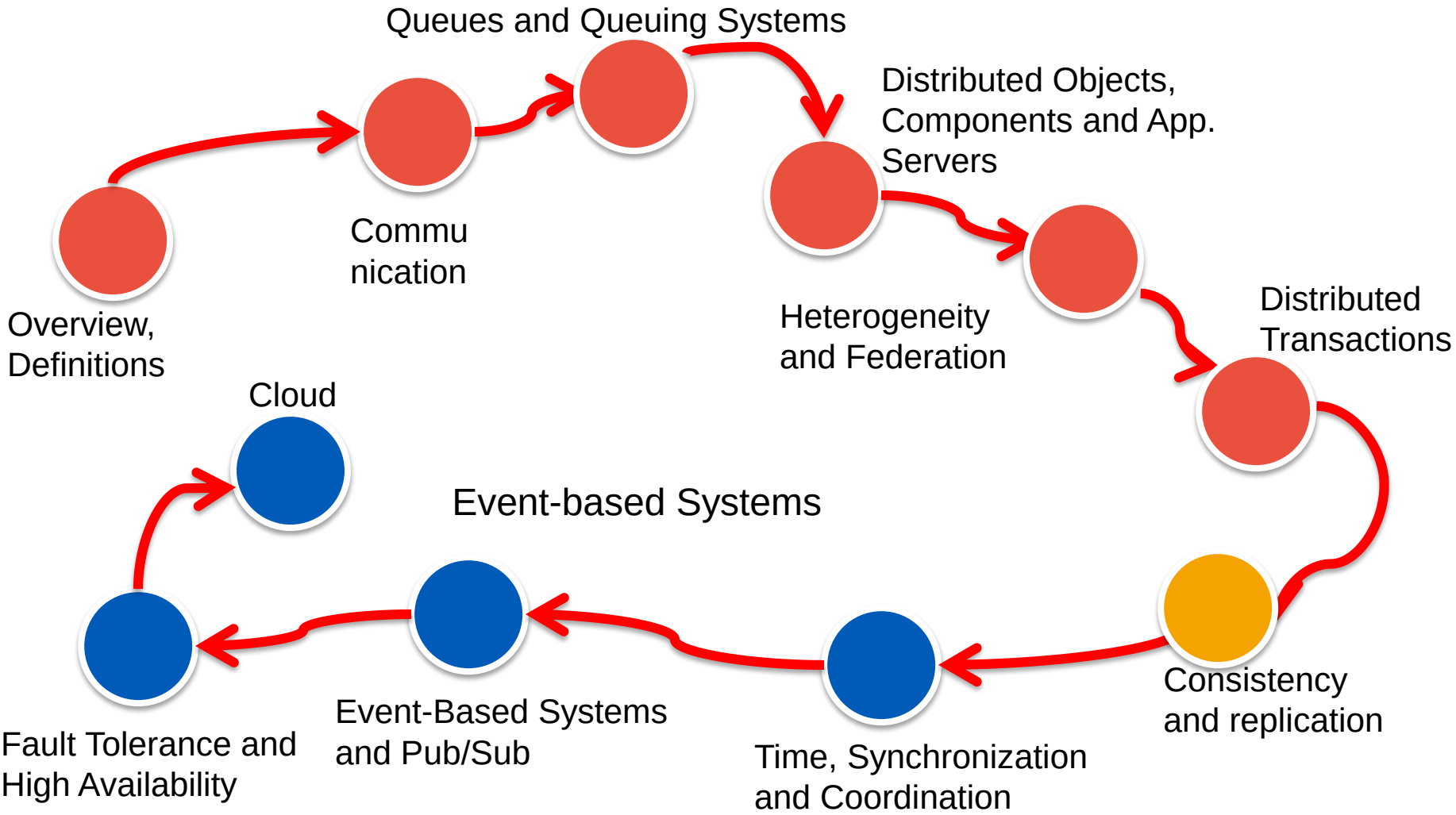
Middleware:
7. Replication and Consistency

TECHNISCHE UNIVERSITÄT DARMSTADT

A. Buchmann
Wintersemester 2014/2015

Topics


Queues and Queuing Systems
Distributed Objects, Components and App. Servers
Commu nication
Overview, Definitions
Heterogeneity and Federation
Distributed Transactions
Cloud
Event-based Systems
Consistency and replication
Fault Tolerance and High Availability
Event-Based Systems and Pub/Sub
Time, Synchronization and Coordination

Topics

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Replication**
- **Consistency**
- **Update management Strategies in replicated systems**
- **Replication protocols**
- **Failure Handing**
- **Quorum Based approaches**

Reading for THIS Lecture

TECHNISCHE UNIVERSITÄT DARMSTADT

- The slides for the lecture are based on material from:
  - M. Tamer Özsu and Patrick Valduriez **Principles of Distributed Database Systems** (3nd Ed.). Prentice-Hall. 2011
    - Chapter 13
  - Andrew S. Tanenbaum and Maarten Van Steen. 2001. **Distributed Systems: Principles and Paradigms.** Prentice Hall.
    - Chapter 6
  - B. Charron-Bost, F. Pedone, A. Schiper (Eds.).
    - **Replication: Theory and Practice**. LNCS Vol. 5959. Springer Verlag, 2010
  - George Coulouris, Jean Dollimore, and Tim Kindberg. 2005. **Distributed Systems: Concepts and Design.** Addison-Wesley Longman.
    - Chapter 15

To replicate or not?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Remember: ‘Lessons from Giant-Scale Services’!

Reading: Eric A. Brewer. 2001. Lessons from Giant-Scale Services. IEEE Internet Computing 5, 4 (July 2001).

- **Why replicate?**
  - ***System availability / Reliability***
    - reduce single points of failure
  - ***Performance***
    - locate data close to client, perform operations locally
    - Lower latency
  - ***Scalability***
    - Geographic distribution
    - Number of parallel requests at low response times
  - ***Application requirements***

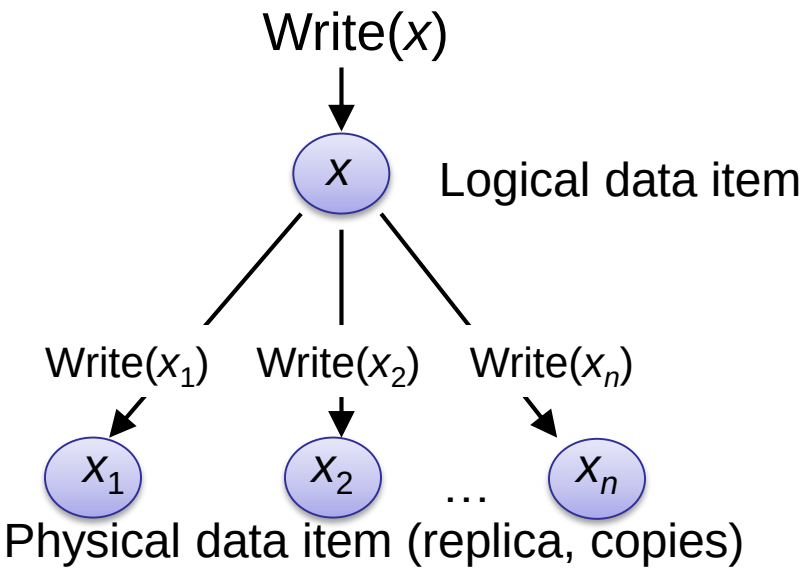
- **Why not replicate?**
  - Replication transparency typically results in lower performance
  - Consistency issues
    - Updates are costly
    - May impede availability

- How are updates distributed and propagated to replicas?

Execution Model

TECHNISCHE UNIVERSITÄT DARMSTADT

- Assume:
  - X is a replicated logical object.
  - X1, X2 … Xn are physical copies of the logical object X in the system
- Operations are specified on logical objects
  - but translated to operate on physical objects.
- One-copy equivalence
  - The effect of transactions performed by clients on replicated objects should be the same as if they had been performed on a single set of objects.



DVS

Replication Issues

TECHNISCHE UNIVERSITÄT DARMSTADT

- Consistency models that are suitable for different types of applications
  - Mutual consistency: Strong/Weak consistency
  - Transactional consistency
- Where are updates allowed?
  - Centralized
  - Distributed
- Update propagation techniques – how do we propagate updates on one copy to the other copies?
  - Eager
  - Lazy

Consistency

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

Mutual Consistency

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Problem**: are values of physical copies of a logical data item synchronized (converging over time)?

- Strong consistency
  - All copies are updated within the context of the update transaction
  - When the update transaction completes, all copies have the same value
  - Typically achieved through 2PC

- Weak consistency
  - Eventual consistency
    - the copies are not identical when update transaction completes
    - eventually converge to the same value
  - Many versions possible:
    - Time-bounds
    - Value-bounds
    - Drifts

Transactional Consistency

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem: How can we guarantee that the global execution history over replicated data is serializable?
- One-copy serializability (1SR)
  - The effect of transactions performed by clients on replicated objects should be the same as if they had been performed *one at-a-time* on a single set of objects.
- Weaker forms are possible
  - Snapshot isolation
  - RC-serializability (RC=Relaxed Consistency) = ‘READ COMITTED’

Reading: B. Kemme, G. Alonso. Don't Be Lazy, Be Consistent: Postgres-R, A New Way to Implement Database Replication. VLDB 2000.

Transactional and Mutual Consistency

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Problem**: Transactional and Mutual Consistency are orthogonal

Site A	Site B	Site C
x	x, y	x, y, z
T_1:$x \leftarrow 20$	T_2:Read(x)	T_3:Read(x)
Write(x)	$x \leftarrow x+y$	Read(y)
Commit	Write(y)	$z \leftarrow (x*y)/100$
	Commit	Write(z)
		Commit

Assumption: Histories at Sites A,B,C

$H_A=\{W_1(x_A), C_1\}$

$H_B=\{W_1(x_B), C_1, R_2(x_B), W_2(y_B), C_2\}$

$H_C=\{W_2(y_C), C_2, R_3(x_C), R_3(y_C), W_3(z_C), C_3, W_1(x_C), C_1\}$

Global history non-serializable: H_B: $T_1 \rightarrow T_2$, H_C: $T_2 \rightarrow T_3 \rightarrow T_1$

Mutually consistent:

Assume $x_A=x_B=x_C=10$, $y_B=y_C=15$, $z_C=7$ to begin;
in the end $x_A=x_B=x_C=20$, $y_B=y_C=35$, $z_C=3.5$

Alternative Approaches to Middleware Transactions

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Do you really/always need ACID Tx?

TECHNISCHE UNIVERSITÄT DARMSTADT

- **In large distributed systems full transactional properties with strong consistency are not feasible (nor always needed)**
  - Depends on: application (requirements, data) and system.

Atomicity, Consistency, Isolation, Durability

ACID

- Strong consistency for transactions highest priority
- Availability less important
- Pessimistic
- Rigorous analysis
- Complex mechanisms

Basic Availability, Soft State, Eventual Consistency

BASE

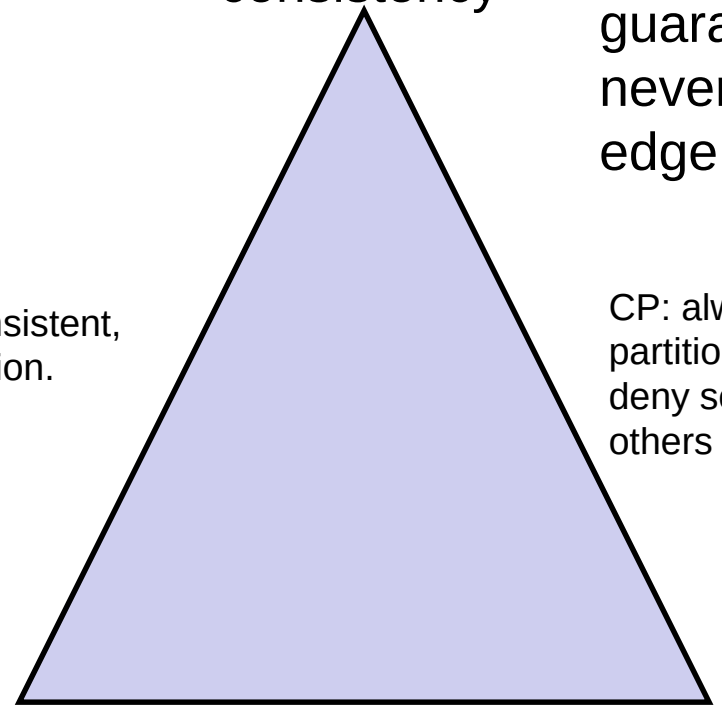
- Availability and scaling highest priorities
- Weak consistency
- Optimistic
- Best effort
- Simple and fast

http://www.johndcook.com/blog/2009/07/06/brewer-cap-theorem-base/

http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

CAP Theorem

TECHNISCHE UNIVERSITÄT DARMSTADT



Statement: every system can simultaneously guarantee at most two, but never all three (is on one edge of the triangle).

CA: available, and consistent, unless there is a partition.

CP: always consistent, even in a partition, but a reachable replica may deny service without agreement of the others (e.g., quorum).

AP: a reachable replica provides service even if there is a partition, but may be inconsistent if there is a failure.

Brewer, E. A. 2000. Towards robust distributed systems. In PODC'00 (Portland, OR, USA, July 16 - 19, 2000).

DVS

Why is differentiation needed?

TECHNISCHE UNIVERSITÄT DARMSTADT

- What goals might you want from a shared-data system?
  - C(onsistency), A(vailability), P(artitioning) → the CAP Theorem [Brewer2000]
  - You can only have two out of these three properties!
    - The choice of which feature to discard determines the nature of your system
    - Databases: sacrifice availability
    - Distributed systems: sacrifice consistency

- **Strong Consistency**: all clients see the same view, even in the presence of updates
- **High Availability**: all clients can find some replica of the data, even in the presence of failures
- **Partition-tolerance**: the system properties hold even when the system is partitioned

Consistency and Availability

TECHNISCHE UNIVERSITÄT DARMSTADT

- Providing transactional semantics requires all nodes to be in contact with each other
- Examples:
  - Single-site and clustered databases
  - Other cluster-based designs
- Typical Features:
  - Two-phase commit
  - Cache invalidation protocols
  - Classic DS style
- What happens if the network connectivity is lost (network partitioning)?

Consistency and Partition-Tolerance

TECHNISCHE UNIVERSITÄT DARMSTADT

- If one is willing to tolerate system-wide blocking, then strong consistency can be provided even when there are temporary partitions

- Examples:
  - Distributed databases
  - Distributed locking
  - Quorum (majority) protocols

- Typical Features:
  - Pessimistic locking
  - Minority partitions unavailable
  - Also common DS style
    - Voting vs primary replicas

- How about response times?

Partition-Tolerance and Availability

TECHNISCHE UNIVERSITÄT DARMSTADT

- Once consistency is sacrificed → easy → but is it really desirable….?
  - It is a simple solution
  - the implications of sacrificing „P“ are difficult. Scenario!
  - sacrificing „A“ is unacceptable for High Availability Systems (Web/Cloud)
  - possible to push the problem to app developer

- „C“ not needed in many applications
  - Airline reservation only transacts reads
  - Many DB (MySQL) use by default lower isolation levels

- When data is noisy and inconsistent anyway
  - making it, say, 1% worse does not matter

Partition-Tolerance and Availability

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Once consistency is sacrificed → easy → but is it really desirable….*
  - Eventual Consistency:
    - In Distributed and replicated DB (why?)
    - In a Query-phase after an update phase in the workload → eventually all replicas will converge to the identical values
  - Replication

- Examples:
  - DNS
  - Web caches
  - Cloud

- Typical Features:
  - Optimistic updating with conflict resolution
  - Internet design style

Techniques

TECHNISCHE UNIVERSITÄT DARMSTADT

- Guaranteed A,P: Expiration-based caching, Eventual consistency
- Guaranteed P,C: Quorum/majority algorithms, Paxos
- Guaranteed A,C: Two-phase commit

Update Management Strategies – Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

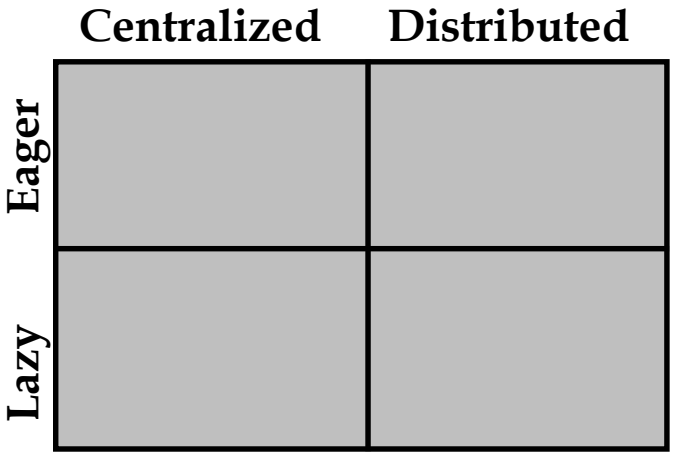
Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Update Management Strategies

TECHNISCHE UNIVERSITÄT DARMSTADT

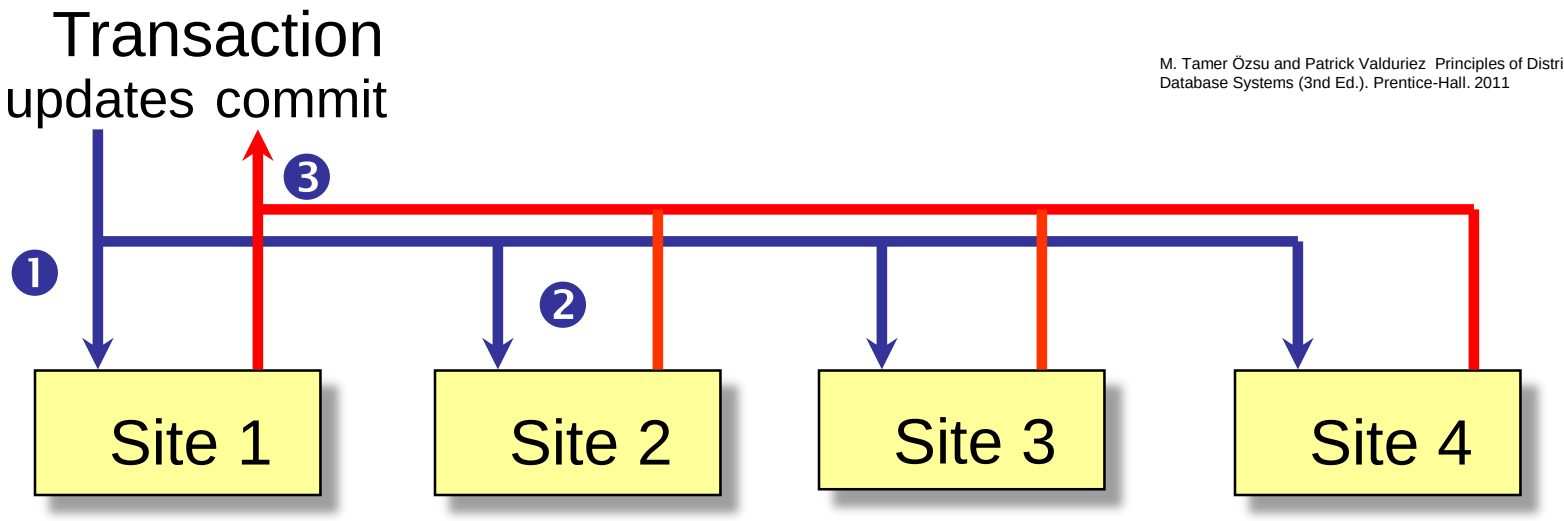
- Replication Protocols are classified according to two criteria
- Depending on when the updates are propagated
  - Eager
  - Lazy
- Depending on where the updates can take place
  - Centralized
  - Distributed



DVS

Eager Replication

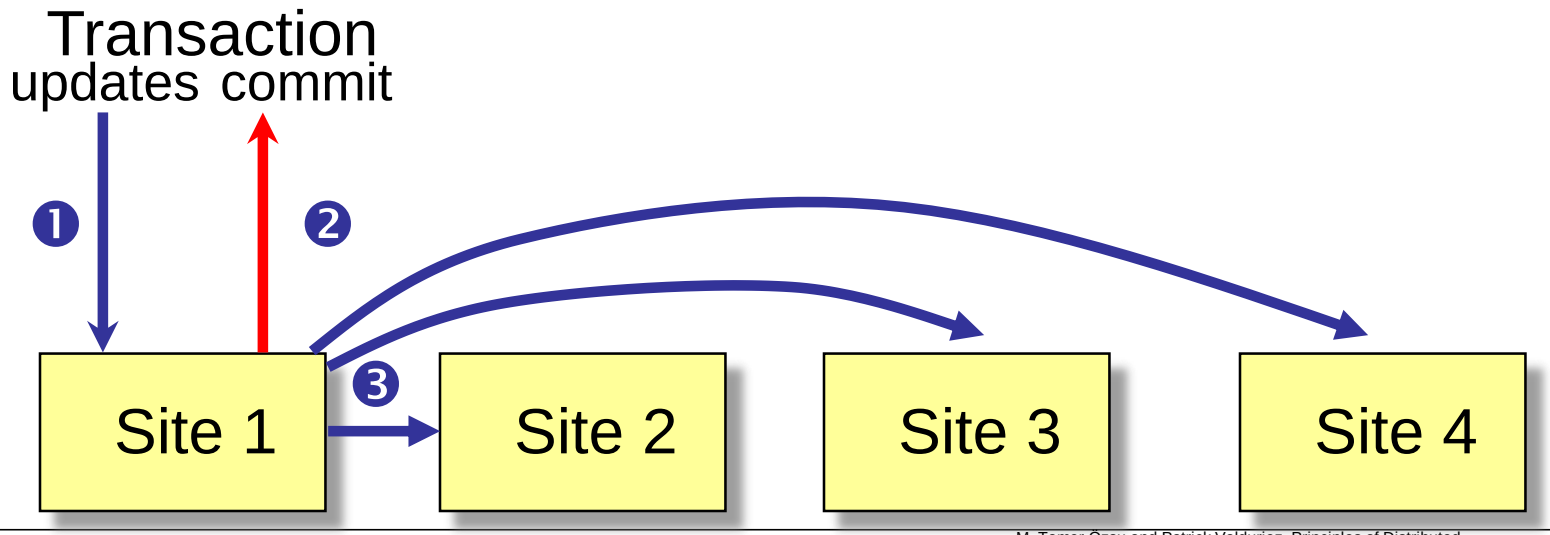
- Updates are propagated within the scope of the transaction making the changes. → 2PC as commit protocol
- The ACID properties apply to all copy updates.
  - Synchronous → apply every update to all replicas
  - Deferred → updates applied to one replica → batched →applied deferred as bulk job to all replicas before commit (Prepare-to-commit)
- ROWA protocol: Read-one/Write-all



M. Tamer Özsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Lazy Replication

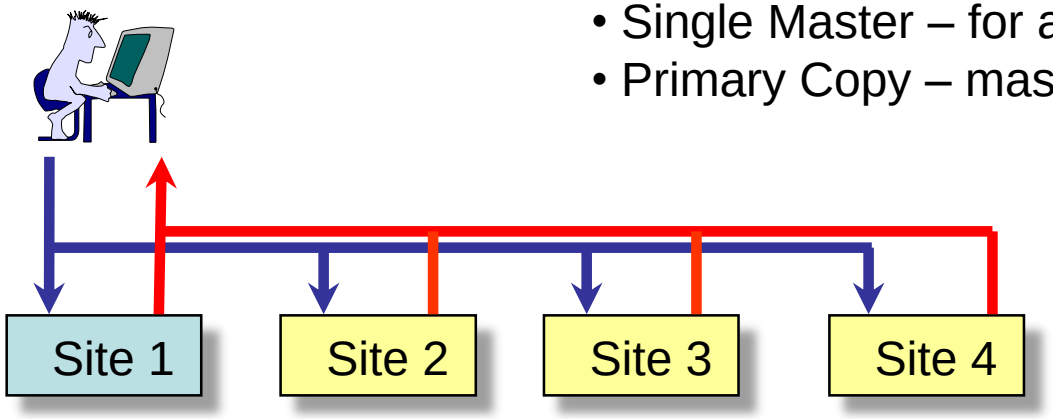
- Lazy update propagation first executes the updating transaction on one copy. After the transaction commits, the changes are propagated asynchronously to all other copies (refresh transactions)
- While the propagation takes place, the copies are mutually inconsistent.
- The time the copies are mutually inconsistent is an adjustable parameter which is application dependent.
- Advantage: Low Response times

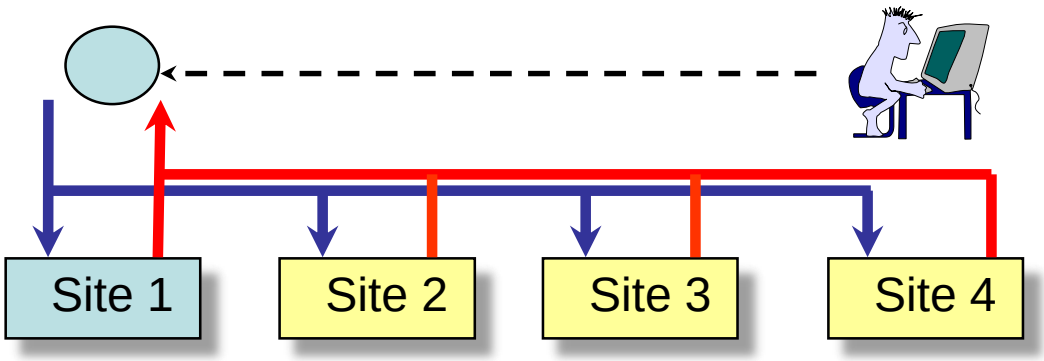


M. Tamer Özsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Centralized

- There is only one copy which can be updated (the master), all others (slave copies) are updated reflecting the changes to the master.
  - Master/Slave Replication
    - Single Master – for all replicated data
    - Primary Copy – master copy for every data item

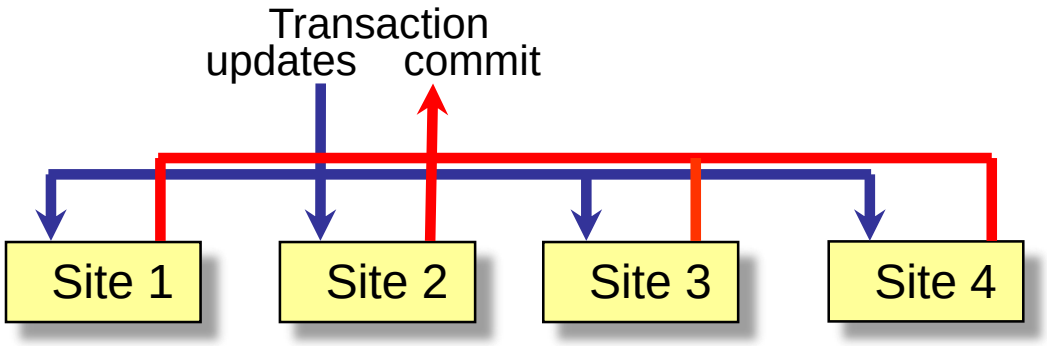




M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Distributed

- Changes can be initiated at any of the copies. That is, any of the sites which owns a copy can update the value of the data item.
- Collaborative, Web 2.0, social Applications, Cloud
- If eager propagation – couple with distr. CC
- if lazy propagation – no 1SR guarantees
  - employ reconciliation methods / Redo Tx at Sites



M. Tamer Özsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Forms of Replication

TECHNISCHE UNIVERSITÄT DARMSTADT

Eager

+ No inconsistencies (identical copies)
+ Reading the local copy yields the most up to date value
+ Changes are atomic
– A transaction has to update all sites
  – Longer execution time
  – Lower availability
  – Potentially blocking

Lazy

+ A transaction is always local (good response time)
– Data inconsistencies
– A local read does not always return the most up-to-date value
– Changes to all copies are not guaranteed
– Replication is not transparent

Centralized

+ No inter-site synchronization is necessary (it takes place at the master)
+ There is always one site which has all the updates
– The load at the master can be high
– Reading the local copy may not yield the most up-to-date value

Distributed

+ Any site can run a transaction
+ Load is evenly distributed
– Copies need to be synchronized

M. Tamer Özsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

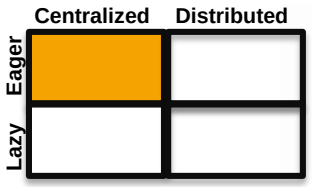
Replication Protocols

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Eager Centralized Protocols

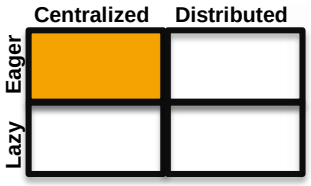


TECHNISCHE UNIVERSITÄT DARMSTADT

- Design parameters:
  - Distribution of master
    - Single master: one master for all data items
    - Primary copy: different masters for different data items. Known to application
  - Level of transparency
    - Limited: applications have full (location) information about master
      - Update transactions are submitted directly to the master
      - Reads can occur on slaves
    - Full: applications and users can submit anywhere and the operations will be forwarded to the master
      - Operation-based forwarding
- Four alternative implementation architectures, only three are meaningful:
  - Single master, limited transparency
  - Single master, full transparency
  - Primary copy, full transparency

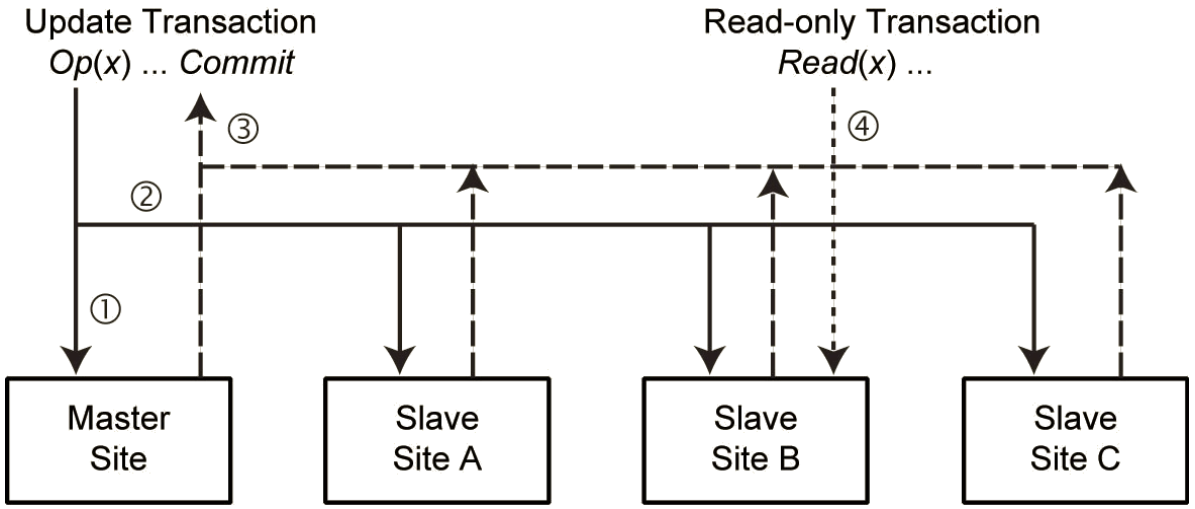
M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Eager Single Master/Limited Transp.



TECHNISCHE UNIVERSITÄT DARMSTADT

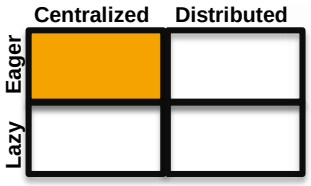
- Applications submit update transactions directly to known master
- Master:
  - Upon read: read locally and return to user
  - Upon write: write locally, multicast write to other replicas (in FIFO timestamp order)
  - Upon commit request: run 2PC coordinator to ensure that all have really installed the changes
  - Upon abort: abort and inform other sites about abort
- Slaves install writes that arrive from the master



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

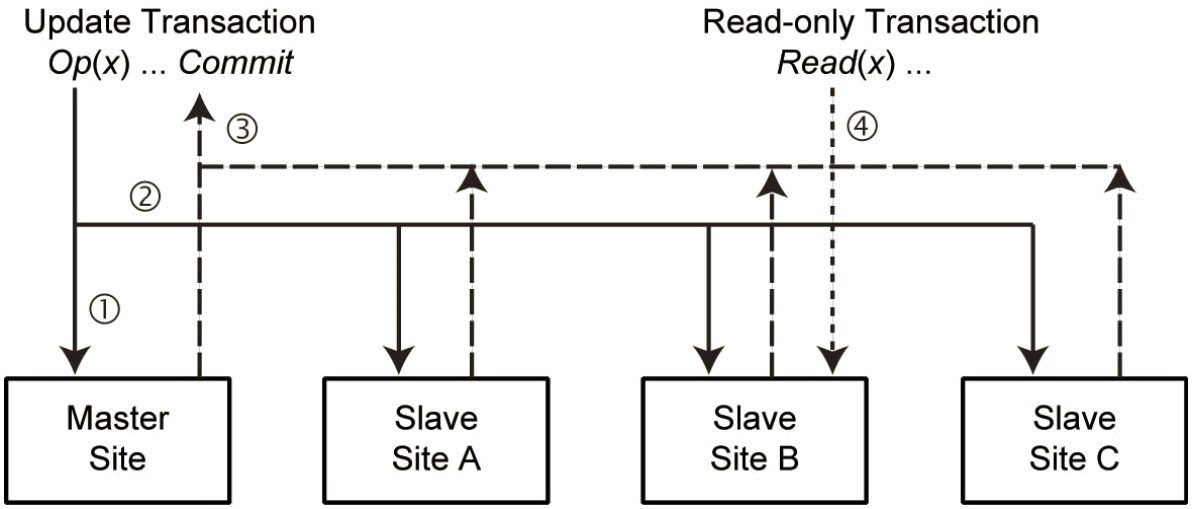
DVS

Eager Single Master/Limited Transp.



TECHNISCHE UNIVERSITÄT DARMSTADT

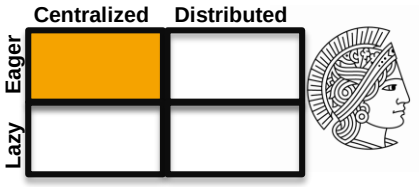
- Applications submit read transactions directly to an appropriate slave
- Slave
  - Upon read: read locally
  - Upon write from master copy: execute conflicting writes in the proper order (FIFO or timestamp)
  - Upon write from client: refuse (abort transaction; there is error)
  - Upon commit request from read-only: commit locally
  - Participant of 2PC for update transaction running on primary



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

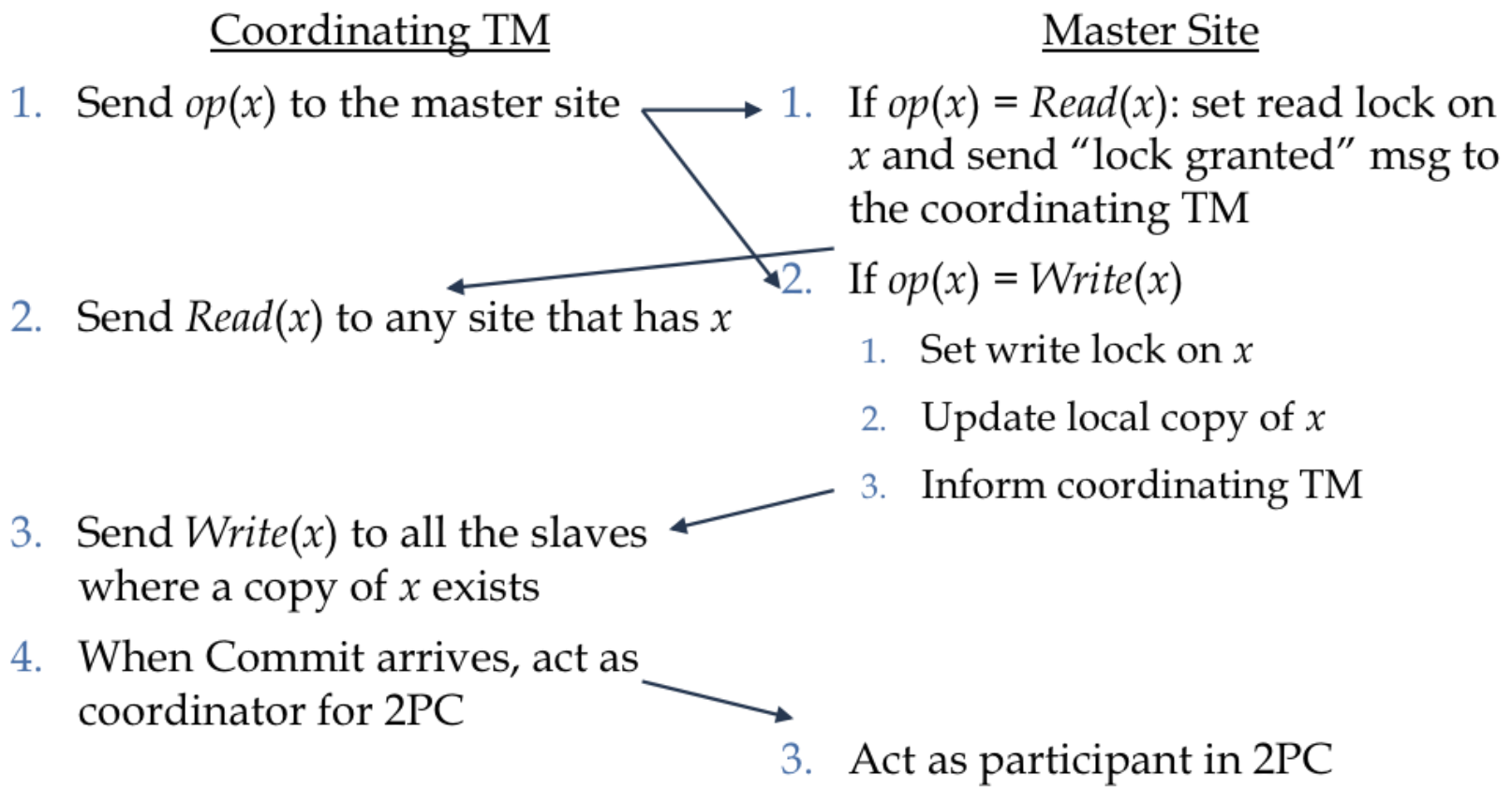
DVS

Eager Single Master/Full Transp.



TECHNISCHE UNIVERSITÄT DARMSTADT

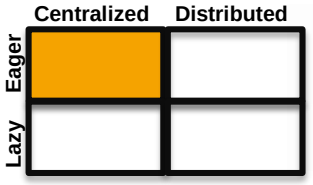
- Applications submit all transactions to the Transaction Manager at their own sites (Coordinating TM)



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

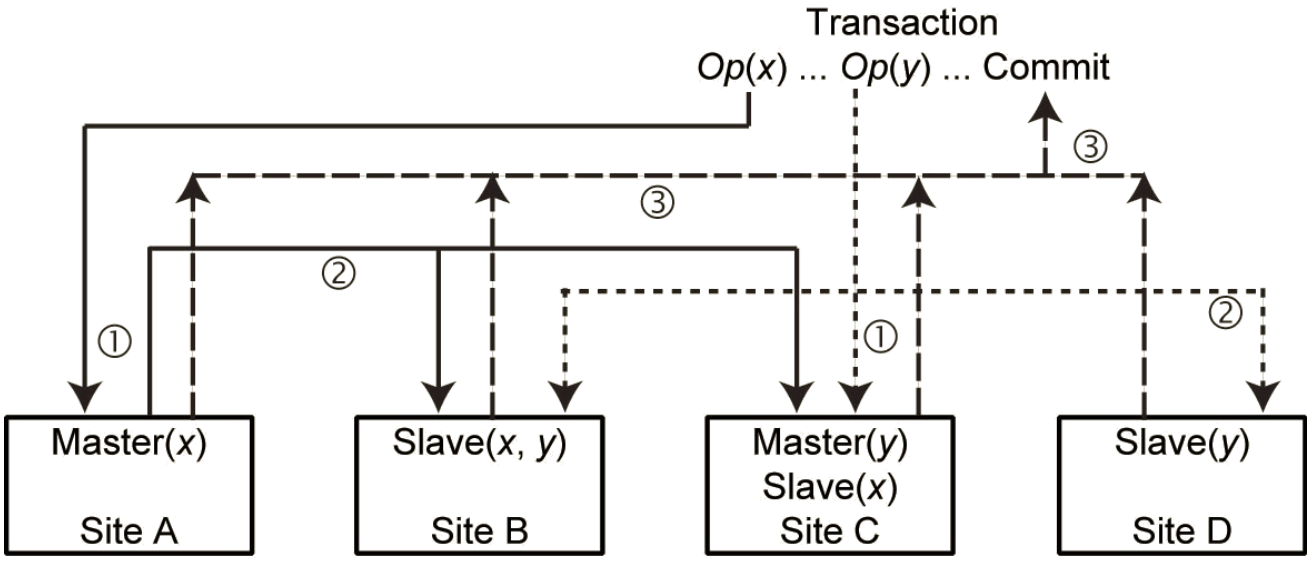
DVS

Eager Primary Copy/Full Transp.



TECHNISCHE UNIVERSITÄT DARMSTADT

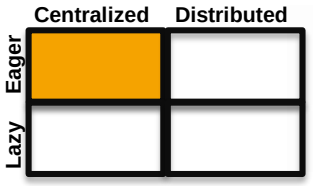
- Applications submit transactions directly to their local TMs
- Local TM:
  - Forward each operation to the primary copy of the data item
  - Upon granting of locks, submit Read to any slave, Write to all slaves
  - Coordinate 2PC



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

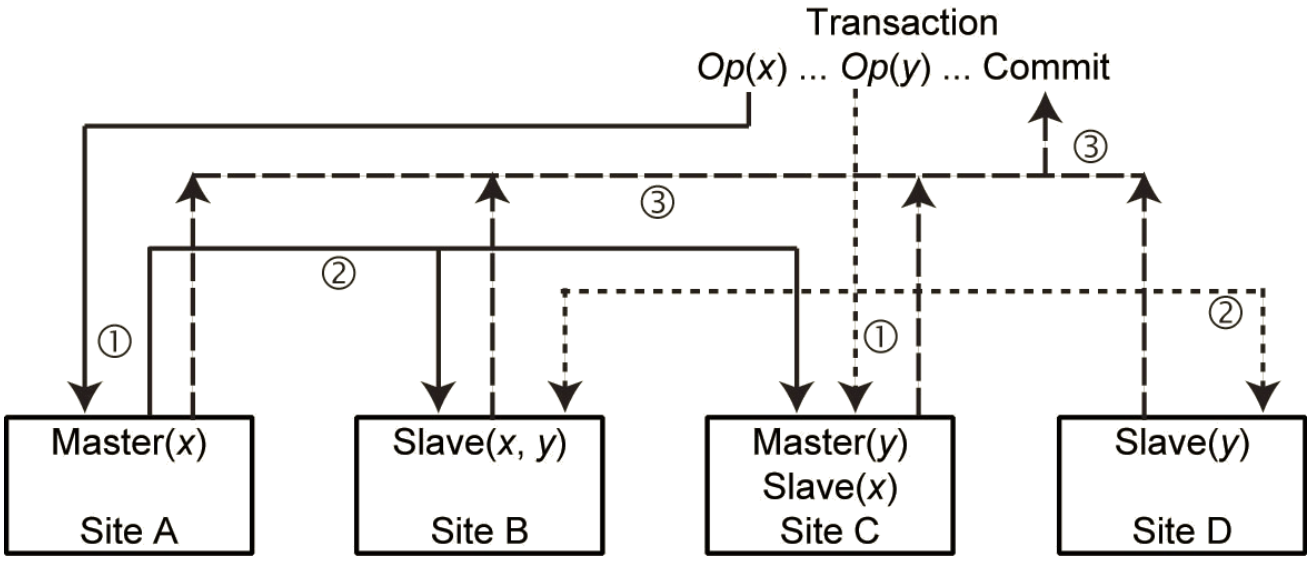
DVS

Eager Primary Copy/Full Transp.



TECHNISCHE UNIVERSITÄT DARMSTADT

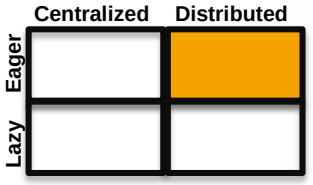
- Primary copy site
  - Read(x): lock x and reply to TM
  - Write(x): lock x, perform update, inform TM
  - Participate in 2PC
- Slaves: as before



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

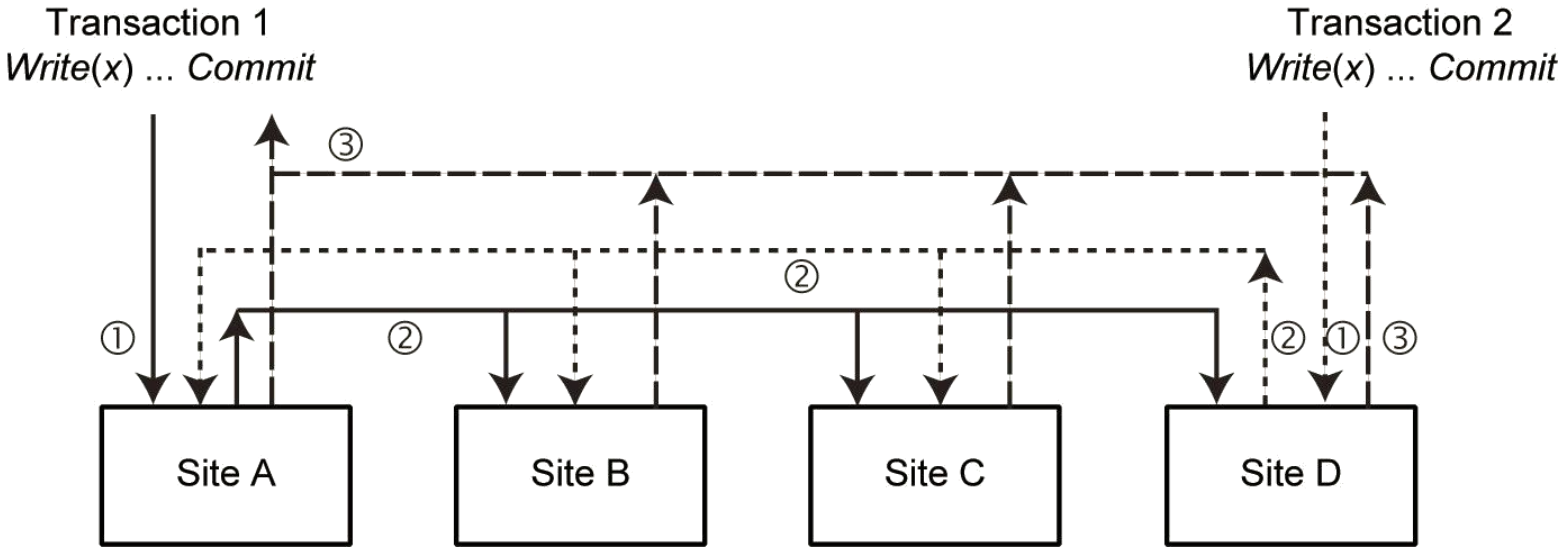
DVS

Eager Distributed Protocol



TECHNISCHE UNIVERSITÄT DARMSTADT

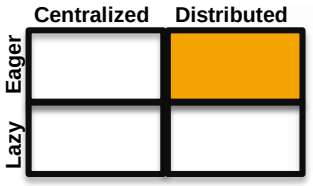
- Updates originate at any copy
  - Each site uses 2 phase locking.
  - Read operations are performed locally.
  - Write operations are performed at all sites (using a distributed locking protocol).
  - Coordinate 2PC
- Slaves: As before



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

DVS

Eager Distributed Protocol



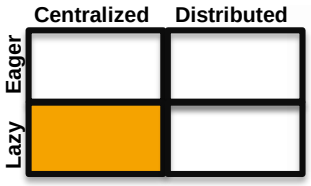
TECHNISCHE UNIVERSITÄT DARMSTADT

- Critical issue:
  - Concurrent Writes initiated at different master sites are executed in the same order at each slave site
  - Local histories are serializable
- Advantage: Simple and easy to implement
- Disadvantage:
  - Very high communication overhead
    - n replicas; m update operations in each transaction: n*m messages (assume no multicasting)
    - For throughput of k [Tx/sec]: k*n*m messages
- Alternative
  - Use group communication + deferred update to slaves to reduce messages

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

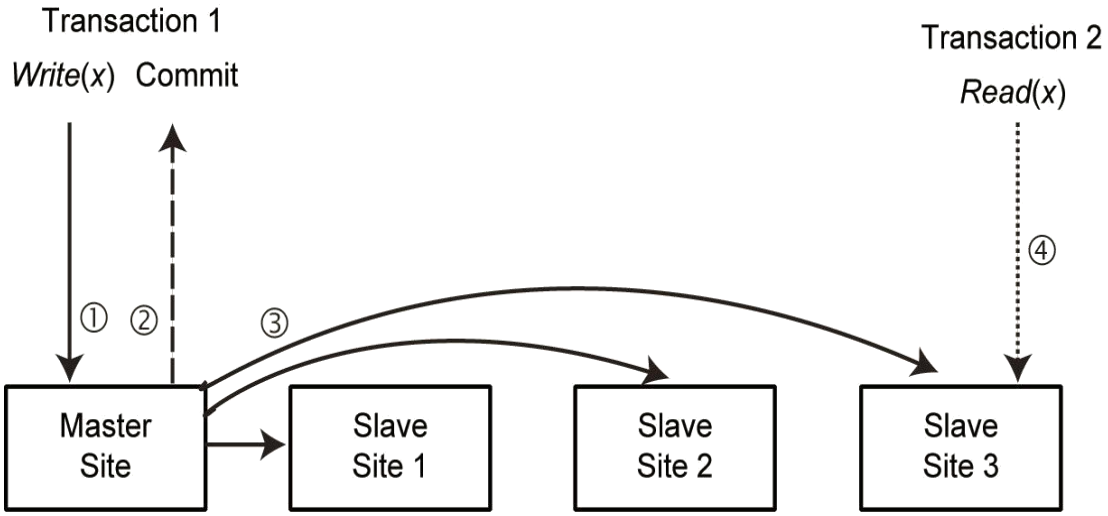
DVS

Lazy Single Master/Limited Transp.



TECHNISCHE UNIVERSITÄT DARMSTADT

- Update transactions submitted to master
- Master:
  - Upon read: read locally and return to user
  - Upon write: write locally and return to user
  - Upon commit/abort: terminate locally
  - Sometime after commit: multicast updates to slaves (in order)
- Slaves:
  - Upon read: read locally
  - Refresh transactions: install updates



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

DVS

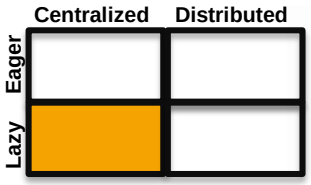
Lazy Primary Copy/Limited Transparency

TECHNISCHE UNIVERSITÄT DARMSTADT

- There are multiple masters; each master execution is similar to lazy single master in the way it handles transactions
- Slave execution complicated: refresh transactions from multiple masters that need to be ordered properly

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Lazy Primary Copy/Limited Transp. – Slaves

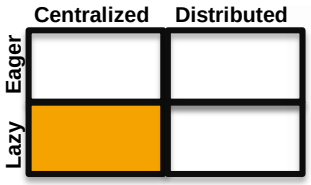


TECHNISCHE UNIVERSITÄT DARMSTADT

- Assign system-wide unique timestamps to refresh transactions and execute them in timestamp order
  - May cause too many aborts
- Replication graph
  - Similar to serialization graph, but nodes are transactions (T) + sites (S); edge ⟨ Ti,Sj⟩exists iff Ti performs a Write(x) and x is stored in Sj
  - For each operation k (opk), enter the appropriate nodes (Tk) and edges; if graph has no cycles, no problem
  - If cycle exists and the transactions in the cycle have been committed at their masters, but their refresh transactions have not yet committed at slaves, abort Tk; if they have not yet committed at their masters, Tk waits.
- Use group communication

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Lazy Single Master/Full Transp.

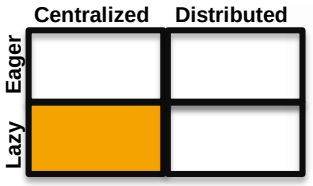


TECHNISCHE UNIVERSITÄT DARMSTADT

- Difficult:
  - Forwarding operations to a master and receiving refresh transactions cause difficulties
  - Read and update Transactions submitted to any site
    - Resulting Reads and Writes forwarded to any site as refresh Tx
- Problems:
  - Violation of 1SR
  - A transaction may not see its own writes
- Problem arises in primary copy/full transparency as well

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Example



TECHNISCHE UNIVERSITÄT DARMSTADT

Site *M* (Master) holds *x, y*; Site *B* holds slave copies of *x, y*

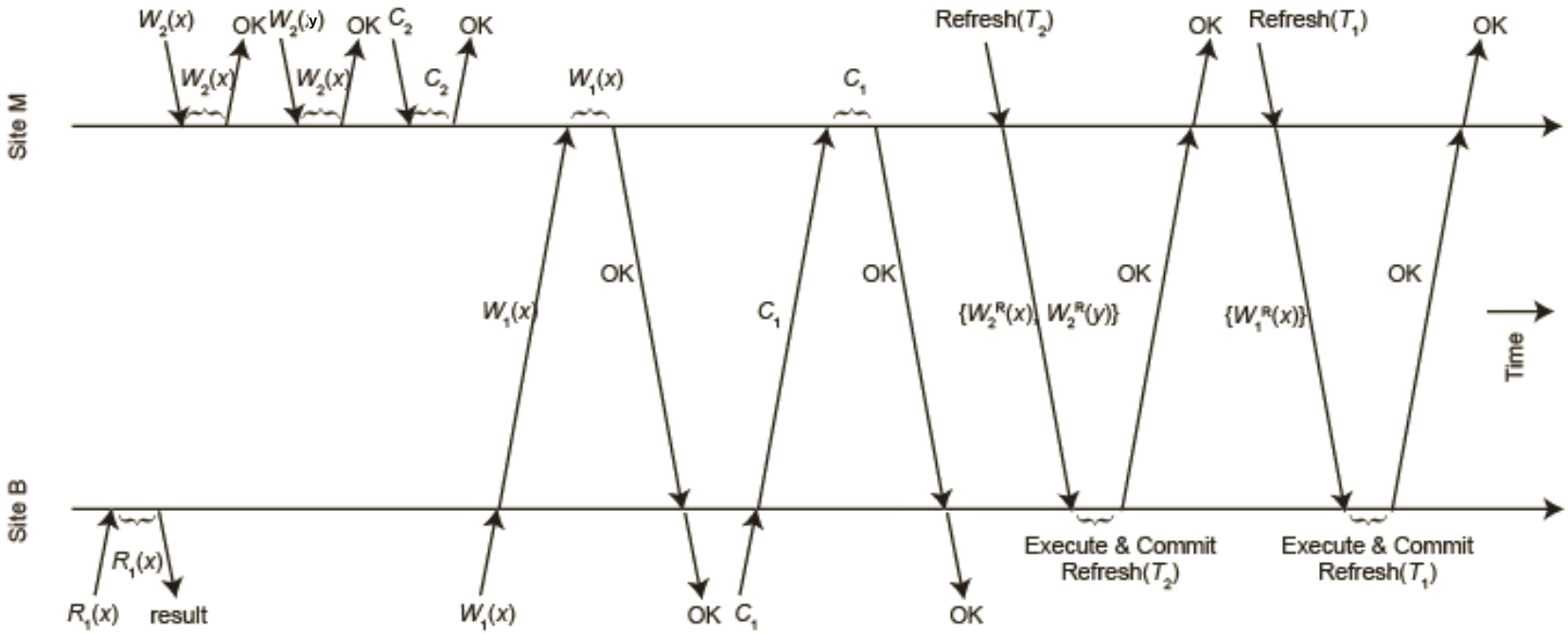
T_1: Read(*x*), Write(*y*), Commit

T_2: Write(*x*), Write(*y*), Commit

Spot mistake in figure!

$$H_M = \{W_2(x_M), W_2(y_M), C_2, W_1(y_M), C_1\}$$

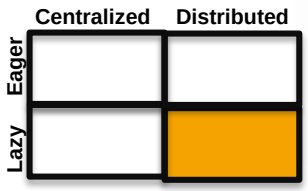
$$H_B = \{R_1(x_B), C_1, W_2^R(x_B), W_2^R(y_B), C_2^R, W_1^R(x_B), C_1^R\}$$



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

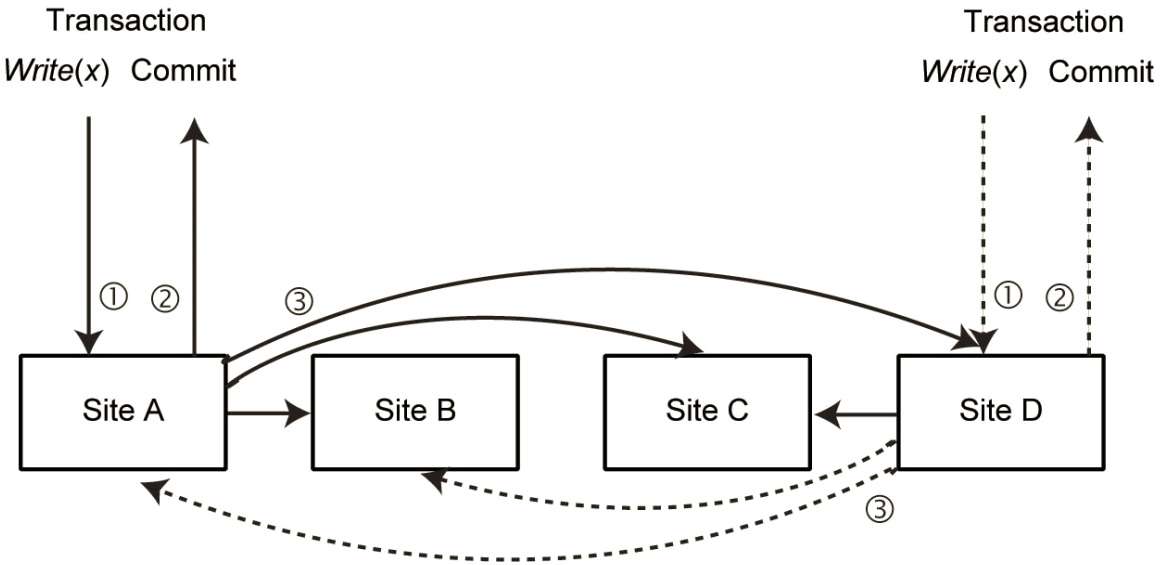
DVS

Lazy Distributed Replication



TECHNISCHE UNIVERSITÄT DARMSTADT

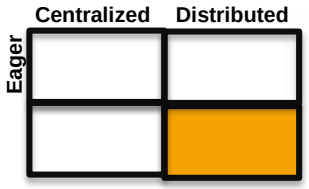
- Any site:
  - Upon read: read locally and return to user
  - Upon write: write locally and return to user
  - Upon commit/abort: terminate locally
  - Sometime after commit: send refresh transaction
  - Upon message from other site
    - Detect conflicts
    - Install changes
    - Reconciliation may be necessary



M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

DVS

Reconciliation



TECHNISCHE UNIVERSITÄT DARMSTADT

- Such problems can be solved using pre-arranged patterns:
  - Latest update wins
    - newer updates preferred over old ones
  - Site priority
    - preference to updates from headquarters
  - Largest value
    - the larger transaction is preferred

- Alternative: ad-hoc decision making procedures
  - Identify the changes and try to combine them
  - Analyze the transactions and eliminate the non-important ones

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Replication Strategies

TECHNISCHE UNIVERSITÄT DARMSTADT

	Centralized	Distributed
Eager	+ Updates do not need to be coordinated + No inconsistencies - Longest response time - Only useful with few updates - Local copies can only be read	+ No inconsistencies + Elegant (symmetrical solution) - Long response times - Updates need to be coordinated
Lazy	+ No coordination necessary + Short response times - Local copies are not up to date - Inconsistencies	+ No centralized coordination + Shortest response times - Inconsistencies - Updates can be lost (reconciliation)

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Group Communication

TECHNISCHE UNIVERSITÄT DARMSTADT

- A node can multicast a message to all nodes of a group with a delivery guarantee
- Multicast primitives
  - There are a number of them
  - Total ordered multicast
    - all messages sent by different nodes are delivered in the same total order at all the nodes

- Used with deferred writes, can reduce communication overhead
  - Remember eager distributed requires **k*m*n** messages (with multicast **k*m**) for throughput of **k** [Tx/sec] when there are **n** replicas and **m** update operations in each transaction
  - With group communication and deferred writes: 2k messages

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Failure Handling

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

Failures

TECHNISCHE UNIVERSITÄT DARMSTADT

- Replication protocols so far were considered in the absence of failures

- **Problem**: How to keep replica consistency when failures occur?
  - Site failures
    - ROWA – in eager approaches → low performance → low availability
    - Read One Write All Available (ROWAA)

  - Communication failures
    - Quorums

  - Network partitioning
    - Quorums

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

ROWAA with Primary Site

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Idea**: execute updates on all available sites
  - If an unavailable site becomes available later on → all updates forwarded

- READ = read any copy, if time-out, read another copy.
- WRITE = send W(x) to all copies.
  - If one site rejects the operation, then abort.
  - Otherwise, all sites not responding inserted into List_missing_writes.

- VALIDATION = To commit a transaction
  - Check if all sites in List_missing_writes are down. If not, then abort the Tx.
    - Check if unavailable sites were available when W(x) was performed
    - There might be a site recovering concurrent with transaction updates and these may be lost
  - Check that all sites that were available are still available. If some do not respond, then abort.

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Distributed ROWAA

TECHNISCHE UNIVERSITÄT DARMSTADT

- Each site has a copy of V
  - V represents the set of sites that are assumed available
  - V(A) is the “view” a site has of the system configuration.
- The view of a transaction T [V(T)] is the view of its coordinating site, when the transaction starts.
  - Read any copy within V; update all copies in V
  - If at the end of the transaction the view has changed, the transaction is aborted
- All sites must have the same view!
- To modify V, run a special atomic transaction at all sites.
  - Take care that there are no concurrent views!
  - Similar to commit protocol.
  - Idea: Vs have version numbers; only accept new view if its version number is higher than your current one
- Recovery: get missed updates from any active node
  - Problem: no unique sequence of transactions

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Quorum-Based Protocol

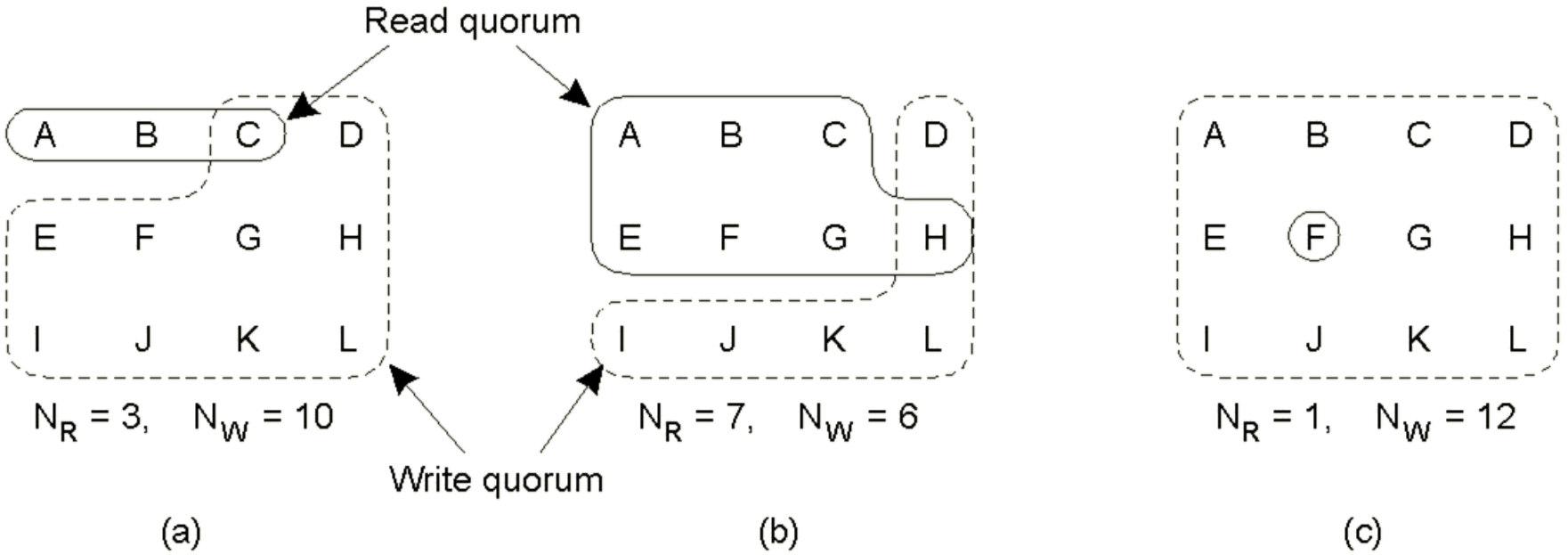
TECHNISCHE UNIVERSITÄT DARMSTADT

- Assign a vote to each copy of a replicated object (say Vi)
  - V is total number of votes → ∑(Vi) = V
- Each operation has to obtain
  - a read quorum (Vr) to read and
  - a write quorum (Vw) to write an object

- Then the following rules have to be obeyed in determining the quorums:
  - Vr+ Vw>V
    - an object is not read and written by two transactions concurrently
  - Vw>V/2
    - two write operations from two transactions cannot occur concurrently on the same object
- Major disadvantage: obtaining read quorum before reads ← Performance
- If no Network Partitioning occurred conditions hold

M. Tamer Ozsu and Patrick Valduriez Principles of Distributed Database Systems (3nd Ed.). Prentice-Hall. 2011

Quorum Example



Three examples of the voting algorithm:

a. A correct choice of read and write set
b. A choice that may lead to write-write conflicts
c. ROWA

For in-depth examples consider: George Coulouris, et al. **Distributed Systems: Concepts and Design.** Chapter 15

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms

PAXOS

TECHNISCHE UNIVERSITÄT DARMSTADT

- Proposed by Lamport (1989)
- Paxos is a whole family of consensus protocols in a network of unreliable processors/communication
  - We will limit discussion to Basic Paxos

- No deterministic fault-tolerant consensus protocol can guarantee progress in an asynchronous network
- Paxos guarantees safety (consistency), and the conditions that could prevent it from making progress are difficult to provoke.

Paxos assumptions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Processors
  - Operate at arbitrary speed
  - May experience failures
  - Processors with stable storage may rejoin the protocol after failures (following a crash-recovery failure model)
  - Processors do not lie and do not collude (i.e. no Byzantine failures)
- Network
  - Processors can send messages to any other processor
  - Messages are delivered without corruption but may be lost, reordered or duplicated
- Number of processors
  - Without reconfiguration, algorithm can make progress using 2F+1 processors in the face of F failed processors

Roles in Paxos

TECHNISCHE UNIVERSITÄT DARMSTADT

- Five different roles that a process can play
  - Client: issues a request and waits for a response (e.g. write a file)
  - Acceptor (Voters): Acceptors are collected into groups called quorums
    - Messages sent to one acceptor must be sent to all acceptors in the group
    - Messages received from an acceptor are ignored unless a copy is received from every acceptor in the quorum
  - Proposer: Proposer advocates a client request and tries to convince the members of a quorum to agree on it; acts as a coordinator
  - Learner: A learner acts on the request once a client request has been agreed upon by the acceptors.

  - Roles can be collapsed, i.e. each process participating in Paxos may be a Proposer, an Acceptor or a Learner.

Basic Paxos

TECHNISCHE UNIVERSITÄT DARMSTADT

- Each instance of the Basic Paxos protocol decides on a single output value.
- The protocol proceeds over several rounds.
- A Proposer should not initiate Paxos if it cannot communicate with at least a Quorum of Acceptors
- A successful round has two phases.
  - Phase 1a: Prepare
    - Proposer creates a proposal identified with a unique ID > any previously used ID
    - Proposer sends a prepare message with the proposal to a quorum of Acceptors
  - Phase 1b: Promise
    - If ID is greater than any previously seen ID at that Acceptor, Acceptor promises to ignore any future proposals with lower ID
    - If Acceptor accepted any proposal in the past it must tell the proposer its ID

Basic Paxos (cont.)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Phase 2a: Accept request
  - If Proposer receives enough promises from a quorum it sets the value at the value of the highest response
  - If no quorum member has accepted a proposal to this point, Proposer sets the value
- Phase 2b:
  - An Acceptor must accept a request message if it has not previously promised not to consider proposals with a higher ID
  - If it accepts a request it sets the value to v and sends an Accepted message to the Proposer and the Learners
  - Even if different values are accepted, eventually they will converge
  - A round fails if conflicting Prepare messages are received or if Proposer fails to get a quorum → restart with a higher ID

Recommended reading: wikipedia

Quorums in Paxos

TECHNISCHE UNIVERSITÄT DARMSTADT

- Quorums are defined in such a way that they contain at least one overlapping Acceptor
- In its simplest form each Acceptor has one vote but it is also possible to assign weights
- If weights are used a quorum must have an aggregated weight greater than half the total possible weight
- Each attempt to define an agreed value v is performed with proposals which may or may not be accepted by Acceptors. Each proposal is uniquely numbered for a given Proposer.

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Consistency of Replicated Databases/Information Systems
- Update Management Strategies
- Replication Protocols
  - Eager Centralized Protocols
  - Eager Distributed Protocols
  - Lazy Centralized Protocols
  - Lazy Distributed Protocols
  - Group Communication
- Replication and Failures
- Quorum-Based Techniques

Thank You!

TECHNISCHE UNIVERSITÄT DARMSTADT

Questions?

DVS