The Byzantine Agreement – An Introduction

Radu Nicolescu
Department of Computer Science
University of Auckland

25 July 2018
1. The Byzantine agreement problem
2. Informal example
3. EIG tree
4. Example
5. Attributes
6. Quiz
7. Triple modular redundancy
**Outline**

1. The Byzantine agreement problem
2. Informal example
3. EIG tree
4. Example
5. Attributes
6. Quiz
7. Triple modular redundancy
The Byzantine agreement problem

- Byzantium history...

- The $N$ generals, basic story $N = 4$
- Complete graph $K_N$ (loopbacks possible), with secure channels
- Generals’ initial choices can be different: attack or withdraw (database: commit or rollback; binary: 1 or 0)
- Agreement required on one of their initial choices
- Generals should either all attack or all withdraw
The Byzantine agreement problem

- Byzantium history...

- The $N$ generals, basic story $N = 4$
- **Complete** graph $K_N$ (loopbacks possible), with **secure** channels
  - Generals’ initial choices can be different: attack or withdraw (database: commit or rollback; binary: 1 or 0)
  - Agreement required on **one of their initial choices**
  - Generals should either **all attack** or **all withdraw**
The Byzantine agreement problem

- Byzantium history...

- The $N$ generals, basic story $N = 4$
- Complete graph $K_N$ (loopbacks possible), with secure channels
- Generals’ initial choices can be different: attack or withdraw (database: commit or rollback; binary: 1 or 0)
- Agreement required on one of their initial choices
- Generals should either all attack or all withdraw
The Byzantine agreement problem

- Byzantium history...

- The $N$ generals, basic story $N = 4$

- Complete graph $K_N$ (loopbacks possible), with secure channels

- Generals’ initial choices can be different: attack or withdraw (database: commit or rollback; binary: 1 or 0)

- Agreement required on one of their initial choices

- Generals should either all attack or all withdraw
The Byzantine agreement problem

• However... among the $N$ generals, there may be $F$ traitors, thus only $N - F$ are loyal

• Typically: $N = 4$, $F = 1$ (or, $N = 7$, $F = 2$)

• In fact, the problem can be solved iff $N \geq 3F + 1$ (we’ll prove this later)

• We need two elves (loyals) for each orc plus one hobbit (loyal): $N \geq F + 2F + 1$ 😊
The Byzantine agreement problem

- However... among the $N$ generals, there may be $F$ traitors, thus only $N - F$ are loyal.
- Typically: $N = 4, F = 1$ (or, $N = 7, F = 2$)
- In fact, the problem can be solved iff $N \geq 3F + 1$ (we’ll prove this later)
- We need two elves (loyals) for each orc plus one hobbit (loyal): $N \geq F + 2F + 1$ ☺
The Byzantine agreement problem

- However... among the $N$ generals, there may be $F$ traitors, thus only $N - F$ are loyal.
- Typically: $N = 4, F = 1$ (or, $N = 7, F = 2$)
- In fact, the problem can be solved iff $N \geq 3F + 1$ (we’ll prove this later)
- We need two elves (loyals) for each orc plus one hobbit (loyal): $N \geq F + 2F + 1$ 😊
The Byzantine agreement problem

- However... among the $N$ generals, there may be $F$ traitors, thus only $N - F$ are loyal
- Typically: $N = 4$, $F = 1$ (or, $N = 7$, $F = 2$)
- In fact, the problem can be solved iff $N \geq 3F + 1$ (we’ll prove this later)
- We need two elves (loyals) for each orc plus one hobbit (loyal): $N \geq F + 2F + 1$ ☺️
The Byzantine agreement problem

- A traitor can:
  - behave correctly (!)
  - stop cooperating (stop sending messages)
  - send confusing messages (different messages to different directions)
  - briefly: anything that could disrupt the agreement!
  - The algorithm must cope with such extremely malevolent adversaries
The Byzantine agreement problem

• A traitor can:
  
  • behave correctly (!)
  
  • stop cooperating (stop sending messages)
  
  • send confusing messages (different messages to different directions)
  
  • briefly: anything that could disrupt the agreement!

• The algorithm must cope with such extremely malevolent adversaries
The Byzantine agreement problem

- A traitor can:
  - behave correctly (!)
  - stop cooperating (stop sending messages)
    - send confusing messages (different messages to different directions)
  - briefly: anything that could disrupt the agreement!
  - The algorithm must cope with such extremely malevolent adversaries
The Byzantine agreement problem

- A traitor can:
  - behave correctly (!)
  - stop cooperating (stop sending messages)
  - send confusing messages (different messages to different directions)
  - briefly: anything that could disrupt the agreement!
  - The algorithm must cope with such extremely malevolent adversaries
The Byzantine agreement problem

- A traitor can:
  - behave correctly (!)
  - stop cooperating (stop sending messages)
  - send confusing messages (different messages to different directions)
  - briefly: anything that could disrupt the agreement!

- The algorithm must cope with such extremely malevolent adversaries
The Byzantine agreement problem

- A traitor can:
  - behave correctly (!)
  - stop cooperating (stop sending messages)
  - send confusing messages (different messages to different directions)
  - briefly: anything that could disrupt the agreement!
- The algorithm must cope with such extremely malevolent adversaries
The Byzantine agreement conditions

- **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 non-faulty processes start with different initial values, then the final decision could be any of these (as long as it is consistent)
The Byzantine agreement conditions

• **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 non-faulty processes start with different initial values, then the final decision could be any of these (as long as it is consistent)
The Byzantine agreement conditions

- **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 [STRONG]
  - if the non-faulty processes start with different initial values, then the final decision could be any of these (as long as it is consistent)
The Byzantine agreement conditions

- **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.

[STRONG]

- if the non-faulty processes start with different initial values, then the final decision could be any of these (as long as it is consistent)
The Byzantine agreement examples

<table>
<thead>
<tr>
<th>Initial</th>
<th>Final</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>required</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 0 0 0</td>
<td>majority rule? NO, required (why?)</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>v v v v v</td>
<td>depending on a parameter $\nu_0$</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 1 1 1</td>
<td>majority rule? NO, required (why?)</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
<td>required</td>
</tr>
<tr>
<td>* 0 0 0</td>
<td>* 0 0 0</td>
<td>required</td>
</tr>
<tr>
<td>* 0 0 1</td>
<td>* 0 0 0 or * 1 1 1</td>
<td>depending on parameter $\nu_0$ and the orc</td>
</tr>
<tr>
<td>* 0 1 1</td>
<td>* 0 0 0 or * 1 1 1</td>
<td>depending on parameter $\nu_0$ and the orc</td>
</tr>
<tr>
<td>* 1 1 1</td>
<td>* 1 1 1</td>
<td>required</td>
</tr>
</tbody>
</table>

- The star (*) represents orc’s arbitrary or malevolent choices
- The algorithm we study – EIG – uses an internal parameter, $\nu_0$, which (1) replaces missing or wrongly formatted messages, and (2) breaks ties
## The Byzantine agreement examples

<table>
<thead>
<tr>
<th>Initial</th>
<th>Final</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>required</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 0 0 0</td>
<td>majority rule? NO, required (why?)</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>v v v v</td>
<td>depending on a parameter ( \nu_0 )</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 1 1 1</td>
<td>majority rule? NO, required (why?)</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
<td>required</td>
</tr>
<tr>
<td>* 0 0 0</td>
<td>* 0 0 0</td>
<td>required</td>
</tr>
<tr>
<td>* 0 0 1</td>
<td>* 0 0 0 or * 1 1 1</td>
<td>depending on parameter ( \nu_0 ) and the orc</td>
</tr>
<tr>
<td>* 0 1 1</td>
<td>* 0 0 0 or * 1 1 1</td>
<td>depending on parameter ( \nu_0 ) and the orc</td>
</tr>
<tr>
<td>* 1 1 1</td>
<td>* 1 1 1</td>
<td>required</td>
</tr>
</tbody>
</table>

- The star (*) represents orc’s arbitrary or malevolent choices
- The algorithm we study – EIG – uses an internal parameter, \( \nu_0 \), which (1) replaces missing or wrongly formatted messages, and (2) breaks ties.
The Byzantine agreement examples

<table>
<thead>
<tr>
<th>Initial</th>
<th>Final</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>required</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 0 0 0</td>
<td>majority rule? NO, required (why?)</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>v v v v</td>
<td>depending on a parameter $\nu_0$</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 1 1 1</td>
<td>majority rule? NO, required (why?)</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
<td>required</td>
</tr>
<tr>
<td>* 0 0 0</td>
<td>* 0 0 0</td>
<td>required</td>
</tr>
<tr>
<td>* 0 0 1</td>
<td>* 0 0 0 or * 1 1 1</td>
<td>depending on parameter $\nu_0$ and the orc</td>
</tr>
<tr>
<td>* 0 1 1</td>
<td>* 0 0 0 or * 1 1 1</td>
<td>depending on parameter $\nu_0$ and the orc</td>
</tr>
<tr>
<td>* 1 1 1</td>
<td>* 1 1 1</td>
<td>required</td>
</tr>
</tbody>
</table>

- The star (*) represents orc’s arbitrary or malevolent choices
- The algorithm we study – EIG – uses an internal parameter, $\nu_0$, which (1) replaces missing or wrongly formatted messages, and (2) breaks ties
Outline

1. The Byzantine agreement problem
2. Informal example
3. EIG tree
4. Example
5. Attributes
6. Quiz
7. Triple modular redundancy
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

- 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)
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)
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

- 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)
Consider that they send to each other their initial values:

- Process #3 cannot differentiate between the left and middle cases and should therefore take the same decision in both cases, i.e., 0.
- Process #1 cannot differentiate between the right and middle 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**

- Consider that they send to each other their initial values:
  - Process #3 cannot differentiate between the left and middle cases and should therefore take the same decision in both cases, i.e., 0.
  - Process #1 cannot differentiate between the right and middle 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

Consider that they send to each other their initial values:

- Process #3 cannot differentiate between the left and middle cases and should therefore take the same decision in both cases, i.e., 0.
- Process #1 cannot differentiate between the right and middle 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

Consider that they send to each other their initial values:

- Process #3 cannot differentiate between the left and middle cases and should therefore take the same decision in both cases, i.e., 0.
- Process #1 cannot differentiate between the right and middle 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

Consider that they send to each other their initial values:

- Process #3 cannot differentiate between the left and middle cases and should therefore take the same decision in both cases, i.e., 0.
- Process #1 cannot differentiate between the right and middle 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...
Consider that on the 2nd round the elves relay to each other the value received from the other process on the 1st round:

- Process #3 still cannot differentiate between the left and middle cases...
- Process #1 still cannot differentiate between the right and 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...
Informal example

- Consider that on the 2nd round the elves relay to each other the value received from the other process on the 1st round:
  - Process #3 still cannot differentiate between the left and middle cases...
  - Process #1 still cannot differentiate between the right and 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...
Informal example

- Consider that on the 2nd round the elves relay to each other the value received from the other process on the 1st round:
  - Process #3 still cannot differentiate between the left and middle cases...
  - Process #1 still cannot differentiate between the right and 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...
• Consider that on the 2nd round the elves relay to each other the value received from the other process on the 1st round:
  • Process #3 still cannot differentiate between the left and middle cases...
  • Process #1 still cannot differentiate between the right and 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...
Consider that on the 2nd round the elves relay to each other the value received from the other process on the 1st round:

- Process #3 still cannot differentiate between the left and middle cases...
- Process #1 still cannot differentiate between the right and 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...
Outline

1. The Byzantine agreement problem
2. Informal example
3. EIG tree
4. Example
5. Attributes
6. Quiz
7. Triple modular redundancy
• EIG = Exponential Information Gathering

• Here, \( F = 1, N = 3F + 1 = 4, L = F + 1 = 2 \)

• Description in Lynch’s monograph
• EIG = Exponential Information Gathering

• Here, $F = 1$, $N = 3F + 1 = 4$, $L = F + 1 = 2$

• Description in Lynch’s monograph
• 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 \texttt{val} ($\alpha$) attributes: first, the levels are filled top-down, according to received messages
  - The bottom-up \texttt{newval} ($\beta$) attributes: next, the levels are recomputed bottom-up, without messaging, according to a local majority rule
  - 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 connected by a red cut
  - The nodes on or above the red cut are **common**: they have the same \texttt{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
EIG tree

- Each non-faulty process maintains its own copy of the EIG tree.
- The top-down val ($\alpha$) attributes: first, the levels are filled top-down, according to received messages.
- The bottom-up newval ($\beta$) attributes: next, the levels are recomputed bottom-up, without messaging, according to a local majority rule.
- 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 connected by a red cut.
- 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.
EIG tree

- Each non-faulty process maintains its own copy of the EIG tree.
- The top-down val ($\alpha$) attributes: first, the levels are filled top-down, according to received messages.
- The bottom-up newval ($\beta$) attributes: next, the levels are recomputed bottom-up, without messaging, according to a local majority rule.
- 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 connected by a red cut.
- 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.
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.
- The bottom-up newval (β) attributes: next, the levels are recomputed bottom-up, without messaging, according to a local majority rule.
- 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 connected by a red cut.
- 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.
EIG tree

- Each non-faulty process maintains its own copy of the EIG tree.
- The top-down val (\(\alpha\)) attributes: first, the levels are filled top-down, according to received messages.
- The bottom-up newval (\(\beta\)) attributes: next, the levels are recomputed bottom-up, without messaging, according to a local majority rule.
- 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 connected by a red cut.
- 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.
EIG tree

- Each **non-faulty** process maintains its own copy of the EIG tree.
- The top-down **val** ($\alpha$) attributes: first, the levels are filled top-down, according to received messages.
- The bottom-up **newval** ($\beta$) attributes: next, the levels are recomputed bottom-up, without messaging, according to a local majority rule.
- 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 connected by a **red cut**.
- 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.
EIG tree

- Each **non-faulty** process maintains its own copy of the EIG tree
- The top-down **val** ($\alpha$) attributes: first, the levels are filled top-down, according to received messages
- The bottom-up **newval** ($\beta$) attributes: next, the levels are recomputed bottom-up, without messaging, according to a local majority rule
- 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 connected by a **red cut**
- 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
EIG tree

- Each non-faulty process maintains its own copy of the EIG tree
- The top-down val ($\alpha$) attributes: first, the levels are filled top-down, according to received messages
- The bottom-up newval ($\beta$) attributes: next, the levels are recomputed bottom-up, without messaging, according to a local majority rule
- 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 connected by a red cut
- 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
Outline

1. The Byzantine agreement problem
2. Informal example
3. EIG tree
4. Example
5. Attributes
6. Quiz
7. Triple modular redundancy
The Byzantine agreement example

- $N = 4$ Byzantine armies, physically separated
- Generals start with their own initial decisions, 0 or 1
- They can communicate via $N(N - 1)/2 = 6$ reliable channels
- They **must** reach a common decision
- Problem: among them there may be $F$ Byzantine traitors
- Deterministic agreement between loyal generals possible iff $N \geq 3F + 1$ and communications are synchronous

Pease, Shostak, Lamport 1980; Lamport, Shostak, Pease 1982; Fischer, Lynch, Paterson 1985
The Byzantine agreement example

- \( N = 4 \) Byzantine armies, physically separated
- Generals start with their own initial decisions, 0 or 1
- They can communicate via \( N(N - 1)/2 = 6 \) reliable channels
- They must reach a common decision
- Problem: among them there may be \( F \) Byzantine traitors
- Deterministic agreement between loyal generals possible iff \( N \geq 3F + 1 \) and communications are synchronous

Pease, Shostak, Lamport 1980; Lamport, Shostak, Pease 1982; Fischer, Lynch, Paterson 1985
The Byzantine agreement example

- $N = 4$ Byzantine armies, physically separated
- Generals start with their own initial decisions, 0 or 1
- They can communicate via $N(N - 1)/2 = 6$ reliable channels
- They must reach a common decision
- Problem: among them there may be $F$ Byzantine traitors
- Deterministic agreement between loyal generals possible iff $N \geq 3F + 1$ and communications are synchronous

Pease, Shostak, Lamport 1980; Lamport, Shostak, Pease 1982; Fischer, Lynch, Paterson 1985
Faulty process $\nu_1$ sends out conflicting messages

![Diagram of faulty process communication]

<table>
<thead>
<tr>
<th>Process</th>
<th>$\nu_1$</th>
<th>$\nu_2$</th>
<th>$\nu_3$</th>
<th>$\nu_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial choice</td>
<td>?</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Faulty</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Round 1 messages</td>
<td>(1, $x$)</td>
<td>(2, 0)</td>
<td>(3, 1)</td>
<td>(4, 1)</td>
</tr>
<tr>
<td>Round 2 messages</td>
<td>(2.1, 0)</td>
<td>(1.2, 0)</td>
<td>(1.3, 0)</td>
<td>(1.4, 1)</td>
</tr>
<tr>
<td></td>
<td>(3.1, $y$)</td>
<td>(3.2, 1)</td>
<td>(2.3, 0)</td>
<td>(2.4, 0)</td>
</tr>
<tr>
<td></td>
<td>(4.1, 1)</td>
<td>(4.2, 1)</td>
<td>(4.3, 1)</td>
<td>(3.4, 1)</td>
</tr>
<tr>
<td>... Final decision</td>
<td>?</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- $x = 0, y = 1$ to process $\nu_2$
- $x = 0, y = 0$ to process $\nu_3$ – try also $x = 1, y = 0$
- $x = 1, y = 1$ to process $\nu_4$

Non-faulty processes are always able to reach a common decision: either all 0, as here – or all 1.
Faulty process $\iota_1$ sends out conflicting messages

```
\begin{align*}
\text{Round 1 messages} &\quad \iota_1 &\quad \iota_2 &\quad \iota_3 &\quad \iota_4 \\
(1, x) &\quad (2, 0) &\quad (3, 1) &\quad (4, 1) \\
\text{Round 2 messages} &\quad (2.1, 0) &\quad (1.2, 0) &\quad (1.3, 0) &\quad (1.4, 1) \\
(3.1, y) &\quad (3.2, 1) &\quad (2.3, 0) &\quad (2.4, 0) \\
(4.1, 1) &\quad (4.2, 1) &\quad (4.3, 1) &\quad (3.4, 1) \\
\text{... Final decision} &\quad ? &\quad 0 &\quad 0 &\quad 0 \\
\end{align*}
```

- $x = 0, y = 1$ to process $\iota_2$
- $x = 0, y = 0$ to process $\iota_3$ – try also $x = 1, y = 0$
- $x = 1, y = 1$ to process $\iota_4$

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

<table>
<thead>
<tr>
<th>Process</th>
<th>( \nu_1 )</th>
<th>( \nu_2 )</th>
<th>( \nu_3 )</th>
<th>( \nu_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial choice</td>
<td>?</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Faulty</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Round 1 messages</td>
<td>(1, ( x ))</td>
<td>(2, 0)</td>
<td>(3, 1)</td>
<td>(4, 1)</td>
</tr>
<tr>
<td>Round 2 messages</td>
<td>(2.1, 0)</td>
<td>(1.2, 0)</td>
<td>(1.3, 0)</td>
<td>(2.4, 0)</td>
</tr>
<tr>
<td>(3.1, ( y ))</td>
<td>(3.2, 1)</td>
<td>(2.3, 0)</td>
<td>(3.4, 1)</td>
<td></td>
</tr>
<tr>
<td>(4.1, 1)</td>
<td>(4.3, 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... Final decision</td>
<td>?</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- \( \alpha \) by top-down messaging
- \( L_1: \) (initial) \( \nu_3 \rightarrow (3,1) \nu_2, \nu_3, \nu_4 \)
- \( L_2: \) (relay) \( \nu_3 \rightarrow (4.3,1) \nu_2, \nu_3, \nu_4 \)
- \( \beta \) by bottom-up local voting
- common final decision
EIG trees for non-faulty processes

(a) $T_{4,2}^2$

(b) $T_{4,2}^3$

(c) $T_{4,2}^4$

<table>
<thead>
<tr>
<th>Process</th>
<th>$\nu_1$</th>
<th>$\nu_2$</th>
<th>$\nu_3$</th>
<th>$\nu_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial choice</td>
<td>?</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Faulty</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Round 1 messages</td>
<td>(1, x)</td>
<td>(2, 0)</td>
<td>(3, 1)</td>
<td>(4, 1)</td>
</tr>
<tr>
<td>Round 2 messages</td>
<td>(2.1, 0)</td>
<td>(1.2, 0)</td>
<td>(1.3, 0)</td>
<td>(1.4, 1)</td>
</tr>
<tr>
<td>(3.1, y)</td>
<td>(3.2, 1)</td>
<td>(2.3, 0)</td>
<td>(2.4, 0)</td>
<td></td>
</tr>
<tr>
<td>(4.1, 1)</td>
<td>(4.2, 1)</td>
<td>(4.3, 1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... Final decision | ? | 0 | 0 | 0 |

- $\alpha$ by top-down messaging
- $L_1$: (initial) $\nu_3 \xrightarrow{(3,1)} \nu_2, \nu_3, \nu_4$
- $L_2$: (relay) $\nu_3 \xrightarrow{(4.3,1)} \nu_2, \nu_3, \nu_4$
- $\beta$ by bottom-up local voting
- common final decision
EIG trees for non-faulty processes

<table>
<thead>
<tr>
<th>Process</th>
<th>$\nu_1$</th>
<th>$\nu_2$</th>
<th>$\nu_3$</th>
<th>$\nu_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial choice</td>
<td>?</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Faulty</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Round 1 messages</td>
<td>$(1, x)$</td>
<td>$(2, 0)$</td>
<td>$(3, 1)$</td>
<td>$(4, 1)$</td>
</tr>
<tr>
<td>Round 2 messages</td>
<td>$(2.1, 0)$</td>
<td>$(1.2, 0)$</td>
<td>$(3.1, y)$</td>
<td>$(4.1, 1)$</td>
</tr>
<tr>
<td></td>
<td>$(3.1, y)$</td>
<td>$(3.2, 1)$</td>
<td>$(4.2, 1)$</td>
<td>$(4.3, 1)$</td>
</tr>
<tr>
<td>... Final decision</td>
<td>?</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- $\alpha$ by top-down messaging
- $L_1$: (initial) $\nu_3 \rightarrow \nu_2, \nu_3, \nu_4$
- $L_2$: (relay) $\nu_3 \rightarrow \nu_2, \nu_3, \nu_4$
- $\beta$ by bottom-up local voting
- common final decision
Outline

1. The Byzantine agreement problem
2. Informal example
3. EIG tree
4. Example
5. Attributes
6. Quiz
7. Triple modular redundancy
The top-down \texttt{val()} attribute

How \texttt{val()} are filled (example):

- \texttt{val(2...) is about what \#2 said}
- \texttt{val(2)} is what \#2 directly said
- \texttt{val(21)} is what \#1 said that \#2 said
- If \#1 is lying about \#2 in \texttt{val(21)}, then \#3 & \#4 will “mask” this by \texttt{val(23)} & \texttt{val(24)}
- Invalid or missing messages are assumed to be \texttt{\nu_0}
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 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 \nu_0
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 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 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

- no messaging anymore

- decision taken by a local majority voting procedure
  - 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

newval()

- computed new value
- no messaging anymore
- decision taken by a local majority voting procedure
  - 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

newval()

- computed new value
- no messaging anymore
  - decision taken by a local majority voting procedure
    - or, $v_0$, if there is no majority
- this “masks” failures
  - if any – within the accepted limits ($n \geq 3f + 1$)
newval()

- computed new value
- no messaging anymore
- decision taken by a local majority voting procedure
  - 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

newval()

- computed new value
- no messaging anymore
- decision taken by a local majority voting procedure
  - 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
Outline

1. The Byzantine agreement problem
2. Informal example
3. EIG tree
4. Example
5. Attributes
6. Quiz
7. Triple modular redundancy
Assume that this is the EIG tree at a non-faulty elf process $i = 2, 3, 4$; $v_0 = 0$; and #1 is a Byz orc

\[ \lambda_i: -: ? \]

For each elf tree $i$, replace $W_i, X_i$ & $Y_i$, s.t. the final decision $\lambda_i$ becomes either (1) 0; or (2) 1

Why shouldn’t we care about the $Z_i$ values?

\[ 1: W_i; Y_i \]
\[ 2: 0: 0 \]
\[ 3: 1: 1 \]
\[ 4: 1: 1 \]

\[ 12: 0: 0 \]
\[ 13: 1: 1 \]
\[ 14: X_i; X_i \]
\[ 21: Z_i; Z_i \]
\[ 22: 0: 0 \]
\[ 23: 0: 0 \]
\[ 24: 0: 0 \]

\[ 31: Z_i'; Z_i' \]
\[ 32: 1: 1 \]
\[ 33: 1: 1 \]
\[ 34: 1: 1 \]

\[ 41: Z_i''; Z_i'' \]
\[ 42: 1: 1 \]
\[ 43: 1: 1 \]

\textbf{Val()} could be distinct at each process

\textbf{Val()} can be changed by the orc, but will still be common
Byzantine quiz: decision 0

final decision = 0

λ: -: 0

- W_2 = 0, for #2 (inferred from 12)
- W_3 = 1, for #3 (inferred from 13)
- W_4 = 0 = X_4, for #4 (to get Y_4 = 0)

1: W_i: 0

12: 0: 0

13: 1: 1

14: 0: 0

21: Z_i: Z_i

23: 0: 0

24: 0: 0

31: Z_i': Z_i'

32: 1: 1

34: 1: 1

41: Z_i'': Z_i''

42: 1: 1

43: 1: 1
Byzantine quiz: decision 1

1: \( W_i: 1 \)

2: \( 0: 0 \)

3: \( 1: 1 \)

4: \( 1: 1 \)

Final decision = 1

- \( W_2 = 0 \) (inferred from 12)
- \( W_3 = 1 \) (inferred from 13)
- \( W_4 = 1 = X_4 \) (for #4 to get \( Y_4 = 1 \))
Outline

1. The Byzantine agreement problem
2. Informal example
3. EIG tree
4. Example
5. Attributes
6. Quiz
7. Triple modular redundancy
Byz vs Triple modular redundancy (TMR)

**Byzantine agreement vs TMR** (more in text).

- In Byz context: Non-faulty modules may well generate different initial values.
- In TMR: We expect that all non-faulty modules generate the same initial value. Only a faulty module will generate a different initial value.
- Can we trust the comparators?