The Byzantine Agreement – part 2

Radu Nicolescu
Department of Computer Science
University of Auckland

12 August 2018

Stopping failures model

- Much simplified version of the Byzantine agreement
- A failed process can only stop sending messages, forever (no intermittent failures, recovery not considered)
- No possibility to send confusing messages (i.e. different messages to different directions)
- The problem can be solved for any $F \leq N - 1$ ☺ (not only when $3F \leq N - 1$)

The Stopping agreement conditions – vs ~~Byz~~

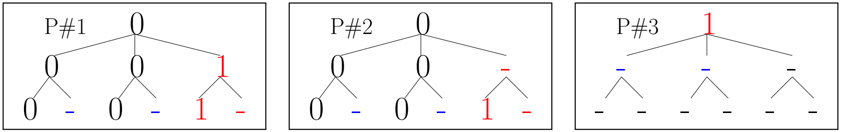
- Termination: all non-faulty processes eventually decide
- Agreement: no two ~~non-faulty~~ processes ever decide on different values
- Validity: if all ~~non-faulty~~ processes start with the same initial value $v \in V$, then v is the only one possible decision value
- If the processes start with different initial values, then the final decision could be any of these (as long as it is consistent)

EIGStop

- EIG tree as in the EIGByz, $F+1$ messaging rounds
  - recall: F can be as high as $N-1$ (not at most $(N-1)/3$)
- Top-down val()'s as in the EIGByz, i.e. via messaging
- No bottom-up newval() attributes
- Final decision: set W of all non-null val()'s in EIG tree
  - all values at all levels! not just leaves
  - nulls discarded! not assumed v_0
- If W is singleton, $W=\{v\}$, then the decision is v
- Otherwise, if W is mixed, $W=\{0,1\}$, then the decision is v_0
  - no voting! no tie breaking

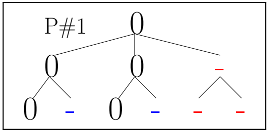
EIGStop example – assuming $v_0 = 1$; nulls as -

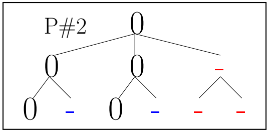
- Process #1 : init 0; decision $v_0 = 1$
- Process #2 : init 0; decision $v_0 = 1$
- Process #3 : init 1; no decision;
  fails after sending one 1st round message, to #1

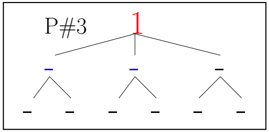


EIGStop example – assuming $v_0 = 1$; nulls as -

- Process #1 : init 0; decision 0
- Process #2 : init 0; decision 0
- Process #3 : init 1; no decision;
  fails before sending any 1st round message

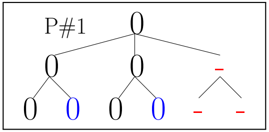


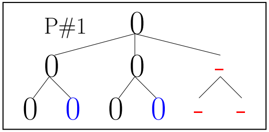


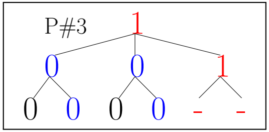


EIGStop example – assuming $v_0 = 1$; nulls as -

- WHAT IF scenario –NOT supported by this EIGStop protocol
- NO agreement
- Process #1 : init 0; decision 0
- Process #2 : init 0; decision 0
- Process #3 : init 1; decision $v_0 = 1$;
  What if P#3 fails before sending any 1st round out-message but would be immediately allowed to recover and decide







EIGStop vs EIGByz vs 3PC – assuming $v_0 = 0$

- x indicates a faulty process, which fails from start, before sending any 1st round message

Initial	EIGStop	EIGByz	3PC
0 0 0 0	0	0	0
0 0 0 1	0	0	0
0 0 1 1	0	0	0
0 1 1 1	0	1	0
1 1 1 1	1	1	1
x 0 0 0	0	0	0
x 0 0 1	0	0	0
x 0 1 1	0	0	0
x 1 1 1	1*	1	0

- * EIGStop: what would happen if the faulty x starts with 0 and would be allowed to recover after the 1st round?

EIGStop vs EIGByz vs 3PC – assuming $v_0 = 1$

- x indicates a faulty process, which fails from start, before sending any 1st round message

Initial	EIGStop	EIGByz	3PC
0 0 0 0	0	0	0
0 0 0 1	1	0	0
0 0 1 1	1	1	0
0 1 1 1	1	1	0
1 1 1 1	1	1	1
x 0 0 0	0*	0	0
x 0 0 1	1	1	0
x 0 1 1	1	1	0
x 1 1 1	1	1	0

- * EIGStop: what would happen if the faulty x starts with 1 and would be allowed to recover after the 1st round?

Byzantine agreement with authentication

- Assume that each process digitally signs its messages in a total safe way, e.g. based on PKI/DSS...
- Is this reasonable?
- Problem with certificate weaknesses: What if a powerful Byzantine faulty process is able to forge such signatures?
- Problem with authority: What if the certification authority itself is hacked or even turns into a Byzantine process?
- Anyway, assuming that such digital signatures are totally safe, Byzantine faulty nodes are not able to wreak much more havoc than a stopped process
- EIGStop can be adapted to solve the (slightly different) Byzantine agreement with authentication
- Faster/better/more general algorithms possible...