Lecture 23 — The Byzantine Generals Problem

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

August 13, 2025

Istanbul was Constantinople

Credit: Wikipedia user Cplakidas

Unreliable, Not Malicious

Where this crosses over into the realm of programming is in dealing with the unreliable nature of systems.

It’s not that we think that somewhere out there one of the processes in the system is malicious and trying to make the system fail.

They have bugs, not ill intent.

The Dude Abides

We need a way to deal with the fact that not everyone agrees and sometimes messages get mixed up.

Organization of the Army

There is a General who is giving the orders.

There are also *n* Lieutenants who each are commanding a group of the Empire's troops.

The individual troops just do what they're told so there's no further concern for their actions.

The lieutenants can communicate with one another through messages, and they get their orders from the general.

All of the lieutenants have to work together for their plan to be a success, otherwise there is chaos.

This is Easy, Right?

No problem, you might imagine. The general tells the lieutenants what to do and they do it!

And if one of the lieutenants does the wrong thing, that lieutenant is a traitor.

Right?

Betrayal!

But it's not so simple, because the general can be disloyal too.

Suppose you are a disloyal general.

Your Emperor has commanded you to attack, but you want the attack to fail, but also seem like it's not your fault.

What do you do?

Contradictory Orders It Is

You can issue different orders to different lieutenants, they will be all confused and uncoordinated.

Oh gee darn, I guess the attack didn't go as planned!

And you can blame the lieutenants who are now dead for going too early, or something.

It's not like they can defend themselves now…

Assessing the General's Loyalty

A loyal general sends the same message to all lieutenants and a disloyal one sends different messages to different lieutenants.

It's key to remember that "loyal" really means "functioning" and "disloyal" means "faulty".

It's also worth noting that being in one state or the other is not permanent.

A functioning unit can be damaged, a malfunctioning one can be repaired, or a problem can be transient.

How Much Uncertainty?

One of the key decisions is about how much disloyalty your system can tolerate.

Given enough bad actors, things will go wrong. If everyone is disloyal, utter chaos will result.

But is it enough to tolerate one disloyal participant? Two?

The line will have to be drawn somewhere…

Should I Stay or Should I Go Now?

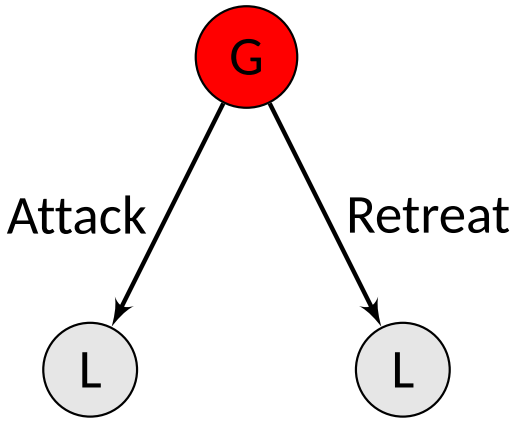
In the examples that we will discuss there are two kinds of command: attack and retreat.

In a real system the options don't have to be quite so binary.

It is possible that our system produces a tie (no matter how many participants or possible actions we have).

This usually necessitates a default action be selected.

Disloyal General, Two Loyal Lieutenants



Red indicates disloyal; grey indicates loyal.

What are lieutenants to do?

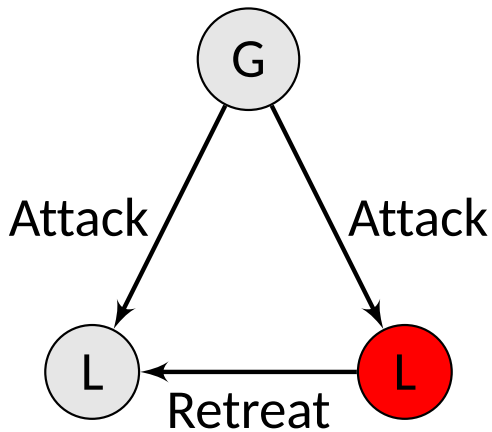
He Said What?

In the imperfect world of Byzantium the lieutenants can try to figure out if they're about to be mismanaged to literal death by communicating with one another.

If they are all on the same page then they can do what they're supposed to do; if they're not, then they will fall back on the default option.

Unfortunately, though, letting the lieutenants communicate is not a complete solution.

Whose Team Are You On



Do the Math

In general if there are d disloyal participants, we will need there to be more than $3d$ participants for the loyal lieutenants to agree on what to do.

If the general is loyal, then at least $2d$ loyal lieutenants are needed to obey the orders.

If the general is disloyal then $2d + 1$ loyal lieutenants are needed so they can come up with a course of action.

One Disloyal Participant

Imagine that there is at most one disloyal participant.

If that's the case then all lieutenants should compare notes and decide on the majority course of action.

Each lieutenant compiles a little table (or array or vector, whatever) of the data received and then decide on what the majority cause of action is.

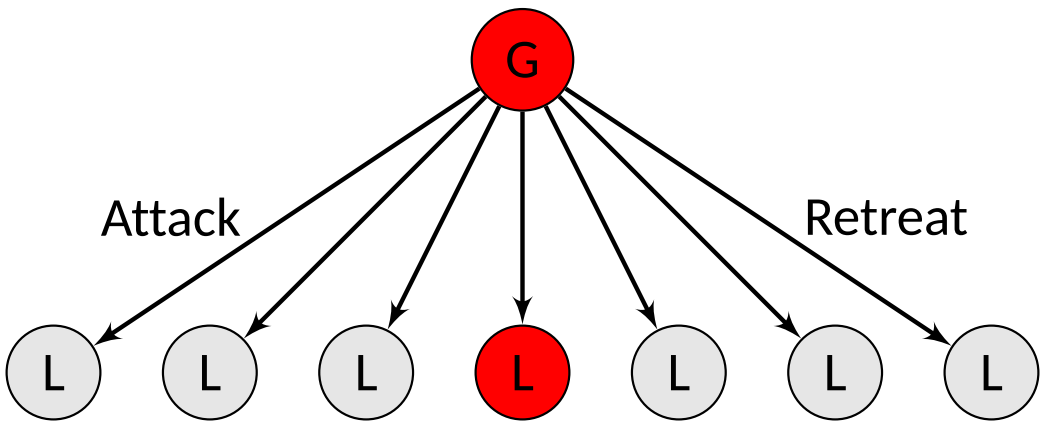
It might be simpler to just count the total number of votes, but it would make it harder to figure out later who the traitor is (if there is one).

Two Disloyal Participants

What if we can have two disloyal participants?

Each lieutenant sending its messages and using a majority-wins vote isn't necessarily going to work here.

It can happen: the general is disloyal and one of the lieutenants is a collaborator.

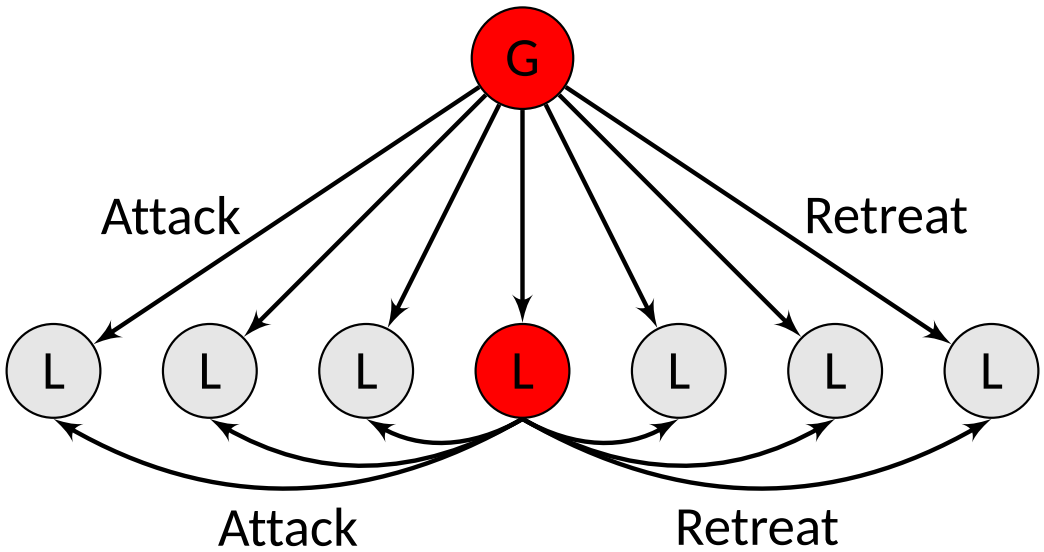


What can the collaborating lieutenant do to make sure that the other lieutenants don't come to an agreement?

Two Disloyal Participants

The lieutenants all send one another their instructions.

The disloyal lieutenant sends one message to half the other participants, and a different message to the other half.



What do we decide?

Uh, This Didn't Go Well

Half the participants think the majority action is attack and half the participants think the majority action is retreat.

Summing up what we heard is not sufficient.

Lieutenants should also talk about what they heard from one another.

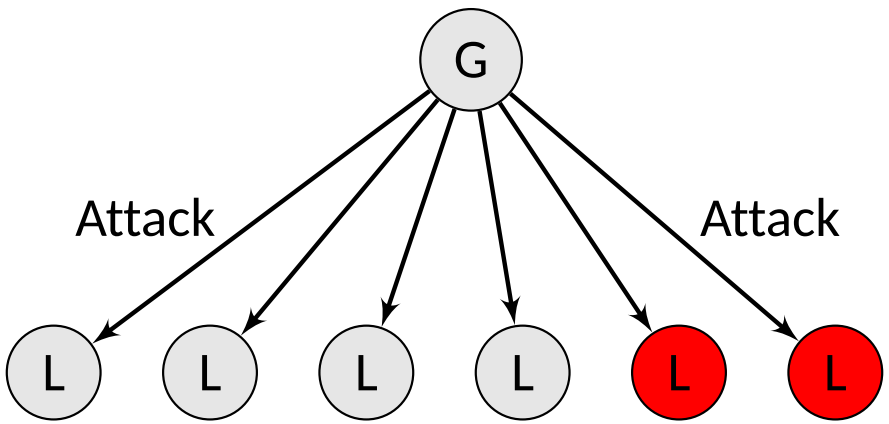
After the general issues instructions, each lieutenant should then communicate with every other to hear what the general said to them.

Then by reviewing this information, they can decide what to do.

Double Treason!

Consider a simple example where the general is loyal and we have two traitorous lieutenants with a total of seven participants.

The general issued an order to attack.



Assemble Your Notes

So each lieutenant constructs the following vector.

The general form is as follows, where v_x is the forwarded order we received from lieutenant x:

General	L1	L2	L3	L4	L5	L6
?	v_1	v_2	v_3	v_4	v_5	v_6

So if we are lieutenant 1, we complete the table as below.

General	L1	L2	L3	L4	L5	L6
?	A	A	A	A	R	R

The value for L1 is what we received from the general, but we get the remaining values from the other lieutenants.

Compare Notes Again

Then the lieutenants just compare notes on this subject as well!

They send to one another their vectors and assemble them into a table.

General	L1	L2	L3	L4	L5	L6
?	$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$v_{1,4}$	$v_{1,5}$	$v_{1,6}$
?	$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$v_{2,4}$	$v_{2,5}$	$v_{2,6}$
?	$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	$v_{3,4}$	$v_{3,5}$	$v_{3,6}$
?	$v_{4,1}$	$v_{4,2}$	$v_{4,3}$	$v_{4,4}$	$v_{4,5}$	$v_{4,6}$
?	$v_{5,1}$	$v_{5,2}$	$v_{5,3}$	$v_{5,4}$	$v_{5,5}$	$v_{5,6}$
?	$v_{6,1}$	$v_{6,2}$	$v_{6,3}$	$v_{6,4}$	$v_{6,5}$	$v_{6,6}$

In the table each value $v_{i,j}$ is interpreted as “Lieutenant i says that Lieutenant j reports the general said v”.

I Agree With Myself

The entries on the diagonal are redundant because of course each lieutenant agrees with itself.

Those “redundant” entries need to be removed from the table.

From the point of view of lieutenant 1, let’s imagine the table looks like this:

General	L1	L2	L3	L4	L5	L6
?		A	A	A	R	R
?	A		A	A	R	R
?	A	A		A	R	R
?	A	A	A		R	R
?	R	R	R	R		R
?	R	R	R	R	R	

Trust In Yourself

Lieutenant 1 does not care what other people THINK they said; so the whole column can be replaced with what was actually said:

General	L1	L2	L3	L4	L5	L6
?		A	A	A	R	R
?	A		A	A	R	R
?	A	A		A	R	R
?	A	A	A		R	R
?	A	R	R	R		R
?	A	R	R	R	R	

Then we'll try to figure out the majority vote for each *j* in the table.

So sum up each column:

General	L1	L2	L3	L4	L5	L6
?	A	A	A	A	R	R

Final Decision

4 lieutenants think the general said attack and 2 think the general said retreat.

The majority wins and the attack proceeds!

Onward to victory, brothers and sisters!

Or Should I Have Used This One?

This might be less historically accurate, though…

Mo' Traitors, Mo' Problems

If there was one less loyal lieutenant, though, we could have a tie here which would result in picking the default value.

If the general ordered something other than the default value (e.g., default is retreat and the order was attack) the general will not be happy about this…

Liar, Liar, Pants on Fire

You may have figured out that since disloyal participants can lie at any step of the equation, we can't rely on their data at all.

That's true!

In the examples in the course we know that two participants are traitors and I've said they are L5 and L6.

Therefore we could replace whatever they say with a question mark in the able rather than any particular answer, because they are liars.

In real life though, *some* message is received – attack or retreat – and it's only later we could identify which participants are the traitors.

We Are Not Amused

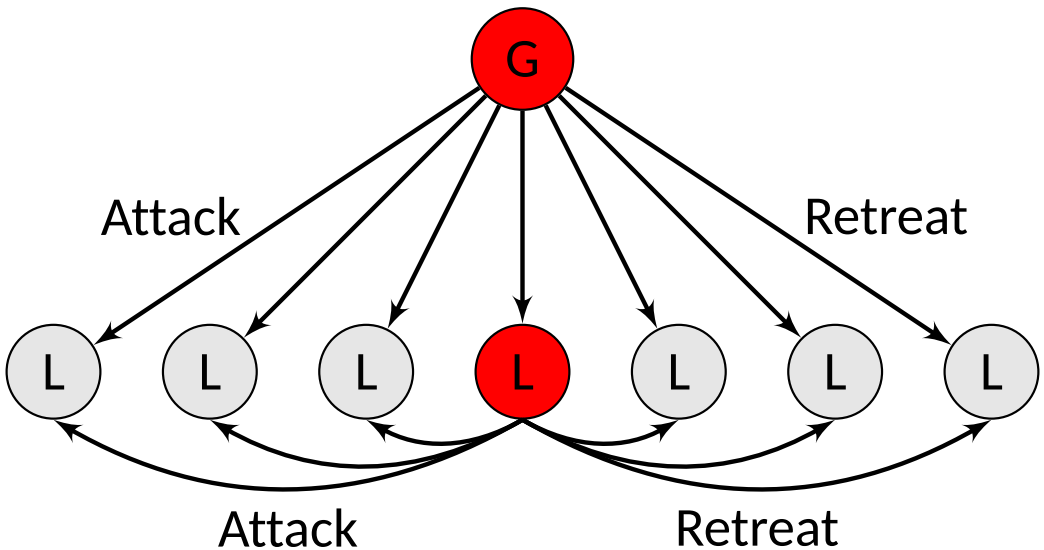
Remember: we aren't actually interested in determining who the traitors are.

The goal is just to reach a decision as a group and carry it out.

(In a real-life scenario, of course, things would be different!)

So, Question Marks

Back to the example of having eight participants.



So, Question Marks

Back to the example of having eight participants.

The value for L1 is what we received from the general, but we get the remaining values from the other lieutenants.

General	L1	L2	L3	L4	L5	L6	L7
?	A	A	A	A	R	R	R

However, if we are lieutenant 7 our vector looks like this instead:

General	L1	L2	L3	L4	L5	L6	L7
?	A	A	A	R	R	R	R

Building the Table

From the point of view of lieutenant 1 the table is formed then as follows:

General	L1	L2	L3	L4	L5	L6	L7
?		A	A	A	R	R	R
?	A		A	A	R	R	R
?	A	A		A	R	R	R
?	?	?	?		?	?	?
?	A	A	A	R		R	R
?	A	A	A	R	R		R
?	A	A	A	R	R	R	

The fourth row is shown as all question marks. Why?

His Majesty the Emperor Says

Lieutenant 4 is the traitor & could also lie about what the other lieutenants said.

Complete the Table and Vector

Then replace the first column.

General	L1	L2	L3	L4	L5	L6	L7
?		A	A	A	R	R	R
?	A		A	A	R	R	R
?	A	A		A	R	R	R
?	A	?	?		?	?	?
?	A	A	A	R		R	R
?	A	A	A	R	R		R
?	A	A	A	R	R	R	

Sum up each column:

General	L1	L2	L3	L4	L5	L6	L7
?	A	A	A	?	R	R	R

It’s A Tie…

Each of the lieutenants figures out that we have a draw and we then choose the default choice.

Well, that’s grim, but it makes sense: the general issued contradictory orders and the collaborating lieutenant did the same.

But now instead of half the lieutenants charging in and dying while the others hang back and watch, everyone takes the same action.

… Even if it’s not the one that helps the Empire the most.

More Traitors!

If there three disloyal participants can exist in the system there's another round of data exchange that needs to take place.

Each lieutenant sends the table formed in the second round to all other lieutenants.

The more disloyalty there could be in the system, the more rounds the process will go on.

Communication grows at n^2 (well, actually worst case dn^2...)

So probably we don't tolerate large amounts of treason.

Trust, But Verify

Part of the difficulty comes from the fact that lieutenants can lie about what the general said.

In the time of Byzantium, wax seals were used (hot wax is poured and then a stamp or ring was used to make an imprint on it).

In modern times messages can be signed using public-key cryptography.

Then participants can check whether the received message was genuine.

An order that does not appear genuine can be disregarded, reducing the ability of disloyal lieutenants to cause confusion.

Applicability

The Byzantine Generals Problem has applicability in all kinds of systems, from space flight to cryptocurrency.

Whenever we have multiple “agents” of some sort that come to their own conclusions about what to do, we can face this issue.

As long as we know this, we can design our system with this in mind, because pretending it’s not going to happen is not a real solution…