Name:Hanliang Jiang
NetID:hj33

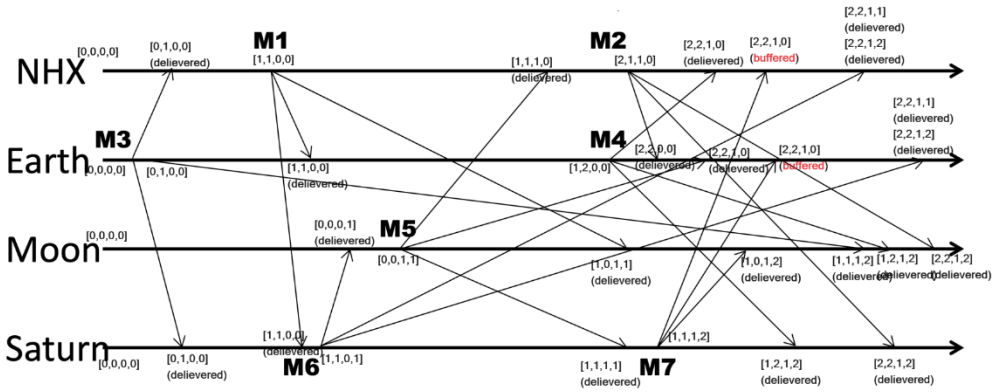
Problem 1:

The proof in a synchronous system has rounds, but in an asynchronous system, as the message transportation time can be arbitrarily long, we cannot decide a proper round time.

Name:Hanliang Jiang

NetID:hj33

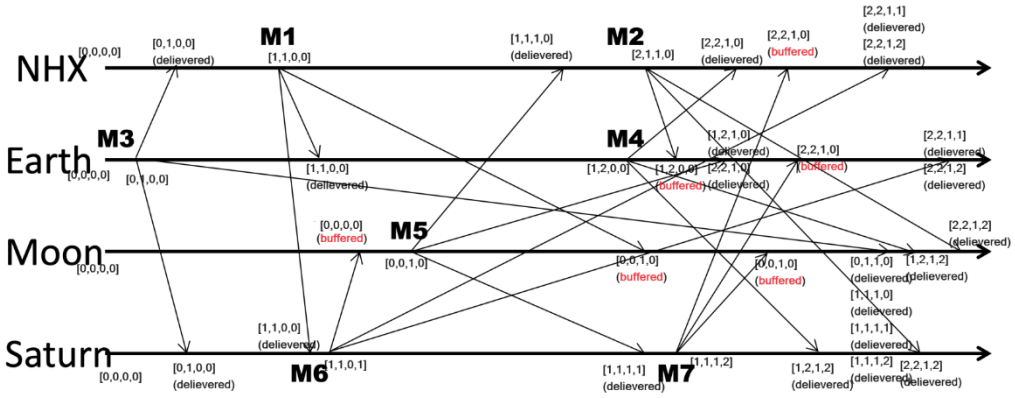
Problem 2:



Name:Hanliang Jiang

NetID:hj33

Problem 3:



Name:Hanliang Jiang
NetID:hj33

Problem 4:

(i) The fact that p=p' doesn't mean e=e'. In fact, if p=p' and m does not equal m', then e!=e', because process p may receive a different message m', which makes e different from e'.

(ii) If C0 never decides, then the system is in a condition where consensus is not solved, and that finishes the whole proof that consensus cannot be solved in an asynchronous system.

(iii) That's true, e can be random selected as long as p doesn't crash and e is applicable

Name:Hanliang Jiang
NetID:hj33

Problem 5:

| | |
|---|---|
| NHX: process state: f | C(Moon, NHX) = <>, C(Earth, NHX) = <R(e)> |
| Earth: process state: c | C(NHX, Earth) = <>, C(Moon, Earth) = <> |
| Moon: process state: b | C(NHX, Moon) = <S(e), R(f)>, C(Earth, Moon) = < S(e), R(f), R(d)> |

Name:Hanliang Jiang
NetID:hj33

Problem 7:

Example: as the system is asynchronous, message transportation can be arbitrarily long. Let's say the node with highest value N80 is alive, but the highest but one node N32 cannot connect to N80, so after timeout, N32 claims N80 dead, but N80 is still the leader for other nodes and N32 also performs as leader for other nodes, and therefore there's conflict.

Besides, N32 cannot decide a proper timeout for N80 since message transportation can be arbitrarily long, so N32 may wait permanently for N80.

Name:Hanliang Jiang
NetID:hj33

Problem 8:

For example, pi, pj, pk are in the same row in a big matrix and pj is inside the critical section, and pi and pk are both waiting in queue. After pj exit(), pj sent release to all processes in his row and column. Each processes in that row reply to both pi and pk, thus pi and pk enter the critical section simultaneously.

Name:Hanliang Jiang
NetID:hj33

Problem 9:

(I) True. It's okay if each of N nodes leave the view separately.

(II) False. Each process in the view should see exactly the same view, so the next view for p2 should be {p1, p2}

(III) True. If the process does not deliver a multicast in the view while other processes do, then it will be forcibly removed from the next view.

(IV) False. Even though the node failed, the message it sent should be considered, and the message's delivery should not violate the conditions

(V) False. It violates the condition that the set of multicasts delivered in a given view is the same set at all correct processes that were in that view