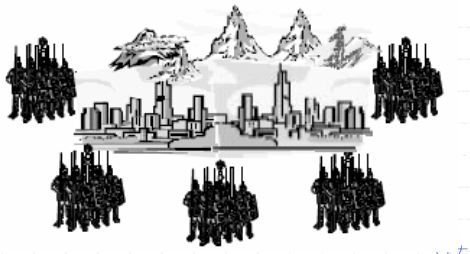
The Byzantine Generals Problem

COP 6614

Harish Ramakrishnan



Leslie Lamport, Robert Shostak, Marshall Pease

Overview

- Motivation
- Problem Definition
- Impossibility?
- Solutions (Oral and signed messages)
- Practical Use?
- Conclusions

Motivation

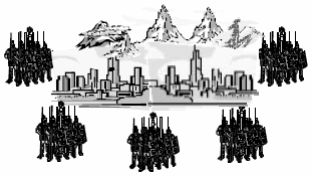
- Reliability of the Computer System
- Coping with failures in computer systems
- Failed component sends conflicting information to different parts of system.
- Agreement in the presence of faults.

P2P Networks?

- Good nodes have to “agree to do the same thing”.
- Faulty nodes generate corrupted and misleading messages.
- Non-malicious: Software bugs, hardware failures, power failures
- Malicious reasons: Machine compromised.

What is the Problem?

Problem Definition



- Generals = Computer Components
- The abstract problem…
  - Each division of Byzantine army is directed by its own general.
  - There are n Generals, some of which are traitors.
  - All armies are camped outside enemy castle, observing enemy.
  - Communicate with each other by messengers.

Conditions

- G1: All loyal generals decide upon the same plan of action
- G2: A small number of traitors cannot cause the loyal generals to adopt a bad plan
  - Note: We

Requirements

- Any two loyal generals must use the same value of v(i) to decide on same plan of action.
- If the i^{th} general is loyal, then the value he sends must be used by every loyal general as v(i).

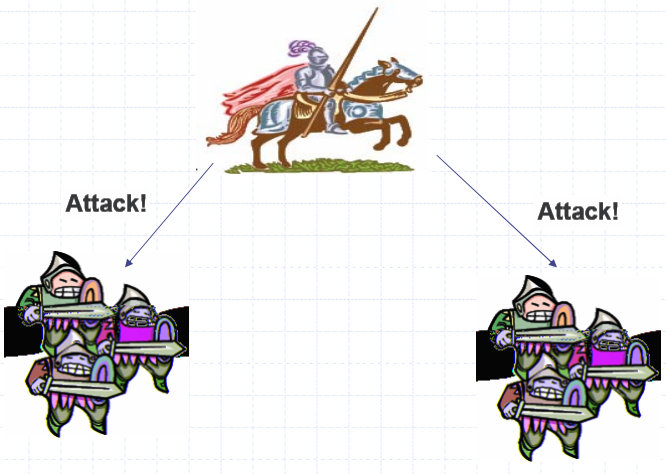
Reduction of General Problem

- **Insight:** We can restrict ourselves to the problem of one general sending its order to others.
- **Byzantine Generals Problem (BGP):**
  - A commanding general (commander) must send an order to his n-1 lieutenants.
- **Interactive Consistency Conditions:**
  - IC1: All loyal lieutenants obey the same order.
  - IC2: If the commanding general is loyal, then every loyal lieutenant obeys the order he sends.
- Note: If General is loyal, IC2 => IC1.
- Original problem: each general sends his value v(i) by using the above solution, with other generals acting as lieutenants.

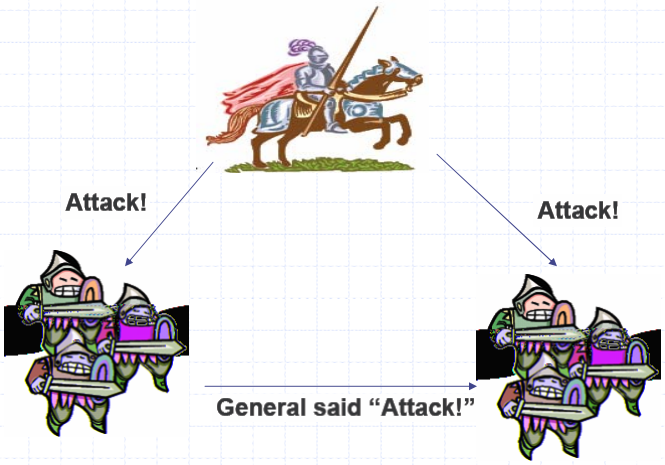
Here comes the Problem!

- Oral, easily changed messages
- No solution works unless more than two-thirds of the generals are loyal
- Even with just three generals, one traitor makes the protocol fail!

Proving the Impossibility

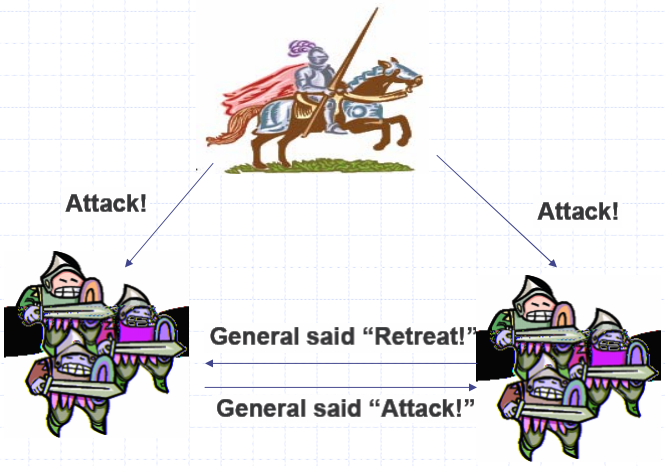


Proving the Impossibility



10/13/2004 Harish Ramakrishnan

Proving the Impossibility



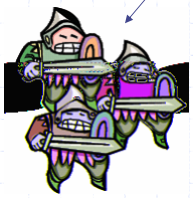
10/13/2004 Harish Ramakrishnan

Proving the Impossibility

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

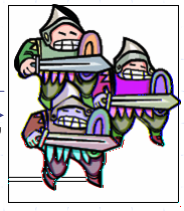
Attack!

Attack!



General said "Retreat!"

General said "Attack!"

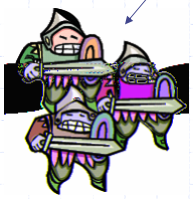


10/13/2004 Harish Ramakrishnan

Proving the Impossibility(2)

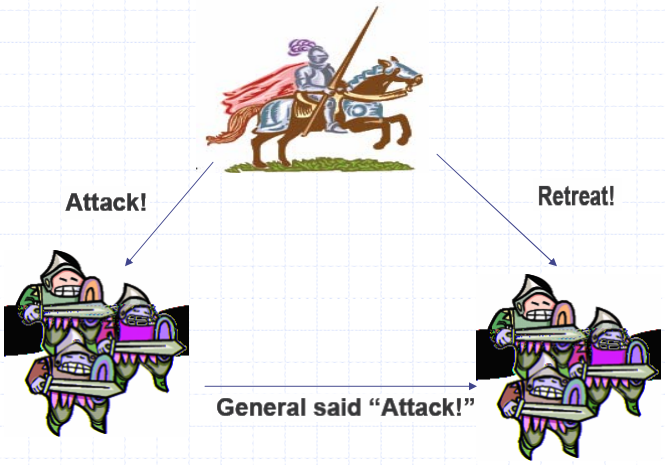
Attack!

Retreat!



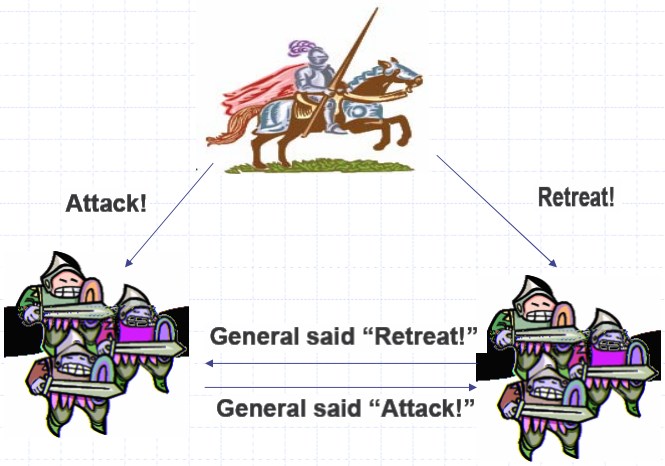
10/13/2004 Harish Ramakrishnan

Proving the Impossibility(2)



10/13/2004 Harish Ramakrishnan

Proving the Impossibility(2)



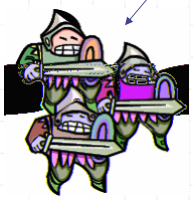
10/13/2004 Harish Ramakrishnan

Proving the Impossibility(2)

IC1: All loyal lieutenants obey the same order

Attack!

Retreat!



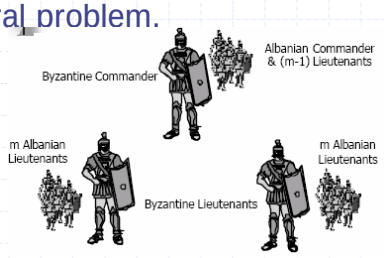
General said "Retreat!"

General said "Attack!"

General Impossibility

- In general, no solutions with fewer than 3m+1 generals can cope with m traitors.
- Proof by contradiction.
  - Assume there is a solution for *3m or fewer Albanians* with m traitors.
  - Reduce to 3-General problem.

- Solution to 3m problem => Solution to 3-General problem!!



Solution I – Oral Messages

- If there are 3m+1 generals, solution allows up to m traitors.
- Oral messages – the sending of content is entirely under the control of sender.
- Default order to “retreat” for silent traitor.

Solution I – Oral Messages(2)

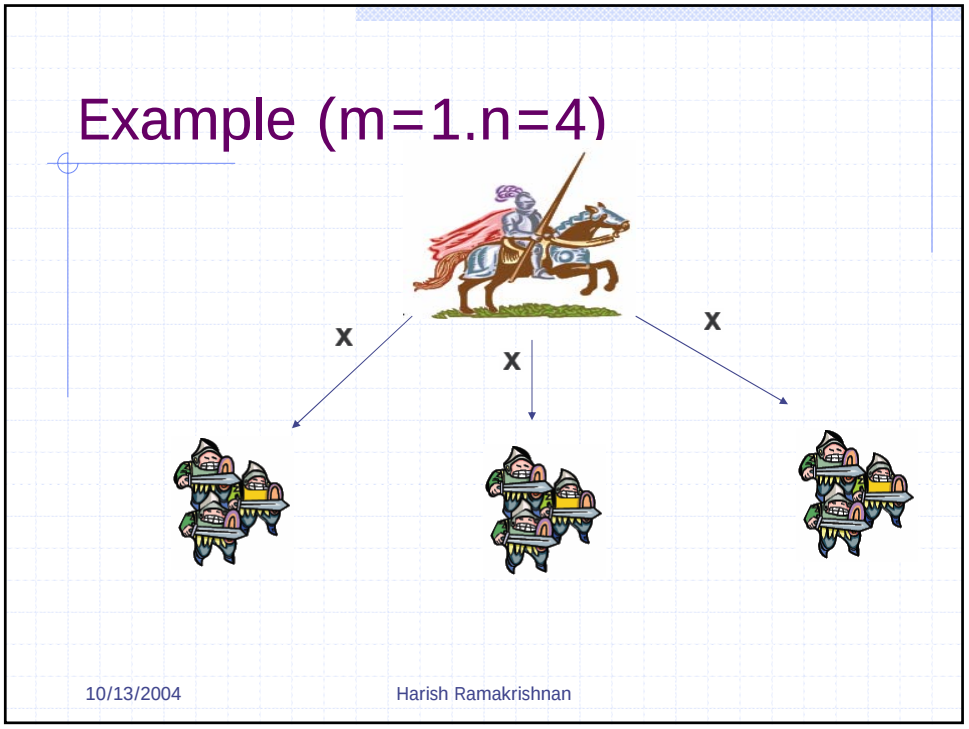
- Assumptions on oral messages:
  - A1 – Each message that is sent is delivered correctly.
  - A2 – The receiver of a message knows who sent it.
  - A3 – The absence of a message can be detected.
- Assures:
  - Traitors cannot interfere with communication as third party.
  - Traitors cannot send fake messages
  - Traitors cannot interfere by being silent.

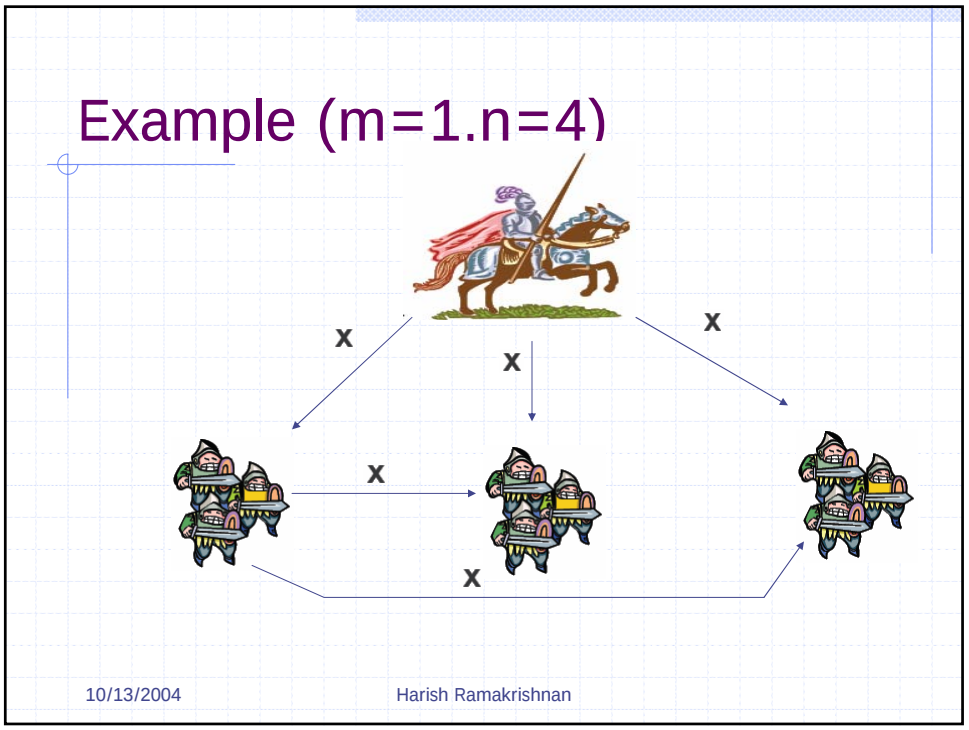
Oral Message Algorithm

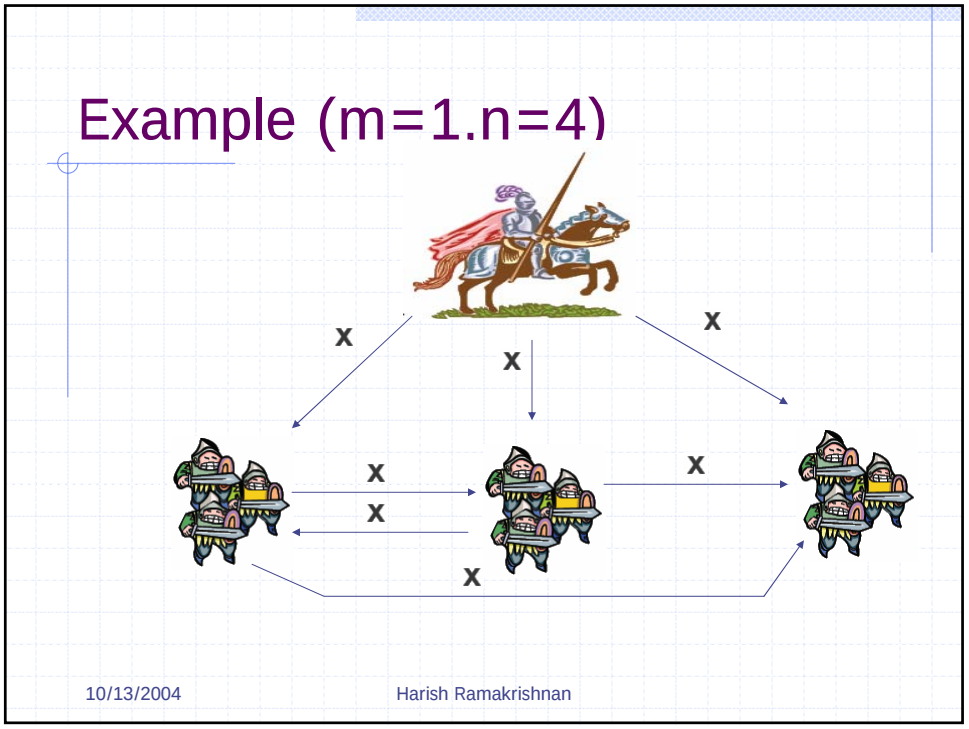
- Algorithm OM(0)
  1. Commander send his value to every lieutenant.
  2. Each lieutenant (L) use the value received from commander, or RETREAT if no value is received.
- Algorithm OM(m), m>0
  1. Commander sends his value to every Lieutenant (v_i)
  2. Each Lieutenant acts as commander for OM(m-1) and sends v_i to the other n-2 lieutenants (or RETREAT)
  3. For each i, and each j<>i, let v_j be the value lieutenant i receives from lieutenant j in step (2) using OM(m-1). Lieutenant i uses the value majority (v_1, ..., v_{n-1}).

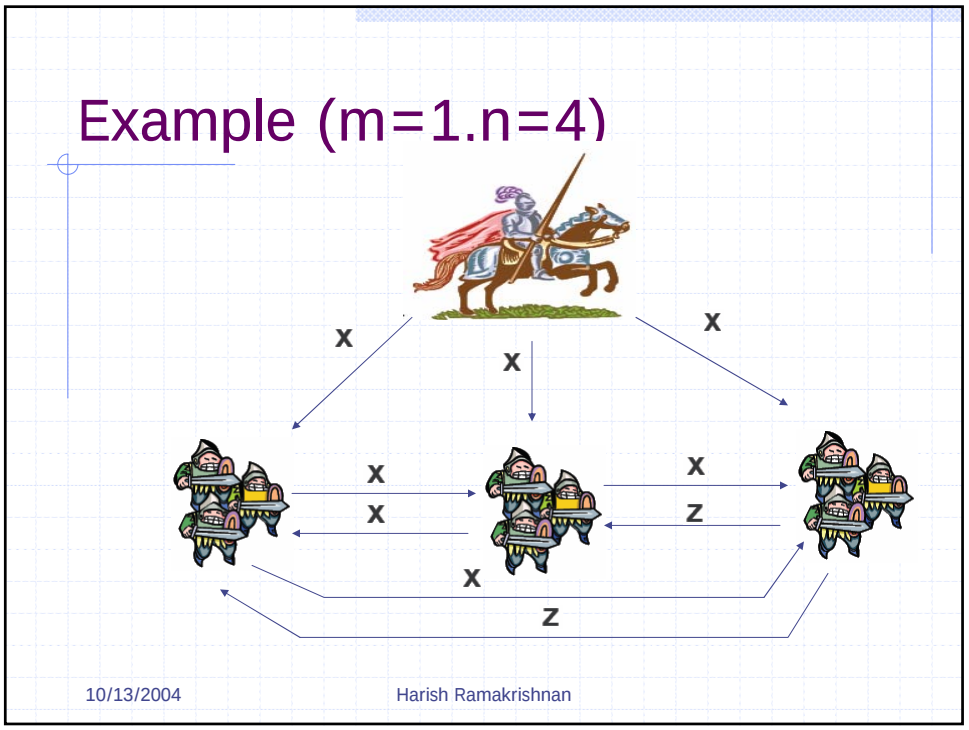
Restate Algorithm

- OM(M):
  - Commander sends out command.
  - Each lieutenant acts as commander in OM(m-1). Sends out command to other lieutenants.
  - Use majority to compute value based on commands received by other lieutenants in OM(m-1)
- Revisit Interactive Consistency goals:
  - IC1: All loyal lieutenants obey the same command.
  - IC2: If the commanding general is loyal, then every loyal lieutenant obeys the command he sends.


Example (m=1.n=4)
x
x
x
10/13/2004
Harish Ramakrishnan

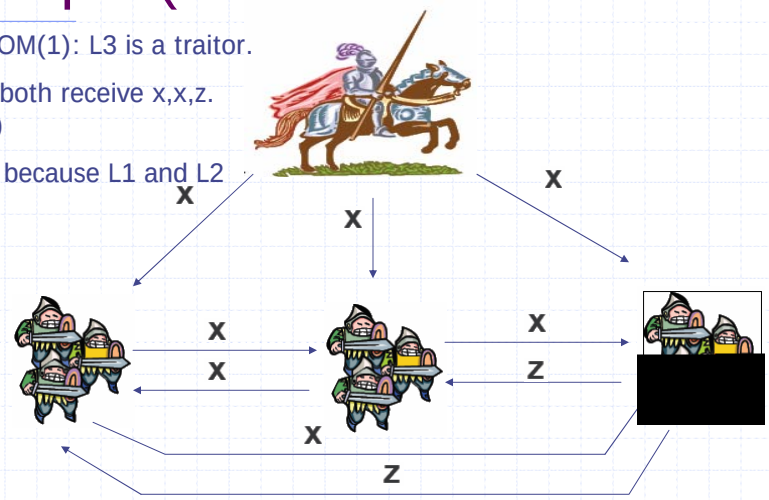

Example (m=1.n=4)
x
x
x
x
x
10/13/2004
Harish Ramakrishnan


Example (m=1.n=4)
X
X
X
X
X
X
X
X
10/13/2004
Harish Ramakrishnan

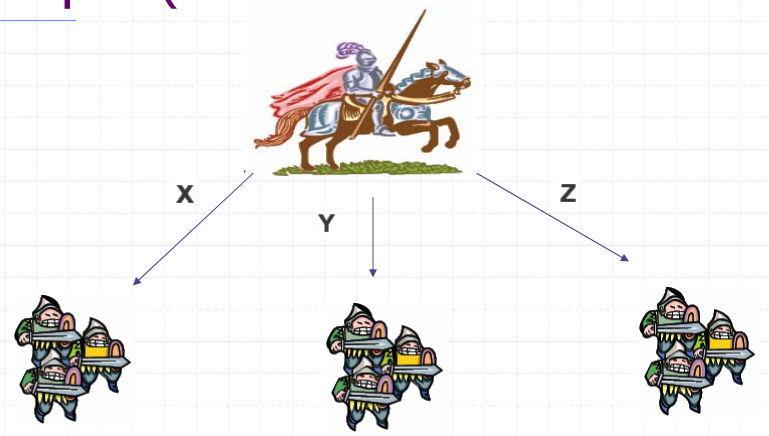

Example (m=1.n=4)
X
X
X
X
X
X
Z
X
Z
10/13/2004
Harish Ramakrishnan

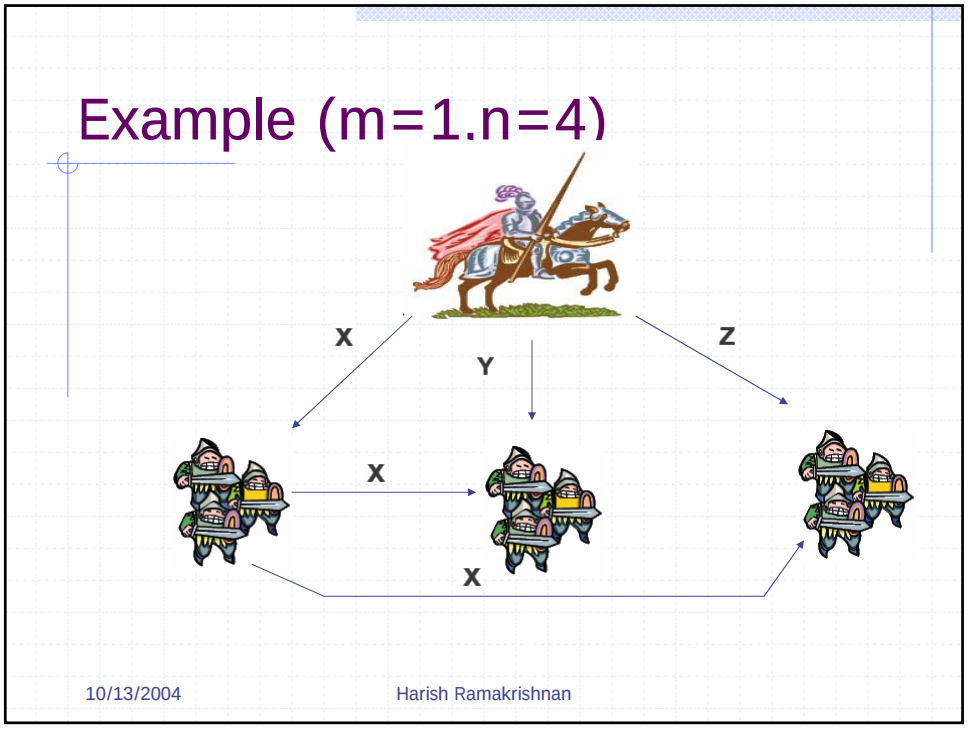
Example (m=1.n=4)

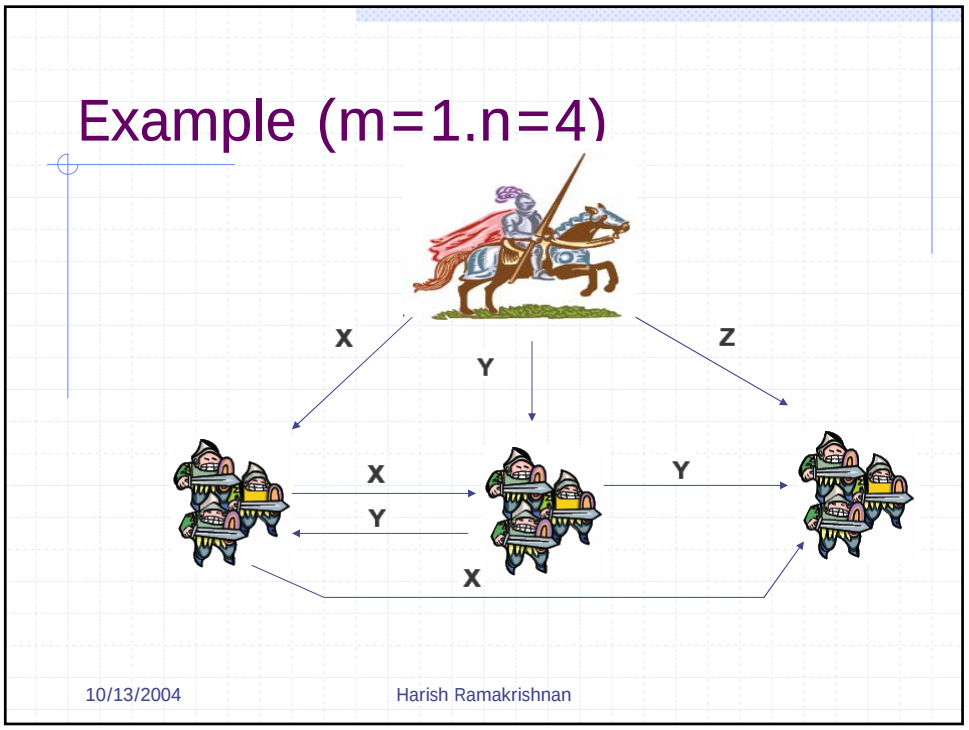
- Algorithm OM(1): L3 is a traitor.
- L1 and L2 both receive x,x,z. (IC1 is met.)
- IC2 is met because L1 and L2 obeys C

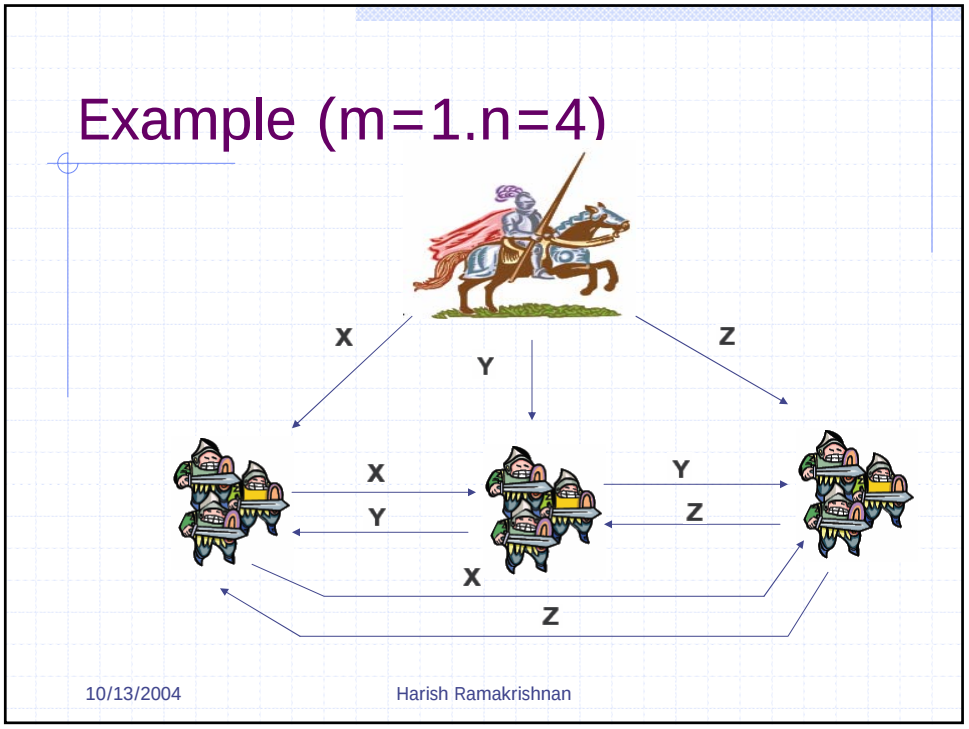


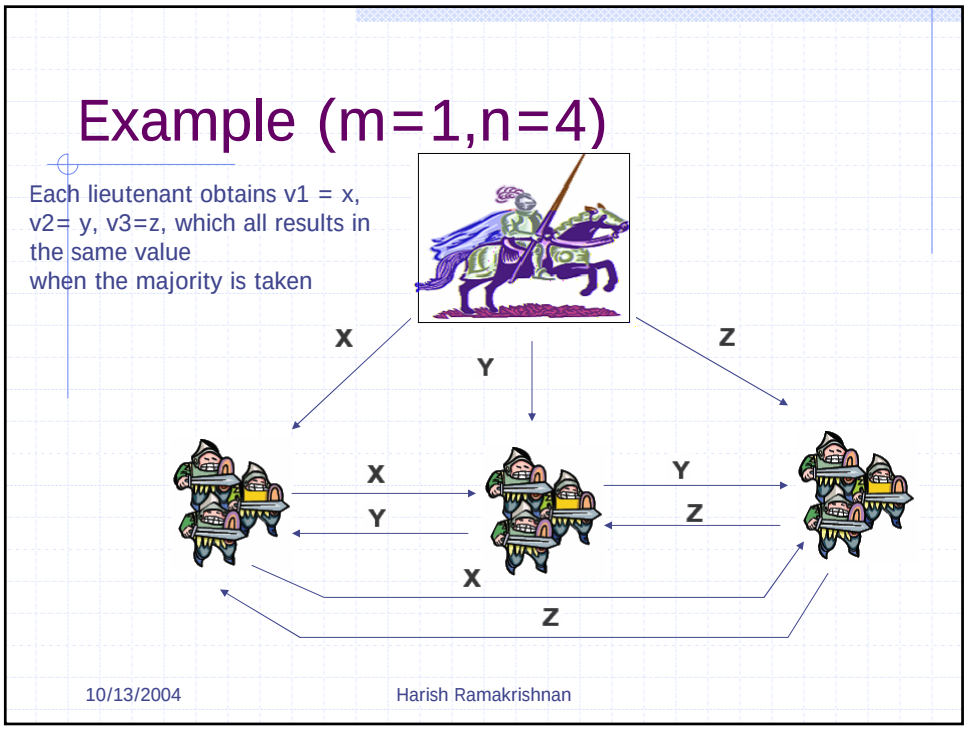
Example (m=1.n=4)




Example (m=1.n=4)
X
Y
Z
X
X
10/13/2004
Harish Ramakrishnan


Example (m=1.n=4)
X
Y
Z
X
Y
Y
X
10/13/2004
Harish Ramakrishnan


Example (m=1.n=4)
X
Y
Z
X
Y
Y
Z
X
Z
10/13/2004
Harish Ramakrishnan


Example (m=1,n=4)
Each lieutenant obtains v1 = x, v2= y, v3=z, which all results in the same value when the majority is taken
X
Y
Z
X
Y
Y
Z
X
Z
10/13/2004
Harish Ramakrishnan

Expensive Communication

- OM(m) invokes n-1 OM(m-1)
- OM(m-1) invokes n-2 OM(m-2)
- OM(m-2) invokes n-3 OM(m-3)
- ...
- OM(m-k) will be called (n-1)...(n-k) times
- $O(n^m)$ – Expensive!

The Dreaded Proof

- For any *m*, algorithm OM(*m*) satisfies conditions IC1 (All loyal lieutenants obey the same order) and IC2 (If the commanding general is loyal, then every loyal lieutenant obeys the order he sends) if there are more than 3*m* generals and at most *m* traitors.
- Induction on m proves true in all cases.

Solution II: Signed messages

- Previous algorithm allows a traitor to lie about the commander's orders (command). We prevent that with signatures to simplify the problem.
- By simplifying the problem, we can cope with any number of traitors as long as their maximum number (m) is known.
- Additional Assumption A4:
  - A loyal general's signature cannot be forged.
  - Anyone can verify authenticity of general's signature.

Signed messages(2)

- Three general solution exist!
- Use a function *choice(...)* to obtain a single order
  - *choice*(V) = v if v if the only elem. in V
  - *choice*(V) = RETREAT if V is empty
- Each lieutenant maintains a set V of properly signed orders received so far.

Signed Message Algorithm

- The commander sends a signed order to lieutenants
- A lieutenant receives an order from someone (either from commander or other lieutenants),
  - Verifies authenticity and puts it in V.
  - If there are less than m *distinct* signatures on the order
    - Augments orders with signature
    - Relays messages to lieutenants who have not seen the order.
- When lieutenant receives no more messages, then use choice(V) as the desired action.
- To protect against more traitors, increase m

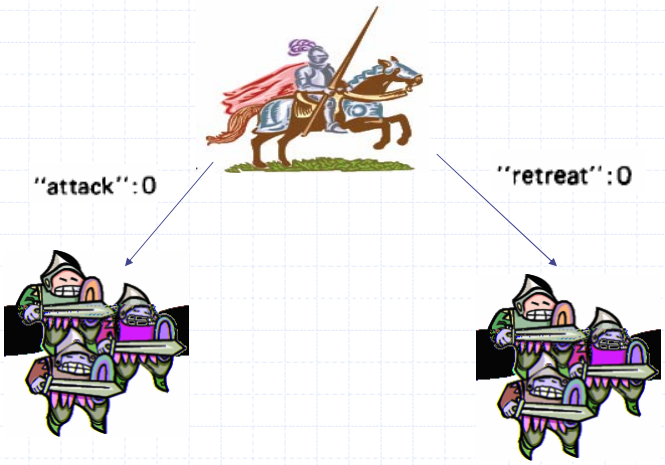
10/13/2004 Harish Ramakrishnan

What's going on here?

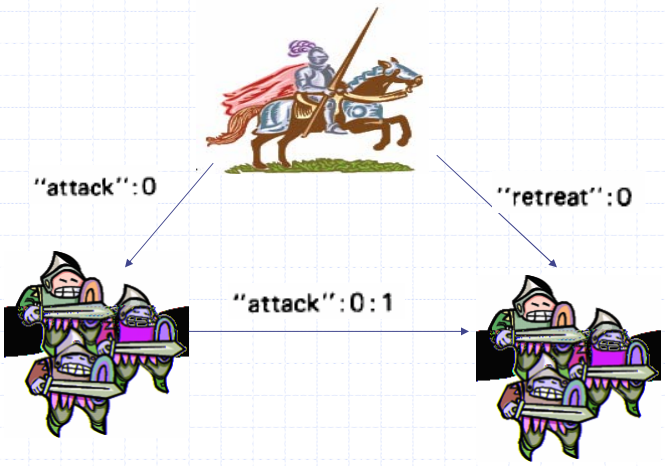
- All loyal lieutenants compute the same set of V eventually, thus choice(V) is the same (IC1)
- If the commander is loyal, the algorithm works because all loyal lieutenants will have the properly signed orders by round 1 (IC2)
- What if the commander is not loyal?

10/13/2004 Harish Ramakrishnan

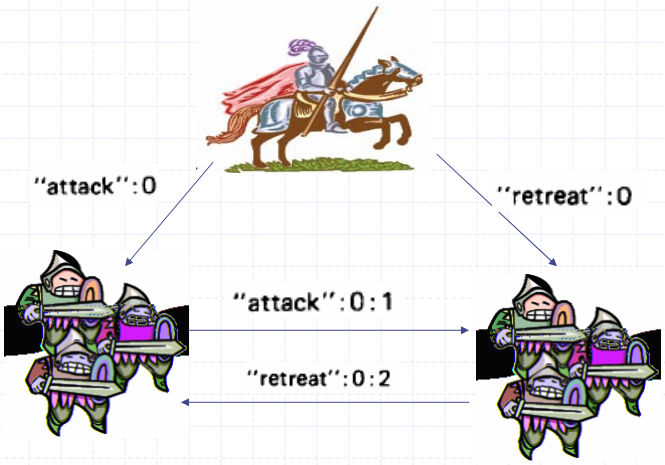
What's going on here?



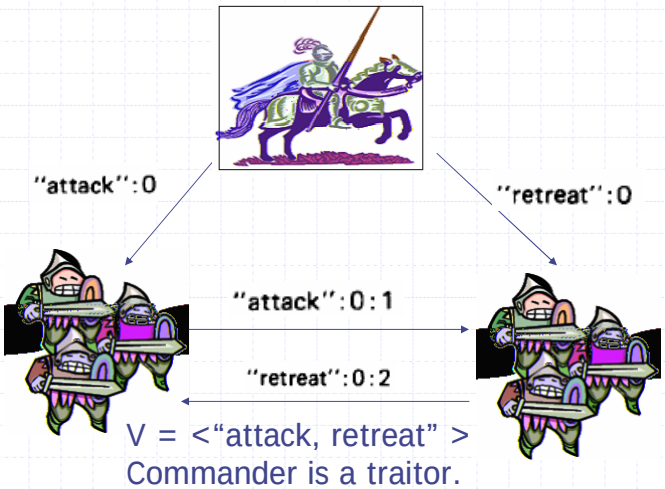
What's going on here?



What's going on here?



What's going on here?



V = <"attack, retreat" >
Commander is a traitor.

Missing Communication Paths

- What if not all generals can reach all other generals directly?
- Simple, Finite undirected graph
- Regular set of neighbors of node *i* if
  - Each element is a neighbor of *i*
  - For any general *k* different from *i*, exists different path from each element to *k* with no node other than *k* in common.
- p-regular graph
  - If every node has a regular set of neighbors with p distinct nodes.

Missing Communication Paths(2)

- P-regular graph – Each node has p regular neighbors.
- 3m-regular graph has minimum of 3m+1 nodes
- Paper shows algorithm for variant of oral message algorithm – OM(m,p). Essentially same algorithm except that each lieutenant forwards orders to neighbors.
- Proves that OM(m,3m) solves BGP for at most m traitors.
- i.e. if the communication graph is 3m-regular, and there are at most m traitors, the problem can still be solved.

Practical use of BGP?

- What does it take for majority voting to work?
  - Input synchronization (order from commander) of non-faulty (loyal) processors to produce same outputs (decisions). (IC1)
  - If input unit (commander) is non-faulty (loyal),all non-faulty processors use the value it provides as input (IC2)

Practical Use of BGP (Cont..)

- A1 – Every message sent by non-faulty processor is delivered correctly.
  - Failure of communication line cannot be distinguished from failure of nodes.
  - but we still are tolerating m failures.
- A2 – A processor can determine origin of message
  - Completely connected network.
  - A4 makes this obsolete.

Practical Use of BGP (Cont..)

- A3 – Absence of a message can be detected.
  - Timeouts or synchronized clocks
- A4 – Unforgeable signatures. Anyone can verify authenticity of signature
  - Message signed by i = (M, Si(M))
  - If i is not faulty, no one can generate Si(M). Faulty processor used for generating signatures?
  - Given M and X, anyone can verify if X=Si(M)

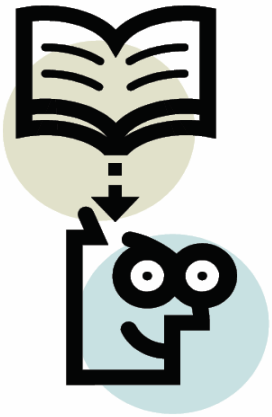
Concluding thoughts

- BGP solutions are expensive (communication overheads and signatures)
- Use of redundancy and voting to achieve reliability.
- What if >1/3 nodes (processors) are faulty?
- 3m+1 replicas for m failures. Is that expensive?
- Tradeoffs between reliability and performance

References

- LESLIE LAMPORT, ROBERT SHOSTAK, MARSHALL PEASE. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems, July 1982, pages 382-401.*
- PRADEEP. K. SINHA Distributed Operating Systems. *IEEE Press.*
- DIFFIE, W., AND HELLMAN, M.E. New directions in cryptography. *IEEE Trans. Inf. Theory IT-22 (Nov. 1976), 644-654.*
- DOLEV, D. The Byzantine generals strike again. *J. Algorithms 3, 1 (Jan. 1982).*
- PEASE, M., SHOSTAK, R., AND LAMPORT, L. Reaching agreement in the presence of faults. *J.ACM 27, 2 (Apr. 1980), 228-234.*
- RIVEST, R.L., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Cornrnun. ACM 21, 2 (Feb. 1978), 120-126.*

Questions???



Thank You!!!