Distributed Systems Midterm Exercises with solutions

Prof. Alessandro Mei. Midterm exercises solved by Alessio Lucciola during the a.y. 2022/2023.

Solutions may contain errors or typos. If you see one, you can contact me using the links in the Github page. If you find this helpful you might consider buying me a coffee 😄.

EXERCISES

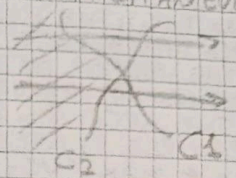
1) LET C_1 AND C_2 BE TWO CONSISTENT CUTS. SHOW THAT THE INTERSECTION OF C_1 AND C_2 IS CONSISTENT:

ES

IT IS NECESSARY TO REMIND THAT A CUT IS CONSISTENT IFF:

$$e \to e' \wedge e' \in C \to e \in C$$

SO IT'S IMPOSSIBLE THAT AN EVENT e' RECEIVES A MESSAGE FROM AN EVENT e WHICH OCCURRES IN THE FUTURE. GRAPHICALLY:



USING THE DEFINITION OF CONSISTENCY:

$$e \to e' \wedge e' \in C_1 \to e \in C_1 \;\&\; e \to e' \wedge e' \in C_2 \to e \in C_2$$

PROOF $e \to e' \wedge e' \in C_1 \cap C_2 \to e \in C_1 \wedge e \in C_2 \to e \in C_1 \cap C_2$

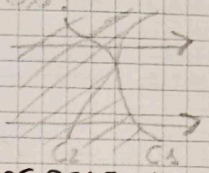
FOR THE DEF OF CONSIST. THAT MEANS THAT

2) LET C_1 AND C_2 BE TWO CONSISTENT CUTS. SHOW THAT THE UNION OF C_1 AND C_2 IS CONSISTENT:

ES

THE SOLUTION IS PRACTICALLY THE SAME AS THE INTERSECTION, JUST USING THE DEFINITION OF CONSISTENCY UNDER THE UNION OPERATION:

WE KNOW THAT $e \to e'$ AND $e' \in C_1 \cup C_2$, WE HAVE 2 OPTIONS:



1) if $e' \in C_1 \to e \in C_1 \to e \in C_1 \cup C_2$

BY CONSISTENCY THAT MEANS THAT

2) if $e' \in C_2 \to e \in C_2 \to e \in C_1 \cup C_2$

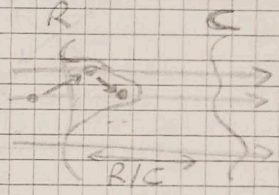
3) SHOW THAT EVERY CONSISTENT GLOBAL STATE (CONSISTENT CUT) CAN BE REACHED BY A CONSISTENT RUN:

ES

LET'S IMAGINE WE HAVE SOME PROCESSES THAT COMPUTE IN SOME WAY. IN THIS COMPUTATION THERE IS A CUT WHICH IS CONSISTENT. WE NEED TO SHOW THAT THERE IS A WAY TO REACH THIS CUT AND THIS RUN MUST BE ALSO CONSISTENT. WE NEED TO OBSERVE THAT:

THE RELATION OF HAPPENED BEFORE IS ACYCLIC.

ASSUME THAT WE HAVE A RUN R: THIS RUN IS EXTENDABLE BY ADDING NEW EVENTS TO THE RUN AND THE RUN IS STILL CONSISTENT AND WE DON'T GO PAST THE CUT C.

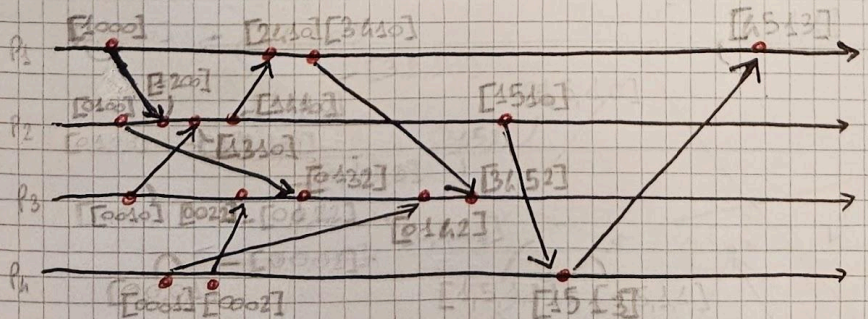


WE HAVE TO TAKE A RANDOM EVENT e_i IN THE SET R/C AND NOTICE IF THERE IS ANOTHER EVENT e_j WHICH HAPPENS BEFORE: IF THAT'S THE CASE THEN WE MUST SELECT THE EVENT e_j i.e. WE MUST ITERATE THIS WAY UNTIL THERE AREN'T EVENTS THAT HAPPEN BEFORE THAT AREN'T PART OF THE RUN. IF WE MANAGE TO BUILD THIS "PATH" WE CAN EXTEND THE RUN TO THE FIRST EVENT SELECTED. THIS RUN IS CERTAINLY CONSISTENT BECAUSE GIVEN AN EVENT, EVERY OTHER EVENTS THAT INFLUENCE THAT PARTICULAR EVENT MUST BE IN THE SAME RUN.

SINCE THE SET OF EVENTS IN R/C IS FINITE THEN WE WILL REACH C AT SOME POINT.

WE CAN IMPLEMENT THIS IDEA IN A COSTRUCTIVE WAY SO WE START FROM $\emptyset$ (WHICH IS A CONSISTENT RUN) AND WE ITERATIVELY EXTEND IT BY TAKING ALL THE EVENTS IN C/R, CHOOSING THE ONE WHICH HAS NO OTHER EVENTS THAT HAPPEN BEFORE AND ADD IT OR AND CONTINUE UNTIL YOU REACH THE CUT C.

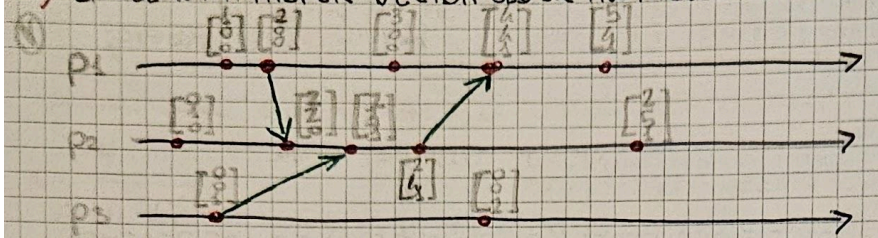
4) LABEL WITH PROPER VECTOR CLOCK ALL THE EVENTS OF:



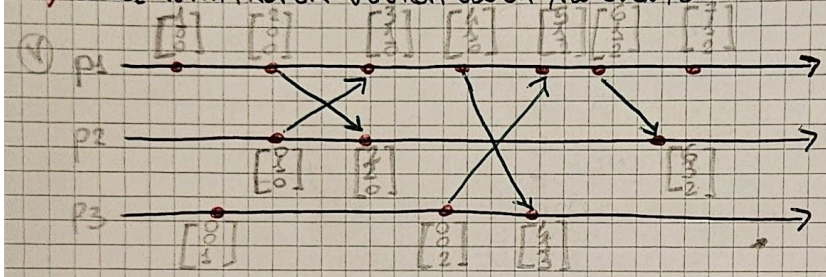
HOW TO:

1) INITIALLY ALL CLOCKS ARE ZERO;
2) EACH TIME A PROCESS EXPERIENCES AN INTERNAL EVENT, IT INCREMENTS ITS OWN LOGICAL CLOCK IN THE VECTOR BY ONE. FOR INSTANCE, UPON AN EVENT AT PROCESS i, IT UPDATES $VC_i[i] \leftarrow VC_i[i] + 1$;
3) EACH TIME A PROCESS SENDS A MESSAGE, IT INCREMENTS ITS OWN LOGICAL CLOCK IN THE VECTOR BY ONE (AS IN THE BULLET ABOVE, BUT NOT TWICE FOR THE SAME EVENT) AND THEN THE MESSAGE PIGGYBACKS A COPY OF ITS OWN VECTOR;
4) EACH TIME A PROCESS RECEIVES A MESSAGE, IT INCREMENTS ITS OWN LOGICAL CLOCK IN THE VECTOR BY ONE AND UPDATES EACH ELEMENT IN ITS VECTOR BY TAKING THE MAXIMUM OF THE VALUE IN ITS OWN VECTOR CLOCK AND THE VALUE IN THE VECTOR IN THE RECEIVED MESSAGE (FOR EVERY ELEMENT). FOR EXAMPLE: IF PROCESS P_j RECEIVES A MESSAGE m FROM P_i, IT UPDATES BY SETTING $VC_j \leftarrow \max(VC_j[k] + 1, VC_i[k])\ \forall k$

5) LABEL WITH PROPER VECTOR CLOCK ALL THE EVENTS OF:

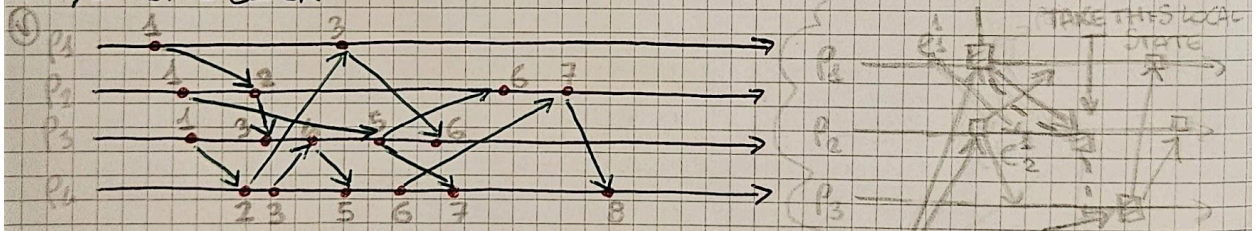


6) LABEL WITH PROPER VECTOR CLOCK ALL EVENTS OF:

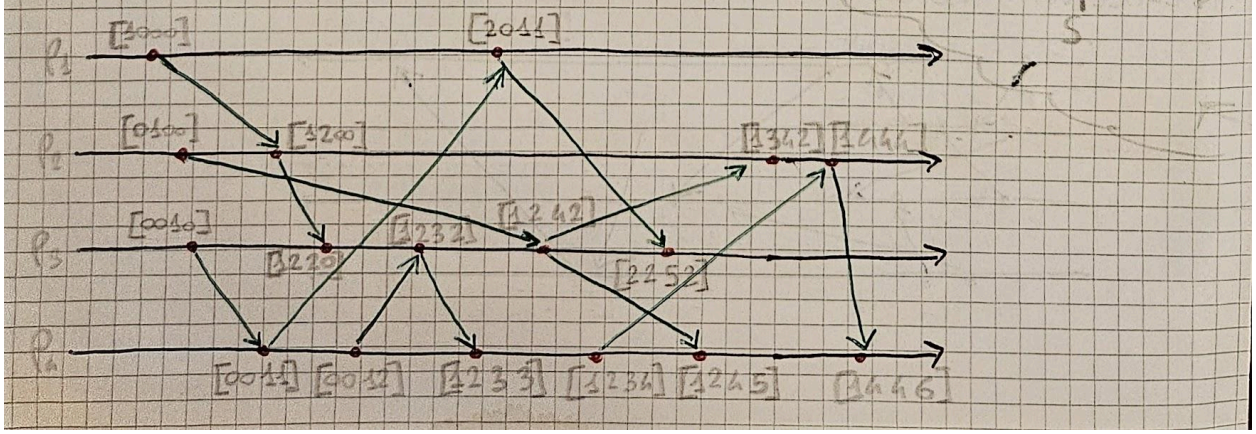


7) TAG WITH LAMPORT AND VECTOR CLOCK THIS DISTRIBUTED COMPUTATION:

1) LAMPORT'S CLOCK:



2) VECTOR CLOCK:



8) LET S_0 BE THE GLOBAL STATE WHEN THE PROTOCOL STARTS, S THE GLOBAL STATE BUILD BY THE PROTOCOL, S_1 THE GLOBAL STATE WHEN THE PROTOCOL ENDS. SHOW THAT S IS REACHABLE FROM S_0 AND S_1 IS REACHABLE FROM S (SO $S_0 \to S \to S_1$):

(ES) (4) LET R BE A RUN IN WHICH THE TWO SNAPSHOTS DO AVOID EACH OTHER:

$$R = \underbrace{e_1 e_2 \dots e_a}_{S_0} \mid \underbrace{e_{a+1} \dots e_b}_{S_1} \mid e_{b+1} \dots e_n$$

WE KNOW THAT R IS CONSISTENT BY DEFINITION (SO INFORMALLY, EACH EVENT IS INFLUENCED SOMEWAY BY THE EVENTS ON ITS LEFT).
WE HAVE TO CHANGE R IN A WAY THAT WE ARE SURE IT GOES THROUGH S.
LET'S BUILD R' THIS WAY:

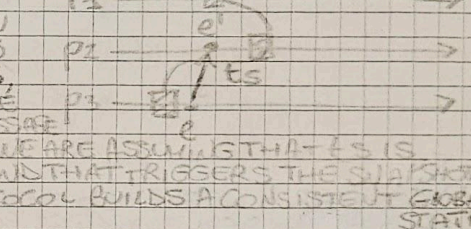
$$R' = \underbrace{e_1 e_2 \dots e_a}_{S_0} \mid \underbrace{\overleftarrow{\quad A' \quad}}_{S} \mid \underbrace{\overleftarrow{\quad A'' \quad}}_{S_1} \mid \underbrace{e_{b+1} \dots e_n}_{F} \quad \left\} \begin{array}{l} \text{WHERE} \\ A = e_{a+1} \dots e_b \end{array} \right.$$

WE ASSUME S TO BE IN THAT POSITION AND SPLIT A THIS WAY:
- A' = ALL EVENTS IN A THAT BELONG TO S IN THE SAME ORDER;
- A'' = ALL EVENTS IN A THAT DON'T BELONG TO S;

WE ARE SURE THAT WE GO FROM S_0 TO S_1 (THROUGH S) AND THE RUN IS CONSISTENT BECAUSE:
1) S AND F ARE CONSISTENT SINCE WE TOOK THEM FROM THE INITIAL RUN WHICH WE KNOW TO BE CONSISTENT;
2) A' AND A'' HAVE THE SAME ORDER AS IN A AND WE KNOW IT TO BE CONSISTENT AS WELL (CL PROTOCOL ACTUALLY MAKE ONLY CONSISTENT CUTS).

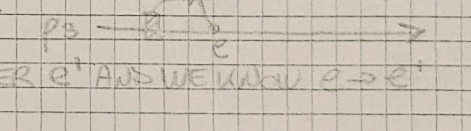
9) SHOW THAT CHANDY-LAMPORT SNAPSHOT PROTOCOL BUILDS A CONSISTENT GLOBAL STATE:

(ES) (4) WE KNOW THAT CHANNELS ARE FIFO (SO MESSAGES FROM THE SAME SENDER ARE DELIVERED IN THE SAME ORDER IN WHICH THEY WERE SENT). NOW WE ASSUME THAT WE HAVE AN INCONSISTENT CUT S MADE FROM A SNAPSHOT SO WE 2 EVENT $e \notin S$, $e' \in S$ AND $e \to e'$. IT IS IMMEDIATE TO NOTICE THAT, SINCE CHANNELS ARE FIFO, THE ES MESSAGE MUST ARRIVE AFTER THE e' EVENT (HERE WE ARE ASSUMING THAT ES IS THE FIRST ES MESSAGE THAT P_2 RECEIVES AND THAT TRIGGERS THE SNAPSHOT) BUT, FOR THIS REASON, $e' \notin S$ $\Rightarrow$ CL PROTOCOL BUILDS A CONSISTENT GLOBAL STATE.



10) SHOW THAT IF CHANNELS ARE NOT FIFO, THEN CHANDY-LAMPORT SNAPSHOT ALGORITHM DOES NOT WORK:

(ES) IF WE ASSUME THAT CHANNELS ARE NOT FIFO THEN WE MAY A SITUATION IN WHICH GIVEN TWO EVENTS $e \in \sigma$ $e' \notin \sigma$ THEN $e \to e' \wedge \sigma' \in S \not\Rightarrow e \in S$ THAT DOESN'T BUILDS A CONSISTENT GLOBAL STATE. IN THE EXAMPLE ON THE RIGHT, P_2 SENDS A ES MESSAGE BEFORE SENDING THE EVENT e AND P_3 RECEIVES IT AFTER e' AND WE KNOW $e \to e'$ SO THIS IS AN INCONSISTENT CUT.



11) WHAT MODIFICATION SHOULD BE DONE TO THE CHANDY-LAMPORT SNAPSHOT PROTOCOL SO THAT IT RECORDS A STRONGLY CONSISTENT SNAPSHOT (I.E. ALL CHANNEL STATES ARE RECORDED EMPTY)?

(ES) A CUT IS STRONGLY CONSISTENT IF, FOR ALL SENDING EVENTS $e \in C$ TO ANOTHER PROCESS, WE ALSO HAVE THE RECEIVING EVENT $e' \in C$. A POSSIBLE MODIFICATION MIGHT BE THAT WHEN THE PROCESS RECEIVES THE ES MESSAGE IT BROADCAST IT TO THE OTHER PROCESSES AND STOPS SENDING OTHER MESSAGES UNTIL THE PROTOCOL TERMINATES. IT MAY HAPPEN THAT AN EVENT IS RECEIVED AFTER TAKING THE SNAPSHOT AND IN THIS CASE THE CUT ISN'T STRONGLY CONSISTENT. LET'S ASSUME IT IS SENT BY P_1 AND FOLLOWING THE PREVIOUS ASSUMPTIONS, IT STILL HAVE TO RECEIVE THE FIRST ES MESSAGE. ONCE IT RECEIVES IT, P_1 BROADCASTS THE ES TO ALL THE OTHER PROCESSES AND IT WILL ARRIVE AFTER THE SENDING EVENT (SINCE CHANNELS ARE FIFO). SINCE THE SYSTEM IS ASYNCHRONOUS, BOTH m AND THE ES COULD ARRIVE AFTER THE RECEIVING PROCESS TAKES THE SNAPSHOT SO INSTEAD OF TAKING THE FIRST SNAPSHOT, WE COULD TAKE THE ONE IN WHICH m ALREADY ARRIVED, MEANING THE OVERALL CUT STRONGLY CONSISTENT SO P_2 RECEIVES A LOCAL STATE AT EVERY ES MESSAGE AND IT WILL GET THE ONE IN WHICH ALL MESSAGES HAVE ARRIVED (IT WILL HAPPEN BEFORE THE PROTOCOL TERMINATES).

12) GIVE AN ACP THAT ALSO SATISFIES THE CONVERSE OF CONDITION AC3. THAT IS, IF ALL PROCESSES VOTED YES, THEN THE DECISION MUST BE COMMIT. WHY IS IT NOT A GOOD IDEA TO ENFORCE THIS CONDITION?
(ES)

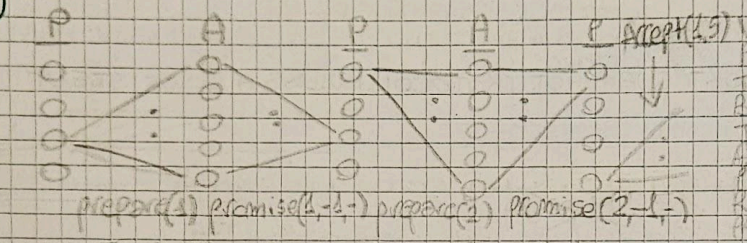
SUPPOSE THAT WE SATISFY THE CONVERSE OF AC3 THEN LET'S MAKE AN EXAMPLE THAT IT IS NOT A GOOD IDEA. ASSUME THAT WE HAVE N PARTICIPANTS WHICH VOTES "YES" BUT ONE OF THE MESSAGES GET LOST. IF THIS HAPPENS, ASSUMING THAT AC3 CONVERSE IS SATISFIED, THEN WE WOULD HAVE A BLOCK SITUATION IN WHICH THE COORDINATOR CAN'T HAVE A DECISION. TO KEEP THE SYSTEM SAFE IT COULD SEND ABORT AFTER A CERTAIN TIMEOUT BUT THIS WAY THE AC3 CONVERSE PROPERTY IS NOT SATISFIED.
CL CL IS A WEAK VERSION OF AC3 CONVERSE THAT ACTUALLY CAPTURE THIS PROBLEM.

13) CONSIDER 2PC WITH COOPERATIVE TERMINATION PROTOCOL. DESCRIBE A SCENARIO (A PARTICULAR EXECUTION) INVOLVING SITE FAILURES ONLY, WHICH CAUSES OPERATIONAL SITES TO BE BLOCKED:
(ES)

LET'S SUPPOSE THE COORDINATOR STARTS THE PROTOCOL BY SENDING A "VOTE REQUEST" MESSAGE. ALL PARTICIPANTS RECEIVE IT AND VOTE YES. THE COORDINATOR RECEIVES ALL THE VOTES BUT CRASHES. IN THIS CASE WE A BLOCK SITUATION SINCE THE PARTICIPANTS DON'T GET THE DECISION. THEN COULD USE THE COOP TERM PROTOCOL TO KNOW WHAT WAS THE OTHERS DECISIONS AND ACT ACCORDINGLY BUT IF ONE OF THE PARTICIPANTS CRASHES AS WELL THEN CAN'T MAKE A DECISION (AND BE SURE THE SYSTEM REMAINS SAFE (THINK OF THE CASE EVERYONE VOTED YES EXCEPT FOR THE CRASHED PARTICIPANT)) SO THIS ANOTHER BLOCK SITUATION.

14) SHOW THAT PAXOS IS NOT LIVE:
(ES)

LET'S ASSUME WE HAVE THIS SITUATION:

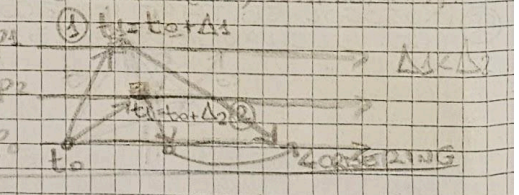


WE A PROPOSER P_2 WHO SENDS A PREPARE(1) MESSAGE WHICH IS SENT TO THE ACCEPTORS AND THEY REPLY BACK WITH A PROMISE MESSAGE. IN THE MEANWHILE A PROPOSER P_1 SENDS ANOTHER PROPOSER P_1 SENDS ANOTHER PREPARE(2) AND THE ACCEPTORS REPLY BACK WITH ANOTHER PROMISE MESSAGE.
IN THE SECOND PROMISE MESSAGE THE PROMISE NOT TO VOTE IN ROUNDS SMALLER THAN 2 SO WHEN P_2 SENDS ACCEPT(1,5) (5 IS A RANDOM VALUE) IT GETS DISCARDED.
THIS SITUATION CAN GO ON FOREVER MAKING PAXOS NOT LIVE.

15) CONSIDER A DISTRIBUTED SYSTEM WHERE EVERY NODE HAS ITS PHYSICAL CLOCK AND ALL PHYSICAL CLOCK ARE PERFECTLY SYNCHRONIZED. GIVE AN ALGORITHM TO RECORD GLOBAL STATE ASSUMING THE COMMUNICATION NETWORK IS RELIABLE (NOTE THAT YOUR ALGORITHM SHOULD BE SIMPLER THAN THE CHANDY-LAMPORT ALGORITHM):
(ES)

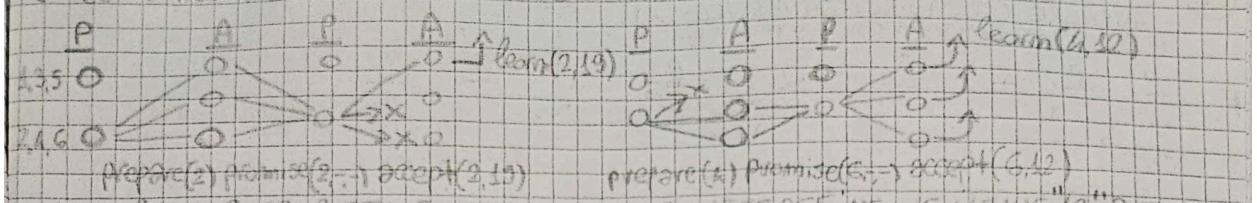
IF EACH CLOCK ARE PERFECTLY SYNCHRONIZED, IT MEANS THAT WE A SYSTEM IN WHICH WE HAVE GLOBAL REAL CLOCK. EACH TIME THE MONITOR WANTS TO TAKE A SNAPSHOT OF THE SYSTEM IT CAN SEND A HIS MESSAGE TO ALL PROCESSES AT TIME t_0 THAT WILL ARRIVE AT TIME $t_0 + \Delta = t_1$ (Δ IS A DELAY). WE ARE SURE MESSAGES WILL EVENTUALLY ARRIVE SINCE CHANNELS ARE RELIABLE. EACH PROCESS SENDS ITS LOCAL STATE BACK TO P_0 AND EVEN IF MESSAGES ARRIVE IN A DIFFERENT ORDER P_0 CAN SORT THEM IN INCREASING TIMESTAMP ORDER. THE ALGORITHM IS ACTUALLY SIMPLER AND ITS COMPLEXITY IS $O(n)$ (CL PROTOCOL HAS $O(n^2)$ COMPLEXITY)

HERE WE ASSUME TO HAVE A MAX. DELAY Δ.



* LET'S ASSUME WE START FROM THE RUN THAT ACTUALLY HAPPENED. ASSUME WE MARK ALL THE EVENTS THAT ARE IN S. IF ALL MARKED EVENTS ARE ON THE LEFT OF THEN WE ARE SURE THE RUN GOES THROUGH S. NOW ASSUME WE HAVE 2 EVENTS IN WHICH THE ONE ON THE LEFT IS NOT MARKED (IT IS NOT IN S) AND THE ONE ON THE RIGHT IS MARKED (IT IS IN S). WE CAN SWAP THEM AND THE RUN WOULD BE STILL CONSISTENT. IF WE DO IT FOR ALL ADJACENT EVENTS SUCH AS THE ONE ON THE LEFT IS IN THE FUTURE AND THE ONE ON THE RIGHT IS IN THE PAST WE FINALLY OBTAIN THE RUN THAT GIVES THE S

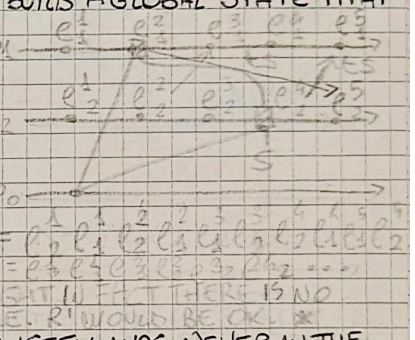
16) BUILD A RUN IN WHICH THE FIRST ACCEPTOR SENDS A LEARN MESSAGE FOR VALUE "19" AND THEN ANOTHER VALUE "12" IN TWO DIFFERENT ROUNDS.



IN THE ROUND 2, PROPOSER P2 MAKES AN ACCEPT MESSAGE UPLINK VALUE "19" BUT TWO OF THE MESSAGE ARE LOST FOR SOME REASONS, SO THE ONLY ACCEPTOR A1 THAT RECEIVED THE MESSAGE SENDS A LEARN MESSAGE BUT THE VALUE IS NOT ACCEPTED SINCE IT DOESN'T HAVE THE MAJORITY.
IN THE NEXT ROUND, P2 SENDS ANOTHER PREPARE TO ALL THE ACCEPTORS EXCEPT FOR A1 (E.G. THE MESSAGE GETS LOST). NOW NOTICE THAT IF A1 WAS THE ONLY ONE THAT VOTED IN THE LAST ROUNDS, SO SINCE IT DOESN'T SEND THE PROMISE MESSAGE, P2 CAN ACCEPT AGAIN ANY VALUE (12). IF A1 RECEIVES THE ACCEPT MESSAGE, IT VOTES THE VALUE 12 THAT HAS NOW THE MAJORITY TO BE LEARNED.

17) HOW MANY MESSAGES ARE USED IN PAXOS IF NO MESSAGE IS LOST AND IN THE BEST CASE? IS IT POSSIBLE TO REDUCE THE NUMBER OF MESSAGES WITHOUT LOSING TOLERANCE TO FAILURES AND WITHOUT CHANGING THE NUMBER OF PROPOSERS, ACCEPTORS AND LEARNERS?
IF WE ASSUME NO MESSAGE IS LOST THEN A ROUND IS ENOUGH TO RECEIVE A VALUE. GIVEN M ACCEPTORS AND REMEMBERING THAT ONLY ONE PROPOSER STARTS A RUN, WE HAVE M PREPARE, M PROMISE, M ACCEPT AND M*L LEARN MESSAGES (WHERE L IS THE NUMBER OF LEARNERS) SO A TOTAL OF M×(3+L) MESSAGES. IF WE MANAGE TO SELECT A COORDINATOR (WHICH IS ONE OF THE PROPOSER) IT IS POSSIBLE TO REDUCE THE COMMUNICATION ROUNDS TO 3 (INSTEAD OF 4) BUT WE WOULD LOSE TOLERANCE TO FAILURES (IF THE COORDINATOR CRASHES). A POSSIBLE SOLUTION MIGHT BE THAT EACH TIME WE HAVE A QUORUM WE SEND MESSAGE ONLY TO THE MAJORITY OF NODES (BUT WE HAVE TO ASSUME WHICH IS THE NUMBER THAT FORMS THE MAJ.)

18) SHOW THAT THE CHANDY LAMPORT SNAPSHOT PROTOCOL CAN BUILD A GLOBAL STATE THAT NEVER HAPPENED:
WE KNOW THAT THE SYSTEM IS ASYNCHRONOUS AND CHANNELS ARE FIFO. THE CHANDY LAMPORT SNAPSHOT PROTOCOL BUILDS A PICTURE OF THE STATE THAT COULD HAVE HAPPENED AND NOT THE ACTUAL STATE. THIS MEANS WE COULD HAVE A RUN IN WHICH THE RECORDED GLOBAL STATE NEVER OCCURRED IN THE EXECUTION. THIS HAPPENS BECAUSE A PROCESS CAN CHANGE ITS STATE ASYNCHRONOUSLY BEFORE THE SENT ES MESSAGES ARE RECEIVED BY THE OTHER PROCESSES AND THEY TAKE THE SNAPSHOT. AN EXAMPLE MIGHT BE THE COMPUTATION ON THE RIGHT IN FACT THERE IS NO POINT IN WHICH THE GLOBAL STATE IS THE ONE IN THE EXAMPLE. R' WOULD BE OK!

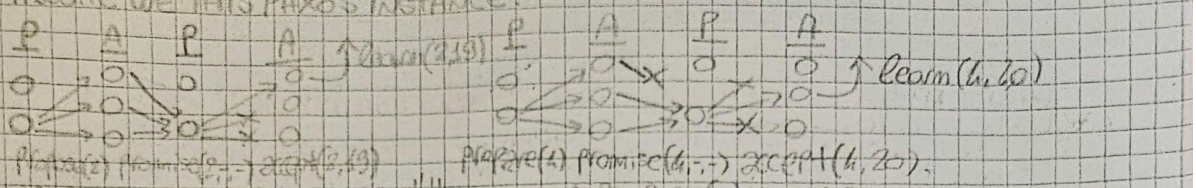


19) WHAT GOOD IS A DISTRIBUTED SNAPSHOT WHEN THE SYSTEM WAS NEVER IN THE STATE REPRESENTED BY THE DISTRIBUTED SNAPSHOT?
SNAPSHOT PROTOCOL BUILDS A PICTURE OF THE STATE THAT COULD HAVE HAPPENED AND NOT NECESSARILY THE ACTUAL STATE AS LONG AS IT IS CONSISTENT (CHANDY LAMPORT PROTOCOL MAKE ONLY CONSISTENT CUTS).
IF THE SYSTEM WAS NEVER IN THE STATE REPRESENTED BY THE SNAPSHOT THEN SOME USEFUL APPLICATIONS MIGHT BE:
1) DEADLOCK DETECTION → A SNAPSHOT CAN BE USED AS A DEBUGGING TOOL TO HAVE DEADLOCK DETECTION.
2) STABLE PROPERTY DETECTION → A PROPERTY OF THE SYSTEM IS SAID TO BE STABLE IF ONCE IT BECOMES TRUE IT REMAINS TRUE (E.G. DEADLOCK).
3) CHECKPOINTING → ALLOWS TO RECONSTRUCT THE SYSTEM STATE IN CASE A FAILURE OCCURS.

20) ASSUME THAT ACCEPTORS DO NOT CHANGE THEIR VOTE. IN OTHER WORDS, IF THEY VOTE FOR VALUE V IN ROUND i, THEY WILL NOT SEND LEARN MESSAGES WITH VALUE DIFFERENT FROM V IN LARGER ROUNDS. SHOW THAT PAXOS DOES NOT WORK ANYMORE (IT CAN REACH A LIVELOCK).

(ES)

ASSUME WE THIS PAXOS INSTANCE:

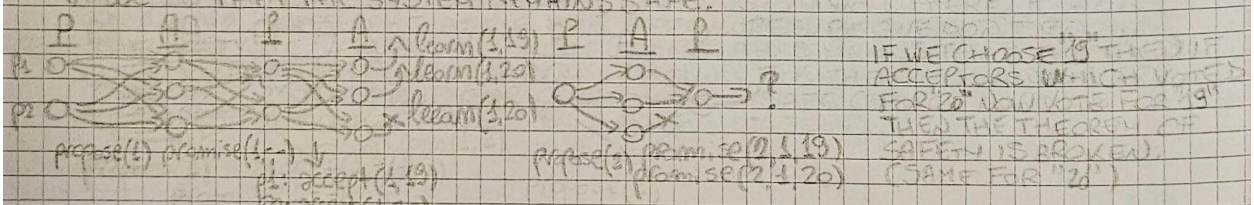


A PROPOSER P_2 STARTS A ROUND 2 SENDING A PREPARE(2) MESSAGE TO ALL THE ACCEPTORS. IT RECEIVES THE PROMISE MESSAGES BACK AND SINCE NOONE VOTED BEFORE, IT CHOOSES "19" AS VALUE V. THE ACCEPT MESSAGE ARRIVE ONLY TO THE ACCEPTOR A_1 (THE OTHERS ARE LOST FOR SOME REASONS) AND IT SENDS A LEARN MESSAGE BUT THE VALUE IS NOT DECIDED SINCE THERE IS NO MAJORITY. IN THE SECOND ROUND "4" P_2 SENDS ANOTHER PREPARE(2) AND THE ACCEPTORS PROMISE BACK EXCEPT FOR A_1 WHOSE MESSAGE IS LOST AND IT WAS THE ONLY ONE WHICH VOTED BEFORE. P_2 SENDS AN ACCEPT FOR VALUE "20" (DIFFERENT FROM BEFORE) BUT ONLY A_2 RECEIVES IT AND SENDS A LEARN MESSAGE. AGAIN IT HE'S NOT THE MAJORITY SO IT IS DISCARDED.
NOW, NOTICE THAT THE SYSTEM IS BUILT TO TOLERATE 1 CRASH. THEN IF WE FOLLOW THE SAME IDEA WE WON'T ABLE TO FIX A_3 TO LEARN ANOTHER VALUE BECAUSE WHEN THE PROPOSER ASKS FOR THE MAJORITY, IT WILL BE EITHER ON 19 OR 20 SO IF WE ASSUME A_2 CRASHES THEN WE REACH A LIVELOCK.

21) ASSUME THAT YOU REMOVE THE PROPERTY THAT EVERY ROUND IS ASSOCIATED WITH A UNIQUE PROPOSER. AFTER COLLECTING A QUORUM OF $n-F$ PROMISES (WHERE n IS THE NUMBER OF ACCEPTORS AND F IS SUCH THAT $n = 2F+1$), THE PROPOSER CHOOSES ONE OF THE VALUES VOTED IN MAX ROUND IN THE PROMISES (OF COURSE IT IS NOT UNIQUE, THE PROPOSER CHOOSE JUST ONE IN AN ARBITRARY WAY). SHOW THAT PAXOS IS NOT SAFE ANYMORE.

(ES)

IF ROUND NUMBERS ARE NOT UNIQUE IT MIGHT HAPPEN THAT TWO PROPOSERS PROPOSE DIFFERENT VALUE IN THE SAME ROUND. IF WE ASSUME TO HAVE A CERTAIN NUMBER OF ERRORS THEN SOME ACCEPTORS MAY ACCEPT ONE VALUE AND SOME THE OTHER. A NEW PROPOSER COULD START A NEW ROUND AND ASK ALL ACCEPTORS THEIR HIGHER ACCEPTED VALUE. SINCE IT WILL GET DIFFERENT VALUE FOR THE SAME ROUND IT WON'T BE ABLE TO CHOOSE A VALUE SO THAT THE SYSTEM REMAINS SAFE.



22) YOU ARE AN OPTIMIZATION FREAK. YOU REALIZE THAT, IN SOME CASES, IT IS NOT NECESSARY THAT THE PROPOSER COLLECTS $n-F$ ("THE FAST PAXOS QUORUM") PROMISES TO MAKE A DECISION. WHICH IS THE MINIMUM QUORUM AND UNDER WHAT HYPOTHESIS THIS MINIMUM QUORUM IS ENOUGH TO TAKE A DECISION?

(ES)

THERE EXISTS A PROPERTY CALLED WEAKENED INTERSECTION THAT SHOWS THAT PHASE 1 OF A FAST ROUND CAN USE THE SAME QUORUM OF A PHASE 1 OF A CLASSIC ROUNDS (THIS IS BECAUSE FAST QUORUMS MUST BE AT LEAST AS LARGE AS CLASSIC QUORUMS). THERE ARE SEVERAL CHOISES WE CAN MAKE:
1) WE COULD HAVE $QC = \lceil 3n/4 \rceil$ AND $QC = \lfloor n/2 \rfloor + 1$ BUT NOTICE THAT A SIMPLE MAJORITY IS NEEDED FOR PHASE-1 OF FAST ROUNDS.
2) WE COULD HAVE $QC = \lceil 3n/4 \rceil$ AND $QC = \lfloor n/2 \rfloor + 1$ BUT NOTICE THAT A ONLY MAJORITY OF ACCEPTORS ARE NEEDED FOR PHASE-2 OF CLASSIC ROUNDS.
WE HAVE SOME FLEXIBILITY AND SINCE PHASE-2 IS USUALLY EXECUTED MORE THAN PHASE-1 WE CAN DECREASE THE NUMBER OF PHASE-2 QUORUMS (FAST AND CLASSIC) BUT INCREASING THE NUMBER OF PHASE-1 QUORUMS.

(ES) $n = 11$ ACCEPTORS, WE CAN USE 2 SO THAT WE USE 7 ACCEPTORS IN FAST ROUNDS AND 3 IN CLASSIC ROUNDS, IF IT USES A QUORUM OF 9 ACCEPTORS FOR PHASE-1.

TO VERIFY THE INTERSECTION REQUIREMENTS AND CONSIDER POSSIBLE

23) ASSUME THAT ALL PROPOSERS ARE LEARNERS AS WELL. LET EVEN ROUNDS BE ASSIGNED TO PROPOSERS WITH THE RULE THAT WE KNOW. IF ROUND $2i$ IS ASSIGNED TO PROPOSER P, THEN ALSO ROUND $2i+1$ IS ASSIGNED TO PROPOSER P. ODD NUMBERS ARE "RECOVERY" ROUNDS. IF ROUND $2i$ IS A FAST ROUND AND IF THE PROPOSER OF ROUND $2i$ SEES A CONFLICT (IT IS ALSO A LEARNER), THEN THE PROPOSER IMMEDIATELY SENDS AN ACCEPT FOR ROUND $2i+1$ WITH VALUE THAT HAS BEEN MOST VOTED IN ROUND $2i$, WITHOUT ANY PREPARE AND ANY PROMISE.
IS SAFETY VIOLATED? IF YES, SHOW AN EXAMPLE. IF NOT, DEMOSTRATE SAFETY.
IF A CONFLICT HAPPENED IN ROUND $2i$ THEN ACCEPTORS COULD RECEIVE CONCURRENT REQUESTS FROM DIFFERENT PROPOSERS CARRYING DIFFERENT VALUES. LET'S SUPPOSE THAT THEY VOTE FOR SOME VALUE (SO THEY SEND A LEARN MESSAGE) BUT A MAJORITY IS NOT REACHED DUE TO THE CONFLICT). NOW WE CAN NOTICE THAT THE MESSAGES OF THIS LAST PART OF THE FAST ROUNDS (ACCEPT AND LEARN) CONTAIN THE VALUE VOTED BY EACH PROCESS AT ROUND i AND ALSO IMPLY THAT THEY WON'T VOTE FOR ANY DIFFERENT VALUE IN BIGGER ROUNDS (ALSO VALID FOR $2i+1$). IF THE COORDINATOR IS ALSO THE LEARNER IT COULD START THE NEW RECOVERY ROUND USING THOSE MESSAGES AS THEY WERE THE PROPOSE/PROMISE MESSAGES IN ROUNDS $2i+1$, SO IT WILL SIMPLY ACCEPT THE VALUE ASSOCIATED WITH THE LARGEST LAST VOTE (THE MOST VOTED IN $2i$). IT WILL SUCCEED IF IT MANAGES TO GET A NON-FAULTY QUORUM. ✗1

24) SHOW THAT FAST PAXOS IS NOT SAFE IF WE ASSUME THE SAME NUMBER OF CRASH FAILURES OF PAXOS:
LET'S MAKE AN EXAMPLE: ASSUME WE HAVE $n=7$ ACCEPTORS AND $F=3$ FAILURES. DURING ROUND i, THE PROPOSER RESPONSIBLE FOR ROUND i COLLECTS A QUORUM OF $n-F=4$ QUORUM FROM THE ACCEPTORS. WE ASSUME THAT IN ROUND i SOMEONE VOTES $V1$ AND OTHERS $V2$ SO WE'LL HAVE PROMISE$(i, V1)$ AND PROMISE$(i, V2)$ ($V2 \neq V1$) MESSAGES (THIS IS POSSIBLE BECAUSE IN FAST ROUNDS MULTIPLE VALUES CAN BE PROPOSED AND VOTED). THE PROBLEM IS THAT THE COORDINATOR DOESN'T KNOW THE VOTE OF THE 3 CRASHED ACCEPTORS AND THIS MAY BE ENOUGH TO FORM A QUORUM IN ROUND i EITHER ON VALUE $V1$ AND VALUE $V2$ (ASSUMING THE SAME NUMBER OF CRASH FAILURES OF PAXOS), SO IN ROUND i A SAFE CHOICE CANNOT BE MADE. THAT'S WHY WE NEED A LARGER QUORUM $n-F'$ WITH $F' = \lfloor (n-1)/3 \rfloor$.

25) SHOW THAT IN PAXOS, IF WE REMOVE THE PROMISE THAT THE ACCEPTOR DOES NOT PARTICIPATE IN PREVIOUS ROUNDS, PAXOS IS NOT SAFE ANYMORE.
IN ORDER TO PROVE THIS, LET'S MAKE AN EXAMPLE: LET'S SUPPOSE A PROPOSER P_1 SENDS A PREPARE(i) MESSAGE TO ALL THE ACCEPTORS AND THEY REPLY BACK WITH A PROMISE$(i, \perp)$ MESSAGE (SO NO ONE HAS VOTED BEFORE). THIS MEANS THAT THEY WILL VOTE FOR A VALUE PROPOSED BY THE PROPOSER P_1 IN ROUND i — PROMISE THAT THEY WON'T VOTE IN PREVIOUS ROUNDS. AT THE MEANTIME ANOTHER PROPOSER P_2 STARTS ANOTHER ROUND $j < i$ AND ACCEPTORS SEND A PROMISE MESSAGE BACK. NOW P_1 PROPOSE A VALUE (IT CAN EVEN IF THE ROUND IS SMALLER) AND ASSUME IT HAS THE MAJORITY THEN THIS VALUE IS ACCEPTED. P_2 NOW PROPOSE ANOTHER VALUE DIFFERENT FROM THE PREVIOUS ONE (SINCE IT DOESN'T KNOW ABOUT IT) AND THE ACCEPTORS ALSO ACCEPT THIS OTHER VALUE. IF THIS IS POSSIBLE WE VIOLATE THE THEOREM OF SAFETY SINCE ACCEPTORS VOTE A DIFFERENT VALUE IN ROUNDS $i \neq j$.

1
✗ (DEM) IN ORDER TO PROVE IT LET'S ASSUME WE ARE IN ROUND $2i+1$ AND WE RECEIVES A MAJORITY Q OF VOTES. IN THIS CASE $j = \max(\text{LASTROUND}) < i$. IN ROUND $2i$ WE HAVE THAT SOME PEOPLE IN Q VOTED FOR DIFFERENT VALUES AND SOMEONE DIDN'T VOTE AT ALL. SINCE THE MAJORITY WAS REACHED IN ROUND $2i+1$ AND $j < i$, TRIVIALLY NOBODY VOTED IN ROUNDS BETWEEN $2i$ AND $2i+1$. NOW V IS THE MOST FREQUENT VALUE IN ROUND $2i-1$ THEN WE ARE SURE THAT THERE IS AT LEAST AN ACCEPTOR THAT VOTED V IN $2i$ AND, BY INDUCTION, IT IS SAFE TO VOTE V (AND WE ARE SURE NO MAJORITY COULD BE REACHED IN SMALLER ROUNDS.

22B) ANOTHER SOLUTION FOR (22):
FAST PAXOS REQUIRES THAT ANY 3 FAST QUORUMS (PROMISE, ACCEPT, LEARN) INTERSECT $Q_1 \cap Q_2 \cap Q_3 \neq \emptyset$ THAT MEANS THAT IN ORDER TO MAKE SAFE CHOICES WE NEED A QUORUM OF $2/3$ OF THE ACCEPTORS THAT IS $n - F$ WHERE $F = \lfloor (n-1)/2 \rfloor$. NOW WE CAN DIFFERENTIATE BETWEEN PHASE-1 QUORUMS Q_1 AND PHASE-2 QUORUMS Q_2 (ACCEPT, LEARN) INVARIANTS: IT IS POSSIBLE TO WEAKEN THE INTERSECTION REQUIREMENTS SO THAT A MAJORITY IS NEEDED FOR PHASE-1 OF A NEW LEADER (SIZE OF A MAJORITY OF $2/3$ OF ACCEPTORS) $\rightarrow \forall Q_1, Q_1', Q_2 \in Q_1 \cap Q_1' \cap Q_2 \neq \emptyset$.

26) MAKE PAXOS LIVE WITH A FAILURE DETECTOR $\Diamond S$:
A $\Diamond S$ FAILURE DETECTOR HAS THE FOLLOWING PROPERTIES:
- STRONG COMPLETENESS: FOR EVERY RUN σ, FOR EVERY PROCESS p WHICH IS CRASHED AND FOR EVERY PROCESS q WHICH IS UP THEN THERE EXIST SOME TIME t AFTER WHICH THE FAILURE DETECTOR S OF q SAY p ARE CRASHED AT TIME $t' > t$.
- EVENTUALLY WEAK ACCURACY: FOR EVERY POSSIBLE RUN σ THERE IS A TIME t SUCH THAT: FOR EVERY TIME $t' > t$ AND FOR EVERY PROCESS q THAT ARE UP, THERE EXISTS A PROCESS p THAT IS UP WHICH IS NOT SUSPECTED TO BE CRASHED BY THE FAILURE DETECTOR OF q.

FIRST OF ALL WE MUST ASSUME THAT THE SYSTEM IS SYNCHRONOUS BECAUSE (IN SUMMARY) THE FLP THEOREM ASSURES THAT A SYSTEM CAN'T BE BOTH SAFE AND LIVE IN AN ASYNCHRONOUS SYSTEM.
IN ORDER TO MAKE PAXOS LIVE WITH $\Diamond S$ WE COULD ELECT A LEADER USING A CONSENSUS PROTOCOL. ONE POSSIBLY MIGHT BE TO ELECT THE PROPOSER p_i WITH THE LOWEST ID. IF IT CRASHES FOR SOME REASON WE COULD ELECT p_j WHICH IS THE ONE WITH THE SECOND LOWEST LOWEST ID AND SO ON. THE PROBLEM IS THAT p_2 MAY THINK p_1 CRASHED EVEN IT IS UP LEADING TO HAVING 2 COORDINATORS AT THE SAME TIME AND THIS IS WRONG.
ASSUMING TO HAVE AN EVENTUALLY STRONG $\Diamond S$ FAILURE DETECTOR WE COULD SOLVE THIS PROBLEM BECAUSE IT WOULD TELL THE PROCESSES WHEN A PROCESS CRASHED: THANKS TO STRONG COMPLETENESS A PROCESS CAN KNOW IF A PROCESS (THE COORDINATOR) HAS CRASHED AND POSSIBLY TAKE THE LEAD. NOTICE THAT p_2 GETS TO KNOW p_1 CRASHED ONLY AFTER A TIME $t' > t$ SO IT MAY HAPPEN THAT FOR A CERTAIN AMOUNT OF TIME THERE IS NO LEADER, BUT IT DOESN'T MATTER BECAUSE AFTER t' WE GET THE LEADER (p_2 GETS THE LEAD) AND IT'S IMPOSSIBLE TO HAVE 2 OF THEM.