Distributed transactions

Distributed transactions are an important paradigm for designing reliable and fault tolerant distributed applications, particularly those distributed applications which access shared data concurrently.

**General observations**

**Isolation**

**Atomicity and persistence**

**Two-phase commit protocol (2PC)**

**Distributed Deadlock**

---

**Distributed Deadlock**

- Multiple transactions may access objects of multiple servers resulting in a distributed deadlock. At object access the server lock manager locks the object for the transaction.

- Deadlock detection schemes try to find cycles in a wait-for graph.

---

**Edge Chasing**

- Distributed approach to deadlock detection

  - No global wait-for graph is constructed.

  - Each involved server has some knowledge about the edges of the wait-for graph.

  - Servers attempt to find cycles by forwarding messages (called probes).

  - Each distributed transaction $T$ starts at a server $\implies$ the coordinator of $T$.

  - The coordinator records whether $T$ is active or waiting for a particular object on a server.

  - Lock manager informs coordinator of $T$ when $T$ starts waiting for an object and when $T$ acquires the lock.

---

**Edge Chasing Algorithm**

**Transaction Priorities**
The algorithm consists of 3 steps: initiation, detection, and resolution.

**initiation**: server X notes that W is waiting for another transaction U; it sends the probe "W → U" to the server of B via the coordinator of U.

**detection**: detection consists of receiving probes and deciding whether a deadlock has occurred and whether to forward the probes.

Server Y receives the probe "W → U"; it notes B is held by transaction V and appends V to the probe to produce "W → U → V"; probe is forwarded to server Z via coordinator of V.

distributed approach to deadlock detection

no global wait-for graph is constructed.

each involved server has some knowledge about the edges of the wait-for graph.
servers attempt to find cycles by forwarding messages (called probes).
each distributed transaction T starts at a server → the coordinator of T.
the coordinator records whether T is active or waiting for a particular object on a server.
lock manager informs coordinator of T when T starts waiting for an object and when T acquires finally the lock.

**Transaction Priorities**

Every transaction involved in a deadlock cycle may cause the initiation of deadlock detection

several servers initiate deadlock detection in parallel

⇒ possible more than one transaction in a cycle is aborted.

**Example:**

transaciton T attempts to access an object A locked by U
transaction W attempts to access an object B locked by V

deadlock detected
distributed approach to deadlock detection
no global wait-for graph is constructed.
each involved server has some knowledge about the edges of the wait-for graph.
servers attempt to find cycles by forwarding messages (called probes).
each distributed transaction T starts at a server \( \rightarrow \) the coordinator of T.

the coordinator records whether T is active or waiting for a particular object on a server.
lock manager informs coordinator of T when T starts waiting for an object and when T acquires finally the lock.

Edge Chasing Algorithm
Transaction Priorities

Introduction
Group communication facilitates the interaction between groups of processes.

Motivation
Important issues
Conventional approaches
Groups of components
Management of groups
Message dissemination
Message delivery
Taxonomy of multicast
Group communication in J189
JGroups

Motivation
Many application areas such as CSCW profit immensely if primitives for a group communication are supported properly.

typical application for group communication
fault tolerance using replicated services, e.g., a fault-tolerant file service.
object localization in distributed systems; request to a group of potential object servers.
conferencing systems and groupware.

functional components (e.g., processes) are composed to a group; a group is considered as a single abstraction.
Important issues of group communication are the following:

- **Group membership**: the structural characteristics of the group; composition and management of the group.
- **Support of group communication**: the support refers to group member addressing, error handling for members which are unreachable, and the message delivery sequence.
- **Communication within the group**: uncasting, broadcasting, multicasting
- **Multicast messages** are a useful tool for constructing distributed systems with the following characteristics:
  - fault tolerance based on replicated services,
  - locating objects in distributed services,
  - multiple update of distributed, replicated data.

**Synchronization**

- the sequence of actions performed by each group member must be consistent.

---

**Groups of components**

**Classification of groups**

Groups can be categorized according to various criteria.

- **Closed vs. open group**
  - Distinction between flat and hierarchical group. A flat group may also be called a peer group.
  - Distinction between implicit (anonymous) and explicit group.
  - In the first case, the group address is implicitly expanded to all group members.

---

**Group management architecture**

Again, there are different approaches for providing the group management functionality.

- **centralized group managers**, realized as an individual group server,
- **decentralized approach**, i.e., all components perform management tasks,
  - requires replication of group membership information, i.e., consistency must be maintained,
  - joining and leaving a group must happen synchronously.

**Hybrid approach**
Introduction
Group communication facilitates the interaction between groups of processes.
Motivation
Important issues
Conventional approaches
Groups of components
Management of groups
Message dissemination
Message delivery
Taxonomy of multicast
Group communication in ISIS
JGroups

Message delivery is an important issue of group communication; two aspects are relevant:

a) who gets the message, and
b) when is the message delivered.

Atomicity
Sequence of message delivery
Ordering for message delivery

It is desired to deliver all messages sent to the group $G$ to all group members of $G$ in the same sequence, because otherwise we might get non-deterministic system behavior.

Example for group reconfiguration

$m_4$ is sent by $C_1$ before the group composition is modified. However, in order to guarantee atomicity, $m_4$ should not be delivered to $S_1$ and $S_2$ (since, due to the crash, it is no longer possible to deliver $m_4$ to $S_3$).
Message delivery is an important issue of group communication; two aspects are relevant:

a) who gets the message, and

b) when is the message delivered.

Atomicity

Sequence of message delivery

Ordering for message delivery

Determination of a correct sequence based on the before relation between two events modeling their causal dependency (see causally distributed breakpoints).

Example

1. \( T_1 \) sends \( N_1 \), and \( T_2 \) sends \( N_2 \) with \( N_2 \) dependent on \( N_1 \)
2. \( T_2 \) sends \( N_2 \) with \( N_1 \) and \( N_2 \) concurrent
3. at \( T_2 : N_2 \) is received before \( N_1 \)
4. at \( T_3 : N_3 \) is received after \( N_1 \)

This approach for message delivery introduces synchronization points. Synchronously ordered messages are delivered to all group members in-sync:

let \( N \) be a synchronously ordered message

all other messages \( N_i \) are delivered either before or after \( N \) has been delivered to all group members.

The ordering method enables the group to synchronize their local states (at synchronization points the group members have a common consistent state).

Sync-ordering

Delivery of messages without delay in the same sequence is not possible in a distributed system \( \Rightarrow \) ordering methods for message delivery.

synchronously, i.e. there is a system-wide global time ordering.
loosely synchronous, i.e. consistent time ordering, but no system-wide global (absolute) time.

Total ordering by sequencer

Virtually synchronous ordering

Sync-ordering