Carnegie Mellon University in Qatar

Distributed Systems

15-440 - Fall 2023

Problem Set 2

Out: September 7, 2023

Due: September 26, 2023

Problem I: RPC Semantics (12 Points)

In each of the following situations, identify the ***weakest*** RPC semantic that can be used. You can think of an *exactly-once* semantic being stronger than *at-most-once*, which in turn, is stronger than *at-least-once*. Explain your answers.

a. Setting up a VoIP (Voice over IP) call to a friend

b. Uploading a blog post

c. Buying a mobile phone from an e-commerce website

d. Checking the stock price of a company

Problem II: Networking (15 Points)

Answer the following questions related to networking:

11pts

a. For each of the following applications, state and justify which of TCP or UDP is best suitable:

   i. Streaming a live football match

   ii. Streaming a YouTube video

   iii. An application that sends very large messages

   iv. An application that requires high throughput

   v. An application that is based on multicasting (one-to-many communication)

4pts

b. We know that transport-layer protocols such as TCP can be used to provide reliability guarantees on the delivery of messages. However, instead of providing these guarantees at the transport-layer of the OSI model, Ahmad, a CMUQ student, wants to apply them at the lowest possible level instead. Discuss the advantages and disadvantages of such an approach.

Handout continues on the next page(s)

Problem III: Where's My Printer? (15 Points)

Mohammad is designing a location-aware printing service for the CMUQ campus. Because there are so many printers on the network, scattered in many departments, it is often difficult to know where the nearest printer is located. Mohammad's printing system will route your document to the printer nearest to you, and tell you where that printer is located. In this design, users send documents from their laptops, tablets or smartphones to a central machine in IT (Information Technology). That central machine is aware of your current location as well as the locations of all printers on campus. It picks the optimal printer, sends your document to it, and returns to you the name of that printer and its location.

4pts

a. Identify the servers and clients in this system.

5pts

b. Mohammad creates an RPC interface on the central machine that implements a single print function. It requires the caller to provide the content and length of the document to print, and a structure that indicates the user's location. The RPC will return the location information for the printer that was actually used. It also sets a status to indicate whether the printer finished successfully or if something went wrong (e.g., the printer is jammed or out of paper). The print function has the following C prototype:

```
location_t* print (char *buf, int len, location_t *user_loc, int *status);
```

Identify and explain what complexities might arise in generating the stub code for this RPC. You can assume that `location_t` is a simple structure that is easy to marshall/unmarshall.

6pts

c. On the CMUQ campus, end-to-end bandwidth between a wireless mobile device and the IT server is typically 50 Mbps. End-to-end latency is typically 20 ms. What timeout value would you recommend for this RPC call? Explain your answer. Based on your answer, suggest an improved RPC interface for Mohammad's printing system.
Assume: a pdf document is 10MB in the worst case.
Assume: Serialization, deserialization, sending reply/result, and network latency are negligible relative to transmission of the document and printing.

Handout continues on the next page(s)

Problem IV: A Very Expensive Operation (22 Points)

Tamim's software for his business course runs on (relatively slow) Android smartphones. This software involves statistical analysis on many `int32` arrays, each containing a million entries. The operation `VeryExpensive(...)` on these arrays is frequently invoked by his algorithms. To speed up execution, Tamim is considering offloading execution of this operation via RPC to a server that uses a high-end GPU for ultra-fast computation of `VeryExpensive(...)`. The C prototype of the RPC is:

```
int32 VeryExpensive (int32 *array)
```

This RPC sends the specified array as input, and returns the single int32 value of `VeryExpensive(...)` on that array.

You may assume the following performance costs:

- marshalling or unmarshalling one integer: 100 ns on smartphone, 10 ns on server
- OS cost of sending a request or receiving a reply: 30 μs on smartphone, 10 μs on server
- one-way client-server network latency (symmetric): 2 ms
- client-server bandwidth (symmetric): 50 Mbps
- compute time for `VeryExpensive(...)` on server for a million-integer array: 5 ms

All other performance costs can be ignored.

8pts a. Suppose Tamim wants to compute `VeryExpensive(...)` on one of his arrays. How long will it take to perform a single RPC call, assuming no failures occur?

5pts b. Suppose Tamim's smartphone is a single-core machine (without hyperthreading). The algorithm being run requires two other computations, each taking roughly 6 ms on the smartphone. These two computations have no dependencies on each other or on `VeryExpensive(...)`. In other words, all three computations can execute in parallel with no synchronization. The final result, however, depends on the result of all three. If Tamim created multi-threaded code to perform both the local operations in parallel with the RPC for `VeryExpensive(...)`, how long would the entire algorithm take? *Remember: although conceptually threads on a given processor are said to run at the same time, they are actually running consecutively in time slices allocated and controlled by the operating system.*

4pts c. If Tamim replaced his old single-core smartphone with a brand new quad-core smartphone, how would your answer to part b. change?

5pts d. Explain what would happen at the client and server if Tamim loses wireless connectivity immediately after his RPC request is sent. Clearly state and justify any assumptions you make.

Problem V: Where Do I Go? (12 Points)

Bob wants to cause trouble for Alice. Alice is working for a startup in Doha that is introducing a new service for last-minute vacations. Her client software first contacts a weather server to identify the best locations in Europe for a short vacation in the next 2 days, taking into account the customer's vacation preferences. For example, if the customer likes skiing, optimal weather would be defined as fresh and deep snow. If the customer prefers a beach vacation, the optimal weather would be sunshine and blue skies with moderate wind. After the top few vacation destinations have been identified, Alice's client software contacts a travel website (like Expedia) to check on air fares to those destinations. Based on the intersection of weather and fares, the system recommends the optimal vacation destination and its price to the customer. Afterward, the customer can select to purchase this recommended vacation fee and book the tickets.

6pts a. Identify two problems Bob can create for Alice if she is not careful in designing her system. Clearly state and justify any assumptions that you make.

6pts b. Is there a scenario in which Alice buys a plane ticket for a suboptimal destination? If yes, give an example of such a scenario, and explain how Alice could fix her implementation to avoid such scenarios. If no, explain why such a problem cannot occur.

Problem VI: Data Backup (12 Points)

WE_NEVER_LOSE_IT is a highly reliable data archiving service located in Doha. To reduce the chances of a catastrophic site failure (such as a fire or sandstorm) wiping out data, the company decides to use off-site mirroring. When archival data is added to the primary site in Doha, copies are also sent via a dedicated network to its two backup servers in Dubai (close to Doha) and Zurich, Switzerland. The network bandwidth on all links is guaranteed to be 800 Mibps (i.e., 800×10^6 bps).

To test the stability of the network, WE_NEVER_LOSE_IT first sends out a test packet of size 10 bytes from the Doha server to each of the backup servers. Each backup server sends a 10-byte ACK in response. The time required for the Doha-Dubai-Doha route is much smaller than for the Doha-Zurich-Doha route. Specifically, after multiple trials under carefully controlled conditions, the total time (from start to ACK) on the Doha-Zurich-Doha is observed to be about 54 ms longer than the value for Doha-Dubai-Doha. You can assume that processing time is negligible, no packets are lost, and no data corruption happens. For your answers below, please explain your reasoning. State and justify any assumptions you make.

6pts a. Why does the Doha-Zurich-Doha interaction take longer than the Doha-Dubai-Doha interaction? From the data provided, can you estimate the distance between Doha and Zurich?
Assume: the network is connected by optical fibre and use the speed of light to estimate the distance between the two cities.

6pts b. Now consider the following protocol for data transfer. Assume all data packets are of size 500 bytes. If necessary, data objects are padded with null bytes to make their length a multiple of 500 bytes. The source sends out exactly 2000 packets to the destination, and then stops to wait for an ACK. Once the ACK is received, the source immediately sends the next 2000 packets, and so on. What is the throughput of this protocol between Doha and Zurich for a multi-gigabyte archived data object? Express your answer as a percentage of network bandwidth.

Problem VII: Tag Storage System (12 Points)

Khaled's startup provides a website for posting pictures of vehicles and tagging them. The persistent storage for the images and tags is provided by a key-value store. Here is the design of his system:

[User : HTTP] → [Web Server : RPC] → [Tag Storage System]

In other words, the user interacts with a webpage using standard HTTP requests. When the user clicks on a widget (e.g., a button to delete an image), the web server issues an RPC to the tag storage system to perform the operation. The RPC interface supports three operations on the tag storage system: INSERT, DELETE and FIND.

You may assume that the tag storage system does not crash. However, the network between the web server and tag storage system is unreliable and may drop packets. Assume that the Web server is single-threaded, and that no other services access the tag storage system. Consequently, the tag storage system only has to process one RPC at a time. Clearly and concisely explain your reasoning in your answers to the questions below:

3pts a. Assume an at-least-once RPC semantic. Can the following situation occur?

A `FIND` RPC performed on the web server is successful, but returns no matching tags even though matching tags exist in the tag store.

3pts b. Again, assume an at-least-once RPC semantic. Can the following situation occur?

A `DELETE` RPC on the web server returns a "NoSuchTag" error, even though it successfully deleted a tag.

3pts c. Now assume an at-most-once RPC semantic. If no reply is received after some time, the RPC terminates with a timeout error code that is then returned by the web server to the user as an error web page. Can the following situation occur?

A `FIND` RPC does not timeout, but fails to return a matching tag even though it exists in the tag storage.

3pts d. Again, assume an at-most-once RPC semantic as in Part C. Can the following situation occur?

A `DELETE` RPC returns "NoSuchTag" error, even though it successfully deleted a tag.