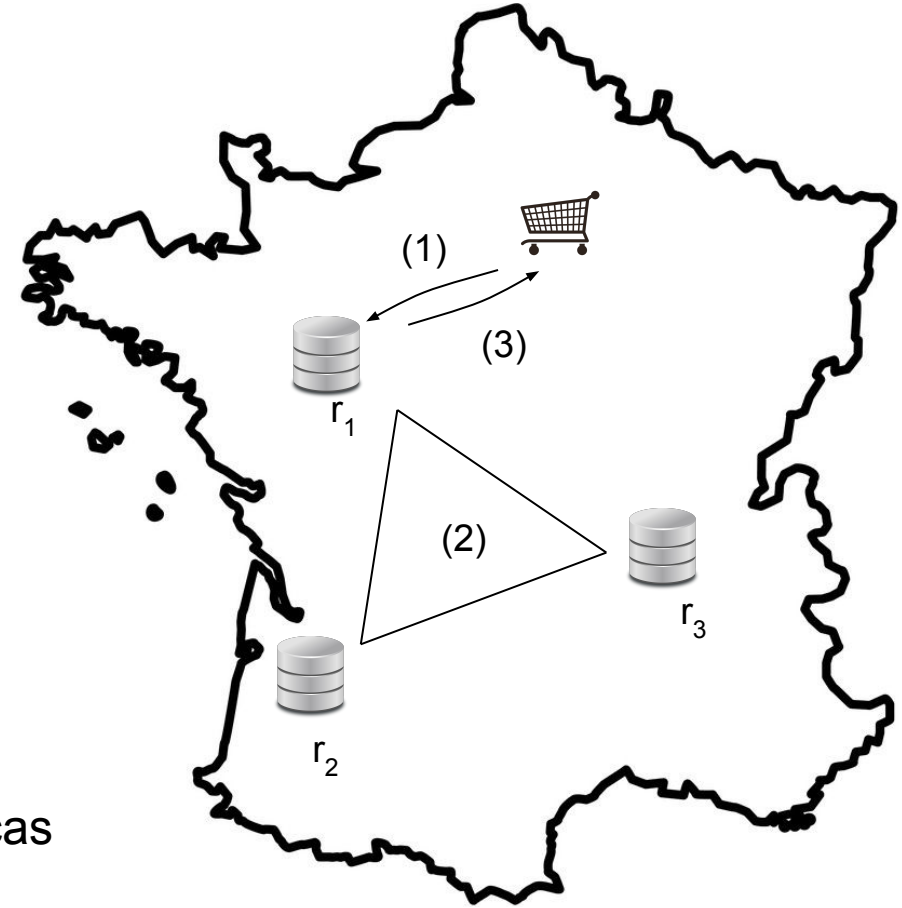
Leaderless State-Machine Replication: An Overview

Pierre Sutra

Télécom SudParis
Institut Polytechnique de Paris

RainbowFS Final Workshop, March 2022

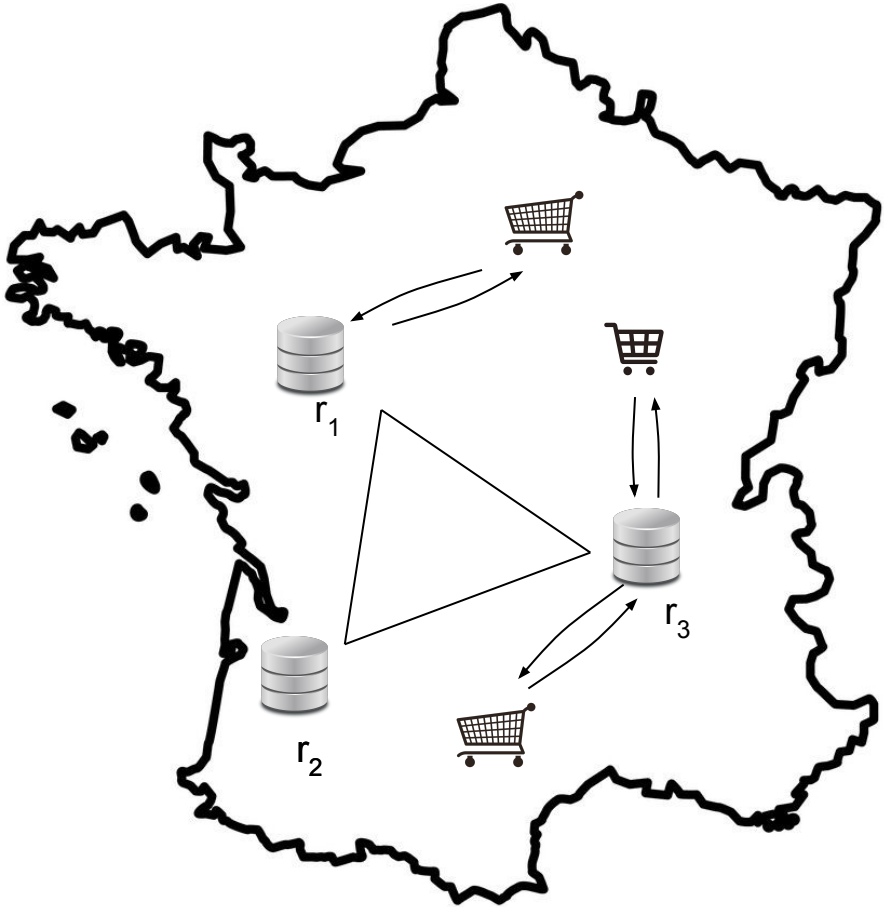
Context: geo-replication



(1) command
(2) some protocol
(3) response

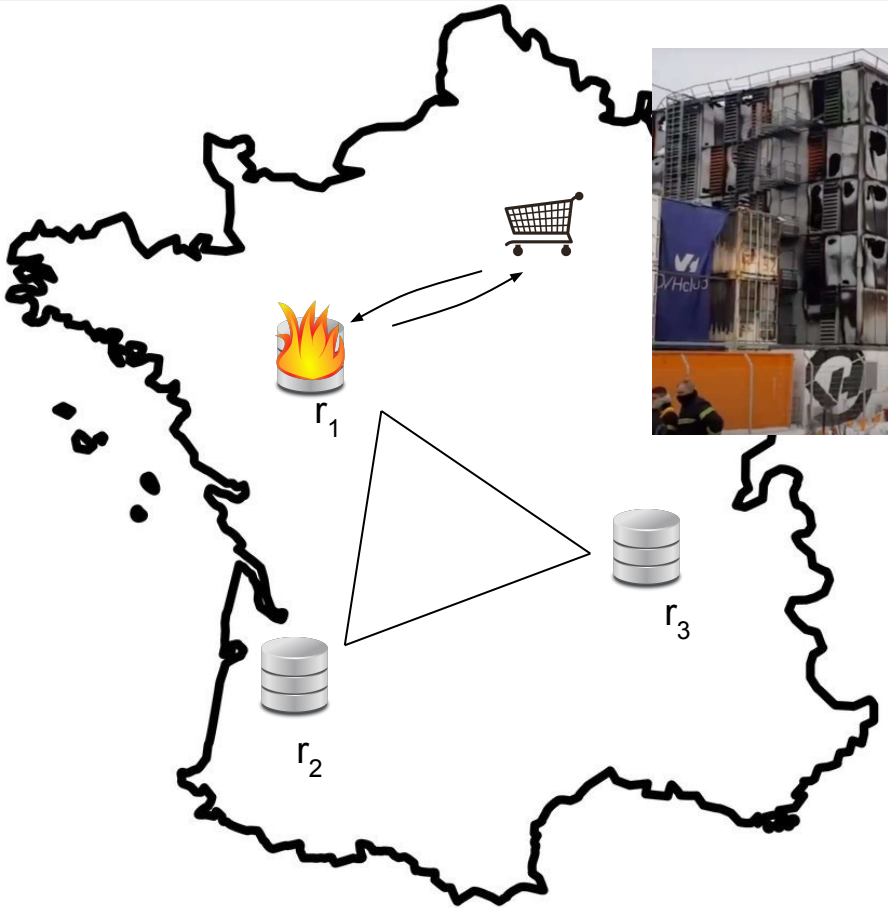
r_1, r_2, .. = data replicas

Context: geo-replication



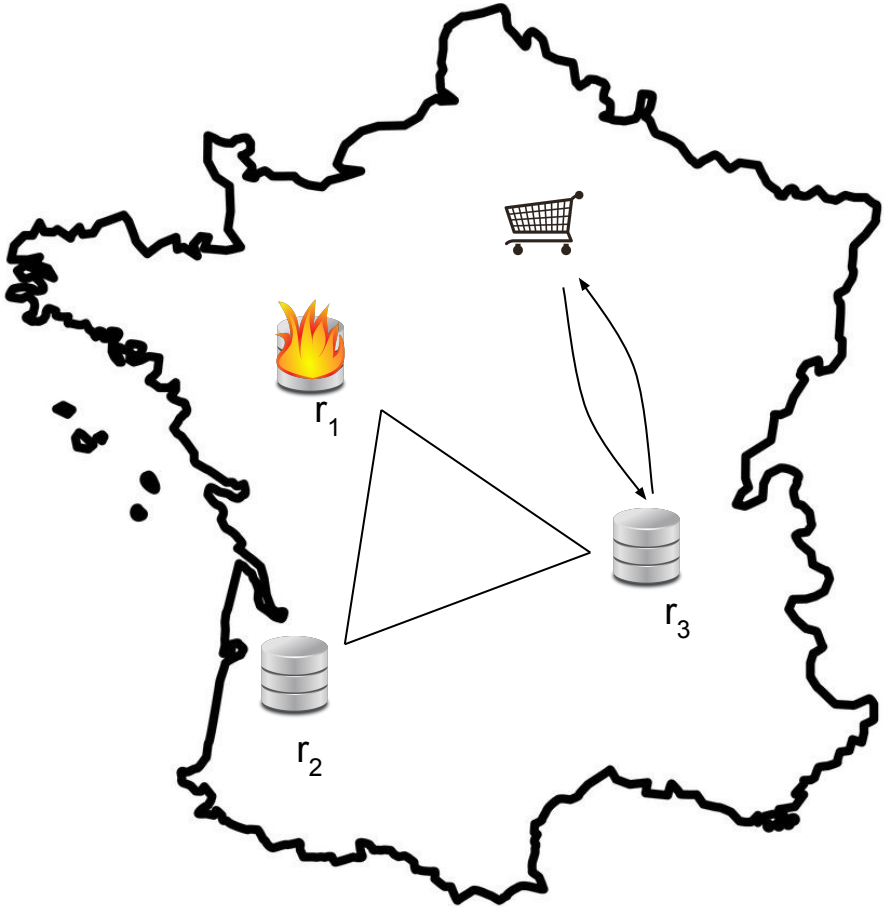
Problematic:
transparent efficient
geo-replication

Context: geo-replication



Problematic:
transparent efficient
geo-replication

Context: geo-replication



Problematic:
transparent efficient
and *robust*
geo-replication

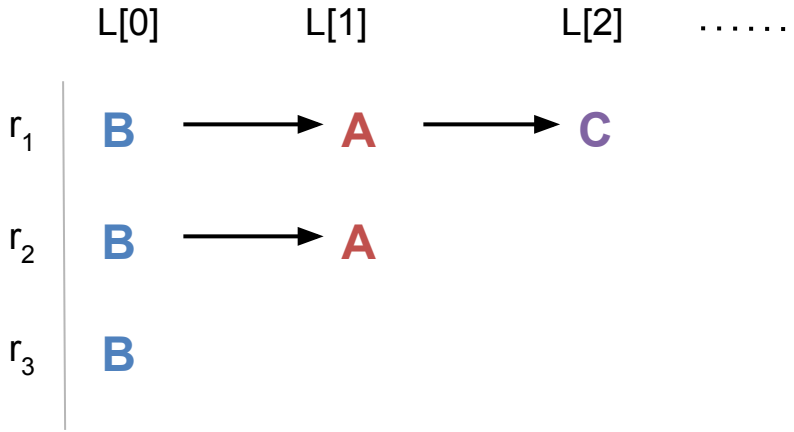
Classic State-Machine Replication [*Paxos*, *Raft*]

Each replica holds a log L

Execute commands in log order

To append a command at position L[i]

- run i-th consensus



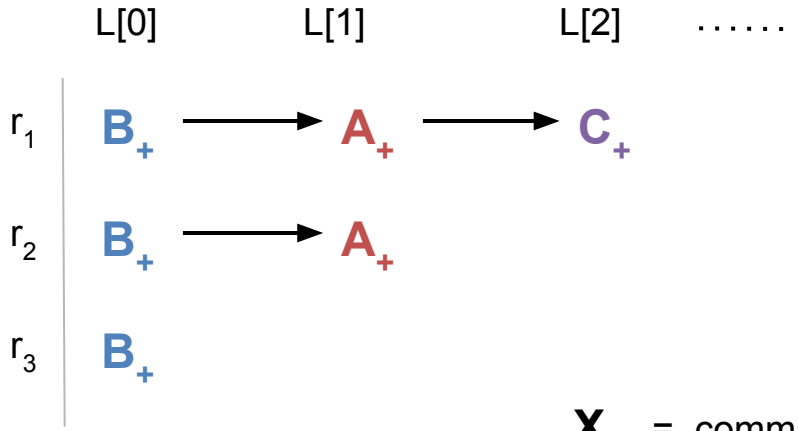
Classic SMR

Each replica holds a log L

Execute commands in log order

To append a command at position L[i]

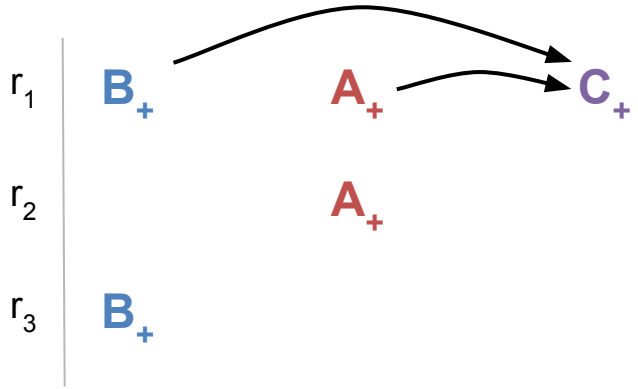
- run i-th consensus



X_+ = command is executed

Generic SMR [*GPaxos*, *GBcast*]

Execute *non-commuting* commands in the same order in the log



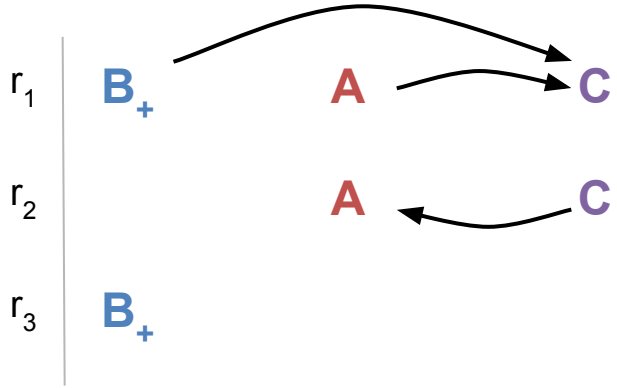
A = x ← 42

B = y ← 7

C = z ← x + y

Leaderless SMR [*DISC'05*, *SOSP'13*]

Execute *non-commuting* commands according to the same graph



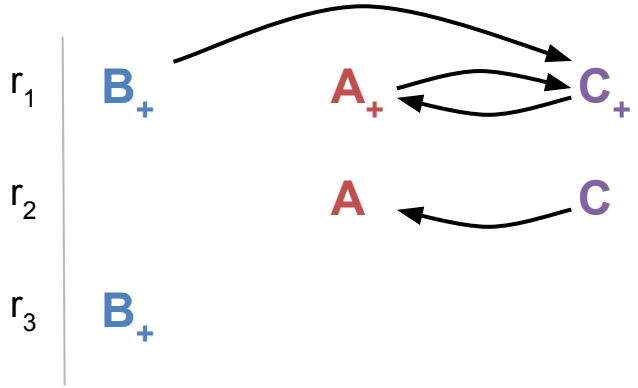
dep(A) = {C}

dep(C) = {B, A}

dep(B) = ⌀

Leaderless SMR

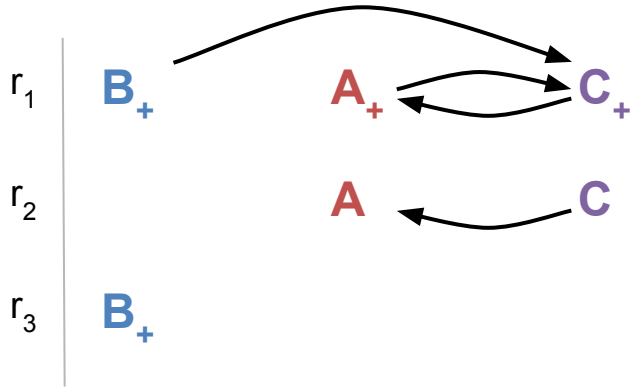
Execute *non-commuting* commands according to the same graph



- operation **X** executed once dep(**X**) transitively closed
- cycles are broken deterministically

Leaderless SMR

Execute *non-commuting* commands according to the same graph

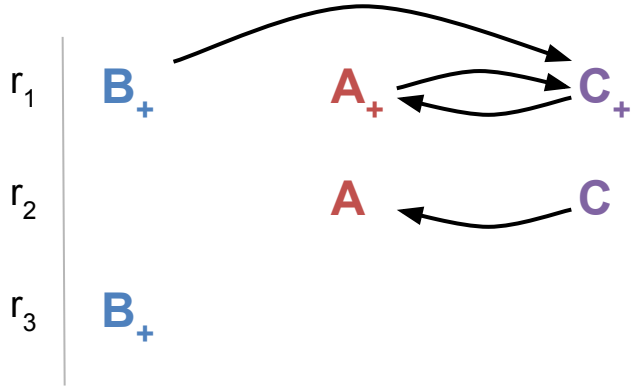


Properties

- replicas agree *on* dep(**X**)
- (**X**, **Y**) non-commuting then **X** ∈ dep(**Y**) or **Y** ∈ dep(**X**)

Leaderless SMR

Execute *non-commuting* commands according to the same graph



Properties

- replicas *agree on* dep(**X**)
- (**X**, **Y**) non-commuting then **X** ∈ dep(**Y**) or **Y** ∈ dep(**X**)

Egalitarian Paxos [*SOSP'13*]

EPaxos uses 2f+1 processes (f = max #failures)

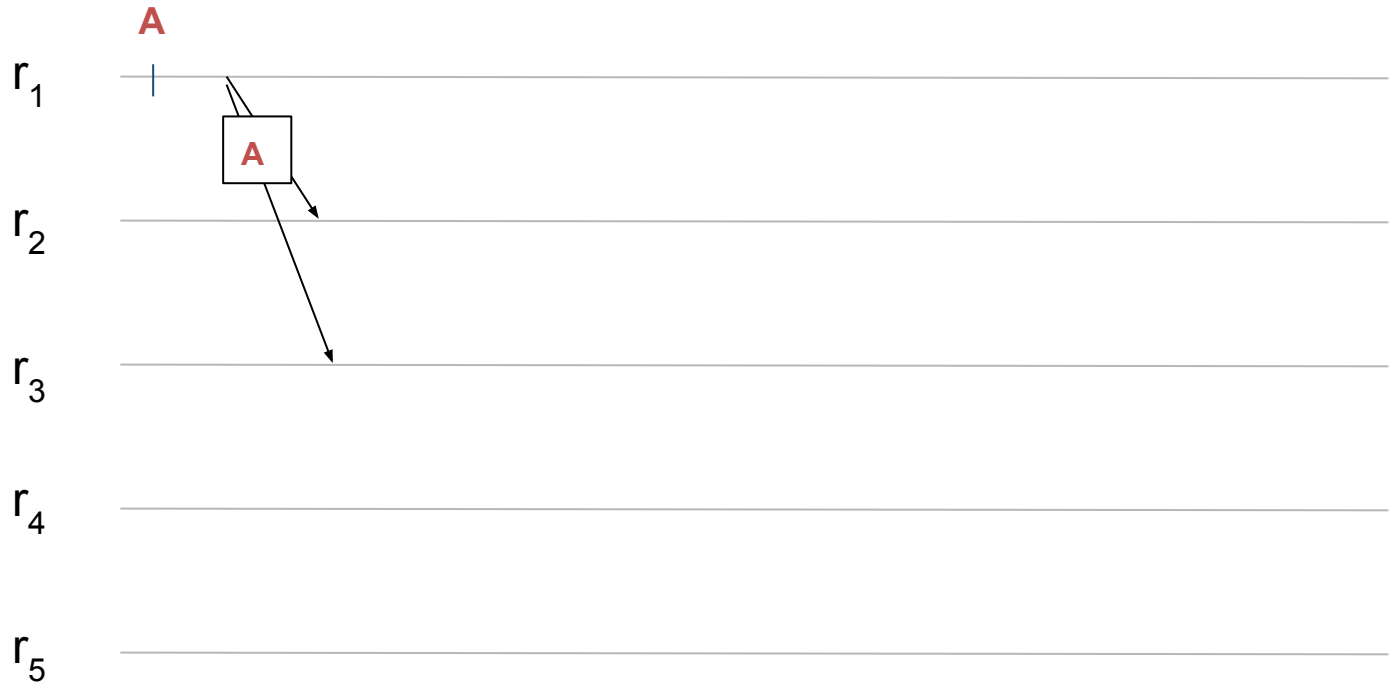
When a client executes command **X**

- pick a replica
- this replica is the *coordinator* for **X**, coord(**X**)
- coord(**X**) runs consensus over dep(**X**)

To do consensus on dep(**X**)

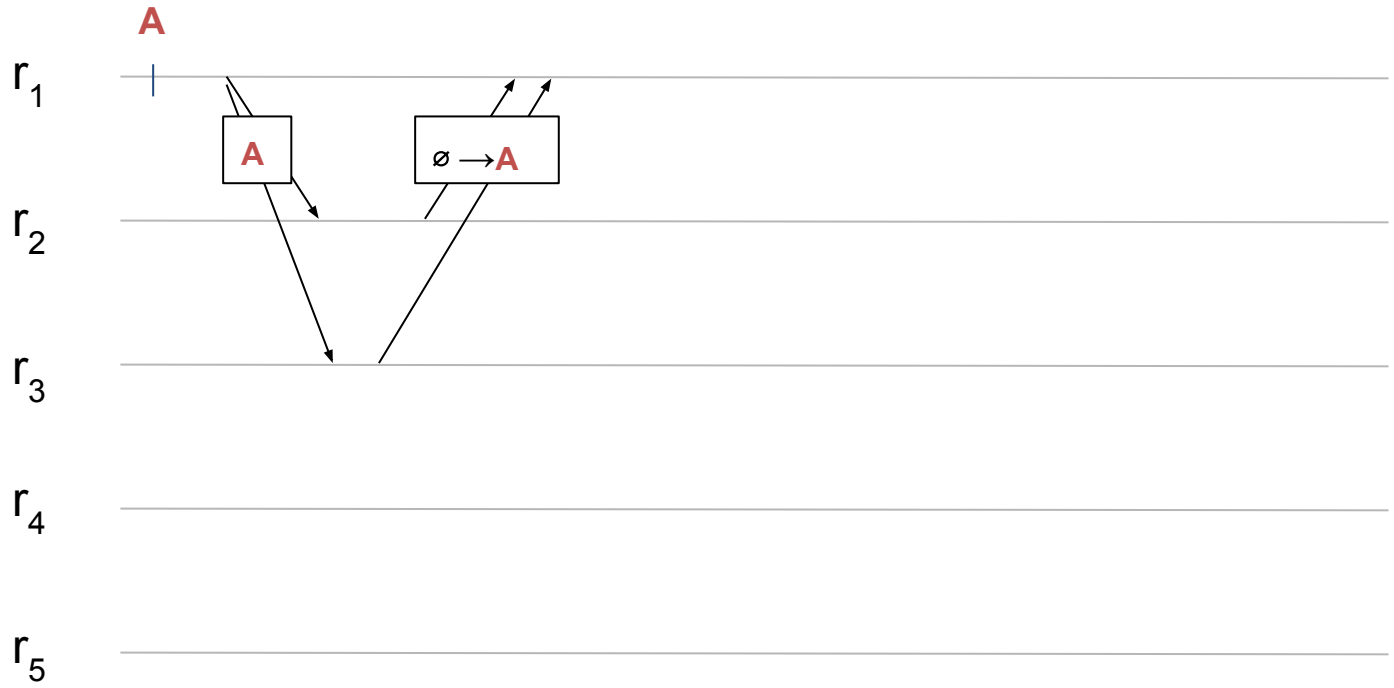
- try to agree spontaneously by contacting a fast quorum (f+f/2 replicas)
  - when contacted, a replica sent back the commands conflicting with **X** seen so far
- if this fails, ask a slow quorum (f+1 replicas)
  - in this slow path, the union of the reported deps by the fast quorum is used

EPaxos



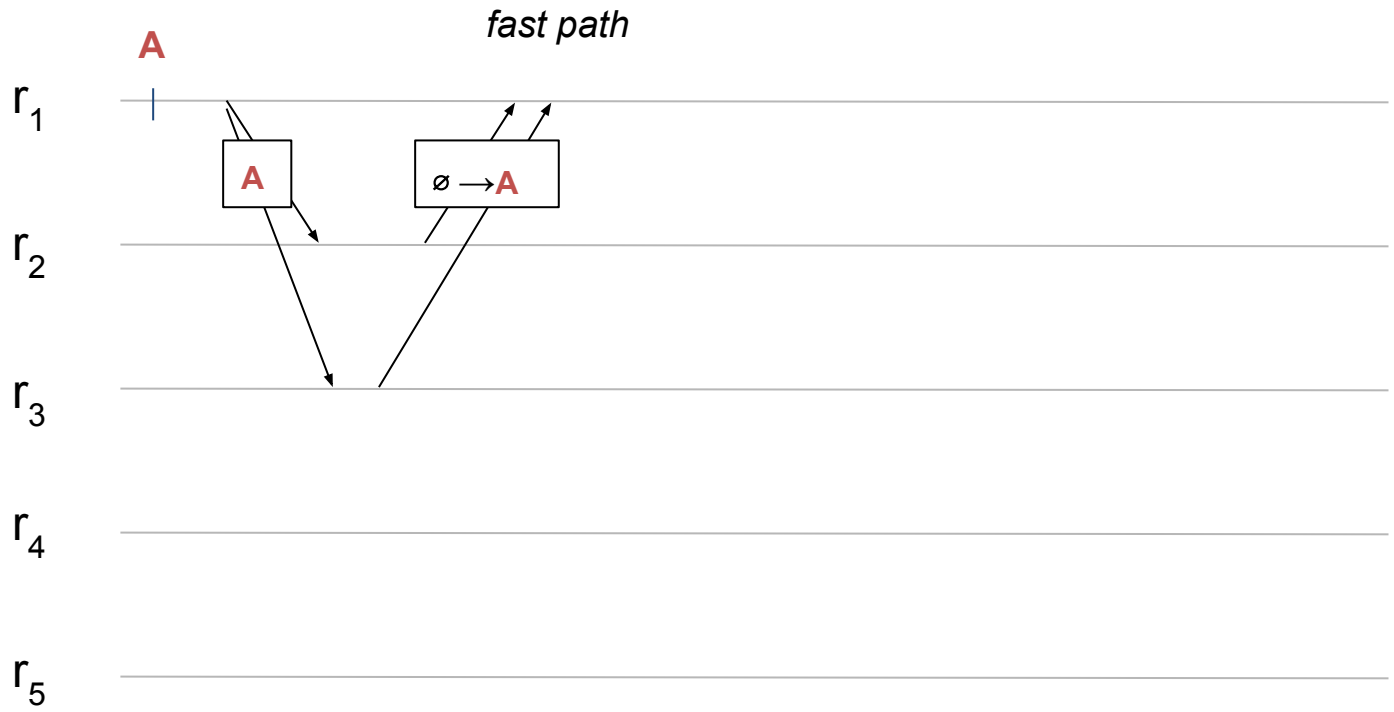
(n=5, f=2)

EPaxos



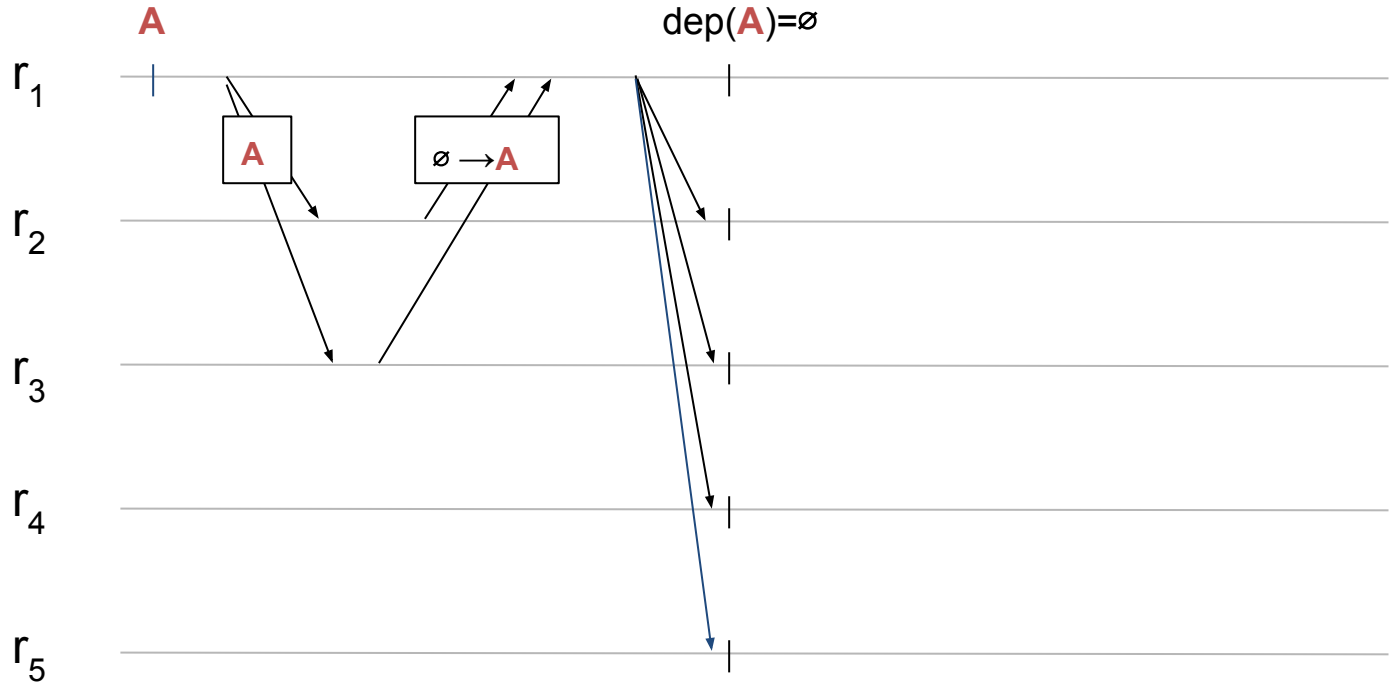
(n=5, f=2)

EPaxos



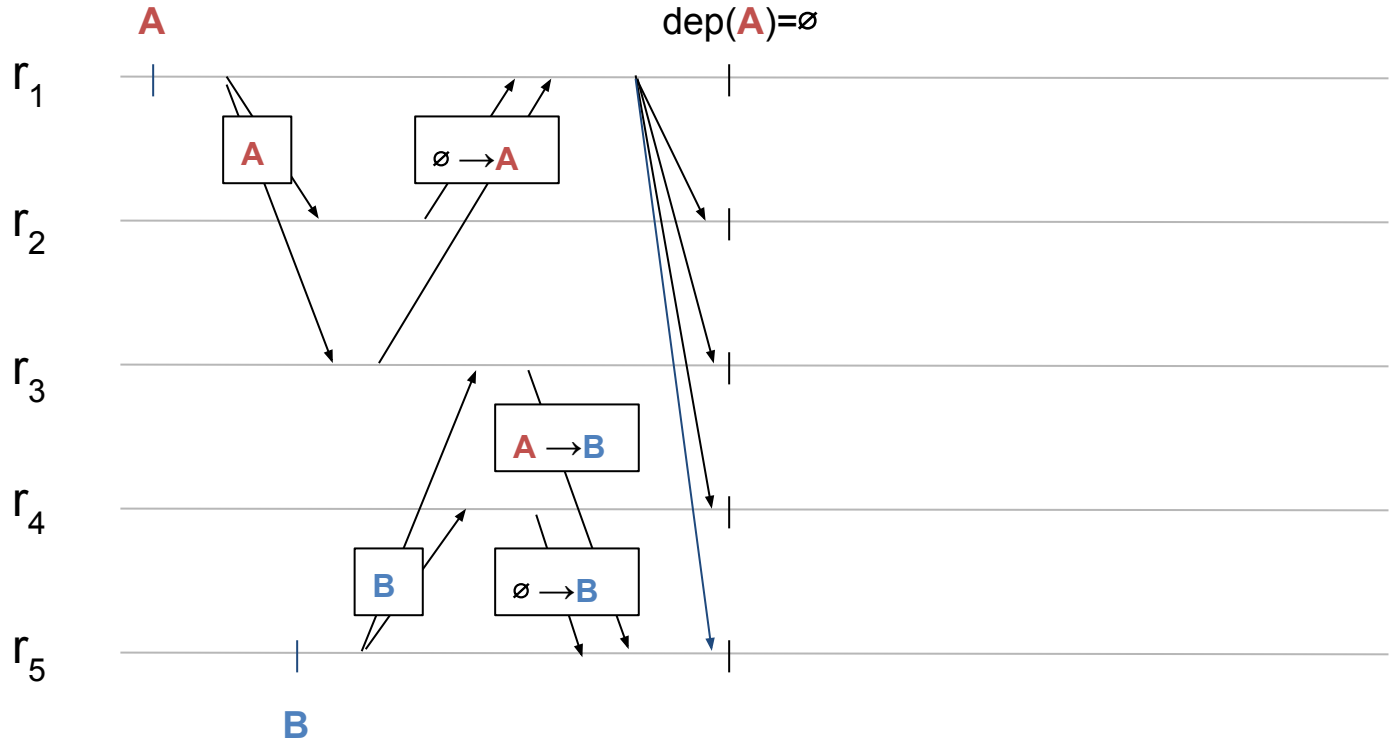
(n=5, f=2)

EPaxos



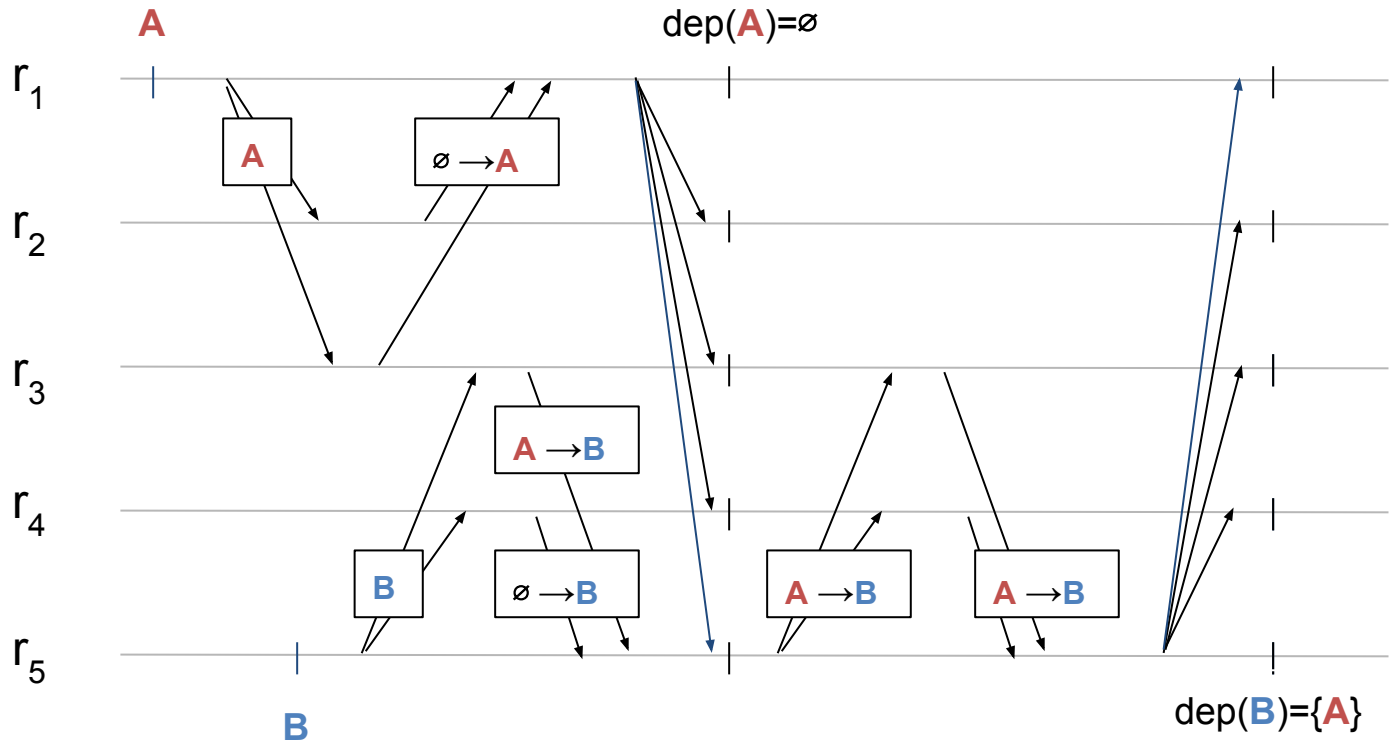
(n=5, f=2)

EPaxos



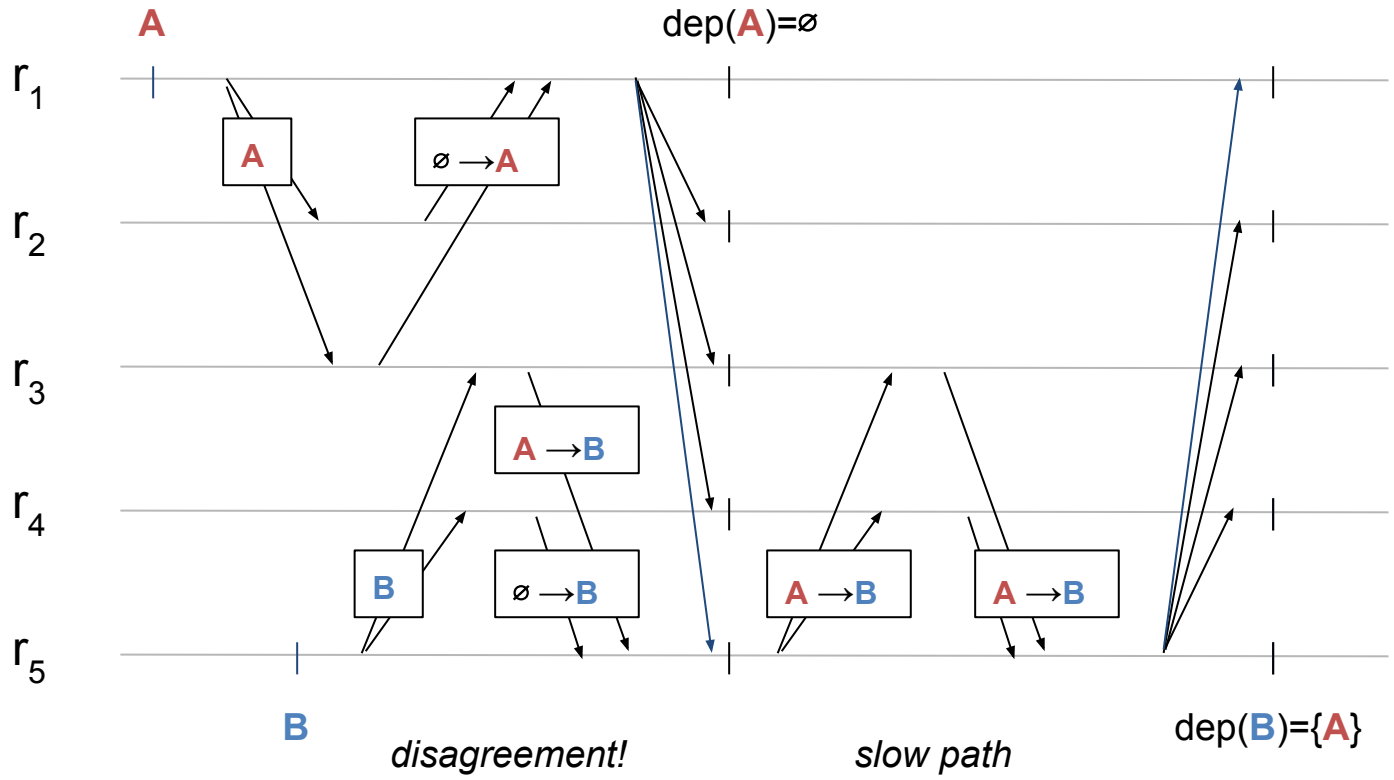
(n=5, f=2)

EPaxos



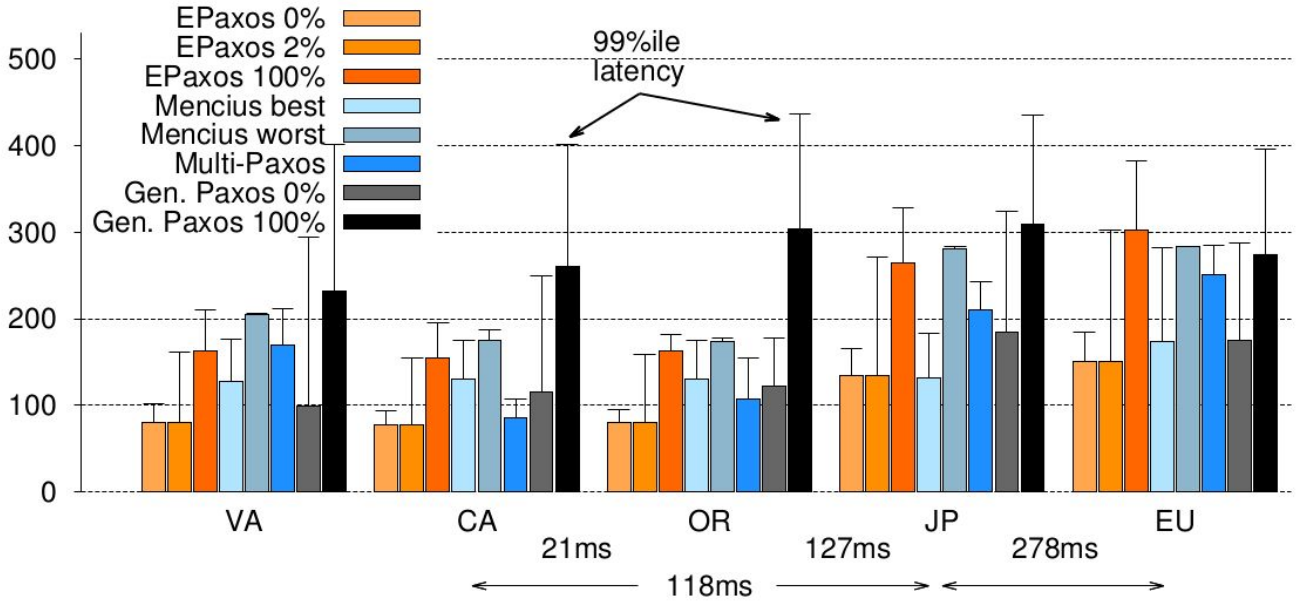
(n=5, f=2)

EPaxos


A
dep(A)=⌀
r1
r2
r3
r4
r5
A
⌀ →A
A →B
B
⌀ →B
A →B
A →B
B
disagreement!
slow path
dep(B)={A}

(n=5, f=2)

EPaxos - AWS experiments



Takeaways:

- leaderless SMR is faster and more fair
- but needs most commands commute (EPaxos 100% is bad)

avoid disagreement

how? threshold union

consider a bag of items E, the k-threshold union of E, written $\biguplus_k E$, are the items reported at least k+1 times in the sets of E

formally,

$$\biguplus_k E = \{ Y : count(Y) \geq k+1 \}$$

$E = \{E_1, E_2, E_3\}$ with $E_1 = \{A,B,C\}$, $E_2 = \{A,C\}$ and $E_3 = \{A\}$

then

- $\biguplus_1 E = \{A,C\}$,
- $\biguplus_2 E = \{A\}$,

Atlas

avoid disagreement

how? threshold union

EPaxos fast path condition:

given $q \in Q$, let dep_q be the dep. reported by q

then

fast-path **iff** $\forall\ q,p \in Q.\ dep_q = dep_p$

Atlas

avoid disagreement

how? threshold union

Atlas fast path condition:

given $q \in Q$, let dep_q be the dep. reported by q

then

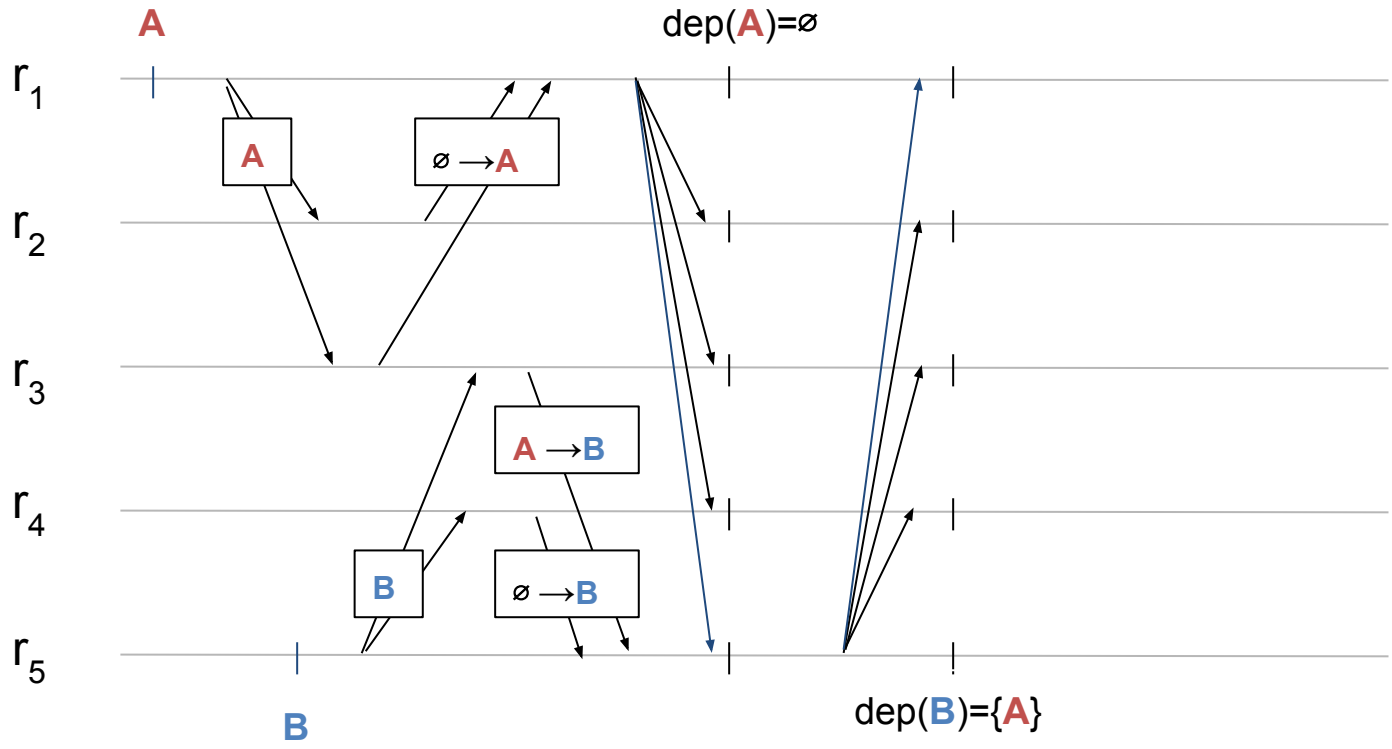
fast-path **iff** $\biguplus_f Q = \bigcup_q dep_q$

(=every dep. is reported at least f+1 times)

why this works?

- if a failure occurs, the dep. reported by <u>any</u> majority quorum in Q is exactly $\biguplus_f Q$

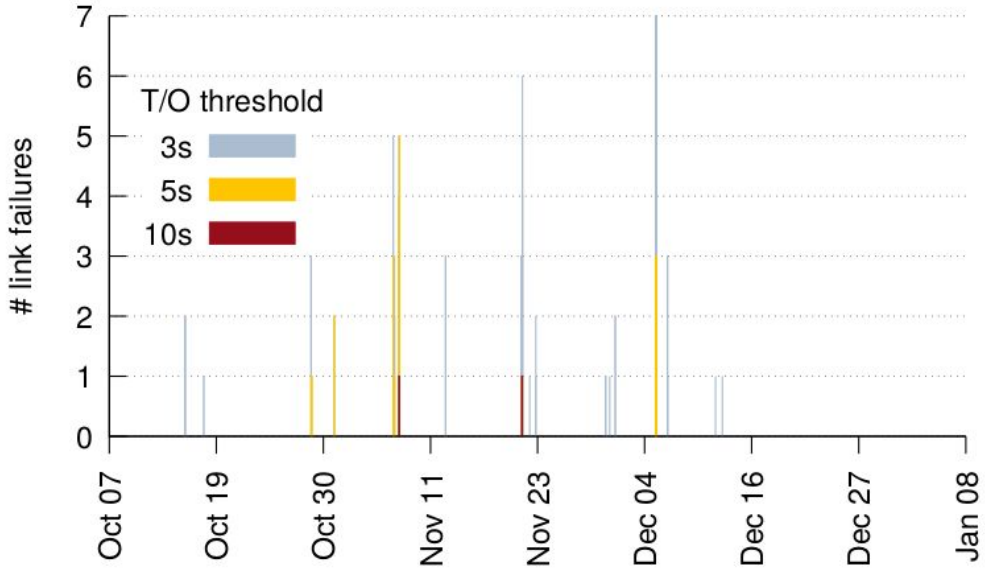
Atlas



** the coordinator takes the union of the reported deps.*

(n=5, f=1)

Atlas - asynchrony in practice

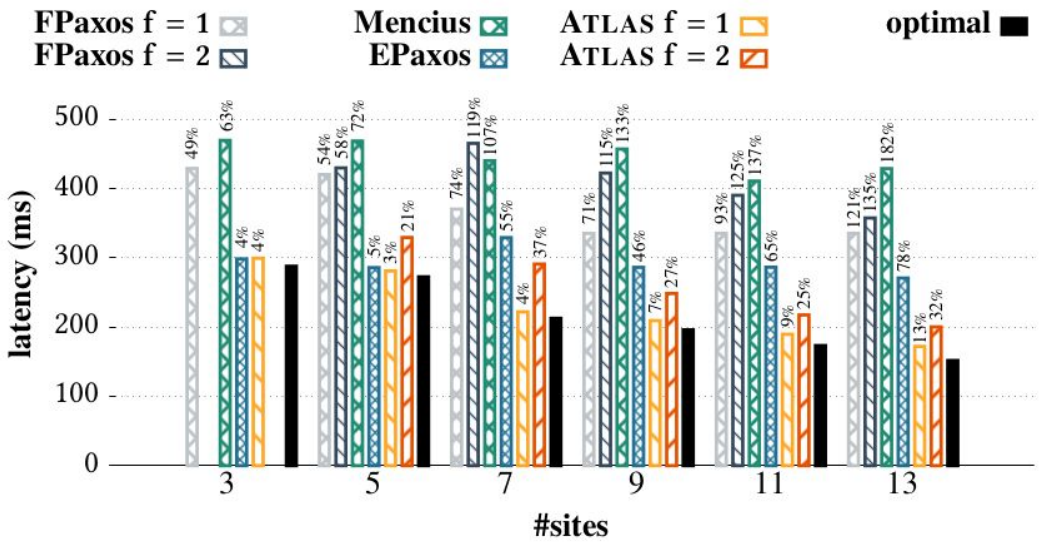


13 GCP sites
all-to-all ping
over 3 months

Takeaways:

- concurrent link failures is a rare event at scale
- at most one slow site during the exp. (f=1)

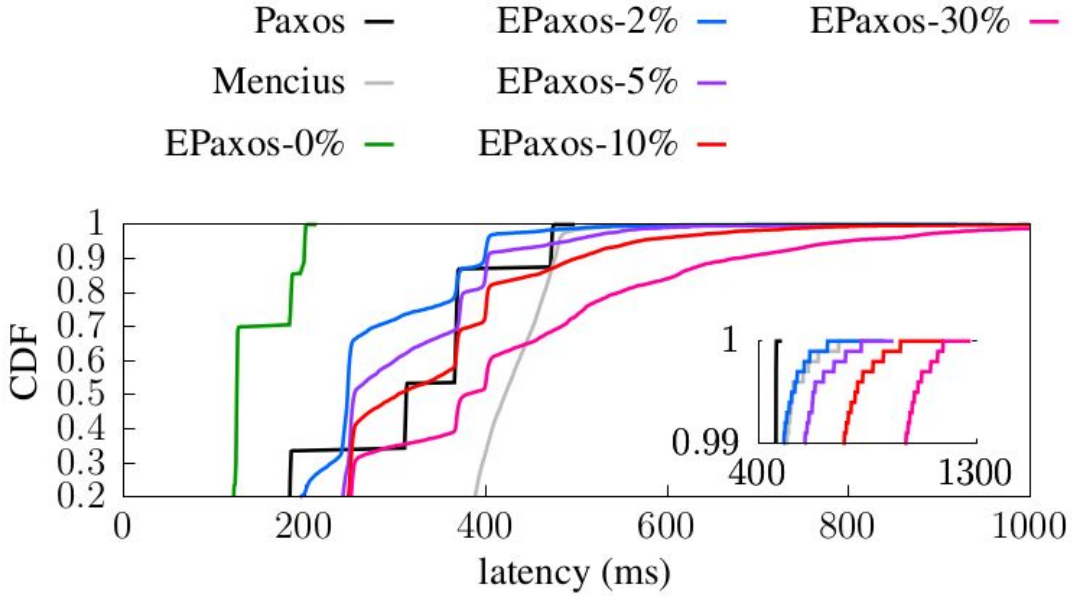
Atlas - GCP experiments



Takeaways:

- Atlas better than EPaxos for large-scale deployment (n ≥ 5)

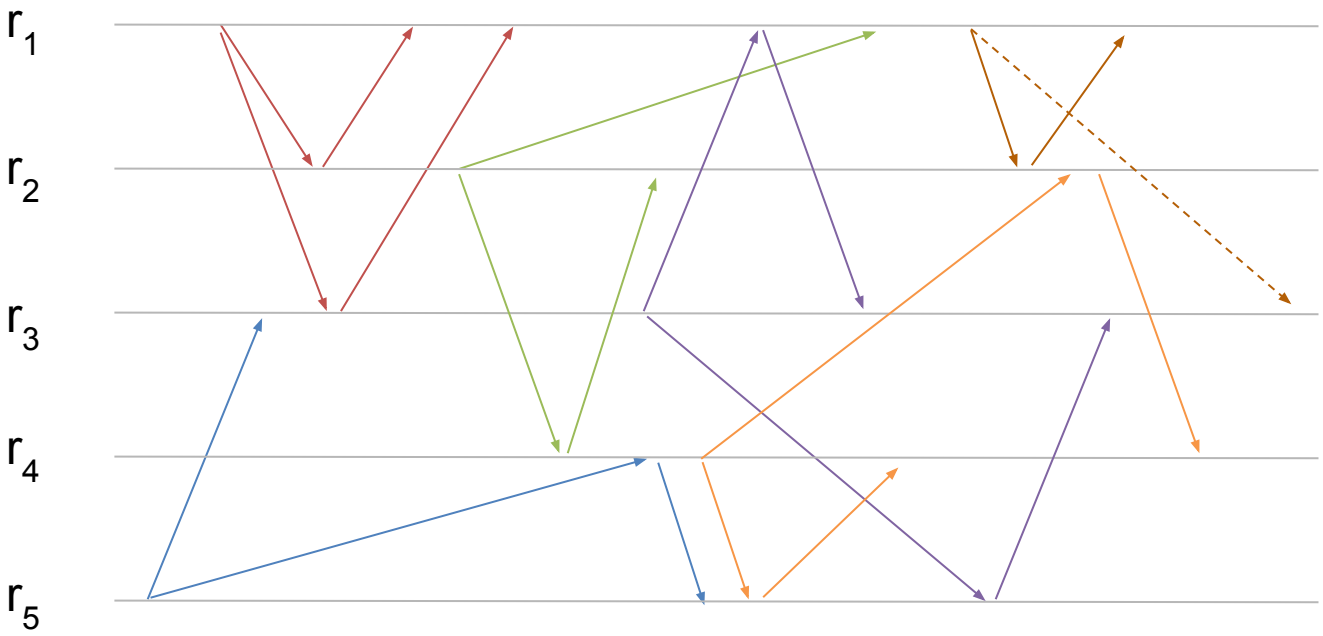
Tail latency [*DISC'20*, *NSDI'21*]

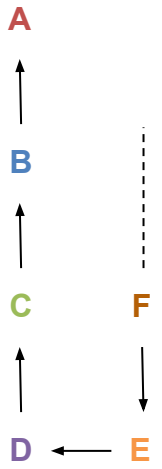


Takeaways:

- Tail latency in leaderless SMR protocols is a problem

Tail latency





(n=5, f=1)

Tempo [*Eurosys'21*]

tame tail latency

how? agree on a timestamp per command

+ make the timestamp stable

Tempo fast path condition:

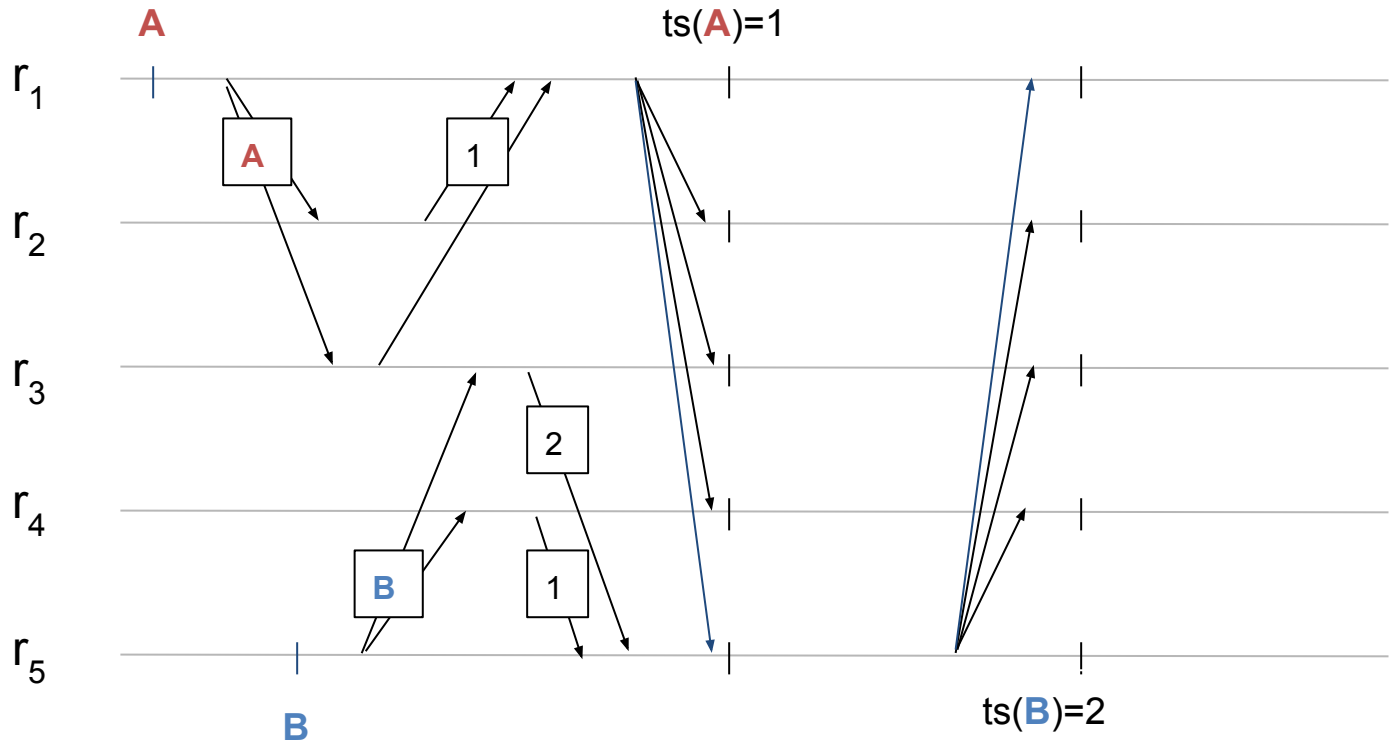
given $q \in Q$, let ts_q be the timestamp reported, or *promised,* by q

then

fast-path **iff** let $t = \max\{ ts_q : q \in Q\}$)

then count(t) ≥ f+1

Tempo



(n=5, f=1)

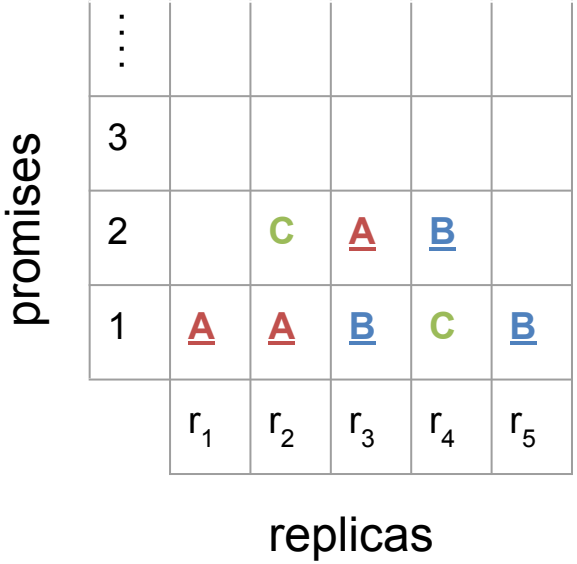
Tempo - background stability mechanism

A command is stable once

- its timestamp, say t, is committed,
- every command with a timestamp lower (or equal) to t is stable
- a quorum reports promises higher (or equal) to t

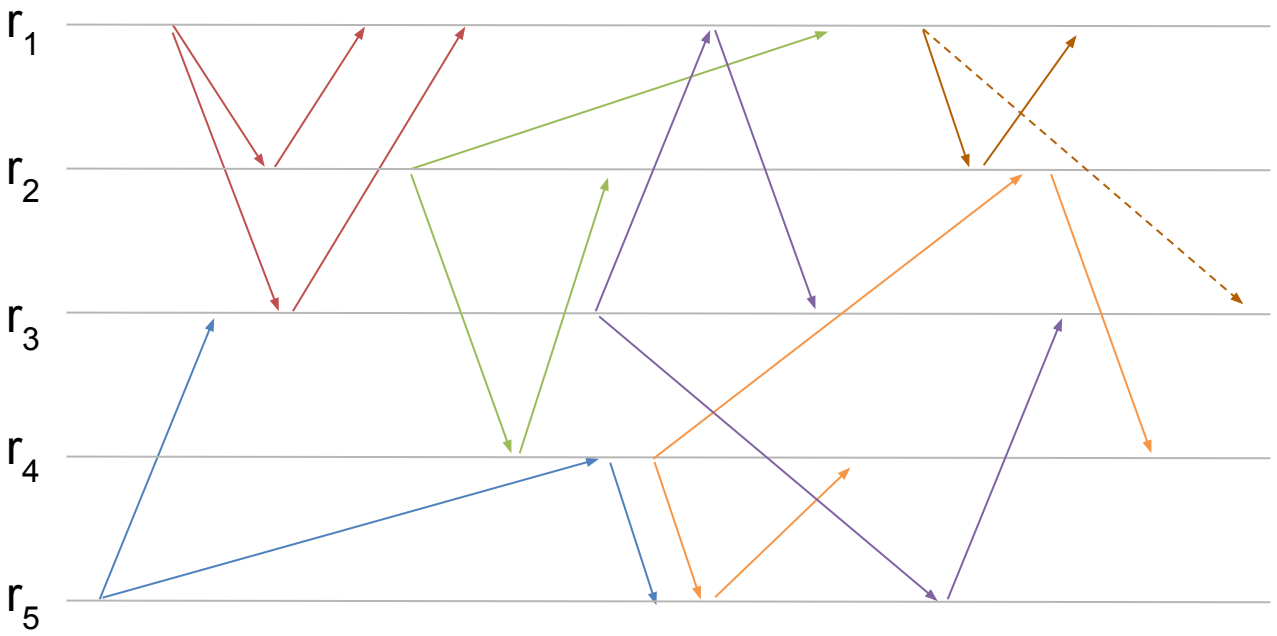
Stable commands are executed in the order of timestamps (ties are broken arbitrarily)

Here, **A**;**B**
as ts(**A**) = ts(**B**) and **A** < **B**



X = command is stable

Tempo - background stability mechanism

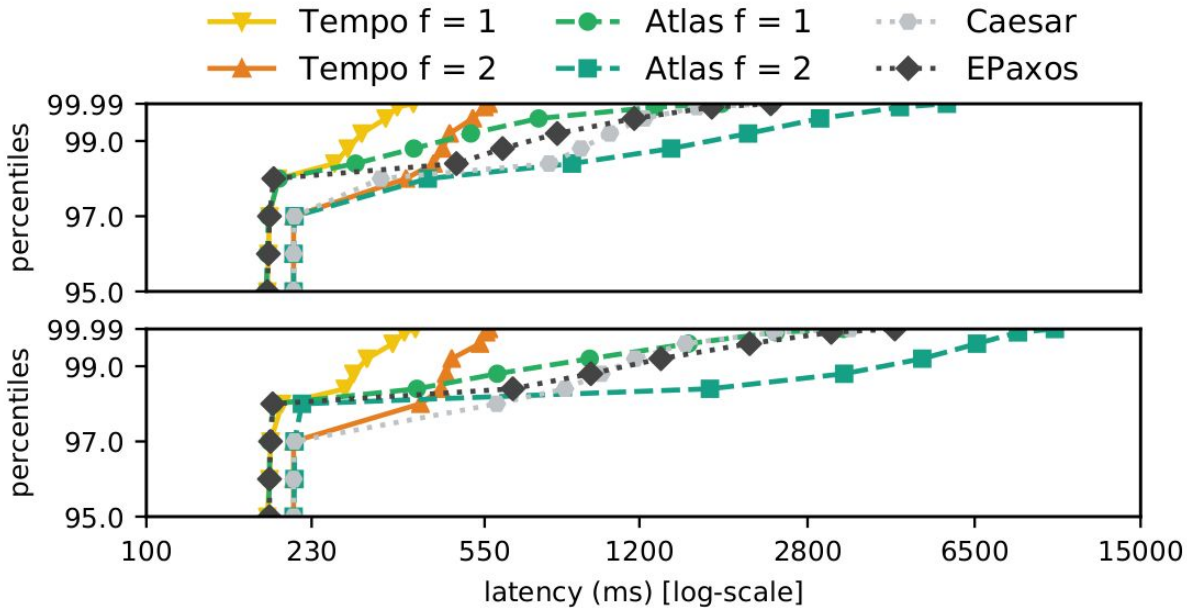


⋮					
3	C		D		D
2	D	C	A	B	E
1	A	A	B	C	B
	r_1	r_2	r_3	r_4	r_5

A;B;C

(n=5, f=1)

Tempo



Takeaways:

- Tempo improves tail latency in leaderless SMR

5 GCP sites
512/256 (top/bottom) clients per site
conflict rate is 2%

Conclusion

Leaderless SMR

- graph-based ordering of commands
- a coordinator per command **X**
  - runs consensus on dep(**X**)
- faster and more fair than Paxos/Raft

Future directions

- scalability
- BFT (blockchain)