Fault Tolerance

**Causes of failure:**
- process failure
- machine failure
- network failure

**Goals:**
- **transparency**: mask (i.e., completely recover from) all failures
- **reliability**: exhibit a well defined failure behavior

**Elements:**
- Atomic Transactions
- commitment (commit protocols)
- Anomalies
  - **atomicity**: either all or none of the effects of the transaction are made permanent
  - **consistency**: the effect of concurrent transactions is equivalent to some serial execution
  - **isolation**: transactions cannot observe each other's partial effects
  - **durability**: once accepted, the effects of a transaction are permanent (until changed again, of course)

**Environment**
- Each node is assumed to have:
  - data stored in a partially/full replicated manner
  - stable storage (information that survives failures)
  - logs (a record of the intended changes to the data; write ahead, UNDO/REDO)
  - locks (to prevent access to data being used by a transaction in progress)

**Solution:**
- what causes blocking?

<table>
<thead>
<tr>
<th>Site Failures</th>
<th>At what point</th>
<th>Actions on recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinator</td>
<td>before writing Commit</td>
<td>Send Abort messages</td>
</tr>
<tr>
<td>Coordinator</td>
<td>after writing Commit but before writing Complete</td>
<td>Send Commit messages</td>
</tr>
<tr>
<td>Cohort</td>
<td>after writing Complete</td>
<td>None.</td>
</tr>
<tr>
<td>Cohort</td>
<td>after writing UndoRedo</td>
<td>None. Abort will occur.</td>
</tr>
<tr>
<td>Coordinator</td>
<td>after writing UndoRedo</td>
<td>Wait for message from Coordinator.</td>
</tr>
</tbody>
</table>

Transaction Model

**Transaction**
- A sequence of actions (typically read/write), each of which is executed at one or more sites, the combined effect of which is guaranteed to be atomic

**Atomic Transactions**
- Isolation: either all or none of the effects of the transaction are made permanent
- Consistency: the effect of concurrent transactions is equivalent to some serial execution
- Isolation: transactions cannot observe each other’s partial effects
- Durability: once accepted, the effects of a transaction are permanent (until changed again, of course)

**Environment**
- Each node is assumed to have:
  - data stored in a partially/full replicated manner
  - stable storage (information that survives failures)
  - logs (a record of the intended changes to the data; write ahead, UNDO/REDO)
  - locks (to prevent access to data being used by a transaction in progress)

Definitions

**Synchronous**
- A protocol is synchronous if any two sites can never differ by more than one transition. A state transition is caused by sending or receiving a message.

**Concurrency Set**
- For a given state, s, at one site the concurrency set, C(s), is the set of all states in which all other sites can be.

**Sender set**
- For a given state, s, at one site, the sender set, S(s), is the set of all other sites that can send messages that will be received in state s.

**Blocking**
- Blocking occurs when a site’s state, s, has a concurrency set, C(s), that contains both commit and abort states.

**Solution**
- Introduce additional states. This implies adding additional messages (to allow transitions to/from these new states). This implies adding at least one more “phase”.

---

**2-phase Commit Protocol**

**Coordinator**
- Before writing Complete: None. Abort will occur.
- After writing Commit but before writing Complete: Send Commit messages
- After writing Complete: Send Abort messages

**Cohort**
- Before writing Undo/Redo: None. Abort will occur.
- After writing Undo/Redo: Wait for message from Coordinator.

**3-phase Commit Protocol**

**Coordinator**
- Before writing Undo/Redo: None. Abort will occur.
- After writing Complete: Send Commit messages
- Before writing Undo/Redo: Send Abort messages

**Cohort**
- After writing Undo/Redo: Wait for message from Coordinator.
Rules for Adding New Transitions

Failure Transition Rule

For every nonfinal state, s, in the protocol, if C(s) contains a commit, then assign a failure transition from s to a commit state; otherwise, assign a failure transition from s to an abort state.

Timeout Transition Rule

For each nonfinal state, s, if site j is in S(s), and site j has a failure transition to a commit (abort) state, then assign a timeout transition from state s to a commit (abort) state.

Using these rules in the three phase commit protocol allows the protocol to be resilient to a single site failure.