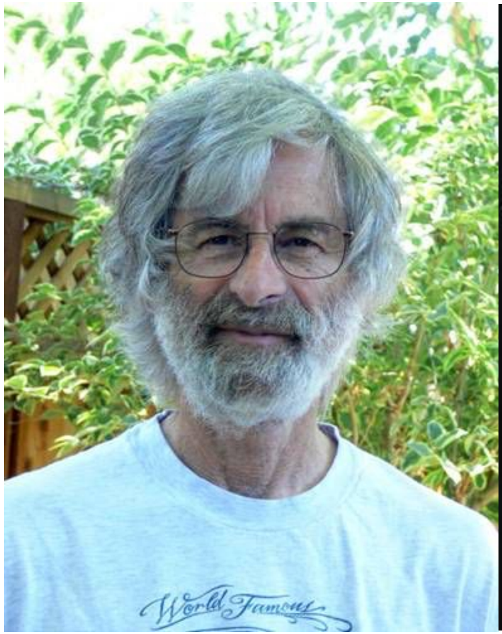
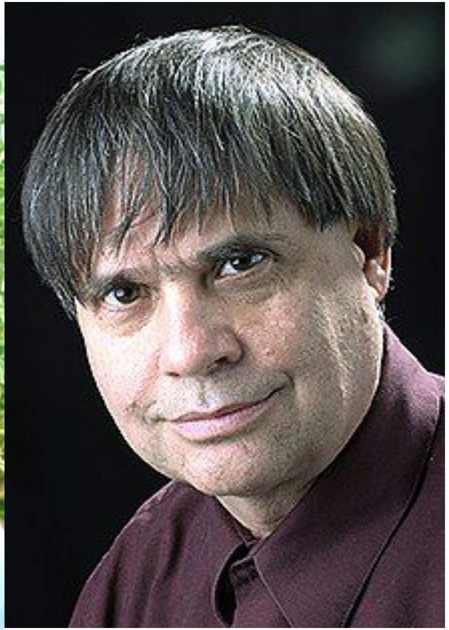
The Byzantine Generals Problem

- Kushal Babel

Authors

- Leslie Lamport
    - Turing Award
    - Paxos, Lamport Clocks, LaTex...
- Robert Shostak
    - PhD at Harvard
    - Entrepreneur
- Marshall Pease





Reinvented & Re-branded many times...

1978
Lamport claims to have first discovered Byzantine faults

The Implementation of Reliable Distributed Multiprocess System

1978
Shostak et al. working on SIFT at SRI formulate the problem and give non-cryptographic solution

1980
Lamport adds cryptographic solution

Reaching Agreement in the presence of Faults

1982
Re-branded to The Byzantine Generals Problem

"I have long felt that, because it was posed as a cute problem about philosophers seated around a table, Dijkstra's dining philosopher's problem received much more attention than it deserves……The popularity of the dining philosophers problem taught me that the best way to attract attention to a problem is to present it in terms of a story"

- Lamport https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/ (Recipient of Dijkstra Prize)

Two Generals Problem	+	Reaching Agreement in the Presence of Faults	=	The Byzantine Generals Problem

Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

Talk Overview

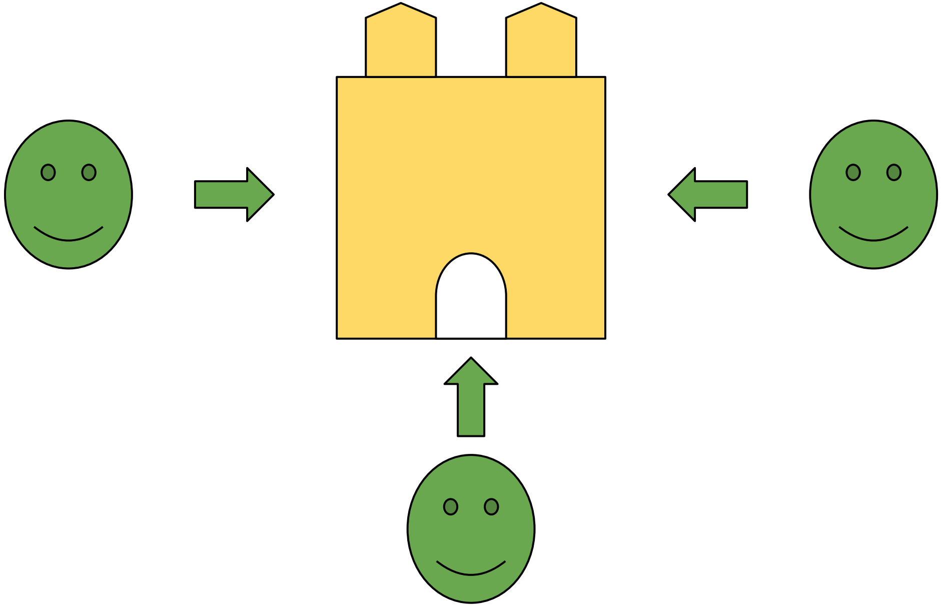
- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

Consensus

- Every process must agree on the same value
- The value should be proposed by some processor i.e. consensus algorithm can't invent a new value

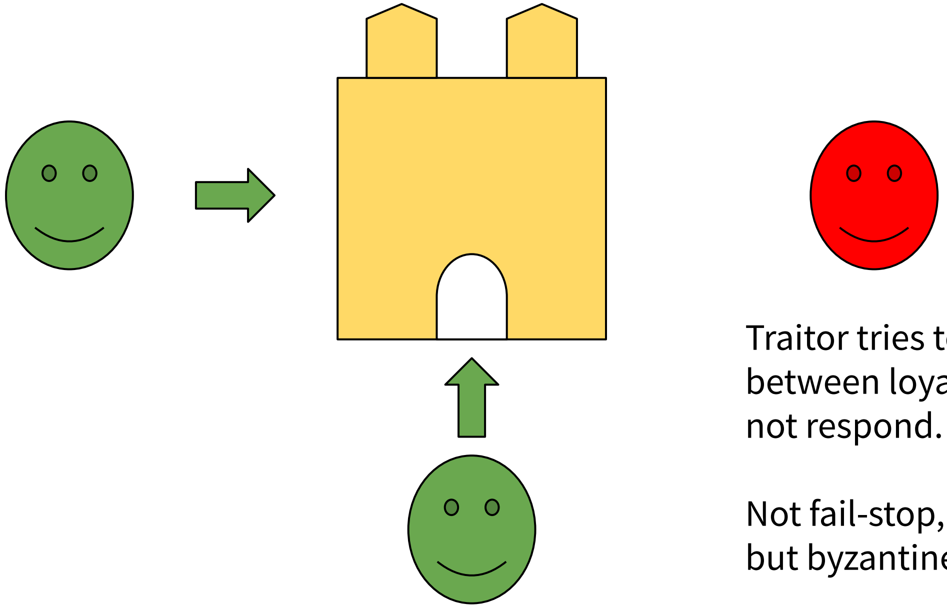
BGP Formulation

Success if all attack or all retreat (common plan of action)



Byzantine behaviour

No general a priori knows if a counterpart is loyal or traitor



Traitor tries to prevent agreement between loyal generals. Can lie or not respond.

Not fail-stop, neither fail-crash but byzantine.

Assumptions

- Absence of message can be detected (Synchronous Communication)
- Every message that is sent is delivered correctly
- Receiver of a message knows who sent it

Objectives

1. All loyal generals decide upon the same plan of action
2. A small number of traitors cannot cause the loyal generals to adopt a bad plan

1. Every loyal general must obtain the same information v(1), ….., v(n)
2. If the i^{th} general is loyal, then the value that he sends must be used by every loyal general as the value of v(i)

1. Any two loyal generals use the same value of v(i)
2. If the i^{th} general is loyal, then the value that he sends must be used by every loyal general as the value of v(i)

Byzantine Generals Problem

A commanding general must send an order to his n-1 lieutenant generals such that:

IC1 : All loyal lieutenant generals obey the same order

IC2 : If the commanding general is loyal, then every loyal lieutenant obeys the order he sends

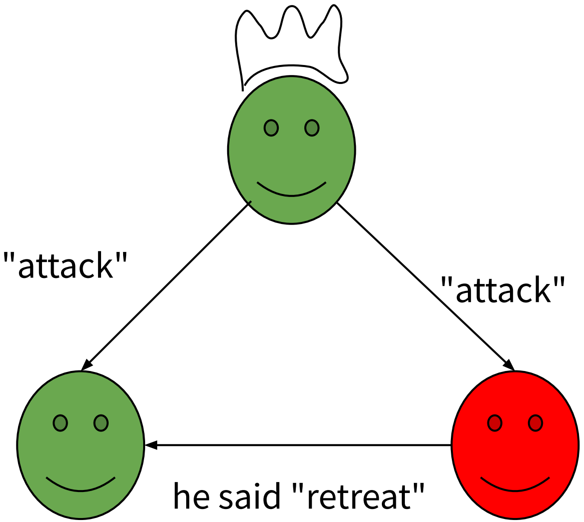
Remark : IC2 implies IC1 if the commanding general is loyal

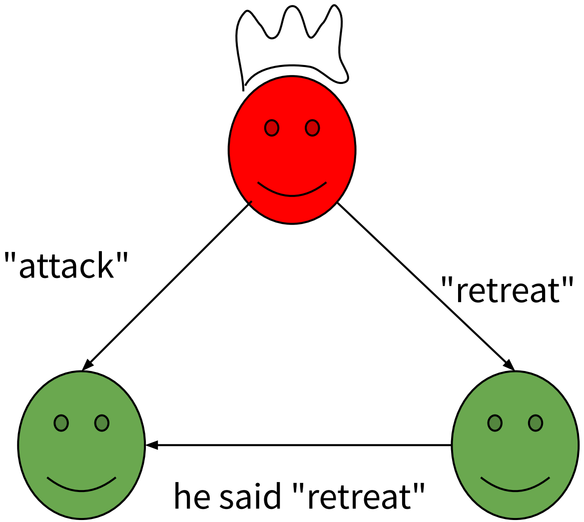
Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

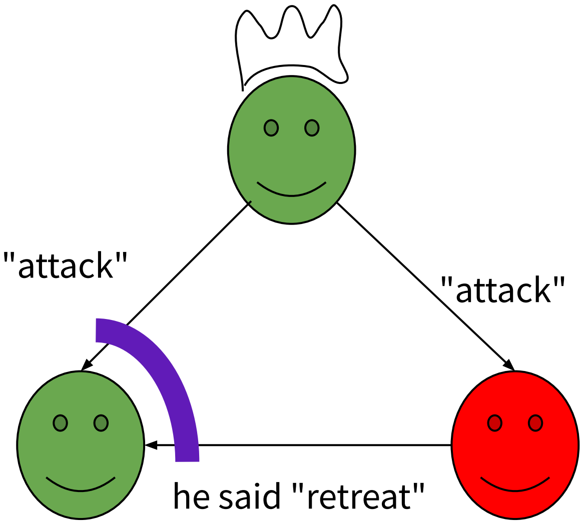
No solution with fewer than 3m+1 generals can cope with m traitors

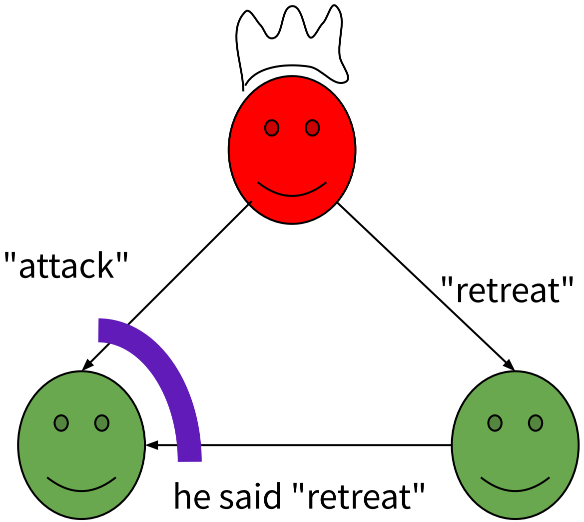
One Traitor


"attack"
"attack"
he said "retreat"


"attack"
"retreat"
he said "retreat"

One Traitor


"attack"
"attack"
he said "retreat"


"attack"
"retreat"
he said "retreat"

Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

Formal Formulation

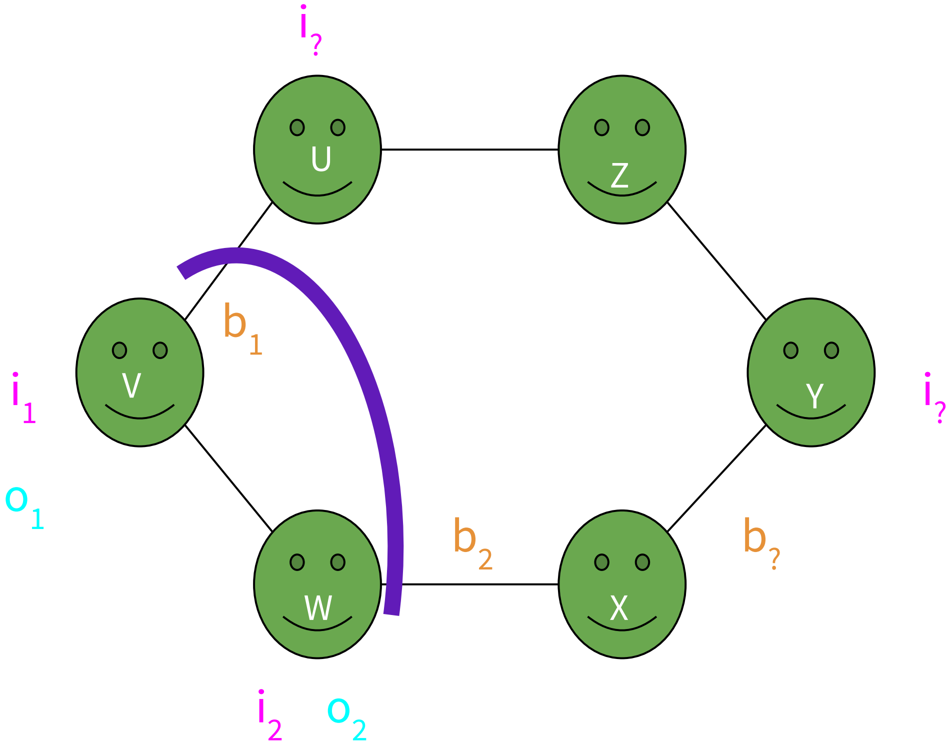
Setting: Communication Graph G with bidirectional edges and each node running a certain type of agreement device. Device is undefined primitive.

Instantiation: Supply a boolean input (1 or 0) to each device. This results in certain boolean output (1 or 0) on each device and certain behaviour of each edge.

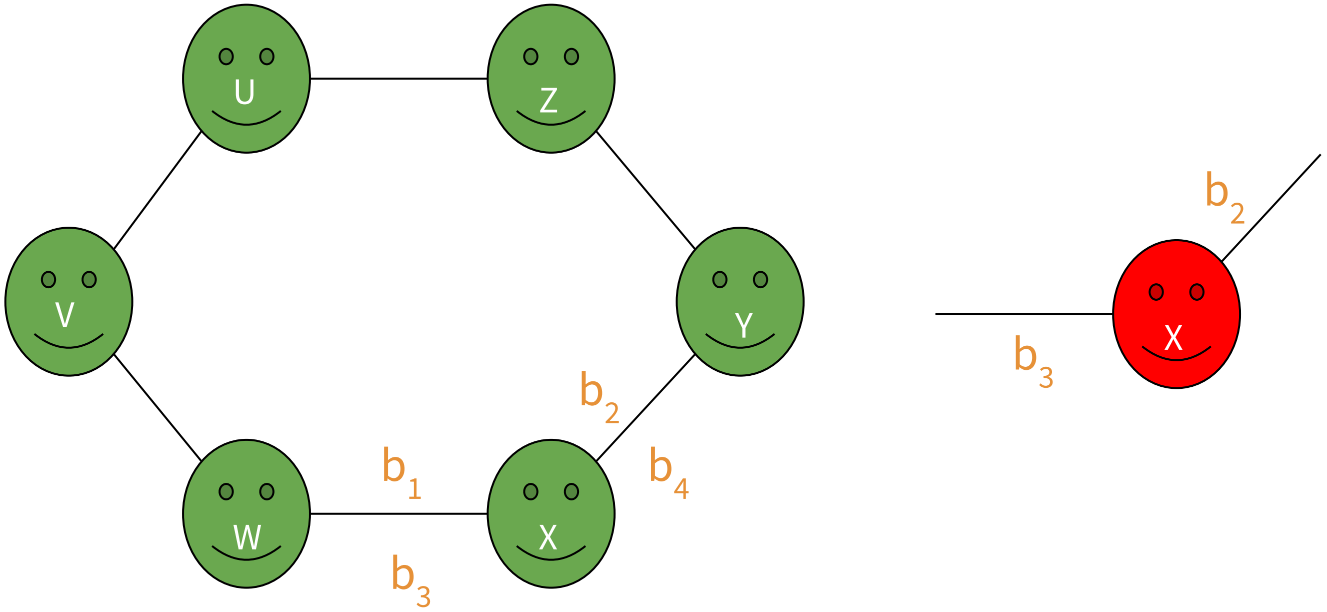
Locality Axiom: The output of every device in any subgraph is determined only by the type of device, the input to the device, and the behaviour of incoming edges from the remainder of the graph.

Fault Axiom: Any behavior exhibited by a device over different edges in different instantiations can be exhibited by a faulty device in a single instantiation.

Locality



Locality



Byzantine Agreement (n,m)

For a graph G with n devices, out of which m are faulty, byzantine agreement is reached if the following two conditions are satisfied:

Agreement: Every correct device chooses the same output

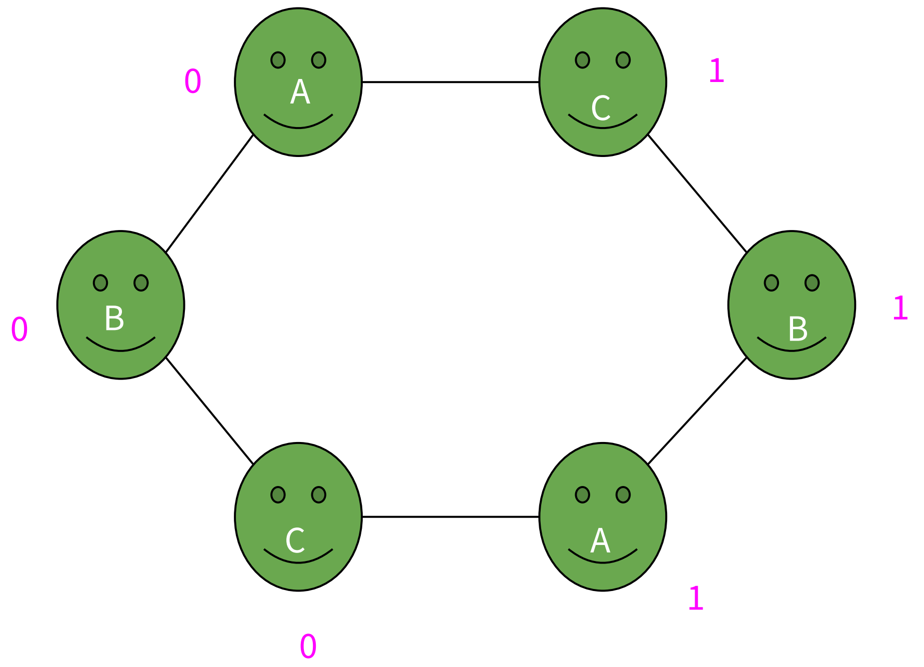
Validity: If all the correct nodes have the same input, that input must be the output chosen.

Sound familiar?

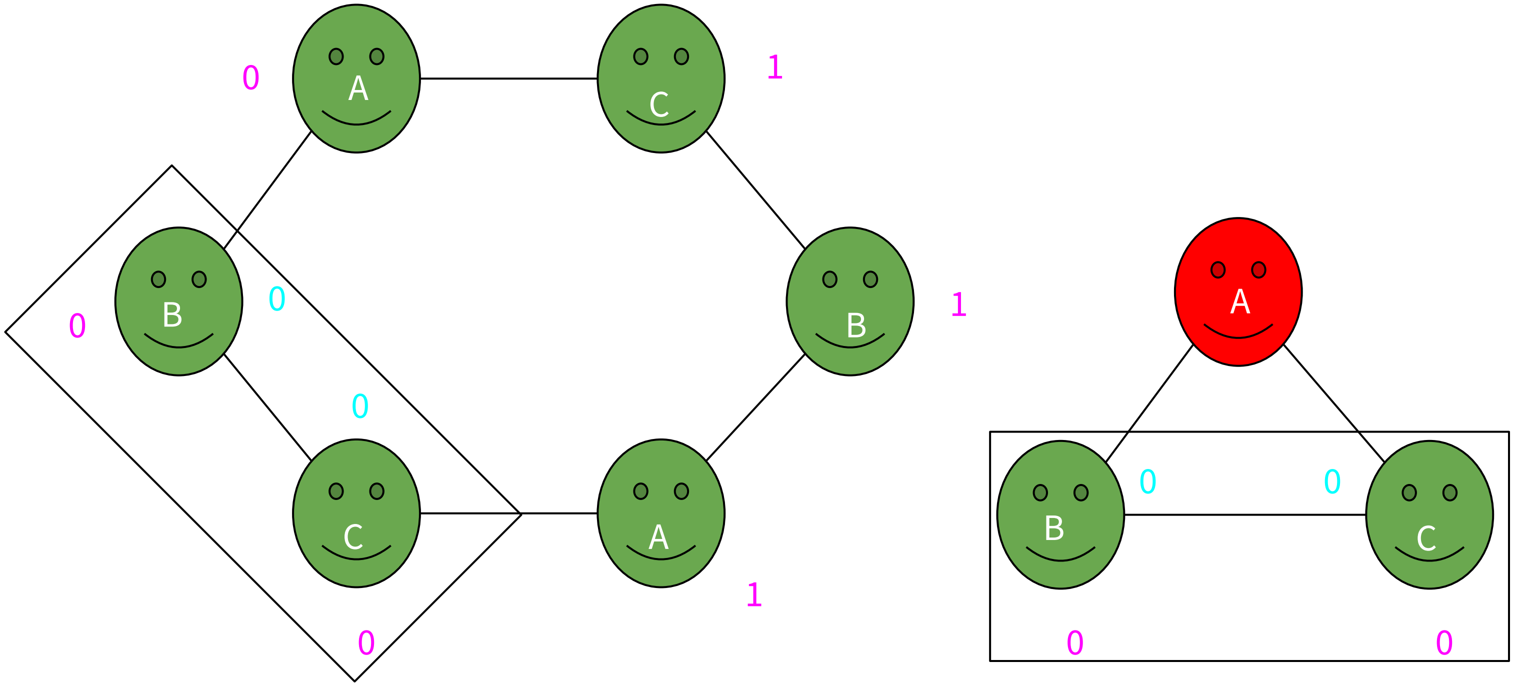
Byzantine Agreement (n,m)

Byzantine Agreement can't be reached if n <= 3m

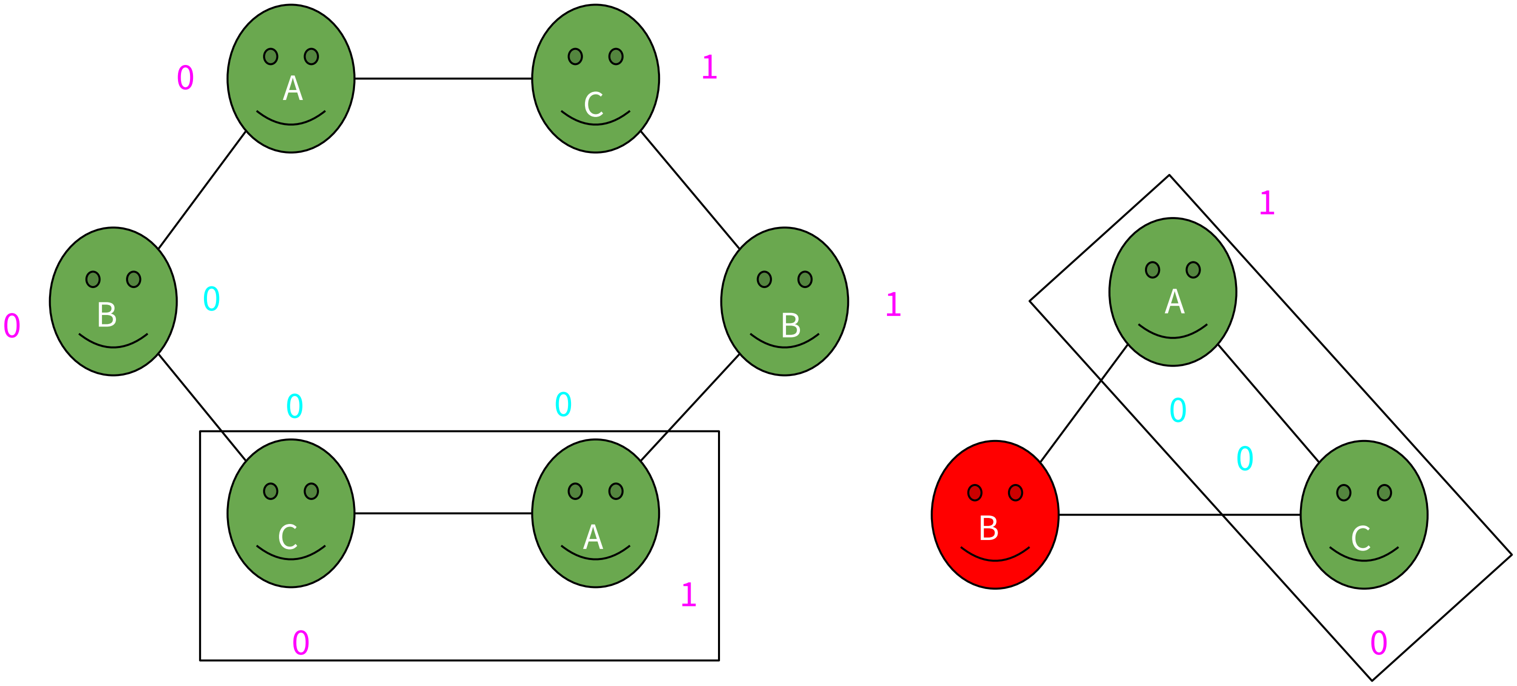
m=1, Proof by Contradiction



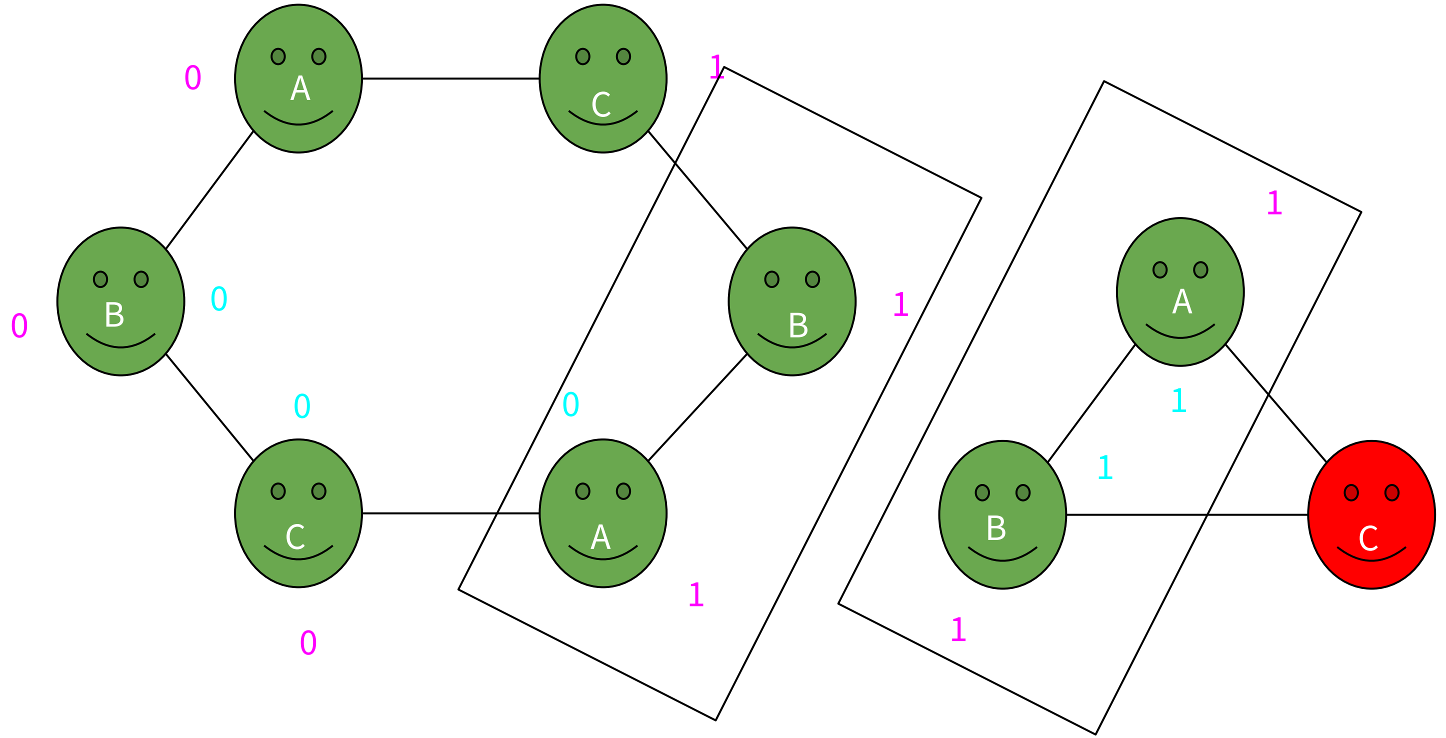
m=1, Proof by Contradiction



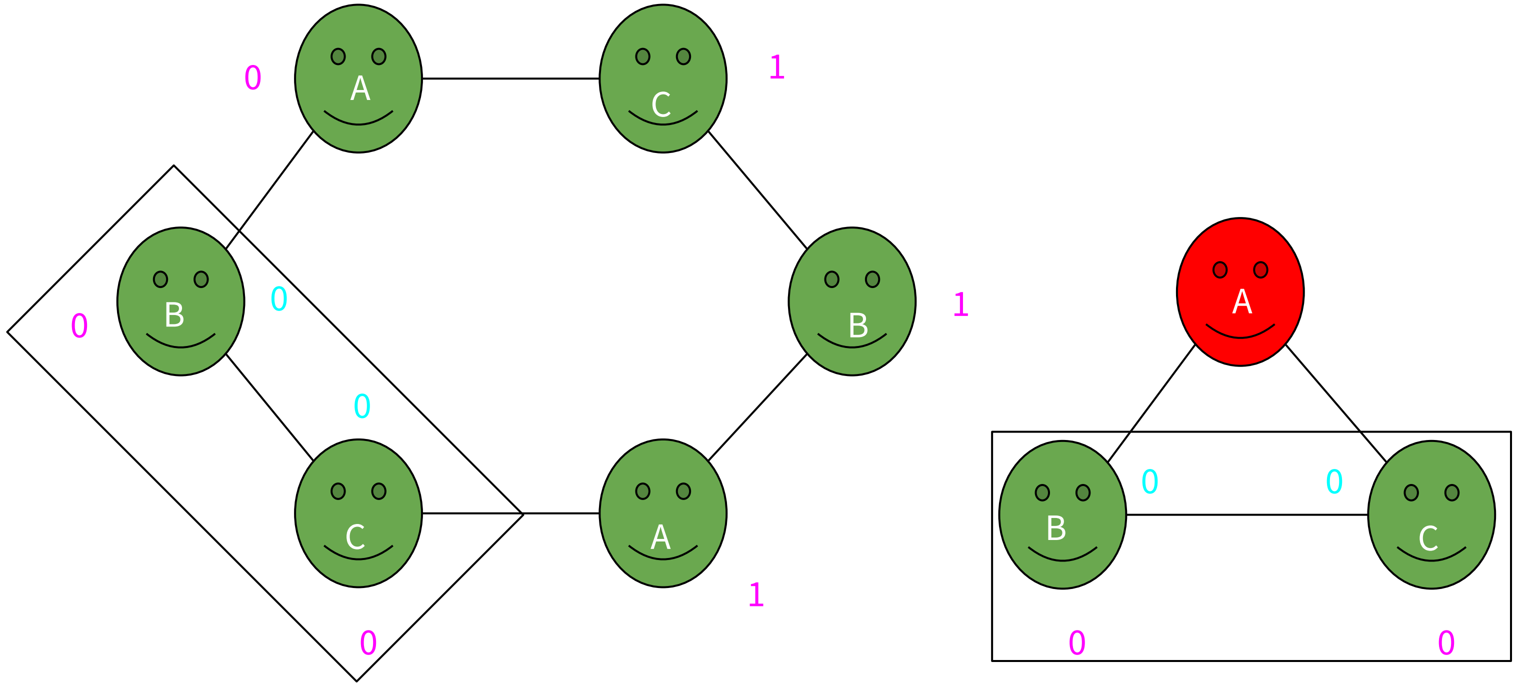
m=1, Proof by Contradiction



m=1, Proof by Contradiction



m, Proof by Contradiction



$n > 3m$ is necessary condition for consensus

Is it sufficient as well?

Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

Assumptions

- Absence of message can be detected
- Every message that is sent is delivered correctly
- Receiver of a message knows who sent it

Recursive Algorithm

OM(0)

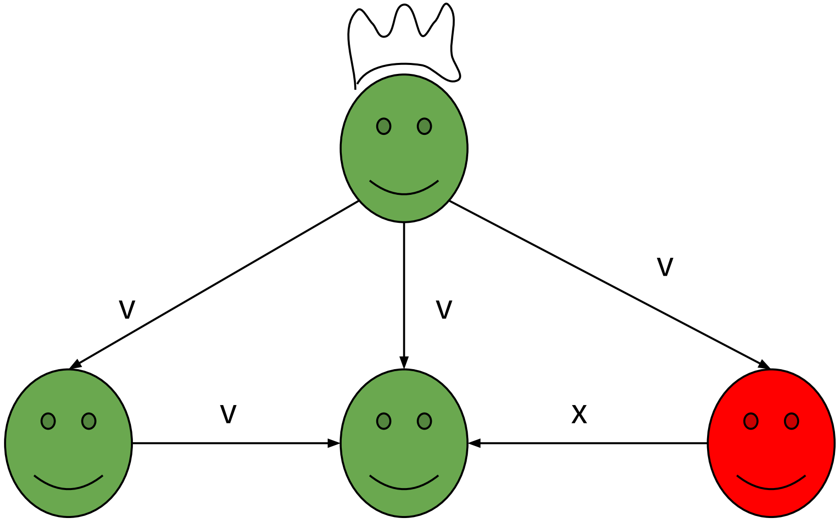
1. Commander sends his value to every lieutenant
2. Each lieutenant uses the received value or "retreat" if no value received

OM(m)

1. Commander sends his value to every lieutenant
2. Everyone runs OM(m-1) and acts as the commander to send the value received in step 1 to all the other lieutenant
3. Each lieutenant uses the majority value out of the values received.

Intuition : On receiving every message, tell others that you have received that message

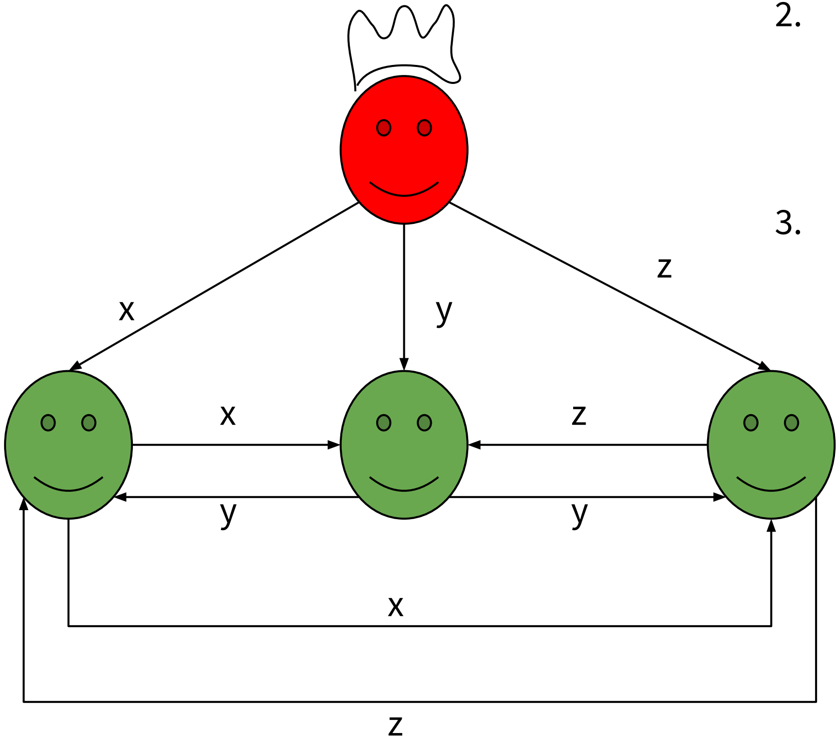
OM(1) for n=4



OM(m)

1. Commander sends his value to every lieutenant
2. Everyone runs OM(m-1) and acts as the commander to send the value received in step 1 to all the other lieutenant
3. Each lieutenant uses the majority value out of the values received.

OM(1) for n=4



OM(m)

1. Commander sends his value to every lieutenant
2. Everyone runs OM(m-1) and acts as the commander to send the value received in step 1 to all the other lieutenant
3. Each lieutenant uses the majority value out of the values received.

Number of messages

$$= (n-1)*(n-2)*....(n-m-1)$$

$$= O(n^m)$$

Exponential in number of traitors!

Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

Authors

- Miguel Castro
  - MSR
- Barbara Liskov
  - MIT
  - Turing Award - OOP
  - Andrew's Advisor

PBFT

- 3-phase commit : Pre-Prepare, Prepare, Commit
- Only 3% slower than non-replicated implementation of NFS
- Requires 3n+1 nodes to cope with n byzantine failures
- Semi-synchronous

Safety

- One replica acts as commander for a particular *view*
- Commander sends a value v to each lieutenant
- Each lieutenant waits for atleast 2m + 1 messages of v from different lieutenants before commiting value v
- Clients need to get m + 1 replies
- Normal-operation and view change works with asynchronous communication

Liveness

- Use local timer to check for timeouts
- Every replica gets to become leader in round robin fashion, called view change
- Synchronous view change if timeout occurs

Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

n >= m+1 is sufficient condition to cope with m traitors

Assumptions

- Synchronous Communication
- Absence of message can be detected
- Every message that is sent is delivered correctly
- Receiver of a message knows who sent it
- A loyal general's signature cannot be forged, and any alteration of the contents of his signed message can be detected
- Anyone can verify the authenticity of a general's signature

Algorithm

SM(m)

Initialise $V_i = \phi$ for each i

1. Commander signs and sends his value to every lieutenant
2. Every lieutenant i : Insert the value in V_i if not present. If chain of signatures has length < m, sign the message and forward to every lieutenant who hasn't signed this value already
3. Choose the majority value from V_i

Number of messages : ${}^{n}P_{m+1}$

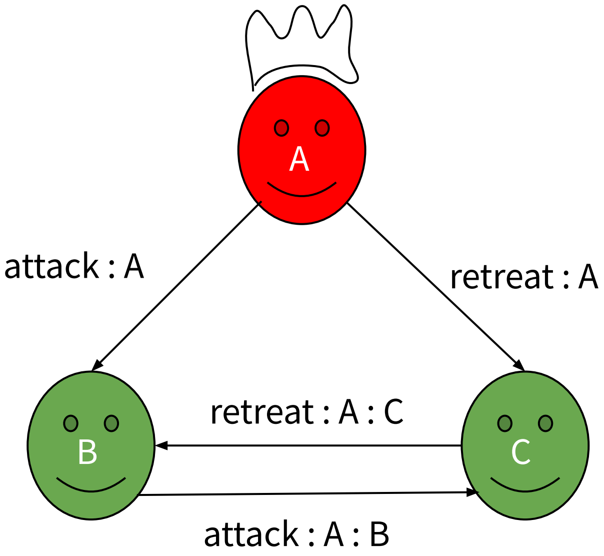
Notation

v : i denotes a value v signed by general i

v : i : j : k denotes i signed v, and the result is signed by j, whose result is signed by k

: is left-associative

One Traitor



SM(m)

Initialise $V_i = \phi$ for each i

1. Commander signs and sends his value to every lieutenant
2. Every lieutenant i : Insert the value in V_i if not present. If chain of signatures has length <= m, sign the message and forward to every lieutenant who hasnt signed this value already
3. Choose the majority value from V_i

General Case Proof

1. IC1 : All loyal lieutenant generals obey the same order

2. IC2 : If the commanding general is loyal, then every loyal lieutenant obeys the order he sends

   Only non-trivial case is to prove IC1 when commander is traitor

   Need to prove that if honest lieutenant i received v, then honest lieutenant j also received v

   lieutenant i forwards v to lieutenant j, except when m+1 signatures are already present in which case some honest lieutenant in that chain must have forwarded it to lieutenant j

Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

Examine Assumptions

- Synchronous Communication -> **Timeout**
- Absence of message can be detected -> **Timeout**
- Every message that is sent is delivered correctly
- Receiver of a message knows who sent it
- A loyal general's signature cannot be forged, and any alteration of the contents of his signed message can be detected
- Anyone can verify the authenticity of a general's signature

Examine Assumptions

- Synchronous Communication -> **Timeout**
- Absence of message can be detected -> **Timeout**
- Every message that is sent is delivered correctly -> **Same as byzantine failure**
- Receiver of a message knows who sent it
- A loyal general's signature cannot be forged, and any alteration of the contents of his signed message can be detected
- Anyone can verify the authenticity of a general's signature

Examine Assumptions

- Synchronous Communication -> **Timeout**
- Absence of message can be detected -> **Timeout**
- Every message that is sent is delivered correctly -> **Same as byzantine failure**
- Receiver of a message knows who sent it -> **Do not use a switching network**
- A loyal general's signature cannot be forged, and any alteration of the contents of his signed message can be detected
- Anyone can verify the authenticity of a general's signature

Examine Assumptions

- Synchronous Communication -> **Timeout**
- Absence of message can be detected -> **Timeout**
- Every message that is sent is delivered correctly -> **Same as byzantine failure**
- Receiver of a message knows who sent it -> **Do not use a switching network**
- A loyal general's signature cannot be forged, and any alteration of the contents of his signed message can be detected -> **Public Key Cryptography or HMAC**
- Anyone can verify the authenticity of a general's signature -> **Public Key Cryptography or HMAC**

Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

Bitcoin

Probabilistic Safety....

Talk Overview

- Byzantine Generals Problem Formulation
- Impossibility Result
- Easy Impossibility Result
- Oral Message Solution
- Practical Byzantine Fault Tolerance
- Signed Message Solution
- Reliable Systems
- Bitcoin
- Conclusion

Necessary & Sufficient number of nodes to cope up with m byzantine failures

	Synchronous	Semi-synchronous	Asynchronous
Unsigned Messages	n >= 3m + 1		Impossible
Signed Messages	n >= m + 1	n >= 3m + 1	Impossible

Questions?