# Problem Set 1

## 15-440/15-640 Distributed Systems Spring 2017

**Assigned:** Thursday February 2, 2017

**Due:** Thursday February 9, 2017 (by the start of class)

**Submission procedure:**

- Create a .pdf of your answers and upload to Autolab**.**
- If you handwrite your answers, scan into .pdf before uploading. Please use a real scanner that outputs easily readable .pdf. Images taken with a camera (e.g., your smartphone or tablet) are not acceptable. Illegible submissions will receive a zero grade.

## Question 1 (16 points)

In each of the following situations, identify the **weakest** RPC semantics that can be used. As explained in class, *exactly-once* semantics is stronger than *at-most-once*, which 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

## Question 2 (20 points)

John is designing a location-aware printing service for the CMU campus. Because there are so many printers on the network, scattered in many buildings, it is often difficult to know where the nearest printer is located. John'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 Cyert Hall. 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.

A. Identify the servers and clients in this system.

B. John 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);
```

You may assume that `location_t` is a simple structure that is easy to marshall/unmarshall. What are the `in`, `out`, `in-out` parameters of this RPC?

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

D. On the CMU campus, end-to-end bandwidth between a wireless mobile device and the Cyert Hall 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 John's printing system.

## Question 3 (24 points)

Terry's software for citizen science 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, Terry 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.

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

B. Suppose Terry'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 Terry created multi-threaded code to perform both the local operations in parallel with the RPC for `VeryExpensive(…)`, how long would the entire algorithm take?

C. If Terry replaced his old single-core smartphone with a brand new quad-core smartphone, how would your answer to part B change?

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

## Question 4 (12 points)

After wreaking havoc on Checkpoint 2 of Project 1 in 15-440, Murphy needs a break. However, he still needs to cause trouble for Alice. She is working for a Pittsburgh startup that is introducing a new service for last-minute vacations. Her client software first contacts a weather server to identify the best locations in the United States for a short vacation in the next 24 hours, 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 Orbitz) 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. Murphy seeks your help to cause trouble for Alice.

A. Identify two problems you can help Murphy create if Alice is not careful in designing the system. Clearly state and justify any assumptions that you make.

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.

## Question 5 (12 points)

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

To test the stability of the network, WE_NEVER_LOSE_IT first sends out a test packet of size 10 bytes from the Seattle server to each of the backup servers. Each backup server sends a 10-byte ACK in response. The time required for the Seattle-Tacoma-Seattle route is much smaller than for the Seattle-Zurich-Seattle route. Specifically, after multiple trials under carefully controlled conditions, the total time (from start to ACK) on the Seattle-Zurich-Seattle is observed to be about 54 ms longer than the value for Seattle-Tacoma-Seattle. 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.

A. Why does the Seattle-Zurich-Seattle interaction take longer than the Seattle-Tacoma-Seattle interaction? From the data provided can you estimate the distance between Seattle and Zurich?

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 Seattle and Zurich for a multi-gigabyte archived data object? Express your answer as a percentage of network bandwidth.

## Question 6 (16 points)

Harry's startup provides a web site 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 web page 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. So 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:

A. Assume at-least-once RPC semantics. 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.*

B. Again, assume at-least-once RPC semantics. Can the following situation occur?

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

C. Now assume at-most-once RPC semantics. 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.*

D. Again, assume at-most-once RPC semantics as in Part C. Can the following situation occur?

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