Carnegie Mellon University in Qatar

Distributed Systems

15-440 - Fall 2020

Problem Set 3

Out: Septebmer 28, 2020

Due: October 12, 2020

Problem I: Chord (44 Points)

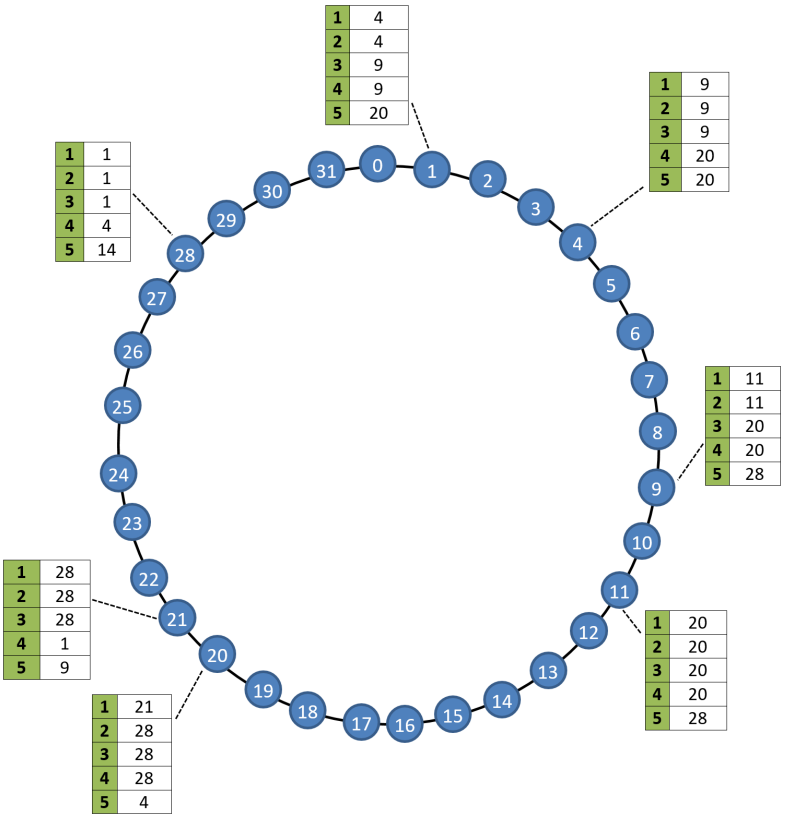

Figure 1: Chord

12pts

a. Consider the Chord system shown in *Figure 1*. Assume each node provides a recursive `retrieve()` RPC to lookup and retrieve content in the DHT. Furthermore, between any pair of nodes in the DHT, assume an average round trip time (RTT) of 100ms, and bandwidth of 16 Mbps. You can also assume that local processing times (for hash computation, scanning finger table, loading content, etc...) are negligible relative to the network times. How much time can we expect `retrieve()` to take, assuming no failures?

14pts

b. Suppose now, that Node **20** leaves the system voluntarily. Describe all the steps that need to be taken to bring the Chord system up-to-date.

10pts

c. Suppose that, instead of **Node 20** leaving the Chord system voluntarily, it *fails* abruptly. This means that, with the current finger tables, the closest successor for **Node 11** would be **Node 28** (since **20** is down). Therefore, **Node 20** disregards all the nodes between **Nodes 21** & **28**, excluding **Node 28** (regardless if these nodes are active or not). How can Chord resolve this problem? (*Hint: you can change or add additional data at each node*).

8pts

d. The given Chord system runs on Windows, Mac, and Linux PCs. However, several users have requested to port it to Android and iOS so it can be run on smartphones. Is this a good idea? Briefly explain with two reasons why or why not.

Problem II: Group Communication & Synchronization (56 Points)

Suppose that we can classify the various kinds of multicasts possible into four classes of source-destination relationships, as illustrated in Figure 2:

- SSSG: Single source and single destination group.
- MSSG: Multiple sources and single destination group.
- SSMG: Single source and multiple, possibly overlapping, groups.
- MSMG: Multiple sources and multiple, possibly overlapping, groups.

Keep in mind the following assumptions:

- A source that is sending messages to any group(s), must also be part of that group(s).
- The source that is sending a message to a group does not receive the message itself.
- There can be some sort of information sharing amongst group members.

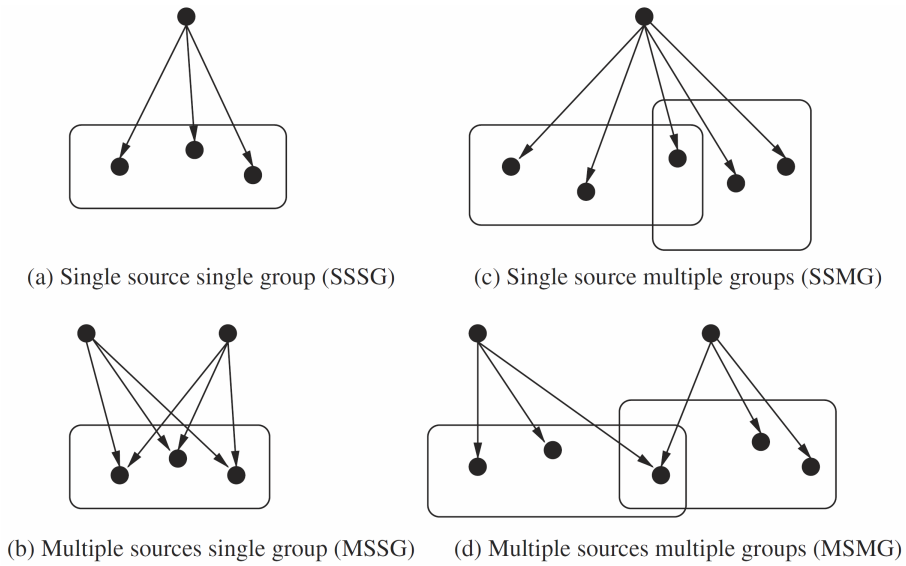

Figure 2: All possible multicasts

12pts

a. Consider two source nodes ***N1*** and ***N2*** in MSSG (see part (b) in *Figure 2*). Assume that these source nodes do not have synchronized clocks, and that the underlying network is a LAN. If these sources were to deliver multiple messages to the designated group, how can we ensure that the messages are received in order? That is, if ***N1*** sends message *M1* before ***N2*** sends message *M2*, the group members on the receiving end must receive *M1* before *M2*. (*Hint: try to logically convert MSSG to SSSG.*)

8pts

b. Consider the following scenario: source nodes ***N1*** and ***N2*** send messages *M1* and *M2* respectively, to Group ***G1***, where ***G1*** has members {N1, N2, N3, N4}. How many messages are sent in total? Is this the optimal number of messages that can be sent? If not, then what would be the optimal number and how can that be achieved?

8pts

c. Suppose that we would also like to have messages received in order in MSMG (see part (d) in *Figure 2*). Without creating a completely new algorithm/solution, can you rely partially or fully on the one you devised for MSSG in 2.A (Q2 Part A)? If not, suggest a completely new solution. Explain your approach in detail.

8pts

d. Consider the following scenario: source node ***N1*** sends messages *M1* and *M2* to group ***G1***, which has members {N1, N2, N4, N5, N6, N8}, and source node ***N2*** sends messages *M3* and *M4* to group ***G2***, which has members {N1, N2, N3, N4, N6, N7, N8, N10}. How many messages are sent in total? Is this the optimal number of messages that can be sent? If not, then what would be the optimal number and how can that be achieved?

8pts

e. Consider the behavior of two nodes in any of the classes depicted in *Figure 2*. Both nodes have clocks that are supposed to tick 1000 times per millisecond. One of them actually does, but the other ticks only 990 times per millisecond. If UTC updates come in once a minute, what would be the maximum clock skew?

12pts

f. Assume a node in any of the classes shown in *Figure 2* is attempting to synchronize with a time server. As a result, it records the round-trip times and timestamps returned by the server in a table as shown below (i.e., *Table 2.1*)

Round-Trip (ms)	**Time** (hr:min:sec)
22	10:54:23.814
25	10:54:25.250
5	10:54:28.442

Table 2.1

1. Which of the times in *Table 2.1* should the node use to adjust its clock? To what time should the node set its clock? Estimate the accuracy of such an adjustment with respect to the server's clock.
2. If it is known that the time between sending and receiving a message in any of the classes portrayed in *Figure 2.1* is at least 8 milliseconds, do your answers (i.e., the time setting and/or the accuracy) change? Explain.