Stochastic Petri Net Models of Service Availability in a PBNM System for Mobile Ad Hoc Networks

Aniket A. Bhat

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Committee:
Dr. Luiz A. DaSilva, Committee Chair
Dr. Scott F. Midkiff, Committee Member
Dr. Kaustubh S. Phanse, Committee Member
Dr. Srinidhi Varadarajan, Committee Member

July 6, 2004.
Blacksburg, Virginia

Keywords: Service availability, policy-based network management, mobile ad hoc networks, non-Markovian modeling, Stochastic Petri Nets

Stochastic Petri Net Models of Service Availability in a PBNM System for Mobile Ad Hoc Networks

Aniket A. Bhat

(Abstract)

Policy based network management is a promising approach for provisioning and management of quality of service in mobile ad hoc networks. In this thesis, we focus on performance evaluation of this approach in context of the amount of service received by certain nodes called policy execution points (PEPs) or policy clients from certain specialized nodes called the policy decision points (PDPs) or policy servers. We develop analytical models for the study of the system behavior under two scenarios; a simple Markovian scenario where we assume that the random variables associated with system processes follow an exponential distribution and a more complex non-Markovian scenario where we model the system processes according to general distribution functions as observed through simulation. We illustrate that the simplified Markovian model provides a reasonable indication of the trend of the service availability seen by policy clients and highlight the need for an exact analysis of the system without relying on Poisson assumptions for system processes. In the case of the more exact non-Markovian analysis, we show that our model gives a close approximation to the values obtained via empirical methods. Stochastic Petri Nets are used as performance evaluation tools in development and analysis of these system models.

Acknowledgements

I would like to take this opportunity to express my deep appreciation and thanks to all those who helped me through the process of conducting this research. First and foremost, I am truly grateful to Dr. Luiz A. DaSilva for his constant support and encouragement throughout this research journey. He was an incessant motivating force in realizing this thesis. He was extremely patient and always provided me with an “active ear” to all the queries and problems I encountered during the course of this research. His relentless efforts to improve my technical writing skills are commendable. I would also like to thank Dr. Kaustubh Phanse for his instrumental role in driving me to conduct research in this area. He truly laid the foundation for this research by his own doctoral work and always provided a guiding hand through all the time he was at Virginia Tech. Away from Virginia Tech, he has been doing all he can in his ability as a mentor and a committee member to help me through various technical and non-technical difficulties I faced.

One person who has always awed me as a teacher and a researcher is Dr. Midkiff. He has always been a fountain of knowledge and has helped me strengthen my basic knowledge of networking by teaching fundamental courses in Computer networks at Virginia Tech. I am also thankful to him for providing me access to the networking laboratory facilities in Torgersen Hall. I would also like to thank Dr. Thomas Hou for his support and help in providing me with necessary facilities. He has been a tremendous influence as a researcher with the amount of effort and time he puts into his research. His technical acumen and ability to understand a problem are phenomenal. Many thanks to Dr. Jahng Park for helping me in logistic issues regarding equipment in the networking lab. Special thanks to Dr. Srinidhi Varadarajan and all members of the committee for agreeing to be a part of the thesis committee and providing time and co-operation in reviewing this work.

I express my deep gratitude to Marco Scarpa at the University of Catania, Italy and Dr. Telek Miklos at the Budapest University of Technology and Economics for their support and patience in answering my queries regarding the use of WebSPN 3.2 and PhFit tools.

Thanks to Karthik Channakeshava, Gaurav Joshi and Vivek Srivastava for being wonderful friends; they helped me through my research at various levels and provided me with some "social life" in an otherwise technical venture. Last, but not the least, heartfelt thanks to all friends, family members and others whose names do not appear here but have been instrumental to my success during this journey of two years.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile ad hoc networks (MANETs) have received considerable interest in recent times owing to their applicability to a whole gamut of areas like disaster management and military missions. These networks are self-deployable and autonomous in the sense that they do not rely on any central entity or existing infrastructure for communication. MANETs however are characterized by certain constraining factors like limited bandwidth, processing power and operational range. Thus management of such networks becomes a daunting task.

One approach that has shown considerable promise in the area of network provisioning and management is policy based network management (PBNM). Policy based network management has traditionally been associated with wired networks [34, 72]. This idea as applied to wireless networks, and mobile ad-hoc networks in particular, however has been studied only recently [52]. In this thesis, we develop analytical models to characterize service availability seen by nodes in a PBNM system developed for a MANET. Section 1.1 of the chapter outlines the motivation to model wireless networks, with a focus on characterizing service being made available to client nodes by certain specialized nodes called servers. We also discuss some of the contributions of this work in this section. In Section 1.2, we describe the PBNM system we model, while Section 1.3 describes the modeling methodology followed. Section 1.4 discusses the structure of this thesis.

1.1 Motivation

In a MANET, certain nodes (servers) provide specialized services to certain other nodes (clients), in support of clustering, policy distribution and enforcement, and management of security mechanisms. To determine the efficiency of the network with respect to these services, it is important to estimate the availability of such services to the client nodes. For instance, in the case of key management mechanisms in wireless networks, it is important to analyze the availability of certificate issuing authorities for client nodes demanding security credentials from a trusted distribution authority. There is a need to develop analytical models that can be used to study service availability before such mechanisms are deployed. In this thesis, we use stochastic Petri nets (SPNs) (Chapter 2) to model a PBNM system for MANETs and cross-validate the results for service availability obtained with this model against results from QualNet [58] simulations (Chapter 3) and test-bed experiments [53]. The modeling methodology developed here is also applicable to the general problem of service availability in MANETs.

Following are some of the key contributions of this research:

- Development of a model for service availability in a PBNM system for MANETs.
- Non-Markovian modeling of the system, providing an analysis methodology for systems in which Poisson assumptions do not necessarily hold.
- Development of a template for modeling and analysis of service availability in a MANET.

1.2 The PBNM System

In this section, we discuss the key architectural elements of policy based management systems in general and then present the architectural specifics of our PBNM system for MANETs [53, 54]. Figure 1.2.1 [53, Chapter 2] shows the fundamental elements found in a typical policy-based system.

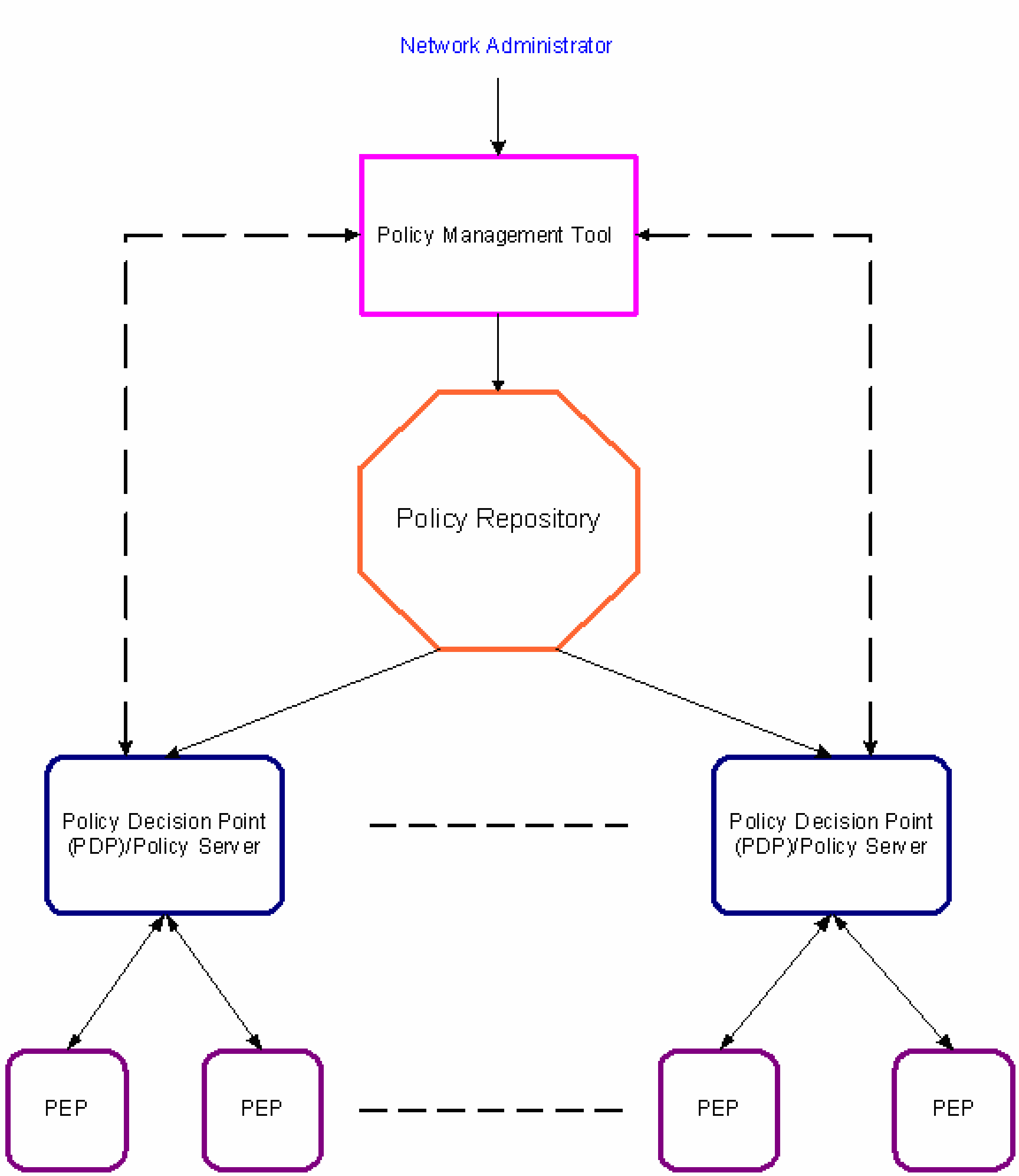

Network Administrator
Policy Management Tool
Policy Repository
Policy Decision Point (PDP)/Policy Server
Policy Decision Point (PDP)/Policy Server
PEP
PEP
PEP
PEP

Figure 1.2.1 Fundamental elements of a policy-based management system

As shown in Figure 1.2.1, the policy management tool (PMT) provides a centralized interface for the network administrators to specify various network policies across all nodes in the system. The policy repository provides a database to store the policy specifications of the network system. These can be group policies or individual policies for the nodes. The policies specified using the PMT are stored in the repository from where they are retrieved by specialized nodes called policy decision points. The policy decision point (PDP), also called the policy server, retrieves the policies from the repository, interprets them and translates them into a format that is compatible to configure one or more policy execution points (PEPs), also known as policy clients. The PDPs must also monitor changes in the policy specification at the PMT or the repository and enforce them at the PEPs. The PEPs are simply policy enforcement points where the policies specified through the PMT are applied. A more detailed description of this architecture can be found in the Internet Engineering Task Force's policy framework working group [55] and in [53].

We now describe the key components that define our policy based network management system for mobile ad hoc networks. Reference [53] identifies seven crucial components for the specification of the PBNM framework for MANETs and provides a comprehensive protocol solution suite to deploy the PBNM system in context of MANETs. In this discussion, we provide an overview of the policy distribution and architecture aspects of the proposed PBNM system. We discuss in detail the controlling and organizing mechanisms that make this protocol solution suite an efficient and an automated control structure.

Policy architecture: This component is fundamental to the framework and defines the type of architecture used to define and implement policies in the PBNM system. The type of policy architecture used to deploy the PBNM system can affect its performance significantly. A taxonomy of policy architectures was suggested in [53] and as a result of analysis of these architectures, a hybrid and distributed approach is suggested. This architecture combines the two distribution models of outsourcing and provisioning and provides an efficient and flexible solution to the policy distribution problem.

Policy distribution: Policy distribution is concerned with the mechanism used to allocate policies to the policy execution points. This involves the choice of the protocol used to apportion appropriate policies to the policy execution points by the policy decision points. An efficient, lightweight and robust mechanism is needed to facilitate this distribution of policies. Common open policy service for provisioning (COPS-PR) [8] is suggested as a policy distribution protocol. It provides a reliable and efficient mechanism to deploy the hybrid architecture mentioned before. A detailed description of the salient features of the COPS-PR protocol is provided in [53, Section 3.2.2].

The PBNM system for MANETs provides an efficient and automated protocol solution suite for network provisioning and management. This enhanced functionality is implemented using three key mechanisms namely, *k*-hop clustering, service discovery and service adaptivity.

- *k*-hop clustering: Cluster management is used to connect a group of nodes to form a virtual network. A set of policy clients are associated with a policy server, forming a cluster. The server node acts as a cluster head and is responsible for policy distribution and implementation of policies at these client nodes. All policy clients within *k*-hops from the server are eligible for service from this server and are a part of this cluster. Figure 1.2.2 shows a dynamic network topology with two servers acting as cluster heads and providing service to their 1-hop neighbor clients. Note that the cluster size in this illustration is limited to 1 hop-count. The cluster size can be changed to increase the coverage of the policy servers.

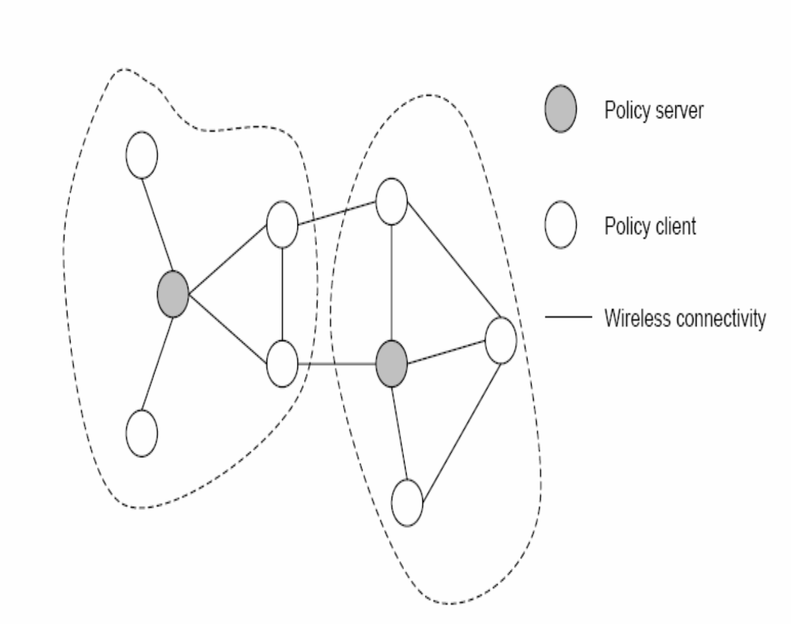


Figure 1.2.2 *k*-hop cluster management algorithm [54]

The advantage of such a clustering algorithm is that it limits the number of wireless hops between policy clients and servers in the PBNM system. This improves performance and works towards optimal utilization of bandwidth and battery life of the intermediate nodes in the network.

In order to implement the k-hop clustering algorithm, [53, and 54] made use of semantics of the COPS-PR protocol. In order to maintain connectivity, a COPS server and a COPS client periodically exchange keep alive (KA) messages. A client connected to the COPS server sends the server a KA message, which is then echoed back to the client by the server. In the *k*-hop clustering algorithm a policy server maintains connections only with clients that are at most k-hops away from it. The *k* value can be advertised by a server in the service advertisement (SA) message broadcasted to the clients or can be set during the connection establishment phase of the COPS protocol. For the clients and servers in the network to perceive connectivity between each other, the client sets the time to live (TTL) field in the KA message to 2*k*. The server, on receiving the KA message from the client, checks the value of TTL field in the KA message. If the value of the TTL field is less than *k*, then the server realizes that the client is more than k-hops away from itself and closes the connection. The KA timer value specified during network configuration decides the periodicity of the KA messages sent by the client.

➢ Service adaptivity: Service adaptivity is concerned with improving service availability by incorporating some intelligence in the protocol used for policy distribution. In a wireless network with typically very few policy servers, all clients in the PBNM system may not receive service from existing servers owing to the limited cluster size. Also, mobility of network nodes reduces the possibility of the clients maintaining their original clusters throughout the life of the network. By including the dynamic service redundancy mechanism (DynaSeR), service availability as seen by policy clients in the PBNM system is improved. This is achieved by two techniques: *redirection* and *delegation*.

Note that the DynaSer mechanism is closely weaved with the clustering mechanism in order to implement the redirection and delegation mechanisms. Redirection is the process in which a policy server guides a client to a new policy server in its vicinity, once the client moves more than *k* hops away from its current server. Once the client's current server detects this, it checks its topology information to determine whether the client has moved within *k* hops of another server. In this case the server redirects the client to the new server.

In the delegation mechanism, whenever a client node, already being served by a policy server, detects one or more clients in need of service, it may volunteer to act as a server for those clients. In the analytical models we develop, we do not consider these service adaptivity mechanisms in order to reduce the complexity to the model.

- Service discovery: Service discovery mechanism is suggested and implemented in order to facilitate automated discovery of policy servers in the PBNM system. This mechanism is implemented using two specialized messages; the client service request (CSRQ) message, which is a broadcast request sent by a client that is currently not in service, and a service advertisement (SA) message, which is also a broadcast message sent by servers in order to advertise their existence. Both these messages are sent periodically and are limited to a *k*-hop broadcast. Associated with the client service request message is a client service discovery timer, which determines the periodicity of the client service request message. Associated with the service advertisement message is a server timer, which determines the periodicity of the SA message.

These three mechanisms explained above are fundamental to the PBNM system for which we develop analytical models. With this basic understanding of the PBNM system for MANETs, we develop analytical models using Petri Nets and networks of queues for analyzing the system with respect to service availability.

1.3 Importance of Modeling

Modeling of systems is essential to represent them and study their behavior before their deployment. Sometimes system models are also developed after the system is deployed in order to study the effect of varying certain parameters of the system and to study the interaction between various parameters that characterize the system. When using a model to represent the system it must be abstracted to an appropriate level of complexity.

The domain of computer and communication systems is full of design questions concerning availability, efficiency, cost, etc. [20]. Addressing such problems before the system is deployed in real life involves the development of analytical models and simulation models at appropriate level of complexity. The development of system models for performance evaluation is critical for a qualitative as well as quantitative understanding of modern day complex and distributed systems. The nature of the traffic seen by such systems cannot be exactly predicted and is better understood and characterized statistically. Hence it is typical to develop stochastic models to represent system parameters [20]. Analytical models purely based on queuing theory or simulation models to describe system behavior have been conventionally used. However, purely analytical models at times rely on unreasonable assumptions, while simulation models may require long characterization time to obtain reliable results. Purely analytical models also face the problem of scalability.

In this thesis, we perform stochastic modeling of the system of interest, parameterized and validated through simulation. We develop stochastic models for the PBNM system for MANETs by using Stochastic Petri Nets as a modeling tool. In particular, we analyze the service availability seen by policy nodes in the PBNM system.

1.4 Structure of the document

This chapter provides an introduction to the research presented in this thesis document. We introduced the PBNM system and discussed the motivation for this work. We also explained the importance of modeling distributed systems and motivated the need to analyze our PBNM system. The remainder of this document is organized as follows. In Chapter 2, we present the fundamental concepts in modeling using Petri Nets. In Chapter 3, we present the basic generalized stochastic Petri net (GSPN) model and discuss the results of a simulation based study of the PBNM system. We perform simulations using QualNet in order to parameterize the analytical models and also try to gain some insight into the behavior of critical system processes. In Chapter 4, we analyze the GSPN model for service availability and validate it using a network of queues analysis. We compare the results of this analysis with the service availability values obtained using simulation and test-bed experiments [53] and motivate the need for a more detailed analysis of the model. Chapter 5 deals with the non-Markovian analysis of the system model. In this modified model, we do not make Poisson assumptions for system processes but instead use the probability distributions observed via QualNet simulations for random variables characterizing these processes. We show that with such a detailed analysis we indeed are able to achieve service availability values close to those obtained empirically and through test-bed experiments. Chapter 6 concludes the thesis by stating the contributions of this work and providing directions for furthering research in this area.

Chapter 2

Background and Related Work

This chapter sheds light on the theory behind Petri Nets and the development of the Discrete Phase Type Stochastic Petri Nets and includes a review of related work in the analytical modeling of network systems using Stochastic Petri Nets. The objective is to help the reader understand the basics of Petri Net theory and motivate the use of this theory for modeling a gamut of systems ranging from manufacturing systems, including process flow and reliability modeling of industrial processes, to networking systems such as asynchronous transfer mode (ATM) and wireless local area networks (WLANs), as well as the modeling of routing problems and service availability. We also describe some of the relevant related work on analytical modeling of networking systems, more specifically mobile ad hoc networks (MANETs), using Stochastic Petri Nets. Thus, the chapter provides the necessary background for this thesis. The fundamental terminology and definitions used in modeling using Petri Nets are introduced and motivation for their usage is provided; one can find a detailed discussion on Petri Nets in [4, 5, 62, 65, and 71].

2.1 Petri Net Theory

Petri nets (PNs) are a mathematical tool to describe concurrent systems and model their behavior. They have been used to study the performance and dependability of a variety of systems. Petri Nets essentially consist of three key components: places, transitions, and directed arcs. The directed arcs connect the places to the transitions and the transitions to the places. There are no arcs that connect transitions to transitions or places to places directly. Each place contains zero or more tokens. A vector representation of the number of tokens over all places defines the **marking** of the Petri Net and hence the state of the system.

Definition 2.1:

A PN is defined by a 6-tuple (P, T, I, O, H, M_0) where:

- P is the set of places $\{p_1, p_2,\ldots,p_n\}$, with cardinality n;
- T is the set of transitions $\{\tau_1, \tau_2,\ldots,\tau_m\}$, with cardinality m;
- I, O and H are, respectively, the set of input, output and inhibition functions mapping the set of transitions T to the bag[†] of places, Bag(p).
- M_0 defines the initial marking of the net, i.e. the initial number of tokens in each place.

A PN can be represented as a bi-partite directed graph G = (V, E) where places and transitions are the vertices of the graph, forming the elements of set V. The set E, consisting of input, output or inhibitor arcs, is the set of edges of the graph. Figure 2.1.1 shows a graphical representation of a PN.

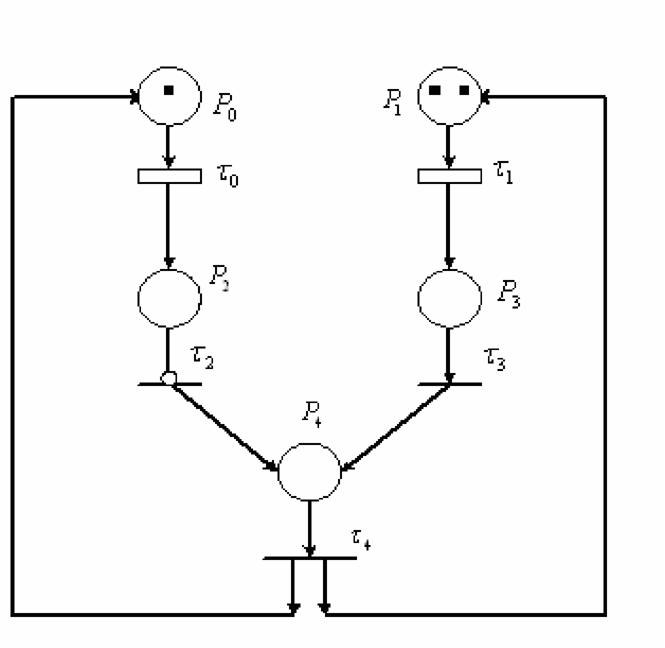


Figure 2.1.1 Graphical representation of a PN

[†] Bags are sets with allowance for multiple occurrences of the same element. They keep a count of the number of occurrences of each element. [50]

As shown in Figure 2.1.1, a place is represented by a circle, with tokens represented by black dots. Thus, in the initial marking M_0 of the PN shown above there are one token in place P_0 and two tokens in place P_1. The initial marking vector is thus given by $M_0= \{1,2,0,0,0\}$. $M_i(p)$ indicates the number of tokens in place p in marking M_i. Thus, in the initial marking M_0, $M_0(p_0) = 1$, $M_0(p_1) = 2$ and so on.

Transitions are represented by rectangular bars, while input and output arcs are represented by a simple line with an arrow head pointing to the destination. Inhibitor arcs are represented by a line with a bubble at its end. Thus, in Figure 2.1.1, the arc connecting place P_2 to transition τ_2 is an inhibitor arc while all other arcs are simply directed arcs. Also, the transitions represented by flat rectangles (for instance, τ_2, τ_3 and τ_4) are immediate transitions while the rectangles with a non-zero width (for instance, τ_0 and τ_1) represent timed transitions. Petri Nets can be classified on the basis of the presence of such different types of transitions; we review some of the relevant classes of Petri Nets in subsequent sections of this chapter. For any place $p \in P$ and any transition $\tau \in T$, the place p is said to be the input place of τ if there is an input arc from p to τ. Similarly, a place p is said to be an output place for a transition τ if there is an output arc from τ to p. p is called the inhibitor place of τ if there is an inhibitor arc from p to τ. Arcs may have multiplicities associated with them. This is done for simplicity of modeling and implies that the number of tokens corresponding to the multiplicity of the input arc will be removed from the input place of the transition when it fires. Also, for an output arc with multiplicity *x*, *x* tokens would be deposited in the output place on the firing of the transition that the arc connects to the output place. The notion of multiplicities is equivalent to having those many input (output) arcs from (to) the input (output) place to (from) the transition that fires.

The behavior of a concurrent system is described by the evolution of its PN model with time. As the PN evolves, the system attains different states that can be completely defined by the marking of the PN model. The dynamic evolution of the PN is defined by the enabling and the firing rules. A transition $\tau \in T$ is enabled in marking M_i if:

- Each input place contains a number of tokens at least equal to the multiplicity of the input arc from the input place to the transition τ; and
- Each inhibitor place has a number of tokens smaller than the multiplicity of the inhibitor arc from the inhibitor place to the transition τ.

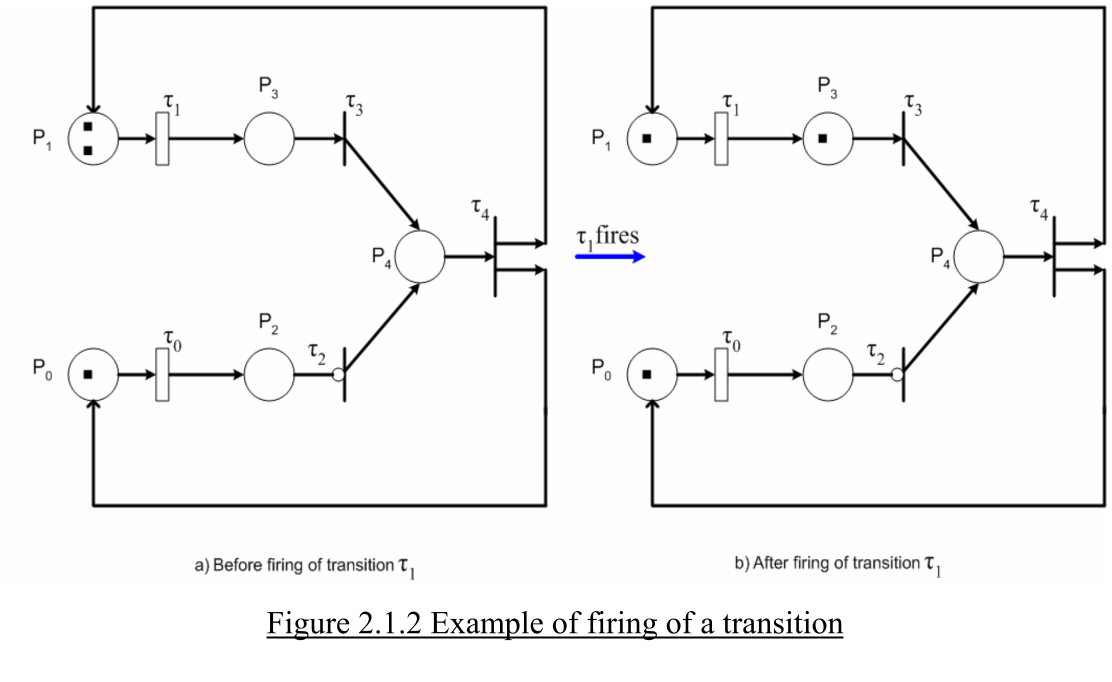
Mathematically, τ_k is enabled in a marking M if and only if [65]:

- $\forall p \in P_i(\tau_k)$, $M(p) \geq I(\tau_{k,p})$ and; (2.1.1)
- $\forall p \in P_h(\tau_k)$, $M(p) < H(\tau_{k,p})$ (2.1.2)

where $P_i(\tau_k)$ and $P_h(\tau_k)$ are the sets of input and inhibitor places of transition τ_k, respectively.

Whenever there are several transitions enabled simultaneously, the decision as to which transition gets fired is made according to the firing policy of the PN. On firing, a number of tokens equal to the multiplicity of the corresponding input arc are removed from each of the input places and put into each the output places of the transition according to the multiplicity of the respective output arcs. The firing of a transition leads to the evolution of the net into a new marking M_j, which is described by the equation [65]:

$$M_j = M_i - I(\tau_k) + O(\tau_k) \quad (2.1.3)$$



a) Before firing of transition τ_1

b) After firing of transition τ_1

Figure 2.1.2 Example of firing of a transition

The firing of a transition is illustrated in Figure 2.1.2, wherein the firing of transition τ_1 removes one token (the multiplicities of the input arc connecting P_1 to τ_1 and that of the output arc connecting τ_1 to P_3 are both 1) from place P_1 and puts it in place P_3. The dynamics of the PN are described by means of a **reachability graph**, which indicates the markings reached by the net by the firing of transitions and is thus the **state diagram** representation of the PN behavior. Given an initial marking M_0, a marking M_k is said to be reachable if there exists a sequence of firing transitions leading from M_0 to M_k. A set of all such possible markings that are reachable from the initial marking M_0 is called the **reachability set** and is denoted by $RS(M_0)$.

2.2 Classification of Petri Nets

Traditional PNs as defined in Section 2.1 have evolved over time, with increase in their modeling power, through which more complex systems can be analyzed for performance limitations and reliability. In this sub-section we focus on certain fundamental types of PNs, formally defining them and explaining their characteristics.

2.2.1 Timed Petri Nets

Reference [62] discusses several different ways in which timing can be associated with Petri Nets. Timing may be associated with the places of the Petri Net, in which case there is a delay attributed to the place. These kinds of nets are called timed place Petri nets (TPPNs). In TPPNs, the tokens in the input place of the transition become available to fire a transition only after a certain time delay. Similarly, delay can be associated with the tokens in the PN. The tokens carry a time-stamp as to when they are available for firing a transition. This time-stamp is then updated after the firing of each transition. Time may also be associated with the arcs. One can imagine a traveling delay for the tokens to move across the length of the arc. Thus when the tokens reach the transition, they are assumed to be available for firing.

Despite these possibilities of associating temporal parameters with the places, tokens or arcs, one class of Petri Nets that is of particular interest is the class of timed transition Petri nets (TTPNs). TTPNs were developed to model the duration of events in real life systems and associate timing parameters with the transitions. Formally, a TTPN is defined as:

Definition 2.2:

A TTPN is a tuple TTPN = $\{P, T, I, O, H, M_0, \Omega\}$ where P, T, I, O, H and M_0 have the same meaning as in Definition 2.1. Ω is a function defined over the set T that assigns timing to each transition $\tau \in T$.

There are two possible types of timing that one can associate with each timed transition. Certain transitions fire after a fixed deterministic delay, while others fire on the basis of a probabilistic firing function defined by a random variable, say $\theta(\tau)$.

2.2.2 Stochastic Petri Nets

Another important and perhaps the most widely used class of Petri Nets are Stochastic Petri Nets, wherein timed transitions are characterized by negative exponential probability distribution functions (PDF). They are described by the parameter λ_k corresponding to each exponentially distributed timed transition τ_k. Since the transitions obey exponential pdfs, they exhibit the memory-less property, wherein the activities associated with the transition in a marking are restarted in each new marking and its respective local clock (that keeps the notion of the time for the transition) is reset to 0. A more formal definition of SPNs is presented in definition 2.3.

Definition 2.3:

A stochastic Petri net (SPN) is a TTPN wherein SPN = $\{P, T, I, O, H, M_0, \Omega\}$ where P, T, I, O, H and M_0 are essentially the same as described in Definition 2.1. $\Omega: T \rightarrow R$ is a function that assigns exponentially distributed timings to each transition $\tau \in T$.

The evolution of SPNs with time can be mapped into an equivalent continuous time Markov chain (CTMC) [43], the states of the CTMC being the markings present in the reachability graph of the PN. The rate associated with the transition arcs defining the CTMC evolution is the sum of the rates of transitions whose firings result in the new marking of the PN (new state of the CTMC). This is one of the most important properties of SPNs. By solving the underlying Markov chain, one can obtain the state probabilities of the marking process *M(τ),* which are reflective of the system parameters in some form and, hence, characterize the behavior of the SPN and therefore the system as a whole.

2.2.3 Generalized Stochastic Petri Nets

Often, the various timing parameters involved in the models differ by several orders of magnitude. For instance, in the modeling of the behavior of a concurrent HTTP server, the time to establish the connection may be much smaller than the amount of time for which the connection is kept active. It therefore makes sense to have activities (like connection establishment) that take a relatively small time be modeled only logically and model the relatively time significant activities with timed transitions. Generalized stochastic Petri nets (GSPN) allow such specifications. The basic idea is to model transitions with relatively short times as immediate transitions (transitions that fire deterministically in zero time) and the ones that fire with significantly larger times as timed transitions. Immediate transitions may be used, for instance, to model almost-instantaneous actions (like a choice of server among a set of servers). A formal definition of the GSPNs is provided below.

<u>Definition 2.4:</u>

A generalized stochastic Petri Net is a tuple GSPN = {P, T, I, O, H, M_0, Ω, Prio} where P, T, I, O, H, M_0 and Ω carry the same meaning as described in Definition 2.3. Prio: T→ N is a function that assigns priorities to each transition $\tau \in T$.

If τ_k is a timed transition, then $\theta(\tau_k)$ is representative of the λ_k parameter of its exponential pdf, while if it is an immediate transition, then it represents the weight of the firing probabilities of immediate transitions. With the presence of priorities, the enabling conditions are modified to incorporate these priorities. The transition τ_k is said to be enabled in marking M_i if and only if:

- $\forall p \in P_i(\tau_k)$ => $M(p) \geq I(\tau_{k,p})$ and; (2.2.3.1)
- $\forall p \in P_h(\tau_k)$ => $M(p) < H(\tau_{k,p})$ (2.2.3.2)

and no other transition $\tau_j \in T_,$ also satisfying the above two equations, has Prio(τ_j) > Prio(τ_k). As a general rule, immediate transitions have higher priority than timed transitions. In case several immediate transitions are enabled in a marking, the associated probabilities are used to break the tie.

Due to the presence of immediate as well as finite time transitions, there are two types of markings that are possible in the GSPN. **Tangible** markings refer to markings wherein there is some non-zero time for which the system remains in that marking and such markings (states of the system) can be perceived tangibly by the user. Thus markings with timed transitions enabled are called tangible markings. **Vanishing** markings, as the name suggests, are markings that exist only for a small instant of time (users perceive this as essentially zero time) and then enter a tangible marking. Thus, vanishing markings are the ones wherein only immediate transitions are enabled. In case of such markings, one needs to associate a policy as to which of the immediate transitions would fire. This arbitration is done by associating probabilities with the transitions as explained before. These probabilities are called **switching** probabilities in GSPN. It is not mathematically possible to analyze the GSPN as a CTMC because of the zero time the net spends in a marking when an immediate transition is enabled.

However, one can remove such vanishing markings from the reachability set and analyze the new net with CTMC without causing any behavioral changes in the model. A GSPN model can thus be analyzed using the evolution through tangible markings of the PN. Figure 2.2.3.1 shows the modification of the original reachability graph to remove the vanishing states. Once all the vanishing markings are removed, the reachability graph obtained is called the reduced reachability graph $\hat{R}$ (M_0).

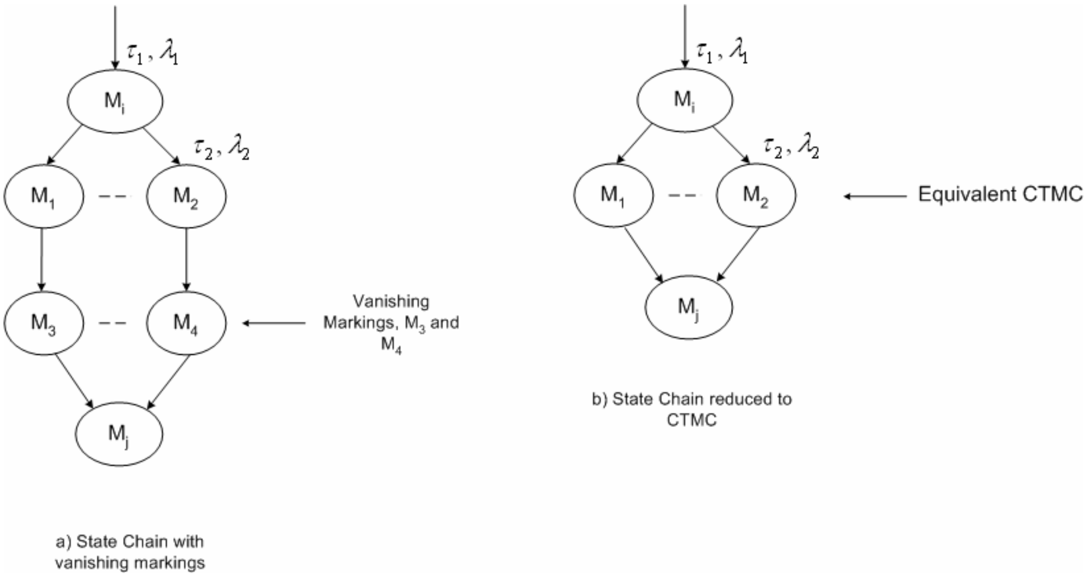


Figure 2.2.3.1 Reducing of GSPN to CTMC

2.2.4 Generally Distributed Transition Stochastic Petri Nets

The usual definition of a GSPN implies that all timed transitions are exponentially distributed and all immediate transitions have probabilistic weights associated with them. As stated in sub-section 2.2.3, these can be effectively mapped into a CTMC. However, most practical systems do not behave in this manner. Not all the timed activities of systems can be appropriately modeled as following an exponential distribution. Over the past few years, several classes of SPN models have been developed that account for the non-exponential characteristics in their definition. These models are broadly classified as generally distributed transition stochastic Petri nets (GDT_SPN).

A more formal definition of a GDT_SPN is given below [65]:

Definition 2.5:

A generally distributed transition SPN is an 8-tuple GDT_SPN = {P, T, I, O, H, M_0, Ω, Mem} where P, T, I, O, H, M_0 carry the same meaning as in Definition 2.1. The function Ω: T→ F(.) characterizes the general cumulative distribution function (CDF) of the random variable associated with each timed transition. F(.) is the operator on a transition τ_k that associates a specific cumulative distribution function with the transition. Mem: T→ {prd, prs, pri}is a function that assigns the memory policy to each transition $\tau \in T$. prd, prs and pri are the types of memory policies and stand for **preemptive repeat**

different, **preemptive resume** and **preemptive repeat identical**, respectively. The stochastic process underlying the GDT_SPN is completely defined by the **execution policy** of a Petri Net. Given the CDFs associated with the timed transitions, the execution policy is comprehensively defined by the **firing policy** and the **memory policy** of the GDT_SPN. The firing policy decides the next timed transition to fire when the net is in a certain marking M. The memory policy decides the rule to store the history of the process as it moves through the different markings of the reachability graph.

The most widely used firing policy for simultaneously enabled transitions in a PN is the race policy. According to the race policy, if more than one timed transition is enabled in any given marking, the timed transition that has the minimum average delay (mean of the random variable characterizing the delay process for the transition) is selected for firing. The memory policy is described by two variables, the age variable σ_k and the threshold variable ρ_k. The threshold variable ρ_k is the minimum value that the clocks associated with each transition should reach for the transition to fire. The age variable σ_k, as the name suggests, is a variable that stores the age of the transition τ_k enabled in the current marking and hence increases with time. The way in which σ_k is related to the past history of the process determines the different memory policies.

The following three different memory policies can be associated with GDT_SPNs:

- preemptive repeat different (prd)
- preemptive resume (prs)
- preemptive repeat identical (pri)

Whenever the PN enters a new marking, the remaining firing time $rft_k = \rho_k - \sigma_k$ is computed for each transition $\tau_k \in T$ that is enabled in the current marking. The transition that has the minimum remaining firing time fires when the age variable σ_k becomes equal to ρ_k, in accordance to the race policy defined above. However this does not completely specify the behavior of the net. The manner in which σ_k behaves when transition τ_k is disabled due to firing of some other enabled transition in the marking is determined by the memory policies specified above.

In the preemptive repeat different memory policy, each time a transition fires, the memory associated with transitions that remain enabled after the firing is retained, i.e. the age variable σ_j of the transitions that are not disabled is retained, whereas the age variable σ_k of the disabled transitions is set to 0. The threshold ρ_k for all transitions is resampled from the CDF each time they are enabled.

According to the preemptive resume policy, if the transition is disabled before the firing threshold is reached, then the value of age variable σ_k accumulated up to that point is maintained. Also, the threshold value ρ_k is sampled only when the transition is enabled for the first time after firing. This is a work conserving policy in the sense that it maintains the age variable σ_k value even if the transition gets disabled. Finally, in the case of the preemptive repeat identical memory policy, the memory variable σ_k is reset when the transition is disabled, but the threshold sampled value ρ_k is maintained and the same value is used when the transition is enabled again.

2.2.5 Phase Type Stochastic Petri Nets

It is difficult to obtain an analytically tractable solution to the marking process represented by a GDT_SPN. A numerically tractable realization of the GDT_SPN is obtained by representing the random firing times ρ_k to be phase (PH) type distributed [46]. In this section, we introduce an analysis technique for GDT_SPNs that assigns a PH distributed random firing time to each transition. This class of SPNs is called phase type stochastic Petri nets (PH-SPNs). A more formal definition will be provided after we discuss the motivation for continuous and discrete phase type distributions. The basic step in solving the GDT_SPN model is to approximate the probabilistic behavior of transitions by continuous or discrete phase distributions.

<u>**Definition 2.6:**</u>

A continuous phase type distribution (CPH) [46] is defined as the CDF of time until the absorption of a continuous time Markov chain (CTMC) with m transient states and one $(m+1)^{th}$ absorbing state, given an initial probability vector.

If there is a CTMC χ with m+1 states (m transient states and 1 absorbing state) and the chain has an initial probability vector $[\alpha_1, \alpha_2, \ldots \alpha_{m+1}]$, then if we define random variable ξ as the time until χ reaches the absorbing state, ξ has a phase type distribution. Let the initial probabilities of the CTMC be defined by $\pi(0) = [\boldsymbol{\alpha}, \alpha_{m+1}]$, where $[\boldsymbol{\alpha}] = [\alpha_1, \alpha_2, \ldots \alpha_m]$, then

$$\alpha_{m+1} = 1 - \sum_{i=1}^{m} \alpha_i \qquad (2.2.5.1)$$

Let $\mathbf{G_I}$ be the infinitesimal generator matrix [46] for the CTMC χ; $\mathbf{G_I}$ can be partitioned as:

$$\mathbf{G_I} = \left[\frac{G}{0} \mid \frac{G^{abs}}{0} \right]$$

where $\mathbf{G} = [g_{ij}]$ is a (m x m) transient matrix that represents the transient state behavior of the CTMC and $\mathbf{G^{abs}}$ is a (m x 1) column vector such that $\mathbf{G^{abs}} = \mathbf{-G.e}$ where $\mathbf{e}$ is an identity column vector (m x 1). The vector $\mathbf{G^{abs}}$ represents the grouping of transition rates into the absorbing state.

The CPH defined as above can be completely defined by $[\boldsymbol{\alpha}, \mathbf{G}]$. Another important property of the CPH as defined by the above equations is [46, Lemma 2.2.1] that the matrix G has to be a non-singular matrix for all the states 1,2, … m to be transient, because only then would the probabilities of an eventual absorption from any initial state to the absorbing state be all 1.

Definition 2.7:

A discrete phase type distribution (DPH) [46] is defined as the CDF of time until the absorption of a discrete state discrete time Markov chain (DTMC) with n transient states and one (n+1)th absorbing state, given an initial probability vector.

The most interesting property of DPH distributions is that they can exactly represent a number of different distributions with finite support as well as infinite support models. Using the conventions we adopted in CPH, we number the transient states as 1, 2, …n and the absorbing state as n+1. A DPH is then represented by the initial probability vector given as [$\boldsymbol{\alpha}$, α_{n+1}] of the DTMC and the one-step transition probability matrix $\mathbf{B_I}$ written in the partitioned form as:

$$\mathbf{B_I} = \left[\frac{B}{0} \Big| \frac{B^{abs}}{0} \right]$$

where $\mathbf{B} = [b_{ij}]$ is the transient probability matrix for transition between transient states. The column vector $\mathbf{B^{abs}}$ stores the probabilities from any state to the absorbing one. The DPH is thus represented by [$\boldsymbol{\alpha}$, $\mathbf{B}$] and has order n.

Now that we have defined the basic classes of PH type distributions, we formally describe PH-SPNs.

Definition 2.8:

PH-SPNs [65] are a class of GDT_SPNs wherein the set Ω: T→ F(.) is a set of phase distributed random variables defining the CDF of the transitions. If the random variables in Ω are continuous phase type distributed, then the SPN is called a continuous phase type stochastic Petri nets (CPH-SPN), while if they are discrete phase type distributed, then it is called a discrete phase type stochastic Petri nets (DPH-SPN).

The solution of CPH-SPNs has been suggested by Cumani [13]. It uses a state space expansion algorithm to compute the measures of interest in the Petri Net. The algorithm uses the representation of CPH-SPNs as finite state Markov chains. An analogous version of the solution for the DPH-SPNs can be achieved using the algorithm developed by Marco Scarpa *et al.* [65, Chapter 4]. The WebSPN package provided by Puliafito, Scarpa *et al.* [28, 57] and the algorithm that they proposed can help solve DPH Stochastic Petri Nets with no restrictions on the concurrency of generally distributed time transitions in terms of their firing being mutually exclusive.

2.3 Applications of Petri Net Theory to Computer Networks

Different formalisms, including networks of queues, Petri Nets and Stochastic Process Algebra, can be used to model complex systems such as computer systems, communication networks, automated manufacturing systems, transportation systems, etc. In this section, we briefly compare two modeling formalisms as applied to computer networks: networks of queues and PNs. The assessment of performance and reliability of a special class of computer networks, namely wireless networks, is discussed separately. Finally, we present facets of our work that differentiate it from the existing work in modeling computer networks.

2.3.1 Analytical Models for Computer Networks

Queuing networks are a classical formalism in modeling of computer systems. In particular, analytical models for computer network systems using queuing networks have been studied for years. For instance, Wong in [73] developed a queuing network model for communication networks. This work focused on modeling packet switched networks for the analysis of end-to-end delay and buffer management. In [68], packet switched networks are modeled for mean time spent by messages in the system using the Jackson network of queues analysis [31]. Reference [26] discusses congestion and flow control model in high level data link control (HDLC) based networks with an assumption of Poisson arrivals for queues. Traffic modeling of bursty network traffic has been studied using queuing network model by Spirn in [66] with exponential arrival and exponential service distribution assumptions. More recently, Garetto *et al.* [17] have developed a closed network of queues model for performance analysis of TCP connections. This work is analogous in modeling concept to the Markovian analysis (Chapter 4) we perform on our PBNM system (Chapter 1, Section 1.2). Many such models of computer networks based on networks of queues appear in literature.

There are limitations to the use of queuing theory to model modern networks. Most importantly, closed form solutions for queuing systems rely on the assumption that arrivals to the queue can be described as a Poisson process. While this may be a reasonable assumption in some cases (e.g., arrivals of telephone calls to a switch, generation of TCP sessions, etc.), many phenomena of interest cannot be appropriately modeled as a Poisson process (e.g., data packet arrivals to a switch, handover of a mobile between points of access, etc.). Furthermore, while modeling distributed systems, traditional queuing networks do not provide the means for representation of synchronization and simultaneous resource possession. Petri Nets provide an alternative for modeling concurrent and distributed systems.

Starting from the introduction of their basic idea by C. A. Petri in 1962, Petri Nets have evolved and grown in their modeling power. Notions of time, preemption and priority have made them suitable for modeling complex distributed systems. We focus on the application of Petri Nets to the modeling of computer networks and computer communication systems. Several Petri Net models for performance and reliability analysis of network systems appear in the literature. These include models for data link protocols [44], local area networks [22], reliability analysis of carrier sensing multiple access/collision detection (CSMA/CD) [37, 40 and 56], and the analysis of TCP queue management algorithms like random early detection (RED) [27]. Reference [71] illustrates the use of Stochastic Reward Nets for studying the effect of early packet discard in ATM networks. The performance of client-server systems has also been analyzed using SPNs [48, 49]. Reference [49] presents a model for analyzing the slow-start mechanism in TCP using WWW traffic as the workload, while [48] characterizes video traffic over the World Wide Web. Applications of Petri Net theory are not limited to these; modeling and measurement of systems using PNs has been extended to almost every aspect of computer networks. With computer networking technology going wireless, there is increased interest in building models to analyze wireless and mobile networks. The following sub-section discusses the modeling of wireless networks using both queuing networks and Petri Nets.

2.3.2 Focus on Modeling of Wireless Networks

Much of the recent work in analytical modeling of wireless networks has employed either traditional queuing models or Markov chains analysis. For instance, the work presented in [39] uses the M/G/1 queuing model to analytically determine the delays incurred in handling different types of queries using the adaptive periodic threshold-sensitive energy efficient sensor network (APTEEN) protocol for wireless sensor networks. Also, in [9], an analytical study of the survivability of ad-hoc networks is performed with an exponential distribution assumed for the time to occurrence of each component failure event in the ad hoc network [9, Section 3]. Konrad *et al.* [33] have focused on developing a Markov-based model for modeling channel parameters in wireless networks. Zhu and Chlamtac [76] have developed a theoretical Markov chain model for calculating traffic priority and throughput, under saturation conditions, for the IEEE 802.11e enhanced distributed coordination function (EDCF) based medium access control (MAC) layer. Researchers have also started applying Petri Net theory to model wireless networks, including wireless LANs, MANETs and mobile agents. Reinhard German *et al.* [19] have developed non-Markovian SPN models for medium access mechanisms in wireless LANs. Reference [19] does not assume Markovian properties for modeling the dynamics of the Distributed Coordination Function in Wireless LANs. Parameters like throughput, mean waiting time for transmission, etc. are modeled. Xiong *et al.* [74] have modeled the ad hoc on-demand distance vector (AODV) routing protocol using colored Petri Nets [62].

Our research focuses on modeling the service (in terms of connectivity to a policy server) received by mobile nodes in a policy based network management framework [54] using PNs as the performance evaluation methodology. Petri Nets are a powerful mathematical technique for modeling complex systems. There are several software tools [70] available for the performance analysis of different classes of PN models. Also, PNs adopt a decentralized paradigm, making them an attractive choice for modeling ad hoc networks. In case of systems where the inter-arrival times for events are not necessarily exponentially distributed, the GDT_SPN class of PNs can be used to specify and solve the system models. Software tools like WebSPN [57] can be used in these cases for model specification and obtaining desired steady state measures. This thesis employs PN theory to characterize service availability in MANETs. We also believe that the research methodology followed here can be applied to capture other performance parameters of wireless ad hoc networks. After developing a simple GSPN model for service availability in MANETs, we add complexity to it in order to incorporate the actual behavior of client nodes in the PBNM system (Chapter 3). We achieve this by resorting to a simulation based study of the actual distributions these client processes follow. This leads us to a more accurate GDT_SPN representation of the model of our PBNM system (Chapter 5). In the simple Markovian scenario (GSPN model), we are able to solve the model and relate it to the classical Markov chain analysis using a closed network of queues (Chapter 4). In the non-Markovian analysis of the model, which incorporates the actual distributions of the client processes, we are able to approximate the general distributions using DPH distributions and solve the model. To the best of our knowledge, such a modeling methodology (Chapter 5, Section 5.3) has never been applied to study performance parameters of a MANET. Following this methodology, we are able to achieve service availability results close to the ones obtained via simulations (Chapter 3). Another contribution of this research is the ability to model a client-server type PBNM system in context of MANETs. A more detailed discussion of significant contributions of this research is presented in Chapter 6.

2.4 Summary

In this chapter, we introduced and defined the basic terms used in Petri Net theory. We discussed the evolution of Petri Nets and demonstrated the increase in modeling power achieved with these advances. Also, the inclusion of notions of time and phase in Petri Nets result in the development of system models that are more accurate in terms of capturing their exact behavior.

With necessary motivation on the basics of using Petri Nets as a modeling tool and their applicability to the modeling of computer network systems, we intend to revisit the concepts explained here and apply them to our benefit in Chapters 4 and 5. A detailed study of the PBNM system using GSPNs is conducted in Chapter 4 while a non-markovian GDT_SPN analysis of the system is presented in Chapter 5.

Chapter 3

Empirical Model Parameterization

The PBNM system for ad-hoc networks (Chapter 1, Section 1.2) provides a unified framework for provisioning and management of QoS in MANETs. It consists of a set of nodes in a MANET, some of which are designated as servers. Other nodes retrieve their policy specifications from the policy servers and are termed as the policy clients. In this research, we develop a performance model that characterizes the behavior of the system with respect to the likelihood that the clients will be covered by a policy server. Chapter 2 provides the necessary background for the development of analytical models using Petri Net theory. Chapter 4 presents the GSPN representation of the model for the PBNM system while Chapter 5 deals with non-Markovian SPN analysis of the same system.

Model parameterization is an important step for an accurate study of the behavior of the system. In this chapter, we discuss various simulation scenarios considered for parameterization of the Markovian and non-Markovian representations of the model we developed in Chapters 4 and 5 respectively and present the results obtained. We assess the impact of variation of different simulation parameters like the mobile pause time, the keep alive (KA) timer value, cluster size and speed of mobile nodes [53] on the parameters of interest (Section 3.1). The remainder of the chapter is organized as follows. In Section 3.1, we introduce the basic model with Markovian assumptions and highlight the parameters needed to characterize the model. Section 3.2 describes the role of QualNet as our network simulator tool of choice. It also discusses the simulation environment used to amass the necessary statistics for parameterizing the model. We also explain the changes made to the OLSR [30] implementation provided by QualNet [58] in this section. Four basic simulation scenarios are explained in detail in Section 3.3. A thorough analysis of the effect of varying simulation parameters like pause time and KA timer value on the parameters of interest is presented in Section 3.4. We summarize the main ideas presented in this chapter in Section 3.5.

3.1 Parameters of Interest

Parameterization is a crucial step in modeling of our PBNM system for MANETs. A careful choice of parameters that describe the behavior of the system is essential. In this section, we introduce the basic GSPN (Chapter 2) model of our PBNM system and identify parameters essential to characterize service availability. Note that the model we introduce here is used as is for Markovian analysis of the PBNM system (Chapter 4). However, additional complexity is introduced in the model later on (Chapter 5) to enable a non-Markovian analysis. Figure 3.1.1 shows a simple Petri Net model of our PBNM system from a client's perspective, but under Markovian assumptions for timed transitions and switching probabilities for immediate transitions. Hence, we call this model the generalized stochastic Petri net (GSPN) model. In creating this model, we omit intricacies like redirection and delegation to come up with a simple model that captures the fundamental aspects of client behavior.

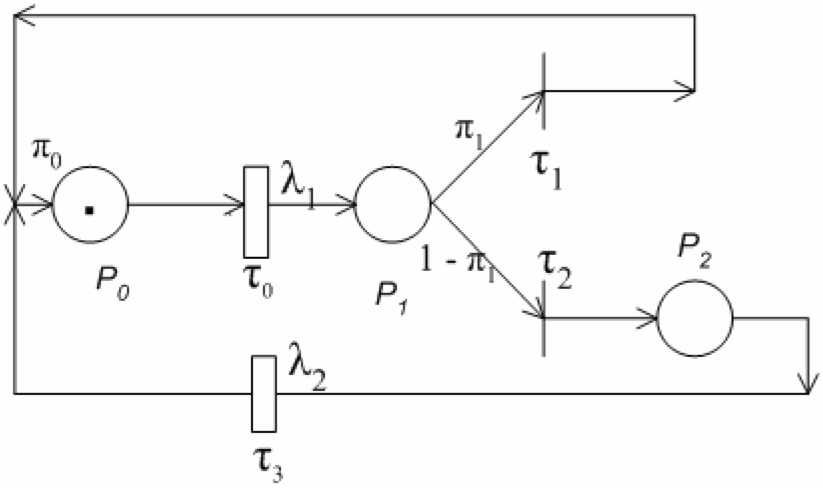


Figure 3.1.1 GSPN Model for policy client in the PBNM system

The model consists of three places P_0, P_1, P_2 and four transitions $\tau_0, \tau_1, \tau_2, \tau_3.$ Two of the transitions, τ_1 and $\tau_{2,}$ are immediate, while the remaining ones are timed with exponentially distributed transition times. Place P_0 represents the state where the client is currently covered by some cluster. We assume this to be our initial condition and this is represented by the presence of a single token in place P_0. Thus, the initial marking of the PN is given by $M_0 = \{1,0,0\}$. The transition τ_0 represents the event of the client moving out of the cluster. The time interval between consecutive occurrences of this event, under Markovian assumptions, is assumed to be exponentially distributed with a firing rate of λ_1. A token in place P_1 represents that the client has moved out of the cluster; this is a vanishing marking (Chapter 2, Section 2.2.4). It enables two immediate transitions τ_1 and τ_2, any of which can fire in accordance to the switching probabilities. π_1 represents the probability with which the client moves back immediately into a new cluster, once it moves out of the current cluster. This is the switching probability associated with the immediate transition τ_1 and, hence, a switching probability of $(1-\pi_1)$ is associated with the immediate transition τ_2. A token in place P_2 represents the scenario where a client, upon moving out of a cluster, has not immediately moved into a new cluster. The transition τ_3 represents the time it takes for this "out of service" client to enter a new cluster. Since this model assumes that the transition times are exponentially distributed, we associate an exponential decay parameter of λ_2 with transition τ_3. We also associate a probability π_0 with the event that there is a token in place P_0. Since a token in place P_0 represents the state of the system wherein the client is covered within one of the clusters, the probability π_0 is the measure of service availability seen by the client. Note that there is just a single token, corresponding to a single client under observation, and the entire system model is based upon characterizing the behavior of this client node in the PBNM system.

The model explained above serves as a reference for identifying **parameters of interest** that characterize system behavior from a service availability viewpoint. The first parameter of interest that we introduce is the rate λ_1 at which the clients leave the k-hop cluster. For the GSPN model shown in Figure 3.1.1, this is the decay parameter of the exponential distribution associated with transition τ_0. In general, the inverse of λ_1 represents the average amount of time for which clients remain inside a cluster. We chose this metric as a parameter of interest because it characterizes the time for which the client remains in service, and is thus indicative of the service availability. Another important parameter in modeling service behavior is the probability π_1 with which the clients, upon leaving a cluster, move immediately into a new k-hop cluster. By immediately, we mean, within an acceptably low amount of time. For simulation purposes, we set this time to be equal to the time out interval of the keep alive (KA) timer (Chapter 1, Section 1.2). As shown in Figure 3.1.1, π_1 is the switching probability associated with transition τ_1. A value of $\pi_1 = 1$ would indicate a service availability of 100% because at this value of π_1 the client theoretically never loses service and each time moves seamlessly into a new cluster. However, the estimated values for π_1 obtained via QualNet simulations (Section 3.3.2) are far less than 1 indicating a service availability fairly below 100 %. If a client does not move back into a new cluster immediately, it enters a state where it is looking for service. The client spends some time in this state before it gets covered within a new k-hop cluster. We characterize this time for which it remains out of service using the third parameter λ_2. Thus, λ_2 is the rate at which the client moves back in to a new cluster, after losing service. For the GSPN model shown in Figure 3.1.1, it is the decay parameter for the exponential distribution associated with the transition τ_3. The reciprocal of λ_2 corresponds to the average time for which clients remain out of service. We use these three parameters to model service availability to policy clients. There are other parameters, like the arrival rates of the clients into a cluster, which are redundant and can be derived from the three key parameters of interest. In our simulations, we do collect these auxiliary parameters to analyze and verify the movement of the clients in and out of the clusters. Also note that the parameter λ_2 cannot be directly obtained through simulations and is derived from the average results obtained for $\lambda_2(1-\pi_1)$ and the estimate of the probability π_1.

3.2 Simulation Methodology

In order to analyze the system performance in terms of service availability, we developed the GSPN (Section 3.1 and Chapter 4) and the GDT_SPN (Chapter 5) models of the PBNM system. As mentioned earlier, parameterization of these models is a crucial step in the analysis. We collected statistical data for the parameters of interest (Section 3.1) by running a set of simulations in order to parameterize our models. These simulations were carried out using the QualNet network simulator tool [58]. In this section, we discuss the QualNet simulation environment used for gathering these statistics. The QualNet simulation code for the PBNM system provided by Phanse [53] was modified to collect the required parameters of interest. The simulation scenario we adopt is a simple scenario without server delegation; service discovery and clustering [54] mechanisms, however, are implemented. We added code to compute instantaneous and average values of λ_1 and $\lambda_2(1-\pi_1)$ and to obtain an estimate of the value of probability π_1 for the entire simulation set. A description of QualNet as a network simulator tool and the models used and modified for our purpose is presented in Section 3.2.1. The actual simulation environment used for data collection and analysis is described in Section 3.2.2.

3.2.1 The QualNet Network Simulator

QualNet Simulator is a C-based network simulation tool. It provides a JAVA based graphical user interface (GUI) and comes with an extensive library of pre-defined and user-modifiable functions and data structures. QualNet comprises four main components:

(a) QualNet Animator: This is the main window of the tool. It allows users to define scenarios and specify node parameters and their topology. Tool bars are available for gathering node statistics, setting up experiments, and collecting simulation results.

(b) QualNet Designer: This component allows users to develop new protocol models graphically by means of finite state machines (FSMs). The design files are saved as *.xml* files.

(c) QualNet Tracer: This module allows users to capture and trace data belonging to a particular layer in the protocol stack. Using the GUI provided for the tracer module, analysis of various packets and their headers can be done. This functionality is similar to traditional trace analysis tools like Ethereal and tcpdump.

(d) QualNet Analyzer: This component of the QualNet simulator lets users analyze the results of a simulation scenario. It provides a graphical tool to plot graphs of desired metrics, compare results over different batch runs, etc.

It is interesting to note however, that as a matter of practicality, the simulation speed is reduced considerably by the use of these Java based GUI tools. Hence, we prefer to run all the simulation scenarios using a command-line interface. Figure 3.2.1.1 shows a snapshot of a sample scenario of our simulation runs. Nodes 2 and 8 are the servers and are marked by arrows, while other nodes act as policy clients.

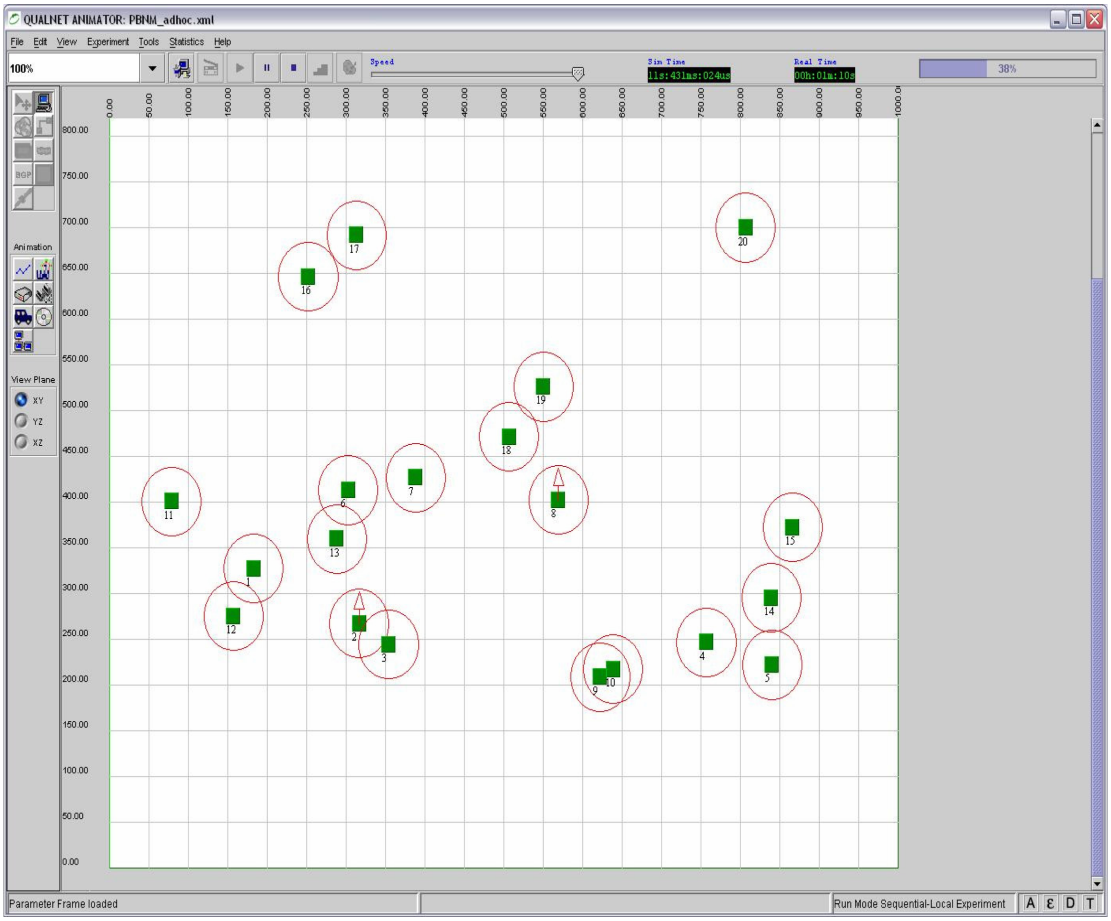


Figure 3.2.1.1 QualNet simulation of the PBNM system - sample scenario

We now discuss the implementation specifics, as to which files were modified and used in the simulation of the PBNM system. As noted before, the originally existing [53] source code files for the PBNM system were used and modified as needed. The following are the key files in our simulation model.

a) cops.h: This file contains definitions and prototypes of all functions, structures and other variables used to implement all the COPS/COPS-PR functionality for QualNet in accordance with the IETF standards [8, 15].

b) pbnm_adhoc.c: This is a key file that contains all the source code for the PBNM system. It contains all the function specifications that define the behavior of the clients and the servers of the system at different layers in the protocol stack. It also implements all the COPS/COPS-PR functionality needed for our application (policy-based management of QoS in a MANET). This core file was modified to collect the statistics of interest.

c) pbnm_adhoc.h: This file serves as a prototype definition file for all functions, structures and variables defined for the simulation model. It contains the prototype definition of a structure called *struct_pbnm_adhoc_str* that stores all the requisite information on a per node basis from an application layer perspective of the system. We introduced several variables in this structure to compute necessary statistics on the parameters of interest. Figure 3.2.1.2 shows a snapshot of the *struct_pbnm_adhoc_str* data structure definition used in our simulations.

```
struct struct_pbnm_adhoc_str
{
    int state;
    int       connectionId;
    int       uniqueId;
    short     sourcePort;
    int       protocol;
    clocktype sessionStart;
    clocktype sessionFinish;
    BOOL      sessionIsClosed;
    double    PBNM_SDP_numBytesSent;
    double    PBNM_SDP_numBytesRecvd;
    double    COPS_numBytesSent;
    double    COPS_numBytesRecvd;
    NodeAddress clientAddr;
    NodeAddress serverAddr;
    int cluster_size;


      clocktype timeConnEstablished;//time at which the connection is
established.
      clocktype timeConnLost;//time at which the connection is lost.
      BOOL conn_First;//flag to check if the connection is getting
established for the first time or not.
      int numConnLost;//Number of the times the client moves out of the
cluster.
      double l1;//Lambda1 for individual client for this run.
      double l2;//Lambda2 for individual client for this run.
      double pi1;//probability of client moving immed. in a new cluster
for this run.
      double il1;//Lambda1 for individual client for this connection.
      double il2;//Lambda2 for individual client for this connection.
      double arr_rate;//arrival process times for clients to enter the
cluster.
      double iarr_rate;//individual arrival rates of clients back in
the cluster.
      double cntl1;//count for calculating the time spent by a client
in a cluster.
      double cntl2;//count for calculating the time required by the
client to move back in the cluster.
      time_InfoData *time_Info;// pointer to the structure that would
store the timings.

      Pbnm_adhocStatsType stats;
      unsigned short seed[3];
      int initStats;
      int printStats;

      NodeAddress PBNM_SDP_serverAddr[10];
      int PBNM_SDP_serverCount;
      BOOL COPS_CONNECTED;
      BOOL PBNM_ADHOC_SERVER;
      BOOL DELEGATED_SERVER;

      int KA_wait_time;

      Message *PBNM_TimerMsg[5]; /*PBNM_TimerMsg[0] [1] [2] [3] [4]
correspond to COPS KA msg timer, connection timeout timer, PBNM_SDP
timer, Statistics timer, SDP FORWARDING Timer respectively */

      KA_DATA *KAdata;

      clocktype serviceAvailability; //To collect total time a client
was being managed
      clocktype SimTime;
      clocktype COPS_Connection_Timer;
      int numCOPSconnections;
      int sessionTimeout;

      PBNM_SDP_INFO *addr_list;

      BOOL COLLECT_DATA;
};
```

Figure 3.2.1.2 Modifications done to the *struct_pbnm_adhoc_str* structure.

The modifications made to the structure are shown in **bold**. As shown in Figure 3.2.1.2, all the performance metrics and variables associated with the nodes in the system are stored in this structure.

d) *application.c*: This file, along with the *application.h* header file provides a means to tie up application level metrics with the simulator. Usually run-time parameter definitions are specified in *.app* files and these need to be associated with nodes or devices in the system. The *application.c* file provides *APP_Initialize( )* and *APP_InitializeApplication( )* functions to achieve this. In addition to this, it handles the application layer specifics (achieved by the *APP_ProcessEvent( )*) of the simulation scenario.

e) *app_util.c*: This file, along with the *app_util.h* header file, is used to register new application layer protocols with the simulator. It is also used to support interaction between the application layer and the underlying transport layer.

f) *pbnm_adhoc.app*: The run-time parameters of the simulation model are specified in the *.app* file. In our model, the values of the KA timer, the service discovery timers, the cluster size and the designation of nodes as client and server are set using the *pbnm_adhoc.app* file.

g) *pbnm_adhoc.config*: This file stores the configuration information of all the layers of the protocol stack for all the nodes in the simulation scenario. The information in this file is loaded by the *application.c* file using the initialization functions.

The *node.h* file is modified to include three new fields essential for gathering topology information from the network layer (Section 3.2.2, Figure 3.2.2.1). There are other files like the *node.c* and *ip.c* that were modified trivially to ensure correct functionality. In the subsequent sub-section, we discuss the simulation environment in detail.

3.2.2 Simulation Environment

The simulation environment used for our PBNM system implementation in QualNet is explained as follows. A 1000 x 1000 sq. meter flat area was considered for simulations with 20 nodes being uniformly dispersed in this area. All the nodes were compliant with the 802.11b MAC layer standard with a data rate of 11 Mbps and a transmit power of 15 dBm. With a statistical path model and a two-ray path loss model, a radio range of 370 meters was achieved. Out of the 20 nodes chosen for simulation in each scenario, two of the nodes (namely, node 2 and node 8) were specified as servers. This was done by specifying the node configuration in a QualNet configuration file called *pbnm_adhoc.app*. The server assignment can be random without noticeable impact on the outcome of the simulation experiments. The mobility model used to characterize the movement of mobile nodes is the random waypoint mobility model [7]. According to this mobility model, a node initially remains stationary for a period specified by the pause time. It then moves towards a randomly chosen destination with constant speed given by a sample of a random variable uniformly distributed between [V_{min}, V_{max}]. V_{min} is the minimum possible speed value chosen by a node and V_{max} is the maximum possible speed value chosen by a node. When the node reaches its chosen destination it again waits for an interval specified by the pause time before choosing a random destination and speed again. This mobility pattern repeats for the entire simulation period for all nodes. As in [53], the V_{min} value is chosen to be 80-90 % of the maximum speed value of V_{max}. This helps in avoiding the average speed decay problem [75] and helps the network achieve a stable state in an acceptably low amount of time (approximately 500 seconds).

Each set of simulation experiments consisted of 100 runs of a scenario. Each simulation experiment was run for 1500 seconds and repeated for 100 such runs to form a set. Over every 10 of these 100 simulation runs, 10 different seed values were chosen from 10 different configuration files to ensure randomness in the initial node placement and mobility. The method of batch means [36] was used to obtain an acceptably tight confidence interval (within ±5% of the reported average value) for the parameters of interest. All the experiments were repeated for cluster size varying from k = 1 to k = 5. The parameters gathered using these simulations are the following; individual and average values for rates λ_1 and $\lambda_2(1-\pi_1)$ reported by all the clients over a single batch experiment (1500 simulation seconds), the estimate of the average probability π_1 of moving into a new cluster immediately (collected over one simulation run) and the individual and average arrival rate of movement of clients inside a cluster (irrespective of whether they move back into a new cluster immediately or not) collected over one simulation run. We also gathered the service availability metric for comparison with the results obtained from models analyzed in Chapters 4 and 5. Table 3.2.2.1 summarizes the default values of simulation parameters. All other metrics like the pause time, the KA timer value, the minimum and maximum mobility speeds and cluster size are variables and are changed to have different simulation scenarios. As shown in Table 3.2.2.1, Optimized Link State Routing protocol is used as the mobile ad hoc network routing protocol.

Table 3.2.2.1 Default values of simulation parameters

Simulation Parameter	**Value**
Simulation Time	1500 seconds
Mobility Model	Random Way-Point Model
Node placement	Uniform
Simulation Area	1000 x 1000
Propagation Model	Statistical
Path-loss Model	Two-ray path loss Model
Radio range	370 meters
Physical Model	Physical 802.11b
Data Rate	11 Mbps
Transmission power	15 dBm
Antenna Model	Omni-directional
MAC layer protocol	MAC 802.11b
Network layer protocol	IP
Ad Hoc Routing protocol	OLSR
PBNM service discovery timer (server)	20 seconds
PBNM service discovery timer (client)	27 seconds
Node Ids of server nodes	Node 2, Node 8

The OLSR implementation in QualNet is incorporated in two files *olsr-inria.c* and *olsr-inria.h*. We modified the protocol implementation for OLSR in QualNet in order to gather topology information. The modification involved passing the routing layer information from the network layer up to the application layer. This is necessary to determine if the client is inside the k-hop cluster or not. Using this information, various timing parameters are computed to subsequently determine λ_1 and λ_2 and the probability π_1 of immediate movement in a new cluster. This was achieved by adding a function called *find_DistFromServer( )* in *olsr-inria.c*. This function takes the individual client node's *node* data structure as a parameter and returns the client node's hop distance from the server in the *hop_Distance* field of the node data structure (Figure 3.2.2.1). We modified the original data structure implementation to include three variables, viz. the current server ID, the hop distance between the client and the current server and the ID of the previous connection of this client node. The modified node data structure (defined in *node.h*) is shown in the code snippet in Figure 3.2.2.1 below:

```
/*
 * The Node structure.
 * This struct includes all the information for a particular node.
 * State information for each layer can be accessed from this
structure.
 *
 * typedef to Node in main.h.
 */
struct struct_node_str
{
    /* Information about other nodes in the same partition. */
    Node      *prevNodeData;
    Node      *nextNodeData;

    /* nodeIndex will store a value from 0 to (the number of nodes -
1);
       each node has a unique nodeIndex (even across multiple
partitions).
       A node keeps the same nodeIndex even if it becomes handled by
       another partition.  nodeIndex should not be used by the protocol
       code at any layer. */
    unsigned       nodeIndex;

    NodeAddress    nodeId;        /* the user-specified node identifier
*/

    unsigned short seed[3];       /* seed for random number generator
*/
    unsigned short initialSeedValue[3]; /* initial seed value for a
node */
    long           numNodes;      /* number of nodes in the simulation
*/
    SplayTree splayTree;

    clocktype* currentTime;
    BOOL          packetTrace;
    BOOL          packetTraceSeqno;

    BOOL          guiOption;

    PartitionData *partitionData;

    int           numberChannels;
    int           numberPhys;
    int           numberInterfaces;


    MobilityData   mobilityData;

    /* Layer-specific information for the  node. */
    PropGlobalData* propGlobalData;
    PropData*       propData;
    PhyData**       phyData;                      // phy layer

    // Users should not modify anything above this line.

      /*Node structure modified: Aniket Bhat*/
      char curr_Server[MAX_STRING_LENGTH];// string to store the node
      address
      unsigned short hop_Distance;//field to store the distance from
      the current server
      int prevConnId;
      /*Modification Ends*/

    MacData**       macData;                      // MAC layer
    MacSwitch*      switchData;                   // MAC switch
    NetworkData     networkData;                  // network layer
    TransportData   transportData;                // transport layer
    AppData         appData;                      // application layer
    TraceData       *traceData;                   // tracing
};
```

Figure 3.2.2.1 Code snippet defining the node structure in QualNet (*node.h*)

We now present the results of running various simulation scenarios in Sections 3.3 and 3.4.

3.3 Basic Scenarios

In this section, we discuss four basic simulation scenarios. The basic scenarios correspond to a pause time value of 10 seconds and KA timer value of 50 seconds. All other parameters are set to their default values as given in Table 3.2.2.1. We vary the speed of the mobile nodes in each of the four scenarios. Scenario I corresponds to a mobility model with a V_{min} value of 4 m/s and V_{max} value of 5 m/s. In scenario II, V_{min} is set to 9 m/s and V_{max} is set to 10 m/s. For scenario III, we set V_{min} to 18 m/s and V_{max} to 20 m/s. Scenario IV is a special case wherein the nodes are stationary. For each of the four scenarios, we vary the cluster size from a value of k = 1 to k = 5. We plot the curves for variation of the average values of λ_1, $\lambda_2(1-\pi_1)$, the estimate of the probability π_1 and the percentage service availability. The plots also show the 95% confidence intervals for each measurement.

3.3.1 Effect of Mobility and Cluster Size on Service Availability

Service availability is measured by varying the cluster size from k = 1 to k = 5 for each simulation set of 100 runs. The effect of varying the speeds of policy nodes on service availability is also studied. Figure 3.3.1.1 shows the plot of service availability as a function of mobility and cluster size. The mobility parameters of the nodes are varied so that service availability is captured at three different values of V_{max} (maximum node speed) of 5 m/s, 10 m/s and 20 m/s. The V_{min} value in each case is kept to 80-90% of the maximum speed. We also considered a special case where in the nodes have no mobility. In this case, however all the parameters of interest were zero as there was no movement inside or outside the cluster and the initial policy distribution was retained throughout the experiment. As expected, the system typically performs well at lower values of mobility (speed of mobile nodes), however we observe an anomaly for the value of cluster size, k =1. At value of k=1, the service availability in-fact increases with increase in the speeds of the mobile nodes.

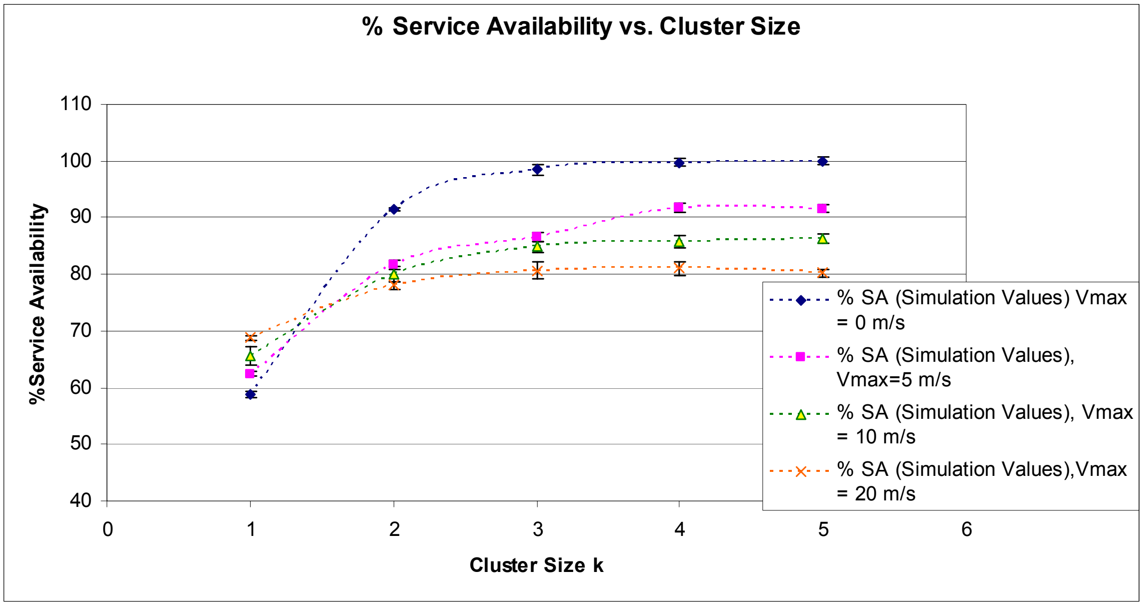


Figure 3.3.1.1 Effect of mobility and cluster size on service availability

This interesting behavior is explained in [53, Chapter 7]. Also, in general, the service availability improves with increase in the cluster size k.

3.3.2 Effect of Mobility and Cluster Size on λ_1

Here we study the effect of mobility and cluster size on the rate at which clients move out of a cluster. This, as stated before, is representative of the time for which the clients remain in service once they move into a new cluster. Figure 3.3.2.1 shows the effect of varying the mobility values and cluster size on the average value of λ_1.

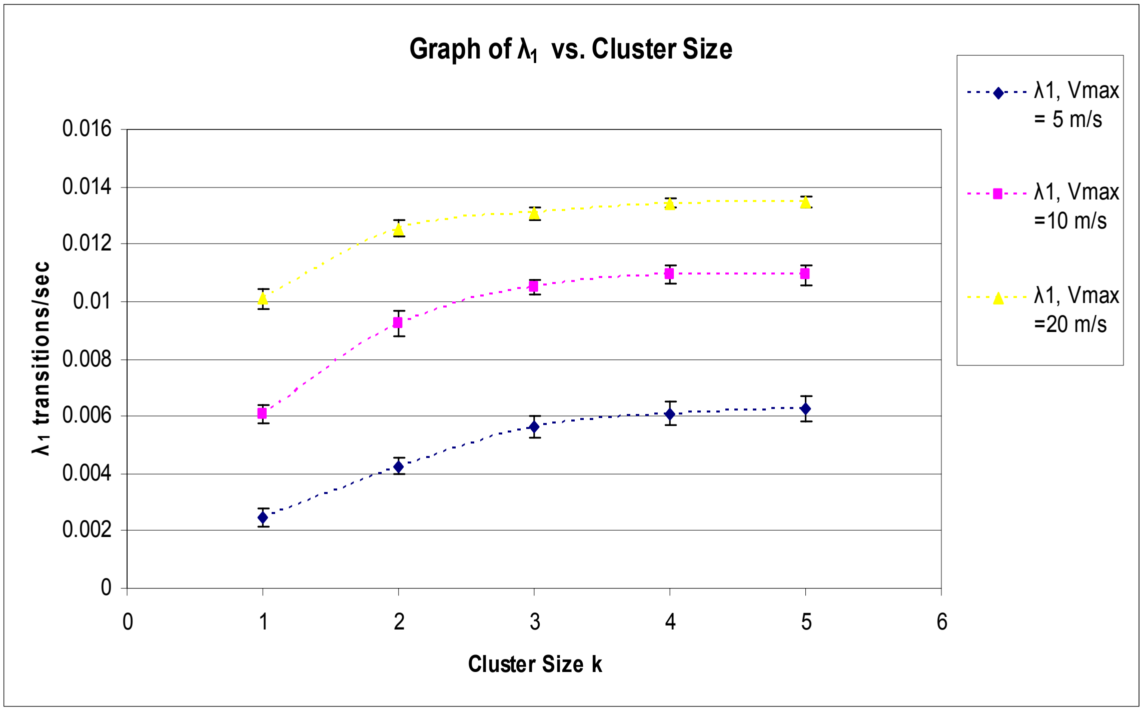


Figure 3.3.2.1 Effect of mobility and cluster size on λ_1

As can be seen from the above graph, the value of λ_1 generally increases with the cluster size. Also, with an increase in the mobility, λ_1 seems to increase. This is because with increase in the speed of the nodes, they move quickly in and out of the k-hop clusters and spend less time within a cluster. This effect also translates into service availability decreasing with increase in mobility, which is verified before.

3.3.3 Effect of Mobility and Cluster Size on $\lambda_2(1-\pi_1)$

As explained before, the way the simulation code was written, λ_2, the rate at which clients move back into a new cluster, cannot be obtained directly. What we have instead is the product $\lambda_2(1-\pi_1)$. We study the effect of varying mobility and cluster size on this product. Figure 3.3.3.1 shows a graph of $\lambda_2(1-\pi_1)$ against cluster size for different values of maximum speed of mobile nodes.

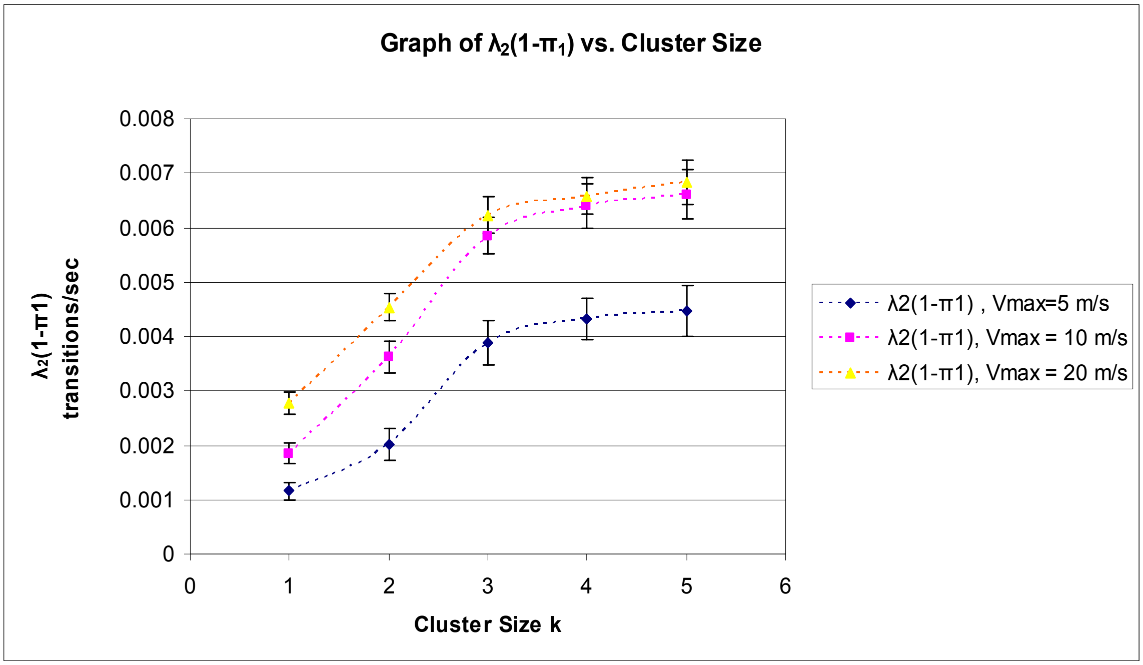


Figure 3.3.3.1 Effect of mobility and cluster size on the product $\lambda_2(1-\pi_1)$

These results match our intuition that increasing the mobility of the nodes increases the product value. This is because the clients stay out of service for a short duration of time and are eventually covered in some other cluster.

3.3.4 Effect of Mobility and Cluster Size on π_1

This sub-section studies the effect of varying speed of mobile nodes and cluster size k on the estimated value of probability π_1 that the clients move into a new cluster immediately after moving out of their current clusters. Figure 3.3.4.1 shows the plot of the estimate of probability π_1 as a function of mobility and cluster size.

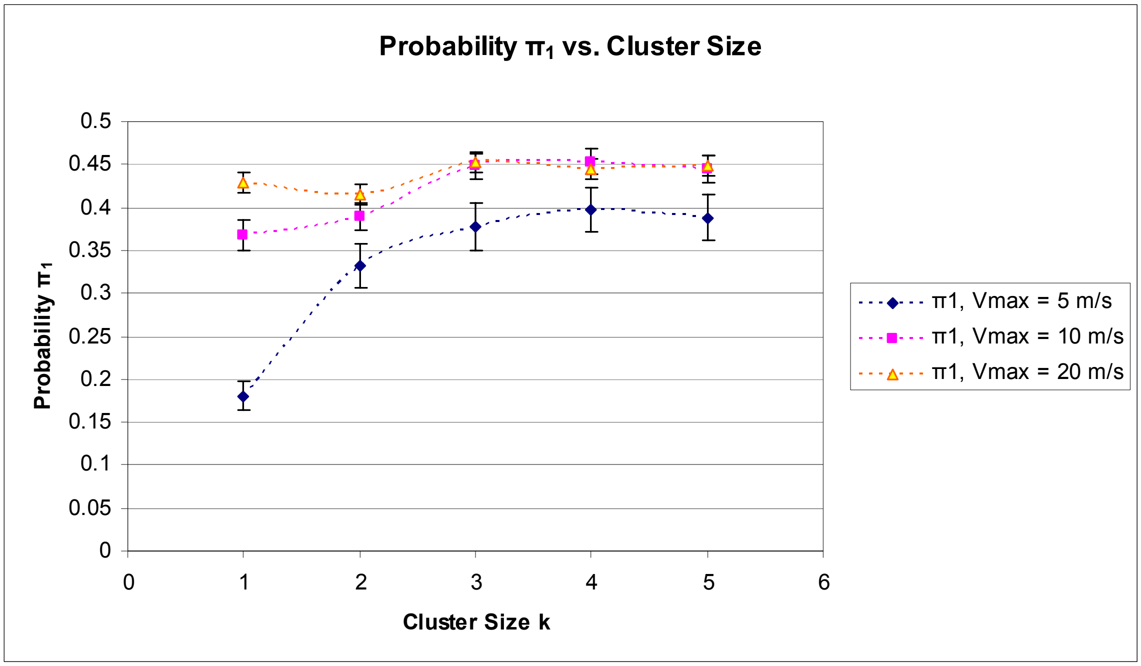


Figure 3.3.4.1 Effect of mobility and cluster size on estimated π_1

The estimated value of probability π_1 is shown to increase with the increase in mobility and cluster size. This is because, with the increase in the cluster size, the probability of the client getting covered in some adjacent cluster increases. With increase in node speeds, the clients move into new clusters immediately and hence the estimated value of π_1 is also seen to increase with mobility. At higher mobility scenarios like V_{max} = 20 m/s, the estimated values of probability π_1 does not seem to vary much. In these cases, mobility of nodes dominates their behavior (as against the size of clusters) and seems to help them enter new clusters approximately with the same probabilities irrespective of the cluster size.

3.3.5 Histograms

In this section we provide histograms for the instantaneous values of the parameters collected for all the nodes in the system individually. The objective is to obtain a better understanding of the probability distribution of these parameters. We discuss a case where cluster size is set to 3, pause time is set to a value of 10 seconds, the mobility value is set to V_{max} = 10 m/s and the KA timer value is 50 seconds. There are three curves that are discussed: the histogram of the time between consecutive arrivals of a client to a cluster, the histogram of the time spent by clients in a cluster (reciprocal of instantaneous λ_1 values) and the histogram curve for time spent by clients outside any cluster (reciprocal of instantaneous values of $\lambda_2(1-\pi_1)$).

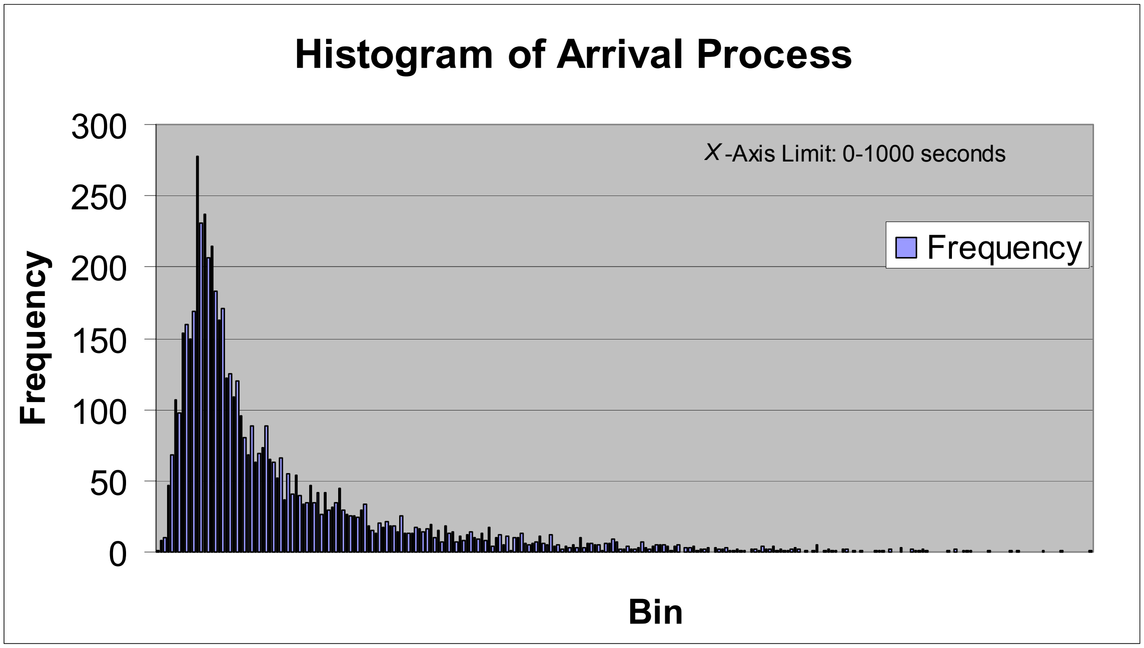


Figure 3.3.5.1 Histogram of inter-arrival times of clients inside a cluster

Figure 3.3.5.1 shows the histogram of inter-arrival times of clients to a cluster. It is clear that these inter-arrival times do not follow an exponential distribution. We tried curve-fitting using the MINITAB Release 14 for Windows.[‡] This software has support for several standard probability distributions. However, the plot obtained does not fit any of the standard distributions and the inter-arrival times for clients entering a cluster are therefore classified as generally distributed.

In Figure 3.3.5.2, we plot a sample histogram for the reciprocal of the rate at which clients depart their cluster (for the same scenario used for Figure 3.3.5.1). This parameter is representative of the time spent by clients inside a cluster, and again follows a general distribution. It was noted that the first peak in this histogram occurs approximately

[‡] MINITAB is a statistical analysis software developed by Minitab Inc.

around the time corresponding to the KA timer value of 50 seconds. In Section 3.4, we provide conclusive evidence that the first peak is indeed dependent on the value of the KA timer. When the KA timer expires, the clients perceive that they have lost the connection with the server and hence are out of the cluster. On a probabilistic basis, approximately one-third of clients are seen to fall out of their respective clusters due to the expiration of the KA timer.

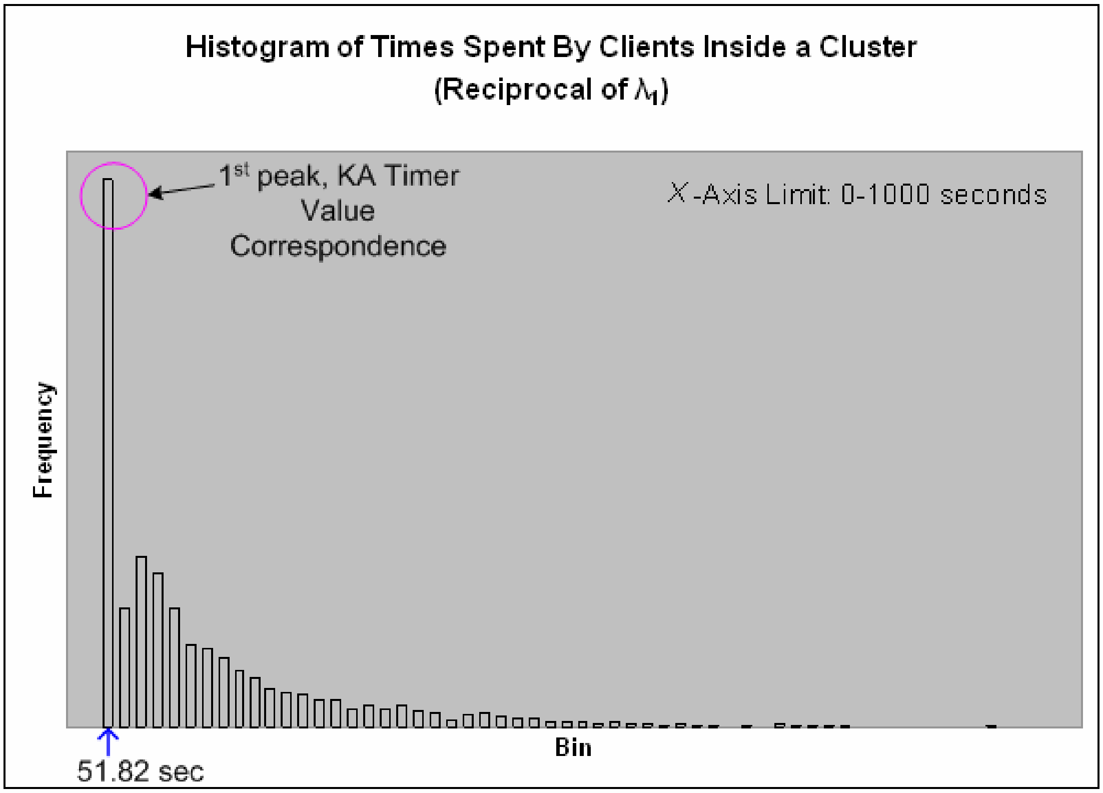


Figure 3.3.5.2 Histogram of time spent by clients inside a cluster

Figure 3.3.5.3 shows the histogram of the time spent by the clients outside any cluster. Again the sample scenario is the one explained at the beginning of the subsection 3.3.5. As can be seen, the times spent by the clients outside the cluster are approximately uniformly distributed. One reason for observing this uniform distribution of random variable associated with the time spent by clients outside the cluster is a possible correlation between the uniform distribution of random speeds chosen by nodes and the possibility of finding a server. Also note that all nodes are initially uniformly placed within the area of interest and follow the same mobility model. We incorporate this uniform behavior in developing a more detailed GDT_SPN model of the PBNM system. Note that the GDT_SPN model is essentially the same model as discussed in Section 3.1 but with system processes being represented by the behavior observed on performing simulation-based analysis.

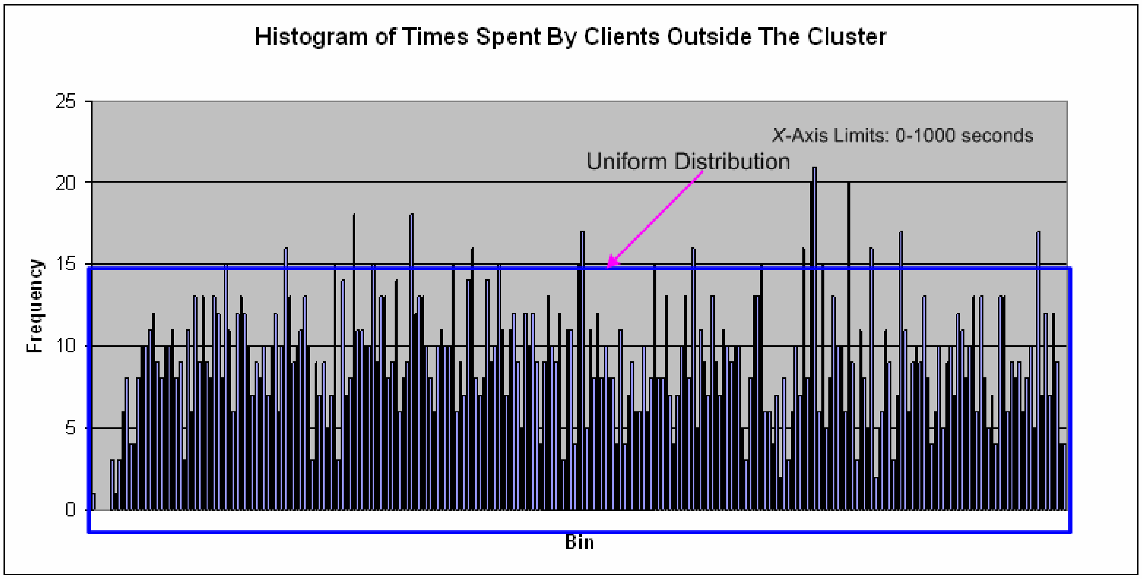


Figure 3.3.5.3 Histogram of time spent by clients outside a cluster

Also, note that as stated earlier this is a histogram of the reciprocal of $\lambda_2(1-\pi_1)$. However, since the probability π_1 is just a scalar value, the nature of the λ_2 plot remains the same although the actual values may be scaled.

3.4 Study of variation of KA Timer and Pause Times

We proceed to study the effects of varying simulation parameters like the KA timer and pause time on the system behavior. The KA timer value was varied between KA = 10 seconds and KA = 60 seconds in steps of 10 seconds. In another set of experiments, the pause time parameter was varied between a value of PT = 0 seconds and a PT value of 20 seconds in steps of 5 seconds.

For the experiment in which we study the effect of variation of the KA timer on the parameters of interest, we tabulate the service availability obtained for different values of KA timer at a pause time value of 10 seconds for the case where V_{max} = 5 m/s in Table 3.4.1. The service availability values do vary somewhat from the basic scenario; however, the occurrence of the first peak (Figure 3.3.5.1) for the times spent in the cluster seems to be dependent directly on the KA timer value. The ratio of the number of connections lost due to time-outs of the KA timer to the total number of connections lost is tabulated in Table 3.4.2, for different cluster sizes and KA timer values.

Table 3.4.1 Variation of service availability with KA timer values

Pause Time = 10 s					
	KA =10s	KA =20s	KA =30 s	KA =40s	KA = 60s
k	% SA	% SA	% SA	% SA	% SA
1	58.72	68.18	76.23	81.991	89.22
2	70.22	77.09	82.18	86.81	91.67
3	73.77	80.21	85.22	88.76	92.58
4	74.83	80.91	85.73	89.34	93.03
5	74.5	81.39	85.747	89.74	93.13

Table 3.4.2 Number of client connections lost due to KA timer time-out

Pause Time = 10 s					
	KA =10 s	KA = 20s	KA =30s	KA =40s	KA =60s
k	Number of time-outs at approx. 10 seconds/Total connections lost	Number of time-outs at approx. 20 seconds/Total connections lost	Number of time-outs at approx. 30 seconds/Total connections lost	Number of time-outs at approx. 40 seconds/Total connections lost	Number of time-outs at approx. 60 seconds/Total connections lost
1	365/3976	223/1340	178/826	163/661	112/426
2	1464/8767	800/4546	487/2453	342/1466	201/779
3	2432/11343	1346/6250	730/3599	508/2381	324/1221
4	2797/12246	1519/6743	990/4120	580/2637	316/1310
5	2838/12354	1662/6934	972/4226	577/2602	351/1411

As can be seen from Table 3.4.2, a considerable number of connections are lost due to time-outs of the KA timer, which explains the peak occurrence in the histogram in Figure

3.3.5.2. Also, at lower values of the KA timer, more connections are lost. This is because, at lower values of KA, more clients get timed out and lose their connections than at higher values. However the time out value should not be large enough that clients having already lost service realize it only some time later.

We also studied the variation of service availability and other curves with respect to the change in pause times in the random way-point mobility model. The variation in pause time did not show any appreciable change in the service availability or in the nature of the histograms presented in the previous section.

3.5 Summary

In this Chapter, we focused on deriving metrics needed to parameterize our PN models (Chapters 4 and 5) by using QualNet simulations. We identified three parameters of interest: the rate of at which clients leave their respective k-hop clusters, the rate at which clients join a new cluster, having not done so immediately (Section 3.1) and the probability of moving back to a new cluster immediately after moving out of the current cluster. Empirical averages obtained through the simulations are used as inputs to the GSPN and GDT_SPN models analyzed in Chapter 4 and Chapter 5, respectively. On a general note, modeling of distributed systems is a complex process that requires careful analysis and choice of appropriate metrics for parameterizing the developed model.

Chapter 4

Markovian Analysis

Most analytical models for network performance assessment have relied on the assumption that the times between consecutive events of the same kind are distributed exponentially (thereby exhibiting the memory-less property). This chapter is concerned with the analysis of the Petri Net model for our policy based network management system from a purely Markovian perspective. Exponentially-distributed random variables are associated with all timed transitions, while switching probabilities are associated with all immediate transitions.

Chapter 2 provided us with the necessary background on the methodology for modeling systems with Petri Nets and Chapter 3 discussed the collection of empirical parameters of interest for the PBNM system. This chapter applies the theory presented in Chapter 2 to analyze and validate the simple GSPN model (Section 3.1) for the system described in Section 1.2 (Chapter1). The results obtained in Chapter 3 are used to parameterize our model. We also express the model as a network of queues (Sections 4.2 and 4.3) and utilize results from that analysis to validate results obtained by analysis of the GSPN model using SPNP [67] in Section 4.4. In Section 4.4, we also motivate the need for a comprehensive and exact analysis of the system using a more detailed Petri Net model. A chapter summary is presented in Section 4.5.

4.1 Generalized Stochastic Petri Net Model For Clients in PBNM system

The policy based network management system has been described in Section 1.2 (Chapter 1). It is a client-server system that consists of a pre-assigned number of server nodes and client nodes in a MANET. The servers are the policy decision points (PDPs) while other nodes retrieve their policy specifications from the PDPs and are called the policy enforcement points (PEPs) or clients. PEPs are the nodes where the policies specified actually get enforced and executed. In the suggested PBNM protocol solution suite, the common open policy service with provisioning (COPS-PR) [8] is used for deploying and distributing policy information to nodes in the network under the policy framework. The client nodes being served by the PDPs in the MANET can stop receiving service if they move out of the service range of the servers. The k-hop clustering algorithm discussed in [53, 54] can be used to completely describe such client behavior. In the k-hop clustering algorithm, a policy server maintains COPS connections with clients that are within k-hops from the server. The server can set the value of k either during service advertisement or while establishing the COPS connection with the client. The main motivation for such a clustering algorithm is to minimize the number of hops between the policy clients and servers. With this algorithm in place, a client can be in one of two states:

- State 1: the client is covered within a cluster created by one of the servers, in which case it is in service; or
- State 2: the client is not covered within k hops of any of the policy servers, in which case it is looking for service.

Figure 4.1.1 shows a state transition diagram for the behavior of clients in the PBNM system. It consists of two states, State 1 and State 2 as explained above. The values adjacent to the transition arcs show the transition probabilities. The values adjacent to the states show the rate of transition of clients out of those states.

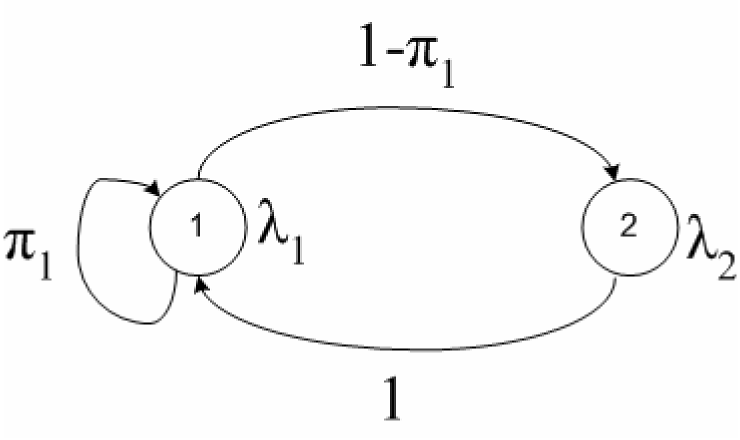
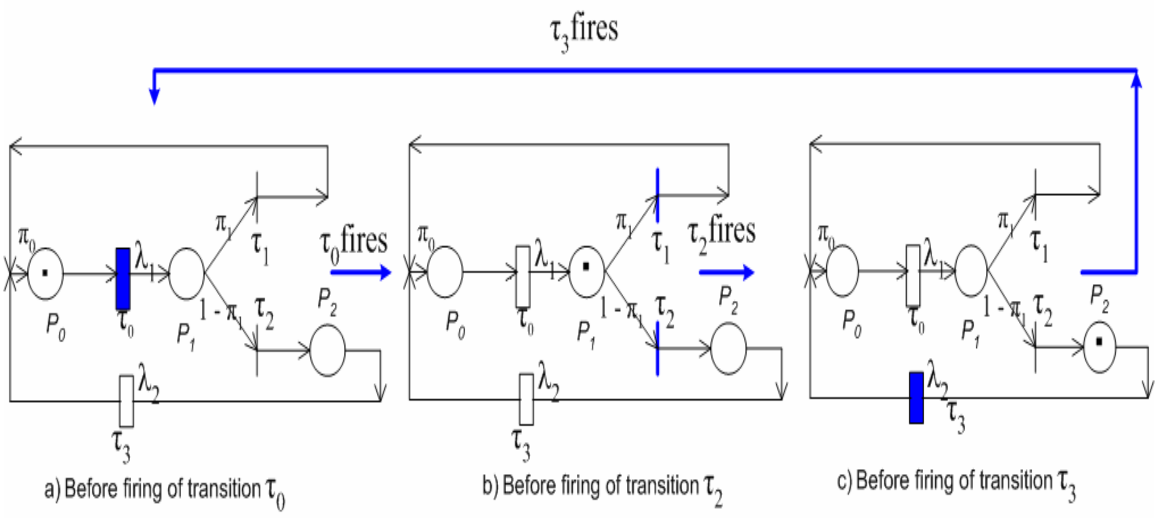


Figure 4.1.1 State diagram for clients in the PBNM system

In Section 3.1 (Chapter 3), we introduced the GSPN model (under Markovian assumptions) for clients in the PBNM system. This model, shown in Figure 3.1.1, serves as the reference model for the Markovian analysis presented in this chapter. Here, we present the dynamics of the GSPN model and describe the client behavior. Figure 4.1.2 shows the evolution of our GSPN model with time.



a) Before firing of transition τ_0 b) Before firing of transition τ_2 c) Before firing of transition τ_3

Figure 4.1.2 Evolution of GSPN model for the PBNM system with time

As shown in Figure 4.1.2.a, the GSPN starts with an initial condition wherein there is a single token in place P_0. When transition τ_0 fires, it removes a token from place P_0 and deposits it into place P_1. The presence of a token in place P_1 is a transient state for the GSPN, shown in Figure 4.1.2.b. When this happens, two immediate transitions, τ_1 and τ_2, are enabled. Either of them can fire according to the switching probabilities. If transition τ_1 fires while in the state represented in Figure 4.1.2.b, the token gets deposited back into place P_0. This represents the case that the client moved out of the current cluster but moved immediately into a new cluster. Note that this occurs with a probability π_1 and is not shown in Figure 4.1.2. We focus on the other possibility, wherein the transition τ_2 fires, removing the token from place P_1 and depositing it into place P_2. This results in the state of the GSPN shown in Figure 4.1.2.c. This state of the GSPN represents a case where the client, once out of service, does not get immediately covered by another cluster and hence remains out of service for a non-zero time.

The evolution of the GSPN with time can be more efficiently represented by a reachability graph (Chapter 2, Section 2.1) as shown in Figure 4.1.3.

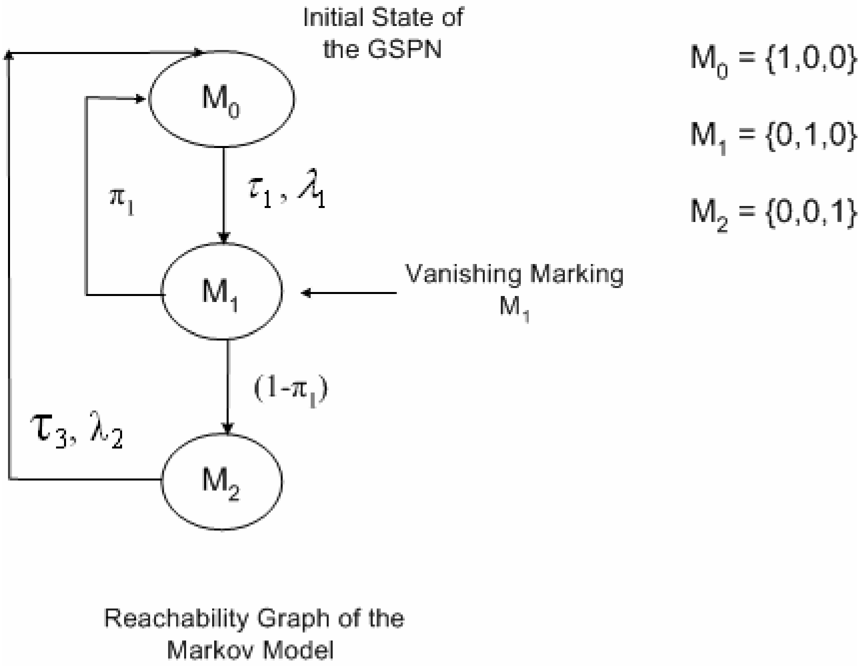


Figure 4.1.3 Reachability graph for GSPN model of the PBNM system

The reachability graph shown in Figure 4.1.3 has three markings: the initial marking $M_0 = \{1,0,0\}$; the marking $M_1 = \{0,1,0\}$, where the client remains for a vanishingly small period of time before it either moves into a state wherein it is immediately covered in another cluster (back to marking M_0 with probability π_1) or to a state wherein it is looking for new service (marking M_2 with probability $(1-\pi_1)$). Finally, the marking $M_2 = \{0,0,1\}$

represents the state where the client is looking for service from one of the servers in the PBNM system.

4.2 Equivalent Network of Queues Model

The GSPN model shown in Figure 3.1.1 can be analyzed using an equivalent closed network of queues representation [32]. Under Markovian assumptions, the network of queues model admits a closed-form solution. Analytical results from this model are useful in validating our simulation results; an example is the percentage service availability graph (Figure 4.4.1), discussed later in this chapter. The network of queues model is shown in Figure 4.2.1. It consists of two queuing systems q_0 and q_1.

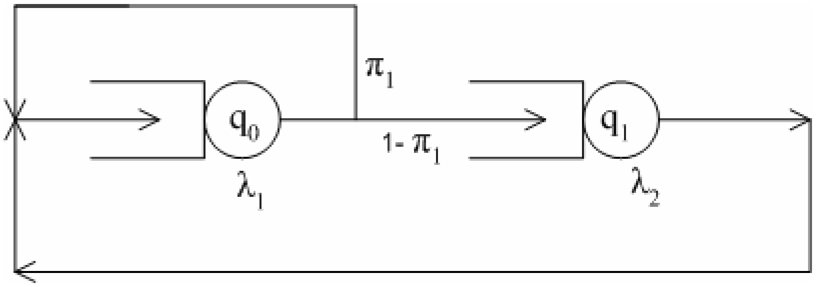


Figure 4.2.1 Network of queues model for policy client in the PBNM system

The first queuing system q_0 represents the period for which the client is in service. Thus the service time in q_0 is analogous to the timing associated with timed transition τ_0. When a client departs this queuing system, it represents the transition from place P_0 to place P_1. The client, upon moving out of the current cluster, is either covered in some other cluster (with a probability π_1) or else remains “out of service” and is represented by the client entering queuing system q_1. Thus the time spent by a client in queuing system q_1 represents the time for which it remains out of service. The service rate of q_1 is the firing rate of transition τ_3, i.e. λ_2.

4.3 Mathematical Solution with Markovian Assumptions

In this section, we derive, from the closed network of queues system shown in Figure 4.2.1, a mathematical expression characterizing service availability in the PBNM system. Let us consider the two queuing systems q_0 and q_1, indicating that the client is "in service" or "out of service," respectively. We are interested in finding the expression for service availability to a particular client in the PBNM system. We solve the system using the Gordon-Newell Algorithm for closed network of queues [21, 32 and 64].

We define the following parameters:

μ_0 = service time of queuing system q_0;

μ_1 = service time of queuing system q_1;

v_0 = number of visits to q_0;

v_1 = number of visits to q_1;

N = total number of times a client arrives into the queuing system.

Let the total service demand on queues q_0 and q_1 be denoted by D_0 and D_1 respectively.

From Figure 4.2.1, we have, $v_0 = N$ (4.3.1)

$\therefore \quad v_1 = (1-\pi_1).N$ (4.3.2)

By the definition of service demand, $D_0 = \mu_0 v_0$ and $D_1 = \mu_1 v_1$ (4.3.3)

$\Rightarrow \quad D_0 = \mu_0 N$ and $D_1 = \mu_1(1-\pi_1).N$ (4.3.4)

Following the Gordon Newell algorithm, we determine a constant α that normalizes the service demand on the two queues.

$$\alpha = \frac{1}{\frac{1}{M}\sum D_i};$$

where M is the number of queues, which in this case is 2. This gives

$$\alpha = \frac{2}{\mathbf{N.\mu_0} + \mathbf{(1 - \pi_1)}.\mathrm{N}.\boldsymbol{\mu}_1}$$

Let y_i represent the normalized or the scaled service demand on queue i. Then,

$y_i = \alpha D_i$

Thus the normalized service demands on queuing systems q$_0$ and q$_1$ are given by Equations 4.3.5 and 4.3.6, respectively,

$$y_0 = \frac{2N\mu_0}{N.\mu_0 + (1 - \pi_1).N.\mu_1}; \qquad (4.3.5)$$

$$y_1 = \frac{2N\mu_1(1-\pi_1)}{N.\mu_0 + (1 - \pi_1).N.\mu_1} \qquad (4.3.6)$$

Table 4.3.1 lists the product terms required for obtaining the normalizing constant $G(N)$.

Table 4.3.1 Tabulation of closed network of queues model

Number of clients in queue		**Product**
q$_0$	**q$_1$**	$\prod_{i=0}^{M-1} y_i^{n_i}$
1	0	$y_0^1.y_1^0 = y_0$
0	1	$y_1^1.y_0^0 = y_1$
$G(N) = y_1 + y_0$		(4.3.7)

$$\therefore p(1,0) = \text{\% service availability (SA)} = \frac{y_0.y_1}{G(N)} \qquad (4.3.8)$$

From equation 4.3.7, we get, $G(N) = 2$ and hence from equation 4.3.8,

$$p(1,0) = \frac{\mu_0}{\mu_0 + (1 - \pi_1).\mu_1} \qquad (4.3.9)$$

From our network of queues model explained in Section 4.2, we have,

$\mu_0 = \frac{1}{\lambda_1}$ and $\mu_1 = \frac{1}{\lambda_2}$

Thus, the service availability equation evaluates to

$$p(1,0) = \frac{\lambda_2}{\lambda_2 + (1 - \pi_1).\lambda_1} \qquad (4.3.10)$$

Thus, the percentage service availability under Markovian assumptions is given by Equation 4.3.10.

4.4 GSPN Model Validation

This section deals with the numeric-analytic solution and validation of the GSPN model we developed for our PBNM system. We discuss the choice of tools possible for analysis of the model. We discuss the use of the empirical results described in Chapter 3 to parameterize our model. Finally, we conclude with validation of our model using analytical expressions obtained for the closed network of queues in section 4.3.

4.4.1 Implementation Tools and Methodology

There are a plethora of tools available for model specification and solution of Stochastic Petri Nets. All of these tools provide an interface to define the specifics of the model and provide a platform to define and capture the behavior of the system for which the model is created. Most of these tools work on either the Microsoft Windows platform or some flavor of the UNIX operating system (Sun, Linux, Free BSD, etc.). The stochastic Petri net package (SPNP) [67], SPN2MGM, POSES++, and EDS Petri Net Tool are some of the tools [70] that can be used to analyze Stochastic Petri Nets under Markov assumptions. Other more advanced modeling tools like TimeNET, GreatSPN [70], WebSPN [57] and ESP [13] are also available and they allow modeling of non-Markovian Petri Nets. In this phase of our work, we chose the SPNP tool for the analysis of the GSPN model that we developed.

SPNP is a widely used modeling tool that can model systems in which transitions are either immediate, deterministic or have exponentially-distributed random variables associated with them. User-defined system models are solved on the basis of stochastic reward net theory using efficient and numerically stable algorithms. There is limited support for other probability distributions such as log-normal, Pareto, truncated Cauchy, Weibull, uniform, Erlang, and hyper-exponential. SPNP uses the C-based model description language CSPL for model specification and solution. A newer version of

SPNP (SPNPv6) also provides a graphical interface called iSPN that makes development of complex models easier and quicker. A number of important PN constructs such as marking dependency, variable cardinality arc and enabling functions (or guards) are supported to facilitate the construction of models for complex systems. SPNP also provides the ability to solve Stochastic Reward Net model [10] for obtaining either steady-state or transient results. SPNP can be used to accurately model Petri Nets that do not extend in complexity beyond the Generalized Stochastic Petri Nets. This makes it a natural choice for modeling and analyzing the simple GSPN model we developed for the PBNM system.

We followed the two-step methodology described below to validate our GSPN model:

- specify the GSPN model using SPNP and analyze the results for service availability.
- compare the results obtained using SPNP and the ones obtained through mathematical formulation (Equation 4.3.10).

In addition to the two step validation of the GSPN model, we also compare the service availability seen by policy clients obtained using four different methods.

1. Using QualNet simulations: Service availability is obtained using QualNet simulations in addition to the collection of the statistics of interest (discussed in Chapter 3).
2. Conducting Test-bed experiments: Service availability is also obtained by conducting experiments on the test bed [53].
3. Analysis of GSPN model using SPNP: The percentage service availability is obtained by estimating the probability π_0 of the GSPN model specified in SPNP (Section 4.4.2.2).
4. Analytical Formulation: The mathematical values for percentage service availability are obtained by substituting the average values of the parameters of interest in Equation 4.3.10.

The validation of the GDT_SPN model (essentially the same model as introduced in Section 3.1 but with added complexity to incorporate probability distributions obtained using QualNet simulations) and the comparison of the service availability results obtained using the above four methods against the ones obtained using the WebSPN analysis of the GDT_SPN model is done in Chapter 5.

4.4.2 Empirical Parameterization of the GSPN Model and Results

The parameterization of the GSPN model i.e. obtaining the average values of rates λ_1 and λ_2 (Chapter 3, Section 3.1) and the estimation of the probability π_1 is done using QualNet simulations explained in Chapter 3. The simulations to gather empirical data were done using the QualNet™ network simulator. The underlying OLSR routing protocol implementation in QualNet was modified to gather service information. We used the pre-existing PBNM model [53, 54] and modified it to run the simulations for 100 runs for each cluster size. The effect of varying various parameters like speed, pause times and keep-alive timer values on the service availability and on the parameters of interest viz. λ_1, λ_2 and π_1 was studied (Chapter 3). This was done in each case for cluster sizes varying from k =1 to k = 5 in a 1000 x 1000 square meter area. A set of 20 uniformly distributed nodes is considered, two of which are chosen to be servers i.e. PDPs. We used the commonly deployed random waypoint mobility model. The mobility parameters were set to account for the speed-decay problem in the mobility model in each case. The pause times were varied from 0 seconds to 50 seconds and the keep-alive (KA) timer value was varied from 10 seconds to 60 seconds and the effect on the three parameters of interest was studied.

Table 4.4.2.1 Simulation parameters gathered for a sample scenario with V_{max} = 10 m/s.

k (cluster size)	λ_1 (transitions/sec)	$\lambda_2(1-\pi_1)$	π_1	λ_2 (transitions/sec)	% SA (QualNet Simulation Values)
1	0.006051	0.001846	0.366821	0.00291545	65.6
2	0.00925	0.003616	0.389583	0.00592382	80.08
3	0.010504	0.005852	0.44891	0.01061896	84.83
4	0.01095	0.006392	0.453535	0.011697	85.72
5	0.010917	0.006611	0.444573	0.01190255	86.3

We show a sample scenario where in the random waypoint mobility model has a maximum speed of the nodes specified to be 10 m /s and a minimum speed of 9 m /s. The pause time parameter for the mobility model is kept to a value of 10 seconds. The above Table 4.4.2.1 shows the average values of the desired measures namely, λ_1, $\lambda_2(1-\pi_1)$, the estimated value of probability π_1 and the corresponding values of the service availability obtained through QualNet simulations for cluster size k varying between 1 and 5.

Using the values obtained through QualNet simulations, we parameterized the GSPN model introduced in Section 3.1 (Chapter 3) and analyzed the net using SPNP. The source code for the analysis of the GSPN model using SPNP is included in Appendix I. Since the presence of a token in place P_0 represents the state that the client is being served by some server in the network, the measure of interest for this analysis is the steady state probability of a token in place P_0. This probability is thus analogous to the service availability parameter for the client. Also, using Equation 4.3.10, the mathematical value of the service availability was computed substituting the parameters from Table 4.4.2.1. The results obtained using the SPNP analysis and mathematical formulation (Equation 4.3.10) are tabulated in Table 4.4.2.2 along with the service availability values obtained using QualNet simulations and by conducting test-bed experiments. We also plot the graph for service availability obtained using four different methods (Section 4.4.1) in Figure 4.4.2.1. As can be seen from Figure 4.4.2.1, the values obtained using the two Markovian analysis techniques match each other very well for all the cluster sizes.

Table 4.4.2.2 Comparison of service availability values

k (cluster size)	% SA (Empirical Parameterization using SPNP)	% SA (Mathematical Formulation)	% SA (QualNet Simulation Values)	% SA (Test-bed experiments)
1	43.2122116	43.17	65.6	47.41
2	51.1990398	51.192	80.08	66.62
3	64.7197167	64.703	84.83	73.35
4	66.1565358	66.134	85.72	77.33
5	66.249885	66.24	86.3	79.39

The results obtained by analysis of the model using SPNP and from the mathematical formulation are seen to be equal for different values of mobility and pause time variations. The comparison of the values obtained using mathematical formulation (Equation 4.3.10) and SPNP analysis of the model validate the GSPN model under Markovian assumptions. However, there is a considerable difference between these values obtained and the service availability measure obtained through QualNet simulations and test-bed experiments. Note that the test-bed experiments were conducted only with nine nodes with one of them acting as a policy server. The values obtained by analysis of our GSPN model are different in the absolute sense from those derived empirically but still follow a similar trend.

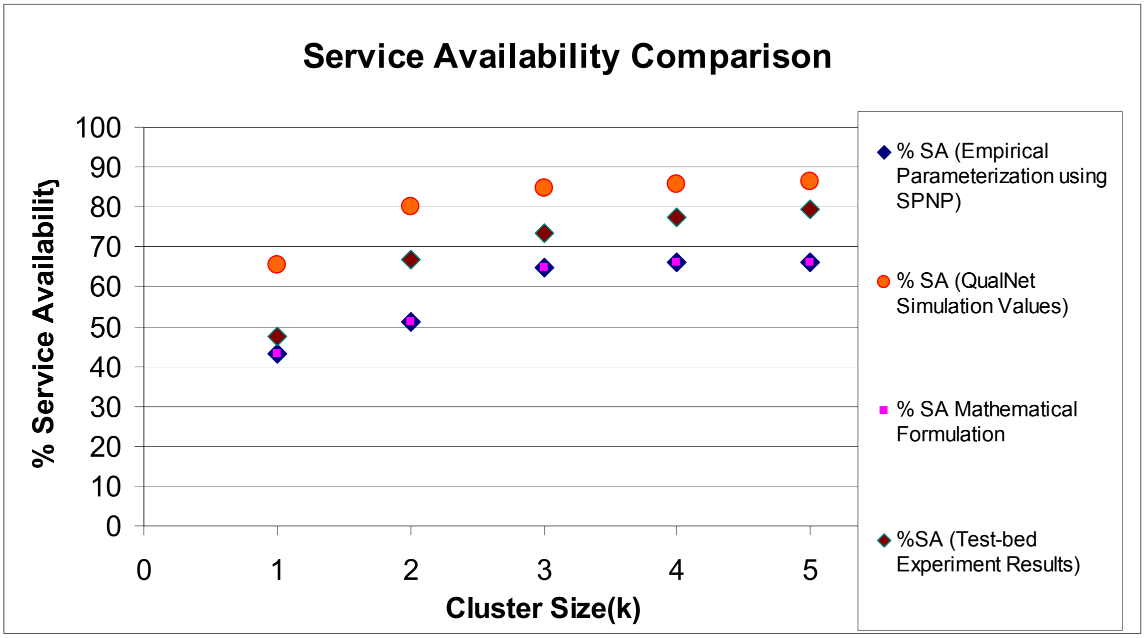


Figure 4.4.2.1 Service availability comparison graph (Markovian analysis)

The SPNP analysis of the model also provides the Markov chain representation of the model behavior as a secondary output. An equivalent Markov chain depiction of the GSPN model of Section 3.1 (Chapter 3) is shown below. It has the two states as explained before. Let they be represented by a state **0** and a state **1**. The 2-state Markov chain is as shown in Figure 4.4.2.2.

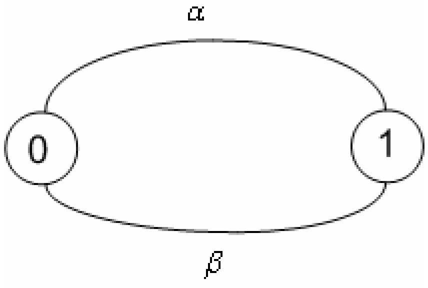


Figure 4.4.2.2 Two-state Markov chain representation of the GSPN model

4.4.3 Limitations of the Markov Model

As we showed in the previous sub-section, the service behavior is not completely captured by the development and analysis of the simple Markov model. This leads to the considerable difference between the service availability values obtained empirically and the ones obtained by analysis of our model. We attribute this to the following two facts:

- The empirical parameters were obtained using information from the network layer (the OLSR routing table update information) whereas the simulation values (the upper curve) are from a purely application layer perspective.
- As discussed in the previous chapter, there is no evidence that the transition by a client from one cluster to another is accurately described by a Poisson process. A more accurate model requires the use of distribution functions for the time in each state that more closely match the results observed in the parameterization phase of this work.

Within the limitations of simplicity of analysis, the Markov model captures the trend of the service availability. For a more detailed analysis of service availability and to capture the exact behavior of policy clients in the PBNM system, we analyze the system using non-Markovian GDT_SPNs (Chapter 2, Section 2.2.4) in the next chapter.

We believe that the first factor listed above is not a dominant factor in terms of performance. A more detailed analysis of the system, by modifying the basic model so that it incorporates non-Markovian processes, gets us close to the simulation values. Chapter 3 in fact proves that the random variables associated with the process of clients entering in to a new cluster (having not immediately done so) and with the departure processes of the clients moving outside the cluster are indeed not exponentially distributed. In Chapter 5, we attest our claim that the second factor mentioned is a more dominant factor in terms of assessment of service availability, by performing non-Markovian analysis of the system.

4.5 Summary

In this chapter, we focused on the development and validation of a simple GSPN model with Markovian assumptions for all non-deterministic timed transitions. We discussed the equivalent network of queues representation and elaborated on the implementation methodology used to validate the model. We then compared the results obtained using the CSPL equivalent of the GSPN model developed in Section 4.1 and the mathematical analysis of Section 4.3 with the simulation values obtained. We were also able to translate the Markov model to an equivalent two-state Markov chain.

Thus, we showed that the Markov model of the PBNM system is a useful model in that it captures the trend of the service availability but is not accurate enough to model the exact service behavior. We present two plausible reasons for this conclusion and suggest a need for a more detailed modeling of the system with generally distributed transition times. Chapter 5 focuses on the modification of the GSPN model and validation of the system using the new non-Markovian representation, which we call the GDT_SPN model.

Chapter 5

Non-Markovian Analysis

The PBNM system that is the focus of this thesis is a classical example of a complex distributed system with various components (nodes in the MANET) exhibiting independent concurrent behavior. The goal of this work is to assess service availability in such a system. As discussed in Chapter 3, we gathered statistics to parameterize the model and investigated the system processes (Chapter 3, Section 3.3.5) to determine the probability distributions followed by the random variables associated with them. Chapter 2 provides us with the necessary background on the methodology for modeling systems with Petri Nets. In Chapter 4, we use this modeling formalism to analyze the GSPN model of our system (Section 3.1, Chapter 3) under Markovian assumptions. We also show (Chapter 4, Section 4.4.2) that the Markov model, though effective in terms of capturing the correct trends for service availability, does not provide a good match with the service availability values obtained through QualNet simulations and test-bed experiments. We thus motivate the need for a thorough analysis of the system without simplified Markov assumptions. In this chapter, we propose and analyze a GDT_SPN model of the PBNM system in a MANET to assess service availability. The GDT_SPN model is a modified representation of the original GSPN model and associates general probability distributions (approximating the distributions obtained via QualNet simulations) to transitions in the model.

This chapter applies the theory of Phase Type distributions introduced in Chapter 2 to develop the GDT_SPN representation of our system. The probability distributions associated with random variables characterizing the system processes (obtained in

Chapter 3) are approximated by discrete PH type distributions [29] using PhFit. PhFit is an approximation tool developed at Budapest University of Technology and Economics by Telek Miklos, Andrea Bobbio, András Horváth *et al.* The approximated distributions are applied as inputs to the GDT_SPN model to obtain the desired performance measures. The remainder of the chapter is organized as follows. In Section 5.1, we discuss the development of our GDT_SPN model from the original GSPN model. Section 5.2 reviews phase type distributions and in Section 5.3 we elaborate on the analysis methodology. The PhFit analysis to approximate general distributions to DPH type distributions is presented in Section 5.4. Validation of the GDT_SPN model is discussed in Section 5.5 and we conclude the chapter with a concise summary in Section 5.6.

5.1 Development of the GDT_SPN Model

The policy based network management system has been described in Section 1.2 (Chapter 1). In this section, we modify the GSPN model of the PBNM system (Chapter 3, Section 3.1) to incorporate general (not necessarily exponential) distributions associated with the random variables describing various transitions in the system. Since it involves generally distributed transitions (albeit approximated with PH type distributions for analyzing the model), we call this model the GDT_SPN model of the system.

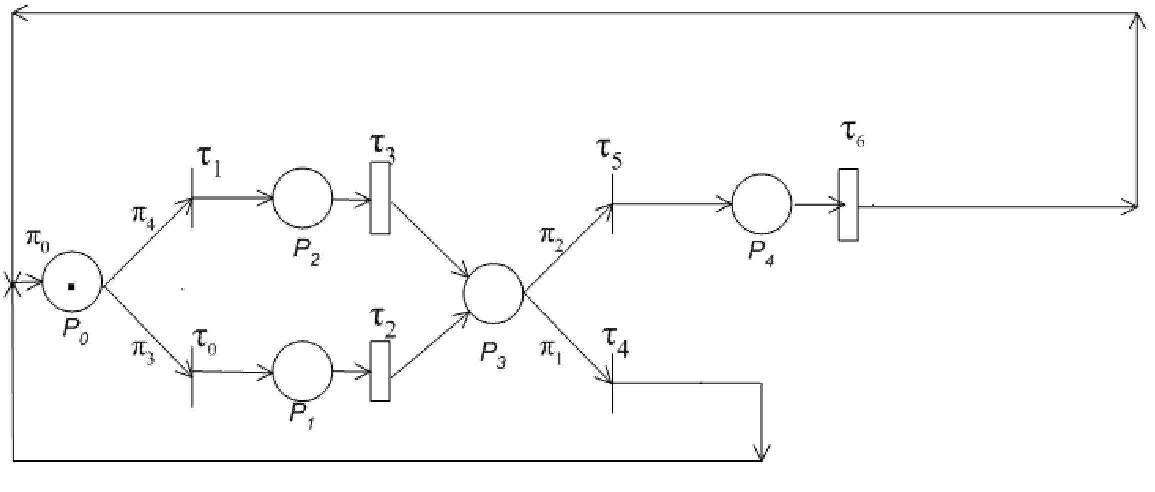


Figure 5.1.1 GDT_SPN Model for policy client in PBNM system

It must be stated here that the GDT_SPN model is conceptually the same as the originally conceived GSPN model of the system, but with necessary modifications to represent the general distributions observed for the parameters of interest in Section 3.3.5. Figure 5.1.1 shows the proposed GDT_SPN model for analyzing the system with non-Markovian assumptions. It consists of a set of places P = $\{P_0, P_1, \ldots, P_4\}$ and a set of transitions T = $\{\tau_0, \tau_1, \ldots, \tau_6\}$. As in the Markov model, we have a single token representative of a particular client in the system. We assume the token is initially in place P_0. The token can be initially placed at any place in the model without a change in the behavior of the system. Unlike the model analyzed in Chapter 4, the presence of a token in place P_0 represents a vanishing condition and the token immediately gets transferred to either place P_1 or place P_2. Therefore the presence of the token in any of the three places signifies that the client is currently being covered inside some cluster and, hence, is in service.

In Chapter 3, we concluded that the random variable associated with the movement of clients outside the cluster (rate λ_1) is generally distributed. It can also be seen that approximately one-third of the times the client nodes move out of the cluster after a fixed time, as determined by the KA timer, while in other cases the clients move out of the cluster after a randomly distributed time. This behavior can be characterized by the combination of a deterministic time transition and a generally distributed time transition. To incorporate this behavior, we split the transition τ_0 of Figure 3.1.1 into two transitions τ_2 and τ_3. Note that transition τ_2 models the movement of clients outside the cluster caused by the expiration of the KA timer. Transition τ_3 is used to model the other case of clients moving out after a generally distributed random time shown in Figure 3.3.5.2 of Chapter 3. The immediate transitions τ_0 and τ_1 and the dummy places P_1 and P_2 are used to model the switching probabilities for transition τ_2 or transition τ_3. Thus, π_3 is the probability with which a client leaves its cluster after a fixed time (modeled by transition τ_2) and $\pi_4 = (1 - \pi_3)$ is the probability with which the client moves out of the cluster after a generally distributed random time. Place P_3 serves as an output place for the two transitions τ_2 and τ_3. π_1, similar to the GSPN model, represents the probability of moving into a new cluster immediately (Chapter 3, Section 3.1). π_2 is simply (1-π_1) and is the probability of not moving immediately into a new cluster. If the client does not move back in a new cluster, then it enters the state where it is looking for service. This state is represented by the presence of a token in place P_4.

Recall that in Section 3.3.5 (Chapter 3), we show that the histogram curve for the reciprocal of λ_2(1-π_1), which is representative of the actual times spent by the client outside the cluster, can be approximated to a PDF curve of a uniformly distributed random variable. Thus, the random variable associated with the time spent by clients outside a cluster follows an approximate uniform distribution. In order to represent this distribution associated with the random variable of the λ_2(1-π_1) process, we employ transition τ_6. Thus, a client will spend a uniformly-distributed amount of time in place P_4 before getting back in place P_0, where it is again covered inside one of the possible clusters and hence starts receiving service. In summary, one can use the basic model defined in Section 3.1 (Chapter 3) and arrive at its non-Markovian representation in accordance with the behavior of system processes. This system behavior can be obtained either via a simulation based study of the system (the approach we follow) or through a heuristic estimation involving analysis of the mobility model (as discussed in Chapter 6, Section 6.2).

5.2 Re-visiting Phase Type Distributions

Phase type distributions are CDFs of time until absorption of the underlying Markov chain with n transient states and an $(n+1)^{th}$ absorbing state, given a vector associating initial probabilities to these states. They are classified into continuous phase type distributions (CPH) and discrete phase type distributions (DPH), depending upon whether the underlying Markov chain is continuous or discrete. In this section, we revisit the phase type distributions defined in Section 2.2.5 (Chapter 2) and introduce a sub-class of the DPHs called the acyclic discrete phase type (ADPH) distributions. We present some mathematical identities associated with ADPH type of distributions and motivate the application of these fundamentals to the non-Markovian analysis of our system model.

DPH distributions (Chapter2, Section 2.2.5) have received little attention in system modeling, mainly due to the focus on research activity and applications using CPH distributions [46]. Recently, there has been renewed interest in DPH distributions [5] due to their wide range of applicability in stochastic modeling of distributed systems following general distributions. We define a DPH random variable as follows. If **Γ** is the time till absorption into the $(n+1)^{th}$ absorbing state in the DTMC, then **Γ** is a DPH random variable of order n and represented by [**α**, **B**], where **α** is the initial probability vector and **B** is the transient state transition probability matrix. The DPH representation of any general distribution is shown [47] to be non unique and non minimal. This has led to the study of a sub class of the DPH distributions called the ADPH distributions, which provide a unique as well as minimal representation of the general distributions. These minimal representations of DPH distributions are called the canonical forms. An ADPH distribution is formally defined in Definition 5.1.

Definition 5.1: Acyclic DPH is a DPH in which the states of the DTMC can be organized such that the transient state probability transition matrix **B** (Chapter 2, Section 2.2.5) becomes an upper triangular matrix.

The DTMC representation of the distribution is therefore an acyclic graph [5] in the case of an Acyclic DPH. This implies that, in an ADPH, each state is visited only once before an eventual absorption. In order to describe the canonical form representation and motivate its use in the analysis of our GDT_SPN model, we introduce some terminology defined and presented by Bobbio *et al.* A detailed description of these concepts can be found in [5].

A path is defined as the sequence of states visited before transition into an absorbing state. For a DPH of order$n$, the number of such paths is finite and at most$2^n - 1$. The length of a path is the number of states visited following that path before absorption. In an ADPH with n transient states, if the Eigen values of the matrix **B** are represented by $q_1, q_2, ..., q_n$, then the values $p_i = 1 - q_i$ represent the exit rate from state i. A **basic path** of the DTMC in an ADPH is then defined as any path r of length l such that it contains the l fastest phases $q_{n-l+1}, ..., q_{n-1}, q_n$. An ADPH has a unique minimal representation in the form of a mixture of its basic paths and is called the canonical form 1 (CF1). The matrix representation of the ADPH in the CF1 form is given by $[a, P]$ where,

$$a = [a_1, a_2, ..., a_n] \qquad \text{and} \qquad P = \begin{bmatrix} q_1 & p_1 & 0 & 0 & . & . & . & 0 \\ 0 & q_2 & p_2 & 0 & . & . & . & 0 \\ . & . & . & . & & & . & . \\ . & . & . & . & & & . & . \\ . & . & . & . & & & . & . \\ 0 & 0 & 0 & 0 & . & . & . & q_n \end{bmatrix} \qquad (5.2.1)$$

Also $\sum_1^n a_k = 1$ and $p_1 \le p_2 \le p_3 ... \le p_n$. (5.2.2)

However, it is sometimes convenient to represent the DPH in a minimal form such that the initial probability is concentrated in the first state. This means that phase 1 is deterministically the starting phase of the DTMC evolution. This results in the canonical form 2 (CF2). In CF2 form, transitions are possible from phase 1 to all other phases and from all other phases i to themselves or to the phase $i+1$. The CF2 form is represented by $[\alpha, B]$ where;

$$\alpha = \begin{bmatrix} 1 & 0 & . & . & . & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} q_n & c_1 & c_2 & c_3 & . & . & . & c_{n-1} \\ 0 & q_1 & p_1 & 0 & . & . & . & 0 \\ . & . & . & . & & & . & . \\ . & . & . & . & & & . & . \\ . & . & . & . & & & . & . \\ 0 & 0 & 0 & 0 & . & . & . & q_{n-1} \end{bmatrix} \qquad (5.2.3)$$

Also, the equivalence between the CF2 and the CF1 forms is given by:

$c_k = a_k p_n$ (5.2.4)

The canonical form 3 (CF3) is another convenient representation of the ADPH. In the CF3, transitions are possible from phase i to itself and phases $i+1$ and $n+1$. Similar to the CF2 form, the initial probability is 1 for phase 1 and $0\,\forall i \neq 1$. The matrix representation of CF3 form is given as $[\delta, M]$ where;

$$\delta = \begin{bmatrix} 1 & 0 & . & . & . & 0 \end{bmatrix} \text{ and } M = \begin{bmatrix} q_n & e'_n & 0 & 0 & . & . & . & 0 \\ 0 & q_{n-1} & e'_{n-1} & 0 & . & . & . & 0 \\ . & . & . & . & & & . & . \\ . & . & . & . & & & . & . \\ . & . & . & . & & & q_2 & e'_2 \\ 0 & 0 & 0 & 0 & . & . & 0 & q_1 \end{bmatrix} \tag{5.2.5}$$

Also, the equivalence between the CF3 and the CF1 forms is given by

$$s_i = \sum_{j=1}^{i} a_j \ , \ e'_i = \frac{a_i}{s_i} p_i \ \& \ e_i = \frac{s_{i-1}}{s_i} p_i \tag{5.2.6}$$

The CF2 and the CF3 forms are important from our perspective for approximating general distributions using PhFit and representing them in a form understandable by the WebSPN tool [57] used to analyze the GDT_SPN model of Section 5.1. We use the CF2 matrix representation of the general transition τ_3 in our model as an input to the WebSPN specification of our GDT_SPN model.

5.3 Non-Markovian Model Analysis Methodology

In this section, we discuss the methodology used to analyze the non-Markovian GDT_SPN model. The following general steps were followed:

- Develop a system model: This involves developing a GDT_SPN model for the system under consideration, with representation of system processes in the original model by corresponding general transitions.
- Identify all the generally distributed transitions in the system model: We identified the transition τ_3 to be the generally distributed transition that needs to be approximated to a DPH type transition. By generally distributed we mean

transitions that cannot be approximated by any of the known standard distribution transitions.

- Approximate all the generally distributed transitions to DPH transitions using a phase approximation tool like PhFit: This involves using the histogram curves gathered from the QualNet simulation results as input to the PhFit tool and obtaining a discrete phase type representation of the distributions.
- Convert the ADPH transitions in the CF1 form to CF2/CF3 form: For the ADPH approximation of each generally distributed transition, it is essential to convert the results obtained to the CF2/CF3 form. This conversion is needed because the provision to specify a DPH transition in the WebSPN tool demands that it must be input in the CF2/CF3 form.
- Identify the memory policies to associate with all transitions: This step ensures that the model behaves in accordance with the memory specifics of the system behavior. It is important to analyze how each transition should behave depending on the history of the system.
- Analyze the model using WebSPN: With all the parameters of the model specified in the WebSPN representation of the system, this step involves obtaining all the desired measures.

This methodology can be used for non-Markovian analysis of any general system modeled as a GDT_SPN. The first two steps have already been discussed in Section 5.1. We present the PhFit analysis (approximation of generally distributed transitions to DPH transitions) and conversion of the DPH transitions from their CF1 form to the CF2 form in the next section and explain the last two steps of the solution in Section 5.5.

5.4 PhFit Analysis

This section deals with approximation of the general distributions to ADPH type of distributions using the phase approximation tool called PhFit [29]. The ADPH approximation of general distributions involves the discretization of the distribution using some discretization step. The ADPH estimation algorithm is then run over the discrete samples. This is done automatically using the PhFit tool; however, we still need to select a suitable value of discretization step and the order (i.e. the number of phases) of the DPH distribution. Most smooth distributions can be well-fitted using few phases (a phase order of 2, 3 or 4). However, more complex distributions may need more phases. We choose the number of phases as 8 for the DPH approximation [69]. For choosing the discretization interval, we follow the bound equation given in [5]. Given a random variable X with a mean $E(\mathrm{X})$ and a coefficient of variation $cv(\mathrm{X})$, if it is to be approximated to a DPH with n phases, the discretization step δ is bounded by

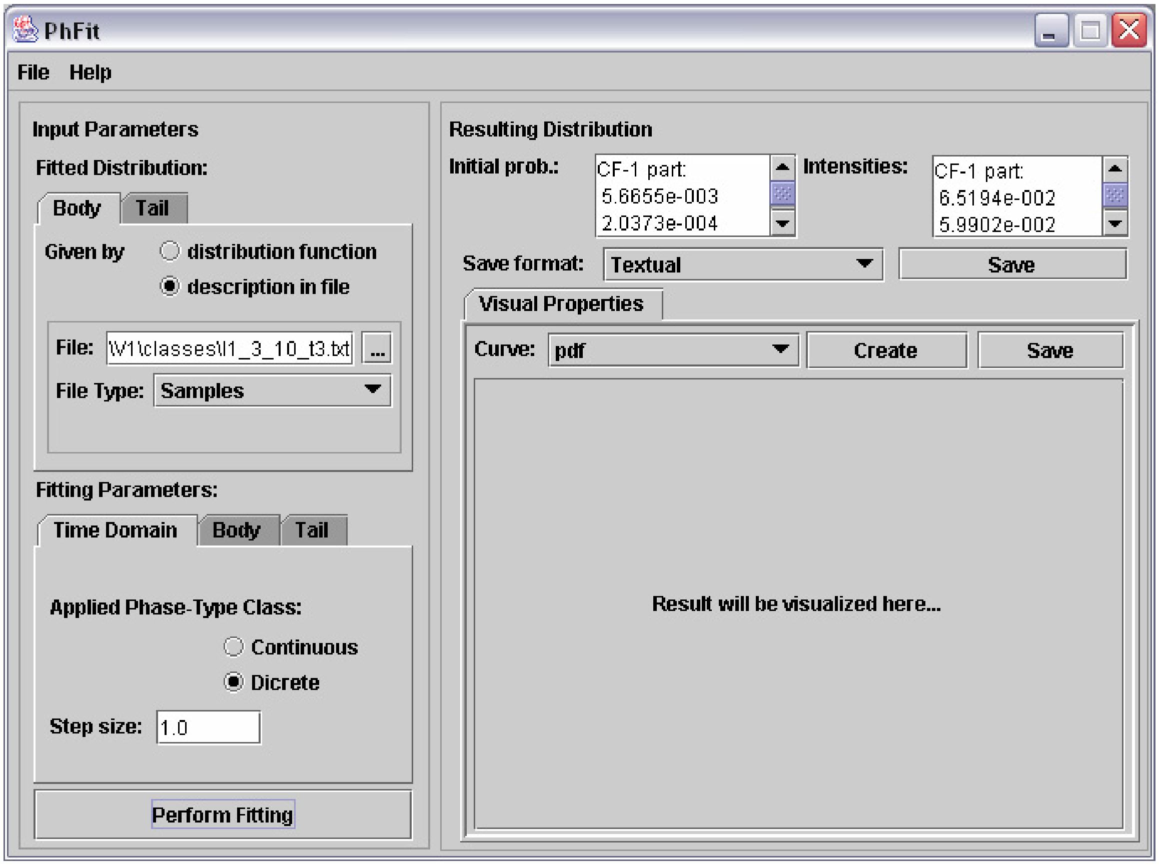
$$\left[\frac{1}{n} - cv^2(\mathrm{X})\right]E(\mathrm{X}) < \delta \le \frac{E(\mathrm{X})}{n-1} \tag{5.4.1}$$

We consider a basic scenario where the $\mathrm{V}_{\mathrm{max}}$ value is set to 10 m/s and the cluster size is varied from k =1 to k =5 as a sample scenario for the entire non-Markovian analysis. The KA timer value is set to 50 seconds, while the pause time is 10 seconds. The histogram curves of λ_1 are processed to remove the deterministic time peak corresponding to transition τ_2. The split curves serve as an input to the PhFit tool. Table 5.4.1 shows the bounds on the discretization step for various values of the cluster size for the transition τ_3.

Table 5.4.1 Bounds on δ for the approximation of general distribution corresponding to τ_3

Cluster size k	$E(\mathrm{X})$ (τ_3)	$cv(\mathrm{X})$ (τ_3)	δ_{ub} (τ_3)	δ_{lb} (τ_3)
1	148.17	0.718	21.167	-57.86
2	173.61	0.7175	24.801	-67.67
3	168.98	0.7227	24.14	-68.34
4	163.78	0.7364	23.397	-68.34
5	169.27	0.7229	24.181	-67.29

Following the bounds equation 5.4.1 and using the bound values from the Table 5.4.1, we choose a discretization step of 1.0 for all the approximations.



Input Parameters

Fitted Distribution:

Given by ○ distribution function ● description in file

File: \V1\classes\l1_3_10_t3.txt

File Type: Samples

Fitting Parameters:

Applied Phase-Type Class: ○ Continuous ● Dicrete

Step size: 1.0

Resulting Distribution

Initial prob.: CF-1 part: 5.6655e-003 2.0373e-004

Intensities: CF-1 part: 6.5194e-002 5.9902e-002

Save format: Textual

Curve: pdf

Result will be visualized here...

Figure 5.4.1 DPH approximation using PhFit

Figure 5.4.1 shows a sample run of the PhFit tool for the cluster size of 3. As can be seen from this figure, the PhFit analysis tool outputs the initial probabilities associated with the DTMC phases and the transient state transition probabilities.

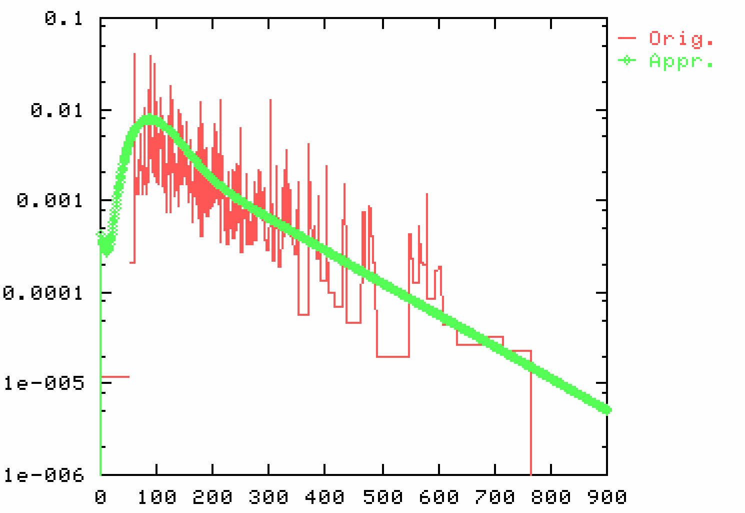


Figure 5.4.2 Approximation plot obtained using PhFit

The tool also gives an approximation plot using the "gnuplot" tool. Figure 5.4.2 shows a sample plot for the distribution of the random variable associated with the τ_3 transition being approximated to an ADPH distribution.

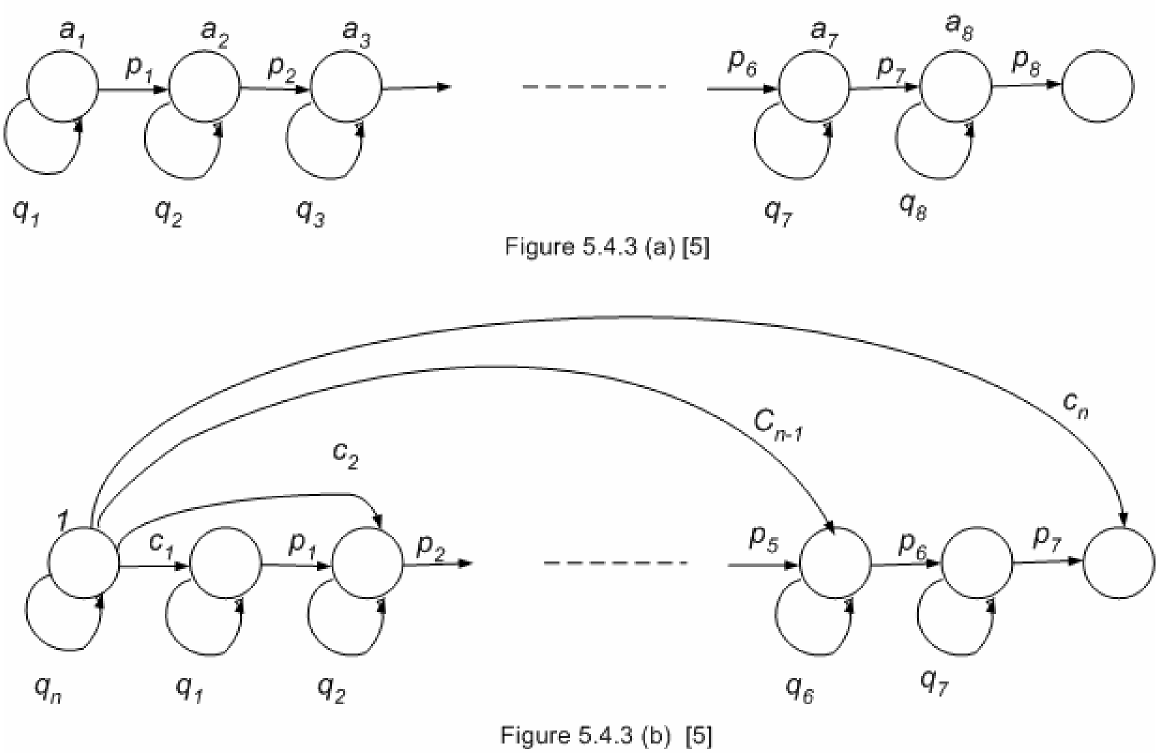


Figure 5.4.3 (a) [5]

Figure 5.4.3 (b) [5]

Figure 5.4.3 Phase representations of the CF1 and CF2 form

Using the method suggested above, we approximated the transition τ_3 for all the cluster sizes and obtained the CF1 form representation for these generally distributed transitions. However, the CF1 representation of the ADPH is not sufficient to serve as an input to the WebSPN tool. Hence, we converted the CF1 output obtained from the PhFit tool into the CF2 form, which is compatible with the WebSPN analysis tool. This is done using the equivalence relation (Equation 5.2.4) between the CF1 and the CF2 forms. Figure 5.4.3 (a) shows the phase representation of the CF1 form and Figure 5.4.3 (b) shows the CF2 form. In Table 5.4.2, we show a sample conversion of the CF1 representation of the transition τ_3 to CF2 form for a cluster size, k= 3. The CF2 matrix obtained is saved to a text file in an appropriate form and used as an input to the model. This procedure is repeated for each value of cluster size for the transition τ_3.

Table 5.4.2 Conversion from CF1 to CF2 form

a	5.67E-03	2.04E-04	5.52E-03	9.22E-03	3.70E-02	4.63E-01	1.35E-01	3.44E-01
α	1	0	0	0	0	0	0	0
p_8	6.52E-02							
CF1 Form	9.35E-01	0.00E+00	0	0	0	0	0	0
	5.99E-02	9.40E-01	0	0	0	0	0	0
	0	5.92E-02	9.41E-01	0	0	0	0	0
	0	0.00E+00	5.90E-02	9.41E-01	0	0	0	0
	0	0.00E+00	0	5.87E-02	9.41E-01	0	0	0
	0	0.00E+00	0	0	5.82E-02	9.42E-01	0	0
	0	0.00E+00	0	0	0	2.00E-02	9.80E-01	0
	0	0.00E+00	0	0	0	0	7.18E-03	9.93E-01
CF2 Form	9.35E-01	3.69E-04	1.33E-05	3.60E-04	6.01E-04	2.42E-03	3.02E-02	8.80E-03
	0	9.93E-01	7.18E-03	0	0	0	0	0
	0	0	9.80E-01	2.00E-02	0	0	0	0
	0	0	0	9.42E-01	5.82E-02	0	0	0
	0	0	0	0	9.41E-01	5.87E-02	0	0
	0	0	0	0	0	9.41E-01	5.90E-02	0
	0	0	0	0	0	0	9.41E-01	5.92E-02
	0	0	0	0	0	0	0	9.40E-01

5.5 Validation of the GDT_SPN Model

We discussed the conversion of general distributions associated with transition τ_3 to their ADPH approximation in the previous section. We also discussed the conversion of the CF1 form of ADPH distribution obtained using PhFit to the CF2 form compatible as an input to the WebSPN tool. In this section, we briefly explain the WebSPN tool used for the solution of the GDT_SPN model. We then discuss the steps involved in validation of the model and present the relevant results obtained.

5.5.1 WebSPN – A Non-Markovian SPN Tool

WebSPN is a non-Markovian SPN tool developed in the Mathematics Department at the University of Messina, Italy, by Puliafito *et al.* [57]. WebSPN employs client-server architecture and uses the Java technology, which makes it easily accessible over the World Wide Web. A server daemon is run on a UNIX based (Sun Solaris, Linux) system using a specific port number and IP address.

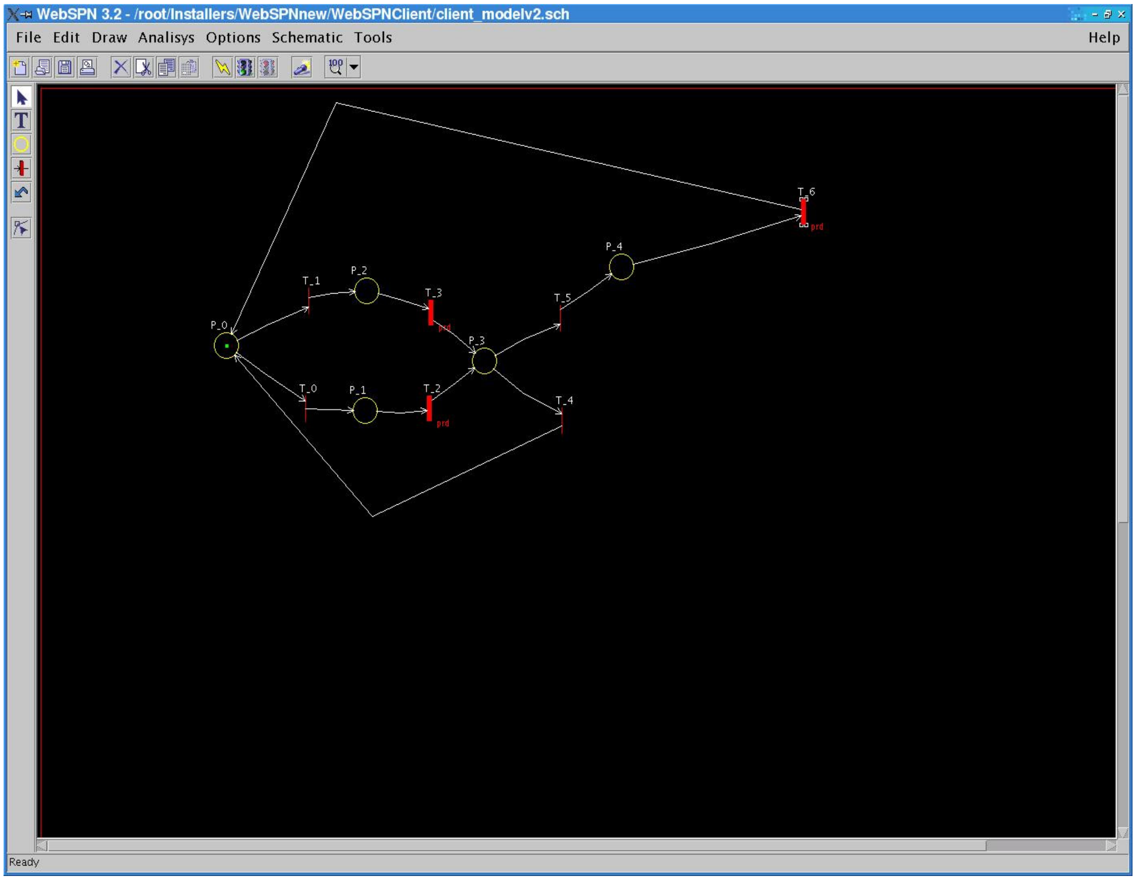

WebSPN 3.2 - /root/Installers/WebSPNnew/WebSPNClient/client_modelv2.sch
File Edit Draw Analisys Options Schematic Tools
Help
T_6
prd
P_4
T_5
P_2
T_1
T_3
prd
P_0
P_3
T_0
P_1
T_2
prd
T_4
Ready

Figure 5.5.1.1 GDT_SPN model specification using WebSPN 3.2[§]

The client provides the interface for specifying a system model and analyzing it, and can reside on the same machine as the server or can be connected to the server over a network. The client, which can be run on a Microsoft Windows or a UNIX platform, binds itself to the server daemon using its IP address and port number. Figure 5.5.1.1 shows the client window with our GDT_SPN model specified. The tool consists of two parts, namely the GUI based model specification interface and the analysis engine. The

[§] Please refer to Figure 5.1.1 for a better view of the model.

GUI based interface is incorporated in the client while the analysis engine works using the client-server model. All the specifications of the PN and the desired measures are specified at the client and computed by the server using the analysis engine. The analysis engine is thus responsible for analyzing the PN and computing the desired measures. The results are then ported back to the client and displayed in a suitable form. We used the recently released version of the tool, WebSPN 3.2, for the analysis of our GDT_SPN model.

5.5.2 Model Validation and Results

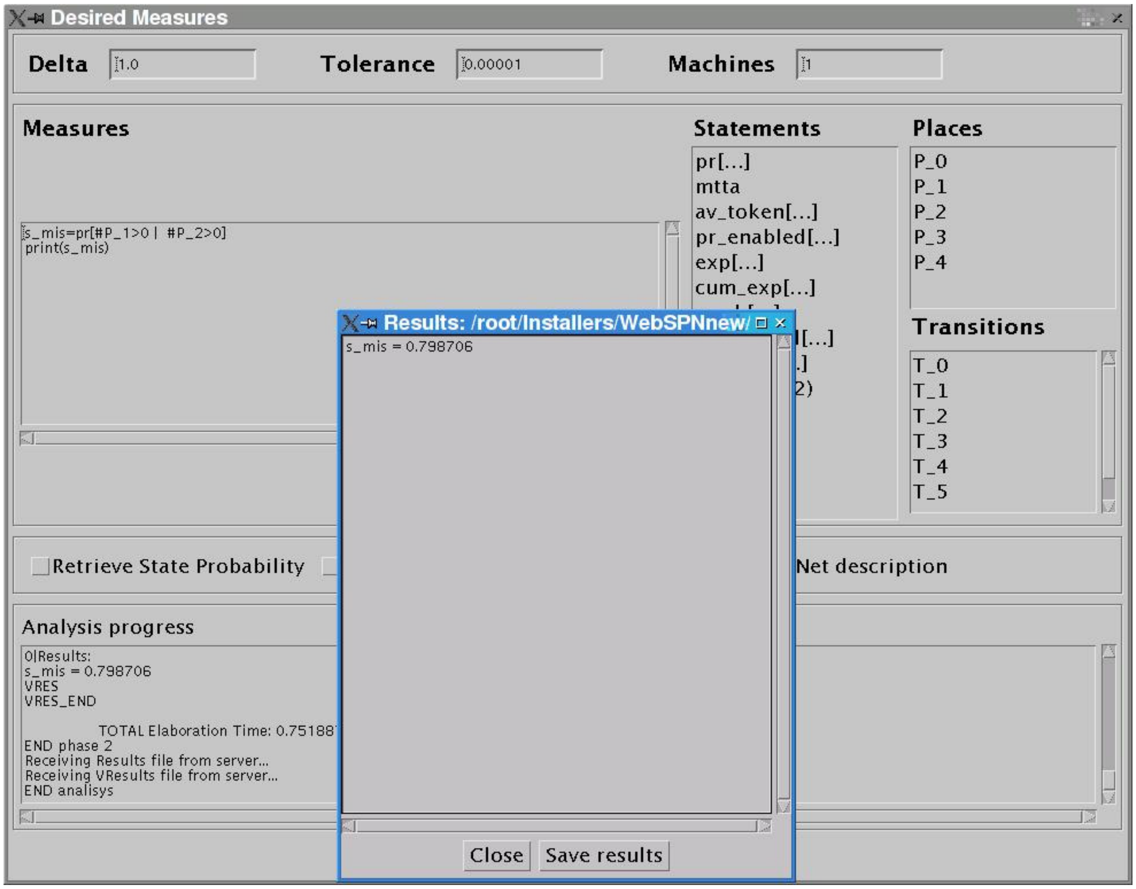
This section focuses on presenting the methodology used to validate the non-Markovian model. We also discuss the results obtained from analyzing the GDT_SPN model of Section 5.1 using WebSPN 3.2. The following steps are used to validate the GDT_SPN model:

- specify the GDT_SPN model using WebSPN 3.2 with a discretization step $\delta = 1$ (same value as the one used in the PhFit analysis), associate the CF2 matrix form representations to the generally distributed transitions and analyze the results for service availability.
- compare the results obtained against the ones obtained through QualNet simulations and test-bed experiments.

Table 5.5.2.1 Input parameters for the GDT_SPN model

Cluster size k	π_1	π_3	Uniform distribution τ_6		Deterministic transition τ_2 Value (seconds)
			Lower Limit (seconds)	Upper Limit (seconds)	
1	0.36682	0.268	3.3097	1000	52.72
2	0.3895	0.3449	2.8903	1000	64.353
3	0.44895	0.2982	3.5247	1000	51.821
4	0.45352	0.3054	2.881	1000	52.55
5	0.4457	0.3247	1.435	1000	58.352

With the model parameters obtained partly from QualNet simulations (Chapter 3) and partly from the post-simulation analysis (analysis of histograms and ADPH approximation), we input the values obtained in each case to the WebSPN model shown in Figure 5.5.1.1 and solve it for steady state service availability. Table 5.5.2.1 tabulates the input parameters of the model for the sample scenario (V_{max} = 10 m/s). All the timed transitions namely, τ_2, τ_3 and τ_6 have a prd memory policy (Chapter 2, Section 2.2.4) associated with them. The immediate transitions have appropriate switching probabilities associated with them. For our GDT_SPN model, the service availability is represented by the probability of presence of a token in place P_1 or place P_2. Unlike the Markov model presented in Chapter 4, in this case, the presence of a token in place P_0 is ephemeral as two immediate transitions τ_0 and τ_1 are enabled in this marking. Figure 5.5.2.1 shows a sample screenshot of the steady state service availability result obtained from running the WebSPN tool for a cluster size of 3.



Delta 1.0 Tolerance 0.00001 Machines 1

```
s_mis=pr[#P_1>0 |  #P_2>0]
print(s_mis)
```

```
s_mis = 0.798706
```

```
0|Results:
s_mis = 0.798706
VRES
VRES_END

            TOTAL Elaboration Time: 0.75188
END phase 2
Receiving Results file from server...
Receiving VResults file from server...
END analisys
```

Figure 5.5.2.1 Service availability for cluster size k =3, V_{max} = 10 m/s

Table 5.5.2.2 presents the results obtained for service availability using the validation methodology explained above for the sample scenario (V_{max} = 10 m/s) described in the PhFit analysis in Section 5.4.

Table 5.5.2.2 Service availability comparison

k (cluster size)	%SA (Non-markovian GDT_SPN analysis)	% SA (Empirical Parameterization using SPNP)	% SA (Mathematical Formulation)	% SA (QualNet Simulation Values)	% SA (Test-Bed experiments)
1	66.84	43.2122116	43.17	65.6	47.41
2	75.378	51.1990398	51.192	80.08	66.62
3	79.87	64.7197167	64.703	84.83	73.35
4	79.2	66.1565358	66.134	85.72	77.33
5	79.06	66.249885	66.24	86.3	79.39

We also compare these values obtained against the Markovian analysis results (Chapter 4, Section 4.4.2) and the service availability values obtained using QualNet simulations and test-bed experiments in the table and plot the values obtained in Figure 5.5.2.2. As can be seen from Table 5.5.2.2 and Figure 5.5.2.2, the service availability values obtained using the WebSPN analysis of the GDT_SPN model of the PBNM system are close to the values obtained via QualNet simulations and the test-bed experiments. Readers must be reminded again that the test-bed experiment results come from a different experimental set-up consisting of 9 nodes with 1 node acting as a policy server. Therefore, a direct comparison of these values with other results is not justified. Even though, these results are from a different set-up, they are important in order to reflect the similar trend obtained via realistic test-bed experiments. This analysis validates our GDT_SPN model with respect to the service availability seen by policy clients in the PBNM system.

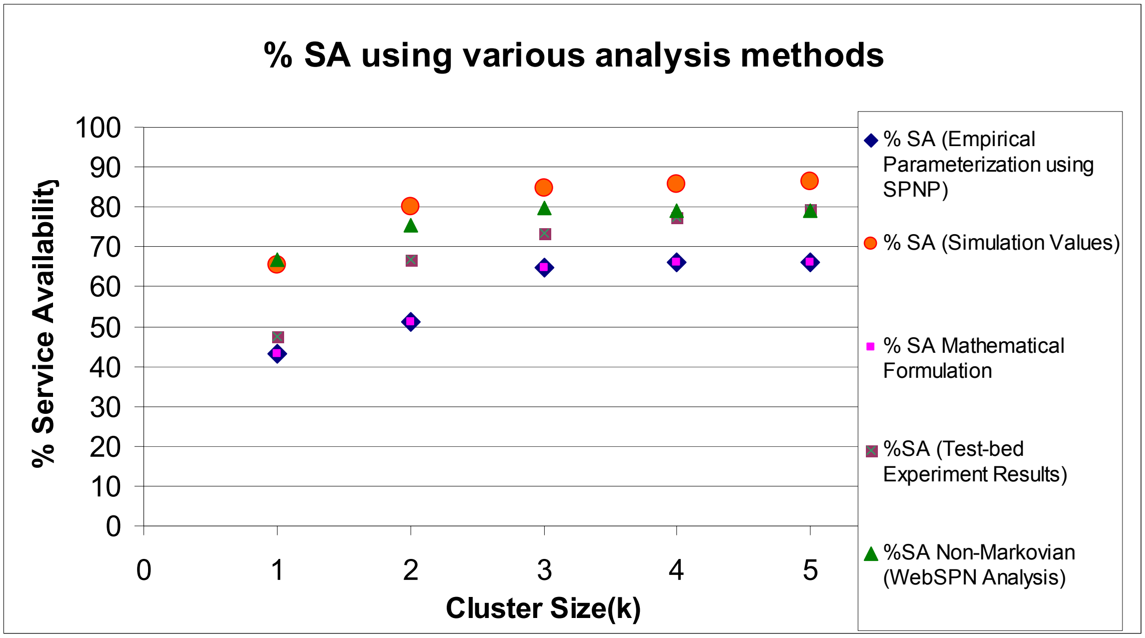


Figure 5.5.2.2 Service availability comparison graph (Non-Markovian analysis)

Using the GDT_SPN analysis, we obtain a better approximation of the simulation values of the service availability. We thus show that a detailed analysis of the PBNM system using actual system processes indeed provides a sufficiently accurate model for capturing system behavior.

5.6 Summary

In this chapter, we focused on the development and validation of the non-markovian GDT_SPN model with DPH approximations for the generally distributed timed transition. We discussed the ADPH class of phase type distributions and presented definitions and results relevant to our research. We also presented the methodology used to transform the generally distributed transitions to a form compatible as input to non-markovian SPN tools like WebSPN. We compared the results obtained by analyzing our model using the WebSPN tool against the ones obtained using the Markovian analysis of Chapter 4.

Thus, we showed that the GDT_SPN model of the PBNM system is a good approximation to the actual behavior of nodes in the PBNM system from a service availability perspective. We justify the need for exact analysis of any distributed system using our model as an example. In Chapter 6, we present the contributions of this research and provide directions for future research in the area of analytical modeling of network systems.

Chapter 6

Conclusions and Future Work

In this thesis, we developed Stochastic Petri Net models for performance evaluation of a PBNM system (Chapter 1, Section 1.2) in the context of mobile ad hoc networks. We developed a simple GSPN model (under Markovian assumptions for timed transitions) of the system in order to estimate the average service availability experienced by client nodes. We concluded that the simple GSPN model does not provide a close approximation to the empirically-obtained values (Chapter 3 and [53]) of service availability. However, under such Markov assumptions, service availability results obtained by analyzing the GSPN model follow a similar trend as the empirical results. This led us to develop a more comprehensive system model called the GDT_SPN model. This model more accurately approximates the processes describing the movement of nodes into and out of clusters in the PBNM system. Evaluation of the GDT_SPN model resulted in a better approximation of service availability in this system. This was verified by comparing the results of the analytical assessment (Chapter 5, Section 5.5) with the values of service availability obtained via QualNet simulations and experiments conducted on a test-bed [53].

This chapter summarizes the main contributions of this research and the key results obtained as an outcome of this thesis. We conclude the thesis by outlining some directions for future work. Section 6.1 discusses the major contributions of this research.

Section 6.2 provides pointers for further research in this area and Section 6.3 summarizes this thesis.

6.1 Contributions of this Research

The analytical modeling of the PBNM system for MANETs involved the development of system models, parameterization of these models and their analysis. The main contributions of this thesis are summarized below.

- Development of a model for service availability in a PBNM system for MANETs: The performance evaluation of the PBNM system in terms of service availability involved development of system models at two levels of complexity. We developed a simple GSPN model with Markovian assumptions for timed transitions which while capturing the correct trends for service availability, did not provide a good approximation to the values obtained empirically through simulations or test-bed experiments. This led to the development of a more complex GDT_SPN model, which provides a better match with the empirically-obtained results. The modeling methodology applied here can be easily extended to analyze the performance of other applications that are concerned with the fundamental problem of service availability in a MANET. Providing such a methodology for performance evaluation of systems that face the common problem of service availability is an important contribution of this research.

- Performance analysis using different modeling tools: We employed different performance analysis tools like mathematical analysis of queuing networks, Petri Nets and a simulation based study to cross-validate the results obtained in this research. A simulation-based study of the system was carried out to parameterize our models. In case of Markovian modeling, we developed a closed network of queues model and formulated a mathematical equation for service availability. We validated our simple GSPN model using this closed-form expression. In case of the non-Markovian modeling of the system, the parameters of interest obtained via QualNet simulations (Chapter 3) were incorporated into the model using different statistical and approximation tools like MINITAB (Chapter 3, Section 3.3.5), PhFit (Chapter 5, Section 5.3), and WebSPN 3.2 (Chapter 5, Section 5.4) to obtain the desired measure of service availability. The effective use of these various tools for analyzing service availability of the PBNM system is another contribution of this work.

- <u>Non-Markovian modeling of systems:</u> The methodology we used for non-Markovian modeling of the PBNM system is novel in terms of its applicability to study the performance parameters of MANETs. We believe that it can be applied for studying other phenomena of interest in a MANET given some heuristic knowledge of the behavior of its various system processes.

6.2 Directions for Future Work

We believe that this research can serve as a pre-cursor to analytical modeling of a wide variety of network systems. With our current research as a strong motivation, we provide the following directions for future work in this area.

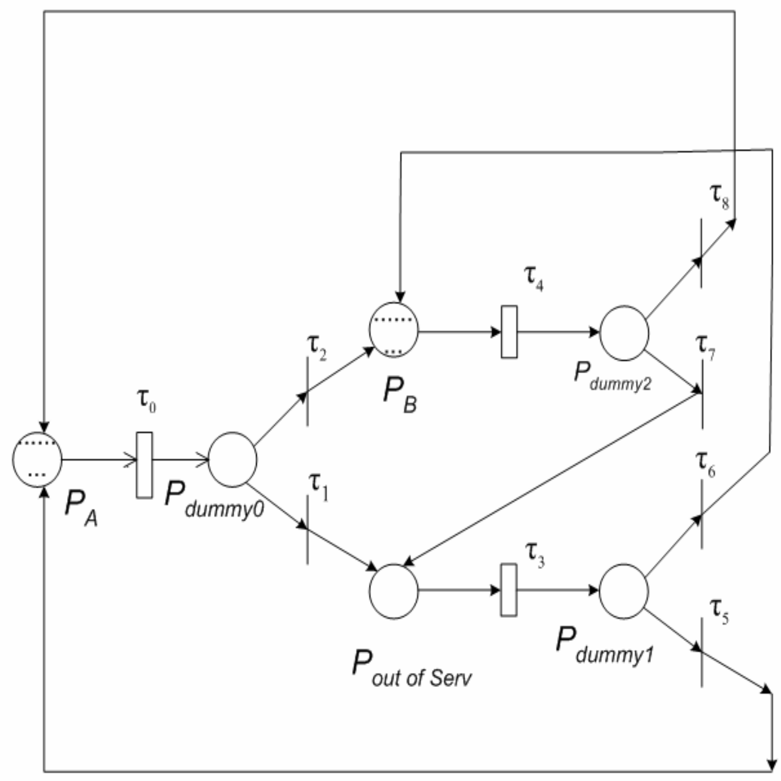
- <u>Service availability as a general problem:</u> The research methodology we follow can be extended to a broad class of wireless networks that are concerned with the problem of service availability. We list below some examples.
    - ➢ In infrastructure-based 802.11 wireless local area networks, the access point (AP) provides access to wireless nodes in the subnet. The connectivity provided by the AP can be thought of as service and can be viewed with a perspective of service availability. Given a mobility scenario for the wireless nodes served by a set of APs (in what is sometimes referred to as a “hot zone”), it would be interesting to develop analytical model for service availability seen by these wireless nodes.

  - In case of wireless sensor networks, certain nodes behaving as cluster heads aggregate data from other nodes and forward them to a central data processing entity. The frequency of service received by the sensor nodes that route data to these aggregating nodes can be modeled using analytical models similar to the ones we developed in this thesis.
  - When certain nodes in wireless ad hoc networks act as certificate authorities (CAs), they provide security services like issuing and managing security credentials and public keys to their client nodes. The notion of service availability in this context is the presence/absence of a CA that can issue security information to client nodes. Analytical models for such client nodes and their server CAs can be developed.

- <u>Analysis of mobility models:</u> In the analysis of the system models we developed, model parameterization was a crucial step. We relied on a simulation based study of the system behavior in order to compute the parameters of interest to characterize our models. Such dependence on simulation is undesirable in terms of the amount of time taken to develop these simulation models and computation of statistically confident averages of the parameters of interest. One way out of this problem is to analyze the underlying mobility models that govern these parameters of nodes in the wireless networks. This is an interesting area of research and can provide considerable insight in to the system behavior even before an analytical model of the entire system is developed.

- <u>Server model of the PBNM system:</u> One possible avenue for furthering our current research is to develop an analytical model of the system from a server's perspective. A preliminary GDT_SPN model for server behavior is presented below. Note that we have two policy servers and eighteen policy clients in the sample scenario of the system we studied (Chapter 3, Section 3.2.2). In the

GDT_SPN model developed from the perspective of a policy server, as shown in Figure 6.2.1, we have two servers, server A and server B, represented by places P_A and P_B. Initially we assume that there are nine tokens in each place corresponding to each server. This implies that there are nine clients being served initially by each server. The initial distribution of tokens can be varied without a consequential change to the resulting service availability.



Server Model for Service Availability in MANETs

Figure 6.2.1 Server model for service availability

Whenever a server, for instance server A, loses a connection with a client, the GDT_SPN model evolves with one token being removed from place P_A and put into either $P_{out\ of\ service}$ or into the other server's place P_B. The same applies for the other server. The transitions τ_0 and τ_4 are therefore representative of the amount of time a client is connected to a server on an average basis. The dummy places are used for simplicity of model development and have no physical interpretation. When a client is not being served by either of the two servers, it is out of service. This is represented by a token in place $P_{\text{out of service}}$. Transition τ_3 represents the amount of time that the clients remain out of service (according to some distribution associated with its random variable). Analyzing the model with appropriate distributions for the random variables associated with the timed transitions and switching probabilities for immediate transitions would lead to service availability metric given by,

$$Serv.Availability = \frac{Average\#Tokens(P_A) + Average\#Tokens(P_B)}{Total\#(Clients)} \quad (6.2.1)$$

Development of such a server model would consolidate the work and provide a comprehensive understanding of the system.

- Adding complexity to the model: Recall that in the models we developed for the PBNM system, we purposely omitted certain intricacies like redirection and delegation to simplify the analysis. There is a possibility to build upon the current model and add requisite amount of complexity to it in order to incorporate such nuances of the PBNM system.

- Analysis of underlying protocols of the PBNM system: In the development of our PBNM system, we chose COPS-PR as the underlying protocol for policy provisioning. One frontier of further research is to analytically model and evaluate the performance of various network provisioning protocols such as COPS-PR, ad hoc network management protocol (ANMP) and simple network management protocol (SNMP) and provide insight into which one performs better.

6.3 Thesis Summary

In this research, we developed analytical models for performance evaluation of the PBNM system for MANETs. We resorted to a simulation based study of the system to parameterize the system models. We went beyond the simple Markovian analysis to more closely model the processes describing the movement of clients in and out of clusters. One of the limitations of this research is its dependence on the simulation results for model parameterization. It can be seen from our analysis that the parameter values and the distributions of various random variables (time spent by client inside a cluster, time spent by client outside a cluster etc.) depend on the underlying random-waypoint mobility model. In our work, we obtain some insight on how mobility affects the coverage of a node by a k-hop cluster. On an end note, analytical modeling of systems serves as a good benchmark of their performance and is an important step in the design and deployment of network systems.

Bibliography

[1] F. Baskett, K. M. Chandy, R. Muntz, and F.G. Palacios, “Open, Closed, and Mixed Networks of Queues with Different Classes of Customers,” *Journal of the ACM (JACM)*, vol. 22 no. 2, pp. 248-260, April 1975.

[2] A. Bobbio and M. Telek, “Computational Restrictions for SPN With Generally Distributed Transition Times,” *In D. Hammer K. Echtle and D. Powell, Editors, First European Dependable Computing Conference (EDCC-1), Lecture Notes in Computer Science*, Springer-Verlag, vol. 852, pp. 131-148, 1994.

[3] A. Bobbio, A. Puliafito, M. Telek, and K. S. Trivedi, “Recent Developments in Non-Markovian Stochastic Petri Nets,” *Journal of Circuits, Systems, and Computers*, vol. 8, no. 1, pp. 119-158, February 1998.

[4] A. Bobbio and M. Telek, “Non-Exponential Stochastic Petri Nets: An Overview of Methods and Techniques,” *Computer Systems Science and Engineering,* vol. 13, no. 6, pp. 339-351, 1998.

[5] A Bobbio, A. Horváth, M. Scarpa, and M. Telek, “Acyclic Discrete Phase Type Distributions: Properties and a Parameter Estimation Algorithm,” *Performance Evaluation,* vol. 54, no. 1, pp.1-32, 2003.

[6] H. W. Braun, “Models of Policy Based Routing,” IETF RFC 1104, June 1989.

[7] T. Camp, J. Boleng and V. Davies, “A Survey of Mobility Models for Ad Hoc Network Research,” *Wireless Communication and Mobile Computing*, special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002.

[8] K. Chan *et al.*, “COPS Usage for Policy Provisioning (COPS-PR),” IETF RFC 3084, March 2001.

[9] D. Chen, S. Garg, and K.S. Trivedi, “Network Survivability Performance Evaluation: A Quantitative Approach With Applications in Wireless Ad-Hoc Networks**,”** *Proc. of the ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pp. 61-68, September 2002.

[10] G. Ciardo, J. Muppala, and K. S. Trivedi, “On the Solution of GSPN Reward Models,” *Performance Evaluation*, vol. 12, no. 4, pp. 237-254, July 1991.

[11] G. Ciardo, R. German and C. Lindemann, “A Characterization of the Stochastic Process Underlying a Stochastic Petri Net,” *IEEE Transactions on Software Engineering*, vol. 20, pp. 506-515, 1994.

[12] D. Clark, “Policy Routing in Internet Protocols,” IETF RFC 1102, May 1989.

[13] A. Cumani, “Esp-A Package for the Evaluation of Stochastic Petri Nets With Phase-type Distributed Transition Times,” *Proc. of International Workshop on Timed Petri Nets*, pp. 144-151, IEEE Computer Society Press no. 674, 1985.

[14] D. Dubois, “A Hierarchical Modeling System for Computer Networks,” *Proc. of Computer Network Performance Symposium*, pp. 147-155. 1982.

[15] D. Durham *et al.*, “The COPS (Common Open Policy Service) Protocol,” IETF RFC 2748, January 2000.

[16] D. Färnqvist, K. Johansson, and J. Hespanha, “Hybrid Modeling of Communication Networks Using Modelica,” *Proc. of the Radiovetenskap och Kommunikation (RVK)*, June 2002.

[17] M. Garetto, R.L. Cigno, and M. A. Marsan, “Closed Queuing Network Models of Interacting Long-lived TCP Flows,” *IEEE/ACM Transactions on Networking*, vol. 12, no.2. April 2004.

[18] H.J. Genrich, K. Lautenbach, and P.S. Thiagarajan, *Elements of General Net Theory*, Springer Verilag, 1980.

[19] R. German and A. Heindl, “Performance Evaluation of IEEE 802.11 Wireless LANs with Stochastic Petri Nets,” *Proc. of the International Workshop on Petri Nets and Performance Models*, pp. 44-53, September 1999.

[20] R. German, *Performance Analysis of Communication Systems-Modeling with Non-markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.

[21] W.J. Gordon and G.F. Newell, “Closed Queuing Systems With Exponential Servers.” *Operations Research,* vol. 15, pp. 254-265, 1967.

[22] E. Gressier, “A Stochastic Petri Net Model for Ethernet,” *Proc. of the International Workshop on Timed Petri Nets*, IEEE Computer Society Press, pp.296-306, 1986.

[23] P. J. Haas and G. S. Shedler, “Regenerative Stochastic Petri Nets,” *Performance Evaluation,* vol. 6, pp.189-204, 1989.

[24] P. J. Haas, *Springer Series in Operations Research. Stochastic Petri Nets Modeling, Stability, Simulation*, Springer, 2002.

[25] P. Harrison and N. Patel, *Performance Modeling of Communication Networks and Computer Architectures*, Addison-Wesley. 1992.

[26] J. Hayes, *Modeling and Analysis of Computer Communication Networks,* Plenum Press, 1984.

[27] Y. Hong, Y. Cao, H. Sun*,* and K.S. Trivedi, “RED Parameters and Performance of TCP Connections,” *Electronics Letters*, 2001.

[28] A. Horvath, A. Puliafito, M. Scarpa, M. Telek, and O. Tomarchio, “Design and Evaluation of Web-based Non-Markovian Stochastic Petri Net Tool,” *Proc. of the International Symposium on Computer and Information Science (ISCIS ’98)*, October 1998.

[29] A. Horváth, A. Bobbio, and M. Telek, “PhFit: A General Phase-type Fitting Tool,” *Proc. of TOOLS 2002, International Conference on Computer Performance Evaluation, Modelling Techniques and Tools,* pp. 82-91, April 14-17, 2002.

[30] INRIA Optimized Link State Routing Protocol (OLSR).
Available at: http://menetou.inria.fr/olsr/

[31] J. R. Jackson, “Networks of Waiting Lines”, *Operations Research*, vol. 5, pp. 518-521, 1957.

[32] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,* Wiley-Interscience, April 1991.

[33] A. Konrad, B.Y. Zhao, A. Joseph, and R. Ludwig, “A Markov-Based Channel Model Algorithm for Wireless Networks,” *Proc. of ACM Modeling Analysis and Simulation of Wireless and Mobile Systems 2001*, pp. 28-36, July 2001.

[34] D. Kosiur, *Understanding Policy-Based Networking*, John Wiley & Sons, Inc., 2001.

[35] *Lecture Notes in Computer Science, Net Theory and Applications*, Springer-Verlag, Berlin, Heidelberg, New York, 1980.

[36] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Addison-Wesley Publishing, 2nd ed., July 1993.

[37] C. Lindemann, “A Stochastic Performance Modeling Technique for Deterministic Medium Access Schemes,” *Proc. of IEEE Workshop on Future Trends in Distributed Computing Systems*, pp. 346-353, April 1992.

[38] C. Lindemann, *Performance Modelling With Deterministic and Stochastic Petri Nets,* John Wiley & Sons Ltd., 1998.

[39] A. Manjeshwar, Q. Zeng, D.P. Agrawal, “An Analytical Model for Information Retrieval in Wireless Sensor Networks Using Enhanced APTEEN Protocol,” *IEEE Transactions on Parallel and Distributed Systems,* vol.13, no. 12, pp. 1290-1302, December 2002.

[40] M. A. Marsan, G. Chiola, and A. Fumagalli, “An Accurate Performance Model of CSMA/CD Bus LAN,” *Advances in Petri Nets*, Springer-Verlag, pp. 146-161, 1987.

[41] M. A. Marsan, G. Balbo, A. Bobbio, *et al.,* “The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets,” *IEEE Transactions on Software Engineering*, vol. SE-15, pp. 832-846, 1989.

[42] M.A. Marsan, M. Meo, and M. Sereno, “GSPN Analysis of Dual-band Mobile Telephony Networks,” *Proc. of International Workshop on Petri Nets and Performance Models,* pp. 54-63, April 2000.

[43] M. K. Molloy, *On the Integration of Delay and Throughput Measures in Distributed Processing Models*, Technical Report, 1981.

[44] M. K. Molloy, “Performance Analysis Using Stochastic Petri Nets”, *IEEE Transactions on Computers*, vol. C-31, no.9, pp.913-917, 1982.

[45] M. K. Molloy, *Modeling and Analysis of LAN protocols Using Labeled Petri Nets,* Technical Report 84-15, Dept. of Computer Science, 1984.

[46] M. F. Neuts, *Matrix Geometric Solutions in Stochastic Models*, Johns Hopkins University Press, Baltimore, MD, 1981.

[47] C. A. O’Cinneide, “On Non-uniqueness of Representations of Phase-type Distributions,” *Stochastic Models*, vol. 5, pp. 247-259, 1989.

[48] A. Ost, C. Frank, and B.R. Haverkort, “Untersuchungen zum Verbindungsmanagement bei Videoverkehr mit Matrix-geometrischen Stochastichen

Petri-Netzen," *Proc. of GI-Workshop on Measurement, Modeling and Evaluation of Computer-Communications Systems*, 1997.

[49] A. Ost, and B.R. Haverkort, "Analysis of Windowing Mechanisms with Infinite State Stochastic Petri Nets," *Performance Evaluation Review*, vol. 26, pp. 38-46, 1998.

[50] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall Inc., 1981.

[51] K. Phanse, L.A. DaSilva, and S. Midkiff, "A Taxonomy and Experimental Evaluation of Policy Architectures for Bandwidth-constrained Networks," Internal Report, Virginia Tech, 2002.

Available: http://www.ee.vt.edu/~kphanse/publications.html.

[52] K. Phanse, and L.A. DaSilva, "Addressing the Requirements of QoS Management in Wireless Ad Hoc Networks," *International Journal on Computer Communications*, vol. 26, no. 12, pp. 1263-1273, July 2003.

[53] K. Phanse, *Policy-based Quality of Service Management in Wireless Ad Hoc Networks*, doctoral thesis, Virginia Polytechnic Institute and State University, Bradley Department of Electrical and Computer Engineering. Alexandria Research Institute, Virginia, 2003.

[54] K. Phanse and L.A. DaSilva, "Protocol Support for Policy-Based Management of Mobile Ad Hoc Networks," *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2004.

[55] IETF Policy Framework Working Group: http://www.ietf.org/html.charters/policy-charter.html.

[56] K. H. Prodromides and W.H. Sanders, "Performability Evaluation of CSMA/CD and CSMA/DCR Protocols under Transient Fault Conditions," *IEEE Transactions on Reliability*, vol. 42, no. 7, pp. 116-127, March 1993.

[57] A. Puliafito, A. Bobbio, M. Scarpa, M. Telek, "WebSPN: A Web-accessible Petri Net tool," *Proc. of conference on Web-based Modeling & Simulation*, January 1998.

[58] QualNet Network Simulator, http://scalable-networks.com.

[59] M. Reiser, "A Queuing Network Analysis of Computer Communication Networks with Window Flow Control," *IEEE Transactions on Communication*27, pp. 1199-1209, August 1979.

[60] M. Reiser, “Performance Evaluation of Data Communication Systems,” *Proc. of IEEE,* vol. 70, pp. 171-196, 1982.

[61] W. Reisig, *Petri Nets An Introduction*, Springer-Verlag, 1985.

[62] W. Reisig, G. Balbo, M. Silva, *et al.*, “Petri Nets 2000-Introductory Tutorial,” *Proc. of the International Conference on Applications and Theory of Petri Nets*, June 26-30 2003. http://www.daimi.au.dk/PetriNets/introductions/pn2000_introtut.pdf .

[63] C. Reutenauer, *The Mathematics of Petri Nets,* Prentice Hall International (UK) Ltd., 1990.

[64] T.G. Robertazzi, *Computer Network and Systems – Queuing Theory and Performance Evaluation*, 3rd ed., Springer, 2000.

[65] M. Scarpa, *Non-Markovian Stochastic Petri Nets With Concurrent Generally Distributed Transitions*, doctoral thesis, Universita di Torino, Dipartimento di Informatica, 1999.

[66] J. R. Spirn, “Network Modeling With Bursty Traffic,” *Proc. of Computer Network Performance Symposium*, pp. 21-28, 1982.

[67] Stochastic Petri Net Package.http://www.ee.duke.edu/~chirel/MANUAL/manual.pdf

[68] B. W. Stuck and E. Arthurs, *A Computer & Communications Network Performance Analysis Primer,* Prentice Hall Inc., 1985.

[69] M. Telek, Re: (PN) Non-Markovian Stochastic Petri-Nets. Email Correspondence (15 January 2004).

[70] H.Rölke, K.H. Mortensen, “Petri Nets Tools and Software,” http://www.daimi.au.dk/PetriNets/tools/.

[71] K.S. Trivedi and H. Sun, “Stochastic Petri Nets and Their Applications to Performance Analysis of Computer Networks,” *Proc. of the International Conference on Operational Research for a Better Tomorrow*, December 1998.

[72] D. Verma, *Policy-Based Networking: Architectures and Algorithms*, New Riders Publishing, 2000.

[73] J. W. Wong, “Queuing Network Model of Computer Communication Networks,” *ACM Computing Surveys (CSUR)* vol.10, no. 3, pp. 343-351, September 1978.

[74] C. Xiong, T. Murata, and J. Tsai, "Modeling and Simulation of Routing Protocol for Mobile Ad hoc Networks Using Colored Petri Nets," *Conference in Research and Practices in Information Technology Series, Proc. of Conference on Application and Theory of Petri Nets: Formal Methods in Software Engineering and Defense Systems,* pp. 145-153, 2002.

[75] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," *Proc. of the Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 1312-1321, March 2003.

[76] H. Zhu, I. Chlamtac, "An Analytical Model for IEEE 802.11e EDCF Differential Services," *Proc. of the International Conference on Computer Communications and Networks (ICCCN'03)*, October 2003.

Glossary

ADPH: Acyclic Discrete Phase Type Distribution

ANMP: Ad Hoc Network Management Protocol

AODV: Ad Hoc On-Demand Distance Vector

AP: Access Point

APTEEN: Adaptive Periodic Threshold-sensitive Energy Efficient sensor Network

ATM: Asynchronous Transfer Mode

CA: Certificate Authorities

CDF: Cumulative Distribution Function

CF: Canonical Form

COPS: Common Open Policy Service

COPS-PR: Common Open Policy Service for PRovisioning

CPH: Continuous Phase Type Distribution

CSMA/CD: Carrier Sense Multiple Access/Collision Detection

CSRQ: Client Service Request

CTMC: Continuous Time Markov Chain

DPH: Discrete Phase Type Distribution

DTMC: Discrete Time Markov Chain

DynaSeR: Dynamic Service Redundancy

EDCF: Enhanced Distributed Coordination Function

FSM: Finite State Machine

GDT_SPN: Generally Distributed Transition Stochastic Petri Net.

GSPN: Generalized Stochastic Petri Net

GUI: Graphical User Interface

HDLC: High Level Data Link Control

KA: Keep Alive

LAN: Local Area Network

MAC: Medium Access Control

MANET: Mobile Ad Hoc Network

OLSR: Optimized Link State Routing

PBNM: Policy Based Network Management

PDF: Probability Distribution Function

PDP: Policy Decision Point

PEP: Policy Execution Point

PH-SPN: Phase Type Stochastic Petri Net

PMT: Policy Management Tool

PN: Petri Net

prd: Preemptive Repeat Different

pri: Preemptive Repeat Identical

prs: Preemptive Resume

RED: Random Early Detection

SA: Service Advertisement

SNMP: Simple Network Management Protocol

SPN: Stochastic Petri Net

SPNP: Stochastic Petri Net Package

TCP: Transmission Control Protocol

TPPN: Timed Place Petri Net

TTL: Time To Live

TTPN: Timed Transition Petri Net

WLAN: Wireless Local Area Network

Appendix-I

Source code for analysis of GSPN model using SPNP

```
/**********************************************************
*************************************
* Markov model for analyzing Service Availability
                                           *
* Written on:06/21/03
                                                     *
* Written By: Aniket Bhat
                                 *
*
                                                          *
***********************************************************
************************************/
//Header files for accessing the CSPL language functions.
#include<stdio.h>
#include<user.h>
#include<stdlib.h>
#include<time.h>

//list of global variables (values for k =3)
double l1=0.010504;
double l2=0.010618;
double pi1=0.44891;
double pi2;

     /* Function to set the options for GSPN solving*/
     void options()
     {
          //options for intermediate files
          iopt(IOP_PR_RSET,VAL_YES);
          iopt(IOP_PR_RGRAPH,VAL_YES);
          iopt(IOP_PR_MC,VAL_YES);
          iopt(IOP_PR_PROB,VAL_YES);

          // options for numerical analytical solution.
          iopt(IOP_MC,VAL_CTMC);
          //options for simulation results.
          iopt(IOP_SIMULATION,VAL_NO);

     }
     /* Function to specify the network */
     void net()
     {
          //defining places
          place("p0");
          init("p0",1);
          place("p1");
          place("p2");

          //defining rates, transitions and probabilities
          rateval("t0",l1);
          rateval("t3",l2);

          imm("t1");
          probval("t1",pi1);
          pi2=1-pi1;


          imm("t2");
          probval("t2",pi2);


          //defining the input and output arcs

          iarc("t0","p0");
          oarc("t0","p1");
          iarc("t1","p1");
          iarc("t2","p1");
          oarc("t1","p0");
          oarc("t2","p2");
          iarc("t3","p2");
          oarc("t3","p0");

     }

     /* Function to check for validity of the GSPN markings
when      constructing reachability graph.*/

     int  assert()
     {
          return(RES_NOERR);
     }

     /* Function to initialize the reachability graph
construction.*/

     void ac_init()
     {
          pr_net_info();
     }

     /*Function to be called after reachability graph is
constructed    to ouput information about it*/

     void ac_reach()
     {
          pr_rg_info();
     }
     void ac_final()
     {

          solve(INFINITY);
          printf("Pi1 is %f\n",pi1);
          printf("Pi2 is %f\n",pi2);
          printf("Lamda 2 is %2f\n",l2);
          printf("Lamda 1 is %2f\n",l1);
          pr_mc_info();

     }
```