The University of Texas
at Austin

CS371D Distributed System

Problem Set 01

Edited by LaTeX

Department of Computer Science

STUDENT

Jimmy Lin
xl5224

INSTRUCTOR

Lorenzo Alvisi

TASSISTANT

Chao Xie

RELEASE DATE

Feb. 8 2014

DUE DATE

Feb. 12 2014

TIME SPENT

20 hours

March 17, 2014

Contents

1 Problem 1: Snapshot Protocol

1.1 Problem Analysis

If we relax the FIFO assumption under the asynchronous system, the problems introduced are as follows:

- Messages in channel will reach destination process after the "take a snapshot" message (the marker message) is delivered to end up channel recording. That is, some messages may stay in channel over the whole snapshot collection period, such that we cannot collect complete channel state. Our strategy to tackle this problem is to delay the reporting of one process to the snapshot initiator, so as to reserve enough time waiting for the arrival of messages in channel.
- Messages sent after the delivery of "take a snapshot" message may arrive at destination process prior to it. That is, some redundant messages may be mistaken as component of channel state. Our strategy of dealing with this problem is to use timestamp.

1.2 Algorithm

The new snapshot protocol is designed as follows: here we choose process p_0 as a snapshot initiator.

- processor p_0 sends "take a snapshot to itself" with timestamp $TS(m)$
- if process p_i receives "take a snapshot" **for first time** from pj
  - records its local state σ_i
  - sends "take a snapshot" with timestamp $TS(m)$ along its outgoing channels
  - starts recording messages received on each incoming channel
  - stops recording a channel **after the delay time** τ
- when p_i receive "take a snapshot" with timestamp $TS(m)$ from **not for first time**
  - ignore all received messages with timestamp greater than $TS(m)$. That is, do not report these messages as part of channel state.
- when p_i stops recording all its incoming channel, it sends collected state to p_0 and then stops

2 Problem 2: Property Specification

2.1 Concepts Interpretation

Before providing solutions for these statement, we need first clarify the semantics of safety and liveness property:

- **Liveness**. Even liveness property does not hold now, it will hold in one future state.
- **Safety**. Once saftey property holds, that property holds for all prefix state.

2.2 Ultimate Answers

1. **Safety**. Once we know that another general will follow the attack decision within five minutes, such decision delay will hold in all previous global state. Thus, this is safety property. This is not liveness property because once another general follows the attack decision over 5 minutes, this property would never hold in the future.

2. **Safety**. If the statement that no general decide to attack holds, no general can decide to attack in the previous state. Hence, this is safety property. But this is not liveness property because once there is one general decides to attack, this statement cannot hold in any future state.

3. **A combination of safety and liveness**. If both generals will eventually decide on the same value, such tendency also holds for all previous states. Therefore, this is safety property. This is also liveness property because even though two generals do not tend to reach the same decision in one state, they may tend to reach the same decision since new decision is made and communication does not fail.

4. **No property at all**. The statement of "Not to attack is a possible decision" does not satisfy any property. Possibility of not attacking at the moment does not indicate the same possibility in the previous state. Impossibility of not attacking does not indicate possibility of not attacking in the future state.

5. **Safety**. This is safety property because one general in previous states can only have less number of messages. No more than 10 messages now indicates no more than 10 messages in the previous states. This is not liveness property because the general may send more than 10 messages in the future.

6. **Liveness**. This statement satisfy liveness property because the even the general B does send 5 messages now, it may sends at least 5 messages in the future. This is not safety property because general will send less messages in the past. Thus, the lower bound of number of messages may not be satisfied in previous states.

3 Problem 3

3.1 Under Bounded Loss Assumption

The algorithm is as follows:

- If one general g are ready to attack, and he does not receive "Yes" message from the other general before:
  - General g should send eleven "Yes" messages to the other general
  - General g decides to attack once it receives "Yes" message from the other general
  - General g decides "not to attack" if it receives "No" message from the other general
- If one general g are ready to attack, and he did receive "Yes" message from the other general before:
  - g should send eleven "Yes" messages to the other general. (acknowledgement)
  - General g make up its decision to "attack".
- If one general g are not ready to attack, and did not receive one "Yes" message from the other general
  - General g make up its decision "not to attack".
  - If received "Yes" message from the other general, g should send eleven "No" messages to the other general
  - If received "No" message from the other general, g should send eleven "No" messages to the other general as acknowledgement.

Proof for Agreement.

If general A is ready to attack, it should send 11 "Yes" messages to the other general. And general B must receive at least one message according to the assumption that at most 10 messages are lossy. Then if general B is ready, he should also send more than 10 "Yes" messages to general A. In this manner, general A is guaranteed to receive general B's decision. And both general decides to attack. (vice versa)

In all other cases, general A and general B will both decide not to attack based on the algorithm shown above.

Proof for Validity.

1. If both generals are not ready, each general will decide "not to attack" by himself without communicating with each other. In this case, no one decides to attack.

2. If both generals are ready, both of them will send 11 "Yes" messages to the other general. And it is guaranteed that each general will at least receive one "Yes" message. And since both of them are ready and did not send "No" messages to the other, neither generals will decide "not to attack" for sure.

In summary, since the point 1 and point 2 hold, the Validity Property holds for the provided algorithm.

Proof for Termination.

If one general are not ready, it will decide "not to attack" at the moment.

If one general are ready, it will send 11 "Yes" message and these messages will be received for sure. The receiver general will respond with 11 messages. If these messages are "Yes" , the sender general will decide to "attack", otherwise, it will decides "not to attack". Thus, since every general eventually make decision as shown above, the termination property for this algorithm holds.

3.2 Under Eventual Delivery Assumption

The difference of this question lies in the fact that each general has no prior knowledge about the number of messages that would be lost. We can first let two generals send redundant messages to fail the channel until it is reliable.

The algorithms designed are as follows:

- If one general is not ready to attack
  - keeps on sending "No" message to the other general
  - decide "not to attack" no matter what it received
- If one general is ready to attack
  - keeps on sending "Yes" to the other general
  - decide "not to attack" if "No" message is received
  - decide "attack" if "Yes" message is received

Proof for Agreement.

If both general A and general B are ready to attack, it will continuously send "Yes" messages to the other general. And general B must eventually receive at least one "Yes" message according to the eventual delivery assumption. General B follows the same analysis. Since general A and B will eventually receive "Yes" message and are both ready to attack, they will both decide to "attack". Therefore, both generals decide the same value if both are ready.

According to the algorithm, if one of two generals (say, general A) is not ready to attack, he will keeps on sending "No" messages to the other general (general B) and he will decide "not to attack". Even if general (B) is ready to attack, since it receives "No" message, it will also decide "not to attack". Hence, two generals decide the same value if either one is not ready.

If both generals are not ready to attack, they will stick to sending "No" message and directly decide "not to attack". Thus, two generals decide the same value if none of them is ready.

In summary, the given algorithm satisfies Agreement Property.

Proof for Validity.

1. If both generals are not ready, they will directly decide "not to attack" and telling the other general their decision. Hence, in this case, no general would decide "attack".

2. If both generals are ready, they both keep on sending "Yes" message to the other general and eventually they will receive "Yes" message sent by the other general. After the recept of "Yes" message, since they are both ready, neither of them will decide "not to attack".

In summary, since the point 1 and point 2 hold, the Validity Property holds for the provided algorithm.

Proof for Termination.

Apparently, the Termination Property holds for the algorithm. This is because no matter whether one general is ready or not, he will keep on sending message to the other general. The other general will eventually receive one of those message and make decision in terms of the recept message and its own readiness.

4 Problem 4

Proof:

$$VC(\Sigma_{min}^{\sigma_i^k}) = VC(\sigma_i^k)$$

By defintion, $\Sigma_{min}^{\sigma_i^k}$ is the earliest consistent state that σ_i^k (event k at process i) can belong to. Thus, we divide the entire proof into two parts. For the first part we show that $VC(\sigma_i^k)$ represents a consistent global state. On the second part , we show that each component in $VC(\sigma_i^k)$ is earliest.

4.1 $VC(\sigma_i^k)$ is earliest

First of all, we prove that **the global state represented by $VC(\sigma_i^k)$ is consistent**. Actually, we can refer to, for convenience, the conclusion made in class note that a vector clock of one event correspond to a consistent global state. But we still provide detailed proof by contradiction.

Assume that the global state represented by $VC(\sigma_i^k)$ is **not** consistent. That is to say,

$$\exists \sigma_j^{s^*}, \sigma_j^{s^*} \rightarrow \sigma_i^k, s.t. \tag{1}$$

$$\sigma_i^k \in C, \sigma_j^{s*} \notin C \tag{2}$$

According to the VC property, the happen-before relation above indicates

$$VC(\sigma_j^{s^*}) < VC(\sigma_i^k)$$

That is,

$$VC(\sigma_j^{s^*}) \neq VC(\sigma_i^k) \; and \; \forall x, VC(\sigma_j^{s^*})[x] \leq VC(\sigma_i^k)[x] \tag{3}$$

However, if we take an event at process j happening before σ_j^{s*} and belong to the consistent cut, let's call it σ_j^s. According the update rule of vector clock, we have

$$VC(\sigma_j^s)[j] < VC(\sigma_j^{s^*})[j] \tag{4}$$

Instantiate the formula (3), we have

$$(\sigma_j^{s^*})[j] \leq VC(\sigma_i^k)[j] \tag{5}$$

Hence, we have

$$VC(\sigma_j^s)[j] < VC(\sigma_j^{s^*})[j] \leq VC(\sigma_i^k)[j] \tag{6}$$

However, according to the property of consistent cut (see the class note), the event σ_j^s and σ_i^k are in one consistent cut and should satisfy

$$VC(\sigma_j^s)[j] \geq VC(\sigma_i^k)[j] \tag{7}$$

Obviously, the conflict lies in (6) and (7). Hence, we should negate the assumption proposed at the very beginning and conclude that **the global state represented by $VC(\sigma_i^k)$ is consistent**.

4.2 $VC(\sigma_i^k)$ is earliest

Then, we prove that **for arbitrary σ_i^k, those each local state in $VC(\sigma_i^k)$ is earliest among all possible ones consistent with σ_i^k.**

We prove this by mathematical induction on number of events E.

For base case of $E = 1$, there is only one event (say, at process i), the vector clock of that event

$$\forall j, 1 \leq j \leq n,\ VC(\sigma_i^1)[j] = \begin{cases} 0 & j \neq i \\ 1 & j = i \end{cases} \tag{8}$$

Obviously, the claim to be proven is true for the case of $E = 1$ since all local states are earliest.

For base case of $E = 2$, we consider two possible derivation according to the update rule of vector clock. One is local update rule, and the other is happen-before update rule. And third possibility is one independent event, regardless of the update rule.

- For local update rule, we have:

$$\forall j, 1 \leq j \leq n,\ VC(\sigma_i^2)[j] = \begin{cases} 0 & j \neq i \\ 2 & j = i \end{cases} \tag{9}$$

  Apparently, this vector clock is earliest.

- For happen-before update rule, we introduce a event e_k^1 at process k, $(k \neq i,\ 1 \leq k \leq n)$ s.t. $e_i^1 \rightarrow e_k^1$, and have:

$$\forall j, 1 \leq j \leq n,\ VC(\sigma_k^1)[j] = \begin{cases} 1 & j = i \ or \ j = k \\ 0 & otherwise \end{cases} \tag{10}$$

  By observation, this vector clock is earliest as well.

- For introducing an independent event, it is obvious that the vector clock of new event is still earliest. Here we omit the detail for convenience.

Now we can conclude that the claim to be proven is true for $E = 2$.

Generalization cases. We assume that for $E = k$, all components in $VC(\sigma_i^k)$ is earliest for any σ_i^k. And prove that for $E = k + 1$, the same property holds. Similarly, we only needs to consider newly introduced event. The detailed proof follows the generalization from $E = 1$ to $E = 2$, involving the case of two VC update rule and one independent case.

4.3 Sum Up

Since we have proven that the global state corresponding to $VC(\sigma_i^k)$ is consistent and also earliest reachable state, it is reasonable to conclude that

$$VC(\Sigma_{min}^{\sigma_i^k}) = VC(\sigma_i^k) \tag{11}$$