JOINT THROUGHPUT AND DELAY OPTIMIZATION IN MULTIHOP NETWORKS

BY

NAVID AGHASADEGHI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Professor Bruce Hajek

ABSTRACT

As the use of the Internet evolves, more and more applications require quality of service constraints such as end-to-end delay requirements, in addition to throughput guarantees. We have developed a framework for rate and delay assignment in order to analyze networks of users with both rate and end-to-end delay parameters. Within this theoretical framework, we have proposed three algorithms, evaluated their performance, and analyzed their convergence rates based on both theoretical and numerical results.

To my parents, for their love and support

ACKNOWLEDGMENTS

First and foremost, I would like to thank my adviser, Professor Bruce Hajek, for his guidance and advice, and for showing me how to become a better researcher. This thesis would not have been possible without his invaluable insights and expertise.

My wonderful parents and sister have been the greatest source of emotional support and motivation for me. I am greatly thankful to my father for helping me excel and always believing in me, my mother for her love and for being my best friend, and my sister for her kindness and thoughtfulness and for making me a stronger person.

I would like to thank my dearest Shadi Kashani, for listening to me patiently, for supporting and encouraging me in this endeavor, and for filling my life with many joyful moments.

I would also like to express my thanks to my friends, Ali and Sina Sirjani and Javad Ghaderi, for always motivating me, and for making the last two years of my education memorable ones.

Lastly, my special thanks go to my colleagues and friends at the Coordinated Science Laboratory (CSL)—Farzad, Homa, Saman, Vineet, Jay, Figen and Sara—for useful discussions and for making CSL a friendly and vibrant research environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction and Motivation

The Internet today is a highly sophisticated and rapidly changing technology. Different applications with various requirements make use of the resources that the Internet provides. These applications and users have various parameters such as their rate, delay constraint, etc., and the network has to assign the users' parameters according to the characteristics of the users, and the available network resources. One accepted way of performing this task is to model users as having utilities over their parameters, and subsequently try to optimize some function of the users' utilities subject to the network's resource constraint. In [1], the authors approach the problem of assigning rates to different users in a network by assigning utilities as a function of rate of the users and optimizing the sum of the utilities of the users subject to the availability of the network resources. This problem poses several challenges, one of the most important of which is that the network has to assign resources in a distributed manner. In other words, the network has to solve the problem without knowledge of the utilities of the users.

As Internet applications grow, more and more applications require quality of service (QoS) constraints such as end-to-end delay requirements. For example, different applications such as video or voice could require different QoS constraints (i.e. delay constraints). While the problem of assigning rates to users in a network has been solved using a theoretical framework and in a provably optimal manner [1], the resource allocation problem where users have different rates and also QoS constraints has not been solved in a similar manner. In current networks only a best-effort level of service is provided, which is not sufficient for some applications. Guerin and Peris [2] discuss some of the methods used in providing service guarantees. It therefore becomes more important than ever to analyze systems with QoS constraints.

The idea of pricing resources has been used in [1] to allocate rates efficiently. In this thesis, we want to exploit the idea of pricing, suggested by [3], to come up with an implicit pricing for end-to-end delays. We will consider the problem of providing delay guarantees, and assigning two parameters for burstiness and rate of flow of users in a multiple hop scenario in an optimal manner. We will subsequently define a utility model that depends on both the burstiness and the rate of flow as suggested by [3]. This also fits into the framework of [4]. We will further define the utilities as functions of end-to-end delay, in order to define a reasonable optimization problem.

This thesis is organized in the following manner. In Chapter 2, we will introduce the formal model of this problem, and we will define the optimization problem that we would like to solve. Subsequently, we will introduce two new formulations of the optimization problem, which will later provide the basis for developing new solutions. We will also provide an example of a network with one link and two delays, in order to clarify the notation. In Chapter 3, we will propose different solutions for the formulations, and we will discuss each separately. In Chapter 4, we will derive the solutions for the one link example. Chapter 5 will present the numerical analysis of the different solutions, and finally, we will conclude in the last chapter.

CHAPTER 2

MODEL

The problem we are considering is N users, denoted by the set of users $S = \{1, 2, ..., N\}$, trying to share K links. We will now define the different values and parameters for the links and the users, and we will generally use a superscript to denote the index of the link and a subscript to denote the index of the user. The K links have capacities C^i where $i \in \{1, ..., K\}$. Each link i offers J^i delays to the users using that link. The sum of all the delays offered is $J = \sum_{i=1}^{K} J^i$. The delays offered by each link i are denoted by D_k^i where $k \in \{1, ..., J^i\}$ and $D_{k1}^i < D_{k2}^i$ iff $k1 < k2$. The routes used by the users are predefined and fixed.

Each user s has a parameter σ_s that represents the burstiness of his flow, a parameter ρ_s that represents the rate of his flow, and a parameter d_s^i which represents the delay guarantee that is offered to the user s by link i. We will let $d_s^i = 0$ if user s is not using link i. Also, we define the parameter k_s^i to be the index of the delay used by user s on link i, and we will let $k_s^i = J^i + 1$ if user s is not using link i. Based on [3], we will be assigning utility functions as functions of the three parameters σ_s, ρ_s and d_s where $d_s = \sum_{i=1}^{K} d_s^i$. For simplicity, we will assume that the utility functions have the following separable form: $U_s(\sigma_s, \rho_s, d_s) = U_s^\sigma(\sigma_s) + U_s^\rho(\rho_s) - P_s(d_s)$, where $U_s^\sigma(\sigma_s)$ and $U_s^\rho(\rho_s)$ are increasing concave functions, and $P_s(d_s)$, the penalty associated with having a sum delay of d_s, is an increasing convex function.

Let the set F be the set of all possible delay allocations for all the users on all links. Note that the set F does not contain allocations where users change the links they are using because in our model, we are assuming that routing is already determined. An element $f \in F$ is defined as a K-by-N matrix, where the element f_s^i from the i-th row and the s-th column is equal to k_s^i. We are also going to define certain operations on the matrices in F and also a certain element $\bar{S} \in F$ in the following way:

- Operation "-": for $f, g, h \in F$ we will define $f - g = h$ where $h_s^i = f_s^i$ if

$g_s^i > f_s^i$ and $h_s^i = J^i + 1$ otherwise $\forall i \in \{1, ..., K\}$ and $\forall s \in \{1, ..., N\}$.

- Operation "$\cup$": for $f, g, h \in F$ we will define $f \cup g = h$ where $h_s^i = \min\{f_s^i, g_s^i\}$ $\forall i \in \{1, ..., K\}$ and $\forall s \in \{1, ..., N\}$.
- Operation "$\cap$": for $f, g, h \in F$ we will define $f \cap g = h$ where $h_s^i = \max\{f_s^i, g_s^i\}$ $\forall i \in \{1, ..., K\}$ and $\forall s \in \{1, ..., N\}$.
- Element "$\bar{S} \in F$": $\bar{S} = \cup_{i=1}^{|F|} f_i$, i.e. $\bar{S}$ is the union (as defined above) of all elements in F. This corresponds to the state where users have the lowest delay possible on all the links that they are using.
- Element "$\emptyset \in F$": $\emptyset = \cap_{i=1}^{|F|} f_i$, i.e. $\emptyset$ is the intersection (as defined above) of all elements in F. This corresponds to the state where users have the highest delay possible on all links that they are using.

The user parameters σ_s, ρ_s and d_s could be functions of the delay allocation of the system. In order to demonstrate this, the following notation will be used. The delay of user s on each link i and the sum delay of user s as functions of the delay allocation of the system will be denoted by $d_s^i(f)$ and $d_s(f)$, respectively. The delay index of user s at link i will be denoted by $k_s^i(f)$. The burstiness and rate of flow of user s will be denoted by $\sigma_s(f)$ and $\rho_s(f)$.

One Link Example: In order to further explain the model, we will also provide formulations for a single link example in the next sections. For the single link example, we are going to define N-dimensional vectors $f \in F$ as the state of system. Also, we will not use a superscript, since there is only one available link. Letting f_s be the s-th element of f, then $f_s = 1$ if user s has $d_s = D_1$ and $f_s = 2$ if user s has $d_s = D_2$. Furthermore, only for the single link example, we are going to interpret f as a set too in the following way. We say $s \in f$ if $d_s = D_1$ and we say $s \notin f$ if $d_s = D_2$. The set F is then the set of all possible vectors or, equivalently, the power set of S. Moreover, only for this example, the penalty function is defined as $P_s(d_s) = -K_s\ I_{\{d_s = D_1\}}$.

Our problem is to allocate the capacities of the links and provide delay guarantees in a way that the sum of utilities of all users is maximized. In the next section, we will formally introduce this optimization problem.

2.1 First and Original Formulation

We can write in the following way the constraints that have to be met for each link to be able to guarantee the QoS of the users of that link. Based on the theory of deterministic network calculus, each link i offering J^i delays will have to satisfy $J^i + 1$ linear constraints. We can therefore describe the constraints in the following compact form. We will define a J^i-by-2 matrix $A_s^{i,f}$ for each $s \in S$ and link $i \in L$ and delay configuration $f \in F$. The first column of this matrix corresponds to the contribution of the burstiness of user s to each of the constraints of link i, and the second column corresponds to the contribution of the rate of user s to each of the constraints of link i. Formally, the elements of this matrix for a given delay allocation f are defined in the following way:

$$
\begin{aligned}
&A_s^{i,f}(j,1) = I_{\{k_s^i(f) \le j \le J^i\}} \\
&A_s^{i,f}(j,2) = (D_j^i - d_s^i(f)) I_{\{k_s^i(f) \le j \le J^i\}} + I_{\{j = J^i+1 \text{ and } k_s^i(f) \le J^i\}} \\
&\text{where } 1 \le j \le J^i + 1.
\end{aligned}
$$

The $I_{\{\}}$ symbol is used to represent an indicator function. Now we define the matrix of constraints of link i under delay allocation f as the horizontal concatenation of all the matrices corresponding to the users, $A^{i,f} = [A_1^{i,f}, ..., A_N^{i,f}]$. Subsequently, we will define the matrix A^f as the vertical concatenation of all such matrices in the following way: $A^f = [A^{1,f}; ...; A^{K,f}]$. Now we can define capacity vectors, for each link $i \in L$, as $\bar{C}^i$, where the j-th element of this vector is defined by $C_j^i = C^i D_j^i$ for $1 \le j \le J_i$ and $C_j^i = C^i$ if $j = J^i + 1$. The vector C is subsequently defined as the vertical concatenation of these vectors: $C = [\bar{C}^1; ...; \bar{C}^K]$. Lastly, we are going to define the matrix of all the σ and ρ parameters under the delay allocation f as the matrix X^f where $X^f(1,s) = \sigma_s$ and $X^f(2,s) = \rho_s$ for $s \in S$.

With these definitions in mind, the optimization problem we are trying to solve is the following:

$$\max_{\sigma\geq 0,\rho\geq 0,f\in F}\sum_{s\in S}U_s(\sigma_s,\rho_s,d_s(f)) = \max_{\sigma\geq 0,\rho\geq 0,f\in F}\sum_{s\in S}\Big[U_s^{\sigma}(\sigma_s)+U_s^{\rho}(\rho_s)-P_s(d_s(f))\Big]$$
$$s.t.$$
$$A^fX^f \leq C \tag{2.1}$$

The constraint $A^fX^f \leq C$ is derived based on the theory of network calculus, and if met, ensures that the network can guarantee the delays in delay allocation f to all the users. For example, the constraint makes sure that the sum of the rates ρ on each link are less than or equal to the capacity of the link. Moreover, there are other constraints imposed on both the σ and ρ of the user, which are used in providing delay guarantees. A summary of deterministic network calculus along with a derivation of the above constraints can be found in [3].

This problem has an optimization over both continuous and discrete parameters. If we hold the delay allocation f fixed, optimization over the continuous parameters could be done using well known convex optimization techniques as discussed in [1]. However, optimization over the discrete delay allocations involves solving a complicated combinatorial problem. Therefore, we will try to change the formulation of the problem in order to come up with more tractable solutions. To do this, instead of finding the optimum delay allocation f^*, we will assign probabilities p_f to each of the delay allocations $f \in F$, and we will try to optimize over the probabilities p_f in a reasonable formulation of the problem.

2.1.1 One Link Example of the First Formulation

The original formulation for a one link example where two delays are offered is as follows:

$$\begin{aligned} \max_{\sigma\geq 0,\rho\geq 0,f\in F} & \sum_{s\in S} U_s(\sigma_s,\rho_s,d_s) \\ & s.t. \\ & \sum_{s:s\in f} \sigma_s \leq CD_1 \\ & \sum_{s\in S}\sigma_s + (D_2 - D_1)\sum_{s:s\in f}\rho_s \leq CD_2 \\ & \sum_{s\in S}\rho_s \leq C \end{aligned} \tag{2.2}$$

2.2 Second Formulation

As mentioned in the previous section, the original formulation of the problem involves solving a complicated combinatorial optimization. In order to simplify the problem, we try to derive other formulations of the problem by assigning probabilities p_f to each of the delay allocations f, and then optimizing over the probabilities according to the framework of [5]. One way to do this is to have the constraints met in a probabilistic fashion, and still have only one σ_s and ρ_s value for each user. Using this idea we relax our constraints by enforcing them on averages of $f's \in F$, rather than for each $f \in F$.

In this formulation, however, each user s has only one σ_s and ρ_s parameter, independent of the delay state f of the system. Therefore, we are going to define the matrix of all the σ and ρ parameters as the matrix X where $X(1,s) = \sigma_s$ and $X(2,s) = \rho_s$, $\forall s \in S$, independent of the delay state f.

With these definitions in mind, the relaxed optimization problem we propose is the following:

$$
\begin{aligned}
\max_{\sigma\geq 0,\rho\geq 0,p\geq 0} \sum_{s\in S} & \left[U_s^{\sigma}(\sigma_s) + U_s^{\rho}(\rho_s) - \sum_{f\in F}\left[p_f P_s(d_s(f))\right]\right] \\
& s.t. \\
& \left(\sum_{f\in F} p_f A^f\right) X \leq C \\
& \sum_{f\in F} p_f = 1.
\end{aligned}
\tag{2.3}
$$

Note that because of the independence of the user parameters σ_s and ρ_s of the delay vector f, we have formulated the problem so that the constraints are met only on average, i.e. there could be times where for a certain delay vector, and for parameters σ_s and ρ_s, the constraints are not satisfied, and the link cannot guarantee providing the QoS requirements. Because of the difference in the formulation of the problem, it is not clear if the answer to this problem is the answer to the original formulation of the problem as well. However, we can argue that this formulation is worth investigating because the solution to this formulation is always larger than the solution to the original formulation (at the cost of having constraints met on average.) This is because the set of parameters that satisfy the constraints of the first formulation is a subset of the set of parameters that satisfy the constraints of the second formulation. In other words, any set of parameters that could be a solution to the first formulation, also satisfies the constraints of the second formulation. Moreover, this solution will be particularly useful if the optimal probability distribution for the second formulation turns out to be concentrated around one particular delay configuration, or possibly few configurations that do not differ by much. In that case, the constraints of the first formulation are not going to be violated by much, further making this formulation a valuable one to investigate. We will further discuss this issue when we develop the solutions and implement the algorithms.

2.2.1 One Link Example of the Second Formulation

To further demonstrate the second formulation, we have included an example of a one link network with two delay offerings:

$$
\begin{aligned}
\max_{\sigma\geq 0,\rho\geq 0,p\geq 0} \sum_{s\in S} & U_s^\sigma(\sigma_s) + U_s^\rho(\rho_s) + K_s \sum_{f:s\in f} p_f \\
& s.t. \\
& \sum_{s\in S} \sigma_s \sum_{f:s\in f} p_f \leq CD_1 \\
& \sum_{s\in S} \sigma_s + (D_2 - D_1) \sum_{s\in S} \rho_s \sum_{f:s\in f} p_f \leq CD_2 \\
& \sum_{s\in S} \rho_s \leq C \\
& \sum_{f\in F} p_f = 1
\end{aligned}
\tag{2.4}
$$

2.3 Third Formulation

As we saw in the previous section, the second formulation cannot guarantee QoS all the time since σ and ρ parameters do not change for different delay vectors. In order to solve this issue, we introduce a new formulation of the problem in this section. In this formulation, we are going to allow different σ and ρ parameters for different delay vectors. We denote the burstiness and rate of a user s under the configuration (delay vector) f as σ_s^f and ρ_s^f respectively. We therefore have more parameters to solve for, and also more inequalities that have to be satisfied. Therefore, in the formulation below, the constraints have to be met $\forall f \in F$. The third formulation then becomes the following:

$$
\begin{aligned}
\max_{\sigma\geq 0,\rho\geq 0,p\geq 0} \sum_{f\in F} p_f \sum_{s\in S} & \left[U_s^\sigma(\sigma_s^f) + U_s^\rho(\rho_s^f) - P_s(d_s^f) \right] \\
& \text{subject to the following constraints} \\
& A^f X^f \leq C, \forall f \in F \\
& \sum_{f\in F} p_f = 1
\end{aligned}
\tag{2.5}
$$

where $\sigma = [\sigma^f : f \in F]$ and $\rho = [\rho^f : f \in F]$, and σ^f and ρ^f are vectors of the parameters for all users under configuration f.

Note that the solution to the third formulation is exactly the solution of the original formulation (assuming the optimal sum of utilities only happens in one particular delay state). This is because the optimal σ and ρ parameters obtained by solving the third formulation will always satisfy the constraints of the first formulation, no matter what delay configuration we have. Also, if a particular delay vector is the optimal delay vector in the first formulation, i.e. the optimal feasible sum of utilities for that delay vector is more than the optimal feasible sum of utilities for any other delay vector, then the solution to the third formulation will also assign all of the probability mass to that delay vector, and therefore both solutions will yield the same optimal sum of utilities.

2.3.1 One Link Example of the Third Formulation

To further demonstrate the third formulation, we have included an example of a one link network with two delay offerings:

$$\begin{aligned}
\max_{\sigma\geq 0,\rho\geq 0,p\geq 0} \sum_{f\in F} p_f \sum_{s\in S} & \left[U_s^\sigma(\sigma_s^f) + U_s^\rho(\rho_s^f) - P_s(d_s^f)\right] \\
& \text{subject to the following constraints } \forall f \in F \\
& \sum_{s:s\in f} \sigma_s^f \leq CD_1 \\
& \sum_{s\in S} \sigma_s^f + (D_2 - D_1) \sum_{s:s\in f} \rho_s^f \leq CD_2 \\
& \sum_{s\in S} \rho_s^f \leq C \\
& \sum_{f\in F} p_f = 1
\end{aligned} \tag{2.6}$$

where $\sigma = [\sigma^f : f \in F]$ and $\rho = [\rho^f : f \in F]$, and σ^f and ρ^f are vectors of the parameters for all users under configuration f.

CHAPTER 3

SOLUTIONS

In this chapter, we discuss three solutions for the three problem formulations in the last chapter. In each section, we explain in detail the proposed solution, and we present relevant theoretical results and numerical simulations to show the performance of each solution. The basic idea we use is that since we have assigned utilities to the σ and ρ parameters of the users with different delays, we now have access to different prices for the σ and ρ parameters of each user, depending on the delay used by that particular user. These prices for the parameters at different delays can implicitly offer prices for the different delays at the different links. We can therefore try to come up with solutions where we allow the users to change their delays given the prices offered by the network.

3.1 First Solution: Direct Utility Comparison

The first solution we offer to this problem is the simplest one. Assuming we start from an initial delay allocation among the users, we solve the maximization in the first formulation of the problem for that fixed delay allocation, i.e. we allow enough time for the distributed algorithms to get close to optimal values of σ and ρ for the given delay vector. Now, for this given setup, we have prices for different constraints. The network can then provide the prices for the different constraints to the users. Subsequently, the network can choose one or a few users at any time instant, and allow them to change their delay allocation according to the prices of the different constraints. In this algorithm, we allow some time after every change period for the parameters to converge close to the optimal values, before we select more users to change their delay allocation.

Each user, if selected by the network, individually looks at the prices offered by the network and determines which delay allocation maximizes its utility. The user makes this decision with the assumption that he pays an amount equal to the value of the σ or ρ parameter multiplied by the price of that parameter. The user then chooses the best delays on all links as its new delay allocation. Note that this algorithm relies heavily on the assumption that prices do not change drastically when one user changes its delay allocation. This assumption is a reasonable one if the network is rather large and the number of users selected in every time slot is not big compared to the size of the network. Otherwise, if a large number of users change their delay allocations in one time slot, the prices they had seen will not be a good indication of the future prices, because the prices might vary a lot after all the users change their allocations.

As already discussed, this heuristic algorithm relies heavily on the assumption that the prices offered to users at a time instant will not change considerably. This assumption therefore imposes a trade-off on this algorithm. On one hand, if the number of users selected by the network is small, the prices are expected to remain close to their values at the time they were offered, and, therefore, the users are more likely to make efficient changes to their delay allocation. On the other hand, if the number of users selected to change their delay allocation is small, the network will have a slower evolution to its optimal or close-to-optimal allocation. Therefore, choosing a number of users to be selected is a critical decision in this algorithm, and it might not be clear how to choose this number in a systematic way a priori. Additionally, this algorithm clearly requires the network to be in close contact with the users in order to inform them whether or not they can change their delays.

3.2 Second Solution: Payment Comparison

In this section, we find implementable solutions to the second formulation. The solution we develop has two major components. One component involves a Markov chain, which we will assume has faster dynamics than the dynamics of the parameters of the users, and will change the delay vector of the system. The second component involves optimizing the σ_s and ρ_s parameters for different users $s \in S$. In this section, we solve for the optimal probability distribution, and we will use the optimal probability distribution as the stationary distribution of a carefully designed discrete time Markov chain. The optimal σ, ρ, and μ parameters are later obtained based on the optimal probability distribution, i.e. given the assumption that the Markov chain converges instantaneously. In fact, the need for the Markov chain to converge instantaneously is why we assume faster dynamics for the Markov chain compared to the parameters and the prices. It is nonetheless clear that in practice we Slater have the Markov chain converge instantaneuosly, and therefore the optimal probability distribution is not going to be achieved completely. Thus, the σ, ρ and μ parameters are going to be time varying.

Since the second formulation of the problem is also complicated, we solve the following approximation to the second formulation introduced in Section 2.2:

$$\begin{aligned} \max_{\sigma\geq 0,\rho\geq 0,p\geq 0} \sum_{s\in S} & \left[U_s^{\sigma}(\sigma_s) + U_s^{\rho}(\rho_s) - \sum_{f\in F}\left[p_f P_s(d_s(f))\right]\right] - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \\ & s.t. \\ & AX \leq C \\ & \sum_{f\in F} p_f = 1. \end{aligned} \tag{3.1}$$

Note that the only difference between this optimization problem and that in the second formulation is the fact that the objective function of this new approximation involves an entropy term scaled by $\frac{1}{\beta}$, which increases with increasing randomness in the distribution p. However, since the entropy term is multiplied by $\frac{1}{\beta}$, it can arbitrarily approximate the second formulation as β gets large, i.e. the effect of the entropy term on the solution decreases as β increases.

3.2.1 Solving for the Optimal Probability Distribution

In order to derive algorithms to solve the second formulation of the problem, we will first solve for the optimal probability distribution p^* given any set of σ and ρ parameters. Now, we can write the optimization in (3.1) only over the probability distribution as the following:

$$\begin{aligned}
&\max_{p\geq 0} \sum_{s\in S}\Big[U_s^\sigma(\sigma_s)+U_s^\rho(\rho_s)-\sum_{f\in F}\big[p_f P_s(d_s(f))\big]\Big]-\frac{1}{\beta}\sum_{f\in F}p_f\log p_f\\
&\quad s.t.\\
&\quad AX\leq C\\
&\quad \sum_{f\in F}p_f=1.
\end{aligned} \tag{3.2}$$

Since the objective and the constraints are linear in p (because A is linear in p), and Slater's condition holds, we can use strong duality and solve the following problem:

$$\begin{aligned}
&\min_{\mu\geq 0}\max_{p\geq 0} \sum_{s\in S}\Big[U_s^\sigma(\sigma_s)+U_s^\rho(\rho_s)-\sum_{f\in F}\big[p_f\ P_s(d_s(f))\big]\Big]-\frac{1}{\beta}\sum_{f\in F}p_f\log p_f\\
&\qquad -\sum_{i=1}^{K}\sum_{j=1}^{J^i}\Big[\mu_j^i\big[\sum_{s\in S}\Big(A_s^i(j,1)\sigma_s+A_s^i(j,2)\rho_s\Big)\big]\Big]+\sum_{i=1}^{K}\sum_{j=1}^{J^i}\mu_j^iC_j^i\\
&\qquad s.t.\\
&\qquad \sum_{f\in F}p_f=1
\end{aligned}$$

which can be expanded and written as

$$\begin{aligned}
&\min_{\mu\geq 0}\max_{p\geq 0} \sum_{s\in S}\Big[U_s^\sigma(\sigma_s)+U_s^\rho(\rho_s)-\sum_{f\in F}\big[p_f\ P_s(d_s(f))\big]\Big]-\frac{1}{\beta}\sum_{f\in F}p_f\log p_f\\
&\qquad -\sum_{i=1}^{K}\sum_{j=1}^{J^i}\Big[\mu_j^i\big[\sum_{s\in S}\Big(\big[\sum_{f\in F}\big[p_f\ I_{\{k_s^i(f)\leq j\leq J^i\}}\big]\big]\sigma_s+\\
&\qquad \big[\sum_{f\in F}p_f\big[\ (D_j^i-d_s^i(f))I_{\{k_s^i(f)\leq j\leq J^i\}}+I_{\{j=J^i+1 \text{ and } k_s^i(f)\leq J^i\}}\big]\rho_s\big]\Big)\big]\Big]\\
&\qquad +\sum_{i=1}^{K}\sum_{j=1}^{J^i}\mu_j^iC_j^i\\
&\qquad s.t.\\
&\qquad \sum_{f\in F}p_f=1.
\end{aligned}$$

We will use μ_j^i to denote the Lagrange multiplier on the j-th constraint of the i-th link. We can then rewrite the problem in the following way, and then maximize with respect to the probability distribution p:

$$\begin{aligned}
\max_{p\geq 0} \sum_{f\in F} p_f \Bigg[\sum_{s\in S} \Bigg(& - P_s(d_s(f)) \\
& - \sum_{i=1}^{K}\sum_{j=1}^{J^i} \Bigg[\mu_j^i \Bigg(I_{\{k_s^i(f)\leq j\leq J^i\}}\sigma_s + (D_j^i - d_s^i(f)) I_{\{k_s^i(f)\leq j\leq J^i\}}\rho_s \\
& + I_{\{j=J^i+1 \text{ and } k_s^i(f)\leq J^i\}}\rho_s \Bigg)\Bigg]\Bigg)\Bigg] - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \\
& s.t. \\
& \sum_{f\in F} p_f = 1.
\end{aligned}$$

For each user s and each delay allocation f we can define the parameter

$$\begin{aligned}
V_s(f) = \Bigg(& - P_s(d_s(f)) \\
& - \sum_{i=1}^{K}\sum_{j=1}^{J^i} \Bigg[\mu_j^i \Bigg(I_{\{k_s^i(f)\leq j\leq J^i\}}\sigma_s + (D_j^i - d_s^i(f)) I_{\{k_s^i(f)\leq j\leq J^i\}}\rho_s \Bigg)\Bigg).
\end{aligned}$$

The maximization problem now becomes:

$$\begin{aligned}
\max_{p\geq 0} \sum_{f\in F} p_f & \Bigg[\sum_{s\in S} V_s(f) \Bigg] - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \\
& s.t. \\
& \sum_{f\in F} p_f = 1.
\end{aligned}$$

Now, we can verify the Karush-Kuhn-Tucker (KKT) conditions to find the optimal stationary distribution, denoted by p^*, as the following:

$$p^*_f(V) = \frac{exp(\beta \sum_{s\in S} V_s(f))}{\sum\limits_{f'\in F} exp(\beta \sum\limits_{s\in S} V_s(f'))} \forall f \in F.$$

Note that if there exists a unique state $f^* = argmax_{f\in F} \sum_{s\in S} V_s(f)$, as β increases to infinity, the optimal probability distribution is going to be only allocated to the delay configuration f^*, therefore ensuring that the constraints of the first formulation are met as well (given optimal σ and ρ parameters.)

How to use this result: One way to use this result is to treat this probability distribution as the stationary distribution of a carefully chosen discrete time Markov chain, and design the transition probabilities to achieve the desired stationary distribution. Moreover, in addition to finding transition probabilities that will give rise to the above stationary distribution, it is also important to choose transition probabilities which will allow a distributed implementation of the Markov chain. We will derive transition probabilities below using the detailed balance equation. We will allow transitions among any two pairs of states $f_i, f_j \in F$. Note that allowing transitions among any two states would intuitively produce a Markov chain that has a fast convergence rate. Moreover, we will see that such transitions could be easily implemented distributedly. Let q_{f_i,f_j} denote the transition probability from state f_i to state f_j. We will write the detailed balance equations in the following way:

$$p^*_{f_i} \times q_{f_i,f_j} = p^*_{f_j} \times q_{f_j,f_i} \text{ and let } q_{f_i,f_j} = \frac{\exp\left(\beta \sum\limits_{s\in S} \big[V_s(f_i \cup f_j) - V_s(f_i)\big]\right)}{\prod\limits_{s\in S} \gamma(1 + \exp(\beta V_s(f_i \cup f_j))} \Leftrightarrow$$

$$p^*_{f_i} \times q_{f_i,f_j} = \frac{exp(\beta \sum_{s\in S} V_s(f_i))}{\sum\limits_{f'\in F} exp(\beta \sum\limits_{s\in S} V_s(f'))} \times \frac{\exp\left(\beta \sum\limits_{s\in S} \big[V_s(f_i \cup f_j) - V_s(f_i)\big]\right)}{\prod\limits_{s\in S} \gamma(1 + \exp(\beta V_s(f_i \cup f_j))} =$$

$$\frac{\exp\left(\beta \sum\limits_{s\in S} V_s(f_i \cup f_j)\right)}{\left[\sum\limits_{f'\in F} exp(\beta \sum\limits_{s\in S} V_s(f'))\right] \prod\limits_{s\in S} \gamma(1 + \exp(\beta V_s(f_i \cup f_j)))} = p^*_{f_j} \times q_{f_j,f_i}.$$

3.2.2 A Distributed Implementation of the Discrete Time Markov Chain

A distributed implementation of the discrete time Markov chain discussed is possible. In this distributed implementation, every user will have a separate Markov chain based on the prices offered by the network and its own parameters. The user will decide to remain in the current delay state or change its delay state based on that Markov chain.

In order to have a distributed implementation of this algorithm, let every user $s \in \{1, ..., N\}$ change delays on every link that it is using according to the following algorithm:

Delay Changing Algorithm: If user s has delay D^i_j on link i, he will change to D^i_k with probability $q^s_{D^i_j,D^i_k} = \dfrac{\exp(\beta[V_s(f_j \cup f_k) - V_s(f_j)])}{\gamma(1 + \exp(\beta V_s(f_j \cup f_k)))}$ and he will not make that change with probability $1 - q^s_{D^i_j,D^i_k}$. In this transition probability, f_j is the current state of the system with user s having delay D^i_j on link i, and f_k is the state of the system where all users remain the same, and only user s changes his delay on link i to D^i_k.

It is easy to verify that if all users act according to the above algorithm, we will have the mentioned transition probabilities between different states in F. Note that these changes can be done in a completely distributed manner, since each user s knows the value of $V_s(f_j \cup f_k) - V_s(f_j)$ (used in the above algorithm), because it only depends on the prices (which could be shared with the users by the network) and parameters of the user who is changing his delay. Therefore, the Markov chain can be implemented in a completely distributed manner.

3.2.3 Solving for Optimal σ, ρ and μ Parameters

If we assume that the discrete time Markov chain designed in the previous section converges to its optimal stationary distribution instantaneously, we can then solve for the optimal σ and ρ parameters, along with optimal μ prices for the network. Assuming we have the optimal probability distribution p^*, we can solve for the user parameters and prices by solving the approximation to the second formulation only with respect to σ, ρ and μ. Since for given p_f^*'s, the problem is convex, and Slater's condition holds, then there is no duality gap and we can solve the following problem:

$$\begin{aligned}\min_{\mu\geq 0}\max_{\sigma\geq 0,\rho\geq 0}\sum_{s\in S}[U_s^{\sigma}(\sigma_s)+U_s^{\rho}(\rho_s)]+\sum_{f\in F}p_f^*\Bigg[\sum_{s\in S}\Bigg(&-P_s(d_s(f))\\ -\sum_{i=1}^{K}\sum_{j=1}^{J^i}\Bigg[\mu_j^i\Bigg(&I_{\{k_s^i(f)\leq j\leq J^i\}}\sigma_s+\\ (D_j^i-d_s^i(f))I_{\{k_s^i(f)\leq j\leq J^i\}}\rho_s&+I_{\{j=J^i+1\text{ and }k_s^i(f)\leq J^i\}}\rho_s\Bigg)\Bigg]\Bigg)\Bigg]\\ &+\sum_{i=1}^{K}\sum_{j=1}^{J^i}\mu_j^iC_j^i.\end{aligned}$$

We can write the following primal-dual algorithm for finding the optimal values of the parameters and the prices.

$$\begin{aligned}\dot{\mu}_j^i =& k_j^i(\mu_j^i)\Big[\sum_{f\in F}\sum_{s\in S} p_f^*\Big(I_{\{k_s^i(f)\le j\le J^i\}}\sigma_s + (D_j^i - d_s^i(f))I_{\{k_s^i(f)\le j\le J^i\}}\rho_s \\ &+ I_{\{j=J^i+1 \text{ and } k_s^i(f)\le J^i\}}\rho_s\Big) - C_j^i\Big]_{\mu_j^i}^+ \\ \dot{\sigma_s} =& h_s(\sigma_s)\Big[U_s^{\sigma\prime}(\sigma_s) - \sum_{f\in F} p_f^* \sum_{i=1}^{K}\sum_{j=1}^{J^i}\Big(\mu_j^i I_{\{k_s^i(f)\le j\le J^i\}}\Big)\Big]_{\sigma_s}^+ \\ \dot{\rho}_s =& r_s(\rho_s)\Big[U_s^{\rho\prime}(\rho_s) - \sum_{f\in F} p_f^* \sum_{i=1}^{K}\sum_{j=1}^{J^i}\Big((D_j^i - d_s^i(f))I_{\{k_s^i(f)\le j\le J^i\}} \\ &+ I_{\{j=J^i+1 \text{ and } k_s^i(f)\le J^i\}}\Big)\Big]_{\rho_s}^+,\end{aligned}$$

where k, h and r are positive and continuous functions, and the function $[x]_y^+ = \max(0, x)$ if $y \le 0$ and equals x otherwise. Note that in the above update equations, the network and each of the nodes need to know the p_f^* values, which could be hard to obtain. However, we can rewrite the update equations in another way, which although it is quite similar, allows for another interpretation and implementation of the algorithm, as will be explained below.

$$\begin{aligned}\dot{\mu}_j^i =& k_j^i(\mu_j^i)\Big[\sum_{f\in F}\sum_{s\in S} p_f^*\Big(I_{\{k_s^i(f)\le j\le J^i\}}\sigma_s + (D_j^i - d_s^i(f))I_{\{k_s^i(f)\le j\le J^i\}}\rho_s \\ &+ I_{\{j=J^i+1 \text{ and } k_s^i(f)\le J^i\}}\rho_s - C_j^i\Big)\Big]_{\mu_j^i}^+ \\ \dot{\sigma_s} =& h_s(\sigma_s)\sum_{f\in F} p_f^*\Big[U_s^{\sigma\prime}(\sigma_s) - \sum_{i=1}^{K}\sum_{j=1}^{J^i}\Big(\mu_j^i I_{\{k_s^i(f)\le j\le J^i\}}\Big)\Big]_{\sigma_s}^+ \\ \dot{\rho}_s =& r_s(\rho_s)\sum_{f\in F} p_f^*\Big[U_s^{\rho\prime}(\rho_s) - \sum_{i=1}^{K}\sum_{j=1}^{J^i}\Big((D_j^i - d_s^i(f))I_{\{k_s^i(f)\le j\le J^i\}} \\ &+ I_{\{j=J^i+1 \text{ and } k_s^i(f)\le J^i\}}\Big)\Big]_{\rho_s}^+.\end{aligned}$$

Now note that by writing the update equations in the above form we can derive the following implementable algorithm that will not require knowledge of the values of p_f^*.

Implementation: In order to implement the above update equations, the users and the network simply do the following. Whenever the system is in a state f, the prices and parameters should be updated according to the setup of that configuration. In other words, when the system is in configuration f, the network and the users can assume that they are permanently in that configuration. Therefore, the network should update prices for each resource depending only on the users that are using that resource under configuration f, and the users should receive prices for a particular resource only if they are using that resource in that configuration. Now, because the system is going to be in that configuration a p_f^* portion of the time, if users use the update equation corresponding to that configuration while in that state, the update equation will automatically be multiplied by the value of p_f^*. This means that everyone just has to act depending on the configuration they are in, and no one needs to know the values of p_f^*. Again, note that in our implementation of this algorithm, we have a Markov chain that converges instantaneously, and, therefore, the system does not spend exactly a fraction p_f^* in each delay configuration f.

In order to implement this algorithm, we will use the following discrete version of the above update equations.

$$\begin{aligned}
\mu_j^i(t+1) =& \mu_j^i(t) + k_j^i \Bigg[\sum_{s \in S} \Bigg(I_{\{k_s^i(f) \le j \le J^i\}} \sigma_s + (D_j^i - d_s^i(f)) I_{\{k_s^i(f) \le j \le J^i\}} \rho_s \\
& + I_{\{j = J^i + 1 \text{ and } k_s^i(f) \le J^i\}} \rho_s - C_j^i \Bigg) \Bigg]_{\mu_j^i}^+ \\
\sigma_s(t+1) =& \sigma_s(t) + h_s(\sigma_s) \Bigg[U_s^{\sigma\prime}(\sigma_s) - \sum_{i=1}^{K} \sum_{j=1}^{J^i} \Big(\mu_j^i I_{\{k_s^i(f) \le j \le J^i\}} \Big) \Bigg]_{\sigma_s}^+ \\
\rho_s(t+1) =& \rho_s(t) + r_s(\rho_s) \Bigg[U_s^{\rho\prime}(\rho_s) - \sum_{i=1}^{K} \sum_{j=1}^{J^i} \Bigg((D_j^i - d_s^i(f)) I_{\{k_s^i(f) \le j \le J^i\}} \\
& + I_{\{j = J^i + 1 \text{ and } k_s^i(f) \le J^i\}} \Bigg) \Bigg]_{\rho_s}^+ .
\end{aligned}$$

Note that the described implementation would be exact if the Markov chain converges to its stationary distribution instantaneously, and if users and network update the parameters and the prices at a slower rate compared to the evolution of the Markov chain. Since this is clearly not possible in practice, we will perform numerical experiments, and evaluate the results to see how the relaxation of this assumption affects the performance of the algorithm. It is now evident that the convergence rate (mixing time) of the Markov chain would be a critical issue in the implementation of this solution. We will therefore devote the next subsection to analyzing the mixing time of the Markov chain.

3.2.4 Bounding the Mixing Time of the Markov Chain

In this section, we use geometric bounds derived by [6] to bound the mixing time of our Markov chain.

The following notation is adopted from [6]. Define Γ to be the set of all one-edge paths between any two states in the Markov chain. This set is going to contain paths between any two states since there is an edge between any two states of the Markov chain. Moreover, the set Γ will only contain paths between different states in the Markov chain. Define the weight of the edge between two states f_i and f_j as $Q(f_i, f_j) = p^*_{f_i} \times q_{f_i,f_j} = p^*_{f_j} \times q_{f_j,f_i}, \forall f_i, f_j \in F$, since the chain is reversible. Also define $\gamma_{f_i,f_j} \in \Gamma$ as the path between two states f_i and f_j. Define the path length $|\gamma_{f_i,f_j}|_Q = \sum_{e \in \gamma_{f_i,f_j}} Q(e)^{-1}$. Since we have defined the Markov chain such that all states communicate, and we have picked Γ as the set of all one-edge paths, this statement simplifies to the following: $|\gamma_{f_i,f_j}|_Q = (p^*_{f_i} \times q_{f_i,f_j})^{-1}$. Now we will define the value of κ in the following way. The value of κ can later help us bound the second largest eigenvalue (not in absolute value sense).

$$\kappa = \kappa(\Gamma) = \max_{e} \sum_{\gamma_{f_i,f_j} : e \in \gamma_{f_i,f_j}} |\gamma_{f_i,f_j}|_Q p^*_{f_i} p^*_{f_j} = \max_{f_i,f_j} \frac{p^*_{f_i} \times p^*_{f_j}}{p^*_{f_i} \times q_{f_i,f_j}}.$$

Let $Z = \sum_{f' \in F} exp(\beta \sum_{s \in S} V_s(f'))$. We can find a bound for κ in the following way:

$$\begin{aligned}
\kappa &= \max_{f_i,f_j} \frac{p^*_{f_i} \times p^*_{f_j}}{p^*_{f_i} \times q_{f_i,f_j}} = \max_{f_i,f_j} \frac{p^*_{f_j}}{q_{f_i,f_j}} \\
&= \max_{f_i,f_j} \frac{exp(\beta \sum_{s\in S} V_s(f_j)) \times \prod_{s\in S} \gamma(1+\exp(\beta V_s(f_i \cup f_j))}{Z \times \exp(\beta \sum_{s\in S} \left[V_s(f_i \cup f_j) - V_s(f_i)\right])} \\
&= \max_{f_i,f_j} \frac{exp(\beta \sum_{s\in S} \left[V_s(f_i \cap f_j))\right] \times \prod_{s\in S} \gamma(1+\exp(\beta V_s(f_i \cup f_j))}{Z} \\
&= \max_{f_i,f_j} \prod_{s\in S} \gamma(1+\exp(\beta V_s(f_i \cup f_j))) \times p^*_{f_i \cap f_j}
\end{aligned}$$

We can solve the maximization, by trying to maximize the first and second term independently. Since the values $\gamma(1+\exp(\beta V_s(f_i \cup f_j)) > 1, \forall s \in S$, we solve the maximization by maximizing the first and second part individually, by choosing $f^*_i = \bar{S}$, i.e. by choosing f^*_i to be a state in F such that all users have the lowest delay on all links that they are using. This will ensure that the first term is as large as possible. Note that by choosing $f^*_i = \bar{S}$ we have not restricted ourselves, since $f^*_i \cap f_j = f_j$, and therefore we can achieve any element in F by choosing f_j. Therefore, we will choose f^*_j such that the first term is maximized. Thus:

$$\kappa = \prod_{s\in S} \gamma(1+\exp(\beta V_s(\bar{S}))) \times \max_{f \in F-\{\bar{S}\}} p^*_f \leq \prod_{s\in S} \gamma(1+\exp(\beta V_s(\bar{S}))) \times \max_{f\in F} p^*_f.$$

The value of κ can now be used to bound the second largest eigenvalue of the probability tranistion matrix. We will denote the largest eigenvalue of the probability transition matrix P as $\beta_0 = 1$, the second largest eigenvalue (not in absolute value sense) by β_1 and the smallest eigenvalue value by $\beta_{min} \geq -1$. Eventually, we are trying to bound the value $\beta_* = \max(\beta_1, |\beta_{min}|)$.

We can now use the inequality $\beta_1 \leq 1 - \frac{1}{\kappa}$ to bound β_1 in the following way:

$$\beta_1 \leq 1 - \frac{1}{\prod_{s\in S} \gamma(1+\exp(\beta V_s(\bar{S}))) \times \max_{f\in F} p^*_f} = 1 - \frac{q_{S,\emptyset}}{\max_{f\in F} p^*_f} \leq 1 - q_{S,\emptyset}.$$

Similar calculations can be done to find a bound for the smallest eigenvalue. Again, based on [6] we define λ_f as the path from state f to itself, where the path has an odd number of edges. Since our Markov chain is irreducible and aperiodic, we are always guaranteed to have paths of this sort. Now let Λ be the set of paths of this form, where there is one path for each state $f \in F$ in the set Λ. Now we define the length of a path as before in the following way: $|\lambda_f|_Q = \sum_{e \in \lambda_f} Q(e)^{-1}$, which simplifies to $|\lambda_f|_Q = (p_f^* \times q_{f,f})^{-1}$. With these definitions, we can now define $\iota = \iota(\Lambda) = \max_e \sum_{e:e \in \lambda_f} |\lambda_f| p_f^* = \max\limits_{f \in F} \dfrac{p_f^*}{p_f^* \times q_{f,f}} = \max\limits_{f \in F} \dfrac{1}{q_{f,f}} = \dfrac{1}{\min\limits_{f \in F} q_{f,f}}$.
Finally, we can use the bound $\beta_{min} \geq -1 + \dfrac{2}{\iota}$, derived in [6], to write

$$\beta_{min} \geq -1 + 2(\min_{f \in F} q_{f,f}).$$

These inequalities can assist us in bounding the mixing time of the Markov chain. Subsequently, bounds on the convergence rate of the Markov chain can further help us in deciding how long to run the Markov chain before updating the parameters, and how close the Markov chain will get to its stationary distribution.

3.3 Third Solution: Stochastic Deceleration

In this section, we find implementable solutions to the third formulation. The solution we develop has two major components. One component involves optimizing the σ_s^f and ρ_s^f parameters for different users $s \in S$ and different configurations $f \in F$. This part of the solution will ensure that the sum of utilities under a certain configuration f is optimal, and the constraints are met deterministically. The second component involves a Markov chain, which we will assume has slower dynamics than the dynamics of the parameters of the users, and will change the delay vector of the system. This is due to the fact that we need the optimal σ and ρ parameters and optimal μ prices in order to find the transition probabilities of the Markov chain.

Since the third formulation of the problem is also complicated, we will solve the following approximation to the third formulation:

$$\begin{aligned} \max_{\sigma\geq 0,\rho\geq 0,p\geq 0} \sum_{f\in F} p_f \sum_{s\in S} \Big[U_s^{\sigma}(\sigma_s^f) + U_s^{\rho}(\rho_s^f) - P_s(d_s^f) \Big] - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \\ \text{subject to the following constraints } \forall f \in F \\ A^f X^f \leq C \\ \sum_{f\in F} p_f = 1 \end{aligned} \tag{3.3}$$

where $\sigma = [\sigma^f : f \in F]$ and $\rho = [\rho^f : f \in F]$, and σ^f and ρ^f are vectors of the parameters for all users under configuration f. Note that the only difference between this optimization problem and that in the third formulation is the fact that the objective function of this new approximation involves an entropy term scaled by $\frac{1}{\beta}$, which increases with increasing randomness in the distribution p. However, since the entropy term is multiplied by $\frac{1}{\beta}$, it can arbitrarily approximate the third formulation as β gets large, i.e. the effect of the entropy term on the solution decreases as β increases. We will later discuss the role of the β parameter in our algorithm implementation, and how it introduces a trade-off to our solution.

3.3.1 Solving for the Optimal Probability Distribution

In order to solve for the optimal probability distribution, first assume that the user parameters converge instantaneously to their optimal values depending on the configuration. The optimal probability distribution will later be used as the stationary distribution of a carefully designed Markov chain. In fact, the reason why the dynamics of the Markov chain are going to be slower than the dynamics of the parameters is that the optimal probability distribution will be a function of the optimal parameters and prices. We will derive algorithms for finding the optimal parameters in the next section. We will denote the optimal parameter values by including an asterisk ($*$) as their superscript. Now, we can write the optimization in (3.3) only over the probability distribution as

$$\max_{p\geq 0} \sum_{f\in F} \left(p_f \sum_{s\in S} \left[U_s^\sigma(\sigma_s^{f*}) + U_s^\rho(\rho_s^{f*}) - P_s(d_s^f) \right] \right) - \frac{1}{\beta} \sum_{f\in F} p_f \log p_f$$

$$s.t.$$

$$\sum_{f\in F} p_f = 1.$$

It can be shown by verifying the KKT conditions that the following distribution is the unique solution to the above optimization:

$$p_f^*(U) = \frac{\exp\left(\beta \sum_{s\in S} \left[U_s^\sigma(\sigma_s^{f*}) + U_s^\rho(\rho_s^{f*}) - P_s(d_s^f) \right]\right)}{\sum_{f'\in F} \exp\left(\beta \sum_{s\in S} \left[U_s^\sigma(\sigma_s^{f'*}) + U_s^\rho(\rho_s^{f'*}) - P_s(d_s^{f'}) \right]\right)} \forall f \in F.$$

How to use this result: Again, one way to use this result is to treat this probability distribution as the stationary distribution of a carefully chosen Markov chain. Moreover, in addition to finding transition probabilities that will give rise to the above stationary distribution, it is also important to choose transition probabilities that will allow a distributed implementation of the Markov chain. We will derive transition probabilities below using the detailed balance equation. We will allow transitions between any two states $f_i, f_j \in F$. Now to come up with transition probabilities, we use the following detailed balance equation, where q_{f_i,f_j} denotes the transition probability of going from state f_i to f_j.

$$\forall f_i, f_j \in F : p^*_{f_i} \times q_{f_i,f_j} = p^*_{f_j} \times q_{f_j,f_i} \Leftrightarrow$$

$$\frac{\exp\left(\beta \sum_{s\in S}\left[U^\sigma_s(\sigma^{f_i*}_s) + U^\rho_s(\rho^{f_i*}_s) - P_s(d^{f_i}_s)\right]\right)}{\sum_{f'\in F}\exp\left(\beta \sum_{s\in S}\left[U^\sigma_s(\sigma^{f'*}_s) + U^\rho_s(\rho^{f'*}_s) - P_s(d^{f'}_s)\right]\right)} \times q_{f_i,f_j} =$$

$$\frac{\exp\left(\beta \sum_{s\in S}\left[U^\sigma_s(\sigma^{f_j*}_s) + U^\rho_s(\rho^{f_j*}_s) - P_s(d^{f_j}_s)\right]\right)}{\sum_{f'\in F}\exp\left(\beta \sum_{s\in S}\left[U^\sigma_s(\sigma^{f'*}_s) + U^\rho_s(\rho^{f'*}_s) - P_s(d^{f'}_s)\right]\right)} \times q_{f_j,f_i}.$$

Therefore let

$$q_{f_i,f_j} = \frac{\prod_{s\in S}\alpha_s}{\exp\left(\beta \sum_{s\in S}\left[U^\sigma_s(\sigma^{f_i*}_s) + U^\rho_s(\rho^{f_i*}_s) - P_s(d^{f_i}_s)\right]\right)(|F|-1)}$$

and similarly

$$q_{f_j,f_i} = \frac{\prod_{s\in S}\alpha_s}{\exp\left(\beta \sum_{s\in S}\left[U^\sigma_s(\sigma^{f_j*}_s) + U^\rho_s(\rho^{f_j*}_s) - P_s(d^{f_j}_s)\right]\right)(|F|-1)},$$

where the α_s parameters are positive constants and $|F|$ is the number of delay states in F.

Note that there are many possible transition probabilities that we can choose. However, many of these transition probabilities could depend on the state we are transitioning to; for example, they could depend on different utilities under the new state (delay configuration). However, such transition probabilities are not useful for our problem since users do not have information about what their utilities are going to be under the new setup. Therefore, we will come up with transition probabilities that are only a function of the information available in the current state, such as the utilities under the current state (delay allocation).

Also note that the above transition probabilities depend on the utilities of all of the users. In the next subsection, we will come up with a distributed implementation of this Markov chain.

3.3.2 A Distributed Implementation of the Markov chain

Note that the transition probability between delay states derived in the last section is still a function of all the utilities of the users in a particular configuration. However, we know that each user only has access to information about his own utility, and the server does not know the utilities of the users. We will rewrite the transition probability in the following way:

$$q_{f_i,f_j} = \frac{\prod_{s\in S}\alpha_s}{\exp\left(\beta\sum_{s\in S}\left[U_s^{\sigma}(\sigma_s^{f_i*}) + U_s^{\rho}(\rho_s^{f_i*}) - P_s(d_s^{f_i})\right]\right)(|F|-1)} =$$

$$\frac{1}{|F|-1}\prod_{s\in S}\alpha_s \exp\left(-\beta\left[U_s^{\sigma}(\sigma_s^{f_i*}) + U_s^{\rho}(\rho_s^{f_i*}) - P_s(d_s^{f_i})\right]\right).$$

In order to be able to implement the Markov chain derived in the last part distributedly, and in discrete time, consider the following algorithm for each user s:

Each user s at any given configuration (delay vector) f will pick to be in either of two states ($S = \{S_1, S_2\}$), with the states being S_1 = No Change and S_2 = Change. Each of these states is like the vote of that user to change the delay allocation or not (more on this later). User s will vote for change with probability $\pi_{S_2} = \pi_{Change} = \alpha_s \exp\big(-\beta[U_s^{\sigma}(\sigma_s^{f*}) + U_s^{\rho}(\rho_s^{f*}) - P_s(d_s^{f})]\big)$ and he will not vote for change with the probability $1 - \pi_{S_1} = \pi_{S_1}$.

Note that the α_s parameters could be used to make sure the values are less than or equal to 1. Moreover, the α_s parameters can affect the speed of the convergence of the Markov chain.

Now assume that the link (the server) changes the delays according to the following rule:

- In each time slot, the server will look at the "vote," i.e. state of of each user.
- If all the users are "voting for change," then the server will uniformly randomly pick one of delay states as the new state.

Given that the server acts as above, then the transition probability between two states f_i and f_j is given as the following:

$$q_{f_i,f_j} = \frac{\prod_{s\in S} \alpha_s}{\exp\left(\beta \sum_{s\in S}\left[U_s^\sigma(\sigma_s^{f_i *}) + U_s^\rho(\rho_s^{f_i *}) - P_s(d_s^{f_i})\right]\right)(|F|-1)}$$

and vice versa, which is the desired transition probability.

As we already mentioned, the parameter α can be chosen to increase the speed of convergence of the Markov chain. We can see this by looking at the transition probabilities, and observing that increasing α in fact increases the transition probabilities, while still ensuring that we have the desired stationary distribution.

3.3.3 Solving for Optimal σ, ρ and μ Parameters

In this section we will derive algorithms to find optimal σ and ρ parameters for every delay allocation in F. We will use primal-dual algorithms to find differential equations that will serve as an update equation for the σ and ρ parameters and the prices.

Assuming the probability distribution is fixed, we can solve for the user parameters and the prices by solving the approximation to the third formulation only with respect to σ, ρ and μ. Since for fixed p_f's, the problem is convex, and Slater's condition holds, then there is no duality gap and we can solve the following problem:

$$\min_{\mu\geq 0} \max_{\sigma\geq 0,\rho\geq 0} \sum_{f\in F} p_f \sum_{s\in S}\left[U_s^\sigma(\sigma_s^f) + U_s^\rho(\rho_s^f) + U_s(d_s^f)\right] - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f$$
$$- \sum_{f\in F}\sum_{i=1}^{K}\sum_{j=1}^{J^i}\left[\mu_j^{i,f}\sum_{s\in S}\left(I_{\{k_s^i(f)\leq j\leq J^i\}}\sigma_s^f + \right.\right.$$
$$\left.\left.\left[(D_j^i - d_s^i(f))I_{\{k_s^i(f)\leq j\leq J^i\}} + I_{\{j=J^i+1 \text{ and } k_s^i(f)\leq J^i\}}\right]\rho_s^f\right)\right],$$

where $\mu_j^{i,f}$s denotes the Lagrange multiplier for the j-th constraint of the i-th link, under the delay vector f.

Now, if the probability distribution is fixed (in practice, we could have the probability distribution converging much slower than the parameters, or fix the probability distribution before solving for a new set of parameters), we can have the following primal-dual algorithm for finding the optimal values of the parameters:

$$
\begin{aligned}
\dot{\mu}_j^{i,f} =& k_j^i(\mu_j^{i,f}) \Bigg[\sum_{s \in S} \Bigg(I_{\{k_s^i(f) \leq j \leq J^i\}} \sigma_s^f + \\
& \big[(D_j^i - d_s^i(f)) I_{\{k_s^i(f) \leq j \leq J^i\}} + I_{\{j = J^i + 1 \text{ and } k_s^i(f) \leq J^i\}}\big] \rho_s^f \Bigg) - C_j^i \Bigg]^+_{\mu_j^{i,f}} \\
\dot{\sigma}_s^f =& h_s^f(\sigma_s^f) \Bigg[U_s^{\sigma\prime}(\sigma_s^f) p_f - \sum_{i=1}^{K} \sum_{j=1}^{J^i} \mu_j^{i,f} I_{\{k_s^i(f) \leq j \leq J^i\}} \Bigg]^+_{\sigma_s^f} \\
\dot{\rho}_s^f =& r_s^f(\rho_s^f) \Bigg[U_s^{\rho\prime}(\rho_s^f) p_f - \sum_{i=1}^{K} \sum_{j=1}^{J^i} \mu_j^{i,f} \big[(D_j^i - d_s^i(f)) I_{\{k_s^i(f) \leq j \leq J^i\}} \\
& + I_{\{j = J^i + 1 \text{ and } k_s^i(f) \leq J^i\}}\big] \Bigg]^+_{\rho_s^f},
\end{aligned}
$$

where k, h and r are continuous and positive functions.

Note that in practice we do not need to keep track of all of the parameters under different configurations. Instead, whenever we transition to a new delay configuration, we can freeze any transitions, remain in that state, and optimize the parameters under that configuration, in order to have optimal utility under that configuration and also to satisfy the constraints. Moreover, note that there is a term p_f needed in the update equations of the σ and ρ parameters. It would be very hard in practice to compute this value; however, we do not really need this value in updating the parameters, because all users have one particular probability multiplied in their update equation, and it is as if we are trying to optimize the sum of utilities under a given setup f all scaled by the probability p_f. The answer to this optimization is the same as the answer to the optimization if the utilities were not scaled by p_f. Therefore, we can simply omit p_f from the above calculations and do not need to compute it. The discrete version of the above update equations used for implementation purposes is as follows:

$$\mu_j^{i,f}(t+1) = \mu_j^{i,f}(t) + k_j^i(\mu_j^{i,f})\Bigg[\sum_{s\in S}\Bigg(I_{\{k_s^i(f)\le j\le J^i\}}\sigma_s^f + \big[(D_j^i - d_s^i(f))I_{\{k_s^i(f)\le j\le J^i\}} + I_{\{j=J^i+1 \text{ and } k_s^i(f)\le J^i\}}\big]\rho_s^f\Bigg) - C_j^i\Bigg]^+_{\mu_j^{i,f}}$$

$$\sigma_s^f(t+1) = \sigma_s^f(t) + h_s^f(\sigma_s^f)\Bigg[U_s^{\sigma\prime}(\sigma_s^f) - \sum_{i=1}^{K}\sum_{j=1}^{J^i}\mu_j^{i,f} I_{\{k_s^i(f)\le j\le J^i\}}\Bigg]^+_{\sigma_s^f}$$

$$\rho_s^f(t+1) = \rho_s^f(t) + r_s^f(\rho_s^f)\Bigg[U_s^{\rho\prime}(\rho_s^f) - \sum_{i=1}^{K}\sum_{j=1}^{J^i}\mu_j^{i,f}\big[(D_j^i - d_s^i(f))I_{\{k_s^i(f)\le j\le J^i\}} + I_{\{j=J^i+1 \text{ and } k_s^i(f)\le J^i\}}\big]\Bigg]^+_{\rho_s^f}.$$

3.3.4 Bounding the Mixing Time of the Markov Chain

In this subsection, we use conductance results from [7] to bound the mixing time of the Markov chain. We can assume that the optimal σ, ρ and μ parameters are available to the Markov chain, and therefore, we can calculate a bound on its mixing time. In practice, we can allow enough time for the σ, ρ and μ parameters to converge close to their optimal values, before we allow a change to the Markov chain.

The paper [7] defines the underlying graph of a Markov chain with transition probability matrix $Q = \{q_{f_i,f_j}\}$ and stationary probabilities $p^*_{f_i}, \forall f_i \in F$ as $G_Q(F, W)$, where F is the set of states and W is the set of weights on the graph. These weights are defined as $w_{f_i,f_j} = p^*_{f_i} q_{f_i,f_j}$. We can calculate the values of these weights for the Markov chain in the third algorithm for the case where transitions exist between any two states $f_i, f_j \in F$ as the following:

$$w_{f_i,f_j} = p^*_{f_i} \times q_{f_i,f_j} = w = \\ = \frac{\prod_{s\in S}\alpha_s}{\sum_{f'\in F}\exp\left(\beta\sum_{s\in S}\left[U_s^\sigma(\sigma_s^{f'*}) + U_s^\rho(\rho_s^{f'*}) - P_s(d_s^{f'})\right]\right)(|F|-1)}$$

Note that the weights of all edges in our graph are the same, and we therefore denote all the edge weights by the symbol w. Subsequently, the paper [7] defines conductance of a subset $A \subset F$ as $\Phi_Q(A) = \dfrac{\sum_{f_i \in A} \sum_{f_j \in F \backslash A} w_{f_i,f_j}}{\sum_{f_i \in A} p^*_{f_i}}$ and the conductance of the Markov chain with probability transition matrix Q as $\Phi_Q = \min\limits_{A \subset F : \sum_{f_i \in A} p^*_{f_i} \leq 1/2} \Phi_Q(A)$, where $F \backslash A$ denotes the set of delay states that are in F and not in A. The conductance of the Markov chain can provide a bound for the mixing time of the chain. Therefore, we provide a bound for the conductance value below:

$$\Phi_Q = \min_{A \subset F : \sum_{f_i \in A} p^*_{f_i} \leq 1/2} \Phi_Q(A) = \min_{A \subset F : \sum_{f_i \in A} p^*_{f_i} \leq 1/2} \frac{w \times |A| \times |F \backslash A|}{\sum_{f_i \in A} p^*_{f_i}}$$

$$\begin{aligned} \Rightarrow \Phi_Q \geq & \frac{\min\limits_{A \subset F : 0 \leq \sum_{f_i \in A} p^*_{f_i} \leq 1/2} w \times |A| \times |F \backslash A|}{\max\limits_{A \subset F : 0 \leq \sum_{f_i \in A} p^*_{f_i} \leq 1/2} \sum_{f_i \in A} p^*_{f_i}} \\ \geq & \frac{\min\limits_{A \subset F : 0 \leq \sum_{f_i \in A} p^*_{f_i} \leq 1/2} w \times |A| \times |F \backslash A|}{1/2} \end{aligned}$$

where $|\cdot|$ denotes the size or number of elements in a set.

In the above minimization, we added a "$0 \leq$" to the constraint, since the denominator could not be zero. Therefore, the minimizing set A^* cannot be the empty set. Moreover, we cannot have $A^* = F$, because in that case we will have $\sum_{f_i \in A^*} p^*_{f_i} = 1$, which will violate the minimization constraint again. Therefore, the minimization becomes: $\min\limits_{1 \leq i \leq |F|} i \times (|F| - i) = |F| - 1$, which can be obtained by picking $A^* = \{f_i\}$ s.t. $p^*_{f_i} \leq 1/2$. We therefore have

$$\begin{aligned} \Phi_Q & \geq 2 \times w \times (|F| - 1) \\ & = \frac{2 \prod_{s \in S} \alpha_s}{\sum_{f' \in F} \exp\left(\beta \sum_{s \in S} \left[U^\sigma_s(\sigma^{f'*}_s) + U^\rho_s(\rho^{f'*}_s) - P_s(d^{f'}_s)\right]\right)}. \end{aligned}$$

If we denote the probability distribution at time $t \geq 0$ by $\vec{x}\,(t)$, and the stationary probability distribution by $\vec{p}^{\,*}$, the paper [7] states that for the case of a strongly aperiodic (i.e. $q_{f_i,f_i} \geq 1/2, \forall f_i \in F$) and time-reversible (i.e. $w_{f_i,f_j} = w_{f_j,f_i}$) Markov chain, the rate at which discrepancy $\vec{e}\,(t) = \vec{x}\,(t) - \vec{p}^{\,*}$

vanishes is bounded as follows:

$$\left\|\vec{e}\ (t)\right\| \leq (1 - {\Phi_Q}^2/2)^t \left\|\vec{e}\ (0)\right\|$$

and the magnitude is given by $\left\|\vec{e}\ (t)\right\| = \sum_f \frac{e_f^2(t)}{p_f^*}$.

We therefore have

$$\left\|\vec{e}\ (t)\right\| \leq (1 - \frac{2(\prod_{s\in S}\alpha_s)^2}{\left[\sum_{f'\in F}\exp\left(\beta\sum_{s\in S}\left[U_s^\sigma(\sigma_s^{f'*}) + U_s^\rho(\rho_s^{f'*}) - P_s(d_s^{f'})\right]\right)\right]^2})^t \left\|\vec{e}\ (0)\right\|$$

First of all, note that the above bound depends on β, and as β is increased, the mixing time of the Markov chain reduces. This introduces an accuracy versus speed trade-off, because the accuracy of our algorithm increases with increasing values of β; however, this comes at a cost of reduced speed. Moreover, note that the speed of the Markov chain will be very slow for large values of $\left[\sum_{f'\in F}\exp\left(\beta\sum_{s\in S}\left[U_s^\sigma(\sigma_s^{f'*}) + U_s^\rho(\rho_s^{f'*}) - P_s(d_s^{f'})\right]\right)\right]^2$. Since this term involves a possible exponential sum of exponential terms, all raised to the power of two, this number is expected to be rather large, and we would expect the Markov chain to have a slow mixing time. Therefore, in order to have implementable versions of this algorithm with faster speed, we will reduce the value of β in our simulations.

CHAPTER 4

EXAMPLE SOLUTIONS

In this chapter, we discuss the different solutions for the one link example with two delays offered. The equations and notation from the previous chapter will be used for this example. We hope that this example can further clarify the solutions, and demonstrate the ease of implementation of the solutions.

4.1 One Link Example: Solving an Approximation to the Second Formulation

In this section, we solve the following approximation to the second formulation for the one link example:

$$\begin{aligned}
\max_{\sigma\geq 0,\rho\geq 0,p\geq 0} & \sum_{s\in S} U_s^{\sigma}(\sigma_s) + U_s^{\rho}(\rho_s) + K_s \sum_{f:s\in f} p_f - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \\
& s.t. \\
& \sum_{f\in F} p_f \sum_{s:s\in f} \sigma_s \leq CD_1 \\
& \sum_{s\in S} \sigma_s + (D_2 - D_1)\sum_{f\in F} p_f \sum_{s:s\in f} \rho_s \leq CD_2 \\
& \sum_{s\in S} \rho_s \leq C \\
& \sum_{f\in F} p_f = 1.
\end{aligned} \tag{4.1}$$

4.1.1 One Link Example: Solving for the Optimal Probability Distribution

In order to derive algorithms to solve the approximation to the second formulation, we first solve for the optimal probability distribution p^* given any set of σ and ρ parameters. Now, we can write the optimization in (4.1) only over the probability distribution as

$$\begin{aligned}
&\max_{p\geq 0} \sum_{s\in S} U_s^\sigma(\sigma_s) + U_s^\rho(\rho_s) + K_s \sum_{f:s\in f} p_f - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \\
&\quad s.t. \\
&\quad \sum_{f\in F} p_f \sum_{s:s\in f} \sigma_s \leq CD_1 \\
&\quad \sum_{s\in S} \sigma_s + (D_2 - D_1)\sum_{f\in F} p_f \sum_{s:s\in f} \rho_s \leq CD_2 \\
&\quad \sum_{s\in S} \rho_s \leq C \text{ (this constraint will be satisfied automatically)} \\
&\quad \sum_{f\in F} p_f = 1.
\end{aligned} \tag{4.2}$$

Since the objective and the constraints are linear in p, and Slater's condition holds, we can use strong duality and solve the problem

$$\begin{aligned}
&\min_{\mu\geq 0}\max_{p\geq 0} \sum_{s\in S}[U_s^\sigma(\sigma_s) + U_s^\rho(\rho_s) + \sum_{f\in F} p_f \sum_{s:s\in f} K_s] - \mu_1 \sum_{f\in F} p_f \sum_{s:s\in f} \sigma_s + \mu_1 CD_1 - \\
&\quad \mu_2 \sum_{s\in S}\sigma_s - \mu_2(D_2 - D_1)\sum_{f\in F} p_f \sum_{s:s\in f}\rho_s + \mu_2 CD_2 - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \\
&\quad s.t. \\
&\quad \sum_{f\in F} p_f = 1,
\end{aligned}$$

where the μ parameters are Lagrange multipliers of the constraints.

We can rewrite the problem in the following way, and just maximize with respect to the probability distribution p:

$$\max_{p \geq 0} \sum_{f \in F} p_f \sum_{s: s \in f} (K_s - \mu_1 \sigma_s - (D_2 - D_1)\mu_2 \rho_s) - \frac{1}{\beta} \sum_{f \in F} p_f \log p_f$$

$$s.t.$$

$$\sum_{f \in F} p_f = 1.$$

We can verify the KKT conditions, and find the following probability distribution as the optimal distribution:

$$p^*_f(K, \sigma, \rho, \mu) = \frac{exp(\beta \sum_{s: s \in f} (K_s - \mu_1 \sigma_s - (D_2 - D_1)\mu_2 \rho_s))}{\sum_{f' \in F} exp(\beta \sum_{s: s \in f'} (K_s - \mu_1 \sigma_s - (D_2 - D_1)\mu_2 \rho_s))} \forall f \in F.$$

For each user s we can define the parameter $V_s = K_s - \mu_1 \sigma_s - (D_2 - D_1)\mu_2 \rho_s$ (Note that V_s here is slightly different than $V_s(f)$ defined in the general notation.) Now the optimal stationary distribution can be written as:

$$p^*_f(V) = \frac{exp(\beta \sum_{s: s \in f} V_s)}{\sum_{f' \in F} exp(\beta \sum_{s: s \in f'} V_s)} \forall f \in F$$

where V is the vector with i-th element equal to V_i for $i \in \{1, ..., N\}$.

How to use this result: Again, as mentioned before, we use the optimal probability distribution as the stationary distribution of a carefully designed Markov chain. We then derive transition probabilities that will enable a distributed implementation of the Markov chain using the detailed balance equation. We allow transitions among any two pairs of states $f_i, f_j \in F$. Let q_{f_i, f_j} denote the transition probability from state f_i to state f_j. We write the detailed balance equations in the following way:

$$p^*_{f_i} \times q_{f_i,f_j} = p^*_{f_j} \times q_{f_j,f_i} \text{ and let } q_{f_i,f_j} = \frac{\exp(\beta \sum\limits_{s:s\in f_j - f_j \cap f_i} V_s)}{\prod\limits_{s:s\in f_i \cup f_j} \gamma(1+\exp(\beta V_s))} \Leftrightarrow$$

$$p^*_{f_i} \times q_{f_i,f_j} = \frac{exp(\beta \sum\limits_{s:s\in f_i} V_s)}{\sum\limits_{f'\in F} exp(\beta \sum\limits_{s:s\in f'} V_s)} \times \frac{\exp(\beta \sum\limits_{s:s\in f_j - f_j\cap f_i} V_s)}{\prod\limits_{s:s\in f_i\cup f_j} \gamma(1+\exp(\beta V_s))} =$$

$$\frac{exp(\beta \sum\limits_{s:s\in f_i \cup f_j} V_s)}{[\sum\limits_{f'\in F} exp(\beta \sum\limits_{s:s\in f'} V_s)] \prod\limits_{s:s\in f_i\cup f_j} \gamma(1+\exp(\beta V_s))} = p^*_{f_j} \times q_{f_j,f_i}.$$

4.1.2 A Distributed Implementation of the Markov Chain

In order to have a distributed implementation of this algorithm, let every user $s \in \{1, ..., N\}$ change delays according to the following algorithm:

Delay Changing Algorithm

- If user s has delay D_1 he will change to D_2 with probability $p^s_{D_1,D_2} = \frac{1}{\gamma(1+\exp(\beta V_s))}$ and he will not change with probability $1 - p^s_{D_1,D_2}$.
- If user s has delay D_2 he will change to D_1 with probability $p^s_{D_2,D_1} = \frac{\exp(\beta V_s)}{\gamma(1+\exp(\beta V_s))}$ and he will not change with probability $1 - p^s_{D_2,D_1}$.

It is easy to verify that if all users act according to the above algorithm, we will have the mentioned transition probabilities between different states in F. Note that these changes can be done in a completely distributed manner, since each user s knows the value of V_s since it only depends on the prices (which could be shared with the users by the netwrok) and parameters of the user who is changing his delay. Therefore, the Markov chain can be implemented in a completely distributed manner.

4.1.3 One Link Example: Solving for Optimal σ, ρ and μ Parameters

Assuming the optimal probability distribution is known, we can solve for the user parameters and prices by solving the approximation to the second formulation only with respect to σ, ρ and μ. Since for given p_f^*'s, the problem is convex, and Slater's condition holds, then there is no duality gap and we can solve the following problem:

$$\min_{\mu \geq 0} \max_{\sigma \geq 0, \rho \geq 0} \sum_{s \in S} [U_s^\sigma(\sigma_s) + U_s^\rho(\rho_s) + K_s \sum_{f : s \in f} p_f^*] - \mu_1 \sum_{f \in F} p_f^* \sum_{s : s \in f} \sigma_s + \mu_1 C D_1 -$$
$$\mu_2 \sum_{s \in S} \sigma_s - \mu_2 (D_2 - D_1) \sum_{f \in F} p_f^* \sum_{s : s \in f} \rho_s + \mu_2 C D_2 + \mu_3 C - \mu_3 \sum_{s \in S} \rho_s.$$

We can write the following primal-dual algorithm for finding the optimal values of the parameters and the prices. Note that in the equations below, we have replaced some of the summations with indicator functions. This change will later help us in deriving simpler forms of the update equations.

$$\dot{\mu_1} = k_1 \left[\sum_{f \in F} \sum_{s \in S} p_f^* \sigma_s I_{\{s \in f\}} - C D_1 \right]_{\mu_1}^+$$
$$\dot{\mu_2} = k_2 \left[\sum_{s \in S} \sigma_s + (D_2 - D_1) \sum_{f \in F} \sum_{s \in S} p_f^* \rho_s I_{\{s \in f\}} - C D_2 \right]_{\mu_2}^+$$
$$\dot{\mu_3} = k_3 \left[\sum_{s \in S} \rho_s - C \right]_{\mu_3}^+$$
$$\dot{\sigma_s} = h_s \left[U_s'(\sigma_s) - \sum_{f \in F} p_f^* \mu_1 I_{\{s \in f\}} - \mu_2 \right]_{\sigma_s}^+$$
$$\dot{\rho_s} = r_s \left[U_s'(\rho_s) - \sum_{f \in F} p_f^* \mu_2 (D_2 - D_1) I_{\{s \in f\}} - \mu_3 \right]_{\rho_s}^+$$

Note that in the above update equations, the network and each of the nodes need to know the p_f^* values, which could be hard to obtain. However, we can actually rewrite the update equations in the following way, to solve this problem, and actually use this fact to our advantage (as explained below).

$$\dot{\mu}_1 = k_1\left(\sum_{f\in F} p_f^*\left[\sum_{s\in S}\sigma_s I_{\{s\in f\}} - CD_1\right]\right)^+_{\mu_1}$$

$$\dot{\mu}_2 = k_2\left(\sum_{f\in F} p_f^*\left[\sum_{s\in S}\sigma_s + \sum_{s\in S}\left[(D_2 - D_1)\rho_s I_{\{s\in f\}}\right] - CD_2\right]\right)^+_{\mu_2}$$

$$\dot{\mu}_3 = k_3\left[\sum_{s\in S}\rho_s - C\right]^+_{\mu_3}$$

$$\dot{\sigma}_s = h_s\left(\sum_{f\in F} p_f^*\left[U_s'(\sigma_s) - \mu_1 I_{\{s\in f\}} - \mu_2\right]\right)^+_{\sigma_s}$$

$$\dot{\rho}_s = r_s\left(\sum_{f\in F} p_f^*\left[U_s'(\rho_s) - \mu_2(D_2 - D_1) I_{\{s\in f\}} - \mu_3\right]\right)^+_{\rho_s}$$

Now note that by writing the update equations in the above form we can derive an implementable algorithm that will not require knowledge of the values of p_f^* as described in section 3.2.3.

4.2 One Link Example: Solving an Approximation to the Third Formulation

In this section, we will solve the following approximation to the third formulation for the one link example:

$$
\begin{aligned}
\max_{\sigma\geq 0,\rho\geq 0,p\geq 0} \sum_{f\in F} p_f \sum_{s\in S} &\left[U_s^\sigma(\sigma_s^f) + U_s^\rho(\rho_s^f) + U_s(d_s^f)\right] - \frac{1}{\beta}\sum_{f\in F} p_f \log p_f \\
&\text{subject to the following constraints } \forall f \in F \\
&\sum_{s:s\in f} \sigma_s^f \leq CD_1 \\
&\sum_{s\in S} \sigma_s^f + (D_2 - D_1)\sum_{s:s\in f} \rho_s^f \leq CD_2 \\
&\sum_{s\in S} \rho_s^f \leq C \\
&\sum_{f\in F} p_f = 1
\end{aligned}
\tag{4.3}
$$

where $\sigma = [\sigma^f : f \in F]$ and $\rho = [\rho^f : f \in F]$, and σ^f and ρ^f are vectors of the parameters for all users under configuration f.

4.2.1 One Link Example: Solving for the Optimal Probability Distribution

The optimal probability distribution for the one link example, and the distributed implementation of the Markov chain, is exactly the same as the one described in the last chapter, and we will therefore not discuss it further in this section.

4.2.2 One Link Example: Solving for Optimal σ and ρ Parameters

Assuming the probability distribution is fixed, we can solve for the user parameters by solving the approximation to the third formulation only with respect to σ and ρ. Since for fixed p_f's, the problem is convex, and Slater's condition holds, then there is no duality gap and we can solve the following problem:

$$\begin{aligned}\min_{\mu\geq 0}\max_{\sigma\geq 0,\rho\geq 0}\sum_{f\in F}p_f\sum_{s\in S}\Big[U_s^\sigma(\sigma_s^f)+U_s^\rho(\rho_s^f)+U_s(d_s^f)\Big]-\frac{1}{\beta}\sum_{f\in F}p_f\log p_f\\ -\sum_{f\in F}\mu_1^f\Big(\sum_{s:s\in f}\sigma_s^f-CD_1\Big)\\ -\sum_{f\in F}\mu_2^f\Big(\sum_{s\in S}\sigma_s^f+(D_2-D_1)\sum_{s:s\in f}\rho_s^f-CD_2\Big)\\ -\sum_{f\in F}\mu_3^f\Big(\sum_{s\in S}\rho_s^f-C\Big)\end{aligned}$$

where the μ_i^fs are the Lagrange multipliers for the i-th constraint under the delay vector f.

Now if the probability distribution is fixed (in practice, we could have the probability distribution converging much slower than the parameters, or fix the probability distribution before solving for a new set of parameters), we can have the following primal-dual algorithm for finding the optimal values of the parameters:

$$\begin{aligned}\dot{\mu}_1^f&=k_1^f\Big[\sum_{s:s\in f}\sigma_s^f-CD_1\Big]^+_{\mu_1^f}\\ \dot{\mu}_2^f&=k_2^f\Big[\sum_{s\in S}\sigma_s^f+(D_2-D_1)\sum_{s:s\in f}\rho_s^f-CD_2\Big]^+_{\mu_2^f}\\ \dot{\mu}_3^f&=k_3^f\Big[\sum_{s\in S}\rho_s^f-C\Big]^+_{\mu_3^f}\\ \dot{\sigma}_s^f&=h_s^f\Big[U_s'(\sigma_s^f)p_f-\sum_{f:s\in f}\mu_1^f-\sum_{f\in F}\mu_2^f\Big]^+_{\sigma_s^f}\\ \dot{\rho}_s^f&=r_s^f\Big[U_s'(\rho_s^f)p_f-(D_2-D_1)\sum_{f:s\in f}\mu_2^f-\sum_{f\in F}\mu_3^f\Big]^+_{\rho_s^f}.\end{aligned}$$

CHAPTER 5

NUMERICAL RESULTS

5.1 Numerical Experiment

Setup: In this numerical experiment, three links are shared among 60 users. Each user $s \in \{1, ..., 60\}$ has a utility of the form $U_s(\sigma_s, \rho_s, d_s) = s\log(\sigma_s) + s\log(\rho_s) - 0.007(d_s^1 + d_s^2 + d_s^3)^2$. We have assigned different weights to different users in order to differentiate them and also to be able to analyze the effect of having different utility functions on the user parameters.

As seen in Fig 5.1, in this setup, the three links being shared all have the same capacity equal to 9. Moreover, all three links offer the same 10 and 40 ms delays. We have used the same capacity for all links in order to be able to analyze the effect of the loads on the links. Therefore, in this experiment, we have one link (link 2) that is the busiest in terms of the number of users it serves, another link (link 1) that serves less users, and the last link (link 3) that serves the least number of users.

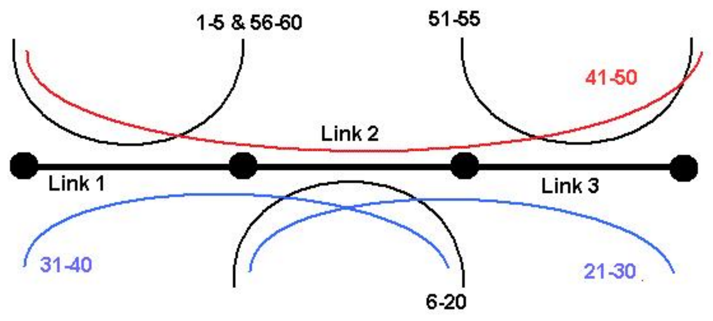

Figure 5.1: Displaying the Links and the Users on Each of Them

Table 5.1 lists the original delay allocation in this experiment. In the tables in this chapter, a value of 1 indicates that the user was allocated delay 1 on the corresponding link. Moreover, a dash (-) indicates that the user was not using the corresponding link.

Table 5.1: Original Allocated Delays on Each Link.

Users 1-10										
Link 1	2	2	2	2	2	-	-	-	-	-
Link 2	-	-	-	-	-	2	2	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 11-20										
Link 1	-	-	-	-	-	-	-	-	-	-
Link 2	2	2	2	2	2	2	2	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 21-30										
Link 1	-	-	-	-	-	-	-	-	-	-
Link 2	1	1	1	1	1	1	1	1	1	1
Link 3	2	2	2	2	2	2	2	2	2	2
Users 31-40										
Link 1	2	2	2	2	2	2	2	2	2	2
Link 2	1	1	1	1	1	1	1	1	1	1
Link 3	-	-	-	-	-	-	-	-	-	-
Users 41-50										
Link 1	1	1	1	1	1	1	1	1	1	1
Link 2	2	2	2	2	2	2	2	2	2	2
Link 3	1	1	1	1	1	2	2	2	2	2
Users 51-60										
Link 1	-	-	-	-	-	1	1	1	1	1
Link 2	-	-	-	-	-	-	-	-	-	-
Link 3	1	1	1	1	1	-	-	-	-	-

Implementation: We implemented the three algorithms in this numerical experiment. We implemented the first algorithm according to the implementation instructions in Section 3.1. We used the discrete update equations in Section 3.3.3, with $h_s^f(\sigma_s^f) = 0.00005\sigma_s^f$ and $r_s^f(\rho_s^f) = 0.000001\rho_s^f, \forall s, f$, and $k_1^{i,f}(\mu_1^{i,f}) = 0.0002, \forall i, f$ and $k_j^{i,f}(\mu_j^{i,f}) = 0.001, \forall i, f$ and for $j = 2, 3$. We initialized all parameters to zero, and allowed 5500 time slots with no delay changes for the algorithm to stabilize. In our simulation, the network picks one user randomly once every 10 time slots and allows that user to change its delay values. The 10 time slot period allows time for stabilization of the different parameters and prices in the network; however, it was chosen arbitrarily based on experiments. The number of users chosen and the wait time before choosing the next user(s) should be adjusted according to the size and structure of each network.

The implementation of the second algorithm involved two parts. We implemented a discrete time Markov chain according to Section 3.2.2. For the Markov chain, we picked $\beta = 1.5$ and $\gamma = 3000$. The second part of the implementation involved update equations for the σ, ρ and μ parameters. We used the discrete update equations in Section 3.2.3, with $h_s(\sigma_s) = 0.00005\sigma_s$ and $r_s(\rho_s) = 0.000001\rho_s, \forall s \in S$, and $k_1^i(\mu_1^i) = 0.0002, \forall i$ and $k_j^i(\mu_j^i) = 0.001, \forall i$ and for $j = 2, 3$. We initialized all parameters to zero, and allowed 5500 time slots with no delay changes for the algorithm to stabilize. In this implementation, users can independently decide to change their delays at any time slot, and the network does not pick certain users to change their delays.

The implementation of the third algorithm, similarly to the second one, involved two parts. One part of the algorithm involved update equations for the σ, ρ and μ parameters. We used the discrete update equations in Section 3.3.3, with $h_s^f(\sigma_s^f) = 0.00005\sigma_s^f$ and $r_s^f(\rho_s^f) = 0.000001\rho_s^f, \forall s \in S$, and $k_1^{i,f}(\mu_1^{i,f}) = 0.0002, \forall i$ and $k_j^{i,f}(\mu_j^{i,f}) = 0.001, \forall i, f$ and for $j = 2, 3$. We initialized all parameters to zero, and allowed 5500 time slots with no delay changes for the algorithm to stabilize. Two delay changes according to the Markov chain were allowed after the intial phase, once every 10 time slots. Note that we purposely allowed only two delay changes, as opposed to possibly all delays changing, in order to have a Markov chain that changes slower and in order to have parameters converge within the next 10 time slots after every change. We implemented a discrete time Markov chain according to Section 3.3.2. For the Markov chain, we picked $\beta = 0.005$, to have a Markov chain with a reasonable mixing time. Also, we picked α_s according to the assumption that the minimum utility value for each user is going to be 20 percent less than the utility value the user converged to after the initial phase. With this assumption, we chose α_s to have a vote for change with probability one when the utility of the user equals its minimum utility.

Expected Properties of Solutions: Different users in this experiment use different numbers of links with different initial delay assignments. A group of users, users 6 to 20, only use link 2. We can therefore anticipate these users to pick the higher delay, because for these users, $d_s^1 = 0$ and $d_s^3 = 0$, and because the penalty function in the users' utility function gets steeper as the sum of delays increases. Similarly, users 1 to 5 and users 56 to 60 also only use link 1. While we anticipate these two groups also to have a large delay (because of the last reason), we anticipate users 56 to 60 to be more inclined to have a larger delay because they have large weights on their σ and ρ parameters, and the cost of transferring these users to the smaller delays is more expensive than for users with smaller weights on their σ and ρ parameters.

Users 21 to 30 use two links, namely link 2 and link 3. We expect these users to have their smaller delay (if they do have one) on the less busy link (link 3) more often than on the busier link (link 2). Similarly, users 31 to 40 will use both links 1 and 2, and we again expect them to have their lower delay more often on link 1 than on link 2. Lastly, users 41 to 50 will use all three links.

Observed Properties of Solutions: Figures 5.2, 5.3 and 5.4 show the evolution of the sum of the utilities of the users versus the number of time slots. Moreover, Tables 5.2, 5.3 and 5.4 represent the final delay state of the system using the different solutions.

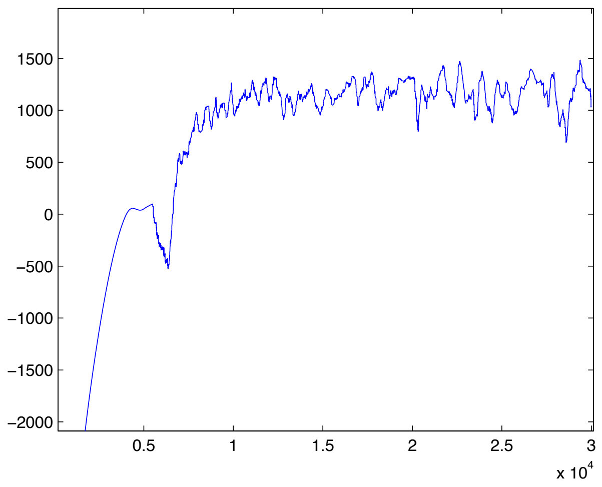

Figure 5.2: Sum of Utilities vs. Number of Time Slots Using Direct Utility Comparison.

The first and second algorithms resulted in similar outcomes. Both algorithms resulted in a significant increase in the aggregate utility of the system. Moreover, the second algorithm, after enough time to stabilize, convereged to one or a few delay states that were not very different. Therefore, we did not see a significant violation in the original constraints of the problem. Namely, although the second algorithm was solving a relaxed version of the original problem with constraints met on average, in practice the algorithm

Table 5.2: Final Delay Allocation Using First Solution.

Users 1-10										
Link 1	2	1	1	2	1	-	-	-	-	-
Link 2	-	-	-	-	-	1	1	1	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 11-20										
Link 1	-	-	-	-	-	-	-	-	-	-
Link 2	2	1	2	2	2	2	2	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 21-30										
Link 1	-	-	-	-	-	-	-	-	-	-
Link 2	2	1	2	1	2	2	2	2	1	2
Link 3	1	1	2	1	1	1	1	1	1	2
Users 31-40										
Link 1	2	2	2	2	1	2	1	1	2	2
Link 2	2	1	2	2	2	2	2	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 41-50										
Link 1	1	1	2	1	2	1	1	2	1	2
Link 2	1	1	1	1	1	2	1	2	2	1
Link 3	1	1	1	1	1	1	1	2	1	1
Users 51-60										
Link 1	-	-	-	-	-	2	2	2	2	2
Link 2	-	-	-	-	-	-	-	-	-	-
Link 3	2	2	2	2	2	-	-	-	-	-

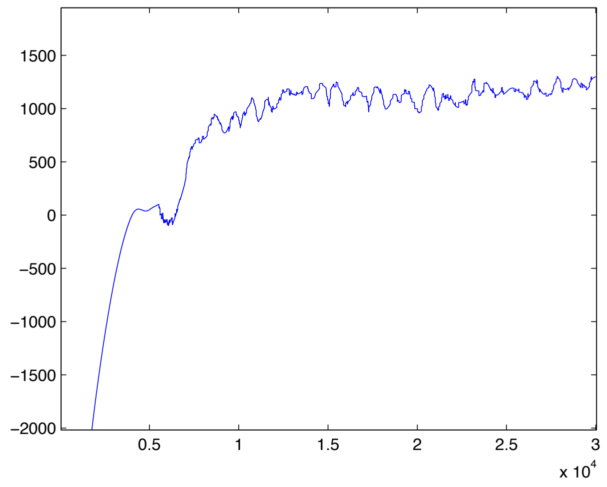

Figure 5.3: Sum of Utilities vs. Number of Time Slots Using Payment Comparison.

Table 5.3: Final Delay Allocation Using Second Solution.

Users 1-10										
Link 1	1	1	1	1	2	-	-	-	-	-
Link 2	-	-	-	-	-	1	2	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 11-20										
Link 1	-	-	-	-	-	-	-	-	-	-
Link 2	2	2	2	2	2	1	2	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 21-30										
Link 1	-	-	-	-	-	-	-	-	-	-
Link 2	1	1	1	2	1	1	2	2	1	1
Link 3	2	2	2	1	2	2	2	2	2	2
Users 31-40										
Link 1	2	1	1	2	1	1	2	2	1	1
Link 2	2	1	2	2	2	1	1	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 41-50										
Link 1	2	1	1	1	1	1	2	2	1	1
Link 2	1	2	1	1	1	2	1	1	2	2
Link 3	1	1	2	2	2	2	1	1	1	1
Users 51-60										
Link 1	-	-	-	-	-	2	2	2	2	2
Link 2	-	-	-	-	-	-	-	-	-	-
Link 3	2	2	2	2	2	-	-	-	-	-

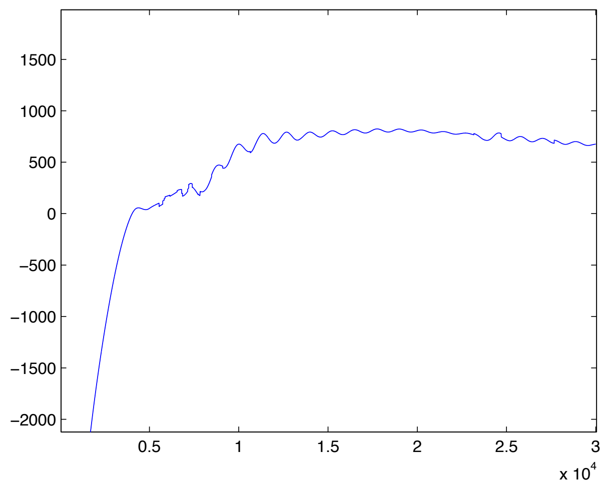

Figure 5.4: Sum of Utilities vs. Number of Time Slots Using Stochastic Deceleration.

Table 5.4: Final Delay Allocation Using Third Solution.

Users 1-10										
Link 1	2	2	1	2	2	-	-	-	-	-
Link 2	-	-	-	-	-	2	2	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 11-20										
Link 1	-	-	-	-	-	-	-	-	-	-
Link 2	2	2	1	2	2	2	2	2	2	2
Link 3	-	-	-	-	-	-	-	-	-	-
Users 21-30										
Link 1	-	-	-	-	-	-	-	-	-	-
Link 2	2	1	1	1	2	2	1	1	1	1
Link 3	2	2	2	1	2	2	2	1	2	2
Users 31-40										
Link 1	2	2	2	2	1	2	2	2	2	2
Link 2	1	1	2	1	1	1	1	1	1	1
Link 3	-	-	-	-	-	-	-	-	-	-
Users 41-50										
Link 1	1	1	1	1	1	1	1	1	1	1
Link 2	2	2	2	2	2	2	1	1	1	2
Link 3	1	1	1	1	1	2	2	2	2	2
Users 51-60										
Link 1	-	-	-	-	-	1	1	1	2	1
Link 2	-	-	-	-	-	-	-	-	-	-
Link 3	1	2	1	1	1	-	-	-	-	-

convereged to states that were not significantly different, subsequently, causing the constraints to be met deterministically.

As we expected, many of the users 6-20 picked the higher delay because they were only using link 2 and therefore still had a small end-to-end delay. Also, most of the users 1-5 changed their delay to the lower delay because they had a smaller utility of σ and ρ and therefore paid a small price for their parameters by picking a small delay. However, users 56-60, who had a similar situation (only using link 1), picked the higher delay because by doing that they had large utilities of σ and ρ and could get lower prices for these parameters with the higher delay. Users 51-55 made a similar change for the same reason. In both cases, many of the users 21-30 and users 31-40 tend to have their lower delay on the less busy link (link 3 and link 1) as we expected. Finally, users 41-50 changed most of their delays to the lower delay, since they have the same delay penalty function as the other users; however, because of using all three links, they started with large end-to-end delays. Again, most of the lower delays tend to be on the less busy links (this is more evident in the first simulation).

We can look at these changes from the network's perspective as well. For example, the network tends to assign users with bigger σ and ρ parameters a larger delay. For example, the network assigned users 51-55 or 56-60 a large delay since these users were more inclined to have large σ and ρ parameters; and by allocating them the larger delay, the network would save more of its resources.

The third algorithm carried out some of the expected delay changes, which resulted in an increased aggregate utility; however, the number of useful delay changes done by this algorithm were less than the number of useful delay changes done by the first and second algorithm, therefore causing the third solution to have a lower final aggregate utility.

Discussion of Algorithm Performance: The first and second algorithms had similar performances in this experiment. The first algorithm is rather simple and performs well; however, implementing the first algorithm requires the constant supervision of the network over the users, monitoring which users change their delays and at what times. The number of users who can change their delays, and how often delays are changed depends on the network, and must be determined empirically. Also, while this method is distributed, it still requires every probed user to do $\prod_{i \in L} J^i$ utility calculations, because every probed user has to calculate its utility for all delay combinations and pick the best allocation, which is a combinatorial problem. This utility calculation could be complex if the number of links and delays used by the user are large.

The second algorithm also had similar performance. The parameters that might have to be set empirically in this experiment are the β and γ parameters, relating to the Markov chain. This algorithm does not require a constant supervision of the network over the users, because any user can choose to change any of its delays at any time according to its Markov chain implementation. Also, every user has to compute $\sum_{i \in L}(J^i - 1)$ probabilities.

The third algorithm didn't perform as well as the first two algorithms, as we can clearly see from the figures. This is mainly because the third algorithm has a slow convergence rate, and we therefore had to choose a small value of 0.005 for β, which in turn caused the resulting solution to be less accurate. Also, in this algorithm, the network has to collect the votes of the users, and subsequently make a decision about which delays to change next.

We have summarized some of the important properties of the three implemented algorithms in Table 5.5.

Table 5.5: Algorithm Properties.

Algorithm Properties	**First Algorithm**	**Second Algorithm**	**Third Algorithm**
Randomness	Selection of user to probe	Delay vector update	Delay vector update
Update Criteria	User utility at new delay	Lower cost given current σ and ρ	Value of current state
Is the change in σ and ρ considered in decision?	Yes	No	No
Number of σ, ρ and μ updates per possible delay updates	10	1	10
Number of possible delays changed at once	All delays of one user	Unlimited	Two delays
Markov chain parameters	Not applicable	$\beta = 1.5$, $\gamma = 3000$	$\beta = 0.005$, α_s depends on min utility
Complexity	Probed user makes $\prod_{i \in L} J^i$ utility calculations	User makes $\sum_{i \in L}(J^i - 1)$ probability calculations	User makes one probability calculation

CHAPTER 6

CONCLUSION

6.1 Conclusion and Future Work

The problem of allocating throughput and delays to users in a network is an important and very challenging problem [2]. The authors in [1] have developed a theoretical framework for allocating rates in an arbitrary network. As suggested in [3], we tried to extend this framework for optimization over rate, burstiness and end-to-end delay. We developed a provably optimal algorithm in Section 3.3 as the solution to the third formulation of the problem for multihop networks. This solution, however, seems to be slow, which motivated us to introduce two new solutions. Both the first and second algorithm have been simulated, and show very promising results and fast convergence times. Of these two algorithms, the second algorithm seems to be easier to implement, since it does not require the network to constantly choose users to change delays; rather, every user can change its delays according to its Markov chain.

Future directions of this research include analyzing the first algorithm more closely, and providing some bounds on the performance of this algorithm. Similarly, the second algorithm could be analyzed futher to see rigorously how far it is from the optimal solution. Lastly, further research can be done on the third algorithm to improve the speed of convergence of the Markov chain.

REFERENCES

[1] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1999.

[2] R. Guerin and V. Peris, "Quality-of-service in packet networks: Basic mechanisms and directions," *Computer Networks*, vol. 31, no. 3, pp. 169–179, February 1999.

[3] B. Hajek and S. Yang, "A mechanism for pricing service guarantees," *IEEE Information Theory Workshop*, June 2009.

[4] R. J. Gibbens and F. P. Kelly, "On packet marking at priority queues," *IEEE Transactions on Automatic Control*, vol. 47, pp. 1016–1020, June 2002.

[5] M. Chen, S. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," in *Proceedings of Infocom Conference*, March 2010.

[6] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of markov chains," *The Annals of Applied Probability*, vol. 1, no. 1, pp. 36–63, February 1991.

[7] M. Mihali, "Conductance and convergence of markov chains - a combinatorial treatment of expanders," *FOCS*, pp. 526–531, 1989.