Practical Client-side Replication: Weak Consistency Semantics for Insecure Settings

NOVALINCS
LABORATORY FOR COMPUTER SCIENCE AND INFORMATICS

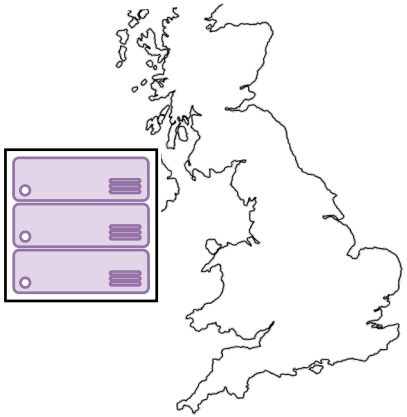
Albert van der Linde, João Leitão, Nuno Preguiça

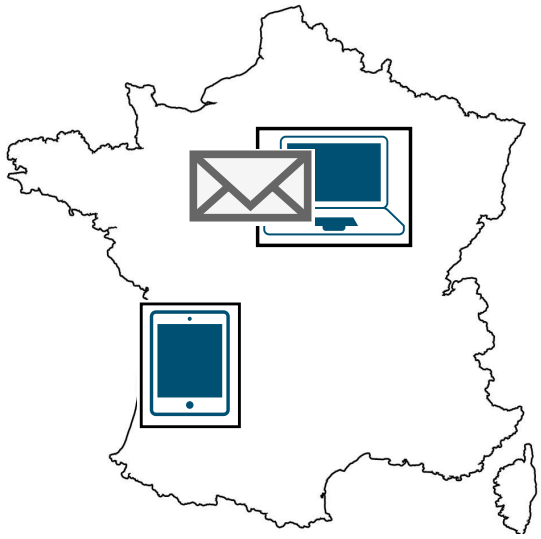
NOVA LINCS
NOVA School of Science and Technology
NOVA University Lisbon

Context

Applications where users interact with each other are highly impacted by client-server latency, especially noticeable if users are very close to each other:

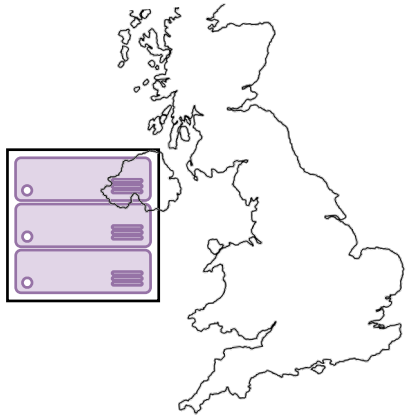
- Editing documents;
- Audience engagement;
- Multi-user games.

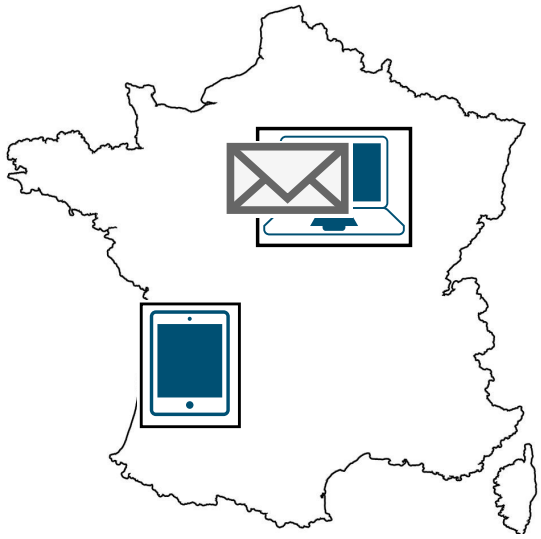




Context

One way to reduce latency among interactions is by letting user's devices communicate directly – P2P.





Context

What is necessary to make this work?

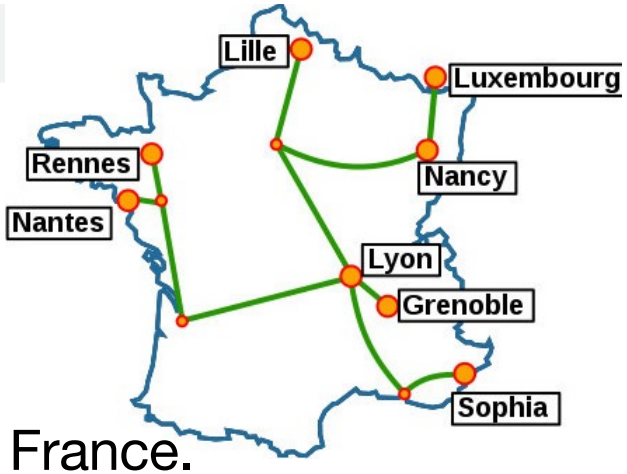
- Clients maintain partial replicas of the DB;
- Generate operations locally and propagate over the p2p network.

How about consistency?

- Need to rely on weak consistency – causal consistency as the strongest available consistency level;
- Need to manage concurrent update – CRDTs.

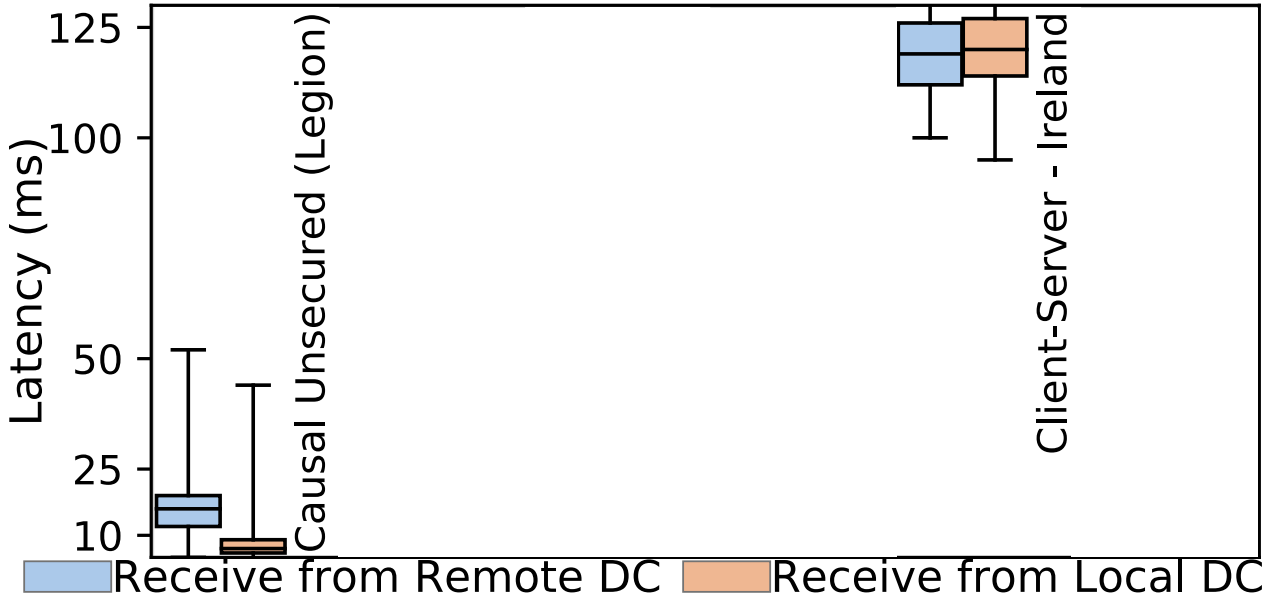
Our solution: Legion, a framework to extend web-based applications with peer-to-peer synchronization.

Context



Comparing the P2P approach vs client-server:

- server in Ireland, clients spread over different DCs in France.



Legion from:
A. van der Linde, et al.
Legion: Enriching internet services with peer-to-peer interactions. WWW '17

Context

Conclusion: P2P provides much lower latency.

Important for interactive applications where low latency is key for user experience.

What can go wrong with clients generating and propagating operations?

- Cheating becomes an issue.

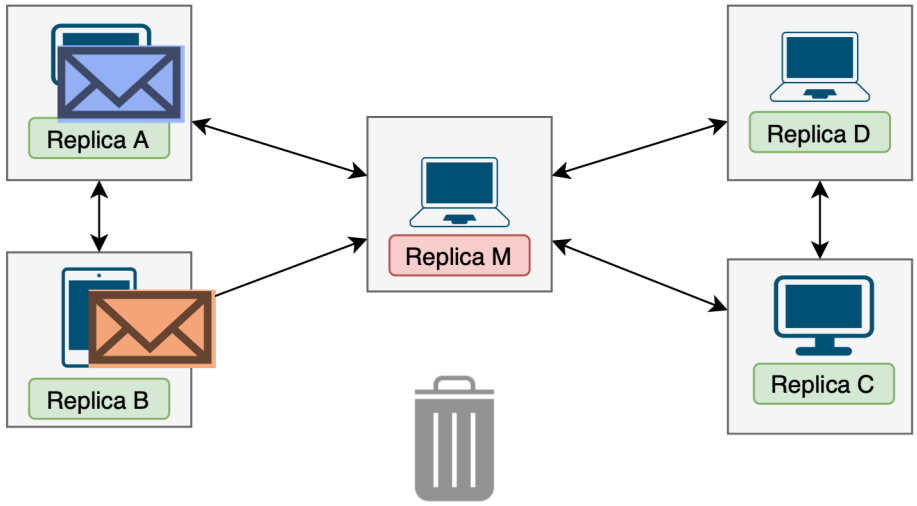
Outline

- Introduction
- Possible attacks
  - Generic attacks to broadcast
  - Attacks to causal ordering
- Secure causal consistency semantics
  - Secure causal consistency
  - Strict secure causal consistency
  - Secure extended causal consistency

Attack: tamper with existing messages

While propagating operations malicious users can, for example:

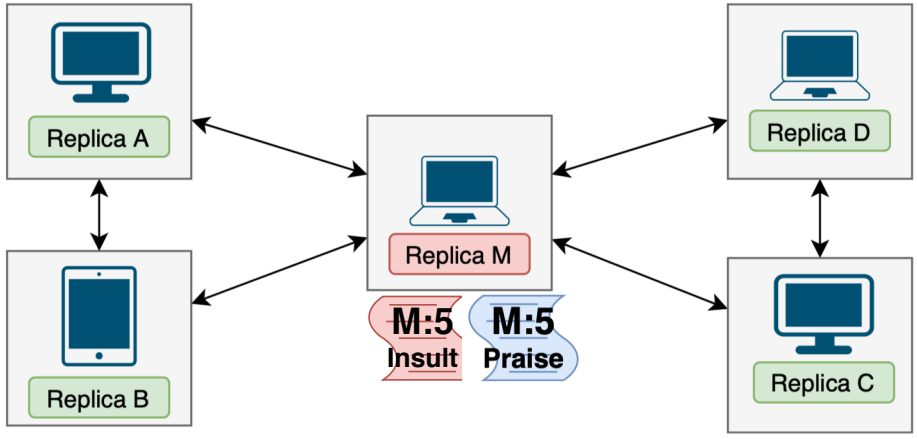
- Not propagate some messages selectively;
- Changing the message contents.



Attack: create different operations with same identifier

Send different messages with the same identifiers to separate groups, leading to diverging state:

- E.g.: send a praise to some users and an insult to others;
- If Replicas A and D connect directly and synchronize, systems using version vectors to summarize the state assume that both replicas are synchronized.



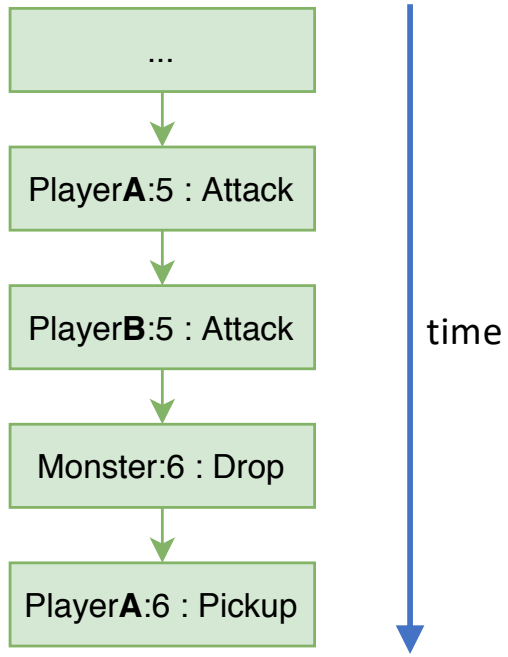
Outline

- Introduction
- Possible attacks
  - Generic attacks to broadcast
  - **Attacks to causal ordering**
- Secure causal consistency semantics
  - Secure causal consistency
  - Strict secure causal consistency
  - Secure extended causal consistency

Attack: depend on the future

Simple game: users are to defeat a monster and attempt to pickup any item that is dropped.

- Players: attack monster
- Monster: drops an item
- Players observe the drop,
- Players attempt to pickup the item



Attack: depend on the future

Simple game: users are to defeat a monster and attempt to pickup any item that is dropped.

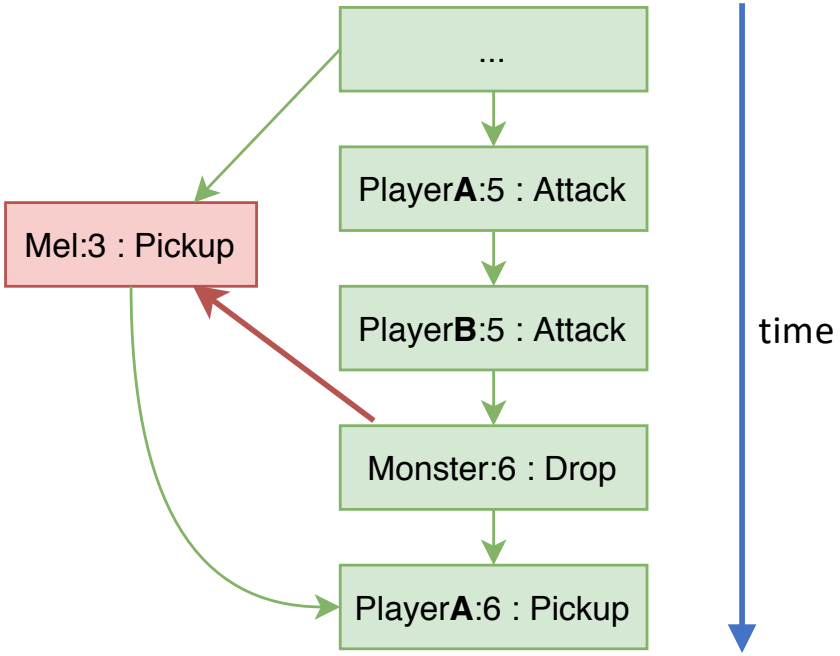
- Players: attack monster

Cheater picks up future drop

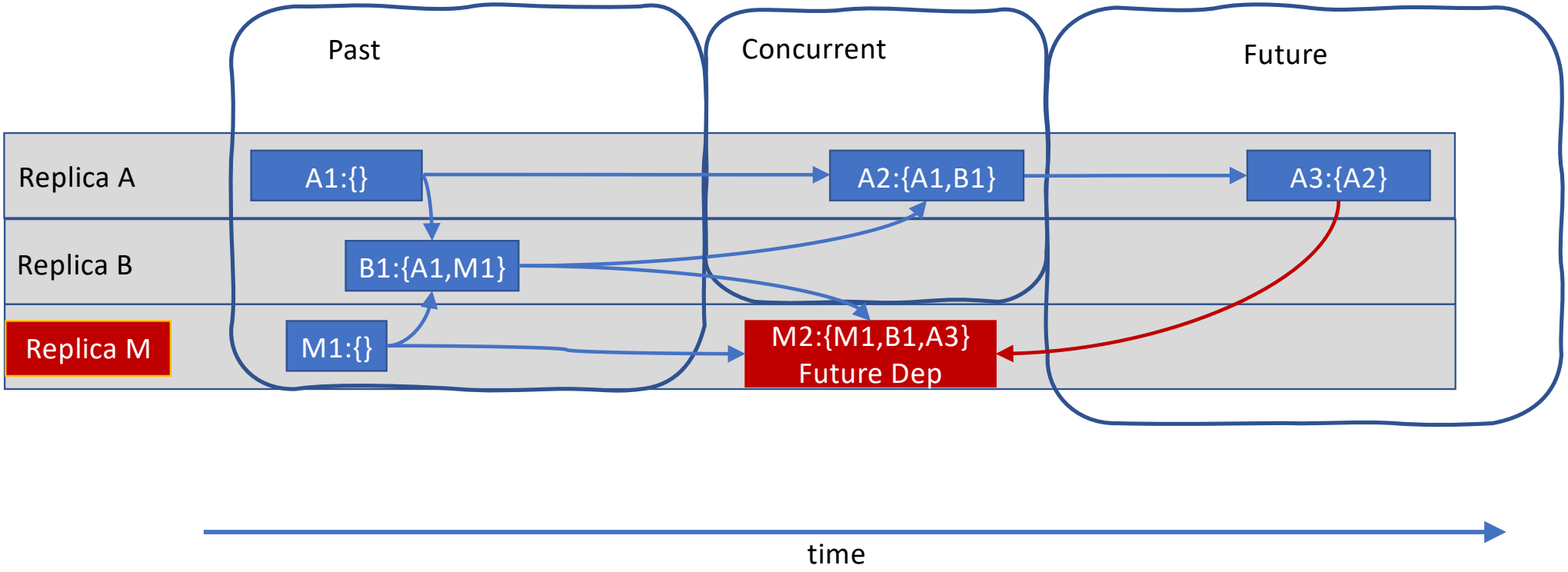
- Monster: drops an item
- Players observe the drop,

  apply Mel:3 immediately

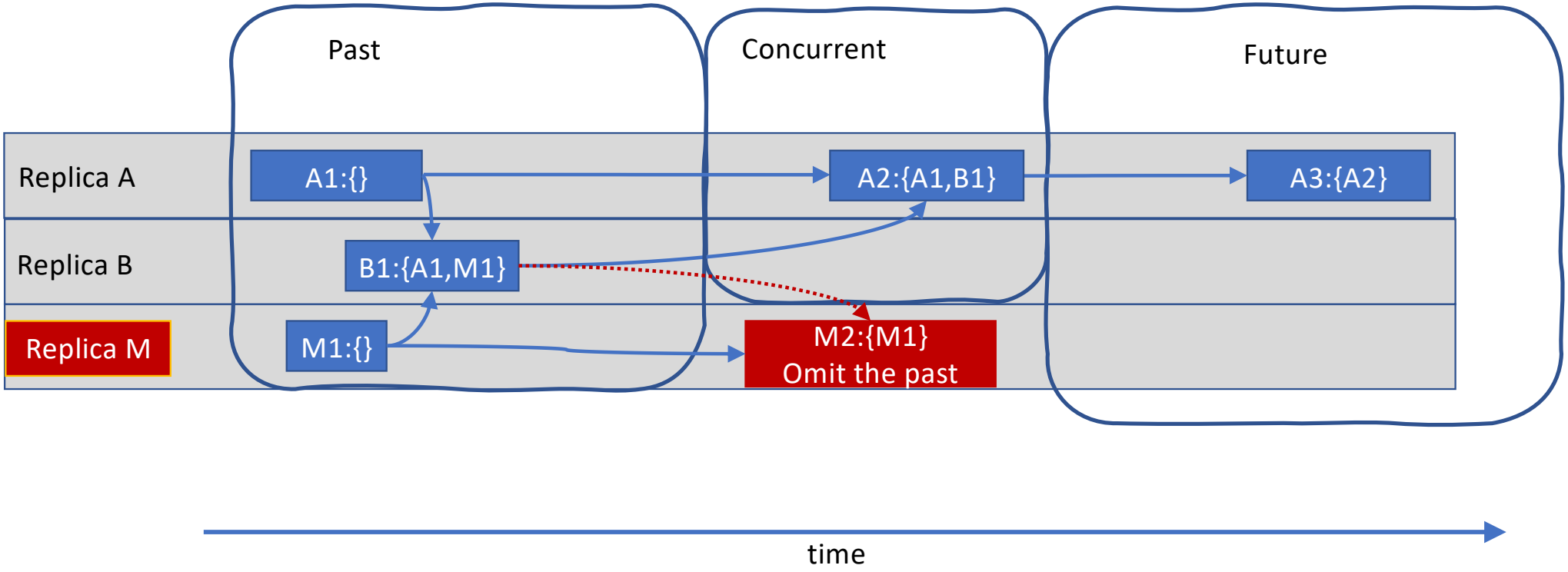
- Players attempt to pickup the item
  - ???



Depend on the future



Omit the past



Defending against these attacks

Byzantine fault-tolerant replication algorithms

- (Typically) Strong consistency;
- (Typically) Require synchronous coordination.

What is possible while keeping as much availability as possible?

Proposal:

- Define secure causal consistency models
- Propose techniques to implement these consistency models

Outline

- Introduction
- Possible attacks
  - Generic attacks to broadcast
  - Attacks to causal ordering
- Secure causal consistency semantics
  - **Secure causal consistency**
  - Strict secure causal consistency
  - Secure extended causal consistency

Secure Causal Consistency

Defends against the following malicious behaviours:

- Creating two operations with the same identifier;
- Tampering with existing operations;
- Fine-grain control over dependencies not to include;
- Creating operations which depend on future operations.

Secure Causal Consistency

Defends against the following malicious behaviours:

- **Creating two operations with the same identifier;**
- **Tampering with existing operations;**
- Fine-grain control over dependencies not to include;
- Creating operations whi

Not possible to avoid, but possible to detect.

- Operations signed.
- Our prototype uses a central server – that sends a periodic signed hash of operations.
- Similar approach can be used in a decentralized way if all replicas can communicate with each other.

Secure Causal Consistency

Defends against the following malicious behaviours:

- Creating two operations with the same identifier;
- Tampering with existing operations;
- **Fine-grain control over dependencies not to include;**
- Creating operations which depend on future operations

Using vector clocks or direct dependencies to record dependencies avoids omitting a specific operation in the past.
Still possible to omit some operations.

Secure Causal Consistency

Defends against the following malicious behaviours:

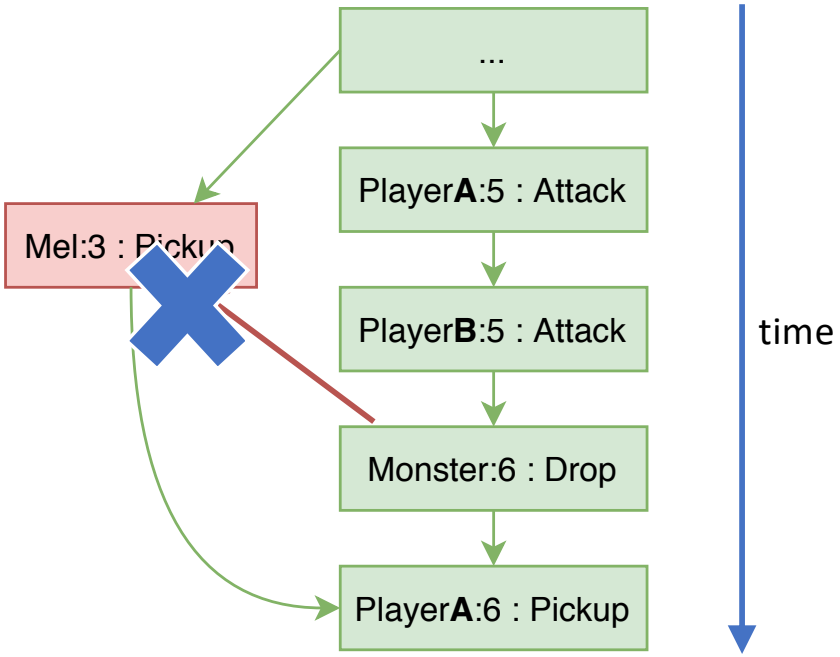
- Creating two operations with the same identifier;
- Tampering with existing operations;
- Fine-grain control over dependencies not to include;
- **Creating operations which depend on future operations.**

Dependencies include a cryptographic summary (hash) of all direct dependencies.
Operations include a random number to make them un-guessable.

Secure Causal Consistency

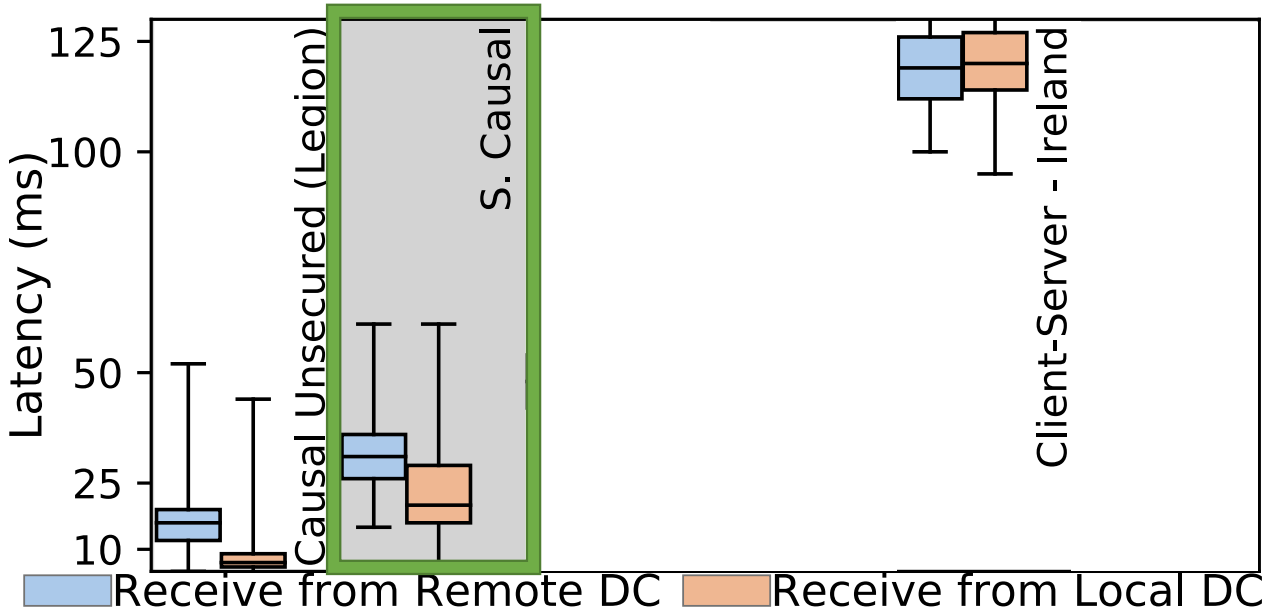
Example: depending on the future monster drop

- The pickup operation must include a hash of the actual operation it depends on;
  - Operations include a random number.
- Mel cannot create a valid hash without observing the drop.



Secure Causal Consistency

The cost in latency: cryptographic overheads.



Outline

- Introduction
- Possible attacks
  - Generic attacks to broadcast
  - Attacks to causal ordering
- Secure causal consistency semantics
  - Secure causal consistency
  - **Strict secure causal consistency**
  - Secure extended causal consistency

Strict secure causal consistency

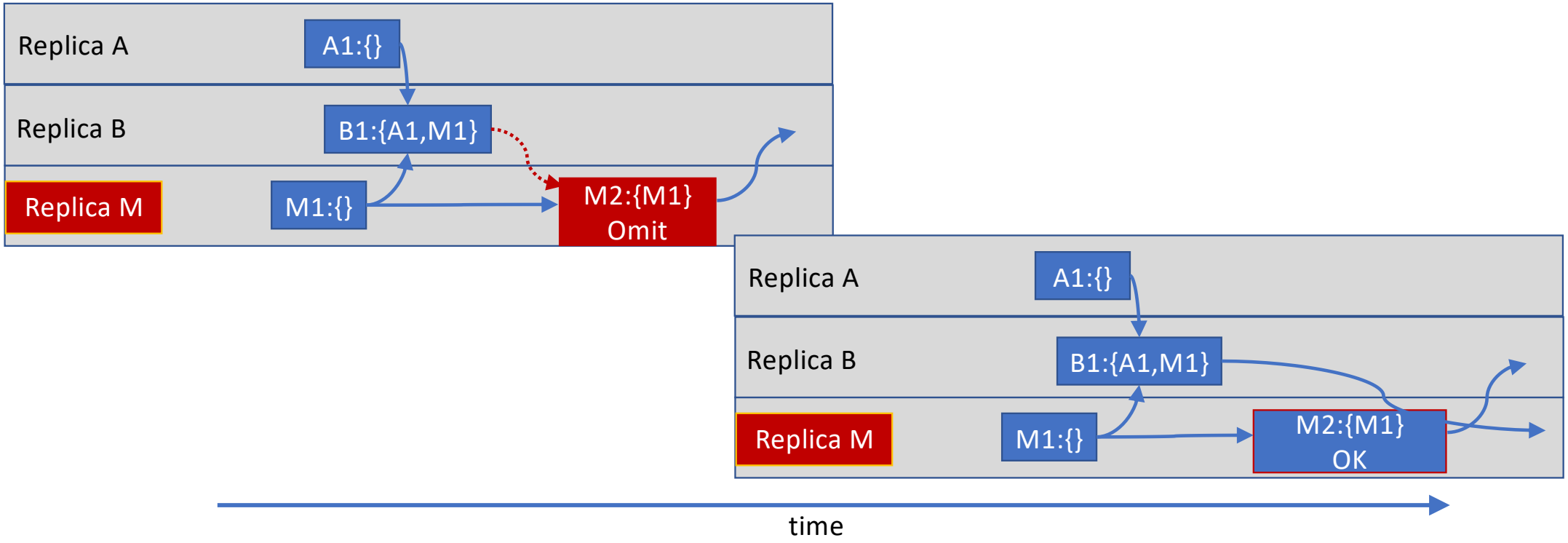
Defends against the following malicious behaviours:

- Secure causal consistency, plus:
- Omitting any dependency.

Requires using exact causal dependencies.

Problem for enforcement

Some correct and incorrect situations are indistinguishable by an external observer.



Strict secure causal consistency

How to implement?

Rely on secure hardware (e.g. SGX) for managing operations in each replica.

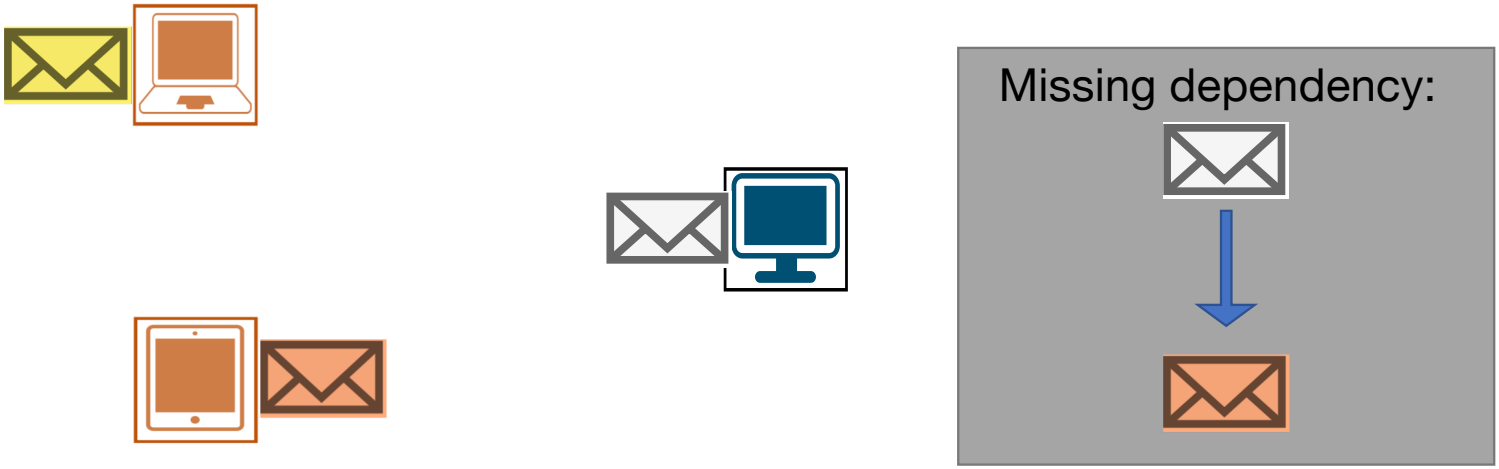
Outline

- Introduction
- Possible attacks
  - Generic attacks to broadcast
  - Attacks to causal ordering
- Secure causal consistency semantics
  - Secure causal consistency
  - Strict secure causal consistency
  - Secure extended causal consistency

Some attacks are still possible

Strict causal consistency seems to require special hardware…

… and collusion can still be used to omit dependencies.



Taking a step back

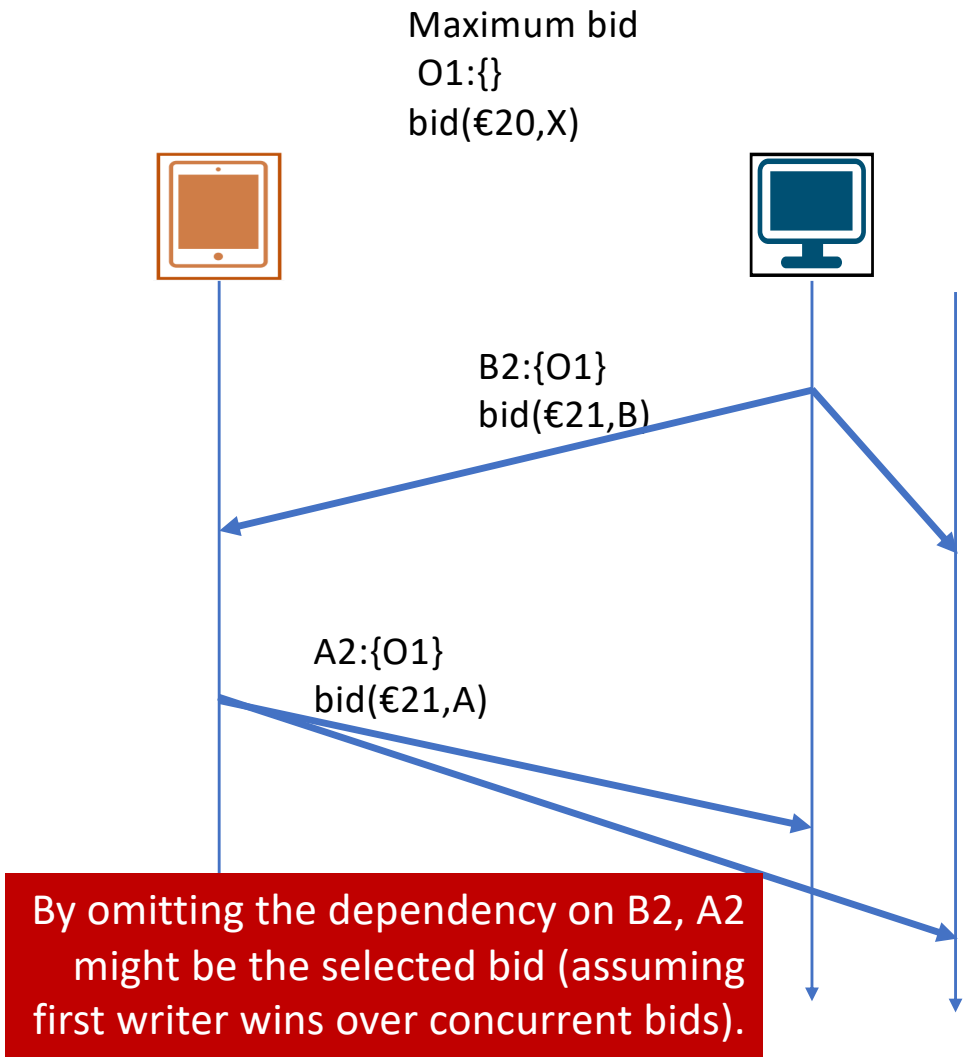
What do we want to avoid?

Having a replica that knows about operation **o** and does not record **o** in the dependencies of its new operations.

This turns out to be very difficult.

What is the problem?

By omitting dependencies, a replica might get advantageous conflicting resolution results.

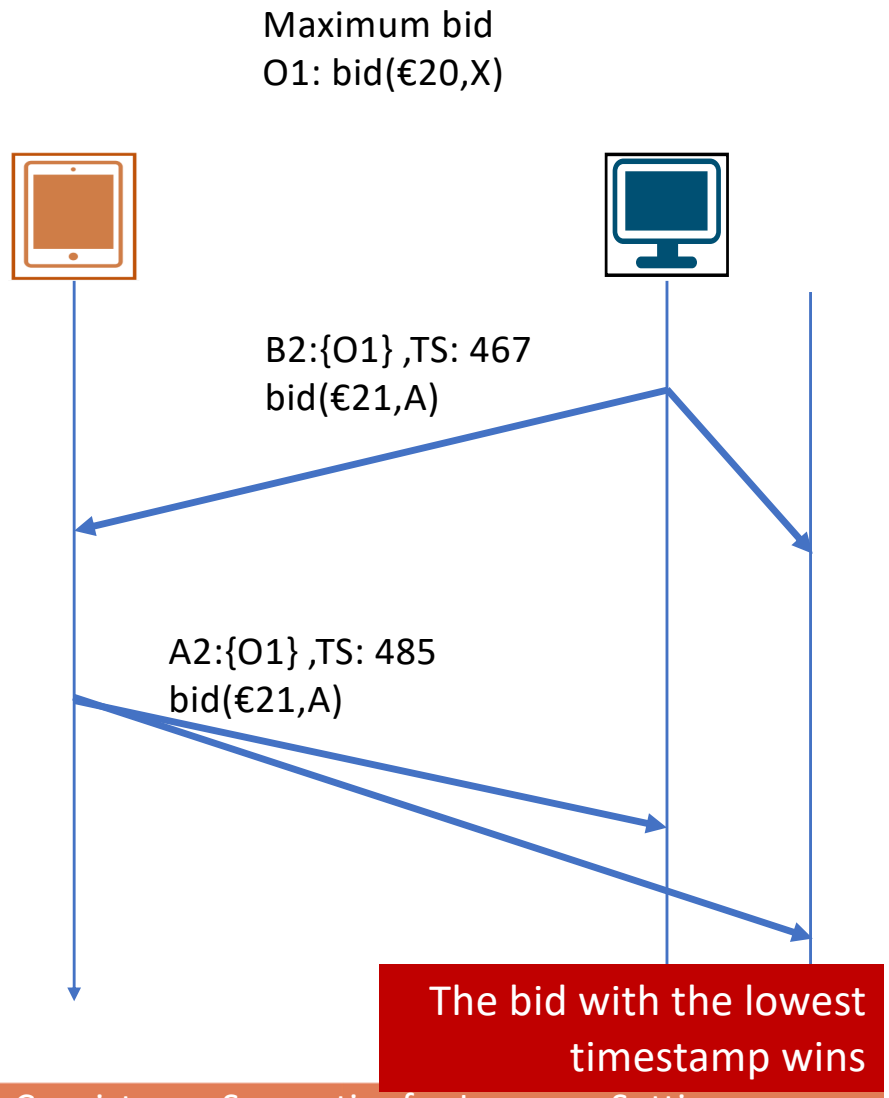


Taking a step back

A close approximation:

Allow a replica to create operations concurrent with some known operation **o**, but…

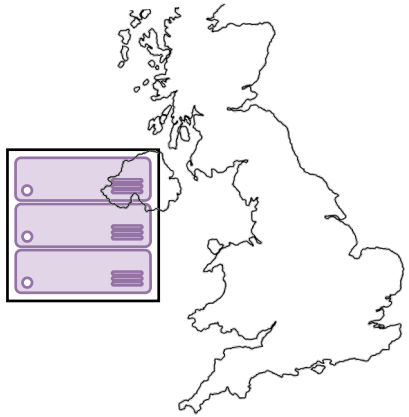
Guarantee that the operation has a timestamp that registers it was created later than **o**.

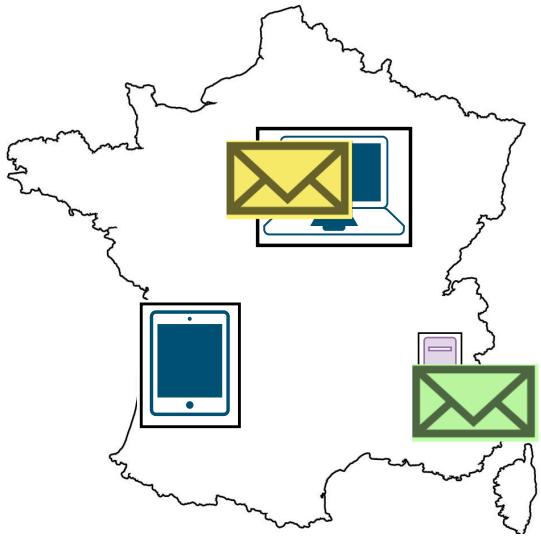


Secure extended causal consistency

Send operation to a timestamping service, which sends back a timestamped operation.

Propagate through the p2p network.

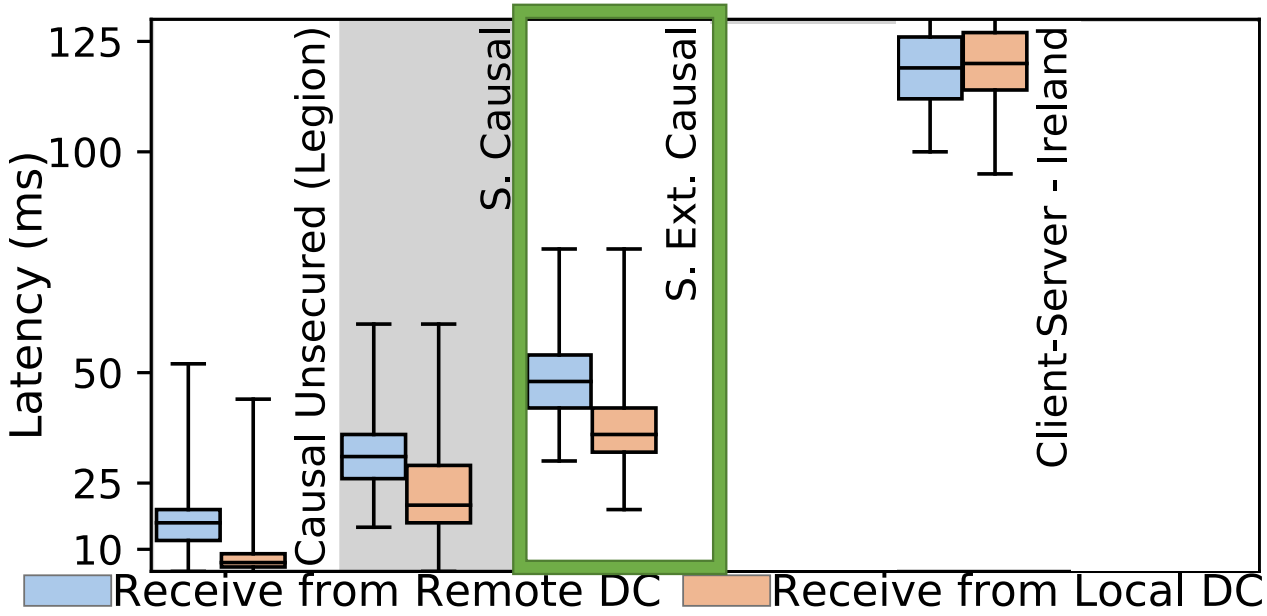




Secure extended causal consistency

The cost in latency: time to reach the TiS

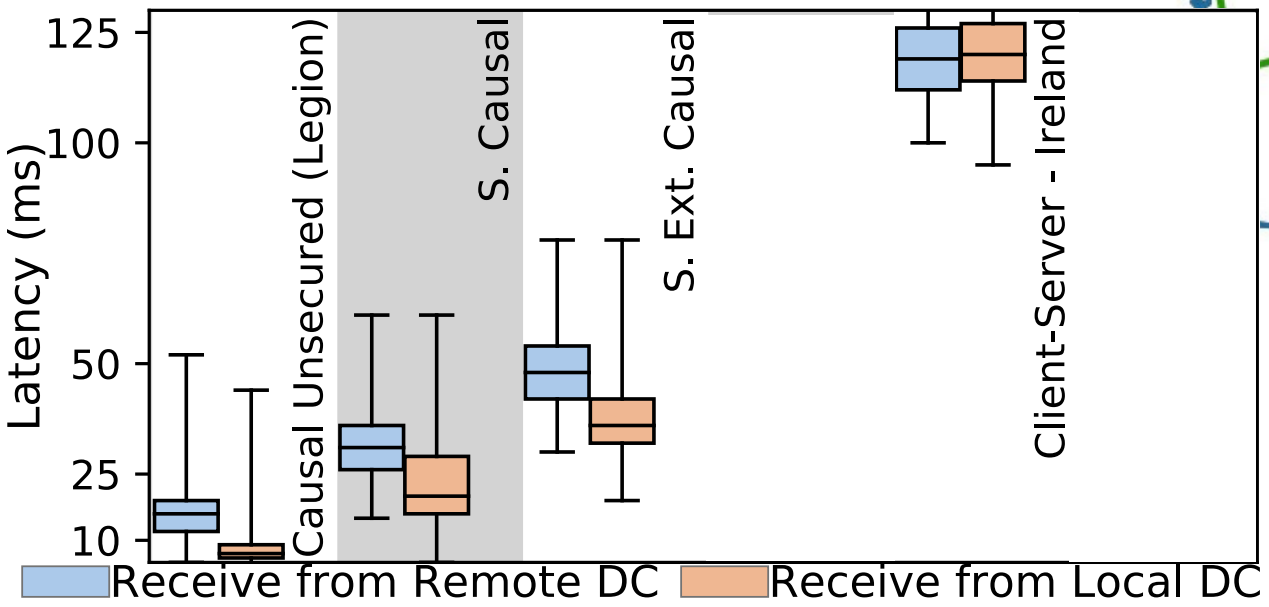
- Closest cloud point of presence - AWS

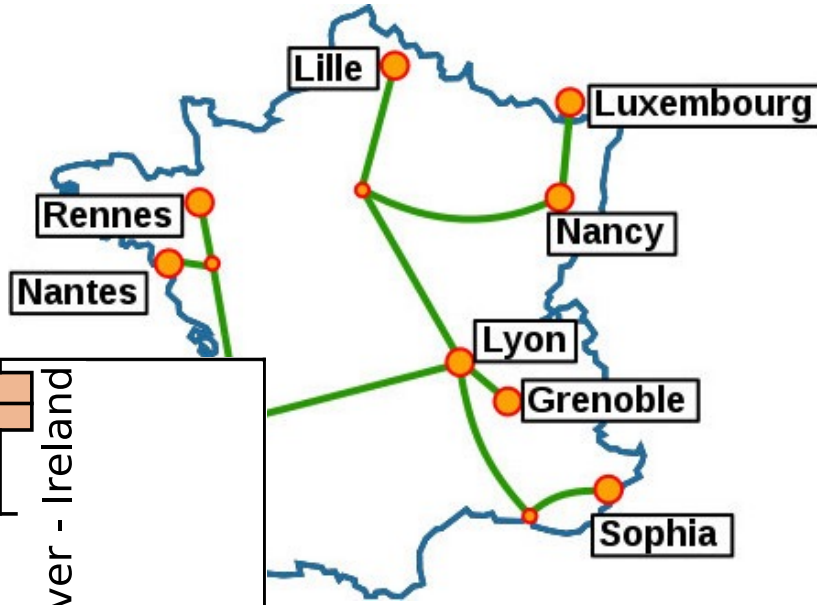


On Server / TiS latency

Latency depends on location:

- Server, TiS instances
- Client proximity



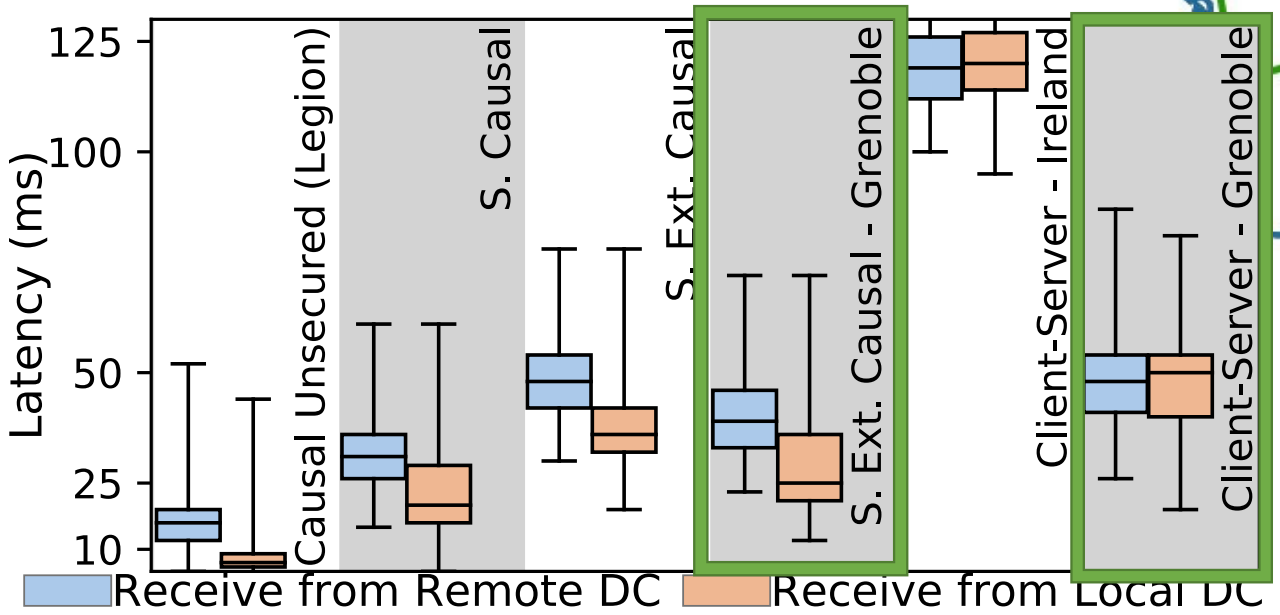


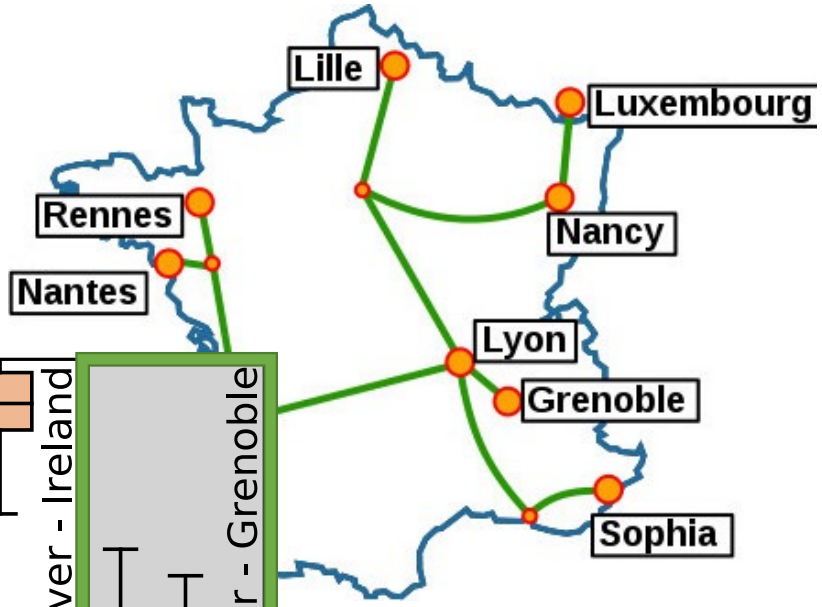
1792 clients on the Grid'5k platform

On Server / TiS latency

Latency depends on location:

- Server, TiS instances
- Client proximity





1792 clients on the Grid'5k platform

On Server / TiS latency

Running a server replica close to every group of clients might not be realistic, when there are plenty of clients.

- Replication becomes complex with too many write replicas.

Running a TiS is much simpler, as it is a stateless service. Possible locations include:

- At edge locations such as 5G towser and ISPs;
- Geo-distributed at multiple cloud points of presence;
- On client devices within secure hardware modules (e.g., Intel's SGX).

Summary

Low latency for interactive applications:

- Partial replicas at clients and direct client-client propagation

Attacks on Causal Consistency

Secure Consistency Models

Evaluation

- Low overhead compared to unsecured
- Low latency compared to client-server

Thank you!

- See our VLDB paper for:
  - Formal definitions of the Attacks on Causal Consistency
  - Formal definitions of Secure Consistency Models
  - Secure Causal Consistency
  - Secure Extended Causal Consistency
    - + Strict Secure Causal Consistency!
    - + Secure Eventual Linearizability!
  - More on how to implement!
  - Many more results
    - Impact of malicious users!