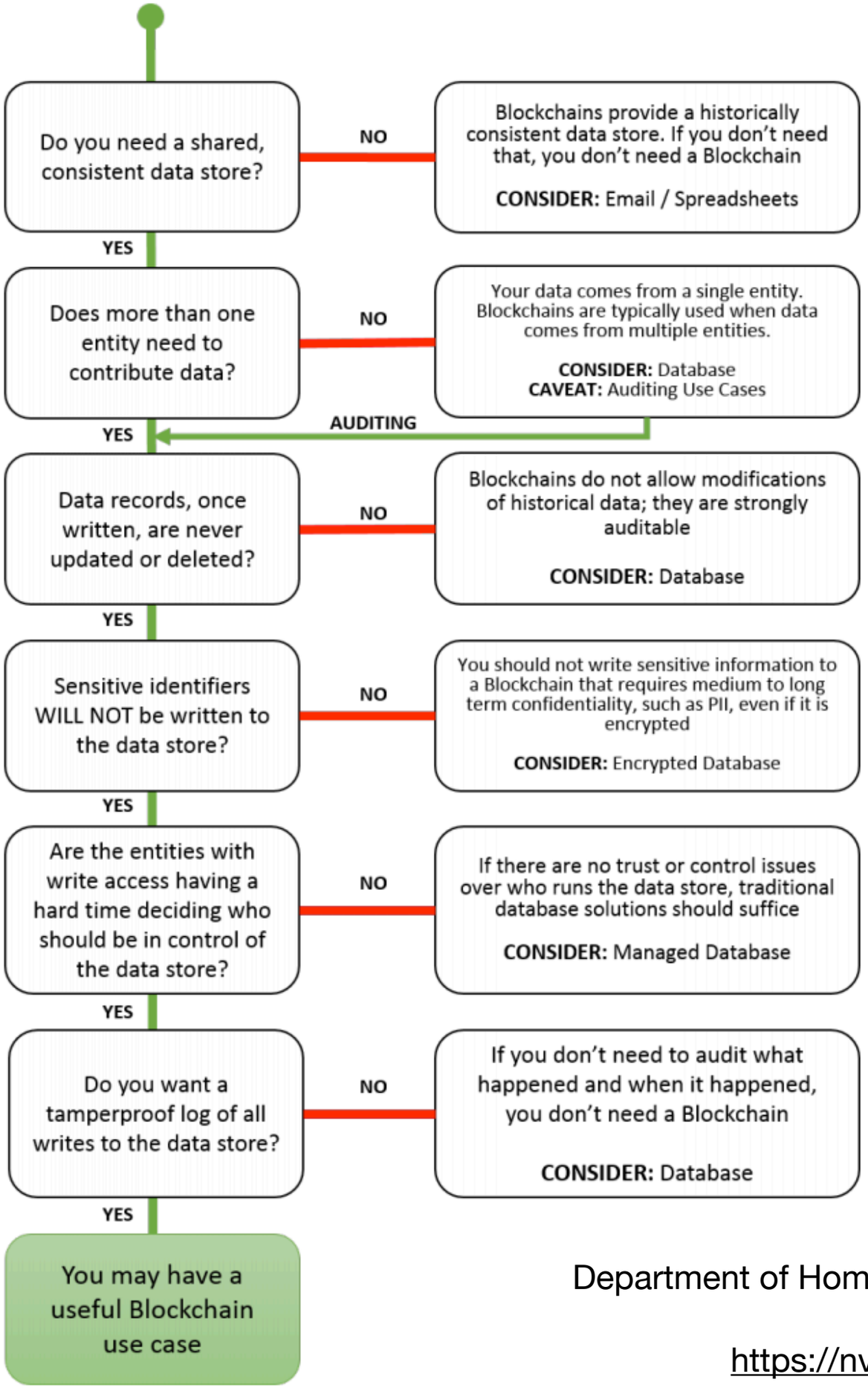
Byzantine Generals Problem

August 26, 2019



source:
Department of Homeland Security, Science & Technology Directorate

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

Do you need a shared, consistent data store?

NO

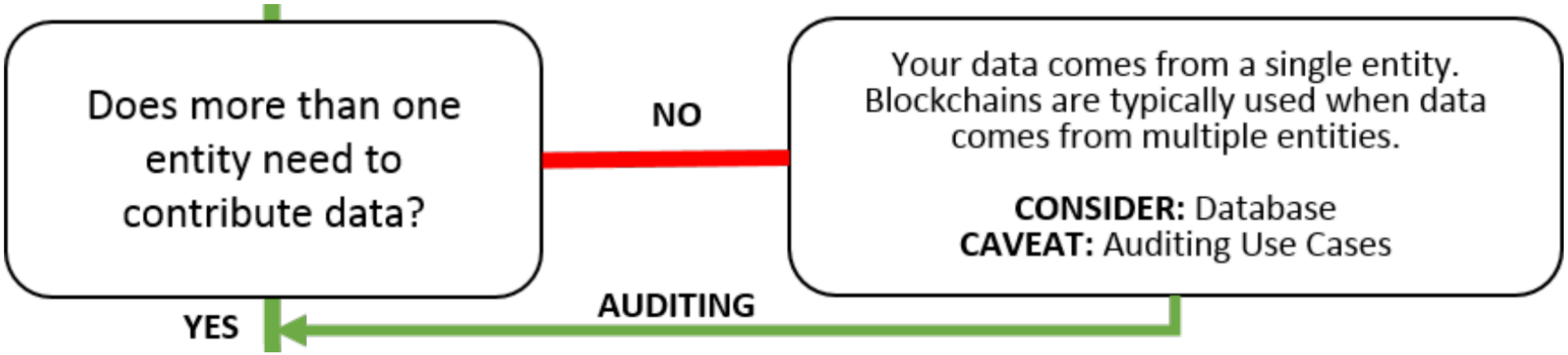
Blockchains provide a historically consistent data store. If you don't need that, you don't need a Blockchain

CONSIDER: Email / Spreadsheets

YES

source:
Department of Homeland Security, Science & Technology Directorate

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf



source:
Department of Homeland Security, Science & Technology Directorate

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

Data records, once written, are never updated or deleted?

NO

Blockchains do not allow modifications of historical data; they are strongly auditable

CONSIDER: Database

YES

source:
Department of Homeland Security, Science & Technology Directorate

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

Sensitive identifiers WILL NOT be written to the data store?

NO

You should not write sensitive information to a Blockchain that requires medium to long term confidentiality, such as PII, even if it is encrypted

CONSIDER: Encrypted Database

YES

PII = Personally Identifiable Information

source:
Department of Homeland Security, Science & Technology Directorate

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

Are the entities with write access having a hard time deciding who should be in control of the data store?

NO

If there are no trust or control issues over who runs the data store, traditional database solutions should suffice

CONSIDER: Managed Database

YES

This is the important issue for this lecture

source:
Department of Homeland Security, Science & Technology Directorate

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

Do you want a tamperproof log of all writes to the data store?

NO

If you don't need to audit what happened and when it happened, you don't need a Blockchain

CONSIDER: Database

YES

source:
Department of Homeland Security, Science & Technology Directorate

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

You may have a useful Blockchain use case

source:
Department of Homeland Security, Science & Technology Directorate

https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

Consensus

- Assume a system with different, independent actors
  - GPS satellites w/ clocks
  - Distributed Databases
  - Politics/Diplomacy
  - Large scale computation (Google’s search engine, …)
- How to reach consensus?

How to reach Consensus?

- How to reach consensus?
  - Send messages between actors
- Issues?

How to reach Consensus?

- How to reach consensus?
  - Send messages between actors

- Issues?
  - Identity, Message Spoofing/Verification, Eves-dropping, Forgery, Dropped/Lost Messages, Delays, Malicious actors/conflicting messages, etc.

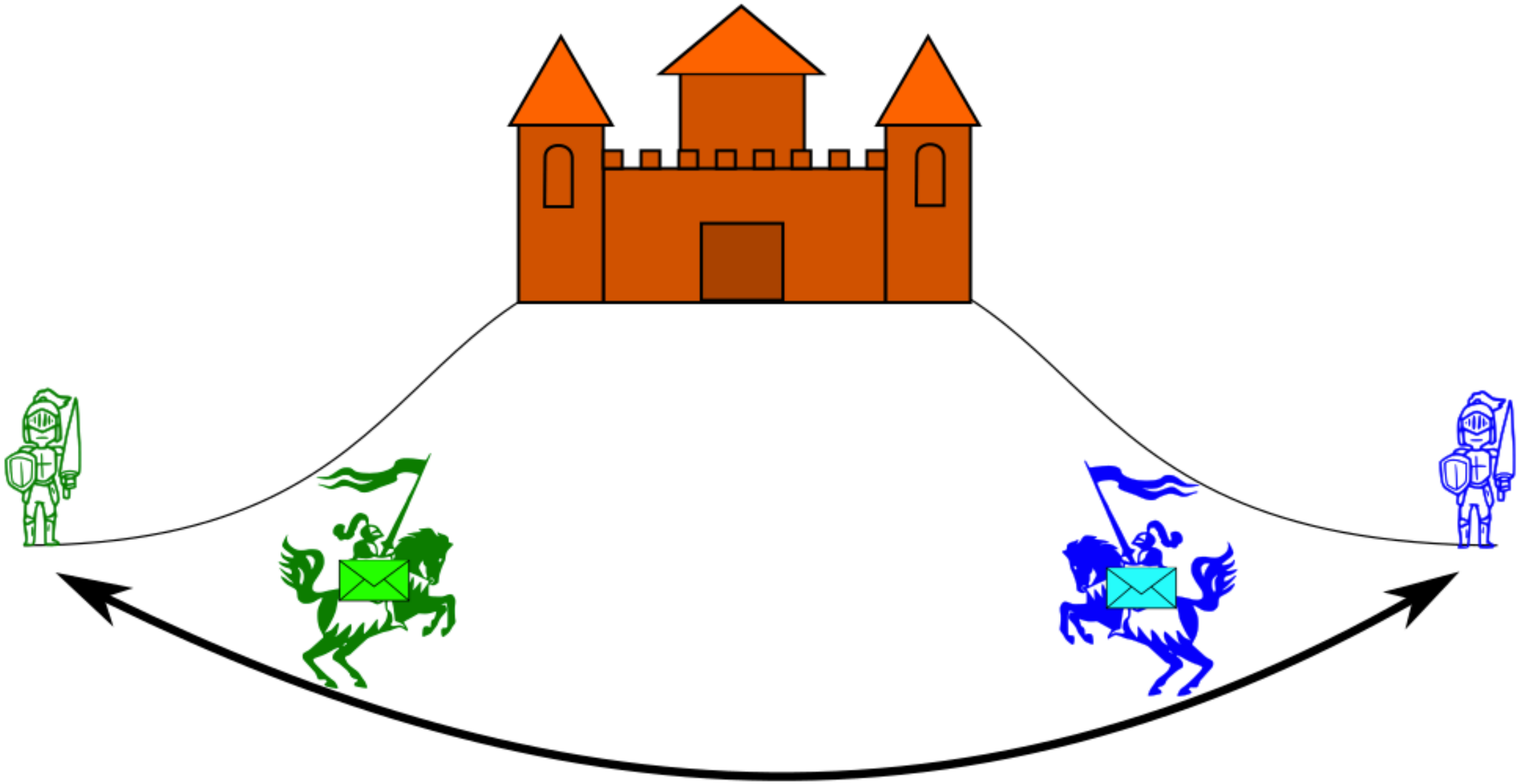
How to reach consensus in distributed systems

- Sending messages

- Harder than it may appear at first glance

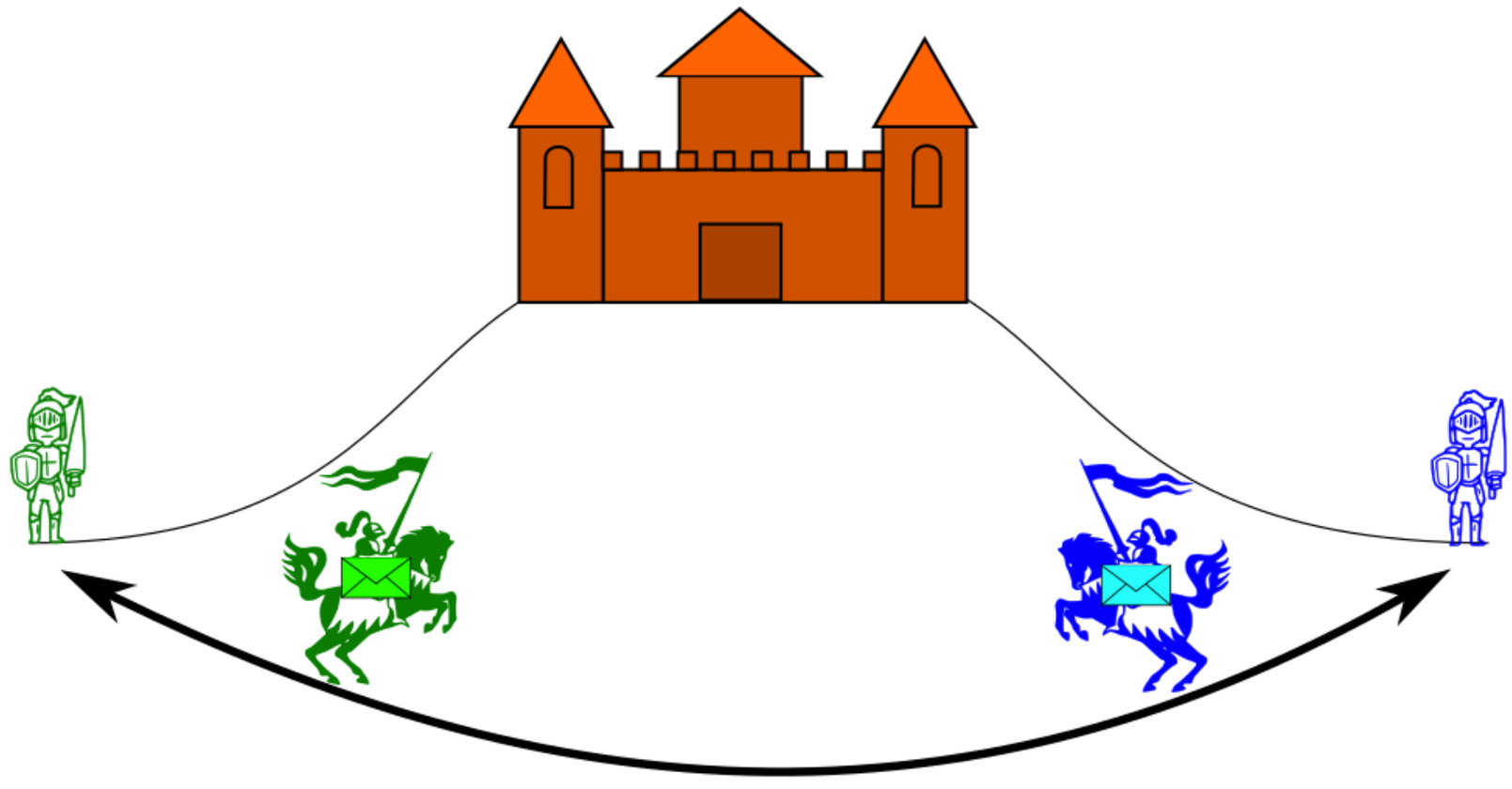
A simple thought experiment

- Two generals/armies/knights want to attack
- All they need to do is to decide when to attack
- What is the most efficient communication protocol/ smallest number of messages to agree on a plan



A simple thought experiment - details

- Both parties are honest
- Need to agree on 1 bit of information (attack/retreat)
  Definite agreement must be reached
  (Green attacks and hopes the blue attack as well is not good enough)
- Messenger might be intercepted
  Communication via unreliable channel

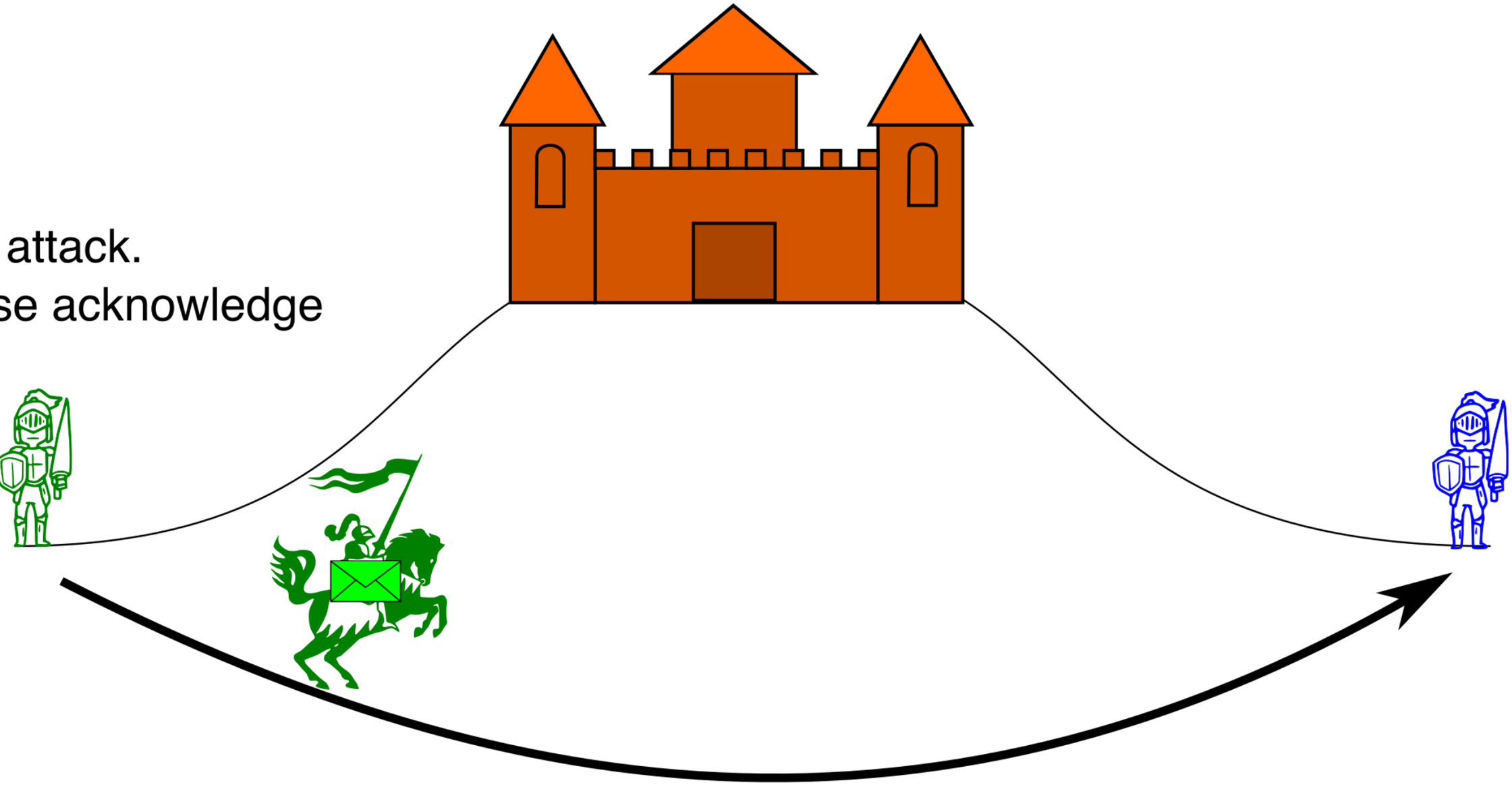


- What is the most efficient communication protocol/smallest number of messages to agree on a plan

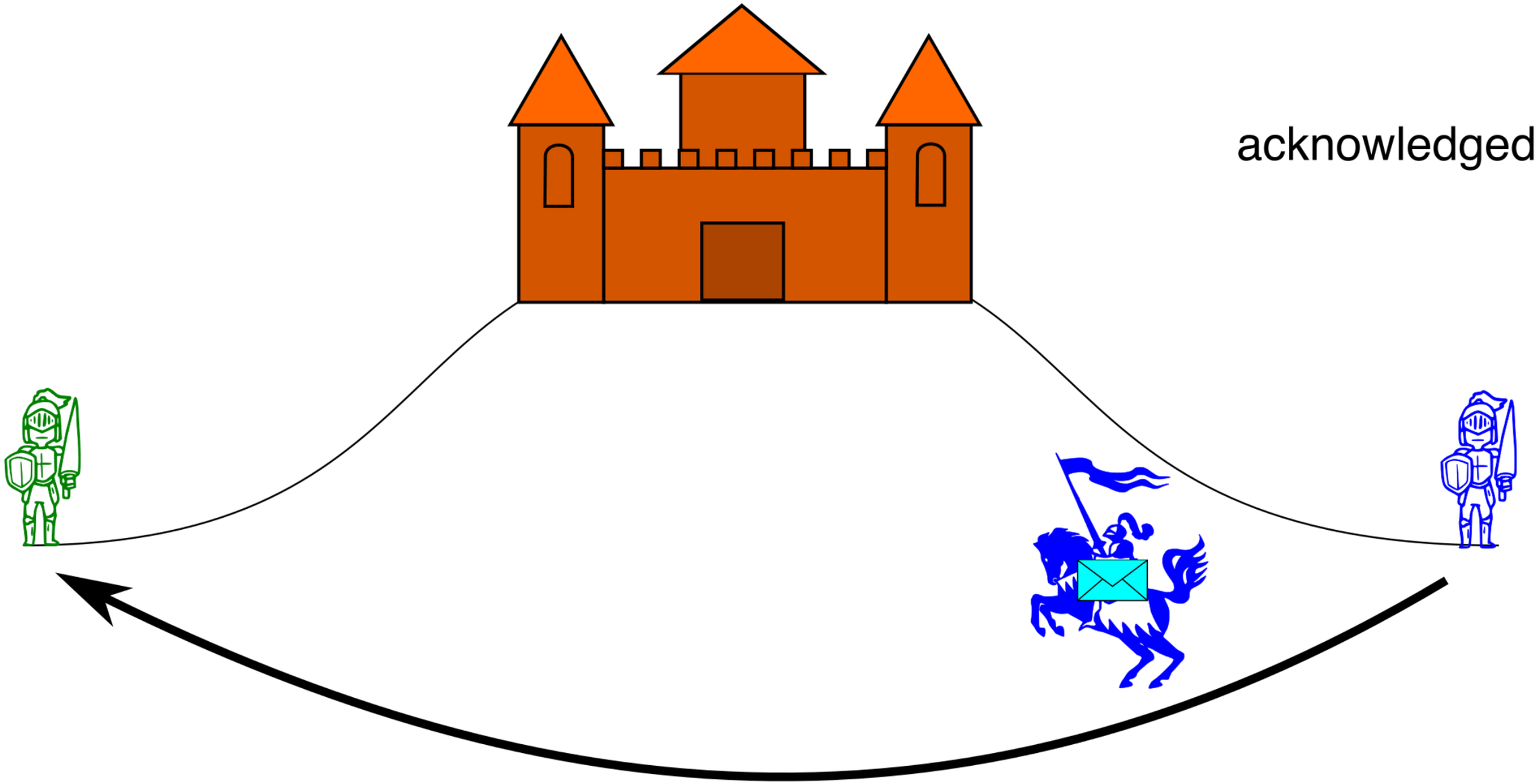
A simple thought experiment - details

- Solution not possible
- To definitely reach consensus (finality), an infinite number of messages must be sent

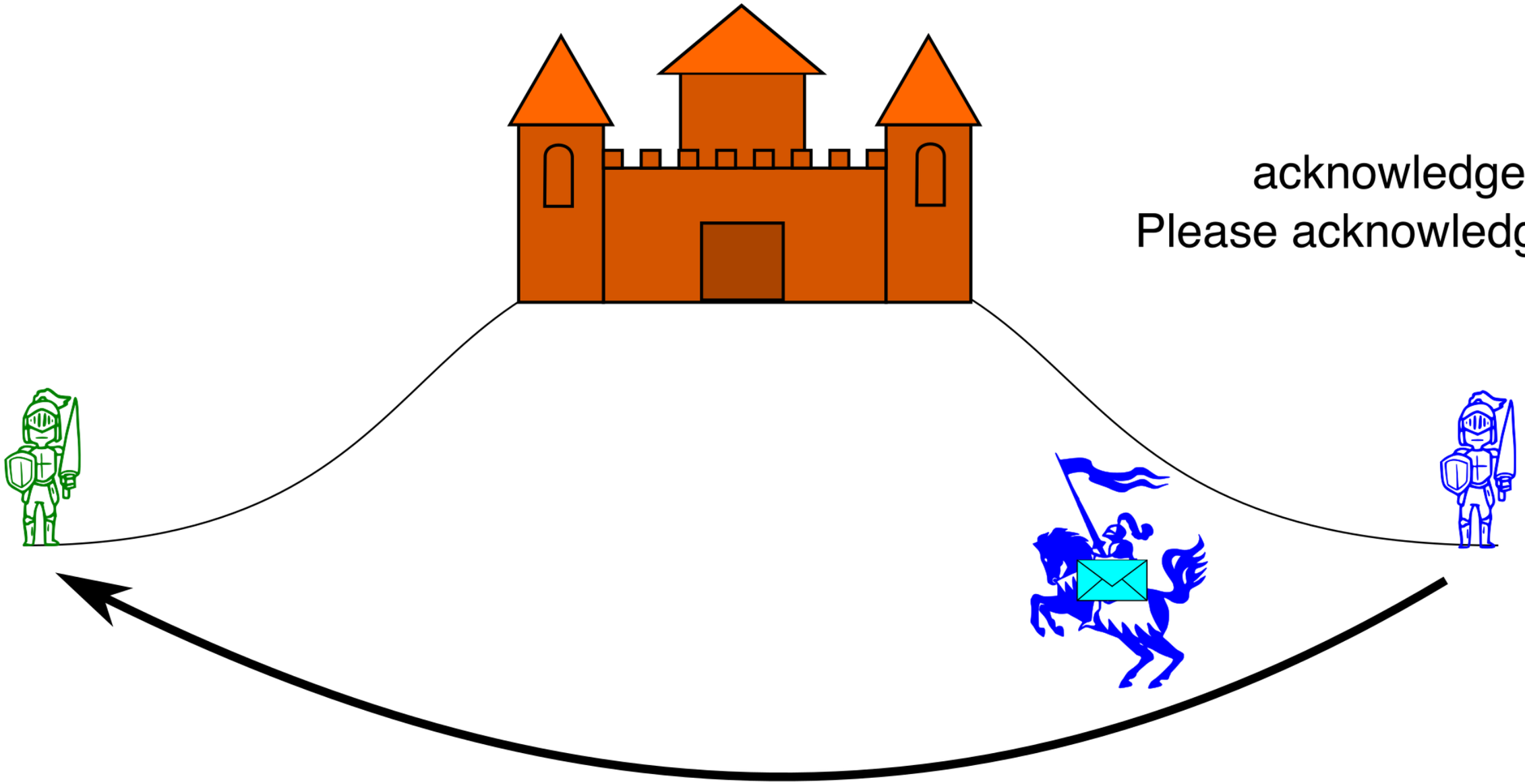
Let's attack.
Please acknowledge



Why is acknowledgement needed?



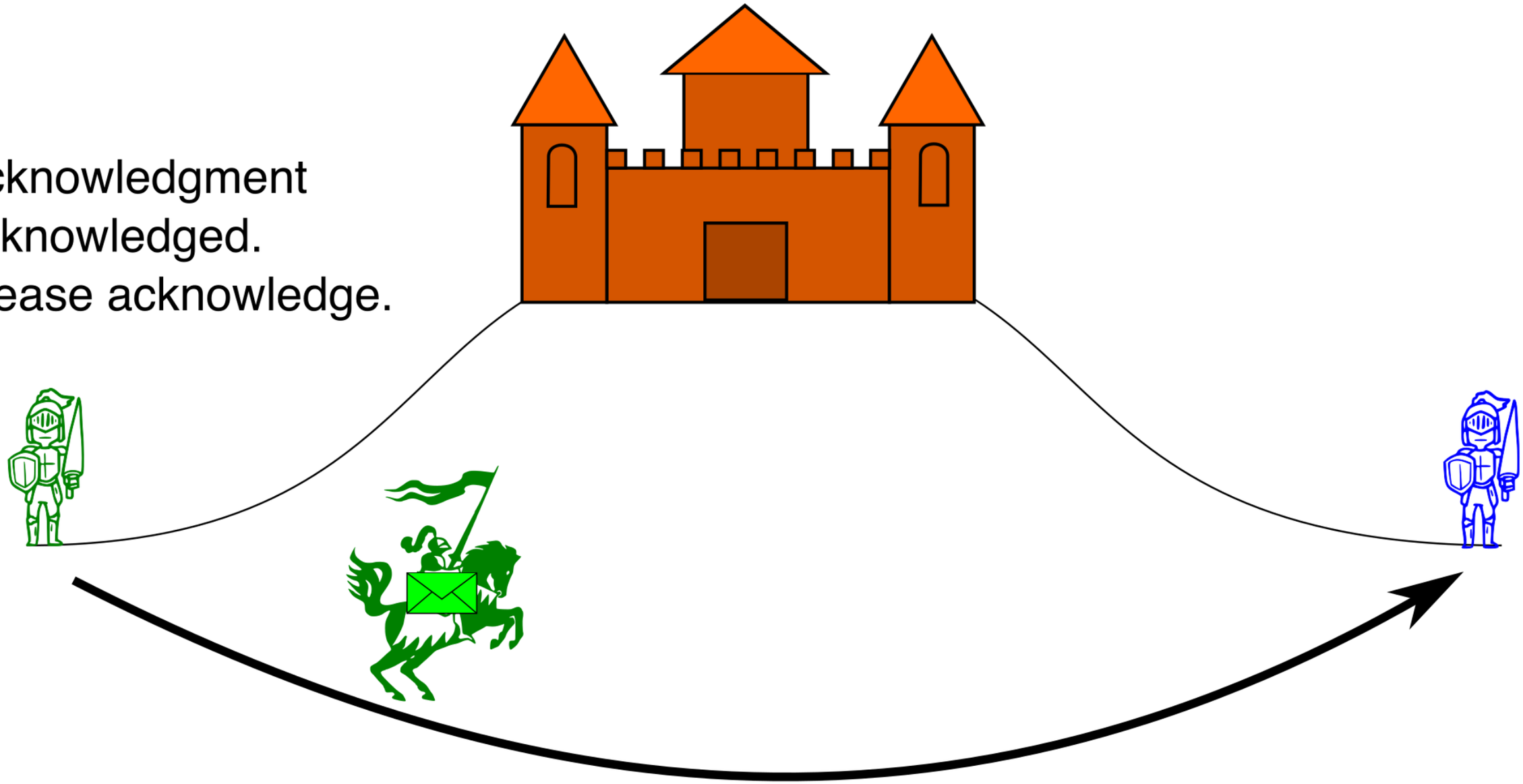
Is that enough?



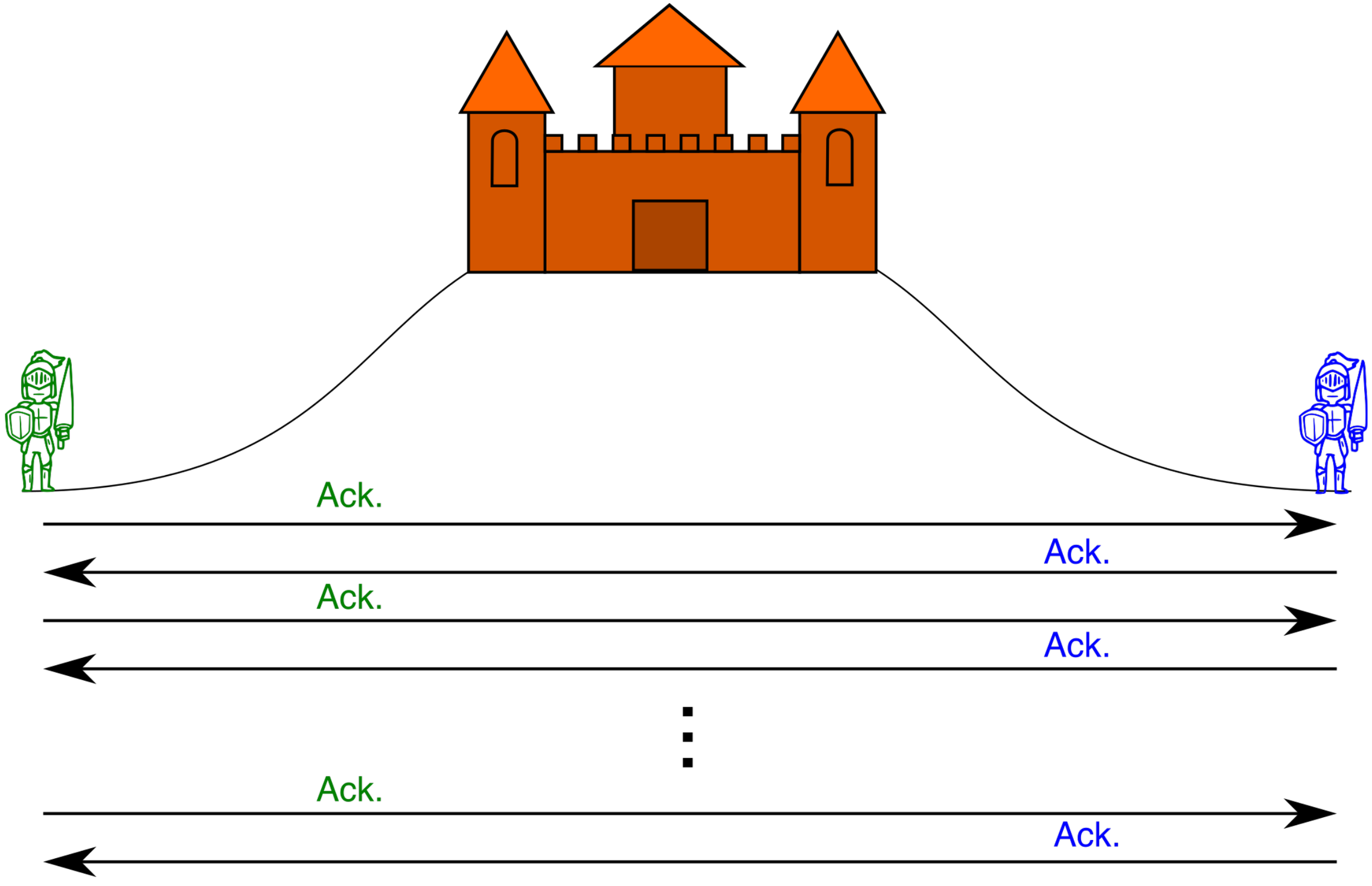
acknowledged.
Please acknowledge

Why is acknowledgement needed again?

Acknowledgment
acknowledged.
Please acknowledge.



Why is acknowledgement needed?



When do we stop?

How do we know that the last message was transmitted correctly?

Byzantine Generals Problem

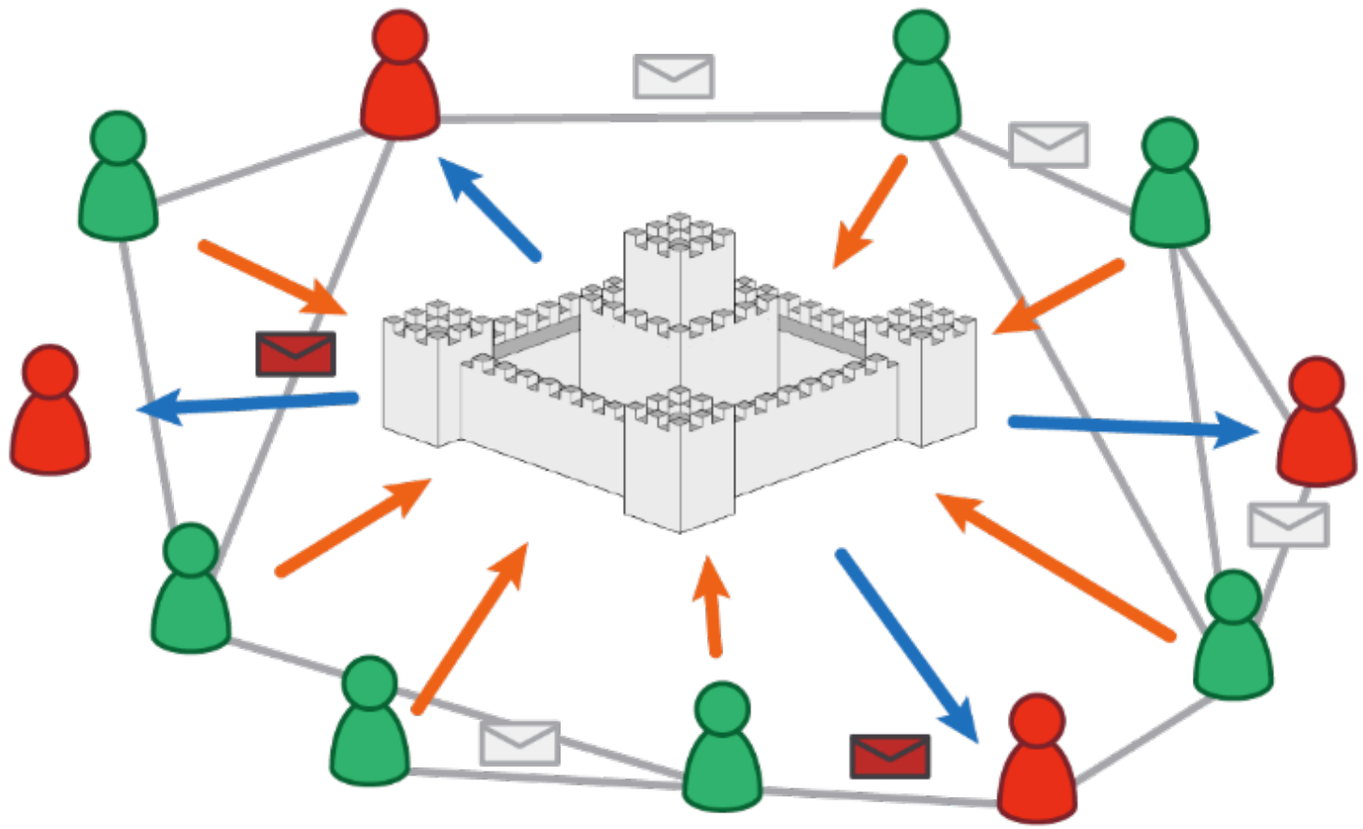
- In case of unreliable message channels (and without a timeout), consensus is impossible, even with well-behaving agents.

- For the remainder of this lecture, assume reliable transmissions

- Actors, however, may be malicious

    - Can we still guarantee consensus?

Byzantine Generals Problem



“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 and (B) A small number of traitors cannot cause the loyal generals to adopt a bad plan.”

- Leslie Lamport, Robert Shostak, Marshall Pease
The Byzantine Generals Problem

Byzantine Generals Problem

- How to reach consensus among different participants in presents of faulty/malicious nodes/traitors

- Name:

  There is a problem in distributed computing that is sometimes called the Chinese Generals Problem, in which two generals have to come to a common agreement on whether to attack or retreat, but can communicate only by sending messengers who might never arrive. I stole the idea of the generals and posed the problem in terms of a group of generals, some of whom may be traitors, who have to reach a common decision. **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.

Important papers

Impossibility of Distributed Consensus with One Faulty Process

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY A. LYNCH

Massachusetts Institute of Technology, Cambridge, Mass

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England

Abstract. The consensus problem involves an asynchronc
unreliable. The problem is for the reliable processes to ag
that every protocol for this problem has the possibility
process. By way of contrast, solutions are known for the
problem.

The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

Categories and Subject Descriptors: C.2.4. **[Computer-Communication Networks]**: Distributed Systems—*network operating systems*; D.4.4 **[Operating Systems]**: Communications Management—*network communication*; D.4.5 **[Operating Systems]**: Reliability—*fault tolerance*

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Interactive consistency

> 30 years ago

Definitions

- General : Node in the system
- Loyal: A general that follows the plan
- Traitor: A general that wants to spoil the plan
    - This "requires saying precisely what a bad plan is, and we do not attempt to do so. Instead, we consider how the generals reach a decision."

Definitions

- Goal: An algorithm that can guarantee
  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.

Simple approach

- Every general i sends its value $v(i)$ to all others
  - $v(i) =$ “attack”/“retreat”
- Every node uses the information it receives to make a decision

Problem

- This algorithm does not work

    A. All loyal generals decide upon the same plan of action

- To fulfill condition A, we require that all loyal generals have the same input values *v(1), v(2), ..., v(n)*

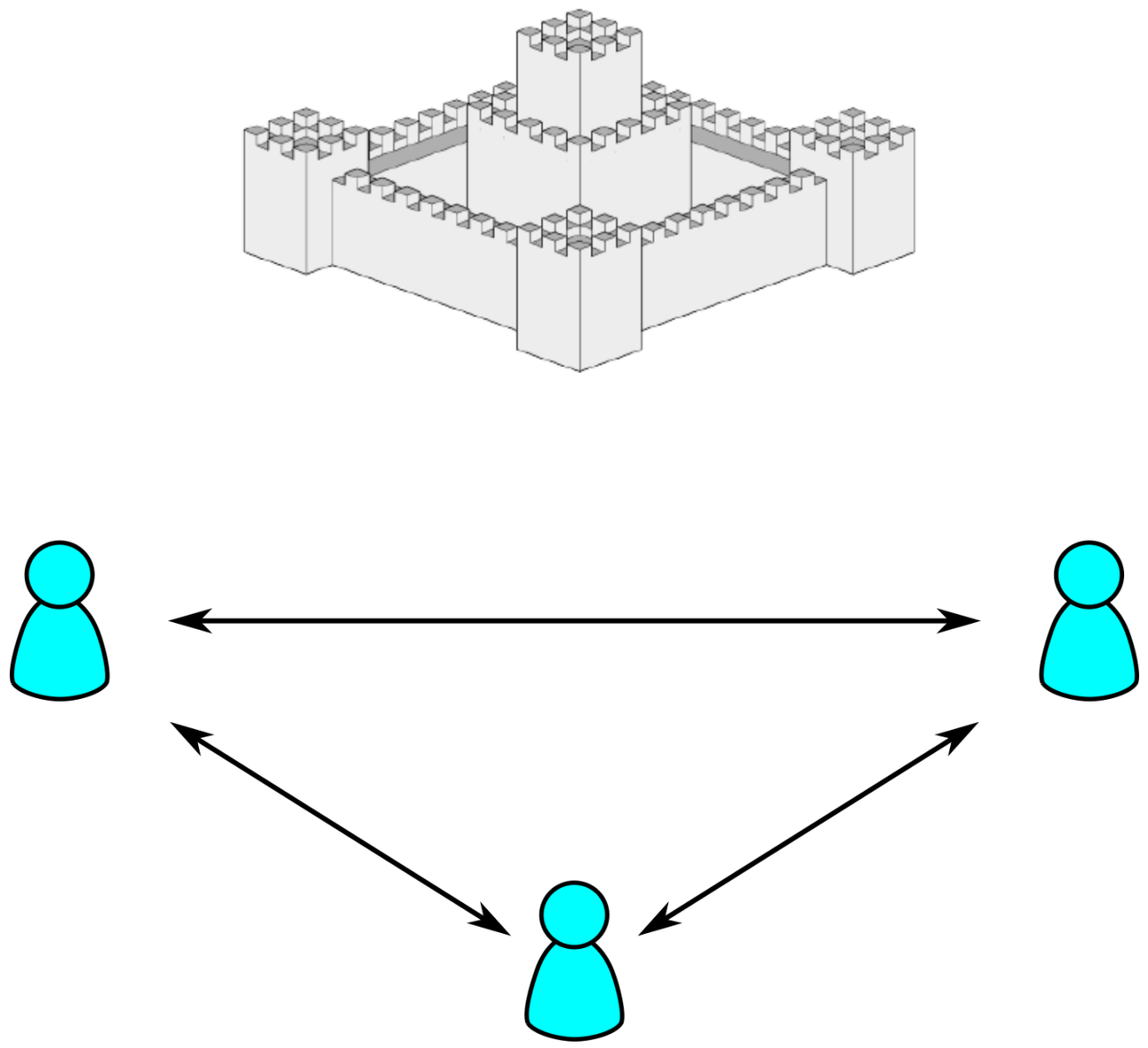
    - Traitors can send different values to loyal generals

- Refined condition

    - Every loyal general must obtain the same information *v(1),...,v(n)*.

Problem

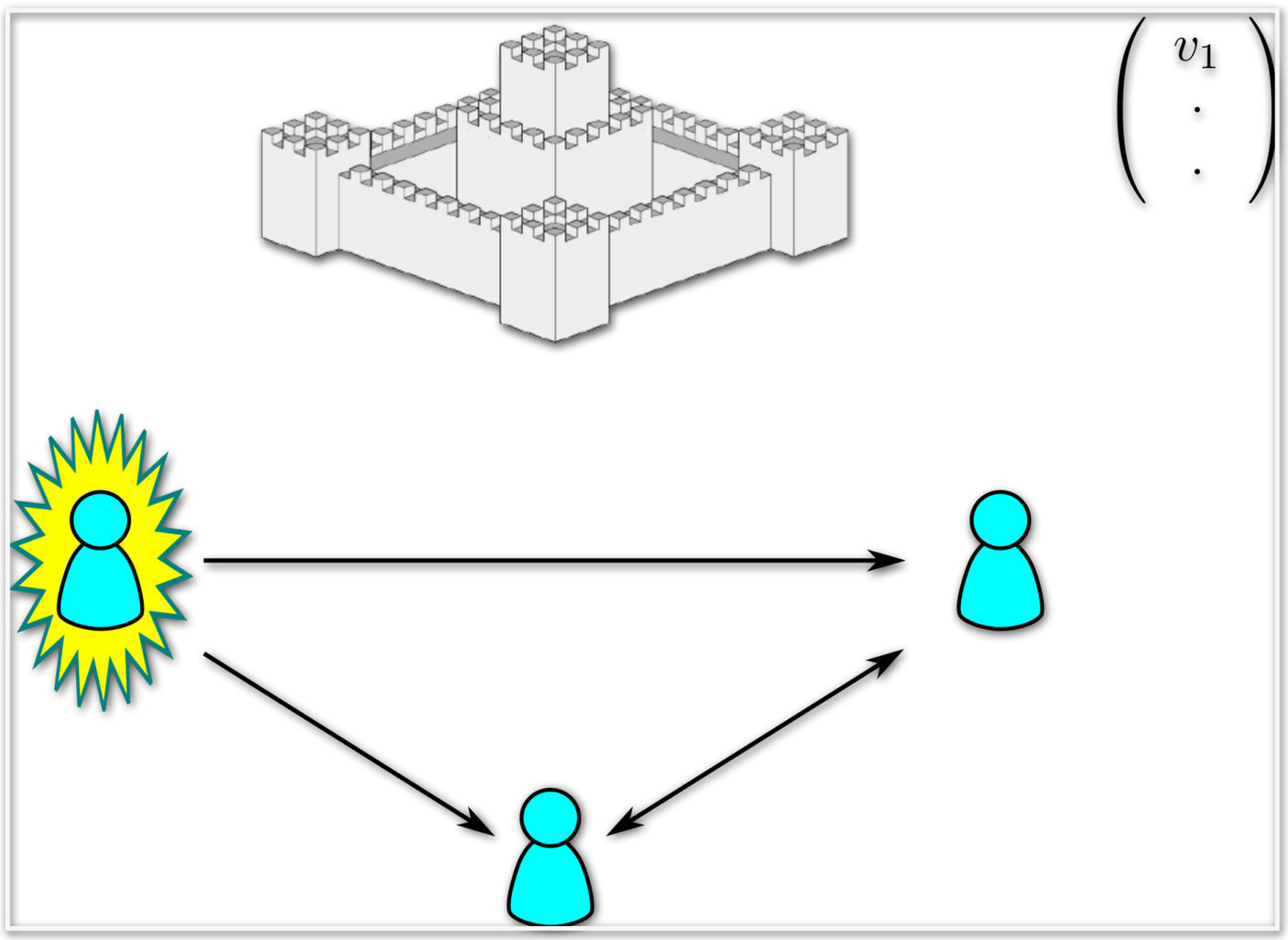
- To fulfill the refined condition, generals must send further between each other
    - Careful not to confuse loyal generals
- Complete refined conditions
    1. Any two loyal generals use the same value of $v(i)$. (Regardless of i loyal or traitor)
    2. If the ith general is loyal, then the value that he sends must be used by every loyal general as the value of $v(i)$.

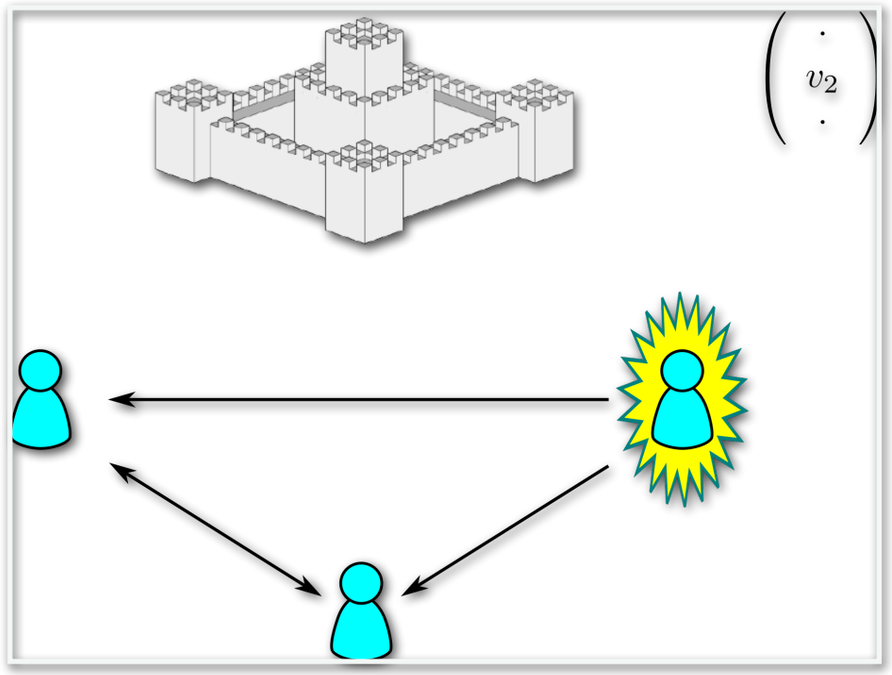


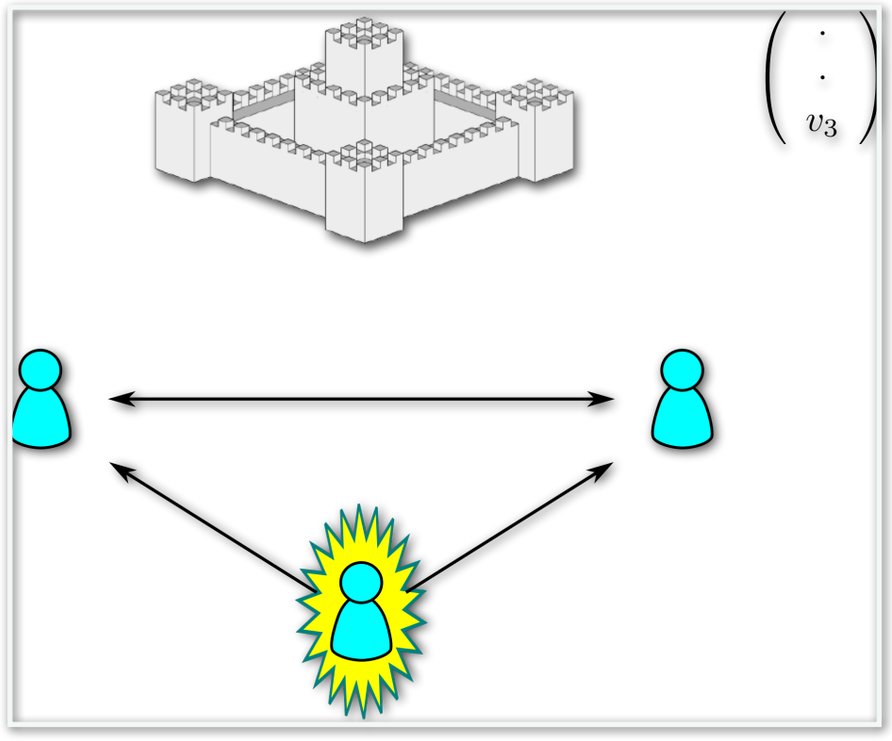
Assume 3 generals

Each sends message to each other

3 parallel situations







New definition

- General: The nodes sending a value
- Lieutenant: all other nodes

Byzantine Generals Problem

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

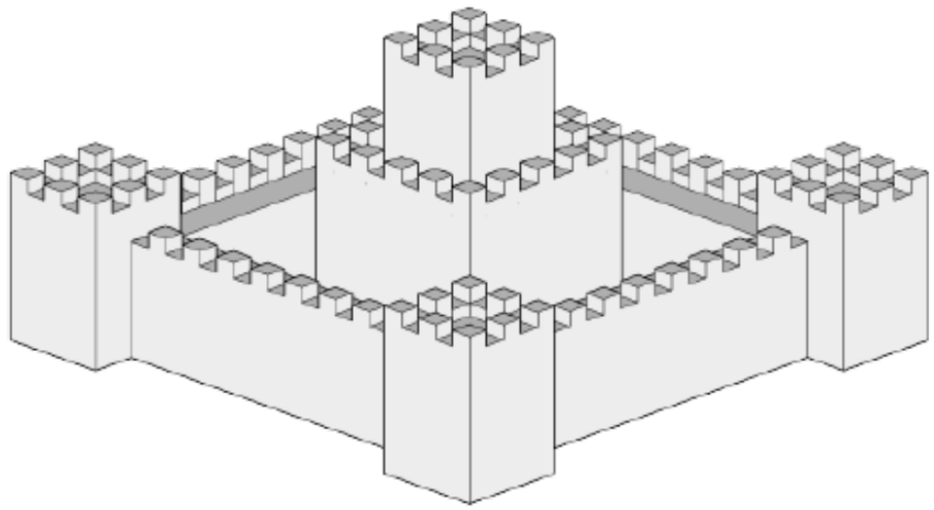
- IC1. All loyal lieutenants obey the same order.
- IC2. If the commanding general is loyal, then every loyal lieutenant obeys the

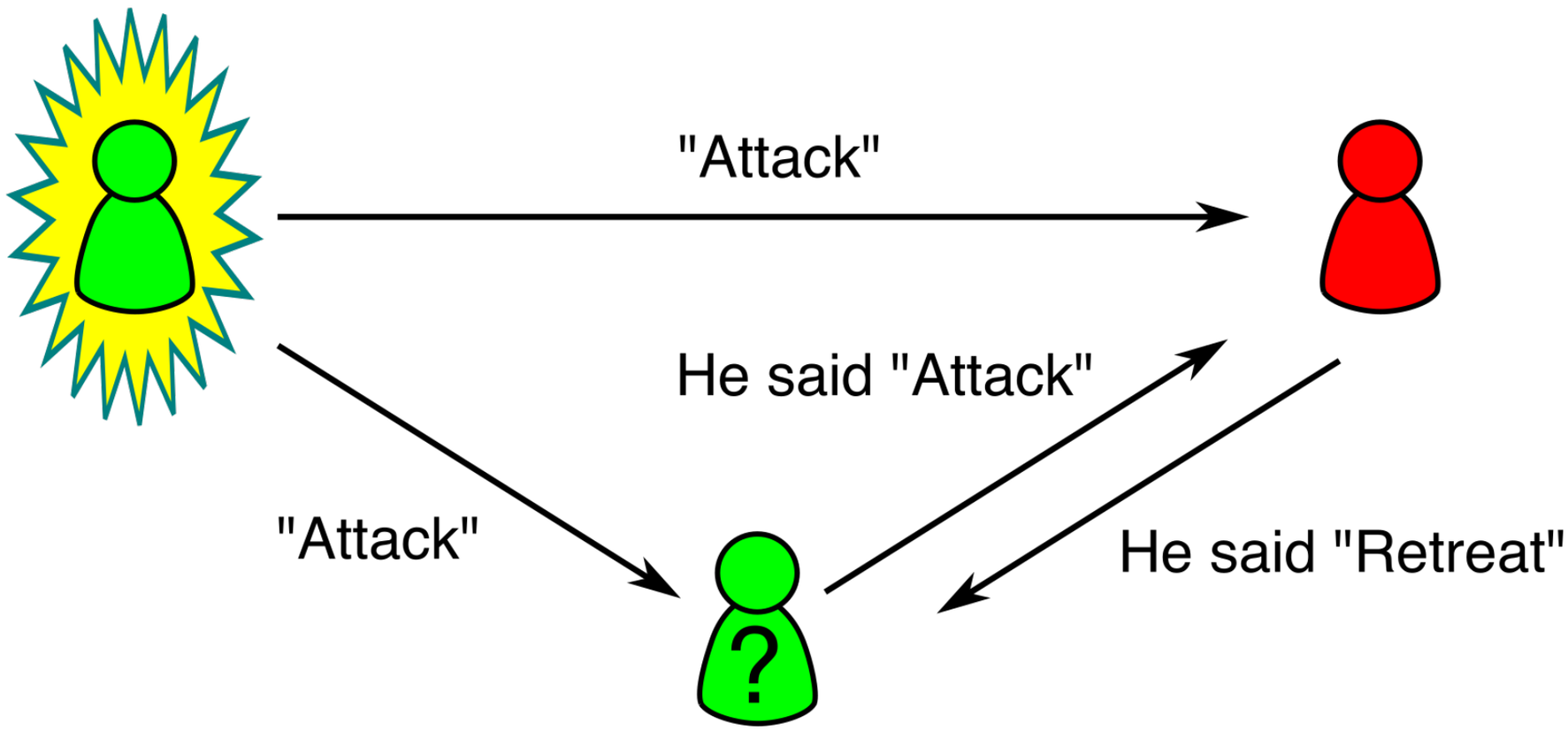
order he sends.

Assume 1 malicious node

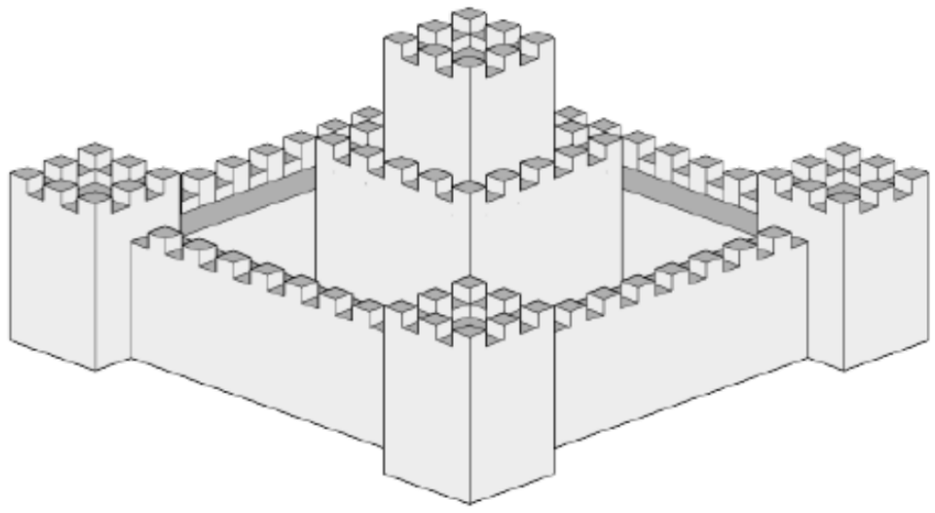
- 2 possible situations
  - One lieutenant is a traitor
  - General is traitor

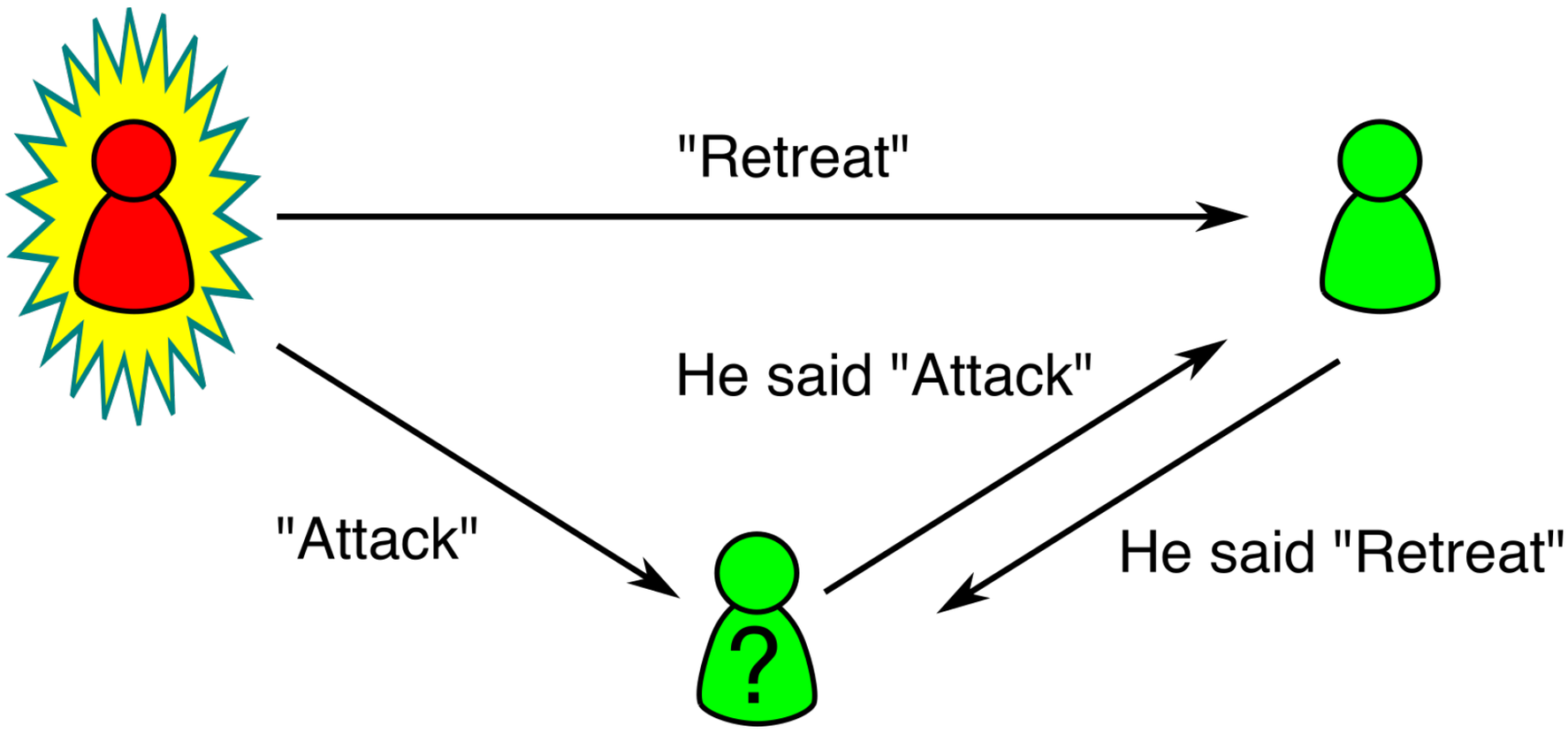
One lieutenant is a traitor





General is traitor





Consequence

- To the third node it is indistinguishable whether the other lieutenant or the general is a traitor
- If the general is a traitor both lieutenants are loyal
  - follow order from general
  - follow different orders
    - violates IC1 “All loyal lieutenants obey the same order”

Consequence

- A system with 3 nodes cannot handle a single malicious actor

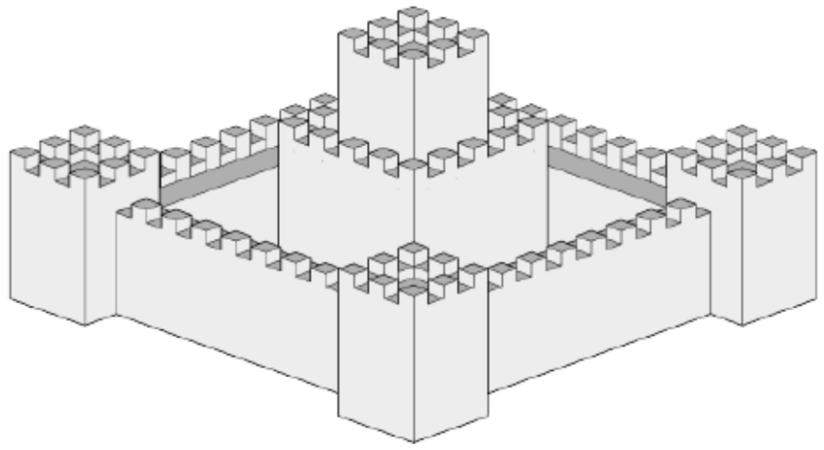
- Even though this hand-wavy argument is correct “[…] we strongly advise the reader to be very suspicious of nonrigorous reasoning. […]. We know of no area in computer science or mathematics in which informal reasoning is more likely to lead to errors than in the study of this type of algorithm.”

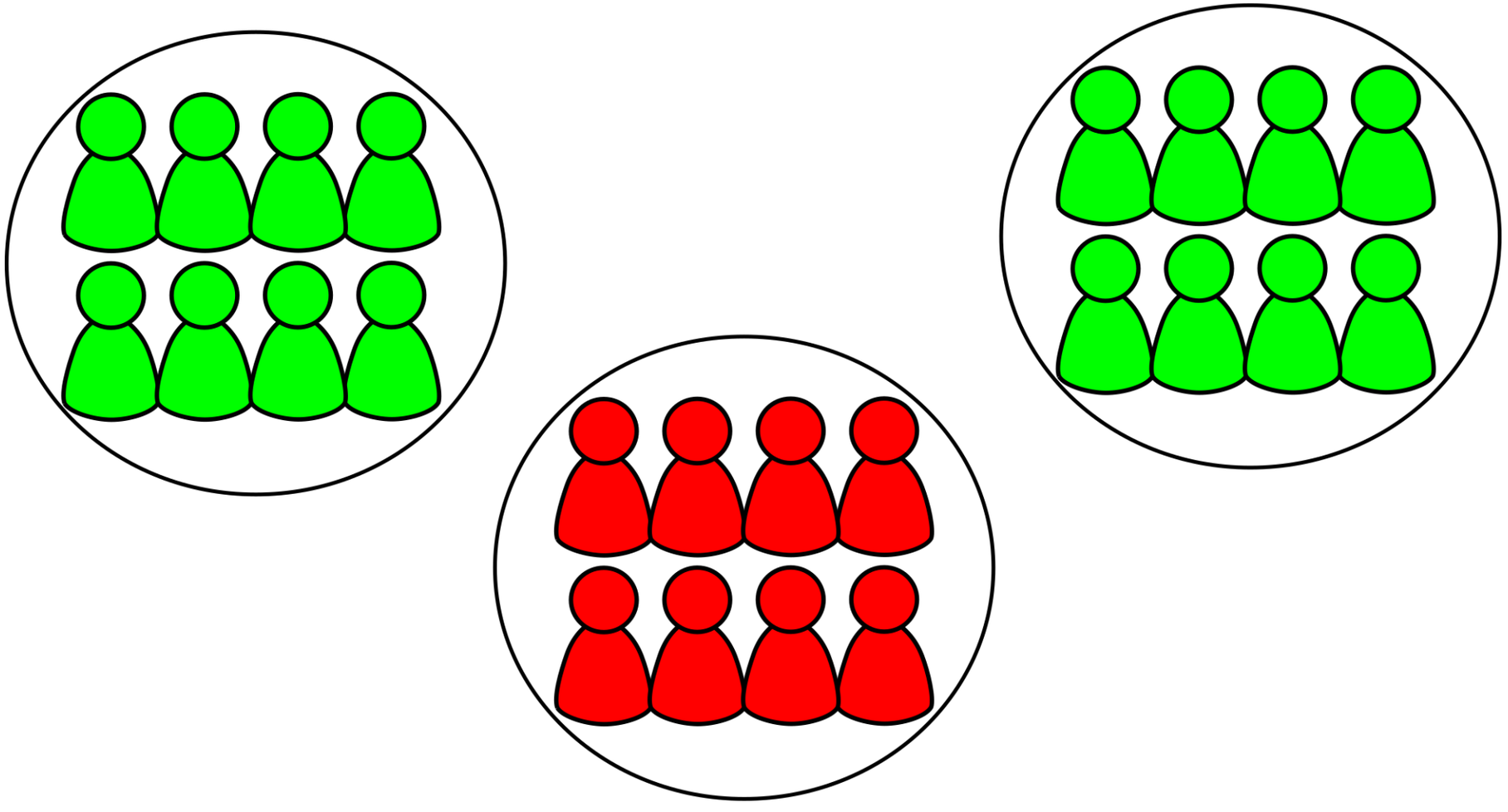
- Leslie Lamport, Robert Shostak, Marshall Pease
The Byzantine Generals Problem

General statement

- No system with <3m+1 can tolerate m traitors
  - If we could find a solution for m traitors, we can construct a solution for 3 nodes

Proof via Contradiction





Solution for <m traitors

- Oral message

  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.

  - A malicious command may not send any order. In absence of an order RETREAT.

Oral Message Algorithm

- for 3m+1 nodes, algorithm *OM(m)*

***Algorithm OM*(0).**

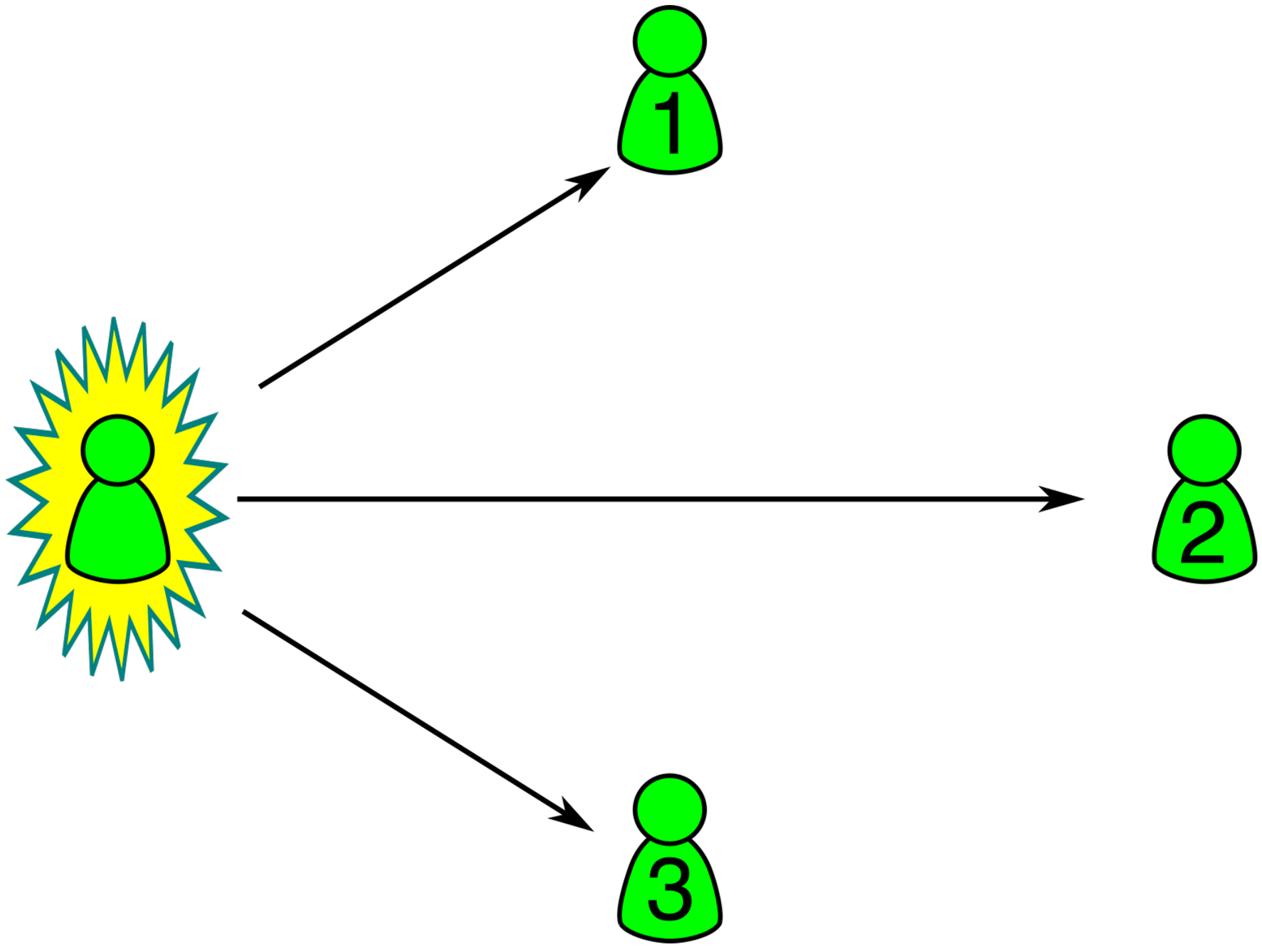
(1) The commander sends his value to every lieutenant.
(2) Each lieutenant uses the value he receives from the commander, or uses the value RETREAT if he receives no value.

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 *majority*($v_1, \ldots, v_{n-1}$).

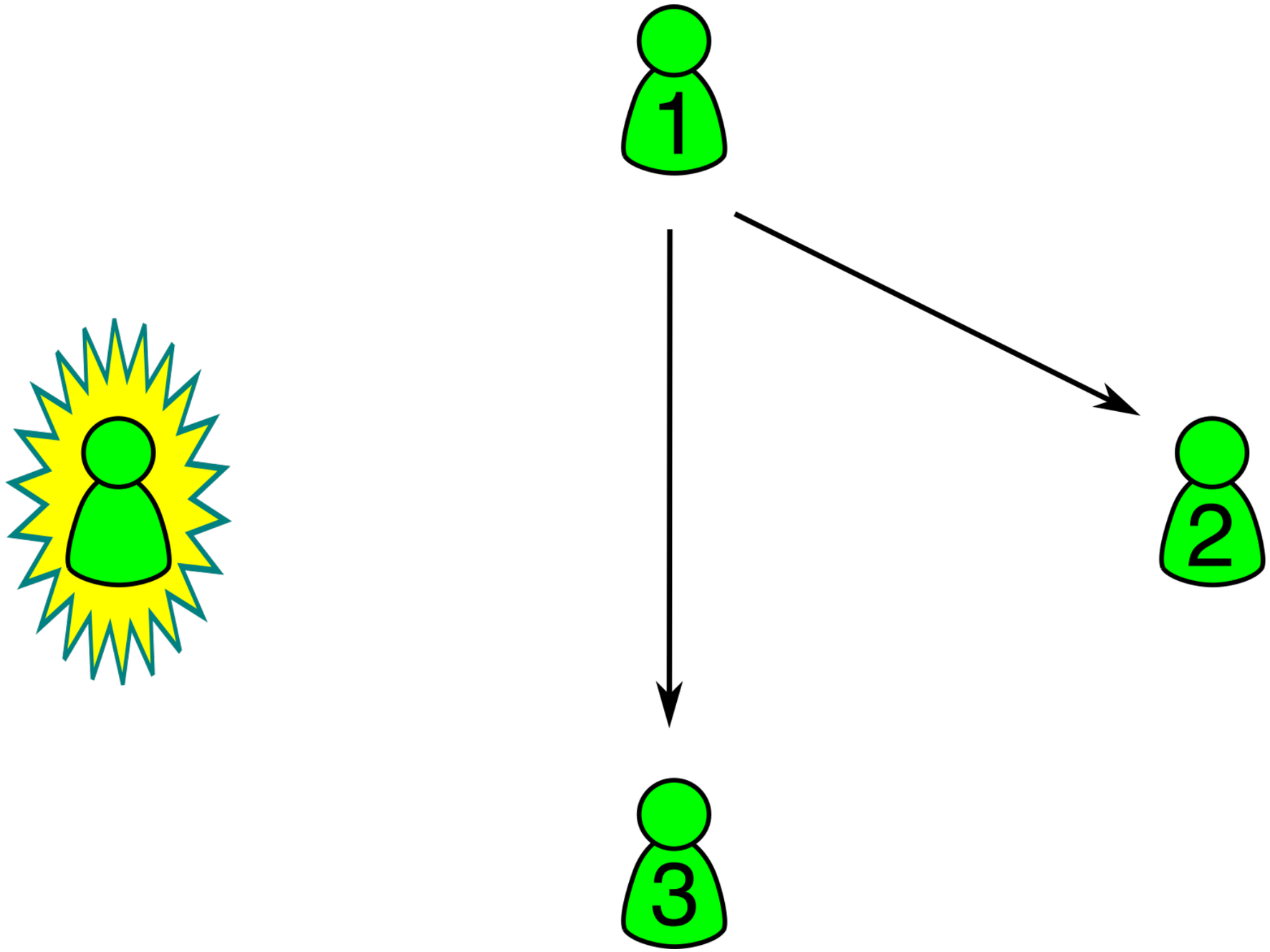
Example m=1, n=4

Step 1



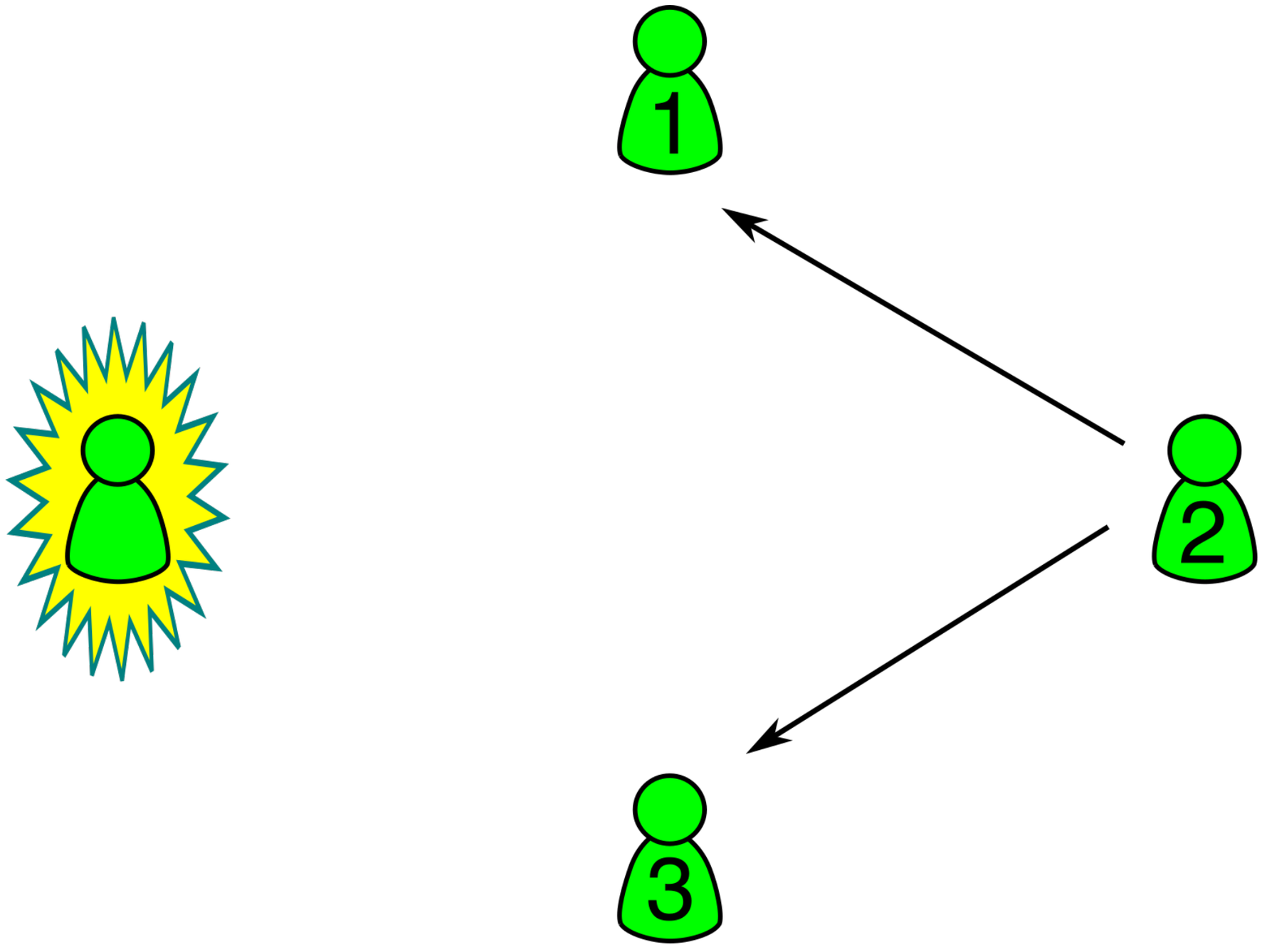
Example m=1, n=4

Step 2



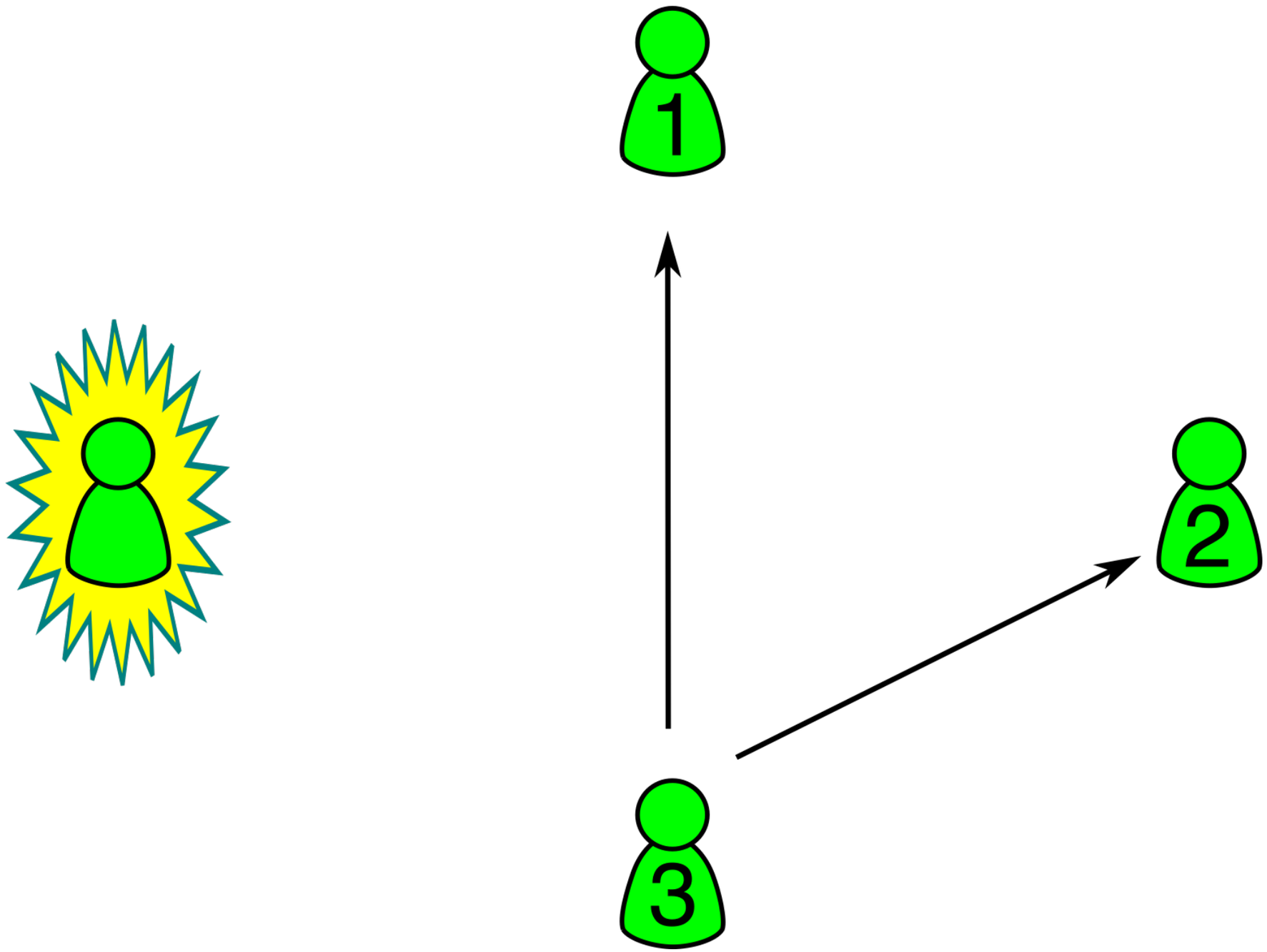
Example m=1, n=4

Step 2



Example m=1, n=4

Step 2



Example m=1, n=4

Step 3

Everybody selects the according to the majority

If no majority select default value, i.e. RETREAT

Example m=1, n=4
L1 is traitor

from

step 1	General	L1	L2	L3
L1	a			
L2	a			
L3	a			

to

Example m=1, n=4
L1 is traitor

from

to

step 2	General	L1	L2	L3
L1	a		a	a
L2	a	b		a
L3	a	b	a	

Example m=1, n=4
L1 is traitor

from

step 3	General	L1	L2	L3	Majority
L1	a		a	a	a
L2	a	b		a	a
L3	a	b	a		a

to

Example m=1, n=4
General is traitor

from

step 1	General	L1	L2	L3
L1	a			
L2	b			
L3	c			

to

Example m=1, n=4
General is traitor

from

step 2	General	L1	L2	L3
L1	a		b	c
L2	b	a		c
L3	c	a	b	

to

Example m=1, n=4
General is traitor

from

step 3	General	L1	L2	L3	Majority
L1	a		b	c	majority(a,b,c)
L2	b	a		c	majority(a,b,c)
L3	c	a	b		majority(a,b,c)

to

all lieutenants obtain the same value, majority(a,b,c) regardless of the actual values

Summary

- For 3 nodes, no solution exists that can tolerate one traitor
  - For <3m+1 nodes, no solution exists that can tolerate more than m traitors
    - i.e. at least 2/3rd of all nodes need to be loyal
  - Otherwise we can construct a solution for 3 nodes
- For 3m+1 nodes, algorithm OM(m) can tolerate m traitors

Summary

- The main difficulty is the lieutenants can forge messages
    - Easier if we can proof that a message has been signed

Cryptographic Signatures

- Each node has a public key (pk), known to everybody and a private (or secret) key (sk)

- A signature is a function that takes a message m and the secret key sk and produces a value $s=sign(m,sk)$
    - Everybody can quickly verify that the owner of sk signed the message m
    - Without knowledge of sk, s cannot be computed*

*(in a reasonable amount of time)

- Mathematical details on signatures next week

Byzantine Generals Problem with Signatures

- Every node knows the public key of every other node
- The general signs the messages with his private key
- Traitorous lieutenants cannot claim that the general send a different message
- Traitorous generals can do whatever they want, even allowing other traitorous lieutenants to forge it (collusion)

BGP with Signatures
Summary of conditions

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.

A4. (a) A loyal general's signature cannot be forged, and any alteration of the contents of his signed messages can be detected.

(b) Anyone can verify the authenticity of a general's signature.

Byzantine Generals Problem with Signatures

- Solution for m traitors and any number of generals
- nonsensical/trivial for <m+2 generals
  - only one loyal node, every other node is a traitor

Byzantine Generals Problem with Signatures

- notation
  - $m{:}i$ message m signed by general i
  - $m{:}i{:}j{:}k$
    - message m signed by general i
      - statement "$m{:}i$" signed by j
        - statement "$m{:}i{:}j$" signed by k