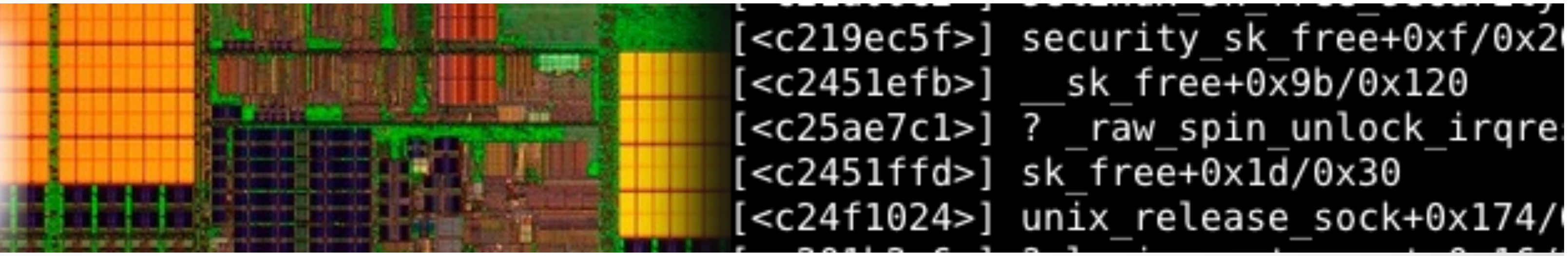
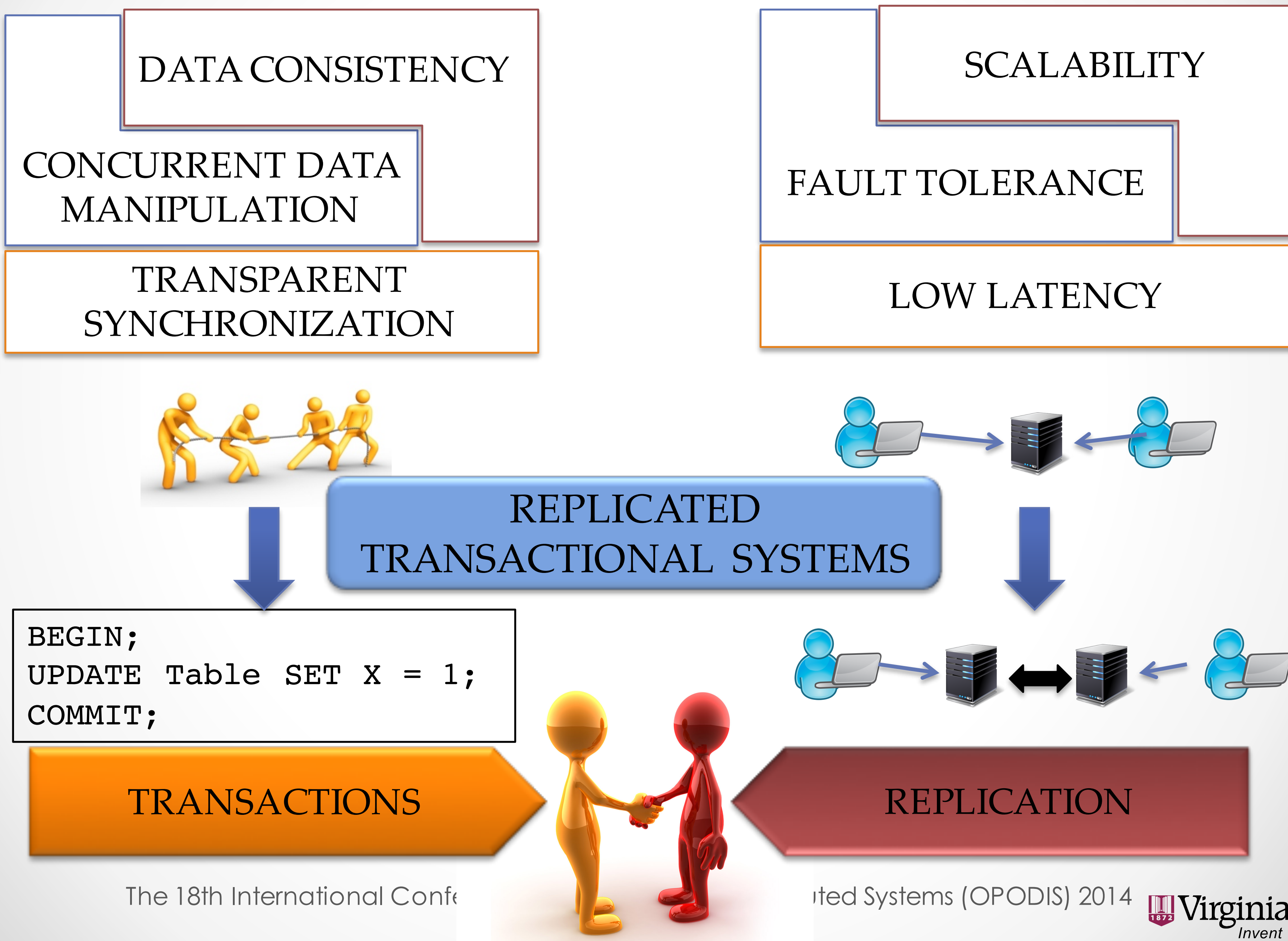


Be General and Don't Give Up Consistency in Geo-Replicated Transactional Systems

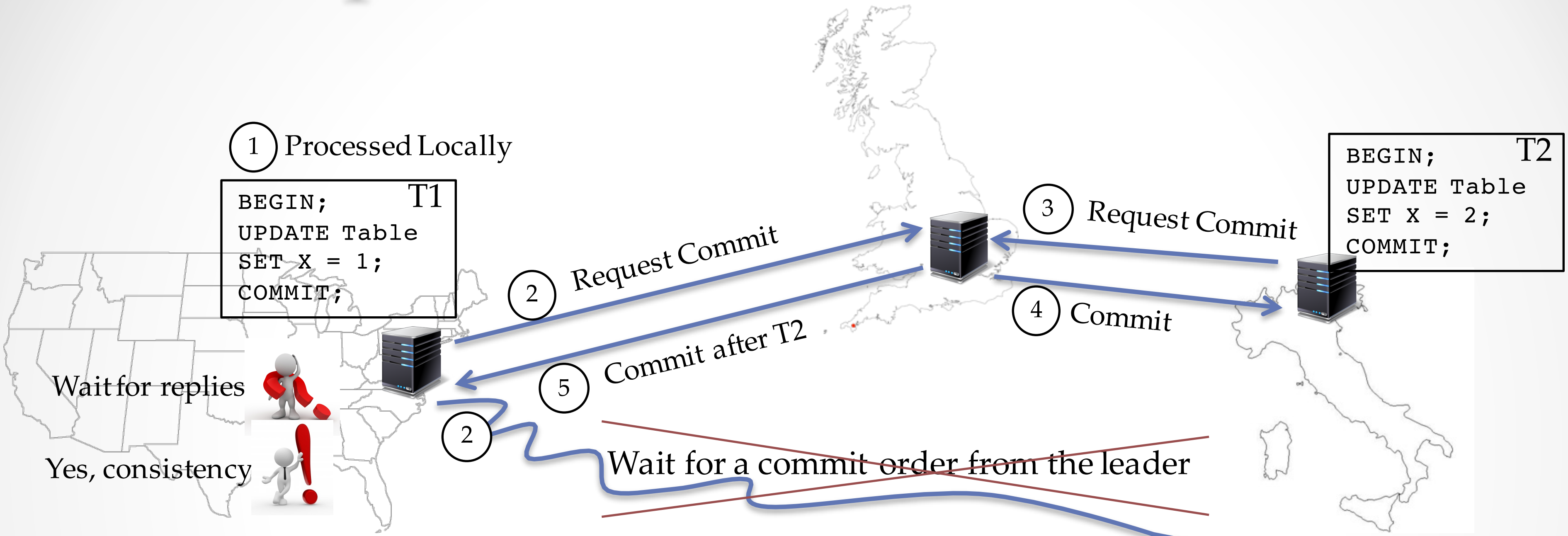
Alexandru Turcu, Sebastiano Peluso,
Roberto Palmieri and Binoy Ravindran

The 18th International Conference on Principles of Distributed Systems (OPODIS) 2014

Replicated Transactional Systems



Geo-Replication: the Whole Picture



- Inter-site delays are predominant: minimize the protocol communication delays.
- Strong Consistency: avoid replicas divergences and provide transparency to the programmers.
- Non-uniform delays: do not define specialized roles for sites because delays can vary and are not uniform.

Key Design Principles

Consistency

- Correct state transitions on all replicas: conflicting transactions committed according to a common order at all replicas.

Require Consensus

Synchrony

- No assumptions on inter-site delays and replicas' clocks speed

Latency

- No partial replication data model
- 2 per-transaction communication delays in case of no conflicts

- Coordination either at the beginning or during the commit.
- No less than 2 communication delays due to the lower bound on consensus.

Parallelism

- No coordination among non-conflicting transactions.
- No designated sites with specialized roles.

No leader-based consensus

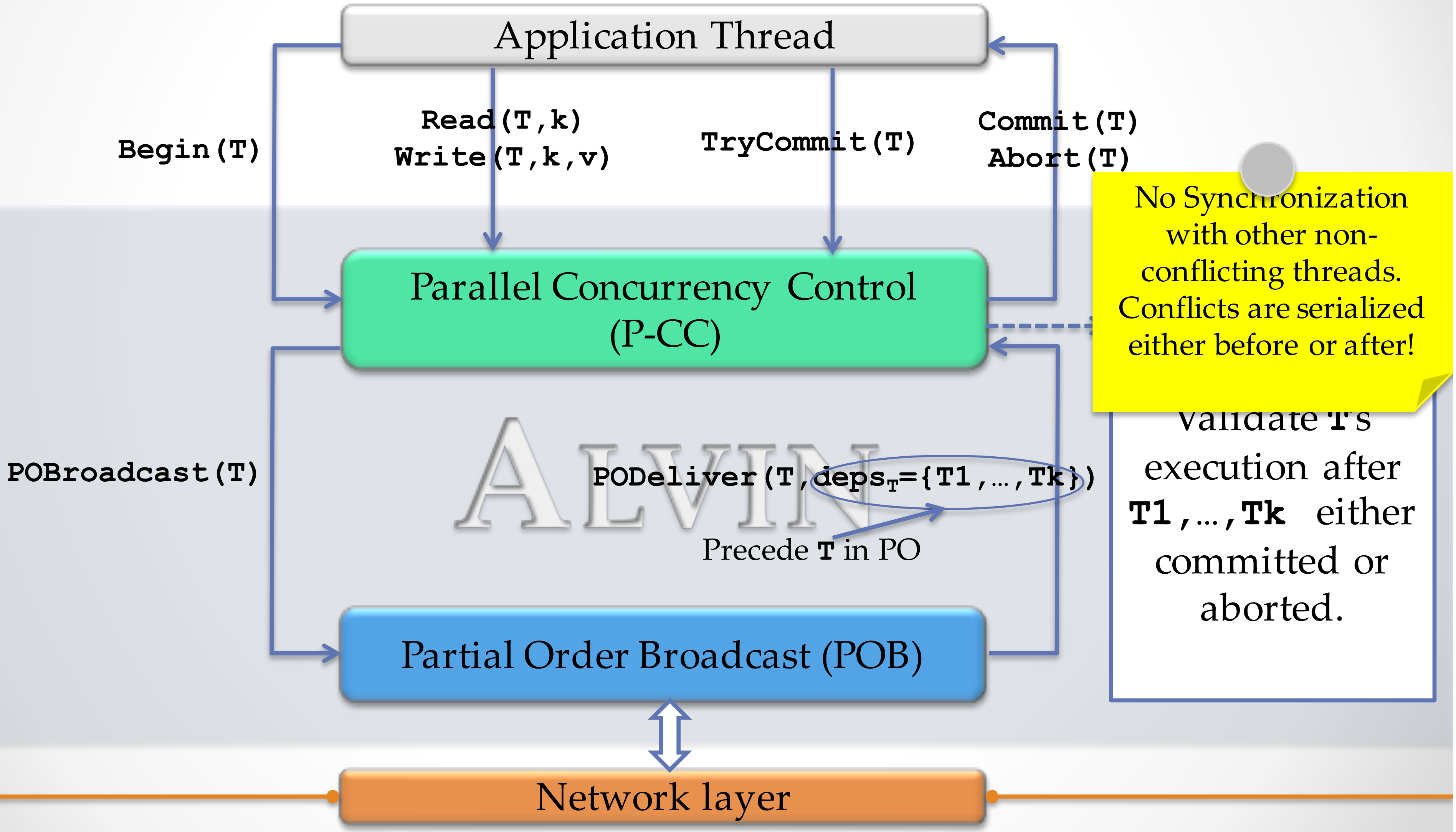
Related Work

- Strong Consistency and efficient transaction execution for restricted types of transactions:
  - Lynx: transactions for piecewise execution [SOSP13a]
  - EPaxos: single operations and write-only transactions [SOSP13b]
- Efficient transaction execution of general-purpose transactions for lower consistency:
  - ChainReaction: Causal+ consistency [EuroSys13]
  - Walter: Parallel Snapshot Isolation consistency [SOSP11]
  - Jessy: Non-Monotonic Snapshot Isolation consistency [SRDS13]
- Consistency for general-purpose transactions on specialized architectures:
  - Spanner: absolute time and uncertainty by relying on specialized hardware components as clock references, i.e., GPS and atomic clocks [TOCS13]

A Step Towards Low-Latency

- Egalitarian Paxos (EPaxos) [SOSP13b]:
  - Multiple leaders and quorum-based
  - Non-blocking if at most *f* faults (where $N=2f+1$)
  - Generalized Consensus a.k.a. it only cares about agreement on conflicting transactions
  - Commit in 2 communication delays if no conflicts
  - Communication is only a part of the story. Consensus reached through graph analysis on dependencies exchanged during communication
- Mencius [OSDI08]:
  - Multiple leaders and possibility of Generalized Consensus
  - Communication phase fully exploited: participants agree on a commit position proposed by the transaction's leader
  - Blocking: a position *p* has to hear about positions less than *p*

ALVIN: Key Ingredients



Partial Order Broadcast

Main Idea

S associated with P_i if S mod N = i, where N number of sites

T_a	T_b	T_c		T_d
P_0	P_1	P_2	P_3	P_4

T_b conflicts with T_a

T_c conflicts with T_d

T delivered in position S if
∀ conflicting T′ delivered in position S′ > S,
$T \in deps_{T'}$, and $T' \notin deps_T$

S	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
			X																	
deps	{ }	{ T_a }			{ }			{ T_d }												

POB: Properties

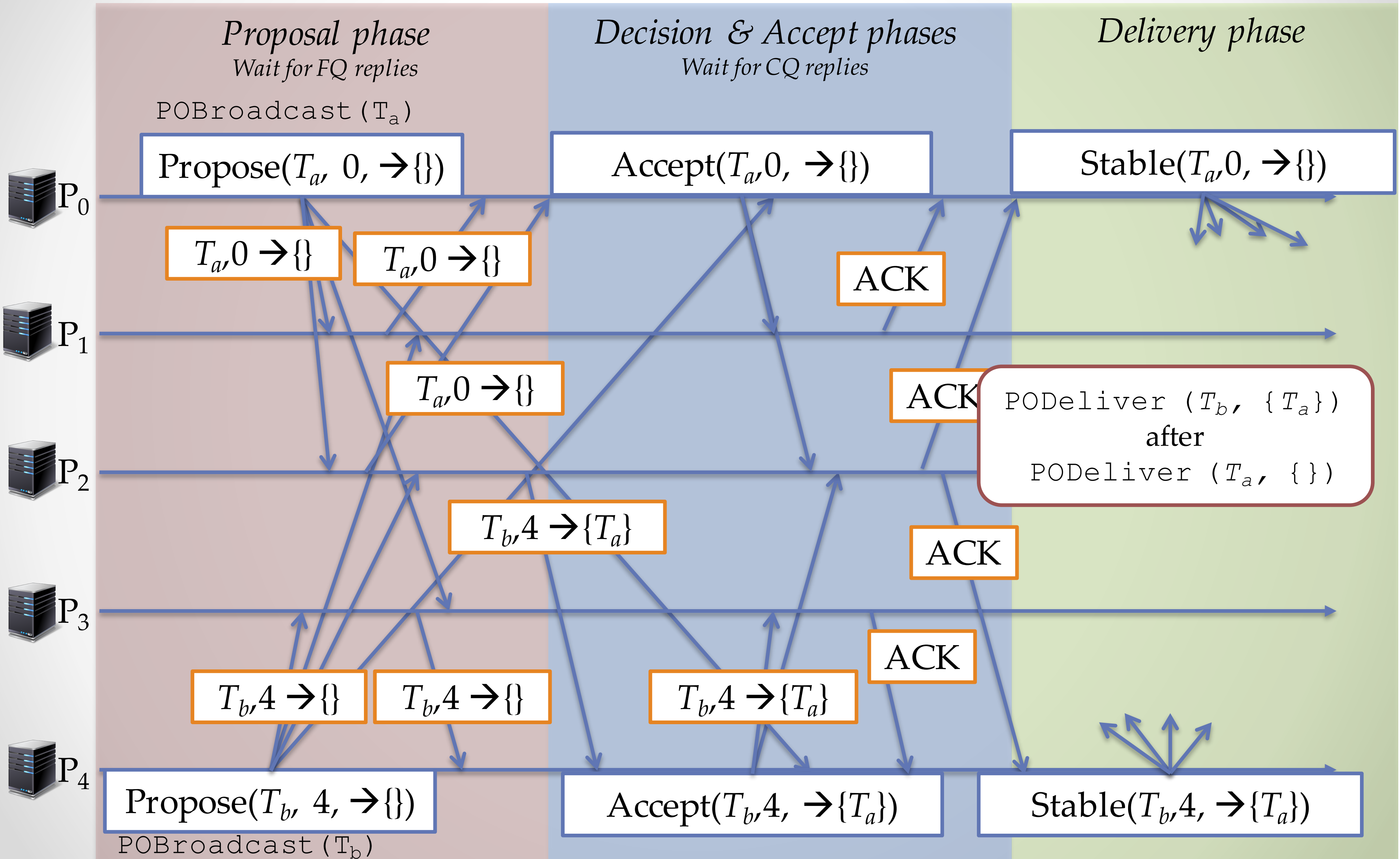
Strong Uniform Conflict Order

If some node executes *PODeliver*(T,$deps_T$) before *PODeliver*(T',$deps_{T'}$), and T and T' are conflicting, then every node executes *PODeliver*(T',$deps_{T'}$) only after *PODeliver*(T,$deps_T$).

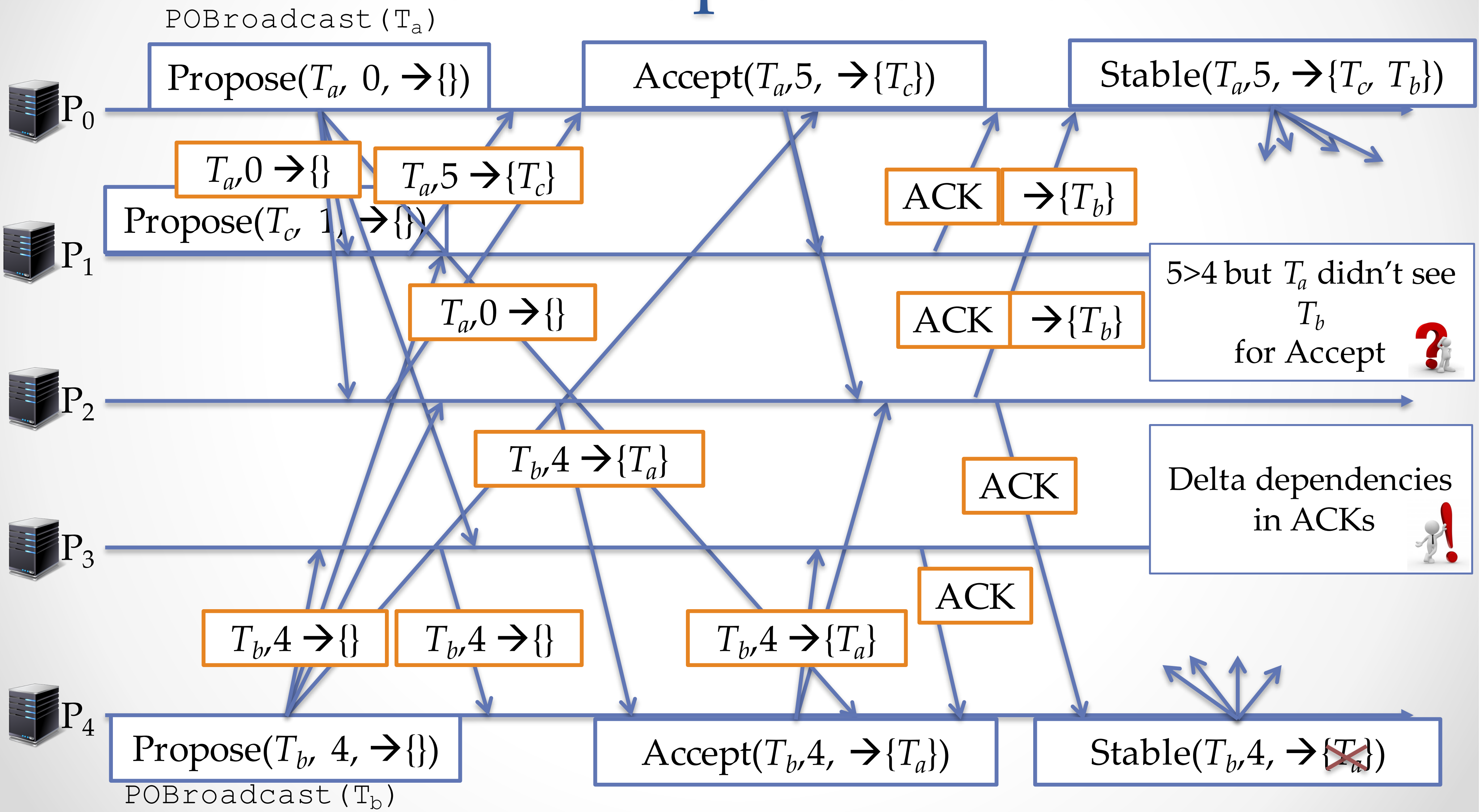
Local Dependency

For any node that executes *PODeliver*(T,$deps_T$) before *PODeliver*(T',$deps_{T'}$), and T and T' are conflicting, then $T \in deps_{T'}$ and $T' \notin deps_T$

POB in Action



Easier Said than Done:
Delta Dependencies



VirginiaTech Invent the Future

Raising the Bar:
2-Communication Delays Delivery

- Base scheme:
  - Wait for $FQ = f + 1$ replies in the Proposal phase
  - Wait for $CQ = f + 1$ replies in the Accept phase
- Fast Transaction Decision:
  - <u>NO Accept phase if all *FQ* replies are the same</u>
  - Wait for $FQ = f + \left\lfloor \frac{f+1}{2} \right\rfloor$ replies in the Proposal phase
  - Wait for $CQ = f + 1$ replies in the Accept phase

$N = 2f+1$ = # of sites
f = max # of faulty sites
FQ and CQ are sizes of quorums

Like Epaxos's quorums.

One node less than *FQ* in Generalized Paxos.

IDEA

- A leader crashes after a fast decision for T, then T's new leader has to take the same decision
  - Every majority in a classic quorum confirms the fast decision

$$N - FQ - 1 < \left\lfloor \frac{CQ}{2} \right\rfloor + 1$$

  - Two new conflicting leaders for T and T' respectively cannot both believe there were two discordant fast decisions for T and T' in the past

$$\left\lfloor \frac{N-f}{2} \right\rfloor + f - 1 < FQ$$

P-CC Layer

Multiversion timestamp-based concurrency control

Execution module

1. Transactions are executed on the snapshot committed before they began.
2. Write operations are buffered.
3. Transactions are submitted to POB layer to request the commit after the execution.

Commit module

1. Upon PODeliver(T,{T1,…,Tk}), wait for {T1,…,Tk} to be either committed or aborted.
2. Validate T, i.e., check that T's snapshot didn't change since T began…
3. …and if so, apply T's updates.

NO synchronization point among non-conflicting transactions!

Correctness Criteria

ALVIN guarantees *Serializability*:
Committed transactions appear as they had been executed sequentially.

Ordered Commits

The commits of all conflicting transactions delivered by POB are totally ordered.

Valid Transactions

Transactions delivered by POB only commit if they didn't observe an outdated snapshot.

Consistent Reads

Transactions always observe a consistent state.

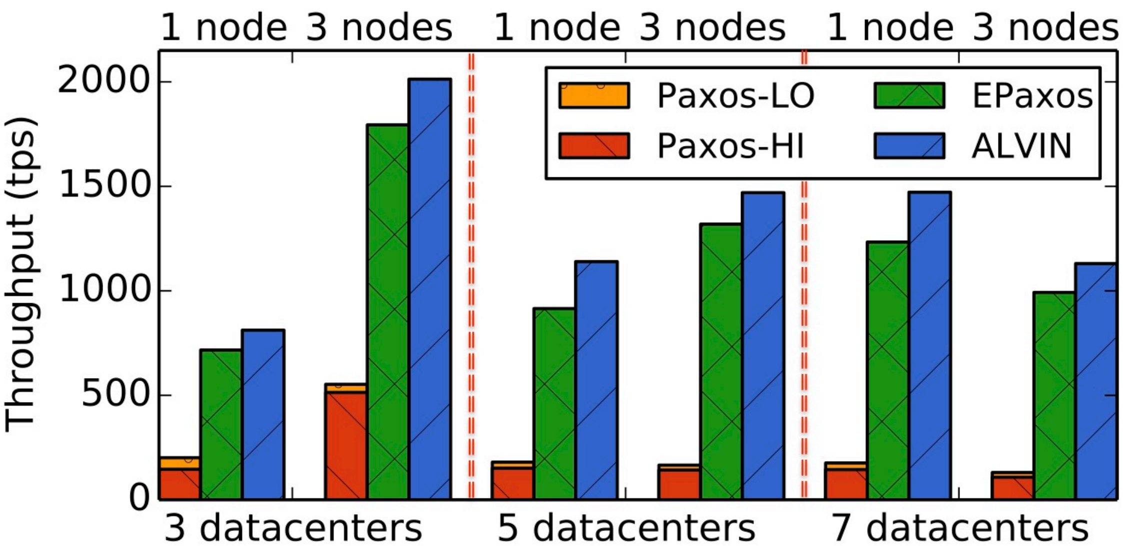
<u>If read-only transactions are not submitted to POB</u>
ALVIN guarantees *Extended Update Serializability (EUS)* *[ICDCS12]*:

- Committed update transactions are serializable
- All transactions always observe a consistent state

Experimental Evaluation

- Implementation: stand-alone framework implemented in the Go Programming language.
- Competitors: certification-based replication protocols by using MultiPaxos [Lamport98] and EPaxos as broadcast layer.
  - MultiPaxos provides total order. Designated leader with point-to-point latency to the other nodes either higher (Paxos-HI) or lower (Paxos-LO) than the average.
  - EPaxos provides total order only among conflicting transactions.
- Benchmarks: TPC-C & Bank.
- Infrastructure: Amazon EC2 with nodes in up to 7 geographically distributed sites.

TPC-C Benchmark

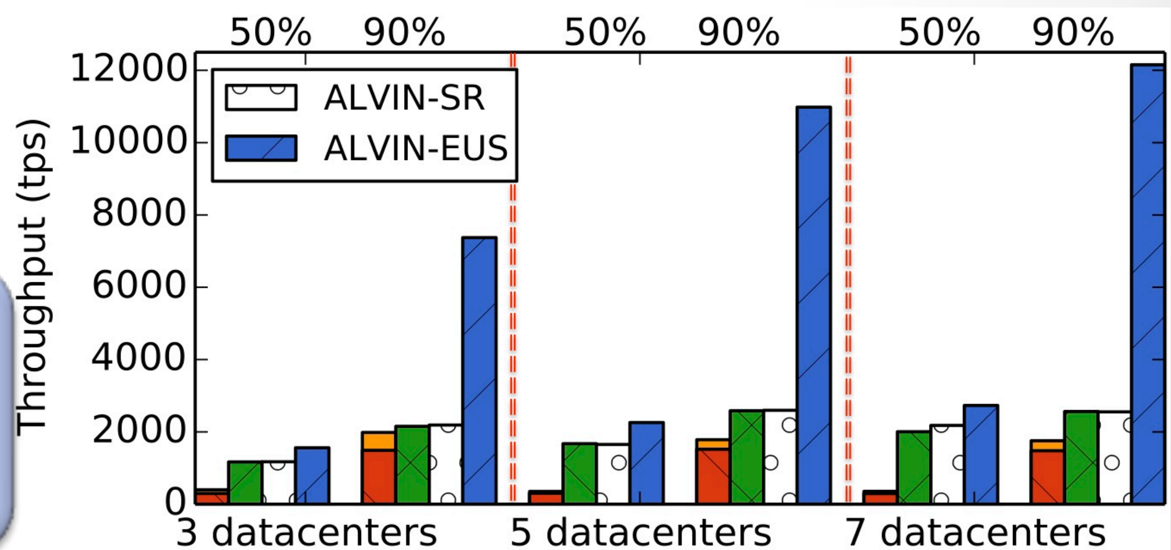


Write-intensive workload
(< 3% read-only transactions)

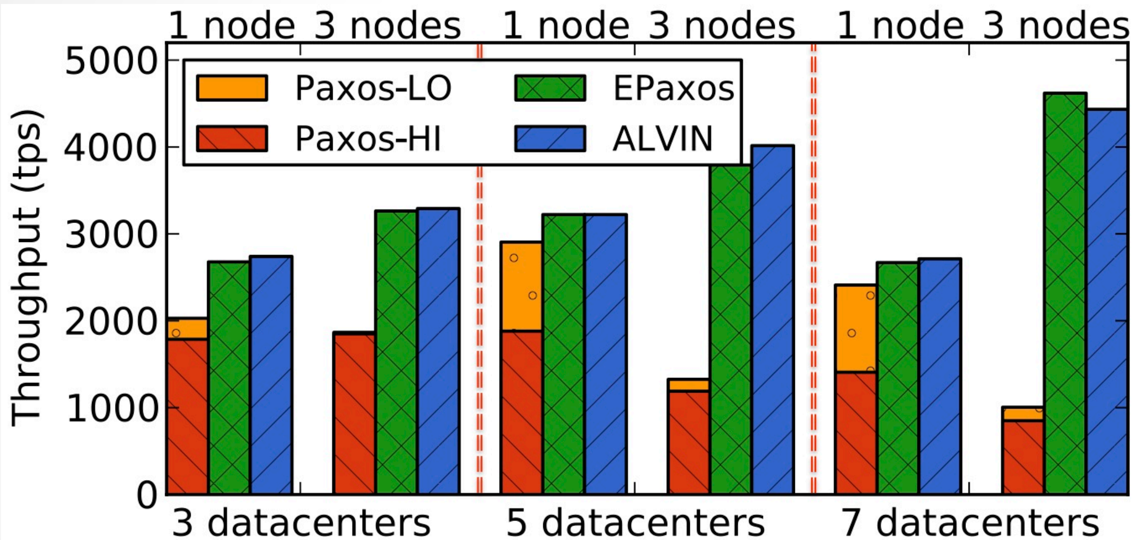
ALVIN gains up to 26% in throughput against EPaxos

Read-intensive workload
(50% and 90% read-only transactions)

EUS provides a speed-up of up to 4.8x in throughput when compared to Serializability



Bank Benchmark



- Write-intensive workload with very small transaction.
- Transactional work negligible if compared to the coordination steps .

- Alvin and Epaxos are comparable.
- Single leader is the bottleneck for MultiPaxos.

Thanks for the attention

peluso@vt.edu

References

- [EuroSys13] Almeida, S., Leitão, J., Rodrigues, L.: ChainReaction: A Causal+ Consistent Data-store Based on Chain Replication. In: 8th ACM EuroSys, pp. 85–98. ACM (2013)
- [ICDCS12] Peluso, S., Ruivo, P., Romano, P., Quaglia, F., Rodrigues, L.: When Scalability Meets Consistency: Genuine Multiversion Update-Serializable Partial Data Replication. In: 32nd ICDCS, pp. 455–465. IEEE Computer Society (2012)
- [OSDI08] Mao, Y., Junqueira, F. P., Marzullo, K.: Mencius: Building Efficient Replicated State Machines for WANs. In: 8th USENIX OSDI, pp. 369–384. USENIX (2008)
- [SOSP11] Sovran, Y., Power, R., Aguilera, M. K., Li, J.: Transactional Storage for Geo-replicated Systems. In: 23rd ACM SOSP, pp. 385–400. ACM (2011)
- [SOSP13a] Zhang, Y., Power, R., Zhou, S., Sovran, Y., Aguilera, M. K., Li, J.: Transaction Chains: Achieving Serializability with Low Latency in Geo-distributed Storage Systems. In: 24th ACM SOSP, pp. 276–291. ACM (2013)
- [SOSP13b] Moraru, I., Andersen, D.G., Kaminsky,M.: There is More Consensus in Egalitarian Parliaments. In: 24th ACM SOSP, pp. 358–372. ACM (2013)
- [SRDS13] Ardekani, M. S., Sutra, P., Shapiro, M., Preguica, N.: Non-Monotonic Snapshot Isolation: scalable and strong consistency for geo-replicated transactional systems. In: 32nd SRDS. IEEE (2013)
- [TOCS13] Corbett J. C. et al.: Spanner: Google's Globally Distributed Database. ACM Trans. Comput. Syst. 31(3), 8:1–8:22 (2013)
- [TOCS98] Lamport, L.: The Part-time Parliament. ACM Trans. Comput. Syst. 16(2), 133–169 (1998)

Key Design Principles

Consistency

- Correct state transitions on all replicas: conflicting transactions committed according to a common order at all replicas.

Require Consensus

Latency

- No partial replication data model
- 2 per-transaction communication delays in case of no conflicts

- Coordination either at the beginning or during the commit.
- No less than 2 communication delays due to the lower bound on consensus.

Parallelism

- No coordination among non-conflicting transactions.
- No designated sites with special rules.

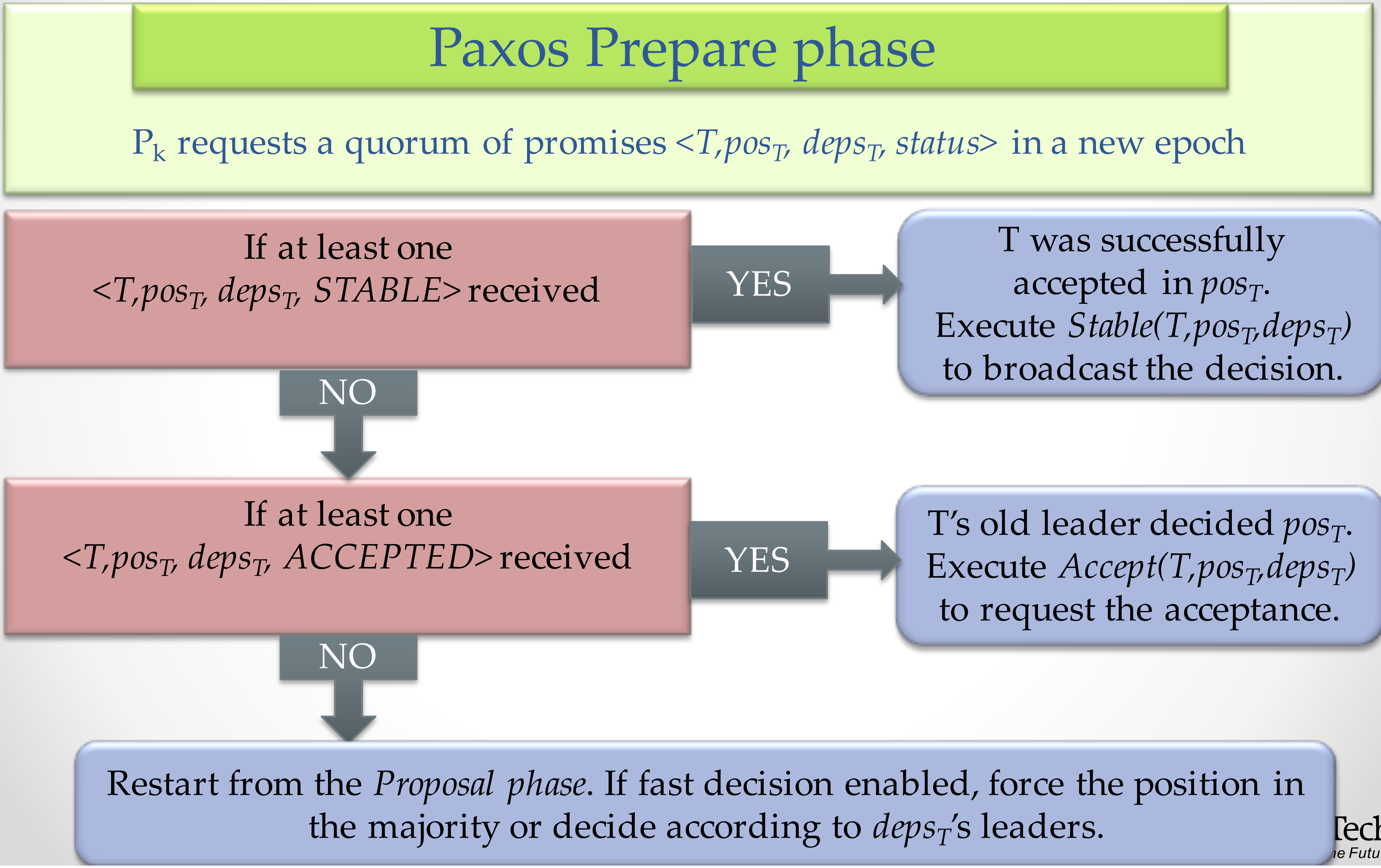
No leader-based consensus

Synchrony

- No assumptions on inter-site delays and replicas' clocks speed

Recovery from Failures

If P_k suspects *T*'s current leader and *T* is not *STABLE*



Geo-Replication: the Whole Picture

- Picture on tx execution. Single node-> cluster -> geo-replication (in a previous slide)
- Challenges in geo-replication: minimize the protocol communication delays during execution and commit. (Far vedere che un nodo richiede il commit e risponde al client dopo ricevuto le risposte. Poi scrivere qualcosa del tipo: posso fare di meglio? No, se vado in crash voglio garantire consistenza…e poi si passa al grafico seguente)
- Consistency: we still require strong consistency to avoid state divergences and to be fully general and transparent to the applications -> the history of committed update transactions has to be serializable (far vedere che le repliche non devono divergere)
- Do not define special roles for sites: delays among sites can vary and are not uniform
- (Sarebbe carino associare un'immagine ad ogni frase.)

Desirable Guarantees

- Full replication still better for geo-replication: transactions are executed locally before the commit -> no remote read/write operations…but the commit has to involve all sites.
- Serializability of update transactions -> conflicting transactions committed according to a common order at all sites
  - Non-conflicting transactions should proceed in parallel
- Consensus on the commit-> two communication delays in case of no conflicts is the best!
- No special roles -> no designed site for helping to reach an agreement on consensus

Desirable Guarantees

- Full replication still better for geo-replication: transactions are executed locally before the commit -> no remote read/write operations…but the commit has to involve all sites.
- Serializability of update transactions -> conflicting transactions committed according to a common order at all sites
  - Non-conflicting transactions should proceed in parallel
- Consensus on the commit-> two communication delays in case of no conflicts is the best!
- No special roles -> no designed site for helping to reach an agreement on consensus