AGENT-BASED SIMULATION OF BEHAVIORAL ANTICIPATION IN

COMPUTER NETWORKS: A COMPARATIVE STUDY OF

ANTICIPATORY FAULT MANAGEMENT

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

_____

Avdhoot Kishore Saple

Certificate of Approval:

_____	_____
Drew Hamilton Associate Professor Computer Science and Software Engineering	Levent Yilmaz, Chair Assistant Professor Computer Science and Software Engineering
_____	_____
Gerry Dozier Associate Professor Computer Science and Software Engineering	Stephen L. McFarland Acting Dean Graduate School

AGENT-BASED SIMULATION OF BEHAVIORAL ANTICIPATION IN COMPUTER NETWORKS: A COMPARATIVE STUDY OF ANTICIPATORY FAULT MANAGEMENT

Avdhoot Kishore Saple

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
May 11, 2006

AGENT-BASED SIMULATION OF BEHAVIORAL ANTICIPATION IN

COMPUTER NETWORKS: A COMPARATIVE STUDY OF

ANTICIPATORY FAULT MANAGEMENT

Avdhoot K. Saple

_____

Signature of Author

_____

Date of Graduation

THESIS ABSTRACT

AGENT-BASED SIMULATION OF BEHAVIORAL ANTICIPATION IN COMPUTER NETWORKS: A COMPARATIVE STUDY OF ANTICIPATORY FAULT MANAGEMENT

Avdhoot K. Saple

Master of Science, May 11, 2006
(B.E., Mumbai University, 2004)

98 Typed Pages

Directed by Dr. Levent Yilmaz

Network fault management is concerned with the detection, isolation and correction of anomalous conditions that occur in a computer network. Present state of art in fault management classifies existing methodologies into two main categories: reactive rule based approaches and intelligent monitoring systems. We explore the concept of anticipatory behavior to develop an intelligent agent-based network management model, which uses an anticipatory agent to proactively detect occurrence of faults using a predictive Bayesian model pertaining to network performance. To analyze the effectiveness of the anticipatory technique, we compare it with alarm correlation and rule-based reactive fault management strategies. Results of the comparative analysis are

presented to demonstrate the potential of the anticipatory technique in detecting network anomalies. Our findings indicate that the anticipatory technique improves network performance significantly better than the reactive techniques. We furthermore describe a methodology for adaptive restructuring of the network based on the simulated annealing process. We observe that adaptive restructuring gives significantly better performance under the reactive rule-based fault-management technique as compared to the anticipatory strategy.

ACKNOWLEDGEMENTS

I wish to thank my advisor, Dr. Levent Yilmaz for his support and guidance along the way, above all for being so patient and understanding with me. Thanks to my committee members for reviewing my thesis. I would also like to thank my colleagues and friends for the wonderful times together. Above all, thanks to my parents for being always there to listen. They have been my constant source of inspiration.

Style manual or journal used APA Style

Computer software used Microsoft Word. Images drawn using Microsoft PowerPoint, UML Diagrams drawn using ArgoUML.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Network fault management (Oates 1995) entails the detection, isolation and correction of anomalous conditions that occur in a network. It can be decomposed into three subtasks: fault identification (Schwarz and Katzela 1995), fault diagnosis (Agre 1986), and fault remediation. Fault identification involves detecting deviation from normal behavior followed by identification of its nature, whereas fault diagnosis involves determining the root cause of the identified problem. Fault remediation is the formulation of a course of action that addresses the problem. All three stages of fault management involve reasoning and decision making based on information about current and past states of the network.

1.1 The Need for Fault Management in Computer Networks

As business and individuals have become increasingly reliant on computer networks, the complexity of those networks has grown along a number of dimensions. The phenomenal growth of the Internet in recent years provides a clear example of the extent to which the use of computer networks is becoming ubiquitous (Oates 1995). As computer networks increase in size, heterogeneity, complexity and pervasiveness, effective management of such networks simultaneously becomes more important and more difficult.

The use of computer networks for business is expanding enormously. The average number of electronic point-of-sale transactions in the United States went from 38 per day in 1985 to 1.2 million per day in 1993. An average $800 billion is transferred among partners in international currency markets every day; about $1 trillion is transferred daily among US banks; and an average $2 trillion worth of securities are traded daily in New York markets. Nearly all of the financial transactions pass over information networks. Consequently, the losses incurred due to faults in these networks are enormously high. Related dollar losses are estimated to be between $100,000 and $10 million (Herdman 1994). Hence it is important to have an effective and efficient network fault management technique to restore the proper functioning of computer and information networks.

1.2 Anticipatory Fault Management

Traditionally, network management activities, such as fault management, have been performed with direct human involvement. However, these activities are becoming more demanding and data intensive, due to the heterogeneous nature and increasing size of networks today. For these reasons, it is becoming necessary to automate network management activities. Artificial intelligence technologies can play an important role in the problem solving and reasoning techniques that are employed in fault management.

Anticipatory fault management involves a novel approach of designing autonomous agents that is based on the idea of anticipatory systems (Ekdhal et al. 1994). An anticipatory system has a model of itself and of the relevant part of its environment

and will use the model to predict the future. The predictions are then utilized to determine the agent's behavior. An anticipatory system is thus a system which uses the knowledge of future states to decide what action has to be taken in the present. Anticipatory fault management can be carried out by having intelligent processing anticipatory agent reside on the network under observation. It makes use of adaptive learning methods (Butz et al. 2003) to detect abnormal behavior before a fault actually occurs. The agent first acquires a picture of the network's health by means of observation processing to process performance variables and obtain probability of each measured variable at a given time. It then combines all the information to build up a predictive model, which provides a method for estimating probabilities and allows the agent to combine observed information with prior knowledge. (Hood and Ji 1998) The agent therefore gets a complete picture of the network's health to carry out adaptive behavior for fault identification and diagnosis.

1.3 Research Objective

Given the importance of network fault management, the goal of this research comprises of coming up with an anticipatory technique for network fault management, followed by comparison of the same with two widely popular techniques: reactive rule based strategy and alarm correlation approach. We compare the three techniques based on the following network performance metrics: throughput, turnaround time, and the drop rate of packets.

To facilitate the experiment, a simulation model of a computer network is developed using the DEVS (Zeigler and Sarjaughian 2003) modeling and simulation framework. Reactive and anticipatory agents are embedded into the network for network

fault management. The reactive agent operates on a simple rule based engine that detects faults based on predefined fuzzy rule-base. We use a Naïve Bayesian classifier (Luger and Stubblefield 1989) as part of the anticipatory agent. The Bayesian classifier acts as a predictive model for the anticipatory agent to facilitate prediction of faults based on past data. Our findings indicate that anticipatory fault management performs significantly better than the reactive and alarm correlation techniques under the experimental conditions the model is tested. Furthermore, we make a study of which technique performs better with network reconfiguration. Network reconfiguration comprises of restructuring the network model as a simulated annealing process (Carley and Svoboda 1996). The restructuring is based on varying network operational parameters such as the operation of switch, the routing strategy of the router, and the link delay of nodes. The results of network adaptation shows that simulated annealing can be applied to the reactive technique since it gives a better performance as compared to the alarm correlation and the anticipatory technique.

The thesis is organized as follows. Chapter 2 reviews the present state of art in fault management of computer networks. Chapter 3 comprises of the design of anticipatory systems. In chapter 4, we discuss the agent based modeling of reactive and anticipatory control in computer networks. Chapter 5 discusses the detailed design of the DEVS model. Chapter 6 comprises of experimentation design, simulation, and results. Finally, in chapter 7 we conclude by discussing the open issues as well as planned future work.

Chapter 2

Overview of Fault Management in Computer Networks

Researches have approached the problem of fault management using various techniques such as artificial intelligence (Corn et al. 1988; Joseph et al. 1989; Wright et al. 1988; Yamahira et al. 1989), machine learning (Langley and Simon 1995) and state space modeling (Rouvellou and Hart 1995). A fault is simply a malfunction in some component of the network, either hardware of software. At an abstract level, fault identification can be thought of as a function, *I*, with inputs and outputs. The input to the function is a description of network state, *S*, and the output is a set of hypothesis, *H*, concerning the existence of *n* different faults. Each hypothesis may specify the indications in S of the corresponding fault and may contain some amount of diagnosis information. That is identification and diagnoses are rarely totally decoupled.

Fault identification, therefore is a process of function that maps from network states to fault hypothesis:

$$I: S \rightarrow H$$

Different approaches to fault identification define S in a distinct manner (Tim Oates 1995).

2.1 Rule Based Approaches

Early work in the area of fault or anomaly detection was based on expert systems. In expert systems, an exhaustive database containing the rules of behavior of the faulty system is used to determine if a fault occurred by matching of predefined rules of network anomalies (Lewis 1993). Rule based systems are too slow for real time applications and are dependent on prior knowledge about the fault conditions of the network. The identification of faults in this approach depends on symptoms that are specific to a particular manifestation of a fault. Examples of these symptoms are excessive utilization of bandwidth, number of open TCP connections, total throughput exceeded etc. (Thottan and Ji 2003). An expert system model using fuzzy cognitive maps (FCMs) (Ndouusse and Okuda 1996) can be used to obtain an intelligent modeling of the propagation and interaction of network faults. Fuzzy expert systems (Kandel 1991) are especially attractive in a dynamic environment because they favor silicon implementation, learning and they avoid the lengthy symbolic graph search in favor of computational inference. Traditional expert systems with symbolic knowledge representation implemented with “IF/ THEN” conditional statements require complicated and lengthy matching schemes, to slow for real-time systems such as networks. Furthermore, traditional expert systems lack support for on-line mathematical analysis, an essential feature common in engineering systems. Fuzzy Expert Systems (FES) provide an alternative to symbolic intelligence. In FES, vague causal reasoning is represented numerically, and hence is amenable to computational processing. In particular, FES which uses a graph-based knowledge representation can easily be converted into causal matrices, thus offering an appealing computational feedback memory recall capability.

2.1.1 Cased-Based Reasoning

Case based reasoning is an extension of rule-based systems (Lewis 1993). It differs from FCM in that, in addition to just rules, a picture of previous of fault scenarios is used to make the decisions. It differs from FCM in that, in addition to just rules, a picture of previous fault scenarios is used to make decisions. A picture here refers to the circumstances or events that led to the fault. In order to adapt the case-based reasoning scheme to the changing network environment, adaptive techniques are used to obtain the functional dependence of relevant criteria such as network load, collision rate, etc., to previous trouble tickets (Lewis and Dreo 1993). The trouble ticketing system is used to perform two functions: Prepare for problem diagnostics through filtering, and infer the root cause of the problem. Using case-based reasoning for describing fault scenarios also suffers from heavy dependence on past information. Furthermore, the identification of relevant criteria for the different faults will, in turn, require a set of rules to be developed. In addition, using any function approximation, such as back propagation, causes an increase in computation time and complexity. The number of functions to be learned also increases with the number of faults studied.

2.2 Alarm Correlation

A fault is a disorder occurring in the managed network. Faults happen within the managed networks while alarms are external manifestations of faults (Rouvellou and Hart 1995). Alarms are defined by vendors and generated by network equipment are observable by network operators. Similar alarm messages with different time stamps are interpreted as separate alarms. Modern telecommunication networks may produce thousands of alarms per day, making the task of real-time network surveillance and fault

management difficult. Due to the large volumes of alarms, network operators frequently overlook or misinterpret them. To reduce the number of alarms displayed on operators' terminals, current network management systems apply alarm filtering procedures or, in the case of bursts of alarms, send them directly to a printer or database (Jakobson and Weissman 1993). Furthermore, a single fault in a large communication network may result in a large number of fault alarms making the isolation of the primary source of failure a difficult task (Katzela 1995).

2.2.1 Alarm Correlation Using Finite State Machines

External observations of alarms may instill an impression that one alarm causes another. However the causality is not between alarms, but rather between faults. Finite state machines model alarm sequences that occur during and prior to fault events. For instance, a probabilistic finite state machine model is built for a known network fault using historical data (Lazar et al. 1992). State machines are designed with the intention of not just detecting an anomaly but also possibly identifying and diagnosing the problem. The sequences of alarms obtained from the different points in the network are modeled as states of a finite state machine. The alarms are assumed to contain information such as the device name as well as the symptom and the time of occurrence. The transitions between the states are measured using prior events (Katzela and Schwarz 1995; Rouvelle and Hart 1995; Bouloutas et. al. 1990). A given cluster of alarms may have a number of explanations and the objective is to find the best explanation among them. The best explanation is obtained by identifying a near optimal set of nodes with minimum cardinality such that all the entities in the set explain all the alarms and at least one of the nodes in the set is the most likely one to be in fault (Lazer et al. 1992; Jackobson and

Weissman 1993). From an observer's point of view, fault detection and identification requires checking whether a network device behaves as the FSM specified and if not, how it deviates from the expected behavior (Lazer et al. 1992). Alarm correlation may be used for network fault isolation and diagnosis, selective corrective actions, proactive maintenance and trend analysis (Jackobson and Weissman 1993).

2.3 Pattern Matching

This approach describes anomalies as deviations from normal behavior and attempts to deal with the variability in the network environment (Feather and Maxion 1993; Papavassiliou et al. 2000). In this approach online learning is used to build traffic profile for a given network. Traffic profiles are built using symptom specific feature vectors such as link utilization, packet loss and number of collisions. These profiles are then categorized by time of day, day of week and special days, such as weekends and holidays. When newly acquired data fails to fit within some confidence interval of the developed profiles then an anomaly is declared. One method includes capturing of normal behavior of time series as templates and setting of tolerance limits based on different levels of standard deviation. These limits are tested using extensive data analysis (Feather and Maxion 1993). The authors also propose a pattern matching scheme to detect address usage anomalies by tracking each address at 5-min intervals. A template of the mean and standard deviation on the usage of each address is then used to detect anomalous behavior. The anomaly vectors from any new data are checked using template feature vector for a given anomaly and id a match occurs it is declared indicating a fault. For simple, unvaried data, a mechanism called the *Performance and Anomaly Monitoring System,* or PAMS is used (Feather 1992; Maxion 1989; Maxion 1990; Maxion and

Feather 1990). PAMS will highlight anomalous points in time series data by developing a prediction of normal behavior, called a template, and tolerance limits called envelopes, based on a model of data variance. Current data that falls outside of the tolerance envelopes is considered anomalous. The efficiency of the pattern matching approach depends on the accuracy of the traffic profile generated. Given a new network, it may be necessary to spend a considerable amount of time building traffic profiles. In the face of evolving network topologies and traffic conditions, this method may not scale gracefully (Thottan and Ji 1993).

2.4 Statistical Analysis

As the network evolves, each of the methods described above require significant recalibration or retraining. However using statistical approached (Thottan and Ji 2003), it is possible to continuously track the behavior of the network. Statistical analysis has been used to detect both anomalies corresponding to network failures (Thottan 2000) as well as network intrusions (Wang et al. 2002). Interestingly, both of these cases make use of standard sequential change point detection approach. The *Flooding Detection System,* (Wang et al. 2002), uses measured network data that describes TCP operations to detect SYN flooding attacks. SYN flooding attacks capitalize on the limitation that TCP servers maintain all half open connections. Once the queue limit is reached, future TCP connection request are denied. The sequential change point detection employed here makes use of the nonparametric cumulative sum (CUSUM) method. Using this approach on trace-driven simulations, it has been shown that SYN flooding attacks can be detected with high accuracy and reasonably short detection times. When detecting anomalies due to failures, we are confronted with the problem of detecting a host of potential scenarios.

Each of these failure scenarios differ in their manifestations as well as their characteristics. Thus, it is necessary to obtain a rich set of network information that could cover a wide variety of network operations. The primary source for such in depth information is in the SNMP MIB data. Designing a failure detection system using MIB data necessitates the use of a general method since MIB variables exhibit varying statistical characteristics (Thottan 2000).

2.5 Intelligent Probing for Fault Management

Intelligent probing makes use of probing technology (CAIDA 2005) for cost effective fault diagnosis in computer networks. Probes are test transactions that can be actively selected and sent through the network. A distributed system can be represented as a "dependency graph" where nodes can be either hardware elements (e.g., workstations, servers, routers) or software components or services, and links can represent both physical and logical connections between the elements. Probes offer the opportunity to develop an approach to diagnosis that is more active than traditional "passive" event correlation and similar techniques. A probe is a command or transaction sent from a particular machine called a probing station to a server or a network element in order to test a particular service. This work addresses the probing problem using methods from artificial intelligence. We call the resulting approach intelligent probing. The probes are selected by reasoning about the interactions between the probe paths. For diagnosis we use a local inference approximation scheme, for instance a Bayesian network (Huard and Lazar 1996) or other probabilistic dependency models (Katzela and Schwartz 1995) that avoids the intractability of exact inference for large networks (Brodie et al. 2002).

2.6 Proactive Fault Detection using Intelligent Agents

Current fault management implementations generally rely on the expertise of a human network manager, which is translated to a set of rules and then to threshold levels on the measurement variables being collected. As networks become more complex and as changes more frequently, the human network manager will find hard to maintain sufficient level of expertise on a particular network's behavior (Hood and Ji 1998). Fault management research has covered approaches such as expert systems, finite state machines, advanced database techniques, and probabilistic methods (Lazer et al. 1992). The drawback to all these approaches is that they require a specification of the faults to be detected, and it is not feasible to specify all possible faults. Also, changes in network configuration, applications, and traffic can alter the type and nature of possible faults, which makes modeling them impractical in many cases. Intelligent agents that reside at network nodes use adaptive learning methods to detect abnormal behavior before a fault actually occurs (Ekaette and Far 2003). In this approach, the intelligent agent processes information collected by Simple Network Management Protocol agents, and uses it to detect the network anomalies that typically precede a fault (Yemini 1994). The SNMP agents collect information about the network node through their management information base, or MIB, which holds a set of variables pertinent to that particular node. The intelligent agents learn the normal behavior of each measurement variable and combine the information in the probabilistic framework of a Bayesian network (Huard and Lazar 1996; Pearl 1998). This yields a picture of the network health form the perspective of the network node, which can be used to trigger local corrective action or a message to a centralized network manager (Hood and Ji. 1998).

Table1.1 Fault Management Techniques

Proposed System	Methodology	Complexity	Scalable	Detect new fault patterns
Rule based approach	Anomaly detection by conventional rule based systems.	Low	No	No
Alarm Correlation	Incorporation of finite state machines to model alarm sequences that occur during and prior to fault events.	Moderate	No	No
Pattern matching	An anomaly is considered as variability in network environment.	Moderate	No	Yes, but introduces overheads
Statistical analysis	Employment of statistical approaches to continuously track the behavior of the network.	Moderate	Yes	Yes
Intelligent probing	Use of probing technology for fault diagnosis.	High	Yes	Yes
Proactive fault detection using agents	Deployment of software agents that detect, correlate and selectively seek to derive a clear explanation of faults.	High	Yes	Yes

Chapter 3

Anticipatory Systems

The idea that *anticipations* influence and guide behavior has been increasingly appreciated over the last decades. Anticipations appear to play a major role in the coordination and realization of adaptive behavior. Various disciplines have explicitly recognized anticipations. For example, philosophy had been addressing the sense of reasoning, generalization, and association for a long time. More recently, experimental psychology confirmed that the existence of anticipatory behavior processes in animals and humans over the last decades (Butz et al. 2003).

Anticipation is an important characteristic of intelligence. Proactive behavior requires anticipatory abilities. A seminal work on anticipatory systems is the one written by Rosen (1985). A brief introduction to and serious concern about anticipation follows: "Strictly speaking, an anticipatory system is one in which present change of state depends upon future circumstances, rather than merely on the present or past. As such, anticipation has routinely been excluded from any kind of systematic study, on the grounds that it violates the causal foundation on which all of theoretical science must rest, and on the grounds that it introduces a telic element which is scientifically unacceptable. Nevertheless, biology is replete with situations in which organisms can generate and maintain internal predictive models of themselves and their environments, and utilize the predictions of these models about the future for purpose of control in the present. Many of the unique properties of organisms can really be understood only if

these internal models are taken into account. Thus, the concept of a system with an internal predictive model seemed to offer a way to study anticipatory systems in a scientifically rigorous way" (Rosen 1985).

3.1 Anticipatory Agents

Perception ability is a required characteristic of agents. Hence, they can be designed to perceive current state of self and others. They can also be designed to create current image(s) of future state(s). Perception requires mechanisms that enable interpretive capabilities. Perception invariably involves sensory qualities, and introspection entails accessing sensations and perceptions that agent would introspect. Perceptions are derived as a result of interpretation of sensory inputs within the context of the current world and agent's self model. The prototype inference, orientation accounting, and situational classification mechanisms could be used to realize the interpretation capabilities of an agent. The interpretation process results in perceptions. An anticipatory agent needs to deliberate upon perceptions through introspection and reflection to anticipate. Introspection is deliberate and attentive because higher-order intentional states are themselves attentive and deliberate. An introspective agent should have access mechanisms to its internal representation, operations, behavioral potentials, and beliefs about its context. Reflection used the introspective mechanisms to deliberate its situation in relation to the embedding environmental context. These features collectively result in anticipation capabilities that orient and situate an agent for accurate future projections. Figure 3.1 presents interpretation and introspection as critical components within the micro-architecture of an anticipatory agent.

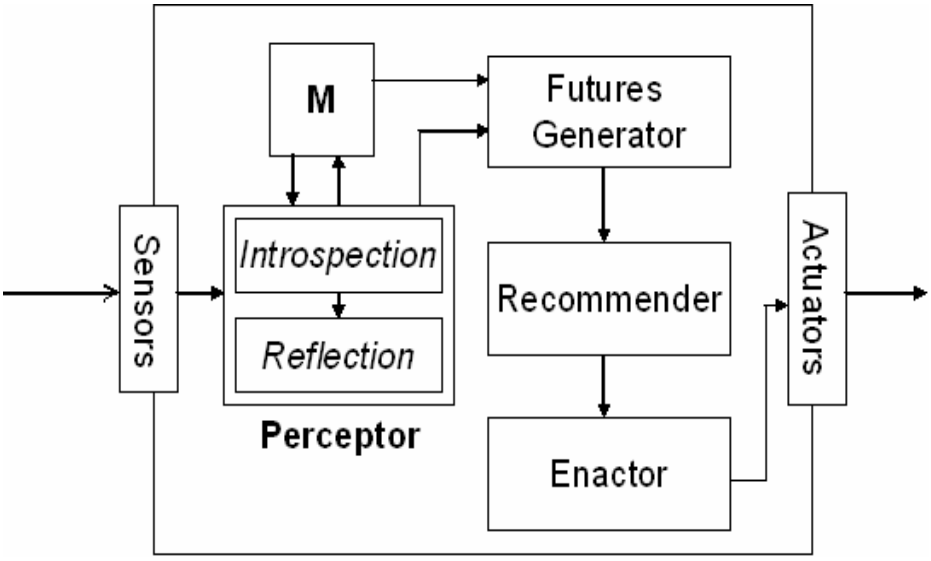

Figure 3.1 Basic Components for Anticipatory Agents

3.2 Preventive State Anticipation

A special kind of anticipation is when an anticipated undesired situation makes an agent adapt its behavior in order to prevent that this situation will occur. For example, assume that we are going out for a walk and that the sky is full of dark clouds. Using our internal weather model and our knowledge about the current weather situation, we anticipate that it will probably begin to rain during the walk. This makes us foresee that our clothes will get wet which, in turn, might cause us to catch a cold, something we consider a highly undesirable state. So, in order to avoid catching a cold we will adapt our behavior and bring an umbrella when going for the walk.

In the suggested framework, an anticipatory agent consists mainly of three entities: an object system (S), a world model (M) and a meta-level component (Anticipator). The object system is an ordinary (i.e., non-anticipatory) dynamic system. M is a description of the environment including S, but excluding the Anticipator. The importance of having an internal model that includes both the agents as part of the

environment and (a large portion of) its abilities has been stressed by, for instance, (Zeigler 1990). The anticipator makes predictions using M and uses these predictions to change the dynamic properties of S. Although the different parts of an anticipatory agent certainly are causal systems, the agent taken as a whole, nevertheless behaves in an anticipatory fashion.

When implementing an anticipatory agent, the component S corresponds to some kind of reactive system similar to the ones mentioned above. This component is referred as the Reactor. The Anticipator corresponds to a more deliberative meta-level component that is able to "run" the world model faster then real time. When doing this, it reasons about the current situation compared to the predicted situations and its goals, and decides whether (and how) to change the Reactor. The resulting architecture is illustrated in Figure 3.2.

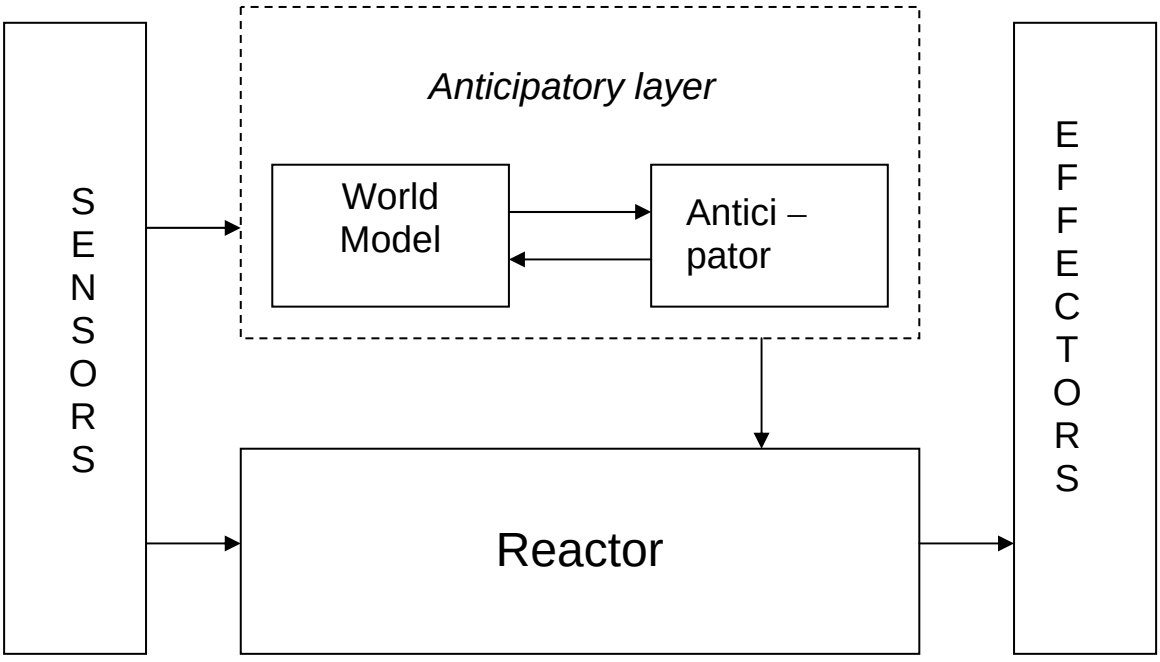

Figure 3.2. The Basic Architecture of an Anticipatory Agent in Our Model.

We can summarize the operation of the architecture as follows: The sensors receive input from the environment. This data is then used in two different ways: (1) to update the World Model and (2) to serve as stimuli for the Reactor. The Reactor reacts to these stimuli and provides a response that is forwarded to the effectors, which then carry out the desired actions(s) in the environment. Moreover, the Anticipator uses the World Model to make predictions, and on the basis of these predictions the Anticipator decides if, and what, changes of the dynamical properties of the Reactor are necessary. Every time the Reactor is modified, the Anticipator should, of course, also update the part of the World Model describing the agent accordingly. Thus, the working on an anticipatory agent can be viewed as two concurrent processes, one reactive at the object-level and one more deliberative at the meta-level (Davidsson 2003).

Chapter 4

Agent-based Modeling of Reactive and Anticipatory Control in Computer Networks

The overall architecture of the simulation is primarily composed of the following components as shown in Figure 4.1

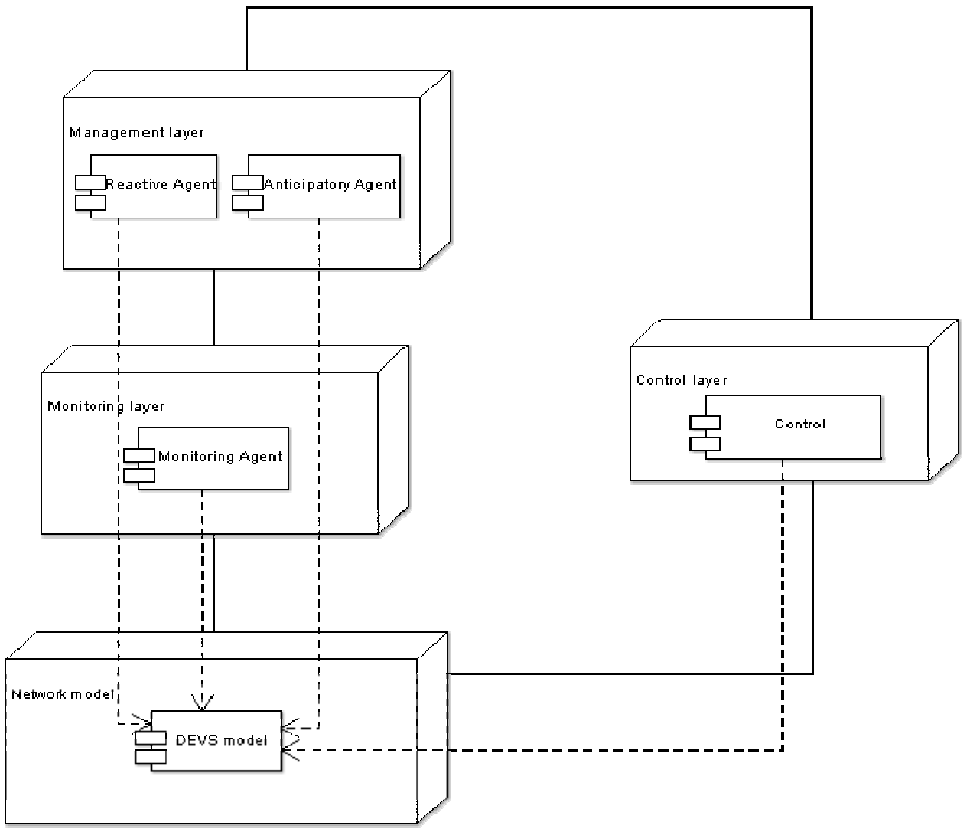

Figure 4.1 Reactive and Anticipatory Control

- **Network Model:** The first component is a basic model of a typical computer network. The network model is the basis of design and experimentation of the

fault management techniques. The network model is designed on a simulation framework and comprises of basic network components that would include switches, routers, hosts and links.

- **Monitoring Layer:** The monitoring layer consists of multiple monitoring agents that are embedded over individual network components or on a group of components. (Eg: a monitoring agent is allocated for each subnet). The monitoring agents may have disjoint functions or potentially overlapping responsibilities for increased reliability.

- **Management Layer:** The management layer comprises of the reactive or the anticipatory agents according to the technique being used. The reactive agent works on a rule based approach. It interprets the data acquired from the monitoring agents and communicates with the control layer to take corrective action (Thottan and Ji, 2003). Similarly, the anticipatory agent works on the principle of a Naïve Bayesian classifier (Luger and Stubblefield, 1989) and interacts with the control layer to take corrective action.

- **Control Layer:** The control layer is responsible for carrying our corrective action with respect to the information if gets from the management layer. The corrective action by the control layer is carried out by triggering local corrective action or a message to the individual components of the network model (Hood and Ji 1998).

4.1 The DEVS Network Model

The network model is developed in the DEVS (Discrete Event System Specification) formalism. A brief description of the simulated network components is as follows:

- Generator: This component is responsible for generation of network packets (payloads to be processed by the hosts).
- Transducer: A Transducer is responsible for calculation of the various network performance metrics.
- Links: Simulation of links is carried out on crucial connections in the network. A link is looked upon as a processor and its overloading is simulated as the increase in processing time of the processor.
- Switch: A switch forms a connection between different subnets to facilitate forwarding of packets among them.
- Router: A router follows routing algorithms such as distance vector routing, link state routing, hierarchical routing, broadcast routing to facilitate forwarding of packets among hosts. We use the *Distance Vector Routing* strategy, by which the packets are forwarded to the best known distance to each destination (the distance is measured in terms of processing time of hosts).
- Hosts: Hosts are entities that process jobs or payload. They can be network clients, servers, printers, plotters etc.

- Monitoring agents: The monitoring agents record performance metrics such as network throughput, latency and packet drop rate. It reports these data to the management layer, where the reactive agents infer using their rules, while the anticipatory agent updates its predictive model.

- Management agents: The management agents are the reactive and the anticipatory agents. They receive data from the monitoring agents and induce the control agent to take respective action.

4.1.1 The DEVS Formalism

The Discrete Event System Specification (DEVS) formalism (Zeigler and Sarjoughian, 2003) provides a means of specifying a mathematical object called a system. Basically, a system has a time base, inputs, states, and outputs, and functions for determining next states and outputs given current states and inputs. Discrete event systems represent certain constellations of such parameters just as continuous systems do. For example, the inputs in discrete event systems occur at arbitrarily spaced moments, while those in continuous systems are piecewise continuous functions of time. The insight provided by the DEVS formalism is the simple way that it characterizes how discrete event simulation languages specify discrete event systems parameters. Having this abstraction, it is possible to design new simulation languages with sound semantics that is easier to understand.

The conceptual framework underlying the DEVS formalism provides is shown in Figure 4.2. The conceptual framework constitutes the following elements:

- Model: It is a set of instructions for generating data comparable to that observable in the real system. The structure of the model is its set of instructions. The behavior of the model is the set of all possible data that can be generated by faithfully executing the model instructions.

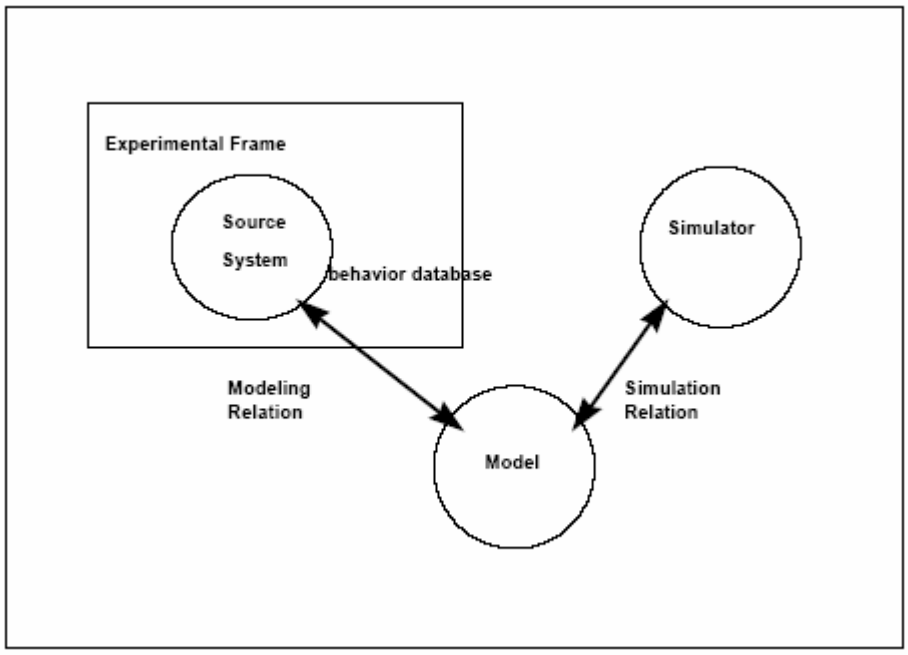

Figure 4.2 Conceptual Framework of the DEVS Formalism (adopted from "Introduction to DEVS Modeling & Simulation with JAVA™", Zeigler and Sarjoughian 2003)

- Simulator: It exercises the model's instructions to actually generate its behavior.
- Experimental Frame: It captures how the modeler's objectives impact on model construction, experimentation and validation. The DEVS experimental frames are formulated as model objects in the same manner as the models of primary interest. In this way, model/experimental frame pairs form coupled model objects with the same properties as other objects of this kind. It will become evident later, that this uniform treatment yields key benefits in terms of modularity and system entity structure representation.

The basic objects are related by two relations:

- Modeling relation linking real system and model defines how well the model represents the system or entity being modeled. In general terms a model can be considered valid if the data generated by the model agrees with the data produced by the real system in an experimental frame of interest.
- Simulation relation, linking model and simulator, represents how faithfully the simulator is able to carry out the instructions of the model.

The basic items of data produced by a system or model are time segments. These time segments are mappings from intervals defined over a specified time base values in the ranges of one or more variables. The variables can either be observed or measured. The structure of a model may be expressed in a mathematical language called formalism. The discrete event formalism focuses on the changes of variable values and generates time segments that are piecewise constant. Thus, an event is a change in a variable value, which occurs instantaneously.

In essence, the formalism defines how to generate new values for variables and the times the new values should take effect. An important aspect of the formalism is that time interval between event occurrences are variable (in contrast to discrete time where the time step is a fixed number).

4.2 Reactive Agents

Fuzzy reactive agents are used in the determination of the proneness of failure. Reactive agents work in a hard-wired stimulus-response manner. Each and every situation must be considered in advance. The reactive agent follows a fuzzy rule based approach to infer the occurrence of a fault. A system becomes fuzzy system when its

operations are entirely or partially governed by fuzzy logic or are based on fuzzy sets. A crisp set is a collection of distinct (precisely defined) elements. In classical set theory, a crisp set can be a superset containing other crisp sets. A superset will represent the universe of discourse if it defines he boundaries in which all elements reside. In any given situation, a new element can be tested to see whether it belongs to any set. On the other hand a fuzzy set is a collection of distinct elements with a varying degree of relevance or inclusion (Berkan and Trubatch 1997). The Reactive Agent gets the network node information through various performance metrics that are being collected by the monitoring agents embedded in the network and uses predefined rules to infer failures based on their degradation. Fuzzy rules consist of antecedents and consequents. The antecedent variables (one or more variables that represent the conditions to be met before any conclusion can be made) comprise of the network throughput and latency. The consequents (set of outputs) comprise of proneness of failure for each of the network component. A sample set of fuzzy rules that are comprised in the reactive agent for a network component (for instance, a host) can be outlined as follows:

- If Throughput is High and Latency is Low then Fault_proneness is Low
- If Throughput is Moderate and Latency is Low then Fault_proneness is Low
- If Throughput is Low and Latency is Low then Fault_proneness is moderate
- If Throughput is High and Latency is Moderate then Fault_proneness is Low
- If Throughput is Moderate and Latency is Moderate then Fault_proneness is Moderate
- If Throughput is Low and Latency is Moderate then Fault_proneness is Moderate

4.3 Anticipatory Agents – A Bayesian Approach

The architectural framework of the Anticipatory agent is described in the previous chapter. It primarily comprises of a predictive model and an anticipator. We make use of a Naïve Bayesian classifier for constructing the predictive model of the anticipatory agent. The strength of the Naïve Bayesian Classifier is that it provides a theoretical framework for combining statistical data with the prior knowledge about the problem domain for making future projections.

Before getting to the Naïve Bayesian, we make an overview of basic probability theory. $p(A/B)$ is known as a conditional probability of event A happening given event B has occurred. We can express the conditional probability, $p(A/B)$ as follows:

$$p(A \mid B) = p(A \cup B) / p(B) \text{ or}$$

$$p(A/B) = (\#\text{ of times A \& B occur}) / (\#\text{ of times B occur}).$$

The following example shows how a fault is detected by the anticipatory agent by making use of the Naïve Bayesian classifier. Consider the sample of evidence specified in Table 4.1.

Table 4.1 Sample Set of Evidence to be Processed by the Naïve Bayesian Classifier

Obs. No	Network	Subnet1	Subnet2	Router1	Host1	Host2	Host3	Router2	Host4	Host5	Host6
1	High	High	Normal	No	Yes	No	Yes	No	No	No	No
2	High	High	Normal	Yes	Yes	No	No	No	No	No	No
3	Normal	Normal	High	No	No	No	No	Yes	No	No	No
4	Normal	Normal	Normal	No	Yes	No	Yes	No	No	No	No
5	High	High	High	Yes	No	Yes	No	Yes	No	No	Yes
6	Normal	Normal	Normal	No	No	No	No	No	No	No	No
7	High	Normal	High	No	No	No	No	No	Yes	No	No
8	Normal	High	Normal	No	No	No	Yes	No	No	No	No
9	Normal	High	Normal	No	Yes	No	No	No	No	No	No
10.	Normal	Normal	High	No	No	No	No	No	No	Yes	No
11.	High	High	High	No	Yes	No	Yes	No	Yes	No	No
12.	High	High	Normal	Yes	Yes	No	No	No	No	No	No
13.	Normal	High	Normal	No	No	No	No	No	No	No	No
14.	High	Normal	High	No	No	No	No	Yes	No	No	Yes

The Network takes value *High* if there is abnormality above a certain threshold in a single or both the subnets and is *Normal* otherwise. The subnet 1 and subnet 2 take value High if any of the component in the respective subnets have failed and is Normal otherwise. The probability that there can be a fault in host 1 provided we have evidence that subnet 1 is high is given by

$$p(\text{Host } 1 = \text{yes} \mid \text{Subnet } 1 = \text{High}) = (\#\text{of times Host } 1 = \text{yes} \,\&\, \text{Subnet } 1 = \text{High}) / (\#\text{of times Subnet } 1 = \text{High}) = 6/8$$

Similarly, the probability that there can be a fault in host 1 provided we have evidence that subnet 1 is high and network is high is given by

$$p(\text{Host } 1 = \text{yes} \mid \text{Subnet } 1 = \text{High}, \text{Network} = \text{High}) = (\#\text{of times Host } 1 = \text{yes} \,\&\, \text{Subnet1} = \text{High} \,\&\, \text{Network} = \text{High}) / (\#\text{of times Subnet } 1 = \text{High} \,\&\, \text{Network} = \text{High}) = 4/5$$

But the number of conditional probabilities in a data set can be very high. Here comes the role of Bayes rule. This is derived as follows:

Given $p(A \mid B) = p(A \cap B) / p(B)$, we know that

$$p(B \mid A) = p(B \cap A) / p(A), \text{ and } p(B \cap A) = p(B / A)\, p(A).$$

Now, since $p(B \cap A) = p(A \cap B)$,

$$p(A \mid B) = \frac{p(B \mid A)\, p(A)}{p(B)},$$

This is known as Bayesian rule.

Consider the following problem with application of Baye's rule:

Given that (Subnet1 = High, Network = High), is there a fault in host 1?

We can express this as:

$$p(Host1 = yes \mid Subnet1 = High, Network = High)$$

$$= \frac{p(Subnet1 = High, Network = High \mid Host1 = Yes)\, p(Host1 = Yes)}{p(Subnet1 = High, Network = Normal)}$$

A general equation for this is:

$$p(C_i \mid A_1 A_2 ... A_n) = \frac{p(A_1 A_2 ... A_n \mid C_i)\, p(C_i)}{\sum_k p(A_1 A_2 ... A_n \mid C_k)\, p(C_k)}$$

However, the conditional probability $p(A_1 A_2 ... A_n \mid C_i)$, may be difficult to compute. If conditional independence among the attributes of the query is assumed, we have the following:

$$p(C_i \mid A_1 A_2 ... A_n) = \frac{p(A_1 \mid C_i)\, p(A_2 \mid C_i) ... p(A_n \mid C_i)\, p(C_i)}{\sum_k p(A_1 \mid C_k)\, p(A_2 \mid C_k) ... p(A_n \mid C_k)\, p(C_k)}$$

The result of Naïve Bayesian Classification is as follows:

$$\text{Result} = \arg\max C_k [\, p(C_k) \Pi p(A_i \mid C_k)], \text{ where}$$

$$p(A_i \mid C_k) = (\#\text{of } A_i \cap C_k) / (\#\text{of } C_k)$$

It can be illustrated by the following example. Suppose we are given that (subnet 1 = High, and Network = High) and we need to know if there is a fault on host 1?

From the above Naïve Bayesian Classifier equation:

$\text{Result}_{(\text{host1 =yes})}$ = p (Host 1 = Yes) * p (Subnet 1 = High $\cap$ Host 1 = Yes) * p (Network = High $\cap$ Host 1 = Yes)

= (6/14)*(1)*(3/6)

= 0.21428

$\text{Result}_{(\text{host1 =no})}$ = p (Host 1 = No) * p (Subnet 1 = High $\cap$ Host 1 = No) * p (Network = High $\cap$ Host 1 = No)

= (8/14)*(3/8)*(3/8)

= 0.08035

Hence we see that $\text{Result}_{(\text{host1 =yes})} > \text{Result}_{(\text{host1 =no})}$, and hence the predictive model predicts the potential of fault in host 1. The Anticipator thereby notifies the control layer to take respective corrective action for host 1. Note that, the set of evidence to the Bayesian classifier is continuously updated according to the events taking place in the network. After a fixed interval of time (say 5 time units), the classifier computes the result (Result = $\arg\max C_k[p(C_k)\Pi p(A_i \mid C_k)]$) based on the state of the components at that time instant.

4.4 Adaptive Restructuring of the DEVS Network Model

Adaptive restructuring can be described as modifying the operating regimes of the DEVS network model in an effort to improve its performance based on the network conditions at a particular instant of time. We intend to find which fault management technique performs better under adaptation. In this section we describe the methodology,

by which operating regimes of the DEVS network model are modified based on certain parameters to preserve the proper functioning of the network.

In case of the DEVS network model, we need to decide the set of operating regimes that we intend to modify in the DEVS environment with the intention that the normal operation of the network model is preserved. Based on the above requirement we come up with the following modes of operation of certain components of the network model. We then club three of those modes to form a particular operating regime.

Following are the modes of operation

Sw_Mode_0: Original operation of the switch.

Sw_Mode_1: Operation of switch with random forwarding of packets to each of the subnet

Ro_Mode_0: Original configuration of the router

Ro_Mode_1: Modification of packet forwarding strategy with forwarding packets to the host with the highest instantaneous value of throughput.

Li_Mode_0: The original value of link delay as defined.

Li_Mode_1: Increase value of link delay by 50%

Li_Mode_2: Decrease value of link delay by 50%

Following are the operating regimes of the DEVS environment

1) Sw_Mode_0 AND Ro_Mode_0 AND Li_Mode_1
2) Sw_Mode_0 AND Ro_Mode_0 AND Li_Mode_2
3) Sw_Mode_1 AND Ro_Mode_0 AND Li_Mode_0
4) Sw_Mode_0 AND Ro_Mode_1 AND Li_Mode_1
5) Sw_Mode_1 AND Ro_Mode_1 AND Li_Mode_0

6) Sw_Mode_1 AND Ro_Mode_1 AND Li_Mode_1

7) Sw_Mode_1 AND Ro_Mode_1 AND Li_Mode_2

Adaptivity can be modeled as a simulated annealing process. Simulated annealing consists of capturing a new state of the DEVS model. The new state is obtained by applying any of the operating regimes. This is followed by recording the performance metric (network throughput) of the network with the new state for a finite amount of time. This recorded metric is then compared with the metric obtained in the previous state (state of the DEVS model before the operating regimes are modified) and the change in performance metric is recorded. The metropolis criterion (Carley and Svoboda 1996) is then used to determine whether or not to adopt the new state. The metropolis criteria states that, a change is always accepted if the forecast performance for a hypothetical organization is better than the known performance of the current organization. A "hypothetical organization" can be interpreted as a new organization that can be obtained by applying design changes to the current organization. Furthermore, when the forecast is poorer that change may still be accepted with a probability which is calculated using the Boltzman equation

$$P = P_0 e^{\Delta \cos t(t)/T}$$

such that cost(t) = 0 – performance (t), and P_0 is the probability of accepting a "bad" design for the previous iteration. The above process is then repeated until the temperature reaches a freezing point or until the simulation time ends, whichever is earlier. Temperature is defined as the model's current level of risk aversion. In other words, the degree to which the DEVS network model is open to accept change of state. The Temperature always drops after every new state has been adopted for the DEVS model.

Freezing point is the point at which a state is in its final form and no more adaptation or change is allowed. (Carley and Svoboda 1996).

To implement the above notion in the DEVS environment, we include an additional component in the network design called as the “annealer”. The total simulation time for the network operation is fixed to 1000 time units and the annealer is made to operate after every 100 time units. After application of a new state, the annealer records the performance metric for a finite amount of time as mentioned above, this finite time is fixed to 50 time units. Each operation of the annealer can be termed as iteration.

The annealer operates based on the following algorithm.

1. Set the initial value of temperature T=0.433 and α = 0.975 where α is the rate at which the DEVS model learns to be risk averse. The initial value of temperature (0.433) corresponds to a probability of 0.9 for changes to be accepted.
2. Derive the new state of the DEVS model by applying an operating regime at random as described above and record the performance metric (network throughput) of the DEVS model in the new state.
3. If the recorded performance metric is better than the one obtained in the old state, continue with step 2, else proceed with step 4

   If the new recorded metric is poorer than the older ones, use the metropolis criteria to determine whether the new state can be adopted in the network.
4. Set the new values of temperature and probability

$$P_0 = P$$

$$T(t+1) = \alpha \times T(t)$$

5. Continue steps 2 thru 5 until a freezing point is reached or the simulation time ends, whichever is earlier. (P = 0.55 and T = 0.345).

The above algorithm, followed by the annealer can be depicted by the following control flow model:

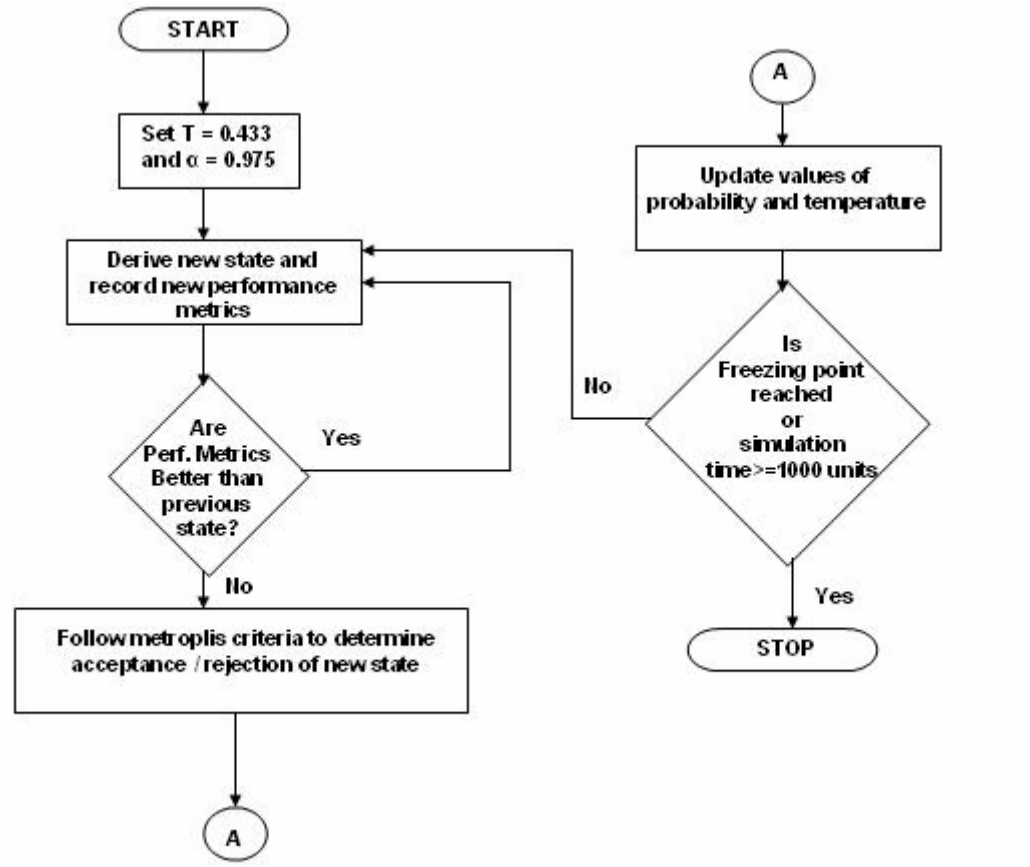

Figure 4.3 Control Flow Diagram Depicting the Operation of the Annealer

We then compare the final results of performance metrics when simulated annealing is implemented, with our original results (without adaptation) for each of the fault management technique to analyze the network performance under adaptive reconfiguration.

Chapter 5

Detailed Design of the DEVS Network Model

Chapter 4 describes the detailed architecture for agent based modeling of reactive and anticipatory control in computer networks. This chapter provides details pertaining to design of the DEVS network simulation model that implements a distributed network monitoring system (DNM) (Prietula et al. 1998).

5.1 The DEVS Basic and Coupled Models

In the DEVS formalism, one must specify 1) basic models from which larger ones are built, and 2) how these models are connected together on hierarchical fashion. A basic model contains the following information

- the set of input ports through which external events are received
- the set of output ports through which external events are sent
- the set of state variables and parameters: two state variables are usually present, “phase” and “sigma” (in the absence of external events the system stays in the current “phase” for the time given by “sigma”)
- the time advance function which controls the timing of internal transitions - when the “sigma” state variable is present, this function just returns the value of “sigma”.
- the internal transition function which specifies to which next state the system will transit after the time given by the time advance function has elapsed.

- the external transition function which specifies how the system changes state which an input is received – the effect is to place the system in a new “phase” and “sigma” thus scheduling it for a next internal transition; the next state is computed on the basis of the present state, the input port and the value of the external event, and the time that has elapsed in the current state.
- the confluent transition function which is applied when an input is received at the same time that an internal transition is to occur - the default definition simply applies the internal transition function before applying the external transition function to the resulting state
- the output function which generates an external output just before an internal transition takes place.

Basic models may be coupled in the DEVS formalism to form a Coupled model. A coupled model tells how to couple (connect) several component models together to form a new model. This latter model can itself be employed as a component in a larger coupled model, thus giving rise to a hierarchical construction. A coupled model contains the following information

- the set of components
- the set of input ports through which external events are received
- the set of output ports through which external events are received
- the external input coupling which connects the input ports of the coupled model to one or more of the input ports of the components
- the external output coupling which connects output ports of components to output ports of the coupled model, thus when an output is generated by a component it

may be sent to a designated output port of the coupled model and thus be transmitted externally

- the internal coupling which connects output ports of components to input ports of other components , hence when an input is generated by a component, it may be sent to the input ports of designated components (in addition to being sent to an output port of the coupled model) (Zeigler and Sarjoughian, 2003).

5.2 The Distributed Network Monitoring System

A distributed network monitoring (DNM) system consists of a hierarchical structure with a set of network components and is endowed with monitoring agents that cooperate in monitoring the network. The network can be divided into several regions or sub networks (in our case we consider two distinct subnets). Within each sub network, a set of monitoring agents are jointly responsible for maintaining up-to-date models of host and router performance and availability. These monitoring agents belong to the monitoring layer as described in the previous chapter. Monitoring agents are responsible for notifying the management layer regarding the status of network components as well as sub networks. The management layer which consist of the reactive and the anticipatory agents, utilize the data acquired from the monitoring layer to make control decisions for management of network faults. Figure 5.1 shows a hypothetical network structure based on the notion of distributed network monitoring system (DNM).

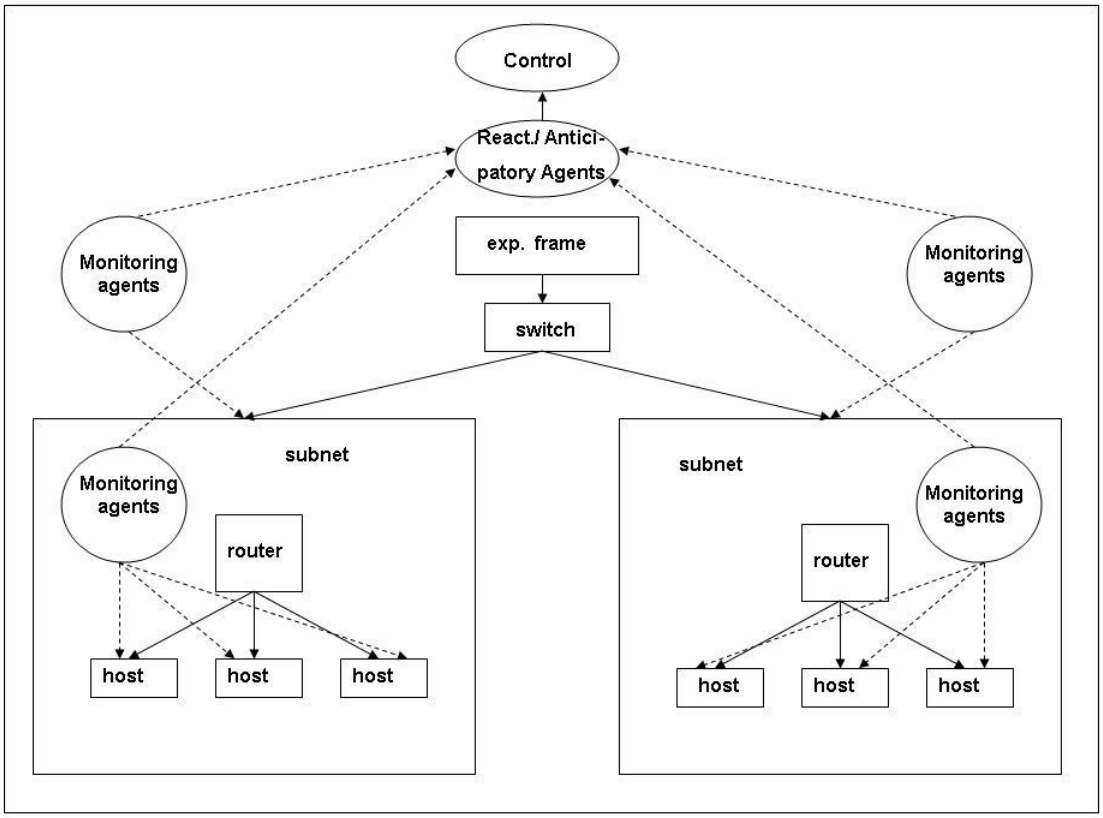

Figure 5.1 Designed Network Model

5.3 Component Overview

We give a brief description and functions of the crucial components in our network model

- Experimental frame

  The experimental frame primarily consists of 3 sub components, the generator, the transducer and the fault injection mechanism. The generator generates packets to be processed by the network components on the basis of a specific inter-arrival time. The transducer is responsible for computation of network performance metrics (throughput, latency and drop rate of packets). The throughput is defined as the average rate of job departures from the architecture, estimated by the

number of jobs processed during the observation interval, divided by the length of the interval. A job's turnaround time is the length of time between its arrival to the processor and its departure as a completed job. The drop rate of packets is defined as the percentage of packets dropped due to network faults. The fault injection mechanism, which is embedded in the experimental frame, generates "fault packets" at a random rate. A "fault packet" when encountered by a network component, induces a certain level of degradation in the throughput and latency of the component.

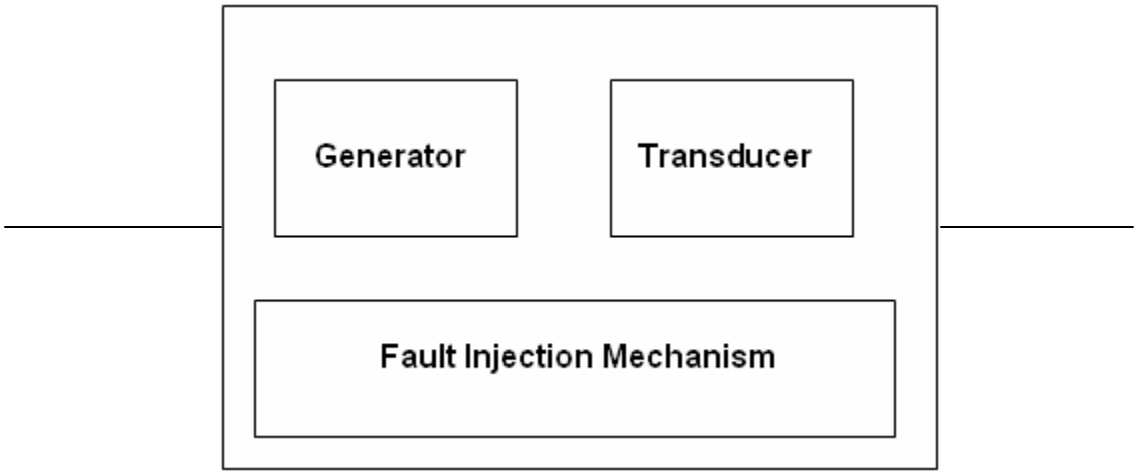

Figure 5.2 Experimental Frame

The activity diagram for the experimental frame is shown in Figure 5.3. The generator and the fault injection mechanism start as soon as the simulation begins. The transducer computes the performance metrics based on the number of packets and the number of faults incurred. The packet generator and the fault injection mechanism cease to operate when the simulation time ends.

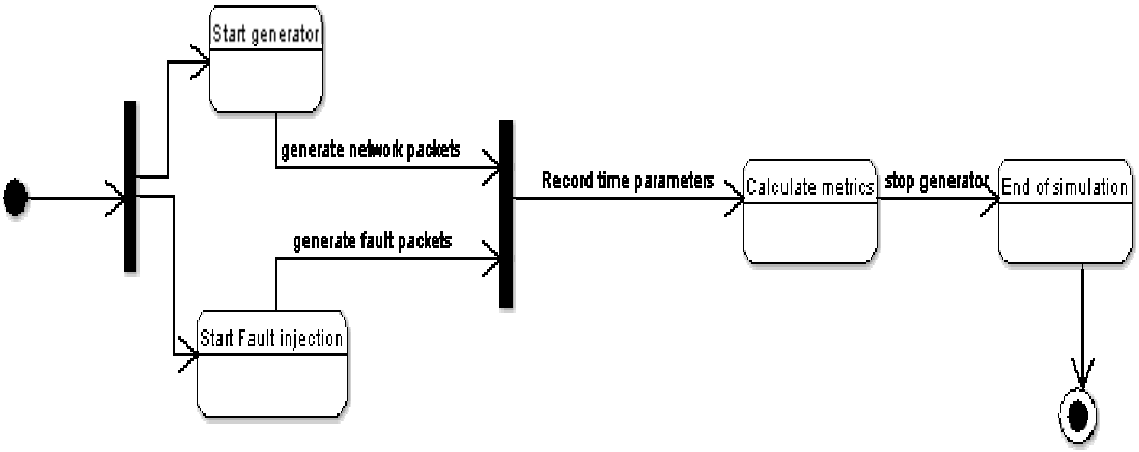

Figure 5.3 Activity Diagram of the Experimental Frame

The interaction among the different components of the experimental frame is shown by a sequence diagram in Figure 5.4. As shown in the sequence diagram, the generator initially starts generating network packets to be processed by the hosts in the network. As soon a packet is generated, the transducer is simultaneously triggered. The transducer records the simulation time the packet is generated. On completion of packet processing by any of the hosts in the network, the transducer records the completion time. Based on the arrival and completion time parameters, it computes the value of throughput and turn around time. If the transducer fails to record the completion time due to packet loss, it records the packet as being lost and appends it to the list of dropped packets which is used to calculate the drop rate of packets. On completion of the simulation time, when no more metrics are to be recorded, the transducer triggers the generator and the fault injection mechanism to cease generation of network packets and fault packets respectively.

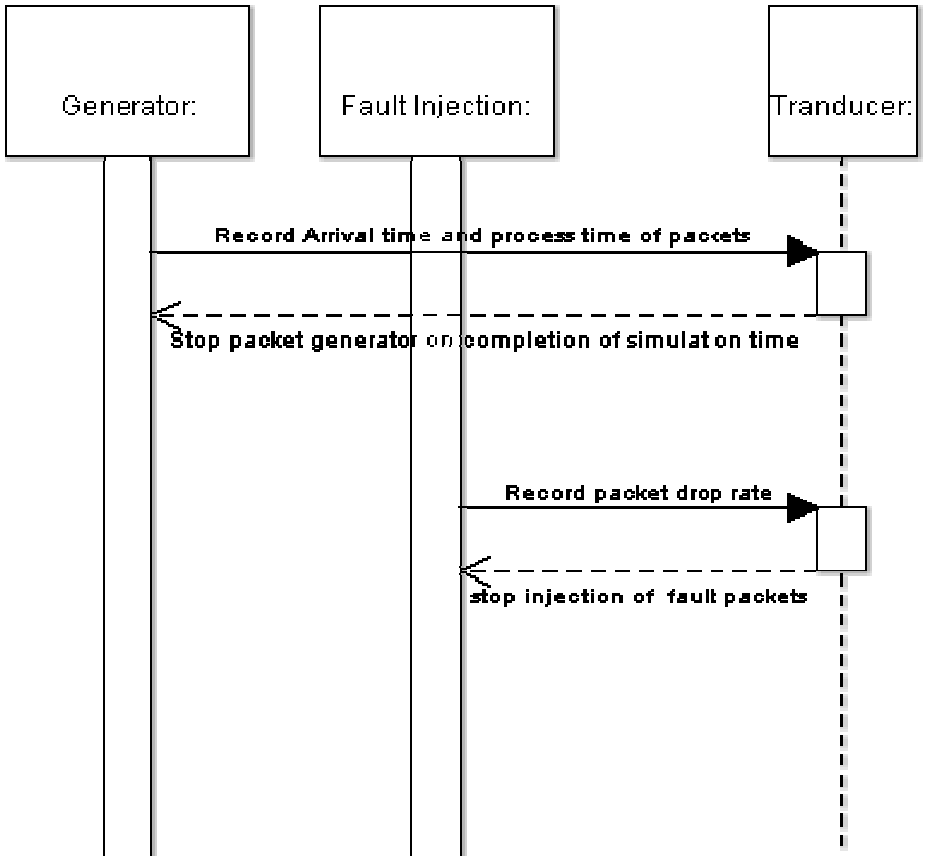

Figure 5.4 Sequence Diagram for Interactions of Various Components within the Experimental Frame

Switches, routers, and hosts:

The operational specifications of the switch, router, and hosts are discussed in the previous chapter. We now give a brief description about the effect on each of these components due to a fault. The switch, router, and the hosts degrade in a similar way when a fault packet in encountered. The degradation can be seen as a 3 step process. On encountering the first fault packet, the component's normal working is disrupted and it's said to change to a "low degradation" state or in other words when a fault is encountered, the processing time of these components is doubled. This can be interpreted as the fact that

degradation causes the components to delay the operation they are carrying out. This in turn affects the throughput, turn around time and drop rate of packets pertaining to that component and hence the subnet to which it belongs and consequently the complete network. On encountering a control packet, the operation is again returned to normal. Similarly, if another fault packet is encountered before a control packet, the components degrade further to a state of "moderate degradation" and furthermore "high degradation" after which it completely ceases to operate. The activity diagram describing the behavior of a network component on encountering fault packet(s) is as shown in Figure 5.5.

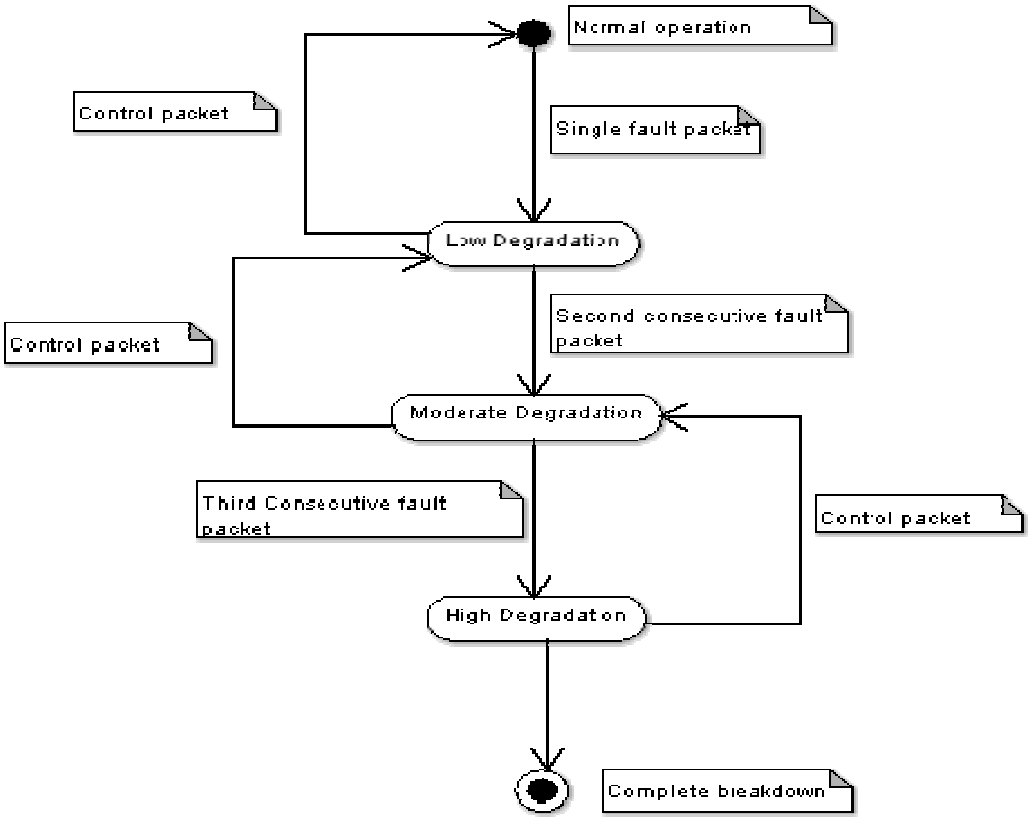

Figure 5.5 Activity Diagram of Network Component due to Fault(s).

- **Links**

  Links are the interface between different components of the network. They regulate the flow of packets. Since we need to vary the delay of the links for experimental purpose, the links should be implemented in such a way that the variation of delay is practical. Hence we implement a link as a form of processor which can be considered as an entity which takes some finite amount of time (link delay) to process a job and forward it. The DEVS processor component, which models a link, has no buffering capability. Therefore, when a job arrives while the processor is busy, it simply ignores it. This affects the drop rate of packets.

5.4 Monitoring Agents

Monitoring agents are deployed within each of the subnets to record the individual performance metrics of components. These metrics include the throughput, the turn around time, and the drop rate of packets. The fault proneness of network components is being reported by the monitoring agents to the reactive or anticipatory agents, which in turn take the required action according to their functionality. The activity diagram depicting the behavior of an individual monitoring agent is shown in Figure 5.6.

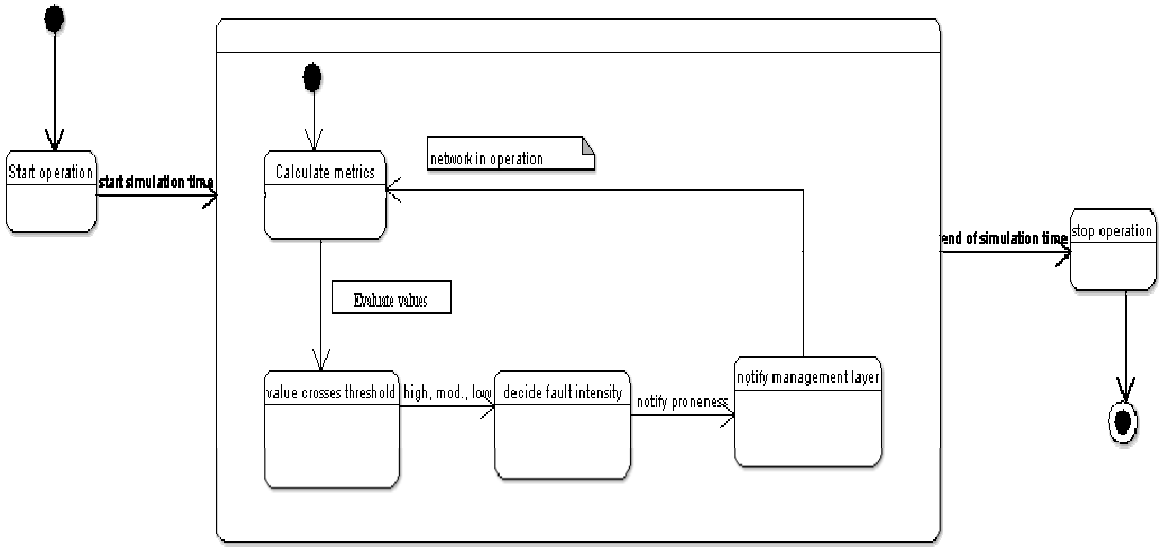

Figure 5.6 Activity Diagram of a Monitoring Agent

The monitoring agents are being deployed at various levels in the DEVS network model. Those include, (1) the component level monitoring agents that monitor the individual performance of routers and hosts, (2) the subnet level monitoring agents that monitor a subnet as whole, and (3) the network monitoring agent that monitors the performance of the complete network. The sequence diagram shown in Figure 5.7 depicts the interaction between the monitoring agents at different levels in the DEVS network model.

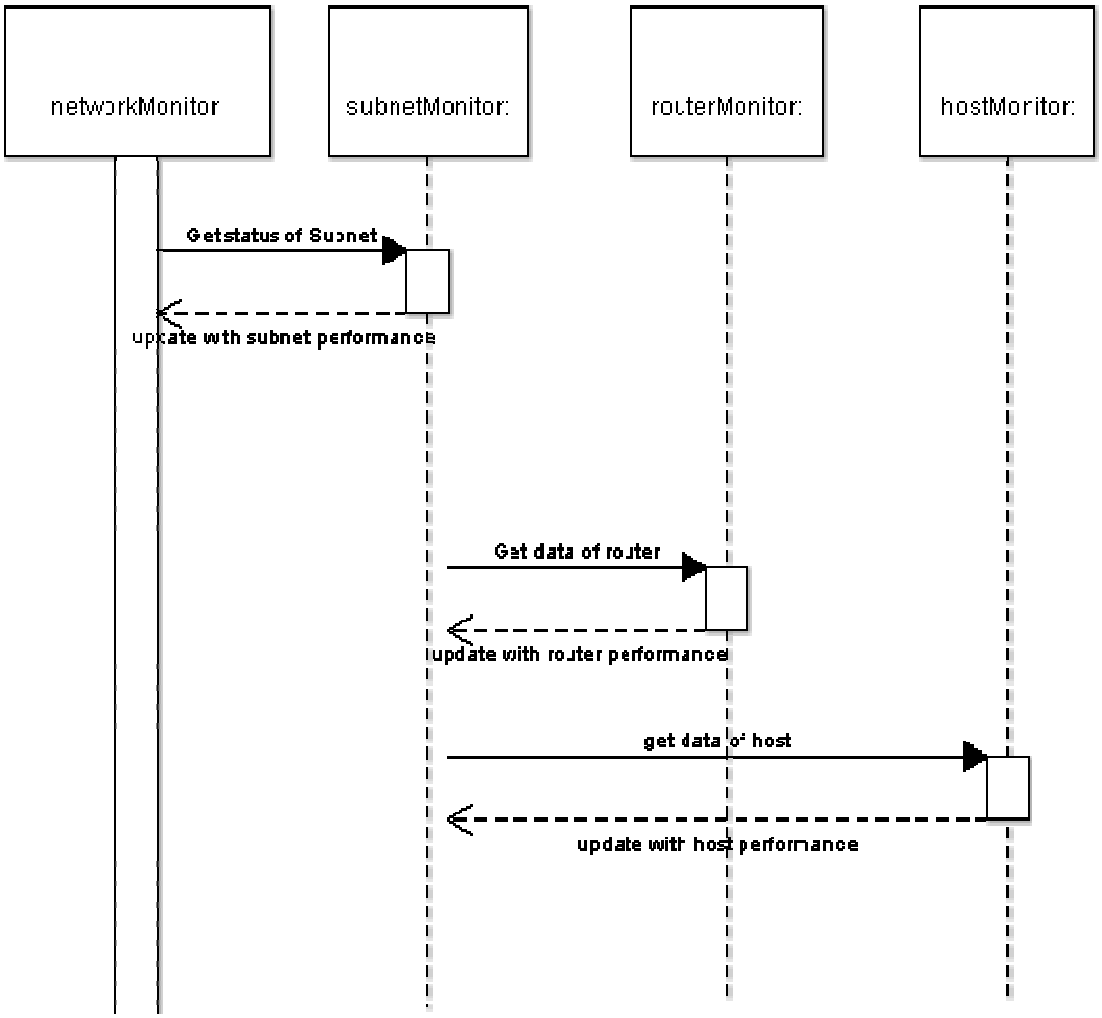

Figure5.7 Sequence Diagram showing Interaction between Monitoring Agents

5.5 Reactive / Anticipatory agents

The data received from the monitoring agents form as inputs to the reactive and anticipatory agents. The reactive agent functions on simple fuzzy rules while the anticipatory agent functions on the basis of a Naïve Bayesian classifier as described in the previous chapter. Figure 5.8 shows the activity diagram of the reactive agent.

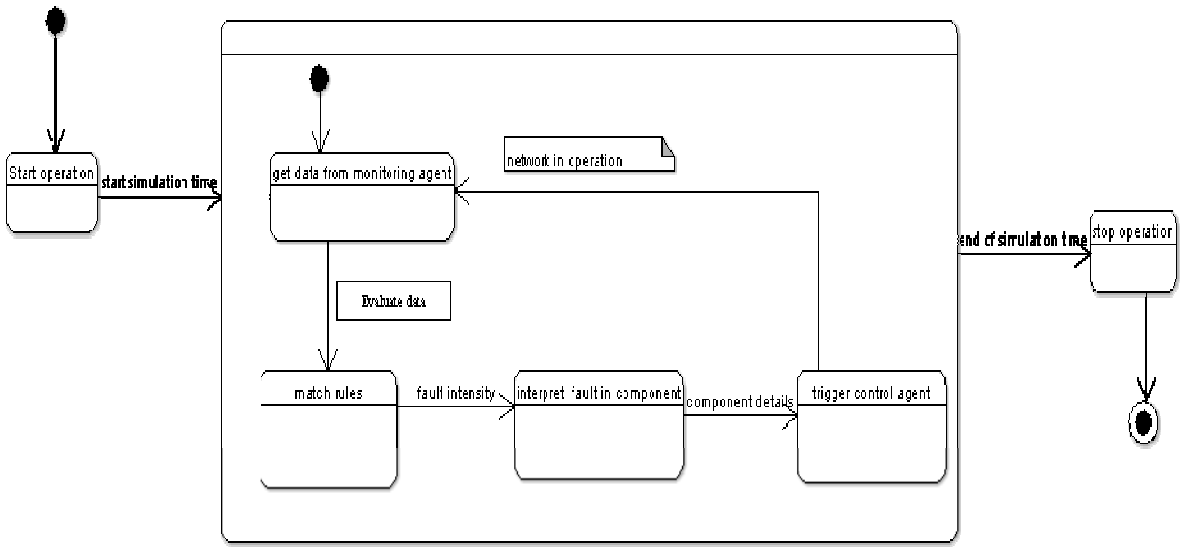

Figure 5.8 Activity Diagram of a Reactive Agent

As shown in the Figure 5.8, the reactive agent gets the data from the monitoring agent, which has the details of proneness of faults in the various components among the network. The reactive agent then matches those with the predefined fuzzy rules and triggers the control agent to take respective action. The activity diagram of the anticipatory agent is as shown in Figure 5.9. In contrast to the reactive agent, the anticipatory agent has an additional “learner” component which builds a predictive model of fault proneness in the network based on the Naïve Bayesian Classifier. It then computes probabilities of failure of the components as described in the Chapter 4. and thereby triggers the control agent to take corrective action.

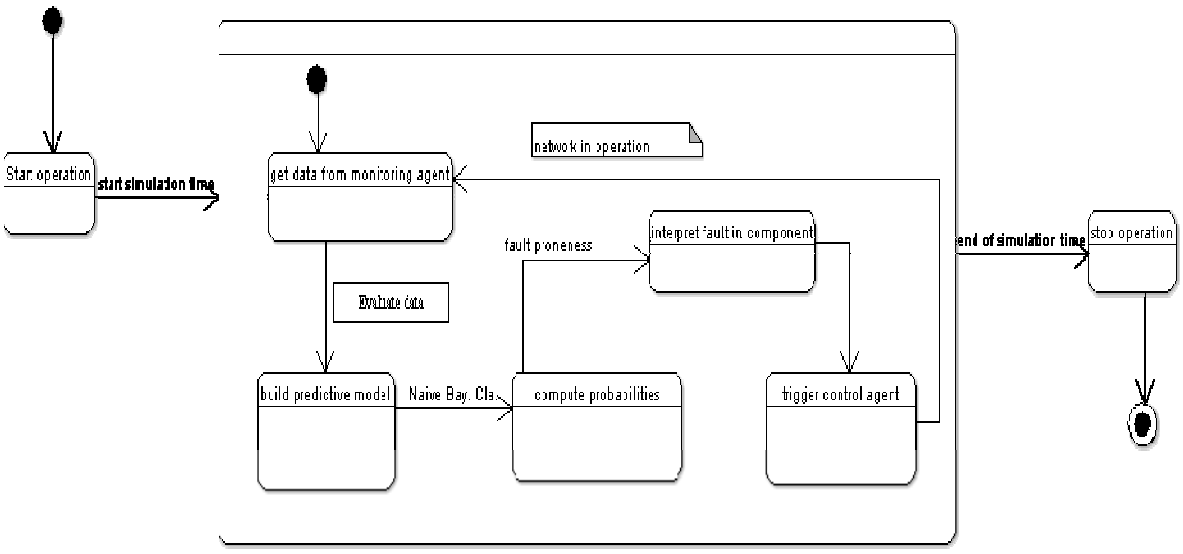

Figure 5.9 Activity Diagram of an Anticipatory Agent

The interactions between the monitoring and the management agents (reactive and anticipatory) are shown with a sequence diagram in Figure 5.10. From the sequence diagram, it can be seen that the reactive agent is supplied data only by the component level monitoring agents. Since the reactive agent works on simple fuzzy rules, it interprets the data obtained by the component level monitoring agents to determine the components that require corrective action. The anticipatory agent, on the other hand, predicts the proneness of fault among the various network components, and hence it needs a complete picture of the network. The monitoring agent at the subnet and the network level help the anticipatory agent to dynamically learn the network behavior for the proneness of faults and thereby constantly update the evidence of the Naïve Bayesian Classifier.

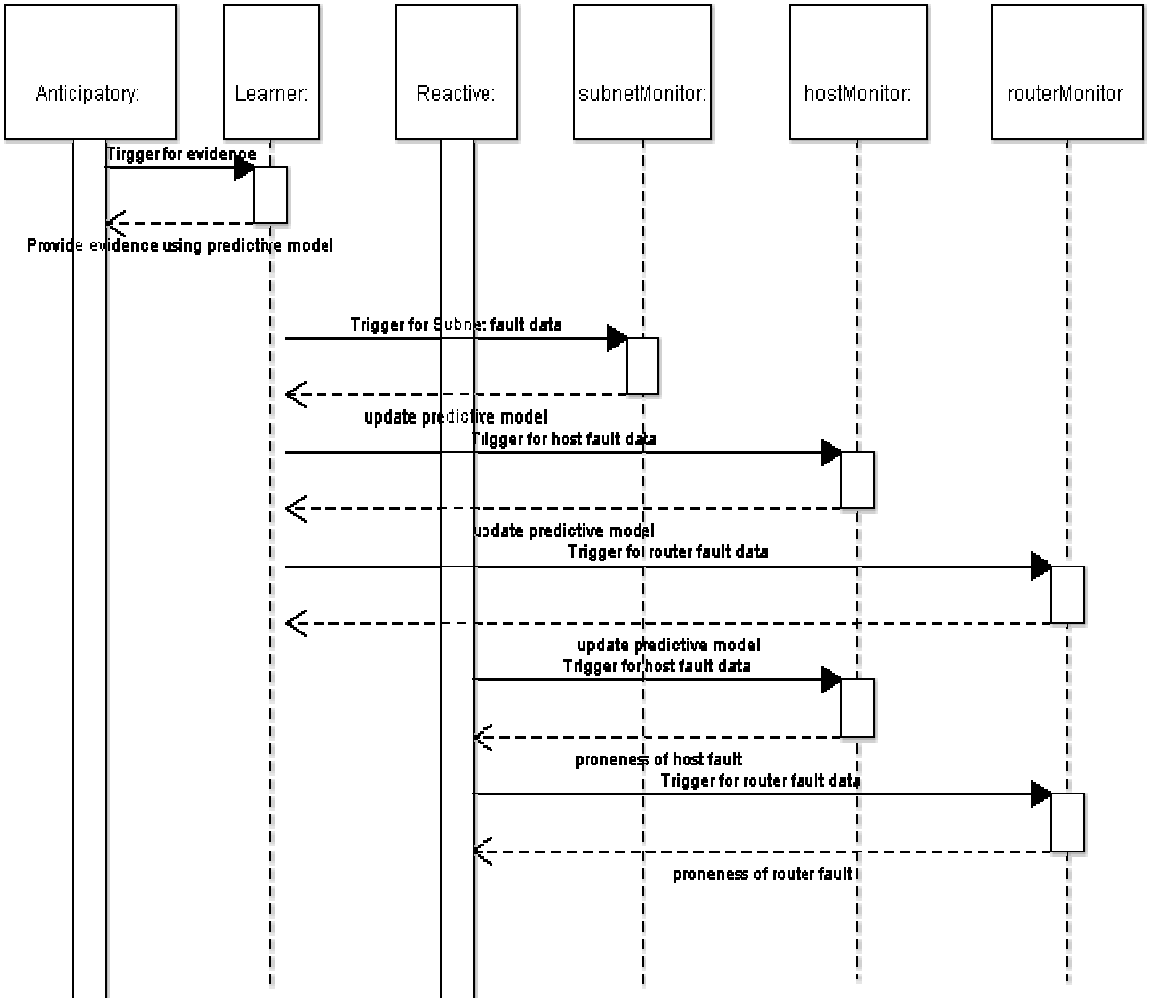

Figure 5.10 Sequence Diagram for Interaction between Monitoring and Management Agents

5.6 Control Agent

The control agent operates on the basis of the output from the reactive and anticipatory agents. The data it obtains from the management agents consists of details pertaining to the network component which is under degradation. The control layer then triggers corrective action to those components in the form of corrective messages (Hood and Ji 1998). When these corrective messages are encountered by network components,

they regain their normal operation. The activity diagram of the control agent operation is shown in Figure 5.11.

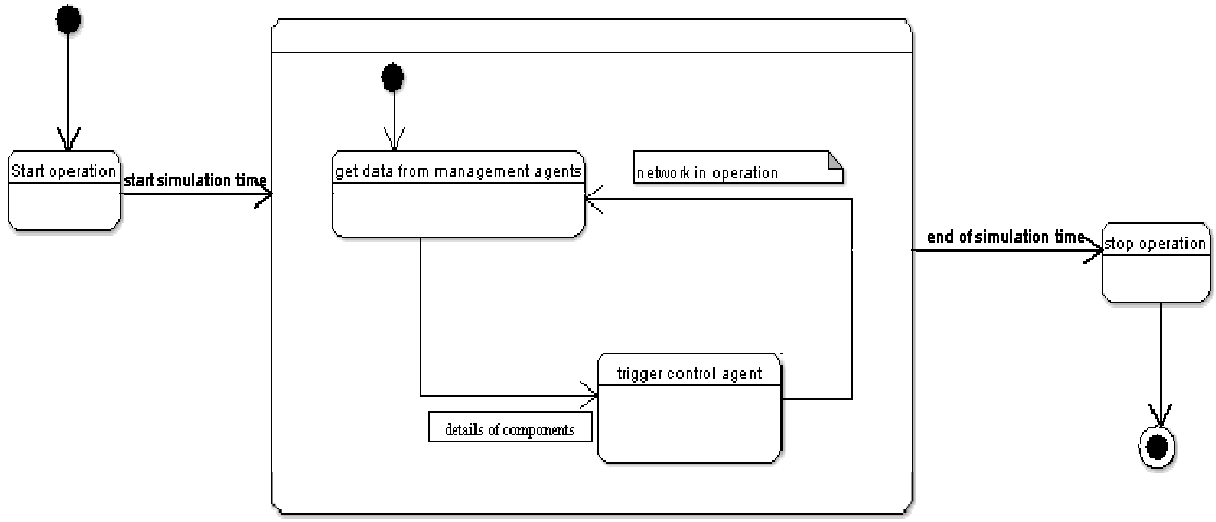

Figure 5.11 Activity Diagram of a Control Agent

The interactions among the control agent and the management agents are as shown below (Figure 5.12). As described above, the control agent is only responsible to trigger corrective action to the respective components that are under degradation or may be prone to degradation based on the output from the reactive or anticipatory agent respectively.

5.7 Annealer

The function of the annealer is to facilitate dynamic updating of the parameters of the DEVS model based on different operating regimes. The annealer operates independently of the management and control agents as described in the previous chapter. The activity diagram depicting the various states traversed is as shown in Figure 5.13. The annealer starts by modifying the operating regimes of the DEVS network based on certain parameters to reconfigure the network. It then records performance metrics of the

reconfigured network for a finite amount of time. If the recorded metrics of the reconfigured network are better than the previous configuration, the new configuration with the new operating regimes is adapted.

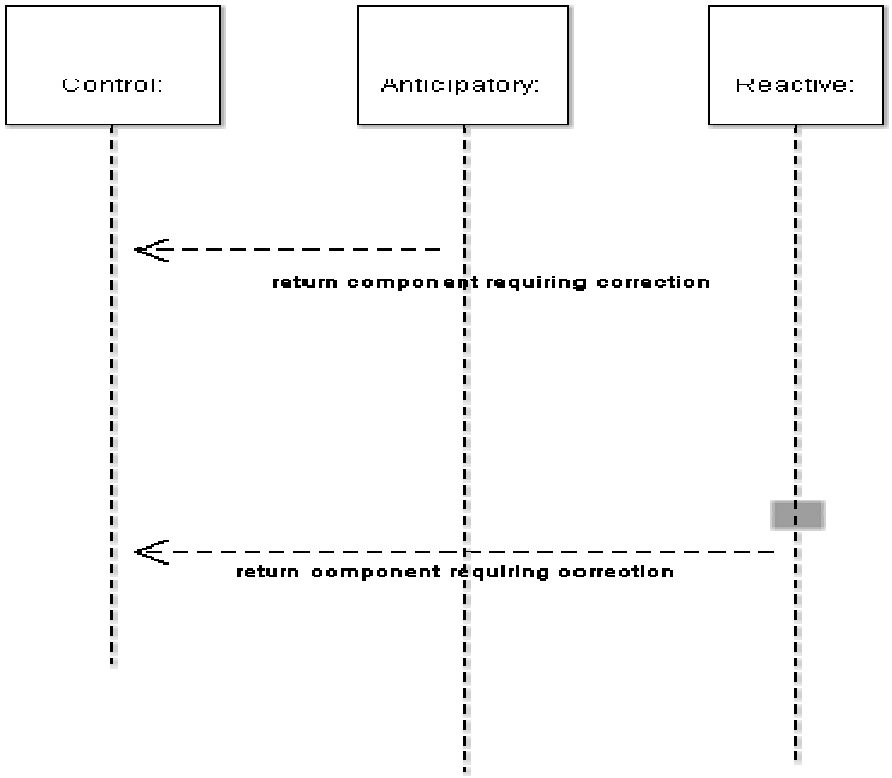

Figure 5.12 Sequence Diagram for Management and Control Agents.

If the metrics recorded are poorer, the metropolis criterion is used to decide whether or not to adopt the new state. According to the metropolis criteria, the new configuration (with poor performance) is accepted with a certain probability that depends on the temperature of the network (the probability decreases as temperature decreases). Temperature is defined as the degree to which the DEVS network model is open to

accept change of state. The temperature drops after every new state has been adopted for the DEVS model.

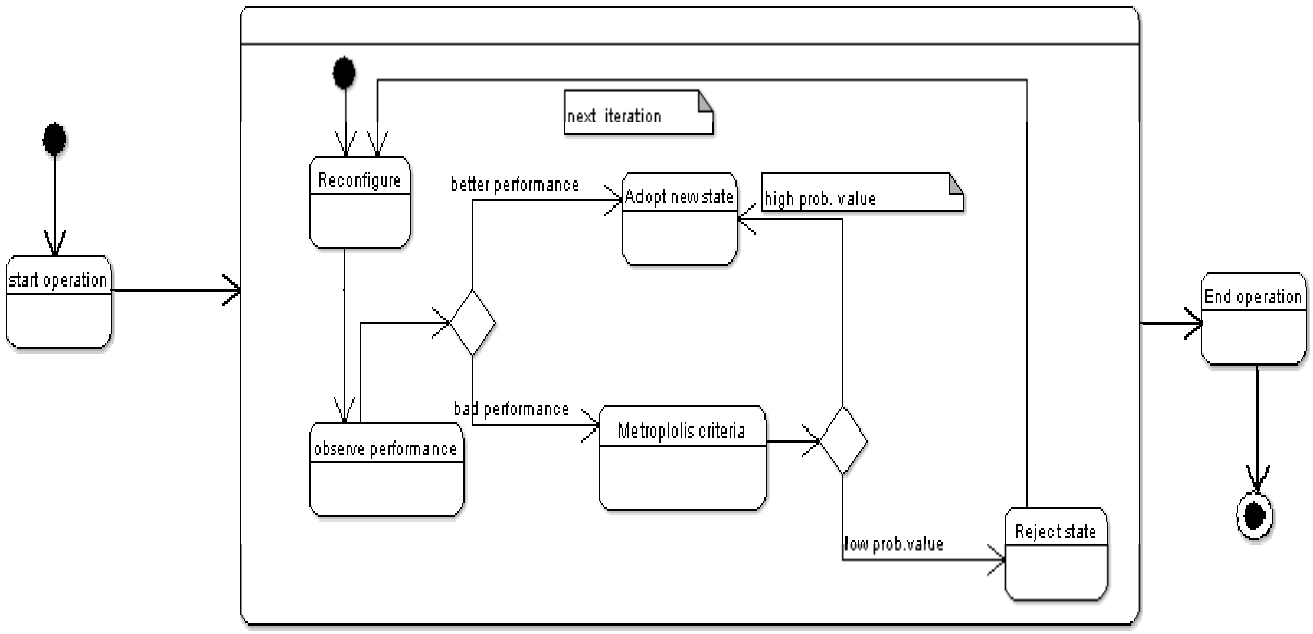

Figure 5.13 Activity diagram of the annealer

A complete class diagram of all the components described above is shown in Figure 1 of Appendix. The class Entity, Devs, Coupled, Viewabledigraph and Viewableatmic form the basic components of the DEVS framework. Entity is the base class of objects to be put into containers. The class Devs contains two main model classes, atomic and coupled. The class atomic realizes the atomic level of the underlying DEVS formalism. It has elements corresponding to each of the parts of this formalism. Coupled is the major class which embodies the hierarchical model composition constructs of the DEVS formalism. A coupled model is defined by specifying its component models. Components are instances of the Devs class thus enabling hierarchical composition. Class Viewable digraph is a derived class of coupled which

enables to define a coupled model in an explicit manner. In addition to components, it enables the specification of the coupling relation, which establishes the desires communication links among the components (internal coupling) and between them and the external world (external input and external output coupling). The processor class is a simple processor representing storage of jobs and passage of time for its execution. The class switch, control, reactive, subnet1, subnet2, generator, transducer, anticipator, monitor have their respective functions as described in the above sections and in previous chapters. The Multiserver coordinator routes incoming jobs for processing and collects results for final output.

Chapter 6

Experiment Design and Simulation Results

The following chapter describes the detailed experimental design of the network model designed in DEVS followed by experimental results. We make use of Borland Jbuilder™ as the Integrated Development Environment (IDE) for implementing the network model in DEVS.

6.1 Experiment Design

The DEVS-based network model comprises of two subnets. Each subnet includes a router and 3 hosts. An experimental frame generates the packets to be processed by the network components on the basis of a specific inter-arrival time. A fault injection mechanism is also embedded in the experimental frame which generates “fault packets” at a random rate. A “fault packet” when encountered by a network component, induces a certain level of degradation in the throughput and latency of the component. Monitoring agents are deployed throughout the network over each of the network components to record the performance metrics (throughput, latency and the drop rate) throughout the simulation. The throughput is defined as the average rate of job departures from the architecture, estimated by the number of jobs processed during the observation interval, divided by the length of the interval. A job’s turnaround time is the length of time between its arrival to the processor and its departure as a completed job. The drop rate of packets is defined as

the percentage of packets dropped due to network faults. A sample screen shot of the DEVS environment is shown in Figure 6.1.

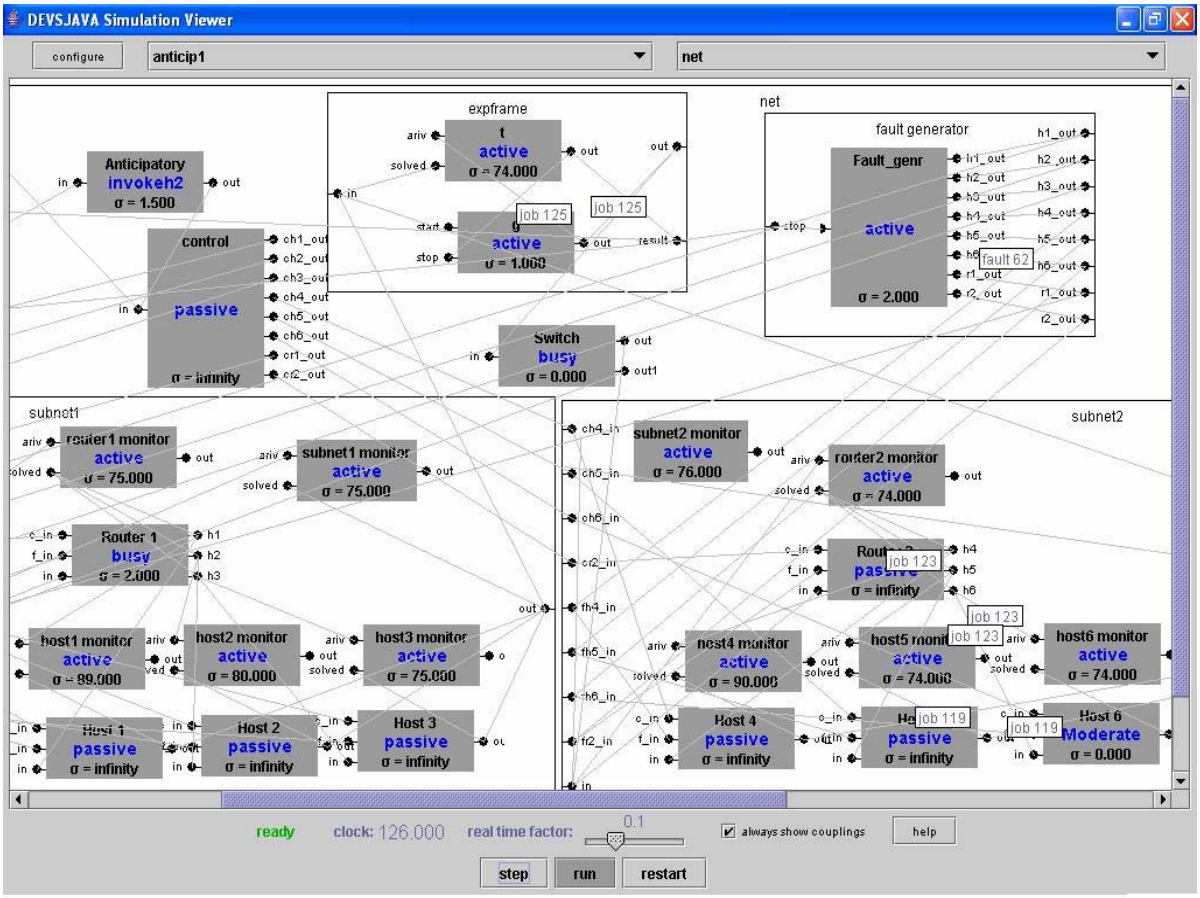

Figure 6.1 Simulated Model in DEVS Environment

6.2 Simulation Results

Each of the fault management techniques (reactive, alarm correlation and anticipatory) are simulated by varying the levels of the link delay and the complexity of the network. The number of replications for each fault management technique is 270. This results from the sum of the replications under each combination of configuration levels (i.e., link delay, network complexity). Each replication is run for 1000 time units. The t test is performed with respect to the mean values obtained for throughput,

turnaround time and the drop rate of packets for each of the fault management technique and the confidence intervals are recorded. From the confidence intervals obtained at 95 percent level, we observe that the intervals obtained for the reactive vs. anticipatory and anticipatory vs. alarm correlation for all the three parameters does not contain zero and hence the difference in their mean values is statistically significant. The intervals obtained for the reactive vs. alarm correlation technique for all the three parameters comprises of zero. Hence the difference between their means is not statistically significant. Table 6.1 shows the results of the t-test.

Table 6.1 Confidence Intervals for Performance Metrics

	Reactive	Alarm Correlation	Anticipatory
Reactive	-------	(-0.001,0.019)	(-0.025,-0.006)
Alarm Correlation	(-0.019,0.001)	-------	(-0.035,-0.014)
Anticipatory	(0.006,0.025)	(0.014,0.035)	-------

6.1.1 Performance of Network Throughput

	Reactive	Alarm Correlation	Anticipatory
Reactive	-------	(-38.69,15.55)	(25.11 , 70.19)
Alarm Correlation	(-15.55, 38.69)	-------	(33.62, 84.81)
Anticipatory	(-70.19, -25.11)	(-84.81, -33.62)	-------

6.1.2 Performance of Network Turnaround Time

	Reactive	Alarm Correlation	Anticipatory
Reactive	-------	(-8.54, 1.96)	(5.67, 9.59)
Alarm Correlation	(-1.96,8.54)	-------	(5.9, 15.93)
Anticipatory	(-9.59, -5.67)	(-15.93, -5.9)	-------

6.1.3 Performance with Respect to Drop Rate of Packets

We analyze the behavior of each of the performance metrics (throughput, turnaround time and drop rate of packets) with respect to the variation of link delay and complexity of the network, for each of the fault management techniques. We plot response surfaces with respect to each of the dependent variables that include throughput, turn around time and drop rate of packets, against the independent variables (link delay and complexity). Figure 6.2 shows the responses obtained.

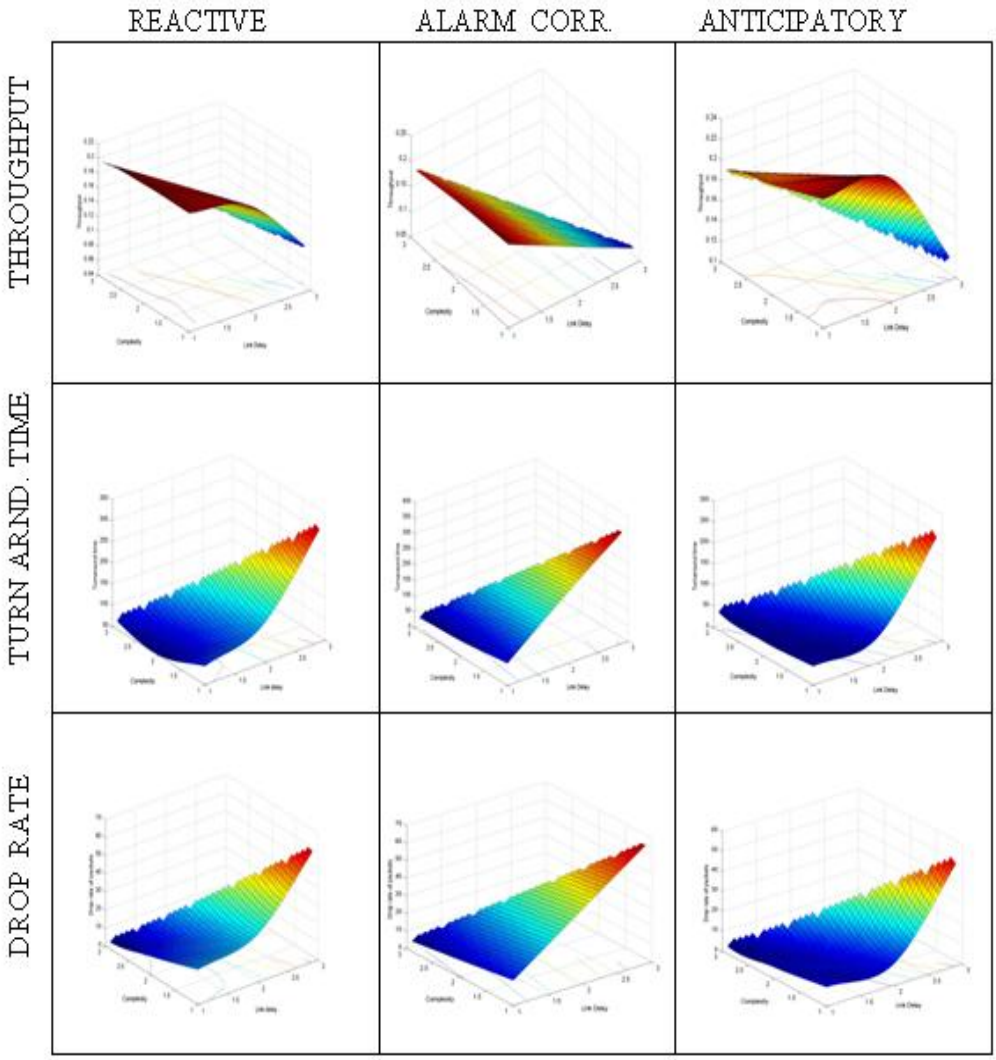

Figure 6.2 Response Surfaces

We observe that the throughput obtained in each of the techniques is significantly better at a lower value of link delay while throughput is less dependent on the complexity of the network. There is a considerable improvement in the turnaround time at higher levels of complexity. For the drop rate of packets, the percentage is significantly less at lower values of link delay; also, there is a significant reduction of drop rate of packets at higher values of complexity. As shown in Figure 6.2, performance parameters for the alarm correlation technique shows a linear dependency with respect to variation of the link delay. Also, reactive and anticipatory techniques are less prone to link delay until a certain threshold. The linearity exhibited by the alarm correlation technique can be explained by the fact that the fault patterns are recorded beforehand and hence the variation of the performance metrics is linear, whereas for the other two techniques this is not the case.

6.2.1 Sensitivity Analyses

We perform sensitivity analysis based on each level of network complexity. We fix the value of complexity and analyze the variation of each of the performance metrics with respect to variation of link delay. The graphs obtained for the reactive, alarm correlation and the anticipatory techniques are shown in Figure 6.3, Figure 6.4, and Figure 6.5 respectively.

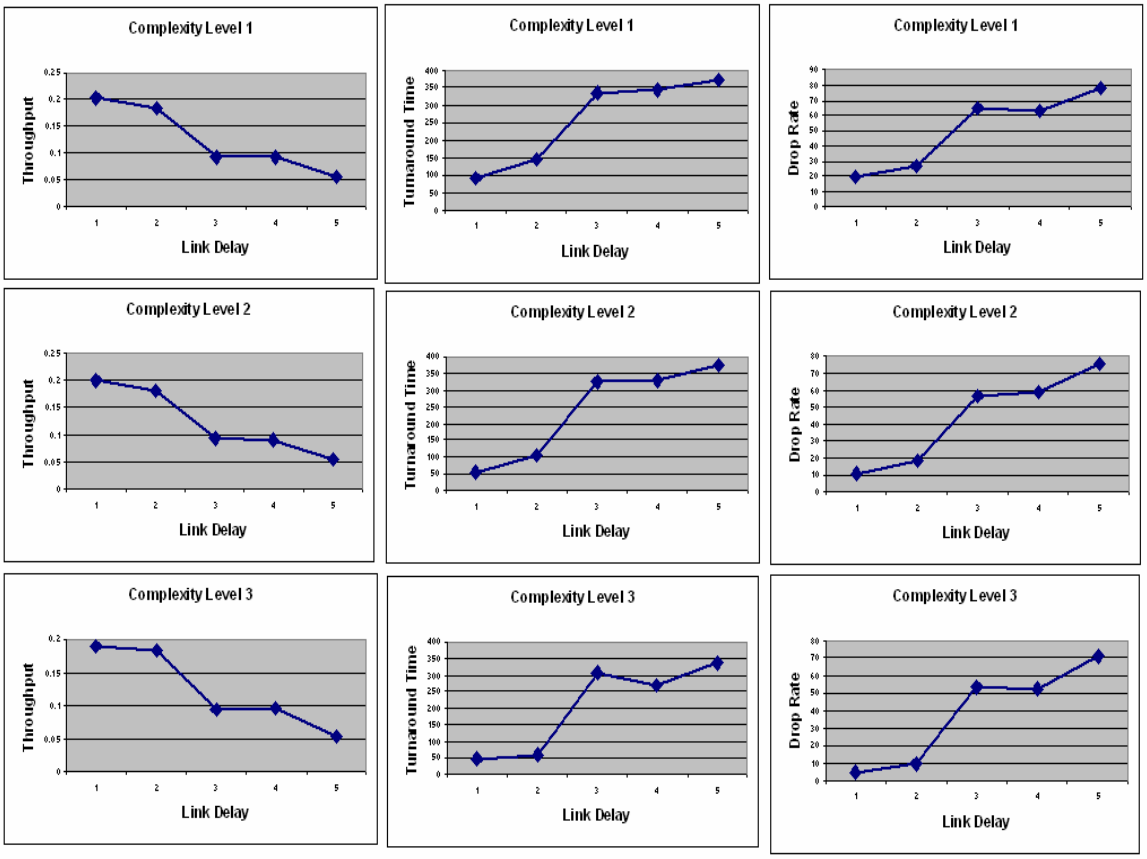

Figure 6.3 Sensitivity Analyses for Variation of Complexity for Reactive Technique

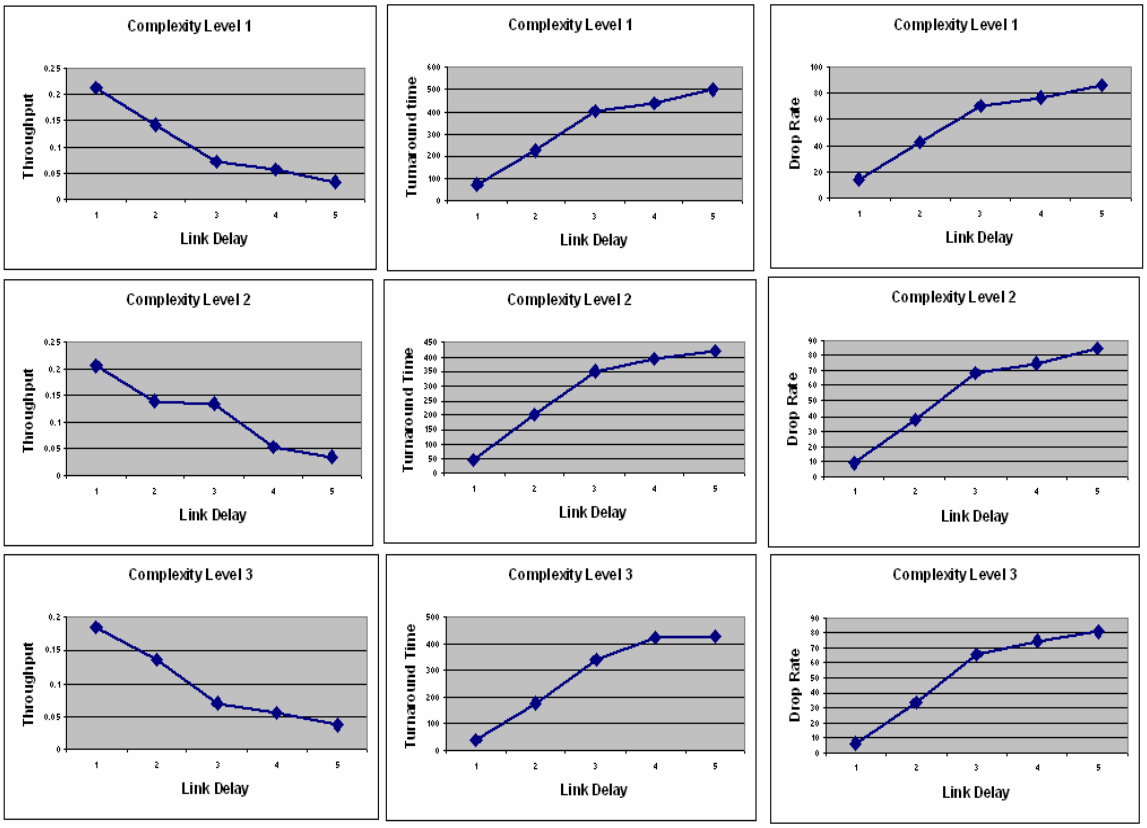

Figure 6.4 Sensitivity Analyses for Variation of Complexity for Alarm Correlation Technique

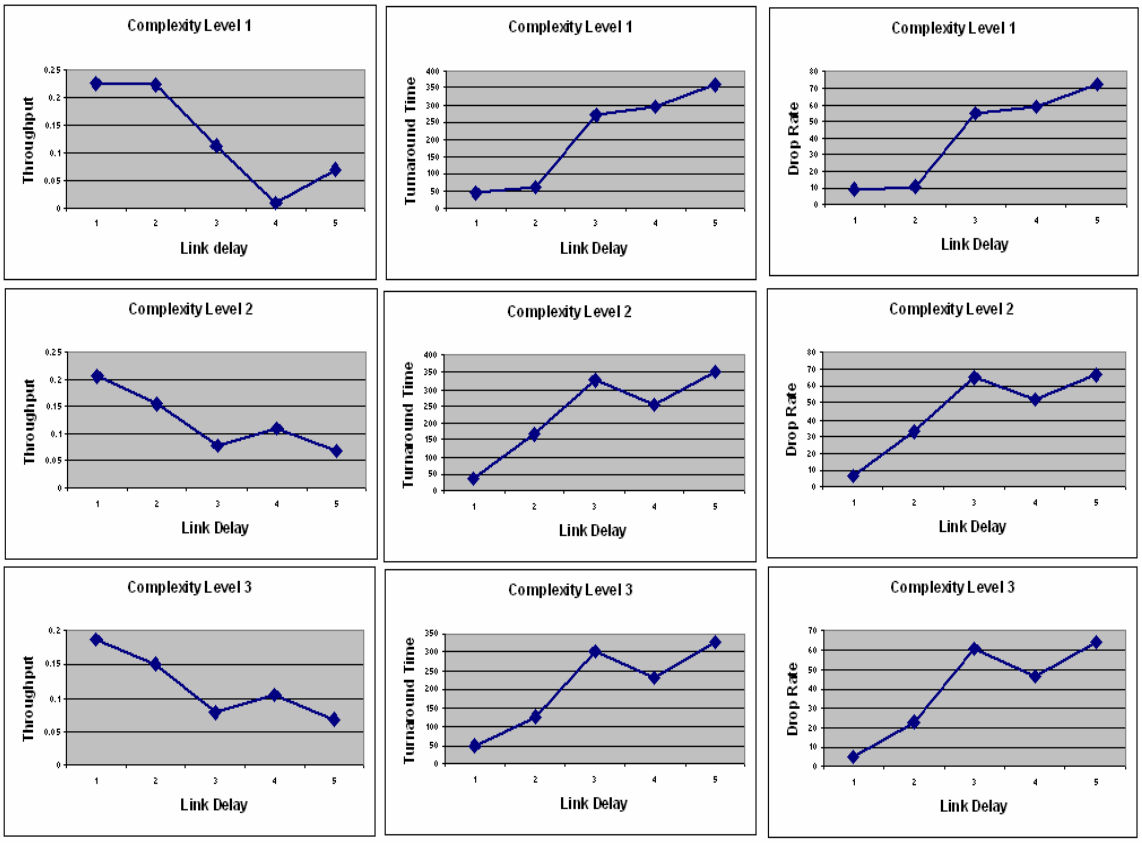

Figure 6.5 Sensitivity Analyses for Variation of Complexity for Anticipatory Technique

From the graphs above, we interpret the effect of complexity on each of the performance metrics for the fault management techniques.

a. Throughput

It can be observed that the complexity has no observable effect on the throughput in case of the reactive technique. The throughput is observed to decline with respect to the increase in link delay of the network and is seen to be constant for increase in link delay from level 3 to level 4. For the alarm correlation technique, variation of throughput is observed to be almost linearly dependent on the link delay. At complexity level 2, the throughput is observed to be constant for level 2

and 3 of link delay. In case of the anticipatory technique, at low complexity level, it can be observed that throughput is not sensitive to link delay at the initial levels, after which it varies linearly with link delay. The throughput is again observed to be independent at very high levels of link delay. For moderate and high levels of complexity, the throughput is observed to be sensitive to link delay at low and moderate levels of link delay. At high levels of link delay, the throughput is less sensitive to increase in link delay.

b. Turnaround Time

For the reactive technique, the turnaround time is less sensitive to link delay at very low and very high levels of link delay. This trend is observed for all levels of complexity for the reactive technique. For the alarm correlation technique, the turnaround time is observed to be highly dependent on link delay for level 1 and 2 of complexity. At level 3 of complexity, the turnaround time is seen to be less sensitive at higher values of link delay. For the anticipatory technique, the turnaround time is less sensitive to very low and very high levels of link delay for low complexity. The turnaround time is seen to be more sensitive to link delay with increase in complexity levels, except for level 3 and level 4 of link delay, where the turnaround time is seen to depreciate to some extent.

c. Drop Rate of Packets

The drop rate of packet is observed to follow a very similar trend as the variation in turnaround time as described in the above section.

6.2.2 Results with Adaptive Control using Annealer

The experimental design is kept the same with addition of the annealer component. We perform 30 replications for each of the fault management technique with network adaptation. The link delay and complexity values are fixed to “Low” and “Level 1” respectively. We then compare the final values of throughput, turn around time and drop rate for each of the replication with enabling and disabling network adaptation. Figure 6.6 and 6.7 show the graphs obtained for the reactive, alarm correlation, and anticipatory technique.

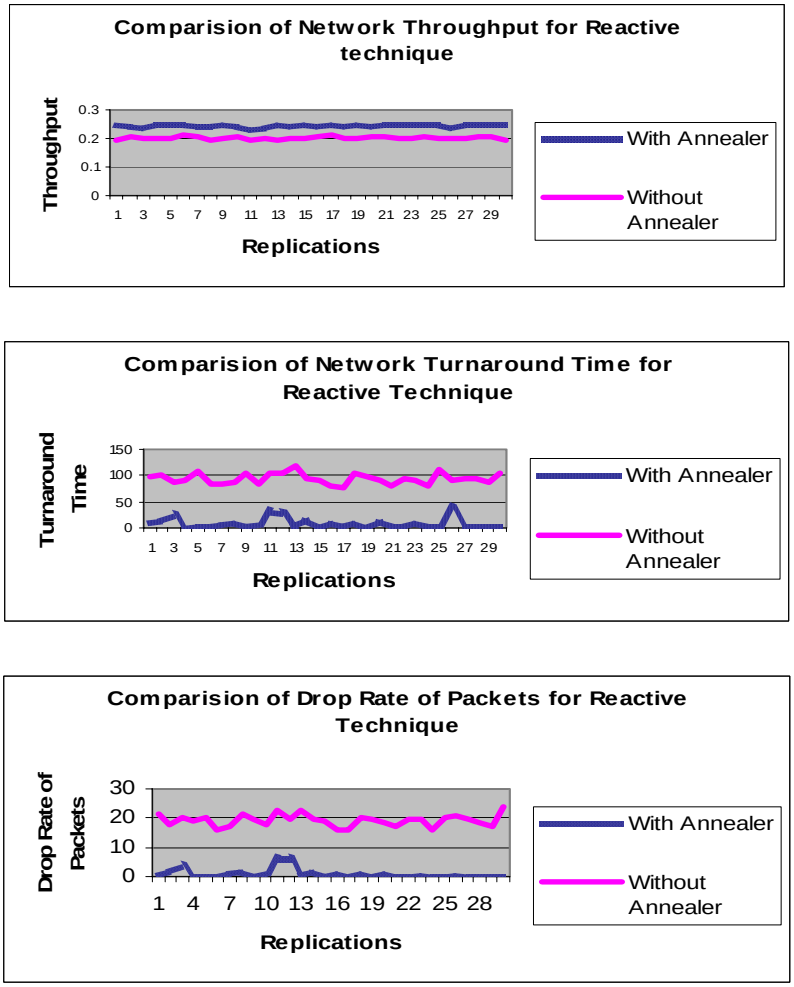

Figure 6.6 Performance Metrics with and without Annealer for Reactive Technique

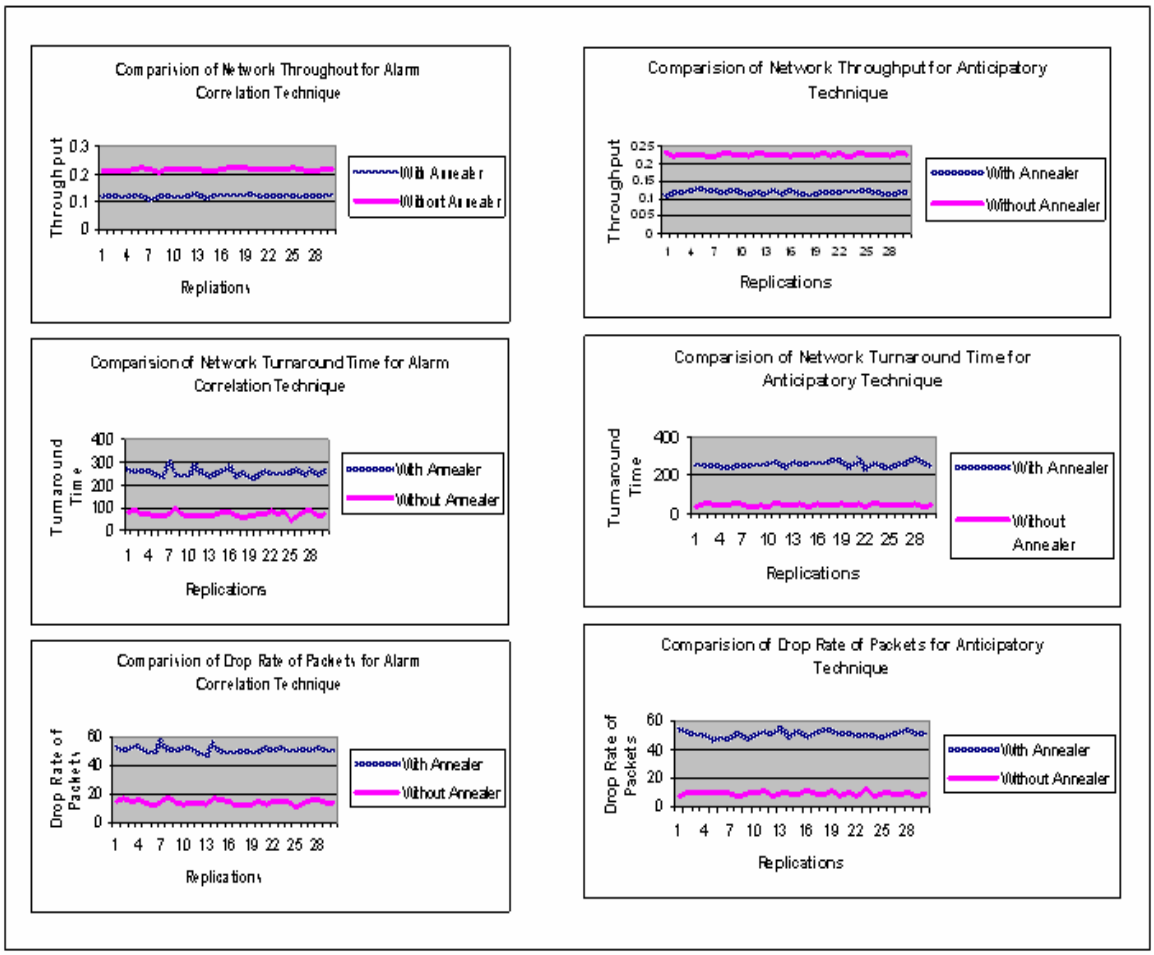

Figure 6.7 Performance Metrics - Alarm Correlation and Anticipatory Technique

From Figure 6.6, we see that the reactive technique performs exceptionally better with network adaptation in terms of performance metrics. In contrast, the performance metrics obtained for the alarm correlation and anticipatory technique are not satisfactory (Figure 6.7). This can be explained by the fact that the adaptation works independently of the recorded behavior of faults in case of alarm correlation technique and independently of the predictive model (the Naïve Bayesian classifier) in case of anticipatory technique. This leads to a high number of control packet generations due to lack of communication between the control agent and the annealer. This is consequently responsible for the

degradation of network performance for the alarm correlation and the anticipatory technique. For example, the naïve Bayesian classifier triggers the control agent based on the evidence it collects. This piece of evidence may not be correct once the network is reconfigured by the annealer.

Chapter 7

Conclusions

Network fault management is a crucial area in the field of computer networks. The goal of fault management is to detect, log, notify users of, and (to the extent possible) automatically fix network problems to keep the network running effectively. Because faults can cause downtime or unacceptable network degradation, fault management is perhaps the most widely implemented element of the ISO network management elements. Our approach towards anticipatory fault management provides a novel methodology of applying agent based behavioral anticipation towards effective fault management. The comparative analyses presented in the previous chapter describe the effectiveness of our technique with respect to reactive techniques. It can be observed that the anticipatory technique performs significantly better compared to the reactive and alarm correlation techniques with respect to network throughput, turnaround time, and the drop rate of packets. The results obtained from the network adaptation methodology shows that the annealer technique performs exceptionally well for the reactive technique as compared to the alarm correlation and anticipatory technique.

7.1 The Limitations of Anticipatory Fault Management

In our approach, the topology of the network is assumed to be fixed and static. The Naïve Bayesian approach will cease to work if the topology is changed (dynamic) and hence cannot be applied to Ad Hoc networks. Furthermore, network adaptation cannot be effectively carried out for anticipatory technique due to lack of communication

between the control agent and the annealer. To get around this problem, an additional component must be incorporated in the architecture to act as an interface between the annealer and the control agent.

Another limitation to our approach is the level of network detail being considered. For instance, we do not consider any hardware details like faults occurring in the network components due to hardware failure or physical wear and tear. One more problem with the adaptation technique is the selection of operating regimes that are to be modified. The operating regimes are decided based on the DEVS toolkit; hence these regimes do not include possible adaptation parameters such as addition, deletion, or modification of component functionality.

7.2 Future Work

Given the limitations from the previous section, the improvement on the anticipatory technique for network fault management requires additional efforts and better tools that would capture additional details about network operation and network components. Furthermore, a framework needs to be devised that would enable anticipatory fault management in Ad Hoc networks, whereby the network topology needs to be modified dynamically. A methodology needs to be invented to undertake hardware implementation of the design. This could be done by embedding the Anticipatory control in an Integrated Circuit (IC) and embedding the fabricated IC in a real time network. Another important issue be taken care is the improvement of the network adaptation framework. The Naïve Bayesian classifier should be designed with additional logic to consider the decisions taken by the annealer. The same can be also achieved by having an additional component that would act as an interface between the control agent and the

annealer. The additional efforts described above would result in a high degree of improvement in network fault management.

The methodology of agent based behavioral anticipation towards fault management in networks can be effectively deployed in the present computer industry and will effectively contribute towards reducing losses incurred due to network faults. Making improvements on our technique as mentioned above will provide a versatile framework for effective fault management in computer networks.

REFERENCES

A. Bouloutas, G. Hart, and M. Schwartz, "On the design of observers for failure detection of discrete event systems," in *Network Management and Control*. New York: Plenum, 1990.

A. Lazar, W. Wang, and R. Deng, "Models and algorithms for network fault detection and identification: A review," in *Proc. IEEE Int. Contr. Conf.*, 1992.

B. Ekdahl, E. Astor and P. Davidson, "Towards Anticipatory Agents", Intelligent Agents: ECAI-94 Workshop on Agent theories, architectures and languages. pp 191-202. 1994.

B. Zeigler and H. Sarjaughian, "Introduction to DEVS modeling and Simulation with JAVATM: Developing component-based Simulation Models", Arizona State University, August 2003.

CAIDA. Cooperative association for internet data analysis. [Online]. Available: http://www/caida.org/Tools.

Carley K. and Svoboda D.," Modeling Organizational Adaptation as a Simulated Annealing Process", Sociological methods & research, Vol. 25 No. 1, August 1996.

Corn, P.A., Dube, R., McMichael, A.F., & Tsay, J.L. (1988), "An autonomous distributed expert system for switched network maintenance". In *Proceedings of IEEE GLOBECOM'88*(pp.1530-1537).

C. Hood and Chaunyi Ji, “Intelligent agents for proactive fault detection”, IEEE, March 1998.

Davidsson P, “A Framework for Preventive State Anticipation”, M.Butz et al. (Eds.): Anticipatory behavior in adaptive learning systems, LNAI 2684, pp. 151-166, 2003.

Edidiong Uyai Ekaette and Behrouz Homayoun Far, “A framework for distributed fault management using intelligent software agents”, CCECE 2003 – CGGEI 2003, IEEE, 2003.

F. Feather and R. Maxion, “Fault detection in an ethernet network using anomaly signature matching,” in *Proc. ACM SIGCOMM*, vol. 23, San Francisco, CA, Sept. 1993, pp. 279–288.

Frank E. Feather. “Fault Detection in an Ethernet Network via Anomaly Detectors”. PhD thesis, Department of Electrical and Computer Engineering, Carnegie MelIon University, 1992.

G. Jakobson and M. D.Weissman, “Alarm correlation,” *IEEE Network*, vol. 7, pp. 52–59, Nov. 1993.

H. Wang, D. Zhang, and K. G. Shin, “Detecting syn flooding attacks,” in *Proc. IEEE INFOCOM*, 2002.

I. Katzela and M. Schwarz, “Schemes for fault identification in communication networks”, IEEE/ACM Trans. Networking, Vol. 3, pp. 753-764, Dec. 1995.

I. Rouvellou and G. Hart, “Automatic alarm correlation for fault identification,” in *Proc. IEEE INFOCOM*, Boston, MA, Apr. 1995, pp.553–561.

J. Agre, “A message-based fault diagnosis procedure,” Proceedings of the ACM SIGCOMM conference on Communications architectures & protocols, Vol 6 Issue 3, Aug 1986.

Joseph, C., Kindrick, J. Muralidhar, K. So, C. & Toth-Fejel, T. (1989) “MAP fault management expert system”. In Meandzija, B.& Westcott, J. (Eds.) *Integrated Network Management, I.*North-Holland: Elsevier Science Publishers B.V.

J. F. Huard and A. A. Lazar, “*Fault Isolation Based on Decision-Theoretic Troubleshooting”*, Technical Report 442-96-08, Center for Telecommunications Research, Columbia University, New York (1996).

J. Pearl, “Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference”, Morgan Kaufmann, San Mateo, Calif., 1988.

Hood C. and Ji C., “Intelligent agents for proactive fault detection”, IEEE Information Communication Conference (Infocom 97). 1997.

K. Carley and D. Svoboda, “Modeling Organizational Adaptation as a Simulated Annealing Process”, Sociological methods and research, Vol. 25 No. 1, August 1996.

L. Lewis, “A case based reasoning approach to the management of faults in communication networks,” in *Proc. IEEE INFOCOM*, vol. 3, San Francisco, CA, Mar. 1993, pp. 1422–1429.

L. Lewis and G. Dreo, “Extending trouble ticket systems to fault diagnosis”, *IEEE Network*, vol. 7, pp. 44-51, Nov 1993. Kandel A, “*Fuzzy Expert Systems.”* CRC press, 1991.

Luger G. F. and Stubblefield W. A., “Artificial Intelligence: Structures and Strategies for Complex Problem Solving”, The Benjamin/Cummings Publishing Company, Inc. 1989

M. Brodie, I. Rish and S. Ma. “Intelligent probing: A cost –effective approach to fault diagnosis in computer networks”, IBM systems journal, Vol 41, No.3, 2002.

M. Thottan and Chuanyi Ji, “Anomaly detection in IP Networks”, IEEE Transactions on signal processing, vol.51, No.8, August 2003.

M. Thottan, “Fault detection in ip networks,” Ph.D. dissertation, Rensselaer Polytech. Inst., Troy, NY, 2000. Under patent with RPI.

M. Butz, O. Siguad and P. Gerard, “Internal Models and Anticipations in Adaptive Learning Systems”, Anticipatory behavior in adaptive learning systems, LNAI 2684, pp. 86-109, 2003.

Pat Langley and Herbert A. Simon. (1995) “Applications of Machine Learning”. Communications of the ACM. Vol.38. No.11

Prietula M., Carley K., and Gasser L., “Simulating organizations: computational models of institutions and groups”, Menlo Park, CA: AAAI Press/MIT Press, 1998.

R. Herdman, “Information security and privacy in network environments”, The Office of Technology Assessment (OTA), September 15, 1994.

Roy Maxion. “Unanticipated Behavior as a Cue for System-Level Diagnosis”, In 8th International Pheonix Conference on Computers and Communications, IEEE, March, 1989.

Roy A. Maxion. “Anomaly Detection for Diagnosis”. In Twentieth International Symposium on Fault-Tolerant Compufing. IEEE, March, 1990.

Rosen. R, “Anticipatory Systems – Philosophical, Mathematical and Methodological Foundations”. Pergamon Press, New York.

R.A. Maxion and F.E. Feather. “A Case Study of Ethernet Anomalies in a Distributed Computing Environment”. IEEE Transactions on Reliability 39(4),433-443, 1990.

T Oates, “Fault identification in Computer Networks: A review and a New Approach”, CS-TR 95-113. 1995.

T. D. Ndousse and T. Okuda, “Computational intelligence for distributed fault management in networks using fuzzy cognitive maps,” in *Proc. IEEE ICC*, Dallas, TX, Jun. 1996, pp. 1558–1562.

Thottan M. and Ji C., “Anomaly detection in IP Networks”, IEEE Transactions on signal processing, vol. 51, No. 8, August 2003.

Wright, J.R., Zielinski, J.E. & Horton, E.M. (1988) “Expert systems development: the ACE system”. In Liebowitz, J. (Ed.) *Expert System Applications to Telecommunications.*New York: John Wiley & Sons.

Yamahira, T., Kiriha, Y. & Sakata, S. (1989) “Unified fault management scheme for network troubleshooting expert system”. In Meandzija, B. & Westcott, J. (Eds.) *Integrated Network Management,* I.North-Holland: Elsevier Science Publishers B.V.

Y. Yemini, “A Critical Survey of Network Management Protocol Standards,” in *Telecommunications Network Management into the 21st Century*, S. Aidarous and T. Plevyak, eds, IEEE Press, Piscataway, N.J.1994.

Zeigler B.P., “Object –Oriented Simulation with Hierarchical. Modular Models – Intelligent Agents and Endomorphic Systems”, Academic Press, 1990.

Zeigler B. and Sarjoughian H. “Introduction to DEVS Modeling & Simulation with JAVA™ : Developing Component-based Simulation Models”, August 2003.

APPENDICES

Appendix A

Design Class Diagram

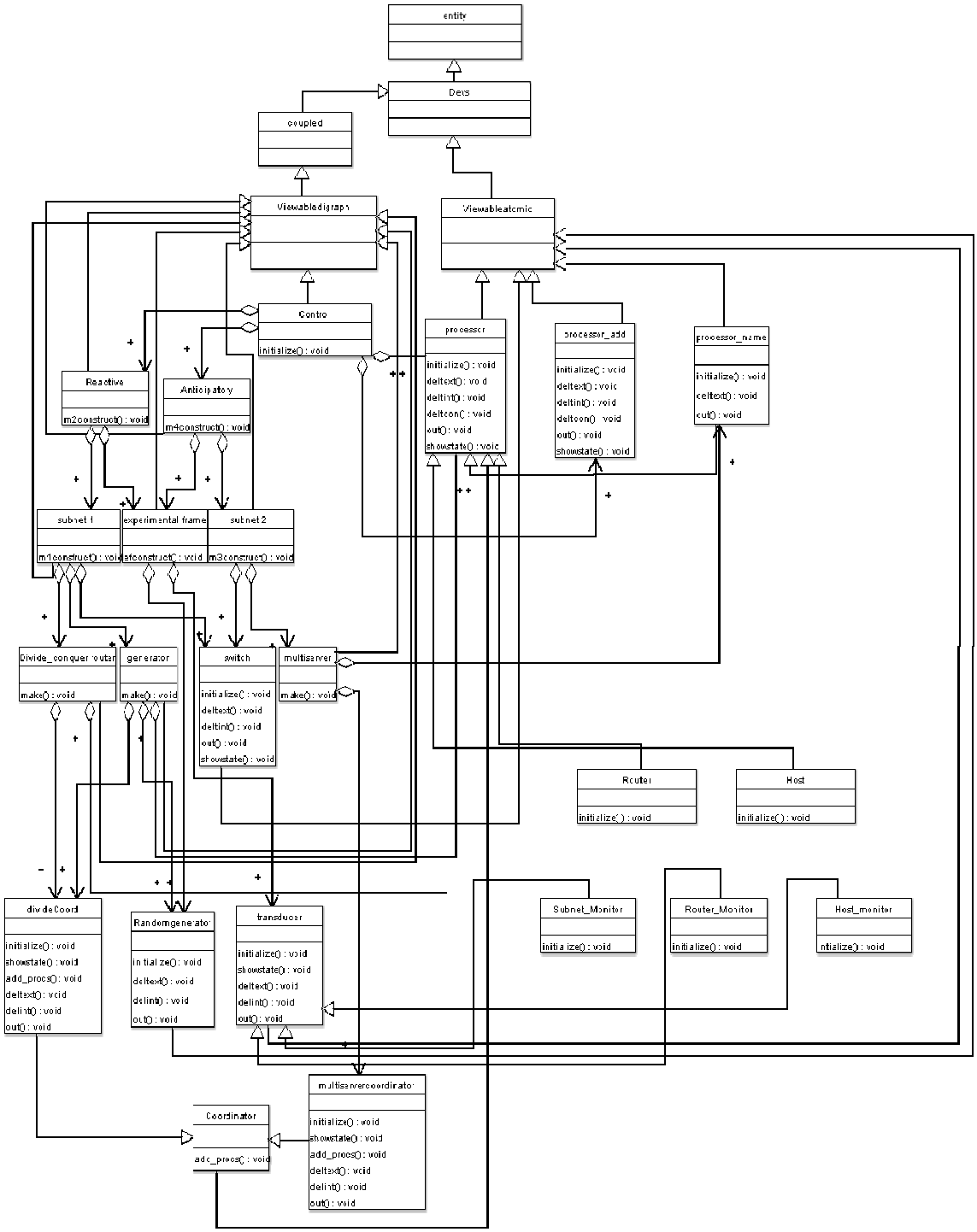

Figure A.1 Design Class Diagram of the DEVS Network Model

Appendix B

Calculation of Confidence Intervals for the T test

A two sided 100(1 - α)% C.I (Confidence Interval) for comparison of means $\theta_1 - \theta_2$ is given by:

$$\overline{Y}_{.1} - \overline{Y}_{.2} \pm t_{\frac{\alpha}{2},\upsilon}\, s.e.(\overline{Y_{.1}} - \overline{Y_{.2}})$$

where,

$$s.e.(\overline{Y_{.1}} - \overline{Y_{.2}}) = S_p \sqrt{\frac{1}{R_1} + \frac{1}{R_2}}$$

R_1, R_2 are the number of replications and

$$S_p{}^2 = \frac{(R_1 - 1)S_1{}^2 + (R_2 - 1)S_2{}^2}{R_1 + R_2 - 2}$$

where, $S_p{}^2$ is an unbiased estimator of the variance $\sigma_i{}^2$ and $\upsilon = R_1 + R_2 - 2$ degrees of freedom. We perform the t test at 95% confidence interval. Tables B.1, B.2 and B.3 shows the tabulated calculations for Confidence Intervals

.

B.1 Calculation of C.I for Reactive vs Alarm Correlation

Technique

	S_1	S_2	*s.e*	*C.I*
Throughput	0.001675	0.002685	0.00538	(-0.001, 0.019)
Turnaround Time	11504.17	17237.6	13.84	(-38.69, 15.55)
Drop Rate	471.63	607.11	2.68	(-8.54, 1.96)

B.2 Calculation of C.I for Reactive vs Anticipatory Technique

	S_1	S_2	*s.e*	*C.I*
Throughput	0.001675	0.001813	0.004822	(-0.025, -0.006)
Turnaround Time	11504.17	8359.43	11.5	(25.11, 70.19)
Drop Rate	471.63	379.17	2.38	(5.67, 9.59)

B.3 Calculation of C.I for Alarm Correlation vs Anticipatory Technique

	S_1	S_2	*s.e*	*C.I*
Throughput	0.002685	0.001813	0.00547	(-0.035, -0.014)
Turnaround Time	8359.43	17237.6	13.06	(33.62, 84.81)
Drop Rate	379.17	607.91	2.56	(5.9, 15.93)

Appendix C

Sample Data Sets

For each experiment, we perform 30 replications with certain levels of complexity and link delay. The data sets obtained for the performance metrics are as follows:

Link Delay: Low Complexity: Level 1									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.1967	96.75	21.28	0.21184	77.45	15.26	0.2319	37.08	7.22
2	0.2052	100.32	17.67	0.20783	90.52	16.86	0.2227	47.8	10.84
3	0.1997	87.74	20.08	0.21184	71	15.26	0.2259	50.17	9.63
4	0.2022	90.46	18.87	0.21084	69.19	15.66	0.2257	42.87	9.63
5	0.19919	109.17	20.08	0.2158	65.29	13.65	0.2247	41.92	10.04
6	0.2102	85.05	15.66	0.2198	63.45	12.04	0.2228	55.77	10.84
7	0.2068	83.23	17.26	0.2128	67.36	14.85	0.2224	41.86	10.04
8	0.1961	86.04	21.28	0.2058	97.35	17.67	0.2315	35.26	7.22
9	0.2002	103.89	19.67	0.2158	73.13	13.65	0.226	39.99	9.6
10	0.2052	84.86	17.67	0.2178	66.64	12.85	0.2267	37.46	9.23
11	0.1931	106.27	22.48	0.2168	60.42	13.25	0.221	59.54	11.6
12	0.2002	105.19	19.67	0.2168	65.4	13.25	0.2305	41.88	7.63
13	0.1931	118.02	22.4	0.2178	61.32	12.85	0.2264	45.48	9.23
14	0.2002	94.38	19.67	0.2088	68.52	16.46	0.2255	46.1	9.63
15	0.2018	90.62	19.27	0.2098	85.2	16.06	0.2277	31.14	8.83
16	0.2092	81.49	16.06	0.2138	74.3	14.45	0.2204	51.51	11.64
17	0.2102	78.17	15.66	0.2218	60.41	11.24	0.227	43.4	9.2
18	0.1991	104.35	20.08	0.2188	59.33	12.44	0.2275	44.38	8.83
19	0.2008	96.78	19.67	0.2188	65.07	12.44	0.2208	52.6	11.64
20	0.2032	89.37	18.47	0.2138	70.87	14.45	0.2295	39.26	8.03
21	0.2078	81.77	16.86	0.2178	66.95	12.85	0.2228	50.56	10.84
22	0.2002	94.92	19.67	0.2128	83.75	14.85	0.2309	34.14	7.63
23	0.2002	90.9	19.67	0.2138	71.99	14.45	0.2168	61.57	13.25
24	0.2092	80.6	16.06	0.2128	81.87	14.85	0.2299	41.45	8.03
25	0.1981	110.4	20.48	0.2228	44.57	10.84	0.226	39.55	9.6
26	0.1977	90.38	20.88	0.2158	61.53	13.65	0.2265	45.72	9.23
27	0.2002	93.19	19.67	0.2098	83.1	16.06	0.2275	43.27	8.83
28	0.2038	93.43	18.47	0.2098	87.67	16.06	0.2238	51.4	10.44
29	0.2072	88.07	16.86	0.2158	64.42	13.65	0.2295	36.87	8.03
30	0.1907	104.28	23.69	0.2168	68.94	13.25	0.2264	41.47	9.23

Figure C.1 Performance Metrics for Low Link Delay and Level 1 Complexity

Link Delay: Moderate Complexity: Level 1									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.1803	160.62	27.71	0.1501	207.69	39.75	0.2224	63.48	10.84
2	0.1853	146.04	25.7	0.1445	220.03	42.16	0.2274	46.37	8.83
3	0.1813	149.43	27.3	0.1481	221.9	40.5	0.2206	74.39	11.64
4	0.1843	145.33	26.1	0.1401	233.13	43.77	0.2216	62.25	11.24
5	0.1813	157.94	27.3	0.1471	233.22	40.96	0.2188	69.96	12.44
6	0.1783	152.07	28.51	0.1415	231.84	43.37	0.2256	63.22	9.63
7	0.1863	134.7	25.3	0.1411	232.59	43.37	0.2166	71.4	13.25
8	0.1843	136.17	26.1	0.1516	218.4	39.35	0.2248	58.22	10.04
9	0.1803	155.51	27.7	0.1441	236.84	42.16	0.2208	61.02	11.64
10	0.1873	139.09	24.89	0.1415	245.14	43.37	0.2246	58.17	10.04
11	0.1783	154.75	28.51	0.1381	257.98	44.57	0.2266	58.4	9.23
12	0.1773	166.11	28.91	0.1485	210.27	40.56	0.2164	74.06	13.25
13	0.1773	155.61	28.91	0.1435	225.01	42.57	0.2276	56.09	8.83
14	0.1903	126.11	23.69	0.1481	236.08	40.56	0.2186	64.47	12.44
15	0.1903	131.75	23.69	0.1441	232.57	42.16	0.2256	50.22	9.63
16	0.1863	137.24	25.3	0.1391	218.44	44.17	0.2216	59.38	11.24
17	0.1883	128.03	24.49	0.1411	236.51	43.37	0.2214	63.12	11.24
18	0.1813	150.76	27.3	0.1425	239.84	42.97	0.2218	55.45	11.24
19	0.1833	135.21	26.5	0.1415	229.98	43.37	0.2228	63.94	10.84
20	0.1723	160.87	30.92	0.1361	250.25	45.38	0.2196	62.83	12.04
21	0.1833	138.52	26.5	0.1391	235.69	44.17	0.2196	68.5	12.04
22	0.1823	145.94	26.9	0.1475	221.27	40.96	0.2206	55.57	11.64
23	0.1793	152.63	28.11	0.1461	207.03	41.36	0.2228	62.82	10.84
24	0.1823	138.25	26.9	0.1506	197.15	39.75	0.2224	60.4	10.84
25	0.1783	162.89	28.51	0.1471	209.57	40.96	0.2258	50.52	9.67
26	0.1883	130.84	24.49	0.1455	211.54	41.76	0.2238	57.74	10.44
27	0.1933	124.68	22.48	0.1451	216.15	41.76	0.2224	61.58	10.84
28	0.1773	158.42	28.91	0.1411	235.3	43.3	0.2238	59.96	10.44
29	0.1853	142.59	25.7	0.1391	227.97	44.17	0.2224	65.54	10.84
30	0.1863	138.33	25.3	0.1451	220.62	41.76	0.2218	65.41	11.24

Figure C.2 Performance Metrics Moderate Link Delay and Level 1 Complexity

Link Delay: High Complexity: Level 1									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.0893	349.25	64.25	0.0773	358.46	69.07	0.1154	273.75	53.81
2	0.0983	319.09	60.64	0.0752	366.62	69.87	0.1104	284.05	55.82
3	0.0913	336.12	63.45	0.0722	648.23	71.08	0.1104	264.8	55.82
4	0.0893	348.3	64.25	0.0742	353.35	70.28	0.1124	264.18	55.02
5	0.0963	326.31	61.44	0.0732	368.32	70.68	0.1084	275.08	56.62
6	0.0903	249	63.85	0.0732	362.57	70.68	0.1124	266.96	55.02
7	0.0903	249	63.85	0.0712	374.85	71.48	0.1084	269.3	56.62
8	0.0953	336.61	61.84	0.0763	355	69.47	0.1134	272.32	56.61
9	0.0933	354.61	62.65	0.0812	338.54	67.46	0.1124	264.38	55.02
10	0.0933	341.4	62.65	0.0732	356.78	70.68	0.1094	281.14	56.22
11	0.1014	313.4	59.43	0.0753	347.38	69.87	0.1105	269.94	55.64
12	0.0903	355.36	63.85	0.0753	371.6	69.87	0.1124	267.95	55.02
13	0.0913	354.32	63.45	0.0742	351.14	70.28	0.1074	272.87	57.02
14	0.0913	356.06	63.45	0.0722	351.48	71.08	0.1084	278.1	56.62
15	0.0883	357.88	64.65	0.0803	353.38	67.87	0.1134	273.43	54.61
16	0.0933	325.25	62.65	0.0753	351.06	69.87	0.1114	275.33	55.42
17	0.0883	356.7	64.65	0.0742	360.75	70.28	0.1124	272.52	55.02
18	0.0933	351.26	62.65	0.0772	350.62	69.07	0.1084	261.23	56.62
19	0.0913	354.76	63.45	0.0735	371.53	70.56	0.1104	276.86	55.82
20	0.0943	345.82	62.24	0.0742	375.94	70.28	0.1124	268.55	55.02
21	0.0943	331.8	62.24	0.0773	349.7	69.07	0.1134	267.37	54.61
22	0.0953	335.55	61.84	0.0692	381.36	72.28	0.1104	280.63	55.82
23	0.0993	330.36	90.24	0.0793	355.05	68.27	0.1104	285.56	55.82
24	0.0923	347.81	93.05	0.0763	351.35	69.47	0.1104	270.72	55.82
25	0.0943	337.95	62.24	0.0773	339.53	69.07	0.1074	274.39	57.02
26	0.0973	330.98	61.04	0.0753	378.02	69.87	0.1094	269.44	56.22
27	0.0893	355.74	64.25	0.0722	373.75	71.08	0.1114	270.22	55.42
28	0.0883	351.42	64.65	0.0732	347.64	70.68	0.1084	270.82	56.62
29	0.0963	321.13	61.44	0.0773	359.14	69.07	0.1114	274.29	55.42
30	0.0953	332.84	61.84	0.0753	372.41	69.87	0.1084	273.07	56.62

Figure C.3 Performance Metrics for High Link Delay and Level 1 Complexity

Link Delay: Low Complexity: Level 2									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.19879	49.32	12.44	0.2118	39.04	8.83	0.2038	41.85	7.63
2	0.1981	68.27	11.64	0.2028	55.21	9.23	0.2136	36.06	6.42
3	0.2058	53.02	11.24	0.2058	49.87	9.23	0.2136	30.77	3.61
4	0.1987	45.46	10.44	0.2138	45.24	8.03	0.2138	23.48	5.22
5	0.1921	62.5	13.25	0.2038	32.39	6.82	0.2006	44.07	8.83
6	0.2018	56.97	12.44	0.1987	61.67	10.84	0.2038	32.18	6.82
7	0.2058	55.75	11.64	0.2108	51.44	9.23	0.2056	28.71	5.22
8	0.1991	49.3	10.84	0.2058	54.08	11.64	0.2066	24.22	5.62
9	0.2058	61.75	12.85	0.2068	40.73	7.63	0.2006	57.68	9.23
10	0.2114	31.02	5.22	0.2108	36.69	5.62	0.2068	32.19	5.22
11	0.1987	51.51	12.44	0.2078	34.16	7.63	0.2046	37.32	6.82
12	0.2032	48.69	9.63	0.2078	41.24	9.63	0.2086	25.81	3.61
13	0.2108	27.44	5.22	0.2078	55.61	11.24	0.2056	37.01	9.23
14	0.2048	46.97	8.43	0.1987	46.87	10.44	0.2088	24.93	6.02
15	0.2022	50.24	10.04	0.2038	52.43	10.44	0.2096	26.61	4.41
16	0.2048	42.29	8.83	0.1957	75.94	12.44	0.2028	33.68	6.42
17	0.1987	58.48	13.25	0.2078	43.8	9.23	0.2048	45.72	8.83
18	0.1967	69.82	11.64	0.2048	47.16	8.03	0.2046	67.55	6.82
19	0.2132	24.62	6.42	0.2048	58.27	11.24	0.2126	57.08	0.4
20	0.1997	62.31	11.64	0.2078	51.05	8.43	0.2178	47.08	1.6
21	0.2012	47.19	11.24	0.2088	39.47	6.82	0.2118	19.07	4.01
22	0.2092	45.29	8.83	0.1967	55.26	12.04	0.2126	22.99	3.61
23	0.1987	66.33	11.24	0.2048	33.8	9.23	0.2068	50.89	5.22
24	0.2082	39.66	6.42	0.2078	55.56	7.22	0.2046	27.33	4.81
25	0.2028	54.54	12.04	0.1937	72.92	14.45	0.2028	39.44	7.63
26	0.2028	56.8	9.63	0.2048	52.51	10.04	0.2098	32.78	5.62
27	0.2022	60.29	11.64	0.2078	49.39	10.44	0.2136	15.96	3.21
28	0.2028	49.51	10.04	0.2038	63.86	12.85	0.2006	30.88	6.02
29	0.1987	64.03	12.04	0.2098	55.46	10.84	0.2076	22.72	5.22
30	0.2012	36.24	7.22	0.2068	51.82	8.83	0.2098	26.57	6.82

Figure C.4 Performance Metrics for Low Link Delay and Level 2 Complexity

Link Delay: Moderate Complexity: Level 2									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.1773	125.25	24.09	0.1291	248.67	43.77	0.1584	173.1	32.53
2	0.1763	112.54	20.08	0.1431	203.47	37.34	0.1472	186.41	36.54
3	0.1823	105.03	15.66	0.1401	210.23	38.95	0.1465	179.77	36.14
4	0.1843	110.74	17.67	0.1371	197.54	38.95	0.1526	165.21	32.93
5	0.1843	115	20.48	0.1405	200.51	37.34	0.1576	160.12	31.32
6	0.1803	98.98	18.47	0.1411	197.04	38.15	0.1536	157.83	32.12
7	0.1873	97.65	15.66	0.1465	192.31	36.14	0.1633	157.81	29.31
8	0.1803	105.56	17.26	0.1435	175.85	35.34	0.1563	170.12	32.53
9	0.1823	105.4	20.48	0.1385	204.55	37.34	0.1556	162.15	30.12
10	0.1863	79.38	13.65	0.1445	184.08	35.34	0.1543	169.12	32.53
11	0.1833	126.61	19.27	0.1405	185.7	38.55	0.1462	173.76	34.53
12	0.1883	97.67	15.26	0.1381	220.14	37.34	0.1485	188.88	34.93
13	0.1823	89.44	17.26	0.1381	196.25	36.54	0.1556	165.41	33.33
14	0.1823	90.82	17.26	0.1451	189.41	32.93	0.1583	163.09	30.92
15	0.1823	113.76	20.48	0.1411	204.17	37.34	0.1594	163.16	31.32
16	0.1853	97.39	15.26	0.1421	204.04	38.15	0.1553	184.24	34.53
17	0.1803	92.83	18.07	0.1421	796.25	35.74	0.1634	179.83	30.52
18	0.1803	111.99	18.07	0.1371	207.78	37.75	0.1584	166.55	31.32
19	0.1823	108.45	18.47	0.1395	209.55	39.35	0.1564	166.38	30.92
20	0.1813	97.76	16.86	0.1405	175.38	37.34	0.1516	162.13	31.32
21	0.1803	116.79	20.88	0.1435	202.47	38.15	0.1523	168.29	32.93
22	0.1833	105.45	18.87	0.1355	211.58	39.35	0.1536	175.9	34.13
23	0.1813	114.46	20.08	0.1381	195.04	36.94	0.1516	174.45	34.13
24	0.1773	113.84	18.47	0.1385	204.51	40.56	0.1534	172.9	32.53
25	0.1833	93.4	17.26	0.1411	202.22	35.74	0.1523	186.18	33.73
26	0.1823	112.26	16.86	0.1351	225.98	39.75	0.1586	162.93	32.12
27	0.1843	85.69	13.65	0.1461	192.31	35.34	0.1594	176.18	32.53
28	0.1803	105.9	19.27	0.1431	205.55	38.15	0.1444	174.55	34.93
29	0.1823	104.13	17.67	0.1431	193.47	37.75	0.1566	152.28	29.71
30	0.1843	102.09	16.06	0.1465	196.71	36.94	0.1523	178.14	32.12

Figure C.5 Performance Metrics for Moderate Link Delay and Level 2 Complexity

Link Delay: High Complexity: Level 2									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.0963	319.97	57.42	0.0702	336.06	67.46	0.0783	339.5	65.86
2	0.0943	327.63	57.42	0.0712	336.62	68.67	0.0763	337.96	65.86
3	0.0923	335.93	57.42	0.07522	350.69	67.06	0.0773	339.33	66.26
4	0.0943	325.72	59.03	0.0773	342.5	65.86	0.0763	334.35	66.66
5	0.0913	314.96	59.03	0.0702	352.41	67.46	0.0793	313.7	63.45
6	0.0863	349.37	61.44	0.0712	355.37	68.27	0.0773	314.03	65.46
7	0.0923	329.15	58.63	0.0722	364.06	66.66	0.0803	322.13	65.06
8	0.0953	322.2	57.83	0.0692	355.7	69.87	0.0783	320.97	64.65
9	0.0933	318.27	57.42	0.0682	354.57	68.67	0.0786	319.73	64.51
10	0.0963	317.88	57.42	0.0661	371.74	70.28	0.0783	339.51	65.06
11	0.0893	343.77	61.84	0.0712	345.55	67.87	0.0783	321.28	65.06
12	0.0933	341.32	60.24	0.0682	349.28	69.47	0.0753	338.15	67.46
13	0.0923	332.06	59.83	0.0652	381.05	69.07	0.0773	333.46	65.46
14	0.0903	351.98	62.24	0.0722	371.06	67.87	0.0793	320.9	65.06
15	0.0983	318.21	57.42	0.0732	357.03	67.46	0.0793	325.15	63.85
16	0.0883	329.51	61.04	0.0682	333.03	69.07	0.0813	322.41	63.45
17	0.0923	340.48	60.64	0.0692	347.15	69.47	0.0793	343.69	64.25
18	0.0983	269.99	54.21	0.0732	334.17	66.26	0.0773	333.25	64.25
19	0.0943	322.26	56.62	0.0702	348.54	68.27	0.0813	328.71	62.65
20	0.0933	326.71	58.63	0.0712	344.25	67.87	0.0753	329.04	65.46
21	0.0933	319.6	58.23	0.0692	359.92	68.27	0.0773	325.1	65.46
22	0.0913	332.37	59.03	0.0712	347.89	69.07	0.0793	322.24	65.86
23	0.0883	332.19	59.43	0.0692	341.2	69.07	0.0773	341	65.46
24	0.0913	341.54	57.83	0.0661	358.39	69.47	0.0813	315.8	65.06
25	0.0983	316.04	59.03	0.0722	365.72	69.07	0.0793	330.28	65.06
26	0.0943	336.42	58.23	0.0712	338.2	67.87	0.0783	344.79	66.66
27	0.0973	303.24	54.61	0.0672	368.36	70.28	0.0793	328	64.65
28	0.0893	343.51	58.63	0.0782	360.45	66.26	0.0763	333.64	66.26
29	0.0913	314.47	58.23	0.0753	342.26	66.26	0.0763	326.77	65.86
30	0.0923	322.23	59.03	0.0722	361.79	69.07	0.0793	338.05	65.06

Figure C.6 Performance Metrics for High Link Delay and Level 2 Complexity

Link Delay: Low Complexity: Level 3									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.1915	32.94	4.01	0.1837	36.88	6.42	0.1841	31.24	4.41
2	0.2052	39.15	1.2	0.1795	63.15	10.44	0.1841	60.24	7.63
3	0.1833	46.23	6.82	0.1945	34.16	4.41	0.1863	43.65	5.62
4	0.1903	89.25	5.22	0.1873	18.58	2.81	0.1863	45.66	6.42
5	0.1847	42.21	9.63	0.1897	46.48	7.63	0.1901	42.13	4.41
6	0.1881	53.58	6.02	0.1805	46.27	6.42	0.1911	40.09	1.2
7	0.1923	60.88	2.4	0.1867	32.83	6.42	0.1921	30.57	3.61
8	0.1877	17.24	4.81	0.1807	55.14	8.43	0.1831	39.83	6.02
9	0.1857	39.42	6.42	0.1925	56.32	6.82	0.2012	99.84	0.8
10	0.1897	55.08	2	0.1847	33.59	6.42	0.1847	50.91	8.83
11	0.2022	25.68	2.4	0.1957	27.88	5.22	0.1841	31.24	4.41
12	0.1935	64	10.04	0.1895	47.94	8.43	0.1841	60.24	7.63
13	0.1985	135.29	2.81	0.1937	19.89	3.61	0.1863	43.65	5.62
14	0.2008	114.31	0.8	0.2046	93.83	1.2	0.1863	45.66	6.42
15	0.1991	48.04	1.2	0.1857	35.98	6.82	0.1901	42.13	4.41
16	0.1985	61.66	2	0.1897	25.5	4.41	0.1911	40.09	1.2
17	0.1977	32.7	2.81	0.1855	45.05	8.43	0.1921	30.57	3.61
18	0.1981	135.5	2.81	0.1985	75.05	3.61	0.1831	39.83	6.02
19	0.2038	138.99	2	0.1907	29.47	5.22	0.2012	99.84	0.8
20	0.2016	304	0	0.1865	32.94	7.63	0.1847	50.91	8.83
21	0.1957	26.38	4.81	0.1947	31.02	6.02	0.1841	31.24	4.41
22	0.2028	79.61	2.81	0.1937	18.25	3.21	0.1841	60.24	7.63
23	0.1981	43.16	6.02	0.1995	26.65	4.01	0.1863	43.65	5.62
24	0.1917	57.69	7.22	0.1867	38.39	6.82	0.1863	45.66	6.42
25	0.1927	34.44	2.4	0.1877	57.46	4.41	0.1901	42.13	4.41
26	0.2006	140.01	3.21	0.1987	43.12	1.2	0.1911	40.09	1.2
27	0.1941	38.25	4.81	0.1837	33.41	6.02	0.1921	30.57	3.61
28	0.1937	36.4	4.81	0.1917	25.02	3.21	0.1831	39.83	6.02
29	0.1957	39.04	5.22	0.1897	39.8	2	0.2012	99.84	0.8
30	0.1981	187.55	4.41	0.1935	47.49	3.61	0.1847	50.91	8.83

Figure C.7 Performance Metrics for Low Link Delay and Level 3 Complexity

Link Delay: Moderate Complexity: Level 3									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.1733	84.37	14.85	0.1321	182.6	34.53	0.1451	141.98	27.3
2	0.1833	63.8	8.83	0.1371	177.97	35.74	0.1556	110.62	22.08
3	0.1933	47.94	7.22	0.1285	181.05	32.93	0.1566	134.29	24.09
4	0.1817	47.43	7.63	0.1395	167.51	33.33	0.1461	131.11	24.89
5	0.1713	70.12	13.25	0.1345	205.48	36.94	0.1465	130.55	24.49
6	0.1893	36.51	6.42	0.1345	181.38	33.73	0.1501	124.67	20.48
7	0.1867	55.47	7.63	0.1345	205.4	34.53	0.1573	108.09	19.67
8	0.1873	47.13	2.4	0.1375	170.37	33.33	0.1492	140.85	22.48
9	0.1757	83.6	15.66	0.1455	154.39	30.12	0.1503	113.26	21.68
10	0.1787	68.23	10.44	0.1335	171.3	35.74	0.1492	112.6	22.89
11	0.1833	51.64	8.43	0.1315	208.74	37.75	0.1471	129.86	23.69
12	0.1767	91.27	14.85	0.1244	217.29	39.35	0.1475	139.36	26.14
13	0.1673	58.06	12.04	0.1291	192.82	38.15	0.1424	127.19	22.08
14	0.1713	85.22	11.64	0.1361	212.2	37.34	0.1492	140.96	24.49
15	0.1736	60.09	10.04	0.1351	203	36.14	0.1541	109.44	22.89
16	0.1813	81.93	14.45	0.1365	188.78	33.73	0.1516	125.58	23.29
17	0.1783	61.64	10.44	0.1234	211.42	38.55	0.1455	123.38	26.5
18	0.1713	77.27	15.26	0.1325	182.92	36.54	0.1516	137.22	23.29
19	0.1706	104.64	16.86	0.1351	176.08	33.33	0.1495	99.43	20.48
20	0.1783	71.64	11.64	0.1241	241.86	40.96	0.1462	158.47	27.71
21	0.1823	45.05	3.61	0.1361	192.46	34.13	0.1536	143.83	28.51
22	0.1757	117.93	15.66	0.1305	194.44	33.33	0.1452	120.6	24.09
23	0.1817	44.56	10.44	0.1321	197.06	36.14	0.1543	124.76	22.89
24	0.1703	105.17	16.46	0.1271	161.73	32.93	0.1495	111.92	21.28
25	0.1773	68.21	12.44	0.1241	196.42	36.14	0.1543	114.16	19.27
26	0.1743	93.02	14.45	0.1345	180.8	35.34	0.1513	121.16	22.08
27	0.1716	108.76	16.86	0.1301	180.51	34.13	0.1504	125.53	23.69
28	0.1843	58.09	9.23	0.1365	165.63	33.33	0.1543	156.54	24.89
29	0.1787	75.93	10.84	0.1295	194.05	38.15	0.1501	133.83	26.9
30	0.1753	81.78	15.26	0.1295	182.03	36.54	0.1541	115.83	19.67

Figure C.8 Performance Metrics o for Moderate Link Delay and Level 3 Complexity

Link Delay: High Complexity: Level 3									
	Reactive Technique			Alarm Correlation Technique			Anticipatory Technique		
Replication	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate	Throughput	Latency	Drop Rate
1	0.0901	306.9	54.21	0.0702	331.38	66.26	0.0832	280.7	57.83
2	0.0963	302.76	52.61	0.0682	355.11	63.85	0.0782	315.12	60.24
3	0.0901	322.47	55.82	0.0672	362.03	67.06	0.0773	307.31	61.84
4	0.0963	282.07	52.2	0.0722	343.15	66.06	0.0803	306.48	61.04
5	0.0941	303.45	53.01	0.0661	342.97	66.66	0.0722	302.07	61.44
6	0.0893	311.13	54.61	0.0751	358.5	65.46	0.0803	282.89	58.63
7	0.0923	321.83	52.2	0.0691	313.2	65.06	0.0813	278.79	60.64
8	0.0933	300.6	55.82	0.0692	326.23	66.26	0.0823	295.4	61.04
9	0.0921	310.13	53.01	0.0682	372.31	65.86	0.0822	319.05	60.64
10	0.0981	284.69	52.61	0.0753	310.83	65.46	0.0782	323.78	61.04
11	0.0911	344.52	55.82	0.0701	341.8	67.06	0.0783	318.11	60.24
12	0.0903	314.64	54.61	0.0742	312.15	62.65	0.0793	319.67	63.45
13	0.0904	328.7	54.21	0.0742	319.32	65.86	0.0792	285.04	57.83
14	0.0901	305.53	53.81	0.0681	341.39	66.66	0.0752	300.74	61.04
15	0.0861	337.18	57.83	0.0681	347.67	67.46	0.0822	304.5	59.83
16	0.0873	325.41	57.02	0.0662	338.23	67.87	0.0802	289.93	58.63
17	0.0911	283.61	52.61	0.0671	392.32	69.07	0.0773	317.88	60.64
18	0.0903	321.56	57.42	0.0682	339.93	68.27	0.0773	314.34	61.44
19	0.0893	312.7	54.61	0.0732	322.31	63.85	0.0773	290.42	61.04
20	0.0853	312.96	58.63	0.0701	307.95	66.66	0.0753	281.12	61.44
21	0.0843	348.12	57.02	0.0652	332.77	67.87	0.0782	296.11	60.24
22	0.0923	322.35	57.42	0.0712	331.37	65.06	0.0813	294.92	59.83
23	0.0833	354.12	61.04	0.0662	354.48	66.66	0.0782	292.98	59.43
24	0.0923	306.39	54.21	0.0751	338.5	63.45	0.0753	281.09	60.24
25	0.0903	324.42	56.62	0.0632	347.23	69.47	0.0803	290.72	57.83
26	0.0883	321.57	57.02	0.0681	366.05	66.26	0.0812	305.53	59.43
27	0.0821	351.9	57.83	0.0692	333.69	67.06	0.0793	310.88	61.44
28	0.0943	308.54	56.22	0.0721	337.16	65.06	0.0813	310.6	62.24
29	0.0883	348.37	58.63	0.0702	337.61	65.46	0.0802	301.87	59.83
30	0.0853	352.8	56.62	0.0731	340.78	66.66	0.0813	305.36	59.83

Figure C.9 Performance Metrics for High Link Delay and Level 3 Complexity