# A Responsive Multi Agent System (MAS) Framework for Optimizing Distributed Network

Manish Sharma
Assistant Professor
IT Graphic Era University
Dehradun, Uttarkhand, India

## ABSTRACT
Multi Agent System introduces an approach that monitoring and response out into the network in support of better scalability and decision building. We depend more and more on computer networks, yet the expansion of networks and their heterogeneous composition make ensuring network reliability a scary task. Using self-directed MAS, the system detects and responds to network degrading events, even those not previously experiential, and even when parts of the network have failed. This paper describes system architecture and gives a case of how this system might achieve.

## General Terms
MAS (Multi agent System), Artificial Intelligence, Prometheus, Subsumption architecture[9].

## 1. INTRODUCTION
Multi Agent Systems [1][2]([Shoham and Leyton-Brown 2008]) are more and more important and active area of inter-disciplinary research on the border of computer science, artificial intelligence, and game theory, as they model a wide variety of phenomena in these fields, including open and interactive systems, distributed computations, security protocols, knowledge and information exchange, etc. Not surprisingly, a number of logical recognized systems have been proposed for specification, verification, and analysis about Multi Agent Systems (MAS).
These recognized systems, broadly speaking, fall into two types: those for analysis about knowledge of agents and those for analysis about abilities of agents

## 2. MOTIVATIONAL SCENARIOS
The task of maintaining network health is compounded by the difficulty of exactly diagnosing problems once symptoms are observed. There is no exact correspondence between network problems and their underlying causes. Faults, attacks and mistreatment may manifest themselves in a variety of ways, and observable symptoms may have a number of possible causes. Problems may be irregular and difficult to consistently reproduce. Relatively minor faults can persist unnoticed, exacerbating and masking the causes of larger events that might occur.

As these issues point toward, computer networks are equivalent to other complex systems, such as automobile engines and industrialized processes, requiring automatic analysis and control in order to remain practical and dependable. In this proposal we present a work in improvement, the Multi-Agent System for Network Resource Reliability.

## 3. PAPER OBJECTIVE AND APPROACH
This system is planned specifically to address the following needs of currently available approaches [8]:

### 3.1 Scalability
Cooperating agents each monitor a part of the network and share information as needed, dropping unneeded messages and eliminating the processing and data transfer bottleneck of a centralized system.

### 3.2 Interoperability
Agents themselves are platform-independent and offer a consistent interface to the administrator. Platform and device specific modules allow an agent executing in one environment to monitor or act upon network elements of different types and having proprietary management interfaces.

### 3.3 Integration of security and management
MAS agents recognize any number of network-degrading events and are able to respond to faults and miss configurations as well as intrusions and attacks. While MAS is not itself an IDS, agents can reason about the output of an IDS (or several different IDS systems) for improved diagnosis and response.

## 4. SUBSUMPTION ARCHITECTURE
Subsumption architecture [9] is a reactive automaton architecture closely associated with behavior-based robotics. [9] Subsumption has been generally dominant in autonomous robotics and in another place in real-time AI [9].

Sensors -> Perception -> Modeling -> Planning -> Task recognition-> Perform

The architecture describes an architectural pattern for the decision making of a single agent. The architecture is organized as a series of parallel working layers each layer is responsible for a specific behavior of the agent. The priority of layers (behavior) increases from bottom to top. Higher layers are able to inhibit lower layers, giving priority to more important behavior. Fig 1 shows Subsumption layered architecture.

Subsumption architecture has used for simple MAS that has to collect packets and deliver them at a destination. MAS must avoid obstacles in the environment. A layer in the architecture directly connects perception to action by means of a finite state machine augmented with timing elements. Each layer

collects its own sensor data that is written in registers. The arrival of specific data, or the expiration of a timer, can trigger a change of state in the interior finite state machine and possibly produce output commands to actuators. Reserve mechanisms resolve conflicts between actuator commands from different layers. The Subsumption architecture pattern allows the design of very efficient agents. However, Subsumption architectures are hard to build for complex agents that have to operate in complex environments. The MAS pattern has successfully been used in many practical MAS.

Popular frameworks such as Jade [10] and Jack [11] have a relative narrow view on middleware support for agent-based systems and basically provide infrastructure for communication or a broker infrastructure. Common middleware services such as security, persistency, and transactions are often considered minimally in multi-agent system development.

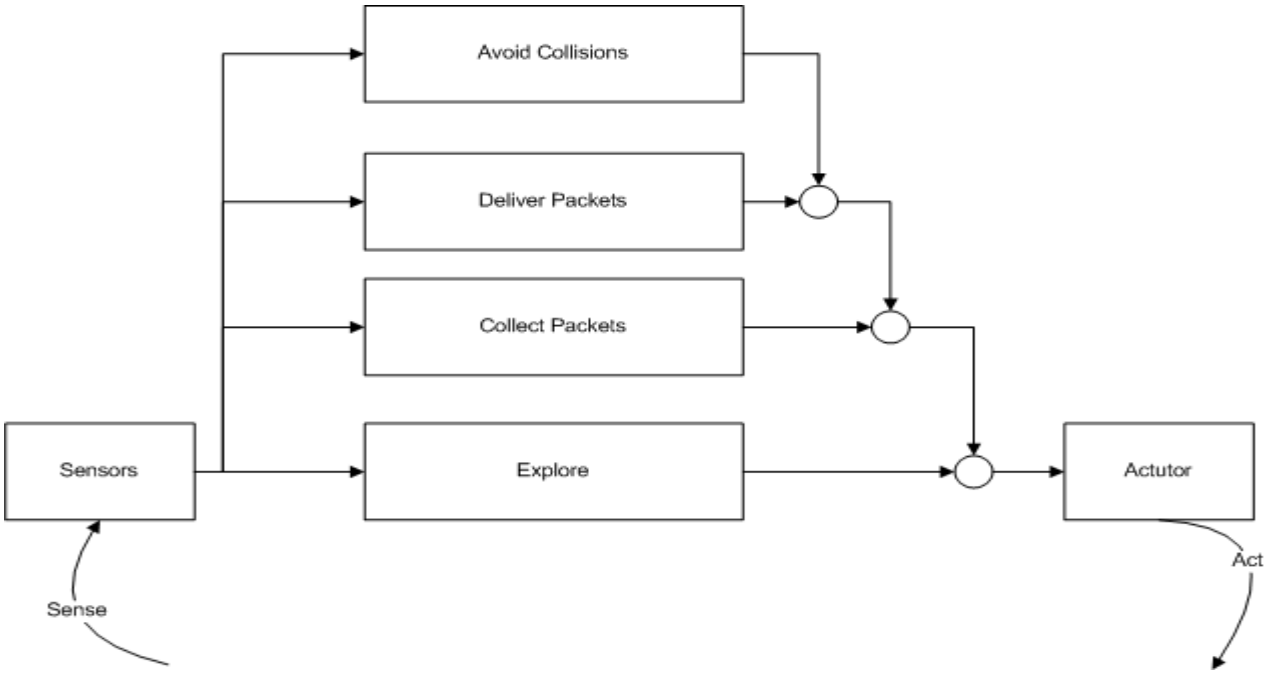


**Fig 1: Architecture for a simple Agent**

## 5. MULTI AGENT SYSTEM

Multi-agent systems (MAS)[1][2][5][9] have been deployed in several domains such as concurrent engineering, knowledge management, communications, air traffic control/flow, space exploration, or e-commerce. They can be used to intelligently assist users in specialized or generic tasks. Specialized tasks include, among others, network management. Generic tasks include handling information (e.g. retrieving, filtering, synthesizing), making decisions (decision support systems) or capturing lessons learned by a project team.

### 5.1 The characteristics of MAS [13] [7]

5.1.1 Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint [7]

5.1.2 There is no system global control

5.1.3 Data are decentralized

Based on the work of Durfee and Lesser, Jennings et al. and Sycara define a MAS "as a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capabilities or knowledge of each problem solver."[13] The problem solver mentioned in the definition is an agent. Sycara K. also describes the abilities presented by multi-agent systems that make them an interesting research subject.

The Prometheus methodology used to implement the Subsumption architecture [9].

## 6. OVERVIEW OF THE PROMETHEUS

The Prometheus methodology [3] defines a detailed process for specifying, designing, implementing and testing/debugging agent-oriented software systems. In addition to detailed processes (and many practical tips), it defines a range of artifacts that are produced along the way. Some of these artifacts are kept, and some are only used as 'stepping stones'. Some of the artifacts are graphical while others are structured text (i.e. forms).

For example, actions and percepts are captured in the system specification phase; the detailed design phase results in plans,

events and beliefs; and the entities used in the various overview diagrams correspond directly to the concepts.

Note that all of the artifacts are structured. This is important in order to be able to provide tool support for the methodology. The Prometheus methodology consists of three phases, depicted in Fig 2.

6.1.1 The system specification phase focuses on identifying the goals and basic functionalities of the system, along with inputs (percepts) and outputs (actions).

6.1.2 The architectural design phase uses the outputs from the previous phase to determine which agent types the system will contain and how they will interact.

6.1.3 The detailed design phase looks at the internals of each agent and how it will accomplish its tasks within the overall system.

6.1.4 A fourth phase is implementation, which is omitted from Figure because its details depend on the implementation platform chosen.

The above description of these phases is intended to give a rough feel for the overall structure of the methodology, so that when reading the paper, where Prometheus is described in detail, you have some idea of how the details fit into the bigger picture.

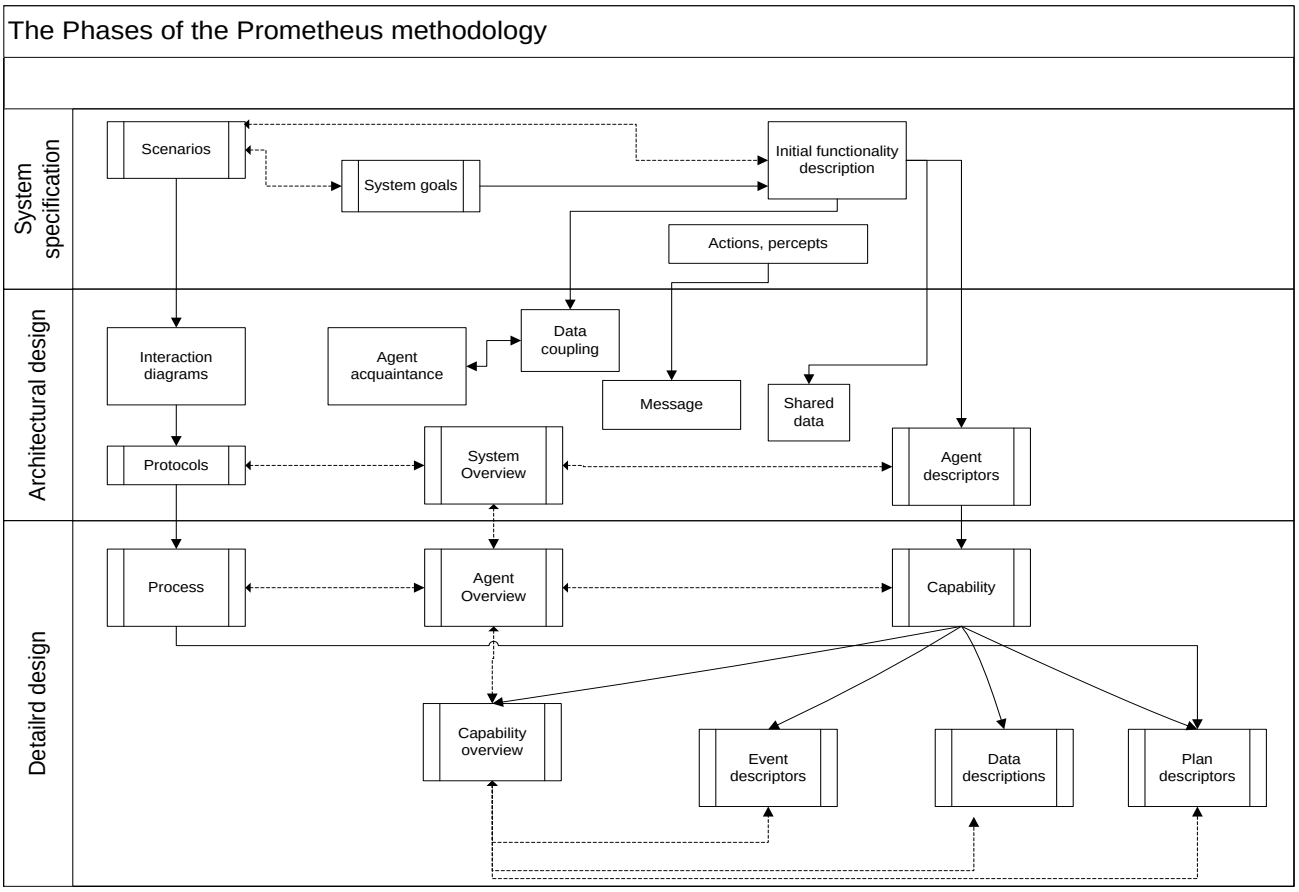


**Fig 2: The phases of the Prometheus methodology [3]**

The above description of these phases is intended to give a rough feel for the overall structure of the methodology, so that when reading the paper, where Prometheus is described in detail, you have some idea of how the details fit into the bigger picture.

## 6.2 Methodology for Optimization

Prometheus differs significantly from object-oriented methodologies include the following:

6.2.1 The provision of a process for determining the types of agents in the system.

6.2.2 Treating messages as components in their own right, not just as labels on arcs. This allows a message (or an event) to be handled by multiple plans, which is crucial to achieving flexibility and robustness.

6.2.3 Distinguishing percepts and actions from messages, and looking explicitly at percept processing. Agents are situated in an environment, and it is important to define the

interface between agents and their environment. Percept processing is often important for agents that are situated in the real world and take their percepts from noisy devices such as video cameras.

6.2.4 Distinguishing passive components (data, beliefs) from active components (agents, capabilities, plans): with object-oriented modeling, everything is modeled as (passive) objects.

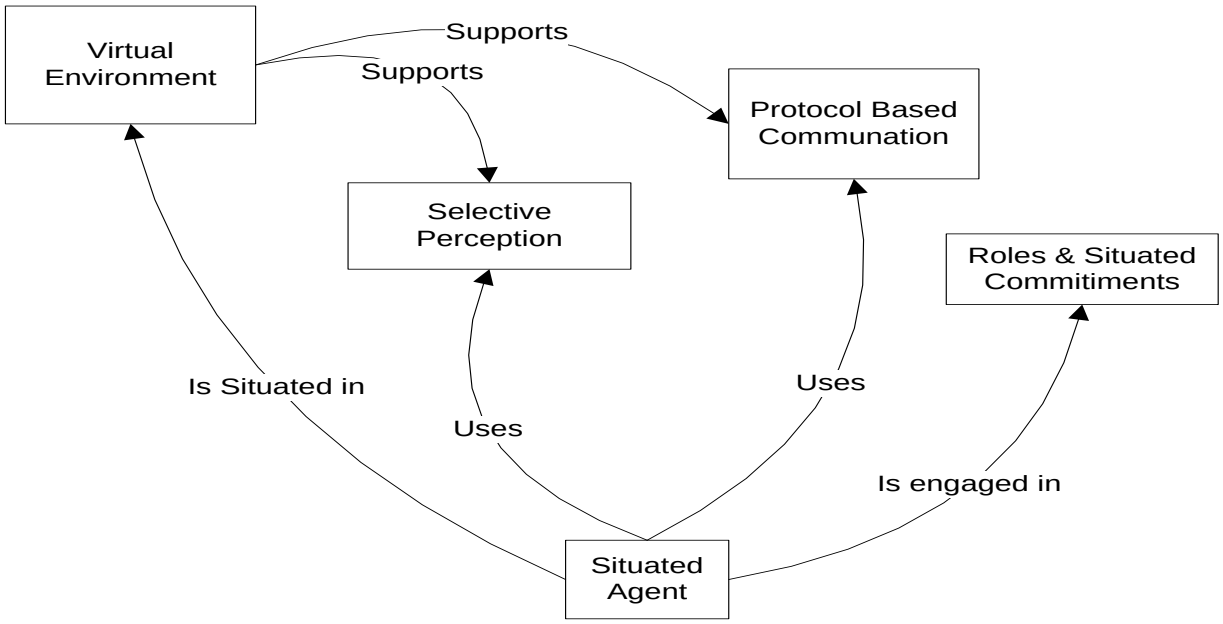


**Fig: 3 Situated Multi-Agent System Patterns**

Fig 3 shows a general overview of the pattern for situated multi-agent systems with the most important associations between the proposed patterns. A situated agent is an autonomous problem-solving body in the system. An agent encapsulates its state and controls its behavior. The responsibility of an agent is to achieve its design objectives, i.e., to realize the application-specific goals it is assigned. Agents are able to adapt their behavior according to the changing conditions in the environment. A situated agent is a helpful entity. The overall application goals result from interaction among agents, rather than from complicated capabilities of individual agents. Agents are situated in a virtual environment. The virtual environment maintains a virtualization of the relevant parts of the world and serves as a coordination medium for the agents, i.e., the virtual environment mediates both the interactions among agents and the access to resources.

## 7. CONCLUSION

In this paper we have described MAS, a framework for optimized distributed network based on the Subsumption architecture, and shown how it is optimized for scalability, Interoperability, Integration of security and management with the help of Promethues mythology for designing and implementations. The simplicity of this design has allowed us to create several identical MAS for different issues in network management.

## 8. ACKNOWLEDGMENTS

This research work is carried out with valuable support by Graphic Era University, Dehradun, Uttarakhand, India.

## 9. REFERENCES

[1] Len Bass “Architecture-Based Design of Multi-Agent Systems” Springer 2010 , ISBN 978-3-642-01063-7

[2] Michael Wooldridge (University of Liverpool) “An Introduction to MultiAgent Systems, 2nd Edition” 2009, Wiley Paperback ISBN 978-0-470-51946-2

[3] Lin Padgham & Michael Winikoff RMIT University, Melbourne, Australia Developing “Intelligent Agent Systems A practical guide” Wiley ISBN 0-470-86120-7 (HB)

[4] Mehdi Dastani, Koen V. Hindriks, John-Jules Charles Meyer(1st Edition., 2010, XVII,) “Specification and Verification of Multi-agent Systems” ISBN 978-1-4419-6983-5

[5] Santhana Chaimontree, Katie Atkinson, Frans Coenen (2011). A framework for Multi-Agent Based Clustering. Published online 1 December2011 Springer Autonomous Agents and Multi-Agent Systems Volume 24/2011. ISSN: 1573-7454

[6] Gilson Yukio Sato, Hilton José Silva de Azevedo, Jean-Paul A. Barthès(2011). Agent and multi-agent applications to support distributed communities of practice: a short review. Published online 05 April 2011 Springer Autonomous Agents and Multi-Agent Systems Volume 24/2011. ISSN: 1573-7454

[7] K. P. Sycara. Resolving goal conflicts via negotiation. In Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88), St. Paul, MN, 1988.

[8] V.K. Verma, R.C. Joshi, B. Xie, D.P. Agrawal, "Combating the bloated state problem in mobile agents based network monitoring applications", Computer Networks 52 (17), pp. 3218-3228, December 2008.

[9] Brooks, R. (1986). "A robust layered control system for a mobile robot". Robotics and Automation, IEEE Journal of [legacy, pre-1988] 2 (1): 14–23. DOI:10.1109/JRA.1986.1087032. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1087032. Retrieved 2008-04-14.

[10] D. Greenwood, G. Caire, F. Bellifemine, Developing Multiagent Systems with Jade. Agent Technology (Wiley 2007)

[11] M. Wood, S. DeLoach, in An Overview of the Multiagent Systems Engineering Methodology. Agent-Oriented Software Engineering I. Lecture Notes in Computer Science, vol. 1957 (Springer, Heidelberg, 2000)

[12] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice-Hall, 1995.

[13] Katia P. Sycara Multiagent System (AAAI-1995)