The Byzantine Agreement – An Introduction

University of Auckland

2 Sep 2020

1. The Byzantine agreement problem
2. Informal example
3. EI
4. Ex
5. Attributes
6. Quiz
7. Triple modular redundancy

Byzantine agreement story

-
-
- http://en.wikipedia.org/wik

Byzantine agreement story

- $N = 4$ Byzantine armies, physically separated
- Generals start with their own initial decisio
- They can communicate via $N \times N$
- They must reach a common decision
- Problem: among them there may be F Byzantine traitors, who may attempt to disrupt the agreement, by any means
- Deterministic agreement between loyal generals possible iff $N \geq 3F + 1$ and communications are synchronous

Byzantine agreement problem

- The N generals, basic story $N = 4$
- Complete graph K_N (loopbacks possible),

- rollback; binary: 1 or 0)

- Agreement required on one of their initial choices
- Generals should either all attack or all withdraw

Byzantine agreement problem

- However... among the N generals, there may be F traitors (faulty), thus only $N - F$ are loyal (non-faulty)

-

-

- We need two elves (loyals) for each orc plus on (loyal): $N \geq F + 2F + 1$ ☺

- Algorithms: Pease, Shostak, Lamport (1 Lamport, Shostak, Pease (1982).

- Impossibility results: Fischer, Lynch, Paterson (1985) – FLP

Byzantine failures

- A traitor can:
  - behave correctly (!)
  -
  -

- briefly: anything that could disrupt the agre
- The algorithm must cope with such extrem adversaries
- The purpose is NOT to identify the traitors, but to ensure that the system continues to work properly (all loyal guys)

Byzantine agreement conditions

- Termination: all non-faulty processes eventually decide
-
-

  value $v \in V$, then v is the only one pos

  [STRONG]

  - if the non-faulty processes start with diff then the final decision could be any of these (as long as it is consistent)

Byzantine agreement scenarios ($N = 4$)

Initial	Final	Notes
0 0 0 0	0 0 0 0	required
0 0 0 1	0 0 0 0	majority rule? NO, required (why?)
0 0 1 1	v v v v	depending on a parameter v_0
0 1 1 1		?)
1 1 1 1		
0 0 0 *		
0 0 1 *	0 0 0 * or 1 1 1 *	depending on parameter v_0 and the orc
0 1 1 *	0 0 0 * or 1 1 1 *	depen he orc
1 1 1 *	1 1 1 *	require

- The star (*) represents orc's arbitrary or malevolent choices
- The algorithm we study – EIG – uses an internal parameter, v_0, which (1) replaces missing or wrongly formatted messages, and (2) breaks ties

Informal example

- The following agreement is required, between the elves:
  - Left: #2 and #3 should decide 0.
  - Right: #1 and #2 should decide 1.
  - Middle: #1 and #3 should reach a consistent decision.

- The orc processes have a perfect disrupting strategy (next)

Informal example

-
  - Process #3 cannot differentiate betwe
    cases and should therefore take the same
    cases, i.e., 0

  - Process #1 cannot differentiate betwe
    cases and should therefore take the same decision in both cases, i.e., 1.
  - Thus, no common decision is possible for the middle case
- Conclusion: 1 round is not enough...

Informal example

- the value received from the other process on the 1st round:
  - Process #3 still cannot differentiate bet middle cases...
  - Process #1 still cannot differentiate bet middle cases...
  - Thus, no common decision is possible for the middle case
- Conclusion: 2 rounds are not enough... arguments can continue for any number of rounds...

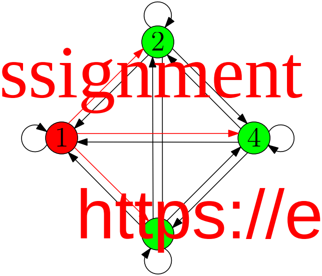
EIG tree

- EIG = Exponential Information Gathering
- Here, $F = 1$, $N = 3F + 1 = 4$, $L = F + 1 = 2$
- Description in Lynch's monograph

EIG tree

- Each non-faulty process maintains its own copy of the EIG tree
- The top-down val(α) attributes: first, the levels are filled top-down, according to received messages
-
- On each branch, there is at least one node with a label ending in the ID of a non-faulty node
- The first such nodes (top-down) are conne
- The nodes on or above the red cut are common: they have the same newval values, in all non-faulty processes
- Thus the final decision is common, for all non-faulty processes
- Full description in Lynch's monograph – also our demo

Faulty process ι_1 sends out conflicting messages

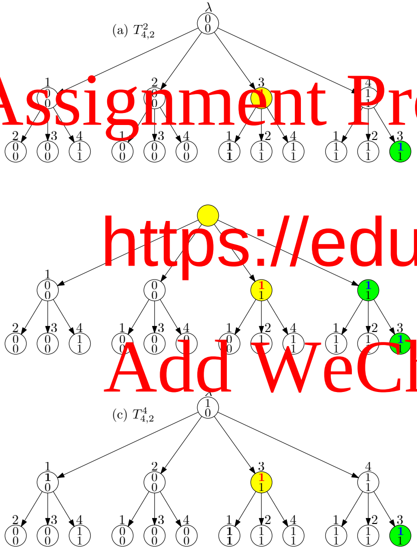


Process	ι_1	ι_2	ι_3	ι_4
Initial choice	?	0	1	1
Faulty	Yes	No	No	No
				$(4, 1)$
	$(3.1, y)$ $(4.$	$(3.2, 1)$		$(1.4, 1)$ $(2.4, 0)$ $(3.4, 1)$
.. Final decision				

- $x = 0, y = 1$ to process ι_2
- $x = 0, y = 0$ to process ι_3 – *try also* $x = 1, y = 0$
- $x = 1, y = 1$ to process ι_4

Non-faulty processes are always able to reach a common decision: *either* all 0, as here – *or* all 1

EIG trees for non-faulty processes



Process	ι_1	ι_2	ι_3	ι_4
Initial choice	?	0	1	1
Faulty	Yes	No	No	No
Round 1 messages	$(1, x)$	$(2, 0)$	$(3, 1)$	$(4, 1)$
			3, 0) 3, 0) 3, 1)	$(1.4, 1)$ $(2.4, 0)$ $(3.4,$ [illegible]$)$
... Final decisi				0

- $, \iota_4$
- L_2: (relay) $\iota_3 \overset{(4.3,1)}{\longrightarrow} \iota_2, \iota_3, \iota_4$
- β by bottom-up local voting
- common final decision

The top-down val() attribute

How val() are filled (example):

- val(2...) is about what #2 said
- val(2) is what #2 directly said
- val(21) is what #1 said that #2 said
- If #1 is lying about #2 in val(21), then #3 & #4 will “mask” this by val(23) & val(24)
- invalid or missing messages are assumed to be v_0

The bottom-up newval() attribute

newval()

- computed new value
-
- [illegible]
  - or v_0 if there is no majority

- this “masks” failures
  - if any – within the accepted limits ($n \geq 3f + 1$)

The bottom-up newval() attribute

Byzantine quiz

Byzantine quiz: decision 0

Byzantine quiz: decision 1

Byz vs Triple modular redundancy (TMR)