CSE 486/586 Distributed Systems
Byzantine Fault Tolerance

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Byzantine Fault Tolerance

- Fault categories
  – Benign: failures we've been talking about
  – Byzantine: arbitrary failures
- Benign
  – Fail-stop & crash: process halted
  – Omission: msg loss, send-omission, receive-omission
  – All entities still follow the protocol
- Byzantine
  – A broader category than benign failures
  – Process or channel exhibits arbitrary behavior.
  – May deviate from the protocol
  – Processes can crash, messages can be lost, etc.
  – Can be malicious (attacks, software bugs, etc.)

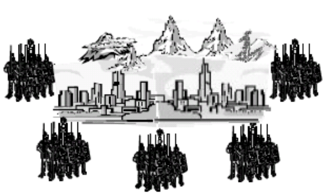
Byzantine Fault Tolerance

- Can we achieve consensus with *f Byzantine faults*?
  – But we're not bypassing the impossibility result (e.g., we still need to mask benign failures.)
- Result: with *f faulty nodes*, we need *3f + 1* nodes to tolerate their Byzantine behavior.
  – Fundamental limitation
  – Today's goal is to understand this limitation.
- How about Paxos (that tolerates benign failures)?
  – With *f* faulty nodes, we need *2f + 1* (i.e., we need a correct majority.)
  – Having *f* faulty nodes means that as long as *f* + 1 nodes are reachable, Paxos can guarantee an agreement.
  – This is the known lower bound for consensus with non-Byzantine failures.

"Byzantine"

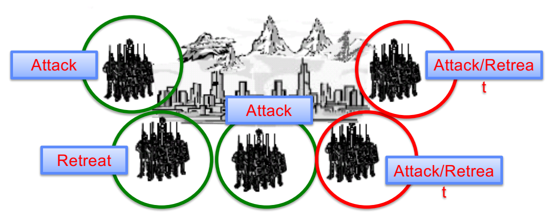
- Leslie Lamport (again!) defined the problem & presented the result.
- *"I have long felt that, because it was posed as a cute problem about philosophers seated around a table, Dijkstra's dining philosopher's problem received much more attention than it deserves."*
- *"At the time, Albania was a completely 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."*
- *"…The obviously more appropriate Byzantine generals then occurred to me."*

Introducing the Byzantine Generals



- Imagine several divisions of the Byzantine army camped outside of a city
- Each division has a general.
- The generals can only communicate by a messenger.

Introducing the Byzantine Generals

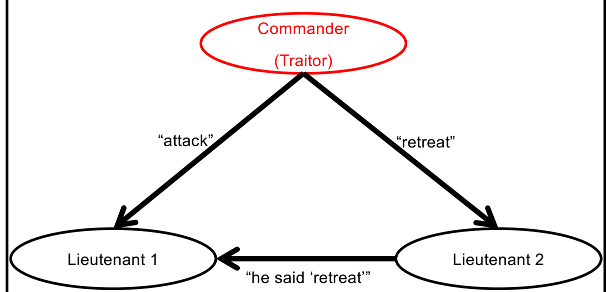


- They must decide on a common plan of action (consensus).
- But, *some of the generals can be traitors*.
- Quick example to demonstrate the problem:
  – One commander and two lieutenants
  – With one traitor, can non-traitors decide on a common plan?

Understanding the Problem

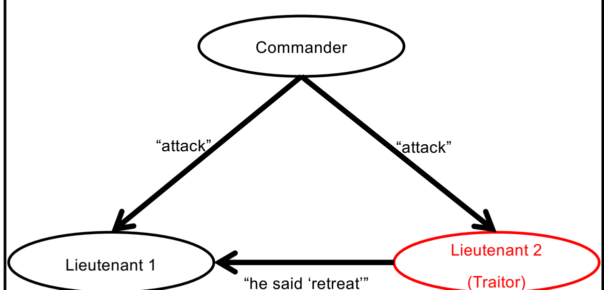
- Setup
  - One commander & two lieutenants
  - One can be a traitor
  - $f = 1$ and $n = 3$ ($2f + 1$)
- Protocol
  - Commander sends a command (either attack or retreat) to the two lieutenants.
  - Each lieutenant forwards the command to the other lieutenant in case messages get lost.
- Goal
  - Deciding on the same plan of action (either attack or retreat)

Understanding the Problem



- Compare this to the next scenario

Understanding the Problem



- For lieutenant 1, this looks exactly the same as the previous scenario.

Understanding the Problem

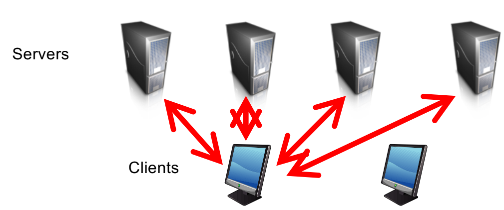
- In the example, one traitor ($f == 1$) makes it impossible to reach consensus with three generals ($2f + 1$ generals).
- Or more generally, when f nodes can behave arbitrarily (Byzantine), $2f + 1$ nodes are not enough to tolerate it.
  - This is unlike Paxos (reaching consensus while tolerating non-Byzantine failures).

CSE 486/586 Administrivia

- PA4 deadline: 5/10
- Final exam: 5/17 @ 11:45 am – 2:45 pm in Knox 109
  - Includes everything
  - True/false questions & multi-choice questions
  - Cheat sheet allowed (1-page, letter-sized, front-and-back)
  - No restroom use
- Survey & course evaluation
  - Survey: https://forms.gle/eg1wHN2G8S6GVz3e9
  - Course evaluation: https://www.smartevals.com/login.aspx?s=buffalo
- Incentive when both have 80% or more participation
  - Currently about 50% for both
- No recitation this week; replaced with office hours

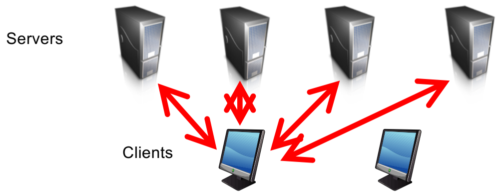
More Practical Setting

- Replicated Web servers
  - Multiple servers running the same state machine.
  - For example, a client asks a question and each server replies with an answer (yes/no).
  - The client determines what the correct answer is based on the replies.



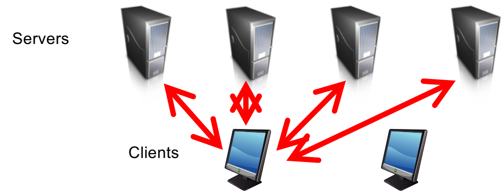
More Practical Setting

- f Byzantine failures
  - At any point of time, there can be up to f failures.
- Ambiguity (many possibilities) of a failure
  - A crashed process, a message loss, malicious behavior (e.g., a lie), etc., but a client cannot tell which one it is.
  - But in total, the maximum # of failures is bounded by f.



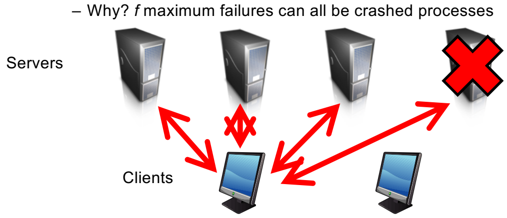
BFT Question

- Given f, how many nodes do we need to tolerate f Byzantine failures?
  - f failures can be any mix of malicious servers, crashed servers, message losses, etc.
  - Malicious servers can do anything, e.g., they can lie (if yes, say no, if no, say yes).



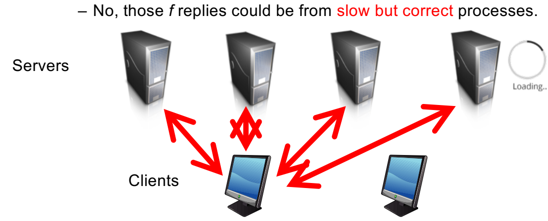
Intuition for the Result

- Let's say we have n servers, and maximum f Byzantine failures.
- What is the minimum # of replies that you are *always* guaranteed to get?
  - $n - f$
  - Why? f maximum failures can all be crashed processes



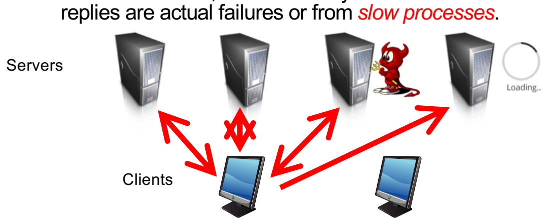
Intuition for the Result

- The problem is that we're unsure what those f failures are. So we have to think about many possibilities.
- Upon receiving $n – f$ replies (guaranteed), are we really sure that f replies will never come?
  - No, those f replies could be from slow but correct processes.



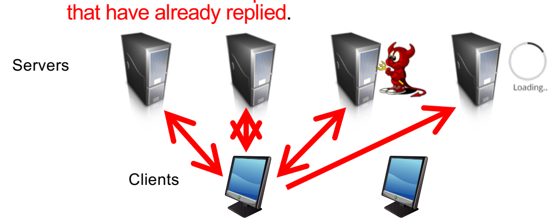
Intuition for the Result

- Let's put it together. We have two possibilities.
- With $n – f$ replies, there is no guarantee that f replies will come, i.e., the client needs to determine what the correct answer is when it has $n – f$ replies.
- At the same time, there's no way to tell if those f replies are actual failures or from *slow processes*.



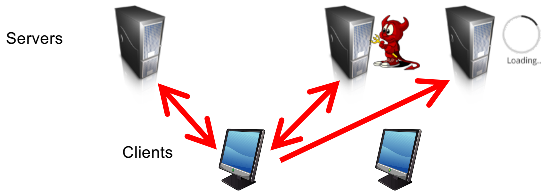
Intuition for the Result

- If those f replies are from *slow processes,* then they are still correct. They don't count towards f failures.
- This means that out of $n – f$ replies, there can still be f replies from f Byzantine nodes.
- This leaves us with f processes that can be malicious that have already replied.



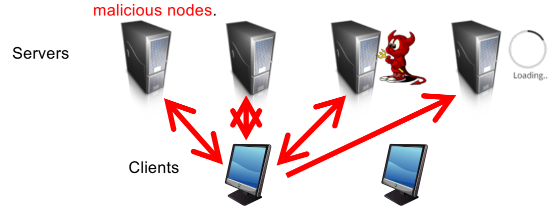
Intuition for the Result

- Then the question is: out of $n - f$ replies and possible f malicious replies contained among them, how can we make sure that we can always determine the correct answer?
  - If we make sure that $n - f$ replies always contain more replies from honest nodes than Byzantine nodes, we're safe.



Intuition for the Result

- Answer: we make sure that we always get $f + 1$ replies from honest nodes, one more than the number of potentially-malicious nodes, f.
  - We set $n == 3f + 1$
  - When we get $n - f$ relies, it is $2f + 1$ replies. At least $f + 1$ *replies from honest nodes*, and at most f replies from malicious nodes.



Write/Read Example

- One client writes to X.
- A malicious node omits it.
- Another client reads X.
- It can still get the latest write.

Summary

- Byzantine generals problem
  - They must decide on a common plan of action.
  - But, some of the generals can be traitors.
- Requirements
  - All loyal generals decide upon the same plan of action (e.g., attack or retreat).
  - A small number of traitors cannot cause the loyal generals to adopt a bad plan.
- Impossibility result
  - In general, with less than $3f + 1$ nodes, we cannot tolerate f faulty nodes.

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).