# Exercise 2

### Exercise 2.3

2.3 (a)

Yes, it implements Uniform Causal Order because it says these lines:

*[Partial algorithm from slides]*

p has prev. executed deliverCA(m’) ∀ m’ ∈ D  
then  
deliverCA(m)  
prevDlvrs := prevDlvrs ∪ {m}  
else  
discard m

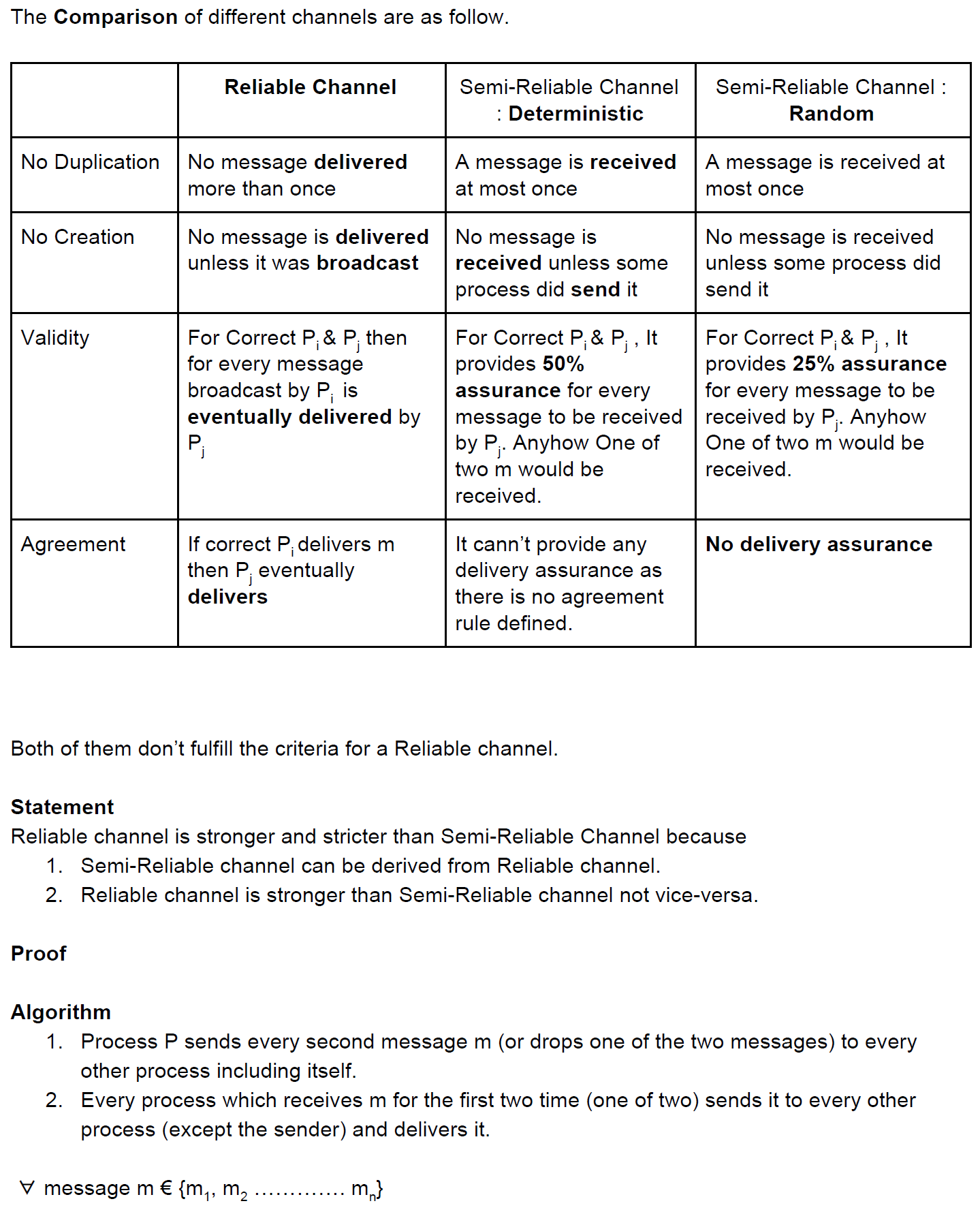
That tells if m' is delivered before m then all other process has to deliver m' before m and if it does not happen it discards m. We can say m' has caused m in the above FIFO atomic broadcast.

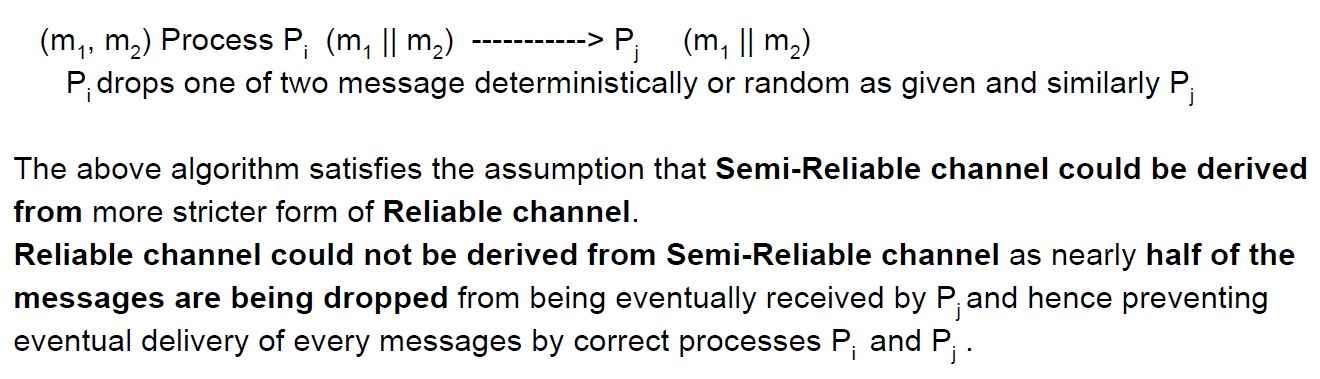
(b)

No, because Uniform agreement(V) says if a CORRECT or NOT process delivers a message m, every correct process eventually delivers m. So for the wrong process it does not accept total order(XIII) i.e a wrong process can deliver m2 before m1 but in actual m1 should be delivered before m2.

We can simplify it if we consider only causal ordering where m1 causes m2 or use consensus.

## Exercise 2.4





### Exercise 2.5

(a)

p1: a,1; b,2; c,3; d,5; e,6; f,7;

p2: g,1; h,2; i,3; j,6; k,7; l,8;

p3: m,1; n,3; o,4; p,5; q,6; r,10;

p3: s,4; t,5; u,5; v,7; w,8; x,9;

(b)

p1[0,0,0,0]: a,[1,0,0,0]; b,[2,1,0,0]; c,[3,1,0,0]; d,[4,1,3,0]; e,[5,1,3,0]; f,[6,1,3,0];

p2[0,0,0,0]: g,[0,1,0,0]; h,[0,2,1,0]; i,[0,3,1,0]; j,[3,4,1,2]; k,[3,5,5,2]; l,[6,6,6,2];

p3[0,0,0,0]: m,[0,0,1,0]; n,[1,0,2,0]; o,[1,0,3,0]; p,[1,0,4,0]; q,[1,0,5,0]; r,[5,3,6,6];

p3[0,0,0,0]: s,[3,0,0,1]; t,[3,0,0,2]; u,[3,3,1,3]; v,[5,3,3,4]; w,[5,3,4,5]; x,[5,3,4,6];

(c)  
No, there is no causal because as per vector time stamp, NONE of the message time stamp is less than local time vector.

(d)   
C1 is inconsistent as v is in cut but e is not where as e has caused the v, so it voilates the consistent cut definition.

(e)  
Ca and Cb are consistent because the cause of the event is in the cut.[refer d.png]

(f)

The algorithm is not fault tolerant because it has no termination detection. One can not d[**istinguish**](http://www.dict.cc/englisch-deutsch/distinguish.html) between.

* message dropped from network
* server is down
* timestamp s is not yet exceeded

With server the singleton initiator process oft the Mattern algorithm is ment.

(g)

You can do three things as described in: An introduction to snapshot algorithms in distributed computing (1995):

1. Let the initiator (server) be a virtual process instead of a physical one
2. Remove the unnecessary ACKs with incrementing s so that no „dummy-message“ is needed
3. Replace the time with a binary value. Because only a binary value is needed to determine wheter a report is needed.

All in all you also need a termination protection and a recovery routine to handle the case that the server is terminated.