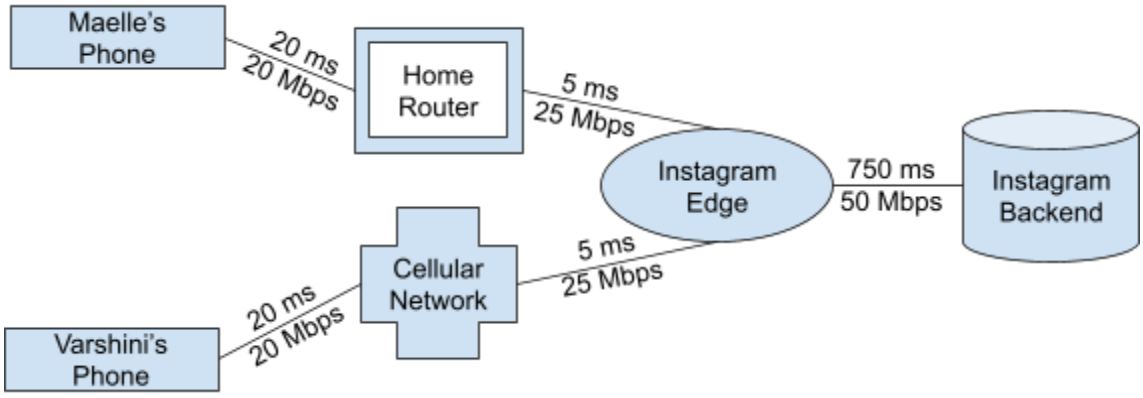
B. Suppose latency is unchanged, but bandwidth improves to 10**B* bps. For each of the two protocols, how long would it now take until Maelle is confident that all her photos have been received?

C. Suppose one-way latency degrades to *5L* ms but bandwidth remains at *B* bps. For each of the two protocols, how long would it now take until Maelle is confident that all her photos have been received?

D. Now assume that the network becomes extremely unreliable, suffering very high packet loss. Using qualitative reasoning, explain which protocol Maelle should use and why.

Question 3 (20 points)

In Sunnyvale, Maelle captures a photo of size *A* on her smartphone and tags Varshini, who is at Yosemite and only reachable via a cellular network. When she taps "Post," the photo is uploaded to the Instagram Edge server, which performs minimal preprocessing (negligible processing time), and then forwards the post to the Instagram Backend. The figure below shows the one-way latency and bandwidth of each network segment that is relevant to this question.



The Instagram Backend server takes 500 ms to process any post, regardless of its size. It cannot begin processing until it has received the entire post. After processing, the Instagram Backend sends a notification of size *B* to Varshini. This travels back to the Instagram Edge, and then through the path shown above to Varshini's phone.

A. Suppose *A* is 50KB and *B* is 10KB. What is the total end-to-end time from when Maelle taps 'Post' until Varshini receives the notification on her phone?

B. Instagram also supports carousels of photos. Suppose that Maelle posts 10 photos, each of size *A* = 70KB, which are sent as a back-to-back stream to the Edge Server. After receiving all the images, the Edge Server passes them all to the Backend Server (again as a back-to-back stream). After processing (taking 500 ms, as before), the Instagram Backend sends a

notification of size *B* = 50KB to Varshini. What is the total end-to-end time from when Maelle taps 'Post' until Varshini receives the notification on her phone?

C. Suppose Instagram upgrades the Instagram Edge to Instagram Backend bandwidth from 50 Mbps to 1 Gbps (latency unchanged). What is the new end-to-end time from when Maelle taps 'Post' to when Varshini receives the notification on her phone for part (B)? Is this network upgrade helpful?

D. Now, suppose that Instagram upgrades the Instagram Edge to Instagram Backend latency from 750ms to 250ms (bandwidth unchanged, remaining at 50 Mbps). What is the new end-to-end time from when Maelle taps 'Post' to when Varshini receives the notification on her phone for part (B)? Is this upgrade helpful?

Question 4 (20 points)

Cha Vitality Hospital (CVH) is designing a new prescription transmission system for securely and reliably passing drug prescriptions from doctors' offices to patients. The requirement for this system is that it preserves *exactly-once semantics:* too many copies of the same prescription could lead to drug trafficking; but not sending a prescription could lower the quality of patient care.

Three different vendors are offering implementations for this prescription transmission system. CVH is conducting a pilot study in which each system is tested on a small number of randomly-chosen patients. The CVH staff carefully and precisely record the experience of each patient. The experiences of three patients are shown below. Viewing the doctor and the patient as the endpoints, explain whether **exactly-once** semantics is achieved. If it is not achieved, explain how you would modify the system to achieve it. Explain your reasoning using the end-to-end principle.

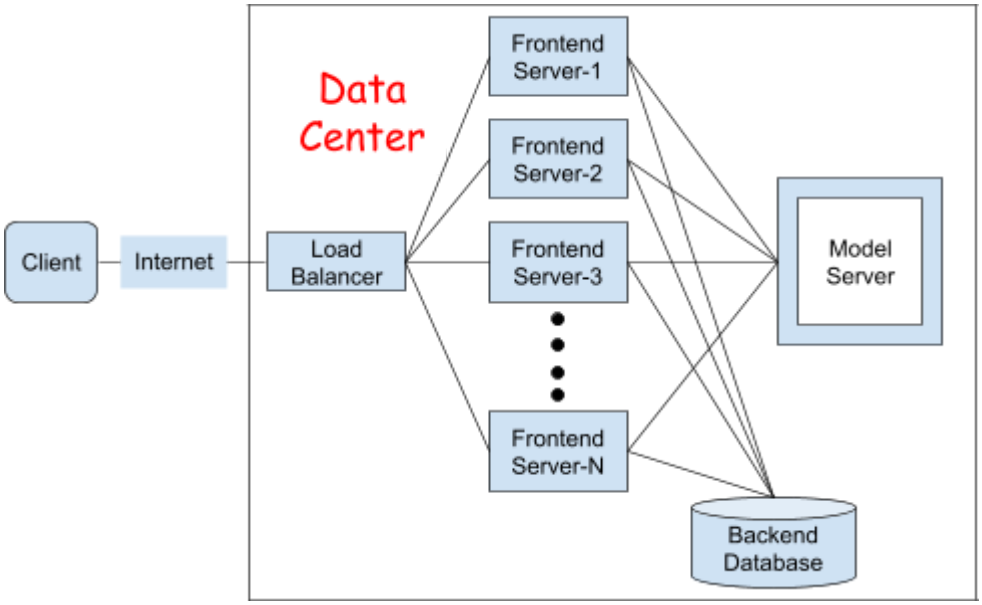
A. Patient Mike meets with Dr. Cindy from CVH and is prescribed a 30-count bottle of Xanax to help him deal with stress. Dr. Cindy sends the prescription via secured USPS mail to Mike's nearest pharmacy. Unfortunately a storm hits and the pharmacy is informed that 50% of their incoming mail has been lost. In this system, the pharmacy is responsible for dealing with unreceived prescriptions. It requests CVH to resend all the sent prescriptions from the relevant day. Dr. Cindy receives this request and resends the prescription again. A week later, the USPS retrieves a missing bag of mail that had fallen off a truck, and it correctly delivers all this much-delayed mail. When Mike finally comes to pick up his order, he is handed two 30-count bottles of Xanax.

B. Patient Katherine is prescribed a refill of a medication by Dr. Daniel of CVH. A prescription is generated with a unique ID, and its hardcopy is faxed to Katherine's pharmacy. CVH enters this unique ID and the pharmacy name in an online portal to track outstanding prescriptions. A prescription is considered complete only when the patient confirms in the portal that they received the medication. If confirmation is not received within 48 hours, CVH will refax the prescription with the same unique ID. The pharmacy fills only one prescription per unique ID. If the pharmacy receives a prescription with an ID it has already processed, it is ignored. After a month-long vacation with no Internet access, Katherine comes to pick up her filled prescription at the pharmacy.

C. Patient Jack is prescribed medication by Dr. Jill of CVH. Dr. Jill sends the prescription via secured USPS mail to Jack's pharmacy. Each prescription includes the patient's name, medication name, and dosage. The pharmacy keeps a log of recently filled prescriptions. It enforces a rule that identical prescriptions for the same patient received within 24 hours of each other are assumed to be duplicates and discarded. A day after Dr. Jill mails the prescription for Jack, she realizes that she needs to double his dosage. So she sends an identical prescription to the pharmacy, with an explanation for Jack to take both prescriptions. Both prescriptions arrive on the same day at the pharmacy. The second prescription (which has the explanation for Jack) is discarded as a duplicate. When Jack arrives the next day to pick up his prescription, he is unaware that he will only be taking half the required dose.

Question 5 (25 points)

You are hired as an engineer at **VibeCode.ai** and assigned to a team rolling out a new LLM service for vibe coding. You are testing this service prior to public release. The application has the distributed architecture shown in the figure above.



A typical user interaction has the following flow.

- Using RPC, a user sends a request from their device. This request is called a "prompt", and it includes a globally unique request ID. The load balancer forwards the request to one of the front-end servers selected at random. To ensure exactly-once semantics, the frontend server saves the request ID and its prompt in a database. After ensuring that the save is complete, it replies to the client that this prompt will be submitted to the model server. When the client receives the reply, the RPC is complete. After replying to the client, the front-end server forwards the prompt to the model server, ensuring that its handling adheres to exactly-once semantics (the details of how this is done are not relevant.)

- Once the AI processing of a prompt begins at the model server, a stream of "tokens" is generated at irregular intervals. This processing may take many seconds, or even minutes. to complete. As tokens are generated, they are tagged with the globally unique request ID of their prompt and sent back to the client. The user sees the output for the original request being slowly completed on-screen as these tokens arrive at the client.

In this AI system, the best measure of end-to-end latency is time to first token (TTFT). You test the system and obtain the following TTFT measurements:

- Mean: 149 ms, Median: 102 ms, 95th percentile: 444 ms

These are very acceptable figures, and your manager approves the release of the application.

A. The product is an instant hit worldwide. Alas, its popularity starts to hurt performance. Customers start complaining about long TTFTs during peak usage periods. You measure TTFT under peak load conditions. While you are conducting your measurements, your ISP informs you that it is performing maintenance and upgrades on the Internet fiber lines into VibeCode.ai' s data center. You obtain the following measurements:
    - Mean: 502 ms, Median: 348 ms, 95th percentile: 1501 ms

    Identify three different sources of delay in the above figure that could plausibly explain the observed increase in end-to-end latency

B. To further understand the issue, you collect more detailed measurements as follows:
    - CPU utilization on each Front-End Server: Mean = 10%, 95th percentile = 30%
    - CPU utilization on Model Server: Mean = 10%, 95th percentile = 30%
    - GPU Utilization on Model Server: Mean = 70%, 95th percentile = 80%
    - Inference time per token on model server: Mean = 79 ms, 95th percentile = 238 ms
    - Network RTT from client to model server: Mean = 10ms, 95th percentile = 30ms
    - Packet loss rate: negligible on all network segments

    In light of these more detailed measurements, reassess the answers you provided in part A. Which possibilities are no longer plausible and why? What is the likely bottleneck?

C. The ISP completes its upgrades to the local fiber infrastructure and reports a 15% increase in the ingress bandwidth of the data center. But the user experience in terms of TTFT has not improved. Within a few weeks, you begin to receive even more complaints about long TTFTs, with the 95% percentile reaching 4900 ms.

    Give a plausible explanation for why the network infrastructure upgrade fails to improve TTFT. Given these observations, what is most likely the performance bottleneck? Why is TTFT getting worse rather than better?

D. Which of the following proposed changes and upgrades could plausibly improve TTFT during peak usage periods? In each case, give a brief explanation for your answer.

    a. Upgrade the Backend Database hardware to use SSDs rather than magnetic disks.

    b. Upgrade the GPU on the Model Server

    c. Add more Front-End Servers.

d. Modify the application to implement at-most-once semantics, and enforce a timeout between the front-end server and the backend database.

e. Modify the Model Server to use a lighter-weight model with lower inference times