The byzantine generals problem

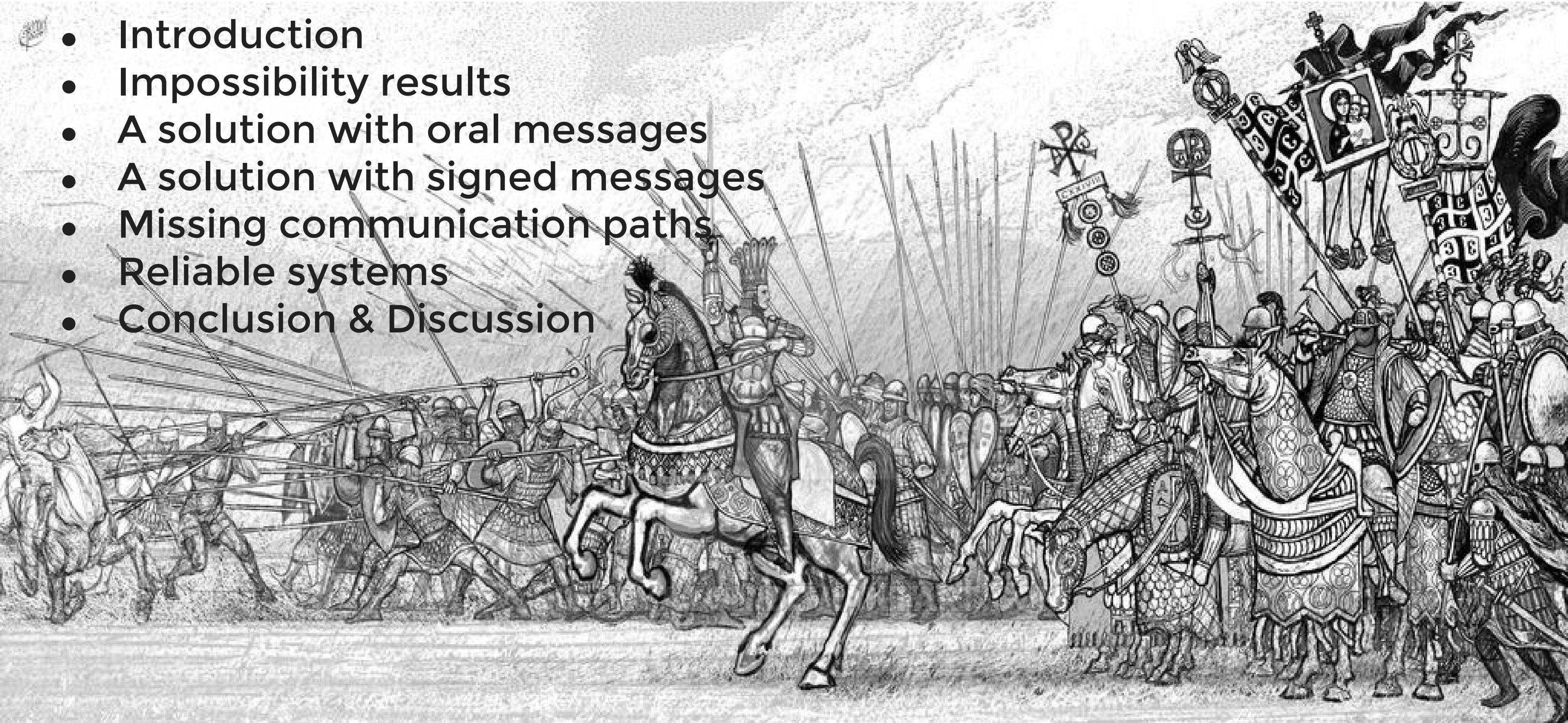
L. Lamport, R. Shostak, and M. Pease @ SRI International ACM Transactions on Programming Languages and Systems, July 1982, pp 382-401

Fernanda Mora
Luis Román

“A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable”

- Leslie Lamport

Contents

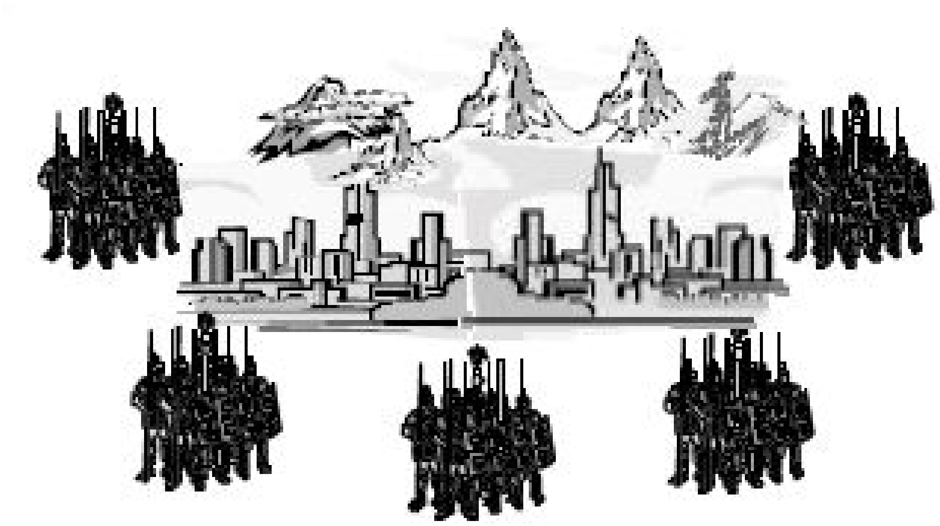


- Introduction
- Impossibility results
- A solution with oral messages
- A solution with signed messages
- Missing communication paths
- Reliable systems
- Conclusion & Discussion

Introduction

Consensus in synchronous faulty systems: BGP

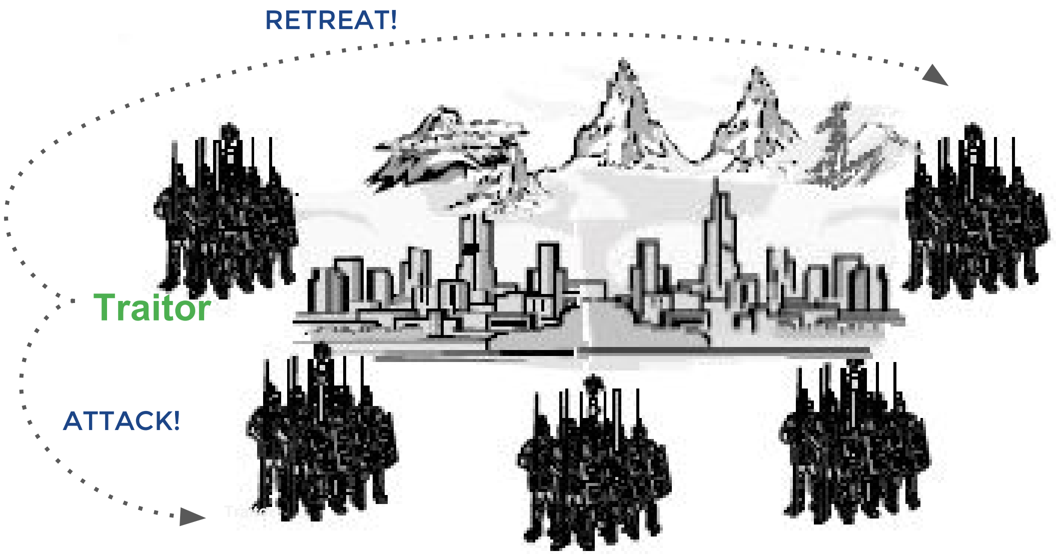
- The problem of coping with arbitrary, random failures (byzantine) is the Byzantine Generals Problem (BGP)



- Byzantine Army is divided in groups leadered by generals (nodes)
- Generals can communicate with each other using a messenger: **ATTACK or RETREAT**

Consensus in synchronous faulty systems: BGP

- The problem of coping with arbitrary, random failures (byzantine) is the Byzantine Generals Problem (BGP)



We must decide upon a **common** plan of action

Problem: some of the generals are traitors (~faulty nodes)!

We don’t know who the traitors are!

We want to guarantee:

1. All loyal generals decide upon the same plan of action.

   They should use the same information v(1),...,v(n)

2. A small number of traitors cannot cause the loyal generals to adopt a bad plan.

   We need a robust method: how does generals reach a decision?

We can have conditions on the *ith* general:

1. All loyal generals decide upon the same plan of action.

   Any two loyal generals use the same value v(i), for all i

2. A small number of traitors cannot cause the loyal generals to adopt a bad plan.

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

Byzantine Generals problem (BGP)

We can restrict on how a single general sends his value to others:

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

IC1. All loyal lieutenants obey the same order.

IC2. If a commander is loyal, then every loyal

Traitor

lieutenant obeys the order he sends.

Byzantine Generals problem (BGP)

To solve our original problem (i.e. decide a plan), the *ith* general sends his value v(i) by using a solution to the BGP to send the order “use v(i) as my value”, with the other generals acting as the lieutenants.

Traitor

Impossibility results

Impossibility of having ⅓ or more traitors using oral messages

- 3 generals: 2 loyal, 1 traitor -> no solution!

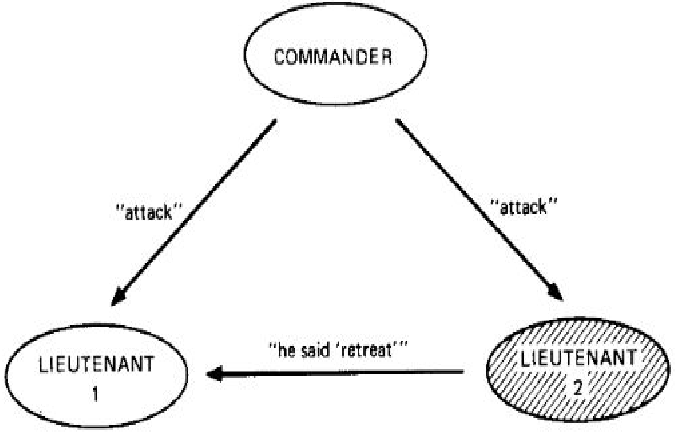


Fig. 1. Lieutenant 2 a traitor.

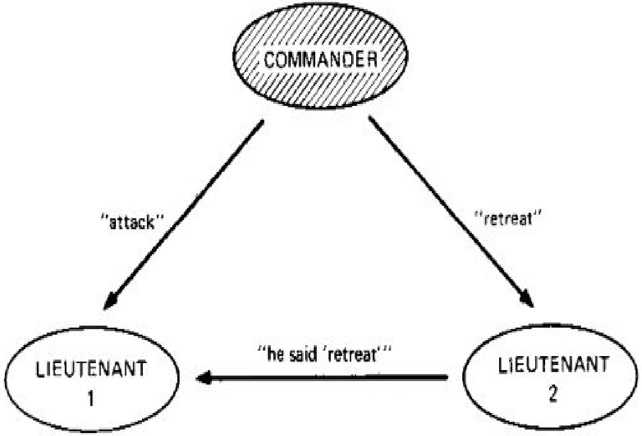


Fig. 2. The commander a traitor.

Traitor

Impossibility of having ⅓ or more traitors using oral messages

- **No solution** with fewer than *3m+1* generals can cope with *m* traitors
- With **m traitors**, we need $n \geq 3m+1$ generals
- Reaching **approximate agreement** is as hard as exact agreement

Traitor

A solution with oral messages

Assumptions

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

Prevent from traitor interfering communication

Prevents a traitor's boycott

* *m* traitors and at least *3m+1* generals

Oral message algorithm (recursive)

A commander sends an order to n-1 lieutenants a majority function such that

- $majority(v_1,\ldots,v_{n-1}) = mode\{v_1,\ldots,v_{n-1}\}$ or RETREAT if not order is received, or
- $majority(v_1,\ldots,v_{n-1}) = median\{v_1,\ldots,v_{n-1}\}$

Algorithm OM(0) (base case):

1. The commander sends its value to every n-1 lieutenants.
2. Each lieutenant uses the value he receives from the commander, or uses RETREAT if he receives no value.

Oral message algorithm (recursive)

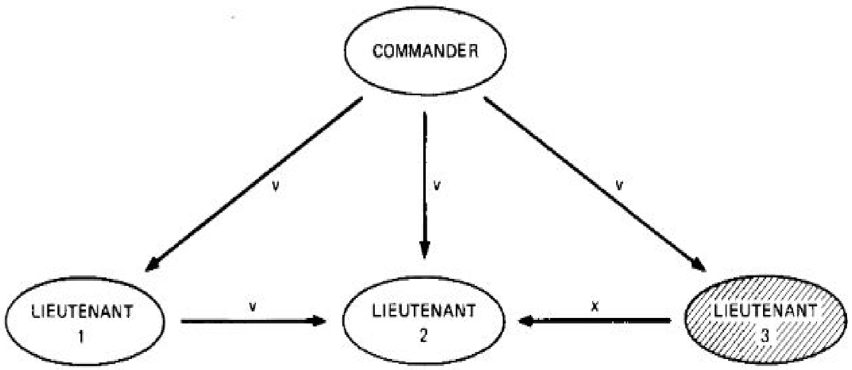
Algorithm OM(m), m>0 (recursive step):

1. The commander sends his value to every lieutenant.
2. For each *i*, let v_i the value lieutenant *i* receives from the commander, or else RETREAT. Lieutenant *i* acts as the commander in algorithm *OM(m-1)* to send the value v_i to each of the *n-2* other lieutenants.
3. $\forall$ i, j, i≠j, let v_i be the value lieutenant *i* received from lieutenant *j* in step 2 (using algorithm OM(m-1)), or else RETREAT. Lieutenant *i* uses the value majority(v_1,v_2,..,v_n).

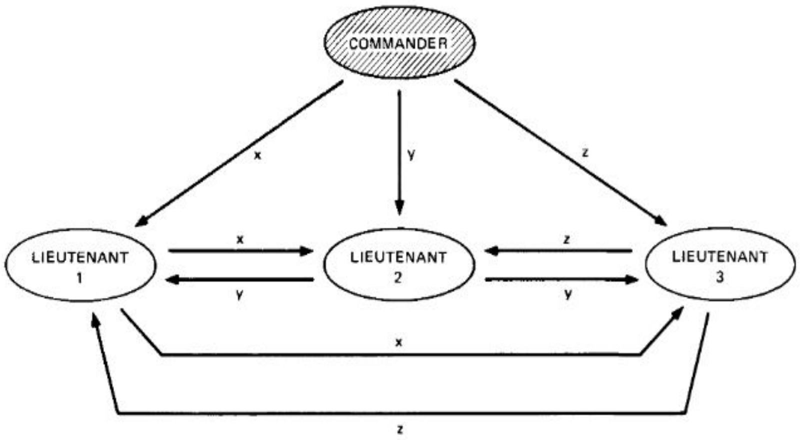
Oral message algorithm: remarks

- Lieutenants recursively forward orders to all the other lieutenants
- Algorithm O(m-k) is called (n-1)*...*(n-k) times to send a value prefixed with *k* lieutenants values
- Commander's order = majority (v_c, v_1, ..., v_n)
- v_i = majority (v_i, $v_{i,2}$,..., $v_{i,n}$), $1 \leq i \leq n$
- $v_{i,j}$ = majority ($v_{i,j}$, $v_{i,j,3}$, $v_{i,j,4}$, ...)
- Unfolding: OM(m) invokes n-1 executions of OM(m-1), which invokes n-2 executions of OM(m-2), ...
- Total number of messages: (n-1)*(n-2)*...*(n-m-1)

Oral message algorithm: example



Algorithm OM(1): Lieutenant 3 a traitor



Algorithm OM(1): Commander a traitor

Two results

Lemma 1. For any m and k, Algorithm *OM(m)* satisfies 1 if there are more than $2k+m$ generals and at most k traitors.

Theorem 1. For any m, algorithm *OM(m)* satisfies conditions C1 and C2 if there are more than $3m$ generals, and at most m traitors.

A solution with signed messages

More assumptions

1. A loyal general's signature cannot be forged.

2. Signatures can be authenticated.

Traitor

Authentication requirements are compensated by a more resilient to faults algorithm

Algorithm SM(m) - overview

1. Commander sends a signed order to its lieutenants.
2. Each lieutenant adds his signature to that order and sends it to the others, who sign the order and send it to others, etc.
3. Each lieutenant maintains a set of orders he has received, i.e., the possible sets are:

   {attack}, {retreat}, {attack, retreat}, Ø
4. Lieutenant takes action according to the value of the set

   {attack, wait} means the commander is a traitor!

   Traitor

Remarks

- We need a function *choice* to choose an order from a set of orders V:
    - If V={v} then choice(V)=v
    - If V={Ø} then choice(V)=retreat
    - else choice(V)=median(V) (for example)

Traitor

Signed messages algorithm SM(m)

$V_i = \emptyset$, General 0 is the commander

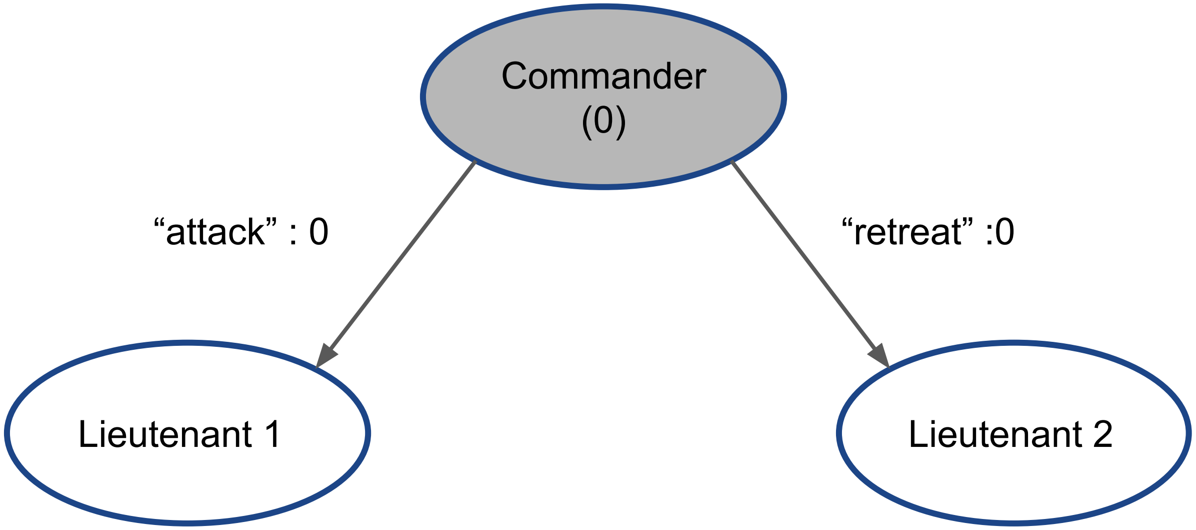
1. Commander signs *v* and sends *v:0* to all lieutenants.
2. For each lieutenant *i*:
   a. If *i* receives *v:0* and $V_i = \emptyset$
      i. $V_i = \{v\}$
      ii. sends *v:0:i* to every other lieutenant.
   b. If *i* receives $v{:}0{:}j_1{:}\ \ldots j_k$ and $v \notin V_i$
      i. Add *v* to V_i
      ii. if $k<m$ sends $v{:}0{:}j_1{:}\ \ldots j_k{:}i$ to all lieutenants $\notin \{j_1{:}\ \ldots j_k\}$
3. When no more messages, *i* obeys order of *choice*(v_i)

signature of mth lieutenant is not necessary

timeout
k>=m

Example: It’s possible to have 3 generals and 1 traitor

<u>Commander is traitor</u>

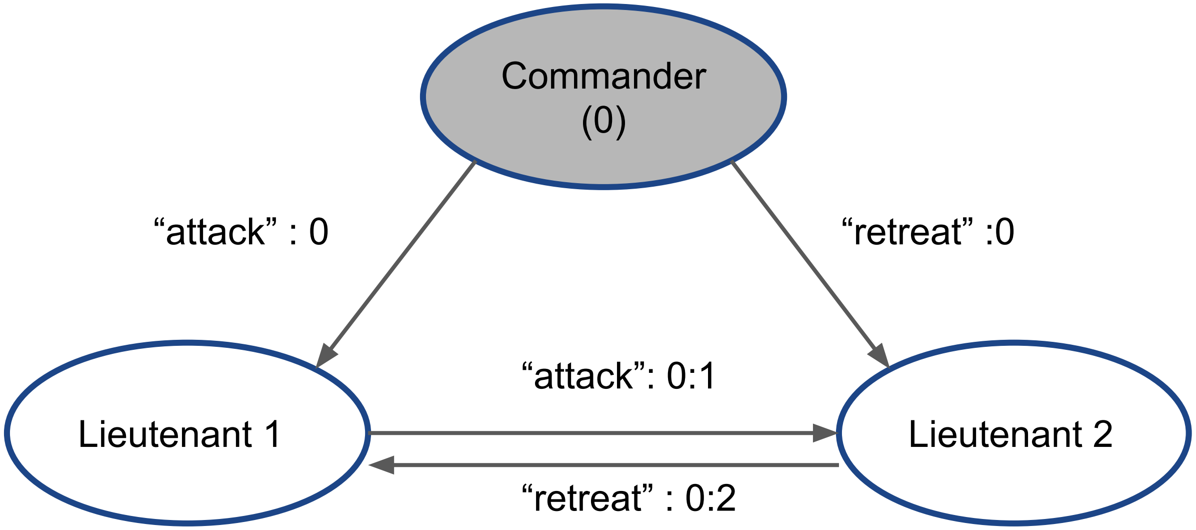


Step 1.

Commander sends signed messages to L1 and L2

Traitor

Example: It’s possible to have 3 generals and 1 traitor



Traitor

Commander is traitor

Step 2.

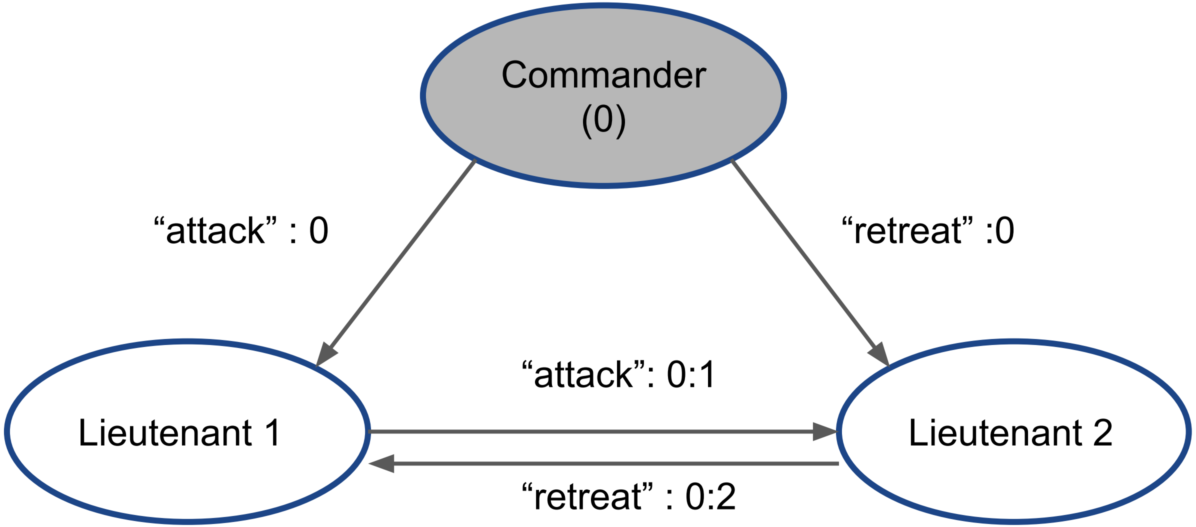
L1: V_1 = {“atack”}, sends “attack”: 0:1 to L2

L2: V_2= {“retreat”}, sends “retreat” : 0:2 to L1

V_1 = V_2= {“attack”,“retreat”}

L1, L2 know commander is traitor! (but no message is forged)

Example: It’s possible to have 3 generals and 1 traitor



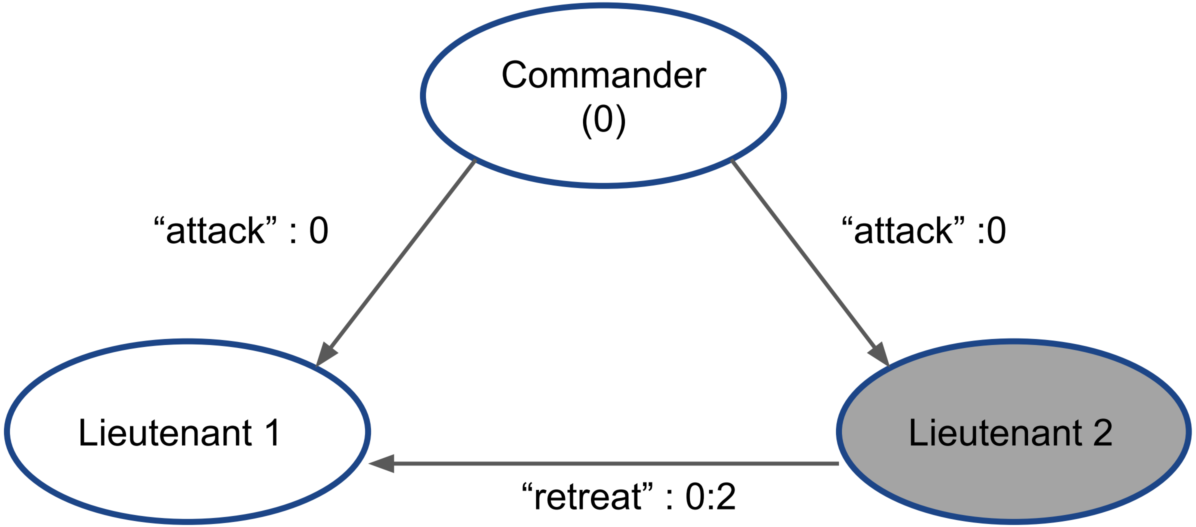
<u>Commander is traitor</u>

Step 3.

L1, L2 obey
choice ({“attack”,“retreat”})

IC1 and IC2 satisfied

Example: It’s possible to have 3 generals and 1 traitor



<u>Lieutenant is traitor</u>

Step 1.
Commander sends signed messages to L1 and L2

Step 2.
L2: V_2= {“attack”}
L2 to L1 “retreat”: 0:2

L1: V_1= {“attack”}
L1 receives “retreat” : 0:2
“L2 forged!”
L1 rejects the message

Step 3.
L1 “attacks”
IC2, IC1 satisfied

Some proofs

Theorem 2. For any *m*, algorithm *SM(m)* solves the BGP if there are at most m traitors an n>=m+2.

Sketch: *Case 1: Commander is loyal: every loyal i receives v, and every other loyal receives v. Then V_i=v, IC1->IC2.*

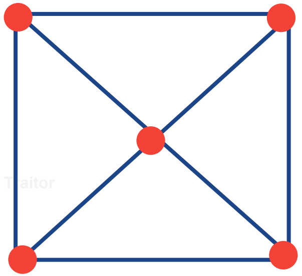
Case 2: Commander is traitor: consider a loyal lieutenant j with signature list |S|=m+1. At least one i in S is loyal. The message accepted and sent by i must be accepted by every loyal lieutenant j, and viceversa. Vi=Vj -> IC1

- *SM(m) sends (n-1) messages, each recipient (n-2), …*
- *$(n-1)(n-2)\cdots(n-m-1)$ messages to reach agreement*
- *Similar message complexity to OM(m) but with better resilience.*

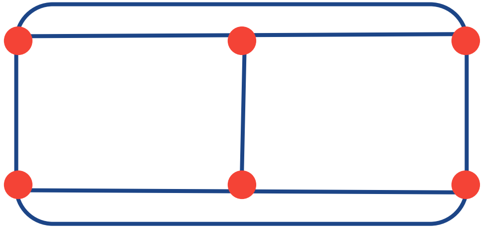
Missing communication paths

Extensions

- Network topology can restrict communication.
- This makes Byzantine problem more general.
- For simple graphs we can extend OM(m), SM(m)
- OM(m,3m) solves BGP in a p-regular graph (3m+1 nodes at least) (is the same as OM(m))
- SM(n-2) solves the GBP for *n* generals if loyal-graph is connected (can have missing links)



G is not 3-regular



G is 3-regular

Reliable systems

Ingredients

- Majority voting as a way to provide reliability
- What does we need?
  - Input synchronization of non-faulty processes (IC1)
  - If input is non-faulty, all non-faulty processes provide same output (IC2)
- A1 – communication line vs node failure
  - No problem: OM(m) or SM(m) can deal with it
- A2 – Fixed lines vs switching network
  - Not needed if we have A4
- A3 - Timeouts
- A4 - Cryptography

Conclusions
&
Discussion

Conclusions

- Reliability involves dealing with failure of components and is expensive: type of failure
- Byzantine failures produce arbitrary output making agreement challenging
- BGP used for input synchronization and handles m faults
- Two solutions: Oral and Signed Messages
- Expensive:

  Time: message latencies and signatures

  Messages: message paths<=m+1 (optimal) , $O(n^{m+1})$ messages

Discussion

- Can we determine m?
- How expensive is to implement a digital signature?
- How different is a “dumb” digital signature from an intelligent digital signature?
- If unfeasible to implement a digital signature, can we have 3m+1 nodes?

Thanks