

## The Byzantine Generals Problem

Phil Gibbons

15-712 F15

Lecture 23

## Today's Reminders

- No class Wednesday. No office hours Wednesday.

2

## The Byzantine Generals Problem [TOPLAS 1982]

- **Leslie Lamport** (MSR)
  - Turing Award, NAE
- **Robert Shostak** (Vocera Communications)
  - Founder, CTO
- **Marshall Pease** (SRI)



Based on "Reaching Agreement in the Presence of Faults"  
[JACM 1980]

- Won Dijkstra Prize in 2005

3

## Lamport's Comments

"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...I believed that the problem introduced in [41] was very important and deserved the attention of computer scientists.

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.**"

[41] Pease, Shostak, Lamport, "Reaching Agreement in the Presence of Faults," JACM 1980

4

## Lamport's Comments

...I wanted to assign the generals a nationality that would not offend any readers. At the time, Albania was a completely closed society, and I felt it unlikely that there would be any Albanians around to object, so the original title of this paper was *The Albanian Generals Problem*. Jack Goldberg was smart enough to realize that there were Albanians in the world outside Albania, and Albania might not always be a black hole, so he suggested that I find another name. The obviously more appropriate Byzantine generals then occurred to me.

The main reason for writing this paper was to assign the new name to the problem. But a new paper needed new results as well..."

5

## Lamport's Comments on "Reaching Agreement in the Presence of Faults"

"Before this paper, it was generally assumed that a three-processor system could tolerate one faulty processor. This paper shows that "Byzantine" faults, in which a faulty processor sends inconsistent information to the other processors, can defeat any traditional three-processor algorithm. In general,  $3n+1$  processors are needed to tolerate  $n$  faults. However, if digital signatures are used,  $2n+1$  processors are enough. This paper introduced the problem of handling Byzantine faults. I think it also contains the first precise statement of the consensus problem.

...My other contribution to this paper was getting it written. Writing is hard work, and without the threat of perishing, researchers outside academia generally do less publishing than their colleagues at universities. I wrote an initial draft, which displeased Shostak so much that he completely rewrote it to produce the final version."

6

## Byzantine Generals

- Generals communicate only by messenger

- Some of the generals may be traitors

- Goals:

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

- Approach:

- Every loyal general obtains the same values  $v_1 \dots v_n$  & uses same method of combining values into plan of action
- If loyal general  $i$  sends  $v_i$ , then all loyal generals obtain  $v_i$



7

## Byzantine Generals Problem

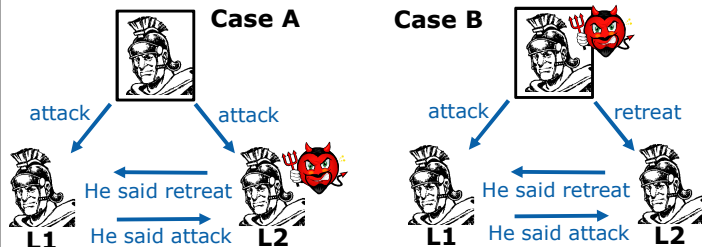
**A commanding general must send an order to his  $n - 1$  lieutenant generals such that**

- **IC1:** All loyal lieutenants obey the same order
- **IC2:** If the commanding general is loyal, then every loyal lieutenant obeys the order he sends

**(Can solve original problem by having each general act as commanding general in order to send its value.)**

8

### 3 Generals Can't Tolerate 1 Traitor



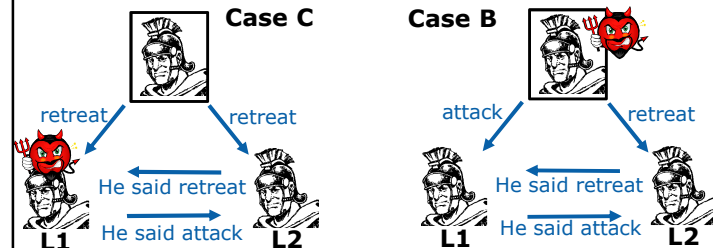
By IC2, L1 must attack

Looks the same to L1,  
so L1 will attack

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

9

### 3 Generals Can't Tolerate 1 Traitor



Symmetric Case for L2:  
By IC2, L2 must retreat

Looks the same to L2,  
so L2 will retreat

Thus, in Case B, L1 will attack & L2 will retreat,  
violating IC1 (Full proof in [41])

10

### Generalization to $m$ Traitors

- **No solution with  $< 3m + 1$  generals can tolerate  $m$  traitors**
  - If were to exist, solution could be used to solve 3-general/1-traitor problem, a contradiction

11

### What About Approximate Agreement?

- **Variant: Agree on approximate time of attack**
  - **IC1'**: All loyal lieutenants attack within 10 minutes of one another
  - **IC2'**: If the commander is loyal, every loyal lieutenant attacks with 10 minutes of the time given in the commander's order
- **If were to exist, solution could be used to solve BGP**
  - Attack: use attack time 1:00; Retreat: use attack time 2:00
  - Lieutenant: if order time  $\leq 1:10$  then attack;  $\geq 1:50$  retreat
  - Else if other lieutenant reached a decision, make same decision; Otherwise Retreat.
  - If commander is loyal: IC2' implies IC2, and IC2 implies IC1
  - If commander is traitor: lieutenants are loyal. By IC1', decide the same in step (1) or in step (2), implying IC1

12

## Oral Messages

**A1: Every 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**

- A1 & A2 prevent a traitor from interfering with communication between two other generals
- A3 foils a traitor who doesn't send messages (Retreat is the default order if commander fails to send an order)

If majority of values  $v_i$  equal  $v$ , then  $\text{majority}(v_1, \dots, v_n) = v$   
Else, return RETREAT (alternative: return median)

13

## OM( $m$ ) for $\geq 3m + 1$ Generals

Algorithm OM(0).

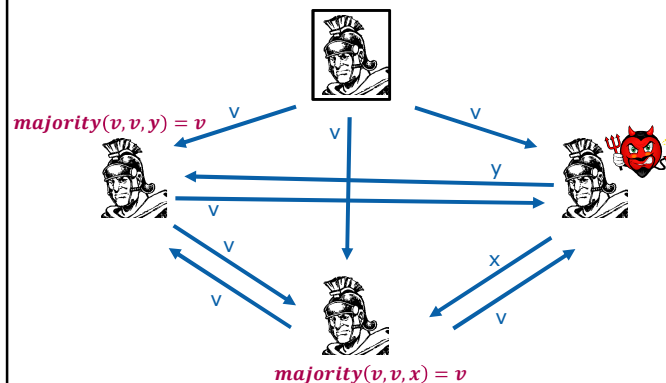
- (1) The commander sends his value to every lieutenant.
- (2) Each lieutenant uses the value he receives from the commander, or uses the value RETREAT if he receives no value.

Algorithm OM( $m$ ),  $m > 0$ .

- (1) The commander sends his value to every lieutenant.
- (2) For each  $i$ , let  $v_i$  be the value Lieutenant  $i$  receives from the commander, or else be RETREAT if he receives no value. 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) For each  $i$ , and each  $j \neq i$ , let  $v_j$  be the value Lieutenant  $i$  received from Lieutenant  $j$  in step (2) (using Algorithm OM( $m - 1$ )), or else RETREAT if he received no such value. Lieutenant  $i$  uses the value  $\text{majority}(v_1, \dots, v_{n-1})$ .

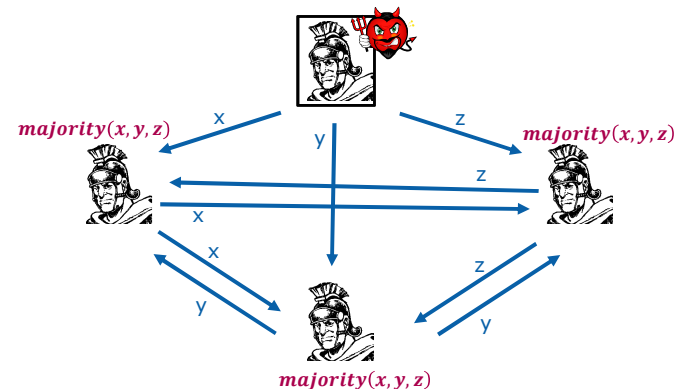
14

## OM(1) with Loyal Commander



15

## OM(1) with Traitor Commander



16

## Signed Messages

**A4: Loyal general's signature cannot be forged & any content alteration of his signed message can be detected. Anyone can verify the authenticity of a general's signature**

$\text{choice}(\text{singleton set } v) = v$ ;  $\text{choice}(\text{empty set}) = \text{RETREAT}$   
(e.g. choice returns median)

- $v:j:i$  denotes the value  $v$  signed by  $j$ , and then that value  $v:j$  signed by  $i$

17

## $\text{SM}(m)$ for $\geq m + 2$ Generals

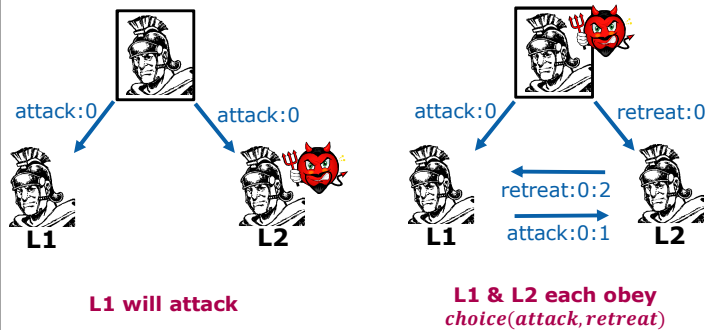
Algorithm  $\text{SM}(m)$ .

Initially  $V_i = \emptyset$ .

- (1) The commander signs and sends his value to every lieutenant.
- (2) For each  $i$ :
  - (A) If Lieutenant  $i$  receives a message of the form  $v:0$  from the commander and he has not yet received any order, then
    - (i) he lets  $V_i$  equal  $\{v\}$ ;
    - (ii) he sends the message  $v:0:i$  to every other lieutenant.
  - (B) If Lieutenant  $i$  receives a message of the form  $v:0:j_1:\dots:j_k$  and  $v$  is not in the set  $V_i$ , then
    - (i) he adds  $v$  to  $V_i$ ;
    - (ii) if  $k < m$ , then he sends the message  $v:0:j_1:\dots:j_k:i$  to every lieutenant other than  $j_1, \dots, j_k$ .
- (3) For each  $i$ : When Lieutenant  $i$  will receive no more messages, he obeys the order  $\text{choice}(V_i)$ .

18

## $\text{SM}(1)$ for 3 Generals



19

## Missing Communication Paths

- Can only send/receive messages to/from neighbors in the communication graph
- If the graph of loyal generals is connected, then  $\text{SM}(n-2)$  solves the Byzantine Generals Problem for  $n$  generals

20

## Message Guarantees in Practice

### A1: Every message that is sent is delivered correctly

- Communication failure is viewed as faulty sender

### A2: The receiver of a message knows who sent it

- Faulty processor cannot impersonate nonfaulty one
- Communicate over fixed lines or consider faulty network nodes

### A3: The absence of a message can be detected

- Use time-outs based on clocks synchronized within a known maximum error

### A4: Loyal general's signature cannot be forged & any content alteration of his signed message can be detected.

Anyone can verify the authenticity of a general's signature

- Standard crypto (simpler if only random malfunction case)

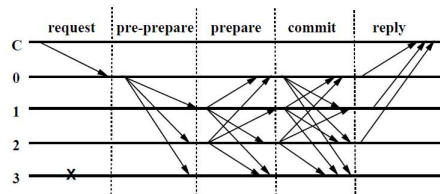
21

## Practical Byzantine Fault Tolerance [Miguel Castro & Barbara Liskov, OSDI'99]

- Asynchronous distributed system, with message failures
- Standard setup for state machine replication
- Adversary cannot delay correct node indefinitely (Needed for liveness, not safety)
- Sign hashed-digest of msg, followed by msg in plaintext
- Even with signed msgs, need  $3m+1$ , due to asynchrony

22

## Practical Byzantine Fault Tolerance



- Client waits for  $m+1$  replies from different replicas with same answer
- If primary is faulty, select new primary
- Many optimizations to reduce overhead
- BFT NFS service with 3% overhead on Andrew benchmark

23