MULTITRACK: A DELAY AND COST AWARE P2P OVERLAY ARCHITECTURE

A Thesis

by

VINITH KUMAR REDDY PODDUTURI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2009

Major Subject: Computer Engineering

MULTITRACK: A DELAY AND COST AWARE P2P OVERLAY ARCHITECTURE

A Thesis

by

VINITH KUMAR REDDY PODDUTURI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2009

Major Subject: Computer Engineering

ABSTRACT

MultiTrack: A Delay and Cost Aware P2P Overlay Architecture. (August 2009)

Vinith Kumar Reddy Podduturi, B.E.;M.Sc., Birla Institute of Technology and Science, Pilani,India

Co–Chairs of Advisory Committee: Dr. Srinivas Shakkottai
Dr. Alex Sprintson

The rapid growth of peer-to-peer (P2P) networks in the past few years has brought with it increases in transit cost to Internet Service Providers (ISPs), as peers exchange large amounts of traffic across ISP boundaries. This ISP oblivious behavior has resulted in misalignment of incentives between P2P networks—that seek to maximize user quality—and ISPs—that would seek to minimize costs. Can we design a P2P overlay that accounts for both ISP costs as well as quality of service, and attains a desired tradeoff between the two? We design a system, which we call MultiTrack, that consists of an overlay of multiple kinds of Trackers whose purpose it is to align these goals. We have mTrackers that form an overlay network among themselves, and split demand from users among different ISP domains while trying to minimize their individual costs (delay plus transit cost) in their ISP domain. We design the signals in this overlay of mTrackers in such a way that potentially competitive individual optimization goals are aligned across the mTrackers. The system could also have a tTracker that acts as a gateway into the system, and ensures that users who are from different ISP domains have a fair chance of being admitted into the system, while keeping costs in check.

We prove analytically that our system is stable and achieves maximum utility with minimum cost. We validated our system design using Matlab simulations, and implemented the system on ns-2 in order to conduct more realistic experiments. We showed that our system significantly outperforms two types of systems, one in which user delay is the only control dimension (forwarding traffic without considering the transit prices) and a second system in which transit prices are the only control dimension (localized traffic only). Thus, we conclude that our system, that operates in two dimensions: (1) user delay and (2) transit prices, results in minimum cost and maximum utility for fixed capacity of the system.

To God and my Parents

ACKNOWLEDGMENTS

I thank my advisor, Dr. Srinivas Shakkottai, for choosing me to work on this research project. As a mentor and critic, he guided me in the right direction throughout the course of this research. His constructive feedback and motivation helped me to challenge myself and realize my true potential, both technically and non-technically as well. I sincerely thank him for helping me out in many ways, through the informative discussions we had or the guidance he provided in choosing my courses, or even the recommendations he provided for getting assistantships in the department. "Thank you very much, Dr. Shakkottai, you made this journey look easy". I thank Younghoon Kim, who is a Phd. student in the Department of Computer Science at KAIST, South Korea, for the invaluable help he provided in the form of implementing our system in ns-2 simulator. I would like to thank Dr. A.L. Narasimha Reddy for his collaboration on this project. His valuable suggestions during the course of this project helped us a lot. I would like to express my sincere gratitude to my co-advisor Dr. Alex Sprintson; he helped me pick the right courses. I thank him for recommending me for department assistantships, which helped me focus on my research. I thank Dr. Gautam Natarajan for agreeing to be on my committee. I would like to extend my gratitude to the professors and staff in computer cngineering group of the ECE department at Texas A&M University for guiding me and helping me in. Finally, I thank my parents - who have always supported and encouraged me to pursue my master's. Last, but not the least, I thank the Almighty for giving me strength to overcome my limitations all the time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The past few years have seen the rapid growth of content distribution over the Internet, particularly using peer-to-peer (P2P) networks. Recent studies estimate that 35-90% of bandwidth is consumed by P2P file-sharing applications, both at the edges and even within the core [1–3]. The use of P2P networks for media delivery is expected to grow still further, with the proliferation of legal applications (*e.g.* Pando Networks [4]) that use P2P as a core technology.

While most P2P systems today possess some form of network resource-awareness, and attempt to optimally utilize the system resources, they are largely agnostic to Internet Service Providers' (ISP) concerns such as traffic management and costs. This ISP-oblivious nature of P2P networks has hampered the ability of system participants to correctly align incentives. Indeed, the recent conflicts between ISPs and content providers, as well as efforts by some ISPs such as Comcast to limit P2P traffic on their networks [5], speak in part to an inability to align interests correctly. Such conflicts are particularly critical as P2P becomes an increasingly prevalent form of content distribution [6].

A traditional BitTorrent system [7] has elements called *Trackers* whose main purpose is to enable peers to find each other. The BitTorrent Tracker randomly assigns a new (entering) user a set of peers that are already in the system to communicate with. This system has the disadvantage that if peers who are assigned to help each other are in the domains of different ISPs, they would cause significant transit costs to the ISPs due to the inter-ISP traffic that they generate. However, if costs are

The journal model is *IEEE Transactions on Automatic Control.*

reduced by forcing traffic to be local, then the delay performance of the system could be impacted. Recent work such as [8–10] has focused on cost in terms of load balancing and localizing traffic, and developed heuristics to attain a certain quality of service (QoS). For example, P4P [9] develops a framework to achieve minimum cost (optimal load balancing) among ISP links, but its BitTorrent implementation utilizes the heuristic that 30% of peers declared to each requesting user should be drawn from "far away ISPs" in order to attain a good QoS.

This leads us to the fundamental question that we attempt to answer in this paper: *Can we develop a distributed delay and cost optimal P2P architecture?*. In this thesis we focus on developing a provably optimal price-assisted architecture called MultiTrack, that would be aware of the interaction between delay and cost. The idea is to understand that while the resources available with peers in different ISP domains should certainly be used, such usage comes at a price. The system must be able to determine the marginal gain in performance for a marginal increase in cost. It would then be able to locate the optimal point at which to operate.

The conceptual system is illustrated in Figure 1, and consists of the following elements. The system is managed by a set of *mTrackers*. Each mTracker is associated with a particular ISP domain. The mTrackers are similar to the Trackers in BitTorrent [7,11], in that their main purpose is to enable peers to find each other. However, unlike BitTorrent, the mTrackers in MultiTrack form an overlay network among themselves. The purpose of the overlay network is to provide multi-dimensional actions to the mTrackers. In Figure 1, mTracker 1 is in steady state (wherein the demand on the mTracker is less than the available capacity [12]), which implies that it has spare capacity to serve requests from other mTrackers. Consider mTracker 3. When a request arrives, it can either assign the requester to its own domain at essentially zero cost, or can forward the user to mTracker 1 and incur a cost for doing so. However,

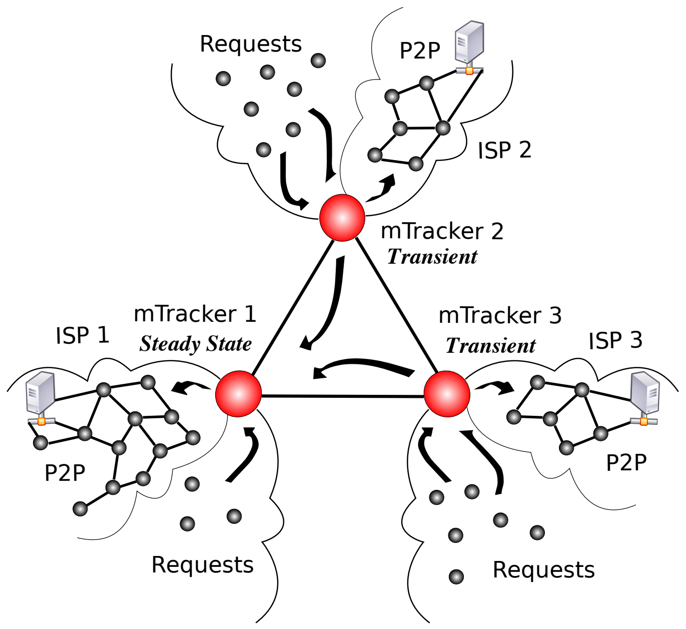

Fig. 1. The MultiTrack architecture. Multiple trackers, each following individual optimizations, achieve an optimal delay-cost tradeoff.

the delay incurred by forwarded users would be less as mTracker 1 has high capacity. Thus, mTracker 3 can trade-off cost versus delay performance by forwarding some part of its demand.

Each mTracker uses *price assisted decision making* by utilizing dynamics that consider the marginal payoff of forwarding traffic to that of retaining traffic in the same domain as the mTracker. Several such rational dynamics have been developed in the field of *game theory* that studies the behavior of selfish users[1]; background on population games and evolutionary dynamics , which we used to model our system, is presented in chapter II. Since our system is based on BitTorrent architecture we dedicate a chapter III to describe about some basic BitTorrent components and terminology.

[1]A good reference on game theory is [13].

We present our system model with its attendant simplifying assumptions in chapter IV. We then design a system in which the actions of these mTrackers, each seeking to maximize their own payoffs, actually results in ensuring lowest cost and highest performance of the system as a whole. We then consider a subsidiary problem of achieving fair division of resources among different mTrackers through admission control. We design an element that we call the tTracker, that tracks the performance of mTrackers, and takes admission control decisions based on the marginal disutility caused by users to the system. The objective here is to ensure that some level of fairness is maintained among the users in different mTracker domains, while at the same time ensuring that the costs in the system are not too high. Users interested in the file would approach the tTracker (which can be thought of as an mTracker search engine and system gateway) that would decide whether or not to admit the user into the system, and if admitted, would direct it to the mTracker whose domain the user belongs (mTracker nearest to the user in terms of ISP domain). We show that our tTracker optimally achieves fairness amongst users, while maintaining low system cost[2].

We simulate our system both using Matlab simulations in chapter V to validate our analysis, as well as ns-2 simulations in chapter VI to show a plausible implementation of the system as a whole. The simulations strongly support our architectural decisions. We conclude with ideas on the future in chapter VII.

[2]Note that switching off admission control would still imply that the total system cost would be minimized by mTrackers, but this could be arbitrarily high.

A. Related Work

There has been much recent work on P2P systems and traffic management, and we provide a discussion of work that is closely related to our problem. Fluid models of P2P systems, and the multi-phase (transient/steady state) behavior has been developed in [12, 14]. The results show how supply of a file correlates with its demand, and it is essentially transient delays that dominate. While they are not concerned with ISP incentives, their model provides the foundation for ours. In particular, we exploit their observation of multi-phase behavior of a P2P system. In our model we have distinct P2P systems for the same file in each ISP domain. Peers connect to the P2P system that is present in their ISP domain. However, a transient state P2P system can potentially forward its peers' requests to a steady state P2P system, i.e. inter-ISP forwarding of traffic is acceptable if expected gain in performance is greater than the cost of traffic exchange.

Traffic management and load balancing have become important as P2P networks grow in size. There has been work on traffic management for streaming traffic [15–17]. In particular, [15] focuses on server-assisted streaming, while [16, 17] aim at fair resource allocation to peers using optimization-decomposition. Similar ideas for traffic engineering for elastic traffic are studied in [18, 19]. But none of these works talk about the cost incurred to an ISP. Closest to our setting is work such as [8–10], that study the need to localize traffic within ISP domains. In [8], the focus is on allowing only local communications and optimizing the performance by careful peer selection, while [9] develops an optimization framework called P4P, that balances load across ISPs using cost information. A different approach is taken in [10], and peers are selected based on inputs on nearness provided by CDNs (if a CDN directs two peers to the same cache, they must be near by). In this thesis we attempt to provide an analytical characterization of the *tradeoff between delay performance and cost.*

Pricing and market mechanisms for P2P systems are of significant interest, and work such as [20,21] use ideas of currency exchange between peers that can be used to facilitate file transfers. The system we plan to design uses prices between agents (or *Trackers*) in P2P systems, which are confined to an ISP domain, that map to real-world costs of traffic exchange, but do not have currency exchanges between peers which still use BitTorrent style bilateral barter.

CHAPTER II

POPULATION GAMES AND EVOLUTIONARY DYNAMICS

Population games, a branch of *Game Theory*, provides a mathematical model to capture the strategic interactions among a large number of agents or players. Population games are used in a variety of disciplines, some of the applications include; multilateral externalities in *economics*, genetic natural selection in *biology*, routing in *computer networks* and many other applications.

Any application or game ($\mathcal{G}$) with the following attributes is called a population game:

1. Many players participate in the game $\mathcal{G}$.
2. The impact of one player's actions on another player's payoffs is very little.
3. Their exists a finite number of population classes ($\mathcal{Q} = \{1, ..., Q\}$) and each player belongs to exactly one population class ($j \in \mathcal{Q}$). All the players within a population class(j) have same set of strategies ($\mathcal{S}_j$) to choose from, and their payoffs are identical functions of their own choices and opponents choices.
4. The payoff of each player is a continuous function.

In the next section we give a formal definition for population games along with its different components. In the later sections of this chapter we describe about *potential games*,a type of population games, that we used to model our MultiTrack system. Later we introduce the notion of equilibrium for these potential games and the different evolutionary dynamics(or strategy dynamics) that can be used to attain this equilibrium.

Much of the discussion presented in the following sections can be found in [22].

A. Populations, Strategy Distributions and Payoffs

A *population game* $\mathcal{G}$, with Q non-atomic populations of players is defined by a mass(or number of players in that population) and a strategy set for each population and a payoff function corresponding to those strategies. By a non-atomic population, we mean that the contribution of each member of the population is infinitesimal. We denote the set of populations by $\mathcal{Q} = \{1, ..., Q\}$, where $Q \geq 1$. The population $q \in \mathcal{Q}$ has mass x_q. The set of strategies for population q is denoted $\mathcal{S}_q = \{1, ..., S_q\}$. A particular *strategy distribution* is a way the population q distribute themselves amongst the different strategies, i.e., a strategy distribution for q is a vector of the form $\vec{x}_q = \{x_q^1, x_q^2, \ldots, x_q^{S_q}\}$, where

$$\sum_{i=1}^{S_q} x_q^i = x_q$$

The set of strategies of a population $q \in \mathcal{Q}$, is denoted by

$$X_q = \{\vec{x}_q \in \mathbb{R}_+^{S_q} : \sum_{i=1}^{S_q} x_q^i = x_q\}$$

We denote the vector of strategies being used by the entire population as $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_Q\}$, where $\vec{x}_q \in X_q, \forall q \in \mathcal{Q}$. The vector $\mathbf{X}$ can be thought of as the state of the system. Let the space of all strategy distributions be $\mathcal{X}$.

Generally a population game is identified by its payoff function. For a given set of populations and their strategies different games can be devised using different payoff functions. A *payoff function* (per unit mass) $F : \mathcal{X} \to \mathcal{R}^Q$ is a continuous and differentiable function that assigns each population class a vector of payoffs, one for each strategy in that population. The payoff obtained by users of population class $q \in \mathcal{Q}$ from using strategy $i \in \mathcal{S}_q$, when the state of the system is $\mathbf{X}$, is denoted by $F_q^i(\mathbf{X}) \in \mathbb{R}$. Note that, the payoff obtained as a result of a strategy taken by a population q can depend on the strategy distribution within population q itself as well as the strategy distribution of other population classes. The total payoff of class q is then given by $\sum_{i=1}^{S_q} F_q^i(\mathbf{X}) x_q^i$, where we assume linearity for exposition. Players may be cooperative or non-cooperative in behavior.

Based on the properties or constraints on payoff functions, population games can be categorized into 3 different classes: *potential games, stable games and supermodular games.* Each of these classes impose a structure on their payoff function which renders their analysis relatively simple. We model our MultiTrack system as a *potential game*, so we dedicate a section to describe about potential games and the structure of their payoff functions.

1. Potential Games

In potential games, all information regarding the payoffs obtained by users of a population class can be captured in a single scalar-valued function. This scalar function is called the games's *potential function.*

Definition 1 Let $\mathcal{G}$ be a population game with payoff function(per unit mass) $F : \mathcal{X} \rightarrow \mathcal{R}^Q$. $\mathcal{G}$ is called a **Potential Game** if there exists a continuously differentiable function $\mathcal{T} : \mathcal{X} \rightarrow \mathcal{R}$ such that

$$\frac{\partial \mathcal{T}}{\partial x_q^i}(X) = F_q^i(X) \tag{2.1}$$

$\forall q \in \mathcal{Q}$ and $i \in \mathcal{S}_q$, where $X \in \mathcal{X}$ is the state of the system. The function $\mathcal{T}$ is called the potential function for game $\mathcal{G}$. It represents the games payoff function in an integrated form.

An important thing to observe in potential games is that the potential of the system increases as players switch to profitable strategies. For example, suppose the state of the system be $X \in \mathcal{X}$ and let $F_q^i(X) > F_q^j(X)$ for some population class $q \in \mathcal{Q}$ and the strategies $i, j \in \mathcal{S}_q$, i.e. strategy i yields more payoff than strategy j for players in population class q. Now let some players in population class q switch from state j to state i, the marginal affect in potential because of this shift is given as:

$$\frac{\partial \mathcal{T}}{\partial x_q^i}(X) - \frac{\partial \mathcal{T}}{\partial x_q^j}(X) = F_q^i(X) - F_q^j(X) \geq 0 \tag{2.2}$$

Thus a potential function characterizes the behaviour of a population of players. The theory behind potential games is analogous to the theory of Lyapunov function in control systems [23].

B. Stability and Equilibrium

As seen in the previous section the theory of population games provide a simple framework for describing strategic interactions among large numbers of players. We now focus on the equilibrium conditions for these games.

A commonly used concept of equilibrium in non-cooperative games is that of the *Nash equilibrium* which is defined as the solution of a game involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only his or her own strategy unilaterally. If each player has chosen a strategy and no player can benefit by changing his or her strategy while the other players keep theirs unchanged, then the current set of strategy choices and the corresponding payoffs constitute a Nash equilibrium. Whereas the Nash equilibrium is appropriate for the case of atomic players, in the context of infinitesimal players, a different concept of equilibrium called Wardrop Equilibrium is used.

1. Wardrop Equilibrium

A commonly used concept in non-cooperative games in the context of infinitesimal players, is the *Wardrop equilibrium* [24], which is defined below. Consider any strategy distribution $\vec{x}_q = [x_q^1, \ldots, x_q^{S_q}]$. There would be some elements which are non-zero and others which are zero. We call the strategies corresponding to those non-zero elements as the *options used by population* q.

Definition 2 A state $\hat{X}$ is a ***Wardrop equilibrium*** if for any population $q \in \mathcal{Q}$, all options used by the members of q yield the same marginal payoff to each member of q, whereas the marginal payoff obtained for options not used by population q would fetch lower payoffs.

Let $\hat{\mathcal{S}}_q \subset \mathcal{S}_q$ be the set of all strategies taken by population q in a strategy distribution when the state of the system is $\hat{\mathbf{X}}$. A Wardrop equilibrium $\hat{\mathbf{X}}$ is then characterized by the following relation:

$$F_q^s(\hat{\mathbf{X}}) \geq F_q^{\bar{s}}(\hat{\mathbf{X}}) \;\; \forall s \in \hat{\mathcal{S}}_q \text{ and } \bar{s} \in \mathcal{S}_q$$

The above concept refers to an *equilibrium condition* for population games. A natural question then would be, how does the system arrive at such an equilibrium state?. This leads us to the concept of evolutionary dynamics.

C. Evolutionary Dynamics

Traditionally, predictions of a players behavior in games are based on some notion of equilibrium, typically Nash equilibrium. These notions are founded on the assumption of equilibrium knowledge, which posits that each player correctly anticipates how his opponents will act. The equilibrium knowledge assumption is difficult to justify,especially in the context of large numbers of players, like population games. So the behavior of players in population games is modeled as a dynamic adjustment process, in which players myopically alter their behavior in response to their current state of the system. This dynamic adjustment process is termed as *Evolutionary Dynamics* or population dynamics, which is generally represented as a differential equation. It takes the current payoff's and the state of the system as input and outputs the rate at which players, belonging to the same class shift from one strategy to another.

We conclude this chapter with the description of two such population dynamics, which are the most widely used models, *Replicator Dynamics* [25] and Brown-von Neumann-Nash dynamics [26].

1. Replicator Dynamics

The rate of increase of $\dot{x}_q^i/x_q^i$ of the action i taken by players in population $q \in \mathcal{Q}$ is a measure of its evolutionary success. Following the basic tenet of Darwinism, we may express this success as the difference in fitness $F_q^i(X)$ of the action i and the average fitness $\sum_{i=1}^{S_q} x_q^i F_q^i(X)/x_q$ of the population j. Then we obtain

$$\frac{\dot{x}_q^i}{x_q^i} = \text{fitness of } i \text{ - average fitness.}$$

Then the dynamics used to describe changes in the mass of population q playing strategy i is given by

$$\dot{x}_q^i = x_q^i \left(F_q^i(X) - \frac{1}{x_q} \sum_{i=1}^{S_q} x_q^i F_q^i(X) \right). \tag{2.3}$$

The above expression thus says that a population would increase the mass of a successful strategy and decrease the mass of a less successful one. The proverb *"The rich get richer and the poor get poore"* exemplify the working of replicator dynamics. Note that the total mass of the population q is x_q which is a constant.

2. Brown-von Neuman-Nash Dynamics

Another commonly used model is called *Brown-von Neumann-Nash (BNN)* dynamics [26], and is defined as follows:

let,

$$\gamma_q^i = \max\left\{F_q^i(X) - \frac{1}{x_q}\sum_{i=1}^{S_q} x_q^i F_q^i(X), 0\right\} \tag{2.4}$$

denote the excess payoff to strategy i relative to the average payoff in population q. Then BNN dynamics are described by

$$\dot{x}_q^i = x_q\gamma_q^i - x_q^i\sum_{j=1}^{S_q}\gamma_q^j. \tag{2.5}$$

An interpretation of the BNN dynamics is that during any short time interval, all players in a population are equally likely to switch strategies, and do so at a rate proportional to the sum of the excess payoffs in the population. Those who switch choose strategies with above average payoffs, choosing each with probability proportional to the strategy's excess payoff. The reason for considering BNN dynamics is that unlike replicator dynamics, it has the property of non-complacency in that it allows extinct strategies to resurface, so that its stationary points are always Wardrop equilibria [22].

CHAPTER III

PEER TO PEER NETWORKS: BITTORRENT

A P2P network is the new wave in network architecture that is being used for many different applications today. The basic idea in a P2P network is to use the upload bandwidth of users or clients to upload pieces of a file. Thus, in a P2P network the peers act as both downloaders and uploaders of data. In P2P systems the total load of the system is shared among different peers where as in a client-server paradigm, the load is shared by limited number of central servers. Thus making P2P more efficient and scalable than classic client-server architecture. There are many P2P network architectures in existence, some of them are *Gnutella, Napster, KaZaA* and *BitTorrent*. Out of these, BitTorrent [7,11] is the most widely used and it is estimated that it accounts for 35% of all internet traffic. Our MultiTrack model is based on BitTorrent architecture so, for completeness, we give an overview of the functioning of BitTorrent architecture and its key components.

BitTorrent was designed in 2001 by Bram Cohen [11]. The following terminology is used in a BitTorrent system:

1. **Chunk**: In the BitTorrent architecture, a large file is divided into multiple chunks, where each chunk size is 256 Kilo-Bytes.

2. **Seeds and Leechers**: A peer that has downloaded the complete file and is willing to upload to other peers is called a seed, whereas if a peer does not want to upload to other peers it is called a Leecher.

3. **Downloaders**: A peer that is in the process of downloading is called a downloader. A downloader can simultaneously upload previously downloaded chunks of the file.

4. **Swarm**: All the peers (seeds and downloaders) that participate in the file exchange process, through a common torrent file. constitute a swarm.

5. **Torrent**: A torrent is a metadata file that contains metadata about the files it makes downloadable, including their names and sizes and checksums of all the chunks of these files. It also contains the address of a tracker that coordinates communication between the peers in the swarm.

6. **Tracker** : A central server which keeps track of all the peers in a swarm is called a tracker. The primary goal of a tracker is to coordinate access between peers.

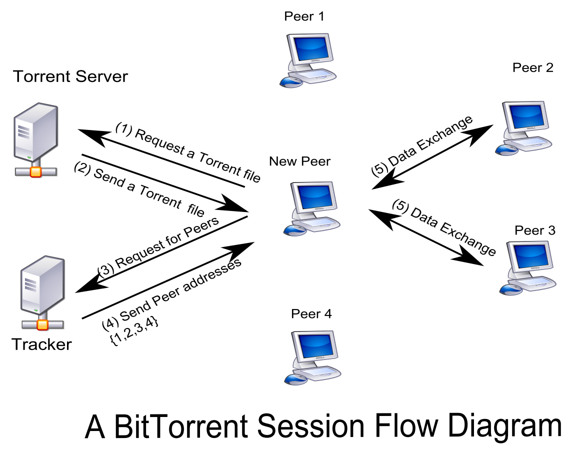

Fig. 2. A typical BitTorrent session consists of the following steps: (1) The new peer requests for a torrent file; (2) The torrent server replies with the torrent file; (3) The peer requests a Tracker for peer addresses to contact; (4) The Tracker responds with peer addresses 1, 2, 3 and 4; (5) Finally the new peer exchanges data with peers 2 and 3.

A BitTorrent session flow depicted in Figure 2. A new peer that wants to download a file needs to know about other peers to start its download process, so it should first locate a Tracker corresponding to the file, which will coordinate access to other peers. Information about Trackers for a file (among other information) is contained in torrent files, which are hosted at free web servers called torrent servers. Thus, the peer downloads the torrent file, and locates a Tracker using the information provided in the torrent file.

When a peer sends a request to a Tracker, corresponding to the file it wants, the Tracker returns IP addresses of a set of peers (*seeds and downloaders*) that the new peer should contact in order to download the file. The peer then connects to a subset of the given peers and downloads chunks of the file from them. While downloading the file, a peer sends updates to the Tracker about its download status (number of chunks uploaded and downloaded). Since peers leave the swarm, in order to maintain connectivity, existing peers should periodically send requests for new set of peers to its tracker and the above sequence repeats. Since a tracker knows about the state of each peer that is present in its peer cloud (or swarm), a tracker can potentially control access to the peer cloud. We exploit this capability of a tracker to design our model that we present in the next chapter.

The distinguishing feature of BitTorrent, which lead to its success over other P2P systems, is the way it handles the problem of *free riders*. A free rider is a peer that downloads data from other peers but do not contribute to other peers, thereby affecting the systems performance. Free riding was a severe problem in traditional P2P systems [27]. In order to overcome this problem, BitTorrent uses a peer selection algorithm called *Choking/Unchoking*. Using this scheme each peer can control to which other peers it can upload data. When a remote peer is selected for upload an *Unchoke* message is sent and, a peer stops uploading to a remote peer by sending a *Choke* message. At any time each peer can upload to a fixed number of other peers(default is four). The peers use a **"tit-for-tat"** strategy in selecting the peers to which it will upload, i.e. it choses to choke those peers from which it gets a poor download rate while uploading to those peers from which it got highest download rates. Thus each peer tries to maintain a download to upload ratio of one. For a seed unchoking is based on the download rate of remote peers rather than the upload rate.

CHAPTER IV

MULTITRACK SYSTEM

MultiTrack is a hybrid P2P network architecture similar to BitTorrent, described in chapter III in many ways. MultiTrack uses two different types of Trackers—tTrackers and mTrackers—that generalize the functions of the BitTorrent Tracker:

1. **tTracker** acts as a gateway into the system, in much the same manner as the server that hosts the .tor file in BitTorrent. Whereas the server in BitTorrent has no control over admission decisions of peers, the tTracker does have this choice. If admitted, the tTracker gives the peer the address of the mTracker nearest to the peer (in terms of ISP administrative domain).

2. **mTracker** acts as a gateway to a particular peer cloud of a file. We associate one or more mTrackers to each ISP, with each mTracker controlling access to its own peer cloud. Note that all these mTrackers are identified with the *same* file. Unlike BitTorrent Trackers, mTrackers are aware of each other, and form an overlay network among themselves. When a peer approaches an mTracker, the mTracker takes a decision on whether to admit it into its own peer cloud (at relatively low cost, but possibly poor delay performance) or to forward it to another mTracker (at higher cost, but potentially higher performance).

The rationale behind this architecture is as follows. At any time a peer cloud has a capacity associated with it, based on the maximum upload bandwidth of a peer in the cloud and the total number of chunks present at all the peers in the cloud (seeds and downloaders). Thus, a peer cloud can be thought of as a server with changing service capacity. In general, a peer-cloud has two phases of operation as mentioned in [12] and is shown in Figure 3:

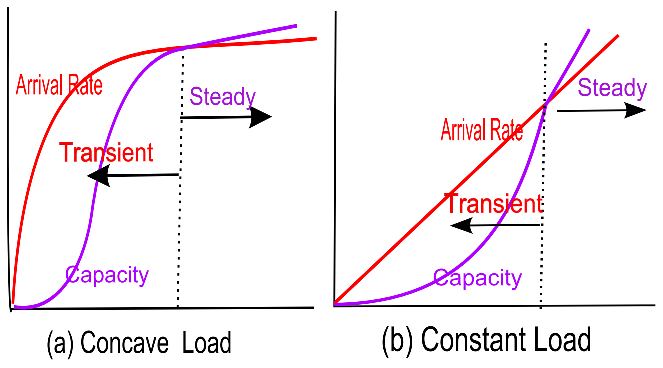

Fig. 3. Steady and transient phases of a P2P system; The evolution of service capacity of a P2P swarm with (a) concave arrival rate and (b)constant arrival rate, of peers into the system and no peer departures.

1. **Transient** phase where the available capacity is less than the demand (in other words, not enough peers with a copy of the file), and a
2. **Steady state** phase, where the available capacity is greater than the capacity required to satisfy demand.

Now, the capacities of the peer-clouds depend on the temporal evolution of demand. One geographic region (say the East Coast of the US) might see the evolution of demand earlier than another (say the West Coast of the US). Thus, there could potentially be steady state peer-clouds that have available capacity, but at high cost in terms of traffic exchange needed between ISPs. This is the core foundation on which our model is built.

We assume *time scale separation* between the dynamics of the two types of Trackers. Our assumption is that the capacity of a P2P system remains roughly constant over intervals of time, with capacity changes seen at the end of these time periods. We divide system dynamics into three time scales:

1. **Large:** The capacity of the peer cloud associated with each mTracker changes at this time scale.

2. **Medium:** tTrackers take admission control decisions at this time scale. They could increase or decrease the number of admitted peers based on feedback from the system. We will study dynamics at this time scale in Section B.

3. **Small:** mTrackers split the demand that they see among the different options (mTrackers visible to them) at this time scale. Thus, they change the probability of sending peers to their own peer-cloud or to other mTrackers at this time scale. We study these dynamics in Section A.

The artifice of splitting dynamics into these time scales allows us to design each control loop while assuming that certain system parameters remain constant during the interval. In the following sections, we present the design and analysis of our different system components.

A. mTracker: Splitting Demand

The objective of the mTracker is to split the demand that it sees among the different options (other mTrackers, and its own peer cloud) that it sees. Since each mTracker is associated with a different ISP domain, it would like to minimize the cost seen by that ISP, and yet maintain a good delay performance for its users.

As mentioned in the last section, peer-clouds can be in either *transient* or *steady-state* based on whether the demand seen is greater than or less than the available capacity. This dual-phase-mode operation was characterized by Yang *et al.* [12] and as shown in Figure 3, our mTtackers could be in one of these modes. An mTracker in the transient mode would like to offload some of its demand, while mTrackers in the steady-state mode can accept load. Thus, each mTracker j in the transient mode maintains a split probability vector $\hat{\vec{y}}_j = [\hat{y}_j^1 \dots \hat{y}_j^Q]$, where Q is the total number of mTrackers, and some of the y_j^i could be zero. We assume that the demand seen by mTracker j is a Poisson process of rate x_j. Thus, splitting traffic according to $\hat{\vec{y}}_j$ would produce Q Poisson processes, each with rate $x_j^i \triangleq y_j^i x_j$ $(i = 1, ...Q)$.

Now, each mTracker in the steady-state mode can accept traffic from mTrackers that are transient. It could, of course, prioritize or reserve capacity for its own traffic; we assume here that it does so, and the balance capacity available (in users served per unit time) of this steady state mTracker is C^i. Then the demand seen at each such mTracker i is the sum of Poisson processes that arrive at it, whose rate is simply $\sum_{l=1}^{Q} x_l^i$, and the M/M/1 delay seen by each peer sent to mTracker i is

$$\frac{1}{C^i - \sum_{l=1}^{Q} x_l^i}. \tag{4.1}$$

While we use an M/M/1 assumption in the paper, we note that our analysis applies to any convex increasing delay function.

Now, the steady state mTrackers are disinterested players in the system, and would like to minimize the total delay of the system. They could charge an additional price that would act as a congestion signal to mTrackers that forward traffic to them. Such a congestion price should reflect the ill-effect that increasing the load by one mTracker has on the others. What should such a price look like? Now, consider the expression

$$D(z) = \frac{1}{C^i - z^i}, \tag{4.2}$$

which is the general form of the delay seen by each user at mTracker i. The elasticity of delay with arrival rate z_i

$$\frac{\partial D(z^i)}{\partial z^i}\frac{z^i}{D(z^i)} = \frac{z^i}{C^i - z^i}. \tag{4.3}$$

The elasticity gives the fractional change in delay for a fractional change in load, and can be thought of as the cost of increasing load on the users. In other words, if the load is increased by any one mTracker, *all* the others would also be hurt by this quantity. Expressing the above in terms of delay (multiply by total delay) to ensure that all units are in delay, the elasticity per unit rate per unit time at mTracker i is just

$$\frac{\sum_{l=1}^{Q} x_l^i}{(C^i - \sum_{l=1}^{Q} x_l^i)^2}. \tag{4.4}$$

The above quantity represents the ill effect that increasing the load per unit time has on the delay experienced on all users at mTracker i. In some sense, the delay cost (4.1) is the disutility for using the mTracker, while the congestion cost (4.4) is the disutility caused to others using the mTracker. The mTracker can charge this price to each mTracker that forward peers to it. Note that a transient mTracker should charge *itself* this price as well, as it indicates the congestion that it is causing on its own users by increasing load.

Since mTrackers belong to different ISP domains, forwarding demand from one mTracker to the other is not free. Indeed, one of the main goals of our system is to tradeoff this cost with that of delay. We assume that the transit cost per unit rate of forwarding demand from mTracker j to mTracker i is p_j^i. Thus, the payoff of mTracker j due to forwarding traffic to mTracker i per unit rate per unit time is given by the sum of transit cost p_j^i with the delay cost (4.1) and congestion price

(4.4), which yields a total payoff per unit rate per unit time of

$$\frac{1}{C^i - \sum_{l=1}^{Q} x_l^i} + p_j^i + \frac{\sum_{l=1}^{Q} x_l^i}{(C^i - \sum_{l=1}^{Q} x_l^i)^2}. \tag{4.5}$$

Note that the mTracker would like as *small* a payoff as possible.

In the next subsections we will develop a game-theoretic framework for our system, and show how rational dynamics when coupled with the payoff function given above naturally results in minimizing the total system cost (delay cost plus transit cost).

1. MultiTrack Game

We model our system as a population game, which we covered briefly in chapter II. Each mTracker(and the peers in its domain) is a population class with a set of options (other mTrackers) to which it can forward its peer requests. The options available to an mTracker are other mTrackers' peer cloud or its own peer cloud. A strategy for each mTracker is then how it should partition its incoming requests among the different options so as to get equal payoff (or cost) from all the options that it uses, and higher cost from options that are not used, i.e., a Wardrop equilibrium.

We define the set of all mTrackers in the system as $\mathcal{Q} = \{1, \ldots Q\}$ and (as defined earlier) the number of mTrackers is $Q = |\mathcal{Q}|$. The service capacity available at mTracker $i \in \mathcal{Q}$ is denoted as C^i *users/sec* where C^i is fixed since we are considering the small time scale. We assume the basic unit of operation in our model is in *users*, so all rates and capacities are measured in *users/sec.*

Let $\vec{x} = [x_1, \ldots x_Q]$ be the total load vector of the system at the small time scale, where $x_i \forall i \in \mathcal{Q}$ is the total arrival rate of new peer requests at mTracker i. As in chapter II, a strategy of an mTracker $j \in \mathcal{Q}$ is to split its load x_j to different mTrackers including itself. We denote a strategy vector of mTracker j as $\vec{x}_j = [x_j^1 \ldots x_j^Q]$, where

$\sum_{i=1}^{Q} x_j^i = x_j$. If a mTracker j is not connected to mTracker i (or if it does not want to use mTracker i), then the rate $x_j^i = 0$.

We denote the vector of strategies being used by all the mTrackers as $\mathbf{X} = [\vec{x}_1 \dots \vec{x}_Q]$. The vector $\mathbf{X}$ represents the state of the system and it changes continuously with time. Let the space of all possible states of a system for a given load vector be denoted as $\mathbb{X}$, i.e $\mathbf{X} \in \mathbb{X}$.

The payoff (per unit rate per unit time) of forwarding requests from mTracker j to i, when the state of the system is $\mathbf{X}$ is denoted by $F_j^i(\mathbf{X}) \in \mathbb{R}$ and is assumed to be continuous and differentiable. As developed in the previous subsection, this payoff is

$$F_j^i(\mathbf{X}) = \frac{1}{C^i - \sum_{l=1}^{Q} x_l^i} + p_j^i + \frac{\sum_{l=1}^{Q} x_l^i}{(C^i - \sum_{l=1}^{Q} x_l^i)^2} \tag{4.6}$$

The total payoff at tracker j is given by $\sum_{i=1}^{Q} F_j^i(\mathbf{X}) \cdot x_j^i$.

We assume that mTrackers use rational dynamics to *learn* about the system, and in particular will focus on Replicator Dynamics (2.3) in this paper [1], repeated here for convenience.

$$\dot{x}_j^i = x_j^i \left(\frac{1}{x_j} \sum_{r=1}^{Q} x_j^r F_j^r(\mathbf{X}) - F_j^i(\mathbf{X}) \right). \tag{4.7}$$

Note that since each mTracker is trying to *minimize costs*, the payoffs described in chapter II are now replaced by costs, and the dynamics followed are the negative of the Replicator Dynamics presented in chapter II and described by (2.3).

2. Convergence of mTracker dynamics

The total cost in the system is defined to be the sum of the total system delay plus the total transit cost. In other words, we have weighted delay costs and transit costs

[1]Results from [22] can be used to generalize our results to a large class of dynamics called positively correlated dynamics.

equally when determining their contribution to the system cost. We could, of course, use any convex combination of the two without any changes to the system design. Hence using the M/M/1 delay model at each tracker, and adding transit costs, the total system cost when the system is in state $\mathbf{X}$ is given as:

$$\mathcal{C}(\mathbf{X}) = \sum_{i=1}^{Q} \left\{ \frac{\sum_{r=1}^{Q} x_r^i}{C^i - \sum_{l=1}^{Q} x_l^i} + \sum_{r=1}^{Q} p_r^i x_r^i \right\}. \tag{4.8}$$

Note that the cost is convex and increasing in the load. We will show that the above expression acts as a Lyapunov function for the system.

Theorem 1 *The system of mTrackers that follow replicator dynamics with payoffs given by (4.6) is globally asymptotically stable.*

Proof 1 *We prove the system stability using Lyapunov Theorem, details of which can be found in Appendix A, with $\mathcal{C}(\mathbf{X})$ defined in (4.8) as the Lyapunov function.*

$$\begin{aligned} \dot{\mathcal{C}}(\mathbf{X}) &= \sum_{i=1}^{Q}\sum_{j=1}^{Q} \frac{\partial \mathcal{C}}{\partial x_j^i} \dot{x}_j^i & (4.9) \\ &= \sum_{i=1}^{Q}\sum_{j=1}^{Q} F_j^i(\mathbf{X}) \dot{x}_j^i, & (4.10) \end{aligned}$$

where the above follows form the definition of $F_j^i(\mathbf{X})$ Eqn.(4.6). Now, let $\tilde{\mathcal{X}}$ be the set of states such that,

$$\dot{\mathcal{C}}(\mathbf{X}) = 0, \forall\ \mathbf{X} \in \tilde{\mathcal{X}}$$

From Eqn.(4.10) it is evident that $\dot{\mathcal{C}}(\mathbf{X}) = 0$, if:

$$F_j^i(\mathbf{X}) = 0 \ \ or \tag{4.11}$$

$$\left(\dot{x}_j^i = 0\right) \Rightarrow \left(\frac{1}{x_j} \sum_{r=1}^{Q} x_j^r F_j^r = F_j^i \right) \quad \forall\ i, j \in Q \tag{4.12}$$

Hence, $\tilde{\mathcal{X}}$ represents the set of equilibrium states of replicator dynamics.

From (4.7) we can substitute the value for $\dot{x}_j^i$ and we have

$$\dot{\mathcal{C}}(\mathbf{X}) = \sum_{i=1}^{Q}\sum_{j=1}^{Q} F_j^i x_j^i \left(\frac{1}{x_j}\sum_{r=1}^{Q} x_j^r F_j^r - F_j^i\right) \tag{4.13}$$

$$= \sum_{j=1}^{Q} x_j \left(\left(\sum_{i=1}^{Q} \frac{x_j^i}{x_j} F_j^i\right)^2 - \left(\sum_{i=1}^{Q} \frac{x_j^i}{x_j} (F_j^i)^2\right)\right) \tag{4.14}$$

Since function $f(x) = x^2$ is convex and $\sum_{i=1}^{Q} \frac{x_j^i}{x_j} = 1$, from Jensen's inequality we have, $\forall\ X \notin \tilde{\mathcal{X}}$:

$$\left(\left(\sum_{i=1}^{Q} \frac{x_j^i}{x_j} F_j^i\right)^2 - \left(\sum_{i=1}^{Q} \frac{x_j^i}{x_j} (F_j^i)^2\right)\right) \quad < \quad 0 \quad \forall\ j \in \mathcal{Q}$$

Thus,

$$\dot{\mathcal{C}}(\mathbf{X}) \quad < \quad 0, \quad \forall\ X \notin \tilde{\mathcal{X}}$$

hence, the system is globally asymptotically stable.

We have just shown that the total system cost (which is convex increasing) acts as a Lyapunov function. It should not come as a surprise then, that the total system cost is minimized by our dynamics. We prove this formally in the next section.

While replicator dynamics is a simple model, it has a drawback. During the different iterations of replicator dynamics, if the rate of forwarding requests from mTracker j to mTracker i (x_j^i), becomes zero then it remains zero forever. Thus, a strategy could become extinct when replicator dynamics is used. To avoid this problem we can use another dynamics called Brown-von Neuman-Nash(BNN) Dynamics,described in chapter II, and repeated here for convenience:

Then BNN dynamics is described as

$$\dot{x}_q^i = x_q \gamma_q^i - x_q^i \sum_{j=1}^{S_q} \gamma_q^j. \tag{4.15}$$

where

$$\gamma_q^i = \max \left\{ F_q^i(X) - \frac{1}{x_q} \sum_{i=1}^{S_q} x_q^i F_q^i(X), 0 \right\} \tag{4.16}$$

denote the excess payoff to strategy i relative to the average payoff in population q.

We can show that the system of mTrackers, which follows BNN dynamics, is globally asymptotically stable. The proof is similar to the proof of Theorem 1.

3. Cost efficiency of mTrackers

In previous work on selfish routing (*e.g.* [28], [29]), it was shown that the Wardrop equilibrium does not result in an efficient system performance. This inefficiency is referred to as the *price of anarchy*, and it is caused primarily because of the selfish strategies of users. However, work on population games [22] suggests that carefully devised price signals would indeed result in efficient equilibria. We show in this subsection that the Wardrop equilibrium attained by mTrackers is efficient for the system as a whole.

The objective of our system is to minimize the total cost for a given load vector $\vec{x} = [x_1, \ldots, x_Q]$. Here the total cost in the system is $\mathcal{C}(\mathbf{X})$ and is defined in (4.8). This can be represented as the following constrained minimization problem:

$$\min_{\mathbf{X}} \mathcal{C}(\mathbf{X}) \tag{4.17}$$

subject to:

$$\begin{aligned}\sum_{i=1}^{Q} x_j^i &= x_j \quad \forall\, j \in \mathcal{Q} && (4.18)\\ x_j^i &\geq 0. && (4.19)\end{aligned}$$

The Lagrange dual associated with the above minimization problem is

$$\mathcal{L}(\lambda, \mathbf{X}) = \max_{\lambda,h} \min_{\mathbf{X}} \Bigg(\mathcal{C}(\mathbf{X}) \quad - \qquad (4.20)$$
$$\sum_{j=1}^{Q} \lambda_j \Big(\sum_{i=1}^{Q} x_j^i - x_j \Big) \quad - \quad \sum_{i=1}^{Q} \sum_{j=1}^{Q} h_j^i x_j^i \Bigg)$$

where $h_j^i \geq 0$ and λ_j, $\forall\, i, j, \in \mathcal{Q}$ are the dual variables. Now the above dual problem gives the following Karush-Kuhn-Tucker first order conditions:

$$\frac{\partial \mathcal{L}}{\partial x_j^i}(\lambda, \mathbf{X}^\star) = 0 \quad \forall\, i, j \in \mathcal{Q} \qquad (4.21)$$

and

$$h_j^i x_j^{\star i} = 0 \quad \forall\, i, j \in \mathcal{Q} \qquad (4.22)$$

where $\mathbf{X}^\star$ is the global minimum for the primal problem (4.17). Hence from (4.21) we have

$$\begin{aligned}\frac{\partial \mathcal{C}}{\partial x_j^i}(\mathbf{X}^\star) - \lambda_j \frac{\partial(\sum_{i=1}^{Q} x_j^{\star i} - x_j^\star)}{\partial x_j^i} + h_j^i &= 0 \quad \forall\, i, j \in \mathcal{Q}\\ \Rightarrow \frac{\partial \mathcal{C}}{\partial x_j^i}(\mathbf{X}^\star) &= \lambda_j + h_j^i \quad \forall\, i, j \in \mathcal{Q} \qquad (4.23)\end{aligned}$$

We know from previous section that $\frac{\partial \mathcal{C}}{\partial x_j^i}(\mathbf{X}) = F_j^i(\mathbf{X})$ (follows from definition of payoff (4.6)). Thus from (4.23) we have

$$F_j^i(\mathbf{X}^\star) = \lambda_j + h_j^i \quad \forall\, i, j \in \mathcal{Q} \qquad (4.24)$$

From (4.22), it follows that

$$F_j^i(\mathbf{X}^\star) \;=\; \lambda_j \quad \text{when } x_j^{\star i} > 0 \;\forall\; i, j \in \mathcal{Q} \tag{4.25}$$

and

$$F_j^i(\mathbf{X}^\star) \;=\; \lambda_j + h_j^i \quad \text{when } x_j^{\star i} = 0 \;\forall\; i, j \in \mathcal{Q} \tag{4.26}$$

Now, consider the replicator dynamics (4.7), at stationary point we have $\dot{x}_j^i = 0$. Thus,

$$\hat{F}_j = \; F_j^i(\hat{X}) \quad \forall\; i, j \in \mathcal{Q} \tag{4.27}$$

or

$$\hat{x}_j^i = 0,$$

where

$$\hat{F}_j \triangleq \frac{1}{\hat{x}_j} \sum_{r=1}^{Q} \hat{x}_j^r F_j^r(\hat{X}) \quad \forall\; j \in \mathcal{Q}, \tag{4.28}$$

and $\hat{X}$ denotes a stationary point. The above equations imply that for mTracker j the per unit cost of forwarding traffic to other mTrackers is same across all the mTrackers that are in use. But for a mTracker (i) not in use the rate of forwarding (x_j^i) is 0 or equivalently, the cost is more than the average payoff. Thus, by definition of Wardrop equilibrium we can say that the above stationary points are indeed the Wardrop equilibria.

We observe that, the stationary point condition of replicator dynamics (4.27) is identical to the KKT first order conditions (4.25) and (4.26) of the minimization problem (4.17) when,

$$\hat{F}_j = \lambda_j \quad \forall\; j \in \mathcal{Q} \tag{4.29}$$

Theorem 2 *The solution of the minimization problem in (4.17) is identical to the Wardrop equilibrium achieved for the non-cooperative potential game $\mathcal{G}$.*

Proof 2 *From (4.27),(4.28) and (4.29) we can conclude that the Wardrop equilibrium achieved for the non-cooperative potential game $\mathcal{G}$ converges to the optimum solution of the Lagrange dual problem (4.20). Thus, to finish this proof we need to show that there is no duality gap between the primal (4.17) and the dual (4.20) problems. This follows immediately from convexity of the total system cost.*

Corollary 1 *All equilibrium states attained using replicator dynamics are not necessarily solutions to the minimization problem (4.17).*

Replacing Replicator dynamics with BNN dynamics, described in chapter II, would result in a Wardrop equilibrium which is exactly identical to the solution of the minimization problem (4.17).

B. tTracker: Admission Control

In the previous section we witnessed how each mTracker tries to reduce the cost in its peer cloud by forwarding requests to other mTrackers. However, minimizing the total delay does not mean that it is bounded. In order to ensure acceptable delay performance, we introduce an element called the tTracker, a Tracker that provides admission control functionality in order to attain an acceptable cost. The tTracker operates in the medium time scale; the mTracker loop is assumed to have converged to yield the lowest cost split at every instant at this time scale. The tTracker is a centralized element whose purpose is to (i) take a decision on whether to admit a requesting peer, and (ii) if admitted, to provide requesting users with the address of the mTracker closest to the user in terms of ISP domain. In some ways the tTracker supplements natural market dynamics—if the delay experienced by requesters were unbearably high, they would simply abort, causing the system to recover. However, such dynamics might cause large swings in quality over time; the tTracker precludes the occurrence of such swings.

One way to formulate an admission control problem is to provide hard constraints on the acceptable system cost. Such a problem could be formulated as a convex optimization problem shown below:

$$\max_{\vec{x}} \quad \textstyle\sum_{j=1}^{Q} w_j \log x_j \tag{4.30}$$

subject to:

$$\mathcal{C}^{\star}(\vec{x}) \leq \kappa \tag{4.31}$$

$$x_j \geq 0$$

where $\vec{x}$ is the load vector and $\mathcal{C}^{\star}(\vec{x})$ is the minimum value of the optimization problem (4.17) for a given load $\vec{x}$. We show in the following lemma, that the constraint set of the above convex optimization problem (4.30) is a convex set.

Lemma 3 *The set of all load vectors $\vec{x}$, satisfying the inequality constraint, $\mathcal{C}^{\star}(\vec{x}) \leq \kappa$ is a convex set.*

Proof 3 *Let $\vec{x}$ and $\vec{y}$ be two load vectors such that,*

$$\mathcal{C}^{\star}(\vec{x}) \leq \kappa \tag{4.32}$$

$$\mathcal{C}^{\star}(\vec{y}) \leq \kappa \tag{4.33}$$

Let X_{min} and Y_{min} be the states, corresponding to load vectors $\vec{x}$ and $\vec{y}$ respectively, *which results in minimum cost to the system, i.e.,*

$$\mathcal{C}(X_{min}) = \mathcal{C}^\star(\vec{x}) \tag{4.34}$$

$$\mathcal{C}(Y_{min}) = \mathcal{C}^\star(\vec{y}) \tag{4.35}$$

Consider,

$$\mathcal{C}(\alpha X_{min} + (1-\alpha)Y_{min}) \leq \alpha\mathcal{C}(X_{min}) + (1-\alpha)\mathcal{C}(Y_{min}) \tag{4.36}$$

the above inequality follows from the convexity of $\mathcal{C}(X)$.
Using Eqns(4.32), (4.33), (4.34) and (4.35), we get:

$$\begin{aligned} \mathcal{C}(\alpha X_{min} + (1-\alpha)Y_{min}) \leq & \quad \alpha\mathcal{C}^\star(\vec{x}) + (1-\alpha)\mathcal{C}^\star(\vec{y}) && (4.37) \\ \leq & \quad \alpha\kappa + (1-\alpha)\kappa && (4.38) \\ \leq & \quad \kappa && (4.39) \end{aligned}$$

if we consider $\vec{z} = \alpha\vec{x} + (1-\alpha)\vec{y}$, then from the definition of $\mathcal{C}^\star$

$$\mathcal{C}(Z_{min}) = \mathcal{C}^\star(\vec{z}) \tag{4.40}$$

where Z_{min} is the state of the system, corresponding to load $\vec{z}$, when the cost is minimum.

Clearly we can represent any state Z, corresponding to the load vector $\vec{z}$, in the form of $\alpha X + (1-\alpha)Y$, and thus it follows from the definition of $\mathcal{C}^\star$ and Eqn(4.39) that:

$$\mathcal{C}^\star(\alpha\vec{x} + (1-\alpha)\vec{y}) \leq \mathcal{C}(\alpha X_{min} + (1-\alpha)Y_{min}) \leq \kappa \tag{4.41}$$

Thus the set is convex.

If we think of $\sum_j w_j \log x_j$ as the total system utility, then $\mathcal{C}^\star(\vec{x})$ is the total system disutility. Instead of hard constraints on the cost, we relax the problem to simply ensure that the difference of utility and disutility (the *net utility*) is as large as possible. In other words, we relax the problem formulation after the manner of [30–32] to produce a formulation

$$\max_{\vec{x}} \quad \left(\sum_{j=1}^{Q} w_j \log x_j - \mathcal{C}^{\star}(\vec{x}) \right) \tag{4.42}$$

subject to:

$$x_j \geq 0$$

A gradient ascent type controller that could be used to solve the above problem is

$$\dot{x}_j = \left(w_j - x_j \frac{\partial \mathcal{C}^{\star}}{\partial x_j} \right) \quad \forall\, j \in \mathcal{Q}. \tag{4.43}$$

We design our tTracker controller around the above differential equation, and use it to tradeoff fair resource allocation versus system delay. Notice that the presence of $\frac{\partial \mathcal{C}^{\star}}{\partial x_j}$ implies that the controller needs to numerically evaluate the impact of a change in admission rates of any one mTracker j on the total system cost. The evaluation is straightforward if the cost expressions are simple convex functions as in (4.8). Here, all that the tTracker need do is run a "thought experiment" by changing x_j slightly, and observing the impact on total delay under the assumption that the mTracker uses Wardrop routing (this is exactly the way we implement it in our simulations). Under this tTracker control loop, we then have the following theorem.

Theorem 4 *Under the time scale separation assumption, the tTracker system with dynamics (4.43) is globally asymptotically stable.*

Proof 4 *We prove the stability of the system using the same Lyapunov technique as* *before. We use the following Lyapunov function*

$$Z(\vec{x}) = V(\hat{\vec{x}}) - V(\vec{x}) \tag{4.44}$$

$$\textit{where } V(\vec{x}) = \Big(\sum_{j=1}^{Q} w_j \log x_j - \mathcal{C}^\star(\vec{x})\Big) \tag{4.45}$$

which is a strictly concave function, and $\hat{\vec{x}}$ is its unique maximum.

Differentiating $Z(\vec{x})$ we get

$$\dot{Z} = -\sum_{j=1}^{Q} \frac{\partial V}{\partial x_j}\dot{x}_j \tag{4.46}$$

from (4.45) and (4.43)

$$\frac{\partial V}{\partial x_j} = \frac{w_j}{x_j} - \frac{\partial \mathcal{C}^\star(\vec{x})}{\partial x_j} \quad = \tfrac{\dot{x}_j}{x_j} \tag{4.47}$$

$$\therefore \;\; \dot{Z} = -\sum_{j=1}^{Q} \frac{\dot{x}_j^2}{x_j} \le 0 \quad \forall\, \vec{x} \tag{4.48}$$

$\dot{Z} = 0$ *when the system is in equilibrium.*

Hence the system is globally asymptotically stable according to Lyapunov theorem [23].

Finally, we note that the equilibrium conditions of the controller (4.43) are the same as the KKT conditions of the convex optimization problem (4.42). Hence, the controller succeeds in maximizing the required net utility.

CHAPTER V

MATLAB SIMULATIONS

We perform simulations in Matlab, on the simple overlay topology illustrated in Figure 4. Our objective is to validate our analytical results, and use the resulting insights to study a more realistic ns-2 implementation which is presented in the next chapter.

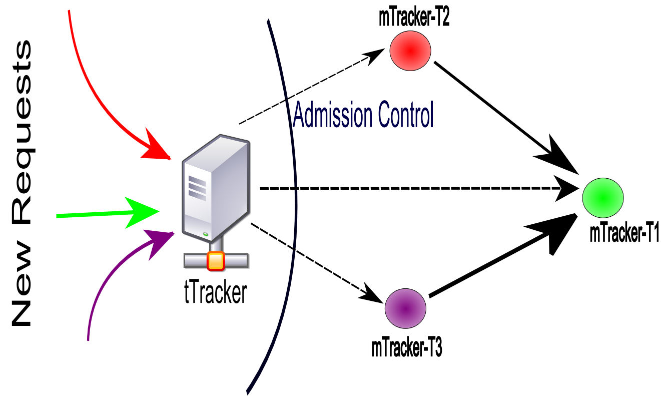

Fig. 4. Simulation topology: three mTrackers, T1, T2 and T3 and one tTracker which does admission control. A solid line from mTrackers T2 and T3 to T1 imply that T2 and T3 are in transient state and can forward requests to steady state mTracker T1. The dotted arrows from tTracker to mTrackers represent the flows admitted by tTracker.

Our system consists of 3 mTrackers (T1,T2 and T3) and a tTracker as shown in Figure 4. The mTracker-T1 is assumed to be in steady state (i.e. it has more capacity than demand in its peer swarm) and the other mTrackers T2 and T3 are in a transient state. Thus, T2 and T3 can forward traffic to T1. Our simulation parameters are chosen as follows. The initial arrival rates at the mTrackers are $x_1 = 10$ users/time, $x_2 = 20$ users/time and $x_3 = 20$ users/time, while the available capacities (fixed) are $C^1 = 30$ users/time, $C^2 = 20$ users/time and $C^3 = 20$ users/time, respectively. There is a transit price for traffic forwarding between mTrackers, and these values are chosen as $P_2^1 = 2$ unit and $P_3^1 = 1$ unit.

A. mTracker Simulations

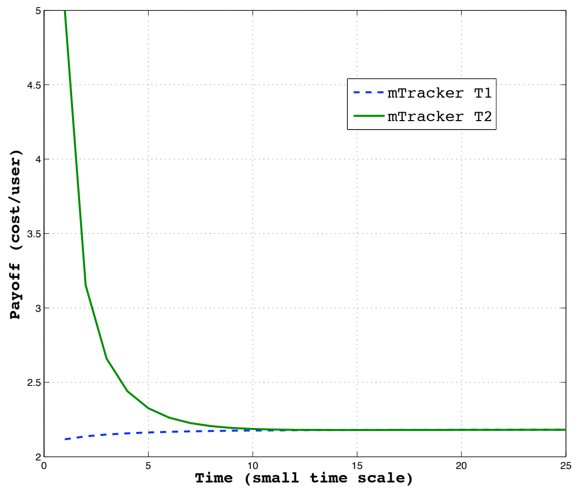

Fig. 5. The trajectory of payoffs of mTracker T2 for the 2 options available (local swarm and T1's swarm). The payoffs eventually equalize, showing that a Wardrop equilibrium has been attained.

Our first simulation is to validate the dynamics of the mTrackers at the small time scale. Thus, the arrival rate at each mTracker remains fixed, and as in Section A, they each use replicator dynamics in order to balance their payoffs among available options. We expect that (i) the per unit payoff for all available options to an mTracker should eventually be equal, and (ii) the total delay of the system would decrease to a minimum.

Figures 5 and 6 show the per unit payoffs and split probabilities, respectively, corresponding to T2. As expected, the per unit payoffs converge to identical values.

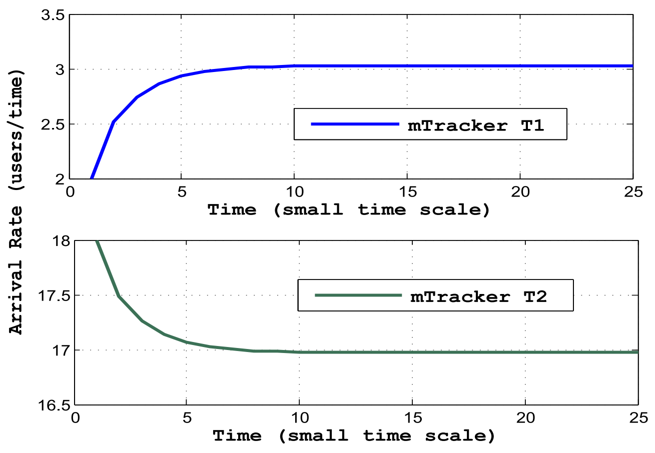

Fig. 6. The trajectory of traffic splits at mTracker T2 for the 2 options.

Finally, we plot the trajectory of total system cost $\mathcal{C}(\mathbf{X})$[1] in Figure 7. As expected it decreases with time, and converges to a minimum value corresponding to the Wardrop equilibrium of split probabilities.

B. tTracker Simulations

We next perform simulations at the medium time scale for the tTracker admission control loop. Here, at each time step the tTracker decides the admission rate (based on the dynamics developed in Section B), and the split probabilities of mTrackers converge essentially instantaneously (i.e., converge at the small time scale). As mentioned in Section B, the tTracker calculates $\frac{\partial \mathcal{C}^{\star}(\mathbf{X})}{\partial x_j}$ numerically to determine the cost of changing the admission rate x_j at mTracker j . We expect the net utility of the system (as defined in (4.42)) would increase to a maximum, and indeed, this is what

[1]Recall that this is the sum of total delay plus total transit cost.

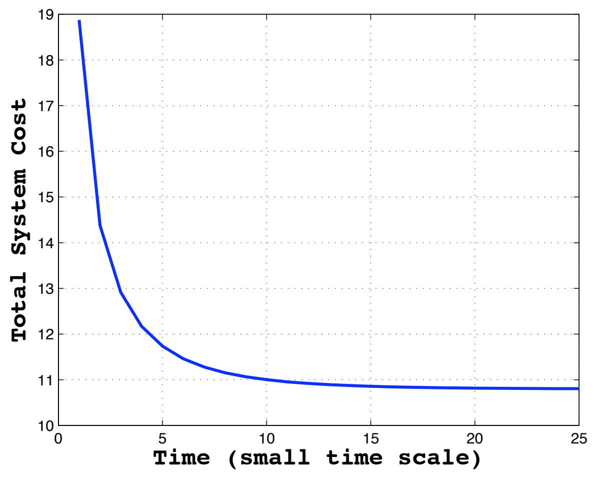

Fig. 7. The trajectory of total system cost in the system. As expected, it decreases over time to a minimum.

we observe in Figure 8.

While our Matlab simulations suggest that our system design is valid, they do not capture the true P2P interactions within each peer-cloud. In the next section, we present with implementation of MultiTrack in ns-2 in order to experiment with a more accurate representation of the system.

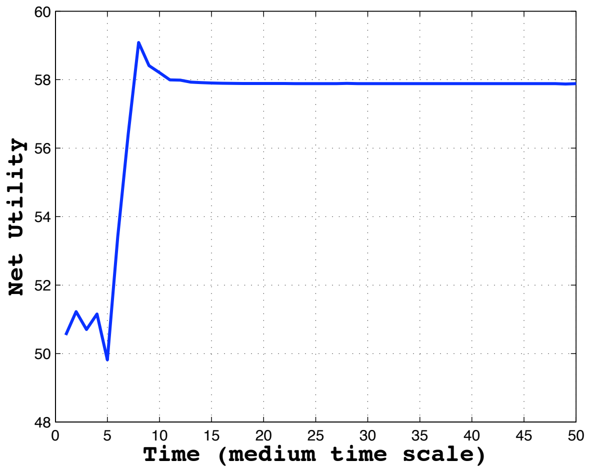

Fig. 8. The trajectory of net utility of the system when tTracker uses admission control. The net utility converges to a maximum.

CHAPTER VI

NS-2 SIMULATIONS

We wanted to test our system in a more realistic setting, so we implemented the MultiTrack system on ns-2. ns-2 is a discrete event driven simulator which is widely used for networking research. We leverage the ns-2 implementation of BitTorrent provided by Eger *et al.*, [33]. They implement a simplified BitTorrent model at packet level and flow level and compare the efficiency of each against optimum analytical models. In their simulations they observed that the flow level behaves similar to the analytical model but the packet level model deviates from the optimal by at most 30%. The packet level implementation is close to the real BitTorrent implementation and the flow level. In the packet level model, the packet exchange between peers and the underlying TCP mechanism is explicitly modeled. But in a flow level implementation, the underlying transport mechanism is ignored.

The differences between packet level and flow level implementation are the following:

1. The packet level model captures the influence of lower level protocols, like TCP, on the BitTorrent performance, whereas in flow level model the primary focus is on the application's performance and lower level protocol implementations are abstracted.

2. The simulation complexity for packet level simulations is high and is not suitable for simulating systems that have more peers, on the other hand flow level simulations are less complex and are ideal for simulating systems involving many peers like our MultiTrack system.

A. Overview of MultiTrack Implementation in ns-2

For implementing our MultiTrack system we extend the flow level model implemented by [33]. We chose the flow level model in the interest of reducing simulation time of our system which involves many peers. We extended the existing BitTorrent Tracker model to support mTracker and tTracker functionality. Our mTracker implementation splits traffic at periodic intervals using replicator dynamics in response to the payoffs that it sees, in an effort to attain Wardrop equilibrium. It forwards peer requests to other available mTrackers based on this split.

We saw in chapter IV the per unit payoff (F_i^j), for each mTracker(i) in forwarding traffic to another mTracker (j), comprises of the delay and the congestion price at mTracker (i) along with the transit price between mTrackers i and j. In our analytical model, presented in chapter IV, we assumed an M/M/1 delay at each mTracker, but in a real network the system need not follow M/M/1 type delay. So we should estimate the delay and congestion price at each mTracker during every small time scale. These values are calculated at the end of each small time scale, before the mTracker's re-calculate their split probabilities using replicator dynamics, as follows:

1. **Delay**: The per unit delay, in each mTracker's peer cloud, is measured by calculating the average download rate obtained by the peers that are admitted in the current time slot. Let NB be the number of bytes of data exchanged between all the peers and NP be the number of peers entering into the mTracker, during the small time interval. The average download rate achieved by an user during the current time slot is $\frac{NB}{NP}$. Hence, the delay experienced per user is calculated as $\frac{F}{NB/NP}$ where F is the size of the file.

2. **Congestion Price**: The congestion price of a system with delay D and arrival rate z is given as $\frac{\partial D}{\partial z} z$, which follows from the *elasticity* Equation(4.3). So we

measure the change in delay and change in arrival rate from the previous and current time unit to calculate the congestion price.

The configuration parameters required for each mTracker are presented in Table I below:

Table I. A list of configuration parameters needed for each mTracker

Parameter	Description
Adjacent Trackers	List of adjacent mTrackers to which this mTracker can potentially forward new peers
Transit Prices	The transit prices between an mTracker and each adjacent tracker
Split Interval	This is the time interval during which mTracker decides to change its splitting rate based on replicator dynamics
Arrival Rate	The arrival process of new peer requests,is a Poisson process with mean specified by this value
Split Probabilities	The initial probabilities of splitting traffic among different mTrackers
Number of Seeds	The number of seeds present in this mTracker domain. The number of seeds along with their upload bandwidth can be used to determine the initial capacity available at an mTracker.

Apart from the above parameters, BitTorrent specific parameters like, the upload bandwidth of each peer, the file size and the size of each chunk should also be configured.

For our simulation we use the same network topology that we used for our matlab simulations, shown in Figure 4, with 3 *mTrackers* T1, T2 and T3 and one tTracker. In our simulation, each peer has an upload capacity of 300 $Kbps$ and their download capacity is not restricted. The requested file size is 5 MB and each chunk has a size of 256 KB. Peer arrivals are created according to Poisson processes of different rates. T1 has 100 seeds in its peer swarm while T2 and T3 have 5 seeds each. We set the update interval for the mTracker to be 8 *sec*. Thus, each mTracker calculates the splitting probabilities for the different options at this frequency. We fix the initial arrival rates to be $x_1 = 3$ users/sec,$x_2 = 5$ users/sec and $x_3 = 7$ users/sec, set transit costs to be $P_2^1 = 20$ and $P_3^1 = 10$. The tTracker, which performs admission control, does so at 40 *sec* intervals. The total simulation time is 400 *sec*.

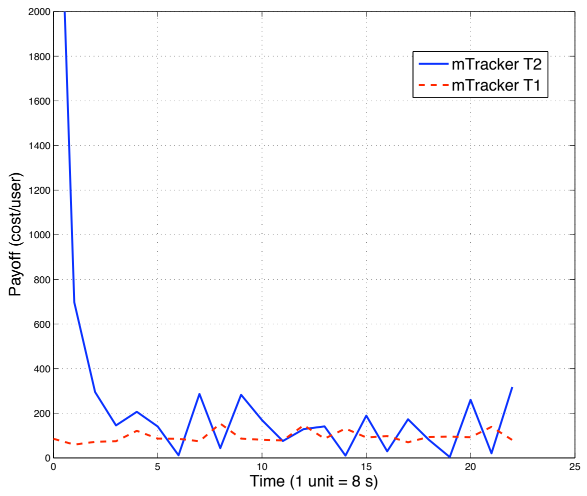

Fig. 9. The trajectory of payoffs of mTracker T2 for the 2 available options (local swarm and T1's swarm).

Figure 9 shows the trajectory of the payoff obtained by mTracker 2 (over the short time scale), while Figure 10 shows same for mTracker 3. As expected, the payoffs converge to the average value, attaining a Wardrop equilibrium.

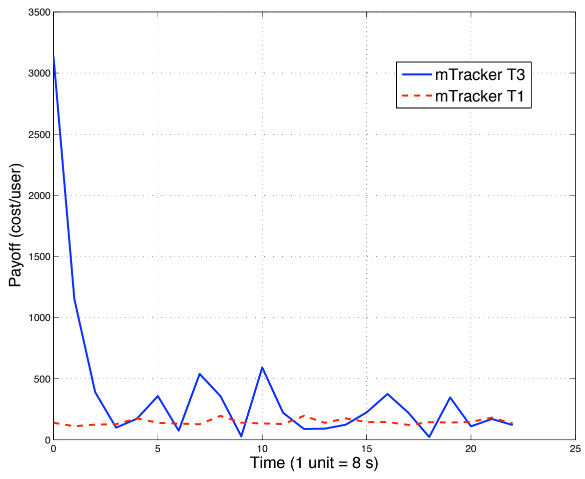

Fig. 10. The trajectory of payoffs of mTracker T3 for the 2 available options (local swarm and T1's swarm).

We next observe the delay performance of the whole system. The temporal evolution of per user delay is shown in Figure 11. We notice that as expected, the delay for the MultiTrack system decreases with time, and converges to a low value.

Next, we present the results of a comparison study of our MultiTrack systems performance, with respect to the transit price,delay and total cost of the system, against the following systems:

1. **No Splitting**: We disable splitting at each mTracker. Thus, all the requests originating in an mTracker domain are served locally. We will see that the delay experienced in such a system is very big.

2. **MultiTrack without transit price**: We consider a MultiTrack system with the transit prices between mTrackers ignored, i.e. the payoff's do not contain

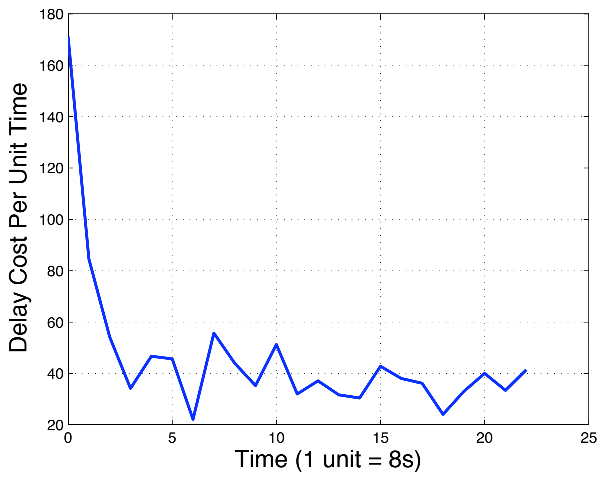

Fig. 11. The trajectory of total system delay with MultiTrack. The delay converges to a low value.

the transit prices. We will observe in figure 12 the transit prices of these systems is highest.

We first compare the transit prices of these three systems. Figure 12 shows the trajectory of the total transit price of the system. Without splitting traffic between ISP domains, there is no transit traffic so the price of transit is zero. On the other hand, splitting traffic without regard to prices, causes a high transit price. MultiTrack achieves a price between these two extremes.

We next study the delay experienced by users. Figure 13 shows the average delay experienced by users. As expected, without traffic splitting, the delay is high. When traffic is split without regard to transit price, the delay is the lowest possible. MultiTrack achieves a price between these two extremes.

We finally observe the cost (transit price plus cost) performance of the whole system. The temporal evolution of cost is shown in Figure 14. The impact of using MultiTrack is clearly illustrated here. The system without traffic splitting has a high

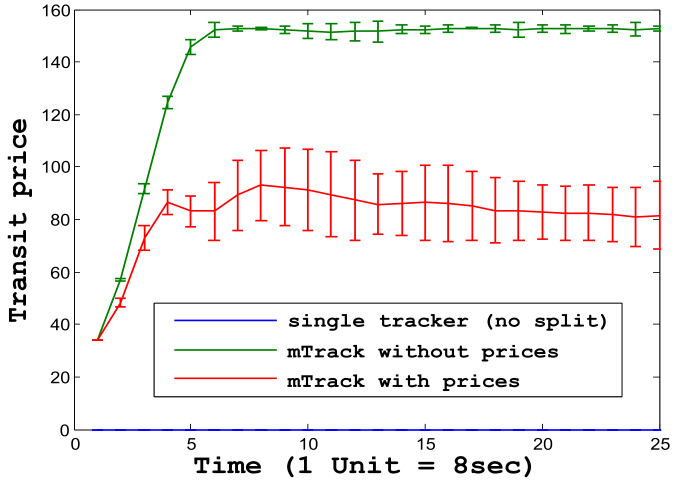

Fig. 12. The trajectory of total transit price. Without traffic splitting, the price is zero. With traffic splitting without regard to price, the price is high. MultiTrack takes prices into account, and has a price between these extremes.

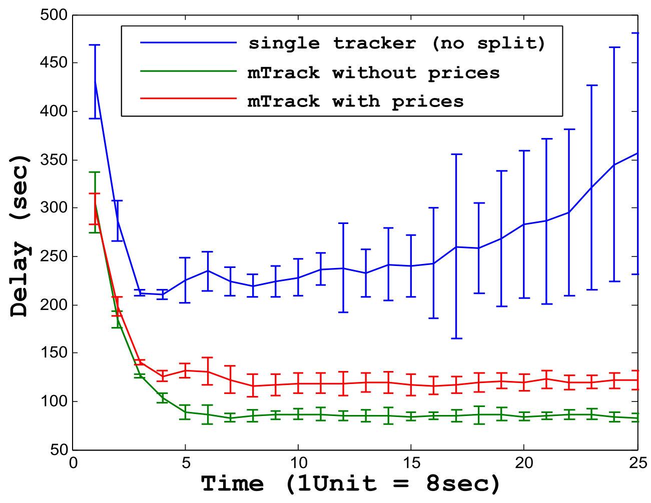

Fig. 13. The trajectory of average user delay. Without traffic splitting, the delay is high. With traffic splitting without regard to price, the delay is low. MultiTrack takes delays into account, and has a delay between these extremes.

cost due to increased user delays, while traffic splitting without regard to prices has a high cost due to excessive transit traffic. MultiTrack takes both transit price and user delay into account, and hence achieves the lowest possible cost.

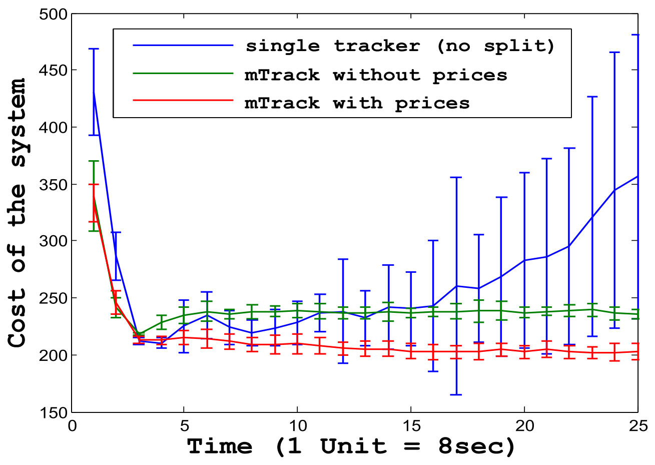

Fig. 14. The trajectory of total user delay. Without traffic splitting, the cost (delay plus transit price) is high. With traffic splitting without regard to price, the delay is low but transit price is high, causing high cost. MultiTrack takes prices and delays into account, and has lowest total cost.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

As the popularity of P2P systems as a medium for content distribution has grown, it has become clear that aligning incentives between the system performance in terms of the user QoS, and the transit costs faced by ISPs will be increasingly important. Fundamental to this problem is the realization that resources may be distributed geographically, and hence the marginal performance gain obtained by accessing a resource is offset in part by the marginal cost of transit in accessing it. In this paper, we consider delay and transit costs as two dimensions and attempt to design a system—MultiTrack–that attains optimal operating point assuming a given weight for each dimension.

Our system consists of two types of Trackers—mTrackers, that form an overlay among themselves and act as gateways to peer-clouds, and a tTracker that takes system admission control decisions. If admitted by the tTracker, users in an ISP domain are directed to the mTracker associated with that domain. The mTracker takes a decision whether the marginal decrease in delay by forwarding the user to a resource rich peer-cloud (perhaps in a different ISP domain) is offset by the marginal increase in its transit cost. We show that a simple price-based controller based on replicator dynamics could ensure that the total system cost is minimized in spite of each mTracker being selfish.

The tTracker runs an admission control loop that calculates the trade-off between the marginal utility in increasing the admission rate in a particular ISP domain to the marginal increase in system cost. It thus allows the correct rate of users into the system to attain optimal performance.

We validated our system design using Matlab simulations, and implemented the system on ns-2 in order to conduct more realistic experiments. We showed that our system significantly outperforms a system in which costs are the only control dimension (localized traffic only).

In the future, we would like to build the MultiTrack system on a testbed in order to study performance-cost trade-offs in a real-world setting of content distribution. We would also like to mathematically characterize the time taken for an mTracker to switch from transient state to steady state,i.e. what is the time required for a transient state mTracker to serve all its peers locally. We would prefer a distributed admission control at each mTracker, so we would like to design an admission control model that is more elegant, unlike the thought experiment which is used in the current tTracker's admission control model.

REFERENCES

[1] A. Moore, J. Hall, C. Kreibich, E. Harris, and I. Pratt, "Architecture of a network monitor," in *Passive & Active Measurement Workshop (PAM2003)*, 2003.

[2] C. Fraleigh, S. Moon, B. Lyle, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, no. 6, pp. 6–16, 2003.

[3] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proc. SOSP*, 2003, pp. 314–329, Bolton Landing, New York.

[4] Pando Networks, Inc., "Download and share . . . BIG," http://www.pando.com/.

[5] A. Broache, "FCC chief grills Comcast on BitTorrent blocking," *C|Net News.com, http://news.cnet.com/8301-10784_3-9878330-7.html*, Feb. 25, 2008.

[6] E. Bangeman, "P2P responsible for as much as 90 percent of all 'Net traffic, http://arstechnica.com/old/content/2007/09/p2p-responsible-for-as-much-as-90-percent-of-all-net-traffic.ars," *ArsTechnica*, Sept. 3, 2007.

[7] "BitTorrent," http://www.bittorrent.com/, 2005.

[8] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can isps and p2p users cooperate for improved performance?," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 29–40, 2007.

[9] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p: provider portal for applications," in *Proc. ACM SIGCOMM Conference on Data Communication*, 2008, pp. 351–362, Seattle, WA.

[10] D. R. Choffnes and F. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 363–374, 2008.

[11] B. Cohen, "Incentives build robustness in bittorrent," in *Proc. of Workshop on the Economics of p2p Systems*, June 2003.

[12] X. Yang and G. de Veciana, "Performance of peer-to-peer networks: service capacity and role of resource sharing policies," *Perform. Eval.*, vol. 63, no. 3, pp. 175–194, 2006.

[13] D. Fudenberg and J. Tirole, *Game Theory*, Cambridge, MA: MIT Press, 1991.

[14] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2004, pp. 367–378, Portland, Oregon.

[15] E. Setton and J. Apostolopoulos, "Towards quality of service for peer-to-peer video multicast," in *Proc. ICIP*, 2007, pp. 81–84, San Antonio, TX.

[16] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," in *Proc. ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2008, pp. 313–324, Annapolis, MD.

[17] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility maximization in peer-to-peer systems," in *Proc. ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2008, pp. 169–180, Annapolis, MD.

[18] J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards multi-layer traffic engineering: Optimization of congestion control and routing," *IEEE J. Selected Areas in Communications*, vol. 25, no. 5, pp. 868–880, 2007.

[19] R. Zhang-Shen, "Designing a predictable backbone network using valiant load-balancing," Ph.D. dissertation, Dept. of Electrical Engineering, Stanford University, Stanford, CA, 2007.

[20] C. Aperjis, M. J. Freedman, and R Johari, "Peer-assisted content distribution with prices," in *Proc. ACM CoNEXT Conference*, 2008, pp. 1–12, Madrid, Spain.

[21] C. Aperjis, M. J. Freedman, and R. Johari, "A comparison of bilateral and multilateral exchanges for peer-assisted content distribution," in *Network Control and Optimization: Second Euro-NF Workshop, NET-COOP 2008 Paris, France, September 8-10, 2008. Revised Selected Papers*, 2009, pp. 1–8, Berlin.

[22] W. H. Sandholm, "Potential games with continuous player sets," *Journal of Economic Theory*, vol. 97, pp. 81–108, January 2001.

[23] H. Khalil, *Nonlinear Systems*, New Jersey: Prentice Hall, 1996.

[24] J. Wardrop, "Some theoretical aspects of road traffic research," *Institution of Civil Engineers, Part II*, vol. 1, no. 36, pp. 352–362, 1952.

[25] J. Hofbauer and K. Sigmund, *Evolutionary Games and Population Dynamics*, United Kingdom: Cambridge University Press, 1998.

[26] G. W. Brown and J. von Neumann, "Solutions of games by differential equations," *Contributions to the Theory of Games*, vol. 24, pp. 73–79, 1950.

[27] E. Adar and B. Huberman, "Free riding on Gnutella," *First Monday, http://firstmonday.org/*, vol. 5, no. 10, 2000.

[28] T. Roughgarden and É. Tardos, "How bad is selfish routing?," *J. ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[29] T. Roughgarden, "The price of anarchy is independent of the network topology," in *Proc. Thiry-fourth Annual ACM Symposium on Theory of Computing*, 2002, pp. 428–437, Montreal, Quebec, Canada.

[30] F. P. Kelly, "Models for a self-managed Internet," *Philosophical Transactions of the Royal Society*, vol. A358, pp. 2335–2348, 2000.

[31] F. P. Kelly, "Mathematical modelling of the Internet," in *Mathematics Unlimited - 2001 and Beyond; B. Engquist and W. Schmid (Eds.)*, 2001, Berlin: Springer-Verlag, pp. 685–702.

[32] S. Shakkottai and R. Srikant, *Network Optimization and Control*, Delft, The Netherlands: Now Publishers, 2008.

[33] K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann, "Efficient simulation of large-scale p2p networks: Packet-level vs. flow-level simulations," in *Proc. 2nd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN'07)*, jun 2007, pp. 9–16, Monterey Bay.

APPENDIX A

LYAPUNOV STABILITY THEOREM

We present an overview of Lyapunov stability theorem and Lyapunov function here. [23] contains more details about Lyapunov theory.

- Let $x = 0$ be an equilibrium point for a system $\mathcal{S}$ that has the following dynamics $\dot{x} = f(x)$.
- Let $V : \Re^n \longrightarrow \Re$, be a continuously differentiable function such that $V(x) > 0$, $\forall x \neq 0$ and $V(0) = 0$, when $x = 0$, then $V(x)$ is called the *Lyapunov function.*

The system $\mathcal{S}$ is *globally asymptotically stable* if $V(x)$ is radially unbounded and $\dot{V(x)} < 0$, $\forall x \in \mathcal{R}^n - \{0\}$. Where a function $V(x)$ is *radially unbounded*, if $x \rightarrow \infty \implies V(x) \rightarrow \infty$.

VITA

Vinith Kumar Reddy Podduturi, is a graduate student in the Dept. of Electrical and Computer Engineering at TAMU. He received dual degrees of B.E. (Honors) computer science and M.Sc. (Honors) mathematics from Birla Institute of Technology and Science, Pilani, India in 2006. He received his M.S in computer engineering at TAMU in 2009. His interests are in communication networks, specifically peer-to-peer networks, internet pricing, congestion control, wireless networks and network coding. Prior to arriving at Texas A&M, he worked for a startup company, Anveshan Telecom Pvt. Ltd., India, from 2006-2007. Before that he worked as an intern in LG Soft India Pvt. Ltd., India, from 2005-2006. He can be contacted at the following address: Department of Electrical and Computer Eng., Texas A&M University, 331F WERC, College Station, Texas 77843-3128. His email-id is reddy.vinith@gmail.com.