Characterization and Performance Analysis
of a Cognitive Routing Scheme
for a Metropolitan-Area Sensor Network

by Esther Jang


Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology

May 2016

Copyright 2016 Esther Jang. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and to distribute publicly paper
and electronic copies of this thesis document in whole and in part in any medium now known
or hereafter created.

Signature redacted

Certified by:
Professor Vincent Chan, Thesis Supervisor
May 20, 2016

Accepted by:
Dr. Christopher Terman, Chairman, Masters of Engineering Thesis Committee
ABSTRACT
This MEng thesis is an exploration of the notion of cognitive methods for routing in a network, and the resulting potential for improvements in network performance. In cognitive routing, individual network nodes gain information about the state of the network in a distributed fashion, by measuring observable data such as packet arrival counts and timing. The nodes then use inference and estimation methods on the network traffic to modify the parameters of their routing protocols and/or routing tables, in order to improve some performance metric such as packet delay or network throughput. In this project we provide an example of the performance improvements possible through cognitive routing, by demonstrating a simple but nontrivial use of network measurement and inference to minimize the maximum average packet delay, and increase the max load that the network can handle. With more information-rich metrics that are available to be passively gathered by a routing protocol, such as source-destination IDs, the sizes of packets passing through a node, and packet loss rates, cognitive routing protocols may be able to predict congestion or link failures, potentially leading to much greater efficiency gains than are described in this project.
## Contents

1 Introduction .......................................................... 6
   1.1 Cognitive Routing .............................................. 6
   1.2 Related Work .................................................. 6
       1.2.1 Cognitive Radio ......................................... 6
       1.2.2 Reinforcement Learning-Based Methods ................. 7
       1.2.3 Traffic Prediction ....................................... 7

2 Network Model ....................................................... 9
   2.1 Graph Structure ............................................... 9
   2.2 Traffic Model ................................................ 10
   2.3 Traffic Re-routing Algorithm ................................ 11

3 Delay ........................................................................ 13
   3.1 M/D/1 Queueing Delay .......................................... 13
   3.2 Delay with routing parameter $f$ .............................. 14
       3.2.1 Optimizing Delay Over $f$ ............................... 15
       3.2.2 Analytical Bounds For Delay ............................ 20

4 Detection ............................................................... 21
   4.1 Bayesian Inference .............................................. 21
       4.1.1 Likelihood Ratio Test .................................. 21
       4.1.2 Neyman-Pearson Test .................................. 22
   4.2 Receiver Operating Characteristic ............................ 23
       4.2.1 Error Probabilities .................................... 23
       4.2.2 ROC curves ............................................. 24
5 Performance

5.1 Discrete-time Approximation for Average Packet Delay ........................................... 26
  5.1.1 Average Delay Time Series ..................................................................................... 27

5.2 Stopping Time ............................................................................................................... 29
  5.2.1 Expected Stopping Time ......................................................................................... 29

5.3 Optimizing Delay over Detection Time T ....................................................................... 30
  5.3.1 \( \rho = 1.1 \) ........................................................................................................... 34
  5.3.2 \( \rho = 0.5 \) ........................................................................................................... 38

5.4 Further Improvements ................................................................................................... 38

6 Conclusion ...................................................................................................................... 41

7 Appendix

7.1 Derivation for Bounds on Delay ................................................................................... 42
  7.1.1 Upper Bound .......................................................................................................... 42
  7.1.2 Lower Bound ......................................................................................................... 43

7.2 MM1 Delay .................................................................................................................... 44
  7.2.1 Bounds .................................................................................................................. 44
  7.2.2 Conditions for needing the algorithm .................................................................... 44

8 Citations ........................................................................................................................... 47
Chapter 1

Introduction

1.1 Cognitive Routing

Cognitive routing is a routing paradigm in which individual network nodes may use distributed network measurement and estimation in order to learn about the state of the network, and modify their routing protocols accordingly to improve performance. With the advent of new, more efficient methods in machine learning, it may become possible to use data analysis to reduce the throughput required by tasks such as link state upkeep.

1.2 Related Work

1.2.1 Cognitive Radio

There has been extensive work on Cognitive Radio in MAC protocols, to predict whether or not a channel (defined by a carrier frequency band) will be busy in the next time step. At least one scheme has been designed and tested in which nodes use hidden Markov models (HMMs) to predict channel state time series online by observing historical data-transmission (Chen, 2011). These predictions may be incorporated into the MAC layer during a contention phase if there is one, to estimate transmission length and increase the probability of successful channel sharing. While these improvements can be implemented without disturbing the upper layers of the protocol stack, the data gathered to be used for the MAC layer could also be incorporated into routing functions.
1.2.2 Reinforcement Learning-Based Methods

Most cognitive routing schemes that have been studied thus far are fairly complex, using artificial neural networks or reinforcement learning based on the classic Q-learning method described by Boyan (Boyan, 1994). The original method used reported delay from acknowledged packets as a learning metric; since then, other routing algorithms have used source and destination IDs, estimated packet loss rates and link quality, and hop count in addition to delay, and achieved some efficiency gains in terms of reducing delay (Gelenbe, 2011). A relevant design feature here is whether the protocol is source routing or route discovering. The former refers to schemes in which the source node specifies the complete path to the destination (e.g. Dynamic Source Routing, DSR), perhaps after a period of sending request packets to determine a path, and the latter to when packets are forwarded directly from router to router based on their best information about the lowest cost path to the destination.

The Cognitive Packet Network (CPN) and Self-Selective Routing (SSR) protocols (Gelenbe et al.) are two examples of cognitive routing protocols that have been implemented in research contexts. Both use neural networks at their nodes and reinforcement learning to train their protocols to find the best routing paths. However, the former uses request packets in order to do route discovery, storing the route information for a particular data flow at the intermediate nodes, using delay and hop count of the eventual route to train the nodes’ internal neural networks. The latter has nodes that assign themselves backoff delays based on their expected distance from the destination node, using source and destination IDs, sequence number of the last packet received form the source, hop count, expected number of hops left, and TTL to compute the backoff delays.

Most real-world implementations of cognitive routing in the near future are likely to be much simpler.

1.2.3 Traffic Prediction

Many different methods have been proposed for using historical time series data to predict trends in network traffic. A protocol could learn periodicities in traffic flow to and from specific areas of the network, and prepare for times of heavy load by allocating more paths, or alerting the network designers to strategically relocate nodes. For example, in the case of an on-demand protocol, a sending node could demand more routes than normally necessary, or routes avoiding specific regions expected to receive heavy traffic. Several papers have attempted to deconstruct real network flows into eigenflows using eigendecomposition methods such as PCA, with the goal of predicting features of traffic patterns and load that can vary predictably over a given period of time such as a week (Azad, 2007).

The values of protocol-specific parameters may also be used as data for traffic prediction, especially
parameters that can be specified by the application layer. For example, the Constrained Application Protocol (CoAP) has a reliable or unreliable transport bit that specifies whether the sender will be wanting acknowledgements for transmitted packets or not (Shelby, et. al., 2014). Other protocols may specify whether communications should be connection-based or connectionless, as well as the source and destination IDs and port numbers being used (which may specify how many other processes are communicating from each node). Sequence numbers may even be used to track how long the sending node has been up, and from there how long they have yet to live. Each of these pieces of information may be provided as input to a traffic model at each node.

The traffic model studied in this thesis is much simpler than any of these implementations, using only packet counts received at each node, a single header field of link-state update, and not considering packet acknowledgements or other overhead. Its simplicity allows us to characterize our cognitive routing algorithm using standard mathematical analysis techniques.
2.1 Graph Structure

The network model represents a simple but reasonable real-world architecture for a metropolitan-area mesh sensor network. All sensor nodes in the network have a wire connection to a central hub node, which collects data from all sensors, and a wire connection to two neighbors to the left and right of it. The resulting overall layout is a ring structure, which can be extended outwards to form a gridded mesh folded into concentric circles, as depicted in the diagram below.

![Diagram of network model](image)

Figure 2.1

However, to simplify and clarify the computations, we will focus on the case of a single-layer ring around the hub. We approximate all of the link lengths to be the same, and each link to be modeled by a channel of the same capacity and error rate, so that there is a single deterministic transmission time for each packet. Each non-hub node has three radios that can send and receive simultaneously over each link; one to each
neighboring sensor node and one to the central hub. Furthermore, there is a separate memory buffer associated with each radio. The central hub, on the other hand, has a dedicated port from each sensor node and may receive simultaneously from all of its neighbors.

![Figure 2.2](image)

**Figure 2.2**

![Figure 2.3](image)

**Figure 2.3**

### 2.2 Traffic Model

Each sensor node generates exogenous packets destined for the central hub as a Poisson process with a baseline rate of $\lambda_0$ in packets per second. However, there is a deterministic transmission time for each packet; therefore, the queueing model used to compute the packet delay is M/D/1.

The phenomenon we aim to detect is that of a single node that suddenly begins to produce large amounts of network traffic, such as a sensor in an alarm system. We call this phenomenon a surge, assuming for ease of computation that only one node surges at a time. The time at which the surge occurs is defined as $t = 0$, at which time there is a step change in the rate parameter underlying traffic arrivals. Packets begin to arrive as a Poisson process with an increased rate of $\lambda_0 + \lambda_s$, where $\lambda_s$ is the expected increase in packet arrival rate over the baseline during a surge.

We are concerned with preventing the potentially large increase in delay for packets from surging nodes
due to congestion and possible buffer overflow on the shortest path link to the destination, in this case the downward link to the central node.

### 2.3 Traffic Re-routing Algorithm

![Figure 2.4](image)

The task of our cognitive routing algorithm is to route packets such that the average delay of surge packets on the network is minimized, given that the only sources of packet delay are queueing and transmission delay. In the baseline case where there is no surge and the network load fraction $p$ is low (significantly less than 1), the minimal-delay routing strategy is to send all packets along the shortest path (the direct downward link to the central node).

During a surge, our algorithm’s re-routing strategy is to split the traffic along multiple paths to the central hub node, which in the ring architecture means routing some fraction of incoming packets to the left and right side neighbors, who will forward the packets to the central node. We refer to the fraction of packets not sent along the shortest path link at a given node as $f$, which we can think of as a parameter of the algorithm that may be optimized. Furthermore, we assume that a given node is able to detect over which link the packets in its buffers have arrived. This means that a non-surging node whose incoming packet rate has increased due to a surge is able to determine the direction from which the surge is coming, and avoid re-routing packets along the link in that direction. As a result, the increase in network load due to the surge packets will be spread out through the network, decreasing geometrically from the source of the surge.

In the following chapters, we compute the average M/D/1 delay of surge packets, assuming that there is enough total capacity in the network to handle the surge, and also that there are enough nodes to diminish the surge geometrically to zero before the spreading traffic wraps around to the opposite side of the ring network.

We further assume that the surging node, in the process of re-routing, may explicitly notify its neighbors (via a small number of header bits) of the value of $f$ being used by its re-routing algorithm. So the surging node itself will send $f/2$ fraction of its total packets to either side. Meanwhile, all non-surging nodes that receive surge packets will send $f$ fraction of the surge traffic (not including its own baseline traffic) along the
lateral link in the direction away from the source. Therefore, we can compute the average delay of surge packets on the network with $f$ as a constant parameter. Note that for ease of computation we have assumed a uniform $f$ for all nodes, but it would also be possible for nodes to individually choose $f$ if they detected the surge on their own, which would result in a more purely distributed algorithm (that we do not analyze here).

We find that the average delay of surge packets is a convex function of the parameter $f$ while $\rho < 1$. We also consider the case when $\rho > 1$, because it may be reasonable to expect a traffic surge to be able to overwhelm any single link, but expect the re-routing algorithm to distribute the load such that the network as a whole can support it. We find that the average delay of surge packets as a function of $f$, if $\rho$ (while greater than 1) is within some reasonable limits, has a convex region that can be exploited to allow the underprovisioned network to handle the traffic increase. A combination of numerical and analytic methods are used to establish these results, presented in detail in Chapter 3.
Chapter 3

Delay

3.1 M/D/1 Queueing Delay

First we review the calculation for the delay of a single packet on a general network. As stated in the previous chapter, the only considered sources of delay are queueing and transmission. The total delay of a packet is therefore the sum over all traversed hops of the expected delay in each queue, plus one unit of transmission delay.

We use the Pollack-Khinchin equation for average wait time in a queue:

$$E[W] = \frac{\lambda E[V^2]}{2(1 - \lambda E[V])}$$ (3.1)

where $W$ is the wait time in the queue, $V$ is the service time for an individual packet, and $\lambda$ is the incoming packet rate to the queue. In our MD1 queueing model, our $V$ is deterministically equal to the transmission time, which we define as $t_p$ in units of seconds. Therefore, our value for $E[W]$ is given by

$$\frac{\lambda t_p^2}{2(1 - \lambda t_p)} = \frac{\rho t_p}{2(1 - \rho)}$$ (3.2)

where the load $\rho$ is equal to $\lambda t_p$ in our model.

The total expected delay of a packet in the network is therefore:

$$\sum_{\text{hops}} \left( \frac{\rho t_p}{2(1 - \rho)} + t_p \right) = \sum_{\text{hops}} \left( 1 - \frac{\rho}{2} \right) \left( \frac{t_p}{1 - \rho} \right)$$ (3.3)
3.2 Delay with routing parameter $f$

According to our re-routing algorithm as described in the previous chapter, during a traffic surge, the surging node will split both surge and baseline traffic to either side, while the other nodes will send $f$ fraction of whatever traffic they receive away from the source of the surge. For the purpose of computing the expected delay of surge packets treated in this manner, we model the packets as traversing an infinite line of nodes to one side of the surging node, with a geometrically decreasing probability of continuing on to the next node rather than being sent from the current node directly to the central hub.

The total expected per-packet delay will be the sum over all potential paths for a surge packet weighted by the fraction of surge packets that take that path. In the following equations, the sum is broken up into three components, representing the delay incurred along all of the lateral links, all of the downward links on non-shortest paths, and the downward link on the shortest path (directly from the surging node to the central node), respectively. Each term is weighted by the expected fraction of packets that will traverse each link.

\[
\text{Delay} = \sum_{j=1}^{\infty} \sum_{k=1}^{j} \frac{(1 - \frac{1}{2} \frac{1}{t_p} \lambda_s f^k) t_p (1 - f) f^j}{1 - \frac{1}{2} \frac{1}{t_p} \lambda_s f^k} + \sum_{j=1}^{\infty} \frac{(1 - \frac{1}{2} \frac{1}{t_p} (\lambda_0 + \frac{1}{2} (1 - f) f^j \lambda_s)) t_p (1 - f) f^j}{1 - t_p (\lambda_0 + \frac{1}{2} (1 - f) f^j \lambda_s)} \\
+ \frac{(1 - \frac{1}{2} \frac{1}{t_p} (\lambda_0 + (1 - f) \lambda_s)) t_p (1 - f)}{1 - t_p (\lambda_0 + (1 - f) \lambda_s)}
\] (3.4)

In an alternate form:

\[
\text{Delay} = \sum_{j=1}^{\infty} \sum_{k=1}^{j} \frac{(1 - \frac{1}{2} \frac{1}{t_p} (1 - f) f^j)}{\mu - \rho_1/t_p} + \sum_{j=1}^{\infty} \frac{(1 - \frac{1}{2} \frac{1}{t_p} (1 - f) f^j)}{\mu - \rho_2/t_p} + \frac{(1 - \frac{1}{2} \frac{1}{t_p} (1 - f))}{\mu - \rho_3/t_p}
\] (3.5)

where $\lambda_0$, $\lambda_s$, and $t_p$ are the incoming baseline traffic rate, surge traffic rate, and transmission time as defined in the previous chapter, $\mu$ is the transmission rate $\frac{1}{t_p}$, $\rho_1 = t_p \lambda_s f^k$, $\rho_2 = t_p (\lambda_0 + \frac{1}{2} (1 - f) f^j \lambda_s)$, and $\rho_3 = t_p (\lambda_0 + (1 - f) \lambda_s)$.

This expression for delay is plotted below as a function of $f$ up to the $j = 100$ set of terms. In most of the figures shown, including the one below, $\lambda_0$, which is a low packet rate representative of the data generation
at a wireless sensor node, and $\lambda_0$ are set such that $\rho = (\lambda_0 + \lambda_s)t_p$ is 0.9. Here, those values are: $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, and $\lambda_s = 108.571$ packets/s such that the surge load $\rho$ is 0.9. The value used for $t_p$ is the total per-packet processing time for the IEEE Zigbee 802.15.4 protocol, reported by Burchfield et. al. (2007).

The given parameter setting is convenient because while the steady-state delay is not infinite, as $\rho < 1$, the load is high enough so that the delay almost certainly has a minimum value as a function of $f$ where $f > 0$. (We will also frequently show plots of parameter settings where $\rho = 0.5$ or $1.1$ to highlight particular points about algorithm performance.) Conditions for this property corresponding to re-routing being useful for minimizing delay are derived in the Appendix for the MM1 queue version of the delay; we expect there to be similar bounds for MD1 queues, but do not derive them here.

![Figure 3.2: Average delay as a function of $f$, for parameter values $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_s = 108.571$ packets/s. Sum of the first $j = 1$ to 100 terms in blue, and the sum of the first $j = 1$ to 20 terms in orange.](image)

The finite summation for delay is convex excluding the peak near $f = 1$, which, as suggested by the figure of delay summed to $j = n$ terms for $n$ from 4 to 20, appears to increase linearly in magnitude with the number of terms. Linear regression on the peak values of delay plots with various parameter settings produced an $R^2$ value of 1. We conclude that the delay at $f = 1$ would approach infinity with an infinite number of terms in the summation, which is the expected result if every packet is passed to the side and never reaches its destination at the hub node.

### 3.2.1 Optimizing Delay Over $f$

Based on knowledge of the rate parameters and transmit time along with an expression for the average delay, we should be able to find the $f$ that will minimize delay.

Since we do not have a closed form expression for delay, we can only approximate it using a finite number
of terms, or develop some other analytic bounds. We obtain a fairly good lower bound simply by truncating the number of terms to 10 or even 4, because the average delay function for $p < 1$ is wide and shallow, as depicted in the figure below. We find that using on the order of 10 terms for small $p$ and then using standard calculus to find the zero of the derivative will achieve a value of $f$ corresponding to a near-optimal average delay. For our parameter setting, we achieve an optimal average delay within a hundreth of the transmission time $t_p$. 
Figure 3.3: Sum of the first $j = 1$ to $n$ terms, where $n$ varies from 4 (the lowest curve) to 20 (the highest curve). Curves below $n = 4$ were not plotted due to having no minimum value. With infinite terms, the peak at $f = 1$ would approach infinity, as expected. $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_s = 108.571$ packets/s.

Figure 3.4: $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 1.1$, and $\lambda_s = 137.143$ packets/s. Sum of the first $j = 1$ to $n$ terms, where $n$ varies from 4 (the lowest curve) to 20 (the highest curve). Curves below $n = 4$ were not plotted due to having no minimum. Note that since $\rho > 1$, the steady state average delay is infinite unless $f$ is greater than some threshold value, below which the downstream transmit buffer of the surging node overflows.
Figure 3.5: $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_a = 108.571$ packets/s. Derivative of delay, sum of the first $j = 1$ to $n$ terms where $n$ varies from 4 (the lowest curve) to 20 (the highest curve). The optimal $f_s$ (at the leftmost zero-crossing of each curve) are clustered tightly even for small $n$.

Figure 3.6: $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 1.1$, and $\lambda_a = 137.143$ packets/s. Derivative of delay, sum of the first $j = 1$ to $n$ terms where $n$ varies from 4 (the lowest curve) to 20 (the highest curve).
Figure 3.7: $t_p = 0.007$s, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_s = 108.571$ packets/s. The delay is shown as a dashed black line, and the orange dots on the line are the average delay resulting from calculating the optimal $f$ using the specified number of terms. The blue dots use the optimal $f$ computed from the analytical upper and lower bounds derived below. The delays resulting from $f$ computed with $n = 4$ to 20 and $n = 100$ are plotted, but $n = 100$ is indistinguishable from $n = 10$ and above. The red dots also show the value of the average delay computed using the given number of terms.
3.2.2 Analytical Bounds For Delay

We also derive analytical upper and lower bounds for delay based on the summation expression, which appear to be tight. We do not require these bounds further as the truncated summation is the best approximation in our region of interest, but the results are provided below, and derivations are provided in the Appendix.

![Graph showing average delay]

Figure 3.8: Average delay: the sum of the first $j = 1$ to 100 terms is shown in blue, the upper bound is in orange, and the lower bound is in green. $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_s = 108.571$ packets/s.

$$\text{Delay} < \frac{(1 - \rho_1') t_p (1 - f)}{1 - \rho_1'} + \frac{(1 - \rho_2') t_p (1 - f)}{1 - \rho_2'} \frac{f}{1 - f} + \frac{(1 - \rho_3') t_p (1 - f)}{1 - \rho_3'}$$  \hspace{1cm} (3.6)

The full upper bound is:

$$\text{Delay} < \frac{(1 - \rho_1') t_p (1 - f)}{1 - \rho_1'} + \frac{(1 - \rho_2') t_p (1 - f)}{1 - \rho_2'} \frac{f}{1 - f} + \frac{(1 - \rho_3') t_p (1 - f)}{1 - \rho_3'}$$  \hspace{1cm} (3.7)

where $\rho_1 = \frac{1}{2} t_p \lambda_s f$, $\rho_2 = t_p (\lambda_0 + \frac{1}{2} (1 - f) f \lambda_s)$, and $\rho_3 = t_p (\lambda_0 + (1 - f) \lambda_s)$.

The full lower bound is:

$$\text{Delay} > \frac{t_p f}{1 - f} + \frac{(1 - \rho_2') t_p (1 - f)}{1 - \rho_2'} \frac{f}{1 - f} + \frac{(1 - \rho_3') t_p (1 - f)}{1 - \rho_3'}$$  \hspace{1cm} (3.8)

where $\rho_2' = t_p \lambda_0$ and $\rho_3' = t_p (\lambda_0 + (1 - f) \lambda_s)$. 

20
Chapter 4

Detection

4.1 Bayesian Inference

In order to detect the surge and trigger the traffic re-routing algorithm, we use Bayesian inference to classify the state of each node as surging or not surging. The Bayesian model requires a prior distribution over the surge and non-surge states, which we parametrize with $p$, the probability of being in the non-surge state. Furthermore, as described in Chapter 2, we assume that surges occur as a step change in the Poisson rate parameter underlying traffic arrivals, so there are only two traffic rates possible in this model, and no transition states.

We may use a number of different traffic rate estimators at each node to make a decision about whether or not there is a surge at that node. The estimator we focus on here is the Bayesian likelihood ratio test based on the number of packet arrivals in a fixed time interval $T$. This estimator, while among the simplest to analyze, is likely to be optimal given the step change in packet rate. However, the proof is not within the scope of this thesis.

Note that in our framework, due to the explicit notification of $f$ sent by a surging node to its neighbors upon detecting its own surge, there is no need for a non-surging node to detect the surges coming over its own left or right side links. However, in the case of no explicit congestion notification, the same estimation framework may be used by each individual node to decide whether to re-route traffic and in which direction.

4.1.1 Likelihood Ratio Test

Assume $K$ is a random variable denoting the number of packet arrivals in the last $T$ seconds, $p$ is the probability of the non-surge state, $\lambda_1$ is equal to the non-surge rate $\lambda_0$, and $\lambda_2$ is equal to the surge rate
The likelihood ratio test is defined as follows:

\[
\frac{\Pr[K = k|\lambda_1]}{\Pr[K = k|\lambda_2]} \overset{\alpha}{=} \frac{1 - p}{p}
\] (4.1)

In the case of Poisson arrival processes for both cases,

\[
\frac{(\lambda_1 T)^k e^{-\lambda_1 T}}{k!} \overset{\alpha}{=} \frac{1 - p}{\lambda_1}
\] (4.2)

This simplifies to

\[
\frac{(\lambda_1 T)^k e^{-\lambda_1 T}}{k!} \overset{\alpha}{=} \frac{1 - p}{\lambda_1}
\] (4.3)

Assuming that \( \lambda_2 > \lambda_1 \),

\[
\frac{e^{(\lambda_2 - \lambda_1)T}}{\lambda_1 \lambda_2} \overset{\alpha}{=} \frac{1 - p}{\lambda_2}
\] (4.4)

Taking the log of both sides,

\[
k \log \left( \frac{\lambda_2}{\lambda_1} \right) \overset{\alpha}{=} \log \left( \frac{1 - p}{\lambda_1} \right) + (\lambda_2 - \lambda_1)T
\] (4.5)

Finally, the detection threshold \( k \) is derived as:

\[
k \overset{\alpha}{=} \frac{\log \left( \frac{1 - p}{\lambda_1} \right) + (\lambda_2 - \lambda_1)T}{\log \left( \frac{\lambda_2}{\lambda_1} \right)}
\] (4.6)

When \( p \) is set to 0.5, the first term on the RHS disappears:

\[
k \overset{\alpha}{=} \frac{(\lambda_2 - \lambda_1)T}{\log \left( \frac{\lambda_2}{\lambda_1} \right)}
\] (4.7)

### 4.1.2 Neyman-Pearson Test

The Neyman-Pearson Test differs from the Bayesian Likelihood Ratio Test in that it does not assume any prior, instead using a parameter \( \gamma \) in place of the constant \( \frac{1 - p}{p} \).

\[
\frac{\Pr[K = k|\lambda_1]}{\Pr[K = k|\lambda_2]} \overset{\gamma}{=} \frac{1 - p}{p}
\] (4.8)
As expected, the threshold expression is very similar:

\[ k \geq \frac{\lambda_2 \log(\gamma) + (\lambda_2 - \lambda_1)T}{\log\left(\frac{\lambda_2}{\lambda_1}\right)} \]  \hspace{1cm} (4.9)

The parameter \( \gamma \) can be used to control the false positive rate of the Neyman-Pearson threshold detector.

### 4.2 Receiver Operating Characteristic

A Receiver Operating Characteristic (ROC) curve can be computed for both the Bayesian LRT and the Neyman-Pearson Test. The ROC is a parametric plot of the probability of correct detection given a surge, \( P_d \), with the probability of false alarm given no surge, \( P_f \), swept over the parameter \( p \) for the LRT, or \( \gamma \) for the Neyman-Pearson test. Since the optimal thresholds for both tests are identical except for the differing parameters \( p \) and \( \gamma \), the curve looks the same for both tests. The value of the parameter determines the operating point along the curve; the Neyman-Pearson \( \gamma \) is often determined by some maximum acceptable false positive rate for a given test. However, there is no natural way to set such a max false positive rate for our algorithm, so we do not develop that strategy here.

#### 4.2.1 Error Probabilities

We use a Gaussian approximation to compute the desired error probabilities and create an ROC curve, because of the difficulty of finding closed-form or computationally inexpensive expressions for Poisson tail probabilities. This approximation is reasonable for high packet arrival counts \( k \) due to the Central Limit Theorem, as the arrival distribution will look approximately Gaussian. It is generally less appropriate for very small packet counts, but we will see that the results involving delay are robust to the approximation. Later, the exact error probabilities are computed for determining the average packet delay over time, because it happens to be feasible to sum over the Poisson distribution for very small time windows \( T \) and thresholds \( k \).

Using \( k \) to denote the optimal Bayesian threshold derived above, the Gaussian approximation for the false positive probability \( P_f \) is:

\[ P_f = \frac{1}{2} \text{erfc}\left(\frac{k - \lambda_1 T}{\sqrt{\lambda_1 T}}\right) \]  \hspace{1cm} (4.10)

where

\[ \text{erfc}(z) = \frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{-t^2} dt \]  \hspace{1cm} (4.11)

or two times the one-sided tail probability of the standard normal distribution, starting at \( z \).
The Gaussian approximation for the probability of detection $P_d$ is

$$P_d = \frac{1}{2} \text{erfc}\left(\frac{k - \lambda_2 T}{\sqrt{\lambda_2 T}}\right)$$

Likewise, the probability of missed detection $P_m$ is equal to $1 - P_d$, and the probability of a true negative is $1 - P_f$.

### 4.2.2 ROC curves

The following is the Gaussian error ROC curve with $p = 0.5$, $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_s = 108.571$ packets/s, the standard parameter setting we have been using.

The ten curves shown correspond to $T$ scaled in terms of the expected time to arrival of one packet in the surge case, $1/\lambda_2$ s/packet. The lowest curve is the ROC with $T = 1/\lambda_2$ s, and the highest is $T = 10/\lambda_2$, with higher ROC curves corresponding to more accurate detectors. With only a window size of ten expected packet arrivals during a surge, the detector is surprisingly accurate due to the large gap between the surge packet rate and the non-surge rate. In the non-surge case, so few packets are expected to arrive in the time window that an anomalous threshold crossing in the number of received packets is unlikely.

![ROC curves](image.png)

**Figure 4.1:** Gaussian error ROC curves with $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_s = 108.571$ packets/s. Gray dots are plotted on the $T = 10/\lambda_2$ curve (top curve) to show the operating point when $p = 0.5$ through 0.9 in increments of 0.1, but the points are so close that they are indistinguishable. Regardless of $p$, the performance of the detector is high, with a $P_d$ of nearly one and a $P_f$ of nearly zero.
Figure 4.2: Gaussian error ROC curves with $t_p = 0.007s$, $\lambda_0 = 20$ packets/s, $\rho = 0.5$, and $\lambda_s = 50.43$ packets/s. We see that the performance of the detector is lower than in the $\rho = 0.9$ case due to the surge and non-surge distributions having closer means. Three of the five gray dots on the top curve for $p = 0.5$ through 0.9 are distinguishable, with $p = 0.9$ being the rightmost point and $p = 0.5$ being the leftmost, with the points moving closer together towards the left.
Chapter 5

Performance

To characterize the performance of the re-routing algorithm, we obtain a time series curve of average packet delay at a given node for various values of the algorithm parameter $T$. We show that there is a value of $T$ for which the average peak delay over all states (surge and non-surge) is minimized, through a balance between early detection (small $T$) and low false alarm probability (large $T$). Through this minimization, the network designer may be able to provide better guarantees on average packet delay, or plan for larger traffic surges or bursts than otherwise possible. For now, we fix the Bayesian prior probability of the non-surge state to $p = 0.5$, which contributes to the optimal threshold $k$. In a real-world setting, $p$ would likely be fixed or continuously updated through longer-term network traffic measurement than described here.

5.1 Discrete-time Approximation for Average Packet Delay

The time series curves for delay are produced using a discrete-time finite-state Markov chain approximation for queue length at the surging node. The discrete time steps have length $t_p$; at every time step, exactly one packet currently in the queue is guaranteed to be transmitted, while there is also some probability (depending on the traffic rate parameter $\lambda$) of one or more packets arriving in the queue during that time step. The initial queue length distribution $\pi_0$ is set to the ceiling of the steady state value with probability 1, and the resulting delay at each time step is the total expected wait time of a packet entering the queue at that time. Total expected wait time is calculated as the queue length times transmission time $t_p$, plus a constant known as the residual, representing the remaining time to process the packet currently being serviced.

In order to find the average queue length at each time step, the discrete-time, finite-state Markov chain allows us to create a finite-sized transition probability matrix $P$, representing the state-dependent probabilities of the queue length increasing or decreasing to a given number in a single time step. The new queue length
distribution at time step $n$ may then be computed as the initial state distribution $\pi_0$ (here, the steady state queue length with probability 1) times $P^n$. The average queue length is found by taking the expectation over the length distribution at time step $n$.

The entries of the matrix $P$ are constructed as follows. Entry $j$ of the first row is the probability of an empty queue growing to a queue of size $j - 1$ (starting with index $j = 1$) in the next time step. For example, for the queue to grow to size one (corresponding to $j = 2$), there must be two arrivals in the next time step, so that one can be transmitted while the other remains in the queue; the entry is the probability of exactly two arrivals according to the Poisson distribution $Pr(n) = (\lambda t_p)e^{-\lambda t_p}$. This holds true for all $j$ in the first row except for $j = 1$. For the queue to remain at size zero, either one or zero arrivals may occur in the next time step, since 1 arrival would be transmitted within time $t_p$. Therefore, the entry at $(1, 1)$ must be the sum of the probabilities of one or zero arrivals.

The second row is more straightforward, with the entry in each column $j$ being the probability of having $j$ arrivals. From the third row onward, the entry in row $i$ and column $j$ is equal to zero if $j < i - 1$, and equal to the probability of having $j - i + 1$ arrivals otherwise.

### 5.1.1 Average Delay Time Series

Separate matrices $P_0$, $P_s$, and $P_f$ are constructed based on the values of $\lambda_0$, $\lambda_s$, $f$, and the conditions of the following scenarios: no surge, surge without detection, surge with detection, and no surge with false detection. If the algorithm detects a surge, the matrix $P_s$ is exchanged for $P_f$ in the computation at the expected time step for detection. The optimal $f$ is chosen based on the infinite summation approximation for the average delay, truncated to 10 terms. The time series curves are generated, weighted according to the probabilities of each scenario—True Negative, False Negative, True Positive, and False Positive—and summed together to
produce the time series for average delay.

Examples of these time series curves (both weighted and unweighted) are depicted below.

Figure 5.2: Components of the average delay time series for $T = 5/\lambda_2$ s, $t_p = 0.007$s, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_s = 108.571$ packets/s. The total curve is solid in blue at top, and the other curves in descending order are True Positive (blue), False Negative (orange), True Negative (green), and False Positive (red) time series weighted by their respective probabilities according to the Gaussian approximation.

Figure 5.3: Center curves: average delay time series for values of $T$ from $1/\lambda_2$ to $10/\lambda_2$ (with some curves overlapping). Other parameter values are $t_p = 0.007$s, $\lambda_0 = 20$ packets/s, $\rho = 0.9$, and $\lambda_s = 108.571$ packets/s. Also shown are the (unweighted) False Negative or Missed Detection curve that asymptotes to the new surge steady state value, the True Negative or Non-surge curve at the bottom, and the False Positive curve just above it.

A subtlety of the model is that the False Positive curve is defined as the delay of packets being routed through neighbors directly to the left and right of a falsely triggered node. This is because packets routed through the false-positive node itself will actually have a lower than baseline delay, because it passes its neighbors a $f/2$ fraction of its rate $\lambda_0$ traffic. Therefore, the packet delay in its neighbors' queues is a more accurate metric for additional delay incurred due to the algorithm. For computational simplicity, we do not average over the delay of surge packets routed through the other nodes along the ring. This implies that we
assume the neighboring nodes, through their own traffic estimation, are able to detect the false positive and refrain from continuing to send surge packets down the line of nodes. While this assumption is reasonable in a real-world scenario, if it does not hold, the weighted average of the four delay curves shown is not a true average delay, and results in a slightly increased penalty for false positives over that resulting from the true average. However, the average delay of the re-routed packets over the whole network is not a perfect metric for the false positive case, as it does not capture other potential disadvantages of having the detector on a hair-trigger, such as node configuration energy and time. This is especially true because the amount of traffic added to a node’s neighbors in the non-surge state is very low compared to in the surge state, and does not have a large impact on the average delay.

5.2 Stopping Time

5.2.1 Expected Stopping Time

In the True Positive case where a surge is detected, the expected stopping time, or the time at which detection occurs, determines the height of the peak delay in the time series. Without the re-routing algorithm, the average delay of surge packets increases from the surge time \( t = 0 \), and asymptotically approaches the steady-state delay determined by the new arrival rate \( \lambda_0 + \lambda_s \). (In the case where \( \rho > 1 \) the delay does not asymptote to a steady-state value, but grows to infinity due to buffer overflow.) With the algorithm, a new steady-state delay is established at the surging node once re-routing is triggered, and the queue length then falls to the new steady-state value.

The expected stopping time can be easily computed as the solution \( x \) to the equation

\[
(\lambda_s + \lambda_0)x + \lambda_0(T - x) = k
\]  

(5.1)

where \( x \) is the time elapsed from \( t = 0 \) until the surge is detected, and \( k \) is the threshold for number of packets within time window \( T \). Since packets have been arriving at rate \( \lambda_0 \) before the surge, at time \( x \), \( \lambda_0(T - x) \) packets are in the portion of the window before \( t = 0 \). The remaining packets in the window have arrived at rate \( \lambda_s + \lambda_0 \) for \( x \) time. If the solution \( x \) is less than the window size \( T \), the re-routing algorithm is triggered on average within \( x \) seconds of the surge time.

\[
x = \frac{k - \lambda_0 T}{\lambda_s}
\]  

(5.2)

Future work includes finding a tractable analytic expression for the distribution over the stopping time,
from which a variance for the peak delay at the stopping time can be derived.

5.3 Optimizing Delay over Detection Time $T$

The goal of our algorithm is to minimize the peak average delay, showing that there is a value of the detection parameter $T$ for which the delay is minimized. We find that for each of the load values $\rho = 0.5, 0.9,$ and $1.1,$ the peak delay is minimized at some value of $T$ between $1/\lambda_2$ and $10/\lambda_2$, with larger optimal $T$ values for closer $\lambda_1$ and $\lambda_2$. Below we show the average delays, peak delay values, and the time index of the peak over all values of $T$ tested for a given $\rho$, for both Gaussian approximated and exact Poisson errors.

![Figure 5.4: Showing average delay curves corresponding to $T = n/\lambda_2$ with $n$ from 1 to 10, using the Gaussian approximation for error. Colors in order from $n = 1$ to 10: Blue, orange, light green, red, bluish-purple, dark orange, light blue/teal, gold, dark green, and pinkish-purple (hidden). Because of the discretization of time steps, the overlapping peaks occur when the expected stopping time is the same for different (consecutive) values of $T$. The curves diverge after the peak because the values of $P_f$ and $P_d$ differ ($P_f$ decreases and $P_d$ increases). Parameter setting: $t_p = 0.007s$, $\lambda_1 = 20$ packets/s, $\rho = 0.9$, and $\lambda_2 = 128.571$ packets/s.]

From the time series plots above, we can see that the peak delay may occur early in the time series or later, depending on whether the surge peak or other erroneous peaks (false alarm and missed detection) dominate the average, respectively. Unsurprisingly, the minimum peak over $T$ corresponds to the value of $T$ for which neither the early nor the late peaks is much higher than the other. The plots of the delay peak values by $T$ and the time indices at which they occur show that regardless of the error approximation used, the minimum peak occurs at around the $T$ for which the peak value switches from occurring late (small $T$) to occurring early (large $T$).

This minimum peak and its timing can be observed in both $\rho = 0.5$ and $1.1$ cases as well as $0.9$. 
Figure 5.5: Showing average delay curves corresponding to $T = n/\lambda_2$ with $n$ from 1 to 10, using the Poisson error. Colors in order from $n = 1$ to 10: Blue, orange, light green, red, bluish-purple, dark orange, light blue/teal, gold, dark green, and pinkish-purple (same as above). The Poisson error gives a higher value than the Gaussian for $P_m$, so the penalty for a missed detection is greater (curves rise higher on the left side). Parameter setting: $t_p = 0.007s$, $\lambda_1 = 20$ packets/s, $\rho = 0.9$, and $\lambda_2 = 128.571$ packets/s.
Figure 5.6: Plot of the peak delay for each $T = n/\lambda_2$ for $n = 1$ to $10$. The peak delay has a minimum around $n = 4$ or $5$ in both cases.

(a) Delay weighted by Gaussian approx. for error probabilities

(b) Delay weighted by Poisson error probabilities
Figure 5.7: Time step at which the peak delay occurs for each $T = n/\lambda_2$ for $n = 1$ to 10. Note the abrupt drop at around 4-6 (with some irregularities in the Poisson case).

(a) Using Gaussian approx. for error probabilities

(b) Using Poisson error probabilities
5.3.1 $\rho = 1.1$

Figure 5.8: Showing average delay curves corresponding to $T = n/\lambda_2$ with $n$ from 1 to 10, using the Gaussian approximation for error. Colors in order from $n = 1$ to 10: Blue, orange, light green, red, bluish-purple, dark orange, light blue/teal, gold, dark green, and pinkish-purple. Parameter setting: $t_p = 0.007s$, $\lambda_1 = 20$ packets/s, $\rho = 1.1$, and $\lambda_2 = 157.143$ packets/s.
Figure 5.9: Showing average delay curves corresponding to $T = n/\lambda_2$ with $n$ from 1 to 10, using the Poisson error. Colors in order from $n = 1$ to 10: Blue, orange, light green, red, bluish-purple, dark orange, light blue/teal, gold, dark green, and pinkish-purple (same as above). Parameter setting: $t_p = 0.007s$, $\lambda_1 = 20$ packets/s, $\rho = 1.1$, and $\lambda_2 = 157.153$ packets/s.
Figure 5.10: Plot of the peak delay for each $T = n/\lambda_2$ for $n = 1$ to 10. The peak delay has a minimum around $n = 4$ to 6 in both cases.
Figure 5.11: Time step at which the peak delay occurs for each $T = n/\lambda_2$ for $n = 1$ to $10$. Note the abrupt drop at around 4-6.

(a) Using Gaussian approx. for error probabilities

(b) Using Poisson error probabilities
5.3.2 $\rho = 0.5$

![Delay Curve](image)

Figure 5.12: Showing average delay curves corresponding to $T = n/\lambda_2$ with $n$ from 1 to 10, using the Gaussian approximation for error. Colors in order from $n = 1$ to 10: Blue, orange, light green, red, bluish-purple, dark orange, light blue/teal, gold, dark green, and pinkish-purple. Parameter setting: $t_p = 0.007s$, $\lambda_1 = 20$ packets/s, $\rho = 0.5$, and $\lambda_2 = 71.43$ packets/s.

5.4 Further Improvements

Finally, note that this analysis considers only the case when there is a single surge at $t = 0$, and the algorithm has one chance to detect the surge within time $T$ of its occurrence. From there, we calculate the probability of false alarm or missed detection. However, in actuality, the algorithm has many chances to detect the surge or lack of surge, and may update its belief continuously (within the limits of not using too much bandwidth notifying its neighbors of its surge status). Therefore, a slightly different analysis of the probability of error over time may be required in order to choose the optimal estimator for minimizing delay.
Figure 5.13: Showing average delay curves corresponding to $T = \frac{n}{\lambda_2}$ with $n$ from 1 to 10, using the Poisson error. Colors in order from $n = 1$ to 10: Blue, orange, light green, red, bluish-purple, dark orange, light blue/teal, gold, dark green, and pinkish-purple (same as above). Parameter setting: $t_p = 0.007s$, $\lambda_1 = 20$ packets/s, $\rho = 0.5$, and $\lambda_2 = 71.43$ packets/s.

Figure 5.14: Plot of the peak delay for each $T = \frac{n}{\lambda_2}$ for $n = 1$ to 10. The peak delay has a minimum around $n = 8$.
Figure 5.15: $\rho = 0.5$; Time step at which the peak delay occurs for each $T = n/\lambda_2$ for $n = 1$ to 10. The minimum peak delay should occur at $T\lambda_2 = 8$ or 9.

(a) Using Gaussian approx. for error probabilities

(b) Using Poisson error probabilities
Chapter 6

Conclusion

There is substantial potential for improvement in network performance using the simplest of cognitive routing protocols. Using only the packet arrival count within a constant-sized window, we are able to detect traffic surges with high reliability, thereby increasing the maximum load that the network can handle. Traffic surges that drive $\rho$ to exceed 1 may be redistributed throughout the network.

However, there are many simplifying assumptions in this thesis that may not be true in the real world. An important example is the assumption of a single node surge, which would likely not be valid in a real sensor network scenario. Any event would likely be detected by multiple sensors in the same area. Multiple sensors triggering the re-routing algorithm would require more complex behavior to ensure the spread of traffic away from the source, and prevent surge oscillation back and forth across the network. Additional network overhead may be required for nodes to collaboratively relieve network load, such as the explicit congestion notification and parameter $f$; this overhead prevents our protocol from being fully distributed, but may provide enough performance improvement to merit the cost in throughput. The tradeoff between overhead and information gain to help with cognitive routing and scheduling is a fertile area requiring further study.

It is also important to note that average delay is not the best or only metric for algorithmic performance, and may not be appropriate for many scenarios or models. Finally, recall from the introduction that the simple cognitive method treated in this project does not resemble the more powerful and complex machine learning methods available, and does not scratch the surface of what is practically possible with modern machine learning features and network monitoring.
Chapter 7

Appendix

7.1 Derivation for Bounds on Delay

Recall that the average delay can be represented as the following, which is the same equation given in Chapter 3 but with \( t_p \) multiplying the top and bottom:

\[
\text{Delay} = \sum_{j=1}^{\infty} \sum_{k=1}^{j} \frac{(1 - \frac{\rho_1}{2})t_p(1 - f)f^j}{1 - \rho_1} + \sum_{j=1}^{\infty} \frac{(1 - \frac{\rho_2}{2})t_p(1 - f)f^j}{1 - \rho_2} + \frac{(1 - \frac{\rho_3}{2})t_p(1 - f)}{1 - \rho_3} 
\] (7.1)

where \( \lambda_0, \lambda_s, \) and \( t_p \) are the incoming baseline traffic rate, surge traffic rate, and transmission time as defined in the previous chapter, \( \mu \) is the transmission rate \( \frac{1}{t_p} \), \( \rho_1 = \frac{t_p}{2} \lambda_s f^k \), \( \rho_2 = t_p(\lambda_0 + \frac{1}{2}(1 - f)f^j\lambda_s) \), and \( \rho_3 = t_p(\lambda_0 + (1 - f)\lambda_s) \). In the following sections, the first, second, and third terms on the RHS are referred to as Term 1, Term 2, and Term 3.

7.1.1 Upper Bound

To find an upper bound on the function above, we approximate each of the 3 terms in the delay as a closed form function of \( f \).

Term 1 is upper bounded by setting \( k \) equal to 1 in the inner summation, because that is the value of \( k \) for which the inner sum is largest. Therefore, the new \( \rho_1' = \frac{1}{2} t_p \lambda_s f \).

Then we replace the inner summation over \( k \) with a multiplication by \( j \), since we have replaced each of the \( j \) terms in the inner summation with the largest of them. The sum is now of the form:

\[
\text{Term 1} = \alpha \sum_{j=1}^{\infty} j f^j = \alpha \frac{f}{(1 - f)^2} 
\] (7.2)
where
\[ \alpha = \frac{(1 - \frac{\rho_1^j}{2}) \tau_p (1 - f)}{1 - \rho_1} \] (7.3)
and
\[ \rho_1' = \frac{1}{2} \tau_p \lambda_s f \] (7.4)

Term 2 is upper bounded by setting \( j \) equal to 1 in \( \rho_2 \), turning the term into a sum that has a known closed form:

\[ \text{Term 2} = \beta \sum_{j=1}^{\infty} f^j = \beta \frac{f}{1 - f} \] (7.5)

where
\[ \beta = \frac{(1 - \frac{\rho_2^j}{2}) \tau_p (1 - f)}{1 - \rho_2} \] (7.6)

and
\[ \rho_2' = \tau_p (\lambda_0 + \frac{1}{2} (1 - f) \lambda_s) \] (7.7)

Term 3 is used in its exact form.

The full upper bound is:

\[ \text{Delay} < \frac{(1 - \frac{\rho_1^j}{2}) \tau_p (1 - f)}{1 - \rho_1'} + \frac{(1 - \frac{\rho_2^j}{2}) \tau_p (1 - f)}{1 - \rho_2'} \frac{f}{1 - f} + \frac{(1 - \frac{\rho_3^j}{2}) \tau_p (1 - f)}{1 - \rho_3'} \] (7.8)

where \( \rho_1 = \frac{1}{2} \tau_p \lambda_s f \), \( \rho_2' = \tau_p (\lambda_0 + \frac{1}{2} (1 - f) \lambda_s) \), and \( \rho_3' = \rho_3 = \tau_p (\lambda_0 + (1 - f) \lambda_s) \).

### 7.1.2 Lower Bound

The lower bound is found in a very similar manner to the upper bound.

Term 1 is lower bounded by setting \( k \) equal to \( \infty \), the value of \( k \) for which the term in the inner sum is smallest. Again, we replace the inner summation over \( k \) with a multiplication by \( j \). The sum is now of the form:

\[ \text{Term 1} = \tau_p (1 - f) \sum_{j=1}^{\infty} j f^j = \tau_p (1 - f) \frac{f}{(1 - f)^2} = \frac{\tau_p f}{1 - f} \] (7.9)

since the new \( \rho_1' = 0 \).

Term 2 is lower bounded by setting \( j \) equal to \( \infty \) in the denominator. (The same result can be found via
the Chebyshev sum inequality.)

\[ \text{Term 2} = \left( \frac{t_p(1-f)}{1-t_p\lambda_0} \sum_{j=1}^{\infty} f^j \right) = \frac{t_p(1-f)}{1-t_p\lambda_0} \frac{f}{1-f} = \frac{t_pf}{1-t_p\lambda_0} \quad (7.10) \]

Term 3 is used exactly.

The full lower bound is:

\[ \text{Delay} > \frac{t_pf}{1-f} + \frac{(1-\frac{\rho_2}{2})t_p(1-f)}{1-\rho_2} \frac{f}{1-f} + \frac{(1-\frac{\rho_3}{2})(1-f)}{\mu - \rho_3/t_p} \quad (7.11) \]

where \( \rho_2 = t_p\lambda_0 \) and \( \rho_3 = t_p(\lambda_0 + (1-f)\lambda_s) \).

### 7.2 MM1 Delay

#### 7.2.1 Bounds

The early results for delay bounds and other features of the algorithm were computed for an MM1 queueing model. The MM1 delay and bounds are derived in the same manner as the MD1 delay and bounds, and are given below:

\[ \text{Delay} = \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \frac{t_p(1-f)f^j}{1-\frac{1}{2} t_p\lambda_s f_k} + \sum_{j=1}^{\infty} \frac{t_p(1-f)f^j}{1-t_p(\lambda_0 + \frac{1}{2}(1-f)\lambda_s)} + \frac{t_p(1-f)}{1-t_p(\lambda_0 + (1-f)\lambda_s)} \quad (7.12) \]

Upper bound:

\[ \text{Delay} < \frac{t_pf}{(1-\frac{1}{2} t_p\lambda_s f)(1-f)} + \frac{t_pf}{1-t_p(\lambda_0 + \frac{1}{2}(1-f)\lambda_s)} + \frac{t_p(1-f)}{1-t_p(\lambda_0 + (1-f)\lambda_s)} \quad (7.13) \]

Lower bound:

\[ \text{Delay} > \frac{t_pf}{1-f} + \frac{t_pf}{1-t_p\lambda_0} + \frac{t_p(1-f)}{1-t_p(\lambda_0 + (1-f)\lambda_s)} \quad (7.14) \]

#### 7.2.2 Conditions for needing the algorithm

The condition on the rate parameters \( \lambda_0, \lambda_s, \) and \( t_p \) that determine whether the re-routing algorithm would decrease the average delay at all is derived below. It is summarized as an inequality involving \( \rho_s = (\lambda_s + \lambda_0)t_p \) and \( \rho_0 = \lambda_0 t_p \), which are the surge load and baseline load respectively. A similar computation may be easily done for MD1 delay, but was not completed within the scope of this project.
The delay as a function of \( f \) has a minimum at \( f > 0 \) when the derivative of the delay at \( f = 0 \) is negative. Although the delay function has many terms, most of them conveniently drop out at \( f = 0 \). We break up the derivatives term by term as in the bounds derivation.

\[
\frac{d\text{Term 1}}{df} = \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \left[ \frac{k(1-f)f^{j+k-1}\lambda_s t_p^2}{2(1-\frac{1}{2}f^k\lambda_s t_p)} - \frac{f^jt_p}{1-\frac{1}{2}f^k\lambda_s t_p} + \frac{jt_p(1-f)f^{j-1}}{1-\frac{1}{2}f^k\lambda_s t_p} \right]
\]

(7.15)

Setting \( f = 0 \) in the expression above leaves only the third part, at the index \( j = 1 \) (and \( k = 1 \)). Though \( f^{j-1} \) becomes \( 0^0 \) which is undefined, we can define it as 1 since we are only interested in very small positive values of \( f \), so \( f^0 \) will evaluate to 1.

Thus, the derivative above at \( f = 0 \) is:

\[
\frac{d\text{Term 1}}{df} \bigg|_{f=0} = t_p
\]

(7.16)

Similarly, only the third part of Term 2 survives due to the \( 0^0 \) term when \( j = 1 \):

\[
\frac{d\text{Term 2}}{df} = \sum_{j=1}^{\infty} \left[ \frac{t_p^2(1-f)^2f^{2j-1}\lambda_s(j-\frac{1}{j+1})}{2(1-t_p(\lambda_0 + \frac{1}{2}(1-f)f^j\lambda_s))^2} - \frac{t_p f^j}{1-t_p(\lambda_0 + \frac{1}{2}(1-f)f^j\lambda_s)} + \frac{jt_p(1-f)f^{j-1}}{1-t_p(\lambda_0 + \frac{1}{2}(1-f)f^j\lambda_s)} \right]
\]

(7.17)

\[
\frac{d\text{Term 2}}{df} \bigg|_{f=0} = \frac{t_p}{1-\lambda_0 t_p} = \frac{t_p}{1-\rho_0}
\]

(7.18)

All of Term 3 survives:

\[
\frac{d\text{Term 3}}{df} = -\frac{(1-f)\lambda_s t_p^2}{(1-t_p(\lambda_0 + (1-f)\lambda_s))^2} - \frac{t_p}{1-t_p(\lambda_0 + (1-f)\lambda_s)}
\]

(7.19)

\[
\frac{d\text{Term 3}}{df} \bigg|_{f=0} = -\frac{t_p^2\lambda_s}{(1-t_p(\lambda_0 + \lambda_s))} - \frac{t_p}{1-t_p(\lambda_0 + \lambda_s)} = -\frac{\rho_s t_p}{(1-\rho_s - \rho_0)^2} - \frac{t_p}{1-\rho_s - \rho_0}
\]

(7.20)

This is the resulting statement that the derivative at \( f = 0 \) is negative:

\[
t_p + \frac{t_p}{1-\rho_0} - \frac{\rho_s t_p}{(1-\rho_s - \rho_0)^2} - \frac{t_p}{1-\rho_s - \rho_0} < 0
\]

(7.21)
Rearranging, we get:

\[
1 + \frac{1}{1 - \rho_0} < \frac{\rho_s t_p}{(1 - \rho_s - \rho_0)^2} + \frac{t_p}{1 - \rho_s - \rho_0}
\]

(7.22)

\[
\frac{2 - \rho_0}{1 - \rho_0} < \frac{1 - \rho_0}{1 - \rho_s - \rho_0}
\]

(7.23)

\[
(2 - \rho_0)(1 - \rho_s - \rho_0) < (1 - \rho_0)^2
\]

(7.24)

Expanding both sides and grouping terms:

\[
2 - 2\rho_s - 2\rho_0 - \rho_0 + \rho_0\rho_s + \rho_0^2 < 1 - 2\rho_0 + \rho_0^2
\]

(7.25)

\[
1 - 2\rho_s - \rho_0 + \rho_0\rho_s < 0
\]

(7.26)

\[
1 - \rho_0 < 2\rho_s - \rho_0\rho_s
\]

(7.27)

\[
\frac{1 - \rho_0}{2 - \rho_0} < \rho_s
\]

(7.28)

Note that this condition is automatically fulfilled if \( \rho_s > 1 \), and \( \rho_s \) can be no smaller than \( \frac{1}{2} \) for any value of \( \rho_0 \).

The following figure demonstrates the threshold \( \lambda_s \) above which \( f > 0 \):

![Figure 7.1](image)

Figure 7.1: A log-linear plot of the optimal value of \( f \) as a function of \( \lambda_s \) packets/s for the parameters \( \lambda_0 = 20 \) packets/s and \( t_p = 0.007s \). The sum of the first \( j=1 \) to 100 summation terms is in blue, the upper bound is in orange, and the lower bound is in green. Note that there is a threshold value of \( \lambda_s \) below which the optimal \( f \) is zero.
Chapter 8

Citations


