Mixed-Mode Adaptation in Distributed Systems: A Case Study

Karun Biyani & Sandeep Kulkarni
Department of Computer Science & Engineering
Michigan State University

Outline

- Classfication of adaptation
- Modeling adaptation
- Case Study
- Conclusion

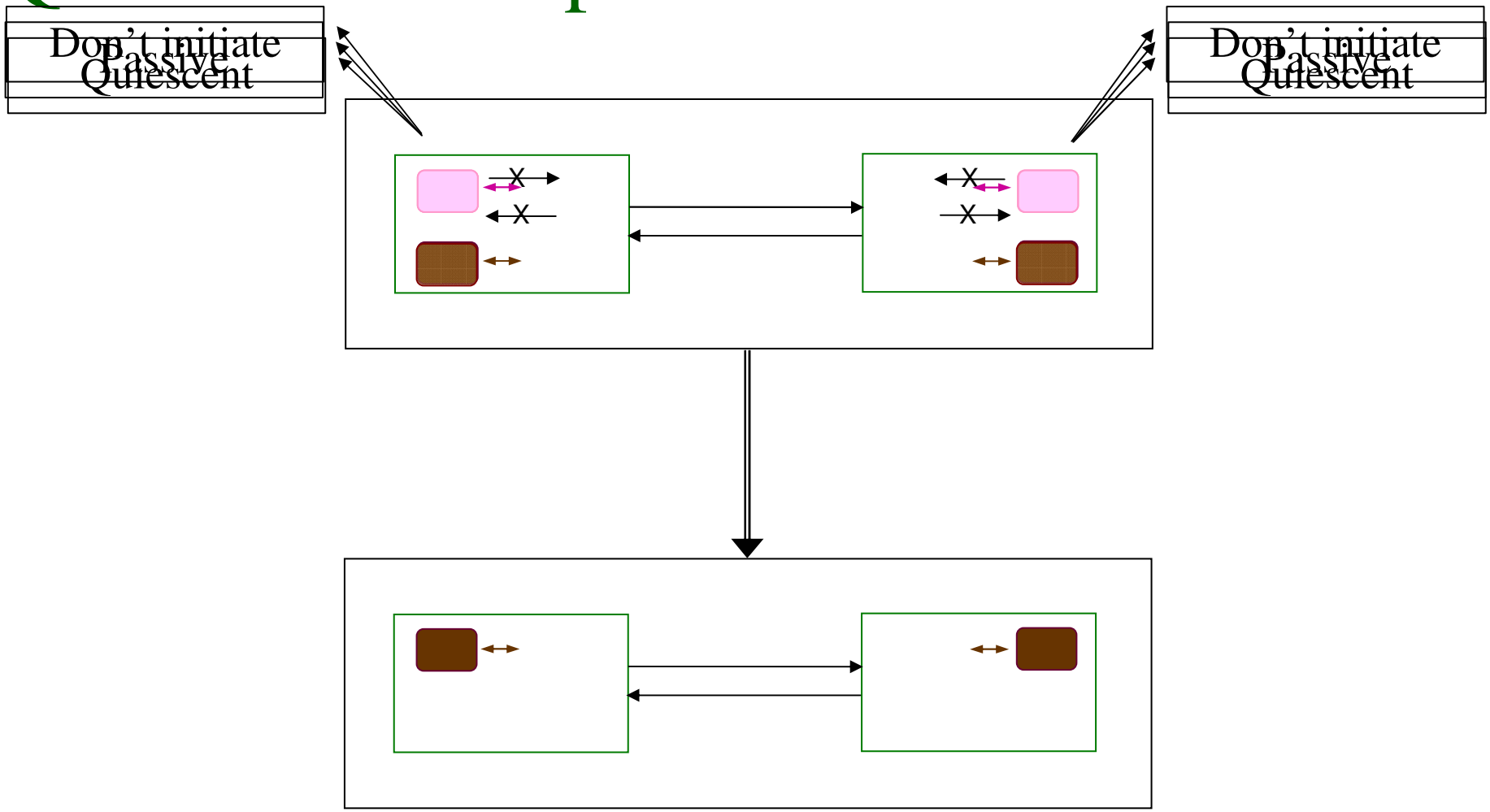
Why adapt?

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

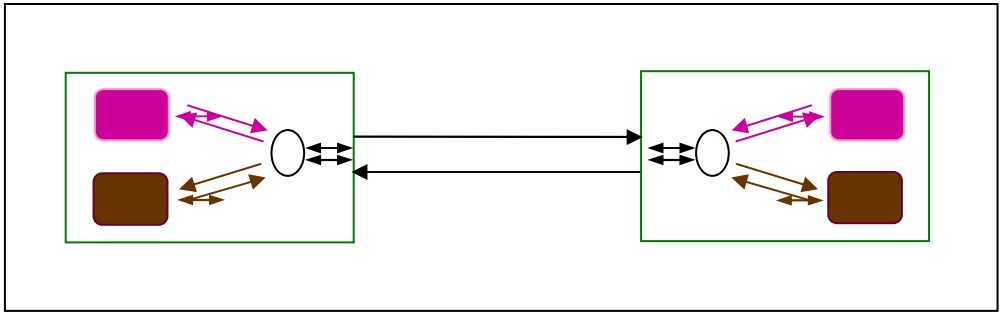
Adaptation Classification

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

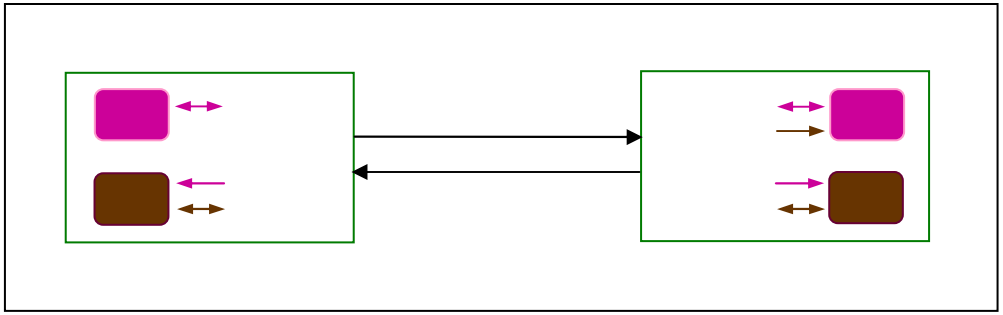
Quiescence Adaptation



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

- Classfication 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

$$\mathrm{A} = (S, \Sigma, \delta, S_0)$$

$S(\mathrm{A}) = set\ of\ states$

$\Sigma(\mathrm{A}) = set\ of\ actions$

$\delta(\mathrm{A}) = state\ transition\ relation\ \delta(\mathrm{A}) \subseteq S(\mathrm{A}) \times \Sigma(\mathrm{A}) \times S(\mathrm{A})$

$S_0(\mathrm{A}) = set\ of\ initial\ states$

- Each element (s, π, s') of $\delta(\mathrm{A})$ is known as a *transition*

Adaptation

- Adaptation is a 5-tuple

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

$I(\Delta) = set\ of\ automata$

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

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

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

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

- Each element $((s, \pi_a), s')$ (equivalently, (s, π_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

- Classfication 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

Variable	***Type***	***Meaning***
N	*list*	List of neighbors
ε	*bool*	Indicates election
ldr	*int (1..n)*	*Id* of the leader node
num	*int*	Index of last election this node started
src‹Num, Id›	*‹int, int›*	Computation index of last election in which this node participated
p	*int (1..n)*	Parent node in last election
ack	*bool*	Indicates if ACK message is sent to parent or not
W	*list*	List of neighbors from whom ACK is being
C	*list*	List of my children in last election
max	*int (1..n)*	Maximum *Id* among my children

max‹Val, Id›* - Type:*(int, int(1..n))

Messages in Leader Election Protocols

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

Message	Message Fields	Meaning
ELECT	*type*, *val* - Type:*(int, int(1..n))*	a spanning tree
ACK	*type*, *src*, *chd*, *id*	To acknowledge receipt of ELECT message
LD	*type*, *src*, *id*	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

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

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 (*ldrId*) to New (*ldrVal*)	Discard	Discard	Accept
New (*ldrVal*) to Old (*ldrId*)	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 ACK 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 LD	Identity mapping
$ldrId.\epsilon \wedge ldrId.W \neq \phi \wedge ldrId.src = i$	in election and waiting for ACK 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 $i[i = 1, .., n]$

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

begin

$idActive \wedge$ [actions of $ldrId$]

[] $valActive \wedge$ [actions of $ldrVal$]

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

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

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

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

end

Correctness of Adaptation

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

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

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

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

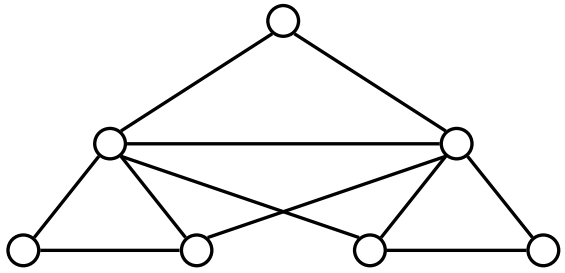
$$P'_L \equiv i.idActive \wedge j.valActive \wedge i.\epsilon \wedge \neg j.\epsilon \wedge i.p = j \wedge \neg i.ack \wedge i.W = \phi \wedge i.src = j.src$$
$$\Rightarrow \texttt{LD}(ldrId) \in C_{j,i} \vee \exists k : k.p = k \wedge k.\epsilon \wedge k.ack \wedge k.W \neq \phi$$

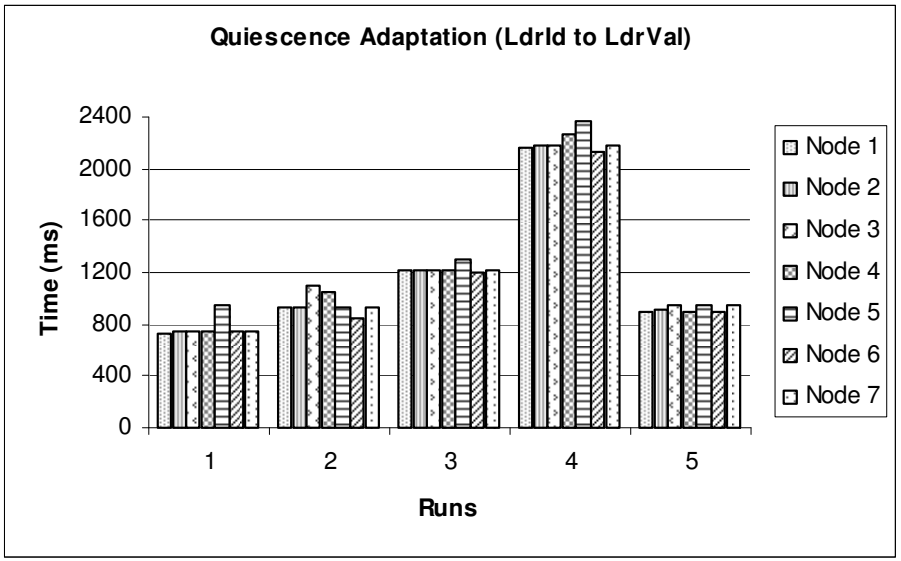
Performance of Adaptation – Setup

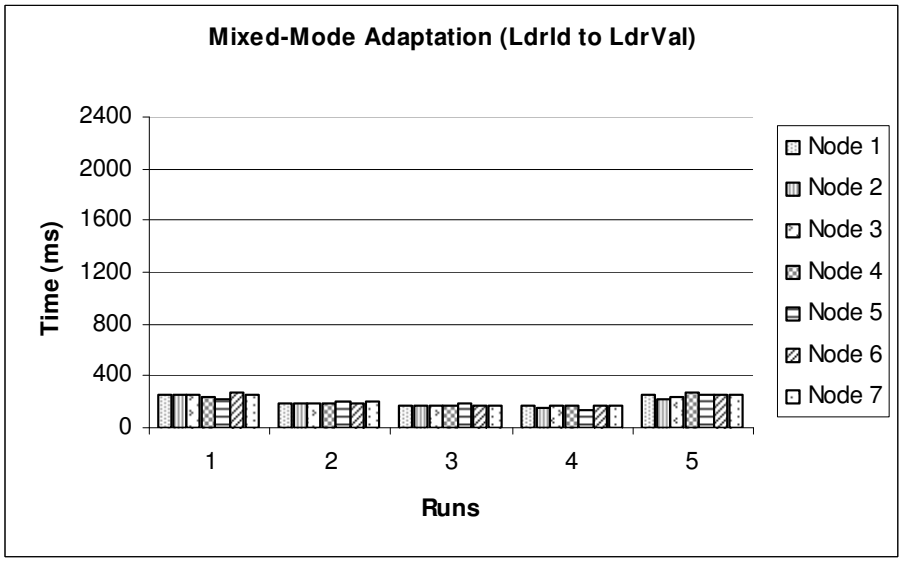
- Machines in our labs (*SENS* south and Linux lab)
- Two configurations
  - 5 nodes, 4 edges (straight line)
  - 7 nodes, 11 edges
- One node is designated as adaptation initiator

Performance of Adaptation…

- Time for adaptation – configuration one

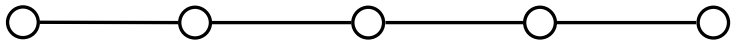


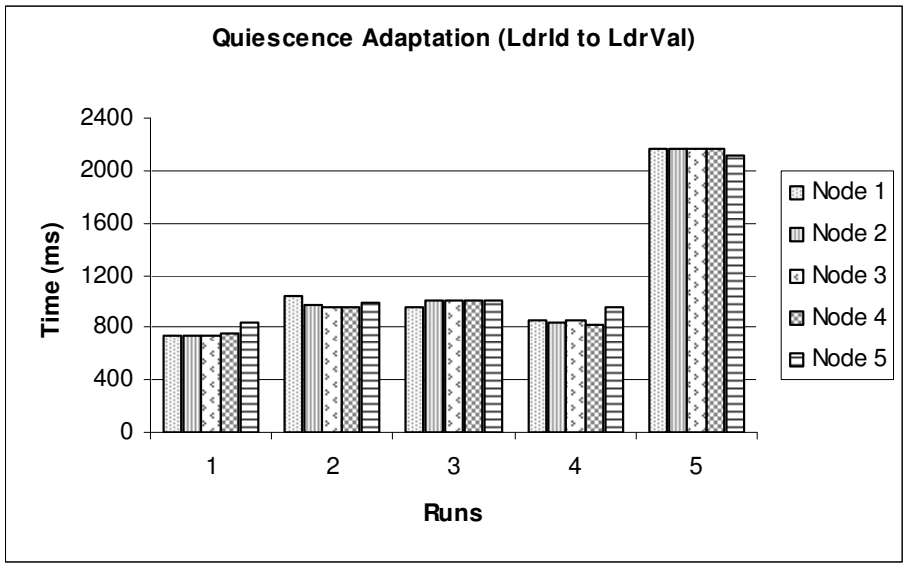


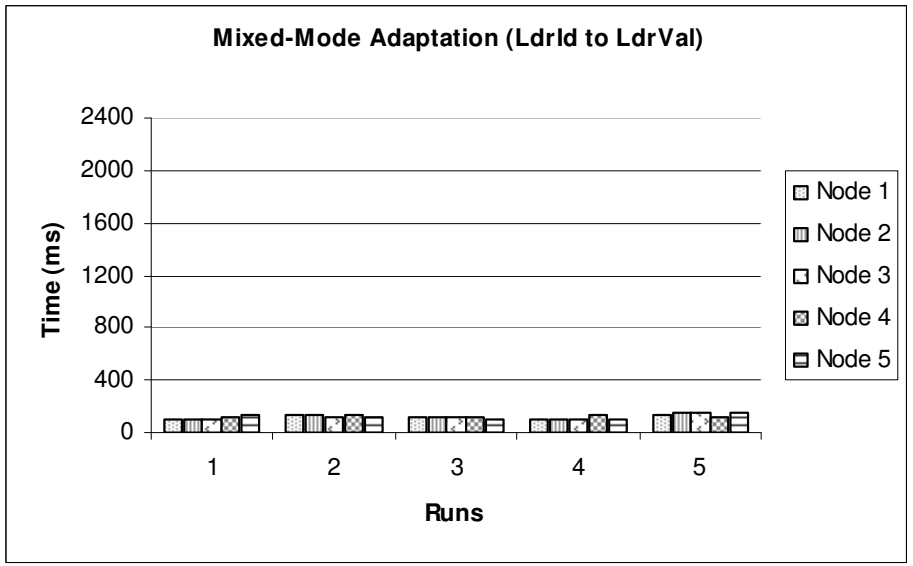


Performance of Adaptation…

- Time for adaptation – configuration two

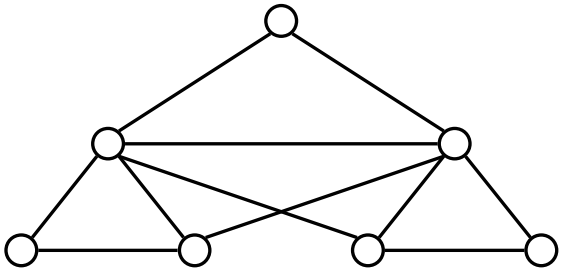


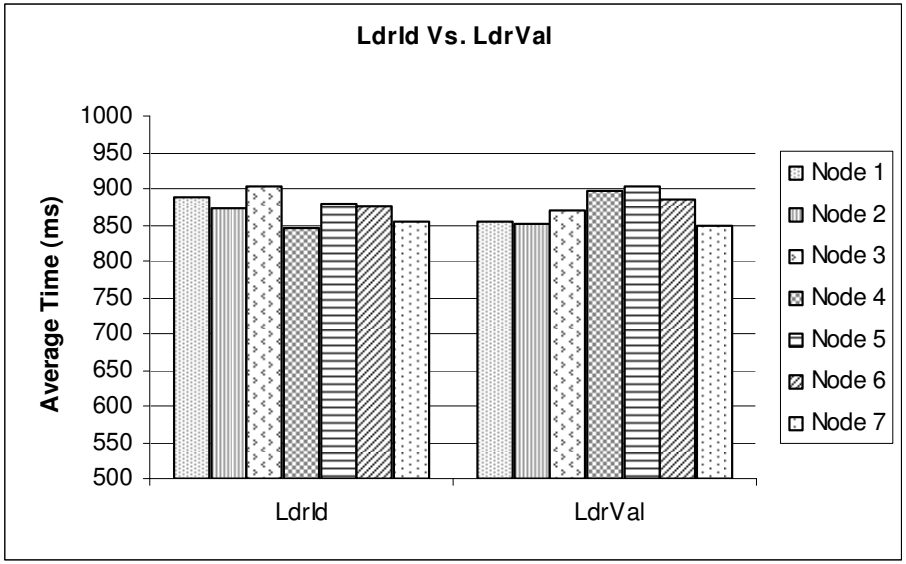


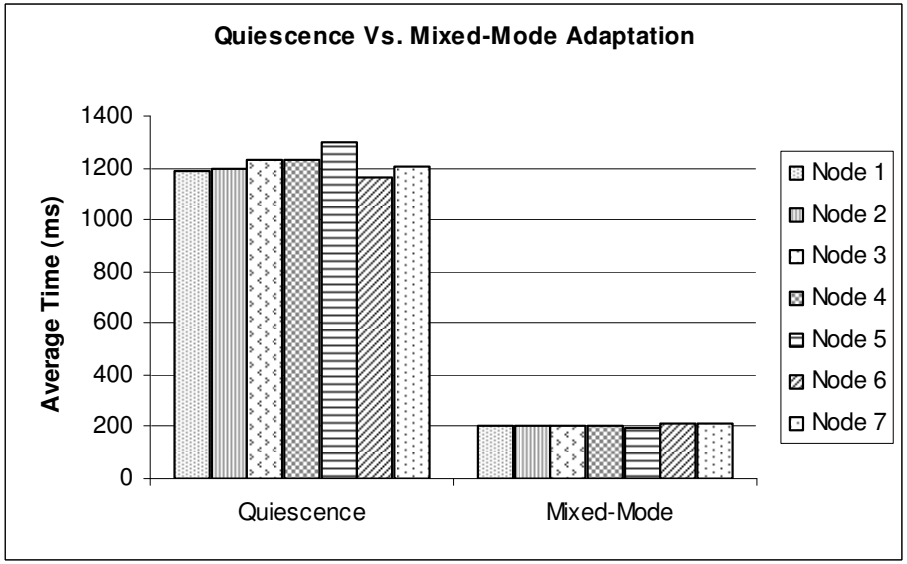


Performance of Adaptation…

- Adaptation transparency – configuration one

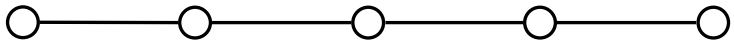


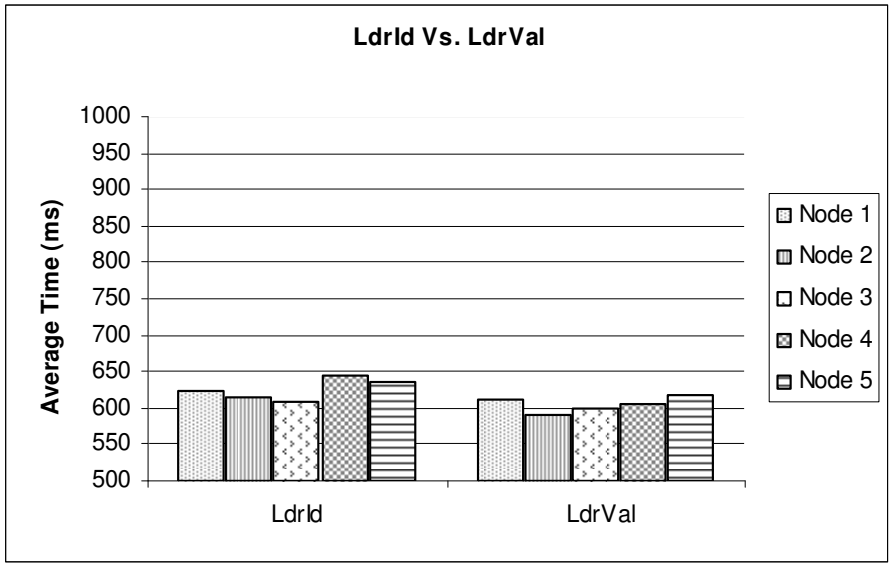


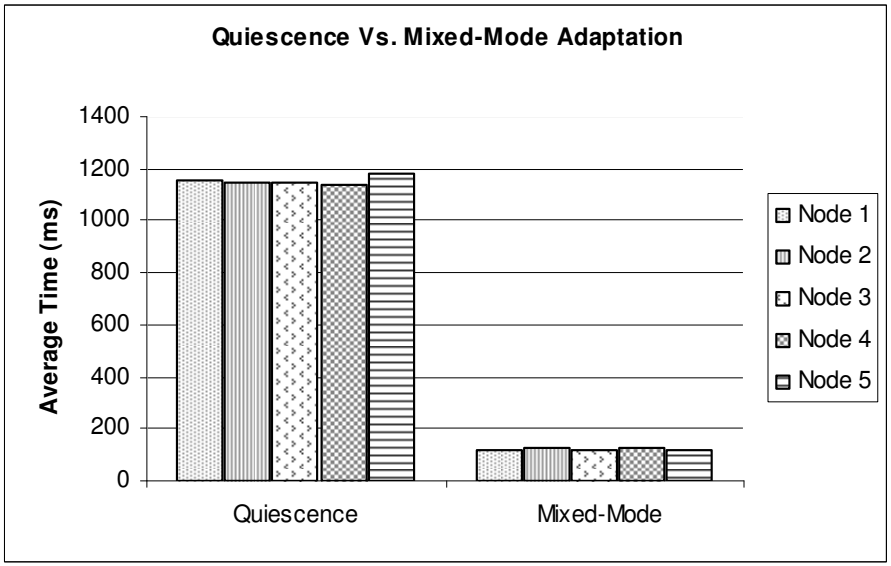


Performance of Adaptation…

- Adaptation transparency – configuration two







Conclusion

- Mixed-mode adaptation provides better performance compared to quiescence adaptation

- Challenges
  - Design & Implementation
  - Assurance

Questions / Comments ?

Thank You !!