reaching agreement in the presence of faults

Written By: Marshall Pease, Robert Shostak, Leslie Lamport

Lecturer in charge: Oded Shmueli

Presentation By: Shahar Yair
The Two Generals Problem

Two generals need to agree on whether to attack the enemy city or retreat. If they come to an agreement then the war is either won or there will be peace. But if one attacks and the other retreats, then the war is lost and everyone is sad 😞.
TCP/IP?

If you respond, I’ll attack!
If you respond, I’ll attack!
We will attack!
There is no perfect solution to the problem!
The Byzantine Agreement Problem
The Byzantine Agreement Problem

“We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. The generals can communicate with one another only by messengers. After observing the enemy, they must decide upon a common plan of action. However, some of the generals may be traitors, trying to prevent the loyal generals from reaching agreement.”
The Basic Byzantine Problem

Assumptions:
• The Network works perfectly.
• There are N=4 generals.
• Each general has a unique name (color).
• One or more of the generals may be a traitor (Red). We’ll say there are M=1 traitors for now.
• All of the non-traitor generals must come to an agreement or we lose the war.
• Assume a previously appointed Super-General.
• Binary Values (A/R).
The Problem With $N=4$, $M=1$
The Problem With N=4, M=1

[Yellow, Green, Red]
The Problem With $N=4$, $M=1$

We win the war!
What If the Super-General is a Traitor?

Rettack!!

Retreat!

Attack!

Attack!
What If the Super-General is a Traitor?

[Yellow, Green, Blue]

[A,A,R] 

Retreat!

Attack!

[A,A,R]

Retreat!

Attack!

[A,A,R]

Retreat!

Attack!

[A,A,R]

Retreat!

Attack!
What If the Super-General is a Traitor?

We win the war again!


The General Byzantine Problem

Assumptions:
• The Network works perfectly.
• There are $N$ processors (shown as nodes).
• Each processor has a unique ID, $(ID_i)$.
• There are $M=1$ faulty processors.
• All of the non-faulty processors must come to an agreement.
• The system is asynchronous.
The Byzantine Agreement
The Byzantine Agreement
The Byzantine Agreement
The Byzantine Agreement
The Byzantine Agreement

\[ [V_1, V_2, V_3, V_4] \]  \[ [V_1, V_2, V_3, \text{NIL}] \]  
\[ [V_1, V_2, V_3, V_5] \]  \[ [\$#@!] \]
The Byzantine Agreement

[ID1, ID2, ID3, ID4]

[V1, V2, V3, V4]  [V1, V2, V3, NIL]

[ID1]  [ID2]

[V1, V2, V3, V5]  [$#@!] [ID4]
The Byzantine Agreement
The Byzantine Agreement

[ID1, ID2, ID3, ID4]

[ID1, ID2, ID3, NIL]

[ID3, ID4]

[ID3, ID4, "$#@!"]
The Byzantine Agreement
Remarks

• For M faulty processors, M+1 rounds are required. The first round is for exchanging the self values, and the others are for “processor X told me…” (We’ll see an example) That means, that the algorithm requires \(O((N-1)^{M+1})\) exchanges – Extremely inefficient!

• The problem is only solvable if \(N \geq 3M+1\).
The Procedure for $N \geq 3M + 1$

The problem is solvable only if $N \geq 3M + 1$

Definitions:
- $P = \{p_1, p_2, p_3, \ldots, p_n\}$: Set of processors
- $V = \{v_1, v_2, v_3, \ldots, v_n\}$: Set of values

We will define a scenario $\sigma$ as a mapping from an alphabet $\Sigma = P = \{p_1, p_2, p_3, \ldots, p_n\}$, to $V$.

Example: For a scenario $\sigma$ and string $w = p_1 p_2 \ldots p_r$
- $\sigma(w) = \text{the value of } p_r \text{ that } p_r \text{ tells } p_{r-1} \ldots \text{ which } p_2 \text{ tells } p_1$.

* Notice that for an arbitrary processor $p$,
a nonfaulty processor $q$ and $w$ over $P$:
- $\sigma(pqw) = \sigma(qw)$
The message a processor $p$ receives in a scenario $\sigma$ are given by the restriction $\sigma_p$ of $\sigma$ to strings beginning with $p$.

For some subset $Q$ of $P$ so that $\text{size}(Q) \geq (N+M)/2$:

* If $\sigma_p(pwq) = v$ for each string $w$ over $Q \Rightarrow p$ records the value of $q$ as $v$.

* Otherwise, the algorithm for $m-1$, $n-1$ is recursively applied with $P$ replaced by $P-\{q\}$ and $\sigma_p(pwq)$ by $\sigma_p(pw)$. If at least $(N+M)/2$ of the $n-1$ elements in the vector obtained by the recursive call agree $p$ records the value of $q$ as $v$, otherwise it records NIL.
If you recall...

\[ \sigma(P_1 P_3 P_4) = V_5 \] is the value of \( P_4 \) that \( P_4 \) told \( P_3 \) that \( P_3 \) told \( P_1 \)

*Notice: \( \sigma(P_1 P_3 P_4) = \sigma(P_3 P_4) = V_5 \)
What about $N<3M+1$?

We can prove that it is impossible to solve the problem for $N < 3M+1$.

Intuition for why it is true:
What if we restrict the values the faulty processor can send?

What is an authenticator?
An authenticator is a redundant augment to a data item that can be created, ideally, only by the originator of the data. A processor $p$ constructs an authenticator for a data item $d$ by calculating $A_p[d]$, where $A_p$ is some mapping known only to $p$. At the same time, it must be easy for $q$ to check, for a given $p$, $v$, and $d$, that $v = A_p[d]$. 
Using Authenticators

Now, the faulty processor has to tell the truth about the values it has received.

* But it can still lie about its own value.

Implication: No restriction on M.
Authenticators - Example

Diagram:
- $P_1$
- $P_3$
- $V_1$
Authenticators - Example

Diagram:
- Node P₁
- Node P₂
- Node P₃
- Edge V₃ from P₃ to P₁
- Edge V₃ from P₁ to P₂
- Edge V₃ from P₃ to P₂
Authenticators - Example

- $P_1$ with $[V_1, V_{21}, V_3]$
- $P_3$ with $[V_1, V_{23}, V_3]$
- $P_2$ with $[V_1, V_{2?}, V_3]$

- $[V_1, V_{2?}, V_3]$
Notice that $P_2$ has to send the same value it sent before!
Authenticators - Example

But we don’t care...
Authenticators - Example

Again, P₂ has to send the same value it sent before to P₃!
Authenticators - Example

P₁
[V₁,NIL,V₃]

P₂
[V₁,NIL,V₃]

P₃
[V₁,NIL,V₃]
We saw that the algorithm is VERY inefficient \(((N-1)^{M+1})\). So how can we improve its efficiency?

A sub committee!
Expansions of the Byzantine Problem

Possible properties to play with:

- Synchronous / Asynchronous (no time limit).
- Kind of fault: Fail-stop / Byzantine.
- Computationally bounded adversary/ private channels/ full information model.
- Static / dynamic adversary (picks who to corrupt during the protocol).
- Message passing / shared registers/ broadcasts.
- Completely connected / sparse network.
- Leader election / Global coin toss.
- Resiliency, Time and Bit complexity.
- Renormalized / Quantum (Hint: QVSS = Quantum Verifiable Secret Sharing protocol).
Questions?