CS5320: Distributed Computing, Spring 2020

Final Exam

SAI HARSHA KOTTAPALLI

CS17BTECH11036

Q1 (8 points). Consider the solution to the crash failure consensus problem that we studied in the class, Algorithm 14.1 of AM book (page 516).

(a) Later in the class, we discussed the early stopping variant of this algorithm, If the upper bound on the number of failures is f and the actual number of failures are f' then the algorithm terminates in number of rounds less than or equal to min(f 0 + 2, f + 1). What is the message complexity of this algorithm?

The worst case occurs when a process fails every round.

In this case, in every round each active process according to the algo has to broadcast its values.

On f failures,

Number of messages sent =

n messages(1st round) + (n -1) messages(2nd round) + ...

+ (n-f) messages(last round)

= n(f + 1) - f(f + 1) / 2

On f' failures(where f' < f),

Number of messages sent =

n messages(1st round) + (n -1) messages(2nd round) + ...

+ (n-f) messages(f' + 1round) + (n - f) messaged(f' + 2 round)

= n(f' + 2) - f'(f' + 3) / 2

Note that, in the last round it is n - f messages because the algorithm is trying to verify again with the active processes if their value is indeed the one agreed upon.

(b) Consider a simplified version of the early stopping algorithm: If the value v decided by a process p i does not change for two consecutive rounds, then p i can stop by deciding v. Will this variant work? Please prove by giving a justification or disprove by giving a counter-example.

Let each process Pi have initial value vi. Lets v1<v2<v3 and so on

Let us assume that in the first round, P1 sends to P2 and fails

and other processes communicate as usual.

At the end of this round, P2 will have v1 as its value and every other process has v2 as its value.

In the next round, P2 sends it value to everyone. Hence, all processes will now have v2 as the value. But here P2 has v2 as its value for two consecutive rounds, hence, it terminates.

In the subsequent rounds, there is no reply from P2 as it has terminated but other processes might assume it has failed. Hence, other processes do not terminate but continue. Hence, they dont stop after f' + 2 rounds but later than that.

Hence, the suggested variant does not work.

2. According to the problem there is only one source process broadcasting its value.

Let source broadcast its value in first round(without crashing else validity doesnt make sense):

Termination: All processes must eventually terminate

Agreement: All non-faulty processes must decide on same value. (which did not crash)

Validity: All non-faulty processes decide on the value proposed by the source(which did not crash).

The algorithm is of the type O(n^2). The source only needs to broadcast values and not worry about any malicious processes. Others dont broadcasr anything (dummy call)Afterwards, All simply just receive. Hence, the proposed value is received and updated by all non-faulty(active) processes.

Q (10 points). Consider the solution to BA problem for malicious processes using signed messages that we studied in the class, Algorithm SM (m) of SG book (page 285). Now answer the following:

(a) We saw in the class that the termination condition (#4) of this algorithm is incorrect. Can you please explain why?

(b) We discussed a corrected version of this algorithm in the class. One of the checks of the corrected version of the algorithm is this: in round r if the length of the SS (signature set) in message m received is less than r then the m is rejected. What is the reason for having this condition?

(c) Consider a simplification of the corrected algorithm discussed in the class. The simplification is that the algorithm will execute only two rounds as follows instead of f + 1: (i) In round 0, the source process will broadcast. (ii) In round 1, all the processes will broadcast to each other the message received in round 0 (while performing all the checks on a message received as mentioned in the algorithm).

Please explain why this simplification will not work with a counter-example.

3. a) Let us assume P0 is the source process which is faulty, and P50 is one of the correct processes. In the first round, If P0 sends a

message to every process except P50, then P50 thinks it as per 4th condition (termination condition) that it did not receive any more messages and decides to terminate with value as default value whereas every other correct process received the value from P0. In the next round, they will broadcast to every other process but by this time, process P50 has already terminated. Hence, even if the correct processes terminate in the next round, P50 is not active to receive the value.

Example: P0 broadcasts 5 to everyone other than P50. P50 terminated thinking no more messages with default value 0. All other correct processes terminate with value 5, later on.

We can then conclude with the above example that the agreement condition is violated.

b) It essentially serves as a check for checking the validity of the messages received. Let us say that the source is sending a message to another process. If P1 was one of the correct processes it would send it to everybody else. In the case that P1 was malicious, it might hold the message in the current round and send it later in the next round. Hence in the check introduced in the corrected algorithm, we treat this message as a corrupted message and reject it.

c) Consider the following scenario:

The commander is malicious, sending a message to some subset of the processes and doesn't send any message to the other subset of the processes where ideally the commander should send a message initially to n-1 processes. So in the current round, the processes which received value (vi) from command have vi while those which did not receive anything have the default value. Now on every subsequent round upto mth round, a non faulty process can receive a new message. Ending the algorithm in the second round, would not allow the correct processes to reach a valid consensus, that is, there might be some set of correct processes which have only the source value and some with just the default value where some faulty processes can decide to hold the messages. This is why we cannot end in two rounds as potentially the agreement condition might be violated.

Detailed example:

- P0 is commander and is malicious.
- P0 can send different messages to each other processes, that is, 1 to P1, 2 to P2 and so on
- Let P{1,m} be malicious and P{m+1, n} are correct processes
- P2 messages in round2, P3 messages in round3, …
- Hence in the worst case, processes keep receiving messages until round m

Q4 (10 points). Consider the oral messages algorithm discussed in the class, Algorithm OM (m) of

SG book (page 281) for the BA problem. This algorithm is also known as EIG algorithm. Please answer the following:

(a) The book mentions that this algorithm work for multi-valued inputs as opposed to binary values.

Please explain the reason.

Since, all non-faulty processes receive same set of values , their median must also be the same.

(b) Show the working of OM (m) algorithm with multi-value input as follows. There are 8 processes in the systems: p0…p7. Here processes p0 and p1 are malicious with p0 being the source process. Processes p2 to p7 are non-faulty processes. p0 broadcasts the following values to the different processes in the system - 5 to p1, 5 to p2, 6 to p3, 6 to p4, 7 to p5, 7 to p6 and 8 to p7. Show the working of this algorithm while showing how the processes the values decided by the non-faulty processes.

Q5 (6 points). Consider the Bitcoin network. What prevents a node in the network (or a group of

nodes) from falsely adding bitcoin to its account?

5. A node in the network cant falsely add bitcoins to its account. This is due to the fact that all the bitcoins are stored as entries in a public ledger shared by all the nodes. The only way to falsely add bitcoins is to write it down on their own ledger. But once he interacts or is involved in any transactions then the majority would deny his changes due to the fact they have their copy of the ledger. Consensus algorithm involved in bitcoin takes care that any changes made into the ledger are to be approved by the majority. It is also not possible for he/she to force the majority to approve his transactions due to the fact that bitcoin is pseudonymous.

Q6 (4 points). Suppose you are given a consistent snapshot S taken by Chandy-Lamport algorithm. Using this snapshot S, how will you detect if the system is deadlocked or not.

Let us assume that the model is a single resource model.

1. For every process create two sets, **wait** and **ask.**

   Construct **wait** such that all processes which are yet to release their resource for letting the current node access are added.

   Construct **ask** such that all processes which are waiting for the current node to release its resources.

2. For above construction, follow the following steps:

   Check every channel Cij (that is channel carrying messages from i to j).

   For all request messages, add Pi to j's **wait** and Pj to i's **ask.**

   For all grant messages, remove Pi from j's **ask** and Pj from i's **wait**.

3. Let us construct a graph such that each process represents a node. For every node to add edges:

   For all Processes x in **wait**, add directed edge to current node from x.

   For all Processes y in **ask**, add directed edge from current node to y.

4. Since the assumed model is a single resource model, any cycle in the graph implies a deadlock. Hence. check for cycles in the graph.

Essentially we are trying to construct a wait for graph and checking if any cycles exist in the graph. A cycle in WFG of single resource model is nothing but a deadlock.