Introduction to CAP Theorem

Ravindu Nirmal Fernando

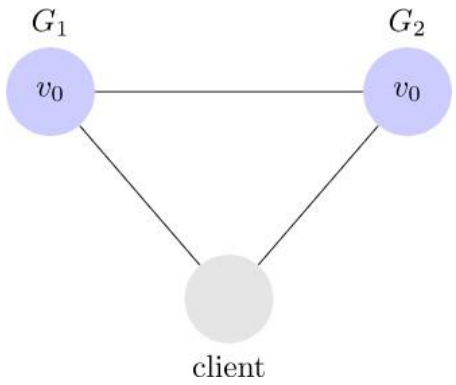
SLIIT | March 2025

CAP THEOREM

- A fundamental theorem in distributed systems.
- Can have at most two of the following three properties,
  - **C**onsistency
  - **A**vailability
  - **P**artition Tolerance

DISTRIBUTED SYSTEM

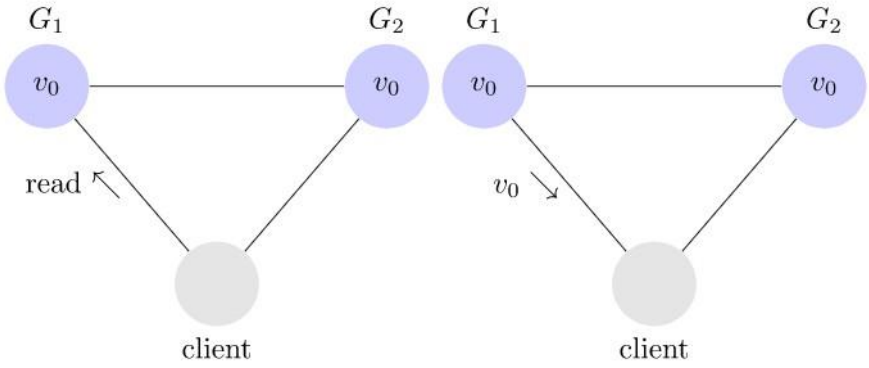
- Consider a simple distributed system with two servers, G_1 and G_2
  - The servers can communicate with each other and connect to remote clients



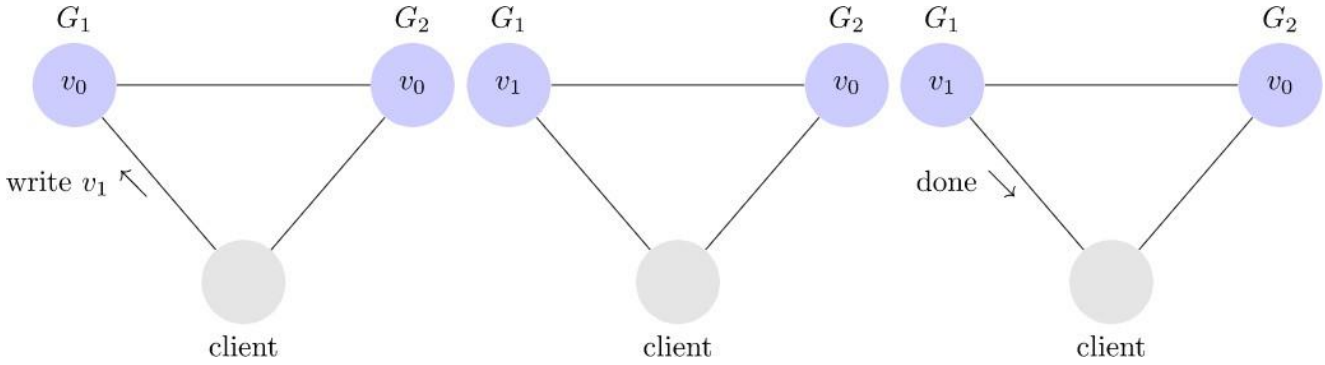
https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

DISTRIBUTED SYSTEM

- Read example



- Write example



https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

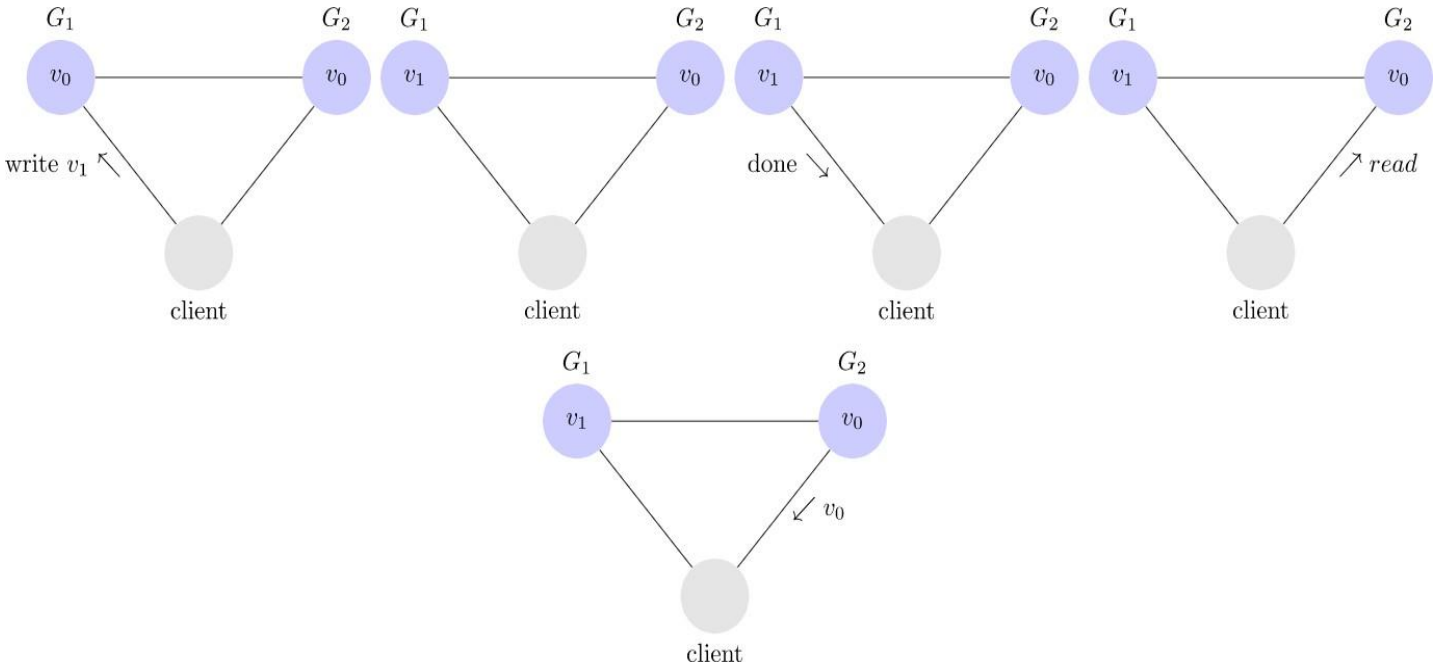
CONSISTENCY

Consistency means that all clients see the same data at the same time, no matter which node they connect to.

For this to happen, whenever data is written to one node, it must be instantly forwarded or replicated to all the other nodes in the system before the write is deemed ‘successful.’

CONSISTENCY

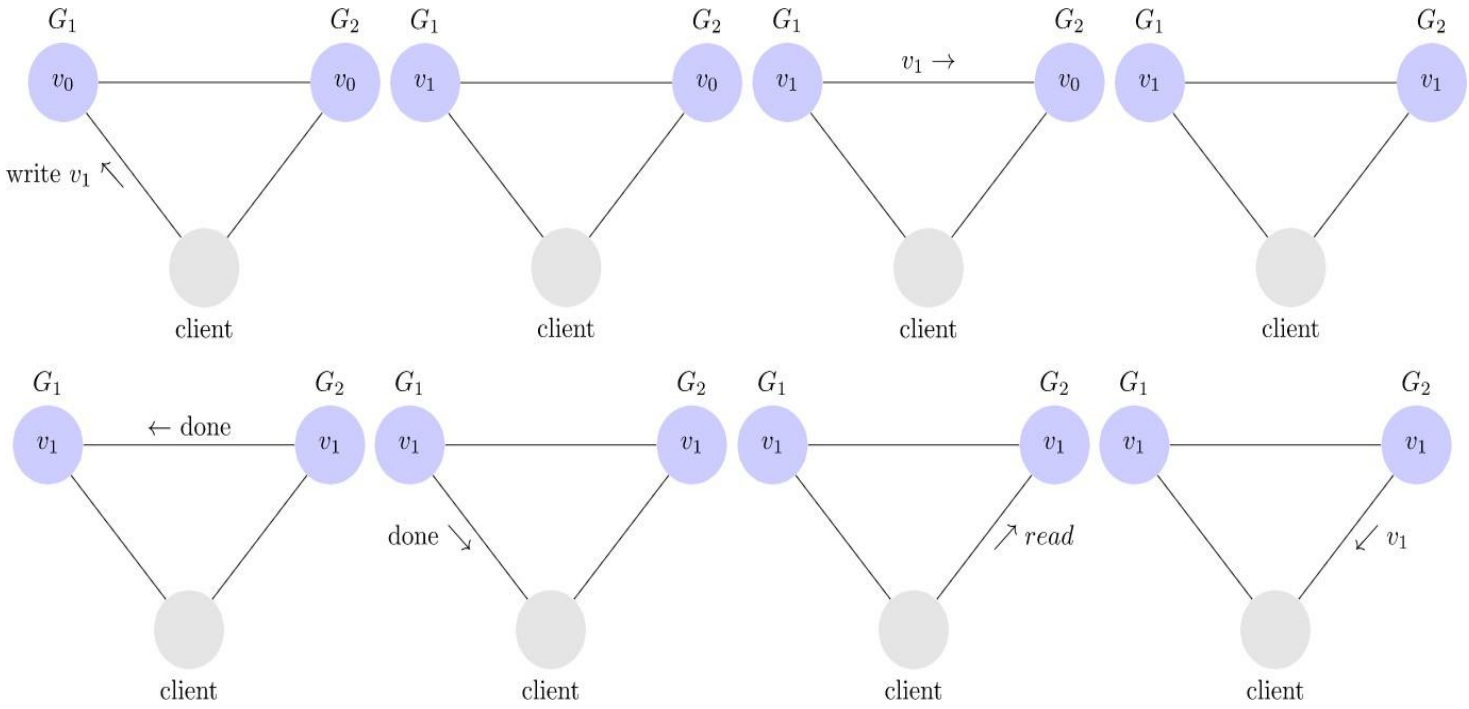
Inconsistent system



https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

CONSISTENCY

Consistent system



https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

AVAILABILTY

Availability means that any client making a request for data gets a response, even if one or more nodes are down.

Another way to state this—all working nodes in the distributed system return a valid response for any request, without exception.

PARTITION TOLERANCE

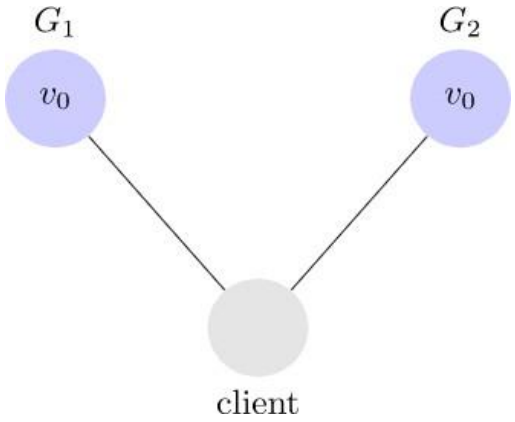
A **partition** is a communications break within a distributed system—a lost or temporarily delayed connection between two nodes.

Partition tolerance means that the cluster must continue to work despite any number of communication breakdowns between nodes in the system.

PARTITION TOLERANCE

The system continues to operate despite network partitions

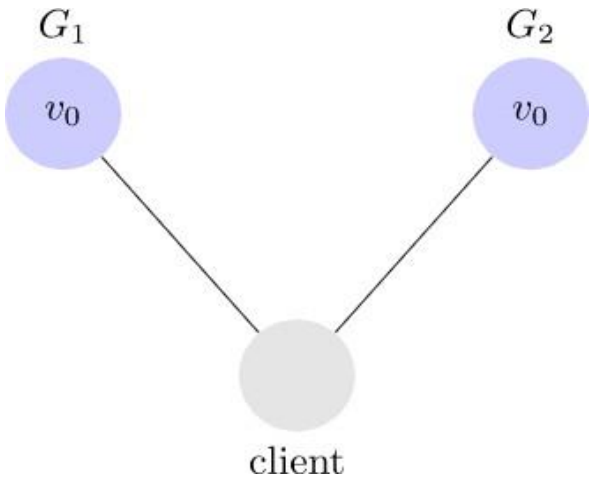
- Communication among the servers is not reliable
- Servers may be partitioned into multiple groups that cannot communicate with each other
- Messages may be delayed or lost forever



https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

PROOF

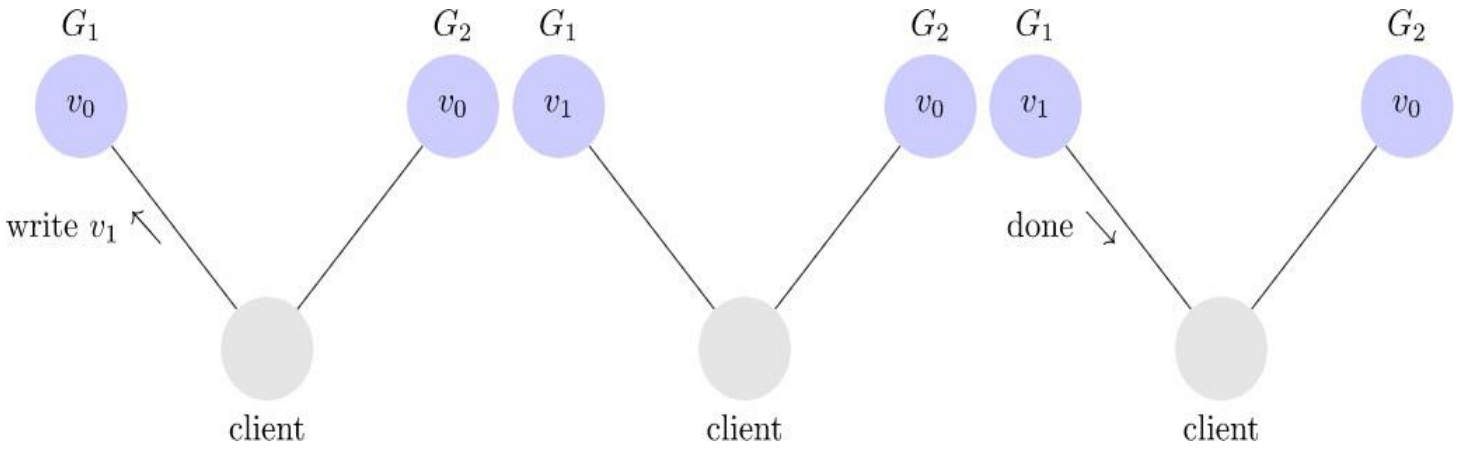
Consider partitioned system,



https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

PROOF

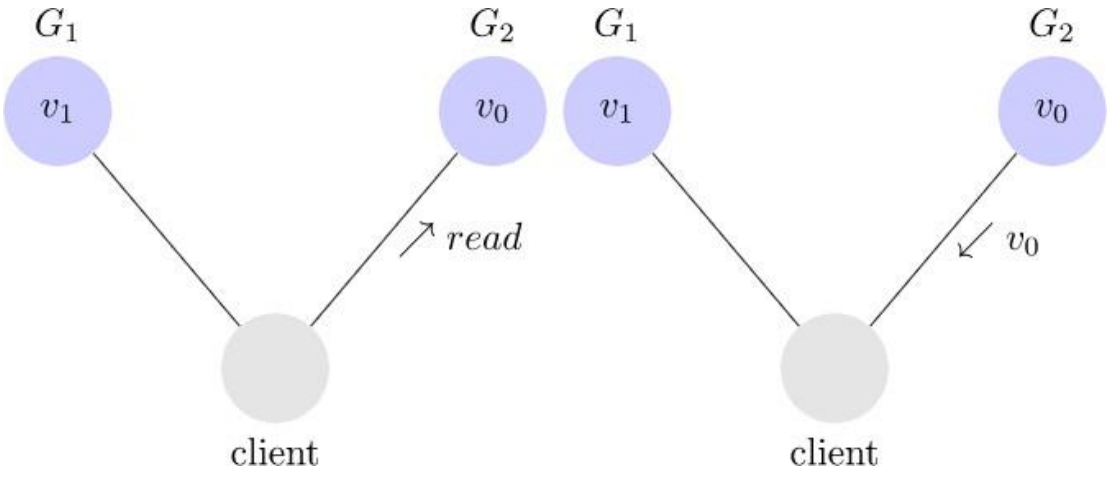
Client writes to G_1



https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

PROOF

Client reads from G_2

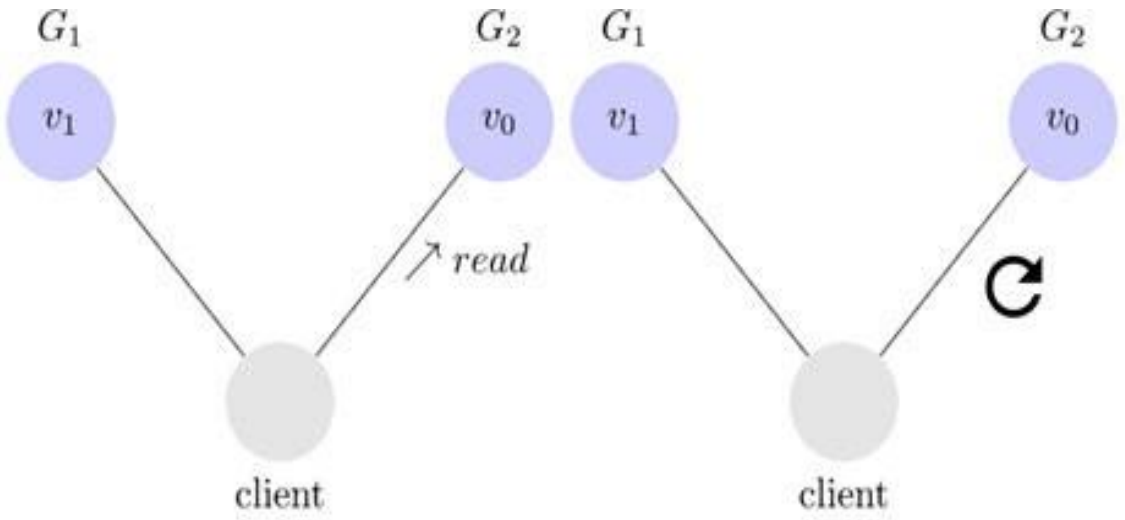


NOT CONSISTENT!

https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

PROOF

Client reads from G_2



NOT AVAILABLE!

https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

SYSTEM DESIGN

This suggests there are three kinds of distributed systems

- CP
- CA
- AP

Why is it important?

- Future of databases is distributed (Big Data Trend)
- CAP theorem describes the trade-offs involved in distributed systems
- Proper understanding of CAP theorem is essential to making decisions about distributed database/system design
- Misunderstanding can lead to erroneous or inappropriate design choices

CONSISTENCY OR AVAILABILITY

- Consistency & Availability is not "binary" decision
- AP systems relax consistency in favor of availability – but are not inconsistent
- CP systems sacrifice availability for consistency- but are not unavailable
- This suggests both AP & CP systems can offer a degree of consistency, & availability, as well as partition tolerance

CONSISTENCY MODELS

Strong Consistency

- After the update completes, any subsequent access will return the same updated value.

Weak Consistency

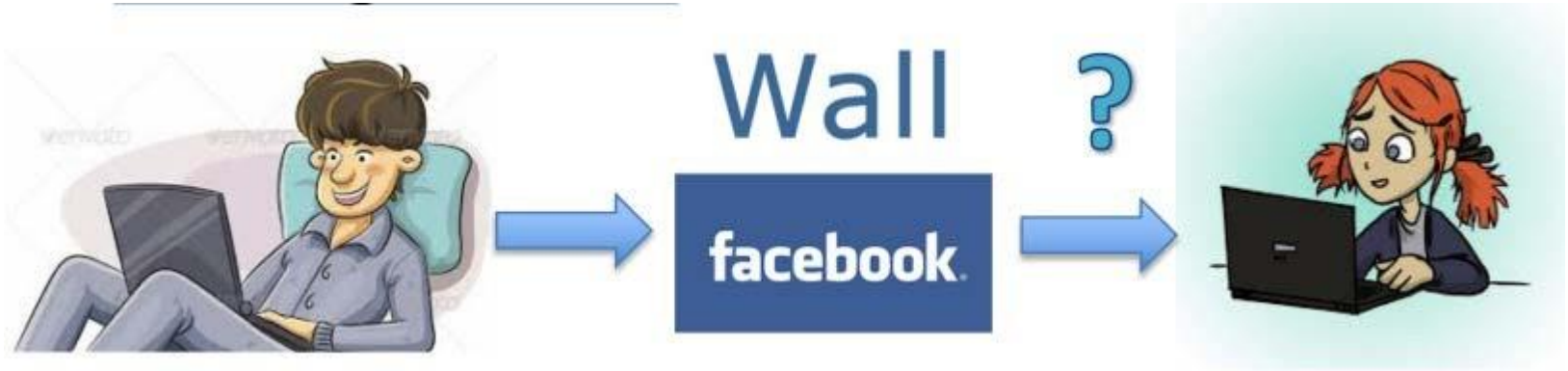
- It is not guaranteed that subsequent accesses will return the updated value.

Eventual Consistency

- Specific form of weak consistency
- It is guaranteed that if no new updates are made to object, eventually all accesses will return the last updated value (e.g., propagate updates to replicas in a lazy fashion)

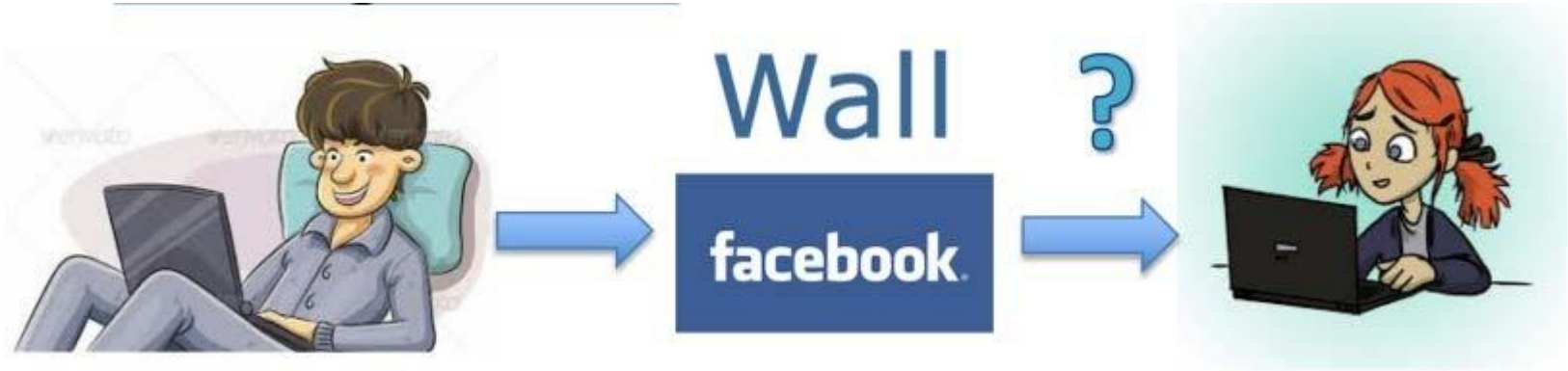
EVENTUAL CONSISTENCY – FACEBOOK EXAMPLE

- Bob finds an interesting story and shares with Alice by posting on her Facebook wall
- Bob asks Alice to check it out
- Alice logs in her account, checks her Facebook wall but - Nothing is there!



EVENTUAL CONSISTENCY – FACEBOOK EXAMPLE

- Bob tells Alice to wait a bit and check out later
- Alice waits for a minute or so and checks back – Finds the wall post!



EVENTUAL CONSISTENCY – FACEBOOK EXAMPLE

- Why would Facebook choose an eventual consistent model over the strong consistent one?
  - Facebook has billions of active users
  - It is non-trivial to efficiently and reliably store the huge amount of data generated at any given time
  - Eventual consistent model offers the option to reduce the load and improve availability

DYNAMIC TRADEOFF BETWEEN C AND A

An airline reservation system:

- When most of seats are available: it is ok to rely on somewhat out-of-date data, availability is more critical
- When the plane is close to be filled: it needs more accurate data to ensure the plane is not overbooked, consistency is more critical

Neither strong consistency nor guaranteed availability, but it may significantly increase the tolerance of network disruption

REFERENCES

- Gilbert, Seth, and Nancy Lynch. "Perspectives on the CAP Theorem." Computer 45.2 (2012): 30-36
- https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/
- https://en.wikipedia.org/wiki/CAP_theorem#:~:text=The%20CAP%20theorem%20says%20that,'P'%20in%20CAP