Distributed Systems Mid Term Paper

Duration - 50 mins

March 2, 2021

Instructions:

1. **Clearly state your roll number**
2. **Clearly state the value of X which is the last digit of your roll number**
3. **The paper is divided into four sections - Mutual Exclusion, Clocks, Global Snapshot and Presentations. Each section has multiple questions. At the beginning of each section; a function that uses X (value calculated by you above) has been clearly mentioned. Please evaluate the function to know the ONLY ONE question per section that you are to attempt. Attempting any other question in the section will allow for grading over only 50% marks on that section.**
4. **Write all the answers in blank sheets WITH PAGE NUMBERS. Scan the sheets and submit the pdf version**
5. **Any query needs to be typed out in the chat box. We will reach you back**
6. **If you run into issues submitting on moodle; then please mail the pdf to lini.thomas@iiit.ac.in before the end of the quiz slot. No submissions on moodle or over mail will be accepted after the time alloted.**

1 Mutual Exclusion

Attempt Question (X mod 3) + 1

1. (15 points) (a) (2 points)Consider the Suzuki Kasami Algorithm. Let process Pi be the process currently holding the token. Process Pj has broadcasted a request for the token. Can process Pj enter CS before Pj's request reaches Pi? Argue.

   (b) Consider the Raymonds Algorithm for ME. Answer the following questions.

   - (1 point) If the process holding the token is interested in entering CS again while holding the token; then, can he quietly just re-enter the CS? Argue.
   - (1 point) Can the process who has received the token and completed critical section(current root)at time t; find his own id on top of his queue before the token travels again to some other process ? Argue.
   - (1 point) Can the process who is executing the critical section (current root) have his own id also in his queue

   (c) (2 points) Consider the Maekawas Algorithm. Prof Z changes to conditions in the Maekawas algorithm slightly as can be seen below

   - for all i, j: Ri $\cap Rj \neq \emptyset$
   - for all i: i $\in$ Ri
   - for all i: $\|Ri\| = N$, where N is the total number of nodes in the system

- Any node i is contained in exactly N request sets.

For such a change what would be your feedback to Prof Z.

(d) (3 points) You are tasked with designing a new distributed mutual exclusion protocol based on voting. You come up with the following solution:

- A node requests permission (votes) from other nodes before proceeding to execute its critical section.
- The node does not proceed unless it receives a majority of replies from other nodes.

What are some problems with this solution?

(e) (5 points) Prof A and Prof B suggested modifications to the Suzuki Kasami algorithm for a process P that has the token (just finished a critical section) and is interested to re-enter the critical section. Assume that when this happens the token queue is not empty.
Prof A's suggestion: Process P puts itself on token queue and then sends the token to the next process on the queue.
Prof B's suggestion: P executes its second CS and then sends token to the next process in the token queue

- Mention why you think each of Prof A and Prof B felt it was worth including their suggestion
- Comment on whether you feel both the modifications give the desired output. If further modifications are required for the idea to work then mention so.

2. (15 points) Consider the below Sarangai-Weiss Algorithm. This algorithm involves counter values. A counter if sent is either sent immediately on receiving a REQUEST or after the critical section based on certain conditions. A counter sent after critical section is called C_Counter and the one sent immediately on seeing a request is called A_Counter.
Initial values of A_Counter and C_Counter = 0 for all sites.
Also note that If two processes have the same time stamp then the tie is broken by using the process id.

Requesting the critical section:
1.When a site S_i wants to enter the CS, it broadcasts a time stamped REQUEST message to all other sites.
2.When site S_k receives a REQUEST message from site S_i, it sends a A_Counter with value 1 to site S_i if
- site S_k is neither requesting nor executing the CS,
- or the site S_k is requesting and timestamp(S_i) < timestamp(S_k)
Else
the decision of replying or not is deferred to after the critical section.

Executing the critical section:
Site S_i enters the CS when the summation of all Counter values received (of both A_Counters and C_Counters together) adds to exactly n-1 (where n is the total number of sites). Note: It need not receive a Counter from every site. As soon as the sum of received counters sums to n-1; he can enter the CS.

Releasing the critical section:
When site S_i exits the CS, it does the below in sequence:
- If he has not received a C_Counter from any site then he sets his C_Counter = 1 else he sets his C_Counter as the received C_counter value incremented by 1.
- sends the C_Counter to a single process which is the next process with lowest time stamp.
- Resets his C_Counter = 0

Algorithm Idea: Consider the sites interested in entering the critical section ordered by increasing timestamp be P1, P2, P3, ... Pi. (If two processes have the same time stamp then the tie is broken by using the process id). According the algorithm the site P2 will be waiting only on the C_Counter of the site P1 to enter CS as every other site would have given its permission using A_Counter = 1. The site P2 after its CS; will let the site P3 know and give it the combined permission of itself and P1 with C_counter = 2 and so on.

Questions:

(a) (3 points) What is the advantage of such an algorithm?

(b) (8 points) Does this algorithm satisfy Mutual Exclusion, fairness, no-deadlock and no-starvation? Explain.

(c) (4 points) What would be any VALUABLE feedback you have about this algorithm?

3. (15 points) Consider the below Maria Seene Modified Algorithm. This algorithm involves counter values. A counter if sent is either sent immediately on receiving a REQUEST or after the critical section based on certain conditions. A counter sent after critical section is called C_Counter and the one sent immediately on seeing a request is called A_Counter.
Initial values of A_Counter and C_Counter = 0 for all sites.
Also note that If two processes have the same time stamp then the tie is broken by using the process id.

Requesting the critical section:
1.When a site S_i wants to enter the CS, it broadcasts a time stamped REQUEST message to all other sites.
2.When site S_k receives a REQUEST message from site S_i, it sends a A_Counter with value 1 to site S_i if
- site S_k is neither requesting nor executing the CS,
- or the site S_k is requesting and timestamp(S_i) < timestamp(S_k)
Else
the decision of replying or not is deferred to after the critical section.

Executing the critical section:
Site S_i enters the CS when the summation of all Counter values received (of both A_Counters and C_Counters together) adds to greater than or equal to n-1 (where n is the total number of sites). Note: It need not receive a Counter from every site. As soon as the sum of received counters sums greater than or equal to n-1; he can enter the CS.

Releasing the critical section:
When site S_i exits the CS, it does the below in sequence:
- If he has not received a C_Counter from any site then he sets his C_Counter = 1 else he sets his C_Counter as the received C_counter value incremented by 1.
- sends the C_Counter to a single process which is the next process with lowest time stamp.
- Resets his C_Counter = 0

Questions:

(a) (2 points) What is the advantage of such an algorithm?

(b) (5 points) Does this algorithm satisfy Mutual Exclusion, fairness, no-deadlock and no-starvation? Explain.

(c) (5 points) Any issues you see with the algorithm and any comments on how you would modify it if you could.

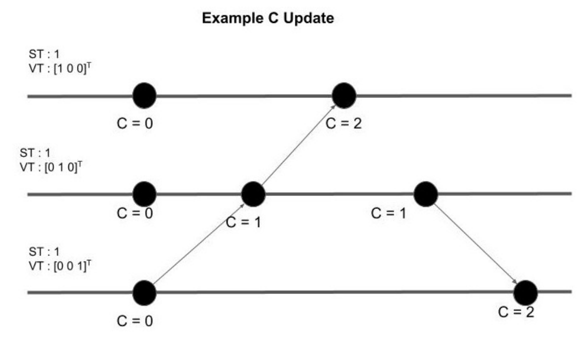
(d) (3 points) Compare and comment on this algorithm compared to the ME algorithms you have studied

2 Clocks

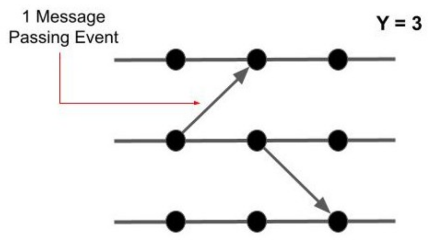
Attempt Question (X mod 3) + 2. If it exceeds 3 then deduct 3

1. (10 points) Kshemkalyani's differential technique for vector clocks:
   - (a) (3 points) Explain the construction of the extra vectors used in this technique.
   - (b) (2 points) What is the message passing condition? Justify.
   - (c) (4 points) Explain how it helps in decreasing storage than the naive method used for reduced message passing.
   - (d) (1 point) What is the limitation of this method?

2. (10 points) Sheldon was trying to find a manner to get the scalar time of an event from its vector time. This was not possible with raw vector clock implementation. So he figured out that only if he had an extra integer C along with the vector time (vt), he would get the scalar time as a function of the vector vt and C.
   The extra term C has its initial value set to 0 for all processes and has the following update rule for a message send event from process Pi to Pj:
   Ci := Ci
   Cj := Cj + Ci + 1
   Then according to Sheldon for a process Pi, the transformation from vector to scalar time would be:
   Scalar Time = $\sum_{j=1}^{n} vt_i[j]$ - C
   Is Sheldon correct? Justify your stance. (Assume step size d = 1 and the logical times start with 1, See example image for better understanding)



3. (10 points) Numericals on logical clocks:
   - (a) (5 points) There are 3 processes having K number of events each where K = X mod 3 + 4. Now in this system 3*(K-1) total number of message passing events are involved (one message pass event constitutes a message sent event by a process and a message receive event by another process). Find the minimum and maximum value of scalar time that is possible and show the order of message passing with a diagram. Take d = 1 and initial scalar time for the first event of each process to be by default 1.



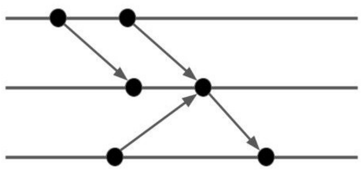
Example explanation for 2(a)

(b) (5 points) For vector clock with step size as d (which is given by d = X mod 2 + 2) , what is the relation between the vector times and the total number of events causally preceding it. Assume that the number of processes are n and the logical time begins with 1 for each process. For a given vector time vt_i you can consider the number of zeros in it to be k_i. For example for $vt_i = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$, k_i = 2 as vt_i has 2 zeros in it.

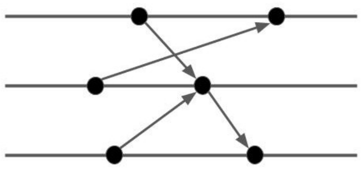
3 Global Snapshot

Attempt Question (X mod 3) + 3. If it exceeds 3 then deduct 3

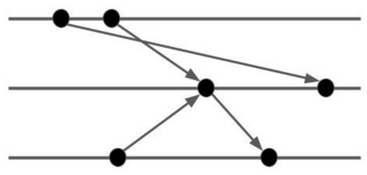
1. (10 points) Consider two cuts whose events are denoted by C1=C1(1),C1(2),.... ,C1(n) and C2= C2(1),C2(2),.... ,C2(n) respectively. (Note: In the above, Ci(j) for integers i and j refer to the local time that cut Ci intercepts the timeline of processor Pj). Define a third cut,C3= C3(1),C3(2),.... ,C3(n), which is the minimum of C1 and C2; that is, for every k, C3(k) is the earlier of C1(k) and C2(k). Then
   (a) will cut C3 be a consistent cut ? Prove or disprove.
   (b) will cut C3 be consistent if C1 and C2 are consistent
   (c) What can you say if we have k cuts C1 through Ck?
2. (10 points) Snapshot Algorithms:
   (a) (6 points) For the following systems state all the global snapshot algorithms (out of Chandy Lamport, Lai Yang and Acharya Badrinath) that could be applied with proper justification (You can safely assume all the algorithms to work in case of a lack of information)



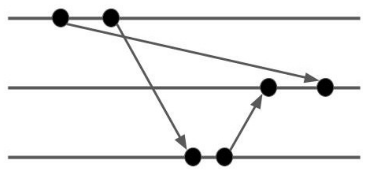
- i)



- ii)



- iii)



- iv)

(b) (4 points) Consider a distributed system where every node has its physical clock and all physical clocks are perfectly synchronized. Give an algorithm to record global state assuming the communication network is reliable. (Note that your algorithm should be simpler than the Chandy–Lamport algorithm.)

3. (10 points) Consider two cuts whose events are denoted by C1=C1(1),C1(2),.... ,C1(n) and C2= C2(1),C2(2),.... ,C2(n) respectively. (Note: In the above, Ci(j) for integers i and j refer to the local time that cut Ci intercepts the timeline of processor Pj). Define a third cut,C3= C3(1),C3(2),.... ,C3(n), which is the maximum of C1 and C2; that is, for every k, C3(k) is the later of C1(k) and C2(k). Then
   (a) will cut C3 be a consistent cut ? Prove or disprove.
   (b) will cut C3 be a consistent if C1 and C2 are consistent
   (c) What can you say if we have k cuts C1 through Ck?

4 Presentations

Attempt Any 1 Question of your choice

1. (8 points) Fill in the blanks (3 points):
   (a) In Birman Schiper Stephenson Algorithm, all messages are _________ messages.
   (b) In Spezialetti-Kearns algorithm, each process keeps track of the initiator using the _________ variable.
   (c) Alagar-Venkatesan algorithm requires _________ time units and _________ * n messages. (n is #processes).

   State True or False with explanation (5 points):
   (a) Spezialetti-Kearns algorithm will work for Causal Channels.
   (b) In Spezialetti-Kearns algorithm, the number of regions might be less than the number of concurrent initiators.
   (c) Message complexity of Singhal's dynamic information structure algorithm can be 5/4 * (n - 1) where n is total number of sites.
   (d) In Singhal's Dynamic Information-Structure Algorithm, the REQUEST and REPLY code sections cannot run concurrently as they both try to modify the logical clock.

(e) Singhal's Dynamic Information-Structure Algorithm will work for non-FIFO channels.

2. (8 points) Fill in the blanks (3 points):

   (a) In Spezialetti-Kearns algorithm, _________ variable of a process contains the identifiers of the neighbouring regions.

   (b) In Mattern's algorithm for Non FIFO channel, _________ is used to detect that no white messages (sent before snapshot) are in transit.

   (c) The synchronisation delay in Singhal's dynamic information structure algorithm is ____ * T.

   State True or False with explanation(5 points):

   (a) Mattern's Algorithm will work for FIFO channels.

   (b) Lai-Yang algorithm uses more memory as compared to Mattern's Algorithm.

   (c) A quorum in Agarwal El-Abbadi Algorithm consists of all processes in the path from root to a leaf excluding the root.

   (d) In Agarwal El-Abbadi algorithm, if a non-leaf node p is unresponsive, permission can be asked from all processes on two paths instead: from each child of p to some leaf.

   (e) Agarwal El-Abbadi algorithm can guarantee formation of a quorum even if number of site failures is between 2 * logn and 3 * logn where n is total number of sites.