The Byzantine Generals Problem

Zixin Chi
Julian Angeles

Motivation

A reliable system must be fault-tolerant.

↓

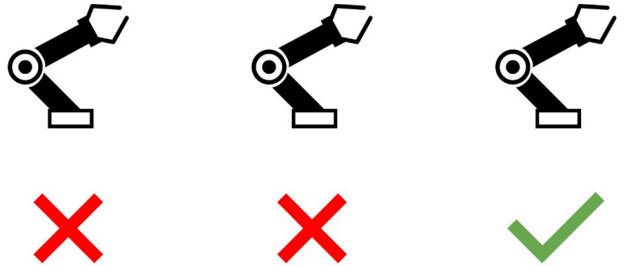
having some degree of redundancy.

↓

Consensus protocol.

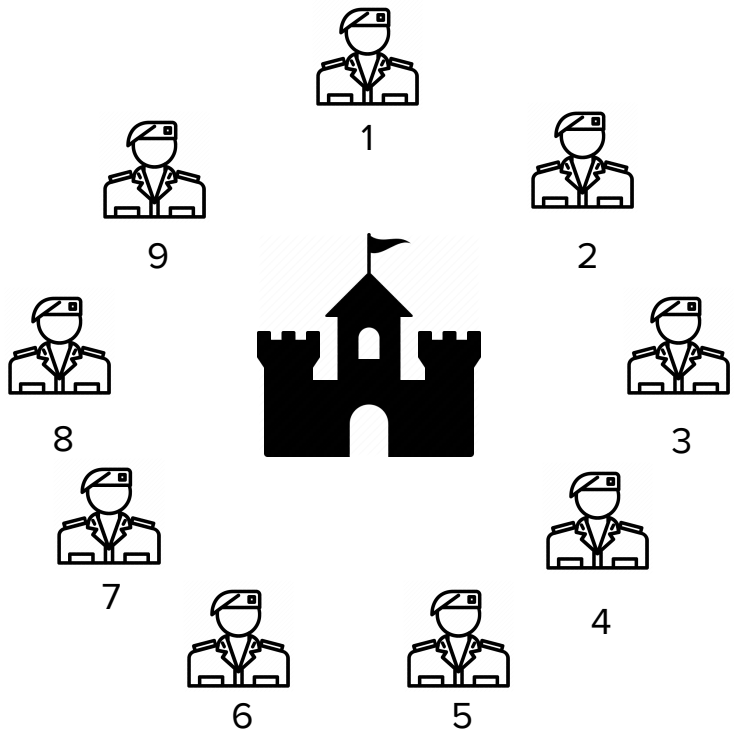
↓

How can we reach consensus?



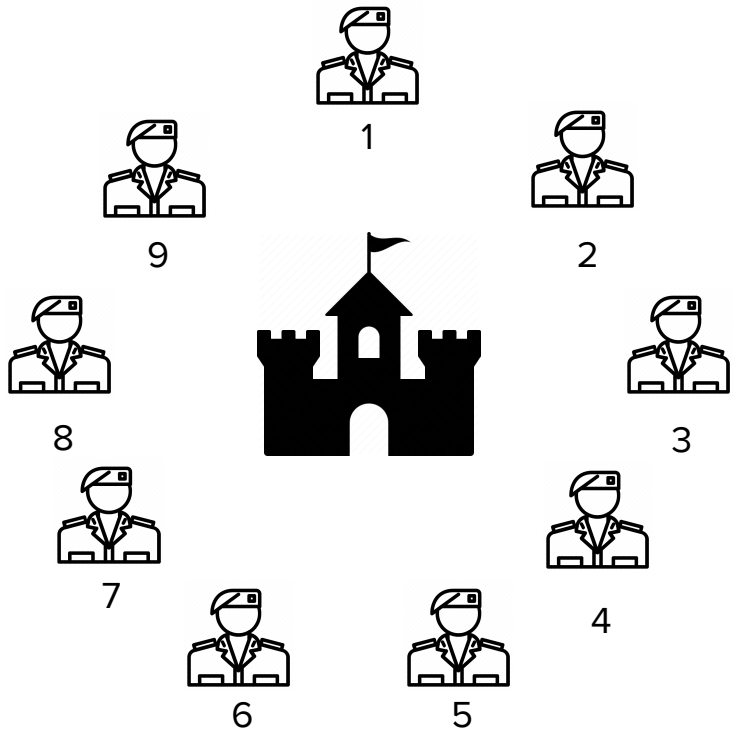
What is the problem?

- First proposed by Lamport, et al in 1982
- Loyal generals vs Traitors.
- Attack? Retreat?
- Reach consensus among “loyal generals”given f “traitors”



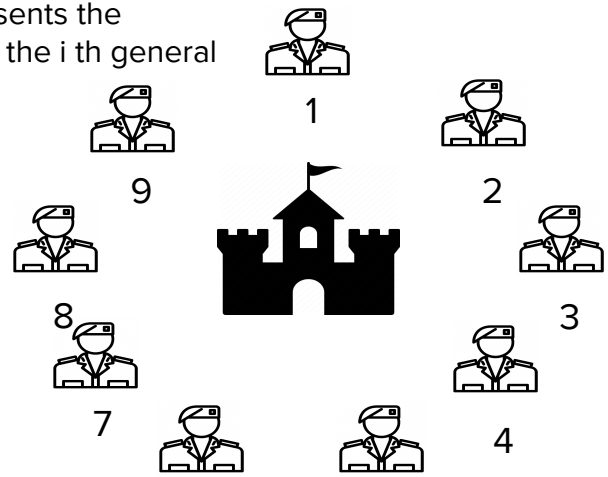
Requirements of the Algorithm

- All loyal generals decide upon the same plan.
- Small number of traitors cannot cause loyal generals to adopt bad plan.



Reach an Agreement

v(i) represents the message the i th general sends



Restate the conditions

1) Every loyal general must obtain same v(1)..v(n)

1') Any two loyal generals use same value of v(i)

2) If ith general is loyal, then the value he sends must be used by every other general as v(i)

Reduce to the final conditions

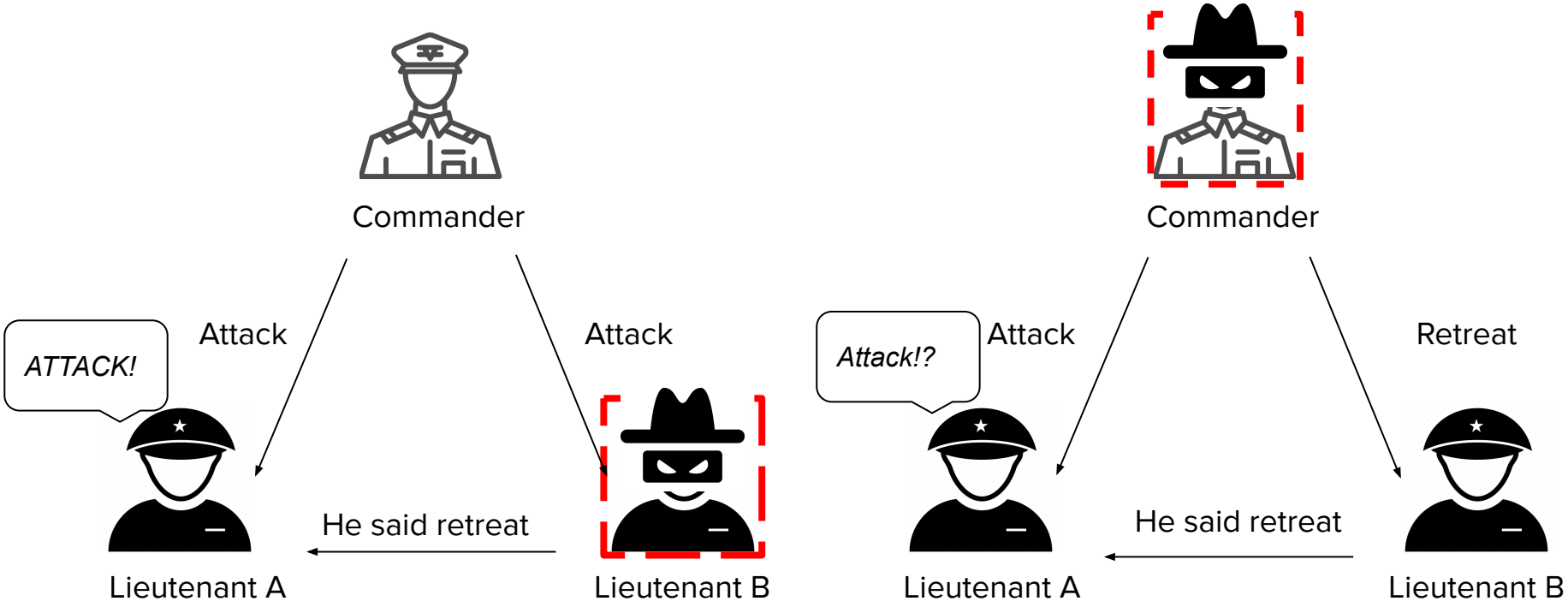
IC 1. All loyal lieutenants obey the same order

IC 2. If the commanding general is loyal, then every loyal lieutenant obeys the order the general sends

Interactive Consistency conditions

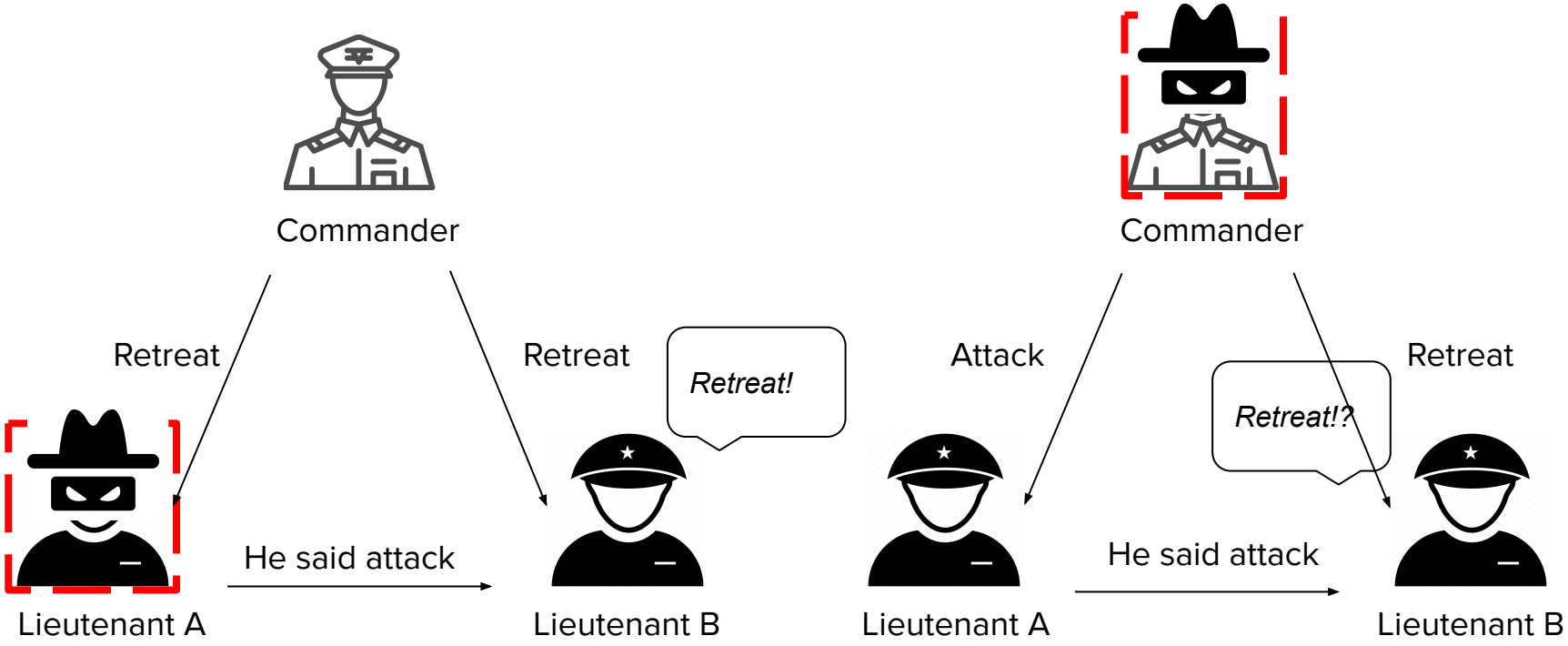
Impossibility results

For oral message communication, traitors must be less than ⅓.



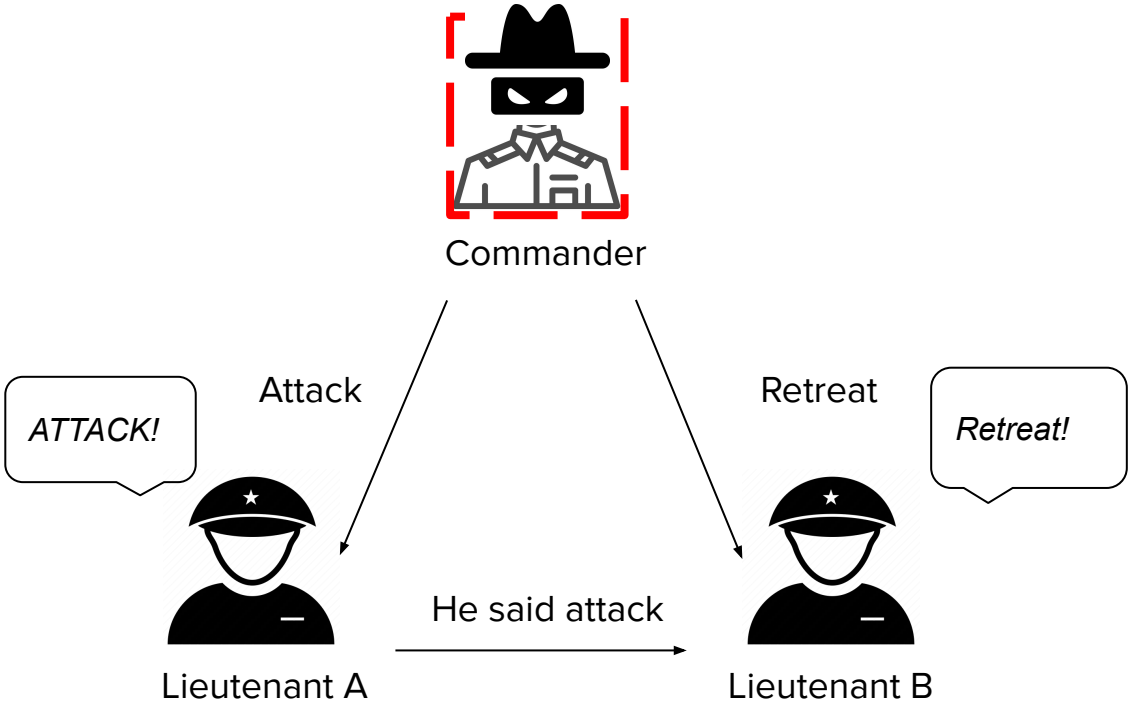
Impossibility results

For oral message communication, traitors must be less than ⅓.



Impossibility results

For oral message communication, traitors must be less than ⅓.



A solution with oral messages

Assumptions of oral messages

A1. Every message that is sent is delivered correctly.

What if it’s not?

A2. The receiver of a message knows who sends it.

Gets nullified later in Sighed Messages

A3. The absence of a message can be detected.

A solution with oral messages

OM(m) : Oral Message algorithms when coping with m traitors (m >= 0)

No traitor OM(0) (1) The commander sends his value.

(2) Each lieutenant follows the value he received

OM(m), m > O.

(1) The commander sends his value to every lieutenant.

(2) every lieutenant act as command to send his value by conducting a OM(m-1)

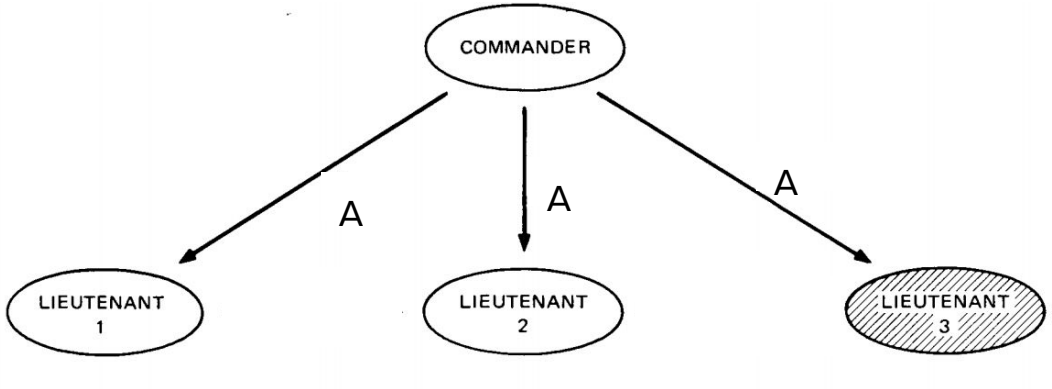
(3) Majority Voting

Default value is Retreat

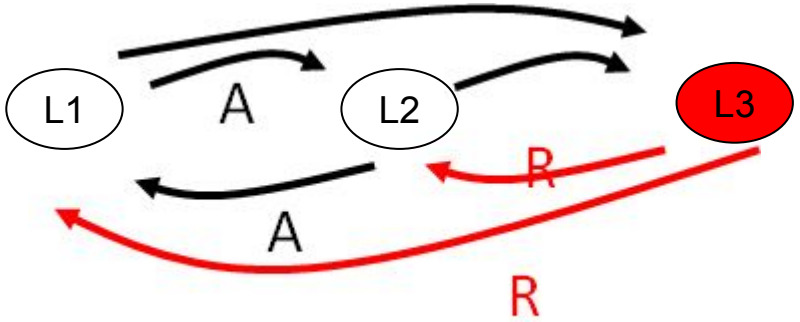
Lieutenant is a Traitor

One traitor, four total(m = 1, N = 4)

OM(1)



OM(0)

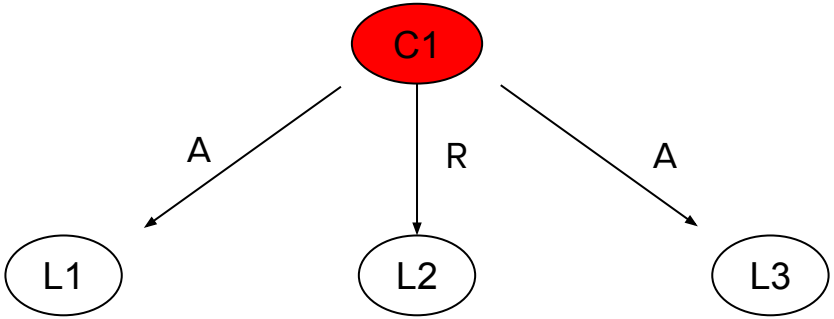


L1 = m (A, A, R); L2 = m (A, A, R); Both attack!

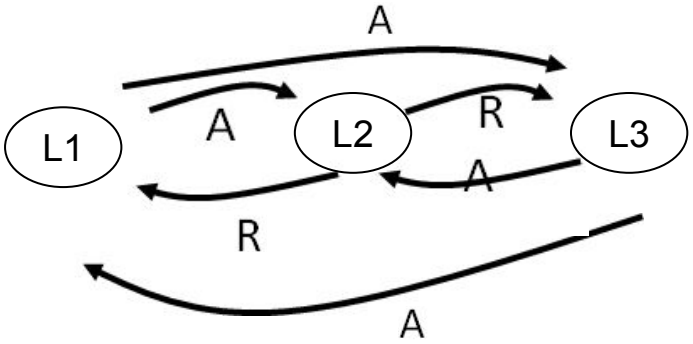
Commander is a Traitor

One traitor, four total(m = 1, N = 4)

OM(1)



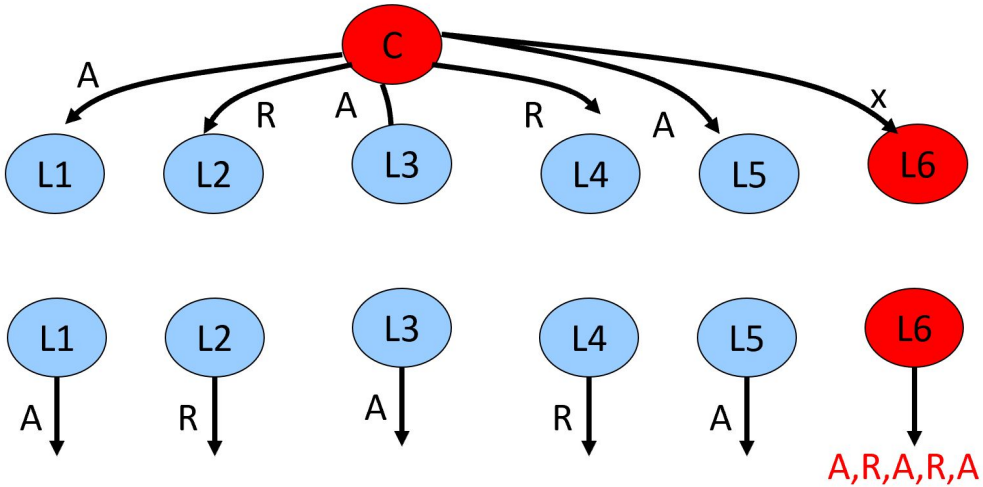
OM(0)



L1=m(A, R, A); L2=m(A, R, A); L3=m(A,R,A); Attack!

Both are Traitors (bigger army)

Two traitors, seven total(m= 2, N = 7)



All loyal lieutenants cannot reach agreement

L1: m(A,R,A,R,A,A) ==> Attack

L2: m(A,R,A,R,A,R) ==> Retreat

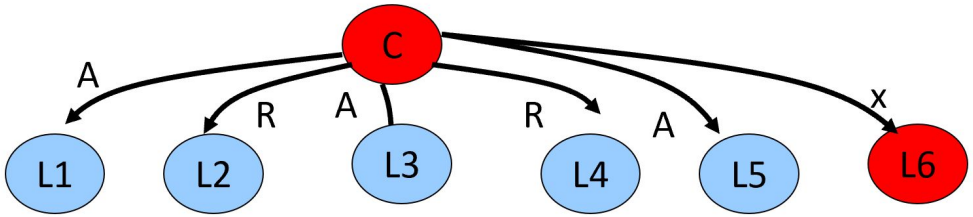
L3: m(A,R,A,R,A,A) ==> Attack

L4: m(A,R,A,R,A,R) ==> Retreat

L5: m(A,R,A,R,A,A) ==> Attack

Both are Traitors (bigger army)

Two traitor, seventotal(M = 2, N = 7)

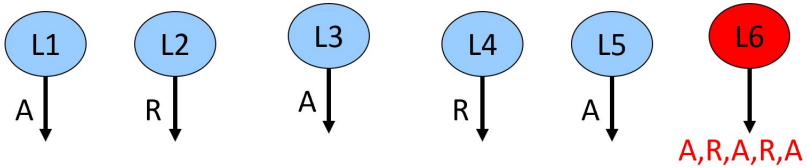


Verify that lieutenants tell each other the same thing

- Requires rounds = m+1

What messages does L1 receive in this example?

- OM(2): A
- OM(1): 2R, 3A, 4R, 5A, 6A (doesn't know 6 is traitor)
- OM(0):
  - 2{ 3A, 4R, 5A, 6R}
  - 3{2R, 4R, 5A, 6A}
  - 4{2R, 3A, 5A, 6R}
  - 5{2R, 3A, 4R, 6A}
  - 6{ total confusion }



L6 is lying!

m(A,R,A,R,A,-) ==> Attack!

Good Enough?

Why so difficult?

Traitor's ability to **Lie**

No More Lying!

Include OM assumptions

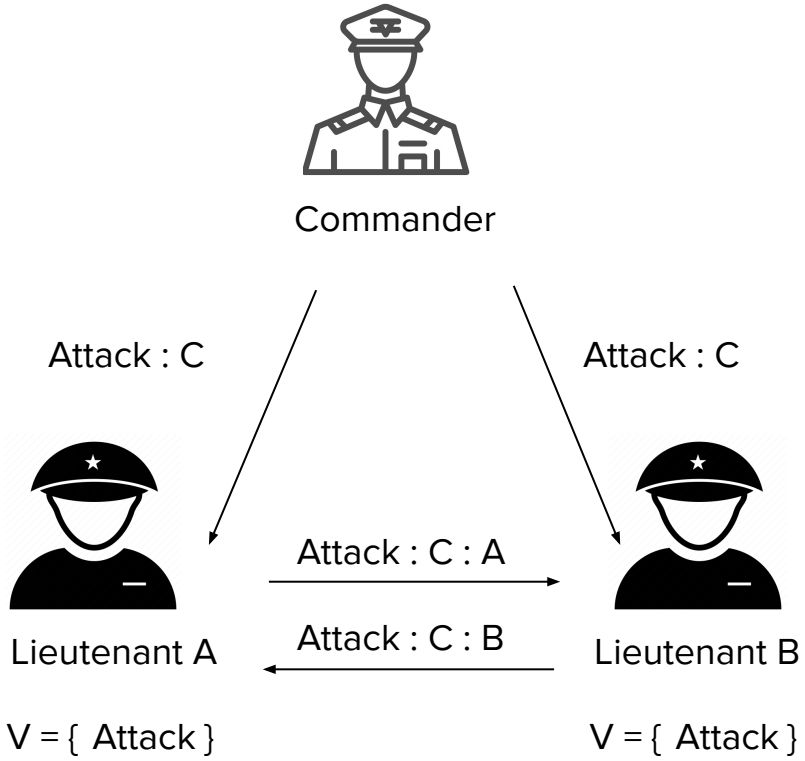
A4

- A loyal general's signature cannot be forged, and any alteration of the contents of his signed messages can be detected.
- Anyone can verify the authenticity of a general's signature.

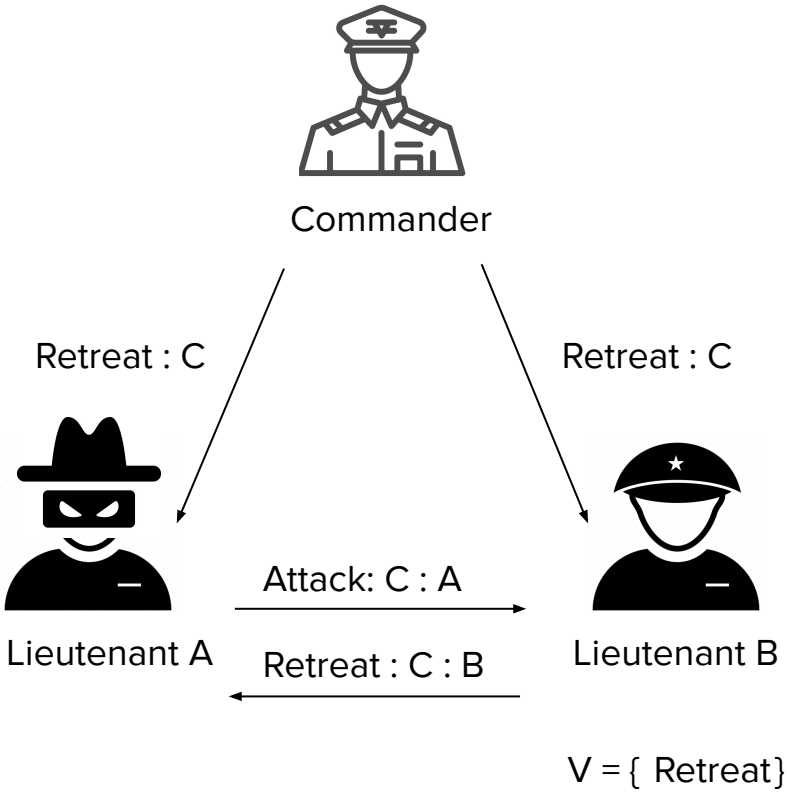
Signed Messages Algorithm - SM(m)

Each Lieutenant has a set of orders V

- Generals send signed order
- Lieutenant receives an order
    - Verifies authenticity & puts in V
    - If m < distinct signatures, sign message
- Sends to Lieutenants that haven't seen
- When no new messages, use choice(V) to decide action



SM(1) - Traitor Lieutenant

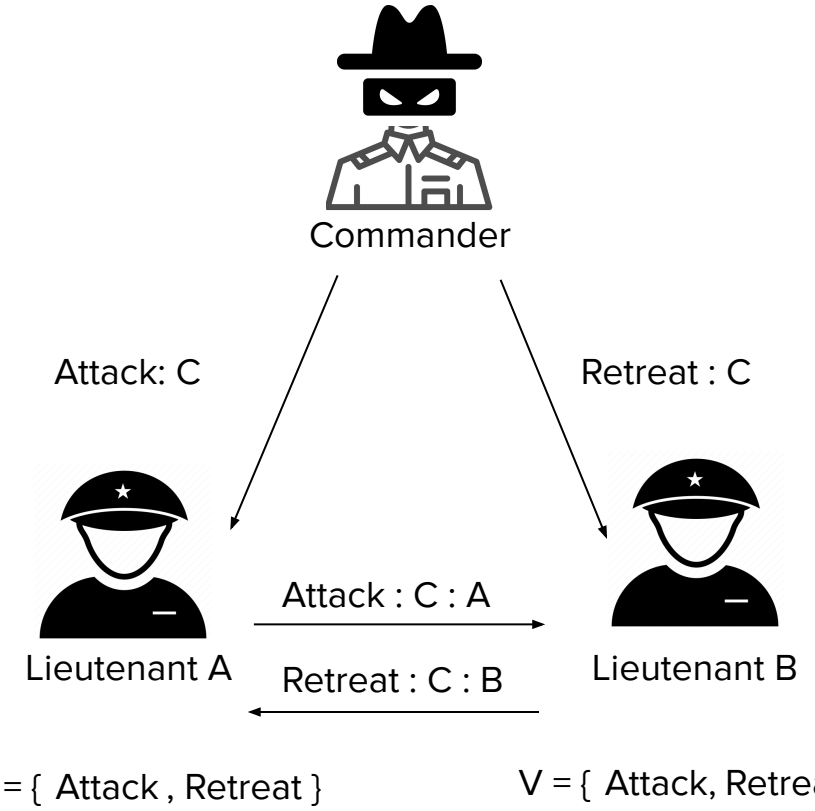


Lieutenant B ignores the traitor's message

Satisfies IC1 & IC2!!

V = { Retreat}

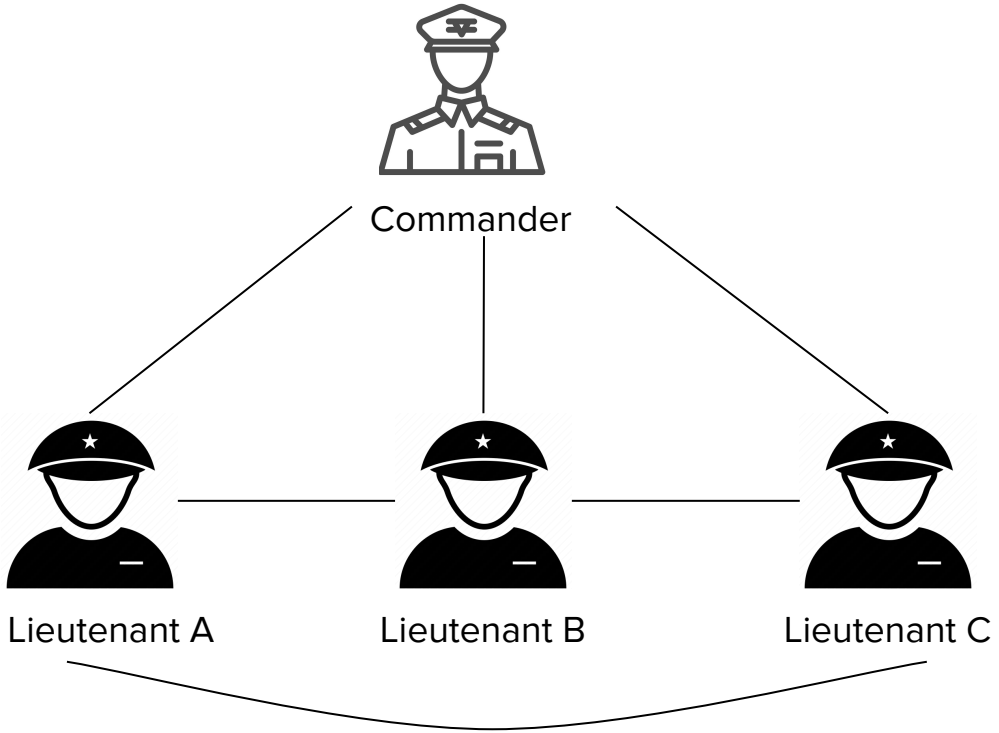
SM(1) - Traitor General



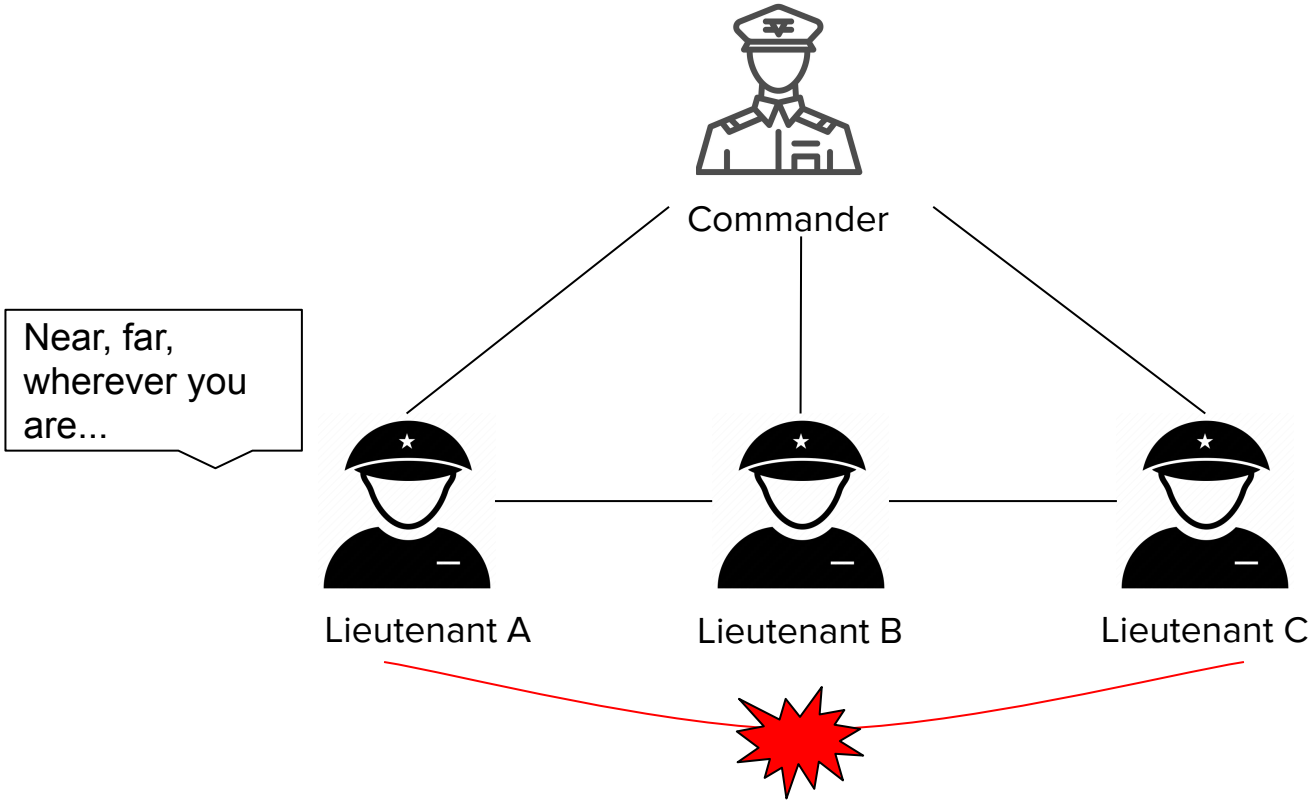
Both Loyal Lieutenants get same set V of orders

Satisfies IC1!!

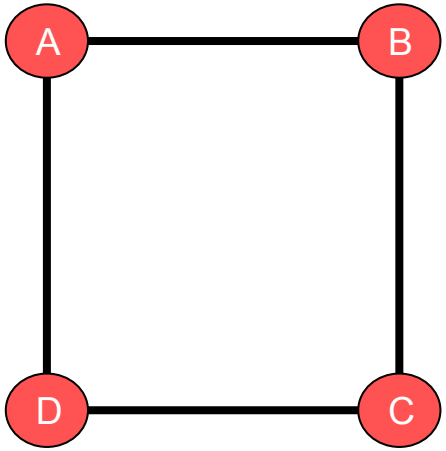
Can We Do Better?



What if...


Commander
Near, far, wherever you are...
Lieutenant A
Lieutenant B
Lieutenant C

A p-regular graph?



2-regular graph

Every node has the same amount of neighbors

Every node's neighbors has a path to some other node, where they share no common node other than the endpoint

p is the amount of neighbors per node

p-graph Examples

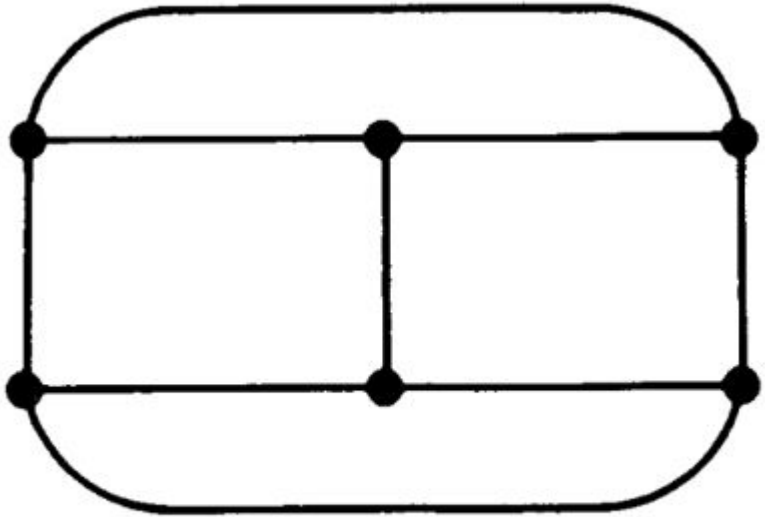


Fig. 6. A 3-regular graph.

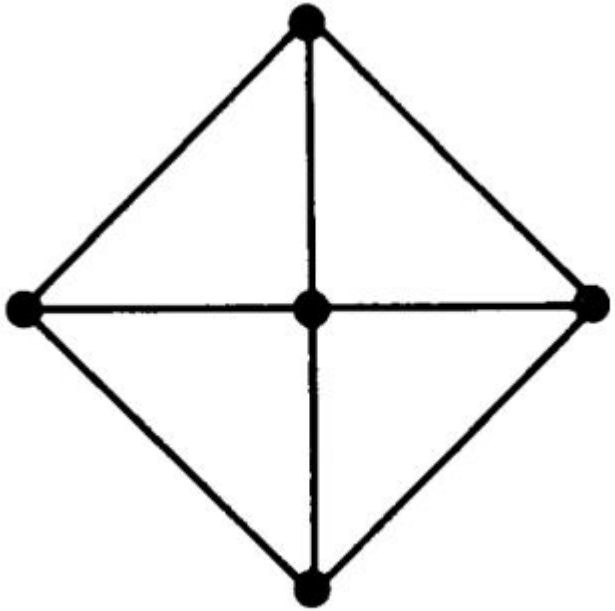


Fig. 7. A graph that is not 3-regular.

Extending Oral Messages for Missing Paths

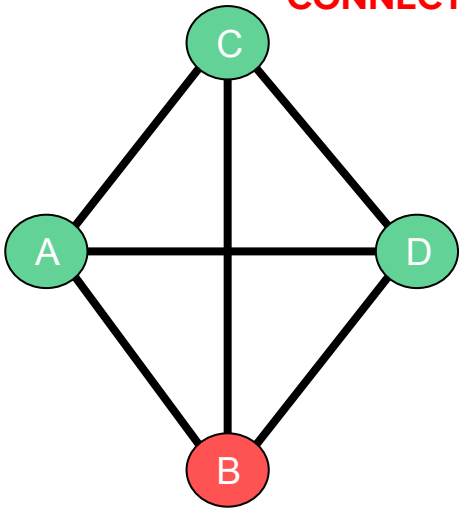
Commander sends message to neighbors only

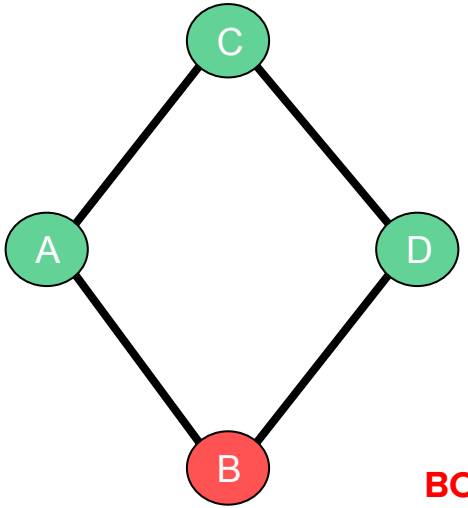
Lieutenants send messages to each other via paths that don’t include the Commander

Solves for p >= 3m

Solved But There's a Catch?

FULLY CONNECTED!

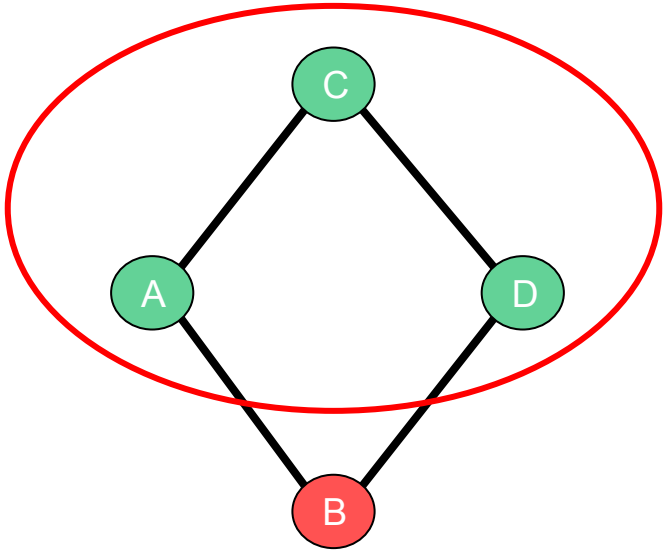




BOTTLENECK

Signed Messages for Missing Paths

Solves if subgraph of loyal generals is connected



WORKS FOR NON P-GRAPHS TOO!

In Terms of Computing Systems

IC1. All non-faulty processors must use the same input value (so they produce the same output)

IC2. If the input unit is non-faulty, then all non-faulty processes use the value it provides as input (so they produce the correct output).

Computing Systems - Assumption 1

Every message sent by a non-faulty processor is delivered correctly

Computing Systems - Assumption 2

Any processor can determine the originator of any message that it received.

Computing Systems - Assumption 3

Absence of a message can be detected

Computing Systems - Assumption 4

Processors must be able to sign their messages in such a way that a non-faulty processor's signature cannot be forged.

Conclusion

- Consensus w/o trust is hard
- Reasonable solutions (Expensive & Complex)
- Practical Application - Reliability vs Performance

Q&A