**17/06/19**

1. **Auth p2p, f byzantine in each neighbourhood (f=1), only s can brd,** 
   * + 1. If the sender is byzantine it can make correct processes to deliver its messages. If s isn’t byzantine, then even with f=1 it can send a fake message to its neighbours (always >= 2) and then achieve condition of third function and repropagate the fake message through all the network.
       2. Yes
       3. It depends on byzantines and also on velocity in receiving messages, the check on sequence numbers is done on the last received, so if s sends a message m with sn = n and m’with sn = n+1, but last[src] = n – 1 it will accept m’ before m if it arrives earlier.

**1-**

M1, m4, m5

M1, m3, m2, m5

M2, m3

M1, m2, m3, m5

**2-**

M1, m4, m5

M1, m3, m2, m4, m5

M2, m3

M1, m2, m3, m4, m5

**3-**

Fifo constraints (only for correct): m1 -> m2, m5->m3

Causal: m2 -> m3

M1, m4, m5

M1, m5, m2, m3, m4

m3, m2

M1, m5, m2, m3, m4,

**3.**

**1-**

Read1() -> 0,1,2

Read2() -> 1,2

Read3() -> 1,2,3

Read4() -> 1,2,3

Read5() -> 2,3

**2-**

Read1() -> 0,1,2

Read2() -> if 1->0,1 => 1,2, otherwise 2

Read3() -> if 2-> 2 => 2,3, otherwise 1,2,3

Read4() -> if 1-> 2 => 2,3, otherwise 1,2,3

Read5() -> if 4-> 3 => 3, otherwise 2,3

**3-**

No, read1 precedes read2, so read2 cannot print 1 after read1 printed 2.

**4.**

**Init**

Correct = PI

Left = get\_left()

Right = get\_right()

Proposals = []N

Receivedfrom = []N

**Upon event Crash<p> do:**

Correct = correct \ {p}

**Upon event new\_leader<p> do:**

Leader = p

**Upon event new\_left(p) do:**

Left = p

if left != null

**trigger** FIFO\_send(proposals[r],r) to left

if right != null

**trigger** FIFO\_send(proposals[r],r) to right

**Upon event new right(p) do:**

Right = p

if left != null

**trigger** FIFO\_send(proposals[r],r) to left

if right != null

**trigger** FIFO\_send(proposals[r],r) to right

**Upon event Propose(v) do:**

Proposals[1] = proposals[1] U v

if left != null

**trigger** FIFO\_send(PROPOSE, <proposals[1],1,self>) to left

if right != null

**trigger** FIFO\_send(PROPOSE, <proposals[1],1,self>) to right

**upon event FIFO\_deliver(PROPOSE, <ps,r,s) from p do**

receivedfrom[r] = receivedfrom[r] U {p}

proposals[r] = proposals[r] U ps

**upon correct in receivedfrom[r] and decision = null do**

if proposals[r] = proposals[r-1]

decision = min(proposals[r])

if left != null

**trigger** FIFO\_send(DECIDED, decision) to left

if right != null

**trigger** FIFO\_send(DECIDED, decision) to right

else

round = round + 1

if left != null

**trigger** FIFO\_send(PROPOSE, <proposals[r-1],r>) to left

if right != null

**trigger** FIFO\_send(PROPOSE,<proposals[r-1],r>) to right

**upon event FIFO\_deliver(DECIDED, v) such that p in correct and decision = null**

decision = v

if left != null

**trigger** FIFO\_send(DECIDED, decision) to left

if right != null

**trigger** FIFO\_send(DECIDED, decision) to right