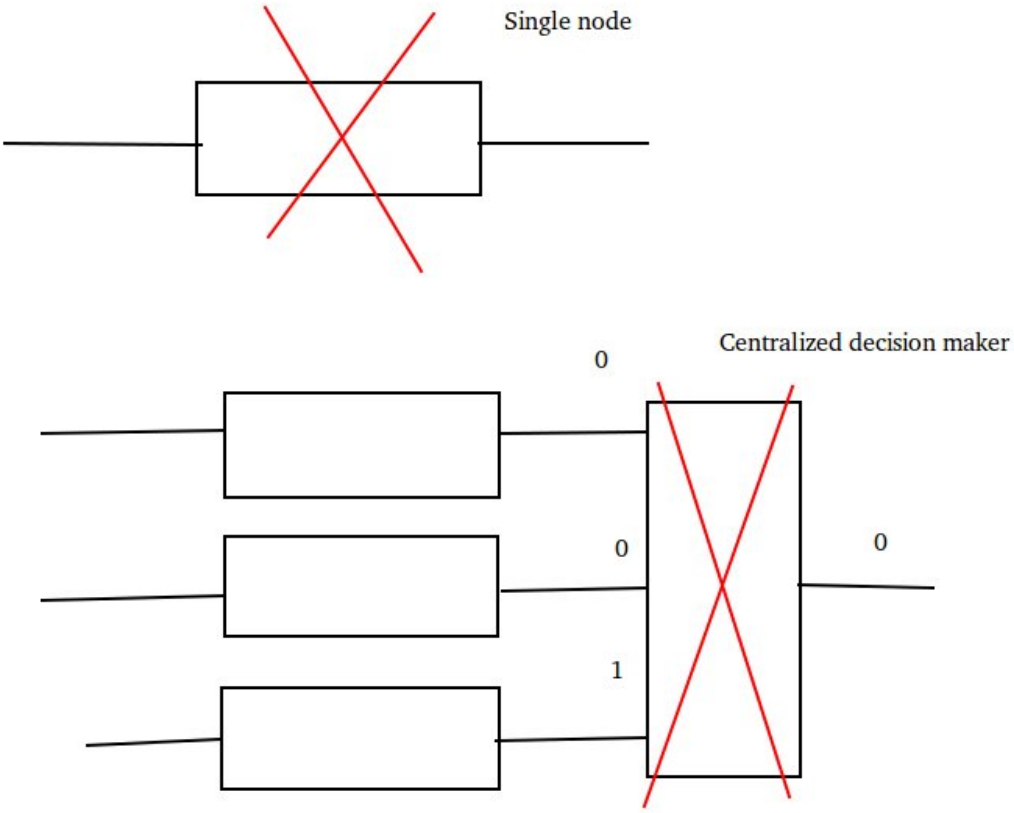
Byzantine Generals Problem (BGP)

Zhuoqun Cheng

Why we need BGP:
Centralized system ->
The single node crashes, the whole system crashes ->
Duplicate the single node and choose the major outputs as the entire system's output ->
Who is gonna decide which output is the major output ->
If there is a single decision maker, what if the decision maker crashes ->
The only way is to let nodes talk with each other and make the decision among themselves ->
BGP = distributed decision making + fault tolerance



Problem mathematical description:
We want to build the system above -> How? What algorithm?
We want to somehow prove that the system can reach a decision with existence of faulty nodes -> How to prove the algorithm and what we need to prove?

Need to first create a mathematical model which can reflect the requirements of the reality!

In 1982, LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE proposed and proved a such algorithm in their classic paper: *The Byzantine Generals Problem*

The model is as following:
Model 1:
We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. The generals can communicate with one another only by messenger. After observing the enemy, they must decide upon a common plan of action. However, some of the generals may be traitors, trying to prevent the loyal generals from reaching agreement. The generals must have an algorithm to guarantee that:
A. All loyal generals decide upon the same plan of action
B. A small number of traitors cannot cause the loyal generals to adopt a bad plan.

Model 2:
This is a conceptual model. It is hard to analyze. So we refine it to the following one:
Let v(i) be the information communicated by the ith general. Each general uses some method for combining the values v (1) ..... v (n) into a single plan of action, where n is the number of generals. The following must be true:
1. Any two loyal generals use the same value of v(i).
2. If the ith general is loyal, then the value that he sends must be used by every loyal general as the value of v (i).
Model 3:
It is still hard to analyze because it is completely distributed. We don't know which general to start from when analyzing it. Actually we can reduce the model above to a centralized model. We name it commander-lieutenant problem. In n generals, there is one general acting as commander and n-1 other generals are lieutenants. 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.

Algorithm:
Now we have our model to depict the problem. It is the time to introduce the algorithm. Under different assumptions, we can have different algorithms. Today we are only gonna talk about solutions with oral messages with following **assumptions**:
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.

We name our algorithm OM:

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 vi 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 vi to each of the n - 2 other lieutenants.

(3) For each i, and each j != i, let vj 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 majority (vl ..... v,-1).

Also we have majority function defined as following:

majority(v1, ..., v,-1):

1. The majority value among the vi if it exists, otherwise the value RETREAT;
2. The median of the vi, assuming that they come from an ordered set.

Examples:

n = 3 and 1 traitor:

We can see that n =3 and 1 traitor is an impossible case. So our algorithm has a **limitation**: n >= 3t + 1 if there are t traitors.

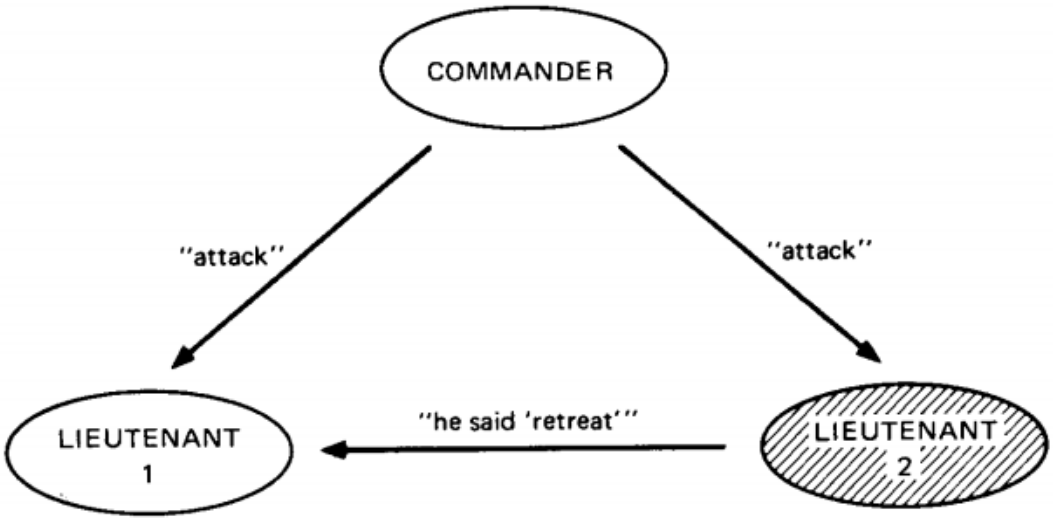


Fig. 1. Lieutenant 2 a traitor.

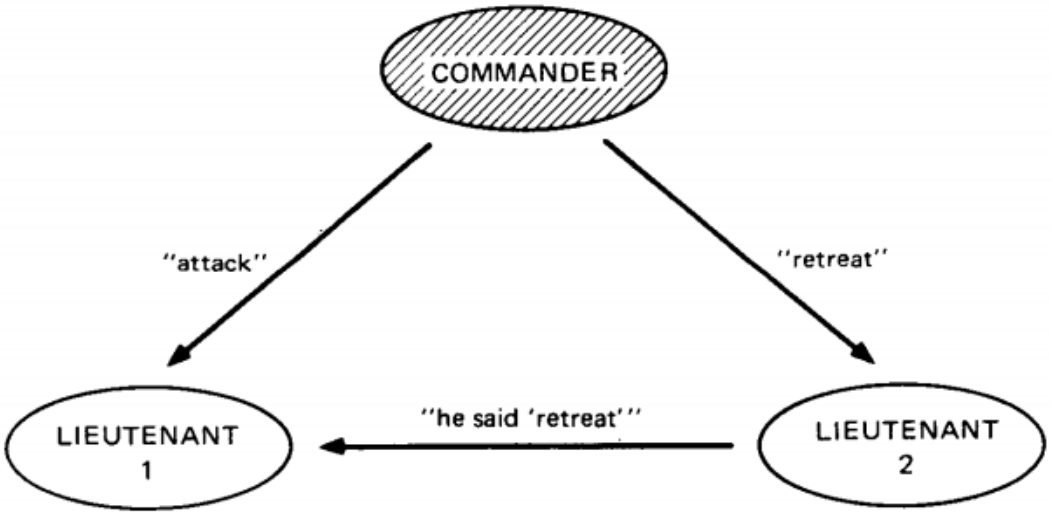


Fig. 2. The commander a traitor.

n = 4 and 1 traitor:

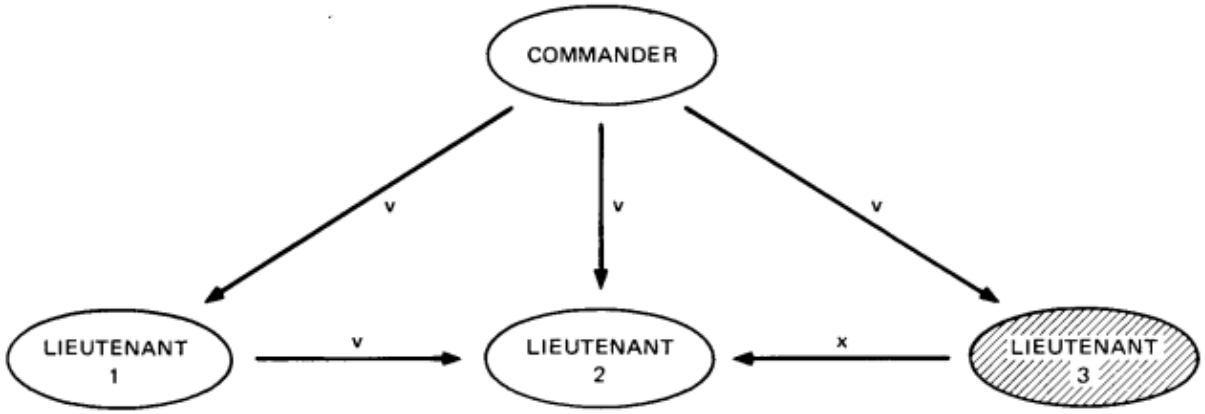


Fig. 3. Algorithm OM(1); Lieutenant 3 a traitor.

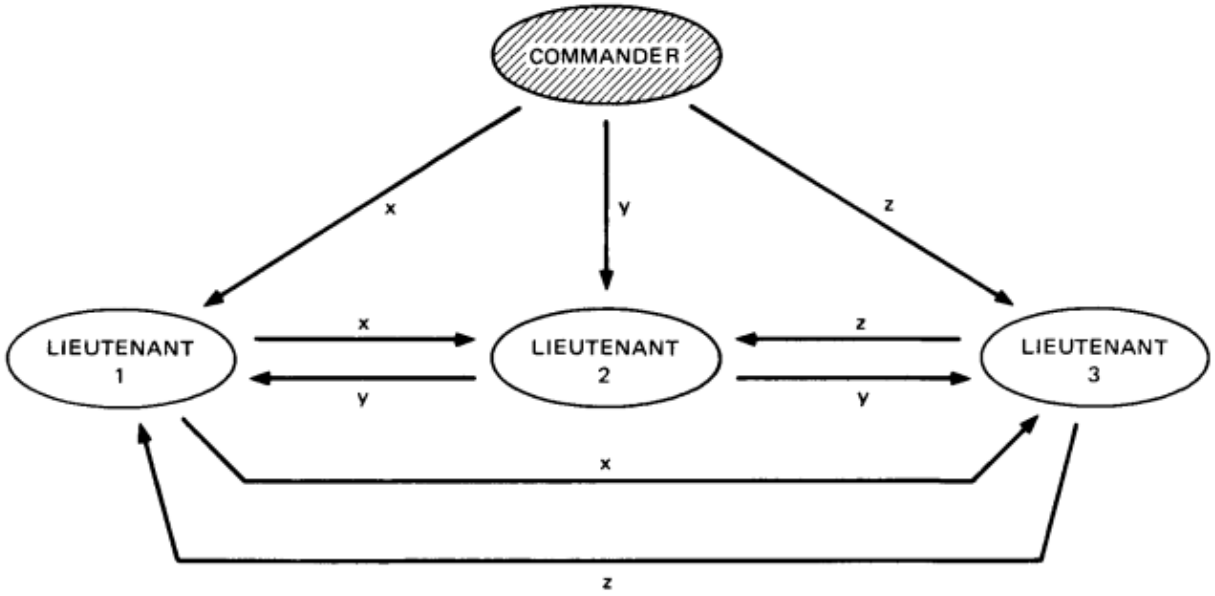


Fig. 4. Algorithm OM(1); the commander a traitor.

Proof:

Finally we need to prove that this algorithm can meet the requirement, which means it can provide us with a fault tolerance system if we implement it in reality.

To prove the correctness of the algorithm OM{m) for arbitrary m, we first prove the following lemma.

LEMMA 1. For any m and k, Algorithm OM (m) satisfies IC2 if there are more than 2k + m generals and at most k traitors.

PROOF. The proof is by induction on m. IC2 only specifies what must happen if the commander is loyal. Using A1, it is easy to see that the trivial algorithm OM(0) works if the commander is loyal, so the lemma is true for m = 0. We now assume it is true for m - 1, m > 0, and prove it for m. In step (1), the loyal commander sends a value v to all n - 1 lieutenants. In step (2), each loyal lieutenant applies OM(m - 1) with n - 1 generals. Since by hypothesis n > 2k + m, we have n - 1 > 2k + (m - 1), so we can apply the induction hypothesis to conclude that every loyal lieutenant gets vj = v for each loyal Lieutenant j. Since there are at most k traitors, and n - 1 > 2k + (m - 1) => 2k, a majority of the n - 1 lieutenants are loyal. Hence, each loyal lieutenant has vi = v for a majority of the n - 1 values i, so he obtains majority(v1 .... , v,-1) = v in step (3), proving IC2.

THEOREM 1. For any m, Algorithm OM (m) satisfies conditions IC1 and IC2 if there are more than 3m generals and at most m traitors.

PROOF. The proof is by induction on m. If there are no traitors, then it is easy to see that OM(0) satisfies IC1 and IC2. We therefore assume that the theorem is true for OM(m - 1) and prove it for OM(m), m > 0. We first consider the case in which the commander is loyal. By taking k equal to m in Lemma 1, we see that OM(m) satisfies IC2. IC1 follows from IC2 if the commander is loyal, so we need only verify IC1 in the case that the commander is a traitor.
There are at most m traitors, and the commander is one of them, so at most m - 1 of the lieutenants are traitors. Since there are more than 3m generals, there are more than 3m - 1 lieutenants, and 3m - 1 > 3(m - 1). We may therefore apply the induction hypothesis to conclude that OM(m - 1) satisfies conditions IC1 and IC2. Hence, for each j, any two loyal lieutenants get the same value for vj in step (3). (This follows from IC2 if one of the two lieutenants is Lieutenant j, and from IC1 Otherwise.) Hence, any two loyal lieutenants get the same vector of values v1..... Vn-1, and therefore obtain the same value majority(v1 ..... Vn-1) in step (3), proving IC1.

Time complexity:

The recursive algorithm OM(m) invokes n - 1 separate executions of the algorithm OM(m - 1), each of which invokes n - 2 executions of OM(m - 2), etc. This means that, for m > 1, a lieutenant sends many separate messages to each other lieutenant. There must be some way to distinguish among these different messages. The reader can verify that all ambiguity is removed if each lieutenant i prefixes the number i to the value vi that he sends in step (2). As the recursion "unfolds," the algorithm OM(m - k) will be called (n - 1) ... (n - k) times to send a value

prefixed by a sequence of k lieutenants' numbers.

We can see that this algorithm has an extremely high time cost. So it is almost impossible to implement it in reality. That is why we need a randomized algorithm to reduce the time cost. Next lecture Sachi will introduce one of **randomized BGP algorithms**.

Thank you!