Evaluation of Internal Delay Inference in Queuing Networks

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

David E. Stoner
Bachelor of Science
The Pennsylvania State University, 2006

Director: Dr. Brian L. Mark, Professor
Department of Electrical and Computer Engineering

Spring Semester 2016
George Mason University
Fairfax, VA
# Table of Contents

<table>
<thead>
<tr>
<th>List of Tables</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>Abstract</td>
<td>x</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Background</td>
<td>2</td>
</tr>
<tr>
<td>2.1 Network Tomography</td>
<td>2</td>
</tr>
<tr>
<td>2.2 Markov Chains</td>
<td>3</td>
</tr>
<tr>
<td>2.2.1 Univariate Continuous-time Markov Chains</td>
<td>3</td>
</tr>
<tr>
<td>2.2.2 Bivariate Continuous-time Markov Chains</td>
<td>4</td>
</tr>
<tr>
<td>2.3 EM Algorithm</td>
<td>5</td>
</tr>
<tr>
<td>2.4 Queuing Networks</td>
<td>6</td>
</tr>
<tr>
<td>2.4.1 Open Jackson Networks</td>
<td>8</td>
</tr>
<tr>
<td>2.4.2 Queuing Networks with MMPP External Arrivals</td>
<td>10</td>
</tr>
<tr>
<td>3 Queuing Network Simulator</td>
<td>12</td>
</tr>
<tr>
<td>3.1 Queuing Network Simulator Design</td>
<td>12</td>
</tr>
<tr>
<td>3.1.1 Packet Length</td>
<td>12</td>
</tr>
<tr>
<td>3.1.2 Queuing Network Arrival Distributions</td>
<td>12</td>
</tr>
<tr>
<td>3.1.3 Server Processing Rate Distributions</td>
<td>13</td>
</tr>
<tr>
<td>3.1.4 Packet Routing</td>
<td>13</td>
</tr>
<tr>
<td>3.2 Python Implementation</td>
<td>14</td>
</tr>
<tr>
<td>3.2.1 SimPy Overview</td>
<td>14</td>
</tr>
<tr>
<td>3.2.2 Modeling Queuing Networks</td>
<td>14</td>
</tr>
<tr>
<td>4 Model and Algorithms for Network Delay Inference</td>
<td>16</td>
</tr>
<tr>
<td>4.1 Bivariate Markov Chain Model</td>
<td>16</td>
</tr>
<tr>
<td>4.2 Application of the EM Algorithm</td>
<td>17</td>
</tr>
<tr>
<td>4.2.1 Initialization</td>
<td>18</td>
</tr>
<tr>
<td>4.2.2 EM Algorithm</td>
<td>19</td>
</tr>
<tr>
<td>4.2.3 Calculation of Equations Containing Integrals with Exponentials</td>
<td>20</td>
</tr>
</tbody>
</table>

iii
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Link Definitions and Transition Probabilities</td>
<td>24</td>
</tr>
<tr>
<td>5.2</td>
<td>Jackson Network Exponential Service Rates for Source/Intermediate Nodes</td>
<td>26</td>
</tr>
<tr>
<td>5.3</td>
<td>MMPP/D/1 Network Deterministic Service Rates for Source/Intermediate Nodes</td>
<td>28</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Server in a Queuing Network</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>M/M/1 Single Queue State Diagram</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Markov Modulated Poisson Process Model with ( r = 2 )</td>
<td>11</td>
</tr>
<tr>
<td>5.1</td>
<td>Topology of Test Network</td>
<td>23</td>
</tr>
<tr>
<td>5.2</td>
<td>Jackson Network Source Destination Delay Results</td>
<td>26</td>
</tr>
<tr>
<td>5.3</td>
<td>Jackson Network Link 6 Delay Density Unstructured</td>
<td>27</td>
</tr>
<tr>
<td>5.4</td>
<td>Jackson Network Link 6 Delay Density Structured</td>
<td>27</td>
</tr>
<tr>
<td>5.5</td>
<td>Jackson Network Estimated Probability Error of Packet Routing Unstructured</td>
<td>28</td>
</tr>
<tr>
<td>5.6</td>
<td>Jackson Network Estimated Probability Error of Packet Routing Structured</td>
<td>28</td>
</tr>
<tr>
<td>5.7</td>
<td>MMPP/D/1 Source Destination Delay Results</td>
<td>29</td>
</tr>
<tr>
<td>5.8</td>
<td>MMPP/D/1 Link 6 Delay Density Unstructured</td>
<td>29</td>
</tr>
<tr>
<td>5.9</td>
<td>MMPP/D/1 Link 6 Delay Density Structured</td>
<td>29</td>
</tr>
<tr>
<td>5.10</td>
<td>Estimated Probability Error of Packet Routing Unstructured</td>
<td>30</td>
</tr>
<tr>
<td>5.11</td>
<td>Estimated Probability Error of Packet Routing Structured</td>
<td>30</td>
</tr>
<tr>
<td>5.12</td>
<td>MMPP/D/1 Source Destination Delay Results with Exponential Packet Length</td>
<td>31</td>
</tr>
<tr>
<td>5.13</td>
<td>MMPP/D/1 Link 6 Delay Density with Exponential Packet Length Unstructured</td>
<td>32</td>
</tr>
<tr>
<td>5.14</td>
<td>MMPP/D/1 Link 6 Delay Density with Exponential Packet Length Structured</td>
<td>32</td>
</tr>
<tr>
<td>5.15</td>
<td>Estimated Probability Error of Packet Routing Unstructured</td>
<td>32</td>
</tr>
<tr>
<td>5.16</td>
<td>Estimated Probability Error of Packet Routing Structured</td>
<td>32</td>
</tr>
<tr>
<td>A.1</td>
<td>Jackson Network Estimated Source/Destination Delay Density - Unstructured</td>
<td>34</td>
</tr>
<tr>
<td>A.2</td>
<td>Jackson Network Estimated Source/Destination Delay Density - Structured</td>
<td>34</td>
</tr>
<tr>
<td>A.3</td>
<td>Jackson Network Estimated Link Delay Density for Link 0 - Unstructured</td>
<td>34</td>
</tr>
<tr>
<td>A.4</td>
<td>Jackson Network Estimated Link Delay Density for Link 0 - Structured</td>
<td>34</td>
</tr>
<tr>
<td>A.5</td>
<td>Jackson Network Estimated Link Delay Density for Link 1 - Unstructured</td>
<td>35</td>
</tr>
<tr>
<td>A.6</td>
<td>Jackson Network Estimated Link Delay Density for Link 1 - Structured</td>
<td>35</td>
</tr>
<tr>
<td>A.7</td>
<td>Jackson Network Estimated Link Delay Density for Link 2 - Unstructured</td>
<td>35</td>
</tr>
</tbody>
</table>
Abstract

EVALUATION OF INTERNAL DELAY INERENCE IN QUEUING NETWORKS

David E. Stoner
George Mason University, 2016
Thesis Director: Dr. Brian L. Mark

Statistical inference of internal computer network characteristics using only externally made measurements is extremely useful in the analysis of highly complex networks. This thesis seeks to implement and test an expectation-maximization (EM) algorithm that uses these observations to estimate total end-to-end network delay density, link delay density and probability of route selection. The EM algorithm in question was tested using source/destination delays generated from a custom queuing network simulator. The parameters of the queuing network were varied in order to determine the effectiveness of the algorithm on Jackson-type networks as well as more realistic networks. The subsequent results of the algorithm are compared against the actual network simulation data to evaluate the performance of the algorithm.
Chapter 1: Introduction

Performing delay inference of queuing networks is a subtopic of the larger field of network tomography. In one of the earliest works on network tomography, the traffic intensity in networks is estimated using a series of observations of the packets flowing through that network \[25\]. Recently, an algorithm to estimate link delays in a network with an unstructured network topology is developed in \[12\] where traffic moving through a network is modeled as a partially observable bivariate Markov chain. In \[12\], this algorithm estimates the partially observable bivariate Markov chain by using a series of source-destination delays sampled from the network’s traffic as inputs to the EM algorithm.

The algorithm in \[12\] is evaluated using observation data generated from a high-order bivariate Markov chain. First, a high order infinitesimal generator matrix was randomly generated and considered the true model of the ”network.” The generator was used to create source-destination delay samples. These samples were fed into the EM Algorithm using a different randomly generated generator matrix of lower order. The resulting generator estimate was compared to the original in \[11\] and conclusions about the accuracy of the algorithm were made. This thesis seeks to take the evaluation of \[11\]’s proposed algorithm a step further by feeding simulated queuing network source-destination measurements into the EM Algorithm. The resulting EM Algorithm estimates are then used to estimate the internal characteristics of the queuing network.

Chapter 2 provides a brief background on network tomography, bivariate Markov chains, the EM algorithm and queuing networks. Chapter 3 details how the queuing network simulations were constructed. Chapter 4 summarizes the link delay estimation algorithm from \[12\]. Chapter 5 summarizes the results of applying the algorithm from \[12\] against the true results from the queuing network simulations. Chapter 6 provides some concluding remarks and recommendations.
Chapter 2: Background

2.1 Network Tomography

Depending on the complexity and size of a network, it is sometimes very difficult, if not impossible, to be able to look inside the network and identify bottlenecks and congestion points. For instance, the Internet is a fast growing network that has very little centralized control. Because of this decentralized authority, obtaining information about specific links means working with various Internet Service Providers who may view network statistics as being confidential [7].

Network tomography is a field of study that aims to perform inference of network performance parameters based on a series of observations of the network. The term network tomography originated in [25] where Vardi presented an approach to estimate node-to-node traffic intensity using the EM Algorithm. Since its conception, there are two general forms of network tomography that are actively being studied; link estimation based on end-to-end observations and traffic intensity estimation based on link traffic measurements [6].

Link estimation, which is the focus of [11], attempts to estimate the delay or loss between two nodes that are directly connected within the network (also referred to as a link). A method for estimating the loss probability associated with each link in a network by sending TCP messages into a network and then collecting statistics on the number of TCP acknowledgement packets are received is presented in [8]. Alternatively, [12], presents a method to estimate individual link delay distributions by injecting packets into a network and measuring the sojourn time of the packet as it travels through the network. Similarly, Internet Tomography Measurement Systems (ITMS) are systems that attempt to measure internal characteristics of the Internet based off probes injected from the boundaries of the Internet. An example of an ITMS is a constellation of devices connected to the Internet.
around a localized region that send test traffic to each other. This traffic is used to provide a general estimate of the performance of the Internet in that particular area [23]. Another approach is developed in [4] that measures multicast traffic through a network at a series of receivers and then estimates loss rates on internal links using a maximum likelihood estimator.

Traffic intensity estimation was studied in [25] where traffic is measured at each node in the network which allows for estimation of traffic intensity between any two nodes in the network. Incoming and outgoing traffic at each router in a network may be measured. For small networks, [5] presented a method to estimate origin-destination traffic counts accurately using a maximum likelihood estimation method. In [20], a method is presented for inferring origin-destination traffic counts using a psuedo likelihood estimation approach.

In addition to computer network estimation, Vardi suggested in [25] that the study of network tomography has far reaching applications such as traffic engineering and urban planning.

2.2 Markov Chains

Markov chains are a stochastic model that describes a statistical process where the future state of the process depends only on the current state, and is independent on the past states [3]. All of the events that have taken place prior are consolidated into a single state. This state changes over time according to given probabilities. Markov Chains may be used to model processes that exist in discrete-time and continuous-time. However, this thesis is primarily interested in continuous-time Markov chains.

2.2.1 Univariante Continuous-time Markov Chains

Univariante continuous-time Markov chains, or simply continuous-time Markov chains, are used to model a single random process that involves transitions between states that occur in continuous-time. In these models, state transitions occur at continuous intervals. The
arrival and processing times of a packet in a queuing network are often modeled using continuous-time Markov Chains [14].

Let \( Z = \{ Z(t), t \geq 0 \} \) denote a continuous-time Markov chain that exists on the finite state space \( Z \). Since \( Z \) consists of a finite number of states, then \( Z \) is considered finite-state. The Markov chain satisfies the following:

\[
P(Z(t_{n+1}) = j | Z(t_n) = i, Z(t_{n-1}) = i_{n-1}, ..., Z(t_0) = i_0) = P(Z(t_{n+1}) = j | Z(t_n) = i)
\]

where \( i, j \in Z \). This is also known as the Markov Property. The Markov Property basically states that the next state of \( Z \) is dependent only on the current state of \( Z \) and independent of all previous states of \( Z \). The Markov chain \( Z \) is homogeneous if \( P(Z(t) = j | Z(s) = i) = P(Z(t-s) = j | Z(0) = i) \) for all \( t, s \geq 0 \).

### 2.2.2 Bivariate Continuous-time Markov Chains

Let \( Z = \{ Z(t), t \geq 0 \} \) denote a finite space, homogeneous continuous-time bivariate Markov chain. The bivariate Markov chain \( Z \) is made up of two random processes \( X \) and \( S \), where \( X \) is observable and exists on the state space \( X = \{ 1, 2, ..., d \} \) while \( S \) exists on the state space \( S = \{ 1, 2, ..., r \} \). The process \( S \) cannot be observed but affects the statistical properties of the observed process \( X \). While neither \( X \) nor \( S \) are necessarily Markov, the processes together are considered jointly Markov [10] and the resulting combination \( Z = (X, S) \) is modeled as a Markov process with state space \( Z = X \times S \). Note that \( Z \) changes state when \( X \) changes state, \( S \) changes state or both \( X \) and \( S \) change state simultaneously. We denote any given state of \( Z \) by \( (a, i) \in Z \) for any \( a \in X \) and \( i \in S \).

The transition matrix, denoted as \( P_t \), of the bivariate Markov chain describes the probability of \( Z \) transitioning from one state to another at time \( t \). The matrix \( P_t \) is continuous and differentiable at \( t = 0 \). The derivative of \( P_t \) at \( t = 0 \) is the infinitesimal generator.
matrix of the bivariate Markov chain and is given by

\[ G = \lim_{t \to 0} \frac{1}{t} (P_t - I) \]  

(2.2)

where the non-diagonal elements of \( G \) are positive and the diagonal elements of \( G \) are such that the sum of any row of \( G \) equals zero. In a bivariate Markov chain, the infinitesimal generator matrix \( G = \{g_{ab}(ij)\} \) has the following properties [10]:

\[ -\infty \leq g_{aa}(ii) \leq 0 \]

\[ 0 \leq g_{ab}(ij) \leq \infty \text{ whenever } (a, i) \neq (b, j) \]  

(2.3)

\[ \sum_{b, j} g_{ab}(ij) \leq 0 \text{ for all } (a, i) \in \mathbb{Z} \text{ with equality if } \sup_{(a, i)} \{-g_{aa}(ii)\} < \infty \]

The matrix \( G \) may be split into \( r \times r \) sub-matrices such that \( G = \{G_{ab}; a, b \in \mathbb{X}\} \) and \( G_{ab} = \{g_{ab}(ij); i, j \in \mathbb{S}\} \). If each \( \mathbb{X} \) state is associated with a different \( \mathbb{S} \) state, \( G \) may be partitioned into submatrices of varying size. In this case, \( G \) is partitioned such that \( G = \{G_{ab}; a, b \in \mathbb{X}\} \) and \( G_{ab} = \{g_{ab}(ij); i \in \mathbb{S}_a, j \in \mathbb{S}_b\} \). This approach to partitioning will be used in chapter 4.

2.3 EM Algorithm

The estimation-maximization (EM) algorithm is an iterative algorithm for performing maximum likelihood estimation [2]. The algorithm, first proposed in [9], is primarily useful for computing estimates from incomplete data. The algorithm is broken into two major steps, the evaluation of conditional expectation (or the E-step) and the maximization (or the M-step).

Let \( Z \) be the set of potential results of an experiment with a density distribution of \( p_{\phi} \)
and let $Y$ be the set of observations of that experiment where $Y_k$ is the $k$th observation from the experiment. After $n$ iterations of the algorithm, the E-step of the $n+1$ iteration consists of finding the following:

$$\sum_{k=1}^{K} E_{\phi_n} \{ \log p_\phi(Y_k, \tilde{Z}_k) | Y_k \}. \quad (2.4)$$

The M-step consists of maximizing equation (2.4).

The EM algorithm generates a series of parameter estimates with increasing likelihood. With each iteration of the algorithm, the likelihood increases. The algorithm continues to run until the likelihood remains constant or only changes slightly.

### 2.4 Queuing Networks

A queuing network is a system that is made up of two sets of entities, customers and servers [19]. Servers are interconnected and make up a system. Customers enter the system via a server, are processed by the server and then routed to another server. In general, a server can only process a finite number of customers simultaneously. Because of this, additional customers for this server wait in a queue until the server becomes available.

![Figure 2.1: Server in a Queuing Network](image)

Figure 2.1 shows the components of a server in a queuing network. The server is made up of two components, the queue and the service node. Customers arrive (at a rate of $\lambda$) at the server’s queue and wait until the service node becomes available. The customer then
enters the service node where it is processed at a rate of $\mu$. Once the customer completes being processed, the customer leaves the server.

Queuing networks may be either open or closed. In an open queuing network, customers arrive from an outside source and eventually leave the network. In a closed queuing network, a finite number of customers exist within the network and the number of customers within the network remains constant.

Queuing networks are described by three main attributes; the input-process, the queue-discipline and the service mechanism [17]. The input-process describes how customers arrive at servers. The queue discipline is how customers behave in the queue prior to being processed by the server. The service-mechanism is how the service node processes a customer before the customer is allowed to move on. The queuing-discipline for all networks in this thesis are a "first come first serve" queue (also commonly known as a first in, first out queue).

In [18], Kendall introduces a shorthand (that is expanded upon by [19]) which is commonly used to describe the behavior of a queuing network. This shorthand describes the arrival time distribution ($A$), the service time distribution ($S$), the number of customers a server can process at a given time ($m$), the maximum number of customers that can wait in the queue ($N$) and size of the input source ($K$). These values are all combined together in the following notation: $A/S/m/N/K$. In the case when the queue size is infinite and the input source is infinite, these values are left off and the notation becomes simply $A/S/m$.

The symbols $A$ and $S$ are subsequently replaced with various letters to describe the distributions. There are several distributions that can be used to model both arrival rates and service rates, however, this thesis focuses on Poisson arrivals (denoted by M), deterministic arrivals (denoted by D) and Markov Modulated Poisson Process arrivals (denoted by MMPP).

The transition probability matrix of a queuing network contains the probability that a customer will be routed from node $a$ to node $b$ once node $a$ has completed processing the
customer. For a network with \( n \) nodes,

\[
P = \begin{bmatrix}
p_{11} & p_{12} & p_{13} & \cdots & p_{1n} \\
p_{21} & p_{22} & p_{23} & \cdots & p_{2n} \\
p_{31} & p_{32} & p_{33} & \cdots & p_{3n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
p_{n1} & p_{n2} & p_{n3} & \cdots & p_{nn}
\end{bmatrix}
\]  

(2.5)

where \( P \) is the \( n \times n \) transition probability matrix of the network and \( p_{ij} \) is the probability of a customer being routed from server \( i \) to server \( j \). Additionally,

\[
S = \begin{bmatrix}
p_{s1} \\
p_{s2} \\
p_{s3} \\
\vdots \\
p_{sn}
\end{bmatrix}
\]  

(2.6)

is the \( 1 \times n \) source transition probability row vector where \( p_{si} \) is the probability of a customer entering the network at server \( i \). If a customer cannot enter the network at server \( i \), then \( p_{si} \) is simply zero.

### 2.4.1 Open Jackson Networks

The simplest network we consider is the Open Jackson Network, named for James Jackson who first described it in [16]. In this network, customers arrive to the network according to a Poisson process with given rate \( \lambda \). Servers process each customer for a service time that is exponentially distributed about \( \mu_i \). Figure 2.2 shows the state changes of a single server within the network as packets arrive at rate \( \lambda_i \) and are processed at rate \( \mu_i \).
Because of the simple calculations necessary, it is possible to calculate the average sojourn time of a customer in the system simply by knowing the transition probability matrix $P$, the arrival rate of customers to the system $\lambda$ and the service node rates $\mu_i$. Equation 6.2-30 from [19] provides the total arrival rate $\lambda_i$ to server $i$. For simplicity, we set $e_i \lambda(n) = \lambda_i$ and equation 6.2-30 becomes

$$\lambda_i = p_{si} \lambda + \sum_{j \in M} p_{ji} \lambda_i.$$  

(2.7)

If we let

$$\Lambda = \begin{bmatrix} \lambda_1 \\
\lambda_2 \\
\lambda_3 \\
\vdots \\
\lambda_n \end{bmatrix},$$

(2.8)
then equation (2.7) can be rewritten as the following system of equations:

\[
\Lambda = \begin{bmatrix}
    p_1 \lambda + \sum_{j \in M} p_{j1} \lambda_1 \\
p_2 \lambda + \sum_{j \in M} p_{j2} \lambda_2 \\
p_3 \lambda + \sum_{j \in M} p_{j3} \lambda_3 \\
    \vdots \\
p_n \lambda + \sum_{j \in M} p_{jn} \lambda_n
\end{bmatrix} = \lambda S^T + P^T \Lambda. \quad (2.9)
\]

Reorganizing equation (2.9) yields

\[
\Lambda = -\lambda (P^T - I)^{-1} S^T \quad (2.10)
\]

where \( I \) is an identity matrix of dimension \( n \times n \). The individual rates for each node are easily computed using equation (2.10). With these rates, the mean sojourn time per server (\( T_i \)), the mean number of customers per server (\( N_i \)) and the mean sojourn time for the entire network (\( T \)) can be calculated using the following equations:

\[
N_i = \frac{\lambda_i}{\mu_i} - \lambda_i \quad (2.11)
\]

\[
T_i = \frac{1}{\mu_i} - \lambda_i \quad (2.12)
\]

\[
T = \frac{1}{\lambda} \sum_{j \in M} \frac{\lambda_i}{\mu_i} - \lambda_i \quad (2.13)
\]

### 2.4.2 Queuing Networks with MMPP External Arrivals

Unfortunately, in data networks, customers do not generally arrive according to a Poisson, but arrive in bursts \[13\]. Markov Modulated Poisson Process (MMPP) can better model
this particular arrival process [15].

The Markov Modulated Poisson Process is simply a Poisson process with time-varying rate governed by a Markov chain with \( r \) states. Each state of the Markov chain has an associated customer arrival rate \( \lambda_i \). When the Markov chain changes from state \( i \) to state \( j \), the arrival rate changes from \( \lambda_i \) to \( \lambda_j \). Figure 2.3 shows the state changes for an MMPP arrival with \( r = 2 \). In this diagram, the Markov chain changes states from 1 to 2 at a rate of \( q_{12} \) transitions back at a rate of \( q_{21} \). While in state 1, customers arrive at a rate of \( \lambda_1 \) and while in state 2, customers arrive at a rate of \( \lambda_2 \).

![Markov Modulated Poisson Process Model with r = 2.](image)

Figure 2.3: Markov Modulated Poisson Process Model with \( r = 2 \).

It is important to note that MMPP arrivals are memoryless. Therefore, when modeling using a MMPP arrival process, when the Markov Chain state changes, any customers that would have arrived at the old rate are eliminated and new arrivals are generated using the new rate.
Chapter 3: Queuing Network Simulator

3.1 Queuing Network Simulator Design

To perform the analysis of the EM Algorithm in [12], a queuing network simulator was required. This simulator was designed to allow the simulation of various internal network characteristics and various external network characteristics.

3.1.1 Packet Length

The simulator first creates \( n \) packets. Each packet is created with a packet length \( l \) that is either exponentially distributed or deterministic. This property is chosen at the beginning of the simulation and the decision applied to all packets in the simulation. In the case of deterministic, all packets have a constant length. In the case of exponentially distributed packet lengths, each packet length is randomly generated and assigned. When packets are generated using exponentially distributed lengths, the packet length is not constrained to the set of integers, therefore it is entirely possible in this simulation to have packets of fractional size.

3.1.2 Queuing Network Arrival Distributions

Packets in the simulator arrive at the network at one of two different arrival distributions; Poisson or Markov Modulated Poisson Process (MMPP). As packets are created, a random start time is generated using one of the two described distributions. This start time is added to the prior packet’s start time in order to determine when the packet enters the network. When it is time for the packet to enter the network, the packet may enter any of the source nodes. The source node is chosen at random according to the probability distribution given by equation (2.6).
Poisson arrivals are simulated by generating a start time for each customer using an exponential distribution about a constant mean arrival rate.

MMPP arrivals are simulated similarly to Poisson arrivals. However, the state of the arrival rate needs to be tracked and updated. Multiple arrival rates are defined along with state transition rates. State transition rates define how often, on average, the state should transfer to another state. Initially, an arrival rate state is chosen at random. The time until state change is calculated using an exponentially distributed random variable. Arrival times are generated in the same manner as Poisson arrivals. Each new arrival time is checked against the desired state change time. If the arrival time occurs after the state change, the arrival time is discarded. A new state is randomly selected and a new arrival time is generated using the new state’s arrival rate. This arrival time is added to the state change time to ensure that the arrivals are memoryless. The next state change time is generated and arrival times continue to be generated until the next state change.

3.1.3 Server Processing Rate Distributions

Upon arrival in the network, the packet immediately enters a server. Each server has an infinite queue and a service node that processes the packet at a given rate. The rate at which a service node processes a packet is either deterministic or exponentially distributed. In the case of deterministic, the packet is held for $\frac{1}{\lambda}l$ seconds before being allowed to move on where $\lambda$ is the service node rate and $l$ is the packet length. If the node provides exponential service, the packet is held on to for $dl$ seconds where $d$ is an exponentially distributed delay about a given mean. As soon as the packet is released from the service node, the next packet in the queue enters the service node.

3.1.4 Packet Routing

Once the packet has been processed by the server, the packet is routed to another server based on the network’s transition probability matrix. This matrix subsequently describes the topology of the network. If there is zero probability that the packet will be routed to
another server, the packet is routed to a destination server where it is immediately processed and exits the network.

### 3.2 Python Implementation

For this thesis, Python was used to implement both the queuing network simulator and the EM Algorithm. Specifically, Anaconda 2.3.0 (64-bit) was used which provides Python 3.4.3 along with several libraries.

#### 3.2.1 SimPy Overview

One package not included in the Anaconda distribution is SimPy. SimPy is a discrete event simulator package for Python. Simulations using SimPy are comprised of multiple processes that interact with various resources that exist within an environment. The environment controls time as it progresses, processes are used to model active components within the environment and resources are utilized by the processes in the environment. SimPy allows multiple processes to be simulated in parallel.

For example, [24] uses SimPy to estimate the number of parking spots required in a parking lot given that cars enter and leave the parking lot at various rates. Each car is modeled by a process and each parking spot is considered a resource. The car enters the parking lot after a random period of time, takes a random amount of time to find a parking spot, parks in the parking spot for a random period of time (thereby utilizing that particular resource) and leaves the parking lot after another random amount of time.

#### 3.2.2 Modeling Queuing Networks

Similarly, SimPy can be used to model a simple queuing network. Each packet (or customer) that is introduced into the network is modeled as a process and every server within the network is modeled as a resource. Start times and lengths for all packets are generated using the methods described in Section 3.1. Once complete, each packet process is created sequentially.
Once the process starts, the process is delayed using the SimPy environmental `timeout` command. The `timeout` command causes the process to wait the prescribed amount of time before progressing. After this initial delay, the customer is routed to one of the source servers according to the distribution given in equation (2.6). Once the customer is introduced to a server, the customer waits until the server becomes available. SimPy’s resource `request` function causes a process to wait indefinitely until the resource becomes available. This simulates the queue at each server. As customers arrive at a server, they are queued up until the server finishes processing its current customer. Once the current customer has been processed, SimPy allows the next packet in the queue to be processed by the service node.

The service time within the service node is simulated by delaying the process again by some amount. Once this timeout is completed, the customer is transferred to another server. If the next server is a destination server, the packet immediately leaves the network, the total sojourn time is recorded and the packet is discarded (i.e., the process terminates). In the simulation, destination nodes are not defined. Instead, when a packet reaches the last server, the server processes the packet (like all other source/intermediate servers) and then the packet exits the network.
Chapter 4: Model and Algorithms for Network Delay Inference

In [12], an algorithm is developed for estimating the link delay densities, routing probabilities and probability of source-destination paths in a unstructured network with random routing. In this Chapter, we provide a brief overview of this algorithm and the associated models.

4.1 Bivariate Markov Chain Model

Traffic in the network is modeled as a bivariate Markov chain \( Z = (X,S) \) where the states of the \( X \)-chain are nodes of the network and the states of the \( S \)-chain are an unobservable underlying process that impacts the sojourn time in each state of the \( X \)-chain. The set of all nodes \( X \) is divided into two sub-sets; \( X_1 \) which contains all source/intermediate nodes in the network and \( X_2 \) which contains all destination nodes in the network. Network traffic, or packets, enter the network at source nodes and are randomly routed to intermediate nodes (nodes interior to the network) or destination nodes. Packets routed to intermediate nodes are randomly routed to other intermediate nodes or destination nodes. Once a packet reaches a destination node, the packet immediately leaves the network. Destination nodes introduce negligible delay to a packet entering it.

Let \( X_1 = \{1,\ldots,d_1\} \) and \( X_2 = \{d_1+1,\ldots,d\} \) where \( d \) is the number of nodes in the network and \( d_2 = d - d_1 \). Because we are only concerned with the sojourn times of source and intermediate nodes, the traffic between those nodes is modeled with \( r \geq 1 \) whereas traffic between destination nodes is non-existent so they can simply be modeled as a scalar \( (r = 1) \). With this in mind, the infinitesimal generator \( G \) of the bivariate Markov chain is partitioned as follows:
\[
G = \begin{pmatrix}
G_{11} & \cdots & G_{1,d_1} \\
\vdots & \ddots & \vdots \\
G_{d_1,1} & \cdots & G_{d_1,d_1}
\end{pmatrix}
\begin{pmatrix}
h_1 & \cdots & h_{d_2} \\
e_{1} & -\epsilon_1 & \ddots \\
\vdots & & \ddots & -\epsilon_{d_2}
\end{pmatrix}
\] (4.1)

where \(\{G_{ab}, a, b = 1, \ldots, d_1\}\) are \(r \times r\) matrices, \(\{h_l, l = 1, \ldots, d_2\}\) are \(rd_1 \times 1\) column vectors and \(\{e_l, l = 1, \ldots, d_2\}\) are \(1 \times rd_1\) row vectors, and \(\{\epsilon_l\}\) are positive scalars. The partitions of \(G\) can then be identified using the following:

\[
G = \begin{pmatrix}
H_{cc} & H_{cd} \\
H_{dc} & H_{dd}
\end{pmatrix}
\] (4.2)

where the subscript of \(c\) is related to source/intermediate nodes and \(d\) is related to destination nodes.

The initial distribution of the bivariate Markov chain is denoted by:

\[
\nu = (\nu_1, \ldots, \nu_d)
\] (4.3)

where \(\nu_i\) is a \(1 \times r\) row vector representing the initial state of node \(i\). The system cannot be initialized with packets in the destination nodes, therefore, \(\nu_{d_1+1} = \ldots = \nu_d = 0\). \(\mu = (\nu_1, \ldots, \nu_{d_1})\) is defined as the initial state of all source/intermediate nodes.

4.2 Application of the EM Algorithm

The EM Algorithm uses \(K\) independent source-destination delay measurements of the network. \(Y = \{Y_1, Y_2, \ldots, Y_k\}\) is the set of measurements made where \(Y_i\) is the elapsed time
starting when the packet is introduced to the network and stopping when the packet reaches a destination node.

The goal of the EM Algorithm is the estimation of the parameter $\phi$ of the bivariate Markov chain through an iterative process where the parameter $\phi$ converges to the eventual estimate. The parameter $\phi$ is made up of the off-diagonal elements of $H_{cc}$, the elements of $H_{cd}$ and the initial distribution $\mu$.

4.2.1 Initialization

To initialize the generator, the non-diagonal elements of $H_{cc}$ and $H_{cd}$ are uniformly distributed between 0 and some upper limit. Networks modeled with the EM Algorithm fall into two categories; structured and unstructured. For structured networks, (i.e., known network topology), $G_{ab}$ is set to 0 when there is no direct link between the source/intermediate nodes $a, b \in X_1$. Similarly, elements in $H_{cd}$ are set to 0 if there is no link between the source/intermediate node ($a \in X_1$) and the destination node ($b \in X_2$). For unstructured networks (i.e. unknown topology) however, all $G_{ab}$ and $h_l$ are randomly initialized.

The diagonal elements of $H_{cc}$ are set so the sum of the row equals 0 (equation (2.3)). Packets do not move from one destination node to any other node, therefore the sojourn times for these nodes needs to be small (simulating a virtually instantaneous service time). Because we model these nodes with $r = 1$, the destination node is modeled as an exponential node with sojourn time of, $1/\epsilon_i$. To make this sojourn time instantaneous, $\epsilon_i$ is chosen to be a large number; [12] recommends $10^5$. The values of $H_{dc}$ are chosen as $\{e_i = \epsilon_i \mu, i = 1, ..., d_2\}$ so the initial distribution remains $\mu$ after the packet leaves the destination node. The parameter $\mu$ is simply initialized using a uniform distribution.
4.2.2 EM Algorithm

The initial state estimate is updated using

$$
\hat{\nu}_{ai} = \frac{1}{K} \sum_{k=1}^{K} \nu_{ai} Y_{ai} e^{H_{yc}y_k} H_{cd} \mathbf{1} \quad (4.4)
$$

where $Y_{ai}$ is a row vector of dimensions $1 \times r d_1$ with a 1 in the $a + i$ position and 0’s everywhere else and $\mathbf{1}$ is a $d_2 \times 1$ column vector of all ones.

Let $D^a_{i}(y_k)$ be the sojourn time in state $(a, i)$. $D^a_{i}(y_k)$ is estimated using

$$
\hat{D}^a_{i}(y_k) = \int_0^{y_k} \frac{[\mu e^{H_{yc}y} 1_{ai}] [1_{ai}' e^{H_{yc}(y_k-t)} H_{cd} \mathbf{1}]}{\mu e^{H_{yc}y} H_{cd} \mathbf{1}} dt \quad (4.5)
$$

where $1_{ai}$ is a column vector of dimensions $r d_1 \times 1$ with a 1 in the $a + i$ position and 0’s everywhere else and $1_{ai}'$ is a row vector of dimensions $1 \times r d_1$ with a 1 in the $a + i$ position and 0’s everywhere else.

Let $M^{ab}_{ij}(y_k)$ be the number of jumps of $Z$ from state $(a, i)$ to $(b, j)$ in the period $[0, y_k]$. An estimation of the number of jumps over this time period where $a, b \in X_1$ is

$$
\hat{M}^{ab}_{ij}(y_k) = \int_0^{y_k} \frac{[\mu e^{H_{yc}y} 1_{ai}] g_{ab}(ij) [1_{bj}' e^{H_{yc}(y_k-t)} H_{cd} \mathbf{1}]}{\mu e^{H_{yc}y} H_{cd} \mathbf{1}} dt \quad (4.6)
$$

For the case where $a \in X_1, b \in X_2$, then $M^{ab}_{ij}(y_k)$ is estimated by

$$
\hat{M}^{ab}_{ij}(y_k) = \frac{[\mu e^{H_{yc}y} 1_{ai}] [1_{ai}' h_{ij}]}{\mu e^{H_{yc}y} H_{cd} \mathbf{1}} \quad (4.7)
$$

In both equations (4.6) and (4.7), $1_{ai}$ is a column vector of dimensions $r d_1 + d_2 \times 1$ with a 1 in the $a + i$ position and 0’s everywhere else and $1_{ai}'$ is a row vector of dimensions...
1 \times rd_1 with a 1 in the b + i position and 0’s everywhere else.

Proofs for equations (4.4), (4.5), (4.6) and (4.7) may be found in [12]'s Propositions IV.1, IV.2, IV.3a and IV.3b respectively.

Once initial estimates have been generated, the EM algorithm runs per Algorithm 1 in [12]. A new estimate of the initial state \( \nu \) is generated using (4.4) for all source/intermediate nodes in the network. The total sojourn time at each node \( a \in X_1 \) is estimated for all \( i \). This estimation is performed using all values of \( Y_k \in Y \). Similarly, the number of jumps from state \( (a, i) \) to state \( (b, j) \) are estimated using (4.6) \( (a, b \in X_1) \) and (4.7) \( (a \in X_1, b \in X_2) \). A new estimate of the generator \( G \) is then calculated

\[
\hat{g}_{abij} = \frac{\sum_{k=1}^{K} \hat{M}_{abij}(y_k)}{\sum_{k=1}^{K} \hat{D}_{ai}(y_k)}, \quad (b, j) \neq (a, i),
\]

and the diagonal elements of \( G \) are calculated in accordance with equation (2.3).

With new estimates of \( \nu \) and \( G \), a new iteration of these calculations are performed with the same set of observed data \( Y \). In [12], 1000 iterations are suggested for the algorithm to converge appropriately.

### 4.2.3 Calculation of Equations Containing Integrals with Exponentials

Van Loan’s Theorem 1 in [21] is used to solve the integrals in equations (4.5) and (4.6). Define

\[
C = \begin{bmatrix}
H_{cc} & H_{cd}1_{\mu} \\
0 & H_{cc}
\end{bmatrix}.
\]

(4.9)

The exponential \( e^{cy_k} \) is then evaluated using Theorem 1 from [21]:

\[
e^{cy_k} = \begin{bmatrix}
e^{H_{cc}y_k} \int_0^{y_k} [e^{H_{cc}(y_k-t)}H_{cd}1][1_{\mu}e^{H_{cc}t}]dt \\
0 & e^{H_{cc}y_k}
\end{bmatrix}.
\]

(4.10)
The matrix exponential can be calculated using Padé approximation \cite{22}.

If we set

\[ J(y) = \int_0^y [e^{H_{cc}(y_k-t)}H_{cd}1] [\mu e^{H_{cd}}] dt, \]  

(4.11)

the terms in equations (4.5) and (4.6) may be rearranged by substituting \( J(y) \) which yields

\[ \hat{D}_{a_i}(y_k) = \frac{1^i_{ai} J(y) 1_{ai}}{\mu e^{H_{cc}y_k} H_{cd}1} \]  

(4.12)

and

\[ \hat{M}_{ij}(y_k) = \frac{g_{ij} 1_{ij} J(y) 1_{ai}}{\mu e^{H_{cc}y_k} H_{cd}1}. \]  

(4.13)

### 4.3 Other Estimates

Several other estimates that use the infinitesimal generator are provided in \cite{12}. The density of the overall source-destination path delay is estimated using

\[ P_{\phi}(Y_k) = \mu e^{H_{cc}y} H_{cd}1 \]  

(4.14)

and the link delay density is computed using

\[ p_{\phi}(t|x_0, x_1) = \frac{\nu_{x_0} e^{G_{x_0}x_0}G_{x_0}x_1 1}{\nu_{x_0} D_{x_0}x_1 1} \]  

(4.15)

where \( D_{x_0}x_1 = -G^{-1}_{x_0}G_{x_0}x_1 \) for \( x_0 \neq x_1 \). The probability that a packet will travel from state node \( x_0 \) to node \( x_1 \) is

\[ p_{\phi}(x_1|x_0) = \frac{\nu_{x_0} D_{x_0}x_1 1}{\nu_{x_0} 1}. \]  

(4.16)
The current iteration’s likelihood of the observation sequence is calculated by

\[
P_{\phi_i}(Y_1, \ldots, Y_K) = \prod_{k=1}^{K} \mu e^{H_{cc}Y} H_{cd} 1. \tag{4.17}
\]

However, it is computationally convenient to calculate the log-likelihood

\[
\log P_{\phi_i}(Y_1, \ldots, Y_K) = \sum_{k=1}^{K} \log(\mu e^{H_{cc}Y} H_{cd} 1) \tag{4.18}
\]

instead \[3\]. The relative log-likelihood is calculated by

\[
\frac{|L[k] - L[k - 1]|}{L[k]} \tag{4.19}
\]

where \(L[k]\) is the log-likelihood calculated for the \(k\)th iteration using equation \(4.18\). The EM Algorithm is run until the relative log-likelihood falls under some threshold \[10\].
Chapter 5: Performance Evaluation

5.1 Simulation Network

A simple test network was designed to test the EM Algorithm described in chapter 4. This network consists of seven source/intermediate nodes \(d_1 = 7\) and two destination nodes \(d_2 = 2\). The topology of the network can be seen in figure 5.1. Packets entering the network were uniformly distributed between nodes 0 and 1. Packets that entered nodes 7 and 8 immediately left the network. Table 5.1 provides the nodes that make up each link and the probability that the packet will travel from the starting node to the ending node.

![Figure 5.1: Topology of Test Network](image-url)
Table 5.1: Link Definitions and Transition Probabilities

<table>
<thead>
<tr>
<th>Link</th>
<th>Starting Node</th>
<th>Ending Node</th>
<th>Transition Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>5</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>6</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>5</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>6</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>5</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>6</td>
<td>( \frac{1}{2} )</td>
</tr>
</tbody>
</table>

5.2 Evaluation

Three simulations were run using the queuing network simulator presented in chapter 3. The behavior of the nodes in the network, the packet arrival rate to the network and the packet length were varied for each simulation. Each network simulation consisted of 10,000 packets being introduced to the network. As each packet left the network, the packet’s overall sojourn time was recorded. Every 5th packet of the 10,000 packets was chosen to create a sampled set of 2,000 source/destination delays. These subsequent samples were used as the input to the EM Algorithm. The EM Algorithm was run for 1,000 iterations.

The EM Algorithm was run on each set of network simulation samples. The elements of \( H_{cc} \) and \( H_{cd} \) were uniformly distributed across the interval \([0, 1]\). The diagonals of \( H_{cc} \), and the elements of \( H_{dc} \) and \( H_{dd} \) were initialized as described in chapter 4. Each network was modeled as both a structured network and an unstructured network in the algorithm. The form of the generator matrix for the unstructured network is
\[ G = \begin{bmatrix}
G_{00} & G_{01} & G_{02} & G_{03} & G_{04} & G_{05} & G_{06} & h_{07} & h_{08} \\
G_{10} & G_{11} & G_{12} & G_{13} & G_{14} & G_{15} & G_{16} & h_{17} & h_{18} \\
G_{20} & G_{21} & G_{22} & G_{23} & G_{24} & G_{25} & G_{26} & h_{27} & h_{28} \\
G_{30} & G_{31} & G_{32} & G_{33} & G_{34} & G_{35} & G_{36} & h_{37} & h_{38} \\
G_{40} & G_{41} & G_{42} & G_{43} & G_{44} & G_{45} & G_{46} & h_{47} & h_{48} \\
G_{50} & G_{51} & G_{52} & G_{53} & G_{54} & G_{55} & G_{56} & h_{57} & h_{58} \\
G_{60} & G_{61} & G_{62} & G_{63} & G_{64} & G_{65} & G_{66} & h_{67} & h_{68} \\
u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & -\epsilon & 0 \\
u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & 0 & -\epsilon
\end{bmatrix} \tag{5.1}
\]

where \( G_{ab} \) is an \( r \times r \) matrix (\( a, b \) are source/intermediate nodes), \( h_{ab} \) are \( r \times 1 \) column vectors (\( a \) is a source/intermediate node and \( b \) is a destination node), \( u_a \) is a \( 1 \times r \) row vector and \( \epsilon \) is a scalar. Similarly, the generator for the structured network is

\[ G = \begin{bmatrix}
G_{00} & 0 & G_{02} & G_{03} & G_{04} & 0 & 0 & 0 & 0 \\
0 & G_{11} & G_{12} & G_{13} & G_{14} & 0 & 0 & 0 & 0 \\
0 & 0 & G_{22} & 0 & 0 & G_{25} & G_{26} & h_{27} & 0 \\
0 & 0 & 0 & G_{33} & 0 & G_{35} & G_{36} & h_{37} & 0 \\
0 & 0 & 0 & 0 & G_{44} & G_{45} & G_{46} & h_{47} & 0 \\
0 & 0 & 0 & 0 & 0 & G_{55} & 0 & h_{57} & 0 \\
0 & 0 & 0 & 0 & 0 & G_{66} & 0 & 0 & h_{68} \\
u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & -\epsilon & 0 \\
u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & 0 & -\epsilon
\end{bmatrix} \tag{5.2}
\]

5.2.1 Open Jackson Network

The first simulation was performed using an Open Jackson Network with constant packet length. Packets were introduced to the network at the rate of 1 packet/second. Each node in
the network is an exponential server with rate given in Table 5.2. The network was initially modeled as an unstructured network for $r = 1, 2, 3, 4$. The estimated source-destination delay density plot along with a normalized histogram of all 10,000 sojourn times collected during the simulation can be seen in Figure 5.2.

Table 5.2: Jackson Network Exponential Service Rates for Source/Intermediate Nodes

<table>
<thead>
<tr>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
<th>Node 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 5.2: Jackson Network Source Destination Delay Results

The EM Algorithm was also run with the network modeled as a structured network for $r = 2, 3, 4$. The EM Algorithm was not run for $r = 1$ because the algorithm failed to run due to a bug in the Python scientific library SciPy [1]. The performance of the structured network matched that of the unstructured network when estimating the source-destination delay density. Appendix A contains the estimated source-destination delay density plots.
and estimated link delay density plots for all links and all simulations.

In general, the structured network link delay estimations performed better than the unstructured link delay estimations. Figures 5.3 and 5.4 show the estimated link delay density for link number 6 along with the normalized histogram of the true link delay. The relative log-likelihood of the EM Algorithm converged monotonically for both models.

![Figure 5.3: Jackson Network Link 6 Delay Density Unstructured](image1)

![Figure 5.4: Jackson Network Link 6 Delay Density Structured](image2)

Structured network modeling yields a considerable improvement in the packet routing estimation over unstructured modeling. Figure 5.5 and figure 5.6 show the mean squared error of estimated packet routing probabilities as compared to the true routing probabilities in table 5.1.

5.2.2 MMPP/D/1 With Uniform Packet Length

The second simulation was performed with an MMPP/D/1 network with uniform packet length. Packets were introduced to the network using a Markov Modulated Poisson Process. This process introduces 3 packets/second for an exponentially distributed amount of time (0.667 seconds on average) and 1 packet/second for a different exponentially distributed amount of time (0.333 seconds on average). Each node in the network was capable of processing 1 packet at the deterministic rate given in table 5.3. The network was modeled
in the same manner as the Jackson Network described previously and source/destination samples were collected in the same manner. The estimated source-destination delay density plot along with a normalized histogram of all 10,000 sojourn times collected during the simulation can be seen in figure 5.7

Table 5.3: MMPP/D/1 Network Deterministic Service Rates for Source/Intermediate Nodes

<table>
<thead>
<tr>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
<th>Node 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Unfortunately, due to the deterministic nature of the nodes in the simulation, the sojourn times stack up considerably in all histogram plots. However, the EM Algorithm does a reasonable job of estimating the source/destination delay density using both structured and unstructured network modeling.

In general, the structured network link delay estimations performed better than the unstructured link delay estimations. Figures 5.8 and 5.9 show the estimated link delay density for link number 6 along with the normalized histogram of the true link delay. It
should also be noted that link delay densities for links 8 and 9 were unable to be calculated due to numerical error. The error appears to be related to values being computed that are either too big or too small for the exponential matrix Python function to handle.

Figure 5.8: MMPP/D/1 Link 6 Delay Density Unstructured

Figure 5.9: MMPP/D/1 Link 6 Delay Density Structured
Structured network modeling yields a minor improvement in the packet routing estimation over unstructured modeling. Figure 5.10 and figure 5.11 show the mean squared error of estimated packet routing probabilities as compared to the true routing probabilities in table 5.1. However, the mean squared error for MMPP/D/1 networks is higher than the mean squared error in the Open Jackson Network. The EM Algorithm did not converge for this case as it did for the Open Jackson Network. The calculated relative log-likelihood of the EM Algorithm was non-monotonic as a function of the number of iterations, a behavior which requires further investigation.

5.2.3 MMPP/D/1 With Exponential Packet Length

The third simulation was performed with an MMPP/D/1 network with an exponential packet length. These parameters were chosen in an effort to mimic the behavior of real world networks where servers will process data at constant rates but packet lengths may vary. Packets were introduced to the network at the same rate used in section 5.2.2. Each packet introduced to the network was given a size that was exponentially distributed with an average of 2. Each node in the network was capable of processing 1 unit of the packet at the deterministic rate given in table 5.3. Samples were collected in the manner previously
Figure 5.12: MMPP/D/1 Source Destination Delay Results with Exponential Packet Length described. The estimated source-destination delay density plot along with a normalized histogram of all 10,000 sojourn times collected during the simulation can be seen in figure 5.12.

The performance of the structured network matched that of the unstructured network when estimating the source/destination delay density. Figures 5.13 and 5.14 show the estimated link delay density for link number 6 along with the normalized histogram of the true link delay. The link delay estimation for the MMPP/D/1 network with exponential packet lengths for structured networks was as good as the estimations for the Open Jackson Network.

Again, structured network modeling provides better packet routing estimation than unstructured modeling. Figure 5.15 and figure 5.16 show the mean squared error of estimated packet routing probabilities in this simulation as compared to the true routing probabilities in table 5.1. The performance of the unstructured network model is comparable to the Open Jackson Network unstructured estimates. The Open Jackson Network structured estimates are slightly better than the MMPP/D/1 with exponential packet length estimate.
The relative log-likelihood of the structured model converged monotonically, however, the unstructured model did not converge monotonically. This behavior requires more investigation.
Chapter 6: Comments

The EM Algorithm outlined in [12] was implemented and tested using three sets of simulated network data generated from a custom queuing network simulator. The internal network characteristics were varied in order to test the algorithm’s performance given various network types. The EM Algorithm performed very well when estimating the source/destination delays through the entire network, regardless of how the network was modeled.

Both link delay estimation and packet routing probability estimation performed better when the network’s topology and flow of data were known (in other words, when the network was modeled as a structured network). In some cases, however, the unstructured model still provided reasonable estimations of the link delays. The EM Algorithm also performed better when estimating networks with more exponential characteristics to them.

There is room for additional evaluation of the EM Algorithm. The EM Algorithm can be enhanced to take into account the source and destination nodes along with the total sojourn time. This may lead to greater accuracy when estimating link delay density and packet routing estimation. This algorithm has only been tested using theoretical results in [12] and using simulated queuing network data in this thesis. Additional testing with real world data could yield interesting results.
Appendix A: Additional Images

Additional plots are available in this appendix for comparison purposes.

A.1 Jackson Network Plots

Figure A.1: Jackson Network Estimated Source/Destination Delay Density - Unstructured

Figure A.2: Jackson Network Estimated Source/Destination Delay Density - Structured

Figure A.3: Jackson Network Estimated Link Delay Density for Link 0 - Unstructured

Figure A.4: Jackson Network Estimated Link Delay Density for Link 0 - Structured
Figure A.5: Jackson Network Estimated Link Delay Density for Link 1 - Unstructured

Figure A.6: Jackson Network Estimated Link Delay Density for Link 1 - Structured

Figure A.7: Jackson Network Estimated Link Delay Density for Link 2 - Unstructured

Figure A.8: Jackson Network Estimated Link Delay Density for Link 2 - Structured
Figure A.9: Jackson Network Estimated Link Delay Density for Link 3 - Unstructured

Figure A.10: Jackson Network Estimated Link Delay Density for Link 3 - Structured

Figure A.11: Jackson Network Estimated Link Delay Density for Link 4 - Unstructured

Figure A.12: Jackson Network Estimated Link Delay Density for Link 4 - Structured

36
Figure A.13: Jackson Network Estimated Link Delay Density for Link 5 - Unstructured

Figure A.14: Jackson Network Estimated Link Delay Density for Link 5 - Structured

Figure A.15: Jackson Network Estimated Link Delay Density for Link 6 - Unstructured

Figure A.16: Jackson Network Estimated Link Delay Density for Link 6 - Structured
Figure A.17: Jackson Network Estimated Link Delay Density for Link 7 - Unstructured

Figure A.18: Jackson Network Estimated Link Delay Density for Link 7 - Structured

Figure A.19: Jackson Network Estimated Link Delay Density for Link 8 - Unstructured

Figure A.20: Jackson Network Estimated Link Delay Density for Link 8 - Structured
Figure A.21: Jackson Network Estimated Link Delay Density for Link 9 - Unstructured

Figure A.22: Jackson Network Estimated Link Delay Density for Link 9 - Structured

Figure A.23: Jackson Network Estimated Link Delay Density for Link 10 - Unstructured

Figure A.24: Jackson Network Estimated Link Delay Density for Link 10 - Structured
Figure A.25: Jackson Network Estimated Link Delay Density for Link 11 - Unstructured

Figure A.26: Jackson Network Estimated Link Delay Density for Link 11 - Structured

A.2 MMPP/D/1 with Static Packet Size Plots

Figure A.27: MMPP/D/1 Estimated Source/Destination Delay Density - Unstructured

Figure A.28: MMPP/D/1 Estimated Source/Destination Delay Density - Structured
Figure A.29: MMPP/D/1 Estimated Link Delay Density for Link 0 - Unstructured

Figure A.30: MMPP/D/1 Estimated Link Delay Density for Link 0 - Structured

Figure A.31: MMPP/D/1 Estimated Link Delay Density for Link 1 - Unstructured

Figure A.32: MMPP/D/1 Estimated Link Delay Density for Link 1 - Structured
Figure A.33: MMPP/D/1 Estimated Link Delay Density for Link 2 - Unstructured

Figure A.34: MMPP/D/1 Estimated Link Delay Density for Link 2 - Structured

Figure A.35: MMPP/D/1 Estimated Link Delay Density for Link 3 - Unstructured

Figure A.36: MMPP/D/1 Estimated Link Delay Density for Link 3 - Structured
Figure A.37: MMPP/D/1 Estimated Link Delay Density for Link 4 - Unstructured

Figure A.38: MMPP/D/1 Estimated Link Delay Density for Link 4 - Structured

Figure A.39: MMPP/D/1 Estimated Link Delay Density for Link 5 - Unstructured

Figure A.40: MMPP/D/1 Estimated Link Delay Density for Link 5 - Structured
Figure A.41: MMPP/D/1 Estimated Link Delay Density for Link 6 - Unstructured

Figure A.42: MMPP/D/1 Estimated Link Delay Density for Link 6 - Structured

Figure A.43: MMPP/D/1 Estimated Link Delay Density for Link 7 - Unstructured

Figure A.44: MMPP/D/1 Estimated Link Delay Density for Link 7 - Structured
Figure A.45: MMPP/D/1 Estimated Link Delay Density for Link 8 - Structured

Figure A.46: MMPP/D/1 Estimated Link Delay Density for Link 9 - Structured
Figure A.47: MMPP/D/1 Estimated Link Delay Density for Link 10 - Unstructured

Figure A.48: MMPP/D/1 Estimated Link Delay Density for Link 10 - Structured

Figure A.49: MMPP/D/1 Estimated Link Delay Density for Link 11 - Unstructured

Figure A.50: MMPP/D/1 Estimated Link Delay Density for Link 11 - Structured
A.3 MMPP/D/1 with Exponential Packet Size Plots

Figure A.51: MMPP/D/1 Estimated Source/Destination Delay Density - Unstructured

Figure A.52: MMPP/D/1 Estimated Source/Destination Delay Density - Structured

Figure A.53: MMPP/D/1 Estimated Link Delay Density for Link 0 - Unstructured

Figure A.54: MMPP/D/1 Estimated Link Delay Density for Link 0 - Structured
Figure A.55: MMPP/D/1 Estimated Link Delay Density for Link 1 - Unstructured

Figure A.56: MMPP/D/1 Estimated Link Delay Density for Link 1 - Structured

Figure A.57: MMPP/D/1 Estimated Link Delay Density for Link 2 - Unstructured

Figure A.58: MMPP/D/1 Estimated Link Delay Density for Link 2 - Structured
Figure A.59: MMPP/D/1 Estimated Link Delay Density for Link 3 - Unstructured

Figure A.60: MMPP/D/1 Estimated Link Delay Density for Link 3 - Structured

Figure A.61: MMPP/D/1 Estimated Link Delay Density for Link 4 - Unstructured

Figure A.62: MMPP/D/1 Estimated Link Delay Density for Link 4 - Structured
Figure A.63: MMPP/D/1 Estimated Link Delay Density for Link 5 - Unstructured

Figure A.64: MMPP/D/1 Estimated Link Delay Density for Link 5 - Structured

Figure A.65: MMPP/D/1 Estimated Link Delay Density for Link 6 - Unstructured

Figure A.66: MMPP/D/1 Estimated Link Delay Density for Link 6 - Structured
Figure A.67: MMPP/D/1 Estimated Link Delay Density for Link 7 - Unstructured

Figure A.68: MMPP/D/1 Estimated Link Delay Density for Link 7 - Structured

Figure A.69: MMPP/D/1 Estimated Link Delay Density for Link 8 - Unstructured

Figure A.70: MMPP/D/1 Estimated Link Delay Density for Link 8 - Structured
Figure A.71: MMPP/D/1 Estimated Link Delay Density for Link 9 - Unstructured

Figure A.72: MMPP/D/1 Estimated Link Delay Density for Link 9 - Structured

Figure A.73: MMPP/D/1 Estimated Link Delay Density for Link 10 - Unstructured

Figure A.74: MMPP/D/1 Estimated Link Delay Density for Link 10 - Structured
Figure A.75: MMPP/D/1 Estimated Link Delay Density for Link 11 - Unstructured

Figure A.76: MMPP/D/1 Estimated Link Delay Density for Link 11 - Structured
Bibliography


Biography

David Stoner grew up in Burke, VA. He attended the Pennsylvania State University, where he received Bachelor of Science degrees in Electrical Engineering and Mathematics in 2006. He currently works as a Systems Engineer with the Boeing Company.