

Adaptive Scene Synchronization for Virtual and Mixed Reality Environments

Felix G. Hamza-Lup

Jannick P. Rolland

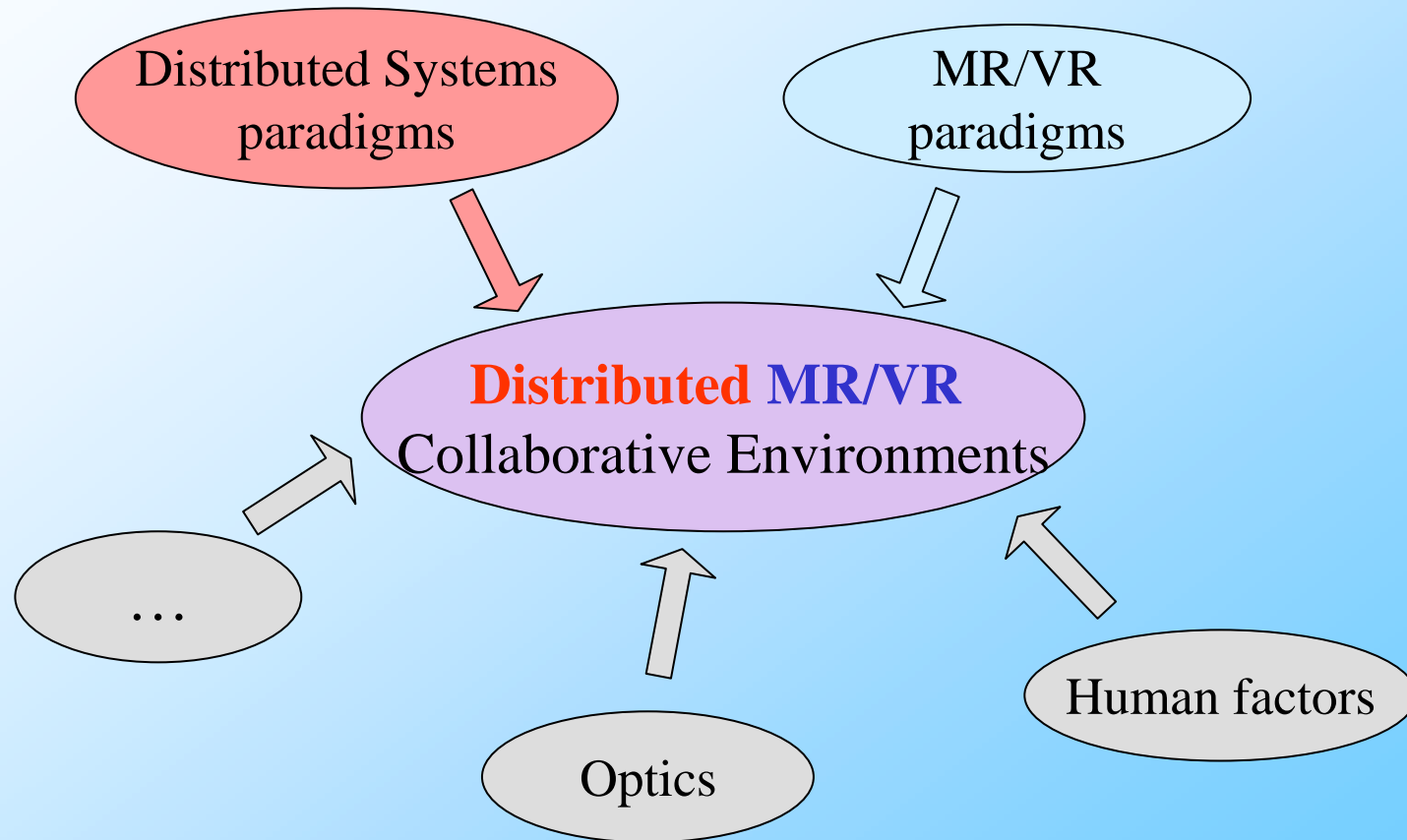
*School of Computer Science, School of Optics,
University of Central Florida*

Outline

- Distributed Collaborative Environments (DCE)
 - Mixed + Virtual Reality (MR/VR), Distributed Systems
 - Examples of MR/VR based DCE & Trend
 - Dynamic Shared State
- Adaptive Scene Synchronization Algorithm
 - Drift Value, Drift Matrix
 - Fixed vs. Adaptive Threshold
 - Quantitative Assessment
- Experimental Results
 - Scenario one – 2 nodes
 - Scenario two – 5 nodes
- Conclusions and Future Work
- Acknowledgments

Distributed Collaborative Environments

- MR/VR & DS paradigms



Distributed Collaborative Environments

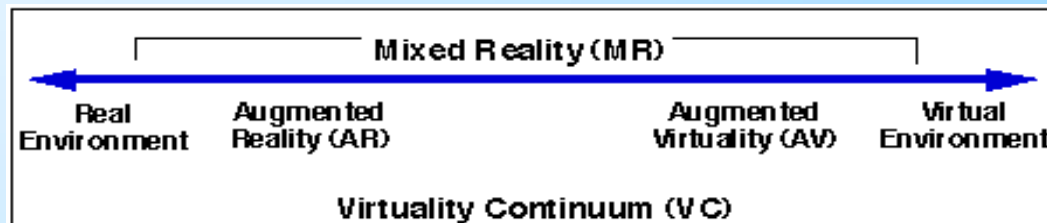
- Examples of MR/VR based DCE & Trend

- DCE application
 - Information/knowledge dissemination
 - Reduced costs, time and risks
 - Increased efficiency through team work
- Examples & Trend
 - Industry
 - Military simulations: (VR) SIMNET, NPSNET, (MR) MOUT ...
 - Entertainment: (VR) networked games, (MR) Project (ISMR'99) ...
 - Medicine: (AR) training tools (MMVR'03) ...
 - Academia: (VR) MASSIVE, DIVE, DEVA, (AR) Studierstube, Coterie...
 - Trend toward Mixed Reality (focus on AR)

Distributed Collaborative Environments

- Dynamic Shared State

“The dynamic shared state constitutes the changing information that multiple machines must maintain about the networked Virtual Environment” (“Networked Virtual Environments – design and implementation”, S. Singhal, M. Zyda)



The cause for inconsistency

- network latency (propagation, transmission, routing)
- computer system latency (rendering, buffering, etc.)

Dynamic Shared State

- Related Work

- Approaches:
 - centralized information repositories (pull/push architectures)
 - dead-reckoning algorithms (convergence & prediction)
 - frequent state regeneration (blind broadcasts, applications that do not require absolute consistency)
- Other techniques for resource management:
 - Communication protocol optimization (packet compression)
 - Visibility of data management (AOI)
 - Human perceptual limitations (LOD)
 - System Architecture

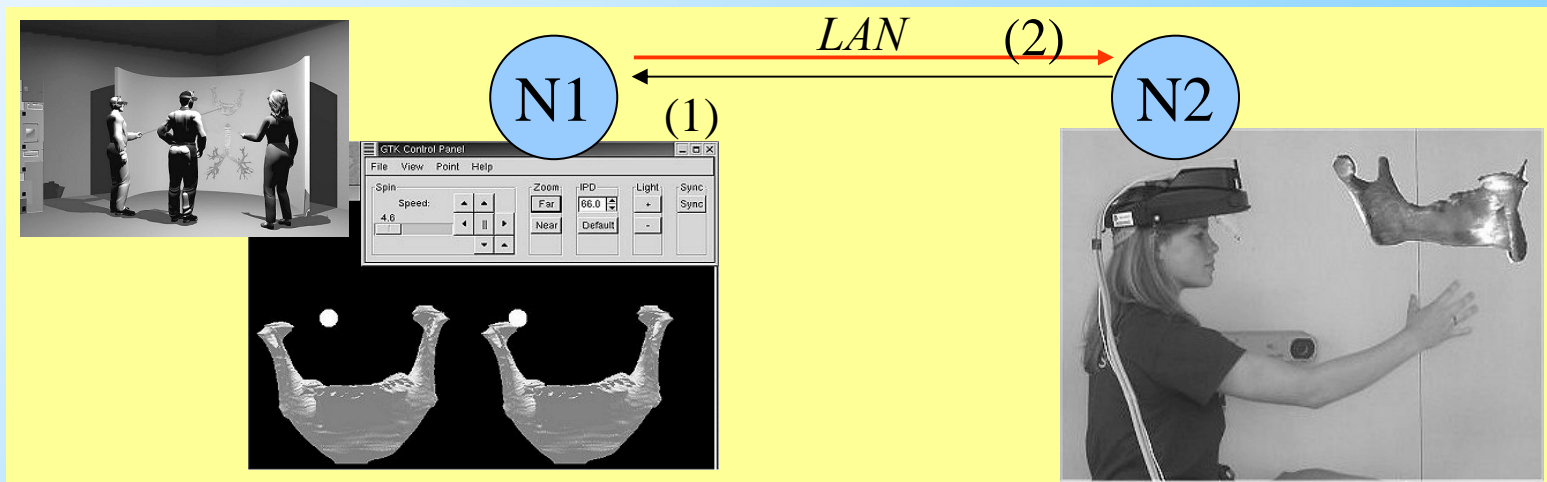
“Networked Virtual Environments – design and implementation”, S. Singhal, M. Zyda.

Outline

- Distributed Collaborative Environments (DCE)
 - Mixed + Virtual Reality (MR/VR), Distributed Systems
 - Examples of MR/VR based DCE
 - Dynamic Shared State
- Adaptive Scene Synchronization Algorithm
 - Drift Value, Drift Matrix
 - Fixed vs. Adaptive Threshold
 - Quantitative Assessment
- Experimental results
 - Scenario one – 2 nodes
 - Scenario two – 5 nodes
- Conclusion and future work
- Acknowledgments

Adaptive Scene Synchronization Algorithm

- Motivation
 - distributed algorithm for shared state maintenance that compensates for the *network latency*
 - takes into account the network infrastructure behavior
 - provides distributed computation combined with distributed system monitoring



Adaptive Scene Synchronization Algorithm

- Overview

- DCE is seen as
 - A distributed system of “n” nodes
 - Each node:
 - runs a set of threads: rendering, interaction, monitoring.
 - has access to a local library of 3D models
 - data is exchanged through software objects (each shared virtual 3D object has a software object associated)
- Two types of nodes
 - “server” nodes (produce/broadcast interaction data, software objects)
 - “client” nodes (consume interaction data, compute delay)
- Each node adjusts the local scene attributes based on
 - delay (between each producer and consumer)
 - information carried in the software objects (e.g. interaction speed)

Adaptive Scene Synchronization Algorithm

- Drift Value, Drift Matrix



- *Drift value* at (N2) - is the product between the action velocity and the network delay
- For a DCE of "N" nodes sharing "M" virtual objects
 - Velocities matrix, $S = [s_i]$, where $i \in [1, M_\tau]$
 - Delays matrix, $T = [t_j]$, where $j \in [1, N_\tau]$
- *Drift matrix*
 - $D(M_\tau N_\tau) = ST^t$

Adaptive Scene Synchronization Algorithm

Client side:

Initialization:

$T_n \leftarrow \mathbf{ComputeNodeDelay}()$

$S_n \leftarrow \mathbf{UpdateAction}();$

$D_n \leftarrow \mathbf{UpdateDrift}()$

$\mathbf{UpdateLocalScene}();$

Main:

if (trigger)

$T_n \leftarrow \mathbf{ComputeNodeDelay}()$

$D_n \leftarrow \mathbf{UpdateDrift}()$

end if

if (changedScene)

$S_n \leftarrow \mathbf{ReceiveChanges}()$

$D_n \leftarrow \mathbf{UpdateDrift}()$

end if

Server side:

for ever listen

if (newClientRequest)

$\mathbf{SendToClient}(S_n);$

end if

if (changedScene)

$\mathbf{BroadcastChanges}();$

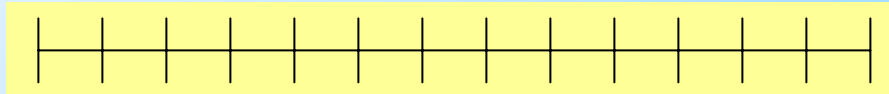
end if

end for

Delay Measurements

- Fixed vs. Adaptive Threshold

- When do we trigger the delay computation ?
- Delay measurements must be triggered whenever significant variations in the network delay appear
 - Fixed Threshold – delay measurements are triggered at regular intervals



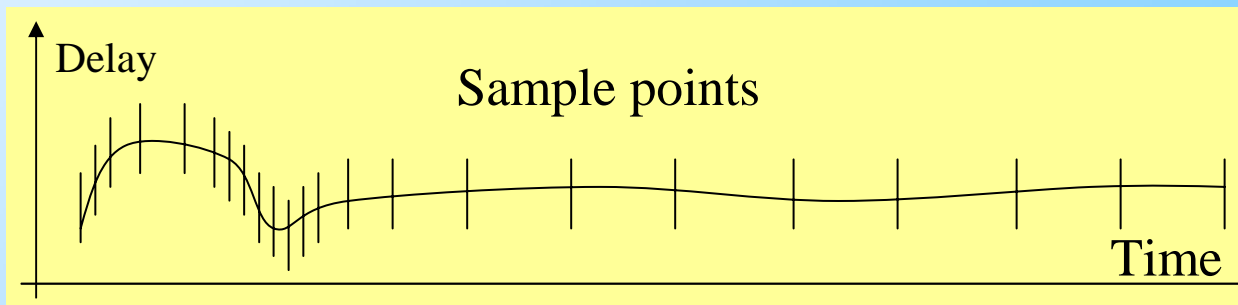
- We propose an *Adaptive Threshold* - delay measurements are triggered based on the delay history - better characterizes the network jitter and the users interaction



Delay Measurements

- Adaptive Threshold

- Let:
 - H_p the delay history
 - σ and h_{mean} be the standard dev. and the mean of H_p
 - h_0 be the most recent delay, i.e. the last entry in H_p
 - γ_0 the current frequency of delay measurements,
(expressed as the number of measurements per second)
- Adaptive approach:
 - decrease γ_0 , if $h_0 \in [h_{mean} - \sigma, h_{mean} + \sigma]$
 - increase γ_0 , if $h_0 \notin [h_{mean} - \sigma, h_{mean} + \sigma]$



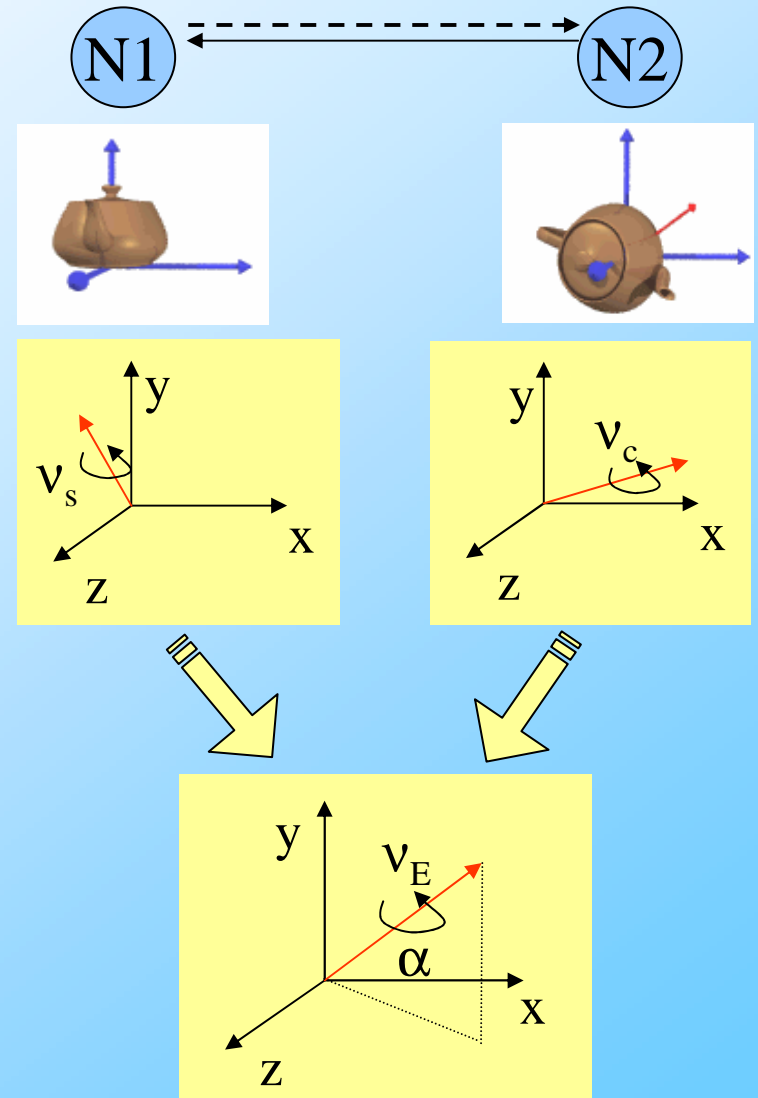
Quantitative Assessment

- Assess *orientation drift* of a shared 3D virtual object
- Let
 - q_s - rotation of an object at N1
 - q_c - rotation of the same object at N2
- Correction quaternion (q_E) - expresses the error between the actual orientation of the object and the desired orientation

$$q_s = q_E q_c$$

$$q_E = (\omega_E, v_E)$$

$$\alpha = 2 \cos^{-1}(\omega_E)$$



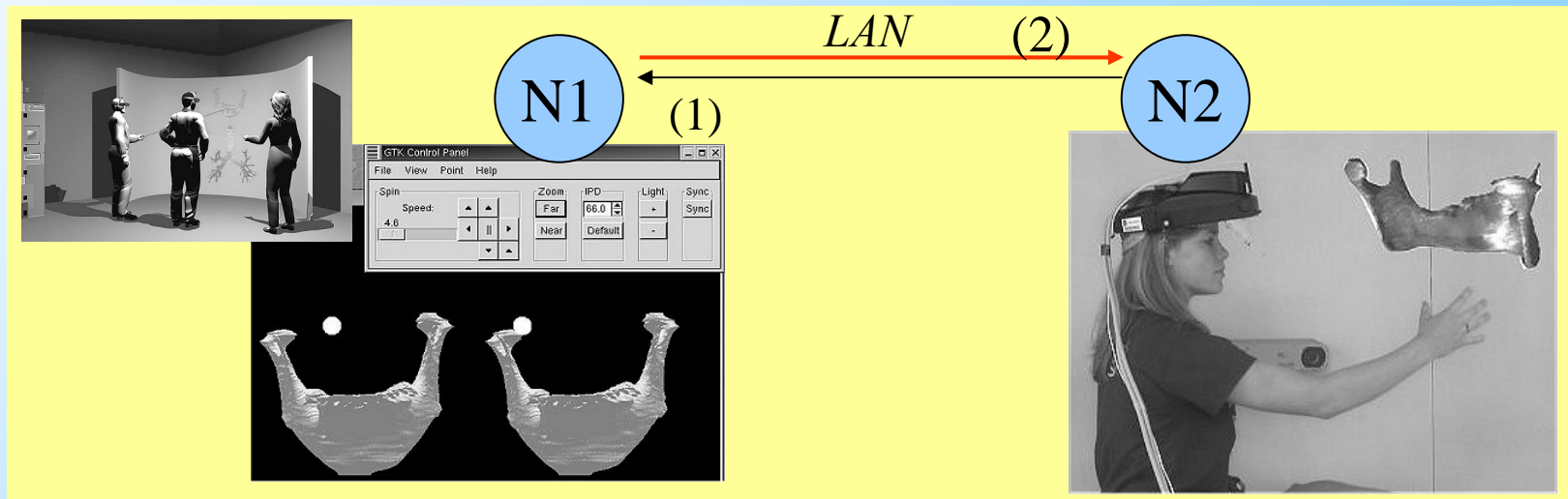
Outline

- Distributed Collaborative Environments (DCE)
 - Mixed + Virtual Reality (MR/VR), Distributed Systems
 - Examples of MR/VR based DCE
- Dynamic Shared State
 - Continuous vs. Discrete Interaction/Updates
- Adaptive Scene Synchronization Algorithm
 - Drift Value, Drift Matrix
 - Fixed vs. Adaptive Threshold
 - Quantitative Assessment
- **Experimental results**
 - **Scenario one – 2 nodes**
 - **Scenario two – 5 nodes**
- Conclusion and future work
- Acknowledgments

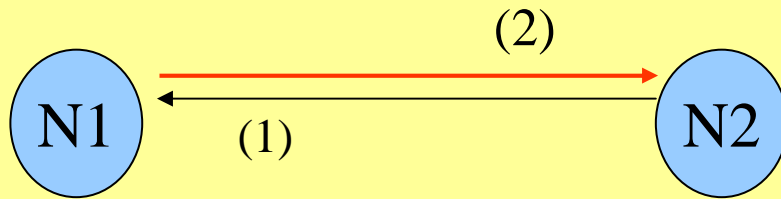
Experimental Results

- Prototype

- Distributed Artificial Reality Environment (DARE)
 - set of OO libraries for 3D rendering, communication, node monitoring, assessment (<http://odalab.creol.ucf.edu/dare>)
- User interacts through a GUI by applying a set of consecutive actions (rotations) on the object



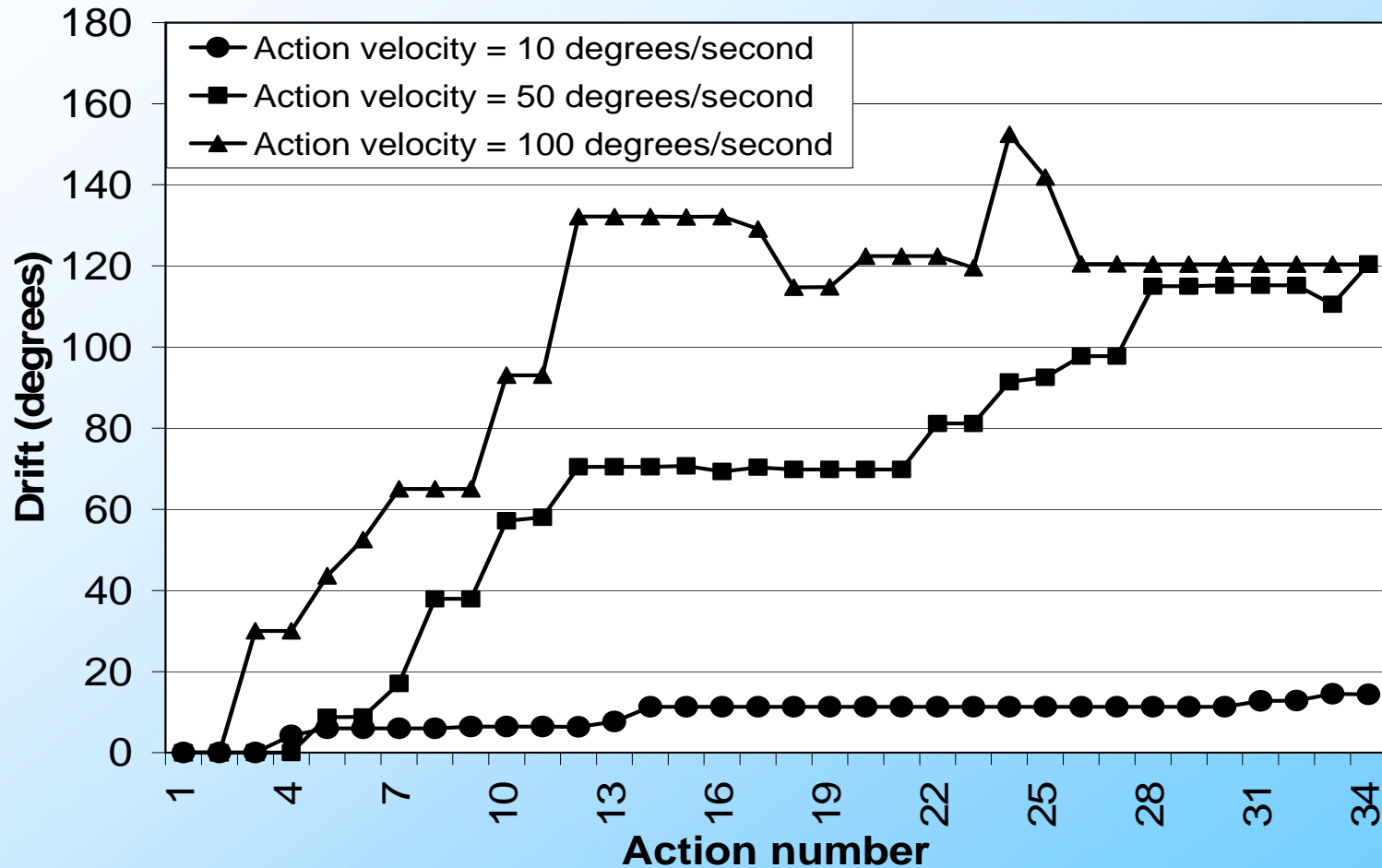
Experimental Results - Scenario 1



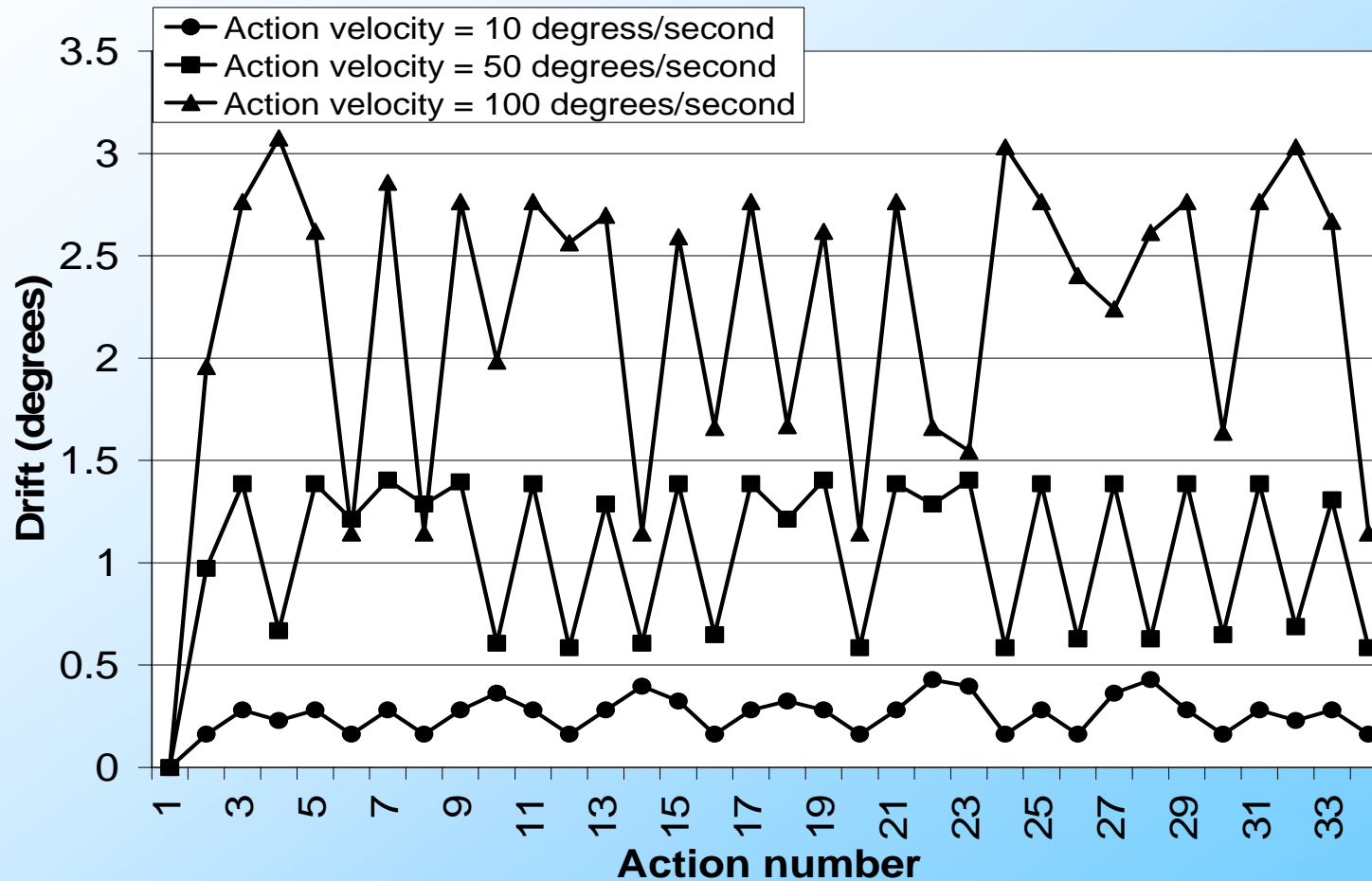
User at N1 rotates the shared object around axes with different velocities (e.g. 10, 50, 100 degrees/second)

- (1) N2 computes the inter-node delay.
- (2) N1 broadcasts updates as the user at N1 interacts with the object.

Angular Drift at N2, Synch. OFF

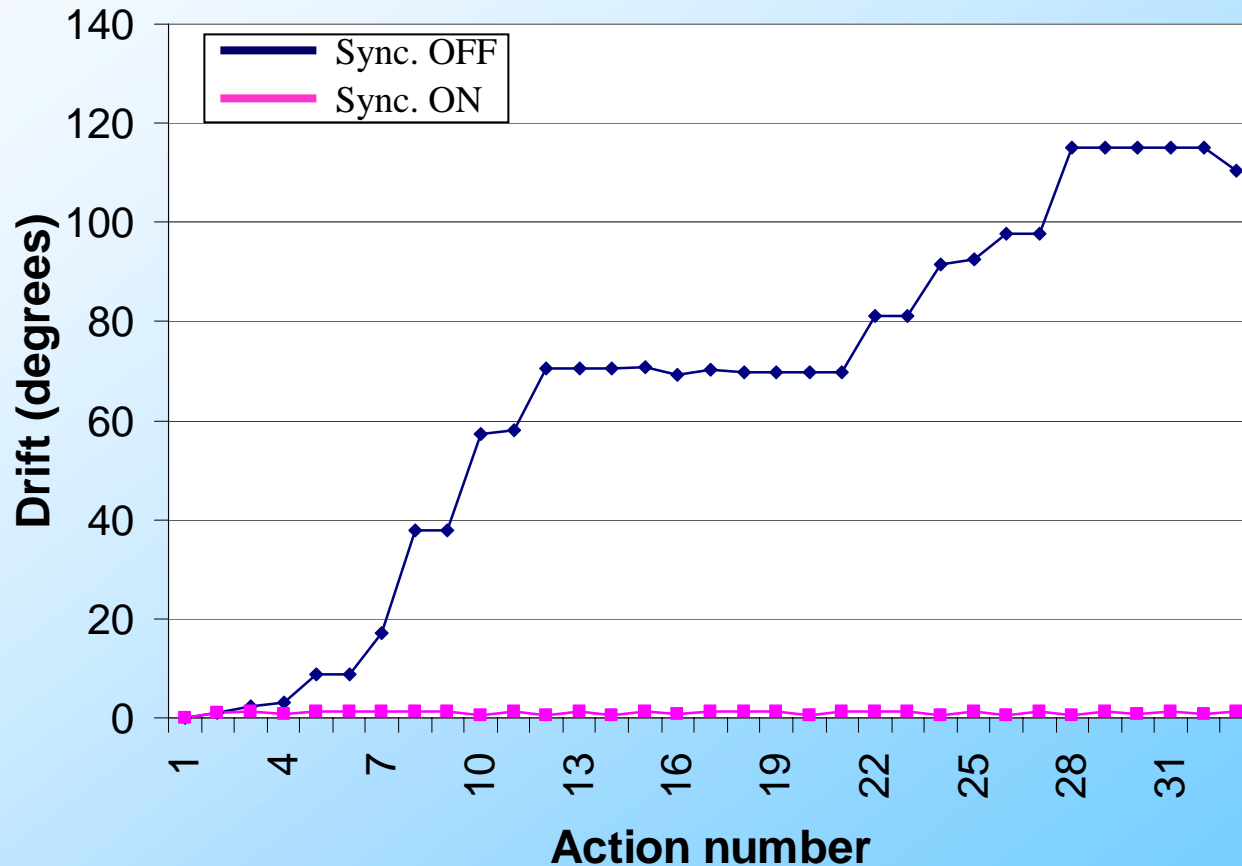


Angular Drift at N2, Synch. ON



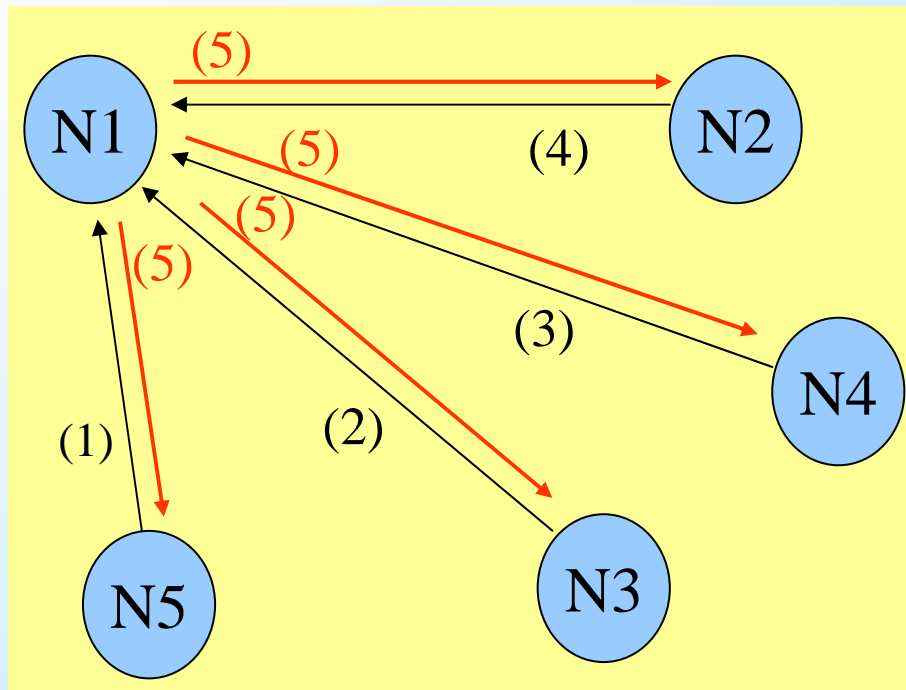
Comparison Synchron. ON/OFF

Drift on N2, action speed 50 degrees/sec



Experimental Results – Scenario 2

- Investigation of Scalability



(1) node 5 joins and uses the *delay probe* to compute the latency between him and the server.

(2), (3), (4) Same

(5) The server (N1) broadcasts data to all participants (as they join the environment)

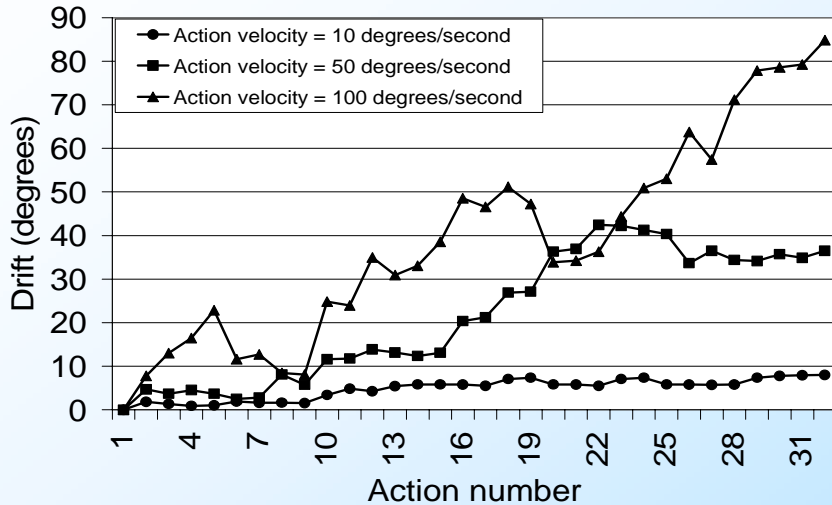
Experimental Results – Scenario 2

- Hardware

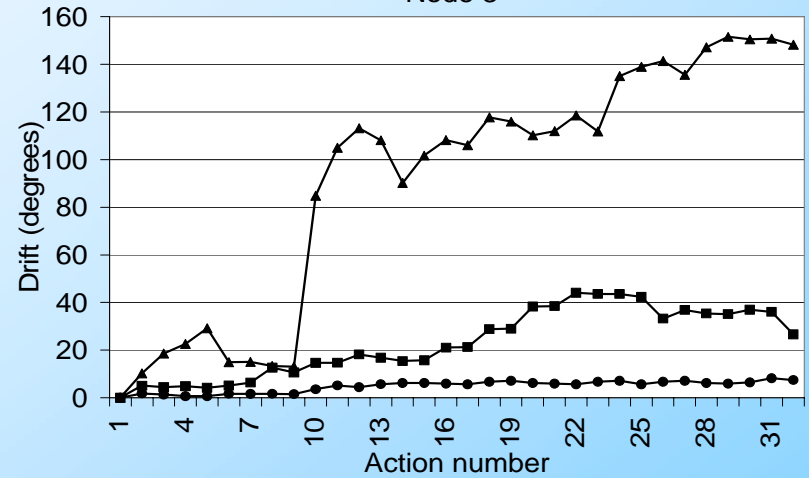
Node no.	Arch	CPU (GHz)	RAM (MB)	Video card
1	Desktop	1.5 AMD	1024	4 Ti4600
2	Desktop	1 P3	1024	2 Mx
3	Desktop	1.7 P4	512	4 Mx 440
4	Desktop	1.7 AMD	1024	4 Ti4600
5	Laptop	2 P4	1024	4 Go440

Drift between client nodes and N1 – Synch. OFF

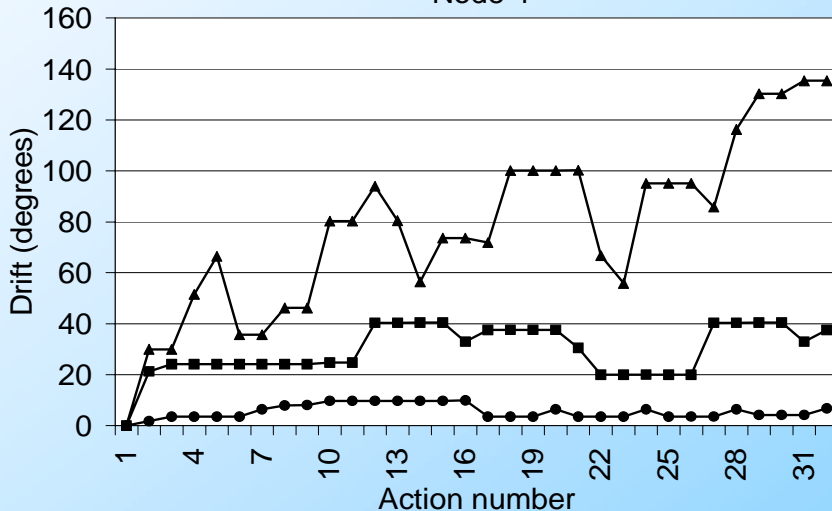
Node 2



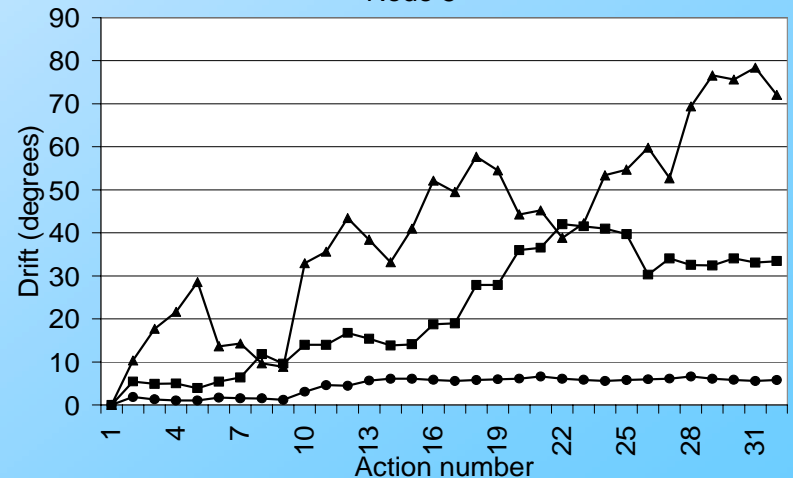
Node 3



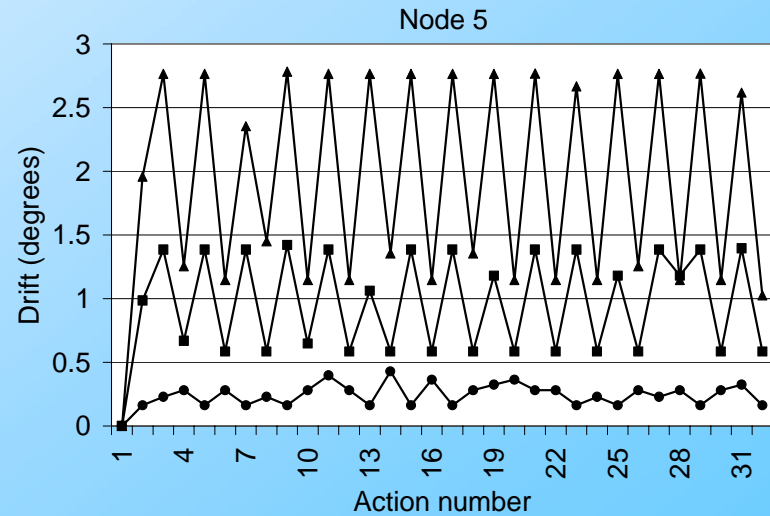
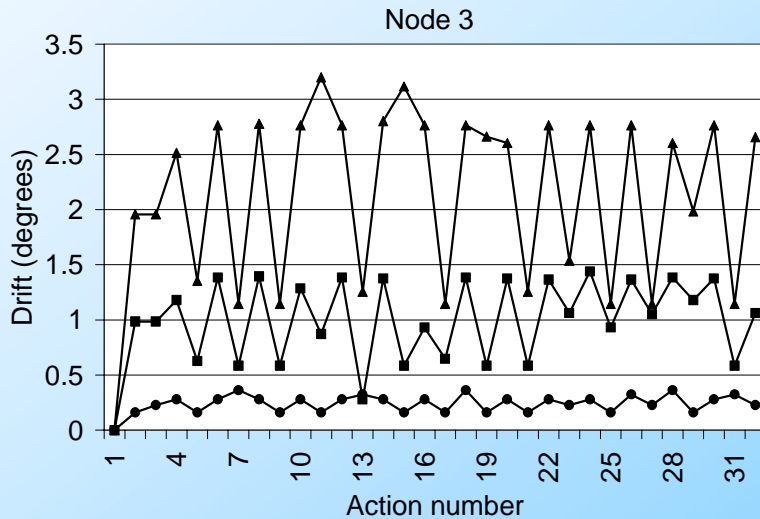
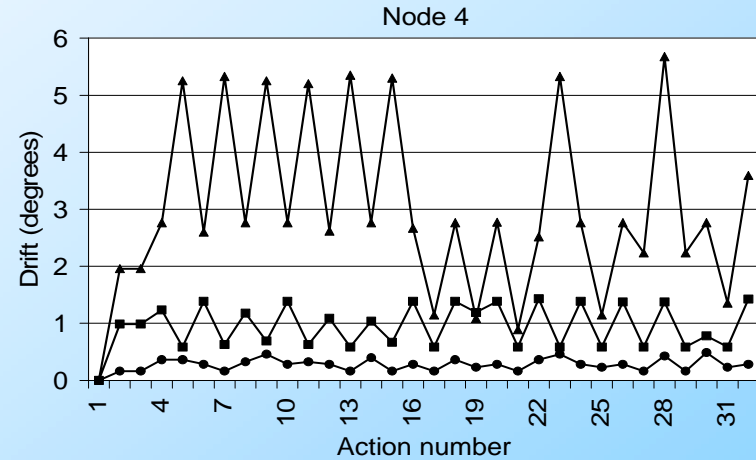
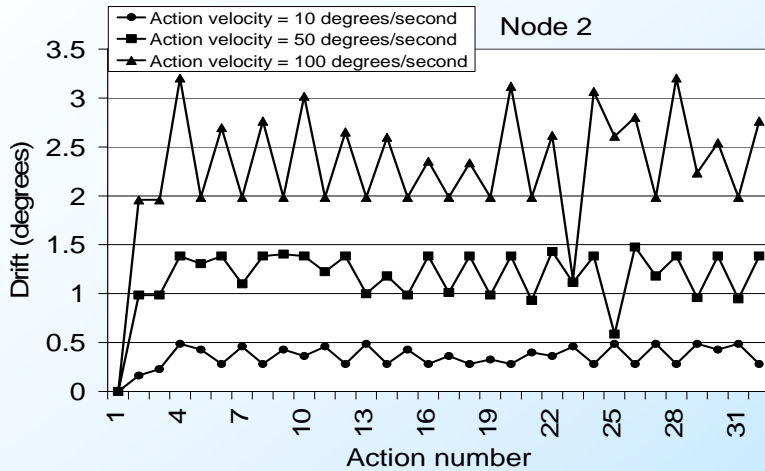
Node 4



Node 5

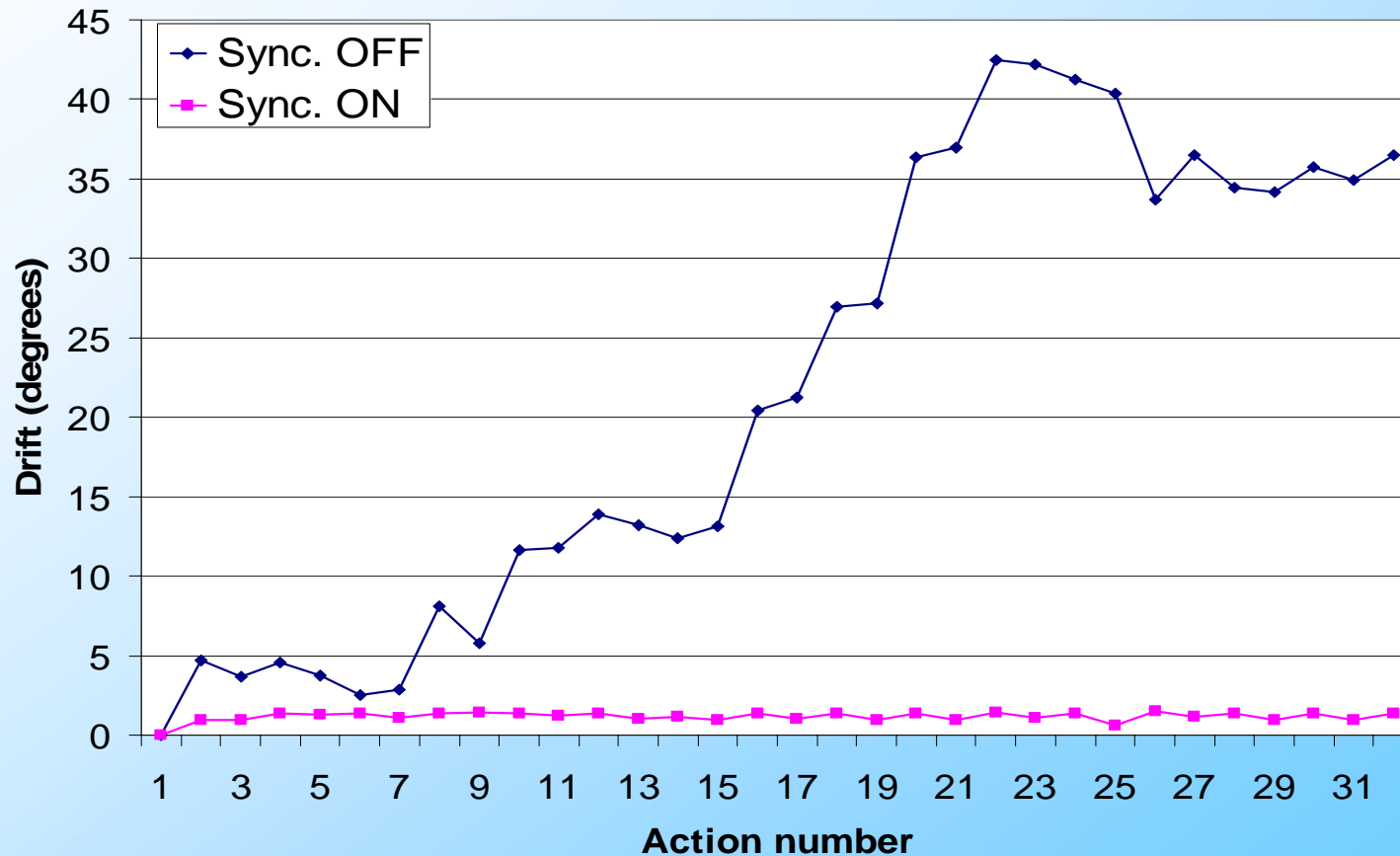


Drift between client nodes and N1 – Synch. ON



Comparison Synch. ON/OFF

Drift on N2 at action velocity 50 degrees/sec



Summary of Results

- Maintains a low and constant drift level
 - 100 degrees/sec, synch OFF, after 34 actions $\Rightarrow \sigma_{\text{drift}} = 22.59$
 - 100 degrees/sec, synch. ON, after 34 actions $\Rightarrow \sigma_{\text{drift}} = 0.48$
- Scalability regarding the number of nodes (ψ average drift)
 - $\Psi_n = n \psi_1$ linear increase, low scalability
 - $\Psi_n \approx \psi_1$ i.e. good scalability
 - experimental results:
 - $\psi_1 = 2.4$ (2 nodes setup)
 - $\psi_4 = 2.9$ (4 nodes setup) $\Rightarrow \Psi_4 \approx \psi_1$

Conclusions and Future Work

- Distributed algorithm for dynamic shared state maintenance
 - takes into account network latency
 - reduces intrusiveness through an adaptive threshold
 - decentralized delay and drift computation approach
- Extend the system infrastructure to multiple interacting nodes:

Hamza-Lup, F. J. Rolland, C. Hughes: “*Hybrid Nodes with Sensors - Architecture for Interactive Distributed Mixed and Virtual Reality Environments*” in press, SCI 2004, July 18-21 Orlando Florida.

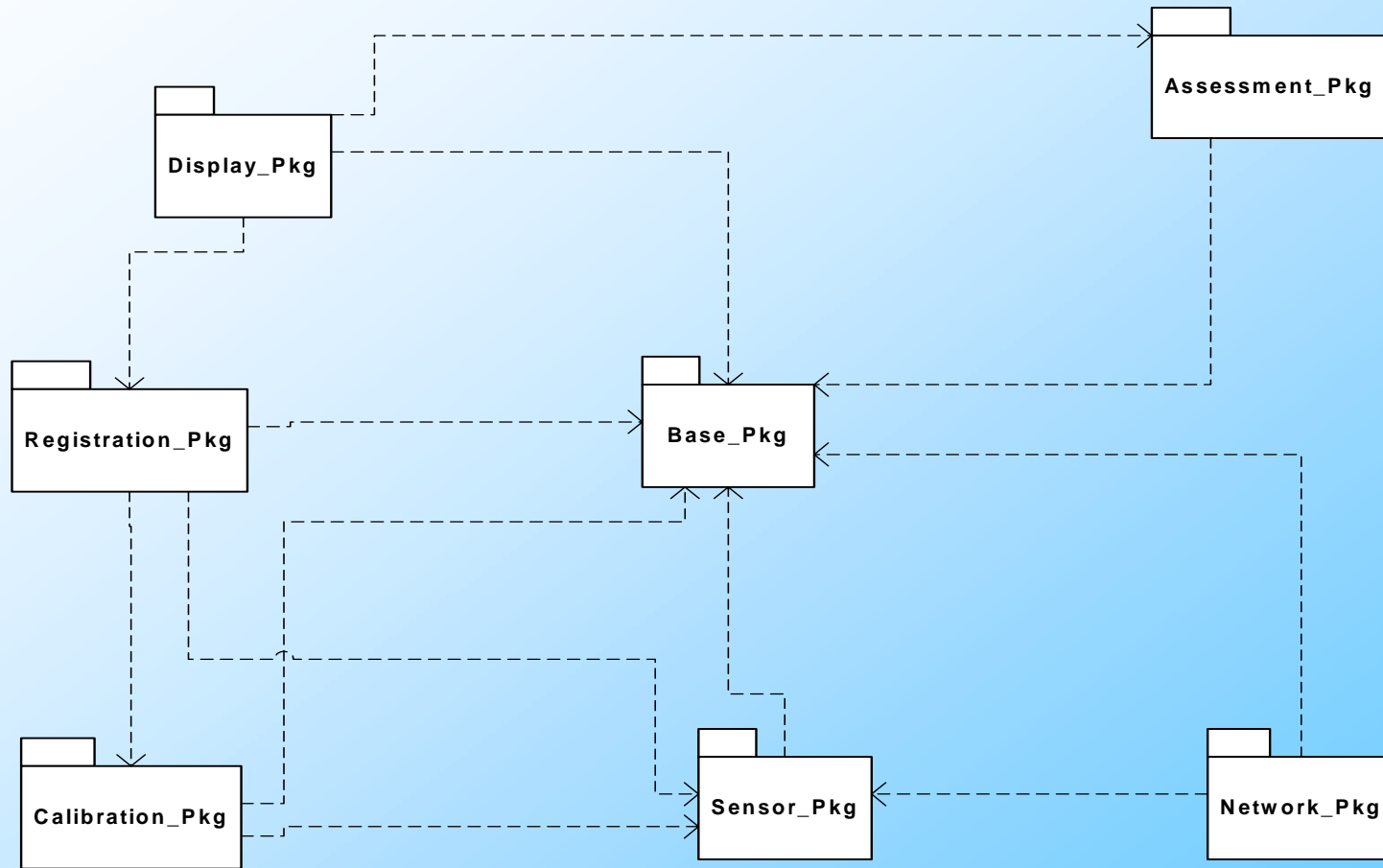
Acknowledgments

- NSF/ITR: IIS-00-820-16
- The Link Foundation
- ODALab team
- Prof. Charles Hughes

Thank you.

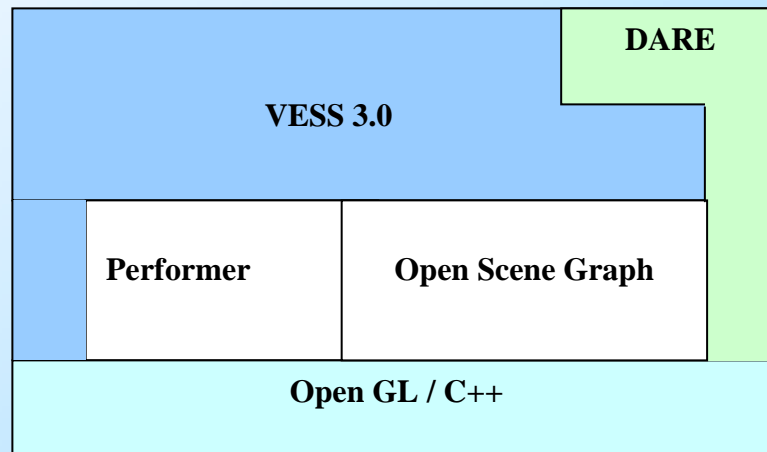
Questions please ...

DARE packages



Distributed AR Environment

- DARE and other VR libraries



Continuous vs. Discrete

- Distributed System
 - Continuous data flow
 - Intermittent (“Discrete” data flow)
- Interaction in MR/VR
 - Continuous (e.g. a sensor feeding data)
 - Discrete (e.g. a user interacting through a GUI)
- System Updates
 - Continuous
 - Discrete

Continuous vs. Discrete

Interaction Update	Continuous	Discrete
Continuous	(-) scalability (+) consistency	(-) scalability (+) consistency
Discrete	(+) scalability (-) consistency	(+) scalability (+) consistency