“I didn’t order that!”

Demystifying the Technology that Supports Online Services

Olivier Marin
Department of Computer Science & Engineering
https://wp.nyu.edu/omarin/

When Things Go Haywire

2018 - Takeaway delivery service (UK) [2]

2020 - Caucus results reporting app (Iowa, USA) [1]

2022 - Pudong PCR test collection on 健康云?

The Usual Suspect

The Usual Culprits

What Computer Science is *not*

Computer Science (CS) is ***not*** the study of computers and programs

Astronomy isn't the study of telescopes
Biology isn't the study of microscopes
Chemistry isn't the study of test tubes

Science is not about tools…

What *is* Computer Science?

Computer Science is the study of *algorithms*

An algorithm is a precisely defined process for solving a problem

Usually in a way that can be automated easily

Formal properties	*Correctness, complexity*
Implementations	*Architecture, programming*
Applications	*Artificial intelligence, network design, bioinformatics, ...*

Boosting the Power Level Over 9000!!!

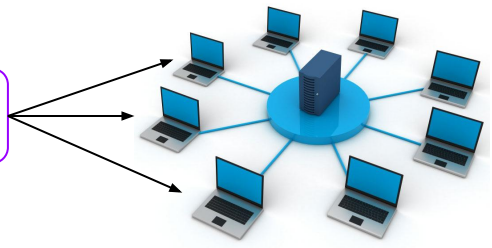
Many disciplines apply the same algorithms to massive amounts of data

Finance, physics, mathematics, …

Computational tools work on a limited scale

Distributing a computation can increase its performance at a very large scale

```
For i = 1 to N
    For j = 1 to N
        #HEAVY-COMPUTATION
```



Distributed Algorithms / Systems

User perspective

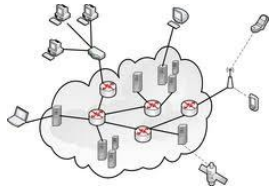
"*A distributed system is a collection of independent computers that appears to its users as a single coherent system.*" (Tanenbaum)





Operational perspective

"*A distributed system is one in which hardware or software components, located at networked computers, communicate and coordinate their actions only by passing messages.*" (Coulouris)

Coordination

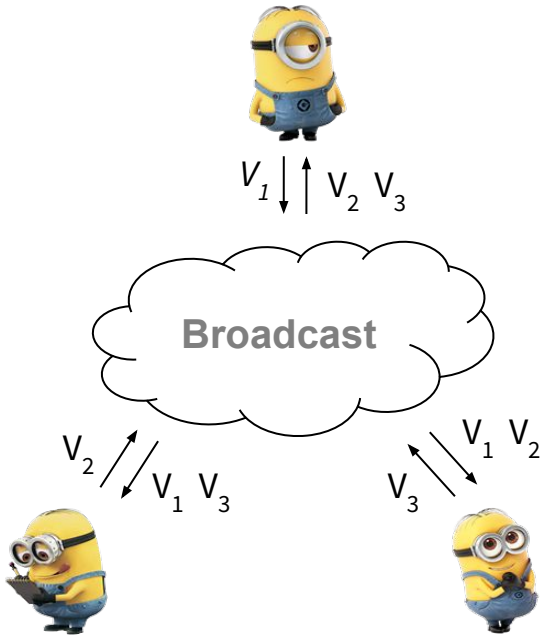
Distributing a computation means that everyone pitches in

Every computer node agrees on:

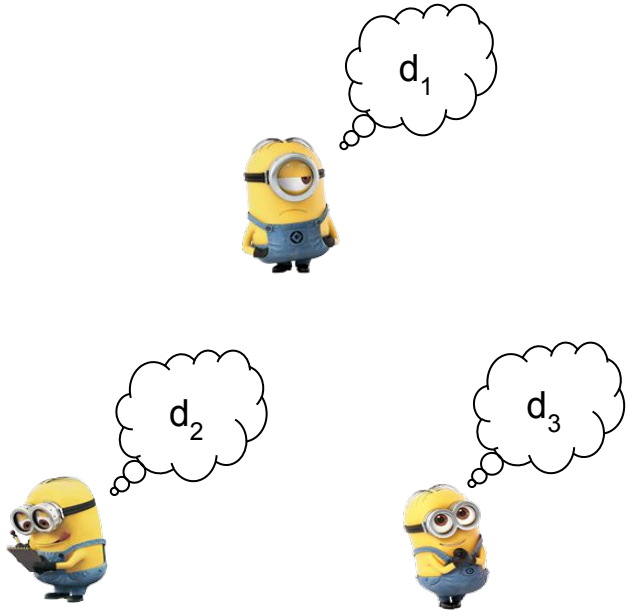
- the common goal
- how to get there (algorithm)
- the distribution of tasks

As a rule, distribution remains transparent to the user

Anatomy of a Consensus



Step 1: Propose



Step 2: Decide

Properties of a Consensus

Termination

All processes eventually decide on a result ***D***

Validity

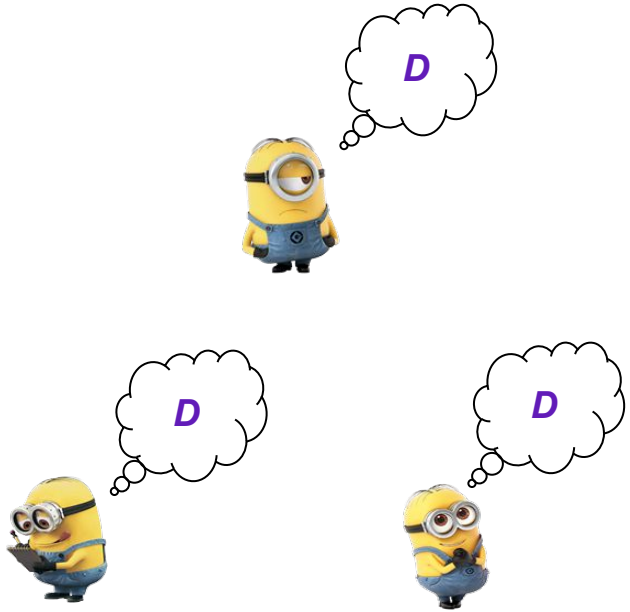
D is a proposed value

Agreement

D is the same for all processes

Integrity

Once a process decides ***D***, it cannot switch to D'



Scalability vs. Fault Tolerance

Higher computing power

More resources available

Big applications

Big data

[Grid/Cloud/Fog] computing

Higher complexity

More failures

The Tale of the Intrepid System Designer

Common fallacies

✗ Everything goes fast

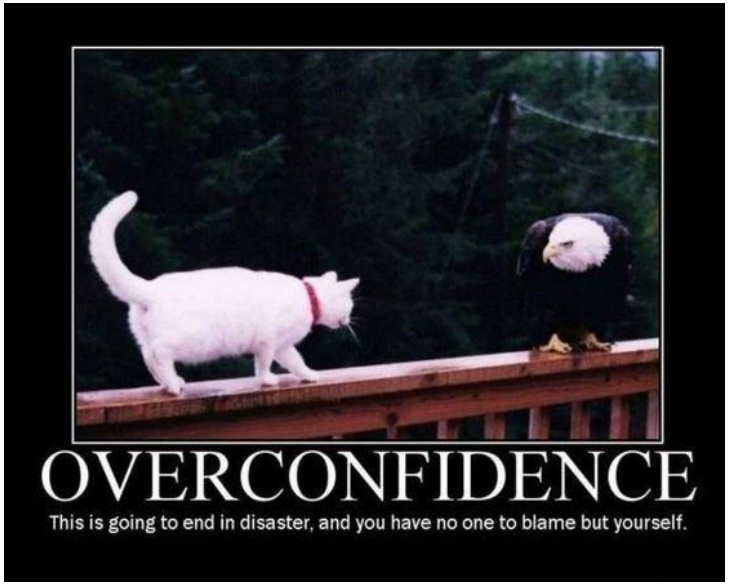
Latency, bandwidth, overhead

✗ Everything is stable and consistent

Topology, hardware, software

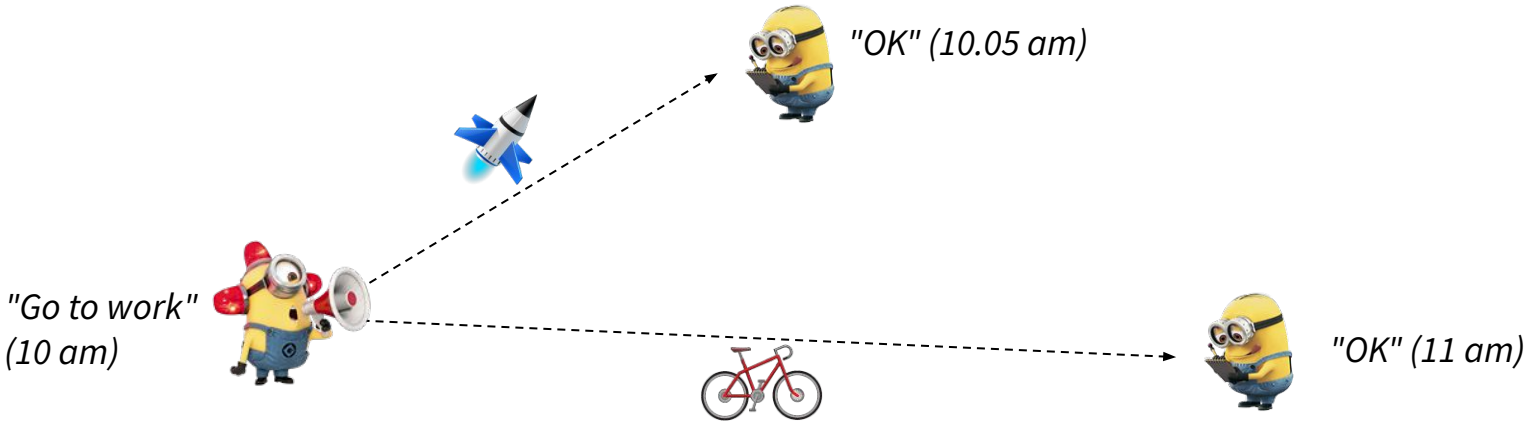
✗ Everything is safe

Server failures, link failures, network partitions



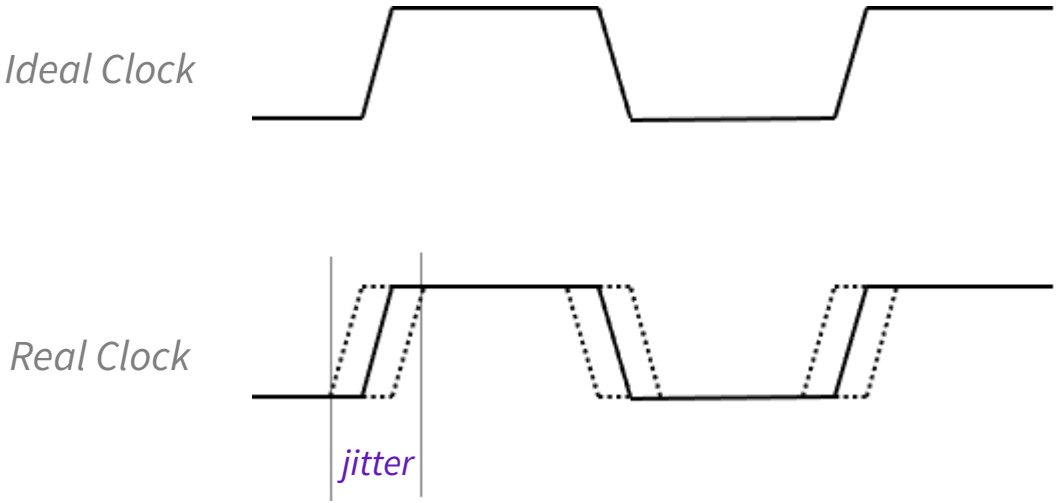
Time is Relative!

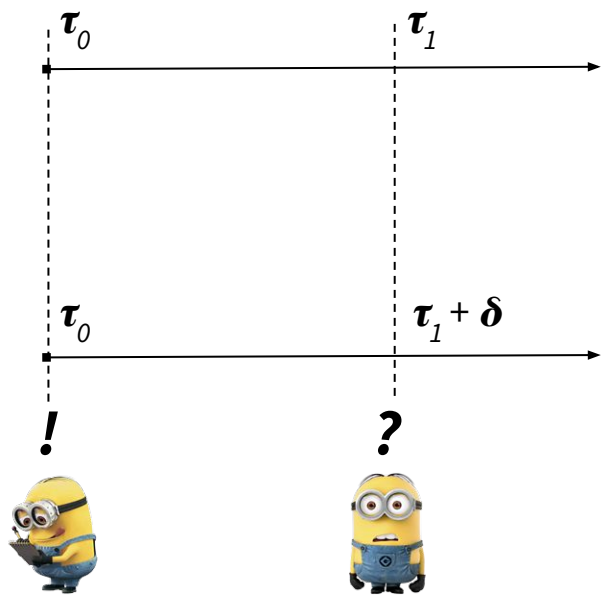
Clock skew



Time is Relative!

Clock jitter



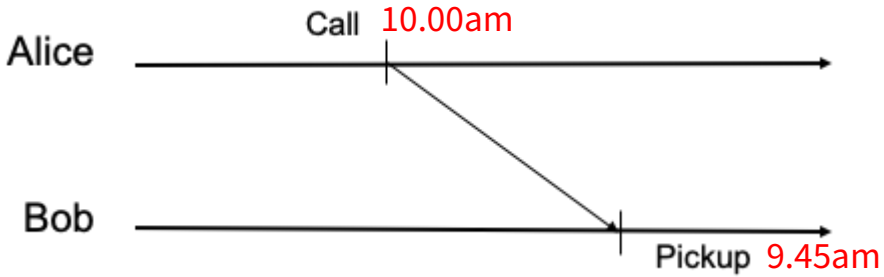


Time is Relative!

Jitter and skew induce clock drift
The drift varies independently for every node

Clock drifts may break causality

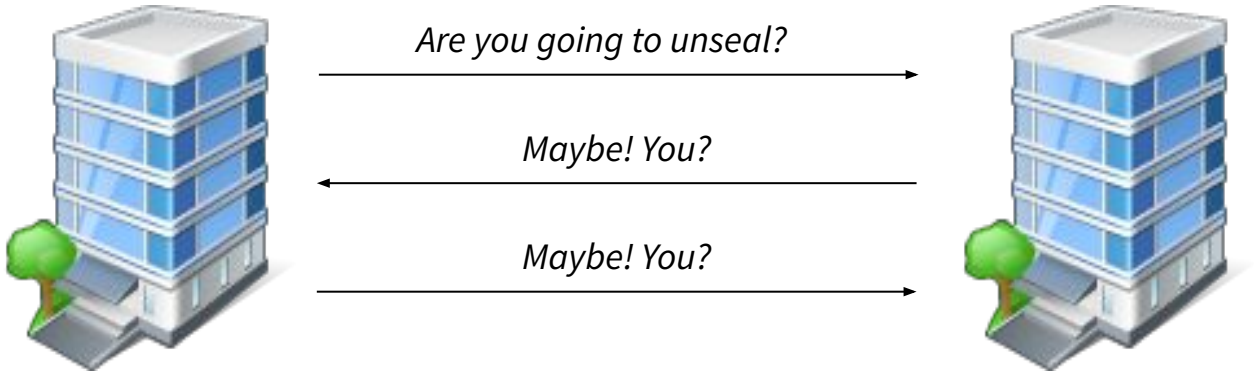
Alice and Bob both wake up at the same time in the real world
At that point their clocks display 8am
Alice's clock drifts forward very fast; Bob's clock drift is marginal
Alice is meant to call Bob at 10 am…



How to (Un)Seal a Deal

The Lockdown Conundrum (aka the *Two Generals' Problem* [3])

- A neighborhood with 2 buildings is in lockdown
- No new cases in the last 7 days
- Each building has a manager, both want to avoid risks at all costs
- They communicate by WeChat to decide whether they should lift the lockdown

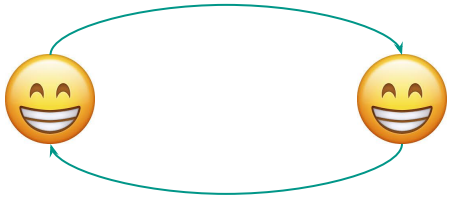


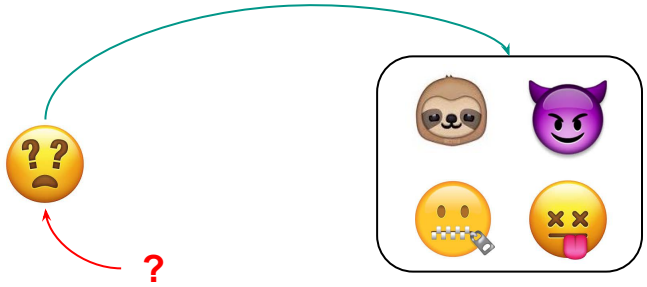
The Internet Can't Exist: Here's Why!

Scaling paradox

1. Probability of failures increases with the number of nodes
2. Dead nodes can bring down the entire system

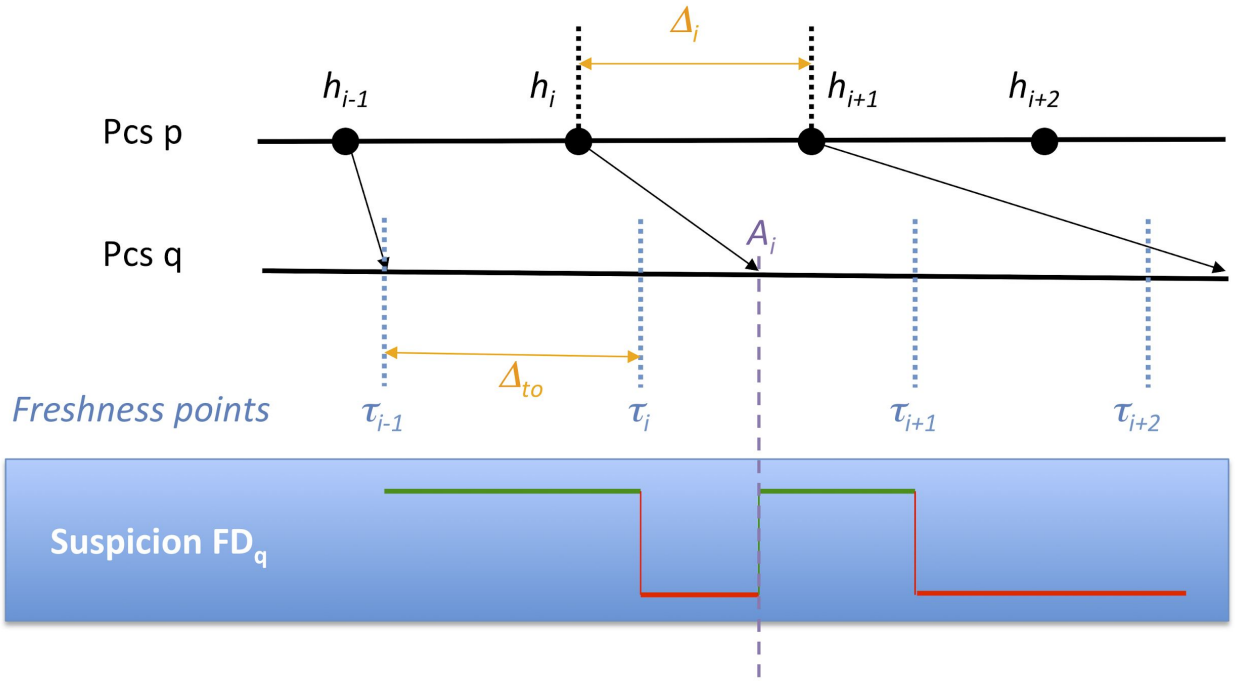
    Theoretical proof that two nodes may never reach consensus [4]



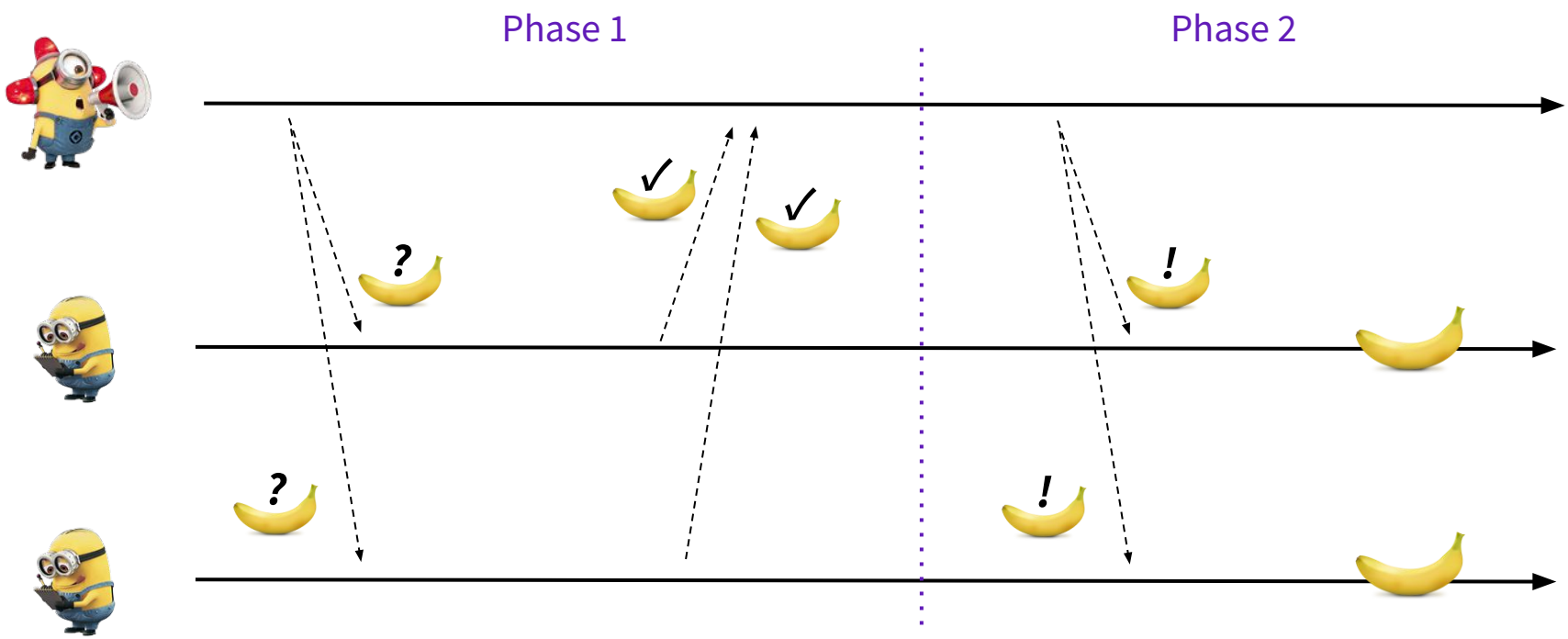


Failure Detection: Monitoring Liveness

Relying on an oracle (aka. a failure detector [6]) enables consensus

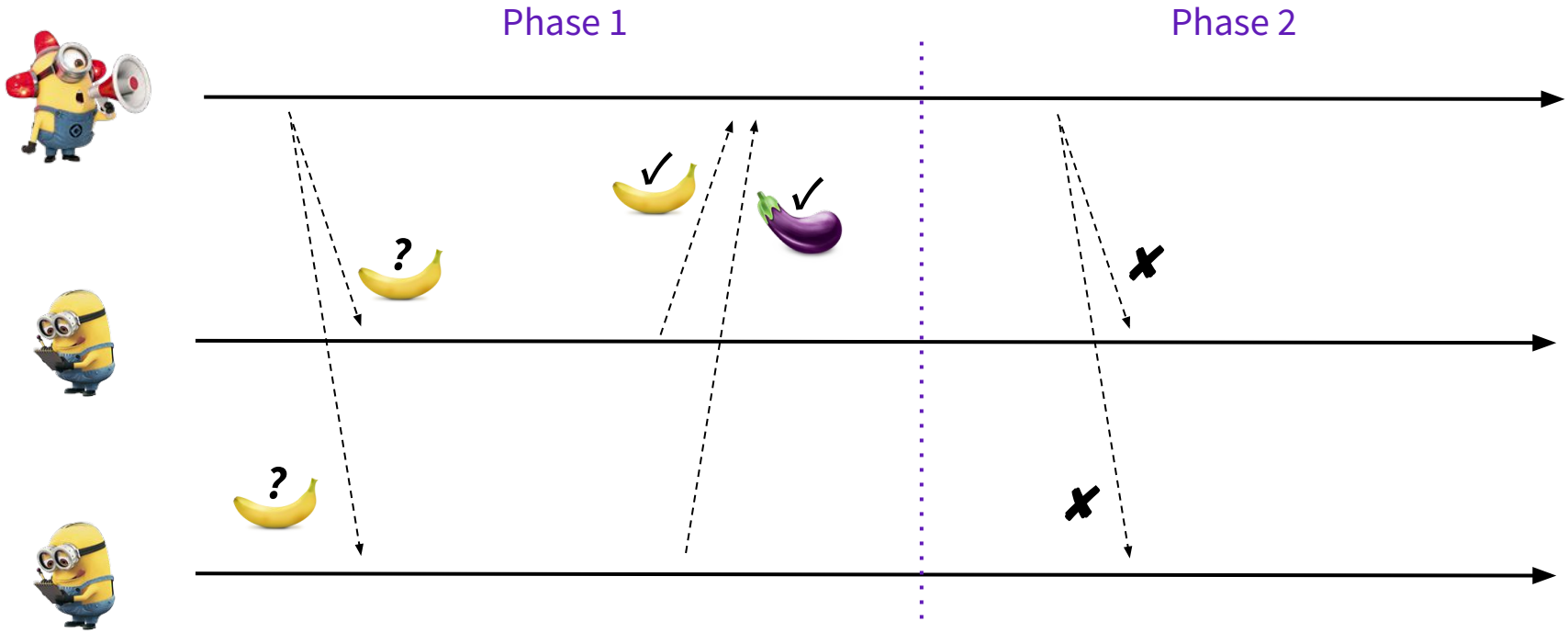


Two-Phase Commit

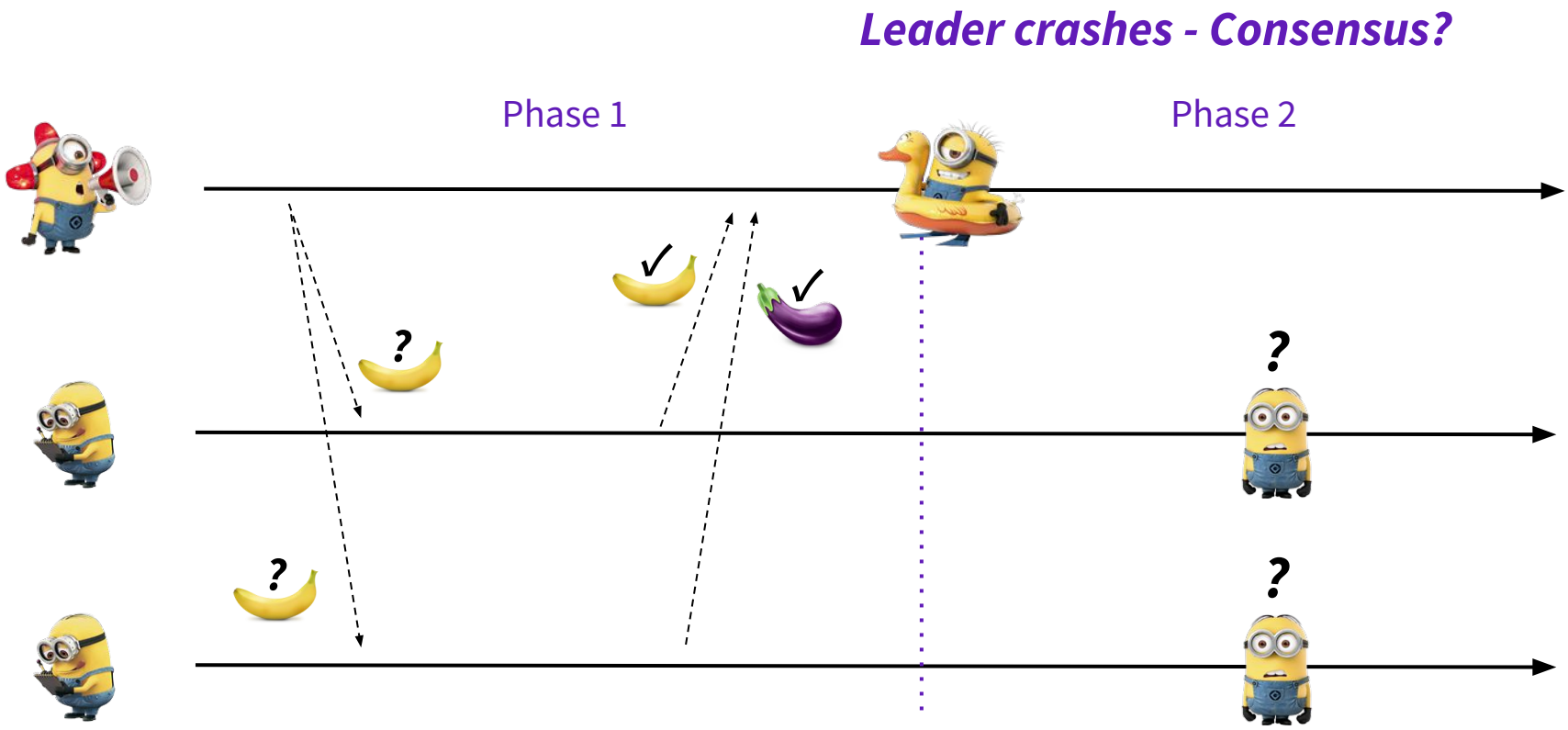


Two-Phase Commit

Consensus: Abort

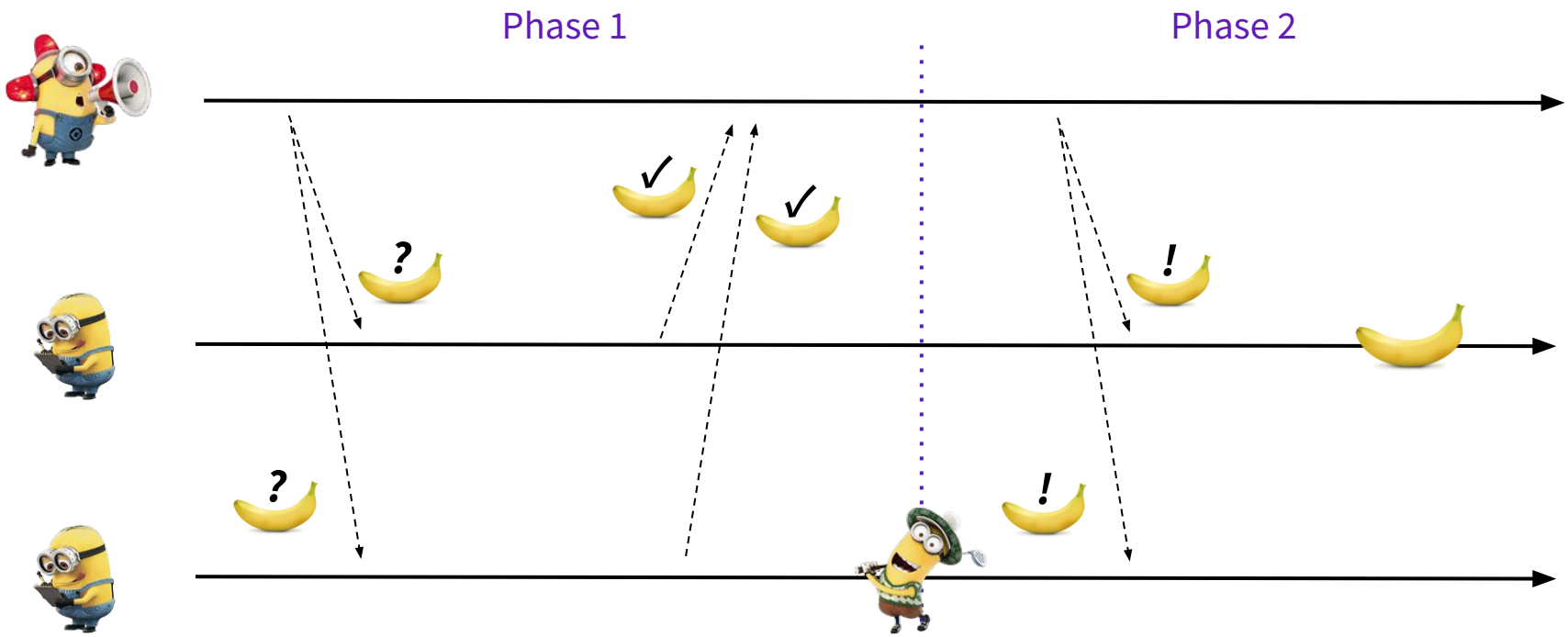


Two-Phase Commit

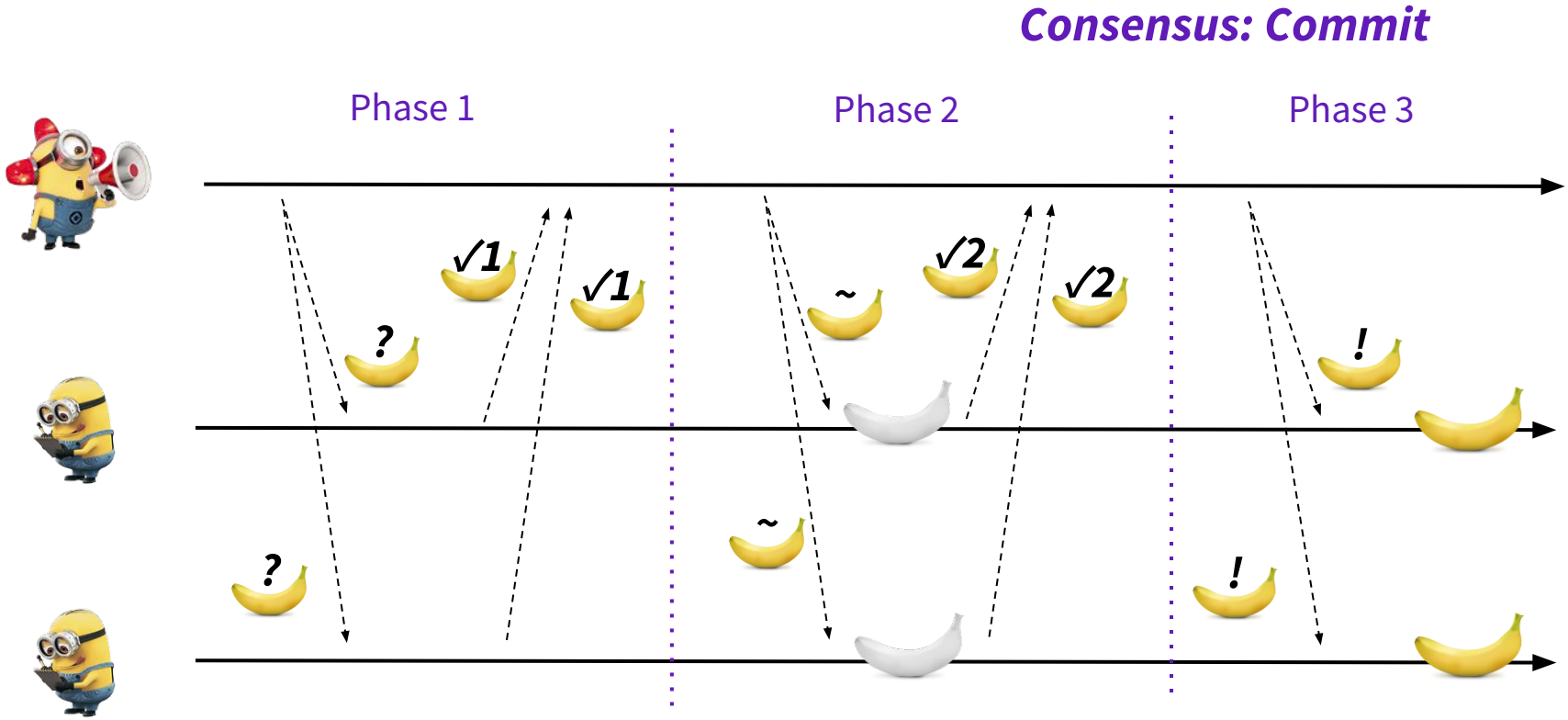


Two-Phase Commit

Follower crashes - Consensus?

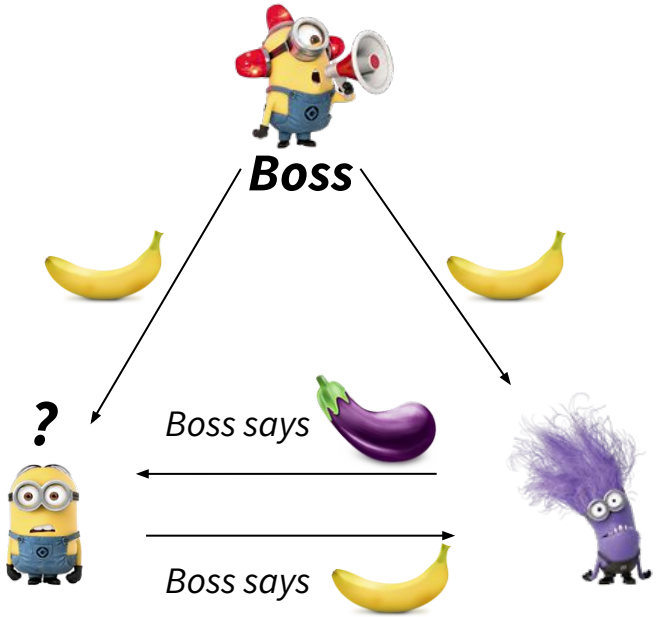


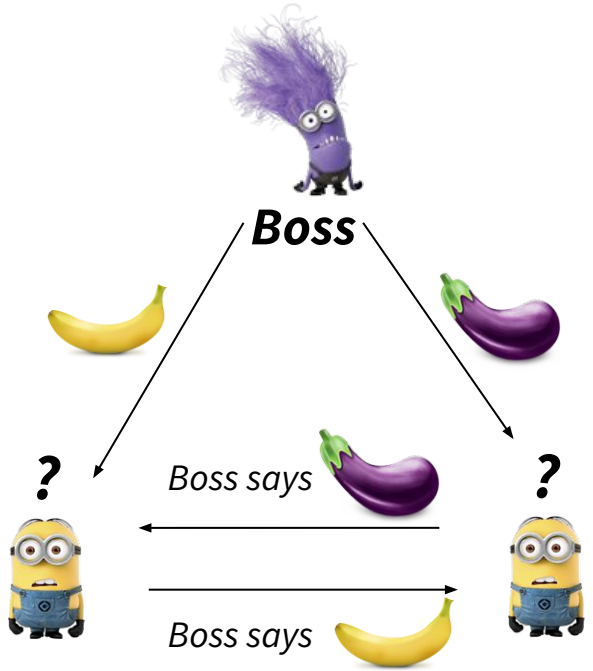
Three-Phase Commit



Consensus with byzantine failures

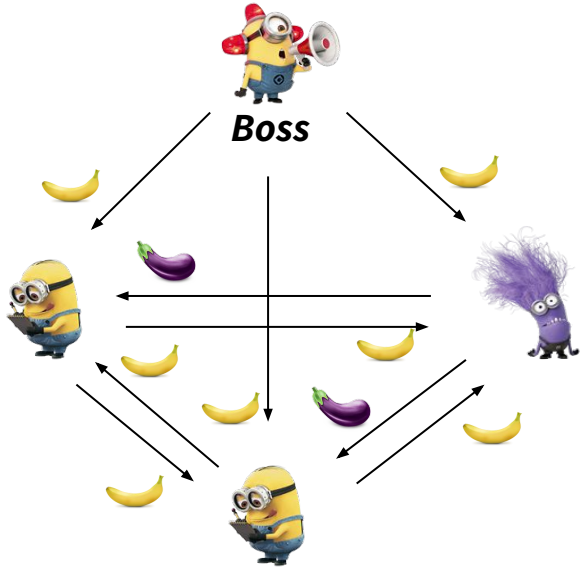
Impossibility: solving byzantine generals with 3 processes [5]

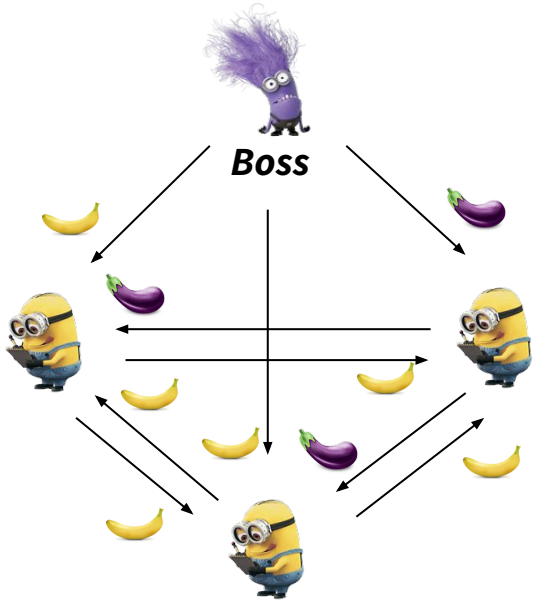




Consensus with byzantine failures

4-process solution for byzantine generals [5]





Takeaway points

None of the online technology you use daily is 100% reliable!

It can never be!

Failures are… ***inevitable***!

The biggest challenge is to maintain consistency through time and space

What to do?

- Be patient
- Read the instructions
- Keep receipts
- Don't trust unverified messages

References

[1] ‘A Systemwide Disaster’: How the Iowa Caucuses Melted Down - NYTimes, Feb. 6, 2020

[2] Deliveroo app crash leaves customers furious - The Sun, Sep. 18, 2018

[3] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber. Some constraints and tradeoffs in the design of network communications. SIGOPS Operating Systems Review 9(5). 1975. pp. 67–74.

[4] M. Fischer, N. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. Journal of the ACM 32(2). 1985. pp. 374-382.

[5] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems 4(3). 1982. pp. 382–401.

[6] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. Journal of the ACM 43(2). 1996. pp. 225–267.