Figure 14.1  A basic architectural model for the management of replicated data

Requests and replies

Clients

Front ends

Replica managers

Service
Figure 14.2   Services provided for process groups

- Join
- Group address expansion
- Group send
- Multicast communication
- Group membership management
- Leave
- Fail
- Join

Process group
Figure 14.3  View-synchronous group communication

a (allowed).

b (allowed).

p crashes

view \((p, q, r)\)

view \((q, r)\)

p crashes

view \((p, q, r)\)

view \((q, r)\)

c (disallowed).

d (disallowed).

p crashes

view \((p, q, r)\)

view \((q, r)\)

p crashes

view \((p, q, r)\)

view \((q, r)\)
Figure 14.4  The passive (primary-backup) model for fault tolerance
Figure 14.5  Active replication
Figure 14.6  Query and update operations in a gossip service
Figure 14.7  Front ends propagate their timestamps whenever clients communicate directly

![Diagram showing front ends (FE) propagating timestamps to resource managers (RM) and service nodes. FE nodes gossip to update RM states, while RM nodes exchange vector timestamps to maintain consistency.]
Figure 14.8 A gossip replica manager, showing its main state components

- Other replica managers
- Gossip messages
- Replica timestamp
- Update log
- Value timestamp
- Value
- Executed operation table
- Stable updates
- OperationID Update Prev
- FE
Figure 14.9  Committed and tentative updates in Bayou

Tentative update $t_i$ becomes the next committed update and is inserted after the last committed update $c_N$. 
Figure 14.10  Transactions on replicated data

Client + front end

getBalance(A)

Replica managers

A
A
A

Client + front end

U

deposit(B,3);

Replica managers

B
B
B
B
Figure 14.11 Available copies

Client + front end

- $T$:
  - `getBalance(A)`
  - `deposit(B, 3)`

Client + front end

- $U$:
  - `getBalance(B)`
  - `deposit(A, 3)`

Replica managers

- $X$: $A$
- $Y$: $A$

- $B$

- $M$: $B$

- $P$: $B$

- $N$: $B$
Figure 14.12 Network partition

Client + front end
withdraw(B, 4)

Network partition

Client + front end
deposit(B, 3);

Replica managers
## Gifford’s quorum consensus examples

<table>
<thead>
<tr>
<th></th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Latency (milliseconds)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replica 1</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>Replica 2</td>
<td>65</td>
<td>100</td>
<td>750</td>
</tr>
<tr>
<td>Replica 3</td>
<td>65</td>
<td>750</td>
<td>750</td>
</tr>
<tr>
<td><strong>Voting configuration</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replica 1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Replica 2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Replica 3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Quorum sizes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R$</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$W$</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

### Derived performance of file suite:

<table>
<thead>
<tr>
<th></th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latency</td>
<td>65</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>Blocking probability</td>
<td>0.01</td>
<td>0.0002</td>
<td>0.000001</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latency</td>
<td>75</td>
<td>100</td>
<td>750</td>
</tr>
<tr>
<td>Blocking probability</td>
<td>0.01</td>
<td>0.0101</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Figure 14.13  Two network partitions
Figure 14.14  Virtual partition

Replica managers

Virtual partition

Network partition

X  V  Y  Z
Figure 14.15  Two overlapping virtual partitions
Figure 14.16 Creating a virtual partition

Phase 1:

- The initiator sends a *Join* request to each potential member. The argument of *Join* is a proposed logical timestamp for the new virtual partition.
- When a replica manager receives a *Join* request, it compares the proposed logical timestamp with that of its current virtual partition.
  - If the proposed logical timestamp is greater it agrees to join and replies *Yes*;
  - If it is less, it refuses to join and replies *No*.

Phase 2:

- If the initiator has received sufficient *Yes* replies to have read and write quora, it may complete the creation of the new virtual partition by sending a *Confirmation* message to the sites that agreed to join. The creation timestamp and list of actual members are sent as arguments.
- Replica managers receiving the *Confirmation* message join the new virtual partition and record its creation timestamp and list of actual members.