15-440/640 Distributed Systems
Homework 1

Due: September 29, 2015, 12:00 PM (in class)

Name:
Andrew ID:

October 8, 2015

1. Blogger Bob decides he wants to set up streaming video. You can treat this video as though it were already stored (like an image). What is the max video bit rate (in bits/s) he can get given the following conditions? Assume the best case transmission time, where there is no processing being done at the packet's destination. Also assume that RTT is symmetric, so if RTT = 200 ms, the time taken in each direction is 100 ms.

   1 Mbps = 10^6 bits/s

   1 Kbytes = 2^{10} bytes

   (a) RTT (Round Trip Time) = 200 ms. Packet size = 10 Kbytes, Bandwidth = 2 Mbps. Packets are sent continuously, as in UDP.

   > **Solution:** max bit rate = 2 Mbps (bandwidth)

   (b) RTT = 200 ms. Packet size = 10 KBytes, Bandwidth = 2 Mbps. After each packet is sent we must wait one RTT (waited for an ACK/Acknowledgement).

   > **Solution:** Packet size=10*1024*8=81920 bits
   > transmission time = (81920 bits)/$(2 * 10^6$ bits/s) = 0.041s
   > max quality = 81920 bits/(0.041+0.2) = 339917 bits/s = 0.340 Mbps

   (c) RTT = 200 ms. Packet size = 10 KBytes, Bob is provided with infinite bandwidth, but only up to 20 packets can be sent per RTT. Packets are sent continuously.

   > **Solution:** Data sent/rtt = 20*81920 = 1638400 bits
   > max quality = 1638400 bits/0.2 = 8192000 = 8.192 Mbps

   (d) RTT = 200 ms. Packet size = 10 KBytes, Bandwidth = 2 Mbps. Video filesize = 100 MB. Bob is experimenting with a new way of sending data. He sends 1 packet and waits for an ACK, then 2 packets and waits for an ACK, then 3 packets, and so on until all the data is sent. He stops increasing number of packets after max bandwidth is reached.

   > **Solution:** We asked for "full bandwidth" to be interpreted as full bandwidth for 1 second. The solution in this case is:

Test to see how much data can be sent for 2 Mbps
number = $2 * 10^6/81920 = 24$ packets
Since 24 packets is well under the 100 Mb to be sent, it stops at 24 packets/rtt
This is the same as problem B only we send 24 packets instead of only 1.

transmission time = 24*(81920 bits)/($2 * 10^6$ bits/s) = 1s
max quality = (24*81920 bits)/(1+0.2) = 1638400 bits/s = 1.638 Mbps

If you interpreted the question so that the number of packets Bob sends is the size of the sliding window, the solution should be:
number of packets in RTT = $2 * 10^6/81920$ * 0.2 = 4.8 packets
Bob stops at 5 packets, and max quality is 2 Mbps.

(e) What is the minimum number of packets that Bob could send before waiting for an ACK and still achieve the full bandwidth? RTT = 200 ms, Packet size = 10 KBytes, Bandwidth = 2 Mbps. Packets are sent continuously.

Solution: number of packets in RTT = $2 * 10^6/81920$ * 0.2 = 4.8 packets
Rounds to 5 packets

2. Jimmy, a student in distributed systems at Carnegie Mellon, wants to build a distributed file system. Aware that this is a major undertaking, he decides to spend some time designing the system before implementing. For his first version, Jimmy assumes that the system initially doesn't contain any files, all files are stored in the same directory, and clients should be unaware of files created by other clients. Being a good friend, you decide to help Jimmy.

   a) In the following client-server interaction, please write down where unexpected behavior could occur. Also describe how Jimmy should build the system so that clients are unaware of each other's files.

   1. Client A creates and opens file1 for writing
   2. Client A writes to file1
   3. Client A loses connection to the server
   4. Client A writes to file1
   5. Client A regains connection to the server
   6. Client A closes file1
   7. Client A opens file2 for reading
   8. Client B creates and opens file2 for writing
   9. Client B's system crashes, and is restarted

Solution: At line 3, when Client A loses connection to the server, updates are no longer being synced with the remote version of the filesystem. One option is to sync files on close(), the way NFS does. However, there would still need to be a way of identifying incompletely synced files in case the connection is interrupted while a sync is in progress. While syncing, the server should mark a file as dirty, and retry until the client sends a message saying that the entire file has been transmitted.

Syncing on close is a good option for intermittent internet connection, and a scenario where users cannot read or write to each others' files.

At line 7, client A attempts to open for reading when it doesn't exist. The distributed file system should return an error message to the client.

At line 9, our design would result in file2 being lost.

Since Jimmy does not want clients to be able to see each others' files, the system could be built with one directory per user, where users cannot see other users' directories.

b) Suppose Jimmy decides that he wants clients to be able to read files created by other clients, but not to write to them. How might your design change?

Solution: We might choose to sync files continuously instead of on close(). The client should retry until it receives an ack from the server.

We could also choose to continue syncing files on close, but to also force a sync when another client reads a file.

3. Every distributed system faces trade offs between consistency, availability, and partition tolerance or resistance.

   a) Define the three terms.

Solution: Consistency - All nodes see the same data at the same time. Performing a read operation will return the value of the most recent write operation causing all nodes to return the same data. A system has consistency if a transaction starts with the system in a consistent state, and ends with the system in a consistent state, though the system might be inconsistent during the transaction.

Availability - Every request gets a response on success/failure. Achieving availability means that the system is operational all of the time. This metric is trivial to measure: either you can submit read/write commands, or you cannot.

Partition Tolerance - System continues to work despite communication failure between two nodes. A system that is partition-tolerant can sustain any amount of network failure that doesn't result in a failure of the entire network. Data records are sufficiently replicated across combinations of nodes and networks to keep the system up through intermittent outages.

b) It is said you can satisfy at most two out of the three aforementioned goals. Why is this the case?

Solution: Consider a system that is partition tolerant. If we want to maintain consistency, we need to ensure that those nodes that are no longer connected do not allow the user to see any data because it could be out of date. Thus we are forced to sacrifice availability. If we want availability, we would allow disconnected nodes to still function and show data even if it could be stale (incorrect). In this case we are forced to sacrifice consistency.

If we want both consistency and availability, that means we cannot allow for nodes to go down or become disconnected. Thus we are sacrificing tolerance of network partitions which is impossible because there is no such thing as a perfect network.

c) Which of these traits might an online banking system optimize? Why?

Solution: Consistency and Partition Tolerance. An online banking system should be consistent to ensure an accurate representation of available funds. It should be partition tolerant so that customers can access their accounts even in the case of network failure.

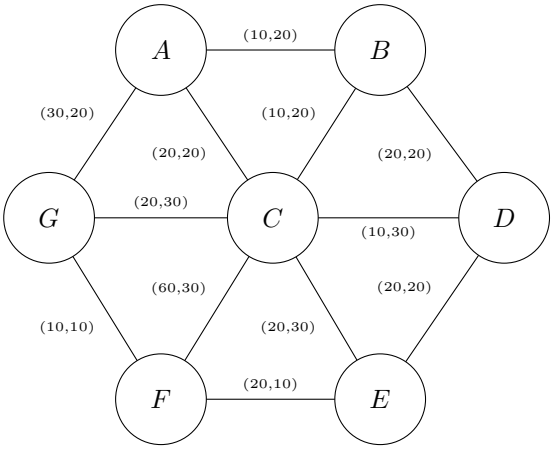
d) Which of these traits might facebook optimize? Why?

Solution: Availability and Partition Tolerance. Facebook data does not need to be completely consistent across all nodes. However, users will complain if the site is not available (even getting a request failure is better than no response), so it is important for the site to be available and tolerant to network failures.

e) Which of these traits might an airport departure system optimize? Why?

Solution: Consistency and Partition Tolerance. The system must replicate data so that network failures will not bring down the system. Otherwise, travelers would be unable to see their flight information. The system must also be consistent, so that incorrect information is not being displayed.

4. In the diagram below, the links are labeled with (latency (in ms), bandwidth (in Mbps)). Also assume that data transmission occurs without loss of data, and sources transmit at rates such that no queueing results.



a) Given this network topology, what path gives the lowest possible end-to-end latency from node B to node F?

Solution: BCGF

b) What is the bandwidth of this route?

Solution: 10 Mbps

c) What would be the bandwidth of this route if data is also being sent from node A to node F (along the route with lowest latency)? Assume that, as in TCP, bandwidth gets allocated equally to both data streams.

Solution: We assume that bandwidth is allocated equally to both streams of data, and that we have full duplex links. Link GF is being shared, and the bandwidth of GF is 10, so half of this would be 5 Mbps.

d) What is the latency of this route if data is also being sent from node A to node F?

Solution: 40 ms (for simplicity, we don't consider queueing delay)

e) What path gives the highest bandwidth for transmitting data from node B to F?

Solution: Any route that reaches F along the link from C to F.

f) What is the latency of this path?

Solution: 70 ms

g) What path gives the lowest possible latency between nodes B and F if C fails?

Solution: BAGF

h) What is the bandwidth of this path?

Solution: 10 Mbps

i) What is easier to improve, bandwidth or latency? Give at least two reasons for your choice.

Solution: Bandwidth.

1) One can simply add more or wider links between nodes.

2) Latency is lower bounded by the speed of light, and cannot be improved much beyond a certain point. Bandwidth grows increasingly expensive to improve, but does not have this restricting bound.

5. In order to store data from a client to a server, the message needs to cross a a sea of routers. Typically a router doesn't store any messages which means they do not ensure that subsequent routers receive the message. This requires the client to write logic to figure out whether or not the message it sent actually reached its destination (TCP or UDP). Imagine you are an angel investor and a start-up proposes the concept of reliable routers have the responsibility of making sure the data they receive is transmitted to the next reliable router (and so on until the message reaches its destination). The selling point of this start-up is that their product allows for a simplification of client code. Is this a good investment? Why or why not?

Solution: This is not a good investment. The client code would still have to ensure that the first router in the chain received the message, so it is no simpler. By the end-to-end principle, this logic would be better implemented in the client and the server.

6. a) The RPC paradigm is useful in many situations. What are some of the motivations for using RPCs?

Solution: Networks are complex. It is convenient to abstract this away into another layer, making it transparent to clients so that application programmers have a way to make remote calls in a way that "looks and feels" like local calls. That is, remote calls now emulate local procedure calls, so all the complexity of the "remote" part is taken care of (mostly). This in turn allows for more flexibility and freedom in application programming, when a lot of this complexity is hidden and related errors are already automatically handled more focus can be given to develop other parts of the system. For example, RPC can be used to develop distributed parallel computing systems. Server(s) can be master scheduler(s) and distribute work to client/worker machines across a network.

b) RPC's are supposed to be transparent. Describe two ways in which this is not true.

Solution: RPCs cannot do the following:

- Global variables cannot be shared (client / server)
- Pointers cannot be used

In addition, failure handling is more complex and performance is more variable.

c) Describe two ways in which an application may need to handle this lack of transparency.

Solution: For example, reading a file will require syscalls. The application will need to packaged up these syscalls and send them to another machine. Exception and error handling will also need to be treated differently from when the procedure call is local. There are a greater variety of possible errors. If an exception occurs remotely, the application will have to check for and respond to these exceptions rather than relying on exception handlers.