

# Mixed-Mode Adaptation in Distributed Systems: A Case Study

Karun Biyani & Sandeep Kulkarni

Department of Computer Science & Engineering  
Michigan State University

# Outline

- Classification of adaptation
- Modeling adaptation
- Case Study
- Conclusion

# Why adapt?

- Self - \* systems adapt for reasons including
  - Discovery of bugs/error
  - Change in requirements
  - Change in environment conditions
- Adaptations are done at run-time

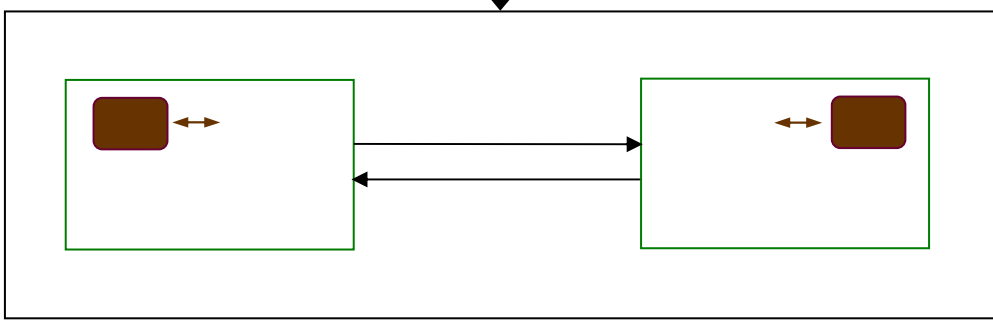
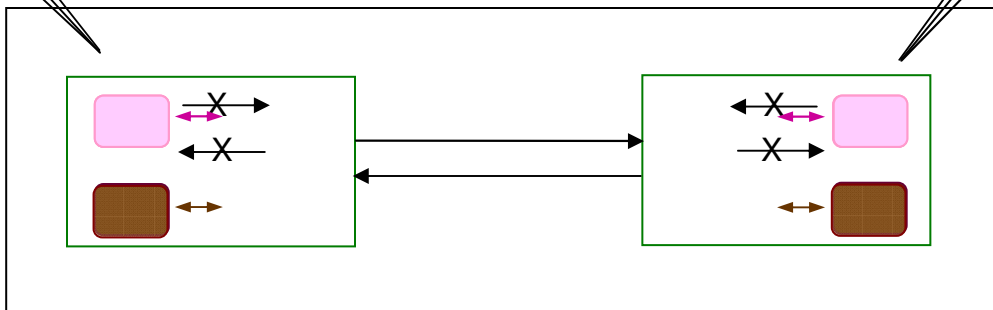
# Adaptation Classification

- Non-overlap adaptation
- Overlap adaptation
  - Quiescence
  - Parallel
  - Mixed-mode

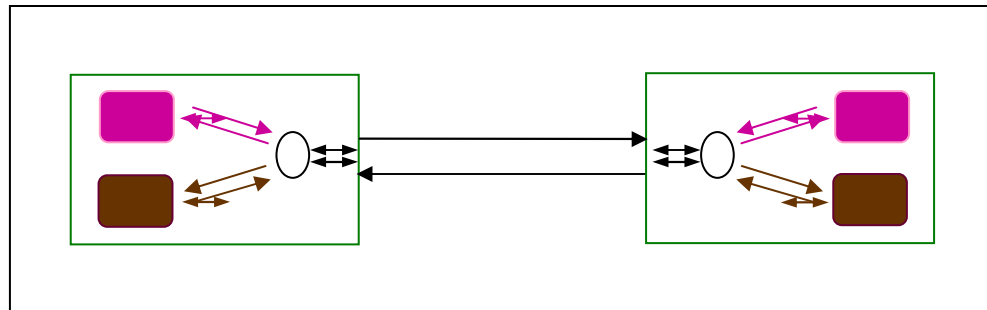
# Quiescence Adaptation

Don't initiate  
Passive  
Quiescent

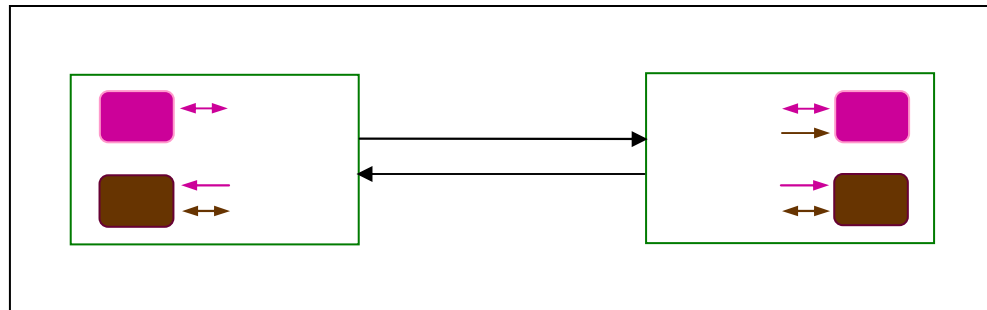
Don't initiate  
Passive  
Quiescent



# Parallel Adaptation



# Mixed-Mode Adaptation



- Expected advantages of mixed-mode adaptation  
*Service-interruption time* and *communication overhead* is reduced due to less need for synchronization

# Outline

- Classification of adaptation
- Modeling adaptation
- Case Study
- Conclusion



# Modeling Adaptation

- Three aspects of verification
  - System before adaptation
  - System during adaptation
  - System after adaptation
- Focus on *during* adaptation
  - Atomic steps in adaptation
  - Intermediate programs

# Program

- Program is a simple state machine or automaton

$$A = (S, \Sigma, \delta, S_0)$$

$S(A)$  = *set of states*

$\Sigma(A)$  = *set of actions*

$\delta(A)$  = *state transition relation*  $\delta(A) \subseteq S(A) \times \Sigma(A) \times S(A)$

$S_0(A)$  = *set of initial states*

- Each element  $(s, \pi, s')$  of  $\delta(A)$  is known as a *transition*

# Adaptation

- Adaptation is a 5-tuple

$$\Delta = (I, P, Q, \Sigma_a, S_{map})$$

$I(\Delta) = \text{set of automata}$

$P(\Delta) = \text{automaton of old program, } P \in I$

$Q(\Delta) = \text{automaton of new program, } Q \in I$

$\Sigma_a(\Delta) = \text{set of special actions known as adaptive actions}$

$S_{map}(\Delta) = \text{state mapping is a partial function } S(A) \times \Sigma_a(\Delta) \rightarrow S(A')$

- Each element  $((s, \pi_a), s')$  (equivalently,  $(s, \pi_a, s')$ ) of  $S_{map}$  is known as an *adaptive transition*

# Adaptation Lattices...

- Based on the definition of adaptation, we define
  - Adaptation lattice
    - For specifying properties of adaptation
  - Transitional-invariant lattice
    - For verifying correctness of adaptation
  - Transitional-faultspan lattice
    - For verifying fault-tolerance properties

# Adaptation as Program

- Adaptation can be viewed as an adaptive program

# Outline

- Classification of adaptation
- Modeling adaptation
- Case Study
- Conclusion

# Case Study

- Adaptation of network or distributed protocols as a special case of adaptation in distributed systems
- Replacement of leader election protocols
  - Id-based leader election protocol
  - Value-based leader election protocol
- Allow mixed-mode behavior

# System Model

- $n$  processes
- Bidirectional channels
- Static nodes
- No node or link failure during adaptation



# Properties of Leader Election Protocols

- Safety
  - No two stable processes can have different leaders
- Liveness
  - Eventually a unique leader is elected

# Variables in Leader Election Protocols

<i>Variable</i>	<i>Type</i>	<i>Meaning</i>
$N$	<i>list</i>	List of neighbors
$\epsilon$	<i>bool</i>	Indicates election
$ldr$	<i>int (1..n)</i>	<i>Id</i> of the leader node
$num$	<i>int</i>	Index of last election this node started
$src\langle Num, Id\rangle$	$\langle int, int\rangle$	Computation index of last election in which this node participated
$p$	<i>int (1..n)</i>	Parent node in last election
$ack$	<i>bool</i>	Indicates if ACK message is sent to parent or
$W$	<i>list</i>	<del>List of neighbors from whom ACK is being</del>
		<b><math>max\langle Val, Id\rangle</math> - Type:(<i>int, int(1..n)</i>)</b>
$C$	<i>list</i>	List of my children in last election
$max$	<i>int (1..n)</i>	Maximum <i>Id</i> among my children

# Messages in Leader Election Protocols

- Both protocols have same type of messages but message format is different

Message	Message Fields	Meaning
ELECT	<i>type,</i> <b><i>val - Type:(int, int(1..n))</i></b>	a spanning tree
ACK	<i>type, src, chd, id</i>	To acknowledge receipt of ELECT message
LD	<i>type, src, id</i>	To announce a leader

## Adaptation: *ldrId* to *ldrVal*

- Replacing Id-based leader election protocol with Value-based leader election protocol
- Specification during adaptation
  - Safety of leader election protocols continues to be true during adaptation

$$\forall p_1, p_2 : p_1, p_2 \in \{ldrId, ldrVal\} : (p_1.i.ldr \neq p_2.j.ldr \Rightarrow p_1.i.\mathcal{E} \vee p_2.j.\mathcal{E})$$

# Overlap Communication Scenarios

- Several overlap communication scenarios exist due to communication between fractions of *ldrId* and *ldrVal*

Message Type Overlap Scenario	ELECT	ACK	LD
Old ( <i>ldrId</i> ) to New ( <i>ldrVal</i> )	Discard	Discard	Accept
New ( <i>ldrVal</i> ) to Old ( <i>ldrId</i> )	Buffer	NA	NA

# State Mapping in Adaptation

- State mapping for each atomic adaptation during adaptation from *ldrId* to *ldrVal*

(old) state $s$ of process $i$	Description of (old) state $s$	(new) state $s'$ of process $i$
$\neg ldrId.\epsilon$	not in election	Identity mapping
$ldrId.\epsilon \wedge ldrId.W \neq \phi \wedge ldrId.src \neq i$	in election and waiting for <i>ACK</i> from children and is not source of election	$\{num, src\}$ - Identity mapping; Initial mapping
$ldrId.\epsilon \wedge ldrId.W = \phi$	in election and waiting for <i>LD</i>	Identity mapping
$ldrId.\epsilon \wedge ldrId.W \neq \phi \wedge ldrId.src = i$	in election and waiting for <i>ACK</i> from children and is source of election	$\{num, \epsilon\}$ - Identity mapping; $\{ldrVal.startElection := true;$ - Functional mapping; Initial mapping

# Adaptation as Program (*ldrId* to *ldrVal*)

**Program**  $\mathcal{P}_{ldrId-ldrVal}$

**process**  $i[i = 1, \dots, n]$

**var**  $idActive$  : **bool** {initially *true*}  
        $valActive$  : **bool** {initially *false*}  
        $ldrId.i.\{\mathbf{var}\} \cup ldrVal.i.\{\mathbf{var}\}$

**begin**

$idActive \wedge [\text{actions of } ldrId]$

[]  $valActive \wedge [\text{actions of } ldrVal]$

[]  $aa_i$  :  $idActive \rightarrow idActive, valActive := false, true;$   
            $s' := \Phi_{aa_i}(s)$

[] **discardElect** :  $valActive \wedge \text{ELECT}(ldrId) \in C_{j,i} \rightarrow C_{j,i}.\text{remove}(\text{ELECT})$

[] **discardAck** :  $valActive \wedge \text{ACK}(ldrId) \in C_{j,i} \rightarrow C_{j,i}.\text{remove}(\text{ACK})$

[] **acceptLd** :  $valActive \wedge \text{LD}(ldrId) \in C_{j,i} \rightarrow true \vee ldrVal.i.\text{setLeader}$

**end**

# Correctness of Adaptation

$I \equiv \boxed{P_E} \wedge \boxed{P'_E} \wedge \boxed{P_L} \wedge \boxed{P'_L}$ , where

$$P_E \equiv i.idActive \wedge j.valActive \wedge j.\epsilon \wedge j.ack \wedge j.W \neq \phi \Rightarrow \text{ELECT}(ldrVal) \in C_{j,i}$$

$$P'_E \equiv j.valActive \wedge \neg j.\epsilon \wedge (\exists i : i \in j.N : i.idActive \wedge j.src = i.src \wedge i.\epsilon \wedge i.ack \wedge C_{i,j} = \phi) \\ \Rightarrow \exists k : k.p = k \wedge k.\epsilon \wedge k.ack \wedge k.W \neq \phi$$

$$P_L \equiv i.idActive \wedge j.valActive \wedge j.\epsilon \wedge j.p = i \wedge \neg j.ack \wedge j.W = \phi \\ \Rightarrow ((i.\epsilon \wedge i.src = j.src \wedge \neg i.ack \wedge i.W = \phi) \vee \\ (\neg i.\epsilon \wedge \text{LD}(ldrId) \in C_{i,j}) \vee \\ (i.\epsilon \wedge i.ack \wedge i.W \neq \phi \wedge i.src = j.src) \vee \\ (i.\epsilon \wedge i.ack \wedge i.W \neq \phi \wedge i.src \neq j.src \Rightarrow j \in i.W))$$

$$P'_L \equiv i.idActive \wedge j.valActive \wedge i.\epsilon \wedge \neg j.\epsilon \wedge i.p = j \wedge \neg i.ack \wedge i.W = \phi \wedge i.src = j.src \\ \Rightarrow \text{LD}(ldrId) \in C_{j,i} \vee \exists k : k.p = k \wedge k.\epsilon \wedge k.ack \wedge k.W \neq \phi$$

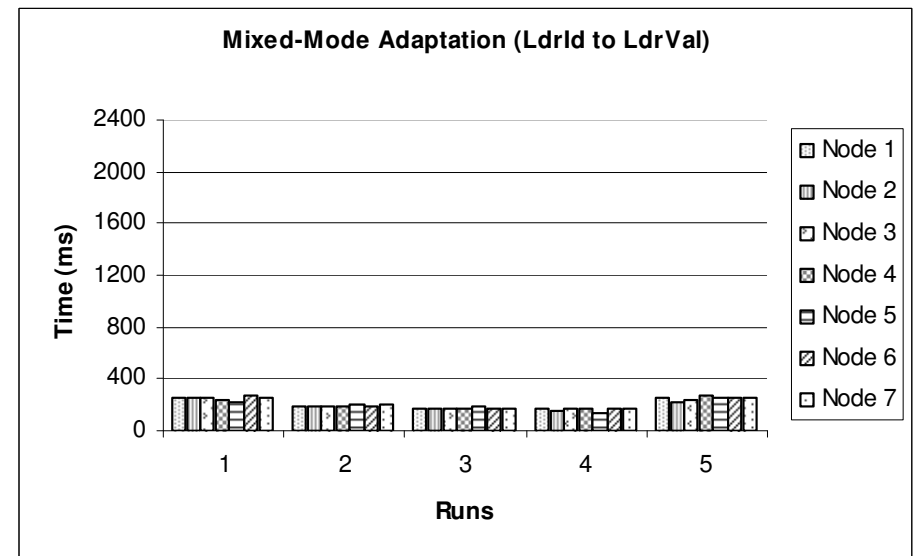
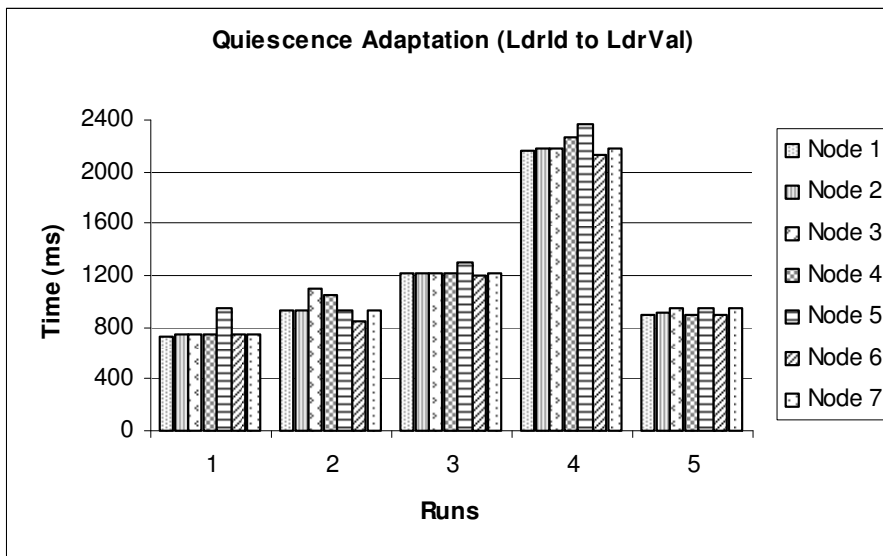
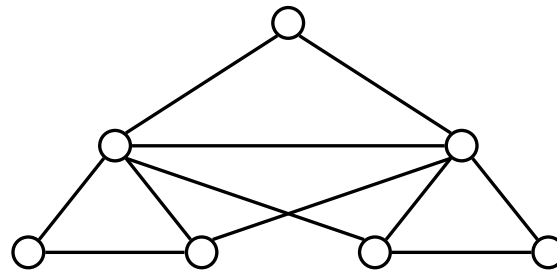


# Performance of Adaptation – Setup

- Machines in our labs (*SENS* south and Linux lab)
- Two configurations
  - 5 nodes, 4 edges (straight line)
  - 7 nodes, 11 edges
- One node is designated as adaptation initiator

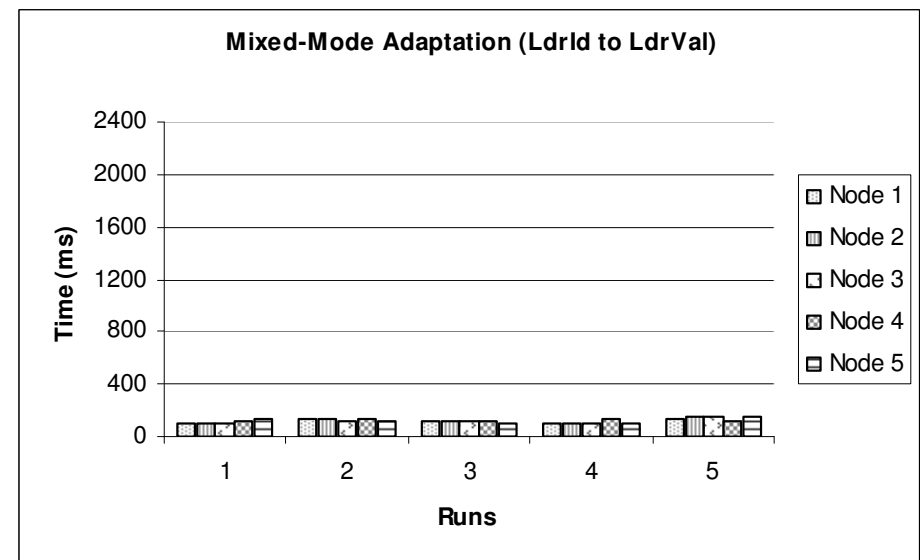
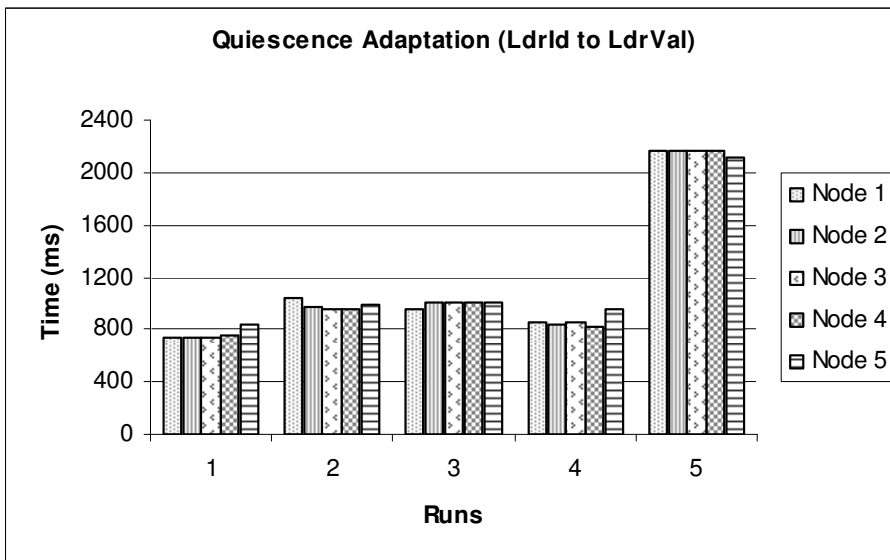
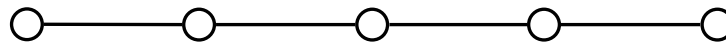
# Performance of Adaptation...

- Time for adaptation – configuration one



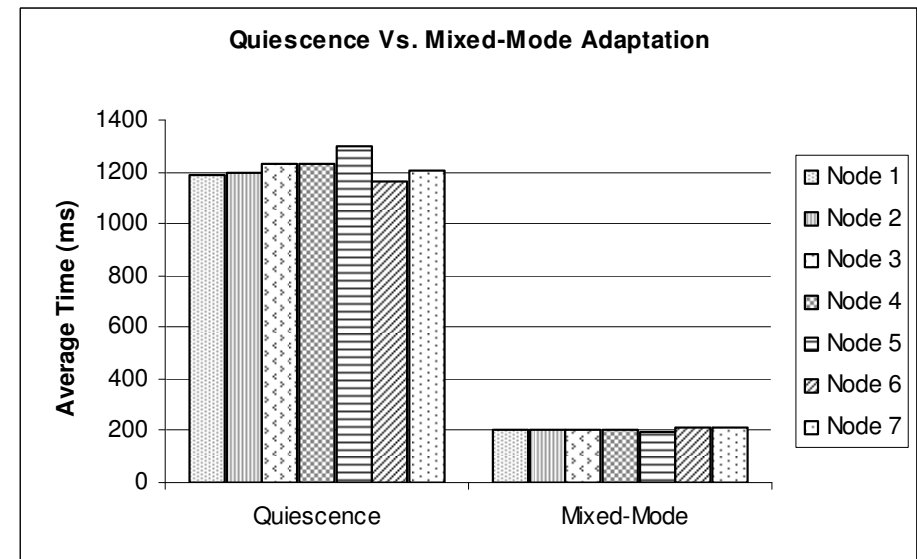
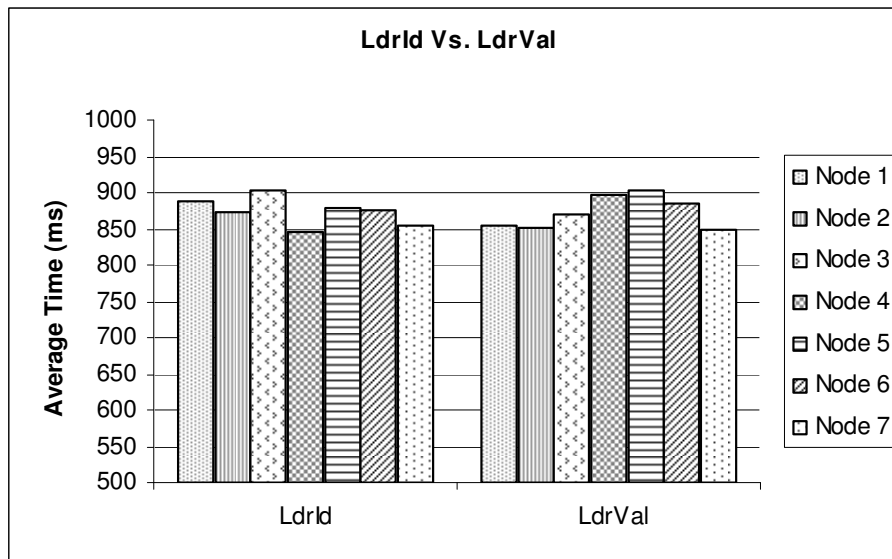
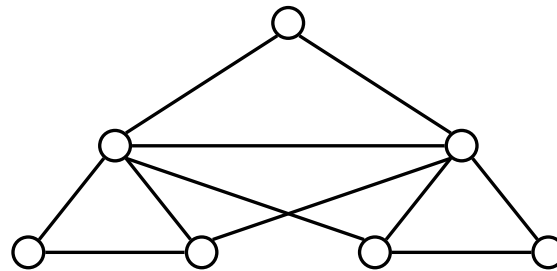
# Performance of Adaptation...

- Time for adaptation – configuration two



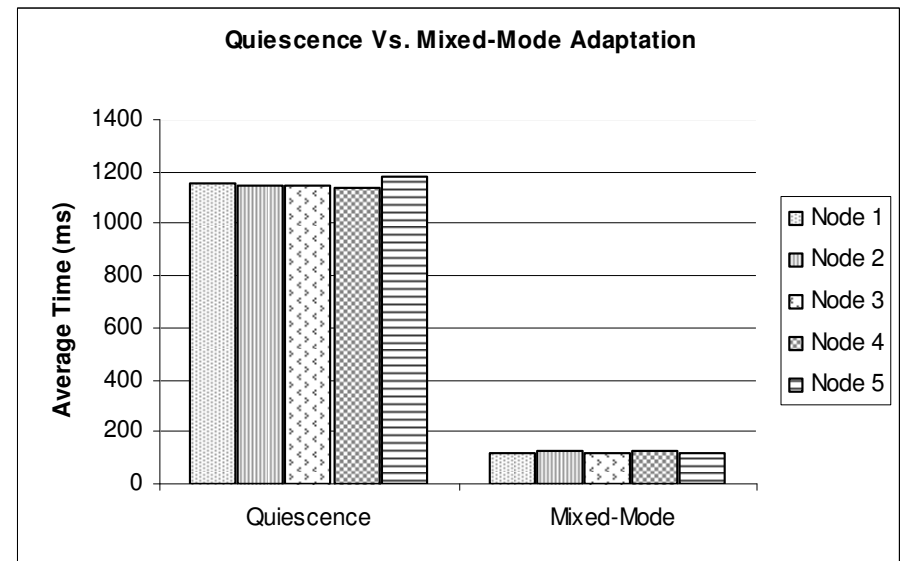
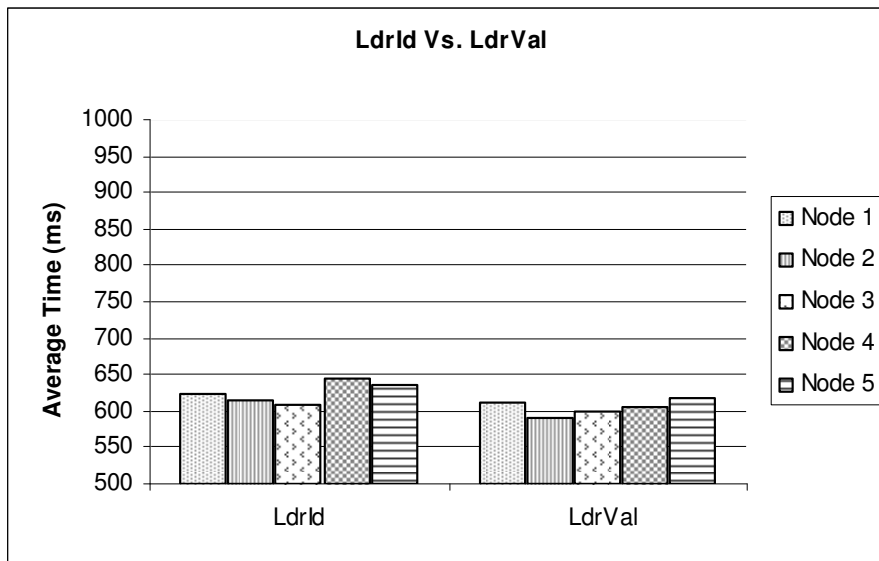
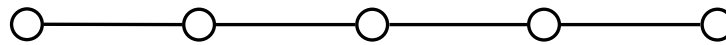
# Performance of Adaptation...

- Adaptation transparency – configuration one



# Performance of Adaptation...

- Adaptation transparency – configuration two



# Conclusion

- Mixed-mode adaptation provides better performance compared to quiescence adaptation
- Challenges
  - Design & Implementation
  - Assurance

# Questions / Comments ?

Thank You !!