KOÇ UNIVERSITY

COMP415/515: Distributed Computing Systems
Spring 2019
Self-Study Homework #2

Important Notes:

- This is a self-study homework i.e., it does not have any due date, no submission required, and hence it is not going to be graded.

- The purpose of this self-study homework is for you to get some idea about the subsequent quiz (also, midterm exam) questions, and being prepared for them accordingly. Therefore, it is highly recommended to devote your time and attention on solving the problems. Although there is no submission requirement for this self-study homework, you are strongly encouraged to deal with that entirely within the week of the announcement time.

Questions:

These questions are mainly from the course textbook, Distributed Systems: Principles and Paradigms, Andrew S. Tanenbaum, Maarten van Steen, **2nd edition** (the pdf is available on the course site).

Chapter 6: Questions: 6, 8, 10 and 14.

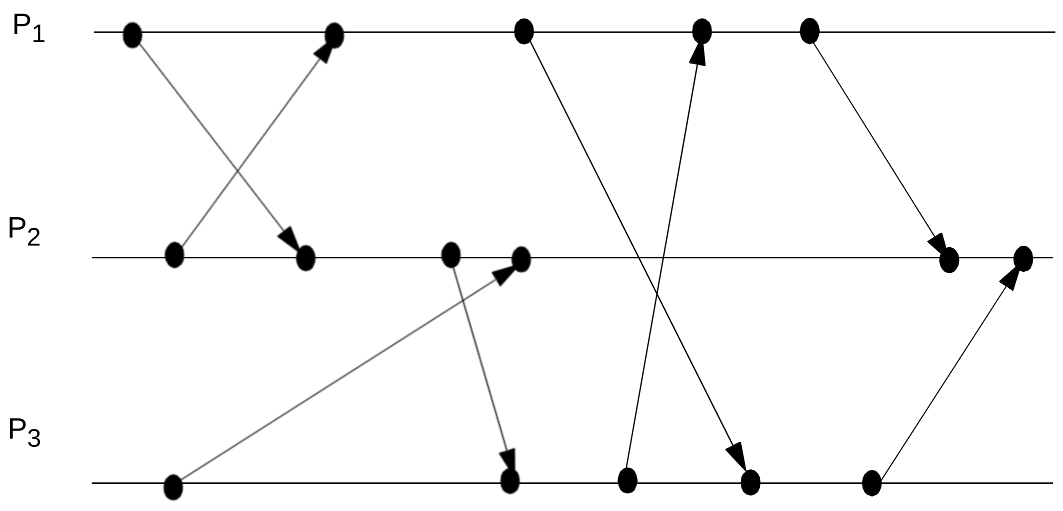
Chapter 4: Questions: 7 and 9.

Question:

The diagram below shows three processes that communicate via messages.

a) Consider **vector timestamps**. Assume that the clock on each system is initially zero (0) and is incremented by 1 for each event. Give the vector timestamp of each event.

b) Indicate each event that is concurrent with the last event at P3. Explain the reason.



COMP415/515: Distributed Computing Systems
Spring 2019
Self-Study Homework #2

Question:

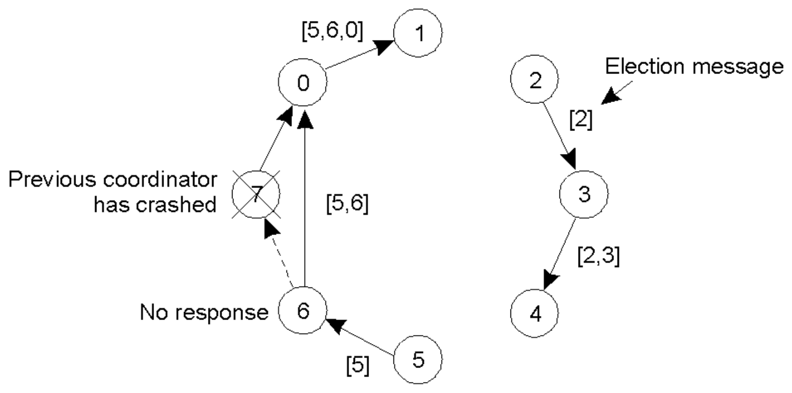
When using **vector timestamps** for enforcing **causally ordered multicasting**, $V_i[i]$ is incremented only when process P_i sends a message, and sends V_i as a timestamp ts(m) with message m. How should we interpret the following two conditions for delivering m when received by process P_j ? Explain your answer.

$$ts(m)[i] \equiv V_j[i]+1.$$
$$ts(m)[k] \leq V_j[k] \text{ for } k \neq i.$$

Question:

Consider the **ring algorithm for distributed election**. Recall that, concurrent elections initiated by different nodes are allowed by the algorithm. As shown in the figure below, there may be multiple **Election** messages circulating simultaneously.

While it does no harm to have multiple elections, it would be more efficient to make sure that only a single election is hold in all cases (i.e. if there happens multiple elections, all but one is cancelled properly). Propose an algorithm for doing this without affecting the operation of the ring election. Explain how your algorithm makes sure that only a single election is hold in all cases.


[5,6,0]
1
0
2
Election message
[2]
Previous coordinator has crashed
7
[5,6]
3
[2,3]
No response
6
4
[5]
5