CS G623: Advanced Operating Systems

Lecture 10

BITS Pilani
Pilani Campus

Amit Dua

August 25, 2018

Topics to be discussed

- Agreement
- System model
- Asynchronous vs synchronous
- Different types of failures
- Solutions

Agreement Protocol: The System model

- The are ***n*** processors in the system and at most ***m*** of them can be faulty
- The processors can directly communicate with other processors via messages (fully connected system)
- A receiver computation always knows the identity of a sending computation
- The communication system is reliable

Communication Requirements

- Synchronous communication model is assumed in this section:
  - Healthy processors receive, process and reply to messages in a lockstep manner
  - The receive, process, reply sequence is called a ***round***
  - In the synch-comm model, processes know what messages they expect to receive in a round
- The synch model is critical to agreement protocols, and the agreement problem is not solvable in an asynchronous system

Processor Failures

- Crash fault

–Abrupt halt, never resumes operation

- Omission fault

–Processor "omits" to send required messages to some other processors

- Malicious fault

–Processor behaves randomly and arbitrarily

–Known as ***Byzantine faults***

Message Types

- Authenticated messages (also called *signed* messages)

–assure the receiver of correct identification of the sender

- Non-authenticated messages (also called ***oral*** messages)

–are subject to intermediate manipulation

–may lie about their origin

Agreement Problems

Problem	Who initiates value	Final Agreement
Byzantine Agreement	One Processor	Single Value
Consensus	All Processors	Single Value
Interactive Consistency	All Processors	A Vector of Values

Practical applicability of BA

- Whether to commit or abort the results of a distributed commit action (database transaction)?

- Based on the readings of multiple altimeters, agreeing on an estimate of airplane's altitude

- Given the results of separate diagnostic tests performed by different processes, agreeing on whether to declare a system component as a faulty component

Origin at Byzantine

- May 29th, 1453
- The Turks are besieging the city of Byzantine by making a coordinated attack.

- Goals
  - Consensus between loyal generals
  - A small number of traitors cannot cause the loyals to adopt a bad plan
  - Do not have to identify the traitors

BA: Impossibility condition

- Theorem: There is no algorithm to solve byzantine if only oral messages are used, unless *more than two thirds* of the generals are loyal.

- In other words, impossible if $\boxed{n \leq 3f}$ for *n* processes, *f* of which are faulty

- *Oral messages* are under control of the sender
  - sender can alter a message that it received before forwarding it

- Let's look at examples for special case of n=3, f=1

Case 1

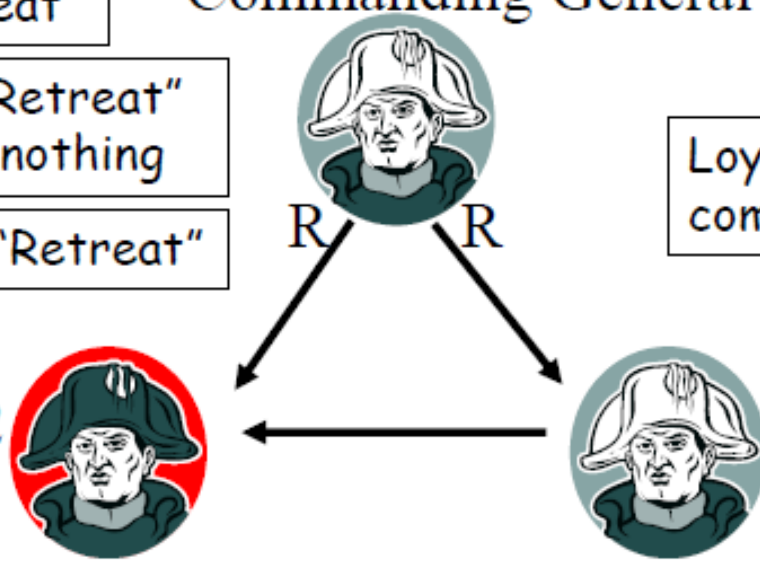
- Traitor lieutenant tries to foil consensus by <u>refusing to participate</u>

"white hats" == loyal or "good guys"
"black hats" == traitor or "bad guys"

Round 1: Commanding General sends "Retreat"

Round 2: L3 sends "Retreat" to L2, but L2 sends nothing

Decide: L3 decides "Retreat"

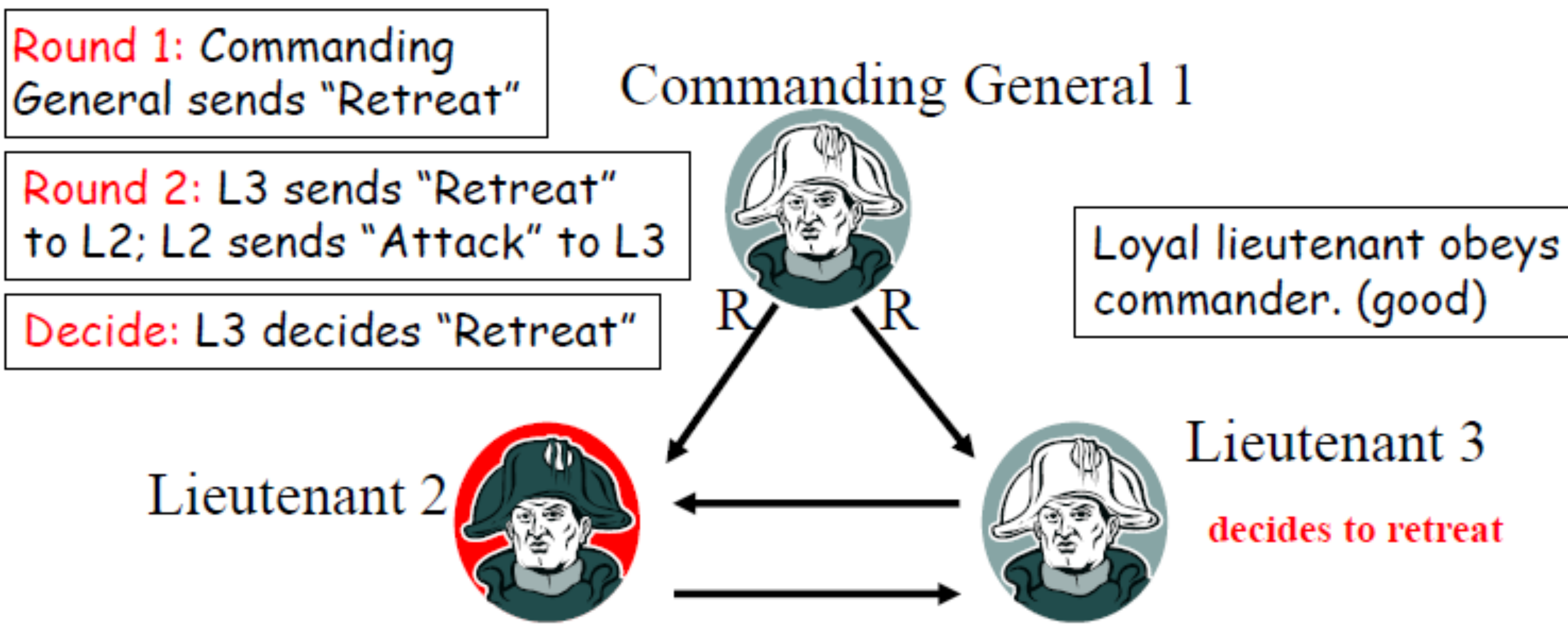


Loyal lieutenant obeys commander. (good)

Acknowledgement: Class lectures of A.D Brown of UofT

Case 2a

- Traitor lieutenant tries to foil consensus by <u>lying about order</u> sent by general



Acknowledgement: Class lectures of A.D Brown of UofT

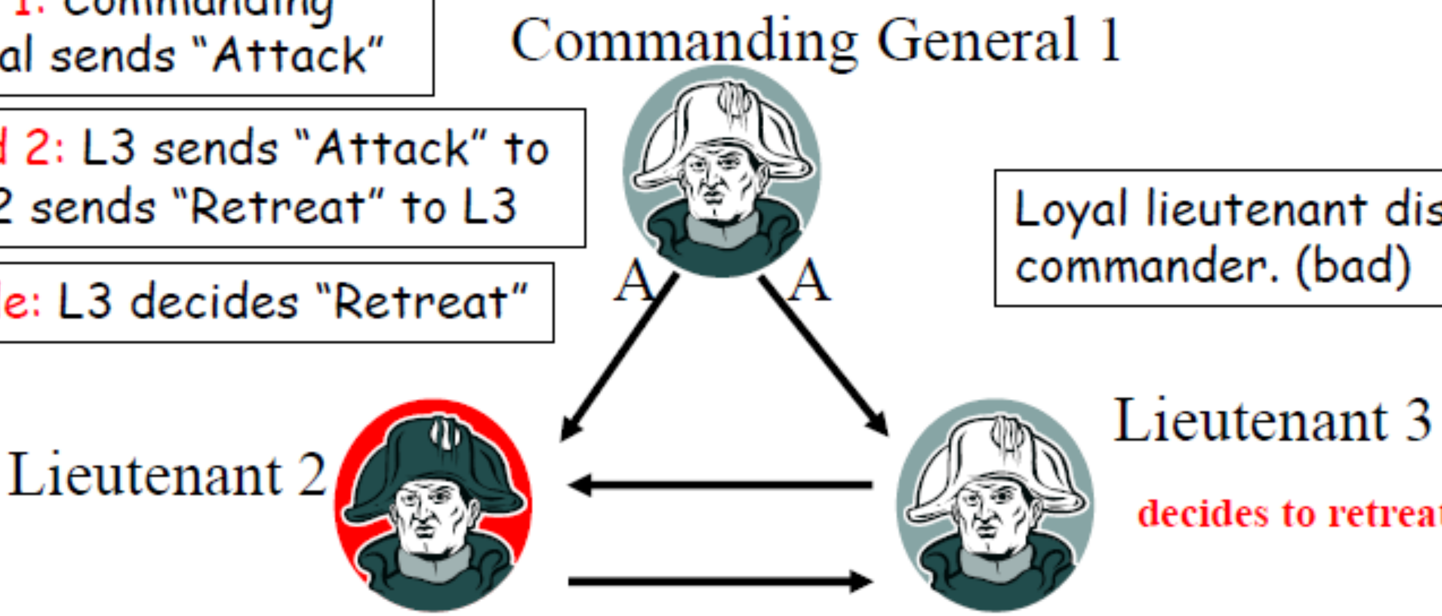
Case 2b

- Traitor lieutenant tries to foil consensus by <u>lying about order</u> sent by general

Round 1: Commanding General sends “Attack”

Round 2: L3 sends “Attack” to L2; L2 sends “Retreat” to L3

Decide: L3 decides “Retreat”



Loyal lieutenant disobeys commander. (bad)

Acknowledgement: Class lectures of A.D Brown of U*of*T

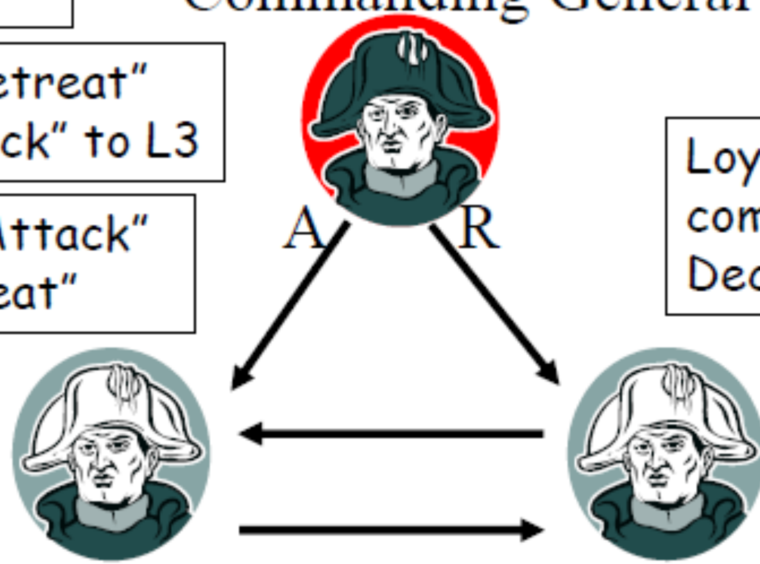
Case 3

- Traitor General tries to foil consensus by <u>sending different orders</u> to loyal lieutenants

Round 1: General sends "Attack" to L2 and "Retreat" to L3

Round 2: L3 sends "Retreat" to L2; L2 sends "Attack" to L3

Decide: L2 decides "Attack" and L3 decides "Retreat"



Loyal lieutenants obey commander. (good)
Decide differently (bad)

Acknowledgement: Class lectures of A.D Brown of UofT

Oral Message Algorithm (LSP)

Oral Message algorithm, OM(m) consists of *m*+1 "phases"

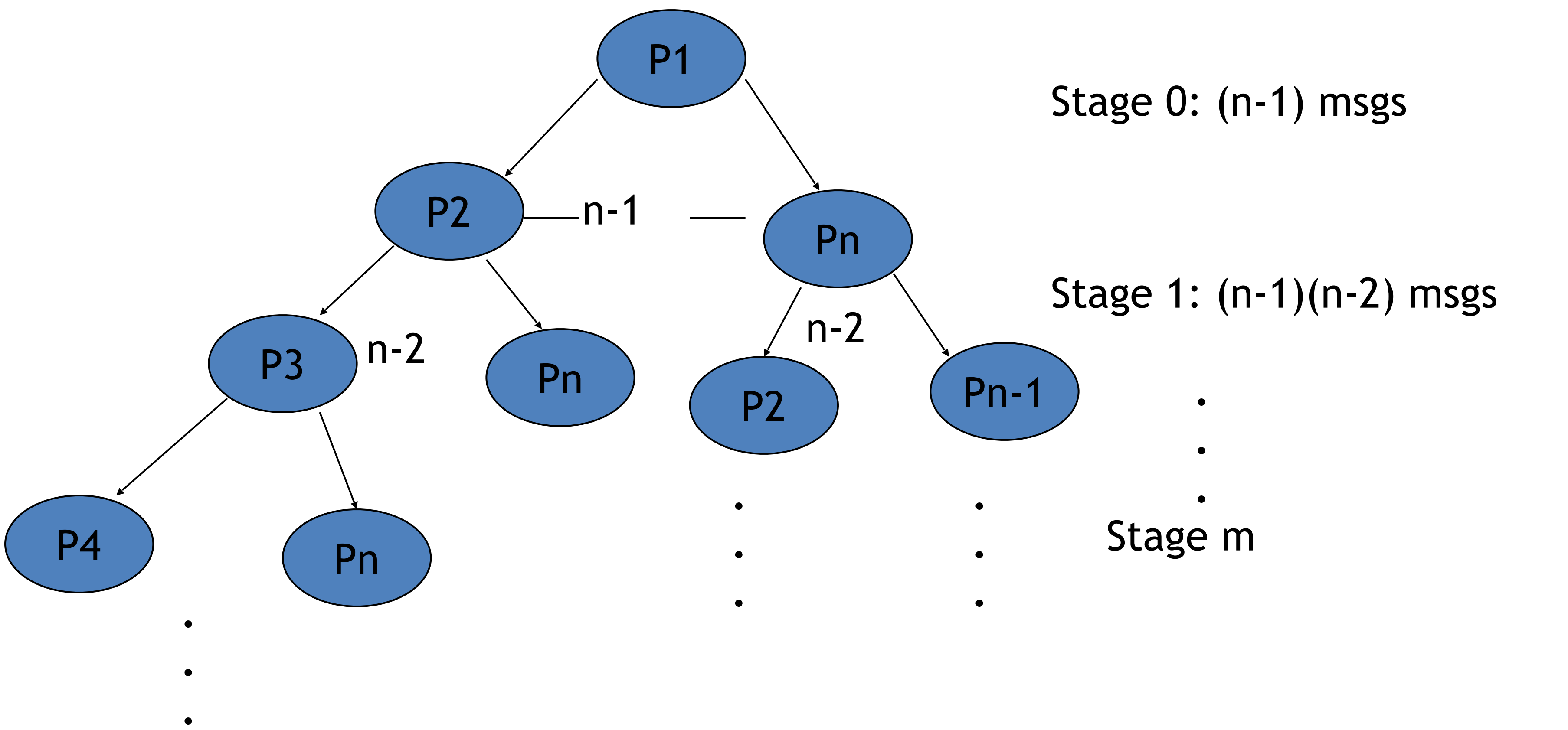
Algorithm OM(0) is the "base case" (no faults)

1) Commander sends value to every lieutenant
2) Each lieutenant uses value received from commander, or default "retreat" if no value was received

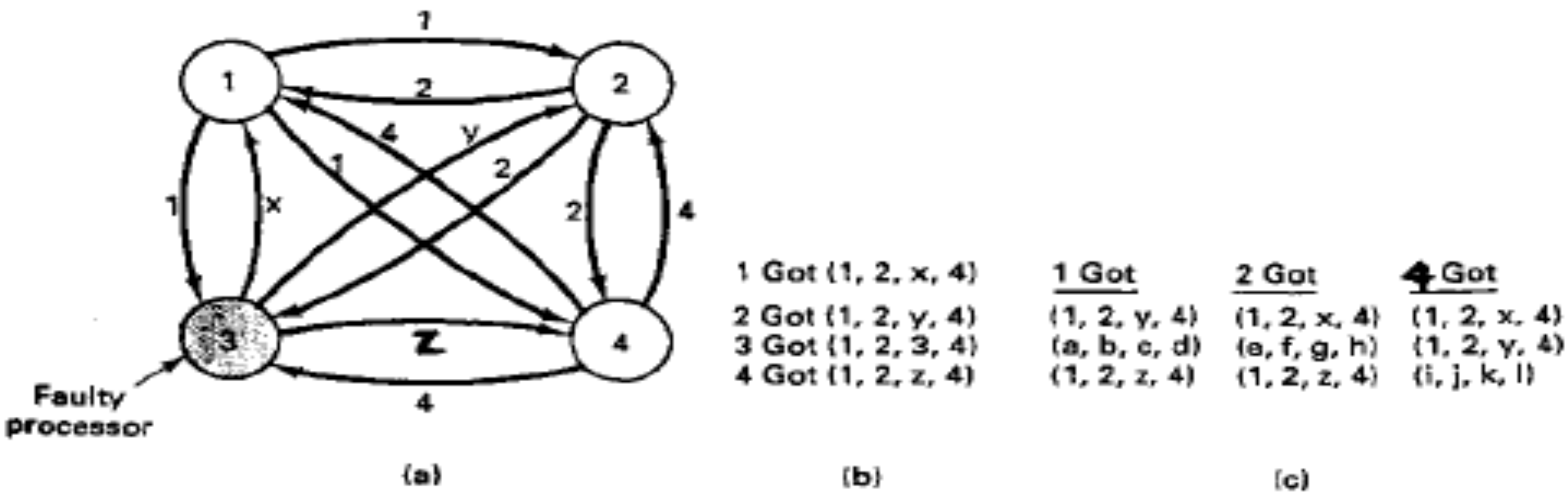
Recursive algorithm OM(m) handles up to m faults

1) Commander sends value to every lieutenant
2) For each lieutenant *i*, let v_i be the value *i* received from commander, or "retreat" if no value was received. Lieutenant *i* acts as commander and runs Alg OM(m-1) to send v_i to each of the *n-2* other lieutenants
3) For each *i*, and each *j* not equal to *i*, let v_j be the value Lieutenant *i* received from Lieutenant *j* in step (2) (using Alg OM(m-1)), or else "retreat" if no such value was received. Lieutenant *i* uses the value *majority*$(v_1, \ldots, v_{n-1})$ *to compute the agreed upon value.*

Stages in Oral message algo^{m}



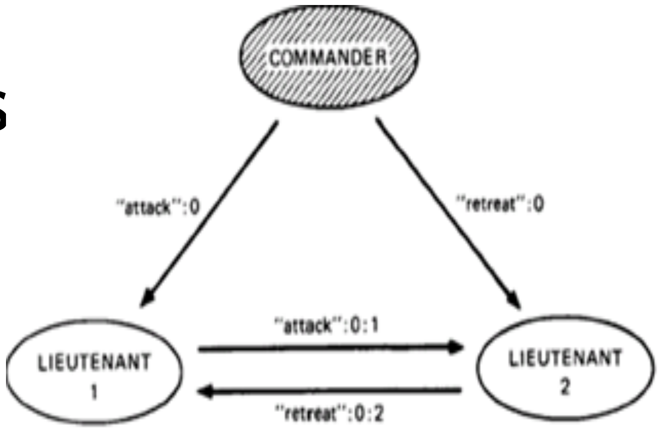
Interactive Consistency (IC)



The Byzantine generals problem for 3 loyal generals and 1 traitor. (a) The generals announce their troop strengths (in units of 1K). (b) The vectors that each general assembles based on (a). (c) The vectors that each general receives in step 2.

Solution with signed messages

- We can cope with any number of traitors
- Prevent traitors lie about the commander's order
- Messages are signed by commander
- The sign can be verified by all loyal lieutenants
- All loyals receive the same set of commands eventually
- If the commander is loyal, it works



Applications of BA

- Building fault tolerant distributed services
  - Hardware Clock Synchronization in presence of faulty nodes
  - Distributed commit in databases

BGP in Distributed systems: Application of BA

- Some misbehave
  - HW Fault, SW bug, Security attack, Misconfiguration

- Goals
  - All correct nodes share the same global info.
  - Ensure that N corrupted nodes can not change the shared global information

innovate achieve lead