Nonblocking commit protocols

Dale Skeen, SIGMOD'81

—

Jingchao Fang, Zhuoer Tong

Abstract

-Protocols that allow operational sites to continue transaction processing even though site failures have occurred are called **nonblocking**.

-This paper investigates the properties of nonblocking protocols.

-This paper also introduced central site nonblocking protocol and decentralized site nonblocking protocol.

Background

-Researchers have started to focus on distributed database systems since *late 70s*.

-Distributed crash recovery is vital.

-Some operations on the data are logically indivisible, and these operations are called **transactions**.

Background

-**Atomic operation**: either executes to completion or it appears never to have executed. By definition, transaction on distributed database system is atomic operation.

-In reality, a transaction is *rarely* a physically atomic operations.

-The gap between logical atomicity and physical atomicity causes significant problems in distributed database implementation.

Background

-Preserving transaction atomicity is well understood in *single site case*.

-Site decides to **commit** or **abort** when reaching **commit point.**

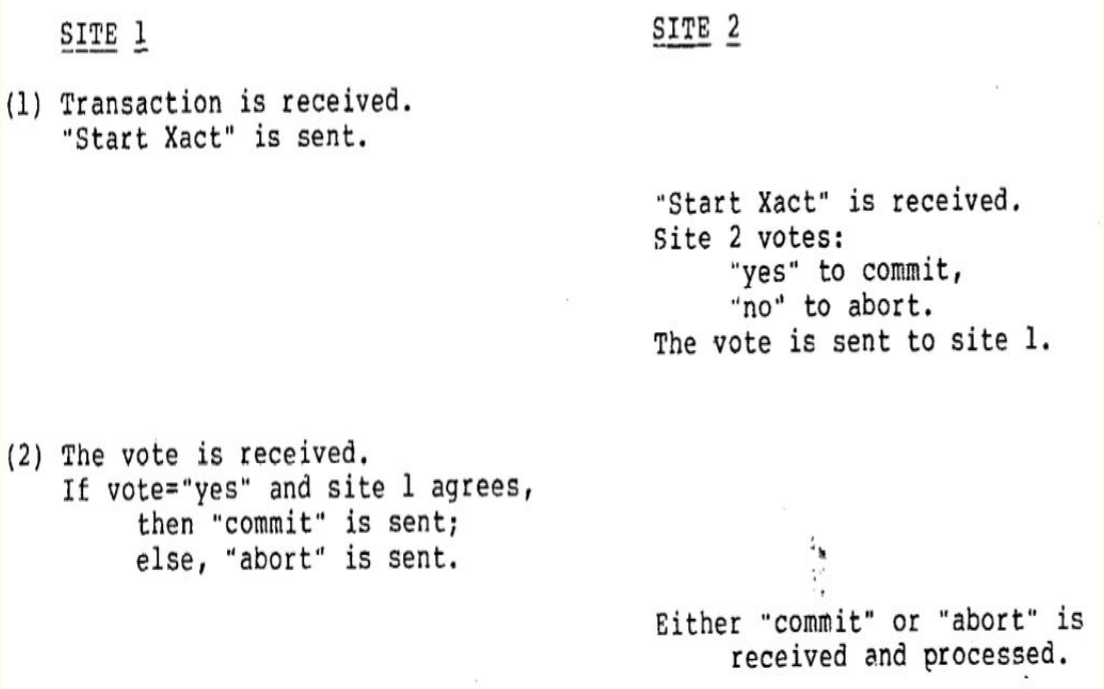
-Unconditionally guaranteed, irreversible.

-Problem occurs when more than one site is involved.

Two Phase Commit Protocol

-The simplest commit protocol that allows unilateral abort.

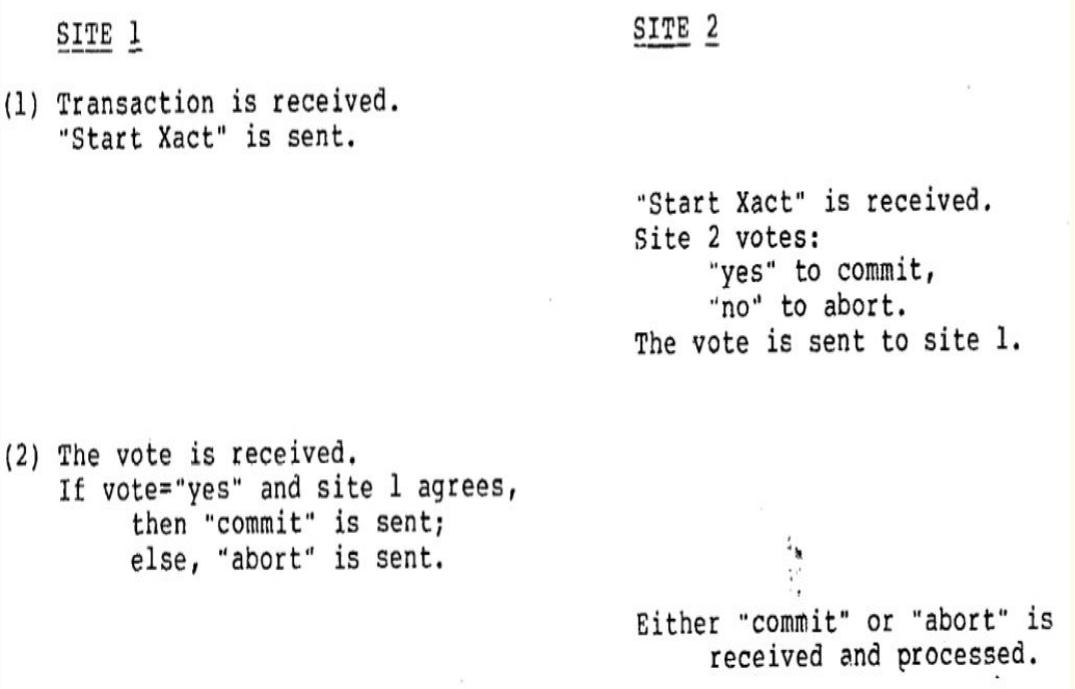
-An example of a blocking protocol.



Two Phase Commit Protocol: Process

-**First Phase**: coordinator distribute the transaction to all sites, and each site individually votes.

-**Second Phase**: coordinator collects all the votes and informs outcome.



Two Phase Commit Protocol: Summary

-Simplest and cheapest in number of messages, but can be block on site failures.

-An example of a **blocking protocol**: operational sites sometimes wait on the recovery of a failed sites. *Locks* must be held on the database while the transaction is blocked.

-A protocol that never requires operational sites to block until a failed site has recovered is called a **nonblocking protocol**.

Termination and Recovery Protocols

-**Termination Protocol**: to terminate transaction execution as quickly as possible at the operational sites.

-**Recovery Protocol**: invoked by failed sites to resume transaction processing. (not discussed in this paper)

Formal Model Describing Commit Protocols

-Transaction execution at each site models as **finite state automaton** (FSA).

-Network models as input/output tapes to all sites.

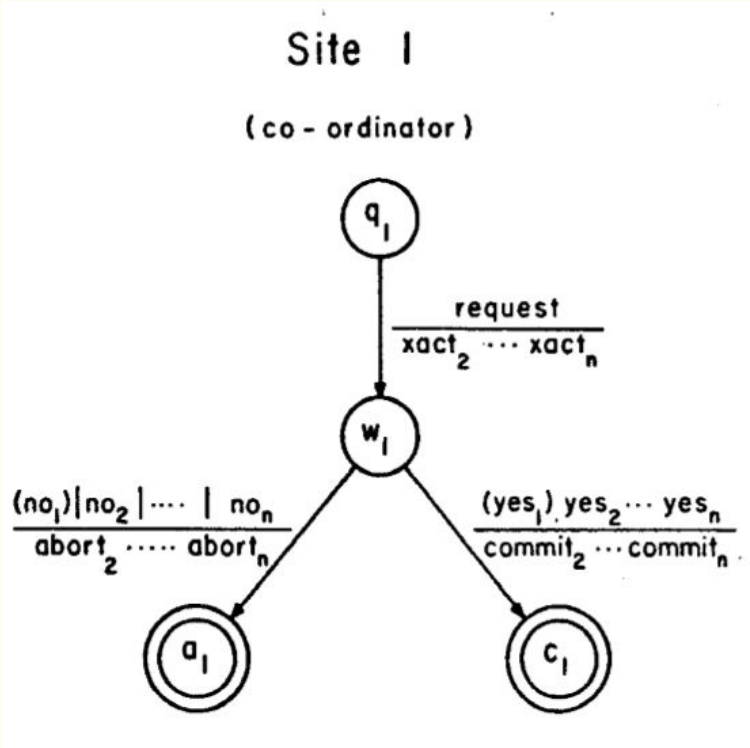
-The states of the FSA for site i are called **local states** of site i.

-State of transition: reading, writing, and moving to the next local state.

FSA for Two Phase Commit Protocol

-Each FSA has four (local) states: initial ($\mathbf{q_i}$), wait ($\mathbf{w_i}$), abort ($\mathbf{a_i}$), and commit ($\mathbf{c_i}$).

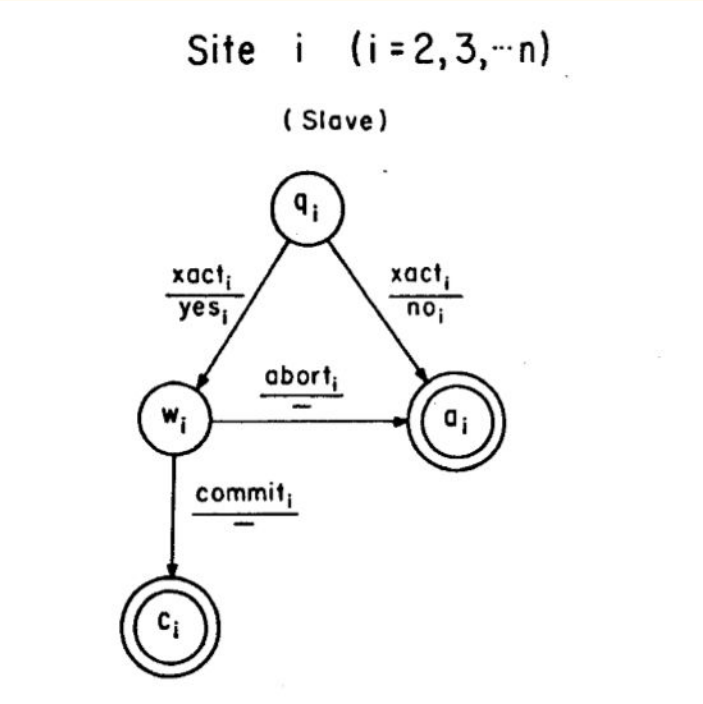
-Abort and commit are final states.



FSA for Two Phase Commit Protocol

-Local state for site **i** are subscripted with **i**.

-Messages sent or received by a slave are subscripted with that slave's site number.



Properties for Commit Protocol

-The FSA's are nondeterministic.

-The final states of the FSA's are partitioned into two sets: the abort states, and the commit states.

-The act of committing or aborting is irreversible.

-The state diagram describing a FSA is acyclic.

Global Transaction States

-**Global state** defines the complete processing state of a transaction.

-A global state is said to be **inconsistent** if it contains both a local commit state and a local abort state.

Global Transaction States

-A global state is said to be a **final state** if all local states contained in the state vector are final states.

-A global state is said to be a **terminal state** if from it there is no immediately reachable successors.

-A terminal state that is not a final state is a **deadlock state.**

-The **concurrency set** of $\mathbf{s_i}$ state is the set of all local states $\mathbf{s_j}$, where $i \neq j$, such that $\mathbf{s_i}$ and $\mathbf{s_j}$ are contained in the same reachable global state.

Committable States

-A local state is called **committable** if occupancy of that state by any site implies that all sites have voted yes on committing the transaction.

-For **nonblocking protocol:** have *more than one* committable state (assert in paper and without proof)

The Central Site Class Commit Protocol

-The central site class: uses one site (**coordinator**) to direct transaction processing at all participating sites (**slaves**).

-**Properties**:

1) Single coordinator;

2) All other participants execute the slave protocol;

3) Slaves can communicate only with coordinator;

4) Coordinator sends the same message and wait.

The Central Site Class Commit Protocol

-Advantages: Relatively cheap, conceptually simple.

-Disadvantage: Vulnerability to a coordinator failure.

-**Synchronous within one state transition:** one site never leads another site by more than one state transition during the execution of the protocol.

The Decentralized Class Commit Protocol

-The (fully) decentralized class: each site participates as an equal in the protocol and executes the same protocol.

-Every site communicates with every other site.

-Process: each site will send the identical message to every other site, then waits until it has received messages from all its cohorts.

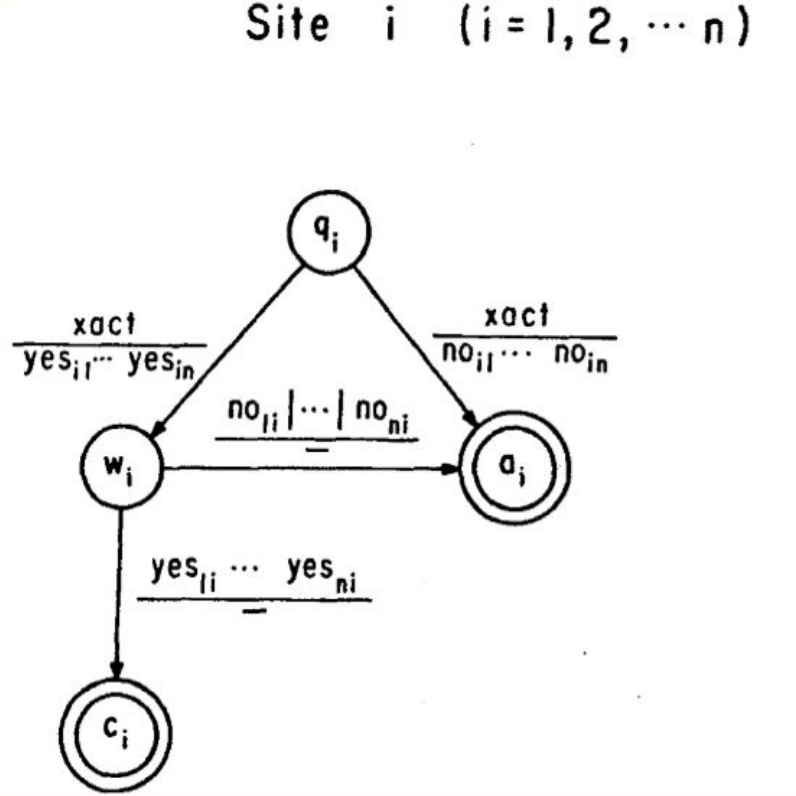
Decentralized Two Phase Commit Protocol

-Simplest decentralized commit protocol.

-**First phase:** each site receives the “start xact”, make the decision and sends to other cohorts.

-**Second phase:** each site accumulates all the abort decisions and move to a final state.

-Also synchronous within one state transition.



The Fundamental Nonblocking Theorem

-Blocking situation arises.

-The fundamental nonblocking theorem.

-A protocol is **nonblocking** *if and only if* it satisfies:

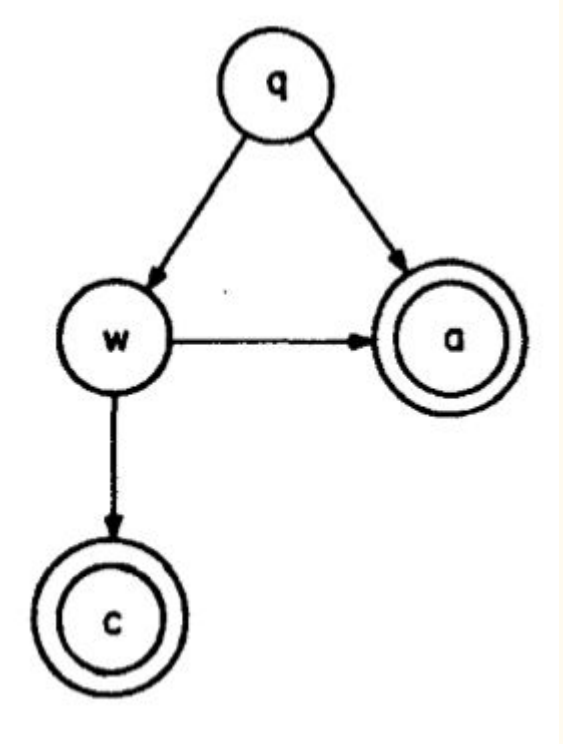
1) there exists no local state such that its concurrency set contains both an abort and a commit state;

2) there exist no noncommitable state whose concurrency set contains a commit state.

The Canonical Two Phase Commit Protocol

-The concurrency set of state **q** contains **q**, **w** and **a**. The concurrency set of state **w** contains all of the local states of the protocol.

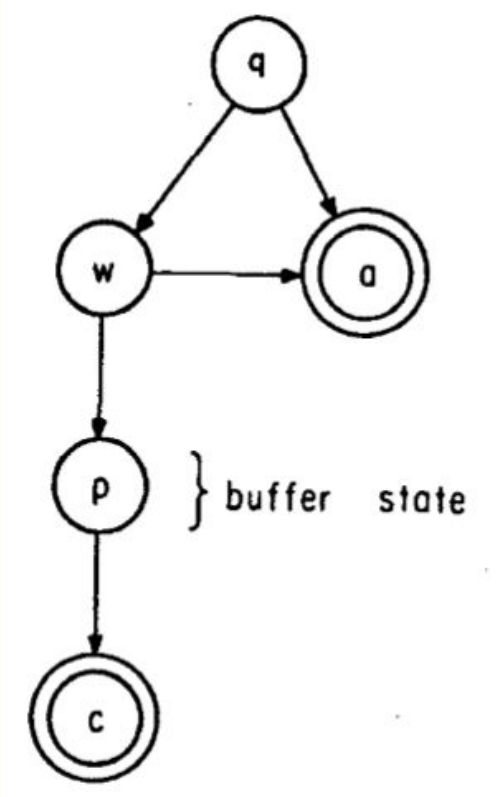
-Lemma: A protocol that is synchronous within one state transition is nonblocking if and only if: 1) it contains no local state adjacent to both a commit and an abort state, and 2) it contains no noncommittable state that is adjacent to a commit state.



Buffer States and Canonical Nonblocking Protocol

-The buffer state can be thought of a “prepare to commit” state (labelled as **p** in the graph).

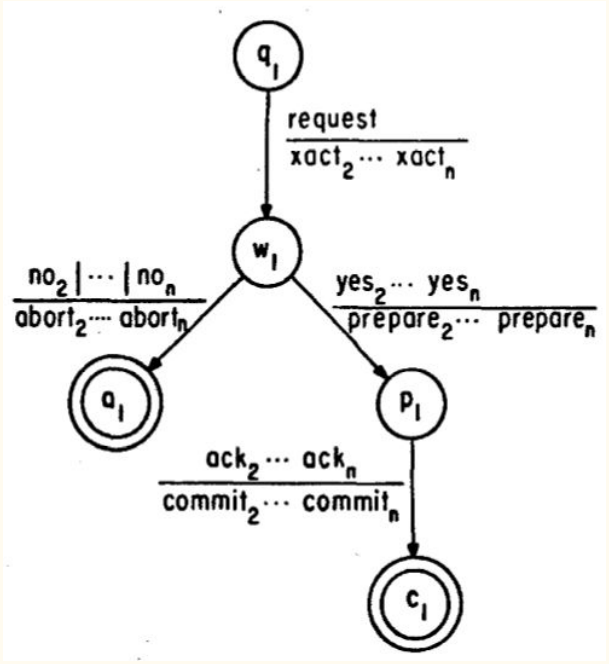
-This protocol can be referred as *canonical nonblocking protocol*, which is a three phase protocol.



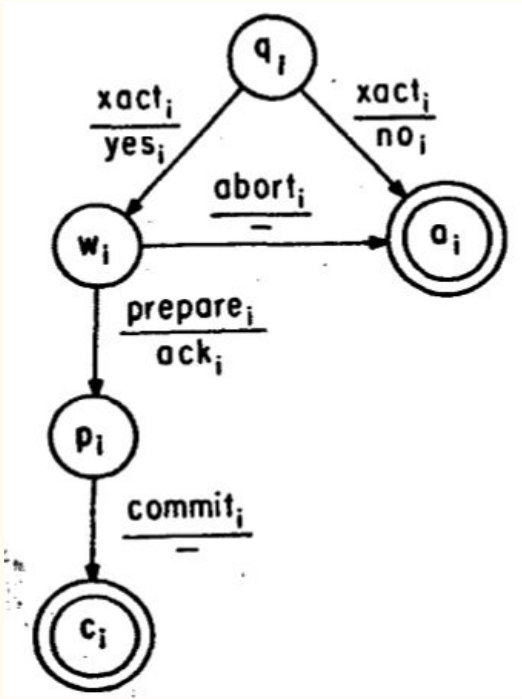
Nonblocking Central Site Protocol

-The slave protocol is the three phase protocol.

-The coordination protocol is also a three phase protocol that is a extension of the two phase coordinator protocol.



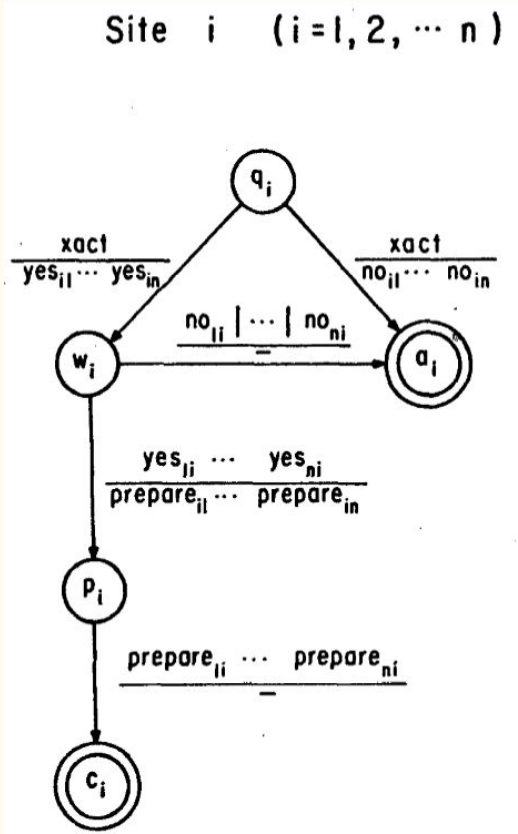
coordination



slave

Nonblocking Decentralized Protocol

-All protocols are the canonical nonblocking protocol.



Termination Protocols

-Site failures may occur and would make the continued execution of the commit protocol impossible. *Solution*: termination protocol.

-A termination protocol can accomplish its task only if the current state of at least one operational site obeys the conditions given in the fundamental nonblocking theorem.

-But in the worst case, it will be able to terminate correctly only if all of the operational sites obey the fundamental nonblocking theorem.

Central Site Termination Protocol and Backup

-Choose a **backup coordinator** from the set of operational sites.

-The backup coordinator will complete the transaction by directing all the remaining sites toward a commit or an abort. The protocol must be reentrant.

-**Decision Rule For Backup Coordinators**: If the concurrency set for the current state of the backup contains a commit state, then the transaction is committed. Otherwise, it is aborted.

-**Phase 1**: Issue a message and wait.

-**Phase 2**: Issue a commit or abort.

Summary

-Two phase commit protocol

-Two most popular commit classes: central site and decentralized

-Fundamental nonblocking theorem

-Three phase commit protocol

-Termination protocol