Distributed Systems Final Exam

Karan Sarkar

December 14, 2018

Question 1

a. We will try to extend the token ring algorithm to the k mutual exclusion problem. The solution is to have k tokens rather than a single token. All of the processes p_i are aranged in a ring. When process p_i has a token, it can access the resource. When it is done with the resource or does not need the resource, it passes the token clockwise to the next process on the ring.

b. Our algorithm works in a model that has reliable asynchronous communication with no crashes. It guarantees both liveness and safety. Liveness means that if every process that has the resource is eventually done with it, then every request will eventually be granted. In our case, if every process releases the token to its clockwise neighbor, the token will eventually make its way around the whole ring. This ensures that every process will eventually have access. Safety means that no more than k processes can access the resource at a given time. Our algorithm guarantees safety because there are only k tokens and a process needs a token in order to access the resource. Therefore, only k processes can possibly access the resource at a given time.

c. In the worst case, all of the tokens are just barely clockwise in front of the requesting processes. Suppose we number the processes in clockwise order $1, 2, \ldots, n$. The tokens could be at $1, 2, \ldots, k$ while the requesting processes could be at $n-k+1, n-k+2, \ldots, n$. We can see that no matter which requesting process is paired with which token, the sum of the distances between tokens and requesting processes is $k(n-k)$. The only messages needed in a token ring algorithm are those that pass the tokens around the ring. Therefore, the worst case message complexity is $k(n-k)$.

Question 2

a. Dr. Science's algorithm does not guarantee a consistent global state. For example, consider two processes: process A and process B. We can assign each process a vector of timestamps representing the greatest timestamp it has associated with each of the processes in alpahabetical order. Suppose A has $\{T, T\}$ and B has $\{T-1, T\}$; Suppose that A sends B message m. Note that A has already created its shapshot but B has not. After A sends the message, we have that A has $\{T+1, T\}$ and B has $\{T+1, T\}$. Now B will make a snapshot. However, the snapshot of B includes receive$m$. However, send(m) is not in the snapshot of A. Because the send happens before the receive, this violates consistency.

b. Dr. Science's algorithm does not guarantee a transitless global state. For example, consider two processes: process A and process B. We can assign each process a vector of timestamps representing the greatest timestamp it has associated with each of the processes in alpahabetical order. Suppose A has $\{T, T-1\}$ and B has $\{T, T\}$; Suppose that A sends B message m. Note that B has already created its shapshot but A has not. After A sends the message, we have that A has $\{T+1, T-1\}$ and B has $\{T+1, T\}$. Now, suppose that B sends A a message. After B sends the message, we have that A has $\{T+1, T+1\}$ and B has $\{T+1, T+1\}$. Now A creates its snapshot. However, the snapshot of B does not include receive$m$. However, send(m) is in the snapshot of A. Because the send is in the snapshot but not the receive, Dr. Science's algorithm is not transitless.

Question 3

Suppose that our two generals are A and B. We claim that if the total number of message failures is no more than three and the model is synchronous, than the problem is solvable. First, A sends four identical messages telling B its opinion. Because our three messages can be lost, we know that at least one message got through. Second, after receiving a message from A, B sends four messages back that say the final decision by combining its opinion with the opinion received from A. Because only three messages can fail we can be sure than at least one got through. Third, after receiving the decision from B, A sends four messages back confirming the decision. As before, we can be sure that at least one message got through. Note that the message are labeled by type. Note that this will always work in a synchronous system but can take indefinitely long in an asynchronous system.

Question 4

a. It is possible for everyone to vote to commit but still to have abort decided. First, the coordinator asks the other processes whether they want to commit and they all vote yes. Second, all of the cohort processes crash fail. As a result, the coordinator never receives any acks and decides to abort.

b. Termination is upheld because every correct process decides some value. The only correct process was the coordinator which decided to abort. Validity means that if all the correct processes proposed the same value v then v must be the agreed upon value. However, because the processes that voted for commit are not correct, validity is upheld. Agreement is upheld because the only remaining process decided to abort.

Question 5

Validity means that if a correct process broadcasts a message m, then it will eventually deliver it. Note that a broadcasting process p sends m to itself. p eventually receives m and subsequently delivers m. Therefore, validity is upheld.

Agreement means that if a correct process delivers m, then all correct processes eventually deliver m. Since a correct process p sends m to all other processes before delivering m if p delivers m then all correct processes which receive m will deliver m. Therefore, agreement is upheld.

Integrity means that if a correct process delivers a message m at most once and only if it was previously broadcasted. After receiving m, we check to see if we have delivered m before. This ensures that m is delivered at most once. Moreover, m can only come from a broadcast message, so some process must have broadcasted it. Therefore, integrity is upheld.

FIFO order means if p broadcasts m_1 before it broadcasts m_2, then no correct process delivers m_2 before unless it has already delivered m_1. Assume for the sake of contradiction that there exist processes that delivered m_2 before m_1. Therefore, these processes must have received m_2 before m_1. Out of these processes that received m_2 first, let p be the one that received m_2 at the earliest time. Suppose p received m_2 from process q. Because p received m_2 first from among the processes that received m_2 before m_1, we now that q must have received m_1 before m_2. Therefore, before sending m_2 to all processes including p, we have that q must have sent m_1 to all processes including p. Because communication channels are FIFO, we have that, p must have received m_1 from q before receiving m_2. This contradicts our assumption. Therefore, the broadcast is FIFO.

Question 6

In atomic broadcast, all correct processes receive the same messages in the same order. This could be implemented by Paxos. Instead of agreeing on log entries, Paxos would be used to ensure agreement on each message in the sequence. Note that Paxos only works if sufficient proposers remain non-faulty. Paxos will work in an asynchronous model with message loss and crash failure. Paxos would be used as follows. Each process would propose the message it wants broadcasted. If its proposal fails, it would try again.

Validity means that if a correct process broadcasts a message m, then it will eventually deliver it. As long as there are finitely many messages and sufficiently many poposers are not faulty, eventually a processe's proposal will be accepted. This ensures validity.

Agreement means that if a correct process delivers m, then all correct processes eventually deliver m. This is guaranteed by Paxos agreement. All of the learned values will be shared across the processes. Therefore, agreement is ensured.

Integrity means that if a correct process delivers a message m at most once and only if it was previously broadcasted. The at most once part is guaranteed because a proposer will stop proposing its message once accepted. The only if previously broadcasted part is guaranteed by Paxos validity that only chosen values can be learned. This guarantees integrity.

Total order means that all processes agree on the order of the broadcasts. This is guaranteed by the nature of Paxos. Paxos reaches agreement on many sequential entries. We are using Paxos to agree on the sequence of broadcasts. Therefore, total order is guaranteed.

Question 7

We will implement the replicated set as a list of entries. There is one entry per set element d. An entry either stores a add(d) or a remove(d). We want to make sure that an add(d) and remove(d) do not occur at the same time and are instead performed atomically. We will use an total weight of 4, an add threshold of 2 and a remove threshold of 3. Note that two add operations will not occur. In our implementation, an add and remove cannot happen at the same time because they require a total voting weight of 5 which exceeds our system total of 4.

Question 8

a. Range queries could be handled by creating a binary tree that contains the prefixes of entries. Moreover, each leaf node should contain a link to the next leaf node via inorder traversal. We would want there to be roughly an equal number of entries per prefix. Then, each all the entries with a specific prefix are assigned to a node by hashing the prefix. Now range queries could be implemented by iteratively querying a list of prefixes associated with the range.

b. Our algorithm first has to find the first find the first prefix bucket. Because we use a binary tree, this should take $\log(D)$ time where D is the number of digits. Next, we iterate through P prefixes. Note that we obtain the next prefix in $O(1)$ time because each leaf in the tree is linked to the inorder successor. It also takes $O(1)$ time to determine which site maps to that prefix becausing of the hashing. Therefore, iterating takes $O(P)$ time total. It is linear in the number of prefixes iterated through. Therefore, the total complexity is $O(\log(D) + P)$ where D is the number of digits and P is the number of prefixes searched.

c. If the set of participating nodes changes. Our approach to range queries will not guarantee to get every element within the range. This is because each node represents a segment of the total range. For example, suppose we are searching for all entries from A-C. If a node that dealt with Bs drops out, we may not be able to react in time. As result, some of the entries may be missing.