Byzantine Agreement

Dipendra K. Misra

Cornell University

dkm@cs.cornell.edu

20th October 2015

Overview

Failure Models

- Fail stop
- Fail crash (Paxos)
- Byzantine Failure

Terminology

Byzantine Fault
Running system can arbitrarily deviate from its protocol.

System can lie, conspire, send wrong messages etc.

Byzantine Failure
The loss of a system service due to a Byzantine fault in systems that require consensus. (Driscoll et al. 2003)

Terminology

Byzantine Fault
Running system can arbitrarily deviate from its protocol.

System can lie, conspire, send wrong messages etc.

Byzantine Failure
The loss of a system service due to a Byzantine fault in systems that require consensus. (Driscoll et al. 2003)

Worst type of failure

Motivating Problem

- You are managing a critical system (power grid, ballistic missile shield)
- There are several systems each listening to input from its sensors/radar or a common source.

Motivating Problem

- You are managing a critical system (power grid, ballistic missile shield).
- There are several systems each listening to input from its sensors/radar or a common source.
- Systems should achieve consensus
  - reduce the load or do not reduce it.
  - fire all missiles at the enemy or fire none.
- Be able to handle a few sensors/radar or systems behaving arbitrarily.

Several Possibilities

1. Single faulty input source, giving different input to different systems.
2. Different input sources with some of them being faulty.
3. Single faulty input source which is consistently lying. [Cannot do anything here]
4. A system getting hacked or corrupt but keeps running.

Situation 1,2,4 come under *Byzantine failure*.

Several Possibilities

1. Single faulty input source, giving different input to different systems.
2. Different input sources with some of them being faulty.
3. Single faulty input source which is consistently lying. [Cannot do anything here]
4. A system getting hacked or corrupt but keeps running.

Situation 1,2,4 come under *Byzantine failure*.

Observation:

- Cannot use majority voting.
- No way to achieve consensus without systems talking to each other.
- Need to tell each other what they observed.

Problem Statement

System:

- Directed graph
- Nodes are devices/processes/complex systems
- Every node has an input
- Edges represent communication

Problem Statement

Byzantine Agreement:
Let there be protocol A_u for every node u in the system.

Every correct node follows the protocol.

Protocols solve the Byzantine Agreement iff

Agreement: Every correct node chooses the same value.

Validity: If all the correct nodes have the same input then that input must be the value chosen.

Impossibility Theorem

Intuition: Consensus should be possible with sufficiently few faulty nodes.

Maybe $2f + 1$ as majority $(f + 1)$ of nodes are not faulty.

Impossibility Theorem

Intuition: Consensus should be possible with sufficiently few faulty nodes.

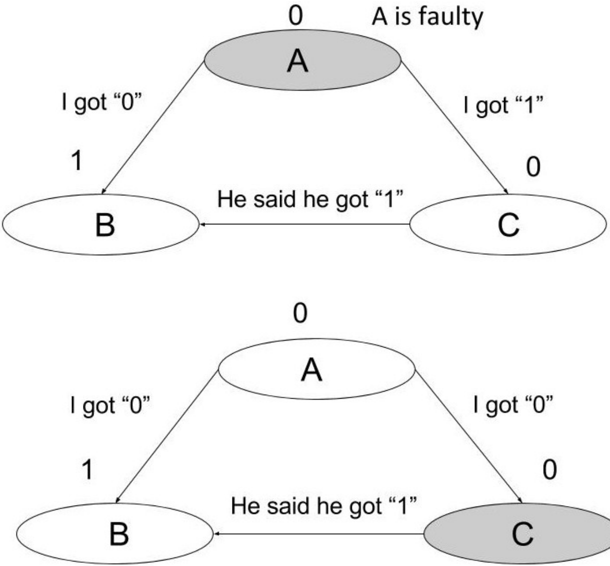
Maybe $2f + 1$ as majority $(f + 1)$ of nodes are not faulty.

Theorem

In order to tolerate f Byzantine faulty nodes, one needs $n \geq 3f + 1$ systems.

Intuition

Special Case: Consensus not possible in 3 systems if 1 is faulty.

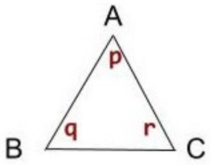


Formal Proof

Special Case: Consensus is not possible with 3 nodes when 1 is faulty.

Known as the *three general problem*.

Say there is a protocol for node p, q, r which solves the problem.

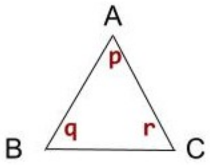


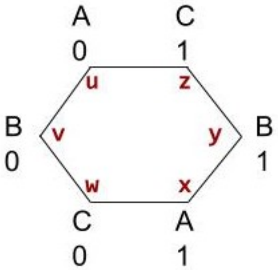
Protocol should work any input and atmost one faulty node.

Formal Proof

Special Case: Consensus is not possible with 3 nodes when 1 is faulty.

Let us say there is a protocol for A, B, C which solves the problem.



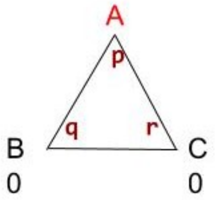


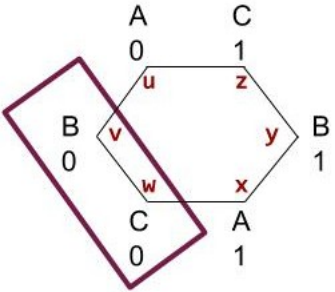
Derive contradiction from a construction.

Formal Proof (Special Case)

Case 1: Consider the nodes v and w

Same condition as q, r with p as Byzantine.

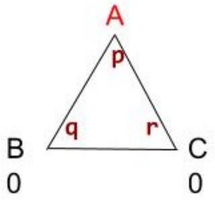


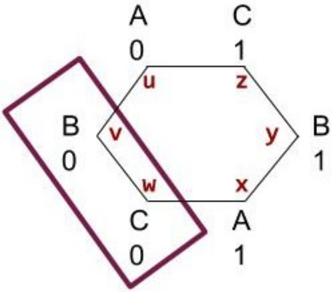


Formal Proof (Special Case)

Case 1: Consider the nodes v and w

Same condition as q, r with p as Byzantine.



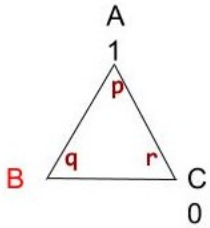


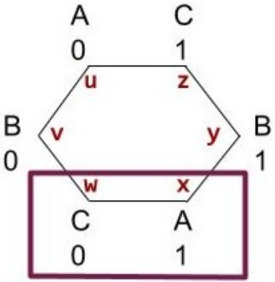
Validity dictates that q, r decide 0 and hence v, w must decide 0.

Formal Proof (Special Case)

Case 2: Consider the nodes w and x

Same condition as p, r with q as Byzantine.

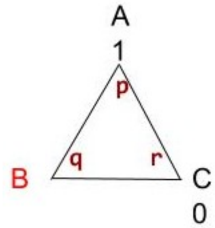


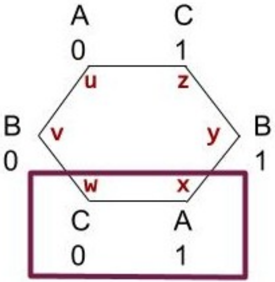


Formal Proof (Special Case)

Case 2: Consider the nodes w and x

Same condition as p, r with q as Byzantine.





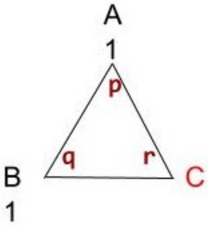
Agreement dictates that p, r decide one value.

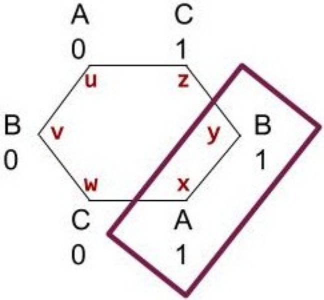
As w decides 0 hence x decides 0.

Formal Proof (Special Case)

Case 3: Consider the nodes x and y

Same condition as p, q with r as Byzantine.

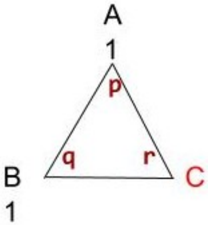


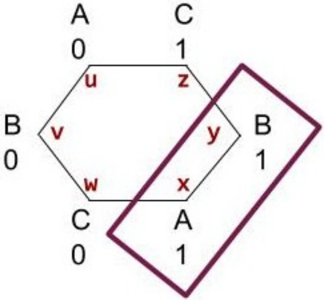


Formal Proof (Special Case)

Case 3: Consider the nodes x and y

Same condition as p, q with r as Byzantine.



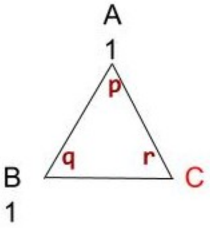


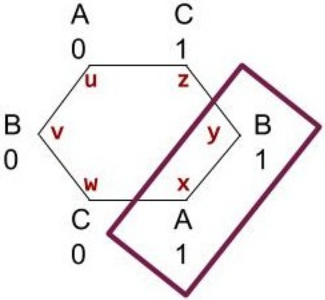
Validity dictates that p, q must decide 1 hence x, y must decide 1.

Formal Proof (Special Case)

Case 3: Consider the nodes x and y

Same condition as p, q with r as Byzantine.



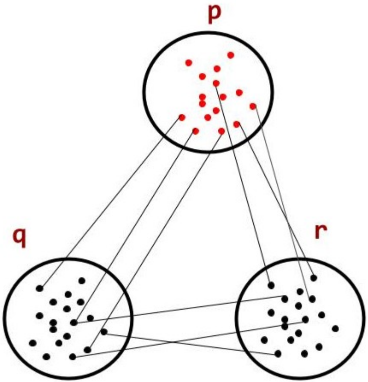


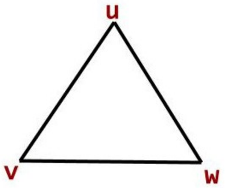
Validity dictates that p, q must decide 1 hence x, y must decide 1.

Wait! we already concluded that x must decide 0

Formal Proof (General Case)

- Say a protocol achieves agreement with $\leq 3f$ nodes ($\leq f$ are faulty).
- Create 3 groups p, q, r containing atmost f nodes each.
- w.l.o.g. all faulty nodes reside in group p.
- Simulate solution for *3 general problem*.

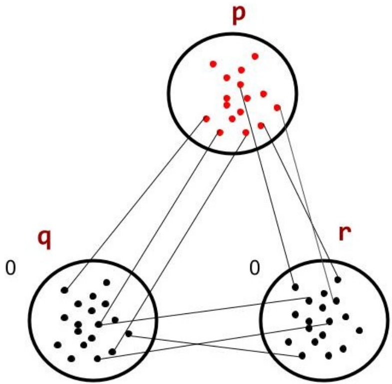


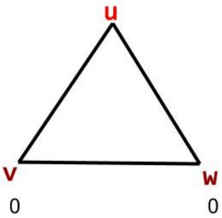


Formal Proof (General Case)

Simulating solution for 3 general problem

- u, v, w simulate group p, q, r resp.
- Given input 0 to node v, w run the protocol with input to all nodes in q, r as 0.

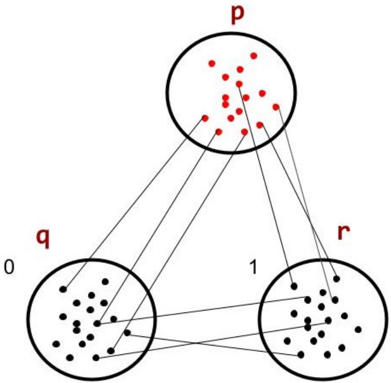


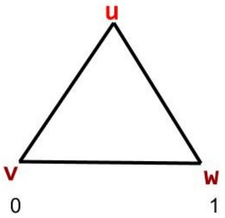


- Eventually all nodes in q, r accept 0 hence v, w accept 0.

Formal Proof (General Case)

- Do similarly when v, w are given input as $0, 1$ resp.





We have found a solution to three general problem. Contradiction.

So how to achieve agreement when $n \geq 3f + 1$

Oral Message Algorithm

Due to Lamport, Shostak and Pease (1982)

Assumption

- Every message that is sent is delivered correctly.
- The receiver of a message knows who sent it.
- The absence of a message can be detected.

So how to achieve agreement when $n \geq 3f + 1$

Oral Message Algorithm
Due to Lamport, Shostak and Pease (1982)

Assumption

- Every message that is sent is delivered correctly.
- The receiver of a message knows who sent it.
- The absence of a message can be detected.

Are these assumptions realistic?

Rephrasing the problem

- System as a graph with nodes taking input.
- **Agreement:** All correct nodes accept same value.
- **Validity:** If all correct nodes have the same input, that input must be the value accepted.

Rephrasing the problem

- System as a graph with nodes taking input.
- **Agreement:** All correct nodes accept same value.
- **Validity:** If all correct nodes have the same input, that input must be the value accepted.

can be reformulated as

- Commander node sending order to a set of lieutenant nodes in a graph.
- **Agreement:** All correct lieutenant nodes accept the same value.
- **Validity:** If the commander is loyal then every loyal lieutenant obeys the order he/she sends.

Rephrasing the problem

- System as a graph with nodes taking input.
- **Agreement:** All correct nodes accept same value.
- **Validity:** If all correct nodes have the same input, that input must be the value accepted.

can be reformulated as

- Commander node sending order to a set of lieutenant nodes in a graph.
- **Agreement:** All correct lieutenant nodes accept the same value.
- **Validity:** If the commander is loyal then every loyal lieutenant obeys the order he/she sends.

From formulation 2 to 1

1. Input to a node is then the order given by the commander.
2. Loyal commander orders and obeys the input given to it.

Problem Statement (Fully Connected Graph)

There are n nodes in a fully connected graph.

One node is a commander and remaining are lieutenants.

Find a protocol for every node such that following holds:

- **Agreement:** All correct lieutenant nodes accept the same value.
- **Validity:** If the commander is loyal then every loyal lieutenant obeys the order he/she sends.

Oral Message Algorithm

Algorithm $OM(0)$

- The commander sends his/her value to every lieutenant.
- Each lieutenant uses the value he/she receives from the commander.

Oral Message Algorithm

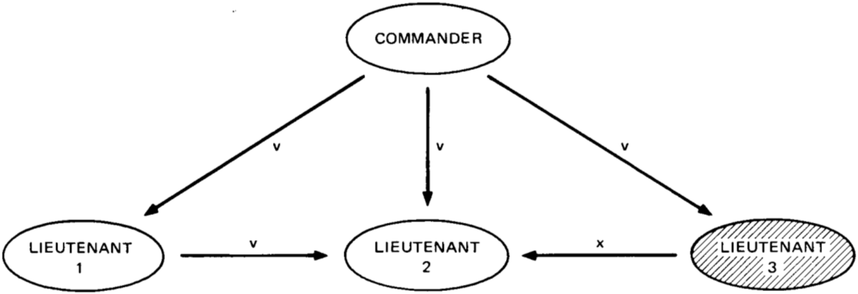
Algorithm $OM(0)$

- The commander sends his/her value to every lieutenant.
- Each lieutenant uses the value he/she receives from the commander.

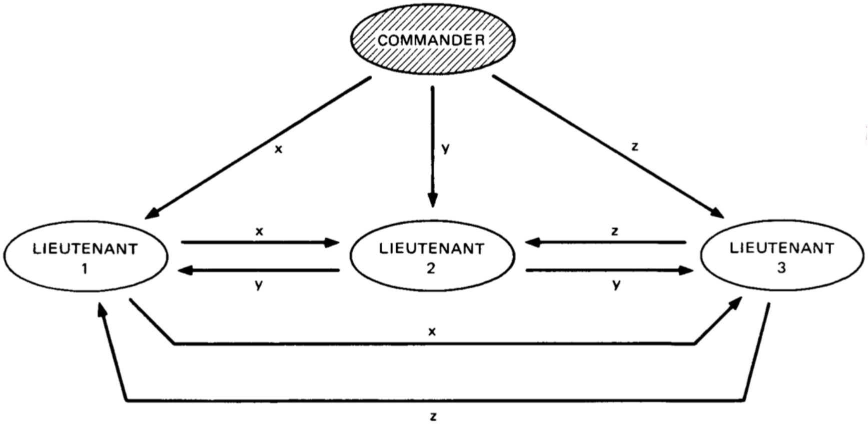
Algorithm $OM(m), m > 0$

- The commander sends his/her value to every lieutenant.
- For each i, let v_i be the value Lieutenant received from the commander else RETREAT if no value is received. Lieutenant acts as the commander and sends the value v_i to each of the $n-2$ other lieutenants using $OM(m-1)$.
- For each i, and each $j \neq i$, let v_j be the value lieutenant received from Lieutenant j in step(2) or else RETREAT if he received no such value. Lieutenant i uses the value $majority\{v_1, v_2, \cdots, v_{n-1}\}$.

Oral Message Algorithm $OM(1)$



Oral Message Algorithm $OM(1)$



OM Algorithm: Proof of Correctness

Lemma

For any m, k Algorithm $OM(m)$ satisfies validity if there are more than $2k + m$ generals and at most k traitors.

OM Algorithm: Proof of Correctness

Lemma

For any m, k Algorithm $OM(m)$ satisfies validity if there are more than $2k + m$ generals and at most k traitors.

- Validity is defined when commander is loyal.

OM Algorithm: Proof of Correctness

Lemma

For any m, k Algorithm $OM(m)$ satisfies validity if there are more than $2k + m$ generals and at most k traitors.

- Validity is defined when commander is loyal.
- Induction on m. For $m = 0$, its trivial.

OM Algorithm: Proof of Correctness

Lemma

For any m, k Algorithm $OM(m)$ satisfies validity if there are more than $2k + m$ generals and at most k traitors.

- Validity is defined when commander is loyal.
- Induction on m. For $m = 0$, its trivial.
- Assume hypothesis works for $m' < m$.

OM Algorithm: Proof of Correctness

Lemma

For any m, k Algorithm $OM(m)$ satisfies validity if there are more than $2k + m$ generals and at most k traitors.

- Validity is defined when commander is loyal.
- Induction on m. For $m = 0$, its trivial.
- Assume hypothesis works for $m' < m$.
- In step 1, loyal commander sends value v to $n - 1$ lieutenant.
- In step 2, loyal lietuenant uses $OM(m - 1)$ and sends v to $n - 2$ other lieutenant.

OM Algorithm: Proof of Correctness

Lemma

For any m, k Algorithm $OM(m)$ satisfies validity if there are more than $2k + m$ generals and at most k traitors.

- Validity is defined when commander is loyal.
- Induction on m. For $m = 0$, its trivial.
- Assume hypothesis works for $m' < m$.
- In step 1, loyal commander sends value v to $n - 1$ lieutenant.
- In step 2, loyal lietuenant uses $OM(m-1)$ and sends v to $n-2$ other lieutenant.
- As $n - 1 > 2k + m - 1$ hence $OM(m-1)$ works in step 2.

OM Algorithm: Proof of Correctness

Lemma

For any m, k Algorithm $OM(m)$ satisfies validity if there are more than $2k + m$ generals and at most k traitors.

- Validity is defined when commander is loyal.
- Induction on m. For $m = 0$, its trivial.
- Assume hypothesis works for $m' < m$.
- In step 1, loyal commander sends value v to $n - 1$ lieutenant.
- In step 2, loyal lietuenant uses $OM(m - 1)$ and sends v to $n - 2$ other lieutenant.
- As $n - 1 > 2k + m - 1$ hence $OM(m - 1)$ works in step 2.
- Therefore, all loyal lieutenant get v from every other loyal lieutenant and the loyal commander.

OM Algorithm: Proof of Correctness

Lemma

For any m, k Algorithm $OM(m)$ satisfies validity if there are more than $2k + m$ generals and at most k traitors.

- Validity is defined when commander is loyal.
- Induction on m. For $m = 0$, its trivial.
- Assume hypothesis works for $m' < m$.
- In step 1, loyal commander sends value v to $n - 1$ lieutenant.
- In step 2, loyal lietuenant uses $OM(m - 1)$ and sends v to $n - 2$ other lieutenant.
- As $n - 1 > 2k + m - 1$ hence $OM(m - 1)$ works in step 2.
- Therefore, all loyal lieutenant get v from every other loyal lieutenant and the loyal commander.
- Hence, each loyal lieutenant receives atleast $n - k$ copies of value v. As $n - k > k + m > n/2$ and hence he/she chooses v.

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm $OM(m)$ satisfies validity and agreement if there are atleast $3m+1$ generals and atmost m traitors.

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm OM(m) satisfies validity and agreement if there are atleast $3m + 1$ generals and atmost m traitors.

- Induction on m. The case $m = 0$ (no traitor) is trivial.

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm $OM(m)$ satisfies validity and agreement if there are atleast $3m + 1$ generals and atmost m traitors.

- Induction on m. The case $m = 0$ (no traitor) is trivial.
- Assume the hypothesis works for all $m' < m$.

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm OM(m) satisfies validity and agreement if there are atleast $3m + 1$ generals and atmost m traitors.

- Induction on m. The case $m = 0$ (no traitor) is trivial.
- Assume the hypothesis works for all $m' < m$.
- When commander is loyal
  - Previous lemma shows that validity holds.
  - When validity holds then agreement holds as well.

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm $OM(m)$ satisfies validity and agreement if there are atleast $3m + 1$ generals and atmost m traitors.

- When commander is a traitor

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm $OM(m)$ satisfies validity and agreement if there are atleast $3m + 1$ generals and atmost m traitors.

- When commander is a traitor
  - In step 2, we have $\geq 3m$ generals and $\leq m - 1$ traitors.

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm $OM(m)$ satisfies validity and agreement if there are atleast $3m + 1$ generals and atmost m traitors.

- When commander is a traitor
  - In step 2, we have $\geq 3m$ generals and $\leq m - 1$ traitors.
  - $3m > 3(m - 1)$ hence $OM(m - 1)$ satisfies validity and agreement.

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm $OM(m)$ satisfies validity and agreement if there are atleast $3m + 1$ generals and atmost m traitors.

- When commander is a traitor
  - In step 2, we have $\geq 3m$ generals and $\leq m - 1$ traitors.
  - $3m > 3(m - 1)$ hence $OM(m - 1)$ satisfies validity and agreement.
  - For every j in step 2, each loyal lieutenant gets the same value v_j.

OM Algorithm: Proof of Correctness

Theorem

For any m, algorithm $OM(m)$ satisfies validity and agreement if there are atleast $3m + 1$ generals and atmost m traitors.

- When commander is a traitor
  - In step 2, we have $\geq 3m$ generals and $\leq m - 1$ traitors.
  - $3m > 3(m - 1)$ hence $OM(m - 1)$ satisfies validity and agreement.
  - For every j in step 2, each loyal lieutenant gets the same value v_j.
  - Each loyal lieutenant accepts the same value given by $majority\{v_1, v_2, \cdots v_{n-1}\}$.

Time Complexity of Oral Message Algorithm

- Let $T(n, m)$ be time complexity of $OM(m)$ for n nodes.

Time Complexity of Oral Message Algorithm

- Let $T(n, m)$ be time complexity of $OM(m)$ for n nodes.
- Step 1: commander sends messages to $n - 1$ lieutenant.

Time Complexity of Oral Message Algorithm

- Let $T(n, m)$ be time complexity of $OM(m)$ for n nodes.
- Step 1: commander sends messages to $n-1$ lieutenant.
- Step 2: each lieutenant runs $OM(m-1)$ algorithm with $n-1$ nodes.

Time Complexity of Oral Message Algorithm

- Let $T(n, m)$ be time complexity of $OM(m)$ for n nodes.
- Step 1: commander sends messages to $n-1$ lieutenant.
- Step 2: each lieutenant runs $OM(m-1)$ algorithm with $n-1$ nodes.
- Each lieutenant computes the majority of values.

Time Complexity of Oral Message Algorithm

- Let $T(n, m)$ be time complexity of $OM(m)$ for n nodes.
- Step 1: commander sends messages to $n-1$ lieutenant.
- Step 2: each lieutenant runs $OM(m-1)$ algorithm with $n-1$ nodes.
- Each lieutenant computes the majority of values.
- $T(n, m) = O(n) + nT(n-1, m-1) + O(n^2) = O(n^2) + nT(n-1, m-1)$
- $T(n, m) = O(n^m)$

Time Complexity of Oral Message Algorithm

- Let $T(n, m)$ be time complexity of $OM(m)$ for n nodes.
- Step 1: commander sends messages to $n - 1$ lieutenant.
- Step 2: each lieutenant runs $OM(m - 1)$ algorithm with $n - 1$ nodes.
- Each lieutenant computes the majority of values.
- $T(n, m) = O(n) + nT(n-1, m-1) + O(n^2) = O(n^2) + nT(n-1, m-1)$
- $T(n, m) = O(n^m)$

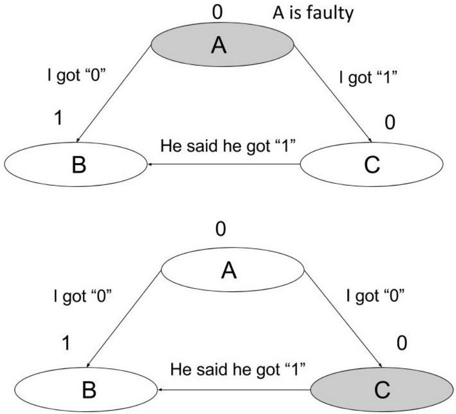
Exponential in number of traitors!

Can we do better?

- Why did we need $\geq 3f + 1$ generals?

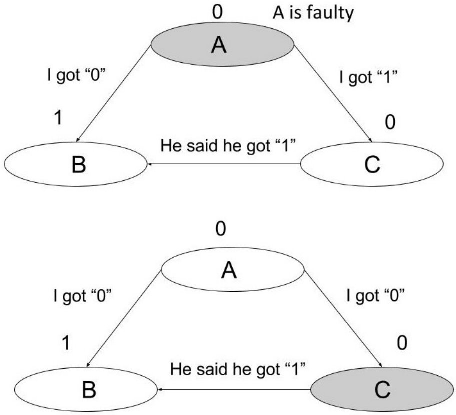
Can we do better?

- Why did we need $\geq 3f + 1$ generals?
- Systems could lie about each other.



Can we do better?

- Why did we need $\geq 3f+1$ generals?
- Systems could lie about each other.



Add digital signature to messages.

Digital Signature Assumptions

- i^{th} general signs a message m as $m : i$ before sending.
- A loyal general's message cannot be forged.
- Anyone can verify the authenticity of a general's signature.

Digital Signature Assumptions

- i^{th} general signs a message m as $m : i$ before sending.
- A loyal general's message cannot be forged.
- Anyone can verify the authenticity of a general's signature.

Theorem

Using above assumptions, one can handle f traitors with $\geq f + 2$ generals.

Digital Signature Algorithm $SM(m)$

- $V_i = \emptyset \ \forall_i \in \{1, 2, \cdots n\}$
- Commander signs and sends his/her value to every lieutenant.

Digital Signature Algorithm $SM(m)$

- $V_i = \emptyset \ \forall_i \in \{1, 2, \cdots n\}$
- Commander signs and sends his/her value to every lieutenant.
- For each i:
  - If a Lieutenant receives a message $v : 0$ from the commander and he/she has not received any order then.
    1. Let $V_i = \{v\}$.
    2. Send message $v : 0 : i$ to other lieutenant.

Digital Signature Algorithm $SM(m)$

- $V_i = \emptyset \ \forall_i \in \{1, 2, \cdots n\}$
- Commander signs and sends his/her value to every lieutenant.
- For each i:
  - If a Lieutenant receives a message $v : 0$ from the commander and he/she has not received any order then.
    1. Let $V_i = \{v\}$.
    2. Send message $v : 0 : i$ to other lieutenant.
  - if Lieutenant receives a message $v : 0 : j_1 : j_2 : \cdots : j_k$ and $v \notin V_i$.
    1. add v to V_i.
    2. if $k < m$ then send message $v : 0 : j_1 : j_2 : \cdots : j_k : i$ to every lieutenant other than $j_1, j_2 \cdots j_k$.

Digital Signature Algorithm $SM(m)$

- $V_i = \emptyset \; \forall_i \in \{1, 2, \cdots n\}$
- Commander signs and sends his/her value to every lieutenant.
- For each i:
  - If a Lieutenant receives a message $v : 0$ from the commander and he/she has not received any order then.
    1. Let $V_i = \{v\}$.
    2. Send message $v : 0 : i$ to other lieutenant.
  - if Lieutenant receives a message $v : 0 : j_1 : j_2 : \cdots : j_k$ and $v \notin V_i$.
    1. add v to V_i.
    2. if $k < m$ then send message $v : 0 : j_1 : j_2 : \cdots : j_k : i$ to every lieutenant other than $j_1, j_2 \cdots j_k$.
- For each i: lieutenant i accepts $majority(V_i)$ (0 if V_i is empty).

Digital Signature Algorithm: Formal Proof

Theorem

For any m, $SM(m)$ solves the Byzantine agreement if there are atmost m traitors.

Digital Signature Algorithm: Formal Proof

Theorem

For any m, $SM(m)$ solves the Byzantine agreement if there are atmost m traitors.

Let commander be loyal

- Each lieutenant receives $v : 0$.
- No lieutenant can forge $v' : 0$ hence every lieutenant receives only value v.
- Every lieutenant end up choosing v.

Digital Signature Algorithm: Formal Proof

Theorem

For any m, $SM(m)$ solves the Byzantine agreement if there are atmost m traitors.

If commander is a traitor

- show that $V_i = V_j$ for every loyal lieutenant i, j.

Digital Signature Algorithm: Formal Proof

Theorem

For any m, $SM(m)$ solves the Byzantine agreement if there are atmost m traitors.

If commander is a traitor

- show that $V_i = V_j$ for every loyal lieutenant i, j.
- let lieutenant i add a message $v : 0 : j_1 : j_2 : \cdots j_k$ to V_i.

Digital Signature Algorithm: Formal Proof

Theorem

For any m, $SM(m)$ solves the Byzantine agreement if there are atmost m traitors.

If commander is a traitor

- show that $V_i = V_j$ for every loyal lieutenant i, j.
- let lieutenant i add a message $v : 0 : j_1 : j_2 : \cdots j_k$ to V_i.
- if $j \in \{j_1, j_2 \cdots j_k\}$ then lieutenant j received the message.

Digital Signature Algorithm: Formal Proof

Theorem

For any m, $SM(m)$ solves the Byzantine agreement if there are atmost m traitors.

If commander is a traitor

- show that $V_i = V_j$ for every loyal lieutenant i, j.
- let lieutenant i add a message $v : 0 : j_1 : j_2 : \cdots j_k$ to V_i.
- if $j \in \{j_1, j_2 \cdots j_k\}$ then lieutenant j received the message.
- else:
  - if $k < m$ then i sends this message to j in next step.
  - if $k = m$ then there is atleast one loyal lietenant in $\{j_1, j_2 \cdots j_m\}$.
  - this loyal lieutenant must have send this message to lieutenant j.

More on Byzantine Agreement

- We assumed fully connected graph in OM, SM algorithm.

More on Byzantine Agreement

- We assumed fully connected graph in OM, SM algorithm.

Theorem

Cannot achieve Byzantine agreement in a graph with $\leq 2f$ node connectivity and f traitors.

Proof technically similar to the one presented.

More on Byzantine Agreement

- Can we solve a simpler problem?
- Can we weaken the validity condition

More on Byzantine Agreement

- Can we solve a simpler problem?
- Can we weaken the validity condition

Weak Validity: Only when all nodes are correct and have the same input, that input is the value chosen.

Theorem

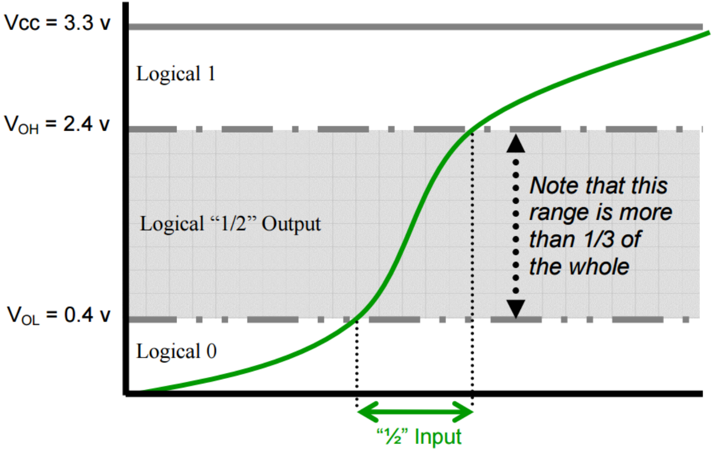
Cannot achieve weak Byzantine agreement in a graph with $\leq 3f$ *nodes with* f *traitors.*

Byzantine Agreement: Take Away

- Used in places where security takes precedence over performance.
- Example credentials system, space shuttle.
- Modern protocols are less expensive than OM, SM algorithms.
- Whenever possible use less expensive models such as *fail-by-halt*.

Byzantine Failure: An example

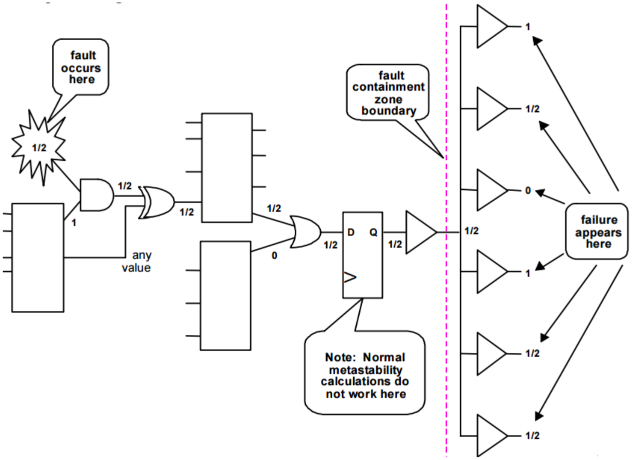
Bit value 1/2



(taken from Driscoll et al. 2003)

Byzantine Failure: An example

Byzantine Failure Propagation



(taken from Driscoll et al. 2003)

Byzantine Failure: Be Realistic

Murphys Law:
"If anything can go wrong, it will go wrong."

Conclusion

- Byzantine fault and Byzantine agreement
- $3f + 1$ theorem
- Oral Message algorithm
- Digital Signature algorithm
- Protocols are expensive
- Byzantine failures can occur in strange places

References

Michael J. Fischer, Nancy A. Lynch and Michael Merritt (1986)
Easy impossibility proofs for distributed consensus problems
Distributed Computing 1.1, 26-39.

Leslie Lamport, Robert Shostak, and Marshall Pease (1982)
The Byzantine Generals Problem,
(TOPLAS) 4.3 : 382-401.

Kevin Driscoll, Brendan Hall, Hakan Sivencrona, Phil Zumsteg (2003)
Byzantine fault tolerance, from theory to reality
Reliability, and Security 12(3), 235-248.

Figure on slide 5-6:
Power Grid: http://www.jmccp.com/strategy/
Ballistic Missile: http://manglermuldoon.blogspot.com/

Backup Slide: Rephrasing the problem

From formulation 1 to 2

1. We go in n rounds.
2. In i^{th} round, node i acts as commander and sends his/her input to the j^{th} node.
3. We then run the protocol for formulation 2.
4. At the end of all rounds, each node accepts the majority decisions of the n rounds.

Backup Slide: Rephrasing the problem

From formulation 1 to 2

1. We go in n rounds.
2. In i^{th} round, node i acts as commander and sends his/her input to the j^{th} node.
3. We then run the protocol for formulation 2.
4. At the end of all rounds, each node accepts the majority decisions of the n rounds.

Why this works?

Agreement: In all rounds, all loyal nodes accept the same value. Hence, at the end of the round; they all accept the same value.
Validity: If all correct nodes have the same input, then that input will be accepted by all loyal nodes in atleast $2f + 1$ rounds and hence will be the majority at the end.