Problem Set 1

15-440/15-640 Distributed Systems Spring 2015

Assigned: Tuesday February 3, 2015
Due: Thursday February 12, 2015 (at the start of class)

Question 1 (15 points)

Suppose you wish to create an RPC mechanism for a Java client on an Android smartphone (ARM hardware) to communicate with server code written in C that is running on a Linux server (Intel Xeon x86 hardware). Relative to the RPC mechanism you just implemented for Project 1, what additional complexity would your new RPC mechanism need to handle? How would you address each of the complexities you identify? Consider both primitive data types (such as ints, floats, chars) and complex data types (such as structs and strings).

Question 2 (20 points)

Jane loves to listen to music. In fact, her mornings cannot begin until she listens to "Comfortably Numb" by Pink Floyd. She has a Spotify music subscription service which streams her favorite melody, which is S seconds long, at T bytes per second. Jane, as you may not be aware, is also one of the first human inhabitants on Mars. Unfortunately, all the Spotify servers are located on Earth. The network between Earth and Mars has a one-way latency of L seconds (i.e. RTT is 2L seconds), and a bandwidth of B bytes per second. Fortunately, end-to-end transmission reliability between Earth and Mars is so good that no acknowledgement packets are sent.

Considering each condition separately, answer the following:

A. Once Jane requests for the song to be played, how long will it take before she can hear the first note of the song?

B. Suppose sunspot activity increases dramatically. It is predicted that Earth-Mars network communication will be disrupted because many packets will be dropped. Jane hates scratchy-sounding music, and decides to wait until the entire song is buffered, before playing it. What is the minimum time that it will take for the song to buffer completely after Jane presses the play button? What is the maximum time?

C. Consider the same conditions as mentioned in (B) but assume that B >> T. What is the shortest time for the song to be completely buffered?

D. Consider the same conditions as mentioned in (2) but assume that T >> B. What is the shortest time for the song to be completely buffered?

Question 3 (25 points)

For a use case involving files of modest size, you are considering whether to add whole-file caching at clients to a distributed file system. Your experiments show that it takes 200ms for a client to

retrieve a typical file from the remote system, but only 5ms for it to retrieve it from the local disk. You also determine that in a trace of 1000 file accesses, 724 were to files that had been previously accessed. Based on these measurements, answer the following questions.

A. What is the cache advantage of your system?

B. What is the hit ratio, miss ratio, and expected cost of a reference?

C. Based on these figures, do you think caching should be implemented for this system? Explain your answer.

D. Identify a situation where you would not use a cache, despite there being a high cache advantage.

E. Identify a situation where you would use a cache, despite there being a low cache advantage.

F. Identify two shortcomings of full replication (e.g. DropBox) relative to on-demand caching.

Question 4 (10 points)

What are some motivations for using the RPC paradigm?

Question 5 (10 points)

Excluding issues that arise from lack of a shared address space across machines, explain two different ways in which programming using RPC is more complicated than programming using local procedure calls.

Question 6 (10 points)

Why is it typically harder to reduce end-to-end latency than to improve throughput in a typical distributed system?

Question 7 (10 points)

NeverLoseIt, a new startup with patent-pending technology, claims that its storage-enhanced network routers can greatly improve file transfer reliability in a network that is composed solely of such routers. Each router has sufficient storage to buffer an entire file. Now, a client application can just hand its file to the closest router, which then takes responsibility for getting it to the next router, and so on, until the last router in the chain hands it to the destination server. Each link of this chain is reliable, because each router retransmits a file to the next router until it is acknowledged. NeverLoseIt claims that client application code can be much simpler because it does not have to check for success acknowledgement from the destination server. Can this possibly be true?
Explain your answer.