

Autonomic Computing: basic concepts and existing infrastructures

PhD Course

Responsible: Elisabetta Di Nitto

Instructors: Luciano Baresi, Roberto
Cascella, Benko Borbala Katalin,
Raffaella Mirandola

Course Organization

Course dates:

24-28 November 2008, 14:00-18:00, Sala riunioni Ed. 23

Course evaluation:

preparation of a presentation on advanced topics

- specific topics will be proposed to the students
- students will present their work in the last lecture (date to be fixed)

Course Outline

- 24/11: Introduction: principles, overview and infrastructures for autonomic computing.
Algorithms for cooperation and interaction among distributed and independent components
(Elisabetta Di Nitto, Raffaella Mirandola)
- 25/11: A specific infrastructure for autonomic computing: the ACE toolkit + experimental usage
(Benko Borbala Katalin)
- 27/11: Supervision approaches (Luciano Baresi)
- 28/11: Security and trustworthiness in autonomic computing (Roberto Cascella)
- Xx/12: Students' presentations

Exams

1. Presentation to be given in the day that we will schedule

2. Possible topics

1. Analyze in detail one of the topics that will be presented through the lectures
2. Discuss on how autonomic computing can be used in your research context
3. Present some technique from your field and argument on how it can be used in the autonomic computing context

Principles, overview and infrastructures for autonomic computing

Algorithms for cooperation and interaction among distributed and independent components

Elisabetta Di Nitto, Raffaella Mirandola
Dipartimento di Elettronica e Informazione
Politecnico di Milano

*With lots of contributions by Daniele J. Dubois, Giorgio
Galvalisi, Silvia Bindelli, Matteo Rossi, ...*

Presentation roadmap

- Motivation and vision
- Overview of autonomic computing
- The CASCADAS approach
- Conclusion

Autonomic computing: motivation and vision – The starting point by IBM

- IT environments more and more complex and heterogeneous
- Need for
 - 24 hours availability
 - Hot deployment
 - Continuous monitoring
- Difficulties in
 - Integrating, installing, configuring, tuning, maintaining systems

Autonomic computing: motivation and vision – Moving toward "open world" software

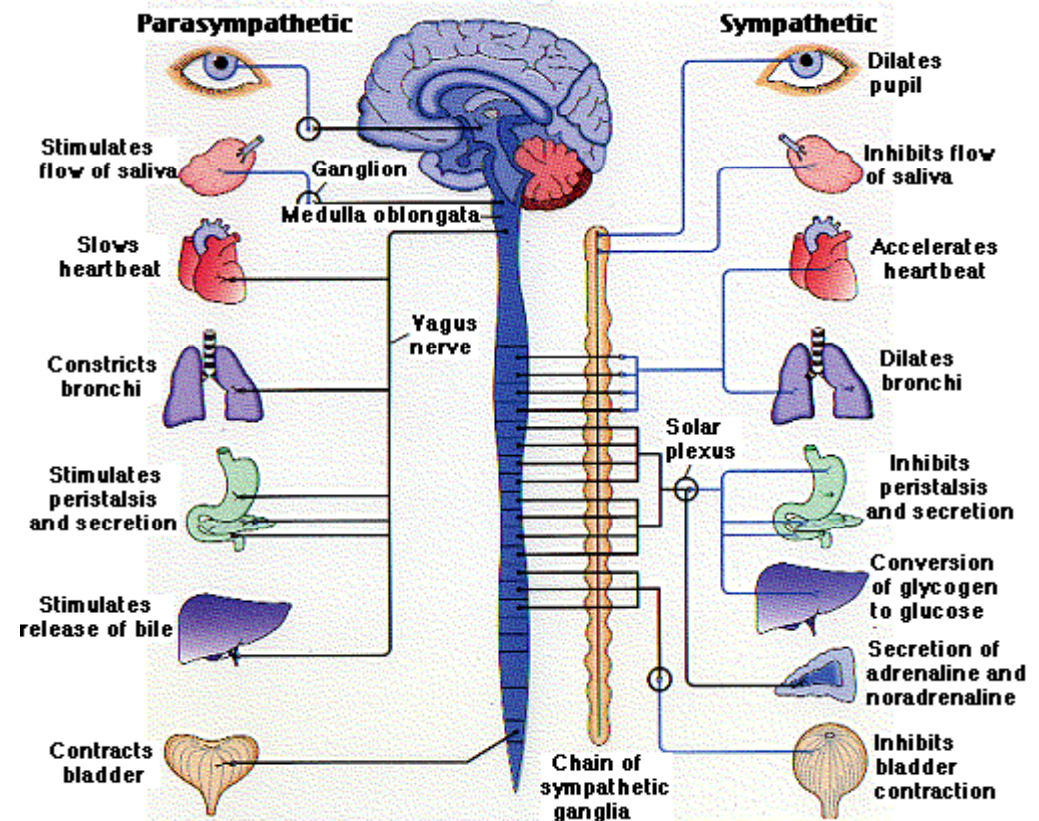
- A world where software elements not necessarily built to work together may interact to fulfill some specific goal
- Examples
 - Ambient intelligence and pervasive systems
 - A room is equipped with some sensors and a control system
 - A person wearing a device to control vital functions enters a room, the wearable device must start "interacting with the room" to acquire information about environmental conditions and communicate the person's needs
 - Compositions of existing (business) services
 - The service integration team identifies some needs, discovers some services, assemble them, and make them available to a large community
- Need to support the operation of highly dynamic systems
- Adaptation to the specific operation conditions becomes an essential issue

Autonomic computing...

- ... aims at addressing the issues mentioned before
- ... limiting human intervention as much as possible
- AC paradox
 - To make autonomic system to work, it needs to be more complicated than non- autonomic one
 - Human intervention is still sometimes necessary
 - We need IT-professionals with deep knowledge of the system-->but there are only few and that's why we want autonomic systems

What do we mean by autonomic computing?

- Inspired by the human autonomic nervous system
- Develops strategies and algorithms to handle complexity and uncertainties
- [Hariri et al 2006]



[<http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/P/PNS.html>]

What is an autonomic system?

- It is a generic system able to (IBM definition):

Self-Configure

- Creates its own configuration automatically

Self-Heal

- It is able to detect and react to malfunctions

Self-Optimize

- It is able to improve its way of working

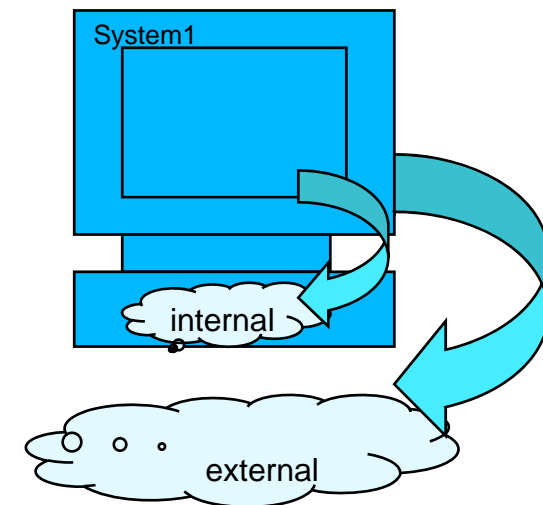
Self-Protect

- Reacts to malicious activity

- Tries to solve difficulties in maintaining complex systems manually
 - A self* architecture reduces human intervention in maintenance

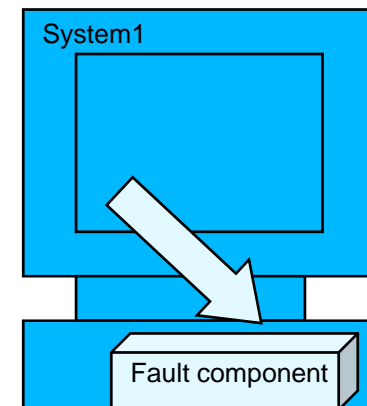
Self-configuration

- Adapt automatically to the dynamically changing environment
- Internal adaptation
 - *Add/remove new components (software)*
 - *configures itself on the fly*
- External adaptation
 - *Systems configure themselves into a global infrastructure*



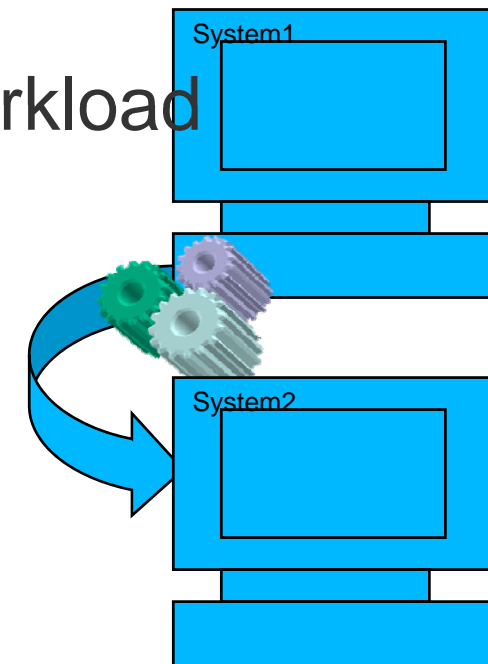
Self-healing

- Discover, diagnose and react to disruptions without disrupting the service environment
- Fault components should be
 - *Detected*
 - *Isolated*
 - *Fixed*
 - *Reintegrated*



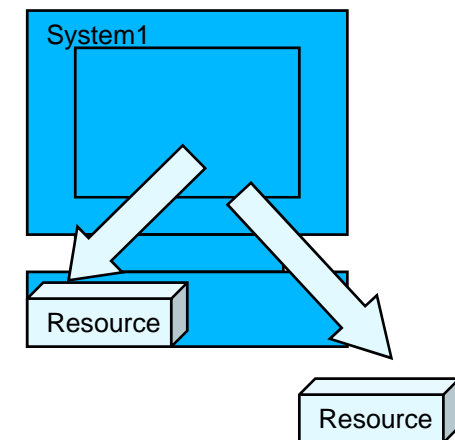
Self-optimization

- Monitor and tune resources automatically
 - *Support operating in unpredictable environment*
 - *Efficiently maximization of resource utilization without human intervention*
- Dynamic resource allocation and workload management.
 - *Resource: Storage, databases, networks*
 - *For example, Dynamic server clustering*



Self-protection

- Anticipate, detect, identify and protect against attacks from anywhere
 - *Defining and managing user access to all computing resources*
 - *Protecting against unauthorized resource access, e.g. SSL*
 - *Detecting intrusions and reporting as they occur*



Autonomic systems: Other characteristics [Hariri et al 2006]

- Self-aware: knows itself, its state, and its behavior
- Contextually aware: is aware of its execution environment
- Open: should be portable across multiple hardware and software architectures
- Anticipatory: should be able to anticipate its needs and behavior and those of its context. Should proactively manage itself

A brief history [M.Huebscher and J. A. McCann 2008]

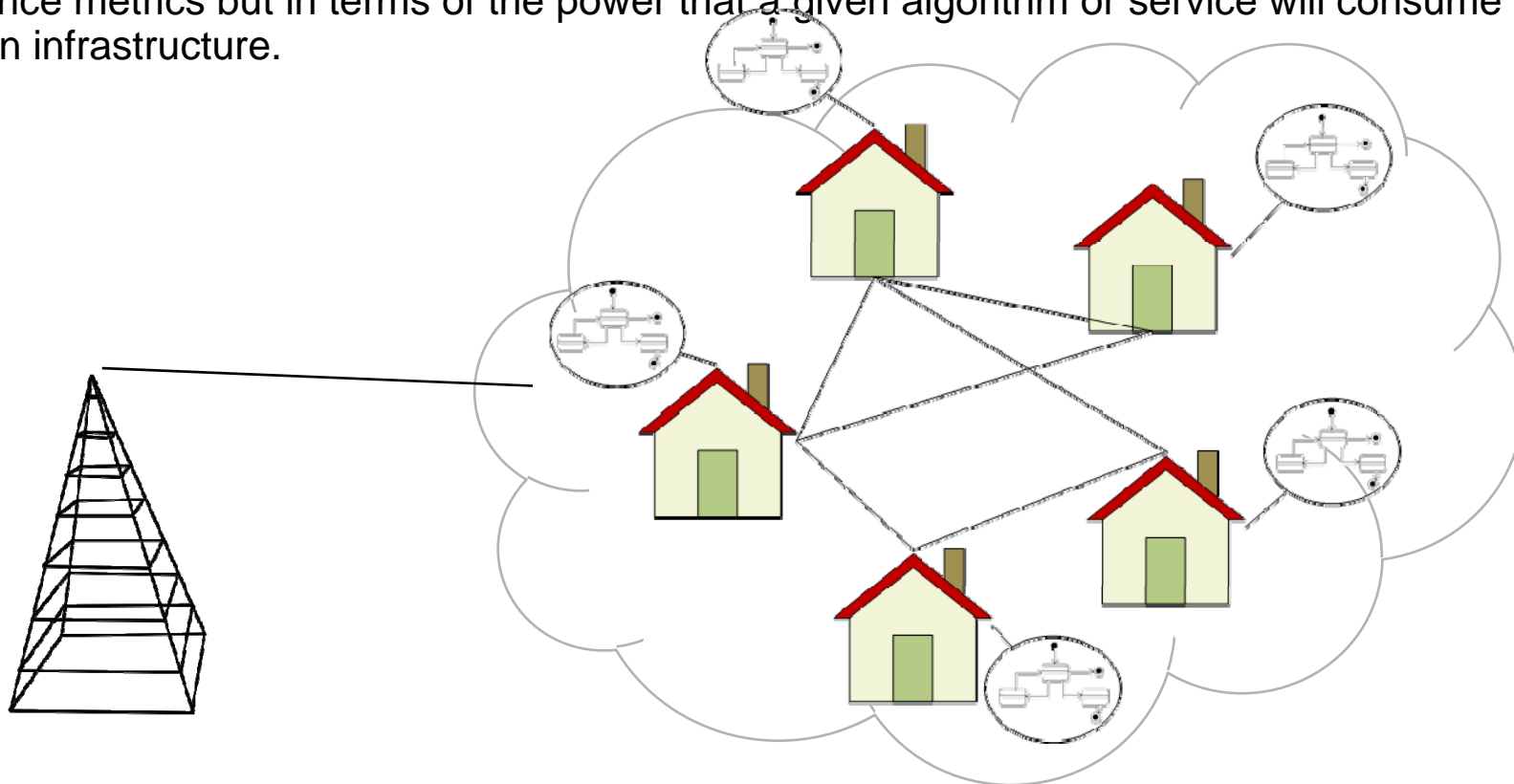
Table I. A Brief Chronology of Influential Self-Management Projects

SAS Situational Awareness System	1997	DARPA	Decentralized self-adaptive (ad-hoc) wireless network of mobile nodes that adapt routing to the changing topology of nodes and adapt communication frequency and bandwidth to environmental and node topology conditions.
DASADA Dynamic Assembly for Systems Adaptability, Dependability, and Assurance	2000	DARPA	Introduction of gauges and probes in the architecture of software systems for monitoring the system. An adaptation engine then uses this monitored data to plan and trigger changes in the system, e.g., in order to optimise performance or counteract failure of a component.
AC Autonomic Computing	2001	IBM	Compares self-management to the human autonomic system, which autonomously performs unconscious biological tasks. Introduction of the four central self-management properties (self-configuring, self-optimising, self-healing and self-protecting).
SPS Self-Regenerative Systems	2003	DARPA	Self-healing (military) computing systems, that react to unintentional errors or attacks.
ANTS Autonomous NanoTechnology Swarm	2005	NASA	Architecture consisting of miniaturized, autonomous, reconfigurable components that form structures for deep-space and planetary exploration. Inspired by insect colonies.

EMERGING APPLICATION AREAS CURRENTLY DOMINATING AUTONOMIC COMPUTING

Power Management

research in self-adaptive systems that not only optimize resource management in terms of performance metrics but in terms of the power that a given algorithm or service will consume on a given infrastructure.

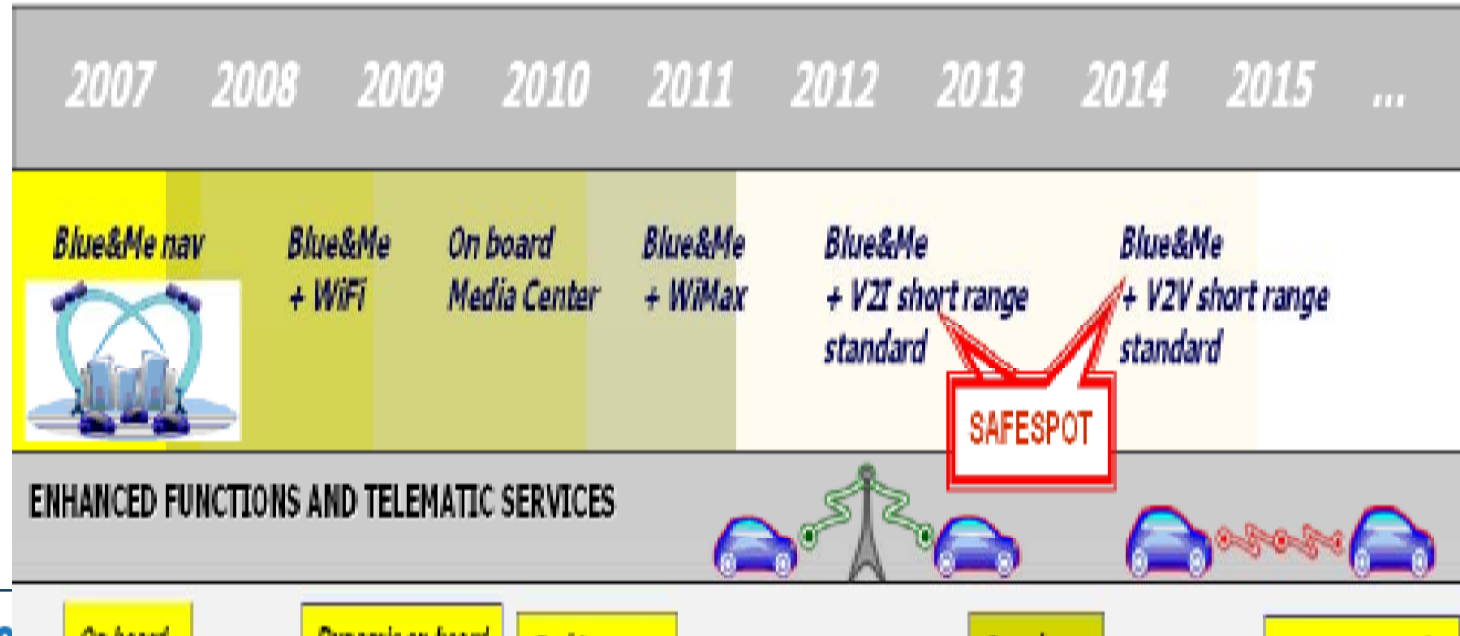


EMERGING APPLICATION AREAS CURRENTLY DOMINATING AUTONOMOMIC COMPUTING

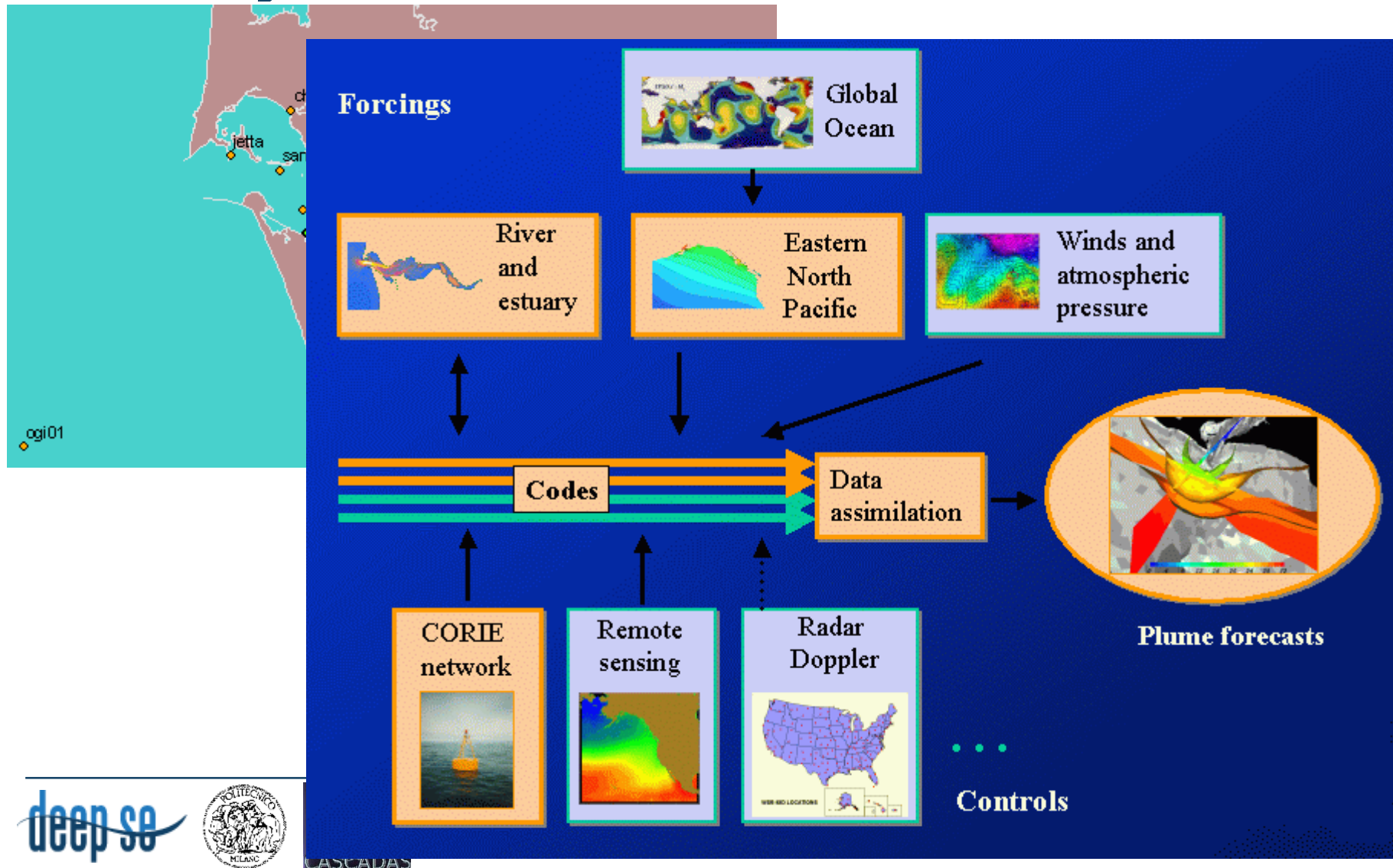
Ubiquitous Computing

A sensor network may deploy a huge number of nodes depending on the nature of the application. Such applications include medical services, battlefield operations, crisis response, disaster relief, environmental monitoring, premises surveillance, robotics etc.

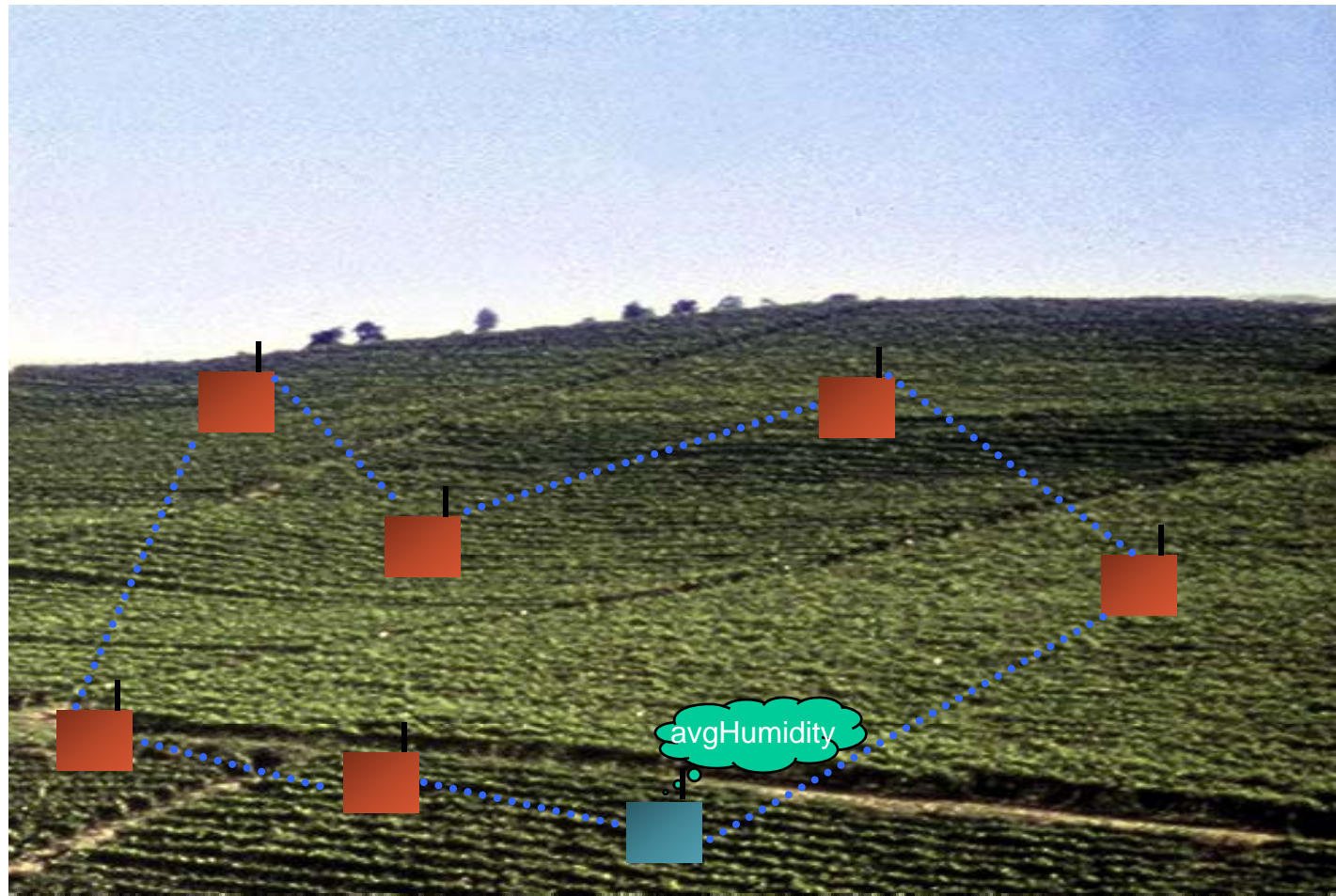
On the other hand, ubiquitous computing concerns the building of intelligent environments from a number of, potentially, heterogeneous devices such as sensor nodes, PDAs, PCs, etc.



Columbia river monitoring and forecast system (http://www.stccmop.org/CORIE/)



Vineyard control



Presentation roadmap

- Motivation and vision
- Overview of autonomic computing
- The CASCADAS approach
- Conclusion

Some of the research areas involved in autonomic computing

- **Software architectures**
 - Support to design, development, and operation of dynamic systems
 - Definition of open standards to support interaction among the various components
 - Definition of clean autonomic element structures that allow them to host various mechanisms to support self* characteristics
- **Multi-agent systems**
 - Partially overlapping with the first point
 - Support to planning, learning, self-awareness, context-awareness
- **Biologically inspired approaches**
 - Offer self-organization approaches derived from the study of emergent behaviors

Autonomic systems characterization

[Kephart 93]

- Autonomic elements
 - Ability to interact with other elements
 - Monitoring, event correlation, rule modeling and execution, optimization, forecasting, planning, learning [Kephart 2005]
 - Taking local decisions on the basis of local knowledge in a way that may affect the global system
- Autonomic systems
 - Ability to guarantee the self* properties
 - In the IBM philosophy this tends to be achieved through some centralized intelligence
- Interaction with humans
 - Policies
 - Suitable user interfaces

Levels of autonomicity [M.Huebscher and J. A. McCann 2008]

Description of autonomic approaches following the MAPE-K reference model

IBM has proposed a set of Autonomic Computing Adoption Model Levels:

Level 1: Basic

system elements are managed by highly skilled staff who utilise monitoring tools and then make the required changes manually

Level 2: Managed

where the system's monitoring tools can detect anomalies in an intelligent enough way to reduce the systems administration burden

Level 3: Predictive

more intelligent monitoring is carried out to recognize system patterns and suggest actions approved and carried out by IT staff

Level 4: Adaptive

the system uses the types of tools available to Level 3 system's staff but is more able to take action. Human interaction is minimized and it is expected that the performance is tweaked to meet service level agreements

Level 5: Autonomic

systems and components are dynamically managed by business rules and policies, thus freeing up IT staff to focus on maintaining ever-changing business needs.

No Autonomicity

Architectural approaches

IBM Model

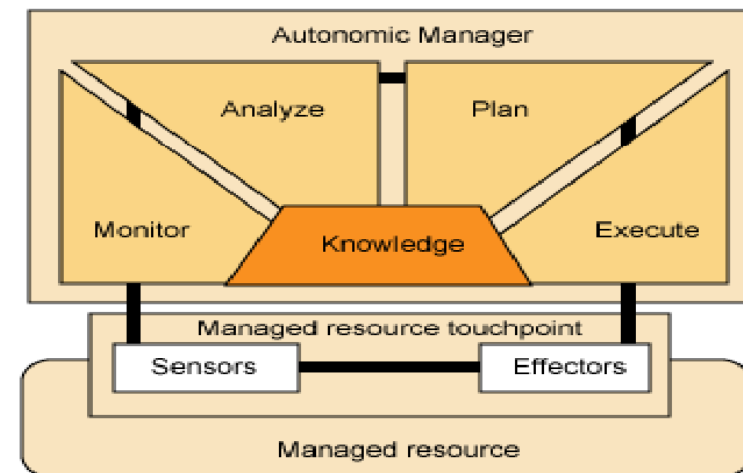
- Autonomic Systems: two autonomic elements

the **managed elements**:

- a single resource like a server or a router,
- or a collection of resources like a pool of servers, cluster or business application

the **autonomic managers**:

- collect and analyse data from managed elements, organise the actions needed to achieve goals and objectives, and control the execution

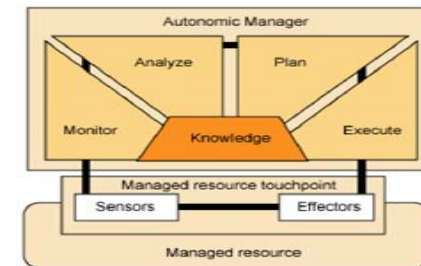


IBM Model

- Autonomic Systems: two autonomic elements

the autonomic manager implements the control loop:

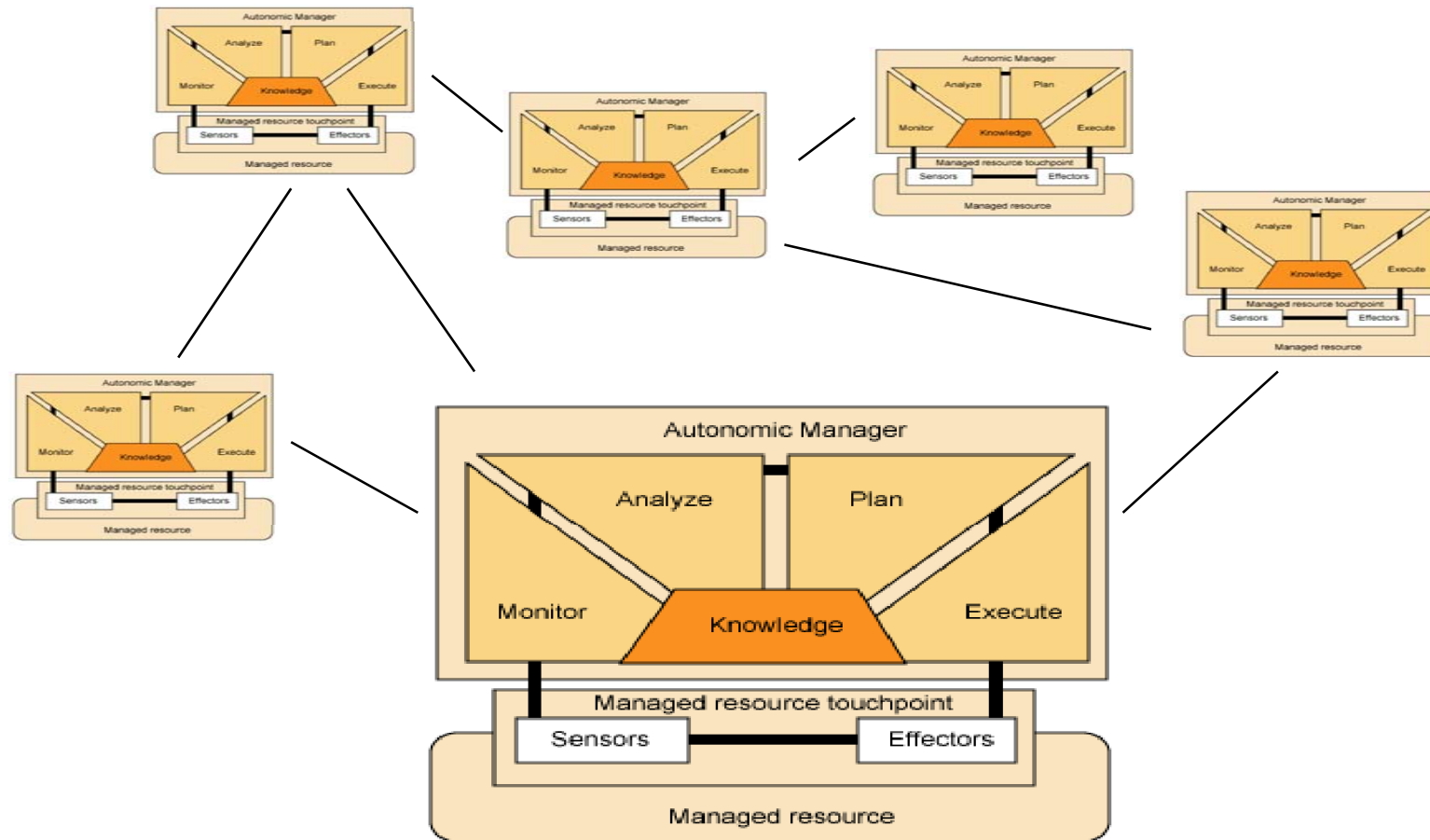
MAPE-K: Monitor, Analyze, Plan, Execute and Knowledge



-The manager collects and analyzes data related to a managed element, organizes the actions needed to achieve goals and objectives, and controls the execution. All the manager's functions consume and generate knowledge, which is continuously shared leading to better-informed decisions.

- The Monitor, Analyze, Plan and Execute engines parts collaborate (using asynchronous communication techniques, like a messaging bus) and exploit the common knowledge to provide the control loop functionality.

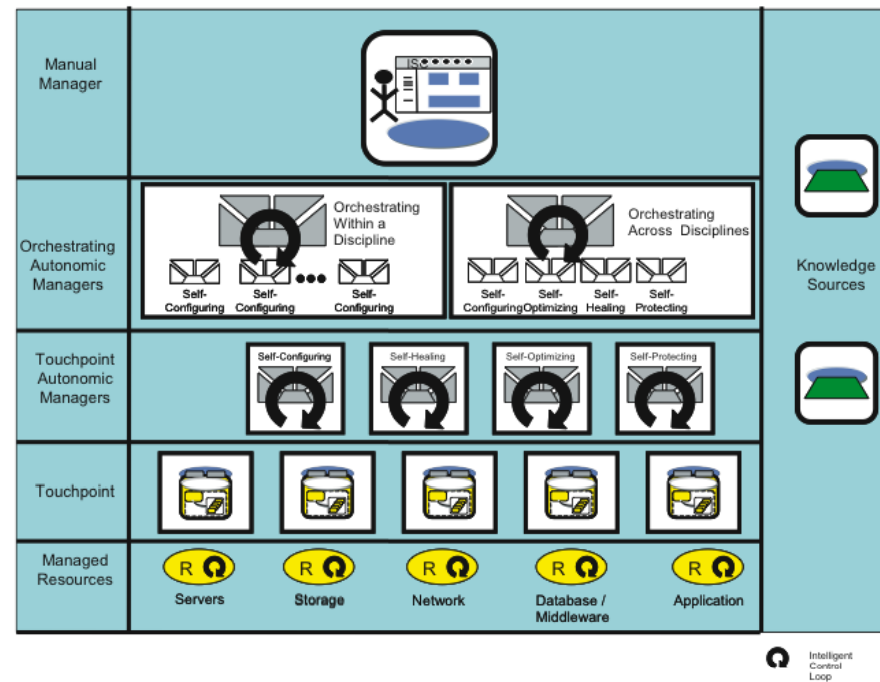
Autonomic system



IBM Model

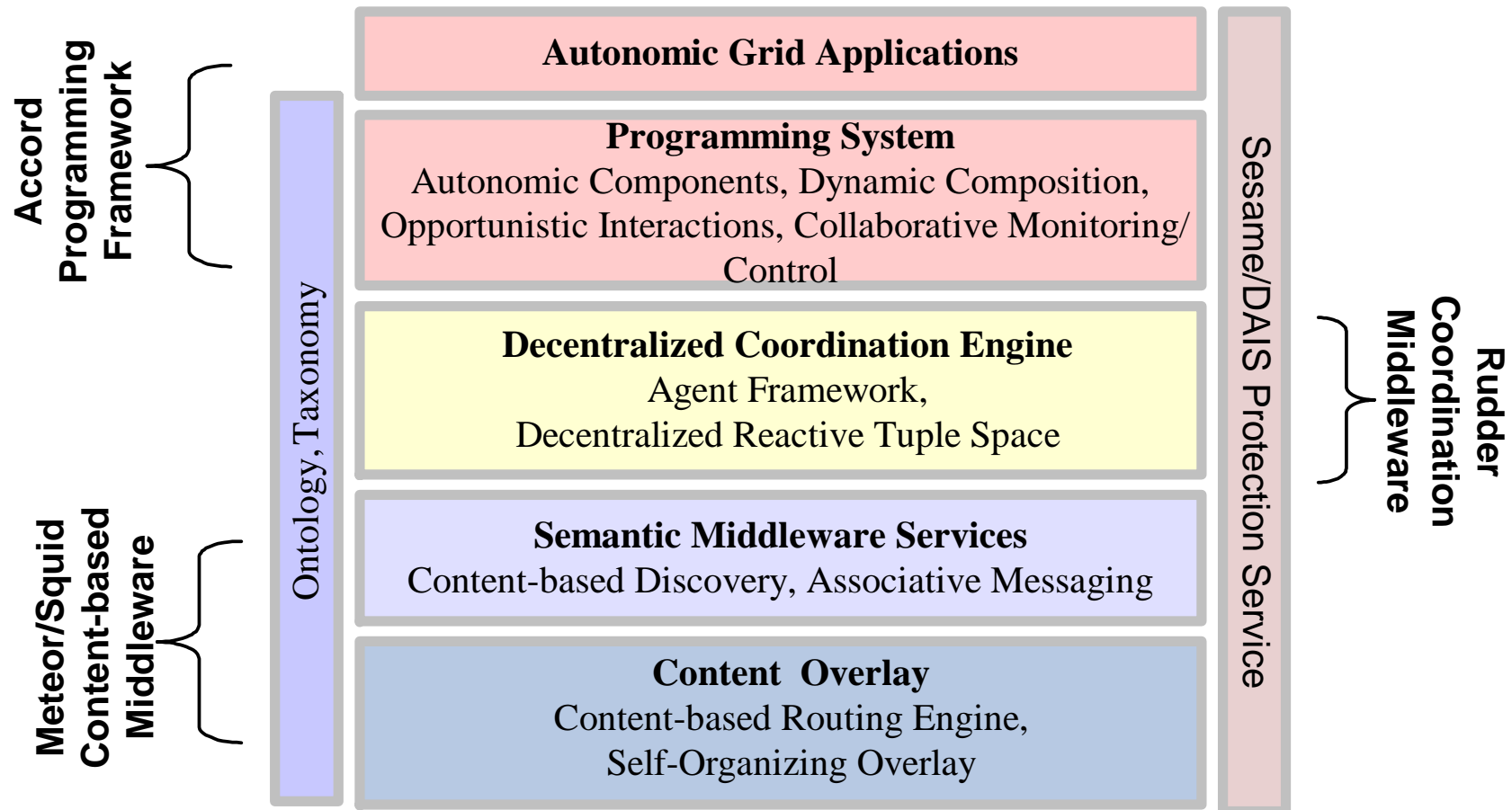
HCI:

a hierarchy of autonomic managers is built to provide common system management interfaces to be used by other autonomic managers and by IT professionals who need to monitor and control the system, and to set the desired degree of autonomicity and the global policies.



Automate: Enabling Autonomic Applications

(<http://automate.rutgers.edu>) [2006]



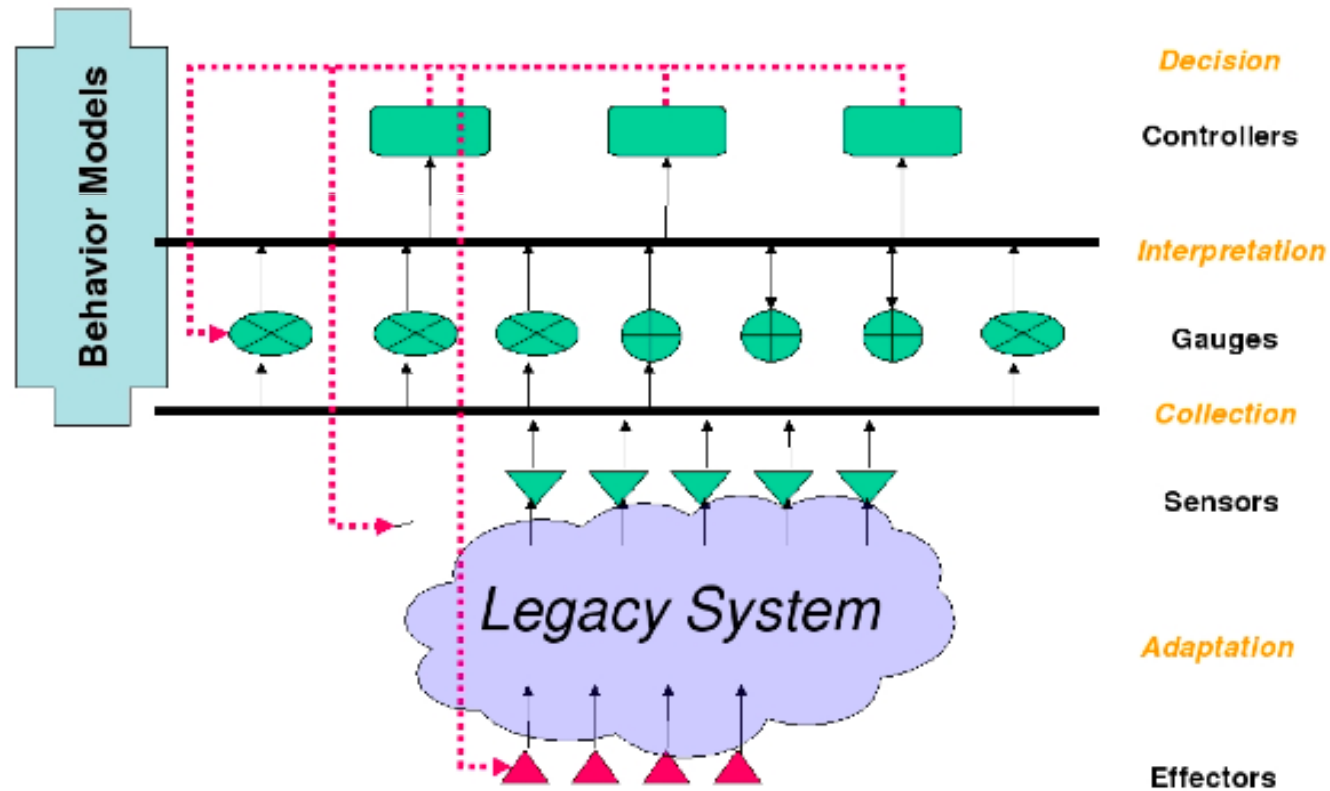
Automate: Enabling Autonomic Applications (<http://automate.rutgers.edu>) [2006]

- multilayer environment:
 - autonomic elements: agents dynamically composed using policies and constraints that are defined, deployed and executed at run time, and considering the available Grid resources (systems, services, storage, and data) and components.
 - The HCI and the coordination among autonomic elements build on a scalable implementation of a decentralised tuple space.

Kinesthetics Extreme [2003]

- Autonomic system: adds autonomic features to legacy systems.
 - several autonomic elements: *probes* (pieces of code that gather data from the running application)
 - gauges, collecting data from probes and using them to generate semantic events about the behaviour of the application;
 - controllers, implementing a control loop which uses the information from gauges to make decisions;
 - effectors, tuning or replacing components when needed.
- In this system the autonomic behaviour is overall achieved through the cooperation of different components, each performing a specific operation but without any intrinsic autonomic behaviour.

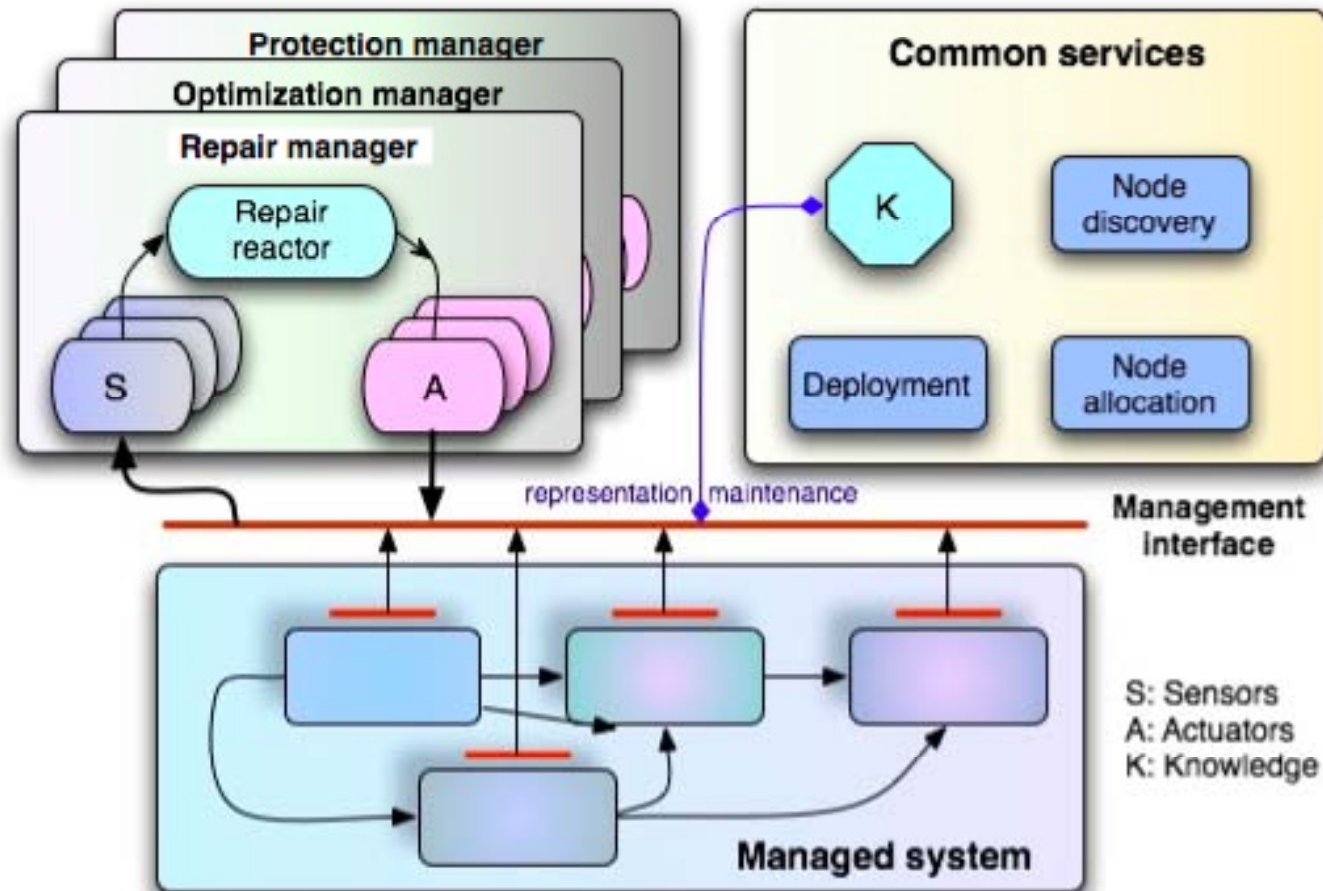
Kinesthetics Extreme [2003]



Jade [2006]

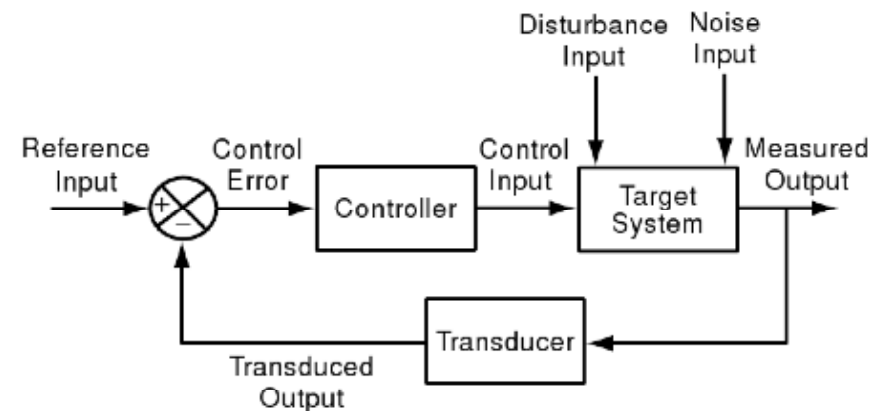
- Environment for the autonomic management of legacy systems whose goal is to simplify the work of system administrators and to optimise the use of resources.
- Jade autonomic elements are of two types:
 - the first one is the *Fractal model* containing pieces of code that provide proper interfaces while hiding the internal structure of the encapsulated component;
 - the autonomic manager that realises the autonomicity implementing feedback control loops.
 - HCI: Jade provides a management layer with a uniform interface, allowing autonomic programs to be built on. These programs reduce the need for the intervention of system administrators, and allow them to interact with a simpler interface, thus limiting the risk of errors.

Jade [2006]



Control-based [2004]

- DTAC framework:
- single autonomic element called *autonomic manager* whose implementation is that of *feedback loop*
 - allowing the achievement of the control goals of each single autonomic element.



Multi-agent systems

Unity: A Multi-Agent Systems Approach to Autonomic Computing [2004]

- the autonomic elements are:
 - entities (called agents) running their individual behaviours and self-managing, interacting with each other to achieve global high-level goals.
 - Different kinds of agents are implemented with different roles, such as resource managers, policy repositories and so forth.
 - Each agent is autonomously responsible for its behaviour and interacts with other agents to accomplish its own goals.

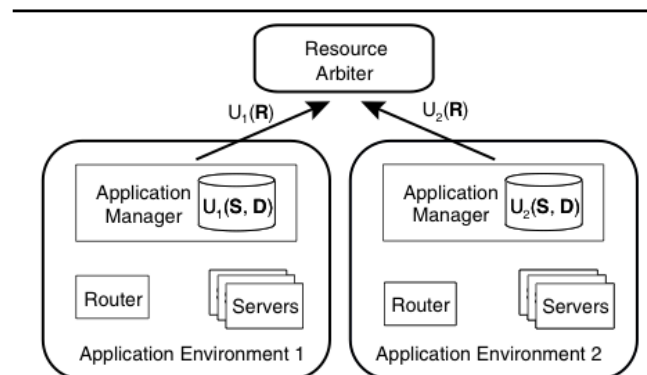


Figure 1. Architecture for self-optimization.

Biologically inspired approaches

Biologically inspired approaches

- Possible sources of inspiration [Nakano, Suda 2007]
 - Immune systems
 - Social insects colonies
 - Cellular systems
- All of them present interesting challenges in terms of
 - Scalability
 - Through local communication
 - Adaptability
 - Through the capability of learning by the past and changing behavior
 - Survivability/availability
 - Through the possibility of replacing dead individuals
- More generally, these systems self-organize

Self-organization

- A spontaneous formation of well-organized structures, patterns and behaviors without central control
- Accomplished by individuals with limited knowledge
- ... usually through very simple behavioral rules
- In ambiguous cases individuals will take random behavior
 - This may introduce noise in communication or even mutation of some individual

Some bio-inspired self-organization approaches

- [Babaoglu et al 2005]: patterns applicable to highly dynamic systems composed by nodes organized in some topology
 - Plain diffusion
 - Replication
 - Chemotaxis
 - Reaction diffusion
 - Stigmergy and learning by reinforcement

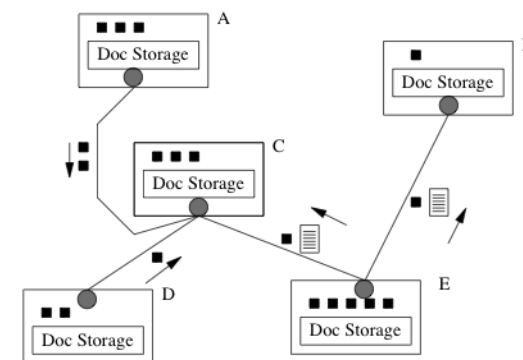
Some bio-inspired self-organization approaches

- [Nakano, Suda 2007] define an idea of Bio-Networking Architecture
 - Composed by cyber-entities (computing resources, software agents, humans, ...)
 - Networks of cyber-entities implement
 - Energy exchange
 - Migration
 - Replication
 - Reproduction
 - Death
 - Hibernation
 - The network evolves according to various rules (e.g, cyber-entities consume energy to live and produce energy when they offer services to others)

Anthill [2001]

- Infrastructure targeted at the study, design and analysis of P2P applications based on ideas such as multi-agent and evolutionary programming.
- The Anthill system is composed of:
 - autonomic elements called nests that can perform computations or store information, and communicate with other nests by exchanging *ants*, which are autonomous agents capable of moving across the network, interacting with the nests they visit.
- The interactions and movements of the *ants* is what achieves a complex behaviour at the system level.

A nest network composed of five nests. Each nest contains a document storage and communicate with other nests through a gateway (gray spots). Some ants (black spots) are managed by ant managers included in nests, while others travel through the network possibly carrying documents with them.



Presentation roadmap

- Motivation and vision
- Overview of autonomic computing
- The CASCADAS approach
- Conclusion

The CASCADAS approach

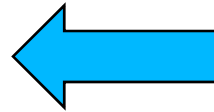
- The development of autonomic systems can be viewed from two perspectives:
 - Top-down intelligent control loop
 - Bottom-up local interaction determine the emergence of global behavior

The CASCADAS approach (2)

- An architectural model for autonomic elements (ACEs – Autonomic Computing Elements)
 - A protocol for ACE engagement (Goal Needed/Goal Achievable)Tuesday lecture
- Some self-organization algorithms (bottom-up)
 - Formal and simulation models to show their utility
 - Libraries to be incorporated within ACEsToday
- Supervision mechanisms (top-down view)
 - They close the loop when self-organization does not work or it is not availableThursday lecture
- Mechanisms to assess security and trustworthiness of ACEs Friday lecture
- A distributed knowledge network

The self-organization approach within CASCADAS

- Some classes of cases being deeply studied and experimented
 - Self-aggregation
 - Self-differentiation
 - Synchronization and collaborative decision making



Self-Aggregation Algorithms

A Self-Aggregation algorithm is defined as an algorithm capable of enabling a spontaneous formation of groups of *compatible* nodes.

Assumption:

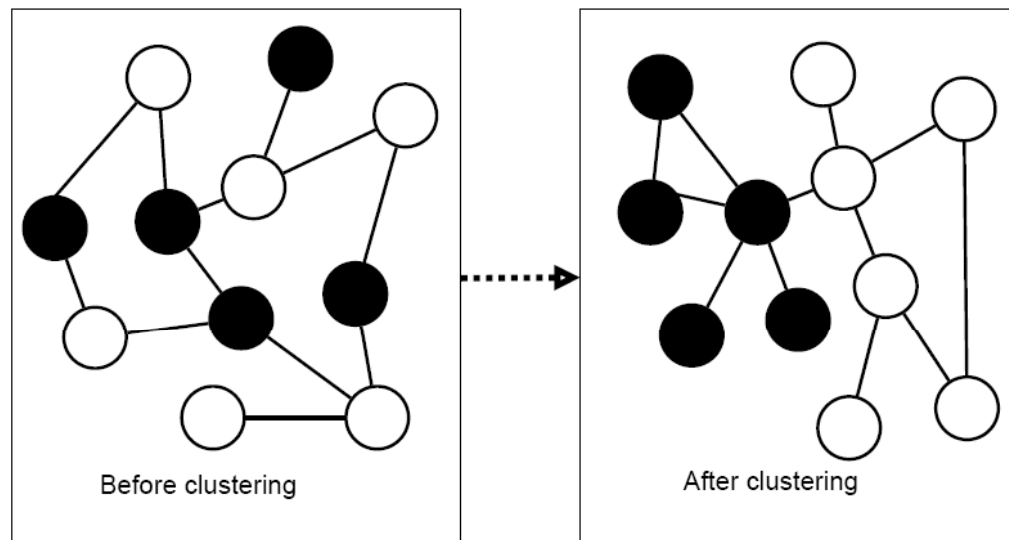
- Each node belongs to a structured type.
- Two nodes are compatible iff their types are compatible.
- Each node has a list of nodes called neighbors.
- Each node can communicate with other neighbors and modify their neighbor list (they are linked).
- There is no centralized control.

• Algorithm Goal:

- Reduce the number of links from incompatible nodes and add new links to compatible nodes.

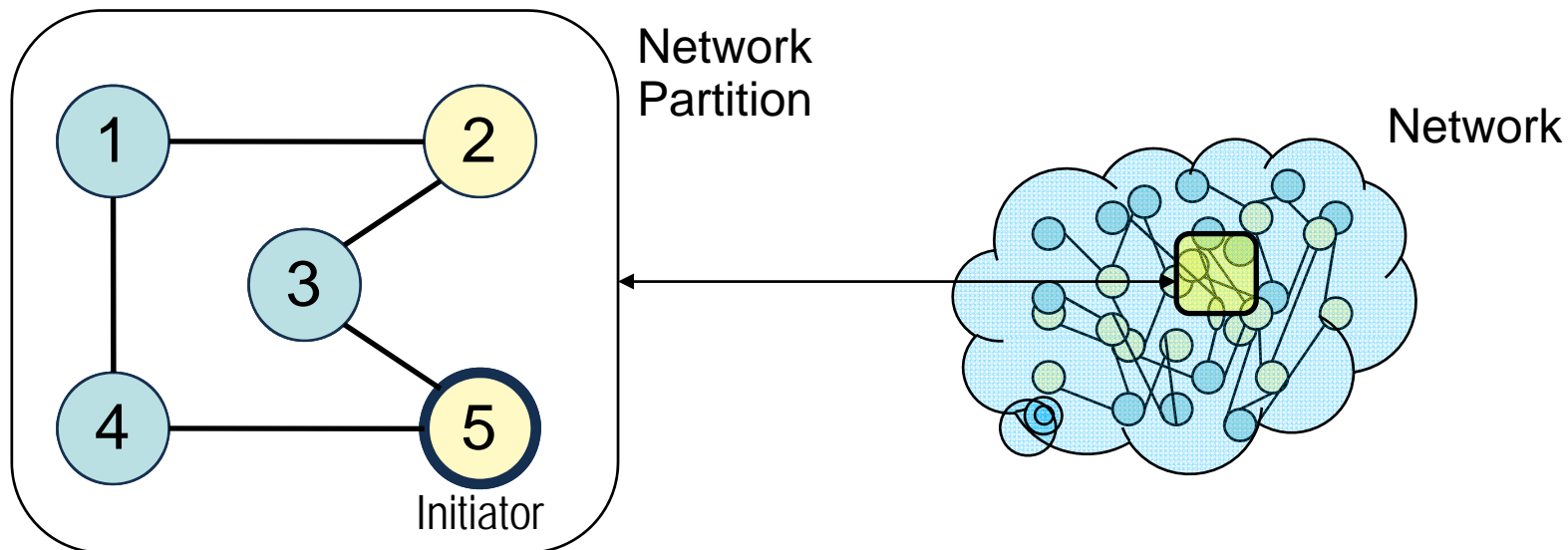
Self-Aggregation Examples

- **Clustering:**
 - Nodes are compatible if they have the same type.
- **Reverse Clustering:**
 - Nodes are compatible if they have different types.



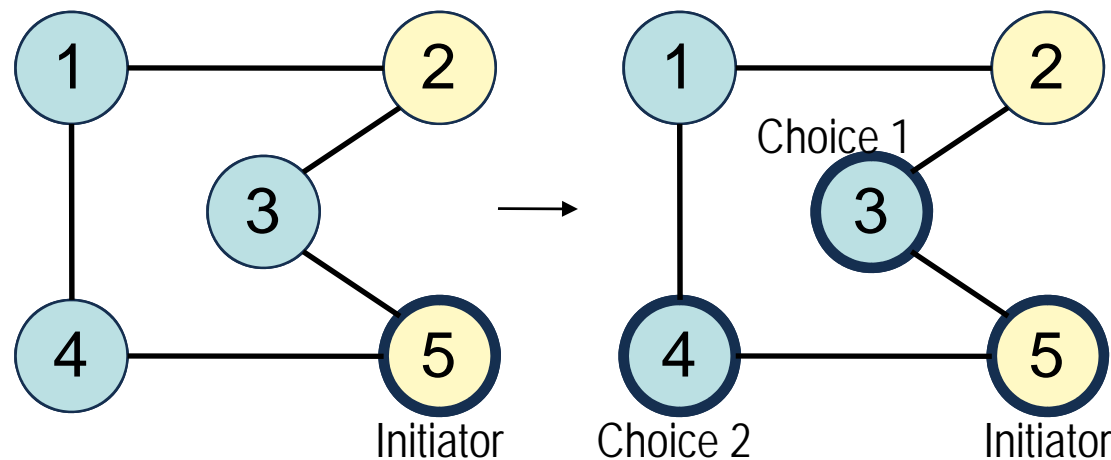
Clustering Algorithm Idea (Saffre et al, 2008)

1. Election of an initiator node: spontaneous in *Passive* mode, requested by a neighbor in *Active* mode;
2. The initiator chooses two neighbors that share the same type and creates a connection between them;
3. The initiator removes a link between itself and one of the chosen neighbors.



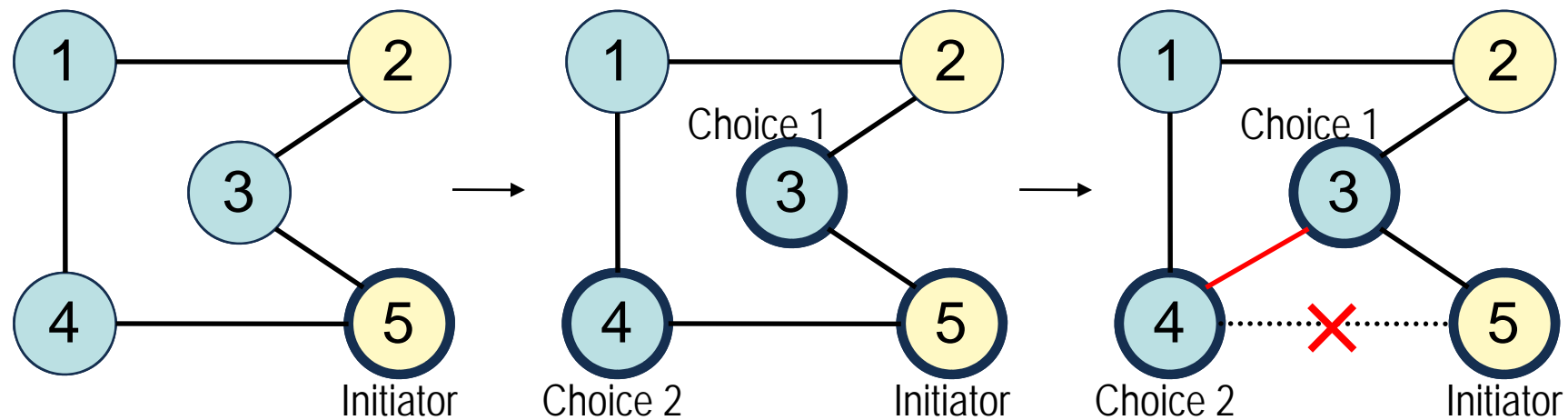
Clustering Algorithm Idea (Saffre et al, 2008)

1. Election of an initiator node: spontaneous in *Passive* mode, requested by a neighbor in *Active* mode;
2. The initiator chooses two neighbors that share the same type and creates a connection between them;
3. The initiator removes a link between itself and one of the chosen neighbors.



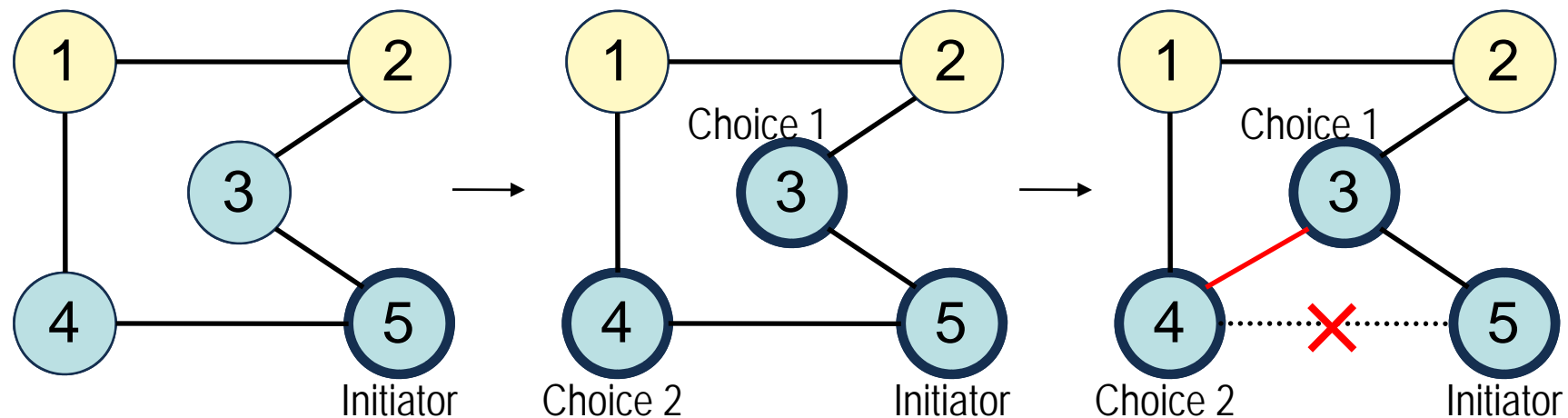
Clustering Algorithm Idea (Saffre et al, 2008)

1. Election of an initiator node: spontaneous in *Passive* mode, requested by a neighbor in *Active* mode;
2. The initiator chooses two neighbors that share the same type and creates a connection between them;
3. The initiator removes a link between itself and one of the chosen neighbors.



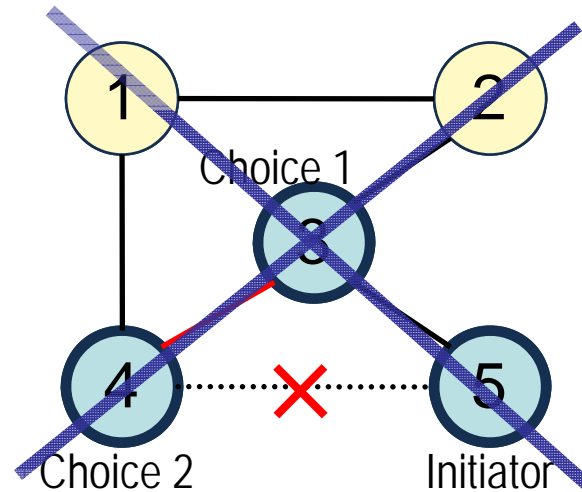
The concept of algorithm “noise”

- The noise in clustering algorithms is defined as the number of algorithm iterations that do not increase the system homogeneity.
- An example of iteration that contributes to algorithm noise:



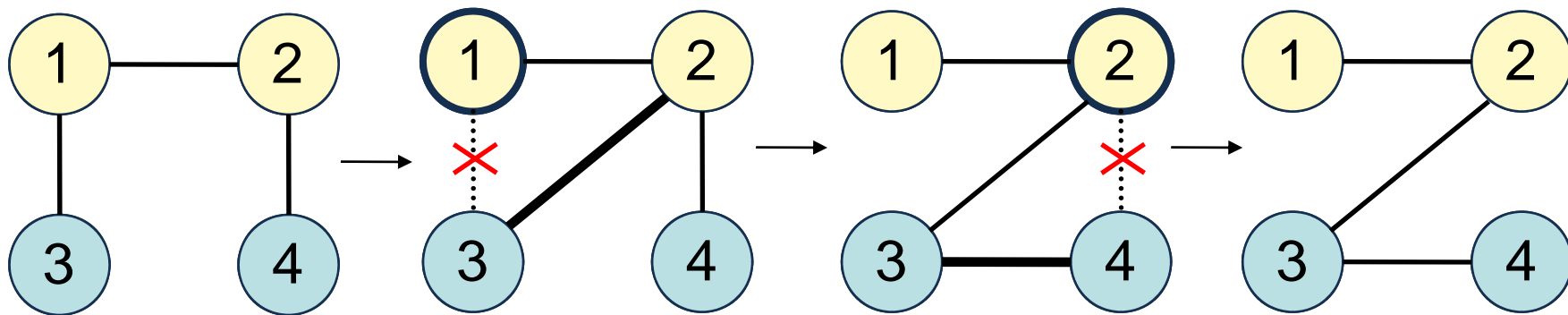
Noise Reduction: Fast Algorithm

- Like original clustering, with this additional constraint:
 - an algorithm iteration can **never** remove a link between compatible nodes (*always increasing homogeneity constraint*).
- **PROS:** less iterations, less messages, faster convergence.
- **CONS:** homogeneity may get stuck in local optima.



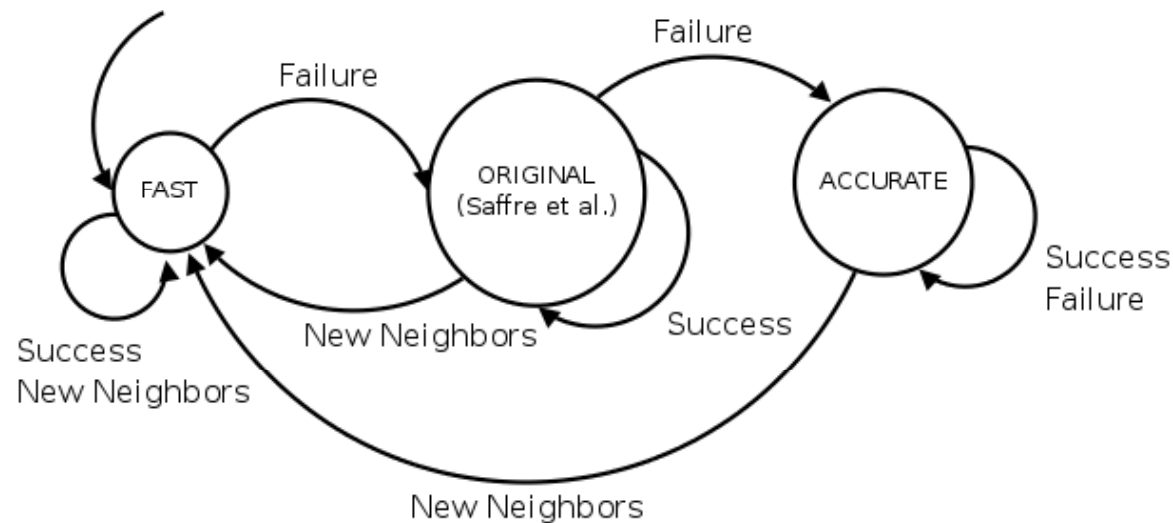
Noise Increase: Accurate Algorithm

- Links exchanges are executed with the following unique constraint:
 - A link between compatible nodes can be removed only if in the same iteration a link between compatible nodes is added (*non-decreasing homogeneity constraint*).
- **PROS:** homogeneity tends to the global optimum.
- **CONS:** more iterations, more messages, slower convergence.

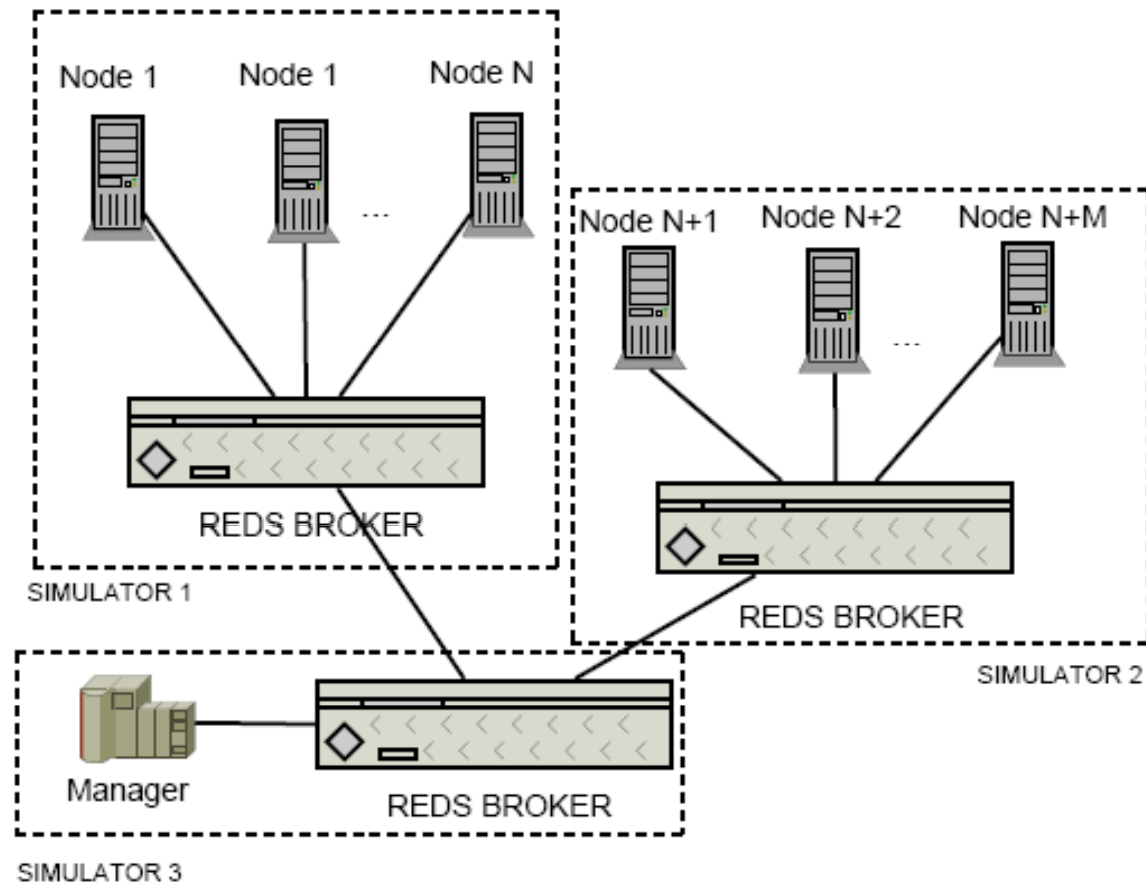


Adaptive Algorithm

- Algorithm noise can be dynamically adapted.
- Modeled as a FSM where:
 - States are the current clustering algorithms.
 - Transitions are the possible events that can be triggered by the algorithms.



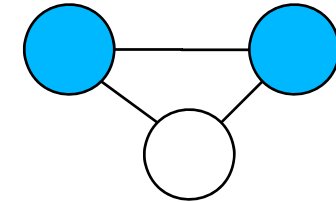
Performance analysis: architecture of the distributed simulator



Performance Analysis: Additional Performance indexes

- Network Homogeneity

$$H = \frac{\sum_{i=1}^N v(\text{node}_i)}{L}$$



- Optimality (clustering):

$$\text{opt}_{clustering} = \frac{H}{\max H}$$

maxH: 67%
opt: 100%

- Links variance:

$$\text{linksv} = \frac{\sum_{i=1}^N (d_i - \frac{2N}{L})^2}{N}$$

- Number of messages:
computed directly by the simulator.

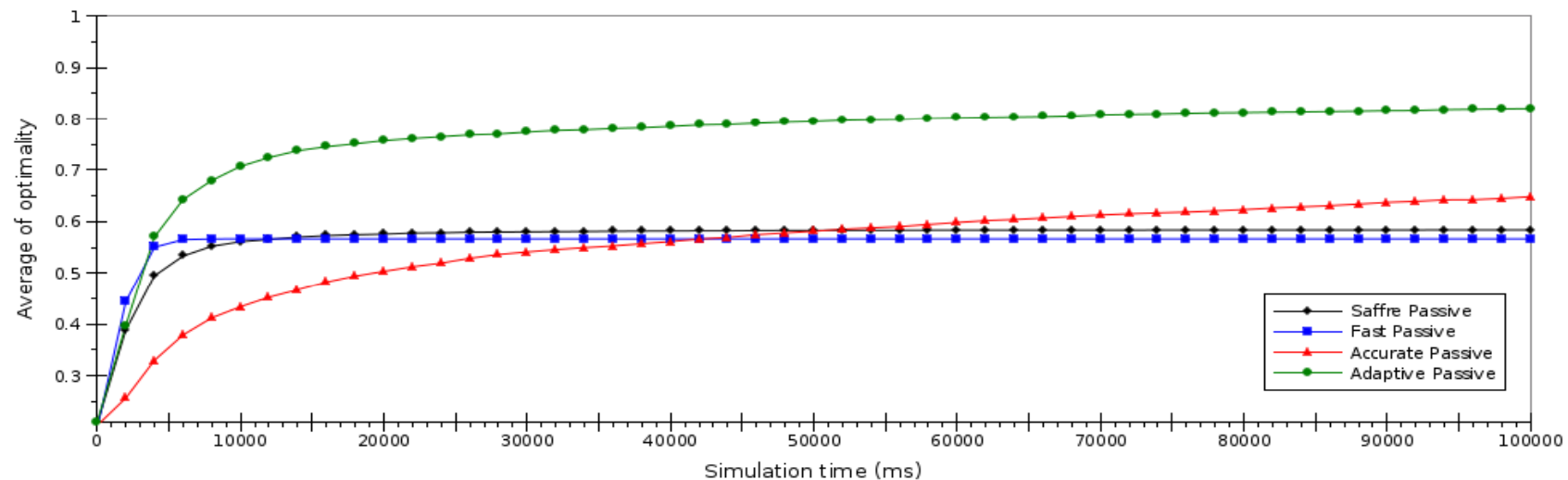
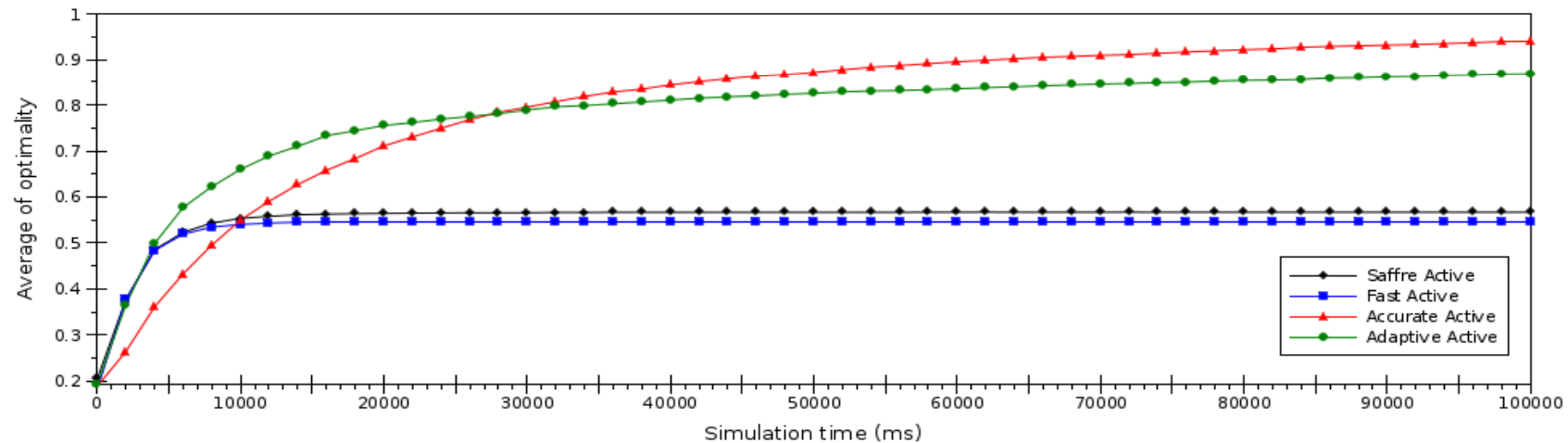
N: total number of nodes.
v(node_i): number of nodes compatible with node_i
maxH: maximum achievable homogeneity.
L: total number of links.
d(i): total number of links of node *i*.

Performance Analysis: Simulations

- *Monte Carlo* Simulations using a distributed simulation framework, 20 runs for each test.
- Input Parameters have been identified through some preliminary simulations:
 - Number of nodes;
 - Number of types;
 - Node degree;
 - Initial topology;
 - Algorithm mode (passive or on demand);
 - Maximum neighbor limit constraint;
 - Definition of compatibility (clustering or reverse clustering).
- Output Parameters: mean and variance of performance indexes.

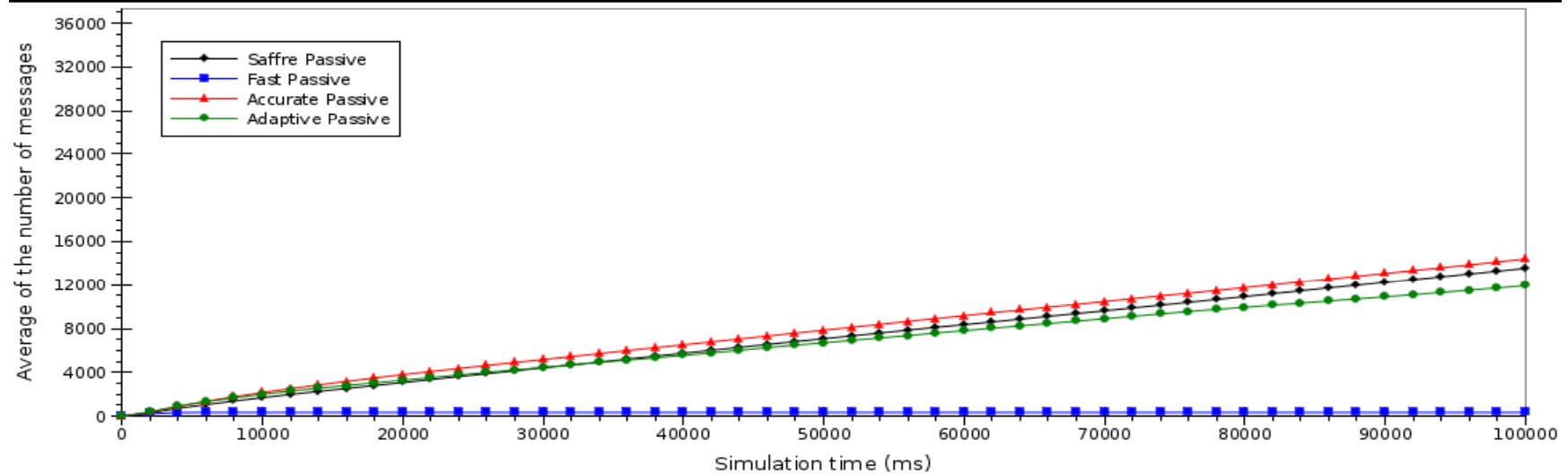
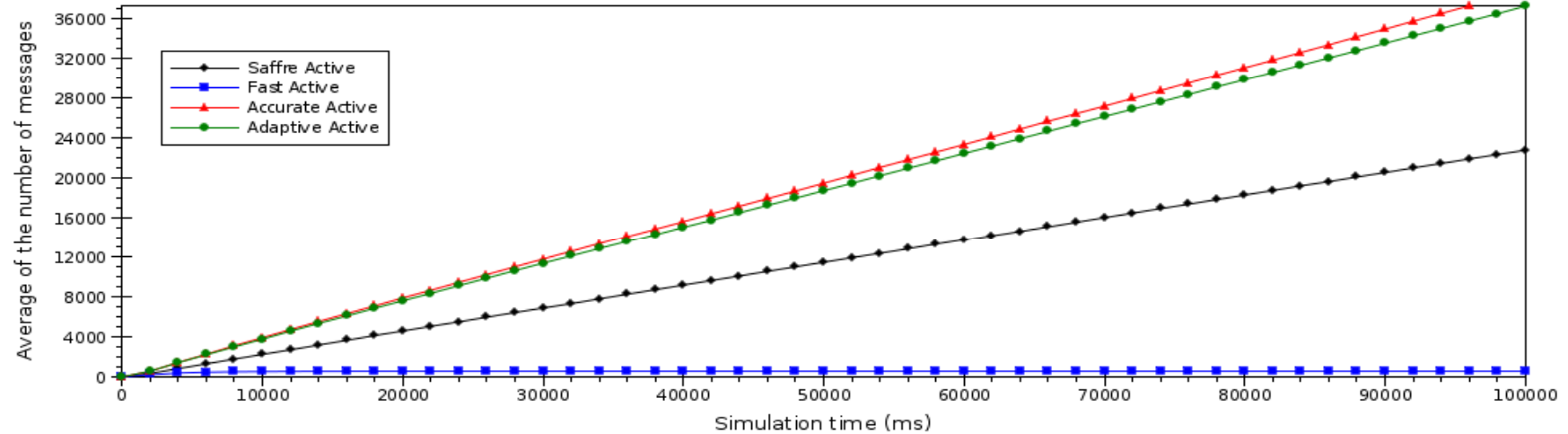
Experiment of reference: optimality average

Nodes=100, Links=4 per node, Types=5, Random topology, Clustering



Experiment of reference: number of messages

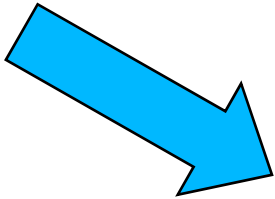
Nodes=100, Links=4 per node, Types=5, Random topology, Clustering



Summary

- **Best speed:**
 - Fast and Adaptive algorithms;
- **Lowest number of messages:**
 - Fast algorithm with passive protocol (with a max neighbors limit to avoid the *scale-free* effect);
- **Highest optimality/homogeneity:**
 - Accurate and Adaptive algorithms.
- **Adaptive algorithm due to its “adaptive” nature performs well in most common situations.**

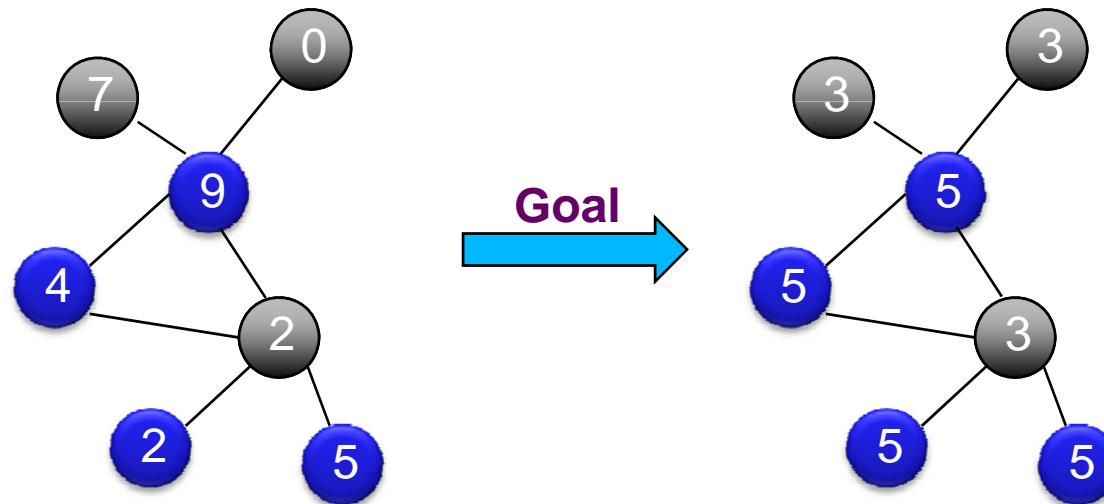
Possible Applications



- **Load-balancing problems:**
 - System nodes are specialized in executing particular tasks.
 - Jobs coming to an overloaded node should be passed to another node that is able to execute the same task.
 - Nodes should be able to self-reorganize their limited knowledge about the environment.
- **Overlay Self-Organization in Publish-Subscribe middleware:**
 - Rewire the broker connections in order to minimize network load.
 - Group together subscribers interested on the same topics.

Our Goal: Load Balancing

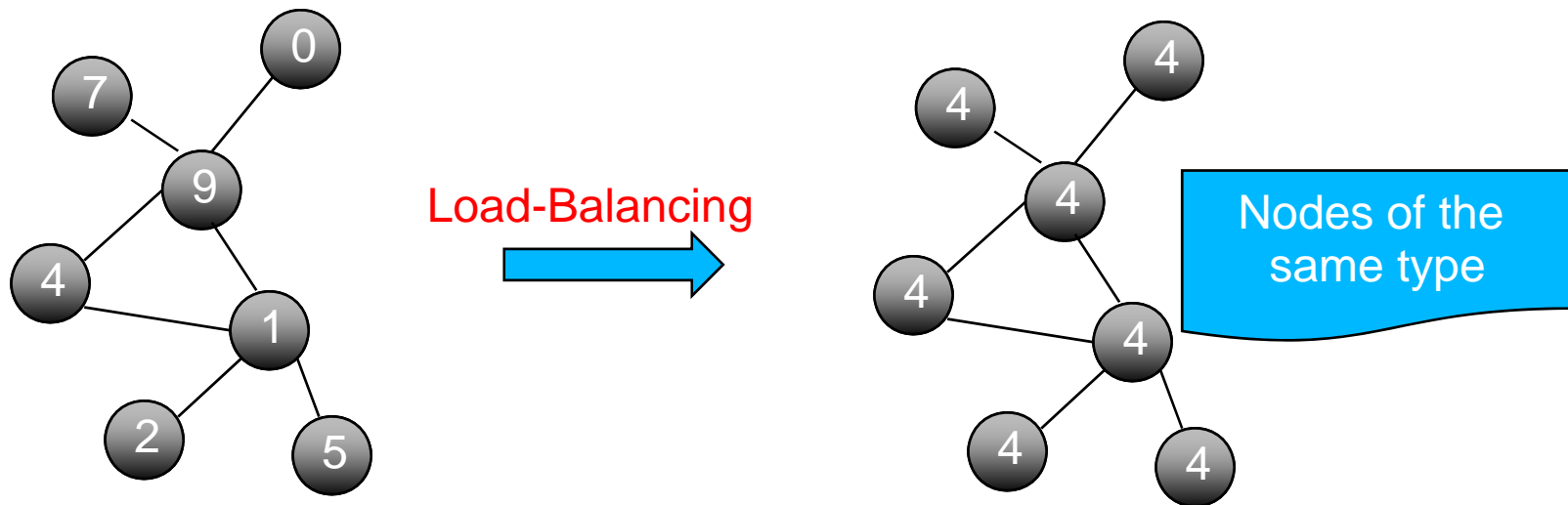
Balancing the workload of the network among the nodes in such a way to parallelize as many jobs as possible.



- Maximize $\text{throughput} = \{\# \text{ completed jobs}\} / \{\text{elapsed time}\}$
- Starting point:
 - Bio-inspired Self-Aggregation techniques;
 - Dimension Exchange Load Balancing algorithms.

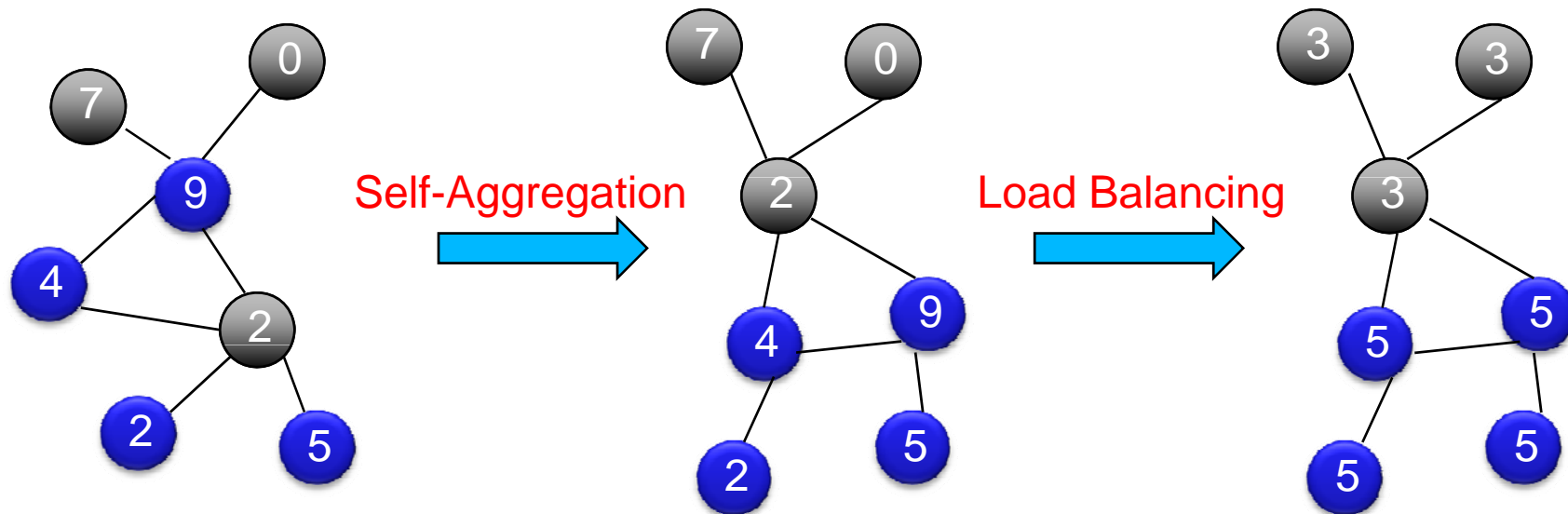
Recall on Dimension Exchange Load Balancing

The Dimension Exchange Load Balancing algorithm balances the workload of each couple of nodes until all the nodes have the same load.



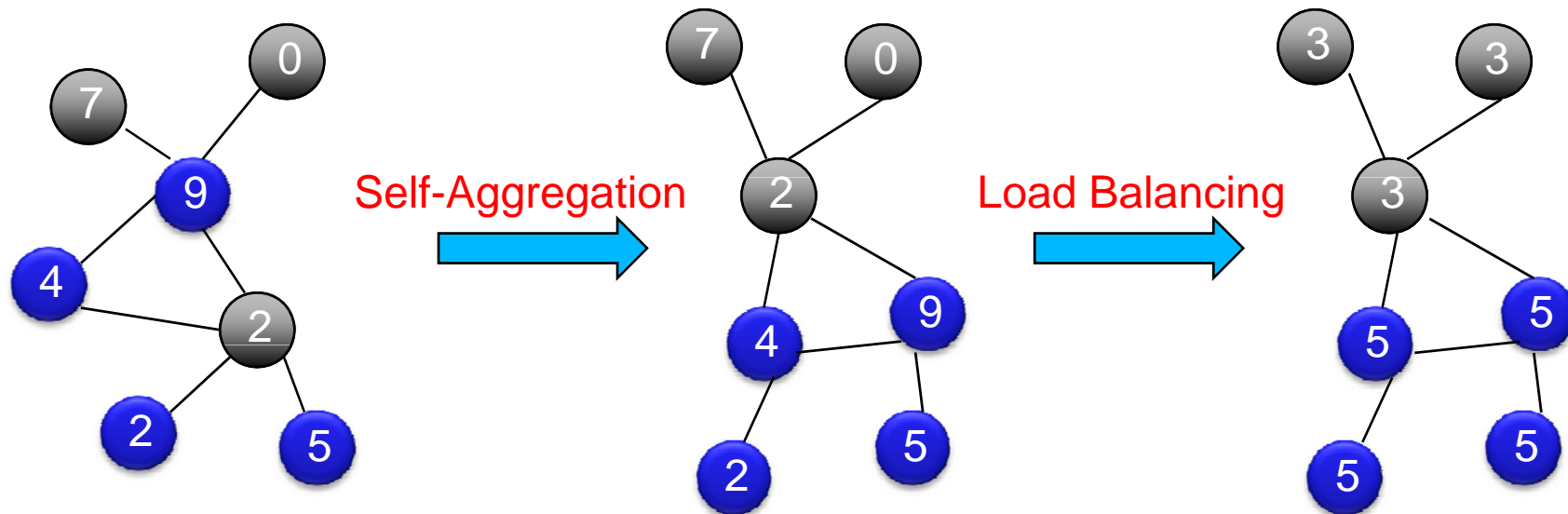
Idea: Combined Approach

1. Group nodes of the same type into homogeneous domains using a self-aggregation algorithm;
2. Balance the workload among homogeneous domains using the Dimension Exchange Load Balancing algorithm.



Idea: Combined Approach

- Both algorithms can run simultaneously without conflicting;
- Both algorithm, as other *bio-inspired algorithms* are able to create the global self-balanced pattern with just those simple local rules.



Simulation Environment

Distributed Simulator written in Java.

20 Monte Carlo Simulations.

Network of **100 nodes**, average node **degree of 4 neighbors**.

Scale-Free topology.

Heterogeneity: 10% (10 types of nodes/jobs).

Job distribution: static load of 400 jobs and continuous insertion of 400 jobs every 20s.

Node processing time: a) 100% 5s, b) 70% 7s, c) 30% 3s (ideal throughput 20jobs/s).

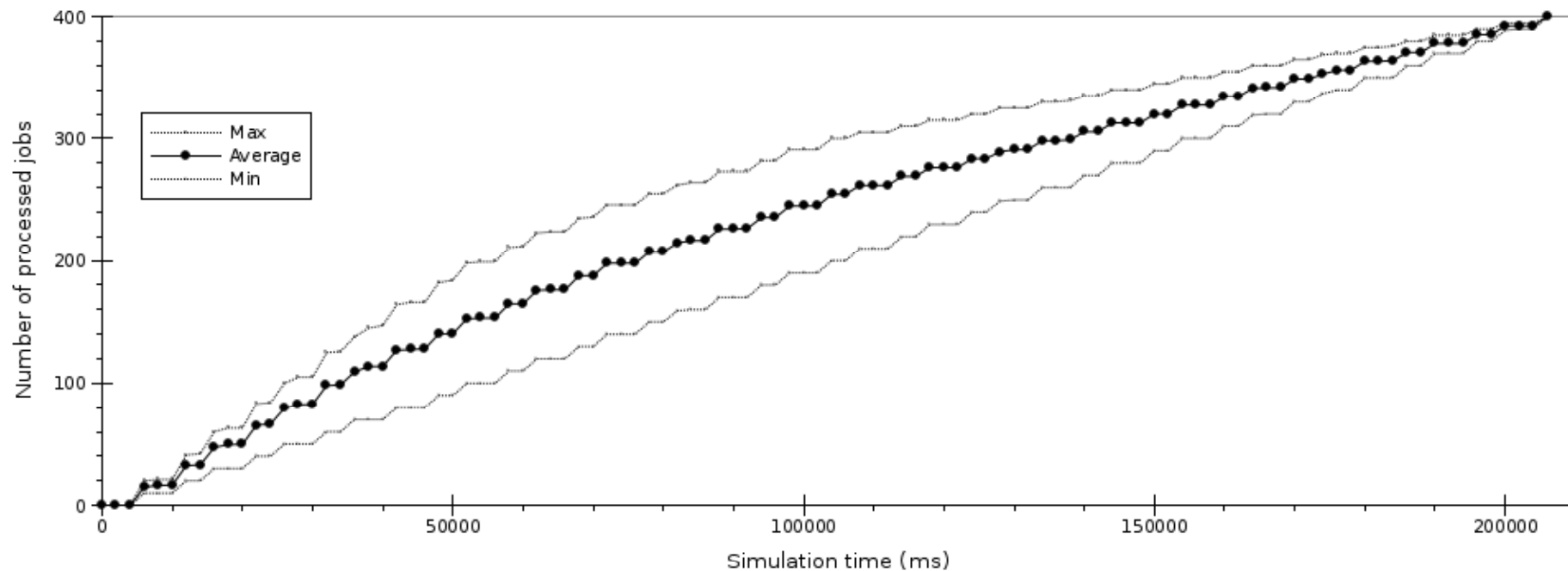
Node churn: every 10s 20% of the nodes disappears, and the same number of new nodes appears.

We evaluate the throughput and the network load in terms of message overhead.

Simulations without Rewiring

Number of processed jobs without rewiring

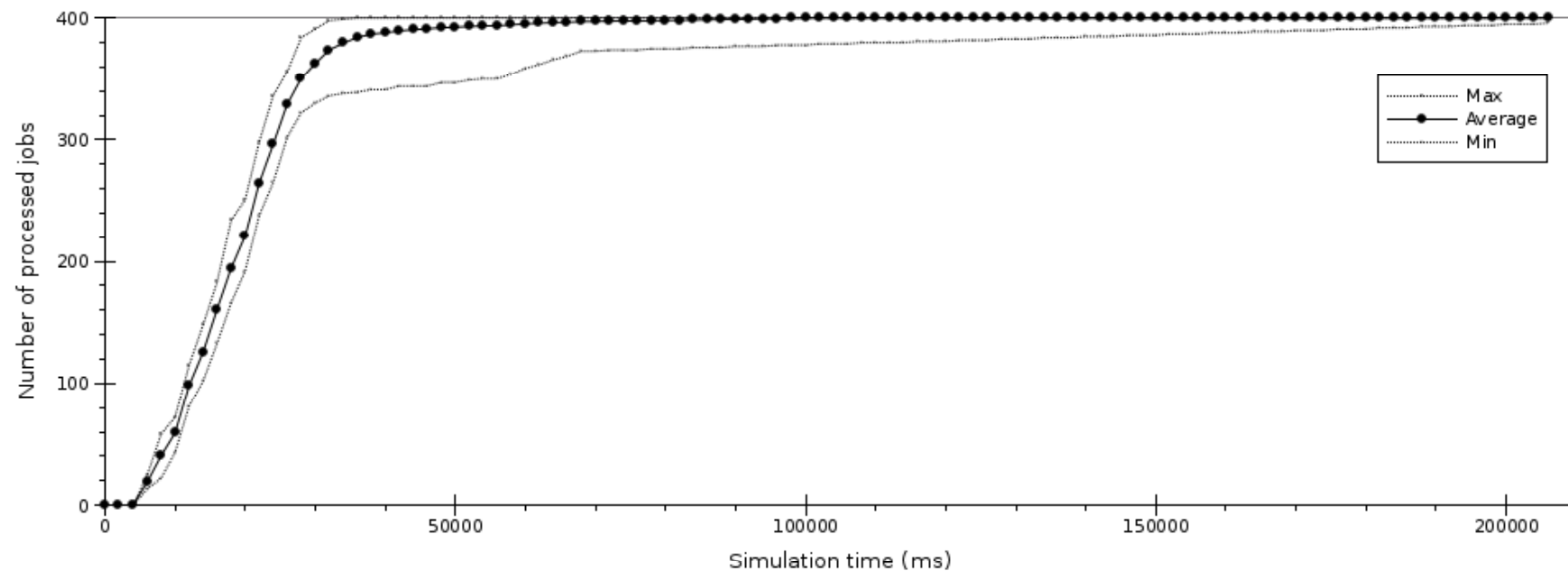
Load balancing iterations are inhibited by the fact that the jobs cannot traverse the nodes having different types.



Simulations with Rewiring: static load

Number of processed jobs with rewiring

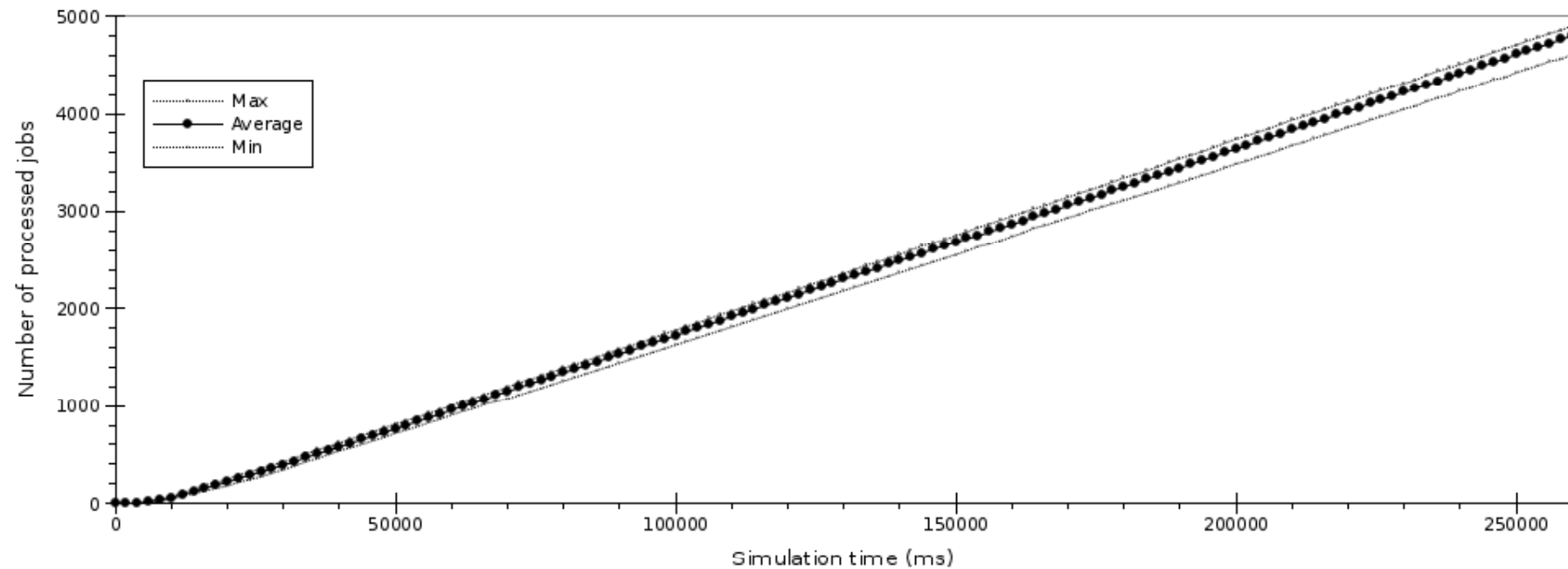
We have a strong improvement with respect to the previous experiments



Simulations with Rewiring: multiple bursts

Number of processed jobs with rewiring and multiple bursts

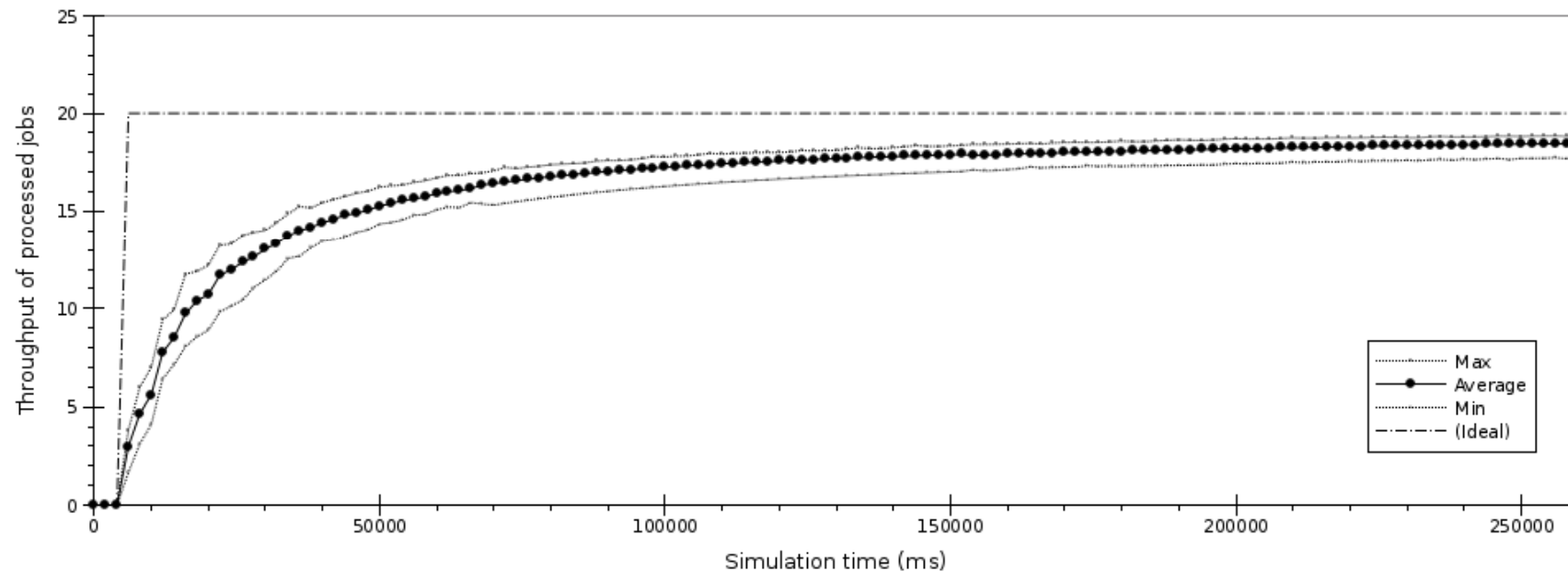
The curve grows as a straight line because of the continuous job bursts that are sent to the nodes



Simulations with Rewiring: multiple bursts

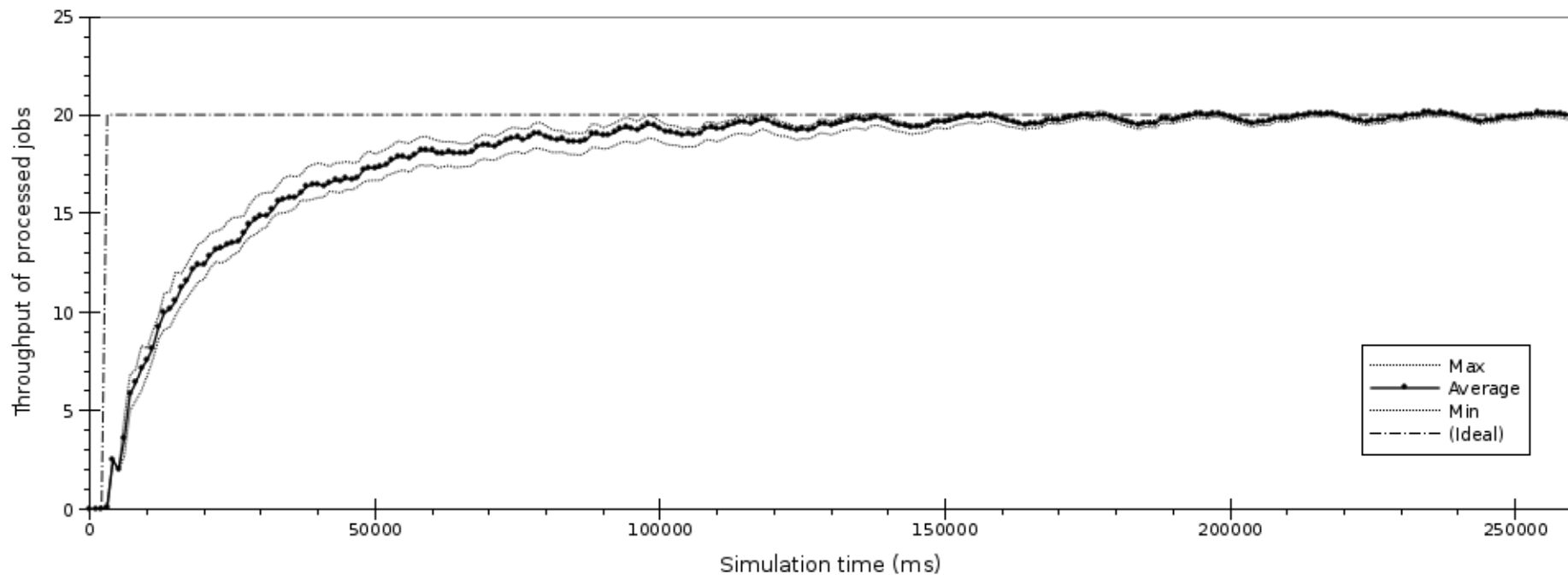
Throughput of processed jobs with rewiring and multiple bursts

The throughput is close to the optimal value (the one obtained in homogeneous networks)



Simulations: multiple bursts and churn

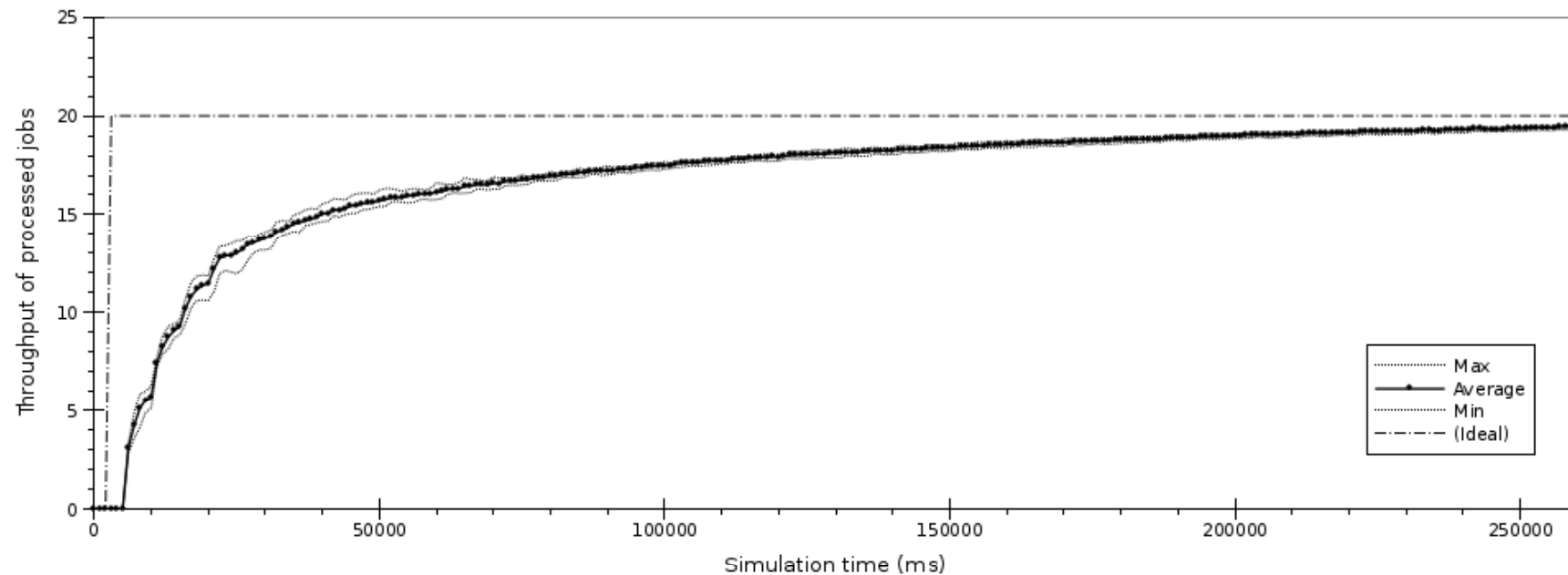
Throughput of processed jobs in presence of a node churn (arrival/departure) of 10%
The throughput is similar to the previous case, therefore it is resistant to this type of uncertainty.



Simulations: different service time

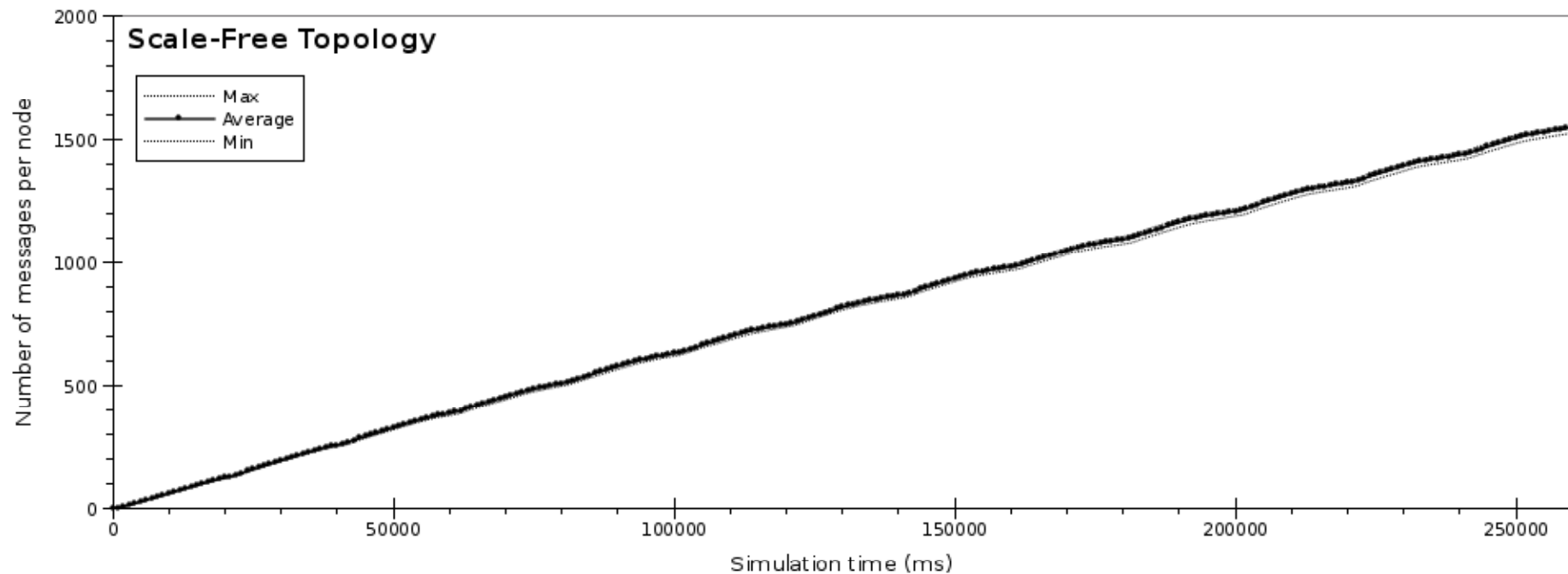
Throughput of processed jobs in presence of nodes of different processing times

Like in the previous case the combined algorithm is still able to obtain nearly-optimal values



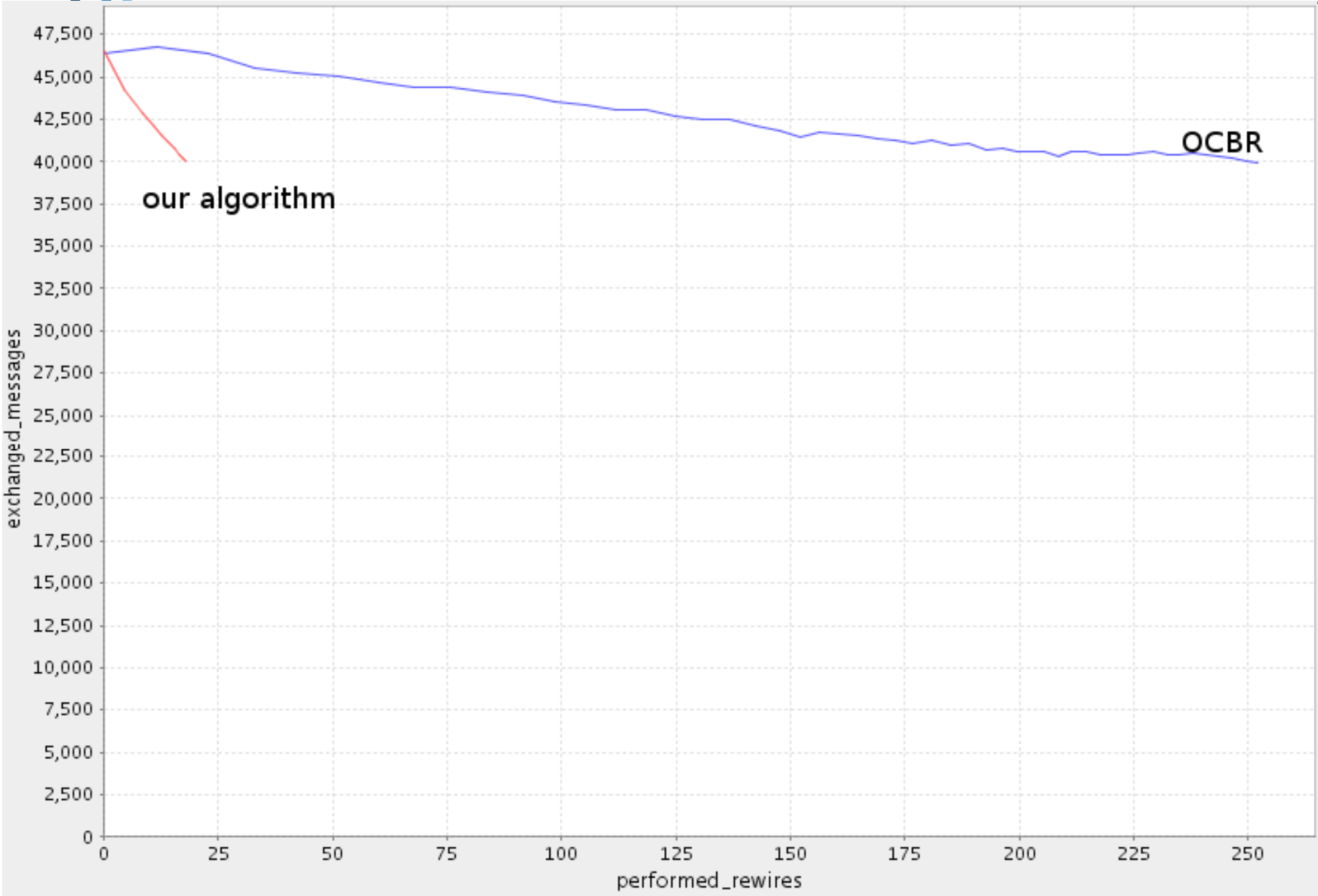
Simulations: network overhead

Message overhead (in terms of number of messages) grows linearly with time in all simulations since it is dominated by the self-aggregation algorithm (5 msgs/sec vs 0.033 msgs/sec).

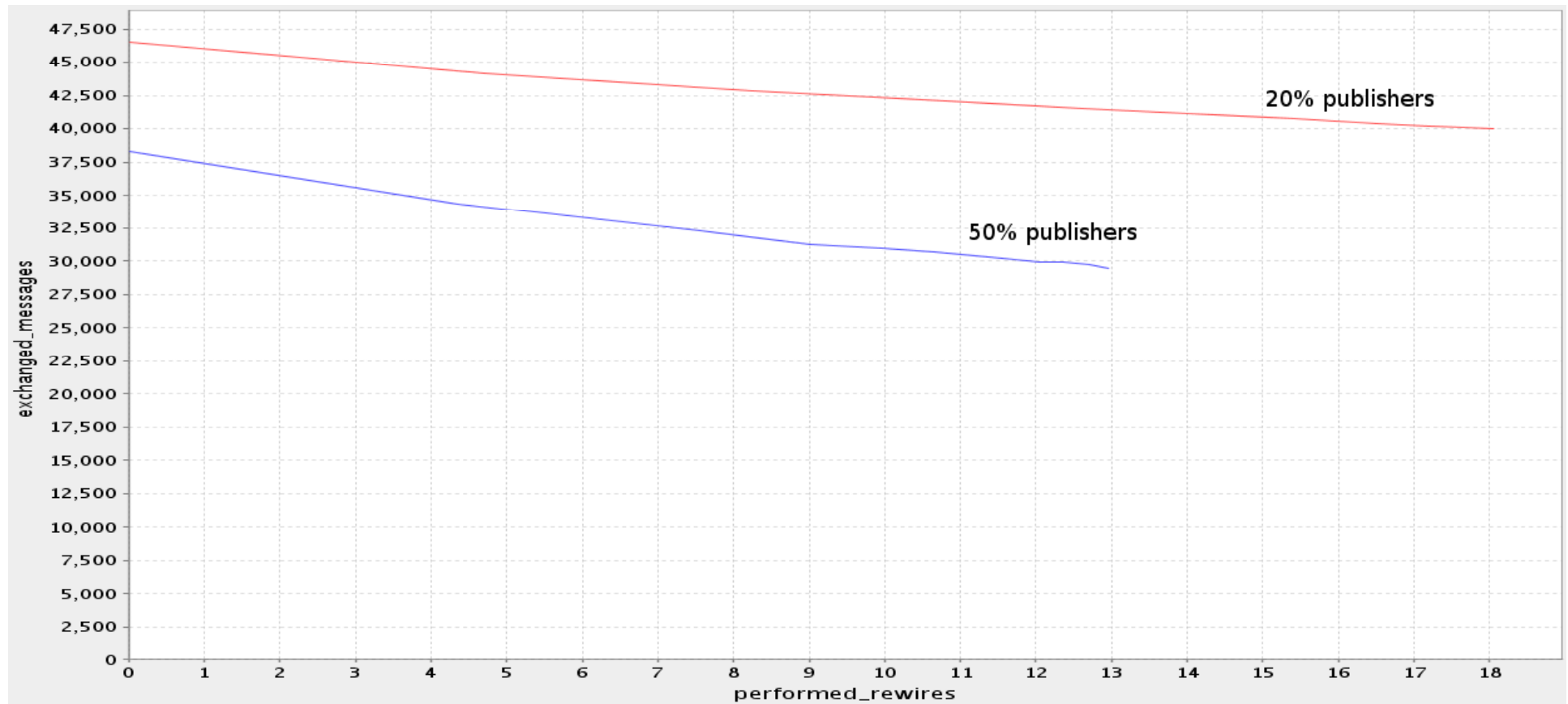


Possible Applications (2)

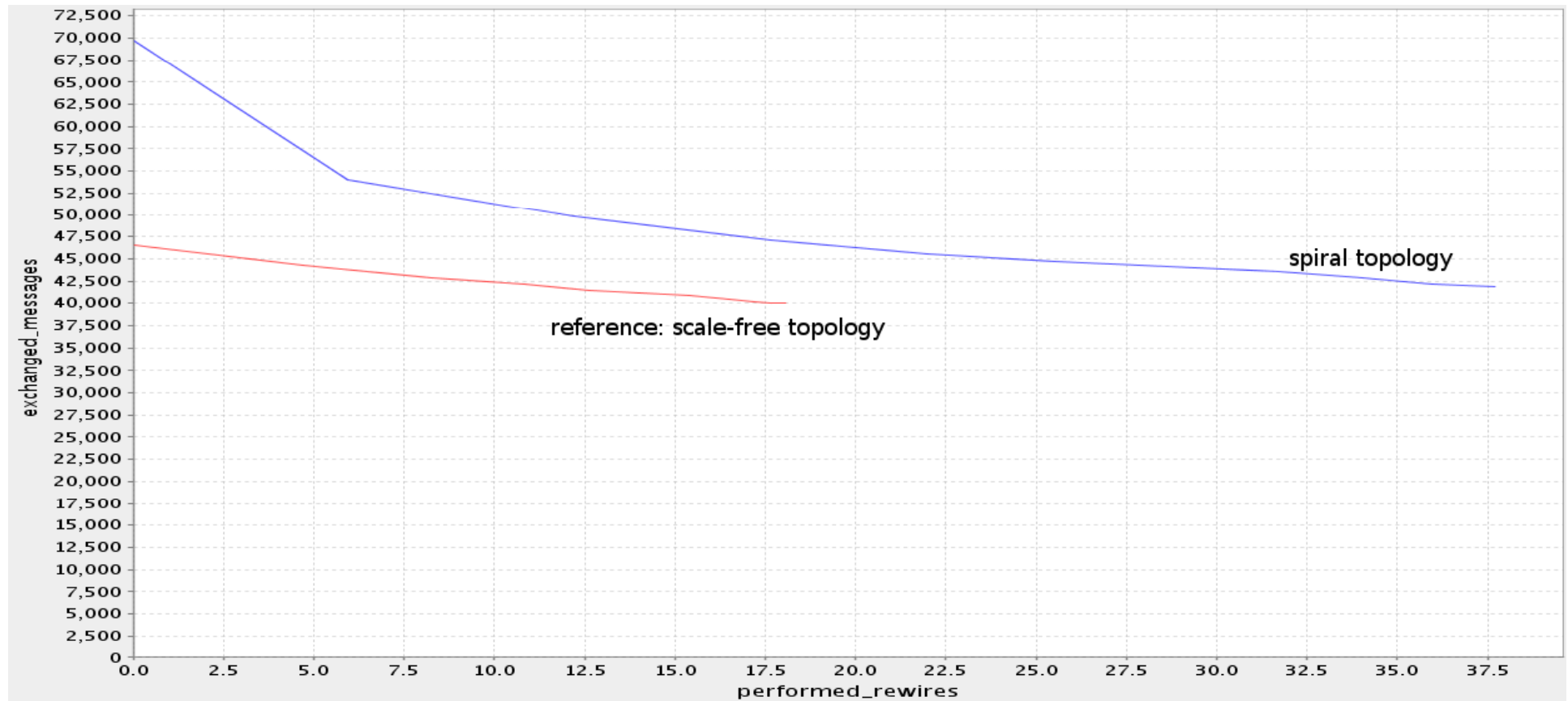
- Overlay Self-Organization in Publish-Subscribe middleware:
 - Rewire the broker connections in order to minimize network load.
 - Group together subscribers interested on the same topics.



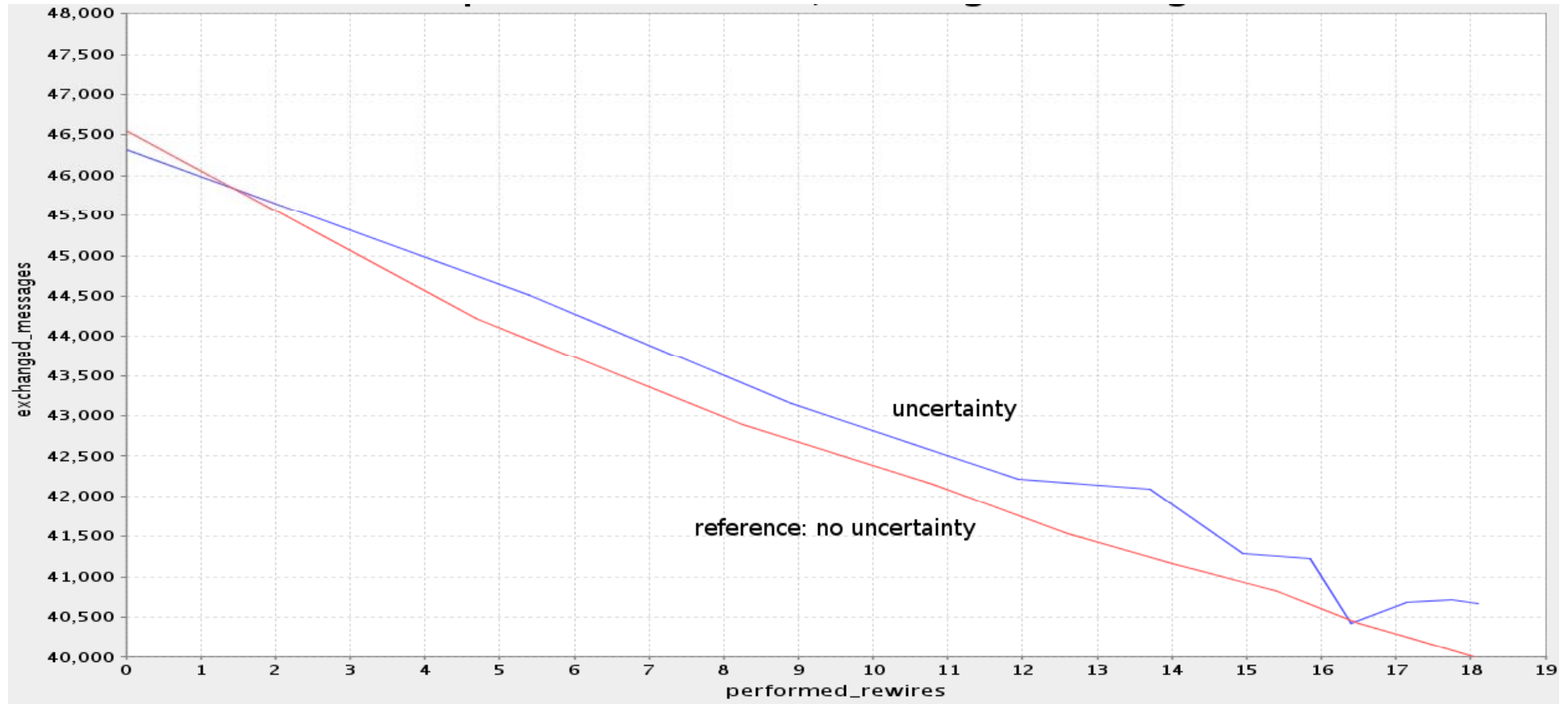
Simulation Results (2)



Simulation Results (3)



Simulation Results (4)



Conclusions...(so far)

- CASCADAS aims at offering a blend of architectural, and bio-inspired approaches to develop various classes of autonomic systems
- Main focus is on open, pervasive systems
- In this context self-organization is essential
- Various open points
 - Development of libraries of self-organization algorithms is on going
 - We need to identify proper case studies that show their potential
 - We plan to assess the suitability of algorithms for networks of ACEs hosted on small devices (tinyACEs)

Bibliography

- P. Horn; "Autonomic Computing: IBM's Perspective on the State of Information Technology"; Technical Report, IBM Corporation, October 15, 2001.
- O. Babaoglu et al, "Design patterns from biology for distributed computing," Proceedings of the European Conference on Complex Systems, November 2005.
- L. Baresi, E. Di Nitto, C. Ghezzi, Developing an Open-World Software Paradigm, IEEE Computer, October 2006.
- S. Bouchenak, N. De Palma, D. Hagimont, and C. Taton. Autonomic management of clustered applications. In IEEE International Conference on Cluster Computing, pages 1-11, Barcelona, Spain, 25-28 Sept. 2006.
- S. Hariri et al, "The Autonomic Computing Paradigm," Cluster Computing: The Journal of Networks, Software Tools, and Applications, Kluwer Academic Publishers, Vol. 8, No. 5, 2006.
- J.L. Hellerstein. Self-managing systems: a control theory foundation. Local Computer Networks, 2004. 29th Annual IEEE International Conference on, pages 708-715, 16-18 Nov. 2004
- M. C. Huebscher and J. A. McCann A survey of Autonomic Computing: Degrees, Models, and Applications, ACM Computing Surveys, Vol. 40, No. 3, Article 7, Publication date: August 2008.
- G. Kaiser, J. Parekh, P. Gross, and G. Valetto. Kinesthetics extreme: an external infrastructure for monitoring distributed legacy systems. In AMS: Autonomic Computing Workshop, pages 22-30, Seattle, WA, USA, 2003.
- J. Kephart and D. Chess, "The Vision of Autonomic Computing," IEEE Computer 36(1): 41-50 (2003).

Bibliography

- JKephart, J.O., "Research challenges of autonomic computing," Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on , vol., no., pp. 15-22, 15-21 May 2005.
- Montresor A. Anthill: a Framework for the Design and the Analysis of Peer-to-Peer Systems, 2001
- M. Parashar and et al. Automate: Enabling autonomic applications on the grid. Cluster Computing, 9(2):161-174, 2006.
- G. Tesauro, D.M. Chess, W.E. Walsh, R. Das, A. Segal, I. Whalley, J.O. Kephart, and S.R. White. A multi-agent systems approach to autonomic computing. AAMAS 2004. Proceedings of the Third International Joint Conference on, pages 464-471, 2004.
- T. Nakano and T. Suda. Applying biological principles to designs of network services. Appl. Soft Comput., 7(3):870-878, 2007.
- F. Saffre, R. Tateson, J. Halloy, M. Shackleton, and J. L. Deneubourg. Aggregation Dynamics in Overlay Networks and Their Implications for Self-Organized Distributed Applications. The Computer Journal, 2008.
- E. Di Nitto, D. J. Dubois, and R. Mirandola. Self-Aggregation Algorithms for Autonomic Systems. In Proceedings of Bionetics I07, Budapest, Hungary, December 2007.
- E. Di Nitto, D. J. Dubois, R. Mirandola, F. Saffre, and R. Tateson. Self-Aggregation Techniques for Load Balancing in Distributed Systems. In Proceedings of SASO 2008 - Poster presentation –
- E. Di Nitto, D. J. Dubois, R. Mirandola, F. Saffre, and R. Tateson. Applying Self-Aggregation to Load Balancing: Experimental Results. In Proceedings of BIONETICS 2008