Byzantine Agreement

CSE 486: Distributed Systems

Ethan Blanton

Department of Computer Science and Engineering

University at Buffalo

Byzantine Failures

We previously mentioned Byzantine failures[1] briefly.

This is when a process displays different behavior to different observers.

E.g., perhaps process p_1:

- Says "my value is 0" to process p_2
- Says "my value is 1" to process p_3
- Fails to respond entirely to process p_4

This is often harder to account for than simpler failures.

[1] Sometimes "Byzantine faults"

Etymology

The term “Byzantine” was coined by Lamport *et al.* [1, 2].

> *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 … Reaching Agreement in the Presence of Faults [3]] 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.*

He has used this tactic several times since.

Failures

All failures we have previously considered were consistent.

A process is either failed, or it is not.

A failed process may give the wrong value, but it does so consistently.

Most of our failures have been fail-stop.

Byzantine Failure

With Byzantine failure, a process may appear differently:

- To different processes
- At different times

It cannot (necessarily) be detected by a failure detector.

It could be caused by (for example):

- A bad bit in memory that reads inconsistently
- A program bug
- A malicious process

Byzantine Adversaries

A Byzantine failure may be a malicious adversary.

In this case, the adversary can give any answer to any process.

It could send the worst possible response in every case!

A Byzantine attacker can be very hard to defeat.

Byzantine Generals

The Byzantine Generals problem is set up as follows:

- Several armies are besieging a city, each led by a general.
- If enough of them attack at once, they will be victorious.
- If too few of them attack, they will fail.
- They can send reliable and timely messages to each other.
- Some of the generals might be traitors.

How, and under what circumstances, can they agree to attack?

The Problem

This is a consensus problem.

Assume that one general is the commander.

The other generals are lieutenants.

We want these properties:

- All loyal lieutenants execute the same order.
- If the commander is loyal,
  all loyal lieutenants follow the commander's orders.

The Model

The messaging model is synchronous.

Messages cannot be forged:

- Generals know if a message does not arrive
- Generals know who sent a message
- The message is received as sent

Loyal generals always behave correctly.

Traitorous generals can lie, and can collude.

Four Generals

Assume there are four generals, with one traitor.

There is a simple solution to this problem.

It is closely related to synchronous consensus with $f = 1$.

It proceeds in two rounds.

The Rounds

Round 1:

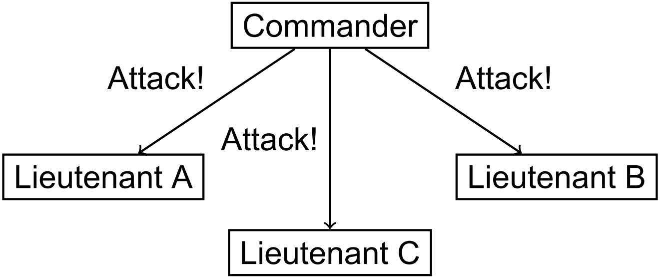
- The commander tells every lieutenant their orders.

Round 2:

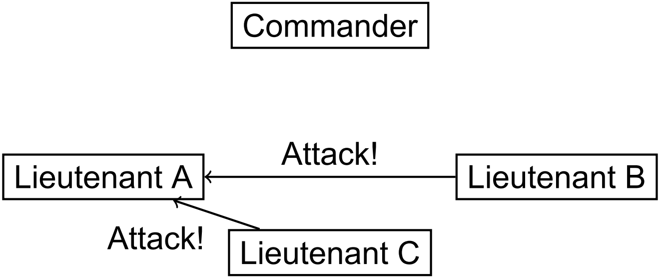
- Every lieutenant tells every other lieutenant their orders.

After round 2, every lieutenant takes the plurality of orders.

Example


Commander
Attack!
Attack!
Attack!
Lieutenant A
Lieutenant B
Lieutenant C

Example


Commander
Lieutenant A
Attack!
Lieutenant B
Attack!
Lieutenant C

Introducing …a Traitor

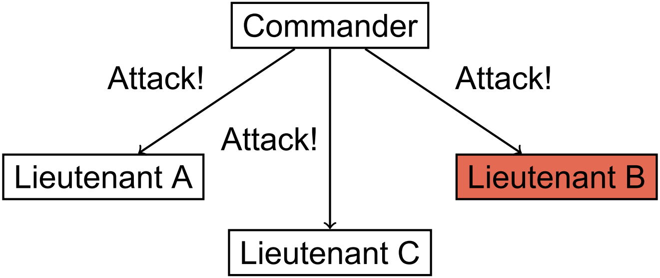
What if one general is a traitor?

There are two cases:

- One lieutenant is a traitor
- The commander is a traitor

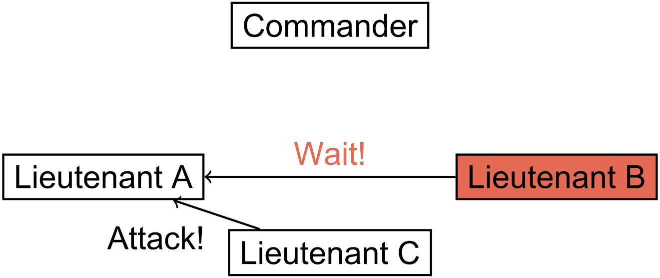
Let’s look at each case.

Traitorous Lieutenant



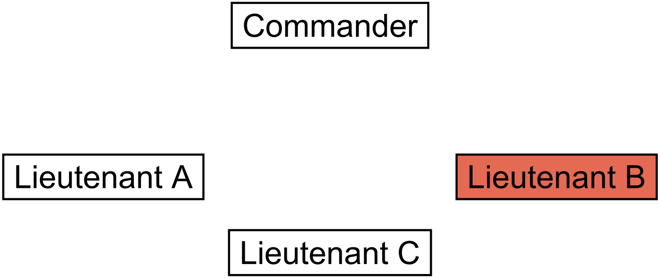
The general sends messages as in the first example.

Traitorous Lieutenant



Lieutenant B is a traitor, and changes the message.

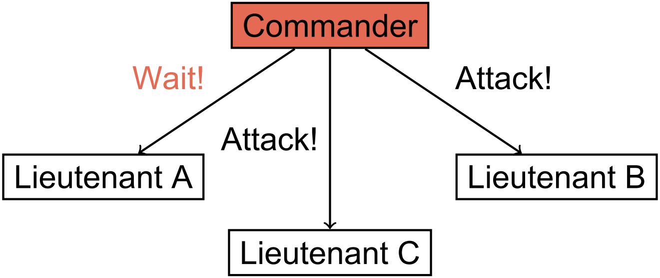
Traitorous Lieutenant



Lieutenant A received: { Attack, Attack, Wait }
Lieutenant A attacks!

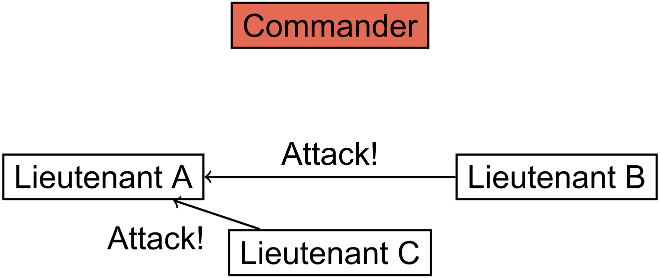
(It is super effective!)

Traitorous Commander



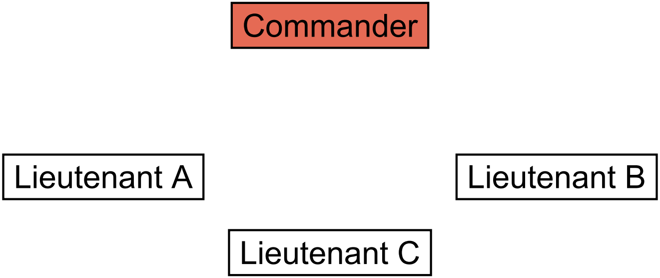
The general sends mixed messages.

Traitorous Commander



Lieutenants B and C repeat what they heard faithfully.

Traitorous Commander



Lieutenant A received: { Wait, Attack, Attack }
Lieutenant A attacks along with Lieutenants B and C.

N Generals

To extend this to *n* generals with no more than *m* traitors:

Round 1 remains the same.

There are *m* additional rounds with particular rules.

Again, this is like synchronous consensus with *f* failures!

The Magic of 1/3

Assume that there are n generals, and m are traitors.

Under this model, $2m + 1$ generals must be loyal.

If fewer than $2m + 1$ generals are loyal, loyal generals may not all take the same action.

Thus, strictly more than 2/3 of the generals must be loyal!

Interestingly, the loyalty of the commander doesn't matter.

Three Generals

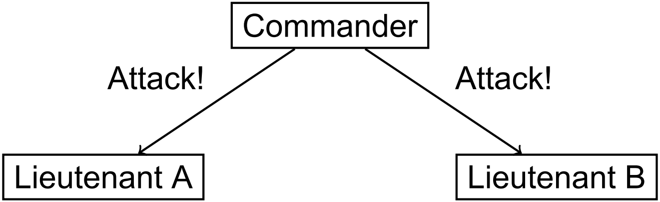
Consider three generals with one traitor.

It is easy to show that agreement is impossible.

We have the same two cases to consider:

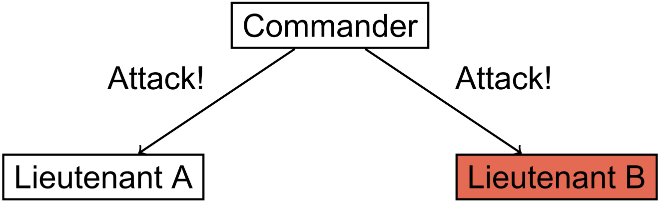
- One of the lieutenants is a traitor
- The commanding general is a traitor

A Loyal Group



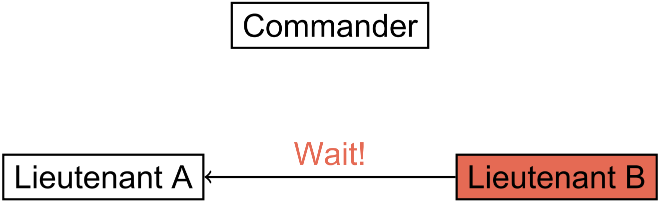
A Loyal Group

A Traitorous Lieutenant



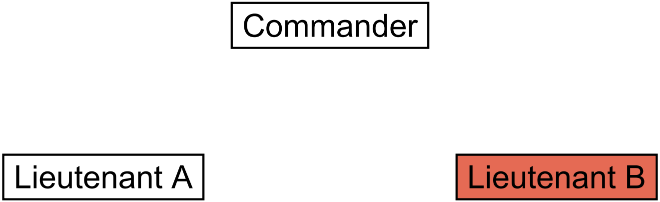
Again, the general proceeds as before.

A Traitorous Lieutenant



Lieutenant B changes the orders.

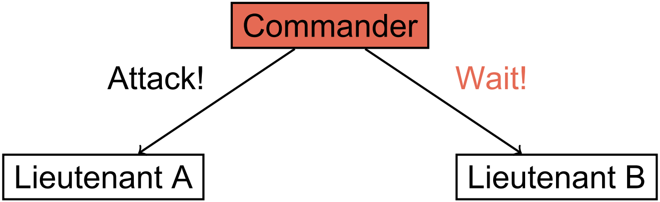
A Traitorous Lieutenant



Lieutenant A received: { Attack, Wait }
Now what?

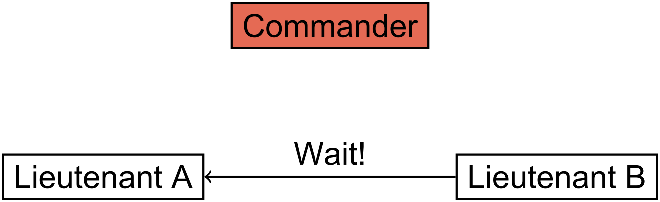
Why can't Lieutenant A simply believe the commander?

A Traitorous Commander



The general sends a different message to Lieutenant B.

A Traitorous Commander



Lieutenant B repeats in good faith.

A Traitorous Commander

Lieutenant A received: { Attack, Wait }

This is exactly the same as the traitorous Lieutenant B!

Generalizing to $3m + 1$

This can be generalized[2] to $3m$ generals.

By contradiction:

1. Assume a solution for $3m$ or fewer generals
2. Divide the loyal generals into two groups, roughly equally
2. Cause the traitorous generals to work in concert
2. Now you have three simulated generals
3. ???
4. Profit by solving the three generals problem!

[2] See what I did there?

Summary

- Byzantine failures present differently in different circumstances
- Storytelling gets you published
- Consensus can be reached even with Byzantine failure (in a synchronous system)
- More than 2/3 of processes must be honest to achieve this

References I

Optional Readings

[1] Leslie Lamport. *The Writings of Leslie Lamport: The Byzantine Generals Problem*. URL: http://lamport.azurewebsites.net/pubs/pubs.html#byz.

[2] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Transactions on Programming Languages and Systems* 4.3 (July 1982), pp. 382–401. DOI: 10.1145/357172.357176. URL: http://lamport.azurewebsites.net/pubs/byz.pdf.

References II

[3] Marshall Pease, Robert Shostak, and Leslie Lamport. "Reaching Agreement in the Presence of Faults". In: 27.2 (Apr. 1980), pp. 228–234. DOI: 10.1145/322186.322188. URL: http://lamport.azurewebsites.net/pubs/reaching.pdf.

To retrieve a copy of this material, or related materials, see `https://www.cse.buffalo.edu/~eblanton/`.