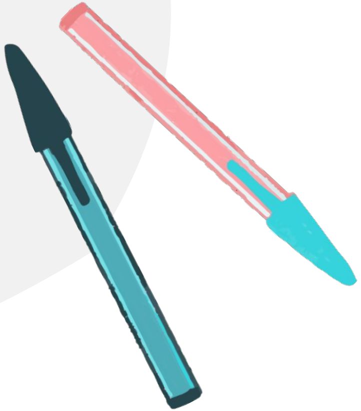
Was Leslie Lamport Right?



Nic Jackson

Developer Advocate @HashiCorp

Member #2 of the Leslie Lamport fanclub

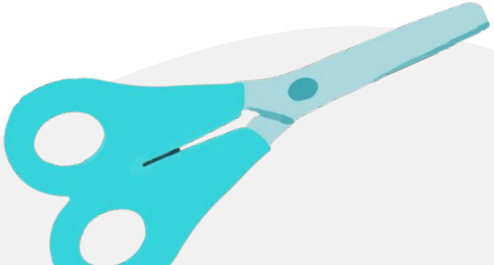
Sarah Christoff

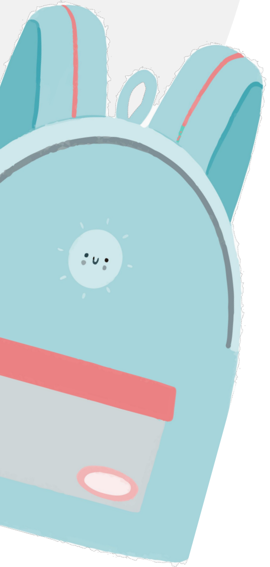
Staff Software Engineer @Edera

Founder & Member #1 of the Leslie Lamport fanclub

Agenda

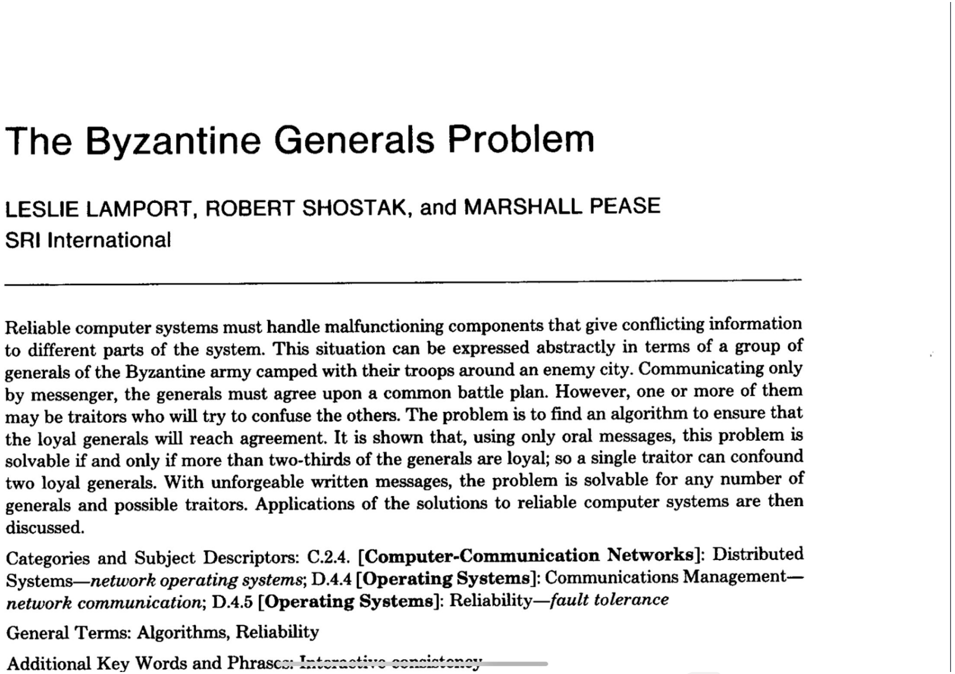
- A little history lesson
- Consensus
- Consistency
- Concurrency
- Clocks
- Gossip





01
Introduction

History



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.

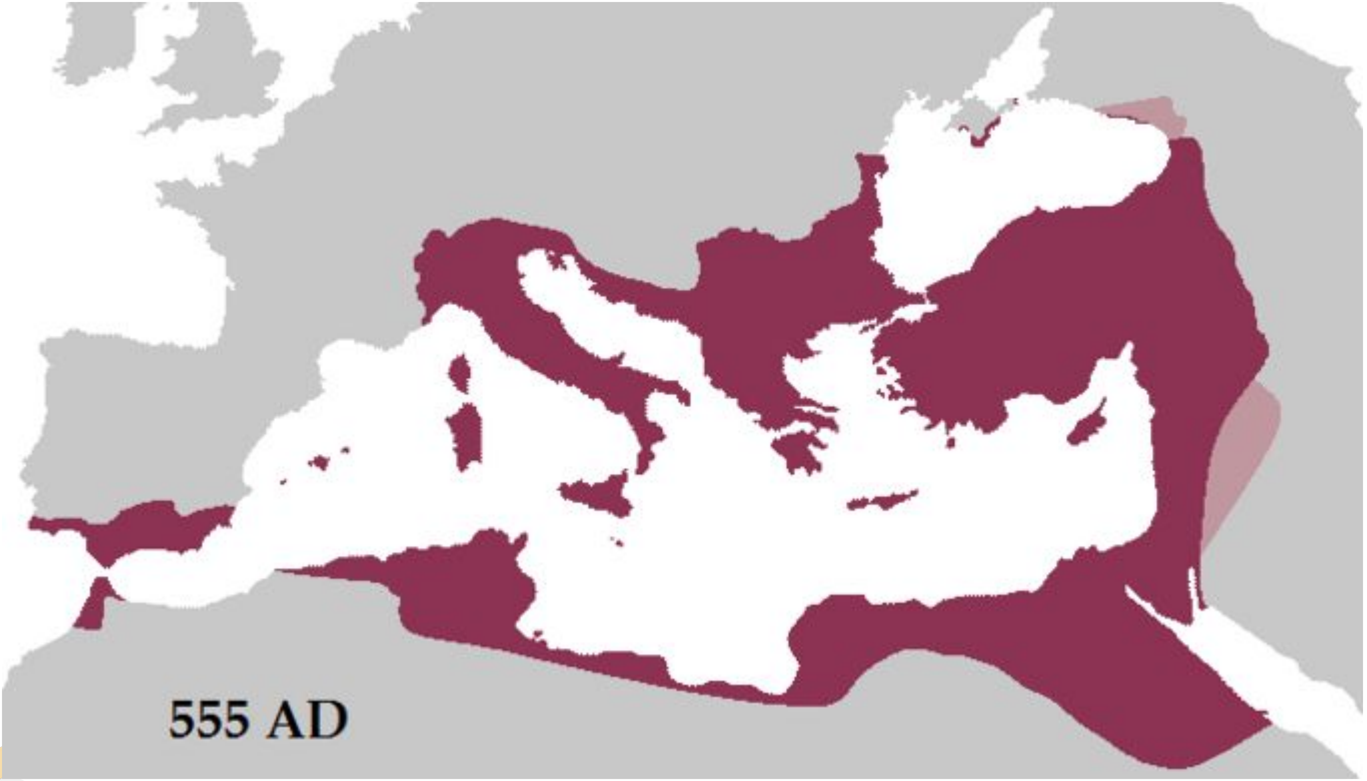
Categories and Subject Descriptors: C.2.4. [**Computer-Communication Networks**]: Distributed Systems—*network operating systems*; D.4.4 [**Operating Systems**]: Communications Management—*network communication*; D.4.5 [**Operating Systems**]: Reliability—*fault tolerance*

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Interactive consistency

Emperor Constantine

Byzantine Empire



Our Cast

Emperor Romanos

In 1034 Emperor Romanos III becomes ill

Empress Zoe

Wife of Romanos and allegedly responsible for poisoning him

Michael the Money Changer

Lover of Zoe brother of John

John the Eunuch

Brother of Michael

George Manilakes

The most feared general in the Byzantine Army

Harald Sigurdsson

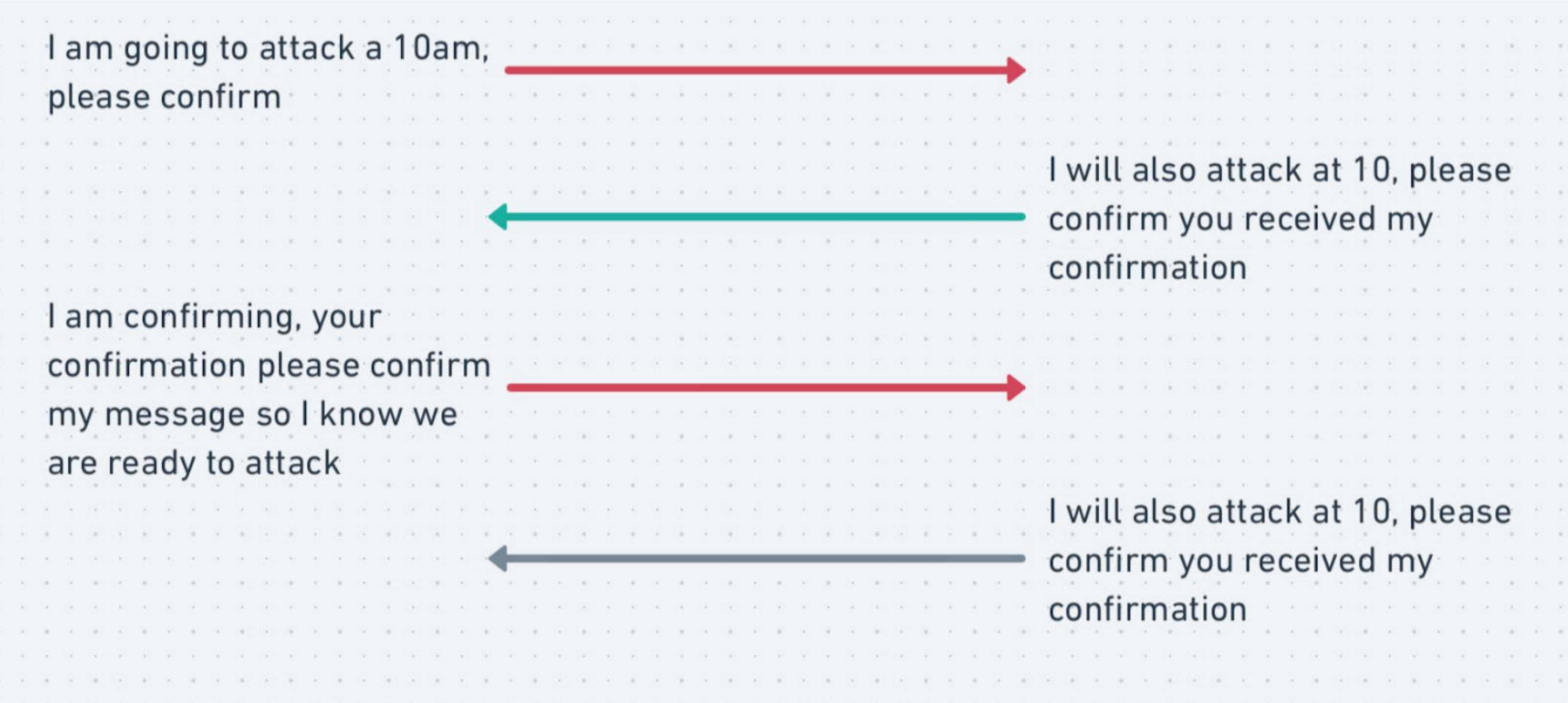
Mercenary soldier, future king of Denmark

The Two Generals Problem

The Two Generals Problem

- If only one general attacks they will be defeated
- If none of the generals attack, they both live to fight another day
- If both generals attack they take the city

The Two Generals Problem



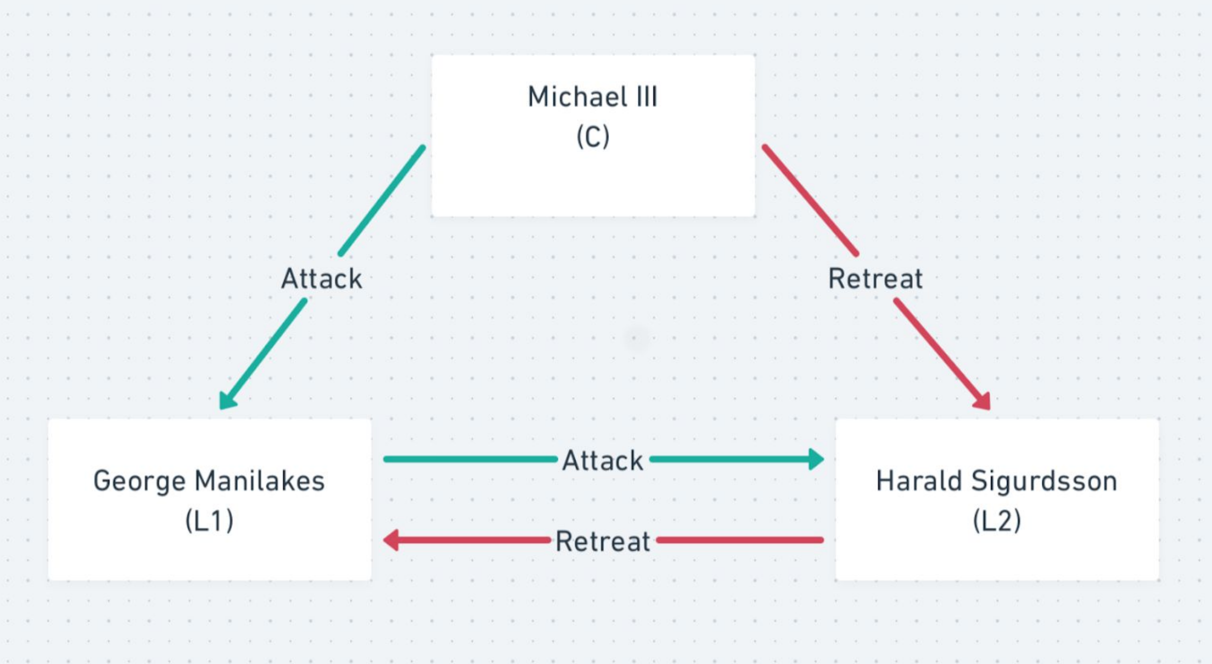
I am going to attack a 10am, please confirm

I will also attack at 10, please confirm you received my confirmation

I am confirming, your confirmation please confirm my message so I know we are ready to attack

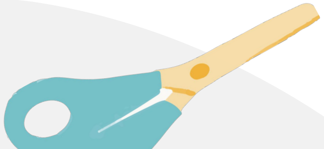
I will also attack at 10, please confirm you received my confirmation

The Byzantine Generals Problem

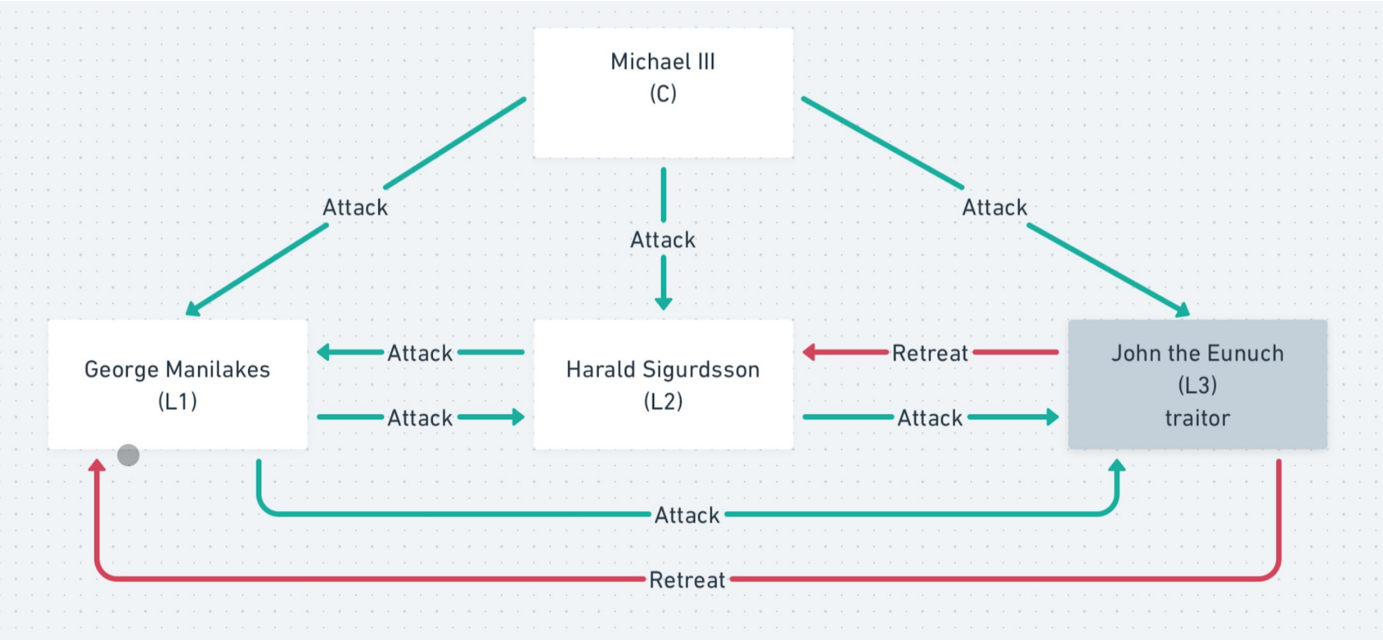


The Byzantine Generals Problem

General	Messages	Outcome
Commander	L1: Attack L2: Attack	
Lieutenant 1	C: Attack L2: Retreat	Inconclusive
Lieutenant 2	C: Retreat: L2 Attack	Inconclusive

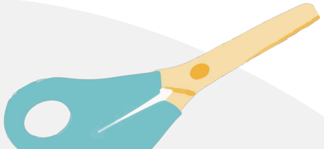


The Byzantine Generals Problem

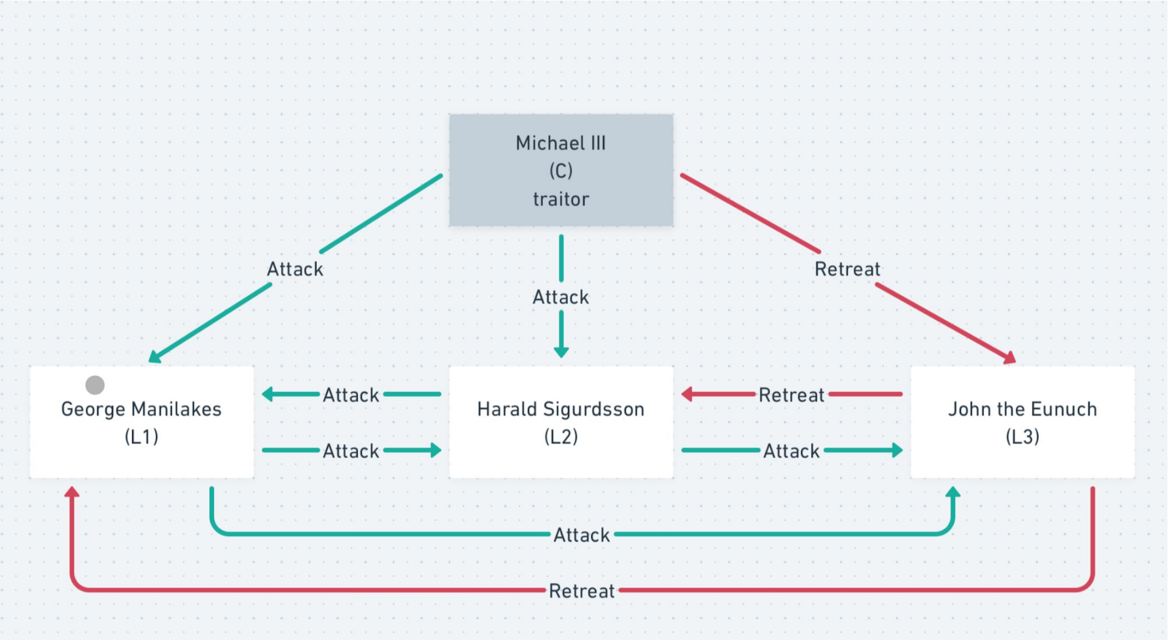


The Byzantine Generals Problem

General	Messages	Outcome
Commander	L1: Attack L2: Attack L3: Attack	Attack
Lieutenant 1	C: Attack L2: Attack L3: Retreat	Attack
Lieutenant 2	C: Attack: L2 Attack L3: Retreat	Attack
Lieutenant 3 T	C: Attack L1: Attack L2: Attack	

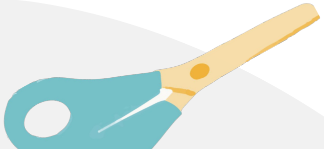


The Byzantine Generals Problem



The Byzantine Generals Problem

General	Messages	Outcome
Commander T	L1: Attack L2: Attack L3: Retreat	
Lieutenant 1	C: Attack L2: Attack L3: Retreat	Attack
Lieutenant 2	C: Attack: L2 Attack L3: Retreat	Attack
Lieutenant 3	C: Retreat L1: Attack L2: Attack	Attack



Step 1

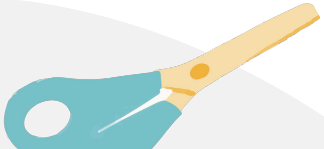
To solve the problem Leslie Lamport proposes that the formula to achieve consensus is as follows

$$3m+1$$

Where m is the number of traitors

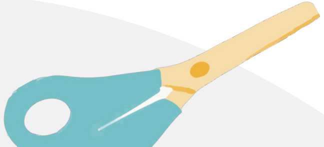
Step 1: Required Generals

Traitors	Generals
1	4
2	7
3	10
4	13



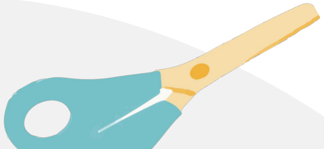
6 Generals

General	Messages	Outcome
Commander	L1: Attack L2: Attack L3: Attack L4: Attack L5: Attack	Attack
Lieutenant 1	C: Attack L2: Attack L3: Retreat L4: Retreat L5: Attack	Attack
Lieutenant 2	C: Attack: L2 Attack L3: Retreat L4: Retreat L5: Attack	Attack
Lieutenant 3 T	C: Attack L1: Attack L2: Attack L4: Retreat L5: Attack	
Lieutenant 4 T	C: Attack L1: Attack L2: Attack L3: Retreat L5: Attack	
Lieutenant 5	C: Attack L1: Attack L2: Attack L3: Retreat L4: Retreat	Attack



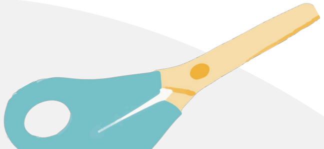
6 Generals

General	Messages	Outcome
Commander T	L1: Attack L2: Retreat L3: Retreat L4: Attack L5: Attack	
Lieutenant 1	C: Attack L2: Retreat L3: Retreat L4: Retreat L5: Attack	Retreat
Lieutenant 2	C: Retreat: L1 Attack L3: Retreat L4: Retreat L5: Attack	Attack
Lieutenant 3	C: Retreat L1: Attack L2: Retreat L4: Attack L5: Attack	Attack
Lieutenant 4 T	C: Attack L1: Attack L2: Attack L3: Attack L5: Attack	
Lieutenant 5	C: Attack L1: Attack L2: Retreat L3: Retreat L4: Retreat	Retreat



7 Generals - Round 2

General	Messages	Outcome
Commander T	L1: Attack L2: Retreat L3: Retreat L4: Attack L5: Attack L6: Retreat	
Lieutenant 1	C: Attack L2: Retreat L3: Retreat L4: Retreat L5: Attack L6: Attack	Retreat
Lieutenant 2	C: Retreat: L1 Attack L3: Retreat L4: Retreat L5: Attack L6: Attack	Retreat
Lieutenant 3	C: Retreat L1: Attack L2: Retreat L4: Attack L5: Attack L6: Attack	Attack
Lieutenant 4 T	C: Attack L1: Attack L2: Attack L3: Attack L5: Attack L6: Attack	
Lieutenant 5	C: Attack L1: Attack L2: Retreat L3: Retreat L4: Retreat L6: Attack	Retreat
Lieutenant 6	C: Retreat L1: Attack L2: Retreat L3: Retreat L4: Attack L5: Attack	Retreat



Step 2

To identify invalid data a number of voting rounds are needed

$$t+1$$

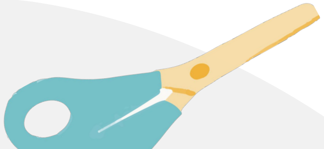
Where *t* is the number of traitors

Example of message rounds with 2 traitors

1. Every lieutenant sends their received value to every other lieutenant
2. The lieutenant attempts to find an outcome and then sends this result to the other lieutenants
3. If lieutenant uses this data to try find an outcome

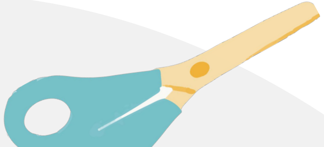
7 Generals - Round 3

General	Messages	Outcome
Lieutenant 1	C: Attack L2: Retreat L3: Retreat L4: Retreat L5: Attack L6: Retreat	Retreat
Lieutenant 2	C: Retreat: L1 Attack L3: Retreat L4: Retreat L5: Attack L6: Attack	Retreat
Lieutenant 3	C: Retreat L1: Attack L2: Retreat L4: Attack L5: Attack L6: Attack	Attack
Lieutenant 4 T	C: Attack L1: Attack L2: Attack L3: Attack L5: Attack L6: Retreat	Attack
Lieutenant 5	C: Attack L1: Attack L2: Retreat L3: Retreat L4: Retreat L6: Retreat	Retreat
Lieutenant 6	C: Attack L1: Attack L2: Retreat L3: Retreat L4 Retreat L5: Attack	Retreat



6 Generals - Round 3

General	Messages	Outcome
Lieutenant 1	C: Attack L2: Retreat L3: Retreat L4: Retreat L5: Attack	Retreat
Lieutenant 2	C: Retreat: L1 Attack L3: Retreat L4: Retreat L5: Attack	Attack
Lieutenant 3	C: Retreat L1: Attack L2: Retreat L4: Attack L5: Attack	Attack
Lieutenant 4 T	C: Attack L1: Attack L2: Attack L3: Attack L5: Attack	Attack
Lieutenant 5	C: Attack L1: Attack L2: Retreat L3: Retreat L4: Retreat	Retreat



Solution

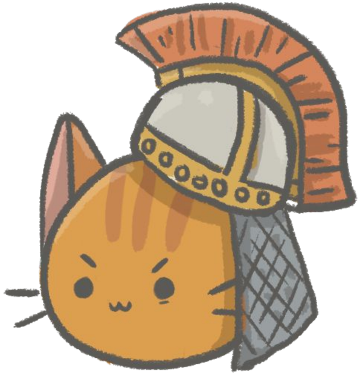
3m+1 generals + t+1 voting rounds

Where *m* and *t* are the number of traitors

Consistency

The best sieges are built on open communication

John the Eunuch



Michael

Message Consistency
(Data Consistency)

Soldier Availability
(System Availability)

Tolerance to bad actors
(Tolerance to network partitions)

Eventual

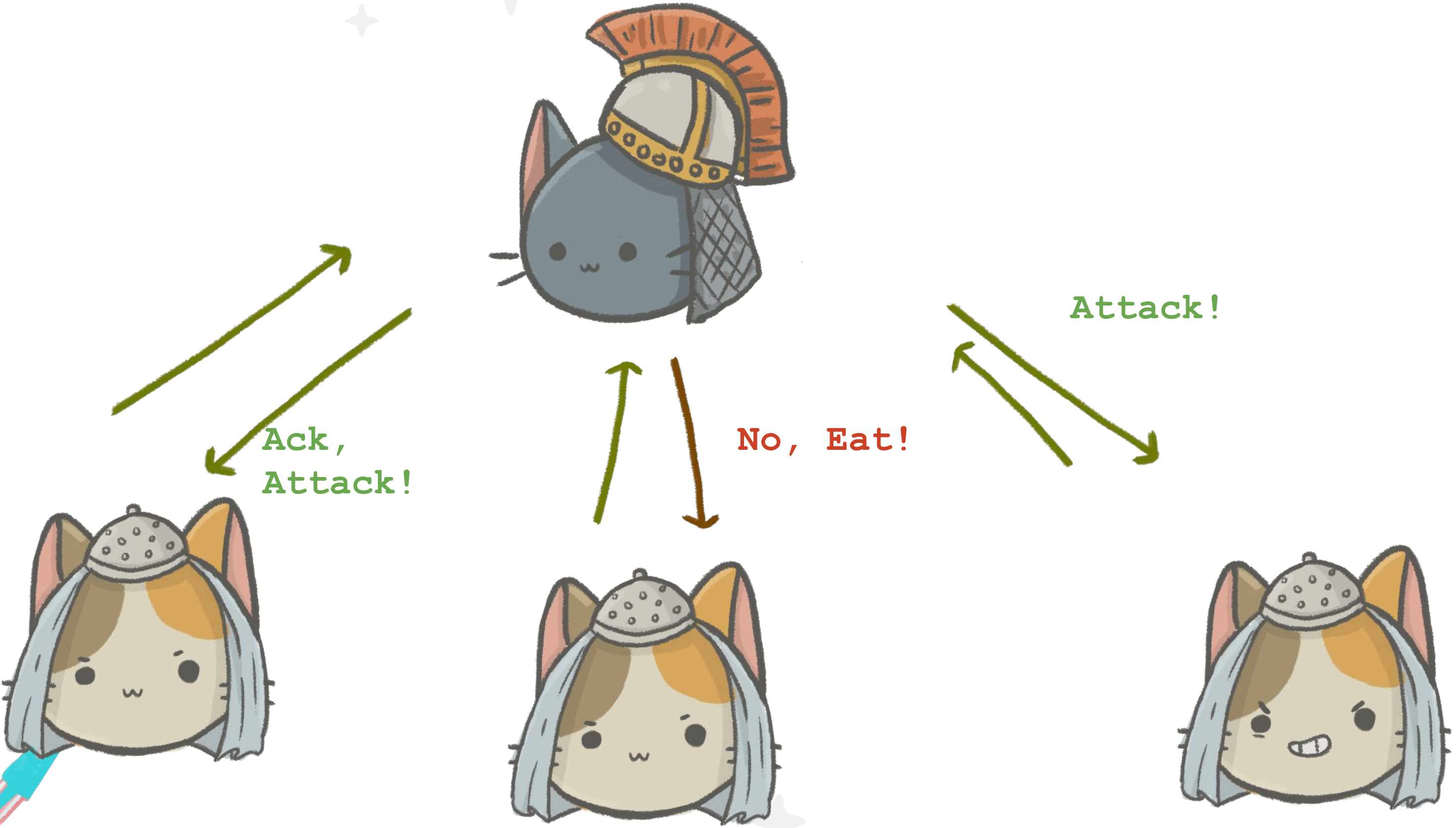
Weak

Strong

Sequential

John the Eunuch

Eventual



Ack, Attack!

No, Eat!

Attack!

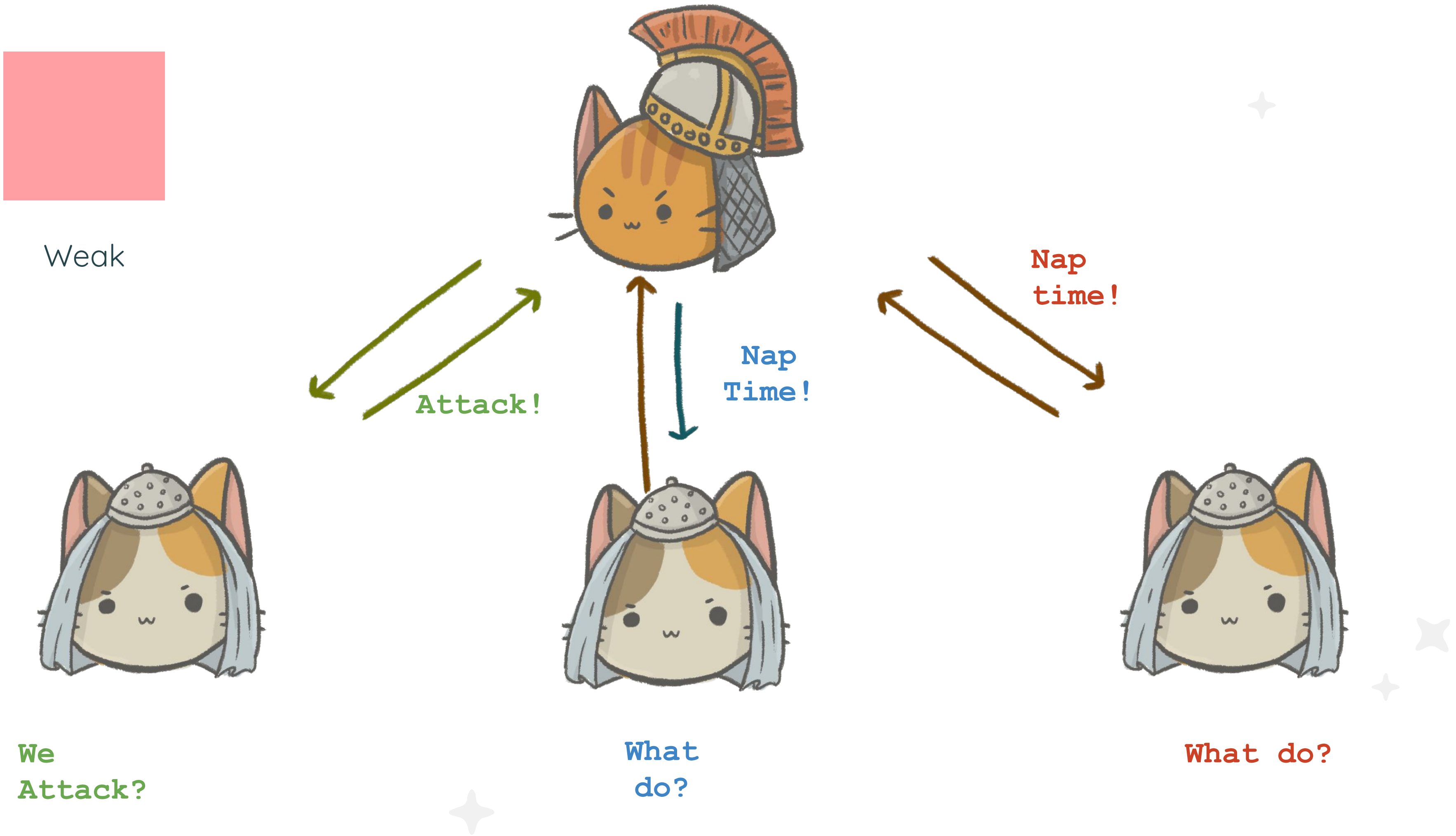
We Attack!

We Attack?

We Attack?

🟥

Weak



Attack!

Nap
Time!

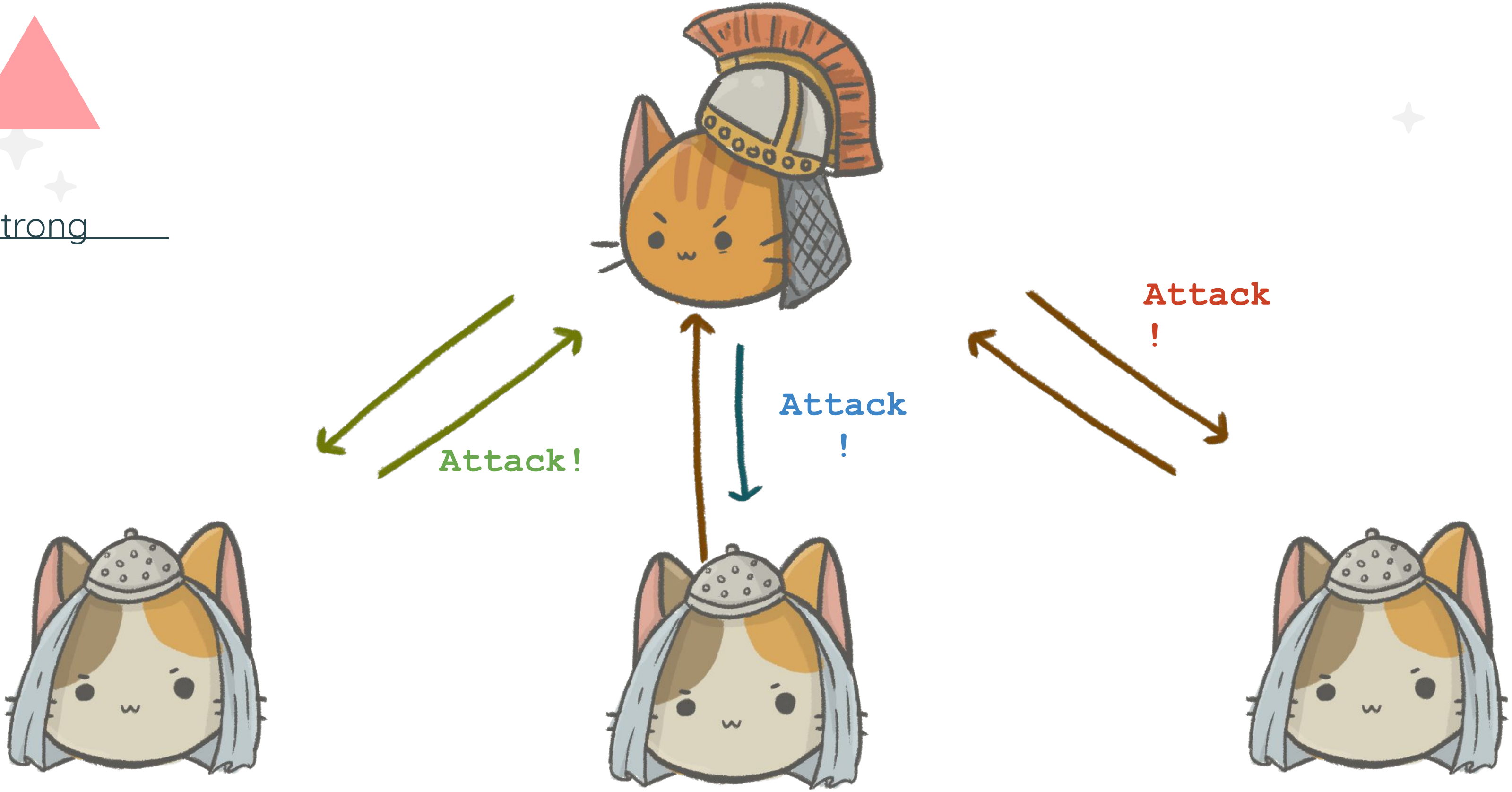
Nap
time!

We
Attack?

What
do?

What do?

Strong



Attack
!

Attack
!

Attack!

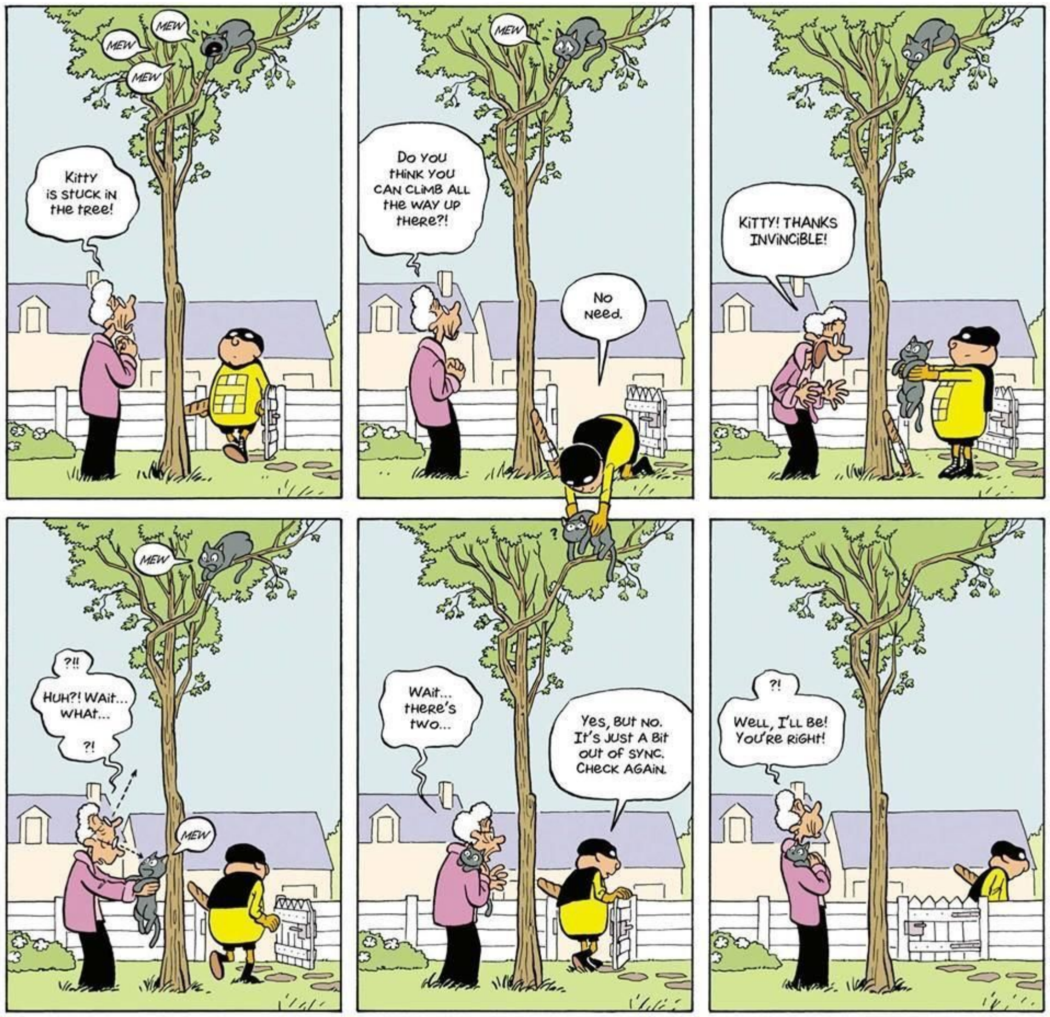
We
Attack?

What
do?

What do?

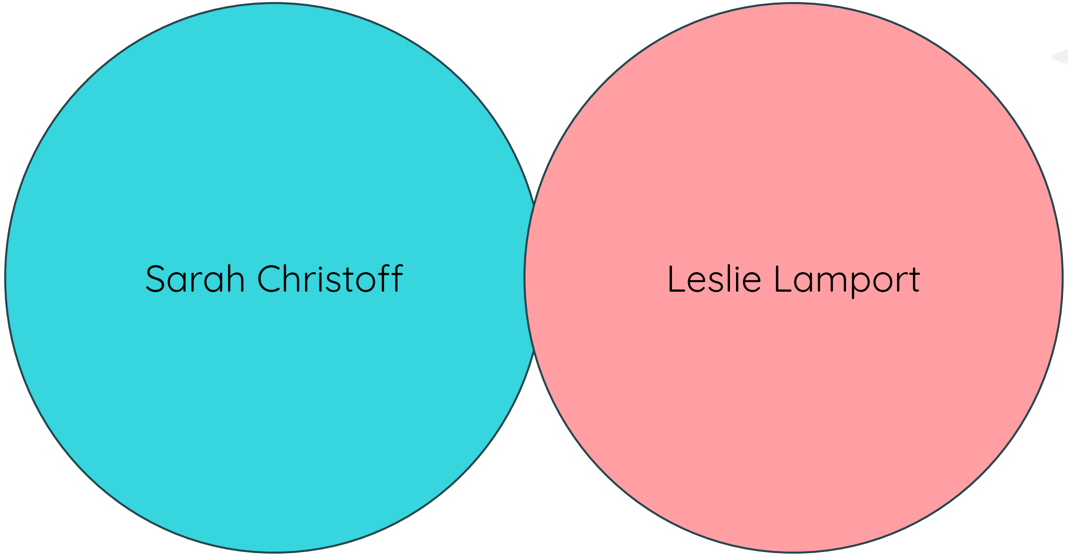
Strong

Sequential


MEW
MEW
MEW
Kitty is stuck in the tree!
MEW
Do you think you can climb all the way up there?!
No need.
Kitty! Thanks Invincible!
MEW
?!!
Huh?! Wait... what...
?!
MEW
?
Wait... there's two...
Yes, but no. It's just a bit out of sync. Check again.
?!
Well, I'll be! You're right!

Concurrency

Things in Common



Sarah Christoff

Leslie Lamport

What is a computation?

What is a computation?

A computation is a sequence of steps.

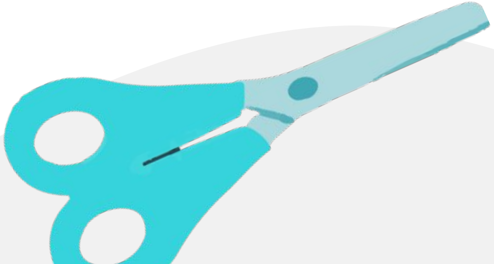
What is a step?

What is a step?

A transition from one state to a next

HEIDELBERG
LAUREATE
FORUM
Leslie

Invariance



Base = n = 1

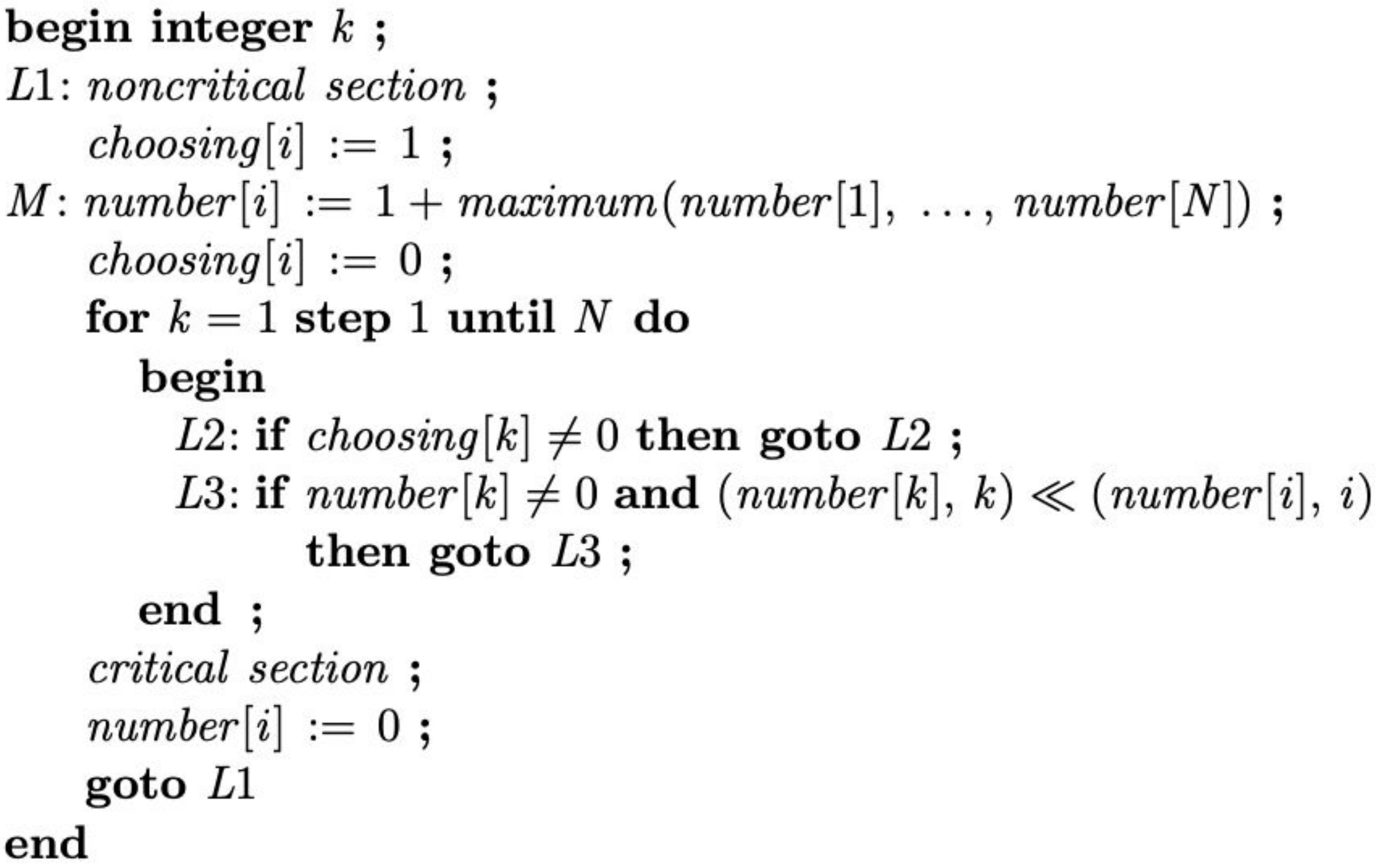
Inductive = n = 1 & n + 1

Once an engineer understands what a computation is and how it is described, she can understand the most important concept in concurrency: invariance.

- Leslie Lamport, Teaching Concurrency

```
“integer $j$;
$Li0$:  $b[i] :=$ false;
$Li1$:  if $k \neq i$ then
$Li2$:  begin $c[i] :=$ true;
$Li3$:  if $b[k]$ then $k := i$;
        go to $Li1$
        end
          else
$Li4$:  begin $c[i] :=$ false;
          for $j := 1$ step 1 until $N$ do
            if $j \neq i$ and not $c[j]$ then go to $Li1$
        end;
        critical section;
        $c[i] :=$ true;  $b[i] :=$ true;
        remainder of the cycle in which stopping is allowed;
        go to $Li0$”
```

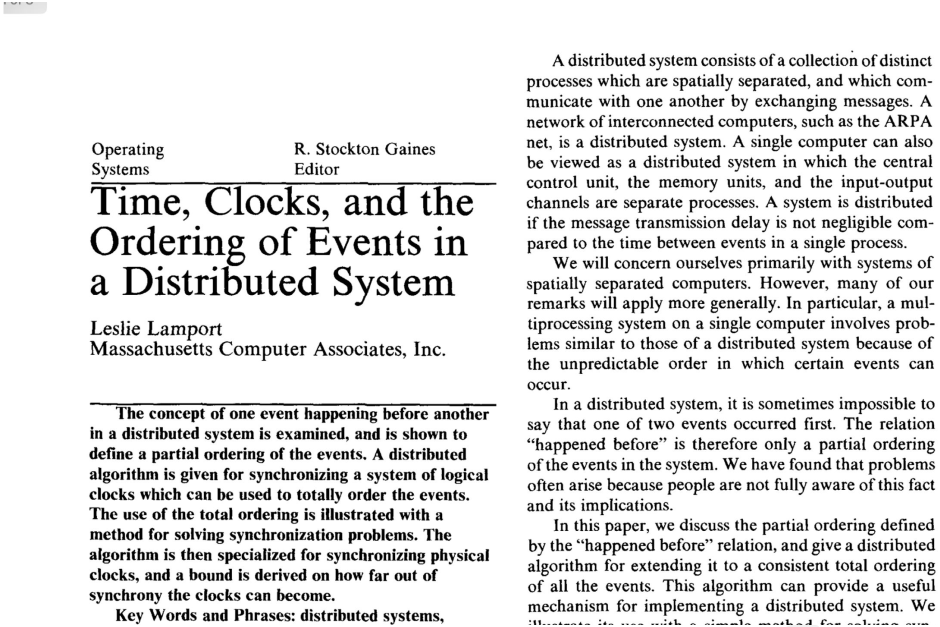
PLEASE
Take A
Number
NOW SERVING
456
855
854



```
begin integer $k$ ;
$L1$: $noncritical\ section$ ;
    $choosing[i] := 1$ ;
$M$: $number[i] := 1 + maximum(number[1], \ldots, number[N])$ ;
    $choosing[i] := 0$ ;
    for $k = 1$ step $1$ until $N$ do
       begin
         $L2$: if $choosing[k] \neq 0$ then goto $L2$ ;
         $L3$: if $number[k] \neq 0$ and $(number[k], k) \ll (number[i], i)$
                   then goto $L3$ ;
       end ;
    $critical\ section$ ;
    $number[i] := 0$ ;
    goto $L1$
end
```

Lamport Clocks

Time Clocks, and the Ordering of Events in a Distributed System



Operating Systems　　R. Stockton Gaines Editor

Time, Clocks, and the Ordering of Events in a Distributed System

Leslie Lamport
Massachusetts Computer Associates, Inc.

The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

Key Words and Phrases: distributed systems,

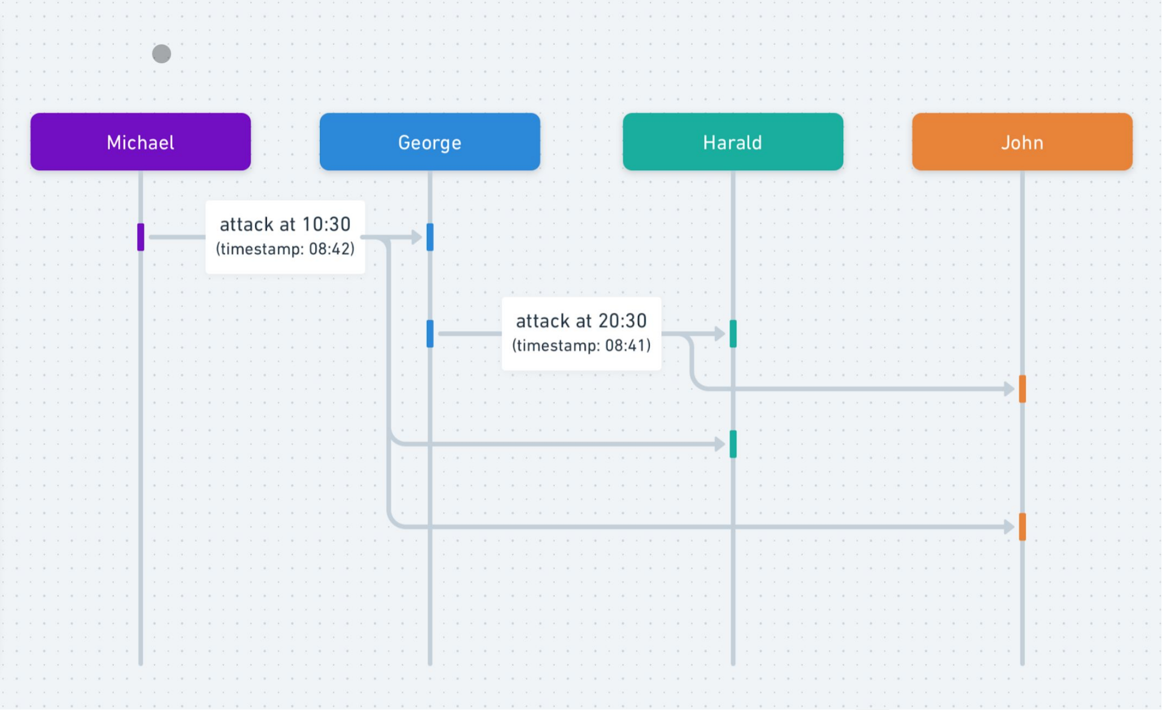
A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages. A network of interconnected computers, such as the ARPA net, is a distributed system. A single computer can also be viewed as a distributed system in which the central control unit, the memory units, and the input-output channels are separate processes. A system is distributed if the message transmission delay is not negligible compared to the time between events in a single process.

We will concern ourselves primarily with systems of spatially separated computers. However, many of our remarks will apply more generally. In particular, a multiprocessing system on a single computer involves problems similar to those of a distributed system because of the unpredictable order in which certain events can occur.

In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation "happened before" is therefore only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.

In this paper, we discuss the partial ordering defined by the "happened before" relation, and give a distributed algorithm for extending it to a consistent total ordering of all the events. This algorithm can provide a useful mechanism for implementing a distributed system. We illustrate its use with a simple method for solving syn-

Lamport Clocks



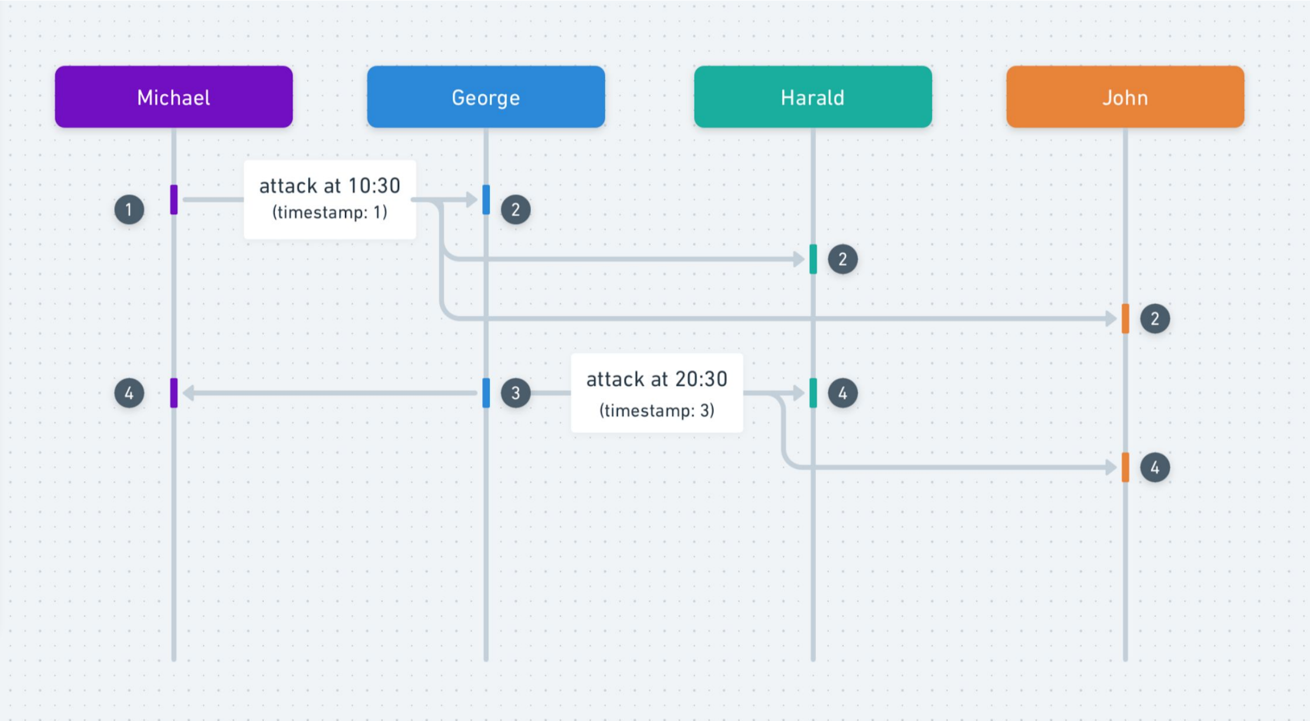
Lamport Clocks

- Every General has a Logical counter ***t***
- Every time an event occurs ***t := t+1***
- Every time a message is sent ***t := t+1,*** **t** *is also attached to the message*
- *When a message is received if* ***t` > t, t:= t`+1***

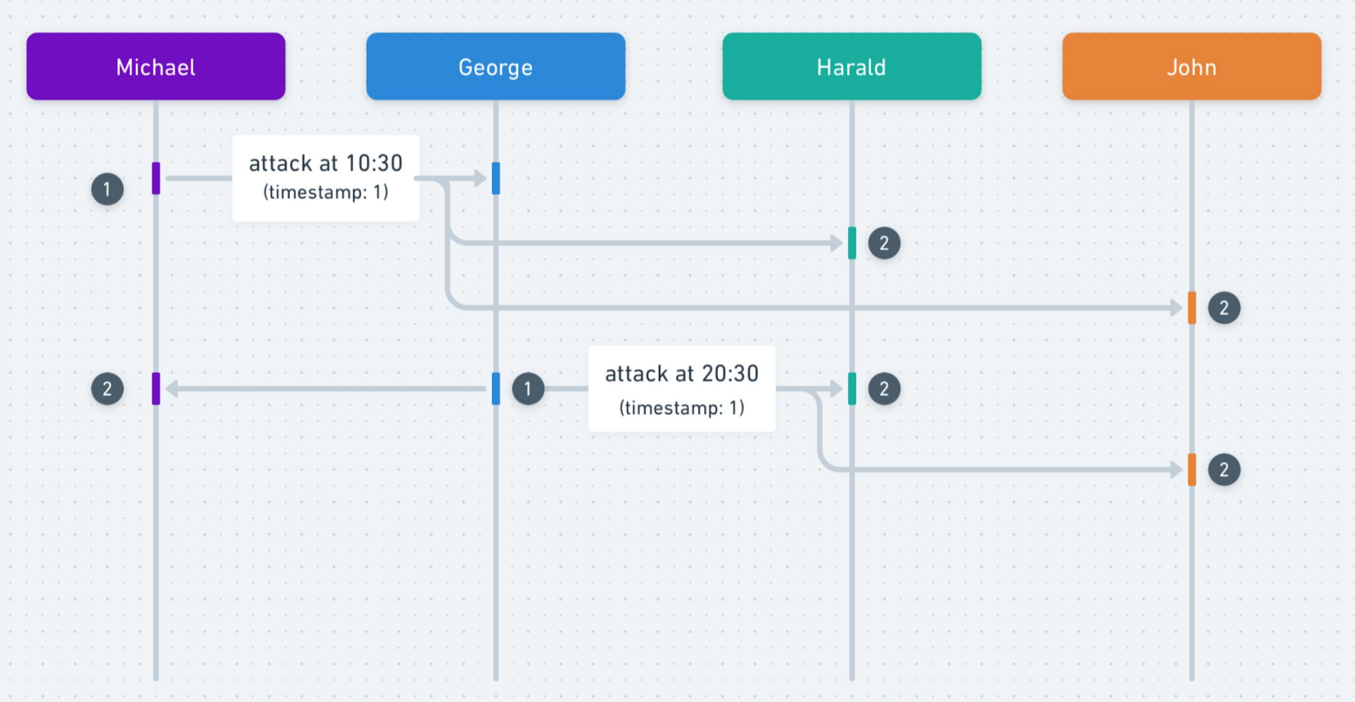
Lamport Clocks

- Given event ***a*** happens before event ***b*** then the Lamport clock for ***a*** will be ***less*** than the Lamport clock for ***b***
- If a Lamport clock for ***a*** is ***less*** than the Lamport clock for ***b*** it does not guarantee the event happened before the other, both events could be concurrent
- It is possible to have two events with the same timestamp

Lamport Clocks



Lamport Clocks: Concurrency



Lamport Clocks: Tiebreaks

- Given two messages with the same timestamp a tiebreak must be used to determine order
- Tiebreak can use any algorithm, common approach is to using a lexicographical order on node name or a shared and pre-defined order

Vector Clocks

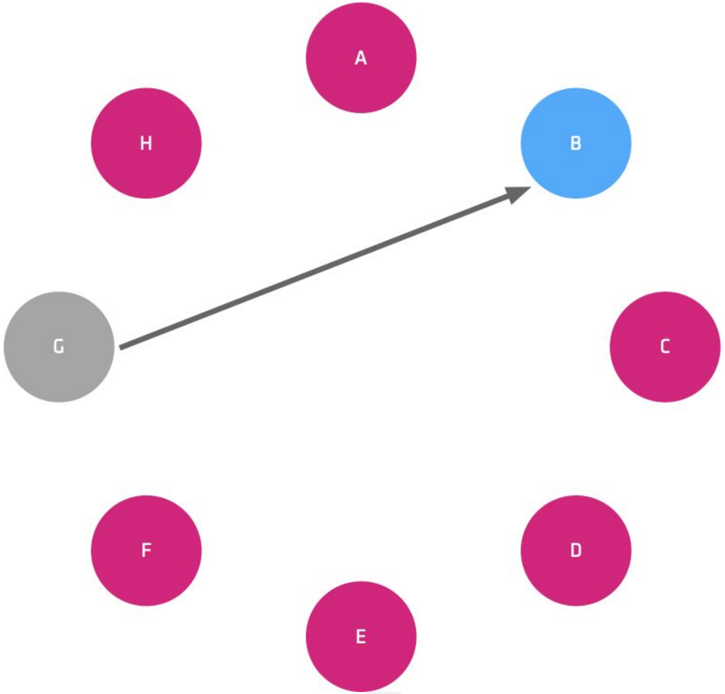
- If a Lamport clock for ***a*** is ***less*** than the Lamport clock for ***b*** it does not guarantee the event happened before the other, both events could be concurrent
- Vector clocks function in a similar way to Lamport clocks except rather than a single scalar value for the timestamp they use a vector of scalar values, with one dimension per node i.e <1,2,3,0>
- Comparing vectors allows concurrency to be identified

Gossip Protocols

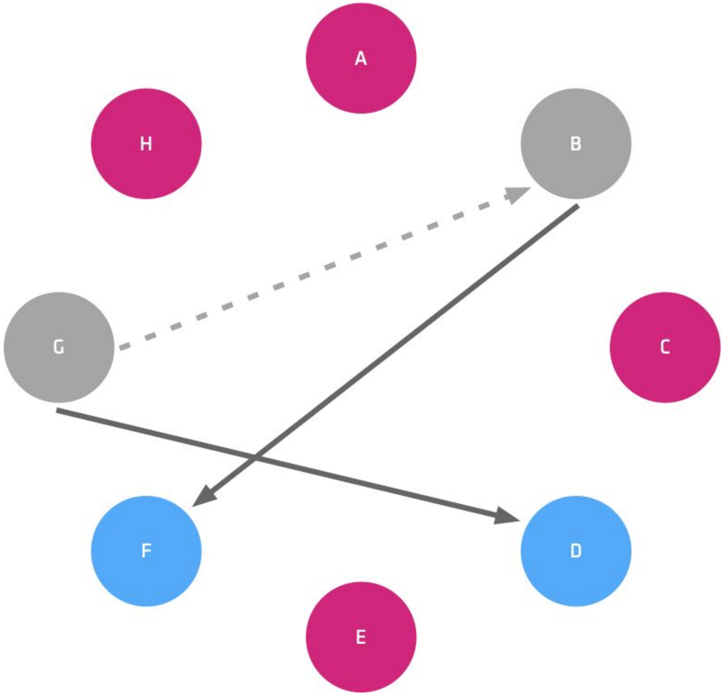
(Slightly Lamport, but more fun!)

A LITT
BIT
DRAMA
TIC

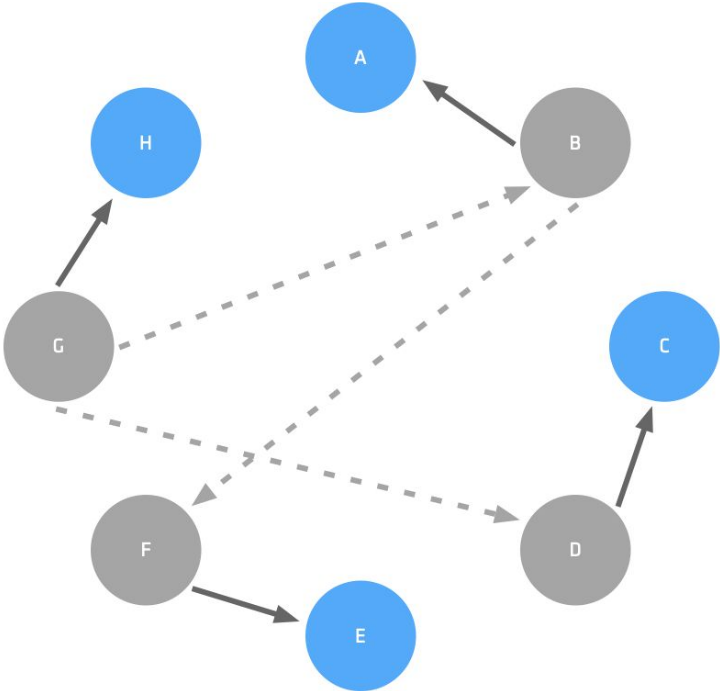
How Gossip Works: Round 1



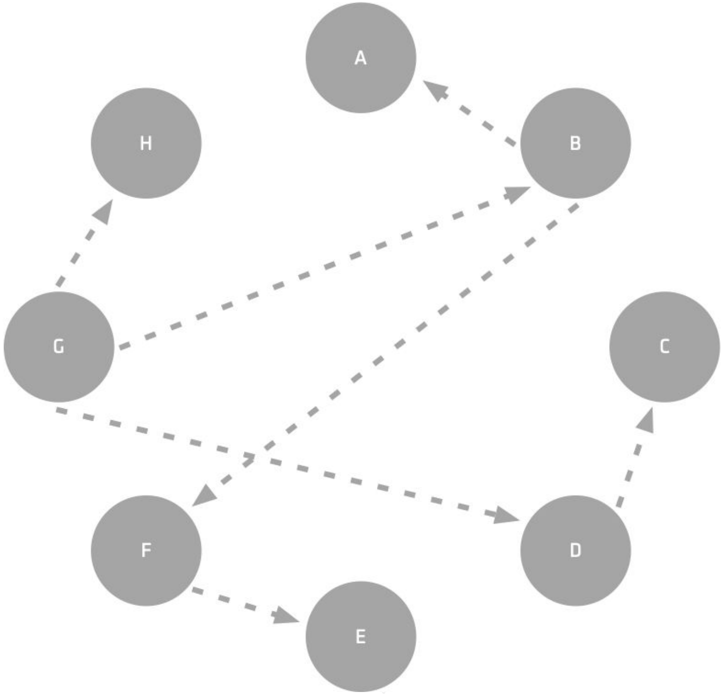
How Gossip Works: Round 2

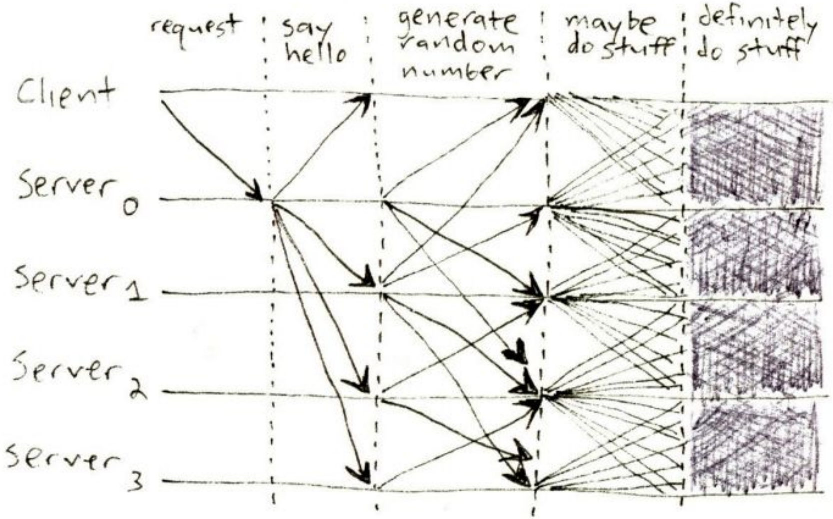


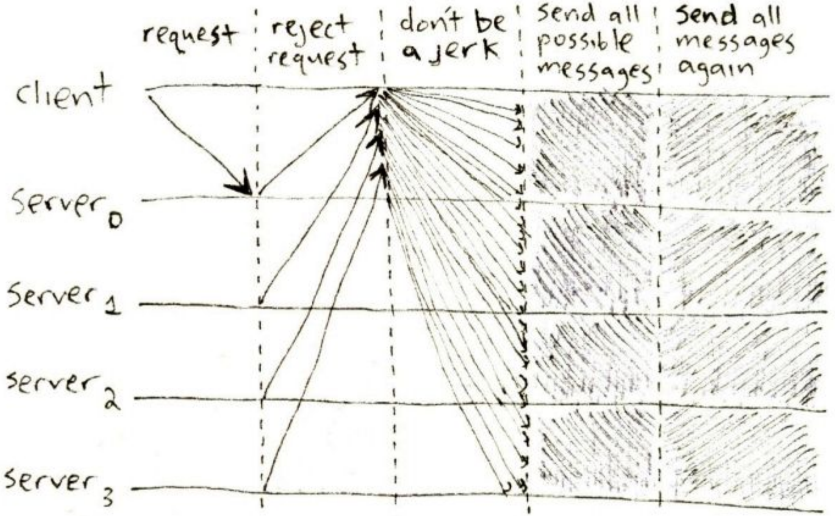
How Gossip Works: Round 3

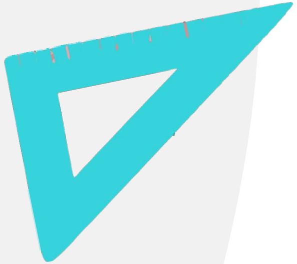


How Gossip Works: Round 4



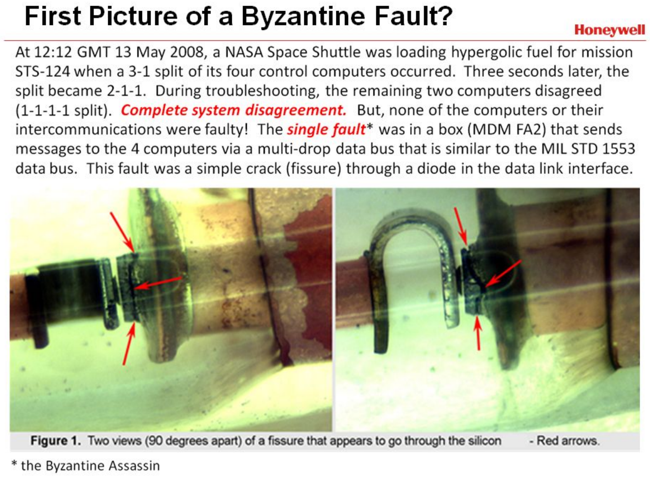

request
say hello
generate random number
maybe do stuff
definitely do stuff
Client
Server 0
Server 1
Server 2
Server 3


request
reject request
don't be a jerk
send all possible messages
send all messages again
client
server 0
server 1
server 2
server 3



Conclusion

- Robinhood Outage in 2020 was caused by *someone* who removed a statement that updated Lamport Clocks
- CloudFlare Byzantine Failure



First Picture of a Byzantine Fault?

At 12:12 GMT 13 May 2008, a NASA Space Shuttle was loading hypergolic fuel for mission STS-124 when a 3-1 split of its four control computers occurred. Three seconds later, the split became 2-1-1. During troubleshooting, the remaining two computers disagreed (1-1-1-1 split). ***Complete system disagreement.*** But, none of the computers or their intercommunications were faulty! The ***single fault**** was in a box (MDM FA2) that sends messages to the 4 computers via a multi-drop data bus that is similar to the MIL STD 1553 data bus. This fault was a simple crack (fissure) through a diode in the data link interface.

Figure 1. Two views (90 degrees apart) of a fissure that appears to go through the silicon - Red arrows.

* the Byzantine Assassin

4

Yes.

Thank You

Nic Jackson

@sheriffjackson.bsky.social

linkedin.com/in/jacksonnic

Sarah Christoff

@schristoff.bsky.social

linkedin.com/in/schristoff

References

- Leslie Lamport
  - The Byzantine General Problem
  - How to make a Multiprocessor Computer That Correctly Executes Multiprocess Programs
  - Teaching Concurrency
  - Deconstructing the Bakery to Build a Distributed State Machine
- E. W. Dijkstra
  - Solution of a problem in concurrent programming control