ATTENTION-FOCUSING ARCHITECTURE FOR SCALABLE NETWORK-SYSTEMS USING DEVS FORMALISM

By

Saurabh Mittal

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN COMPUTER AND ELECTRCICALENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

2003

STATEMENT BY AUTHOR

This thesis has been submitted in the partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotations from the thesis are allowable without special permission, provided that accurate acknowledgment of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate college when in his or her judgement the proposed use of material is in the interests of scholarship. In all other instances, permission must be obtained from author.

SIGNED: _____

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

_____ _____

Prof. Bernard P. Zeigler Date
ACIMS Center, Electrical and Computer Engg.
University of Arizona

ACKNOWLEDGEMENTS

I would like to thank Professor Bernard P. Zeigler, my advisor, for his invaluable guidance and moral as well as financial support all through the MS studies. He gave the freedom to explore uncharted waters and provided the encouragement to complete this endeavor. I also would like to express my appreciation to Professor Srinivasan Ramasubramanian whose suggestions helped in improving the thesis content, and Professor Salim Hariri, who provided me the with the knowledge in networking domain through various discussions, and both of whom served on my defense committee.

My sincere thanks to our family friend, Sunita Suri – Sports teacher at my High School, for sponsoring my studies and V. P. Gupta – General Manager, State Bank of India, without whose timely financial advice and help, it would be impossible to embark upon this task. I shall be forever indebted towards their generous contribution.

I would like to express my gratitude to my family back at home that includes my parents, my brother Vaibhav, Vandana mamiji, my sisters and my grandfather Moti Ram Garg, who taught me the meaning of life and that simplicity and hardwork can accomplish any dream that I could envision.

And, above all, I would like to express my thanks to God, who made it all possible.

To,

Madhu, Anand, and
Moti Ram Garg

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

This research investigates the application of Modeling & Simulation as a potential tool in the design and development of scalable Intelligent Systems. With the evolution of Internet, scalability has gone to new dimensions and the journey isn't complete yet. The current tools aren't equipped to study evolving networks, as the software architecture laid underneath is not scalable and evolvable. This thesis outlines an architecture that is both scalable and evolvable. The architecture is built on Discrete Event System Specifications (DEVS) Formalism and is implemented in Java. The example considered in this research is that of Homeland Defense Intelligence Surveillance and Reconnaissance. The inspiration has been taken from Neuroscience and Brain Cognitive Science. The architecture is named *AdSNet* implying Adaptive Sensor Network. It is designed to display intelligent behavior and provide dynamic allocation of resources e.g. bandwidth, channel capacity, to numerous computing decision-making components that are laid out in a hierarchical manner. The smallest computing entity in the system is a Sensor that has the capability to detect and sample information inside a surveillance area. Its ability to detect any abnormal activity is based on its sensitivity, which is tunable. The sensor follows the activity occurring in the area assigned to it and its sampling rate increases if there is high activity and it decreases if the activity becomes low. The surveillance area, assigned to a sensor, is designed to be a random phenomenon incorporating random loss factor but displaying qualitative behavior. It's a coupled model, composed of DEVS[1]-

[1] Discrete Event System Specifications

cells (finite state machines) arranged in grid fashion. The architecture is modeled along the geographical distribution of area and at the top level of the hierarchy is a State that constitutes many counties, which constitutes many cities, which then is divided into surveillance areas. The communication between these levels is filtered and condensed as it travels up the hierarchy. The architecture is a multi-level hierarchically organized system and its mapping to geographical area distribution is one example of its application. It is capable of focusing attention to highly active components and allocating resources to them and withdrawing resources from components that show under-activity. The capability of system to evolve has been modeled by addition of new sensors during high activity and removal of sensors corresponding to low activity, where it is not predicted beforehand when the period of high activity is scheduled in Surveillance areas. The resulting *AdSNet* Framework enables system architects and network designers to study the pre-design issues concerning the scalable-evolvable networks and of hybrid-autonomous systems. As the framework itself is scalable and evolvable, it provides a tool to model and simulate the growth in any evolvable network composed of heterogeneous components.

1. Introduction

With the Internet expected to touch 1 Billion nodes in the near future, scalability has achieved new dimensions. Nobody ever envisioned that ARPANET [42] which was designed as a test network between 5 Universities will evolve towards this huge network, shrinking this globe into a village. As this network evolved, so did the discipline of Computer Engineering specializing into a new branch of Computer Networks that encompassed research areas like Queuing Theory, Distributed Computing, Network Traffic Engineering etc. Initially when the networks were small, they were studied easily in the labs, partly by creating prototype networks and partly by modeling and simulation. The earlier simulators like NS-2 [43] and later Opnet [44] proved to be an ideal platform to test small networks. Now, as the network systems are growing exponentially, these static topology-generating simulators are ill equipped and are unable to predict the performance of such evolving systems accurately. Their inability may be attributed to insufficient understanding of these evolvable networks and also by the architectural limitations inherent in these simulators. Although there are simulators like Glomosim [45], [46] pdns [48], etc. that can model hundreds of thousand of nodes but none is able to model a network that is evolvable, resizable and adaptive. Moreover, most of the architectures are flat in operation i.e. all components at the same level with no abstractions and hierarchies, and generate great overheads when it comes to message passing as the messages are at the packet level resolution that is the lowest resolution possible. The comparable analysis of these simulators can be seen at [22].

In the league of large scale, simulation architectures SSFNet [47] and DEVS-DOC [49], [50] are the only architectures that are truly scalable and hierarchically constructed but these two are also not capable to study the evolving nature of a network-system. As a result, they do not have the functionality to study the scalability-evolvability aspect of the system.

1.1 Scalability

So, how does one define a Scalable System? Is it just the increase in number of components of a system? What if each component of the system is intelligent and has the capability to take decisions independently? How is a "**Scalable Intelligent System**" defined?

According to Zeigler [1],

> "A Scalable Intelligent System is a system that retains or evolves
> its capability as its size increases dramatically."

Likewise, a scalable software system is a system that retains its basic services and the inner components' qualities, as the number of its components increases, for example, a million. Scalability focuses on the architectural traits of a system that allow scaling up of

its basic information processing algorithms to continue to work, or to even work better, as it grows by orders of magnitude. One such attempt to model a scalable Network System ([21],[23]) was made as a part of this research.

Consider the example of Internet. Its current architecture doesn't support scalability, but still its becoming ubiquitous. Its success is attributed to different factors but the failures that have occurred in the past are due to the architectural makeup that have stalled the Internet (for ex. Code Red Virus), which is flat. The point in consideration is that as the Internet grows and increases in size, the Internet architecture requires disproportionate amount of house-keeping i.e. maintenance of link information at routers, which is generally more than the necessary house-keeping when compared with the hierarchical architecture. This was used when foundations of Internet were being laid. This housekeeping cumulatively adds up at the router, overloading it and finally resulting in its crash or slower end-to-end response. The information flows through whole of the network like a ripple thereby forcing each of the router to update its routing table. If this router happens to be at the Backbone of the Internet and it crashes due to overload, its periodic crashing can put the routing table of the entire backbone in a route-flap storm which is triggered by its advertisement of "I'm dead…I'm alive". This kind of behavior that occurred at the backbone of the Internet has not occurred after 2001 (Code Red and Nimda Worms.) Its degree of propagation was upto the lowest possible level i.e. home PC. There is no hierarchy in the Internet architecture that could prevent and filter out the

information that was propagated from the backbone to the end-user (from the /8 CIDR block to /24 CIDR block [27], [28]).

Some other examples of Scalability from real life [1] are depicted in Figure 1. The y-axis denotes the number of nodes and increasing complexity. It represents four systems:

a) The Biological Brain
b) Human's Knowledge System
c) E-Commerce
d) Homeland Defense

Its not surprising that most of the systems mentioned above are hierarchically organized. Nature supports hierarchical architecture when considered the example of the biological brain! At each level of the hierarchy in each of the system described above, the information is processed, filtered, and then transferred to the higher level.

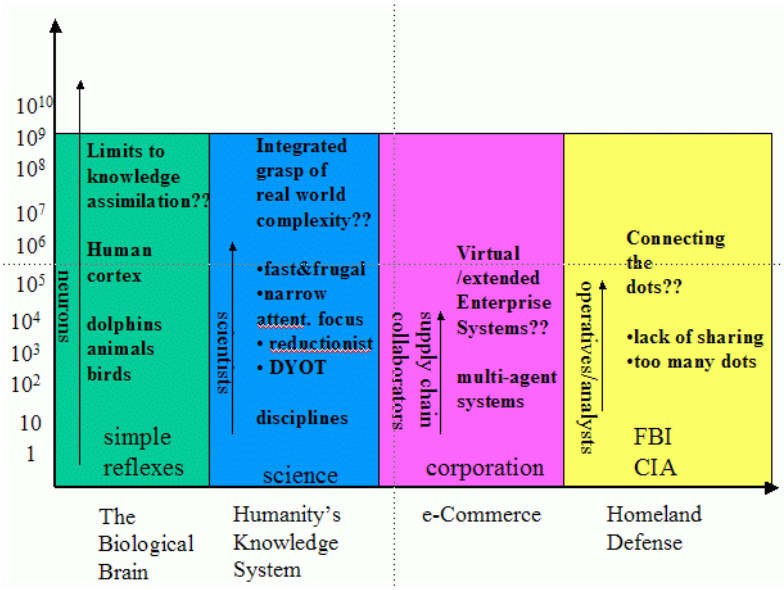


Figure 1: Scalability of Intelligent Systems

1.2 Towards Intelligent Systems

When Scalability and measurement of Intelligence is viewed from the Computer Science perspective, then intelligent processing can be viewed as a subset of universal computation. So, intelligence is measured by standard computational complexity. Intelligent Systems are organization of Algorithms, each of which may become a bottleneck for growth. Unfortunately, the only means to measure intelligence in a system is through simulation, as it cannot be measured in advance and is not semantically rich enough.

An Algorithm in such a system is considered practical if it completes in polynomial execution time proportional to the input data. Likewise, a problem is tractable if it has a

polynomial algorithm solution. But there are intractable problems also known as NP complete problems. These problems have non-deterministic polynomial execution time.

The problem of searching a visual space is one such intractable problem when the targets are not known in advance. However, if the targets are given the problem becomes linear and tractable ([5], [6]). Tsotos also showed that an attentional selection mechanism could provide subsets to try, each subset being processed in real-time. Although, there is no guarantee of its validity in general as the attention might be focused on a wrong area, it provides a method to reduce a NP complete problem to a tractable one.

This paves the way for Decision Making Systems (DMS) [2] that make time-critical decisions in real-time. The system is composed of semi-autonomous agents/components i.e. each component has its own behavior and is able to change decisions on inputs by environment and other components. There exists coordination between these components to enable DMS to work towards group agenda. This coordination requires processing, filtering of information as it travels, and changes hands between different components. Thus, DMS has the capability to focus attention i.e. allocation of resources to certain components relative to others. These components are distributed in space, sense environment locally and have their own life cycle. DMS components are heterogeneous, playing varied roles and having varied skill sets. All resources e.g. computational platforms, communication links, etc. are maintained only if the DMS succeeds in

maintaining, repairing, replacing them, i.e. survival as an implemented architecture, along with evolvability.

1.3 Motivation

The prime motivation of this thesis came from the involvement of Lockheed Martin's interest in the development of a decision-making simulation architectural framework capable of attention management. Figure 2 gives a general idea of what they intent to achieve.

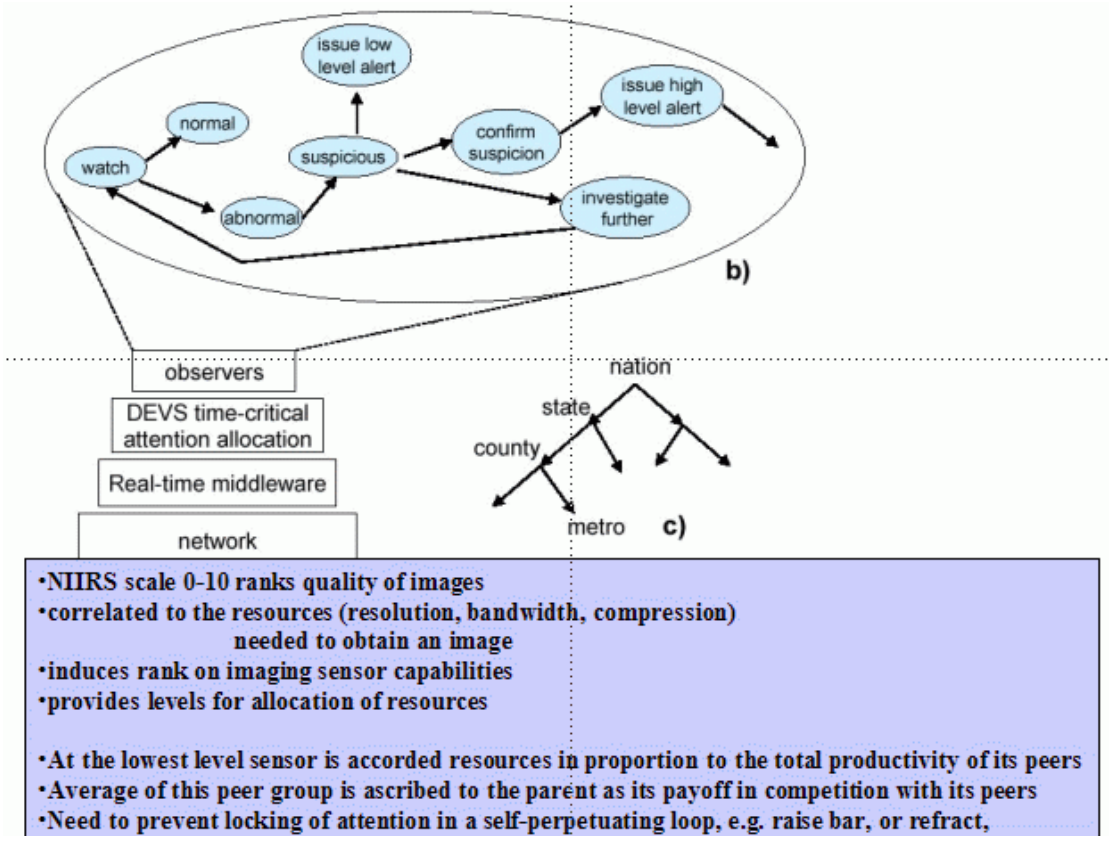


Figure 2: Focusing Attention on the Dots and their Aggregation

They are working on **How to design an attention management infrastructure to direct computational resources to sensors in a large distributed network.** Figure 3 shows an infrastructural view of the system. The system receives input from various sensors dispersed in the search Area. The system processes the information and sends commands to the actuators located in the area that perform their respective jobs. It's a battlefield scenario where the intelligence is gathered through sensors and control is undertaken by the actuators under direct control of the central Mission command. This is a flat architecture. It is not clear from this diagram as to what is hierarchy of the decision-making. Figure 4 shows the hierarchical representation of the system.

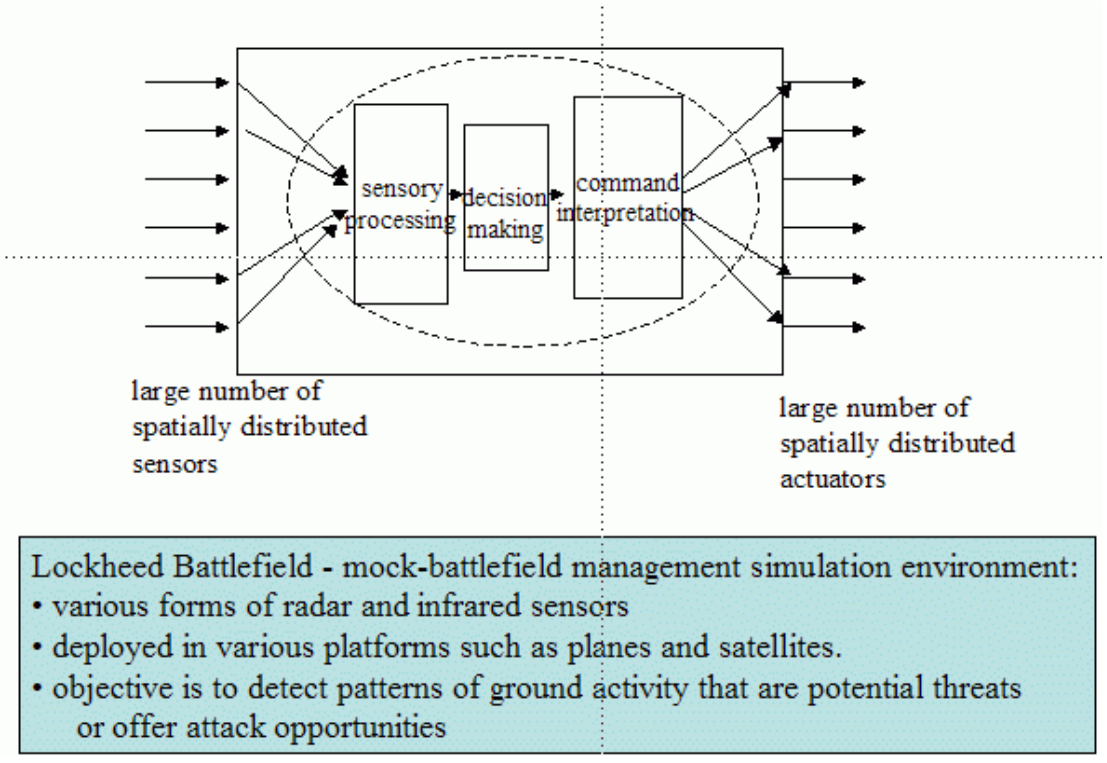


Figure 3: A blackbox view of the system

To specify these attention management systems in a little more detail, we specialize these systems in the set of intelligent systems called **LDRRT systems** [3]**.** They are characterized as:

a) **L**arge number of components

b) **D**istributed in physical space

c) **R**eal-time Behavior

d) **R**eactive

e) **T**ime-critical reaction

Basic architectural insights from the LDRRT system are:

a) Focusing on bandwidth, latency, compute power constraints

b) To reduce communication data load, increase pre-processing at sensors

c) This requires increased local compute power

d) Requires hierarchy to relieve bottleneck at central decision node

e) Develop abstraction to reduce communication in upward direction

f) Develop dynamic models (with predictive capability) to increase local decision-making power

g) Evolve attention allocation infrastructure to allocate bandwidth and compute power to focus activity on the needs of the moment

Figure 4 depicts the layered-system approach and presents the hierarchical layout of LDRRT system. The goals and commands follow the top-down approach, while the

resource allocation follows the bottom-up approach. At the lowest levels are sensors that receive resources like bandwidth, channel capacity, sampling-rates from the Resource Allocation Managers (acting as bandwidth brokers).

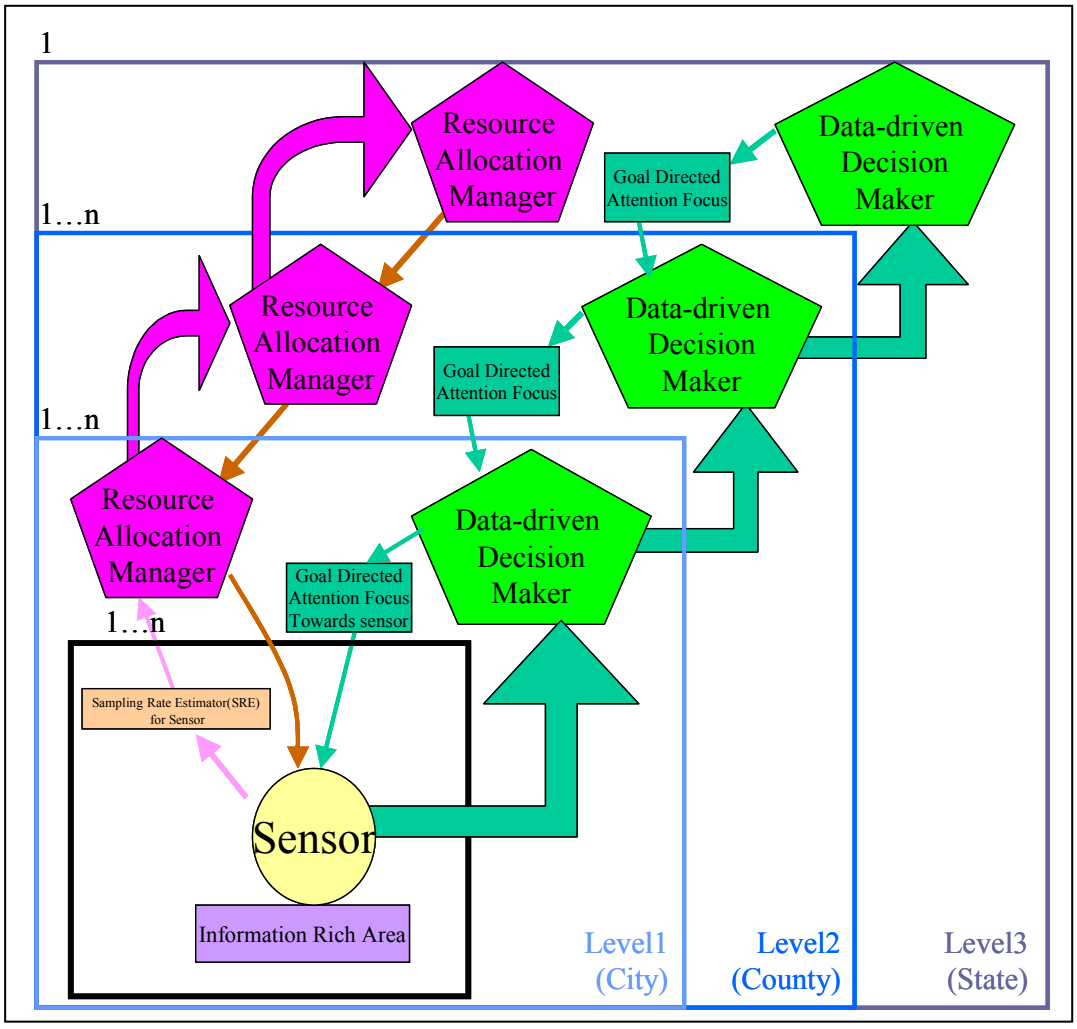


Figure 4: System model for Hierarchical multi-level architecture

Figure 4 shows different levels of the system mapped to geographical levels (based on area covered). Taken the complete picture*, the task is to design an Autonomous System,*

that can get "information" from numerous wireless sensor components, static or mobile, with the system working on a central goal and every sensor being fed (the resources) and be controlled (active or dead) according to this goal, at appropriate times, and still maintaining the self-constructed dynamic network.

The present thesis will focus on the bottom-up approach. It can be represented in figure 7 that shows the system deployment. At the lowest levels are different areas containing numerous sensors and Resource Allocation managers. These sensors report the activity in their assigned areas to these Allocation Managers that respond by allocating bandwidth to the reporting sensors.

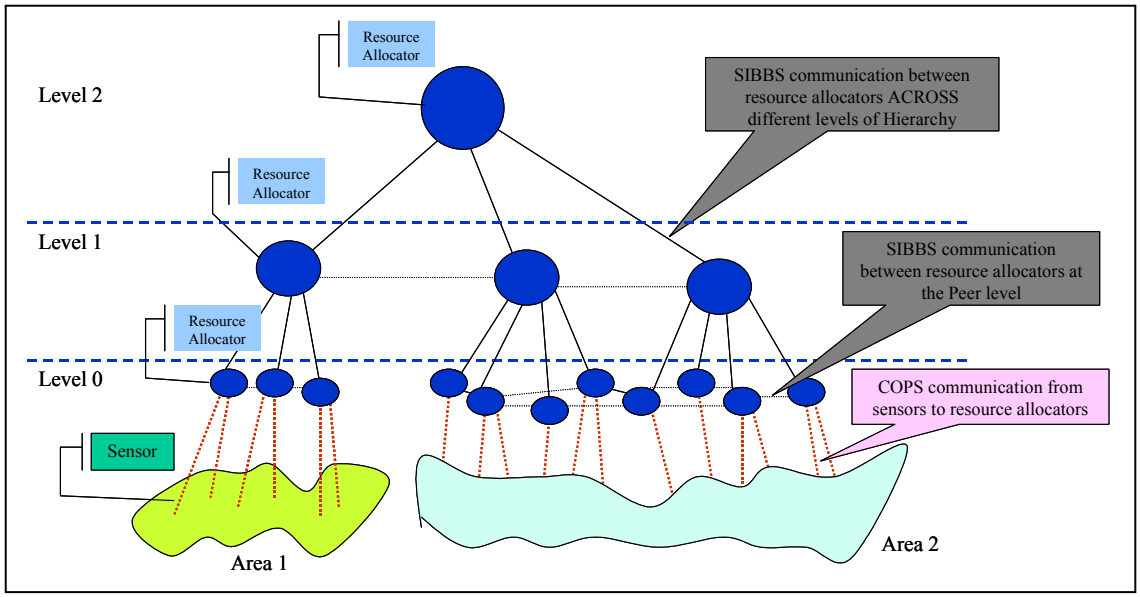


Figure 5: the deployment of sensors and Allocation Managers for the AdSNet system

There exist protocols like *Common Open Policy Service* (COPS) [38] that take care of policy based message passing between different components of the system. The Resource

Allocation Managers may be hierarchically organized or on same level. They communicate among themselves using protocols like *Simple Inter-domain Bandwidth Broker Signaling* (SIBBS) [37] that enable them to make reservations regarding bandwidth across different Allocation Managers. The focus of this thesis is not to model these communication protocols but to assume that such type of communication exists when the system is deployed.

1.4 System Design & Tools

The behavioral complexity associated with any ultra-large heterogeneous system arises from both the dynamics of the individual components and the structural relationships between components. Design decisions affecting the individual components often have significant impact on the overall behavior of the system that can be either favorable or disastrous. For example, activity in a search-area governing the sampling rate of the assigned sensor. The sampling rate of a sensor is defined as the measurable activity of a particular sensor in sampling the information out of the search space. Similarly, the collective activities of all sensors in an area drive the traffic parameters. Exploring these design decisions and alternatives is the focus of our interest.

The results of this work provide a modeling and simulation framework to explore the dynamic behavior consequences of design decisions. This framework enables the system designers to model sensors, surveillance areas and the communications hierarchy

according to the guidelines of the Homeland Security Systems and Attention-Management paradigms. We call the resulting framework as **Ad**aptive **S**ensor **Net**work: **AdSNet**. In the following sections, we briefly review the technologies used to design the AdSNet.

The AdSNet architecture is different from Priority-bases architectures. In priority-based systems, each activity is assigned a priority level and its then accordingly dealt with based on the policy-mechanisms. In Attention-focusing architecture, the concept of 'focusing' attention is highlighted, which conversely, leads to the suppression of any other running activities in the system. There is no suppression of any background activity in policy-based mechanism.

1.4.1 Java

Java provides the most familiar form of portability – source code portability. A given Java program produces identical results regardless of underlying CPU, Operating System, or Java compiler. The JVM provides CPU architecture portability. To facilitate OS/GUI portability, Java provides a set of packages (awt, util, and lang). Java programming language runs on object-oriented paradigm and provides a means for the development of software objects and applications. The software objects in AdSNet maps to the sensors and managers and other different components that are physically realizable in the real world. AdSNet is designed to model these components and simulate the resulting models

to support design decisions and observe the system behavior both qualitatively and quantitatively.

1.4.2 Unified Modeling Language

The Unified Modeling Language (UML) [51] is a graphics based language for specifying, visualizing, constructing and documenting models that have numerous components interacting with each other. It provides a means to depict the relationship and collective activity graphically. UML is well accepted in the Software Engineering Community and is fast becoming a standard due to its inherent advantages that aid in comprehension of the designed structure, behavior and interactions with the software System Under Development (SUD). It provides visual aides like Collaboration and Activity diagrams that help in understanding the message and control flow between the components of the SUD. Also, there are sequence diagrams that help understand the collective behavior on a time-line basis.

UML diagram complements the AdSNet architecture design with its graphical representation. Extending these UML diagrams and appending details according to the DEVS specification provides a complete set of parameters needed for AdSNet modeling and simulation.

1.5 Modeling & Simulation Formalisms

Four Fundamental methods for specifying dynamical system of models are Differential Equation System Specification (DESS) [19], Discrete Time System Specification (DTSS) [18], Qualitative System Specification (QSS) [19] and Discrete Event System Specification (DEVS) [18]. The figure below depicts the basics of these four fundamental specification methods. In order to develop a dynamic specification for AdSNet models, we need to work with these fundamental specifications. In AdSNet, we use QSS and DEVS formalisms. The QSS is used to model the search space that is under the sensor surveillance and has a random activity associated with it. Details of the implemented QSS are explained in the Section 4.1.1. For the behavior of the overall system DEVS is used, as the behavior of the components is piecewise constant over variable periods of time. Thus, we see that in the design of architecture of a framework that involves unpredictable environments (search-space) with predictable components (sensors and actuators), we need to exploit both the QSS and DEVS specifications.

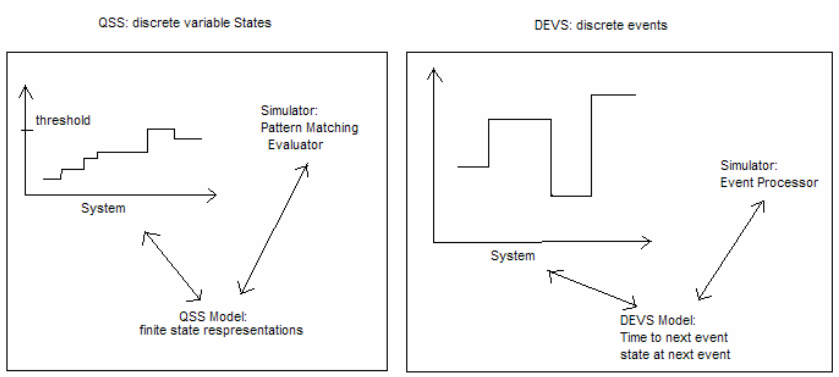


Figure 6: System Specification used in AdSNet Architecture

1.6 Plan of Thesis

The next chapter of the thesis discusses the background and related work. In Chapter 3, the conceptual constructs used to develop AdSNet modeling environment are outlined. Chapter 4 discusses the AdSNet Modeling and Simulation Environment and the process of creating a scalable sensor Network. The components that make up the system are discussed in detail and are analyzed using behavioral approach. Chapter 5 presents the simplest of scenarios in which the designed network-system could be put. The theoretical analysis of the System under design is taken up in Chapter 6 and boundary conditions of various parameters involved are defined. This analysis is then followed by Chapter 7, which deals with the Quantitative aspect of the system. The system is tested in two different manners, namely, flat architecture, hierarchical. This chapter also compares the performance of each of the experiments and the results are presented graphically. Chapter 8 introduces a new concept in modeling of Network delay. The designed AdSNet system is run over this model and quantitative results are produced and compared with the results of previous chapter. Chapter 9 mentions some application where this intelligent system could be put to use. Finally, Chapter 10 concludes the thesis with a discussion for future work.

2. Background and Related Work

How do we define intelligence and what are the parameters over which it can be measured? Is it a behavior that emerges as a result of some problem-solving technique and where does intelligence exists in the first place? Can we infuse intelligence in any other organism or, say in a machine.

It is a fact that human beings are the most intelligent of all species and the level of intelligence has increased with evolution. During the process of evolution from ape to man, there has been a marked increase in the reasoning faculties and the ability to draw conclusions in *homo sapiens*. Their problem-solving mechanism has become refined in the process of maintaining survival and countering threats. Consequently, intelligence in essence, is a problem-solving mechanism dedicated to the life and death of organism in the real world [13]. Of course, animals do have their survival strategies (flight or fight response) but the quality of reasoning is not as developed as it is in man. Apart from the existence of remains of the reptilian brain in man, there exist structures, like thalamus, that play integral role in the problem-solving mechanism.

In the brain, all the regions in cortex are connected with the thalamus and it is highly probable that the triangular circuit exists wherever cortical columns in one area communicate with columns in another area, completing this triangular circuit [16]. The proposed triangulation circuit functions to amplify cortico-cortical connections in both

bottom-up and top-down directions. Taken together, and with other PET (Positron Emission Tomography) studies show activations both in thalamic nuclei and the cortical regions. We see here that in order to produce a reactive response, the attentional expression (cortical areas activated by sensory neurons) has to contact thalamus, with sufficient intensity [16]. Then the thalamus amplifies the signal and directs various neurons in the motor cortex to facilitate an act of attentional control. Thalamus plays an important part in getting attention of the sensory inputs and focusing of the brain's resources to address the activity that has newly gathered [17]. Now, which cortical motor-areas need to be activated towards generating a reactive response depends upon the concluding faculties of the thalamus and its decision making power [17], for it's the humans who have the "choice" with respect to any situation unlike other animals where the response is generally hard-wired, as a result of millions of years of evolution.

Therefore, in a sense, the thalamus brings attention to any activity and awareness that the activity is being addressed. In order to display an intelligent reactive time-critical responsible behavior its important to have a component in the system, at every level in the hierarchy, that is capable of analyzing, deducing, concluding, accumulating information, directing, commanding, inhibiting, controlling and even enhancing the signal or information, working in background continuously receiving inputs from all types of sensors/lower levels.

Biologically implementable behaviors correspond to the environment in which the organism is situated and it directly amounts to the survival of the organism. Fast and Frugal (FFAs) heuristics work well with ecologically rational problem data sets that are extracted from real-world situations [13]. FFA refers to *fast, frugal and accurate* response behavior. FFAs are structured from the start to exploit ecologically rational behavior. It's a response behavior where an organism arrives at a decision that is fast, accurate and made by its limited mental resources that facilitate its survival in its environment. An organism's FFAs are essentially reflections of the real environment in which it has found its niche and to which it has been adapted.

FFA operates at the cognitive level where sequential processing predominates, but at the perception level, processing is much finer grained and parallel. Its been concluded that ecological rationality, fast and frugal heuristics at the cognitive level of processing and the attention directed perceptual processing are the keys to understand the characteristics of biological information processing [13]. Pinker [11] provides the fundamental guidelines of an artificial system that tries to imitate natural systems.

"Any intelligent agent incarnated in matter, working in real-time and subset to the laws of thermodynamics must be restricted in its access to information. Only the information that is relevant should be allowed in. This does not mean that the agent should wear blinkers or become amnesiac. Information that is irrelevant at one time for one purpose might be relevant at another time for another purpose. So information must be *routed*. Information

that is always irrelevant to a kind of computation should be permanently sealed off from it. Information that is sometimes relevant and sometimes irrelevant should be accessible to the computations when it's relevant, in so far as it can be predicted in advance".

Pinker also presents evidence that our visual systems are tuned to recognize the landscapes and the texture of scenes that are consistent and characterize the environments of our evolutionary ancestors. Tsotsos had shown that even in such a scenario, the problem of visual processing is inherently intractable and is NP-hard if the targets are not identified in advance [5],[7],[8]. He shows that an attentional selection mechanism is needed to deal with such intractability. Once the targets are given then the problem becomes linear in the size of the display [5],[6]. However, this mechanism is not foolproof, and attention may become focused at a wrong target, a dynamic system capable of switching attention may prove to be a reasonable solution.

Our work also takes help from the prototype systems that Tsotsos had done earlier with attention-management systems [14], [15], and [10]. This research takes off form where he left off by introducing new decision-making components in the system and giving the system the additional capability to allocate resources and suggesting a means to channelize them. This architecture is built using DEVS formalism that gives the advantage of using component modeling. The use of variable-structure modeling [20] makes the design of the system highly dynamic and evolvable. It gives us a tool to

change the interface of different components as well as the structure of the system to manage resources efficiently.

Other work has been done by Hannon [29], [30] in the area of Agent Systems integrating the emotions in the biological systems (particularly the humans) and the agent technology. His work deals with Cognitive-based Agent Architectures. He has used the emotion-based approach to improve the control mechanisms of several of his cognitive-based agents models. This approach [34] when combined with his current models for attention arousal, vision processing and natural language use and learning, using his Goal Mind Architecture environment, (the early results) demonstrates a marked improvement in the model's ability to fuse language and vision in a sensor rich environment. A number of efforts to incorporate emotions as a control mechanism in an overall agent model are ongoing [31], [32] and [33]. However, Scheutz [36] points out that unless an approach is cognitively deep enough to be explanatory, they may do little overall good. Hannon's approach is based on such explanatory mechanisms wherein the emotions are considered not only for determining the reasoning strategies used by the Higher Order Processes (HOPs) but also how they can control the amount and type of information presented to the HOPs by the periphery processing layer, which may be the sensors. When viewed a system as a whole, the decision-making is goal oriented and follows the top-down approach (from HOPs to periphery) while the information flows according to the bottom-up approach. One thing that is missing in the Hannon's work is the unpredictability in the environment under observation and the appearance of the

emergent phenomenon called “attention” in such complex sensor environment. His architecture has the advantage of Goal-directed execution where the system goals are decided by HOPs. The Emotion Control Model [34] use “fear” as a control mechanism to control both the attention and arousal in a dangerous situation for the agent. Details of attention–arousal model can be found in [30]. In the Goal-Mind Architecture [36], the constituent models draw their explanatory depth from the environment’s ability to support hierarchical cognitive processing. Using adaptive distributed processing and generalized inter-process communication; cognitive functions can be modeled at different levels of abstractions without changing the logical relationship between these functions. Thus, a perception model of the world and the self can be created/simulated with a reasoning and knowledge storage system that has far less capacity than that of real human.

In the area of developing wireless sensor networks, many approaches have come forward and all are working on creating “smart sensors”. The idea of sensor isn’t new and sensors have been around for over a decade but the idea of sending the refined information to its peers or superiors is what this research is all about. Rather than typecasting the sensor as a dumb component that relays whatever it detects, the focus is towards arriving at a localized scenario that is created by the sensor at its local end before its communicated across the network channel. As every bit of communication requires power consumption, reducing the amount of information at the sensor end is the only solution possible, and if the sensor processes and refines this information, it makes the sensor apparently

"intelligent". This research identifies sensor as an active component that has the capacity to deliver information, not just numbers and it calls for a data-processor inside a sensor (coupled with it) to validate data and make conclusions about the sensed activity. One other major component is their networking. Being ubiquitous and pervasive, self-organization is an important part of sensor networks and must be dealt quite fairly.

Various research groups have developed sensor applications [41]. Some of them are:

a) Intel and University of California, Berkeley have developed a wireless pager-sized "chassis" that can be customized with various types of sensors. This is being used to track microclimates and pests in vineyards, monitor the nesting habits of rare sea birds, and control heating and ventilation systems.

b) At UCLA, researchers are deploying wireless sensors to gain detailed measurements of the effects of seismic waves on buildings.

c) *Sensoria* in San Diego, is developing sensors that could turn cars into traveling nodes urban wireless networks, allowing groups of vehicles to automatically assemble real-time information about local destinations.

d) Culler's Group at Berkeley and Intel along with Crossbow Technology in San Jose has developed a potentially dominant design called "mote" and its operating System "TinyOS". These motes, tested by hundred of research groups around the world, are smaller and use less power than other commercial sensors.

Research is also undergoing in the area of developing protocols and the hardware technology to bring about the realization of these sensor agent systems. Protocols like Simple Inter domain Bandwidth Broker Signaling (SIBBS) protocol [37] have come up that can create an overlay network of resource-managers and communicate among themselves as well as to the levels above them. Similarly, protocols like Common Open Policy Service Protocol (COPS) have been designed [38] that can communicate between the sensors and the resource-managers/bandwidth-brokers. The work is currently in progress and so is the design of a system that can maximize these protocols. Architectures have been proposed [35] that are designed using these protocols in the underlying network but a system is yet to be implemented on the magnitude of Internet scale and be controlled remotely.

This works takes off from the research done in the area of Attention-focusing by Tsotos, Cognitive architecture by Hannon and the general architectural problems faced during the allocation of resources (like bandwidth, power, link capacity) and getting the required information from a vast sensor network, by proposing a complete architecture. It brings forward an architecture that is capable of both the peripheral level processing as well as goal directed system execution. The experimentation for the goal-directed simulations is beyond the scope of the present work and will be taken in the due course.

2.1 Attention-focusing Experiment

Tsotsos has conducted experiments with visual image pattern recognition. [5], [6] and [10]. In [6] he has conducted an experiment where a light beam is made to focus on a part of an image (target) after locating the target inside the image. He has demonstrated that if a target is known in advance, then the problem of finding the target from a search space can be solved in realistic time and the problems is no more NP-complete. Figure 7 explains the steps the light beam takes until it reaches the target. Consider an image in Figure 7 at the bottommost layer of the triangle pyramid. Given a target image (at a particular resolution, we need to find this target and focus a light beam as we increase our resolution in approaching the target. Consider the different resolutions as different layers in the pyramid (Figure 7b). As we localize our target within an area, at any particular resolution, we also allow the light to pass through this area.

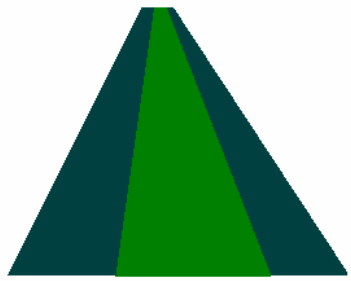


Figure 7a: Triangle representation of an Image

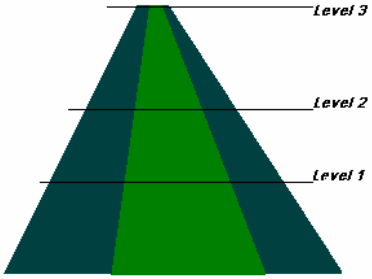


Figure 7b: Layers as representation of resolutions. Layer 1 has higher resolution than layer 2

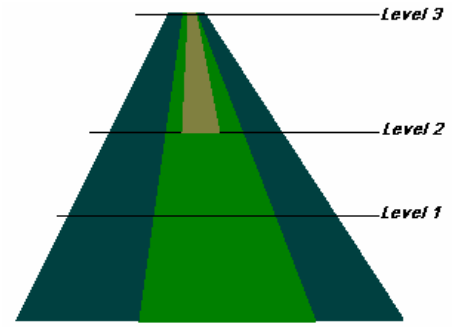


Figure 7c: Light beam through Layer 3

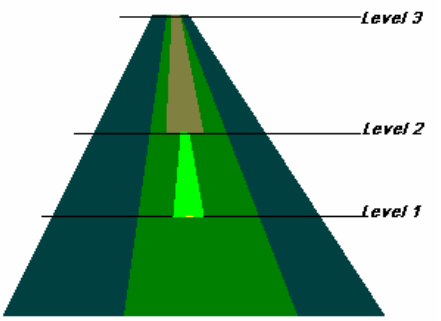


Figure 7d: Light beam through Layer 2

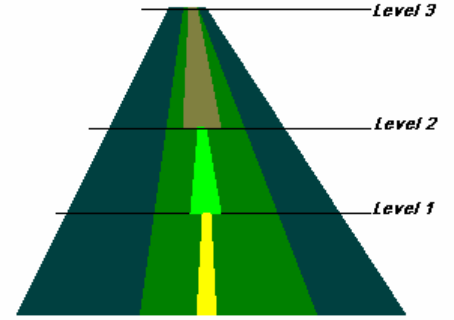


Figure 7e: Light beam through Layer 1

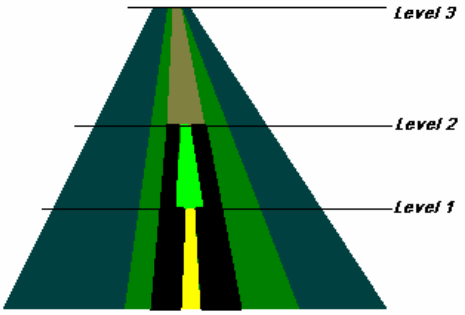


Figure 7f: Black area representing inhibited solution sets

Figure 7: Focusing of Light beam on a target within an image

At each resolution, an area is obtained, and then a winner is calculated within that area. The winner is decided based on *weighted-sum* rule [6]. All those pixels that qualify their proximity to the target, at this particular resolution, are chosen as winners within that area. This process is performed using an algorithm called *Winner-take-all* (WTA) algorithm [6] and [10]. WTA is executed until a winner can be determined at any particular resolution. Figure 7c, 7d and 7e show the path of the light beam as it reaches close to the target. Another effect of WTA algorithm is the inhibition of the various other solution sets amongst which the winner was chosen. The winner takes all the attention (passing of light beam) while the others in that layer are ignored.

We attempt to implement this approach in our sensor network where at the bottom of the pyramid are numerous sensors sending their activity rate up the hierarchical network. The attention is focused in terms of assigning a sampling rate to the sensor that shows high or important activity.

3. AdSNet Building Blocks

3.1 DEVS Specification

DEVS exhibits the concepts of systems theory and modeling and affords a computations basis for implementing behaviors that are expressed in the other basic systems formalisms – discrete time [18] and differential equations [19]. It supports capturing the systems behavior from both the physical and behavioral perspectives. A *Discrete Event System Specification* (DEVS) is a structure.

$M = < X, Y, S, \delta_{int}, \delta_{ext}, 8, ta >$

where,

X is the set of input values
Y is the set of output values
S is a set of States
δ_{int}: S → S is the internal transition function
δ_{ext}: Q X X →S is the external transition function, where
Q = {(s,e)| s ε S, 0 <= *e* <= ta(s)} is the total state set
e is the time elapsed since last transition
8: S →Y is the output function
ta :S → $R_{0^+,infinity}$ is the time advance function

Figure 8: Definition of DEVS model

A DEVS model can either be atomic or coupled, which is composed of many atomic models [18]. Atomic and coupled models can be simulated using sequential computation or various forms of parallelism. A parallel DEVS atomic model is defined as:

$M = < X_M, Y_M, S, \delta_{int}, \delta_{ext}, \delta_{conf}, 8, ta >$

where,

$X_M = \{(p,v) \mid p \varepsilon IPorts, v \varepsilon Xp\}$ is a set of input ports and values

$Y_M = \{(p,v) \mid p \varepsilon OPorts, v \varepsilon Xp\}$ is a set of output ports and values

S is a set of States

$*_{int}$: S → S is the internal transition function

$*_{ext}$: $Q \times X_M^b \rightarrow S$ is the external transition function

$*_{conf}$: $Q \times X_M^b \rightarrow S$ is the confluent transition function

8: $S \rightarrow Y_M^b$ is the output function

ta :$S \rightarrow R_{0^+, infinity}$ is the time advance function

with

$Q = \{(s,e) \mid s , S, 0 <= e <= ta(s)\}$ is the set of total states

e is the time elapsed since last transition

X_M^b is a bag of inputs (a set with possible multiple occurrences)

Y_M^b is a bag of outputs (a set with possible multiple occurrences)

Figure 9: Definition of DEVS Atomic model

As shown in the above figure, each DEVS atomic model has a state representation and it exists in one state. It passes through different states on the basis of its internal function after spending a finite time in a particular state. It executes its internal function in the absence of any input at its interfaces. If the model receives any input at its ports in any particular state, it executes its external transition function, and goes to another state dictated by the external transition function. Thus, internal transition function dictates the system's new state when no events occurred since the last transition and the external transition function dictates the system's new state when the external event has occurred. At this point the new state is determined by the input, current state and time duration for which the system will stay in this new state. In both cases, the system is in some new state and it will stay there for a duration dictated by its time advance function. This cycle is repeated thourghout the life of the system. There is another transition function called

the confluent transition function which comes into play what the system is on the verge of going to another state and an external event occurs. This allows the system to define its behavior in case of this simultaneous occurrence of external event and internal transition. It allows the system to execute either the external transition function or the internal transition function first, depending on the behavioral characteristics of the system.

Just like the Atomic model, the coupled model is also defined in the similar manner. The specification includes the external interface (input and output ports and values), the components (DEVS models), and the coupling relations.

$N = < X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$

where

X is a set of input values
Y is a set of output values
D is a set of the DEVS component names
For each *i* , D,
M_i is the DEVS component model
I_i is the set of influencees for I,
For each $_j \varepsilon I_i$,
$Z_{i,j}$ is the *i-to-j* output translation function

Figure 10: Definition of DEVS Coupled model

The sequential formalism treats component transition functions (δ_{ext} and δ_{int}) sequentially, while the parallel DEVS treats them concurrently. By adding couplings between output/input ports of different components, messages can be passed from one component to another. Under the property of closure under coupling, a coupled model itself can be treated as a sub-component of other models. This kind of hierarchical modular construction makes each DEVS model a self-contained component that can be

easily reused. Because of this, DEVS component-based modeling and simulation environment does not rely on the underlying implementation language.

Another addition to the current DEVS formalism is the Variable Structure DEVS [20] with the capability of dynamic reconfiguration. For component-based modeling and simulation, dynamic reconfiguration provides several advantages:

- It provides a natural and effective way to model those complex systems which exhibit structure and behavior changes to adapt to different situations.
- It provides the additional flexibility to design and analyze a system under development.
- Dynamic reconfiguration makes it possible to load only a sub-set of system's components for simulation. This is very useful to simulate very large systems with tremendous number of components as only the active components can be loaded dynamically to conduct simulation.
- It provides a mechanism to see the evolution of the system if the system has the capability to expand and contract during its lifetime.

In DEVS, they are usually referred to as variable structure modeling. The update of a component means a component is updated by a new version that might have a very different behavior or interface from that of the old one. This new component can also have additional functionality than its previous version. In a sense, its capable of adaptation.

This research exploits the advantages of the variable-structure DEVS and attempts to build an evolving architecture where the components have adaptive behavior.

3.2 Experimental Frame

An Experimental Frame is a specification of the conditions under which a system is observed and experimented with. It is the operational formulation of the objectives motivating a modeling and simulation project. Simulation based performance analysis requires careful design of experimental frame, the concepts of which has been proposed in [18]. It basically consists of three models: a generator model for generating inputs to a model under evaluation, a transducer model for collecting output data and an acceptor model for controlling simulation run.

Design of experimental frame is objective-driven, meaning that different design objectives require different experimental frames. More specifically, a set of design objectives is transformed into a set of performance indices. Then, an experimental frame is designed such that the desired performance indices are measured by simulation. Thus, simulation is done in a way that a model is simulated with a set of different experimental frames, or a set of candidate models are simulated with an experimental frame until a desired design is found.

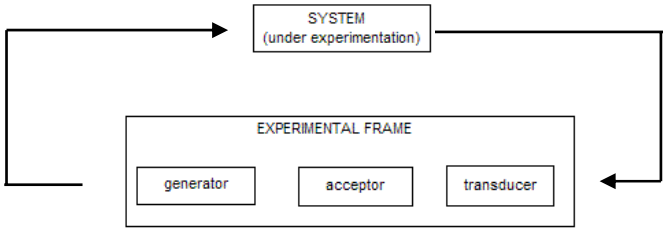


Figure 11: Experimental Frame

Figure 12 demonstrates the existence of various models (Model 1, Model 2, etc.) that exissts in the system and the corresponding Experimental Frame to test them.

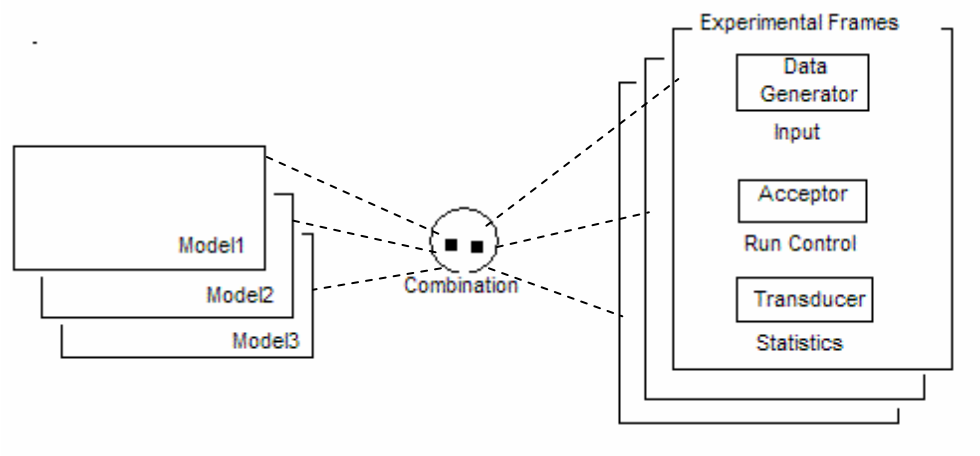


Figure 12: Different models require different Experimental Frames

3.3 AdSNet Model

This section describes the problem using semantic language. The use of UML comes handy at this stage and a general functionality of the system is described as follows:

We represent the components of the system in **Entity-Relationship diagram** using UML as the tool in figure 13.

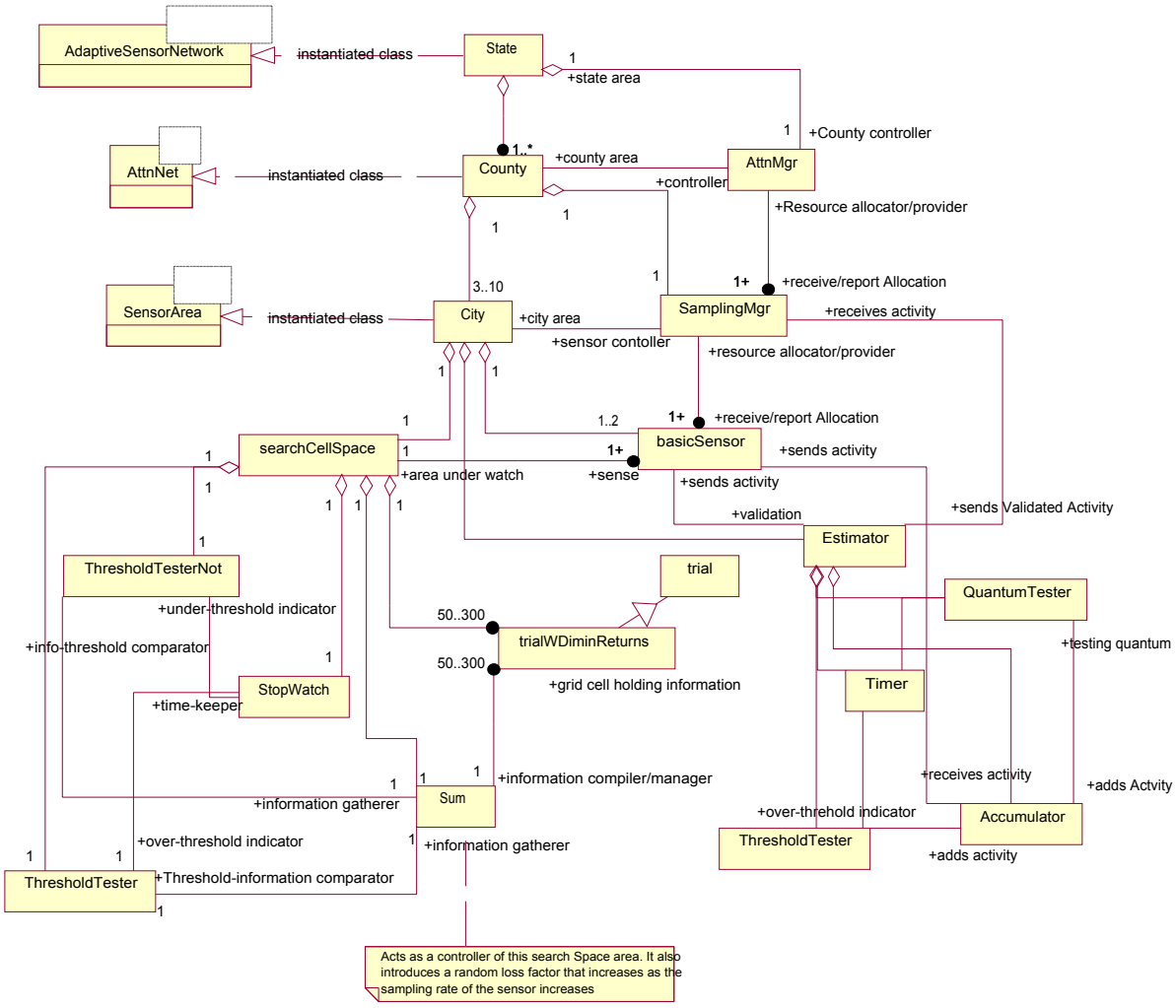


Figure 13: the entity-relationship diagram for the adaptive sensor system

4. AdSNet Component Behavior

This chapter deals with the behavioral specification of the components that constitute the AdSNet system. In particular, we focus on presenting the state charts for various entities that behave due to the objectives laid under the problem definition area. We also view these entities working in collaboration with other entities under the structural hierarchy shown in **Figure 13**, using of the collaboration diagrams (UML) and the activity charts. These behavioral representations were developed and refined all throughout this research work even in defining the objectives.

4.1 Topological Behavior

The hierarchy of the AdSNet system maps with that of the land-planning administration in a typical battlefield scenario. At the lowest level are the localized areas that are under the investigation by the sensors. These are dealt under as Surveillance areas. Many surveillance areas exists within a city but for simplification purposes there exists one Surveillance area per city. The cities under consideration fall within the County limits and there are numerous cities within a county. Likewise, we extend the topology one more level up by having many counties under a state.

4.1.1 Surveillance Area (searchCellSpace)

A Surveillance area is an area that is under the observation by one or more sensors. This is a highly dynamic environment where the activity level is unpredictable. Each surveillance area is a collection of cells organized as a two dimensional grid. Since the basic unit of this area is a Cell, its actually a Cell Space.

The basic characteristics of a single Grid cell are defined according to the requirements in the problem statement. Its salient features are:

a) It represents a finite area which collectively taken (with all cells in the Cell Space) amounts the total Surveillance area.

b) The Cell is either active or passive and these events are mutually exclusive.

c) The Cell is associated with a resource parameter (between 1 and 100).

d) The probability of the Cell becoming active is directly proportional to this resource parameter

e) Every time the cell becomes active, the resource parameter is decremented, giving the impression of exhaustion of resource as the cell is projecting active behavior. Since the resource value gets decremented with time, the probability of this cell becoming active also decreases.

f) Any Cell inside the Surveillance area can become active. Becoming active is totally a random phenomenon and all cells are bound by their individual

probability of becoming active as each cell has a unique storage of resources (resource parameter).

g) The Cell continues in the receptive mode till there is finite number of resources left in the cell. Once the resources are reduced to zero, the cell turns passive.

Along with this basic functional unit there are other components that integrate these individual cells towards an area. The area is basically a DEVS coupled model with the following four components in closed couplings with the grid Cells (as shown in the figure). These components constituting an area, along with the Grid Cells are:

a) StopWatch

b) Sum

c) ThresholdTester

d) ThresholdTesterNot

The **StopWatch** keeps track of the time this area remains active. As long as the cells of the area remain productive and the number of cells becoming active is over the minimum threshold it stays active and outputs the time only when the area becomes dead. The **Sum** component adds the number of grid cells becoming active at a particular instant. It's a gateway to the outer world that tells the sensors about the activity inside the cell space. It is responsible to report activity to the sensors. It also introduces a loss factor which diminishes the activity as it is relayed. The loss factor is a random parameter and it varies

according to either linear or square function. **ThresholdTester** checks if the number of cells becoming active are over a certain threshold before it can trigger off the stop watch to declare that they are dead. Likewise, **ThresholdTesterNot** keeps track of the situation when the threshold hasn't been reached. It keeps the stopwatch engaged in the active state so that it continues to note time. The following table delineates their state diagrams. Figure 14 shows the Cell Space structure containing 5 Grid cells. According to the specifications given in the problem statement, the number of cells can be anywhere between 50 and 300. The functionality of this surveillance area can best be comprehended by the UML Collaboration Diagram below.

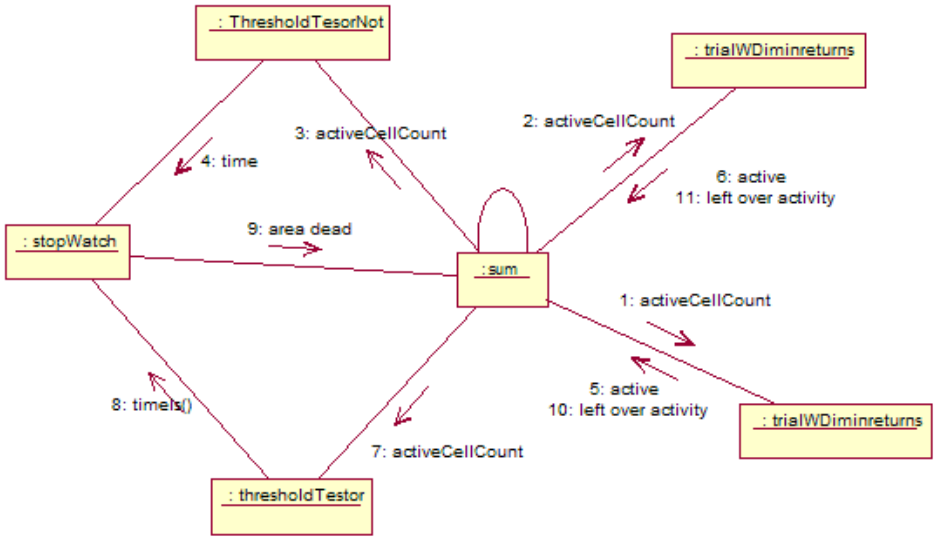


Figure 14: The objects of cell space in communication with each other

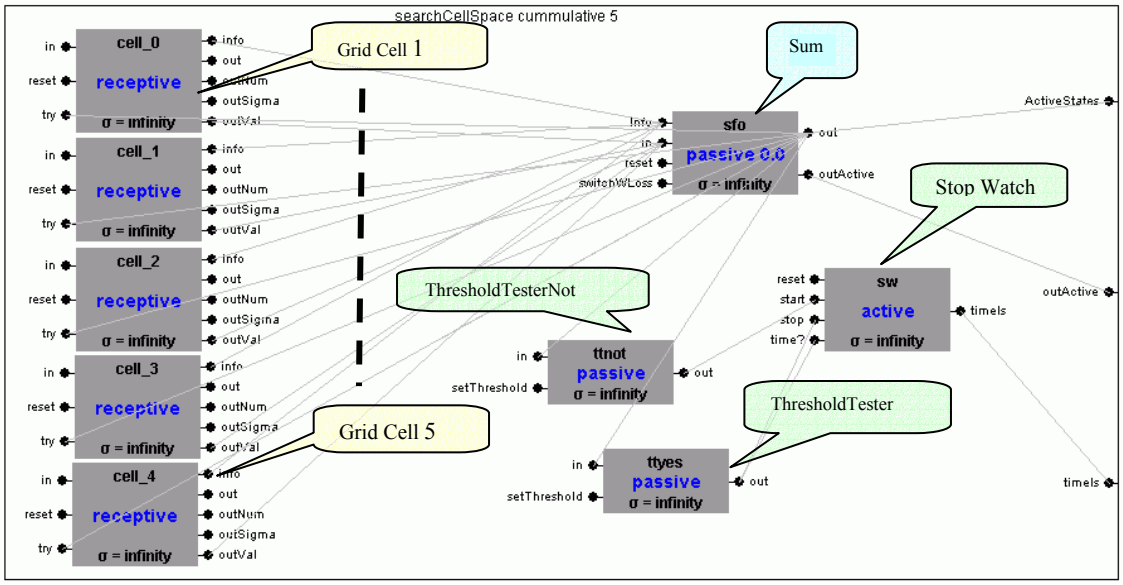


Figure 15: Digraph Coupled model showing the Cell Space area components with 5 cells

The whole of the cell space behaves according to the Qualitative Systems Specification (QSS) where in it produces output only when this closed system reaches a certain threshold. It operates within a certain boundary and gives a qualitative appearance of showing 'activity' as a whole. When it produces output, it advertises to the external world that this closed-coupled system is undergoing a quantum change in its state. However, its internal components work according to their threshold values, the aggregated output value that comes out of this closed system is not dependent on these threshold values. The output value may be lower than the threshold of the components and it results due to the loss factor that creeps in as the information (activity) leaves this closed system. The incorporation of error factor and the system maintaining its behavior despite introducing

error and undergoing a quantum state change is what differentiates this system with the normal DEVS.

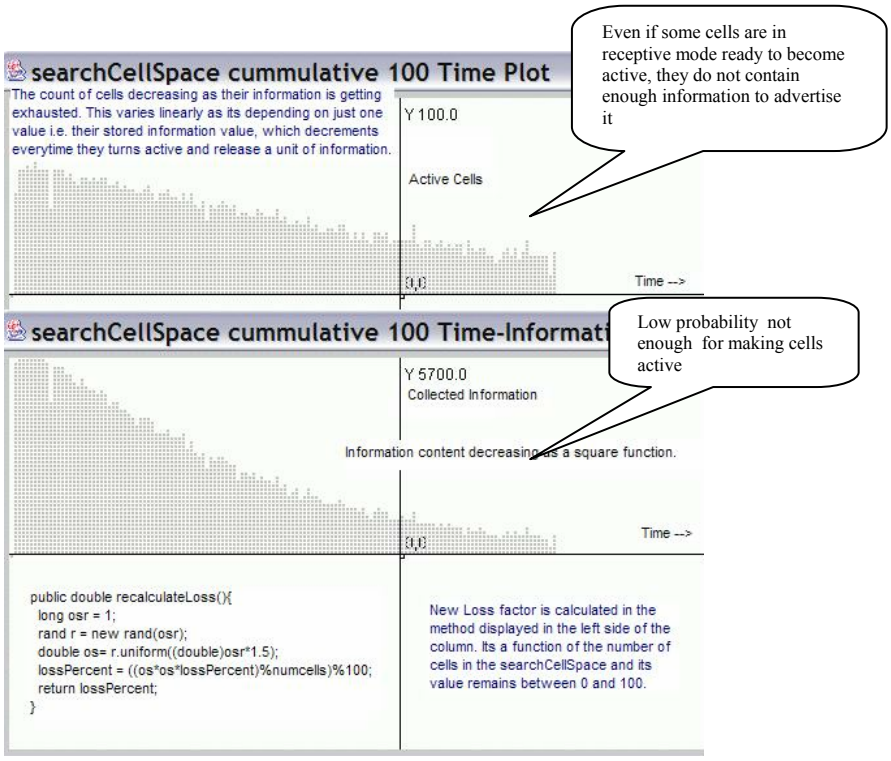


Figure 16: Activity associated with the Surveillance Area

4.1.2 City (sensorNet)

The city structure is actually a sensor network wherein sensors are assigned a Surveillance area. The details of Surveillance area are dealt in the previous section. For simplicity purposes only one area is there in the City structure though in reality there may be numerous areas within a city. Again for simplicity, the sensors that are assigned to a surveillance area can be a maximum of 2. Attached to every sensor is an *Estimator* that acts as a 'thinking element' capable of drawing conclusions about the sensor's behavior.

It has a threshold level defined by the system parameters, that acts as a baseline to deduce results about the sensor activity.

Sensor

A sensor is an atomic entity whose primary purpose is to acknowledge any activity in the area that it is investigating. It has its own sensitivity level over which it can detect any activity. The sensitivity in this context may be considered as the activity within the Surveillance area at which the sensor detects this activity as the "activity" which then calls for a corresponding response from the sensor. The first task that the sensor does is to communicate the activity to the Estimator for further processing. This process is repeated indefinitely and the sensor maps the area activity with the corresponding sampling rate. The sampling rate of a sensor is the ability of the sensor to process information in unit time. As in reality, these sensors do not always work an maximum allowable power but at an 'optimum' level and can go to the maximum level if the need arises. They work at a rate proportional to the information under processing and as a percentage of the maximum capability. The sampling manager sends them this percentage value and the sensor then works according to it. The sensor is both an active and a passive element. Its "active" in the sense that it has its own discretion to detect activity and can differentiate between a normal and an abnormal activity (though it needs to be programmed in terms of its sensitivity) and "passive" as it calls upon the sampling manager for an updated Sampling rate to process the information coming. A sensor has a typical structure as shown in the Figure 17:

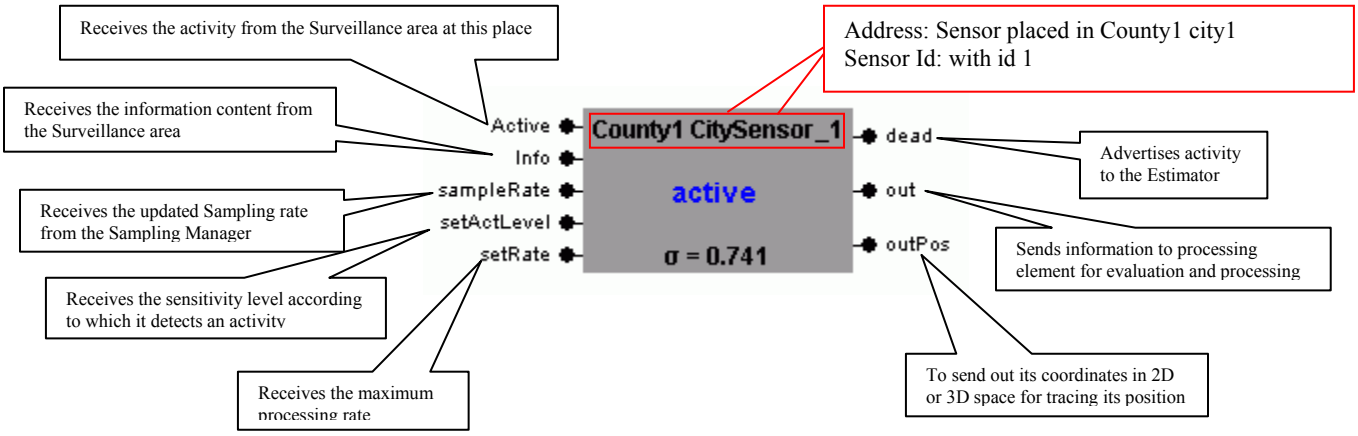


Figure 17: A Sensor with its interfaces and I/O handling

The Sensor has four states as shown in Figure 18. It continues to stay in the active mode as it needs to continue to sample information at the default rate. It's necessary as by being in the active mode will enable it to detect any abnormal behavior in the surveillance area.

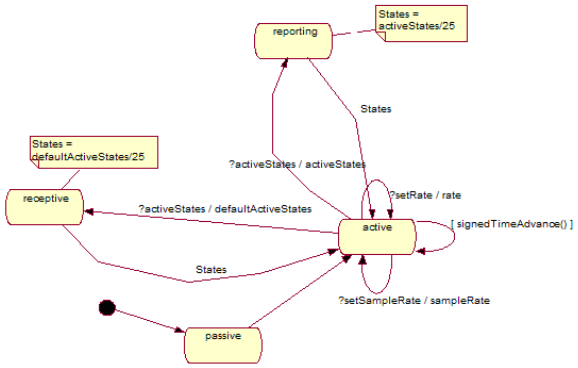


Figure 18: State diagram of a sensor staying in default activity as long it is alive

The ideal behavior of a sensor is displayed in Figure 19. It shows that the sensor continues to stay in the receptive mode with a default sample rate and accumulates information. It is when the information reaches/surpasses the sensitivity level, it switches to the reporting mode, wherein it reports the activity to the Sampling Manager and is assigned a new Sampling rate. This new sampling rate is higher than the default sampling rate and the sensor starts sampling the information more rapidly. The different curves in the figure show pathways of information gathered in different Surveillance areas. The leftmost depicts the behavior of highly information loaded region containing large number of cells and consequently a high amount of time to sample this information by staying in the active mode. While the curve in bold depicts the boundary condition where the sensor reports an activity that just equals its sensitivity threshold. It then receives the sampling rate from the Sampling manager and processes the information. In the next step, it turns from an active mode to receptive mode again.

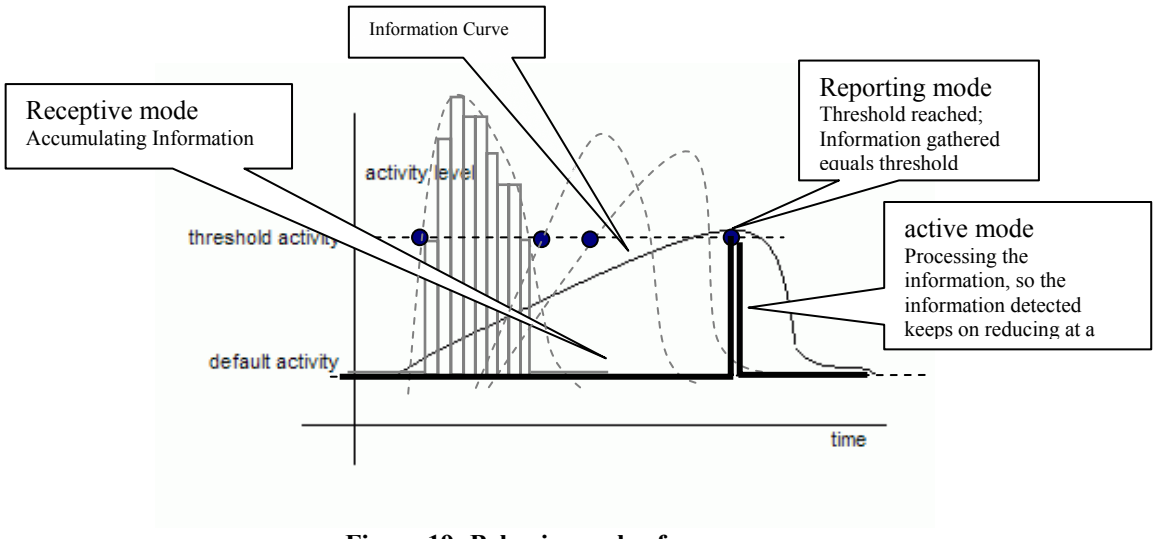


Figure 19: Behavior cycle of sensor

The behavior of the sensor in action and its interaction with different components of the system can best be seen in the UML collaboration diagram Figure 20. Apart from the surveillance area, other component that it needs for its successful operation and desired behavior is the Sampling Manager, a detail of which is presented in the Section 4.1.3.

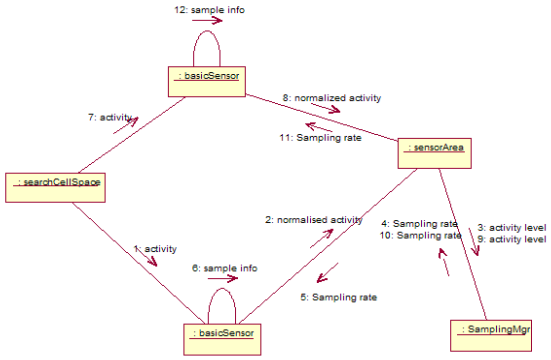


Figure 20:The collaboration of City coupled model. The sensor area object shown above refers to the City boundary that the sensor messages have to pass through to reach Sampling Manager and vice versa

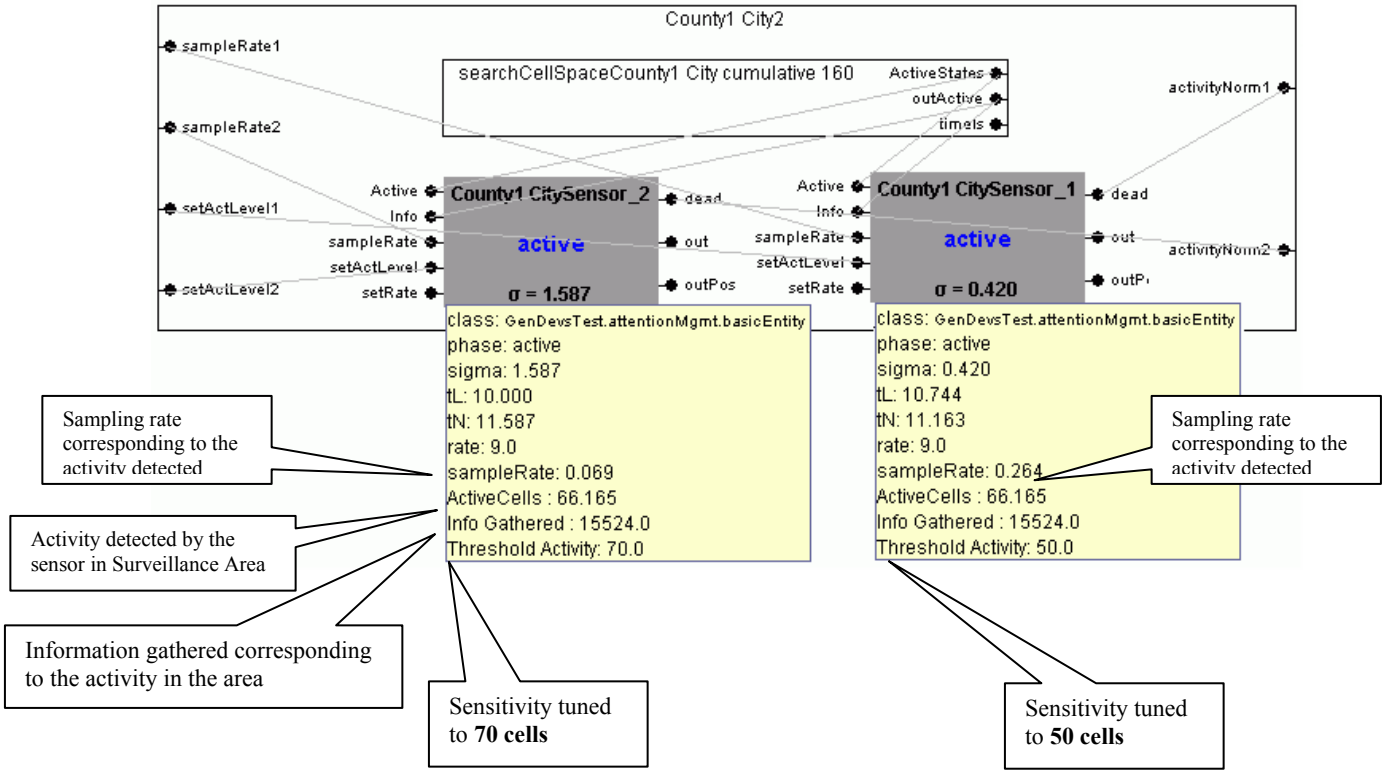


Figure 21: Snapshot of the city in action displaying the statistics of each of the Sensor

The UML diagram shown in Figure 20 describes only one sensor in operation. Figure 21 shows a typical city boundary that is composed of a surveillance area and two accompanied sensors.

Notice that the two sensors are investigating the same area but each of them has unique sensitivity index. Count1 CitySensor_1 has a sensitivity of 50 while the second sensor County1 CitySensor_2 has a sensitivity of 70. Also notice, the time advance (sigma) of

the next event. The sensor that has a high Sampling rate is scheduled for a next event first. However, the information reaching both the sensors is same, it is their sensitivity that gives them the criteria to decide as to what will be their sample rate. Sensor 2 doesn't consider the activity in the area enough to declare it as an abnormal activity as its threshold is 70 and the activity in the area is 66.165. On the other hand, the sensitivity of the other sensor is 50. It also detects an activity of 66.165 in the area and declares it as abnormal increase, which is above its threshold and starts sampling at a higher rate. A typical snapshot of the system is presented in Figure 21 and the comparison of the instantaneous values is shown below:

Table 1: Comparison of the sensitivity behavior of the two sensors observing same activity

Behavioral Statistic	County1 CitySensor_1 (max Rate = 9.0)	County1 CitySensor_2 (max rate = 9.0)
Sensitivity	50 cells	70 cells
Activity detected	66.165	66.165
Information Gathered from the active cells	15524 units	15524 units
Sampling Rate	0.264 of max rate	0.069 of max rate
Time Advance of next event	0.42 sec	1.587 sec

Thus, this section describes the City containing one Surveillance area with two sensors in action.

Estimator

Each sensor has an attached rate estimator. It is a compound element composed of the following basic functionalities:

a) Accumulation: This component accumulates the sensor activity sent by the sensor for the processing inside the Estimator.

b) Threshold Estimation: This component evaluates the accumulated activity with the threshold parameter that has been setup by the system specifications. This is different from the sensitivity of the sensor (sensitivity applies when the sensor detects an activity). This comes into operation when the area-sensor combination reaches a meaningful threshold. It works on the accumulated activity in the accumulation process and sends a signal when the area-sensor combination, the accumulated activity, surpasses the defined threshold.

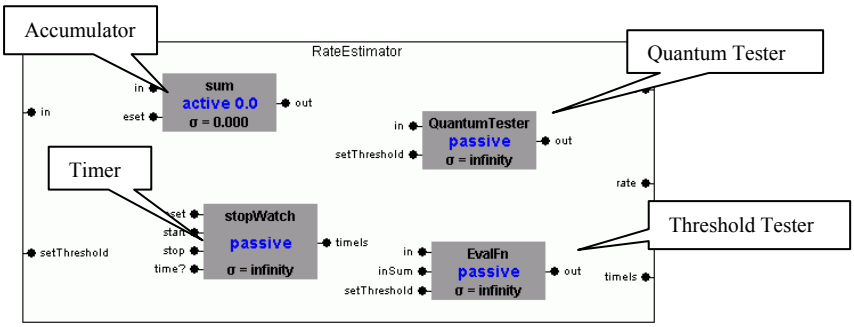


Figure 22: Components inside a Rate-estimator

c) Strength of the signal received from sensor: This is a timer. It keeps an account of the time period over which the accumulator has been in session. This helps in getting the strength of the accumulated activity. The more the time elapsed in reaching the threshold, the less the strength of the activity and vice versa.

d) Quantum Testing: This is the component that helps integrating all the above three components. This component advertises the signal along with its strength to the outside world and generates a snapshot of the area depending upon the

information gathered by the accumulator. This component sends the processed signal to the Sampling Manager that then sends the updated Sampling rate to the sensor. At the instant it sends the communication to the Sampling Manager, it creates the snapshot of the accumulated activity.

The working of the Estimator is analogous to the working of thalamus, the decision-making entity that takes decision of the validity of the inputs from the sensors. The sensor is independent of the operation of the Estimator and it continues to provide input to the Estimator. This working is again taken from the biological domain where the sensors (ear, eyes, touch, smell, taste) keep on sending information to the brain, but it's the decision making capability of the brain to process, integrate and give a comprehensive picture [17]. In our model, every sensor is provided with an estimator to validate the results of the sensor. The model of the Estimator is a basic and simple model to map the analytical faculties of brain and has a very high level of abstraction. It is designed on the FFA Heuristic mechanisms such that it sends a message as soon as it has enough information. This is implemented using threshold mechanism. The message sent is actually the strength of the activity received by the sensor.

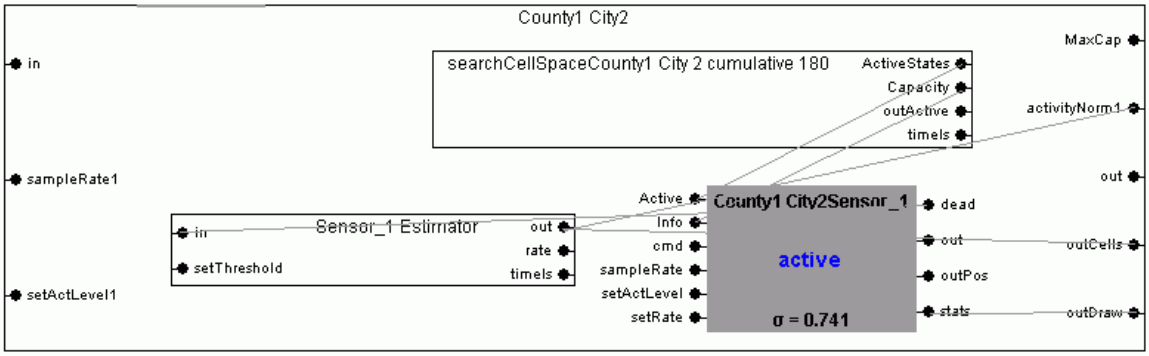


Figure 23: Sensor Estimator Pair

4.1.3 County (attnNet)

The County is an aggregation of many cities. Apart from a collection of cities, a county contains another important component, Sampling Manager. This component is responsible for controlling and release of valuable resources to the sensors placed in the city. As indicated in the problem statement, this manager has a maximum of resources corresponding to the total area of the cities in its control i.e. the whole county. It assigns the maximum allowable resources to the respective cities proportional to their area. As the sensors report the state of activity in their assigned areas, the manager assigns them their sampling rate to process the information released by their areas. The Sampling Manager has a quantum over which any change reported by the sensors is considered for a new assignment of the sampling rate to the sensor. The sensors continuously report about their activity (through the Rate-estimator) to the manager. If the level of change in activity is below this quantum, the Sampling Manager doesn't revise the sampling rate of the sensor. But, if the activity reported surpasses its quantum, then it revises the sampling

rate for that particular sensor. When the sensors do not report any abnormal increase or decrease in the activity level of the area, i.e. the activity level is well within the quantum range of the Sampling Manager, the continue to sample at their current rate, the rate at which they reported the activity. If the activity level is below their individual threshold, then the Sampling Manager assigns them a default Sampling rate, which make them continue to sample at a minimum possible rate. This is required as the sensor can detect any abnormal activity only when they are always investigating. To keep them in the ON state, they are assigned this default sampling rate. A typical County scenario is displayed in Figure 23. It contains three cities, the address of each is printed inside the top area of each box e.g. County1 City1, county1City2, etc.

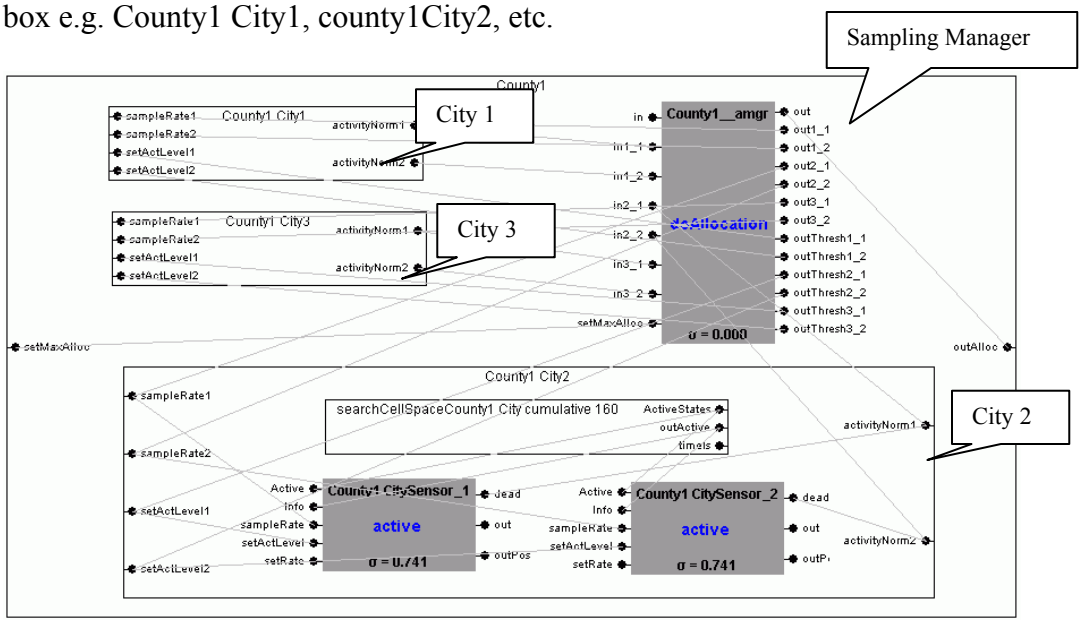
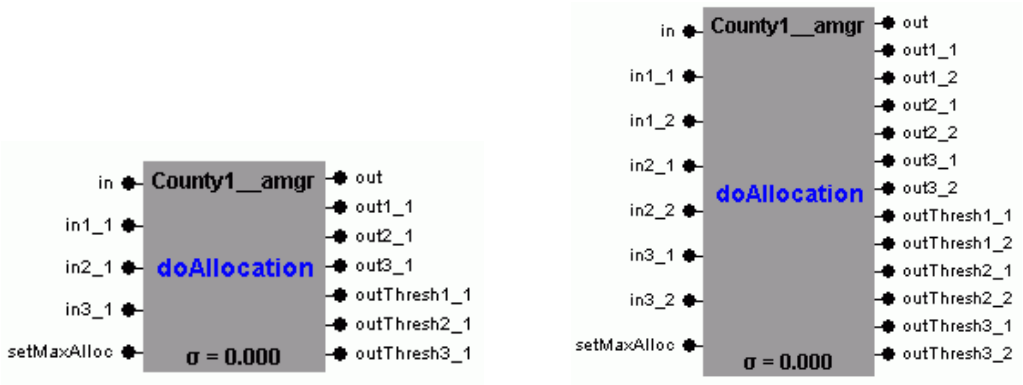


Figure 24:A snapshot of the County structure. City 2 is expanded for illustration

Figure 24 shows three cities in direct control of a Sampling Manager. The city as a black box has 4 incoming ports and 2 outgoing ports. This corresponds to the number of

sensors contained in the city. Each sensor requires 2 incoming ports and one outgoing port. The only information that comes out of the city boundary is the activity level detected by the sensor, which then goes to the Sampling Manager.



(a) (b)

Figure 25: Showing the evolution of the Sampling Manager as sensors increase

The Sampling Manager shown in the Figure 25(a) above shows the simplest construction of the component. The left figure shows the structure of the interfaces when there exists only one sensor assigned to the Surveillance area. The notation *in2_1* implies that the port *in* belongs to *city 2* and within it, the sensor *1*. The same address notation belongs to the outports. So each sensor has been assigned one input port and two output ports:

Input port:

- in + location of sensor

(used as the interface for receiving activity from sensor)

Output port(s):

- Out + location of sensor

(used by Sampling manager to notify the sensor of its revised Sampling rate)

- outThresh + location of sensor

(used by Sampling Manager to updated the sensitivity of the particular sensor)

The rest of the ports (Inport(s): *in* and *setMaxAlloc*; Outport: *out*) are for the functioning of the Sampling Manager and receiving and sending messages to the superior level. It sends its resource distribution to higher level using the *out* port and receives its maximum allocation from superior level at the inport *setMaxAlloc*. The Figure 25(b) shows a modified version of Sampling Manager. This Sampling Manager is managing cities that contain two sensors each. As in Figure 25(a), the number of cities is same, i.e. three. Notice the address location as indicated by in2_1 and in2_2 (city 2 containing sensor 1 and sensor 2). The Sampling Manager is a *dynamic structure that has the capability to expand and contract on a need-to-basis*. It is provided with a capability to evolve as the sensor system evolves and modify its interfaces, though keeping the logic of providing and distributing resources intact. This capability is facilitated by variable Structure DEVS [20] that was built as a part of this research. The logic inside the Sampling Manager is described in Section 6.2.

As shown in the City structure in the previous section, each city has its boundary and the Sampling Manager and other entities communicate to the City using this boundary. The information that comes into the city is the Sampling rate of the sensors inside this boundary and the sensitivity index of the sensor. Both of these values are sent by the Sampling Manager. Figuratively speaking, the sensors are acting with the outer world,

Sampling Manager specifically, and getting updates about their behavior. The Surveillance area inside the city is a completely independent identity with its own behavioral map, as discussed in Section 4.1.1

4.1.4 State (adaptiveSensorNet)

The State is one level up in hierarchy over the County level. Its structure maps that of a county, except the fact that a State is aggregation of many counties controlled by an Attention Manager. The description of county is addressed in the previous section. The function of Attention Manager is analogous to Sampling Manager. At this level, each county is allotted a maximum allocation of resources, which gets distributed to the sensors inside the county via Sampling Manager. A typical state looks like:

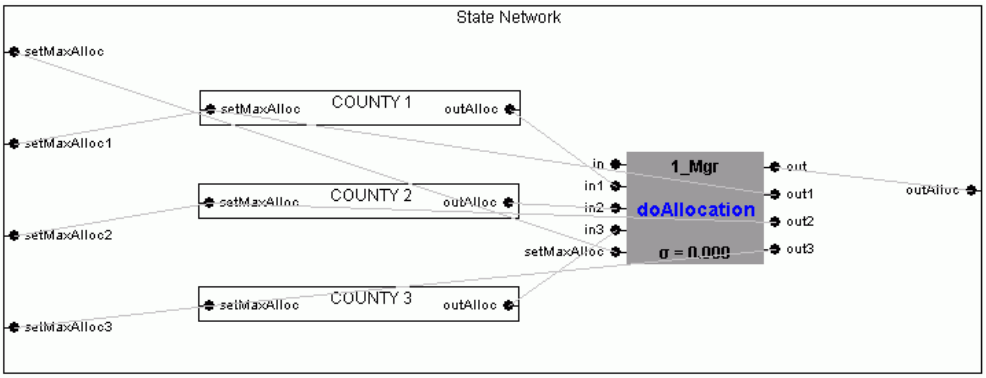


Figure 26: Top level view of a State constituting Counties

It shows three counties under the control of a single Attention Manager, which assigns them their maximum allocation rate. The manager may be considered as containing a

communication backbone that distributes bandwidth among different counties depending upon their total activity. It is not responsible how the resources are distributed inside the county and is totally ignorant of the structure inside the counties. The manager sees the counties as they appear at this level, a black-box. Each county shown above have different behavior and are not identical to each other. Each has a different loss factor of how information gets transferred to the sensors. Its behavior along with the behavior chart of the counties is shown below.

We expect a behavior of distribution of resources from this level to lower levels (inside the counties) till the allocation reaches the sensor that displays the highest activity. The algorithms implemented at different levels are independent of implementation of the hierarchical construction as the Manager at every level distributes what's available with it. We observed a similar behavior when we ran experiments.

4.2 Experimental Frame Component Behavior

Each Simulation scenario is a controlled experiment. We have seen that the system when left alone to run, displays a certain behavior. We need to test the boundary of our system when its subjected to various strains through the experiments conducted using Experimental Frame. In order to introduce external conditions that will drive the simulation, we introduce a component called **Scheduler**. The job of this component is to schedule events that bring about the change in the topology or the sensitivity of the system. Although there is no real 'generator' in the system, as the Experimental Frame formalism puts it, the scheduler may be considered as generator that generates selective events pertaining to the scenarios that are discussed in the Section 5. The other component that holds a place in the Experimental Frame is the Analyzer (transducer) that gathers the results of the simulation and provides analysis. The Scheduler and the Analyzer are the specialized versions derived from Generator and Transducer described in Section 3.2.

5. Scenarios

As mentioned earlier, the experiments are conducted by using the concepts of Experimental Frame. Our Frame has three components:

a) Scheduler

b) Model under Investigation

c) Analyzer

The component (b) changes according to the scenarios and so does the scheduler tasks. The tasks of scheduler are explained in detail with reference to the scenarios discussed below:

5.1 Static Topology and fixed number of Sensors

In this scenario, the topology of the system remains constant. There are no additional sensors added, no additional cities added. The boundary of the model under consideration ends at the county level and experimentation is done within the county. The number of components in the experiment doesn't fluctuate. Now, keeping the structure fixed we introduce a disturbance in the system, instigated by the Scheduler. The disturbance may be constructive or destructive depending upon the task it accomplishes and its effect on the resource allocation.

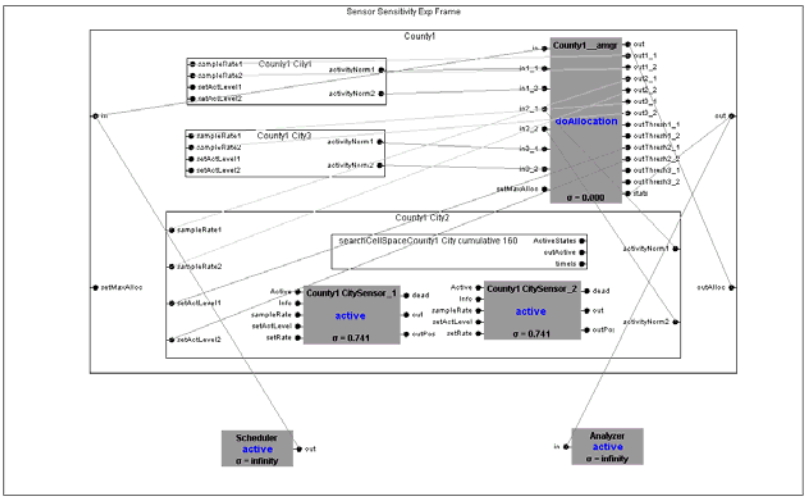


Figure 27: Experimental Frame showing the model with Scheduler and Analyzer

The basic job of the Scheduler here is to cause external event into the system that would bring about the change in the sensitivity of one of the sensors chosen. When the sensitivity of one of the sensor changes, we expect to observe a change in the behavior of the sensor Area as a whole.

5.2 Static Topology and variable number of Sensors

In this scenario, we introduce more sensors into a city when the activity of the city is over certain threshold. The system is capable of adding and removing sensors and keeps the sampling rate of sensors well within a range. As mentioned earlier, the activity in the area is unpredictable; the decision to add/remove a sensor is taken on the run-time. This experiment is conducted at the County level, composed of cities containing sensors. The experiments demands the presence of a Sensor Manager inside the city that keeps the

count of the number of sensors in play and take the corresponding decisions based on the activity of the area. The figure below explains the Experimental Frame.

The top part of Figure 28 displays the system at its normal state. To show a clear picture only the couplings related to the expanded city 2 are shown, though couplings between city1 and city3 with the Sampling Manager do exist. The lower part of Figure 28 shows the resulting system after the addition of the third sensor in city2. Notice the updating of city interfaces and the interfaces of the Sampling Manager to handle the new sensor that has appeared on the scene. When the job of the sensor is complete the system is restored to its original states. The status of the Sensor Manager is also shown in the figure which displays that the system is aware of the arrival of the third sensor. Only the new couplings that resulted from the arrival of third sensor are shown in the figure. The Scheduler and the Analyzer do their respective jobs.

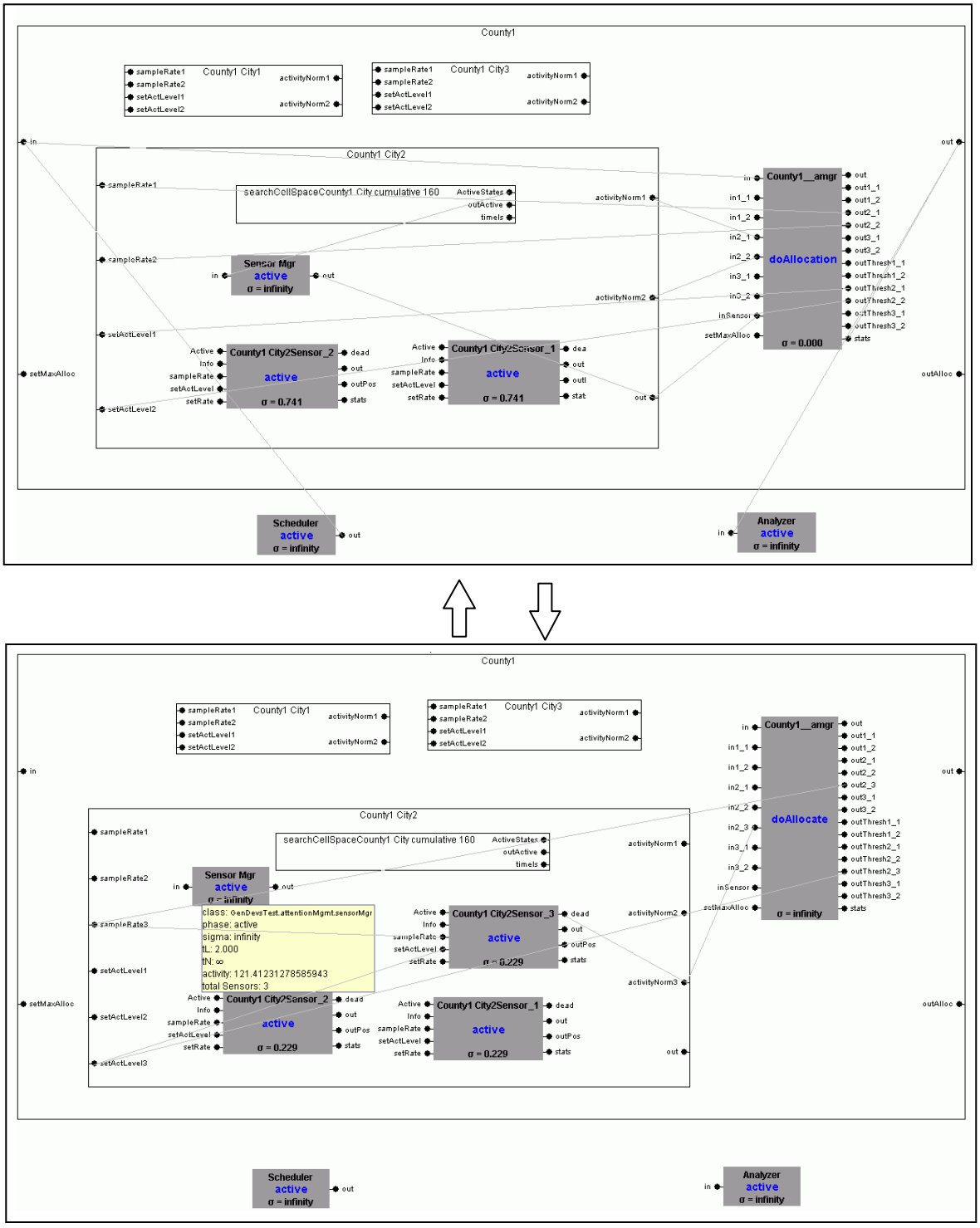


Figure 28: Experimental Frame structure showing variable structure

6. Theoretical Analysis

This chapter deals with the mathematical analysis of some of the key factors of the system described in previous sections. It also tries to define the boundaries conditions for the parameters involved therein. The parameters that will be discussed in the following subsections are:

a) Incoming activity: **A**

b) Sampling Rate of Sensor: **S**

c) Threshold of Rate Estimator: **T**

d) Number of Data Messages in system per sec: **m**

e) Number of House-keeping Messages in system per sec: **z**

f) Rate of producing jobs (System Snapshot data) by sensor: **R** ($R \alpha S$)

6.1 Presence of Rate-Estimator with Sensor

6.1.1 Estimator Threshold

The AdSNet System has a working delay of 1 sec. This implies that any change in activity has to persist for around 1 sec to actually receive a new Sampling rate from the Sampling Manager. It is kept as such so that any small accidental change in the Activity measure doesn't produce change in the Sampling rate updates thereby making system more stable. The system makes sure that the incoming activity has persisted long enough before it communicates this change to the next level. This buffering/working delay can be

defined based on the system requirements. It is approximately 1 sec for the default case. Having set that, let us also define the maximum Job rate of sensor. Each sensor has a maximum production-rate of R Jobs/sec. This implies that given a Sampling Rate of 1, the number of jobs produced per sec is R. The default Sampling rate of a sensor is certainly less than 1 and consequently, the default Job-rate will be less than R.

Let us assume that the default activity (background) activity is A units. So every time a sensor reports activity (to the Rate-estimator), it reports this value (A units). Given a system delay of 1 sec,

Total amount of Activity reported per second $\xi = \mathbf{A} \mathbf{X} \mathbf{R}$

Now the Rate-Estimator reports the Activity to the Sampling Manager when the Activity surpasses the threshold T. As a result, if the delay factor is 1 sec, then $\mathbf{T} > \xi$. This condition will enable the Rate-estimator to report the Activity (after processing it, see Section 4.1.2) after atleast 1 sec (iff the sensor is operating at Maximum Sampling Rate).

6.1.2 Number of messages in System (Bandwidth Usage)

This section discusses the effect of the Rate-estimator on the bandwidth usage. The Bandwidth usage is determined by the number of messages in the network queue at a particular instant of time (see Section 7). Let that number be N.

So,

N = No. of Data Messages + Housekeeping Messages

Consider the case when Rate-Estimator is not present. In such a scenario, every data message (reporting of Activity) from the Sensor reaches the Sampling Manager. Let the number of sensors in the System be *n*.

∴ Total data messages per sec = *n* X jobs produced by sensor per sec at default rate

$$m = n\,R \qquad (R <\neq \text{Max Job rate})$$

and, Total number of housekeeping messages = z per sec

Hence, Total messages in Network Queue $\mathbf{N_{\not\in}} = \mathbf{m} + \mathbf{z}$ **(1)**

Now, considering the case when Rate-estimator is present,

Total data message per sec $= n \times \frac{R}{T} = \frac{m}{T}$ (using 1 from above)

As all the data message goes from the Sensor into the Rate-estimator and the Rate-estimator produced only one data output once the threshold is reached, the effective data message is reduced by a factor of T. The Rate-estimator is an integral part of the sensor and the data messages from the sensors are not released into the network. In other words, only an output from the Rate-estimator is used for communicating information.

And, House Keeping messages per sec = z

They are taken to be same for the above scenario. This assumption is based on the simulation experiments that we performed in the following section.

Hence, Total messages in Network Queue $\mathbf{N_{\in}} = \frac{m}{T} + \mathbf{z}$ **(2)**

The network bandwidth usage is proportional to the number of messages in the network queue. Taking ratios of $N\in$ with $N\notin$, gives the ratio of bandwidth usage for a system with Rate-estimator and without Rate-estimator.

$$\therefore \quad \gamma = \frac{N_{\in}}{N_{\notin}} = \frac{\frac{m}{T} + z}{m + z} = \frac{\frac{1}{T} + k}{1 + k}$$

where,

$$k = \frac{z}{m}$$

Clearly,

γ can be approximated to $\frac{1}{T}$ when $m >> z$ i.e. when the data messages are greater than the housekeeping messages. This is a practical assumption when the sensors are in a steady state and no housekeeping is required to maintain the system•. Recall that in a discrete event scenario, message passing only occurs when there is a change in the state of the system. Consequently,

$N_{\in} \cong \frac{1}{T} N_{\notin}$, which implies that the messages in the Network Queue are reduced by a factor of T, when Rate-Estimator is in operation.

• There are health-check protocols that maintain the system and are often periodic. Since we can't reduce the number of these messages, they are not considered for developing the relation for Bandwidth usage for either case (using Rate-estimator or without Rate-estimator).

6.2 Algorithms inside the Sampling Manager

The Sampling Manager implements the algorithm to assign the Sampling Rates to different sensors inside the Sensor-Areas. As described in Section 4.1.3, it receives the Activity-strength from the Sensor-Area and responds back with an updated Sampling Rate for the sensor. The underlying logic implemented has the general algorithm described as follows:

1. Get activity value *currVal* (*at the in+location_of_sensor port, see Section 4.1.3*)
2. Get previous activity value associated with this sensor from its own database
3. Calculate the difference
4. Update the *Sum* parameter which reflects the total amount of resources being used by all the sensors.
5. Update its own database with this new value *currVal*
6. Get the *lastSum* value from the database (total sum of resources in the previous iteration and during the last assignment of sampling rates to sensors)
7. Compare *lastSum* and *Sum* values and get the difference
8. if the difference is greater than the quantum *outQuant* of this manager, then get ready to send revised Sampling rate (*outQuant* may be referred as the sensitivity of the Sampling Manager and its sends the revised rate to the sensors only if the difference value in the above step i.e. the new total allocation *Sum* has made a quantum change)
9. replace the value of *lastSum* by *Sum*
10. Get the previous Sampling rate for this sensor
11. recalculate new Sampling rates based on either of
    a) *Normalized*-Sum (NS) Rule (Section 6.2.1)
    b) *Normalized*-Max (NM) Rule (Section 6.2.2)
    c) *Tunable Alfa*-Beta (TA) Rule (Section 6.2.3)
12. Update the Sampling rate of this sensor in its own tables
13. Send new Sampling rates to the sensors.
14. Repeat the cycle with the above 13 steps indefinitely

Figure 29: Algorithm inside the Sampling Manager

Depending upon the experiment performed or the environment of AdSNet System (constricted resources or unlimited resources), appropriate algorithm in Step 11 is chosen and the sensors are assigned the updated Sampling Rates.

6.2.1 Normalized-Sum Rule (NS Rule)

The NS Rule is based on the condition of Limited Resources (the situation when there are no free resources available and all the resources are already distributed). The Sampling Manager has fixed resources (eg., bandwidth, channel capacity etc.) and it has to distribute them among the sensors. Consequently, the most active sensor (Sensor-Area reporting the maximum Activity) should be granted maximum amount of resources as compared to other sensors. It can be represented as:

$S_k = \frac{A_k}{\sum_j^i A_i}$	where, $i < k \leq j$ $i,j \subset N$ and $j \geq 2$ S_k = sampling rate allotted to sensor k A_k = the incoming activity rate of sensor k

This Rule has the following properties:

a) As Sampling Rate S is directly proportional to the percentage of resource allocated, $\sum S_k = 1$

b) Any increase in the Sampling rate of any sensor brings about the decrease in the rates of other sensors and vice versa. This is due to the limited capacity of the system and the Sampling Manager having limited resources. The Sampling Manager distributes 100% of its resources to the sensors. Consequently, in order to address in demand, it has to withdraw some of the resources from other sensors.

c) Every update in the Sampling Rate of a single sensor re-adjusts the Sampling rates of all other sensors in the AdSNet system. Any single update disturbs the current configuration of system. This is unacceptable when the incoming activity **A** is of competing nature (**A** not very high as compared to background activity. See Section 7.1.1)

d) As Job Rate R of sensor is also proportional to the Sampling rate of sensor, high Sampling rate produces high Job Rate. If the numbers of sensors in system is large, and in light of the property (a), the Job Rate greatly reduces.

6.2.2 Normalized-Max Rule (NM Rule)

The NM Rule is based on the condition of unlimited resources (situation when there is always an amount of free resources available with the Sampling Manager). When the resources are freely available with the Sampling Manager, it can distribute them to the sensors and let them Sample at the maximum rate. This would enable each sensor working at its maximum Sampling Rate. In order to focus and bring attention to the highest Activity or highly active sensor, all the Activity reported by the sensors is normalized by the Maximum Activity amongst them. As a result, the Sampling rate becomes a function of Maximum Activity present in the system at a given time. It can be represented as:

$$S_k = \frac{A_k}{A_{\max}}$$

where,

S_k = sampling rate allotted to sensor k

A_k = Incoming Activity and

$A_{\max}$ = Current Maximum Activity available with the Sampling Manager for any particular sensor.

This Rule has following properties:

a) The Maximum activity $A_{\max}$ at any given instant defines the state of the system. If any sensor has a high Activity, it forces other sensors in the neighborhood (under the same Sampling Manager) to increase their Sampling rate (through the Sampling Manager).

b) If the incoming $A_k < A_{\max}$, then the sensor is assigned the Sampling Rate according the definition said above, without changing or updating any other Sampling rates of other sensors (provided $A_{\max}$ is constant during the arrival of new activity). The arrival of any new Activity does not disturb the configuration of the system as long as its less than the current maximum value.

c) If the incoming $A_k > A_{\max}$, then the incoming Activity replaces the current value of $A_{\max}$ and becomes $A_{\max}$ itself. As the system has new Normalization factor, all the Sampling rates of other sensors are updated based on this change, with the sensor reporting this Activity A_k receiving the Sampling rate equal to 1 (according to the definition). As a result, the new activity grabs

hold of the maximum resources that it can utilize iff it has the Maximum Activity as compared to other sensors in the system.

d) The definition of NM Rule is independent of the number of sensors in the system as it is dependent only upon the new incoming activity A_k and the maximum activity $A_{\max}$ amongst the sensors already present. The assigned Sampling rate is independent of the number of sensors operating at any given time in the system. As a result, the Job rate is also independent of the number of sensors. **The NM Rule promotes scalability**.

e) Each sensor can work to its maximum potential (producing the max possible Job Rate) unlike the NS Rule (where it has an upper bound) irrespective of the loading of the system.

f) It helps us to define that **providing resources** and **providing attention** are two different operations. Providing attention leads to providing resources but not vice versa. Property (c) makes it more evident when the new incoming activity registers itself with the Sampling Manager and causes reconfiguration of the entire Sampling Rate table based on its value. On the other side, property (b) provides the resources to the incoming activity but no attention.

6.2.3 Tunable Alfa-Beta Rule (TA Rule)

The TA Rule exploits the benefits of the NM Rule and makes it more realistic by using negative feedback mechanism. It incorporates the previous sampling rate of the sensor

(reporting the new activity) in the determination of the new sampling rate. Though the overhead increases at the Sampling Manager end as it has to maintain a separate table for storing the previous Sampling rates, it makes the system more realistic. Keeping the previous sampling rate in the calculation brings a little amount of "inertia" in the system. This inertia factor is tunable and is presented as follows:

Let the change in Sampling Rate be defined by, $\frac{dS_k}{dt}$

$$\frac{dS_k}{dt} = \frac{A_k}{A_{\max}} - S_k$$

$$S_k(t + dt) = S_k(t) + dt\left(\frac{A_k}{A_{\max}} - S_k\right)$$

$$= S_k(t).\,(1 - dt.\,S_k) + dt\left(\frac{A_k}{A_{\max}}\right)$$

which gives us,

$$\boldsymbol{S}_k' = \alpha\, \boldsymbol{S}_k + \beta\, \frac{A_k}{A_{\max}}$$ where S_k' is the new sampling rate of Sensor k

The TA Rule has the following properties

a) The determination of Sampling rate is tunable. Based on the values of α and β, the system can be made more sensitive or more sluggish.

b) α and β are correlated by the function $\alpha+\beta = 1$.

(boundary condition when $S_k' = S_k$)

c) Increasing the sensitivity (β) automatically reduces the inertia factor (α)

d) This Rule makes the transition to new Activity level smoothly. There are no sharp rises and falls in the values of new Sampling rate when compared with previous Sampling rate.

e) Higher the β, the fast the responsiveness of the System (through Sampling Manager) and higher the α, the more inertia the system has and more averse to change to new Sampling rate.

f) Striking a balance between α and β defines the responsiveness of the System.

7. Quantitative Results

This section presents the analytical results obtained after conducting experiments in the fixed topology scenario with minimum ten sensors at level 1 (County level). The system has two independent invariants:

a) The new incoming activity that tries to grab the attention

b) Estimator-threshold of the sensors already working.

The threshold of the activity considered here is the threshold of the attached Estimator that advertises the findings of the sensor. Various experiments are devised that show the relationship between the said invariants and their effect on the response time of the new incoming activity. The response-time is defined is the time it takes for the new activity to grab attention and the required resources.

7.1 Flat Simulation

7.1.1 Response Time for a new Activity to grab attention

This experiment depicts a scenario when the sensors are running at a default activity and a new activity tries to break in the system. The new activity is varied from an activity value lower than that of the default activity level and is increased to a value much higher than the default level. Response time is defined as the time from the instant when the sensor starts reporting the increased activity to the instant it receives the increased

sampling rate from the Sampling Manager. The resulting response time is plotted against the corresponding activity value. Figure 30 shows the results in the graphical manner. The threshold of Rate-estimator of all the sensors is kept constant and the only parameter that is varied is the activity level of the new incoming activity. This is a single level simulation and the Sampling Manager has finite resources. The environment consists of three sensors at their default activity value 100 and Rate-estimator threshold =3500. Maximum job production rate = 9 jobs/sec. The simulation experiment was run twice and the same values were obtained in both the simulations.

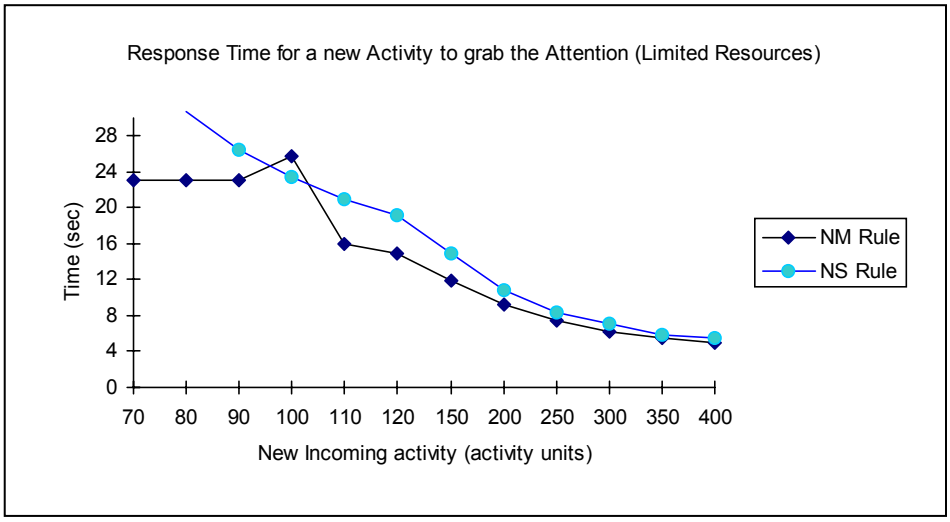


Figure 30: Response Time for new activity to grab the attention

Comments and Inferences:

a) It takes less time for the new activity to break into the system when its activity level is much higher than the activity level of already running sensors. As the activity value is increased, the response time decreases (see the curve for activity higher than activity =100).

b) When the new incoming activity level equals 100, there is a marked increase in the response time, as the system takes quite a while to provide attention to an activity of similar potential(competing nature). A bell curve behavior is spotted at around value 100. As discussed in the Section 6.2, NM Rule is more responsive and sensitive than NS Rule, we see much expressed competition in the graph above.

c) The NM Rule takes less time for every reading on the scale (except at incoming value of 100). The reason is attributed to the variation in the job rate. NM rule has higher job rate (it can vary from 0job/sec to 9 jobs/sec and it does go upto 9 jobs/sec when the activity gets Sampling rate equal to 1 while the NS rule can never go to its maximum job rate).

d) For values lower than the default activity level, and region away from the bell-curve, the system takes long time to provide attention and resources. This is attributed to the fact that an incoming activity with low potential takes more time to produce meaningful information and as a result high response time.

7.1.2 Estimator Threshold v/s Response Time

This experiment calls for the behavior of the estimator attached to the sensor. It shows the effect of the threshold of the estimator on the response time of the incoming activity. As discussed earlier, the estimator is an abstract model for the validation model of the sensor, its threshold value is a symbolism of its decision making power. Lower the threshold, powerful and quick is the ability of the Estimator (with the caveat that it satisfies the condition in Section 6.1.1) to come to conclusion and validate the information sent by the sensor and vice versa. NM rule is used in this experiment for

better responsiveness. Maximum Job Rate is 9 jobs/sec and Maximum Activity values taken by any incoming new activity is 300 units. The minimum threshold level of any Rate-estimator can theoretically be not lower than 2700 (according to the conditions described in Section 6.1.1). This experiment has ten sensors each working at an estimator-threshold of 4000, while the incoming activity has its estimator-threshold taking different values. Different lines in the graph depict different values of the incoming activity. In addition, the default activity level of the sensors is again 100. The response time is plotted against the estimator-threshold of the incoming value.

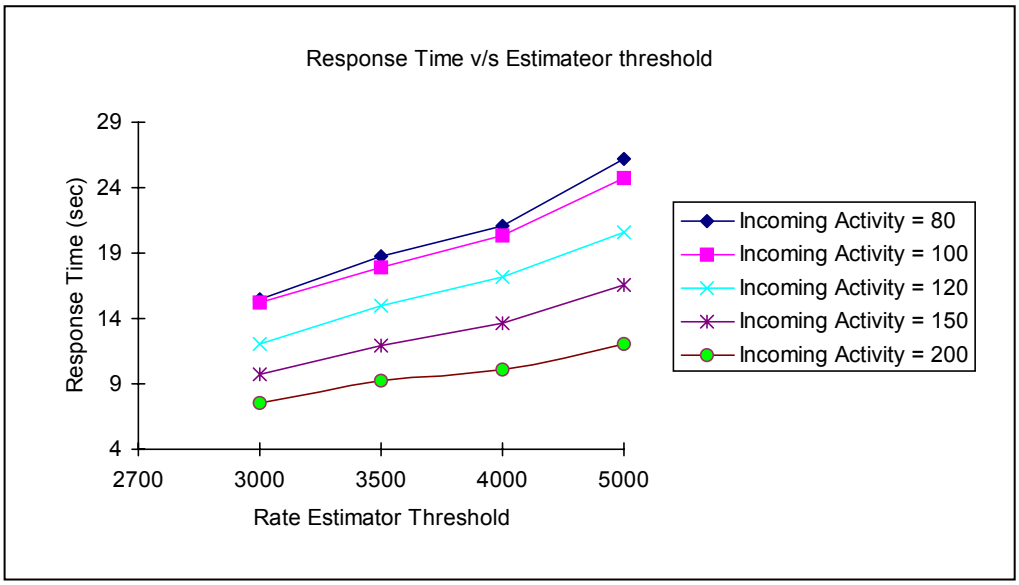


Figure 31: Graph for Response Time v/s Estimator Threshold

Comments and Inferences:

a) The threshold of the estimator has a strong impact on the response time of the incoming activity and the higher the threshold, higher the response time.

(assuming that the threshold of the estimators of already running sensors remains constant).

b) If the incoming activity is higher than the populating sensor, the response time increases along with the threshold of the estimator.

c) Maximum variation in response time occurs when the incoming activity is same or close to the already running sensors-activity (incoming activity = 100 or a little less than 100).

d) An area irrespective of its Estimator-threshold value can grab attention if its activity level is higher than the already running sensors.

e) When the incoming activity is lower (incoming values: 90 in the graph above) than the activities of the older populating sensors then response doesn't show much of a different in response time as in the case when incoming activity is greater than 100 (eg, 120). Notice that in Section 7.1.1 the response time stays constant and close to activity value 100 for values less than 100.

In an environment where sensors are operating at a certain threshold, a new sensor can report its activity and can get noticed by tuning its Threshold value. In case of noise channels or chattering environment, this feature facilitates focusing of attention to the new incoming activity.

7.1.3 Scalability v/s Response Times

This experiment calls for the scalability issues and the effect on the response time of the incoming activity as the number of sensor increases. Sensor number is increases from a total value of 3 to a number of 300. Different lines in the figure below show different estimator-threshold values of the incoming activity. The default activity level of the

sensors is 100 units and their estimator-threshold values equal 3000 units. The incoming activity value is 150 units. As a result, the incoming activity level being higher, it must break into the system and grab the attention and the resources. The figure below shows the plot of Response time with the number of sensors.

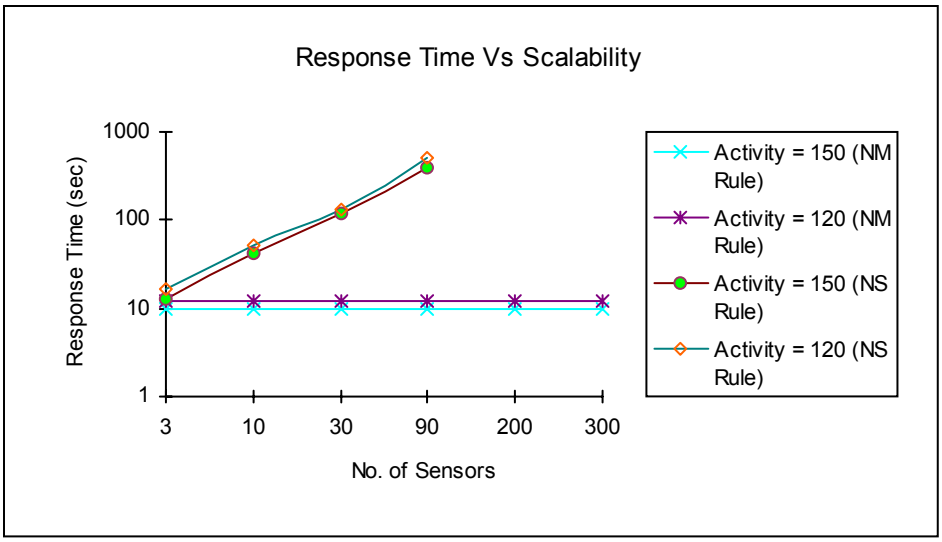


Figure 32: Response Time v/s Scalability

Comments and Inferences:

a) Response time of any incoming activity at any threshold value is independent of the number of sensors already active for NM Rule and directly proportional for NS Rule

b) If the activity level of the incoming activity is higher, it will grab the attention irrespective of the current resource distribution

c) The architecture with NM Rule is scalable and maintains its qualitative behavior in the range of 3 – 300 sensors.

7.1 Hierarchical Simulation (Response time v/s new incoming activity)

This section deals with the experiments conducted with a 2-tier system. Previous section consisted of only cities under the control of a Sampling Manager. This section analyses the system of many cities under the control of another manager at County level. This manager provides resources to the constituent cities. For more description, see section 4.1.4.

7.2.1 Limited Resources at Level 1 and at Level 2 (NS Rule at both Levels)

This experiment deals with a finite capacity system that has fixed resources at any level of the hierarchy. As a result, there exists competition with sensors in grabbing attention when it comes to same activity level. The system is expected to behave according to the results of experiment 7.1.1. Since this is the simplest n-tier architecture, the lower hierarchy calls for resources from the higher level which again has the same behavior. This experiment is designed to see two levels, one of them under the other's supervision, in interaction and their emergent behavior. This particular experiment is conducted with three counties and each county contains 5 cities and each city contain one surveillance area and one sensor. All the 15 sensors are working at the default activity level 100 and the estimator-threshold 3500. The interaction of both Level 1 (city level) and Level 2 (County level) is studied by bringing a new incoming activity level and observing the response time it takes to break-into the system. The experiment is done on the same lines as that of Experiment 7.1.1 with adding another superior level to the system. The

simulation results support the findings of Tsotsos (see [4],[9]). The legends in the graph below have their usual meaning.

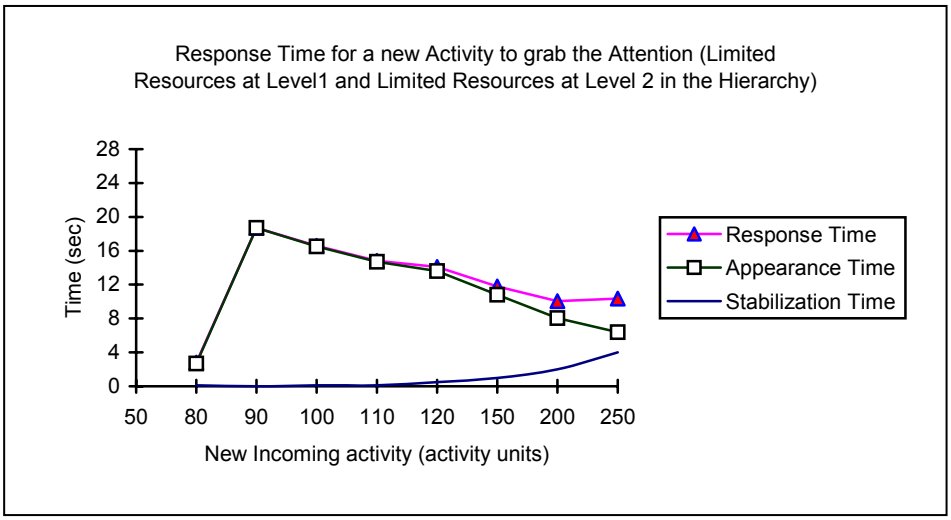


Figure 33: Response time v/s New incoming activity (Limited Resources at Level 1)

Comments and Inferences:

a) Resources are demanded from the higher level 2 by the level 1 Sampling Manager and is then allocated to the sensors at its own level

b) The Value of response time is less when compared to the values in experiment 6.1.1. (which constitutes only one level). Notice the Stabilization time in particular. The Stabilization time is high corresponding to the high incoming value. Higher the incoming value, the longer it takes for the system to come back to steady state, after it has been disturbed by the advent of this high activity.

c) The competition is now mitigated as the resources are provided by the above level on demand by the City.

d) The response time of the new incoming activity still observes a bell-curve when its activity level is equal to that of the sensors (activity value =100), but the value of response time is less as compared to experiment 6.1.1

e) The system maintains its characteristic behavior with this hierarchical setup with reallocation and redistribution of resources taking place dynamically.

The plot obtained above requires some explanation. Notice that it has three curves depicting the Appearance time, the Stabilization time and the Response time. All the plots in previous experiments showed only the Response time. In this particular experiment, the Response time is the sum of the Appearance time and the Stabilization time. The Appearance time is defined as the time at which the new activity grabs the attention and receives its new updated Sampling rate. The Stabilization time is referred to as the time taken till it continues to get a revised Sampling rate even after grabbing attention or it the time taken for the background sensors to become steady in their operation. This behavior is *an emergent behavior* of the system. We described NS Rule in Section 6.2.1 that any new change in the system results in adjustment of the whole system. This is evident in the Stabilization time as the system takes time to settle down. The more the value of new activity, the more it takes time for the system to settle down to steady state. On the contrary there exists no Stabilization time in NM Rule but there do exist Stabilization in TA rule (for experiments see Section 8.2.2).

7.2.2 Unlimited resources at Level 1 with limited resources at Level 2

This experiment is designed on the same lines with the same specifications of the system parameter as that of the previous experiment. The only difference lies in the way the Sampling Manager at the City level behaves. Though, their inherent logic remains the

same, the manager at City level is different to that of the manager at the County level. The Manager at the County level has no change in the implementation but there is change in one of the steps (Step 11, defined in section 6.2) for the City level Manager. The step 11 says that:

> Recalculate new Sampling rates based on either of
> *a)* *Normalized*-Sum (NS) Rule (Section 6.2.1)
> *b)* *Normalized*-Max (NM) Rule (Section 6.2.2)
> *c)* *Tunable Alfa*-Beta (TA) Rule (Section 6.2.3)

This experiment calls for the implementation of NM Rule at Level 1 of the hierarchy. Whatever requirement Level 1 has, it is directed to Level 2 which is then addressed accordingly. The experiment is done with the following setup. Each of the sensor has a Rate-estimator threshold of 3500, with a maximum Job rate of 9 jobs/sec. Each County has 5 sensors each and each sensor is having an activity of 100 units. The plot depicts the Response time to gather attention by the Incoming Activity (when the Incoming activity is greater than 100) and to gather resources (when the incoming activity is less than or equal to background activity).

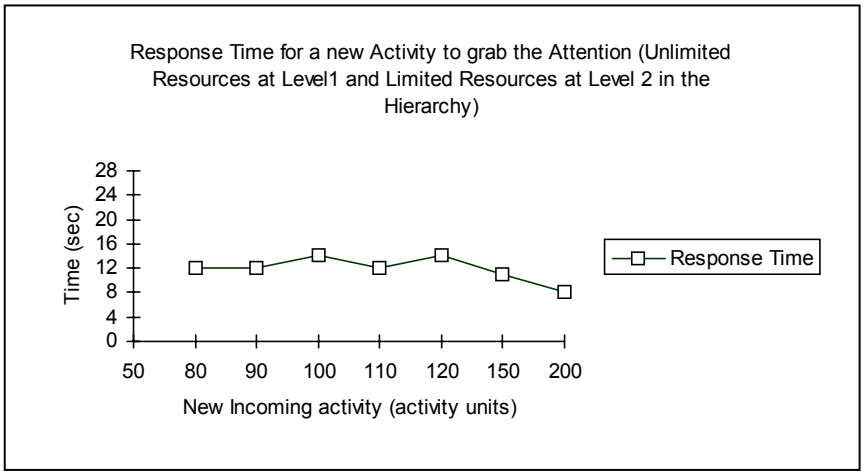


Figure 34: Response time v/s new Incoming Activity (Unlimited resources at Level 1)

Comments and Inferences:

a) The response time decreases as the incoming activity increases in its value

b) There exits fairly less competition when the incoming value (activity =100) equals the already running sensors at the same value. The plot practically has same Response time for values in the vicinity of activity-value 100.

c) The resources are demanded from the level 2 by the Sampling Manager at Level 1 in case of any competition as the resources are always available on-demand. The request is just forwarded to the next higher level and it is met with, always.

The results of this experiment supports the finding of the work done by Tsotsos [9], which says that the emergence of attention happens only in systems or organisms with fixed limited capacity for information processing. As there is no fixed capacity at Level 1 and any requirement of resources is fed to the superior level, it gives an impression of unlimited amount of resources available with the Sampling Manager of that County. This implies that the sensors have never to compete with each other to grab the attention in order to get the resources. As a result, no bell-curve behavior is ever observed in this experiment even if the incoming activity is of competing nature.

8. Experiments with a Network model having link-delays

This section describes the modeling of the underlying network over which the said AdsNet system will be laid out. Its conjectured that the sensor network is more likely to be a wireless network. Nonetheless, this network model works for wired systems too.

8.1 Network Model

The network is generally considered a queue. Every data packet passes through some network delay, which constitutes the time spent by that packet in the network-links or through network-components (e.g. Routers and switches). Since, it is difficult to get a measure of how many links or components a packet goes through, it is impractical to get a cumulative count of time spent. We also know that a packet suffers delay proportional to the network loading. If the network is heavily loaded, the packet takes more time to reach the destination. Consequently, we can represent the network as a single queue.

We could implement the network as a single queue such that each packet comes into the queue and goes out of it in a FIFO manner, but this would restrict us to have only one network channel that stays same for every network packet. It would bind us with a network (a model of it) that has same properties and characteristics for every single flowing entity, which is packet (message) in this case. We developed a network model where in the network characteristics are programmable for every source destination pair and we can have different network characteristics in different sub-networks. As the

AdSNet is a hierarchical network and even in a single level we have different independent areas. Although, the network model is incomplete without having a network queue, it works in conjunction with this programmable model.

The conceptual models is shown in the figure below:

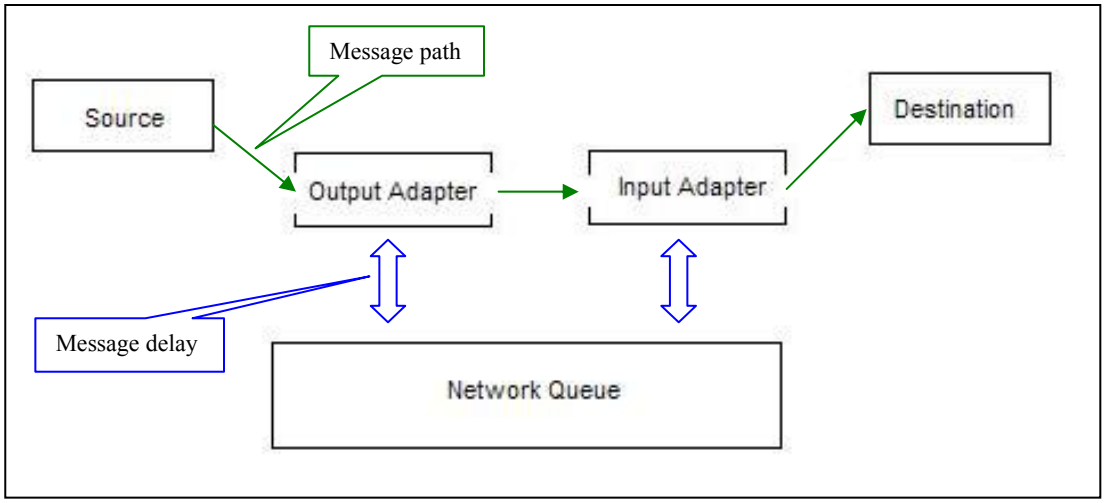


Figure 35: Conceptual Model of Adapter System with Network Queue

The model is implemented as a DEVS model shown in Figure 36. It has the following components:

a) Network Queue

This *atomic* component keeps track of how many packets are there in the network. It sends the Adapter system (of a source-destination pair) the delay time based on the current active packets in the network.

b) Adapter System

Each source-destination pair has an Adapter system between them that puts on additional delay to the default delay (value based on the Network Queue) in the packet-transit time. It can be programmed as desired. It contains of the following two components:

a) Output Adapter

This atomic component receives packet from the source and holds it for a certain period. This time-period is the network delay. Once it gets the packet it asks for the *duration to hold* from the Network Queue. If it is programmed, it adds or subtracts the time from the default time that it received from the Network Queue. After this duration has elapsed it passes over the packet to the Input Adapter.

b) Input Adapter

This is another atomic component as a part of Adapter System. It receives the packet from the Input Adapter and relays it to the destination. If there is another delay that needs to be put up at the destination end, it is programmed here.

The following figure shows a sample network of three source-destinations connected through the Adapter System. The network channel is defined by the *Network Queue-Adapter System* pair.

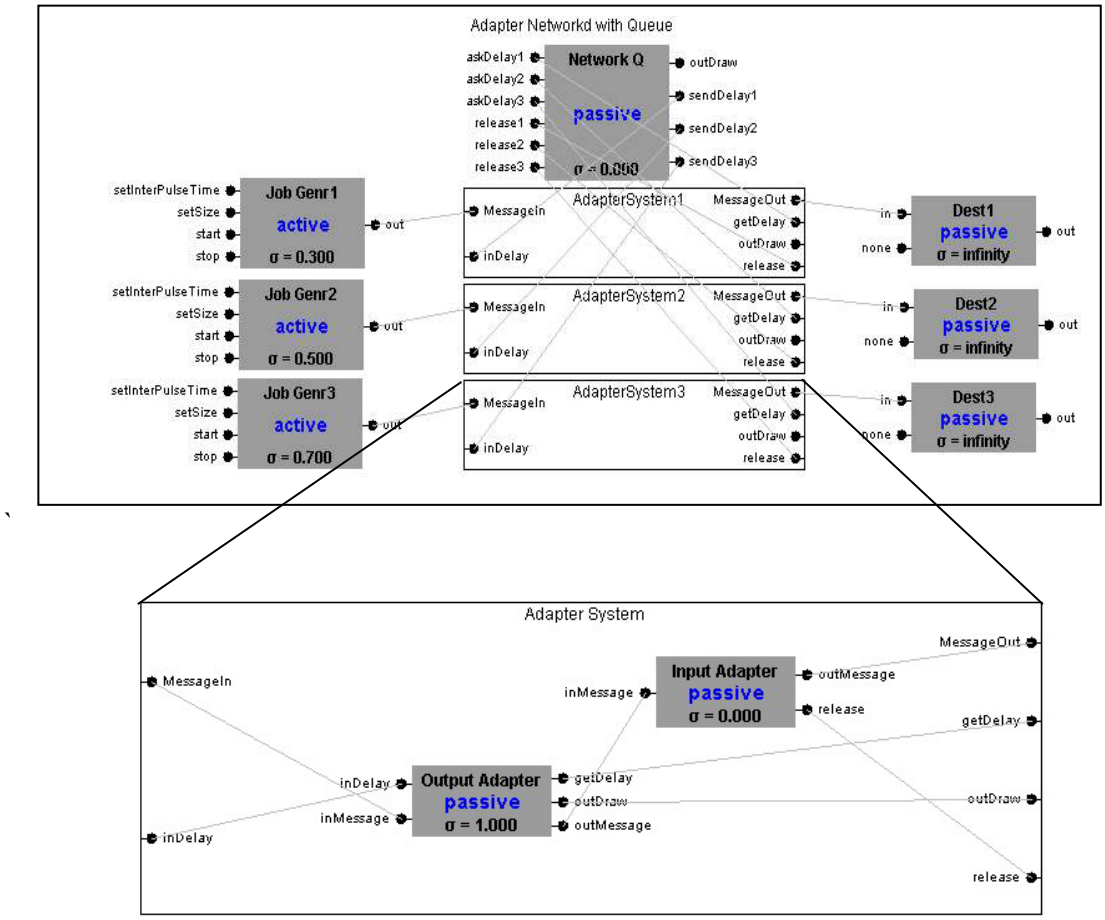


Figure 36: A Sample Network with source-destination pair connected through the Adapter System and Network Queue.

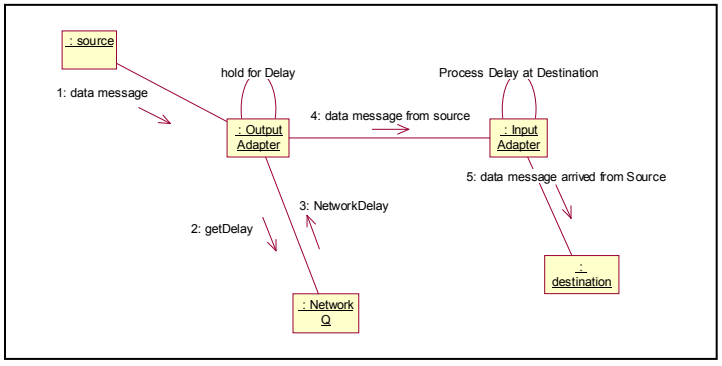


Figure 37: Collaboration diagram for a source destination pair showing message path

Figure 37 shows a collaboration diagram for a source-destination pair with a message flowing from source to destination. When applied to our AdSNet System, the resulting system looks like Figure 38.

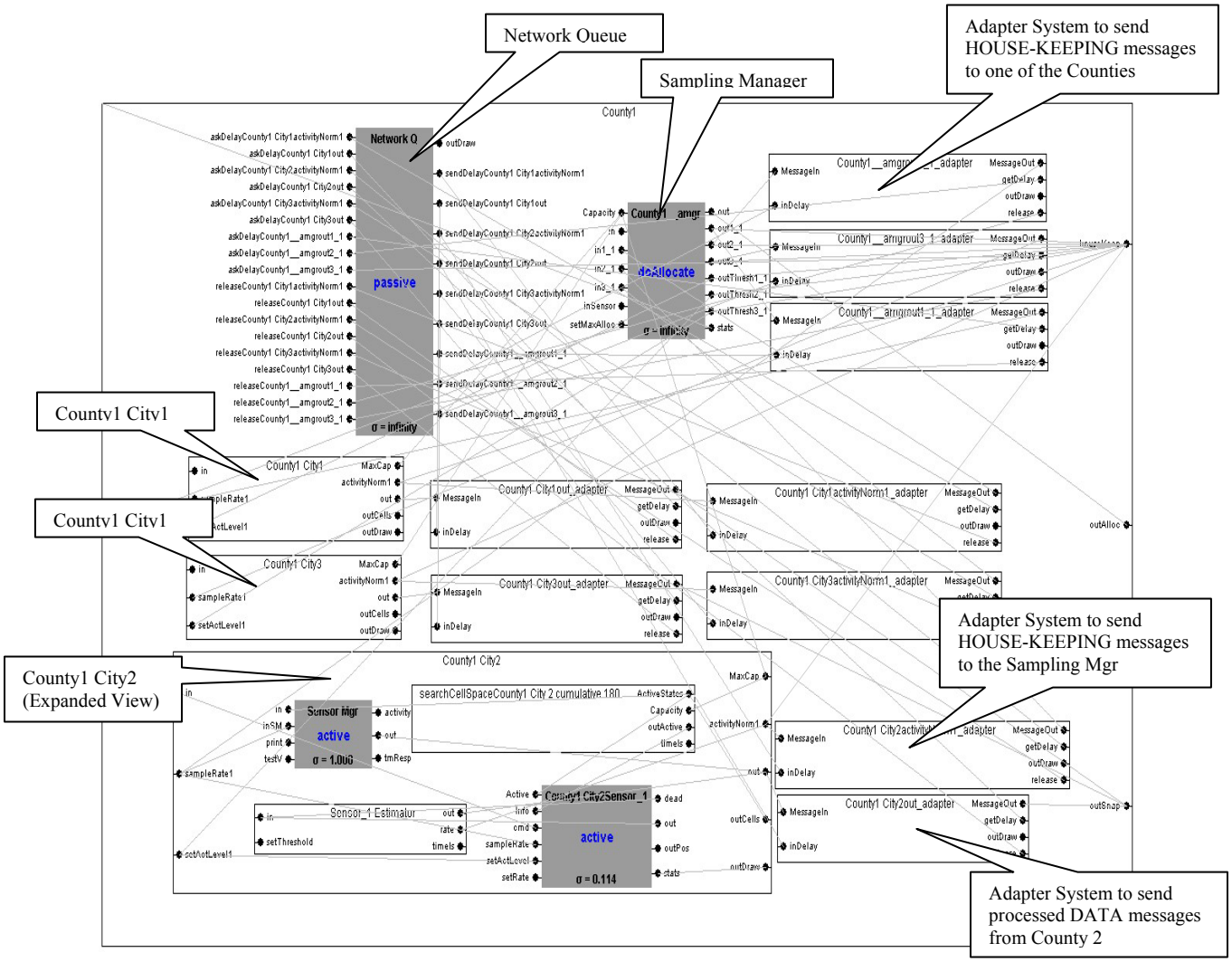


Figure 38: AdSNet System consisting of three cities in a County connected to the Sampling Manager using the Network Model described above.

The figure above is a layout of the AdSNet System that is laid out on the Network Model that is described previously. As described in Section 5, each city has two output ports: the housekeeping port (that sends the Activity rate to the Sampling Manager) and the Data output port (that sends out a meaningful snapshot of the County). Each of these ports is associated with an Adapter System that couples the output ports to their respective destinations. The figure above is similar to figure in Section 4.1.3 with the additions of Adapter System and the Network Queue.

8.2 Experiments with AdSNet laid out on Network Model

This Section deals with simulation studies on AdSNet System running on the Network Model described in the previous section. Some of the experiments done in Section 7 are repeated again to compare the results and provide more understanding to the functioning of AdSNet System. This section also contain new experiments that require Network Model be a component in the System design.

8.2.1 Response Time vs. New Incoming Activity

This experiment is exactly same as done in Section 7.1.1. The environment consists of five sensors at their default activity value 100 and Rate-estimator threshold = 3500. Maximum job production rate = 9 jobs/sec. The experiment is done for both NM Rule and NS Rule.

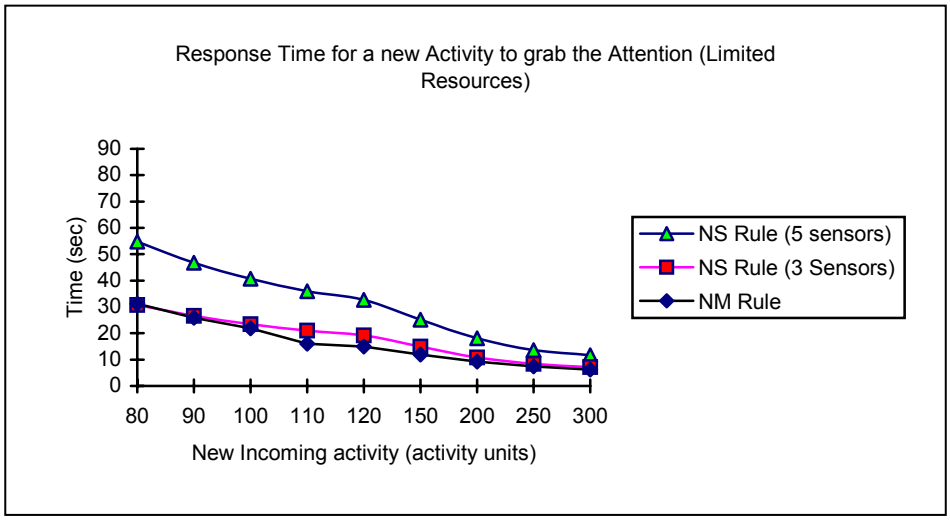


Figure 39: Response Time vs. Incoming Activity with Network Model

Comments and Inferences:

a) The above plot maps the behavior of experiment done in Section 7.1.1

b) It also maps the behavior of experiment in Section 7.1.3 which says that NS Rule take more Response Time as the number of sensors increases while NM Rule is independent of number of sensors in the network

c) The NM Rule shows a sharp rise in the vicinity of activity level 100 as to NS Rule. (NM Rule being more expressive than NS Rule)

d) The AdSNet system works well with Network model and an activity can break-into the system after passing through the network channel

8.2.2 Response Time with TA Rule

This experiment evaluates TA Rule. The previous experiment helps us arrive at a conclusion that AdSNet System produces same behavior with or without the Network

model, but its advisable to do all the simulation with the Network model as it makes the system more practical. The default setup is as follows: All the Rate-estimators have threshold of 3500. The maximum job rate/sec is 9 and all the other parameters have their usual meaning. The Simulation results are graphically plotted in Figures 50 and 51. In order to comprehend these plots we need to understand the response time behavior of the sensors. The following figure shows the actual plot during a simulation run with alfa= 0.9 and incoming activity of 200 (Fig. 50) and alfa= 0.7 and incoming value of 120 (Fig 51).

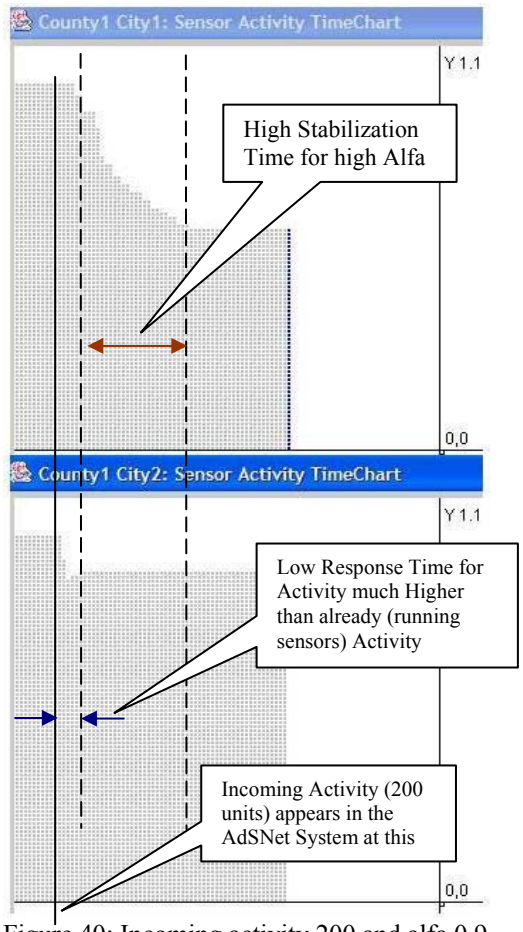


Figure 40: Incoming activity 200 and alfa 0.9
(top) Sampling Rate plot of the background sensors
(bottom) Sampling Rate plot of Incoming Sensor

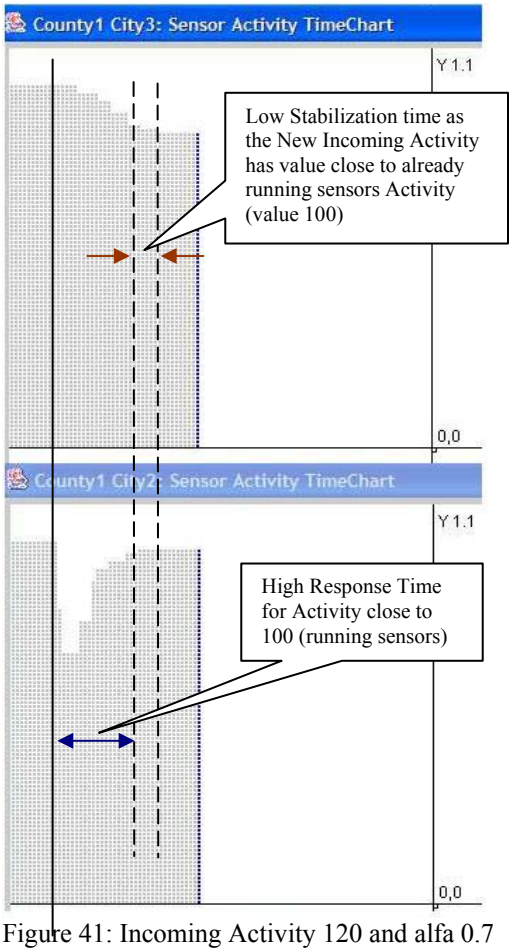


Figure 41: Incoming Activity 120 and alfa 0.7
(top) Sampling Rate of the background Sensors
(bottom) Sampling Rate plot of Incoming Sensor

Figure 40 depicts the case when the Incoming activity is very high as compared to the already running Activity values of different sensors. Experiments in section 7.1.1 and 8.1.2 show that the response time is inversely proportional to the incoming Activity value (for both NS and NM Rules). This graph shows that it is true for TA Rule also. The description of TA Rule in Section 6.2.3 says that higher Alfa α is responsible for increasing inertia of the system. This implies that the already running sensors will take time to adopt any change in the system. Even after the new Incoming activity has grabbed the attention, even then also the background sensors haven't accepted this change and they take their own time to come to steady condition. This time is termed as *Stabilization time*. The figure has two plots; the top curve shows one of the sensors already running in the system with Activity 100 and it represents the behavior of all other sensors having same Activity value (as in this case of controlled experiment); the below curve shows the plot of Sampling rate of the sensor with Incoming activity. Notice that after the Response time, there is no change in the Sampling rate of the sensor (of Incoming activity) and its constant but the Sampling rate of sensor in the top curve is continuously decreasing till it gains steady state. To see the numerical value of Stabilization time, look at Figure 43.

Similarly, Figure 41 depicts a case when the Incoming activity has a value close to the activity value of the background sensors (value 100). As the Incoming value is of competing nature, the response time is more as compared to when the value is much higher. The reasoning has been done in Section 7.1.1 and this graph shows the same behavior for TA Rule. Notice that the Stabilization time is less in this case as the system

doesn't have to shift to a completely new level (like in left graph with Incoming value of 200).

Figure 42 depicts the Response (Appearance) time of Incoming Activity at different Alfa values.

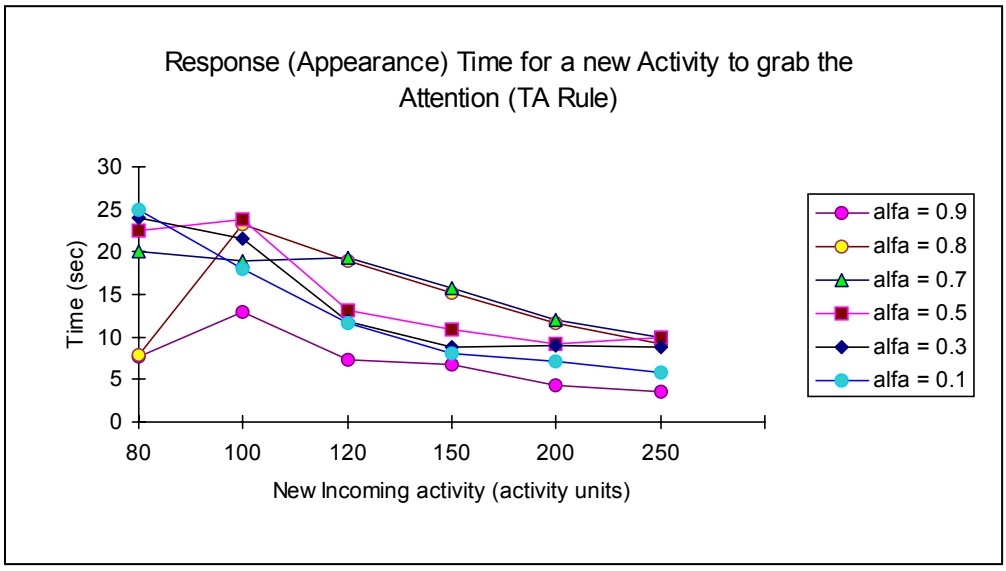


Figure 42: Response Time for Incoming Activity with respect to different Alfa values

Comments and Inferences:

a) When the Incoming activity value is same as that of the background sensor Activity (value 100), the Response time takes maximum value (when compared to Response time values for activity higher than background activity), irrespective of alfa value.

b) Alfa = 0.9 takes the least Response time and there is a marked difference in Response time values for alfa >0.5.

c) Response time increases for 0.1≤ alfa ≤ 0.7 and for alfa > 0.7 Response time starts decreasing.

Figure 43 plots the Stabilization time after the new Incoming activity has grabbed attention.

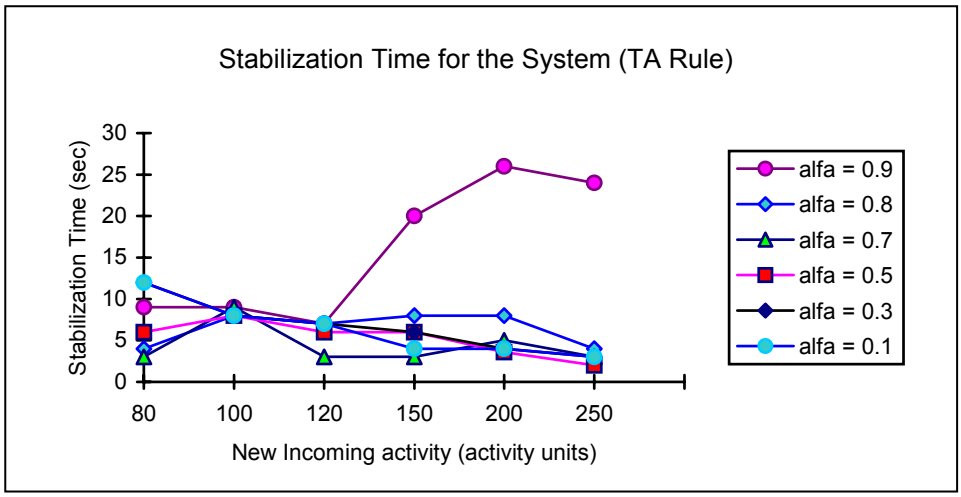


Figure 43: Stabilization Time for Incoming Activity with respect to different Alfa values

Comments and Inferences:

a) Stabilization Time again takes a steep curve at value equal to 100 indicating the competing nature of new Incoming Activity and is same for every value of alfa when the Incoming activity is same as that of background activity

b) Stabilization time for alfa = 0.9 is unusually high

c) Stabilization time decreases when alfa increases from 0.1 to 0.7 (0.1≤ alfa ≤ 0.7) and starts increasing for alfa greater than 0.7. Alfa = 0.7 has the lowest Stabilization time.

d) The value of Stabilization time for alfa = 0.8 is in the same range as the Stabilization time for 0.1≤ alfa ≤ 0.7

The above two plots bring about two surprising results:

a) the Response time is lowest when alfa is at maximum, which is contrary to the TA Rule that says the Higher the alfa, more sluggish the system becomes.

b) The system is not at its maximum sensitivity when alfa is lowest i.e. 0.1 (Beta is highest simultaneously as $\alpha + \beta = 1$)

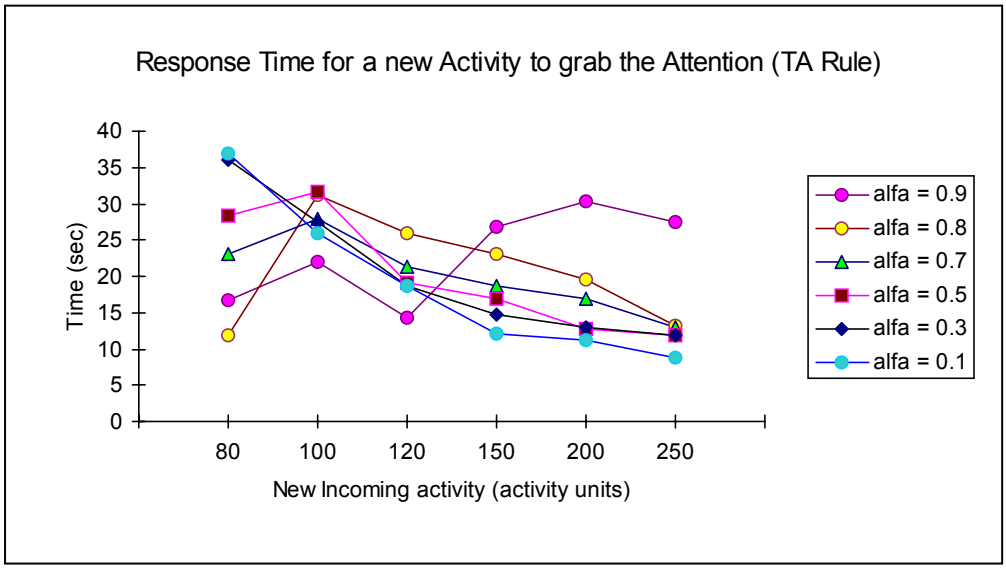


Figure 44: Sum of Appearance time and Stabilization Time v/s Incoming activity

Both the above anomalies are completely in accordance with the definition of TA Rule. If we consider the sum of Response Time and the Stabilization Time, higher alfa do end up making the system sluggish and lowest alfa (highest β) do makes the system more responsive and bringing it quickly to steady state. In order to tune the values of alfa and

beta such that we can have minimum Response time as well as optimum Stabilization time, we have to exclude all alfa > 0.8. The simulation results in Figure 44 indicate that the range of alfa must confirm to the condition 0.3≤ alfa ≤ 0.8 for better Response time and Stabilization time behavior in conjunction. The Response time in this range of values displays the characteristic behavior that has been obtained previously with NS rule and NM rule i.e. as the incoming activity is of competing nature, there is increase in the Response time otherwise Response time decreases as the value of incoming activity increases. For alfa ≤ 0.3, the Response time increases much sharply as the incoming activity equals that of background activity and it continues to rise when the incoming value is less than the background activity. This maps the behavior of NS rule (Figure 30). For alfa > 0.8, the graph shows anomalous results and hence is discarded.

8.3 Network Utilization

This section deals with the experiment done to verify the result of Section 6.1.2 which says that the Network Queue is loaded by T (*threshold*) times the number of messages when the Rate-estimator is not operational in conjunction with the sensors. The background threshold T is 3500 and the background activity is 100 units. The figure on the top shows the case when Rate-estimator is not used in the system and the right side plot the same Queue activity when estimator is operational. Notice the difference in queue usage. As the data messages keep on coming directly to the Sampling Manager without any estimator as an intermediate component, we see a continuous activity of the

Network Queue as the Job rate R Job/sec. When the estimator is operational, all the data messages are filtered through it and as a result the number of data messages flowing through the network are greatly reduced.

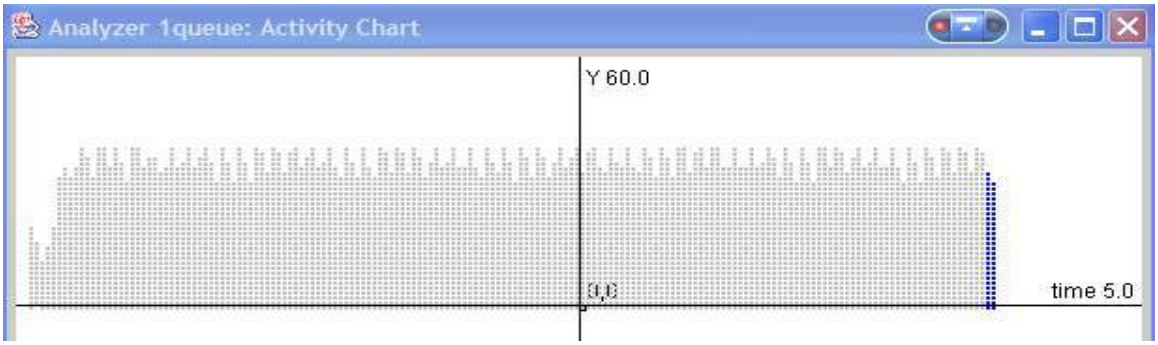


Figure 45: Queue Activity when Rate-estimator is not operational (20 sensors in system)

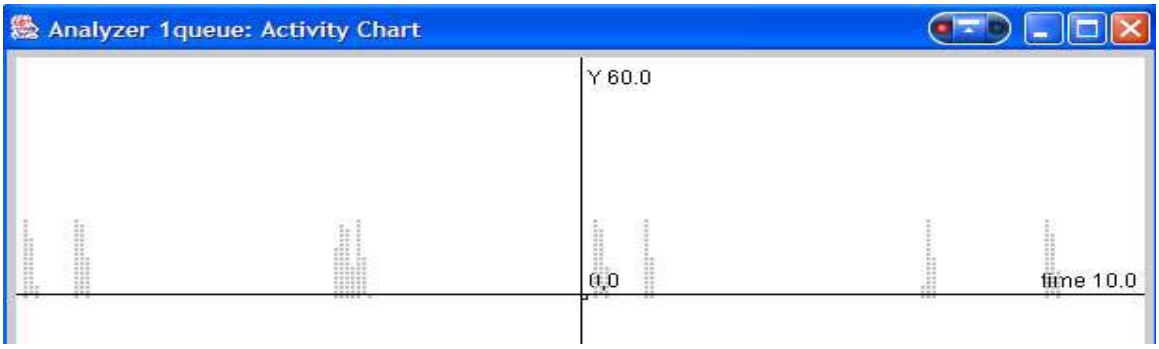


Figure 46: Queue activity when Rate-estimator is used (20 sensors in system)

8.4 Synopsis

This chapter has implemented AdSNet System over a Network Model implementing single network queue. The system behaved according to the behavior described in Section 4 when the communication between components happen through a single network channel and each packet undergoing network-delay depending upon the state of the network (determined at the run-time). The implementation of Network model enabled to monitor the bandwidth usage and the simulation results concluded that even under constant loading of the system, the high Incoming Activity is able to break-into the system and grab attention as well as resources.

9. Application Areas

This architecture is a prototype of an intelligent system capable of focusing attention and directing resources to the focused activity. Any natural or artificial system that has finite amount of resources can be mapped onto this architecture. Any artificial system is composed of sensors and actuators. A sensor is a component that detects the activity and an actuator is a component that reacts to that activity. What happens in between and how quickly it happens, is what this system attempts to accomplish. The capability to acknowledge the detected activity, register it and then releasing commands to the appropriate actuators is what this system is about. Following are some of the application where this system could be put to test:

a) Human decision-making System

b) Autonomous Systems & Robotic Systems

c) Bandwidth Provisioning & QoS Management Systems

d) Applications involving Hierarchically distributed Genetic Algorithms

Since, its inspiration is mostly taken from the Cognitive Psychology domain, the first application that comes to mind is a system that can map **Human decision-making system**. Humans have quite evolved sensory apparatus that enables humans to see, hear, taste, smell and touch. Corresponding to this there are cortical areas in brain dedicated to processing information coming in from these highly complex sensory systems. The most important feature of the brain, that integrates this multifarious information, is its capacity

to handle these parallel inputs and bring forward a complete picture rich in visual and audio components along with their subtle and gross relationships. The sensory systems can't provide an integrated picture and they just communicate to the brain, what they detect. The brain with its integration mechanism and with the help of memory systems creates a unified picture. The ability to take a specific decision and giving us the power of choice is provided by Thalamus that has the capability to analyze and evaluate the situations by using information stored in various parts of the brain [17] and recalling them on-demand. It helps us perceive the environment based on the information in our memory. Once the decision has been made volitionally, it directs the cortical motor neurons to engage the actuators to react to the environment and complete the cycle of sense, perceive, decide, act. Any system that is provided with the components that validates sensory inputs and is able to draw conclusions based on some formal symbolic basis, like memory, threshold mechanism etc, along with a component that directs resource energy to the appropriate apparatus, is an intelligent system. At the lowest level, the *AdsNet* architecture has Estimator attached to every sensor that has the capability to validate the sensory inputs based on threshold-mechanism. At the higher levels and at every level, the AdSNet System contains a data-driven Decision Maker (not in scope of this thesis) that helps the system to proceed and interact on achieving some specific goals both local and global. It also has a component called Attention Manager that directs the resources to the appropriate place. In a sense, the system has the basic functionality to display intelligent behavior.

The other league of systems that this architecture can be used to is the **Autonomous Systems and the Robotic Systems**. These artificial systems that respond to the situations presented to them, but there scope is limited and much of their response is hard-wired and generally predicted. The sense-respond pair is explicitly defined and the systems face limitations due to the scalability and evolvability issues. The *AdsNet* system with is variable-structure definition has capability to evolve both in terms of system structure and component metamorphosis. The variable structure implementation help us analyze the behavior of the network-system when its components count is increased or decreased in run-time, for example, in a robot application consisting of swarm of robots focused over a particular target. The target may be considered as an Activity area and the robots may be considered sensors with different sensitivities and different processing powers. A good amount of work has been done in this area that uses the modeling approach to realize physical robotic systems [39].

The other major application of *AdsNet* architecture is in the area of **Network Management and QoS provisioning**. With the tremendous growth of Internet, Network Engineering & Management have developed in whole areas in itself and require sufficient monetary and human effort to maintain network systems. Providing QoS is still a challenge and appropriate measures and means are under research to do it efficiently. The problem occurs with incomplete understanding of the traffic behavior and its categorization with respect to network utilization, application specific requirements and business application priorities. The basic task is to manage the link bandwidth capacity to

be allocated according to the priorities and economics. In a recent white-paper by Cisco [25], a five step approach have been suggested for QoS provisioning and Collecting & Reporting Capacity Information:

1. Determine your needs
2. Define a process
3. Define Capacity Areas
4. Define Capacity Variables
5. Interpret the data

This 5-step approach has been advised to System Administrators, Network Mangers, as there's a marked difference between what the software tools can do and the way they are implemented by people. In defining the capacity areas in Step 3, they suggest that:

a) Different areas may have different thresholds (for example, LAN bandwidth is much cheaper than WAN bandwidth so utilization thresholds should be lower)

b) Different areas may require monitoring different MIB variables (for ex, FECN and BECN counters in frame relay are critical in understanding frame-relay capacity problems)

c) It may be more difficult or time consuming to upgrade areas of network (for ex, international circuits can have much longer lead times and need corresponding higher level of planning)

The *AdsNet* system has the basic functionality to define and work on these capacity areas (traffic rates) and act and assign bandwidth capacity to the area with high priority or importance. The quantitative results show experiments done with threshold variation,

sensor sensitivities, and fixed capacity networks that map with the requirements presented above as:

QoS Provisioning requirement	AdSNet feature
Different Areas may have different thresholds	The sensors lie in a specific area and the attached Rate-estimator works on the Threshold mechanism. The "Surveillance Area" in this case is the Network Utilization or Bandwidth usage
Different Areas may require monitoring of different variables	In an area we have numerous sensors. Each sensor has its sensitivity index which allows it to detect any activity higher than this index value. Different 'variables' in the left columns can be quantified to different sensitivity indexes.
Difficult and time-consuming to upgrade some areas of network (generally international circuits)	The expanding-contracting nature of the AdSNet System structure is limited to County levels and the levels underneath it as its impractical to add/remove a whole County from a Network-system operation.

The last application of AdSNet System that deals with resource allocation, that comes to mind are the **Design Applications involving Genetic Algorithms**. On such work has already been done in the area of Intelligent Control towards the design of a Fuzzy Logic Controller [28]. It has been built on an architecture called Intelligent Machine Architecture (IMA), which integrates non-deterministic symbol processing and computationally intensive numerical processing. Hierarchical Distributed Genetic Algorithm (HDGA) was developed within the IMA Framework. Genetic Algorithms (GAs), which uses principal of biological evolution, simultaneously evaluate many points in parameter space and converge towards a global solution. For optimal results, the

algorithm tunes each parameter to great precision. This issue has been addressed by using multiresolutional search paradigm and by employing a variable structure (changes in internal structure to achieve the goal).

The HDGA architecture consists of multi-level clusters that are created on the fly depending upon the resolution demanded by the problem search space. Each cluster consists of a Controller (a kind of expert system) and the agents (that solve an abstracted version of the given problem). These clusters can collect information, process information and pursue a goal. It also creates parent-child relations with other clusters. HDGA uses a multiresolutional search strategy in which a higher-level cluster investigates a search space at low resolution and creates child clusters with high resolutions. This results in the creation of clusters in a hierarchical fashion, with each level dealing with a certain problem space at a particular resolution (lower level in the hierarchy with highest resolution). The Controller is a knowledge-based expert system that is an autonomous entity in the cluster and one of its major task is to coordinate the execution of agent GAs to solve a given problem. The job of the Controller is to execute and coordinate GAs in a goal-oriented fashion and displaying a top-down control. An agent GA is linked to a simulator and performs computationally intensive simulation required to evaluate individual parameter sets in real-world problems. They report their results to the Controller. GAs don't need large population or chromosome sizes which helps the HDGA architecture an efficient way of managing computing resources during a search, thus allocating more resources to promising search regions.

The following figure shows the mapping of the HDGA architecture with that of the AdSNet architecture. This mapping enables the HDGA system to manage the resources also. Notice the presence of RAM in AdSNet system and the analogous behavior of Controller in HDGA system mapped to the combination of GAFS and DDM. The function of Rate-estimator provides a mechanism to allocate resources to the particular sensor in the AdSNet System. When mapped with the HDGA System this Estimator may contain the logic for threshold-mechanisms with respect to small population size or chromosome size to the attached sensor (GA).

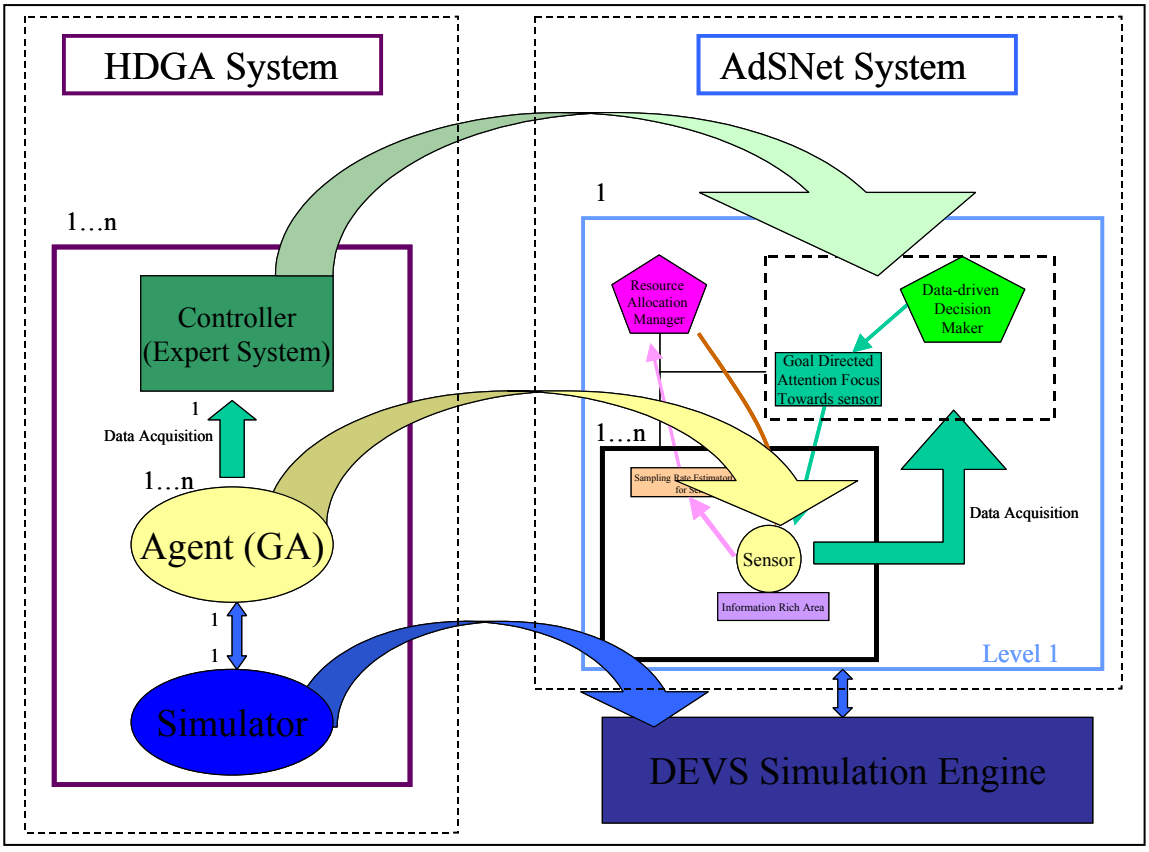


Figure 47: HDGA system and AdSNet System together

Both the architectures displayed above support variable structure and can expand & contract at the run-time or on a need-to-basis. The AdSNet architecture takes the leverage from DEVS Simulation Engine. The generic architecture of AdSNet sensor system facilitates the mapping of its sensor apparatus in various settings, a Genetic Algorithm in this case.

10. Conclusions and Future Work

This research has successfully implemented a scalable framework *AdSNet* capable of focusing attention to components displaying high activity and directing resources towards them so that they can accomplish their task efficiently. The framework has been built over DEVS formalism and exploits the advantages of variable-structure DEVS, which was designed during this research. This provides the components to exhibit adaptive behavior as their environment changes. The architecture is multi-level hierarchically organized modular system and can be mapped to any real life system that is hierarchically organized working along the rule of "chains of command", which implies that the information is filtered and condensed as it travels up in the hierarchy. Here in this research it has been mapped to geographical area distribution under the control of managers that allocates resources (like bandwidth and channel capacity) to areas under their control, which in turn distribute the resources to the end-user. The distribution is done intelligently with the most active component receiving the maximum number of resources. As the sensors display high or low activity (Figure 40 and 49) so does the assignment of resources allocated to them, resulting in increased or decreased sampling rate, respectively. The simulation results have confirmed that the system is capable of directing and switching its focus to components that become active during simulation and also can withdraw attention from components who are not displaying any activity. The architecture has also displayed the capability to evolve, by bringing new components in the system during the simulation run.

This work combines the WTA (Winner Take All algorithm) proposed by Tsotsos and the Berger model to develop a system where the attention is focused on an activity of highest importance. The criteria for deciding an activity important is based on the sensitivity of the sensor and the threshold of the "Attention Control Center" associated with every sensor, which in this case is called Rate-Estimator that validates the importance of any activity sensed by the sensor. The WTA model enables the system to continually shift focus and direct its attention to the most active component. This system has been extended towards a Resource Allocation system, where the resources are directed towards a focused activity, which makes the system a generalized architecture capable of focusing attention and concentrating on the job at hand by providing more resource to it by redistribution and reallocation. It calls for two entities in the system:

a) Entity (Rate Estimator) that is capable of drawing conclusions and analyzing situations based on threshold mechanism
b) Entity (Sampling Manager) capable of focusing attention to the most needy one

The system has a n-tier hierarchical organization where there exists a Manager at every level to direct focus and attention and an Estimator coupled to every sensory element. The Estimator may or may not be present at the intermediate levels in the hierarchy but it must be at the coarsest level, to deduce and validate what the sensors are witnessing. The system also allows resources and peripheral attention to the ongoing working sensors and doesn't inhibit or stalls their operation in the pursuit of focusing attention to the important

one. For different WTA mechanisms, the sensor population is met accordingly and in no case, the resources are completely withdrawn from the running sensors as its not predictable which sensor might produce important information the next instant. The system let the other sensors keep working at their default settings and provide resources for their operation and when an activity of high importance is encountered as advertised by any sensor, it provides resources required by that sensor.

Future work

As mentioned before, the present AdSNet system only studies the bottom-up flow of information, the next step of the research is to cater the top-down communication aspects of the System that drive the system towards a central goal. Once this part of the research is done in the near future, the system will be ready to develop an autonomous system capable of "perception" and responding towards the perceived situation, based on the reasoning developed due to stored information and local memory. The other area where this system can be extended is in the area of On-demand systems. A plausible scenario is the situation when a Surveillance area displays a higher level of activity that its neighboring areas. A little description of this kind of scenario has been presented in Section 5.2. The further work will run rigorous experiments to test this scenario and the Attention-Focus mechanism in this dynamic network.

The distributed operation of the system and its architecture being component based facilitates its deployment in the real world in terms of federates. The system can be made

predictive and robust with more detailed modeling of the Estimator and WTA mechanisms implemented in the Attention Manager, supported with efficient synchronization strategies.

REFERENCES

[1]. Zeigler, B.P, "Scalability Considerations in Measuring Intelligence: Insights from Modeling and Simulation", ACIMS, University of Arizona, 2002

[2]. Zeigler, B.P., Hariri, S. , "Discrete Event Architecture for Time-critical Decision Making", ECE Dept, University of Arizona, 2002

[3]. Zeigler, B.P., "Design of Attention Management for Homeland Defense Intelligence Surveillance and Reconnaissance", Supplement to NSF Grant with Steve Hall of Lockheed Martin, Advanced Simulation Center, Sunnywale, CA, 2002

[4] Tsotsos, J.K. , "Computational Resources do constrain behavior", Behavioral Brain Sc., pg 506-507, 1991

[5] Tsotsos, J.K. , Modeling Visual Attention via Selective Tuning, Artificial Intelligence 78, 507-545, 1995

[6] Tsotsos, J.K. , An Inhibitory beam for attentional selection, In: Harris, L., Jenkin, M. (Eds), Spatial Vision in Humans and robots, Cambridge University Press, Cambridge, pg 313-332. 1993

[7] Tsotsos, J.K., Triangles, Pyramids, Connections and Attentive Inhibition, *PSYCHE: An Interdisciplinary Journal of Research on Consciousness* (http://psyche.cs.monash.edu.au/v5/psyche-5-20-tsotsos.html), July 1999.

[8] Tsotsos, J.K., On Behaviorist Intelligence and the Scaling Problem, *Artificial Intelligence 75*, p 135 - 160, 1995

[9] Tsotsos, J.K., Limited Capacity is a Sufficent Reason for Attentive Behavior, *Cognition and Consciousness 6*, (invited Commentary on A.H.C. van der Heijden and S. Bem) , p.429 - 436, 1997

[10] Tsotsos, J.K., Milios, E., Selective Attention within a Visual Processing Pyramid, IEEE Workshop on Nonlinear Signal and Image Processing, Neos Marmaras-Halkidiki, Greece, June 20-22, 1995.

[11] Pinker, S, How the Mind Works, W.W. Norton, New York, NY

[12] Zeigler B. P., Discrete event Abstraction: an emerging paradigm for modeling complex adaptive systems, Festschrift in honor of John H. Holland, L Brooker

[13] Zeigler, B. P. , The brain-machine disanalogy revisited, BioSystems 64 (2002), 127-140

[14] Culhane, S. M., Tsotsos, J. K., A prototype for Data-driven Visual attention, Prodeedings of the 11th IAPR International Conference on Pattern Recognition, the Hague, The Netherlands, September 1992, IEEE Computer Society Press, Los Alamitos, CA pp 36-40

[15] Tsotsos, J.K., Computation, PET Images and Attention, *Behavioral and Brain Sciences 18:2*, (invited Commentary on Images of Mind, Posner and Raichle) p372 , 1995

[16] LaBerge, D. , Attention, Awareness and the Triangular Circuit, Cognition and Consciousness, 6, 149-181, 1997

[17] Nader, T., Human Physiology-Expression of Veda and Vedic Literature, Maharishi Management University Press, 1995

[18] Zeigler, B.P., Kim, T.G., et al., 2000. Theory of Modeling and Simulation, Academic Press, New York, NY

[19] Cellier, F.E., Continuous System Modeling, Springer-Verlag, NY, 1991

[20] Hu, X., Zeigler, B.P, Mittal, S. , Dynamic Configuration in DEVS Component-based modeling and Simulation, to appear in Transactions: Society of Modeling and Simulation International, 2003

[21] Zeigler, B.P., Mittal, S., Modeling & Simulation Architectures for Autonomous Computing, Autonomic Computing Workshop: The Next Era of Computing, January 2003

[22] Zeigler, B.P., Mittal, S., Modeling and Simulation of Ultra-large Networks: A Framework for New Research Directions, supported by NSF Grant ANI-0135530, ULN Workshop, July 2002 (addendum to the ULN Workshop 2001)
http://www.acims.arizona.edu/EVENTS/ULN/ULN_doc2.pdf

[23] Mittal, S., Hierarchical scalable DEVS architecture for a Service discovery Network, ECE575 Project, 2001

[25] Cisco-Capacity and Performance Management: Best Practices, White Paper, 2003

[26] DeLong, D.F., Code Red Virus: "Most expensive in the history of Internet", 2001

[27] http://www.caida.org/analysis/security/code-red/newframes-small-log.mov

[28] Kim, J. and Zeigler, B. P., Hierarchical Distributed Genetic Algorithms: A Fuzzy Logic Controller Design Application, IEEE Expert, 1996

[29] Hannon, C. and D. J. Cook., " A Parallel approach to Unified cognitive Modeling of Language Processing within a Visual context." In Hamilton, H. (Ed.) *Advances in Artificial Intelligence*, 1822. Springer Verlag, 151 –163, 2000

[30] Hannon, C., "Biologically Inspired Mechanisms for Processing Sensor Rich Environments". In *Proceedings of FLAIRS 02*, AAAI Press, 3-7, 2002

[31] Aylett, R and C. Delgado. "Emotion and Agent Interaction". In *AAAI Fall Symposium,* 2001

[32] Reilly, S. W. and J. Bates. "Emotion as part of a Broad Agent Architecture", http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/wsr/Web/reserch/waume93.html, 2003

[33] Velasquez, J. D. "From Affect Programs to Higher Cognitive Emotions: An Emotion-Based Control Approach", In Proceedings of EBA 99.

[34] Hannon, C., "Emotion-based Control Mechanisms for Agent Systems", *International Conference on Information Systems and Engineering*, 2003

[35] Hwang, J and Revuru, R., "Inter-domain Diffserv Dynamic Provisioning and Ineterconnection Peering Study using Bandwidth Management Point – A Simulation Evaluation. ", *International Conference on Information Systems and Engineering*, 2003

[36] Scheutz, M., "Agents with or without Emotions?", *In Proceedings of FLAIRS 02*, AAAI Press, 89-94, 2002

[37] Qbone Signaling Design Team "Simple Inter-domain Bandwidth Broker Signaling (SIBBS)", http://qbone.internet2.edu/bb work in progress

[38] Durham, D., Boyle, J., Cohem, R., Heroz, S., Rajan, R., Sastry, A., "Common Open Policy protocol (COPS),", *IETF RFC 2478*, 2000

[39] Jamshidi, M., El-Osery, A., Fathi, M., et al., V-Lab :A Distributed Intelligent Discrete-Event Environment For Autonomous Agents Simulation, Intelligent Automation and Soft Computing, Vol.9, No.3, pp. 181-214, 2003

[40] Zeigler, B. P., Object-Oriented Simulation with Hierarchical, Modular Models, 1990, San Diego: Academic Press

[41] http://www.technologyreview.com/articles/downloads/huang0703.html

[42] http://www.dei.isep.ipp.pt/docs/arpa.html, History of ARPANET

[43] Network Simulator-2, http://www.isi.edu/nsnam/ns/

[44] Opnet Modeler, http://www.opnet.com/products/modeler/home.html

[45] Xiang Zeng, Rajive Bagrodia, Mario Gerla; "GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks", *Proceedings of the 12th Workshop on Parallel and Distributed Simulations -- PADS '98*, May 26-29, 1998 in Banff, Alberta, Canada

[46] Glomosim: http://pcl.cs.ucla.edu/projects/glomosim/

[47] SSFNet, http://www.ssfnet.org/

[48] Parallel/Distribute NS, http://www.cc.gatech.edu/computing/compass/pdns/

[49] DEVS-DOC http://acims.eas.asu.edu/PUBLICATIONS/PDF/hild_phd.pdf

[50] Hild, D.R., H.S. Sarjoughian, B.P. Zeigler, "DEVS-DOC: A Co-Design Modeling and Simulation Environment." IEEE SMC-Part A, Vol. 32, No. 1, pp. 78-92.

[51] Unified Modeling Language, www.omg.org/uml/