**Pg.8 ex.2**

**1)**

Causal: m1->m2, m1->m4

TO: agreement, order

M1, m2, m4, m3

M1, m2, m3, m4

M1, m3, m2, m4

M1, m4, m2, m3

M1, m4, m3, m2

M1, m3, m4, m2

**2)**

Causal: m2->m1 or m4->m1

TO(UA, WNUTO) -> P3 sequence can be differently ordered.

Same sequences putting P2 or P4 before P1

**Pg.8 ex.3**

**Total order:**

* Properties: Validity, Integrity, Agreement, Order
* Types:
  + Agreement: UA, NUA
  + Order:
    - Uniform: SUTO,WUTO
    - Not uniform: SNUTO, WNUTO

**TO(NUA,SUTO) and not TO(UA, SUTO):** sequencies with same order but different deliveries for crashed processes.

**TO(UA,WUTO) and not TO(UA,SUTO):** suto wants same prefix:

M1,m2,m3

M1,m3 crash

**Pg.9 Ex.3**

**1)** TO(NUA,SNUTO)

**2)**

M1, m2, m3, m4, m5 for corrects

M1,m2,m4 for faulty

**3)**

M1, m2, m3, m5, m4 for corrects

M1, m2, m4, m5 for faulty

**Pg.9 Ex.2**

**TO but not causal order**

**Fifo:** m3->m4

**Local:** m2->m4,m1->m2

M1,m2,m4,m3

**Pg.11 Ex.2**

Fll, pfd => Uniform reliable broadcast: Validity, no duplication, no creation, uniform agreement.

**Init**

Correct = PI

Pending = empty

Ackm = empty (for every message)

Delivered = empty

**Upon event Broadcast(DATA, s,m) do**

For all p in correct:

trigger fll\_send(DATA,s,m) to p

**upon event fll\_deliver(DATA, s,m) do**

if <m,s> not in pending:

pending = pending U {<m,s>}

ackm = ackm U {self}

for all p in correct:

for all <m,s> in pending:

trigger fll\_send(DATA,m,s) to p

**upon exist <m,s> in pending such that correct incluso in ackm do**

if m not in delivered:

delivered = delivered U m

trigger Deliver(DATA, m)

**upon event Crash(p) do**

correct = correct \ {p}

**27/1/2021**

**Pg.11 ex.2**

Flp2p, pf

**Init**

Correct = PI

Sent = [empty]N

Pending = empty

Ackm = [] (for every message)

delta

**Upon event URBcast(DATA,m,s) do**

Forall p in Correct do

**Trigger flp2pSend(DATA, m, s) to p**

Senti = Senti U {m}

Starttimer(delta)

**Upon event flp2pDeliver(DATA, m) from pi do**

If m is not in pending, then

Pending = pending U {m}

ackm = ackm U {pj}

**upon timeout**

timer = delta

forall m in senti do

Forall p in Correct do

**Trigger flp2pSend(DATA, m) to p**

**upon event Crash(p) do**

correct = correct \{m}

Forall mm in Sentp do

**Trigger URBCast(DATA, mm)**

**upon correct incluso in |ackm| do**

if m not in deliveredi then

**trigger URBDeliver(m)**

deliveredi = deliveredi U {m}

**pg.12 ex.4 Consensus, ring, pp2p**

**Init**

V= null

Nexti = p(i+1) mod n

Propose[p] = []N

**Upon event Propose(PROPOSAL,v,s) do**

V = value

**Pp2pSend(‘PROPOSAL’,v,s) to next**

**Upon event pp2pDeliver(‘PROPOSAL’,v,s) from p**

If s == self

**Trigger Pp2pSend(‘DECIDE’,v,s) to next**

If s != self

Propose[self] = v

**Trigger Pp2pSend(‘PROPOSAL’,v,s) to next**

**Upon event pp2pDeliver(’DECIDE’,v,s) from p and decision == NULL do**

If s==self

Dec = min(propose)

**Trigger Decide(Dec)**

Decision = Dec

If s!=self

**Trigger Pp2pSend(‘DECIDE’,v,s) to next**

**Pg.13 ex.4 TOBRD, LEADER, RB**

**1)**

**Upon event Init(TOBrd) do**

Delivered = empty

Unordered = empty

Ordered = empty

Wait = false

**Upon event TOCast(DATA,m) do**

**Trigger RBCast(DATA,m)**

**Upon event RBDeliver(DATA,m) from p do**

if m not in delivered and m not in unordered then

unordered = unordered U {m}

**upon unordered != empty and wait = false do**

wait = true

if Leader(self) = true then

ordered = sort(unordered)

**trigger RBCast(LIST, ordered)**

**upon event RBDeliver(LIST,ordered) from p do**

for all m in ordered do

**trigger TODeliver(m)**

delivered = delivered U m

unordered = empty

ordered = empty

wait = FALSE

**2)** no because only delivery is considered, not also sending. We should consider local timestamps.

**3)**

**Upon event Crash(p) do**

correct = correct \{p}

**If p is Leader then**

trigger Leader(maxrank(Correct))

**pg.15 ex.4**

**fifo rb, flp2p, line, p-1, p+1, no crashes.**

**Init**

Lsn = 0

Pending =empty

Next[p] = [1]N

Delta = T

Senti = empty

**Upon event fifoRBCast(m) do**

lsn = lsn + 1

**trigger flSend(self, m,lsn) to pi+1 if exists**

**trigger flSend(self, m,lsn) to pi-1 if exists**

senti = senti U {self, m, lsn}

starttimer(delta)

**upon timeout do**

for all {s,m,sn} in senti do

**trigger flSend(self, m,sn) to pi+1 if exists**

**trigger flSend(self, m,sn) to pi-1 if exists**

starttimer(delta)

**upon event flDeliver(s,m,sn) from pj do**

**trigger flSend(s, m, sn) to pi+1 if exists**

**trigger flSend(s, m, sn) to pi-1 if exists**

pending = pending U (s,m,sn)

if exists (s,m,sn) in pending such that sn = next[s] do

next[s] = next[s] + 1

pending = pending \ {s,m,sn}

**trigger fifoRBDeliver(s,m)**

**Graphical user interface, application

Description automatically generated**

1.

**Validity (for beb):** if a correct process brd a message, then every correct process eventually delivers it. => not respected, if a received message is not the “next” it will be discarded and not delivered (need of a pending array).

**Validity (for rb):** if a correct process p brd a message, then p eventually delivers it. => same, we should add a deliver just after the first send.

**No duplication:** no message delivered twice => respected

**No creation:** if a message has sender s, then s has previously sent it. => respected straight by perfect p2p.

**Agreement (for rb):** if a message is delivered by a correct process, then every correct will deliver it. => not respected, since validity is not respected.

NONE OF BEB OR RB ARE RESPECTED.

2.No ordering is respected.

1.

**Validity (for beb):** if a correct process brd a message, then every correct process eventually delivers it. => y.

**Validity (for rb):** if a correct process p brd a message, then p eventually delivers it. => y

**No duplication:** no message delivered twice => y

**No creation:** if a message has sender s, then s has previously sent it. => y

**Agreement (for rb):** if a message is delivered by a correct process, then every correct will deliver it. => no, no crash manage.

ONLY BEB IS RESPECTED.

Text

Description automatically generated

**Upon event Init() do**

Right = pi+1

Left = pi-1

Correct = N

Value = 0

Ackv = []N

**Upon event write(v) do**

**Trigger** pp2pSend(VALUE, v, self) to right if exists

**Trigger** pp2pSend(VALUE, v, self) to left if exists

**Upon event pp2pDeliver(VALUE, v, s) from p do**

Value = {v,ts}

If p = right do

**Trigger** pp2pSend(‘ACK’, self, s) to right if exists

**Trigger** pp2pSend(VALUE, v, s) to left if exists

If p = left do

**Trigger** pp2pSend(‘ACK’, self, s) to left if exists

**Trigger** pp2pSend(VALUE, v, s) to right if exists

**Upon event pp2pDeliver(‘ACK’, p, s) do**

If s = self do

Ackm = Ackm U {p}

If |ackm| = N do

**Trigger** WriteReturn(v)

Ackm ­= empty

Else if p = right do

**Trigger** pp2pSend(‘ACK’, p, s) to left if exists

Else

**Trigger** pp2pSend(‘ACK’, p, s) to right if exists

**Upon event read() do**

**Trigger** ReadReturn(v)

**Pg.6**

**Fifo:** m3->m4

**Local:** m1->m2, m2->m4

M1,m2,m3,m4

M1,m3,m2,m4

M3,m1,m2,m4

1. Causal order
2. Fifo not causal

Fifo: m1->m2

Local: m1->m2, m2->m4

M1->m3 already delivered

1.

M1, m2, m3, m4

M1, m3, m2, m4

M1, m2, m4, m3

2.

P1: M1,m3,m4,m2

P2: M1, m2,m4,m3

P3: M3,m1,m2,m4

1. Total and causal
2. Total not causal
3. Fifo no causal nor total

Fifo: m3->m4

Local: m2->m4, m1->m2

M3->m4 already delivered by p4

M1->m2->m4

1.

M1,m2,m3,m4

M1,m3,m2,m4

M3,m1,m2,m4

2.

M1,m3,m4,m2

3.

M1,m3,m4,m2

**Pg.7**

1. NURB
2. URB
3. ORDERING?

1.

M1,m2,m3,m4

M3,m5

M1,m2,m3,m4

2.

M1,m2,m3,m4,m5

M3,m5

M1,m2,m3,m4,m5

3.

Fifo: m1->m2,m3->4

Local: m3->m5

Both fifo and causal

**Pg.8**