Performance Evaluation of Mobile-Agent Middleware

Marios Dikaiakos

Department of Computer Science
University of Cyprus
Nicosia, Cyprus

http://www.cs.ucy.ac.cy/mdd/

Performance Analysis and Distributed Computing
Dagstuhl Seminar No. 02341, August 19-23, 2002
Schloss Dagstuhl, Wadern, Germany

Acknowledgments

- Collaborators:
  - prof. G. Samaras (U. of Cyprus)
  - M. Kyriakou (U. of Cyprus)
  - C. Spyrou (U. Cyprus)

- Funding: Research Foundation of Cyprus (PENEK 23/2000)

The context…

- Explosion of systems and services available on Internet/Web.
- Remarkable increase in size and pervasiveness of computer networks.
- Demand for personalized, instant, context-aware, ubiquitous services.
- Support of these trends by an evolving *communication infrastructure*.
- What about the necessary *computational infrastructure ?*
  - Scalability
  - Reconfigurability and Extensibility
  - Adaptability
  - Physical Mobility
  - Fault-tolerance

Mobile Agents

- *Programs that can migrate from host to host in a wide-area network at times and to places of their own choice. The state of the running program is saved, transported to the new host and restored, allowing the program to continue where it left off.*
- Distinctive characteristics:
  - Autonomous migration of code and state.
  - Location is a first-class element, exposed at the programming level.
- Java-based Mobile Agents:
  - Java objects running in Java-based execution environments, taking advantage of Java's distributed computing features to achieve code mobility.
- JAVA-based Mobile Agent Middleware:
  - Environments supporting MA execution, management of local-resource access, and programming via higher level API's.

Why Mobile Agents?

- *Autonomy*
  - Support for disconnected operations and weak connectivity.
  - A new design style for distributed applications.
- *Enhanced Flexibility*
  - Extend the client-server model of distributed computing.
  - Enabling client-systems to customize their access to remote resources.
  - Extend clients dynamically by code coming from remote sites.
- *Performance*
  - Reduce bandwidth consumption in network management systems.
  - Efficient distributed database access over the Web.
  - Distributed information retrieval and filtering.
- *Open Issues*
  - Security; Interoperability with existing middleware and protocols; Lack of wide acceptance and many real applications; Lack of rigorous evaluations and comparisons.

Motivation and Summary

- Goals:
  - Investigation.
  - Comparison.
  - Discovery.
  - Prediction.
- Summary:
  - A framework to evaluate MA middleware performance quantitatively.
  - An implementation of this framework as a hierarchy of benchmarks.
  - Reference implementations on commercial MA platforms.
  - Experiments.

Performance Analysis of Software Systems

- Typical approaches: Experimentation, simulation, modelling,…
- For complex software systems some modelling is required:
  - A hierarchical structure of interacting modules (subsystems and objects).
  - Each module is assigned:
    - A performance model.
    - A description of the underlying architecture and workload.
  - Model development usually "top-down."
  - Experimentation and/or simulation at various levels to specify values of modelling parameters.

The Case of Mobile-Agent Middleware

- Quantitative evaluation of mobile-agent-based distributed systems is even harder:
  - Absence of global time, control and state information.
  - Heterogeneity/complexity of platforms: difficult to describe performance properties via small sets of metrics.
  - Variety of distributed computing (software) models.
  - Diversity of operations found in distributed applications: hard to construct simple and portable benchmarks.
  - Agility of system configuration: hard to provide concise representation of system resources.
  - Issues affecting performance of JAVA.

Our Proposal: A Hierarchical Approach

…inspired by the structure of MA-applications, which is determined by:

- The MA middleware upon which an application is implemented: differences in functionality, API, performance, underlying implementation details.
- The higher-level abstractions representing the design choices made at the development of a particular application.

Our Proposal: A Hierarchical Approach

- *“Bottom-up”* instead of *“top-down:”*
  - Isolate performance properties of MA middleware as measurements of platform-independent metrics.
  - Investigate the performance of “popular” program structures commonly used in MA applications.
  - Enrich the functionality of “popular” program structures and investigate the interplay of the MA technology with other systems (databases, information retrieval, networking infrastructures, etc.).
- Our abstractions:
  - Basic Elements
  - Application Frameworks

Basic Elements

Set of basic abstractions representing the fundamental functionalities commonly found in MA platforms.

- Agents: State, Implementation (code), Interface, Identifier, etc.
- Places (environment where agents execute): Virtual Machine, Network Connection, Resources, Services available
- Behaviors (within and between places): Creation, Dispatch, Transfer, Communication via messages and agents, Multicasting, Synchronization.

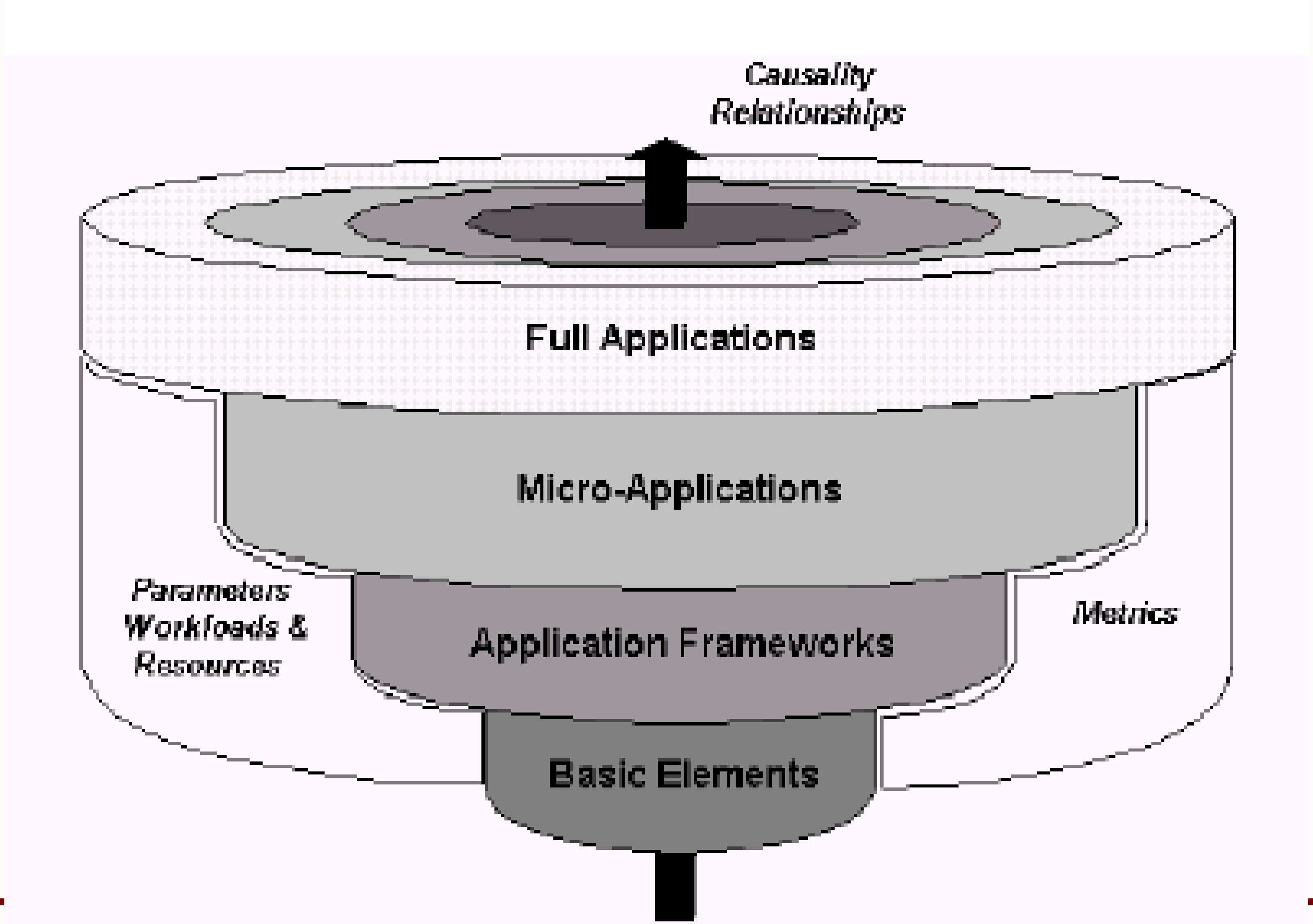
Application Frameworks

- Software frameworks (OO)
  - Ways of structuring generic solutions to a common problem by providing the structure of a program but no application-specific details.
- Application Frameworks (MA)
  - Define scenarios common to various problems of MA application design, and are defined in terms of places participating in a scenario, agents placed at or moving between these places, and interactions of agents and places.
  - Distributed-computing models
  - MA Design Patterns

Application Frameworks (ctd')

- Client-Server model and extensions:
  - Client-Server
  - Client-Agent-Server
  - Client-Intercept-Server
- Roaming (multi-hop) MA
- Master-Slave
- Agent Design Patterns:
  - Forwarding
  - Meeting

A Hierarchical Framework



Implementation through Benchmarks

- **Micro-benchmarks:** short loops designed to isolate and measure performance properties of basic elements.
- **Micro-kernels:** short, synthetic codes designed to measure performance properties of application frameworks.
- **Micro-applications:** instantiations of micro-kernels with particular functionalities and representative workloads.
- **Parameters:** platform, workload, resources
- **Metrics:** total time, average time, peak rate, sustained rate
- **Platforms:** Concordia, Aglets, Voyager, Win95, WinNT

Micro-benchmarks

- Key software components:
  - Mobile Agents to materialize components of C/S, C/A/S, etc.
  - Messenger Agents for flexible communication.
  - Messaging for efficient communication and synchronization.
- Metrics:
  - Total and average runtime.
  - Peak and sustained rates.
- Parameterized by:
  - Number of iterations.
  - Configuration of places.
  - Configuration of channels.

Micro-benchmark Suite

- Agent Creation and Launching:

  CL measures the overhead of local agent-creation.

  CR measures the overhead of remote agent-creation.

  AD measures the overhead of agent transportation.

- Messaging:

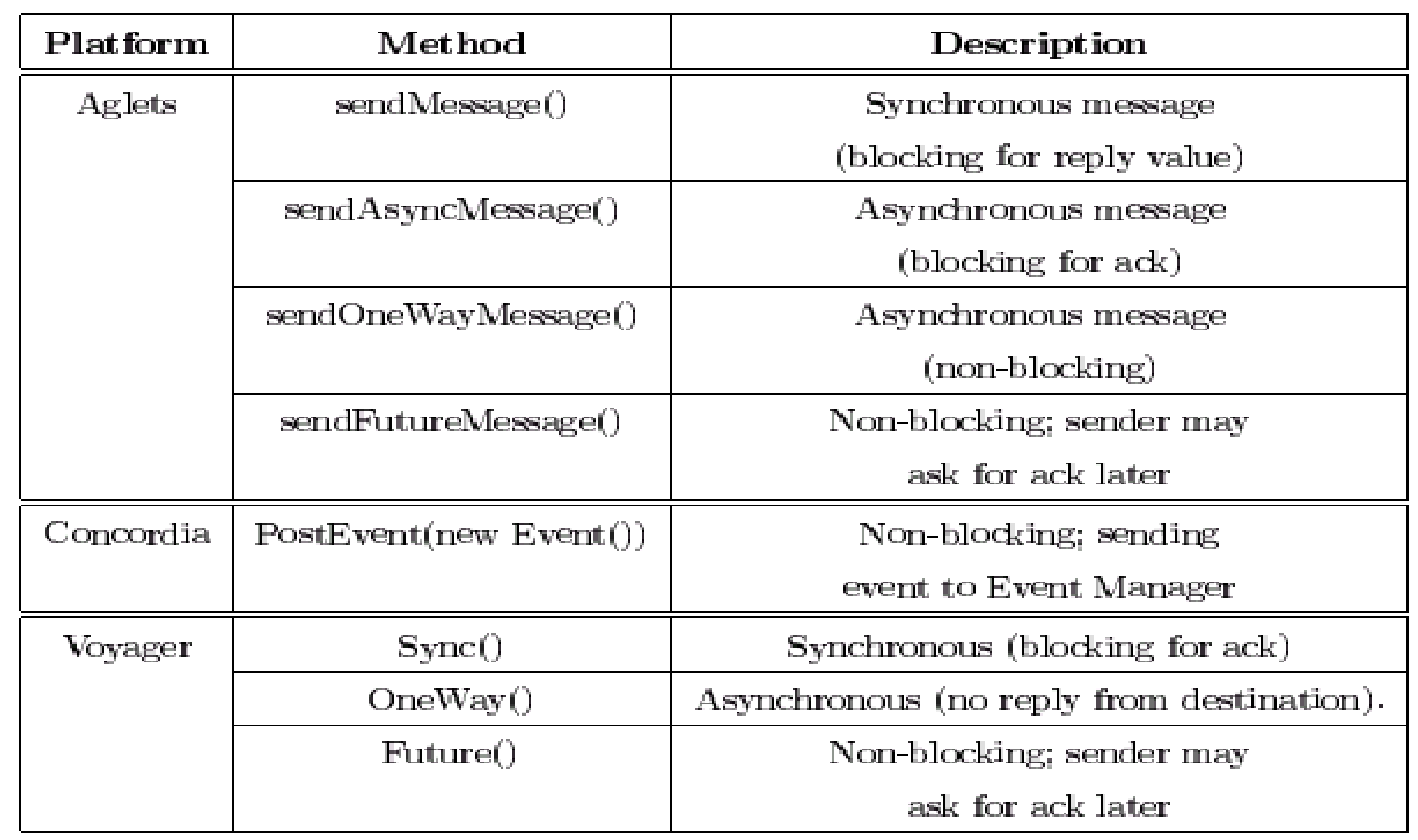
  MSG-1W point-to-point, non-blocking messaging.

  MSG-2W point-to-point, non-blocking with asynchronous ack.

  SYNCH point-to-point, blocking (ping-pong).

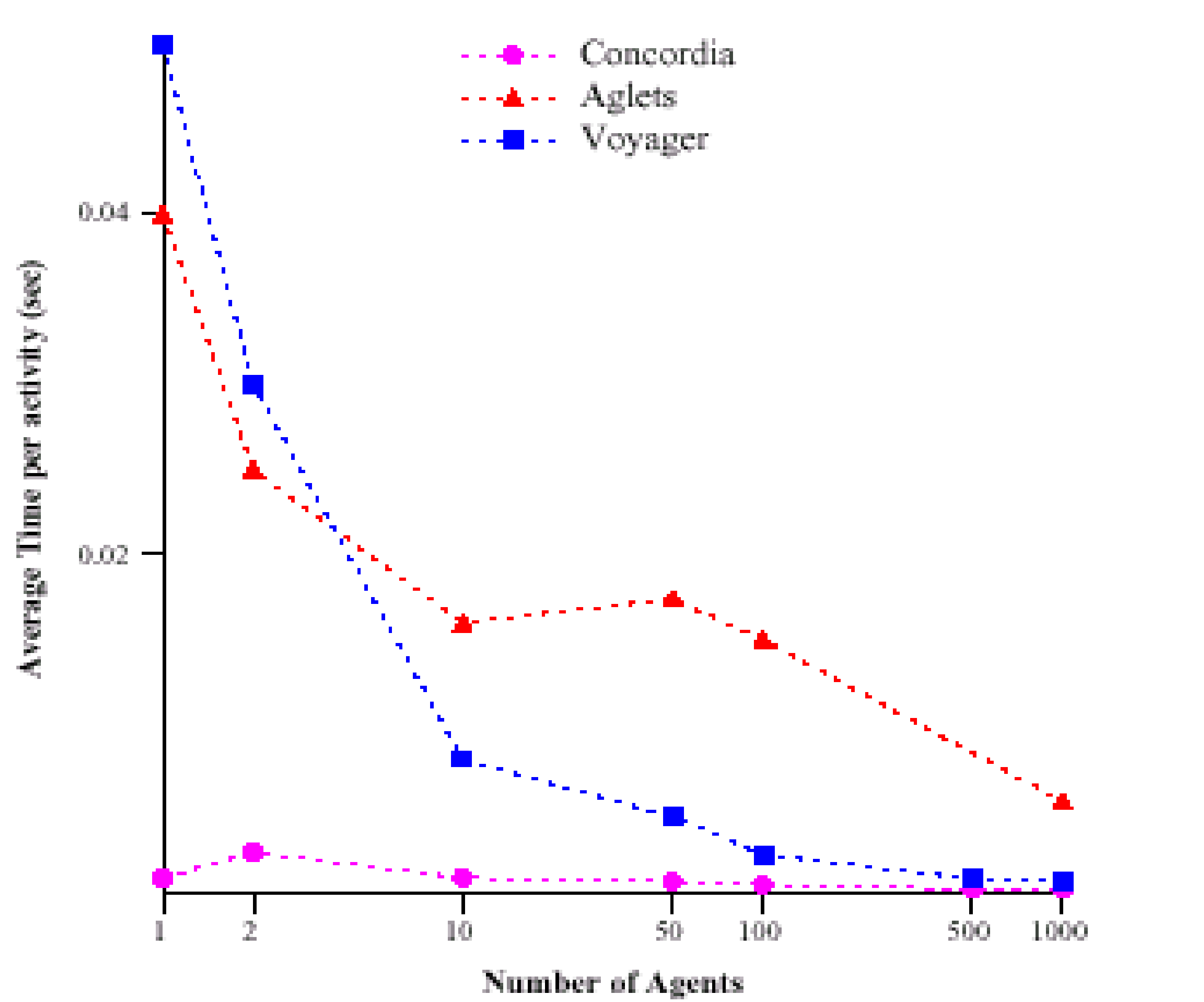
  MSG-MA point-to-point, blocking with messenger agent.

Messaging between MA's



Platform	Method	Description
Aglets	sendMessage()	Synchronous message (blocking for reply value)
	sendAsyncMessage()	Asynchronous message (blocking for ack)
	sendOneWayMessage()	Asynchronous message (non-blocking)
	sendFutureMessage()	Non-blocking; sender may ask for ack later
Concordia	PostEvent(new Event())	Non-blocking; sending event to Event Manager
Voyager	Sync()	Synchronous (blocking for ack)
	OneWay()	Asynchronous (no reply from destination).
	Future()	Non-blocking; sender may ask for ack later

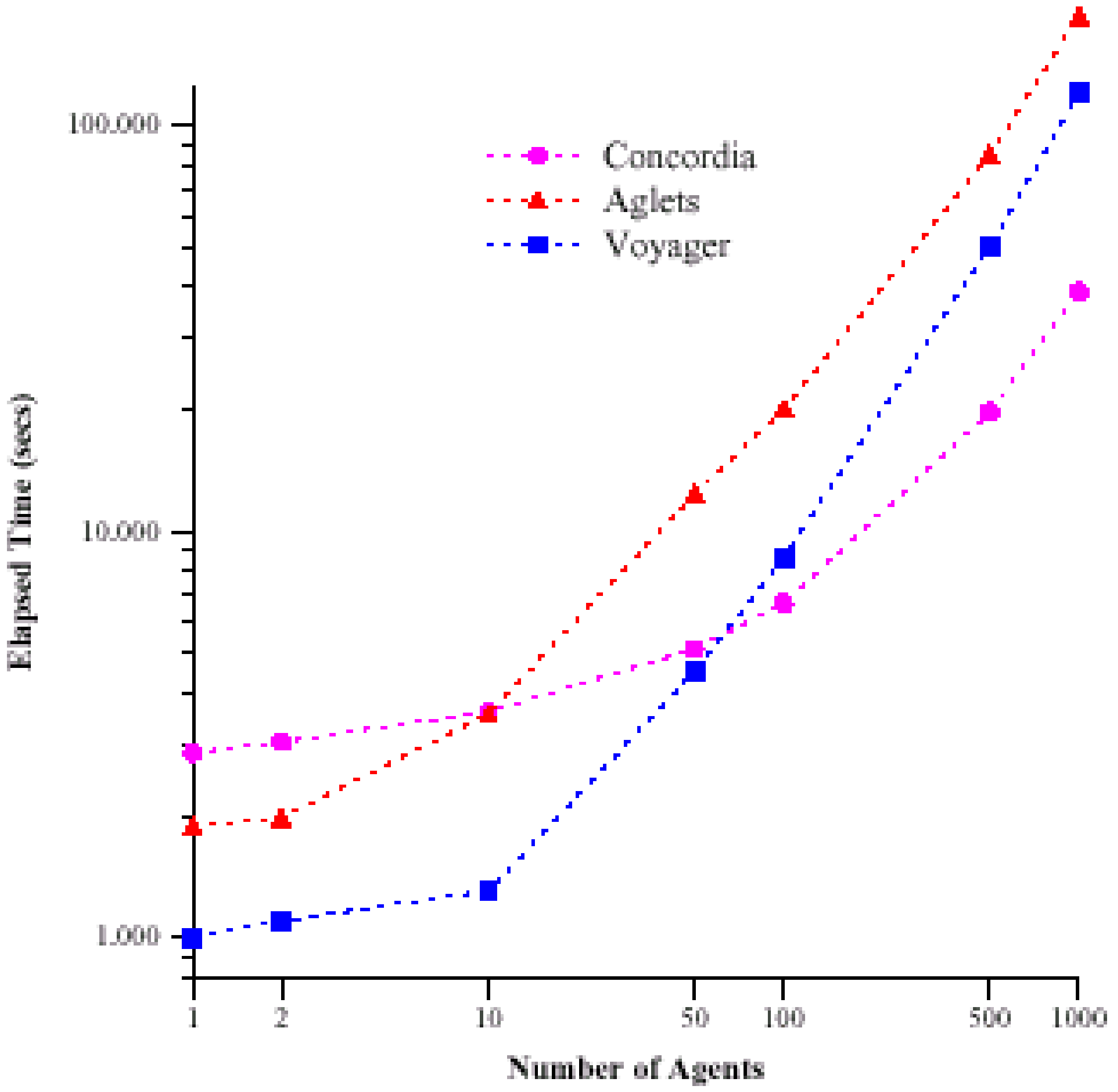
CL: *CreationLocal* experiments



Average Time:[CL] Benchmark (log scale)

- Caching
- Memory management

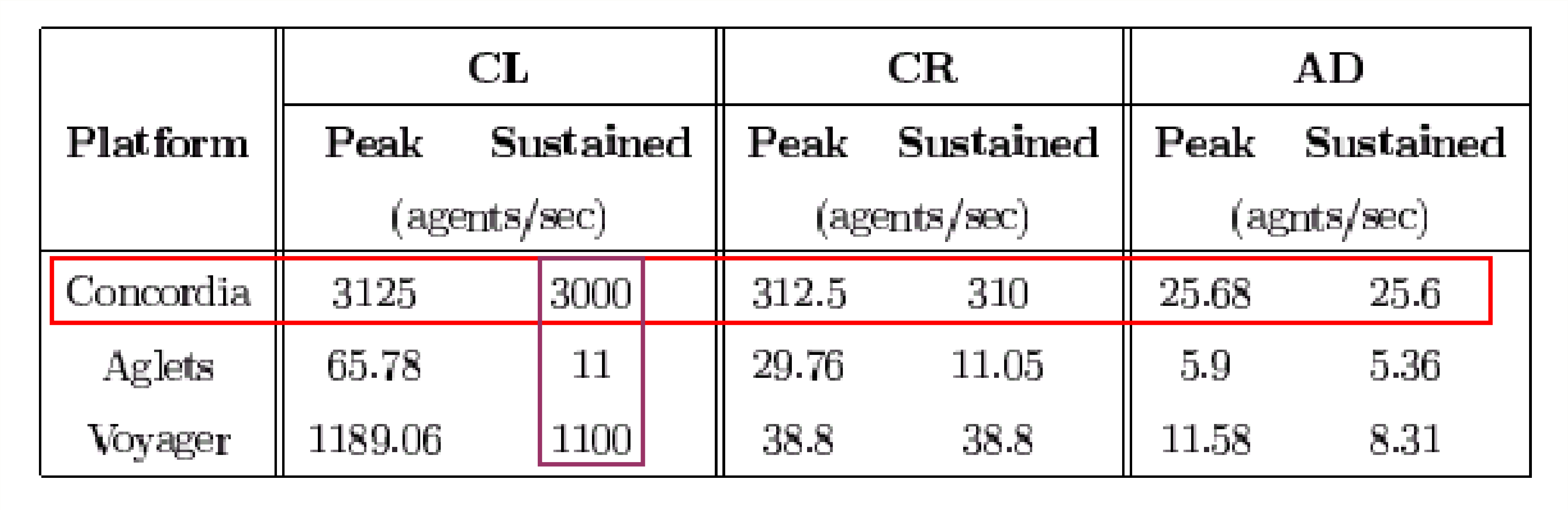
AD: *AgentDispatch* experiments



[AD] Benchmark (log-log scale)

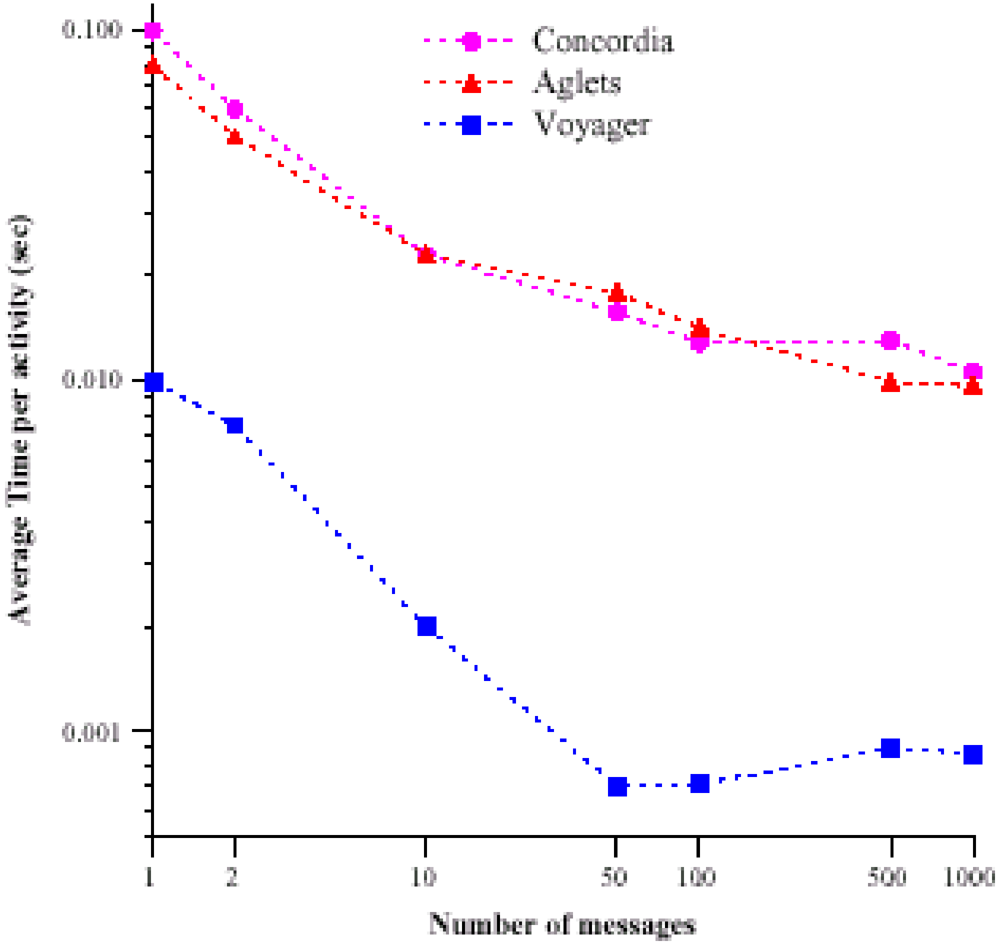
- Aglets transportation based on ATP. Carries all reachable objects.
- Concordia agent transportation based on RMI. Carries and caches objects on a need-to-use basis.
- Voyager agent transportation uses agent-serialization. Agent and all its non-transient parts copied to new location.

CL, CR and AD: Peak and Sustained Rates

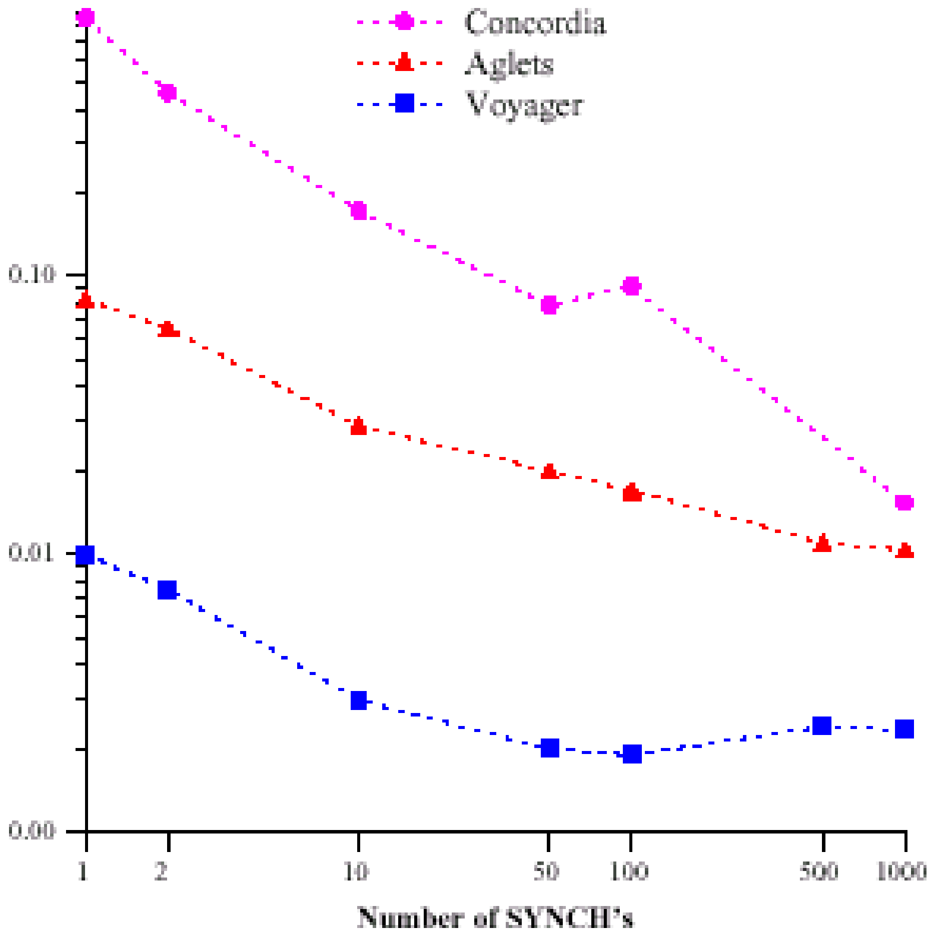


Platform	CL		CR		AD	
	Peak	Sustained	Peak	Sustained	Peak	Sustained
	(agents/sec)		(agents/sec)		(agnts/sec)	
Concordia	3125	3000	312.5	310	25.68	25.6
Aglets	65.78	11	29.76	11.05	5.9	5.36
Voyager	1189.06	1100	38.8	38.8	11.58	8.31

Messaging Timings

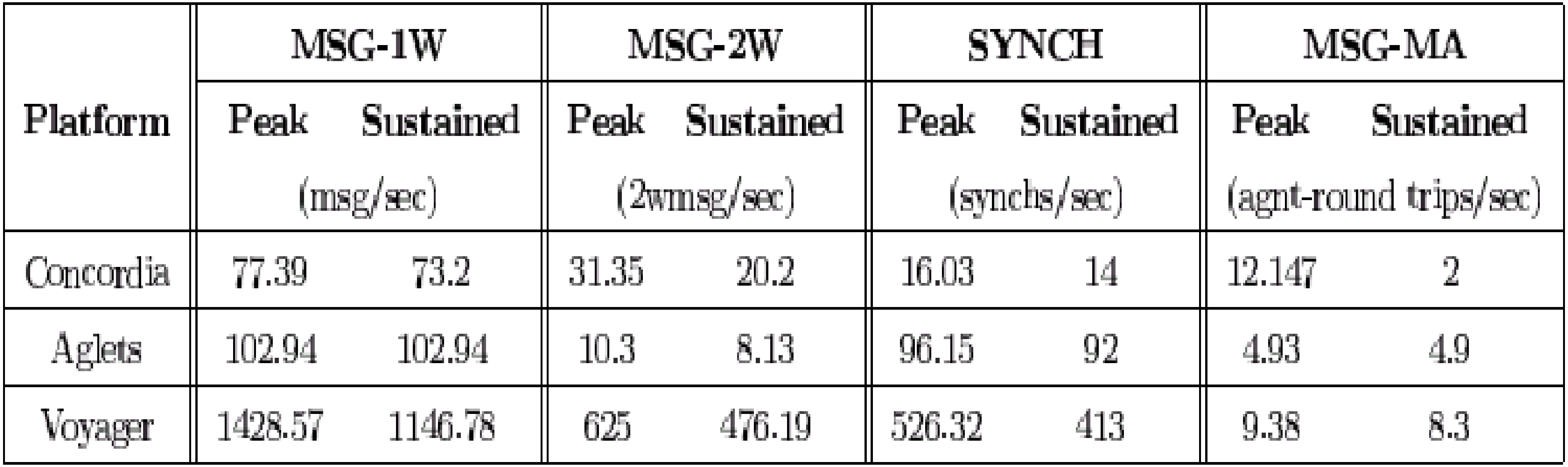


[MSG-1W] Benchmark (log-log scale)



[SYNCH] Benchmark (log-log scale)

Messaging Performance



Platform	MSG-1W		MSG-2W		SYNCH		MSG-MA	
	Peak	Sustained	Peak	Sustained	Peak	Sustained	Peak	Sustained
	(msg/sec)		(2wmsg/sec)		(synchs/sec)		(agnt-round trips/sec)	
Concordia	77.39	73.2	31.35	20.2	16.03	14	12.147	2
Aglets	102.94	102.94	10.3	8.13	96.15	92	4.93	4.9
Voyager	1428.57	1146.78	625	476.19	526.32	413	9.38	8.3

Micro-kernels

C/S Captures the performance of an agent acting as server in a C/S setting.

C/A/S Captures the capacity of a place to host intermediary agents and the performance thereof, acting in a C/A/S setting.

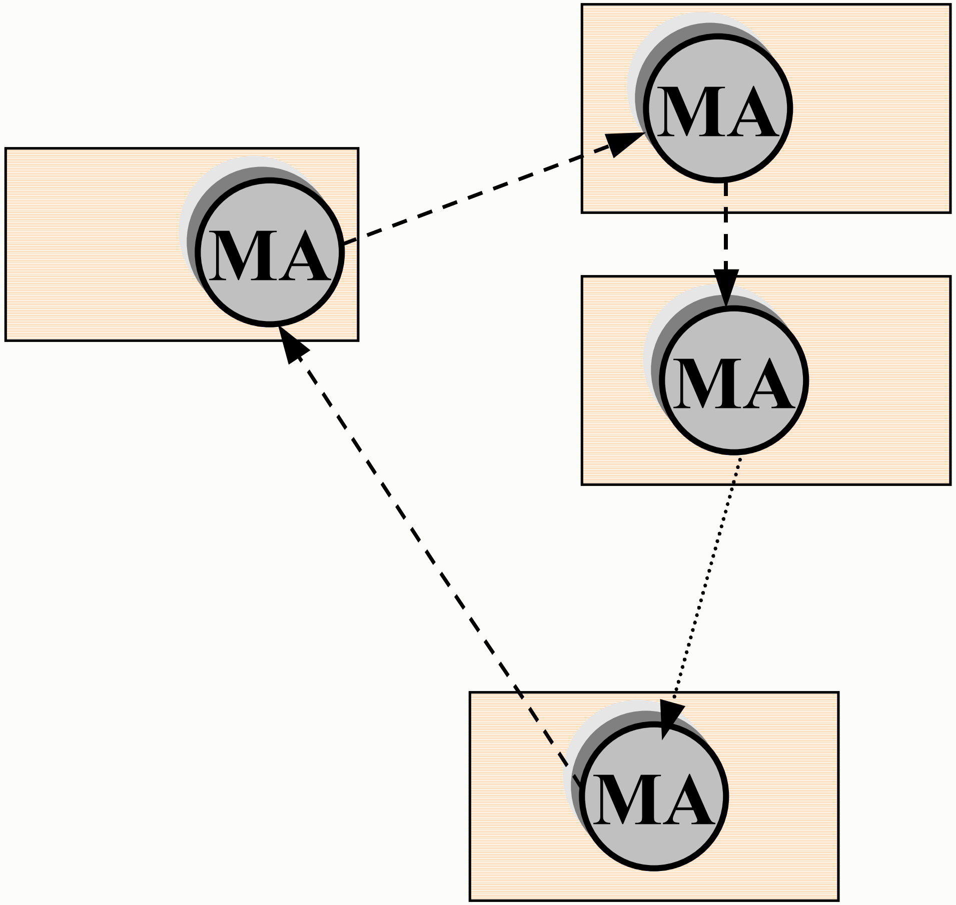
ROAM Captures the overhead of an agent roaming across different places.

M/S Captures the overhead of an agent acting as master in a M/S setting.

FORW-MSG Captures the performance of an agent acting as a router of msg. requests towards a farm of server-agents.

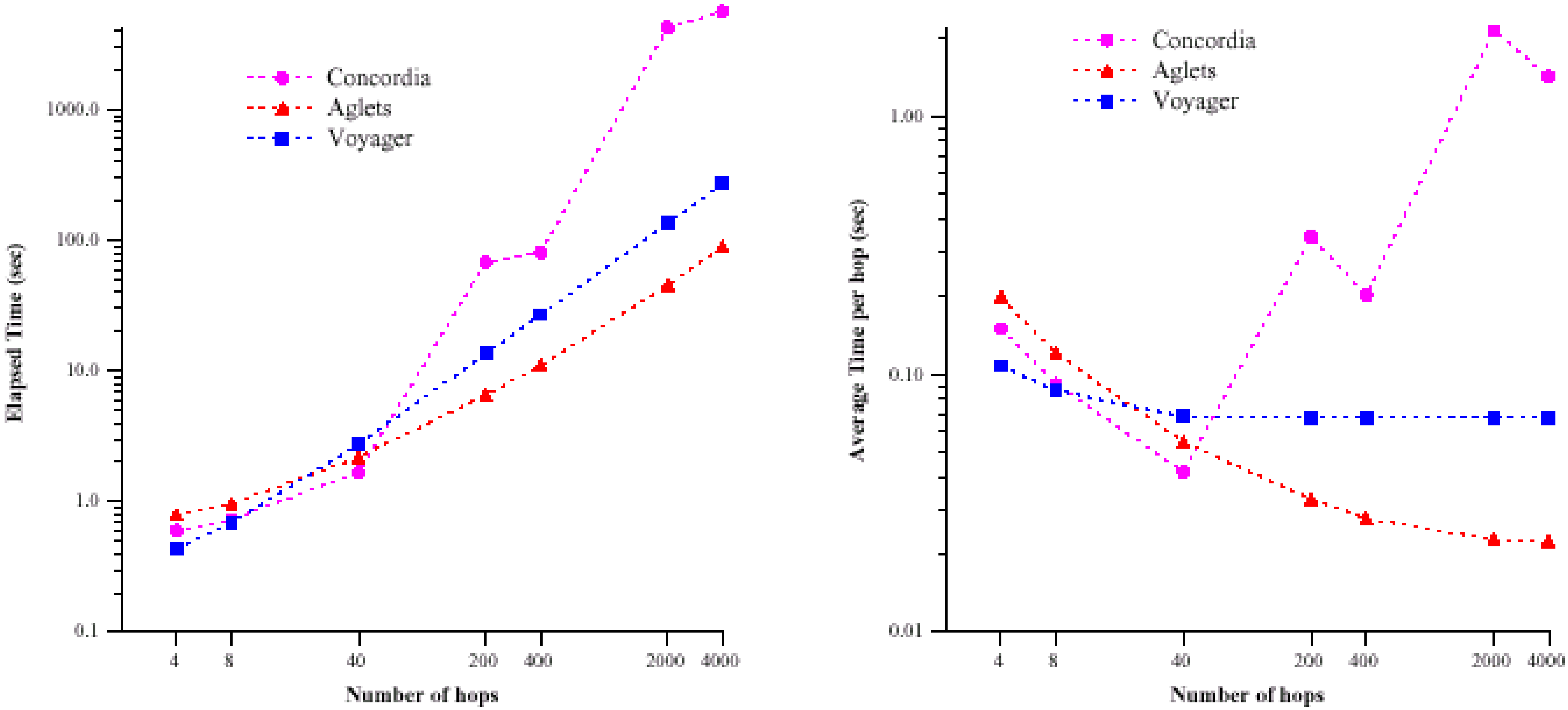
FORW-MA Captures the performance of an agent acting as a router of messenger agents towards a farm of server-agents.

ROAM

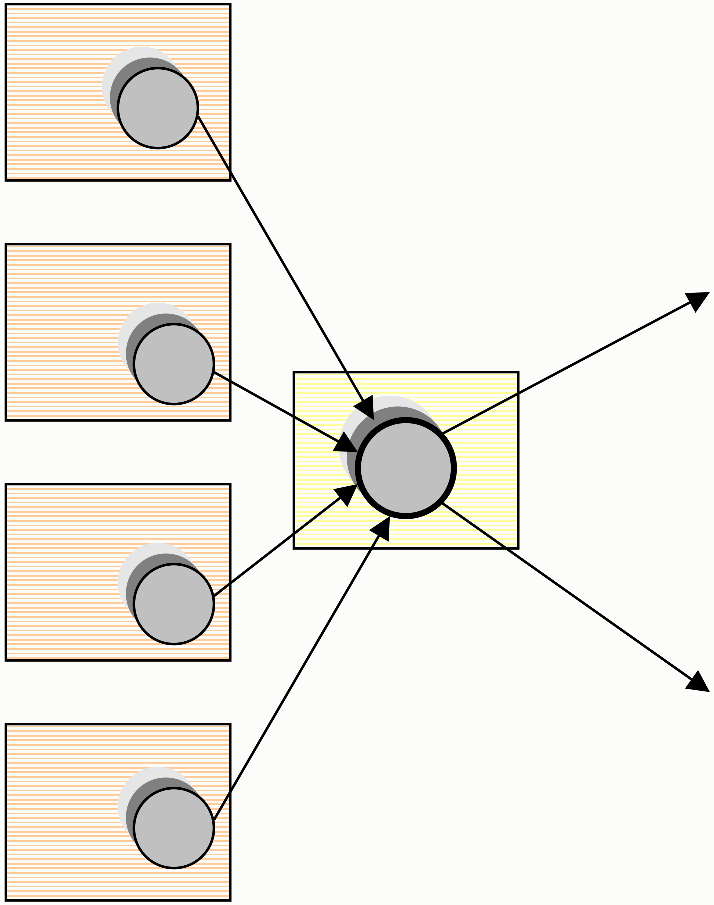


- Additional Parameters:
  - Number of places.
  - Number of hops.
- Metrics:
  - Total elapsed time.
  - Hops per second.
- Sustained Rates (4000 hops):
  - Voyager: 14.7 hops/sec
  - Aglets: 44.64 hops/sec
  - Concordia: 0.7 hops/sec (peak: 23.92 hps)

ROAM timings (4 places)

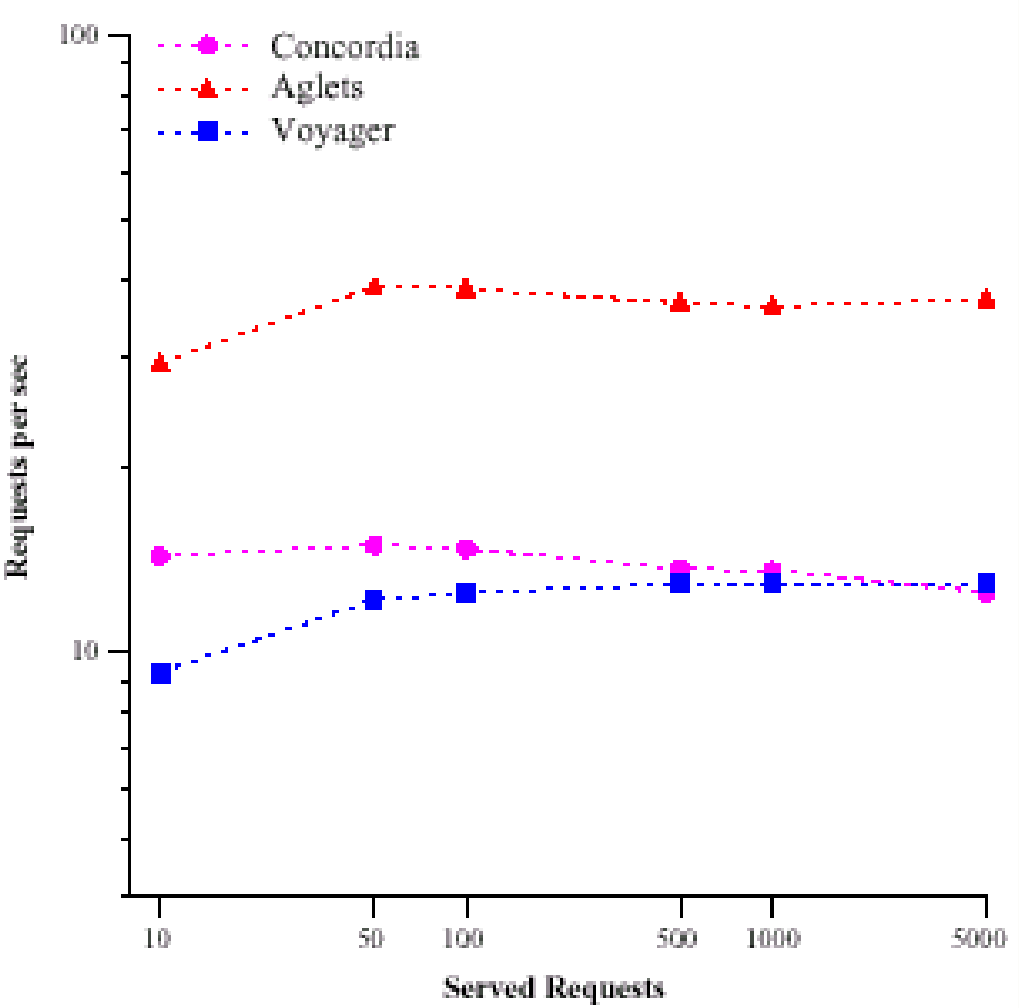


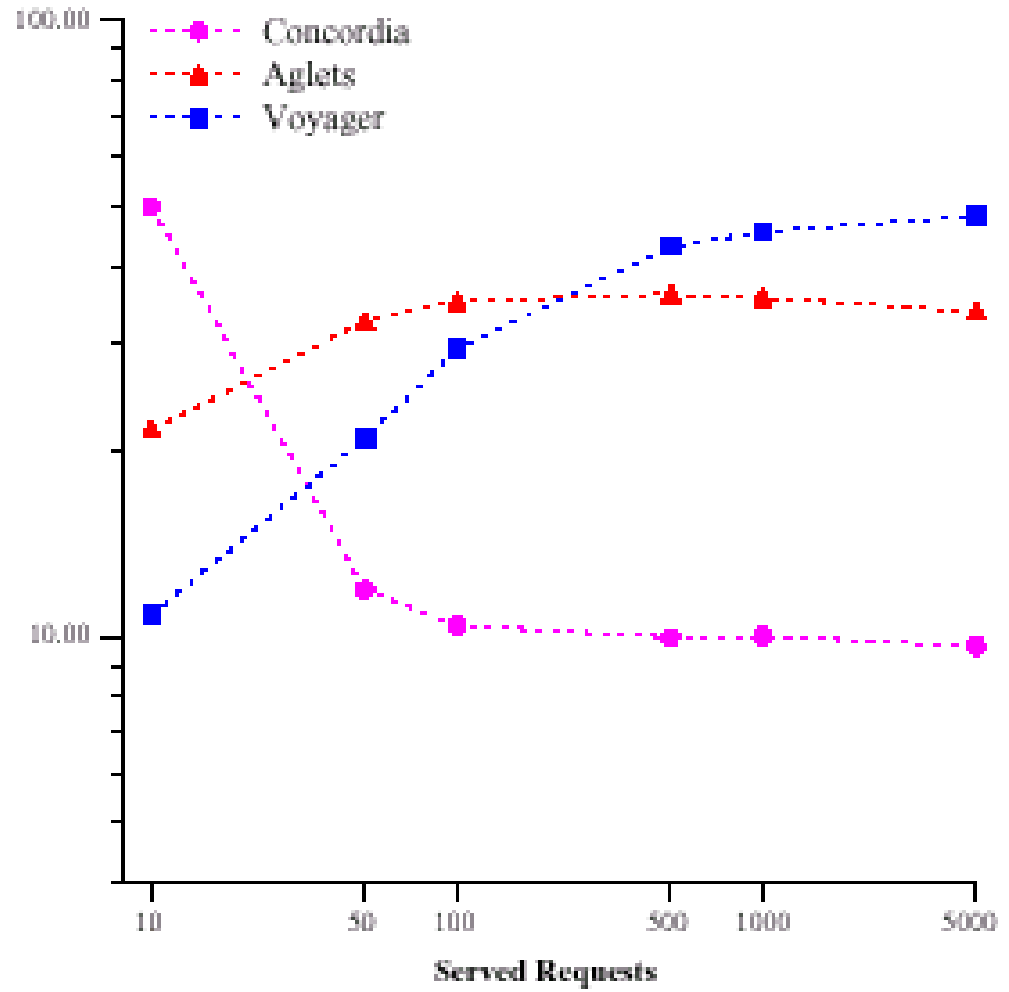
FORW-MSG



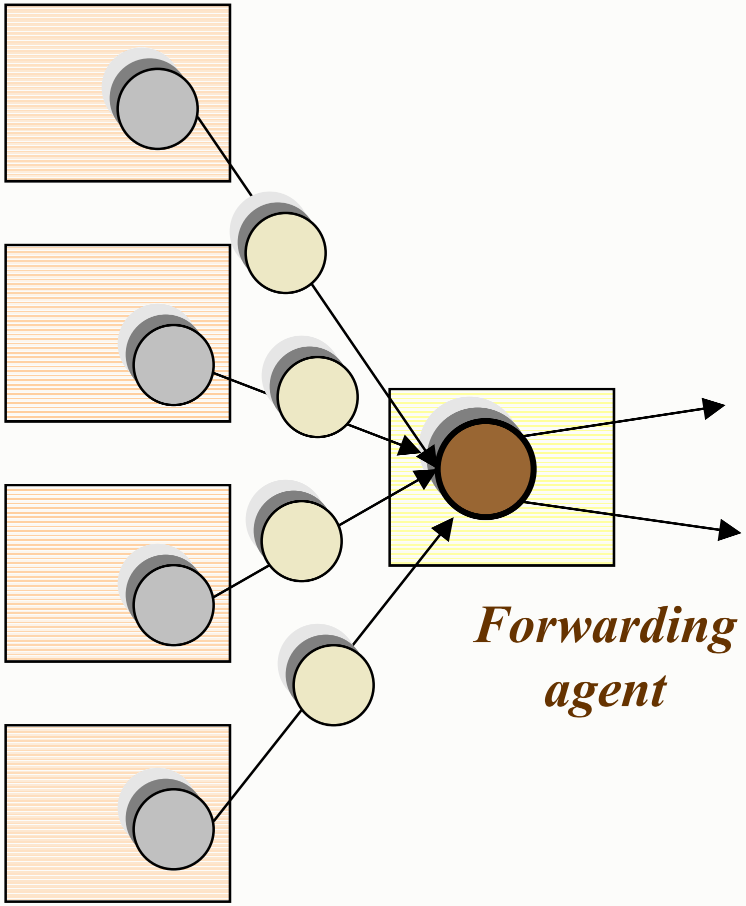
- Additional Parameters:
  - Number of clients.
  - Total number of requests.
- Metrics:
  - Total time to receive and forward requests.
  - Rate of request-handling.
- Measurements (3 clients, 3 servers):
  - Concordia: 12.48 requests/sec.
  - Voyager: 12.91 requests/sec.
  - Aglets: 37.33 requests/sec.

FORW-MSG (3/3, 12/3)





FORW-MA



- Additional Parameters:
  - Number of clients.
  - Total number of agents re-routed.
- Metrics:
  - Total time to receive and forward agents.
  - Rate of request-handling.
- Measurements (1 client, 1 server):
  - Concordia: 19.84 requests/sec.
  - Aglets: 9.54 requests/sec.
  - Voyager: 5.76 requests/sec.

Conclusions

- A framework for studying performance of MA middleware that:
  - Captures basic performance properties.
  - Isolates performance problems arising from lower-level implementation decisions.
  - Describes the performance capacity of MA systems.
  - Compares different middleware platforms quantitatively.
  - Helps design and programming decisions based on performance.
- Transporting and caching agent-state is a crucial factor that determines performance of mobility, messaging, etc. Caching mechanisms are hidden.
- O/S and JVM affect MA performance and robustness.
- Configuration of experiments is a real headache.

Current and Future Work

- Providing a definition of benchmarks compliant to the MASSIF standard.
- Further experiments with micro-applications under "realistic" workloads (e.g., TCP-W).
- Doing experiments at a wider-scale.

References

- *Tracker: A Universal Location Management System for Mobile Agents*. G. Samaras, C. Spyrou, E. Pitoura, M. Dikaiakos. European Wireless 2002 Conference, Italy, February 2002.
- *Performance Evaluation of Mobile-agent Middleware: A Hierarchical Approach.* M. Dikaiakos, M. Kyriakou, G. Samaras. 5th IEEE International Conference on Mobile Agents, December 2001.
- *Performance Evaluation of Mobile Agents: Issues and Approaches.* M. Dikaiakos, G. Samaras. In Performance Engineering. State of the Art and Current Trends. LNCS, Springer, May 2001.
- *Mobile Agent Platforms for Web Databases: A Qualitative and Quantitative Assessment*. G. Samaras, M. Dikaiakos, C. Spyrou and A. Liverdos. Joint Symposium of the First International Symposium on Agent Systems and Applications (ASA`99) and the Third International Symposium on Mobile Agents (MA`99), ASA/MA`99, November 1999.