Peer-to-Peer Networks – Protocols, Cooperation and Competition

Hyunggon Park
Signal Processing Laboratory (LTS4), Institute of Electrical Engineering, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland

Rafit Izhak Ratzin
Computer Science Department, University of California, Los Angeles (UCLA), Los Angeles, USA

Mihaela van der Schaar
Multimedia Communications and Systems Laboratory, Electrical Engineering Department, University of California, Los Angeles (UCLA), Los Angeles, USA

1. INTRODUCTION

Peer-to-peer (P2P) networks connect many end-hosts (also referred to as peers) in an ad-hoc manner. P2P networks have been typically used for file sharing applications, which enable peers to share digitized content such as general documents, audio, video, electronic books, etc. Recently, more advanced applications such as real-time conferences, online gaming, and media streaming have also been deployed over such networks. Unlike traditional client-server networks, where servers only provide content, and clients only consume content, in P2P networks, each peer is both a client and a server.

It has been observed that P2P file sharing applications dominate Internet traffic usage. In fact, a wide range of measurements, which were performed in 8 different geographic regions during the years of 2008-2009, show that P2P networks generated most of the traffic in all monitored regions, ranging from 43% in Northern Africa to 70% in Eastern Europe (http://www.ipoque.com/). The same study also identified that BitTorrent (Cohen, 2003) is the most popular protocol on the Internet, generating most of the traffic in 7 out of 8 regions ranging from 32% in South Africa to 57% in Eastern Europe. The details of the BitTorrent protocol will be discussed in Section 3. Recently, media delivery and streaming services over the Internet such as YouTube (http://www.youtube.com/), PPLive (http://www.pplive.com), and Internet video broadcasting (e.g., AOL broadcast, MSNBC, CBS, etc.) have emerged. These services have become very popular, as they can deliver video to a large number of receivers simultaneously at any given time. In order to reduce infrastructure, maintenance, and service costs, and provide more reliable services, the content providers often implement their services using P2P network.

While several designs for P2P systems have been successfully deployed for file sharing and real-time media streaming, key challenges such as the design of optimal resource reciprocation strategies among self-interested peers still remain largely unaddressed. For example, pull-based techniques (Cohen, 2003; Pai et al., 2005; Zhang et al., 2005) are designed assuming that peers are altruistic and are willing to provide their available data chunks (pieces) whenever requested. However, such assumptions may be undesirable from the perspective of a self-interested peer, which aims to maximize its own utility. Thus, efficient resource reciprocation strategies need to be deployed, which can also provide incentives to the peers for their contributions.

In BitTorrent systems, incentive strategies are based on the so-called tit-for-tat (TFT) strategy, where a peer selects some of its associated peers (i.e., leechers), which are currently uploading at the highest rates, and provides them its content for downloading (Cohen, 2003). This simple strategy is currently implemented in BitTorrent systems, and provides good performance. However, a key disadvantage of this resource reciprocation strategy is that peers decide how to determine their resource reciprocation based on only the *current* upload rates that it receives from its associated peers, and does not consider how this reciprocation will impact their upload rates in the future. In other words, the resource reciprocation based on the TFT is *myopic*. Since peers in P2P networks are generally involved in repeated and long-term interactions, such myopic resource reciprocation strategy can result in a suboptimal performance for the involved peers.

More advanced resource reciprocation strategies have been recently proposed in (Park & van der Schaar, 2009), where the resource reciprocation among the interested peers is modeled as a stochastic game (Fudenberg & Tirole, 1991). In this framework, peers determine their resource distributions by explicitly considering the probabilistic behaviors (reciprocation) of their associated peers. Unlike existing resource reciprocation strategies, which focus on myopic decisions, it formalizes the resource reciprocation game as a Markov Decision Process (MDP) (Bertsekas, 1976) to enable peers to make foresighted decisions on their resource distribution in a way that maximizes their cumulative utilities, i.e., the sum of their immediate and future utilities. Thus, this strategy can improve the performance of the peers, which are generally involved in long-term and repeated interactions. When the foresighted strategies are deployed in practice, the peers' *bounded rationality* should be considered, because perfectly rational decisions are often infeasible in practice due to their memory and computational constraints. Peers can only have limited knowledge of the other players' behavior and limited ability to analyze their environment. Therefore, it is essential to study the impact of the peers' bounded rationality on 1) the performance degradation of the proposed resource reciprocation strategy and 2) their repeated interactions (resource reciprocation).

This chapter is organized as follows. In Section 2, we overview various P2P network structures and discuss their advantages and disadvantages. In Section 3, we discuss the BitTorrent system, which is one of the most popular file sharing protocols. We also discuss the limitations of BitTorrent systems. In Section 4, we overview existing P2P-based media streaming applications, and discuss how several mechanisms have been developed to support real-time media streaming requirements. In Section 5, we show the recently proposed foresighted resource reciprocation strategies, which can improve the performance of P2P-based applications. In Section 6, we discuss new directions for game-theoretic approaches to incentive design in P2P networks. Conclusions of this chapter are drawn in Section 7.

2. OVERVIEW OF P2P SYSTEM STRUCTURES

P2P systems can be classified into two different classes: *structured* P2P systems and *unstructured* P2P systems. In structured P2P systems, connections among peers in the network are fixed, and peers maintain information about the resources (e.g., shared content) that their neighbor peers possess. Hence, the data queries can be efficiently directed to the neighbor peers that have the desired data, even if the data is extremely rare. Structured P2P systems impose constraints both on node (peer) graph and on data placement to enable efficient discovery of data. The most common indexing that is used to structure P2P systems is the Distributed Hash Tables (DHTs) indexing. Similar to a hash table, a DHT provides a lookup service with (*key, value*) pairs that are stored in the DHT. Any participating peers can efficiently retrieve the value associated with a given unique key. However, this may result in higher overhead compared to unstructured P2P networks. Different DHT-based systems such as Chord (Stoica et al., 2001), Pastry (Rowstron & Druschel, 2001), Tapestry (Zhao et al., 2004), CAN (Ratnasamy et al., 2001) are different in their routing strategies and their organization schemes for the data objects and keys.

Unlike structured P2P systems, in unstructured P2P systems, connections among peers in the network are formed arbitrarily in flat or hierarchical manners. In order to find as many peers that have the desired content as possible, peers in unstructured P2P systems query data based on several techniques such as flooding (e.g., among the super-peers in KaZaA (http://www.kazaa.com/)), random walking (Gkantsidis et al., 2004), and expanding-ring (e.g., Time-To-Live counter in Gnutella (http://wiki.limewire.org/)). Three different designs of unstructured P2P systems exist: centralized unstructured P2P systems, hybrid unstructured P2P systems, and decentralized (or pure) unstructured P2P systems.

In a centralized unstructured P2P system, a central entity is used for indexing and bootstrapping the entire system. In contrast to the structured approach, the connection between peers in the centralized unstructured approach is not determined by the central entity. A BitTorrent network discussed in Section 3 is an example of a centralized unstructured P2P network. Napster (http://www.napster.com/), the network that pioneered the idea of P2P file sharing, is another example of a centralized design. In Napster, a server (or server farm) is used to provide a central directory. A peer in the network informs the directory server of its IP address and the names of the contents that it makes available for sharing. Thus, the directory server knows which objects each peer in the network have, and then, creates a centralized and dynamic database that maps content name into a list of IPs. The main drawback of Napster's design is that the directory server is a single point of failure. Hence, if the directory server crashes, then the entire network will also collapse. Moreover, increasing the size of the network may cause a bottleneck in the directory server, due to the need of responding to many queries and maintaining a large database for this meta-data information. The bottleneck can only be resolved by adding more infrastructure (e.g., more servers), which may be expensive.

The decentralized (or pure) unstructured P2P network is an overlay network. An overlay network is a logical network. An edge in this network exists between any pair of peers that maintain a TCP connection. The decentralized unstructured overlay network is flat, meaning that all peers act as equals. There is neither a central server that manages the network, nor are there preferred peers with a special infrastructure function. The network has a single routing layer. Gnutella (http://wiki.limewire.org/) is an example of a decentralized unstructured P2P network. In order to join the Gnutella network, a user initially connects to one of several known-bootstrapping peers. The bootstrapped peers then respond with the information about one or more existing peers in the overlay network. This information includes the IP address and port of each peer. The peers in Gnutella are aware only of their neighbor peers. Peers that are connected with

each other in the overlay network have a common virtual edge in the overlay network. In Gnutella, queries are distributed among the peers using a variation of a flooding mechanism. A peer that is interested in specific content sends a query to its neighbors in the overlay network. Every neighbor then forwards the query to all of its neighbor peers. The procedure continues until the query reaches a specific depth of search limit (counted by Time-To-Live counter). Upon receiving a flood query, a peer that has a copy of the desired content sends a 'query hit response' to the peer that originated the query, which is an indication of having the content. The response is sent on the reverse path of the query, using pre-existing TCP connections. The peer that originated the query then selects one peer from the responded peers, and downloads the desired content through a direct TCP connection from the selected peer. Although Gnutella design is simple, highly decentralized, and does not require peers to maintain information related to location of contents, it is often criticized for its non-scalability. This is because query traffic can grow linearly with the total number of queries, which in turn grows with the system size. In addition, another drawback of the protocol is that a peer that originated the query may not find the desired content especially if the content is rare.

A hybrid unstructured P2P network allows the existence of infrastructure nodes, often referred to as *super-peers* (or *super-nodes* or *overlay nodes*). This creates a hierarchical overlay network that addresses the scaling problems on pure unstructured P2P networks such as Gnutella. A peer in such network can typically change roles over time. For example, a regular peer can become a super-peer that takes part in coordinating the P2P network structure. KaZaA (http://www.kazaa.com/), which is based on the FastTrack (http://www.fasttrack.nu/) protocol, is an example of a hybrid unstructured P2P network. This network uses specially designated super-peers with high bandwidth, disk space and processing power. When a peer joins the network, it is assigned to a super-peer. The peer then informs its super-peer about the content that it will share. The super-peer facilitates the search by maintaining a database that maps content to peers, and tracks only the content of its assigned peers. Similar to the centralized design, the super-peer plays the role of a directory server, although only to its assigned peers. The super-peers together create a structured overlay of super-peers, which makes search for content more efficient. A query in this network is routed to a super-peer. Then, as in the decentralized design, the query is flooded in the overlay super-peer network. The super-peer then responds to the peer that originated the query with a list of peers having the content. The hybrid networks are no longer dedicated to a single server, since the database is distributed among the super-peers. Moreover, the size of the database is relatively small, since each super-peer tracks only the contents of its assigned peers. However, the drawback of this approach is that it is considerably complicated, and requires non-trivial maintenance of the overlay network. Moreover, the fact that super-peers may have more responsibilities than ordinary peers can result in a bottleneck.

3. P2P-BASED FILE SHARING: BITTORRENT SYSTEMS

BitTorrent is a popular peer-to-peer file sharing protocol that was created by Cohen (2003). BitTorrent has been shown to scale well with large number of participating end hosts. Ipoque (http://www.ipoque.com/) measurements for years 2008-2009 show that BitTorrent is the dominant protocol in the Internet, and that it accounted for approximately 20-57% of all Internet traffic depending on the geographical location.

3.1 System Description

BitTorrent is a centralized unstructured system, which consists of two interacting units:

1) A "control" level describes control methods such as the required file sharing preparation, which takes place prior to sharing the actual content, and the coordination among end hosts, which is performed by a central entity during the downloading process.
2) A "reciprocation" level describes the actual data exchange among the end hosts.

These two design levels are described in detail next.

3.1.1 The BitTorrent System Design – Control Level

The BitTorrent content distribution system consists of the following components:

- Data content
- An original content provider
- The metainfo file
- The tracker
- The end hosts or peers or clients

A *torrent*, or *swarm*, is a collection of end hosts (or peers) participating in the download of ***content***, where content may refer to one (e.g. the Linux operating system) or multiple (e.g. several video or audio) files. ***The tracker*** is a server that coordinates and assists the peers in the swarm. It maintains the list of peers that are currently in the swarm, as well as statistics about the peers. The tracker listens on a BitTorrent TCP port for coming client requests. While the default BitTorrent port is port 6969, several trackers may use different ports.

Prior to the content distribution, ***a content provider*** divides the content into multiple *pieces*, where each piece is typically 256KB. Each piece is further divided into multiple *subpieces* with a typical size of 16KB. The content provider then creates *a* ***metainfo file***. The metainfo file contains information that is necessary for initiating and maintaining the download process. For example, the metainfo file contains the URL of the tracker, the name of the data file (files), the length of the data file (files), and the length of a piece. The metainfo file may also contain information related to multiple data files, and optional information such as creation date, author's comments, name and version of the .torrent creator, etc. The metainfo file also contains a special string, which is a concatenation of 20-byte encoded hash values. Each value is a SHA-1 hash of a piece at the corresponding data content, which is used for data integrity.

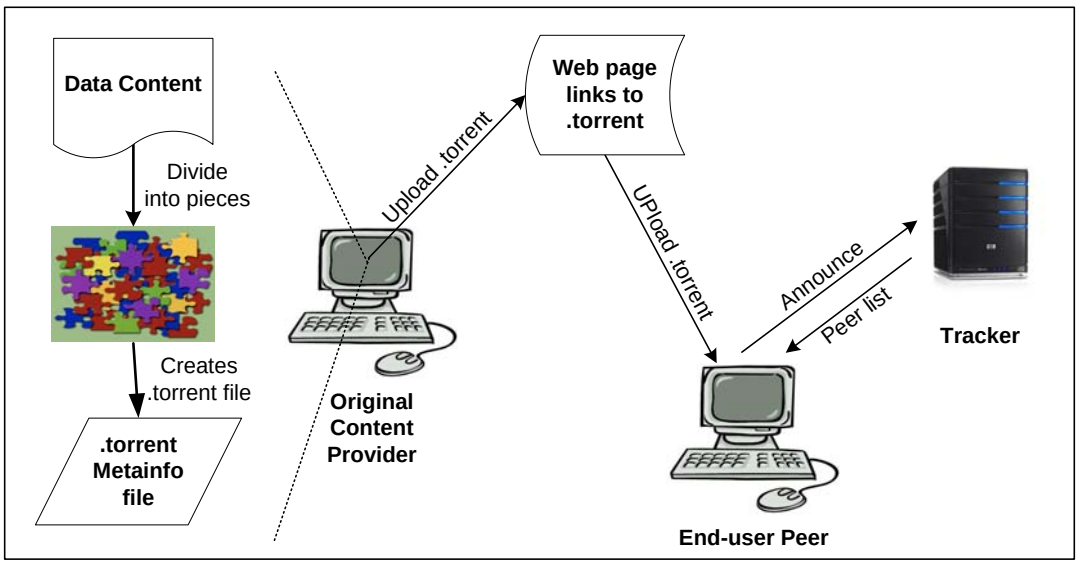

Figure 1: BitTorrent System - Prior to File Sharing

The metainfo file is usually uploaded to a website, thereby making it accessible to peers that are willing to download content by joining the swarm.

A peer that is willing to join the swarm first retrieves the out-of-band metainfo file. Then, this peer contacts the tracker by sending an "announce" HTTP GET request. The request in general may include necessary information such as the total amount uploaded, the total amount downloaded, the number of bytes the peer still has to download, an event description such as *started* – if it is the first request to the tracker, *completed* — if the peer shut down gracefully, *stopped* – if the download completed. This information helps the tracker keep overall statistics about the torrent (e.g., number of seeds, number of leechers, life time of a seed, etc). The tracker responds back with a "text/plain" document, which includes a randomly selected set of peers that are currently online. A typical size of a peer set is 50. The random peer set may include both *seeds* and *leechers*. Seeds are the peers who already have the entire content and are sharing it with others. Leechers are the peers who are still in the process of downloading (i.e. they do not possess the entire file). The new peer can then initiate new connections with the peers in the swarm and start to exchange data content pieces. The maximum number of connections that a peer can open is limited in BitTorrent to 80 in order to avoid performance degradation due to competition among concurrent TCP flows. In addition, the new peer is also limited to establish a fixed number of outgoing connections, typically 40, in order to ensure that some connection slots are kept available for new peers that will join at a later time. Figure 1 portrays the preliminary steps that need to be performed before starting to distribute the data content.

A peer that has already begun downloading may contact the tracker and ask for more peers if its peer set falls below a given threshold, which is typically set to 20 peers. Moreover, usually there is a minimum interval between two consecutive peer requests to avoid overwhelming the tracker. In addition, peers contact the tracker periodically, typically once every 30 minutes, to indicate that they are still present in the network. If a peer does not contact the tracker for more than 45 minutes, the tracker assumes that the peer has left the system and will remove the peer from the torrent list.

3.1.2 The BitTorrent System Design – Reciprocation Level

The connection between two peers starts with a handshake message followed by control and data message exchanges. The control messages between peers in the swarm as well as data messages are transferred over the TCP protocol. The range of TCP ports, which is used by BitTorrent clients is 6881-6999. The connection between peers is symmetric and the control messages in both directions have the same format. The data can flow in either direction. The messages that are used in a connection between two peers are:

- Handshake: the "**handshake" message** is a required message that ensures connections from both sides and must be the first message that is sent by the peer.
- Bitfield: the "**bitfield**" **message** is an optional message and may only be sent after the handshaking sequence is completed, and before any other messages are sent. A peer can choose not to send this message if it has no pieces. Using a single bit for every piece, bits that are set indicate valid and available pieces, which can be shared with other peers, and bits that are cleared indicate missing pieces, which must be downloaded from other peers.
- Interested: an "**interested" message** is a notification that the sender is interested in some of the receiver's data pieces.
- Not-interested: a "**not-interested**" **message** is a notification that the sender is not interested in any of the receiver's data pieces.
- Choke: the term *choke* is commonly used in BitTorrent as a verb that describes a temporary refusal to upload. A "**choke**" **message** is a notification that the sender will not upload data to the receiver until *unchoking* happens.
- Unchoke: An "**unchoke**" **message** is a notification that the sender peer will upload data to the receiver if the receiver is interested in some of the sender's data pieces.
- Request: a "**request**" **message** is used to request a subpiece.
- Piece: a "**piece**" **message** is sent in response to a request message, and contains the requested subpiece.
- Have: a "**have**" **message** describes the index of a piece that has been downloaded and verified via the SHA-1 hash function.
- Keep-alive: Peers may close the TCP connection if they have not received any messages for a given period of time, generally 2 minutes. Thus, the "**keep-alive**" **message** is sent to keep the connection between two peers alive, if no message has been sent in a given period of time.
- Cancel: a "**cancel**" **message** is used to cancel subpiece requests. It is mostly sent towards the end of the download process (see more details in Section 3.3.3).

A peer *A* in the swarm maintains a 2-bits connection state for every associated peer *B* that it is connected to. The first bit is the choking/unchoking bit. The second bit is the interested/not-interested bit. The connection state is initialized to choked and not interested. Peer *B* transfers data to peer *A* only if the state of the connection with *A* is unchoked and interested. Peer *B* responds to peer *A*'s "request" messages with encapsulated subpieces in "piece" messages. After peer *A* finishes downloading a piece, it verifies that the piece is uncorrupted. It calculates the SHA-1 value of the downloaded piece and compares this value with the encrypted reference value

of the piece that is given in the metainfo file. Since the SHA-1 value is assumed to be unique, a corrupted piece's hash would not match the reference hash value. After verifying that the piece is uncorrupted, peer A announces that it has the piece to all of its associated peers using the "have" message.

Figure 2 shows an example of a possible message flow among peers that have an active connection in a BitTorrent overlay network. In the example, the connection is established after peer *A* sends a "handshake" message, and *B* responses with one as well. Then, peer *B* sends a "bitfield" message but peer *A* does not. Such a scenario might happen if *A* has no piece ready to be shared. Peer *B* sends a "not interested" message to *A,* and *A* sends a "choke" message to *B*. Thus, data will not flow from peer *A* to peer *B* until both messages are replaced. On the other hand, data does flow from peer *B* to peer *A* because peer *A* sends an "interested" message to peer *B* and peer *B* sends an "unchoked" message to peer *A*. Then, peer *A* requests subpieces of a particular piece and *B* responds with "piece" messages, uploading the requested subpieces. Once peer *A* obtains the entire piece and confirms the validity of the piece, it sends "have" messages to all the peers that it is connected to in the BitTorrent overlay network.

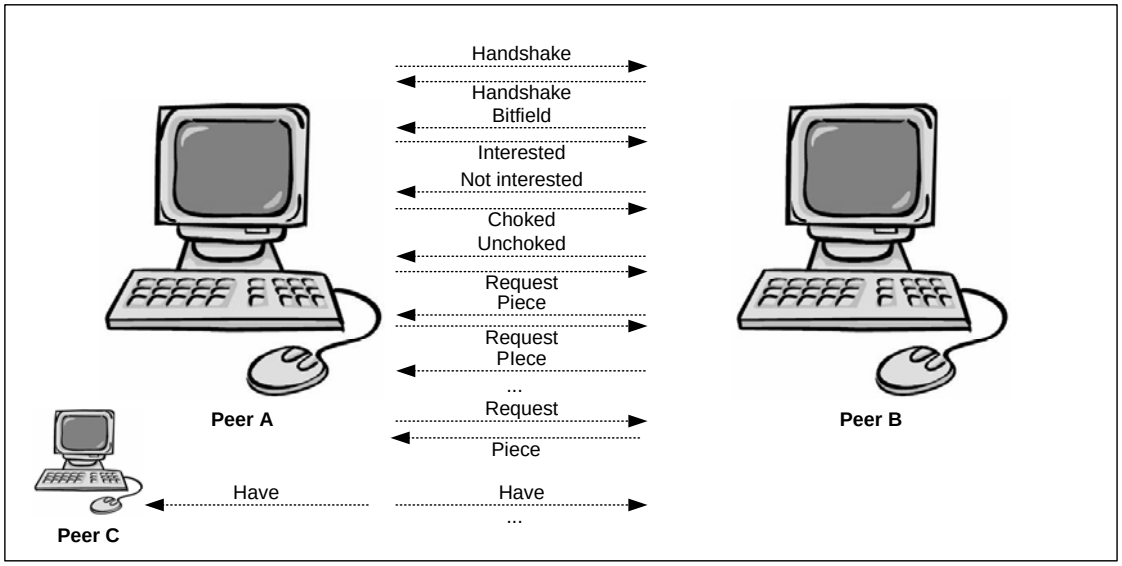

Figure 2: An illustrative example for a message flow among peers

3.2 Piece Selection Mechanisms

In the BitTorrent system, peers download the data content in a random order, unlike other protocols such as http or ftp, where an end host downloads a file from beginning to end. In order to facilitate such a downloading process, when a BitTorrent application is activated in a peer, the peer first allocates space for the entire content. Then, the peer tracks the pieces that each of its associated peers possess. A peer is able to identify what pieces its associated peers have by exchanging "bitfield" messages upon establishing new connections and by tracking the "have" messages that its associated peers send after downloading and verifying pieces. In this way, a peer is able to select a particular piece to download from a particular associated peer.

The piece selection mechanism is fundamental in achieving efficient P2P networks. A poor selection strategy can lead to an inability to download, e.g., when a peer is not interested in any of the pieces its associated peers have to offer, and vice versa, it can lead to the inability to upload,

e.g., when all associated peers are not interested in the pieces that a peer has to offer. More generally, it can prevent the peer selection mechanism from reaching an optimal system capacity (Legout et al., 2006). BitTorrent applies the strict priority policy for subpiece selection. Once the first subpiece of a piece is requested, the strict priority policy prioritizes subpieces that are part of the same piece. This ensures that a complete piece is downloaded as quickly as possible.

The piece selection mechanism in BitTorrent is composed of three different algorithms that are applied in different stages of the downloading process. The three algorithms are Random Piece First, Rarest Piece First, and End Game.

3.2.1 Rarest Piece First Selection

The rarest piece is the piece that has the least amount of copies in the peer set. For every piece, the peer maintains a counter of the number of copies that exists in its peer set. A peer, which runs the rarest piece first selection algorithm, selects the rarest missing piece as the next piece to download. If there are multiple equally-rare missing pieces, then the peer chooses at random to download one of the rarest pieces. A leecher that uses the rarest piece first algorithm will:

1. Upload pieces that many of the associated peers are interested in, such that uploading can be performed when needed.
2. Increase the likelihood that peers will offer pieces through the entire downloading process by leaving pieces that are more common to a later download.
3. When downloading from a seed, a leecher downloads new pieces first, where new pieces are those pieces that no leecher has. This is crucial, especially when the system has a single seed that may eventually be taken down, since this can lead to the risk that a particular piece will no longer be available. This is also important when the seeds in the system are slower than the leechers in the system. In this case, a redundant download wastes the opportunity of a seed to upload new pieces to associated peers with faster uploader speeds.

In (Legout et al., 2006), the authors studied the efficiency of the rarest piece first selection algorithm in BitTorrent. More specifically, they evaluated the efficiency of the rarest piece first selection strategy by characterizing the entropy of the system, with peer availability. They defined peer availability as the ratio of time that a peer is interested in its associated peer. They showed that the rarest piece first strategy can achieve a close to ideal entropy, when each leecher is almost always interested in all other leechers.

3.2.2 Random Piece First Selection

The download time of a random piece will be shorter on average than the download time of the rarest piece. A piece that is chosen at random is likely to be more replicated than the rarest piece, and thus, its download time will be shorter on average by downloading simultaneously from more peers. Despite this, the download time of complete piece may not affect the performance of a peer that uses the rarest-first piece selection strategy if the peer has other complete pieces to share. However at the beginning of the downloading process, a leecher has no pieces to share, and thus, the leecher should download pieces faster than in the rarest-first piece selection strategy, as it is important for a new peer to obtain some complete pieces and to start reciprocate pieces. Hence, at the beginning of the process the peer selects a piece to download at random, while applying the random piece first selection algorithm. Once the peer downloads *C*

pieces that are ready to be shared (*C* is a constant that may vary in different BitTorrent client implementations), the leecher switches to the rarest piece first selection algorithm.

3.2.3 End Game Piece Selection

The end game piece selection algorithm is performed after a peer has requested all the subpieces of the content. In this phase, a peer sends a request to all of its associated peers for all of the pending subpieces (i.e., those subpieces that have not been received yet). This step is performed in order to avoid potential delays at the end of the content download, which can occur if a request has been sent to a peer having a very slow upload rate instead of a peer having a fast upload rate. Since multiple requests for the same subpieces are sent out, once a subpiece is downloaded in the end game phase, the peer sends "cancel" messages to its associated peers so they do not waste upload bandwidth by sending redundant data. The end game is performed at the very end of the process, and thus, it may have only a small impact on the downloading process.

3.3 Peer Selection Mechanisms

In BitTorrent, peers download from whom they can, and upload simultaneously to a constant number of peers. The number of associated peers, which a peer uploads to, is limited in order to avoid sending data over many connections at once, which may result in poor TCP congestion control behavior. Thus, peers need to make decisions on which peers to unchoke. The default number of peers to unchoke (*unchoke slots*) is four. However, this number may increase unless a peer's upload bandwidth is saturated. A peer independently makes the decision regarding whom to unchoke and whom to choke, in every *unchoke period* which is typically ten seconds. The peer uploads to unchoked peers for the duration of the unchoke period.

The peer selection mechanism, which is also referred to as the choking mechanism, can affect the performance of the system. A good choking mechanism should:

1) Motivate peers to contribute and upload data to the network,
2) Utilize all available resources,
3) Be robust against free-riding behaviors where peers only download and do not upload.

In BitTorrent, the peer selection (choking) mechanism is applied differently to peers that are leechers and those that are seeds.

3.3.1 Leecher's Peer Selection Mechanism

The leecher's peer selection mechanism has two parts: The TFT mechanism and the optimistic unchoke mechanism.

3.3.1.1 The TFT Mechanism

In the TFT peer selection mechanism, a leecher decides to unchoke peers from which it currently downloads data. It chooses the peers who have the highest upload rate. The idea of TFT is to have several connections that actively transfer data in both directions at any time. In order to avoid wasting of resources due to rapidly choking and unchoking peers, the designer of the protocol sets the rechoke period to 10 seconds, claiming that "Ten seconds is a long enough period of time for TCP to ramp up new transfers to their full capacity" (Cohen, 2003).

3.3.1.2 The Optimistic Unchoke Mechanism

BitTorrent applies the optimistic unchoke mechanism in parallel with the TFT mechanism. The goals of the optimistic unchoke mechanism are: 1) to enable a continuous discovery of better peers to reciprocate with, 2) to bootstrap new leechers that do not have any content pieces to download some data and start reciprocate pieces with others. The optimistic unchoke mechanism chooses to unchoke a peer randomly regardless of its current upload rate. Optimistic unchoke is rotated every optimistic unchoke period, when an optimistically unchoked peer is unchoked for the entire optimistic unchoke period. The designer of the protocol chose the optimistic unchoke duration to be 30 seconds, because 30 seconds is enough time for the upload to get to full capacity, for the download to reciprocate, and finally for the download to get to full capacity. Optimistic unchoke is typically applied on a single unchoke slot while TFT is applied on the rest of the unchoke slots.

3.3.1.3 Anti-snubbing

If a peer has received no data from a particular peer for a certain period of time, typically 60 seconds, it marks the particular peer's connection as snubbed. A peer does not upload to an associated snubbed peer through the TFT peer selection mechanism. This may result in more than one simultaneous optimistic unchoke, when the peer is choked by many of its associated peers. In such a case, the peer may experience poor download rates until the optimistic unchoke finds better peers. Thus, increasing number of optimistic unchokes in this scenario is important.

3.3.2 Seed's Peer Selection Mechanism

Seeds, which do not need to download any pieces, follow a different choking mechanism than the leechers. The most common mechanism is based on a round-robin mechanism, which strives to distribute data uniformly.

3.3.3 Modeling the Peer Selection Mechanism

Many researchers studied the choking mechanism in BitTorrent by suggesting mathematical and game theoretical models. Qiu & Srikant (2004) studied a fluid analytical model of BitTorrent systems. They analytically studied the choking mechanism and investigated how it affects the peer performance. They showed that the optimistic unchoke mechanism may allow free-riding. Fan et al. (2006) characterized the design space of BitTorrent-like protocols capturing the fundamental tradeoff between performance and fairness. Other works such as (Izhak-Ratzin, 2009; Izhak-Ratzin et al., 2009; Neglia et al., 2007) model the choking mechanisms in BitTorrent as games with strategic peers. Massoulié & VojnoviĆ (2005) introduced a probabilistic model of coupon replication systems, and they argued that performance of file sharing system such as BitTorrent does not depend critically on altruistic behavior or on piece selection strategy (e.g., the rarest first algorithm). Levin et al. (2008) presented an auction based model of the peer selection mechanism. They claimed that the insight behind their model is that BitTorrent uses an auction model to decide which peers to unchoke and not to tit-for-tat as widely believed.

3.4 Limitations of BitTorrent Systems

In addition to the analytical models of BitTorrent, some measurement studies have been performed (Guo et al, 2005; Izal et al., 2004; Pouwelse et al., 2005; Piatek et al., 2007; Piatek et al., 2008). Most of these studies were performed on peers that were connected to public torrents. These studies provide interesting results about the overall behavior of deployed BitTorrent

systems. The studies also pinpoint several limitations of the BitTorrent protocol. In this section, we discuss two of the limitations of BitTorrent protocol that were shown in these studies:

1) The feasibility of free-riding,
2) The lack of fairness.

3.4.1 Free-Riding in BitTorrent Systems

Free riders are the peers who attempt to circumvent the protocol mechanism and download data without uploading data to other peers in the network. Researchers have argued that free-riding in BitTorrent is feasible through the optimistic unchoke mechanism and through seeds.

The first to pinpoint that effective free-riding in BitTorrent is feasible was Shneidman et al. (2004). They briefly described a scenario in which peers can attack the tracker while they exploit involved leechers by lying about the pieces they have. Jun et al. (2005) also argued that free-riding is feasible in BitTorrent by investigating the incentives in BitTorrent choking mechanism. Liogkas et al. (2006) showed that free-riding in BitTorrent is feasible by implementing three selfish BitTorrent exploits that allow free-riders to achieve high download rates and evaluate their effectiveness under specific circumstances. Locher et al. (2006) extended these results and presented the BitThief, a free-riding client that combines several attacks. They demonstrated that free-riding is feasible even in the absence of seeds. More recently, Sirivianos et al. (2007) evaluated an exploit based on maintaining a larger-than-normal view of the system, which affords free-riders a much higher probability of receiving data from seeds and optimistic unchokes. They argued that the large view exploit is effective and has the potential for wide adoption. Wide adoption of free-riding strategy can result in a "tragedy of the commons", where overall performance in the system will decrease.

Several works have attacked the ability of free-riders to download data from seeds without uploading in return, intending to considerably hurt free-riders' performance. Locher et al. (2007) proposed a source-coding scheme. Seeds in this scheme only upload a fixed number of content pieces to each leecher they connect to, thereby placing a hard limit on the data that free-riders can obtain in this manner. Chow et al. (2008) presented an alternative modified seed unchoking algorithm that gives preference to leechers that are either at the beginning or the end of their download.

Complementary to these approaches that modify the seed strategy, other works suggested to replace or limit the optimistic unchokes. Izhak-Ratzin et al. (2009) suggested a team mechanism that limits the optimistic unchokes as the collaboration with peers having similar upload rate increases. A foresighted resource reciprocation mechanism was suggested in (Izhak-Ratzin et al., 2009b) to replace the choking mechanism in BitTorrent. Finally, reputation systems such as (Buchegger & Le Boudec, 2004; Xiong & Liu, 2004; Yang et al., 2005) that use a peers' reputation history to make the choking decision were also suggested in order to help limit the ability to free-ride.

3.4.2 Lack of Fairness in BitTorrent Systems

Fairness in BitTorrent is commonly defined as "receive as much as they give." Fairness among peers participating in content distribution encourages peers to actively collaborate in disseminating content. Thus, fairness is an important factor, which can lead to improved system performance. However, research studies, such as (Guo et al., 2005; Piatek et al., 2007; Bharambe et al., 2006; Legout et al., 2007), show that BitTorrent does not provide fair resource reciprocation, especially in node populations with heterogeneous upload bandwidths. Two of the

mechanisms that contribute to the lack of fairness are the TFT and optimistic unchoke mechanisms, which are used for peer selection in BitTorrent.

The TFT mechanism is based on a short-term history, i.e., upload decisions are made based on the most recent observations of resource reciprocation. Thus, a peer can follow the TFT policy only if it continuously uploads pieces of a particular file and as long as it receives pieces of interest in return. However, this is not always possible because peers may not have any pieces that other peers are interested in, regardless of their willingness to cooperate (Piatek et al., 2008). This behavior is still perceived as a lack of cooperation.

The impact of optimistic unchokes on fairness in the BitTorrent system is shown in Figure 3 [the Figure is taken from (Izhak-Ratzin, 2009)], which shows the impact of optimistic unchokes on the expected download rate as a function of the peer's upload rate. The peer upload rate distribution is based on observed bandwidth distribution given in (Piatek et al., 2007). It is assumed that one unchoke slot is used by the optimistic unchoke mechanism, and that the regular unchoke mechanism works perfectly in terms of fairness, i.e., the download rate and the upload rate through TFT unchokes are equal. Clearly, we can see that the sub-linear behavior of the expected download rate leads to unfairness for high capacity leechers that are forced to interact with low-capacity peers, and low-capacity leechers that benefit from this unfairness. Note that in the observed bandwidth distribution, the majority of the leechers are low-capacity leechers with 88% of the leechers having less than 300KB/s upload capacity.

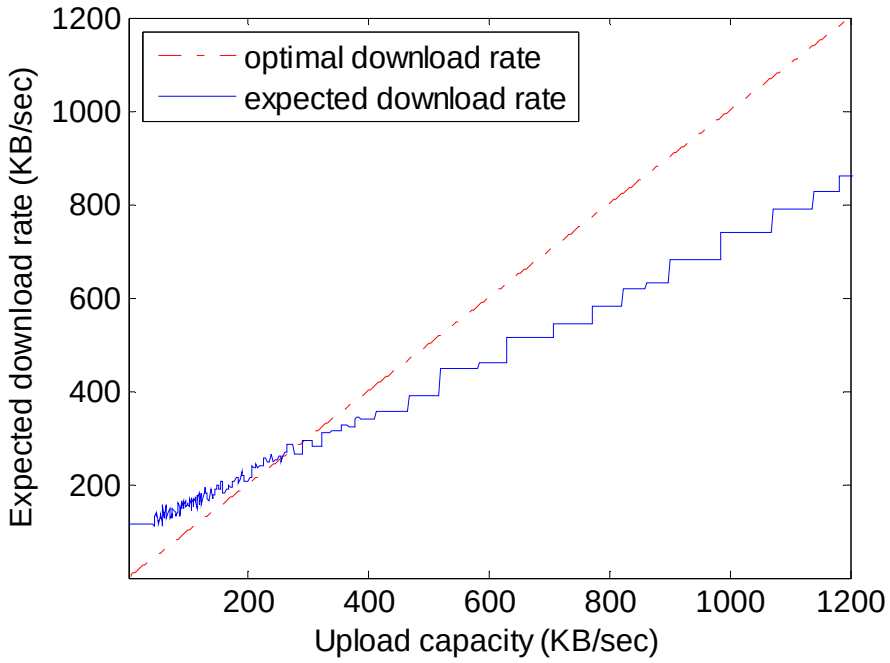

Figure 3: Upload and download in BitTorrent network

Finally, the number of unchoke slots may also lead to a lack of fairness in BitTorrent. The number of unchoke slots that a leecher uses for regular unchoke is a function of the leecher's ability to fully utilize its upload capacity. This again may lead to unfairness, since typically, in real torrents, the download capacity of a leecher may be greater than the upload capacity. Thus,

high capacity leechers may upload in full capacity, but not be able to download as much, due to upload constraints of the downloading leechers and limited number of unchoke slots.

Fairness in BitTorrent systems has been largely discussed in the literature. Guo et.al. (2005) performed extensive measurements of real torrents and pinpointed several BitTorrent limitations including lack of fairness. Bharambe et al. (2006) utilized a discrete event simulator to evaluate the impact of BitTorrent's mechanisms such as the peer selection mechanism, and observed that rate-based TFT incentives cannot guarantee fairness. They suggested a block-based TFT policy to improve fairness. Legout et al. (2007) studied clustering of peers having similar upload bandwidth. They observed that when the seed is underprovisioned, all peers tend to complete their downloads approximately at the same time, regardless of their upload rates. Moreover, high-capacity peers assist the seed to disseminate data to low-capacity peers. Piatek et al. (2007), observed through extensive measurement on real torrents the presence of significant altruism, where peers make contributions that do not directly improve their performance. They proposed the BitTyrant client, which adopts a new peer selection mechanism that reallocates upload bandwidth to maximize peers' download rates. Izhak-Ratzin (2009) identified the potential of there being a significant difference between a leecher's upload and download rates and proposed the Buddy protocol that matches peers with similar bandwidth. Other researchers acknowledged the importance of cooperation incentives through additional reputation mechanism. Anagnostakis & Greenwald (2004) extended BitTorrent's incentives to $n - way$ exchange among rings of peers, providing incentive to cooperate. Later, Lian et al., (2006) proposed multi-level TFT incentives as a hybrid between private and shared history schemes. More recently, Piatek et al., (2008) proposed a one-hop reputation system, in which peers that are not interested in the current available content perform data exchanges for the assurance of future payback.

4. P2P-BASED MEDIA STREAMING

4.1 Challenges and Requirements for Media Streaming

As discussed in Section 3, several protocols and incentive mechanisms have been developed for efficient data dissemination over P2P networks. However, they may only provide a limited performance for media streaming, because media streaming needs much more consideration such as real-time and high bandwidth requirements than general data dissemination. Extensive studies on video transmission over IP multicast have discussed in the literature (e.g., Liu et al., 2003; van der Schar & Chou, 2007), and various P2P based overlay multicast systems have also been recently proposed (e.g., Zhang et al., 2005; Liang et al., 2009; Lin et al., 2009). Unlike proxy-assisted overlay multicast systems, where several proxies are coordinated and placed such that an efficient overlay can be constructed (Zhuang et al., 2001; Guo et al., 2004; Hefeeda et al, 2004; Shi & Turner, 2002), the P2P based systems do not need such dedicated nodes and coordination. Rather, these systems consist of autonomous and often self-interested nodes that self-organize into groups, exchange information, and share their data.

Designing efficient protocols and mechanisms for the media streaming applications over P2P networks is challenging, because:

- the real-time and continuous delivery of content to a large number of participants should be supported in dynamically changing network conditions and groups of interacting peers, and
- a higher bandwidth is required in general.

Therefore, the media streaming applications require different design rules and approaches that consider such requirements. For example, the objective of the file sharing applications such as BitTorrent is to *completely* download the content data as fast as possible. Therefore, the timely downloading of data objects is not critical. Rather, downloading data objects that can be rarely found is more important. However, media streaming applications should consider stringent real-time constraints. Thus, a mechanism that enables peers to download data objects that meet the playback deadline is required, such that media streaming is uninterrupted. Second, each data object has an equal importance for file sharing P2P applications, as all the data objects should be completely downloaded. However, for media streaming applications, each data object has different importance based on the coding structure, playback deadline, etc. Thus, data objects that have higher quality impact or that have stringent playback deadline can have higher priority. Therefore, packet prioritization and scheduling mechanisms should be developed in the media streaming applications over P2P networks. Finally, a mechanism that provides graceful quality degradation also needs to be implemented, which enables adaptive and flexible media streaming in dynamic networks where heterogeneous peers having different bandwidth interact with each other.

In order to address these challenges, several solutions for media streaming have been proposed in the literature (see e.g., Vlavianos et al., 2006; Zhang et al., 2006; Banerjee et al., 2002; Castro et al., 2003; Deshpande et al., 2001; Trans et al., 2004; Heffeeda et al., 2003; Kostic et al., 2003; Pai et al., 2005; Padmanabhan et al., 2003; Tian et al., 2005; Venkataraman et al., 2006; Zhang et al., 2005b). The implementation of these solutions varies depending on their focuses, which can be broadly classified as 1) peer clustering strategies, 2) overlay constructions and 3) incentive strategies on resource reciprocations.

4.2 Peer Clustering Strategies

Several approaches for efficient and reliable delivery of media streams over P2P networks have focused on developing peer clustering strategies. As discussed, the media content is directly shared by peers interacting with each other. Hence, it is important to select good peers and form clusters (or groups) with them. For example, Purandare & Guha (2007) proposed that peers can form groups based on alliance formation process, where this process considers the time and resource constraints. In (Venkataraman et al., 2006), the participating nodes form a graph, where the degree of the graph is determined proportionally to its desired transmission load. In addition, several other strategies such as SpreadIt (Deshpande et al., 2001), NICE (Banerjee et al., 2002), and ZIGZAG (Banerjee et al., 2002) have been proposed, where these algorithms perform hierarchically from clusters such that a minimized transmission delay can be achieved.

In order to enhance the efficiency and robustness of the media delivery, various coding structures and packetizing strategies can also be implemented in conjunction with the abovementioned clustering strategies. For instance, Multiple Description Coding, or MDC (Goyal, 2001), is an illustrative example of coding structure for P2P based media delivery, and its deployment for media delivery has been studied in e.g., Padmanabhan et al., 2003; Padmanabhan

et al., 2002b; Chu et al., 2004; Li et al., 2004. Alternatively, network coding techniques are also deployed for efficient media delivery (e.g., Nguyen et al., 2007; Wang & Li, 2007; Wang & Li, 2007b; Gkantsidis & Rodriguez, 2005).

4.3 Overlay Constructions: Tree-based and Data-driven Approaches

4.3.1 Tree-based Approach: Description

One of design approaches of data dissemination over P2P networks is based on tree structure. In this approach, peers are organized into tree structure, where each peer is participating in disseminating data. In general, peers (or nodes) in the tree are hierarchically organized (i.e., parent – children node relationship) and data is disseminated by typically push-based approach, where the data is forwarded from parent nodes to children nodes. Because the data is disseminated in a structured way, it is important to

1) design optimal tree constructions that provide an efficient performance to each peer,
2) develop approaches for tree construction that are robust to nodes' (unexpected) joining and leaving trees and provide easy tree repair, and
3) construct non-cyclic trees, i.e., trees that have no loops.

Note that one of the major concerns in this approach is the failure of nodes. If a node cannot appropriately perform forwarding data due to node crash or node malfunction, its children nodes in the tree receive no data packets, resulting in significantly poor performance. Moreover, since a significantly large number of nodes are located in the end of the trees, an efficient utilization of outgoing bandwidth of them is challenging. In order to address these issues, several approaches have been proposed (e.g., Chu et al., 2000; Deshpande et al., 2001; Padmanabhan & Sripanidkulchai, 2002; Padmanabhan et al., 2002b; Padmanabhan et al., 2003; Li et al., 2004). In order to successfully implement the tree-based approaches, the following mechanisms have been proposed.

4.3.1.1 Group Management Mechanism

To maintain a robust and efficient tree, each peer (or node) may need the information about a set of other nodes and the path from a source. Peers in a tree keep exchanging the information that they have, which may include the group member information, and thus, they can have updated information about the tree. The information exchange can be implemented based on gossip-like protocols (e.g., End System Multicast, ESM, (Chu et al., 2000)). Peers already participating in a tree can switch their parent nodes if the performance (e.g., download rates) achieved in a current position is not satisfied. If a peer newly joins the network (i.e., a tree), it contacts the server, and obtains the (partial) information about the existing peers and the corresponding paths. Then, the peer can determine its parent nodes. Finally, the records of peers that left the tree are simply removed from the group member list.

4.3.1.2 Parent Selection Algorithm

As discussed, the way of selecting parents is important for both peers in a tree and newly joining peers. In order to select the parent nodes, a peer can contact a node that is randomly chosen from

its group member list. Then, it retrieves the information that includes the currently achieved performance, the number of children nodes, etc. from the contacted node, while estimating the round-trip time. The peer evaluates the contacted node only if the contacted node is not included in its children nodes, and the contacted node has not exceeded its children nodes limit. The evaluation of the contacted node can include the achievable performance and induced delay. Several criteria can be deployed in order to evaluate the node based on goals of applications. For file sharing applications, the performance (i.e., bandwidth) can be one of the most important criteria. For media streaming applications, however, the delay may be the most important criterion for parent selection. Alternatively, the obtained media quality can also be a criterion for parent selection (Park and van der Schaar, 2009d).

4.3.2 Data-driven Approach

Alternative approaches for data dissemination over P2P networks are based on data-driven structures (Pai et al., 2005; Zhang et al., 2005). Unlike the tree-based structures, where well-designed trees need to be constructed and continuously maintained for efficient data dissemination in dynamically changing networks, data-driven approaches focus on

1) how to efficiently exchange information about the data availability,
2) how to form groups of peers and share their data, and
3) how to develop group management algorithms and scheduling algorithms.

In data-driven approaches, a newly generated data of a peer can be forwarded to its randomly selected neighbor nodes, and the neighbor nodes also forward the data to their neighbor nodes, which can be continued. Finally, the messages can be distributed to all the nodes. This approach can be implemented based on gossip algorithms (Eugster et al., 2004). While such random *push* approaches enable the P2P systems to be robust to the random node failures, which is one of challenges for tree-based approach, these approaches may result in significant redundancy for receiving nodes (Gkantsidis et al., 2004).

In order to reduce the redundancy, *pull-based* techniques are proposed (Cohen, 2003; Pai et al., 2005; Zhang et al., 2005). In pull-based techniques, peers self-organize into groups and each peer requests necessary data packets from other peers in its group. Thus, each peer can avoid receiving redundant packets from its neighbor peers. The metadata that may include information about the peers can be downloaded from a server (e.g., trackers in BitTorrent systems), and then, each peer forms a group (e.g., swarm) with the other randomly selected peers or joins an existing group. The groups can be maintained by periodically exchanging the information about data availability with the group members. This approach has been actually implemented in BitTorrent (Cohen, 2003), and has shown efficient performance for general file sharing. In this example, a source distributes its content to peer A and peer B, and then, they exchange the content with their group members. Note that peer C also has its own group (which is not shown in this example), which also includes peer A.

However, this approach cannot be directly deployed for media streaming applications that require explicit playback deadlines, unless appropriate scheduling algorithms are implemented. Thus, for efficient media streaming over P2P networks based on data-driven approaches, several approaches have been proposed in (Pai et al., 2005; Zhang et al., 2005), where they explicitly

implement scheduling algorithms. These mechanisms that support the data-driven approaches are discussed next.

4.3.2.1 Group Management Mechanism

One of the key mechanisms required for data-driven approaches is the *group management mechanism*, which enables peers to maintain their groups, such that they can continuously exchange necessary data packets among the peers. For newly joining peers, they can contact a server to obtain an initial set of group member candidates (e.g., End System Multicast (Chu et al., 2000), BitTorrent, CoolStreaming (Zhang et al., 2005)). Based on the initial information, each peer can form a group with partial set of peers or join an existing group. Then, each peer maintains its group by exchanging the information about the data availability. For example, CoolStreaming deploys an existing gossip membership protocol for exchanging the information about the data availability, which is referred to as membership message. The information is generated based on a Buffer Map (BM) that represents active data segments. More details about the BM are discussed next.

4.3.2.2 Prioritization and Scheduling Mechanisms

Unlike file sharing applications, it is critical for media streaming applications to deliver data segments timely and continuously. Thus, piece selection mechanism (i.e., the rarest first mechanism) implemented in BitTorrent systems can only provide limited performance, as it does not consider the priority of each data segment when it is downloaded. In order to explicitly consider the priority of data segments, scheduling mechanisms can be deployed. For example, in CoolStreaming, a sliding window (see Figure 4), or BM, is used, which consists of 120 segments that correspond to 120 seconds of media data (i.e., 1 segment includes media data of 1 second). In this illustration shown in Figure 4, the peer is focusing on downloading the segment that is not in the window (empty segment) with the highest priority at this playback point.

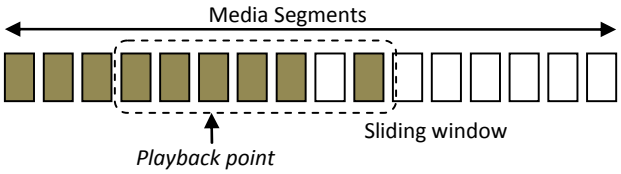

Figure 4. Sliding window in scheduling mechanisms

Thus, the scheduling algorithms should be designed while taking into account the playback deadline (i.e., priority) for each segment, and the available upload bandwidth of associated peers. This is because peers in a P2P system may be heterogeneous, having different upload bandwidth, etc. In CoolStreaming, a simple and heuristic algorithm has been proposed.

4.3.2.3 Group Update Mechanism

In general P2P systems, peers can leave the systems, and thus, leave their groups, at any time, which changes the groups that the peers have been associated. Thus, an efficient group update

mechanism needs to be deployed, such that peers can easily capture the group changes and efficiently respond to the changes. This may include an efficient rescheduling algorithm for data segment request, and the exchanged information about data availability that can be used. For instance, in CoolStreaming, the groups are periodically updated by randomly contacting peers in local membership list, and the rescheduling algorithms are performed based on exchanged BM information.

4.4 Incentives for Resource Reciprocation

While several approaches discussed in the previous sections provide efficient solutions to media streaming over P2P networks, they are designed based on an implicit assumption that peers are unconditionally cooperative and collaborative – they are willing to provide correct and true information, exchange and share their resource actively, by honestly following the pre-designed protocols. However, in reality, this is not always true. Rather, peers are often self-interested and are trying to maximize their own benefits (e.g., download rates, media quality, etc.) by deviating from the pre-determined rules in the systems. One of the representative examples for self-interested behaviors of peers is the *free-riding*, which is discussed in Section 3. The free-riders may not contribute their resources while downloading content from the other peers (data-driven approach), or they try to become leaf nodes by intentionally announcing that their upload bandwidth is poor. Moreover, it has been observed from several P2P based systems that the majority of uploading requests are concentrated on a small set of peers (Ali et al., 2006; Adar & Huberman, 2000), which may prevent potentially cooperative peers from actively contributing their resources.

One of the reasons why such behaviors that can result in significant degradation of both the system and individual peers' performance are allowed is the lack of well-designed *incentive mechanisms*. In order to provide incentives for each peer's contribution, simple incentive mechanisms have been developed based on TFT strategy for a general file sharing in BitTorrent-like protocols (e.g., Cohen, 2003; Legout et al., 2006; Piatek et al., 2007) or for a media streaming in BiToS (Vlavianos et al., 2006). Incentives can also be provided based on each peer's reputations, which can be managed in various ways (e.g., Credence (Walsh & Sirer, 2006), DARWIN (Jaramillo & Srikant, 2007), EigenTrust (Kamvar et al., 2003), PeerTrust (Xiong & Liu, 2004), DCRC and CORC (Gupta et al., 2003), Despotovic & Aberer, 2005). Finally, several game theoretic approaches have been adopted for providing incentives (see e.g., Chen et al., 2007; Fan et al., 2006; Zhang et al., 2007; Buragohain et al., 2003; Jun & Ahamad, 2005; Lai et al., 2003; Golle et al., 2001; Feldman et al., 2004; Park & van der Schaar, 2009; Park & van der Schaar, 2009c).

While simple incentive mechanisms have been developed and deployed in practice (e.g., TFT strategy in practical BitTorrent systems), it is still reported that existing incentive mechanisms do not efficiently prevent such self-interested behaviors in P2P systems. In the next section, we will discuss recently proposed game theoretic approaches for incentive mechanisms in detail, which model the resource reciprocation among self-interested peers as a stochastic game and find an optimal strategy that maximizes each peer's long term benefit in P2P systems

5. FORESIGHTED RESOURCE RECIPROCATION STRATEGIES

5.1 Challenges of Incentive Design – Cooperation and Competition

As discussed in Section 4, one possible solution for providing cooperation incentives is the TFT strategy, which has been actually implemented in practical file sharing P2P systems such as BitTorrent. The TFT strategy is deployed in BitTorrent as a choking algorithm, which effectively encourages peers' cooperation and penalizes peers' selfish behaviors (see Section 3.1.2 for more detail). While it has been shown that this strategy performs effectively in various P2P network scenarios, it may not completely eliminate selfish behaviors such as free-riding (Liogkas et al., 2007; Locher et al., 2006; Sirivianos et al., 2007). Moreover, the resource reciprocation based on TFT is *myopic*, because a peer unchokes some of its associated peers (i.e., leechers) that are *currently* uploading at the highest rates. Thus, the resource reciprocation decisions based on TFT may not provide optimal solutions that maximize the long-term rewards. Since peers in P2P networks are generally involved in repeated and long-term interactions, such myopic decisions on peer selection and bandwidth allocation can result in a sub-optimal performance for the involved peers. Finally, the TFT strategy in BitTorrent is based on the *equal* upload bandwidth distribution (Cohen, 2003; Legout et al., 2007) which is neither a fair nor an optimal resource allocation for heterogeneous content and diverse peers with different upload/download requirements (Park & van der Schaar, 2007).

To address these challenges, a foresighted resource reciprocation strategy has been recently proposed, which enables the peers to determine their resource reciprocation decisions that maximize a long-term utility. The interactions among the peers are modeled as a stochastic game, where they determine their resource distributions by considering the probabilistically changing future behaviors of their associated peers. The resource reciprocation process of a peer is modeled as a Markov Decision Process (MDP), and the foresighted resource reciprocation strategy is obtained by solving the MDP. It has been shown that the foresighted strategy provides improved performance compared to existing incentive mechanisms in BitTorrent or heuristic algorithms in several media streaming approaches (e.g., Lee et al., 2009).

5.2 Formalization of Resource Reciprocation as Stochastic Games

5.2.1 Group-based Resource Reciprocation

Resource reciprocation games in P2P networks are played by the peers interested in each other's media content. A resource reciprocation game is played in a *group*, where a group consists of a peer and its associated peers. The group-based resource reciprocation has been discussed in the literature, where a group could be formed of swarms (Cohen, 2003; Legout et al., 2006), partnerships (Zhang et al., 2005), or neighbors (Pai et al., 2005).

The group members associated with a peer i is denoted by C_i, where a peer $k \in C_i$ also has its own group C_k which includes peer i. Due to the dynamics introduced by peers joining, leaving, or switching P2P networks, the information about groups needs to be regularly (periodically) updated or it needs to be updated when group dynamics change (Cohen, 2003; Legout et al., 2006). This process is shown in Figure 5. In this diagram, the group dynamics change when group members are changed or their behavior are changed, etc. Moreover, resource reciprocation in Figure 5 may include several modules for its actual implementation, such as an estimation of the state transition probability and a level of accuracy for behavior estimation. These will be discussed in the following sections.

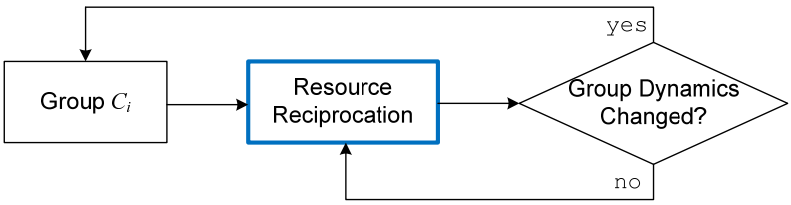

Figure 5. Block diagram for group-based resource reciprocation.

5.2.2 Modeling Resource Reciprocations as Stochastic Games

The resource reciprocation game in a group C_i consists of

- a finite set of players (i.e., peers): $C_i \cup \{i\}$
- for each peer $l \in C_i \cup \{i\}$, a nonempty set of actions: $\mathbf{A}_l$
- for each peer $l \in C_i \cup \{i\}$, a preference relation (i.e., utility function) of peer l: $U_l(\cdot)$.

To play the resource reciprocation game, a peer can deploy an MDP, defined as follows.

For a peer i, an MDP is a tuple $\langle \mathbf{S}_i, \mathbf{A}_i, P_i, R_i \rangle$, where $\mathbf{S}_i$ is the state space, $\mathbf{A}_i$ is the action space, $P_i : \mathbf{S}_i \times \mathbf{A}_i \times \mathbf{S}_i \rightarrow [0,1]$ is a state transition probability function that maps the state $s_i \in \mathbf{S}_i$ at time t, corresponding action $\mathbf{a}_i \in \mathbf{A}_i$ and the next state $s_{i'} \in \mathbf{S}_i$ at time $t+1$ to a real number between 0 and 1, and $R_i : \mathbf{S}_i \rightarrow \mathbb{R}$ is a reward function, where $R_i(s_i)$ is a reward derived in state $s_i \in \mathbf{S}_i$. This MDP-based resource reciprocation model and its variations in P2P networks have been discussed in (Park & van der Schaar, 2009; Park & van der Schaar, 2009b; Park & van der Schaar, 2009c). We review the details in the following.

5.2.1.1 State Space

A state of peer i represents the set of received resources from the peers in C_i, which is expressed as

$$\left\{ (x_{1i},\ldots,x_{N_{C_i}i}) \middle| 0 \le x_{ki} \le L_k, \forall k \in C_i \right\} \tag{1}$$

where x_{ki} denotes the provided resources (i.e., rate) by peer k in C_i and L_k represents the available maximum upload bandwidth of peer k. The total received rates of peer i in C_i is thus $\sum_{k \in C_i} x_{ki}$. Due to the continuity of x_{ki}, the cardinality of the set defined in (1) can be infinite. Hence, we assume that peer i has a function ψ_{ik} for peer k, which maps the received resource x_{ki} into one of n_{ik} discrete values, i.e., $\psi_{ik}(x_{ki}) = s_{ik} \in \{s_{ik}^1,\ldots,s_{ik}^{n_{ik}}\}$. These values are referred to as *state descriptions* in this chapter. Hence, the state space can be considered to be finite. The state space of peer i can be expressed as

$$\mathbf{S}_i = \left\{ s_i = (s_{i1},\ldots,s_{iN_{C_i}}) \middle| s_{ik} = \psi_{ik}(x_{ki}), k \in C_i \right\}, \quad (2)$$

where s_{ik}^l denotes the l th segment among n_{ki} segments that corresponds to the l th state description of peer i. For simplicity, we assume that each segment represents the uniformly divided total bandwidth, i.e., $\psi_{ik}(x_{ki}) = s_{ik}^l$ if $(l-1)\frac{L_k}{n_{ik}} \le x_{ki} < l\frac{L_k}{n_{ik}}$ for $1 \le l \le n_{ik}$.

5.2.1.2 Action Space

An action of peer i is its resource allocation to the peers in C_i. Hence, the action space of peer i can be expressed as

$$\mathbf{A}_i = \{\mathbf{a}_i = (a_{i1},\ldots,a_{iN_{C_i}}) \Big| 0 \le a_{ik} \le L_i,\ 1 \le k \le N_{C_i}, \textstyle\sum_{k\in C_i} a_{ik} \le L_i\}, \quad (3)$$

where $a_{ik} \in A_i$ denotes the allocated resources to peer k by peer i in C_i. Hence, peer i's action a_{ik} to peer k becomes peer k's received resources from peer i, i.e., $a_{ik} = x_{ik}$. To consider a finite action space, we assume that the available resources (i.e., upload bandwidth) of peers are decomposed into *unit*s of bandwidth (Jain et al., 2007). Thus, the actions represent the number of allocated units of bandwidth to the associated peers in their groups. We define the *resource reciprocation* as a pair $(\mathbf{a}_i, s_i) = \left((a_{i1},\ldots,a_{iN_{C_i}}),(s_{i1},\ldots,s_{iN_{C_i}})\right)$ comprising the peer i's action, a_{ik}, and the corresponding modeled resource reciprocation s_{ik}, which is determine as $s_{ik} = \psi_{ik}(x_{ki})$ for all $k \in C_i$.

Note that various scheduling schemes can be used in conjunction with the resource allocation (i.e., actions) deployed by peers in order to consider the different priorities of the different data segments (chunks). We assume that the chunks that have higher quality impact on average media quality have higher priority and are transmitted first when each peer takes its actions. However, other scheduling algorithms, such as the rarest first (Cohen, 2003; Legout et al., 2007) method for general file sharing applications or several scheduling methods proposed in e.g., (Zhang et al., 2005) for media streaming applications, can also be adopted. It is important to note that appropriate scheduling schemes need to be deployed in conjunction with our proposed resource reciprocation strategies, depending on the objectives of multimedia applications (e.g. maximizing achieved quality, minimizing the playback delay etc.).

5.2.1.3 State Transition Probability

A state transition probability represents the probability that by taking an action, a peer will transit into a new state. We assume that the state transition probability depends on the current state and the action taken by the peer, as peers decide their actions based on their currently received resources (i.e., state). Hence, given a state $s_i \in \mathbf{S}_i$ at time t, an action $\mathbf{a}_i \in \mathbf{A}_i$ of peer i can lead to another state $s_i^{'} \in \mathbf{S}_i$ at $t'(t' > t)$ with probability $P_{\mathbf{a}_i}(s_i, s_i^{'}) = \Pr(s_i^{'} \mid s_i, \mathbf{a}_i)$. Hence, for a state $s_i = (s_{i1},\ldots,s_{iN_{C_i}})$ of peer i in C_i, the probability that an action $\mathbf{a}_i$ leads the state transition from s_i to $s_i^{'}$ can be expressed as

$$P_{\mathbf{a}_i}(s_i, s_i^{'}) = \prod_{l=1}^{N_{C_i}} P_{a_{il}}(s_{il}, s_{il}^{'}) \quad (4)$$

The state transition probabilities of peers are identified based on the histories of past resource reciprocation. One approach that efficiently builds the state transition probability functions is discussed in (Park & van der Schaar, 2009).

5.2.1.4 Reward

Reward of a peer i represents the derived utility from its state s_i. The utility of peer i can be differently defined depending on the goals of applications. For example, in (Park & van der Schaar, 2009; Park & van der Schaar, 2009b), the reward $R_i(s_i)$ for a peer i in state s_i is the total received resources in C_i, defined as

$$R_i(s_i) = R_i(s_{i1},\ldots,s_{iN_{C_i}}) = \sum_{k\in C_i} r_i(s_{ik}) \tag{5}$$

where $r_i(s_{ik})$ is a random variable that represents the received resource in state s_{ik}. If the media quality is the ultimate goal of the application, the utility that corresponds to reward $R_i(s_i)$ of peer i in state s_i is $U_i\left(\sum_{k\in C_i} r_i(s_{ik})\right)$. Utility function $U_i(\cdot)$ of peer i downloading a demanded content from its peers at rate x_i is defined as

$$U_i(x_i) = \begin{cases} 0, & \text{if } x_i < R_i^{req}, \\ \rho_i \cdot Q_i(x_i), & \text{otherwise} \end{cases} \tag{6}$$

where R_i^{req} is the minimum rates to successfully decode the downloaded content and ρ_i is a constant representing the preference of peer i for the downloading content. The derived quality $Q_i(x_i)$ with downloading rate x_i is represented by a widely used quality measure, Peak Signal to Noise Ratio (PSNR), which is a non-decreasing and concave function of x_i for multimedia applications (van der Schaar & Chou, 2007). Alternatively, in (Park & van der Schaar, 2009c), the reward of a peer is defined as a guaranteed rates (and thus, guaranteed media quality) that can be achieved from its associated peers.

Since each peer in a P2P network that consists of N total peers can individually deploy the MDP, the resource reciprocation game in the network can be described by a tuple $(\mathcal{I},\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R})$, where $\mathcal{I}$ is the set of N peers, $\mathcal{S}$ is the set of state profiles of all peers, i.e., $\mathcal{S} = \mathbf{S}_1 \times \cdots \times \mathbf{S}_N$, and $\mathcal{A} = \mathbf{A}_1 \times \cdots \times \mathbf{A}_N$ denotes the set of action profiles. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is a state transition probability function that maps from the current state profile $s \in \mathcal{S}$, corresponding joint action $a \in \mathcal{A}$ and the next state profile $s' \in \mathcal{S}$, into a real number between 0 and 1, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^N$ is a reward function that maps an action profile $a \in \mathcal{A}$ and a state profile $s \in \mathcal{S}$ into the derived reward. Thus, we can focus on the resource reciprocation game in a group, as this resource game can be extended to the resource reciprocation game in a P2P network.

5.3 Foresighted Resource Reciprocation Strategy

The solution to the MDP described in Section 5.2 is represented by peer i's optimal policy π_i^*, which is a mapping from the states to optimal actions. The optimal policy can be obtained using well-known methods such as value iteration and policy iteration (Bertsekas, 1976). Hence, peer i can decide its actions based on the optimal policy π_i^*, i.e., $\pi_i^*(s_i) = \mathbf{a}_i$ for all $s_i \in \mathbf{S}_i$. Note that policy π_i^* enables peer i to make *foresighted* decisions on its resource reciprocation.

A conventional approach is *myopic* decision making. Myopic peers only focus on maximizing the immediate expected rewards, i.e., a myopic peer i takes its action $\mathbf{a}_i^*$ (i.e., upload bandwidth allocation) such that the action leads to the maximum immediate expected reward, i.e.,

$$\begin{aligned} \mathbf{a}_i^* = \arg\max_{\mathbf{a}_i \in \mathbf{A}_i} & \sum_{s_i^{(t+1)} \in S} P_{\mathbf{a}_i}\left(s_i^{(t)}, s_i^{(t+1)}\right) R\left(s_i^{(t+1)}\right) \\ \text{subject to} & \sum_{k \in C_i} a_{ik} \le L_i \end{aligned} . \tag{7}$$

Unlike the myopic peers, the foresighted peers take their actions considering the immediate expected reward as well as the future rewards. Since future rewards are generally considered to be worth less than the rewards received now (Watkins & Dayan, 1992), the foresighted peers try to maximize the cumulative discounted expected rewards. Hence, a foresighted peer i in state s_i at time t_c given a discount factor γ_i tries to maximize its cumulative discounted expected rewards, i.e.,

$$\text{maximize} \quad \sum_{t=t_c+1}^{\infty} \gamma_i^{(t-(t_c+1))} \cdot E\left[R(s_i^{(t)})\right] \tag{8}$$

where $R(s_i^{(t)}) = \sum_{l=1}^{N_{C_i}} r_i(s_{il})$ for $s_i^{(t)} = (s_{i1},\ldots,s_{iN_{C_i}})$. More precisely, the expression in (8) can be rewritten as

$$\begin{aligned} \text{maximize} \quad & \underbrace{\sum_{s_i^{(t+1)} \in S} P_{\mathbf{a}_i}(s_i^{(t)}, s_i^{(t+1)}) R(s_i^{(t+1)})}_{\text{immediate expected reward}} + \underbrace{\sum_{t'=t_c+2}^{\infty} \gamma_i^{(t'-(t_c+1))} \cdot \sum_{s_i^{(t')} \in S} P_{\mathbf{a}_i}(s_i^{(t')}, s_i^{(t'+1)}) R(s_i^{(t'+1)})}_{\text{discounted future expected reward}} \\ \text{subject to} \quad & \sum_{k \in C_i} a_{ik} \le L_i \end{aligned} \tag{9}$$

The discount factor γ_i in the considered P2P network can alternatively represent the validity of the expected future rewards, as the state transition probability can be affected by system dynamics such as peers' joining, switching, or leaving groups (Park & van der Schaar, 2009c). Hence, for example, if P2P network is in transient regime, a small value of discount factor is desirable, while a large value of discount factor can be used if the P2P network is in stationary regime (de Veciana & Yang, 2003). We note that the myopic decisions are a special case of the foresighted decisions when $\gamma_i = 0$.

Illustrative performances achieved based on different resource reciprocation strategies are shown in Figure 6. As discussed in this section, resource reciprocation based on the myopic strategy is aiming to maximize the immediate expected rewards (Eq. (8)). However, resource reciprocation based on the foresighted strategy targets on maximizing the cumulative (discounted) expected rewards (Eq. (9)). Figure 6 confirms this, as the rewards obtained by the actions of myopic policy are always higher (or equal) than the other policies for immediate rewards (left of Figure 6). However, a peer can achieve the highest cumulative discounted expected rewards based on the foresighted policy. Thus, the foresighted resource reciprocation strategy may be beneficial for peers, as they are generally interacting with each other in a long period of time. In (Lee et al., 2009), the foresighted resource reciprocation strategies in conjunction with scheduling algorithms have been deployed for media streaming over P2P, which shows improved performance.

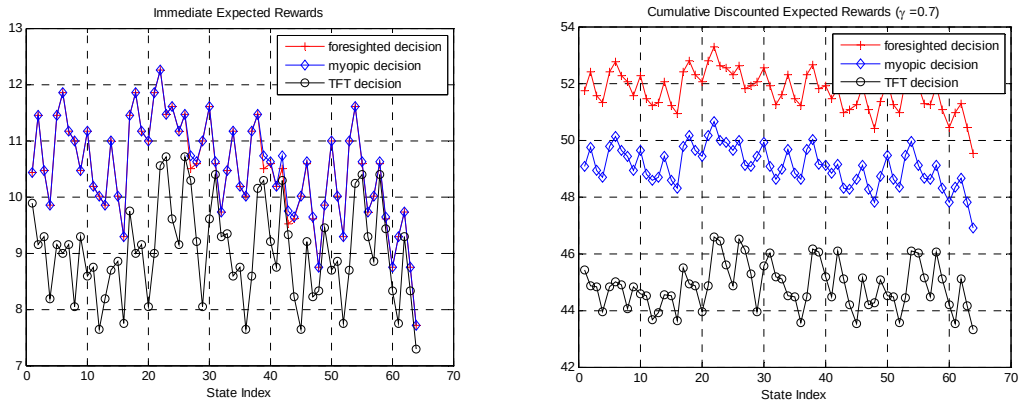

Figure 6. Expected rewards achieved by different resource reciprocation strategies

5.4 Bounded Rationality on Resource Reciprocation

Conventional stochastic games have been developed based on the implicit assumptions on players' rationality, where players have the abilities to collect and process relevant information, and select alternative actions among all possible actions. However, peers are often *boundedly rational* in practice (Simon, 1955; Haruvy, 1999). This is because perfectly rational decisions are often infeasible in practice due to memory and computational constraints. Due to the bounded rationality, peers may have incorrect beliefs on the other players' behavior and limited ability to analyze their environment. Therefore, it is essential to study the impact of the bounded rationality of peers on 1) the performance degradation of the proposed resource reciprocation strategy and 2) their repeated interactions (resource reciprocation) for practical implementations. We overview several studies, which investigate how the bounded rationality of peers can impact the peers' interactions and the corresponding performances.

5.4.1 Bounded Rationality: Attitude towards Resource Reciprocations

In (Park & van der Schaar, 2009), the bounded rationality is represented by the resource reciprocation *attitude* of peers, and its impact on resource reciprocation and the corresponding individual peers' performances is discussed.

Peers in the considered P2P networks are characterized based on their attitudes towards the resource reciprocation, which are pessimistic, neutral, or optimistic (Park & van der Schaar, 2009). These characteristics determine how peers can respond to their resource reciprocations. Let $\left(a_{ik}, s_{ik}\right)$ be recent resource reciprocation between peer i and peer k, which is a pair of peer i's action to peer k and peer k's response to peer i. Peer i is referred to as *neutral*, if it presumes that peer k linearly changes its response (i.e. resource allocation) corresponding to peer i's next action ($a_{ik}^{'}$). Peer i is referred to as *pessimistic* if it presumes that peer k reduces its resource allocation to peer i fast for $a_{ik}^{'} \le a_{ik}$ but increases the resource allocation slowly for $a_{ik}^{'} \ge a_{ik}$. Finally, peer i is *optimistic* if it presumes that peer k reduces its resource allocation to peer i slowly for $a_{ik}^{'} \le a_{ik}$ but increases the resource allocation fast for $a_{ik}^{'} \ge a_{ik}$. Illustrative examples of these characteristics of resource reciprocation are shown in Figure 7.

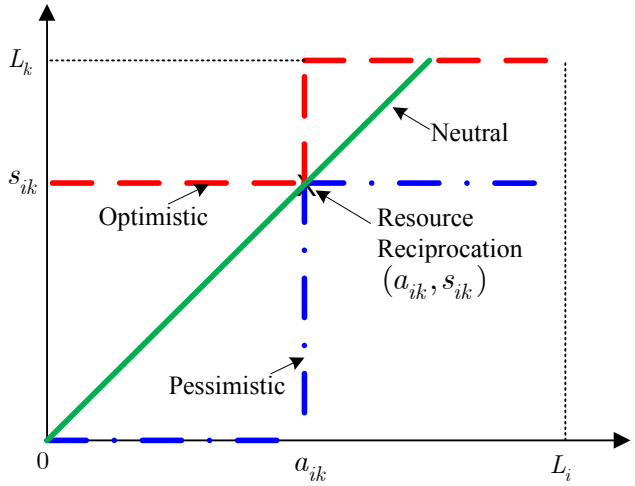

Figure 7. Attitudes for resource reciprocation of boundedly rational peers

These types of peers discussed above obviously affect their resource reciprocation strategies. In (Park & van der Schaar, 2009), it has been analytically shown that if a peer has only one attitude for resource reciprocation and makes myopic decisions, it cannot efficiently reciprocate its resources. Thus, it has been concluded that peers can improve their performance by considering various reciprocation attitudes and multiple observations for past resource reciprocations and making foresighted decisions.

5.4.2 Bounded Rationality: Limited Memory and Computation Power

In (Park & van der Schaar, 2009b; Park & van der Schaar, 2009c), the bounded rationality of peers induced by their limited memory for storing the resource reciprocation history and limited computation power is discussed. Recall that a peer's received resources from its associated peers are captured by its state. Boundedly rational peers can have limited ability to characterize their resource reciprocation with other peers (i.e. they can distinguish their received resources using only a limited number of states). This is due to the large complexity requirements associated with their decision making processes.

In (Park & van der Schaar, 2009b), it has been investigated how this bounded rationality of peers can impact the accuracy of the long-term expected rewards. It is obvious that using more states (i.e., finer states) enables each peer to compute the actual long-term rewards more accurately. However, increasing the number of states also leads to higher computational complexity to find the optimal resource reciprocation strategy. Therefore, it is important for each peer to find the minimum number of states, while achieving a tolerable accuracy of the actual long-term rewards. In this study, the impact of the number of states on the accuracy of the long-term rewards is analytically quantified, and shows how to determine an optimal number of states that achieves the tolerable accuracy.

Alternatively, (Park & van der Schaar, 2009c) has focused on how the heterogeneous peers having different abilities to refine their states can interact with each other, and the corresponding long-term rewards. This study analytically shows that a peer may have multiple actions that are optimal because these actions do not alter its associated peers' states, and thus, they do not alter

the resource reciprocation of these peers. This is because peers cannot differentiate among all possible download rates from their associated peers due to the limited number of state descriptions. It is observed that peers can mutually improve their long-term rewards (i.e., download rates) only if they simultaneously refine their states. It also studies the impact of the heterogeneity of peers on their group formations, and concludes that peers prefer to form groups with other peers, which not only have similar or higher upload rates but also have similar abilities to refine their state descriptions.

5.4.3 Practical Implementation of Foresighted Strategies

The foresighted resource reciprocation strategy discussed in Section 5.3 has been actually implemented in BitTorrent-like system. The foresighted strategy replaces the TFT resource reciprocation strategy and the optimistic unchoke mechanism, which have been implemented in the BitTorrent protocol. By deploying the foresighted resource reciprocation strategy, the following advantages against the regular BitTorrent protocol:

- It improves the fairness - the peers that contribute more resources (i.e., higher upload capacities) can achieve higher download rates. However, the peers that contribute less resources may achieve limited download rates
- It promotes cooperation among high-capacity peers.
- It discourages free-riding by limiting the upload to non-cooperative peers.
- It improves the system robustness by minimizing the impact of free-riding on contributing peers' performance.

Several illustrative experiment results are shown in the following. The experiments host 54 Planet-Lab nodes, 50 leechers and 4 seeds with combined capacity of 128 KB/s serving a 100 MB file. All peers start the download process simultaneously, which emulate a flash crowd scenario. The initial seeds are stayed connected through the whole experiments. A leecher disconnects immediately after it completes its downloads, and reconnects immediately while requesting the entire file again. This enables our experiments to have the same upload bandwidth distribution during the entire experiment time.

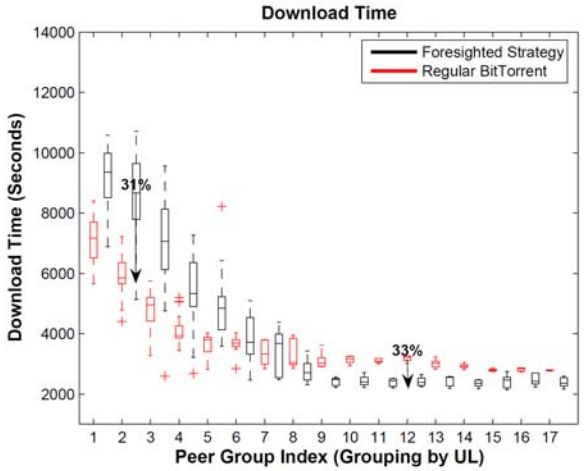

Figure 8. Download Completion Time for Leechers

Fig. 8 shows the download completion time of leechers. The results show the clear performance difference among high-capacity leechers, which are the fastest 20% leechers, and low-capacity leechers, which are the slowest 80% leechers. High-capacity leechers can significantly improve

their download completion time. Unlike in the regular BitTorrent system, where leechers determine their choking decisions based on the TFT that uses only the last reciprocation history, the leechers adopting the foresighted strategy determine their choking decisions based on the long-term history. This enables the leechers to estimate the behaviors of their associated peers more accurately. Moreover, since part of the choking decisions is randomly determined in the regular BitTorrent, there is a high probability that high-capacity leechers need to reciprocate with the low-capacity leechers. However, the randomly determined choking decisions are significantly reduced in the proposed approach, as the random decisions are taken only in the initialization phase or in order to collect the reciprocation history of newly joined peers. As a result, the high-capacity leechers increase the probability to reciprocate resources with the other high-capacity leechers.

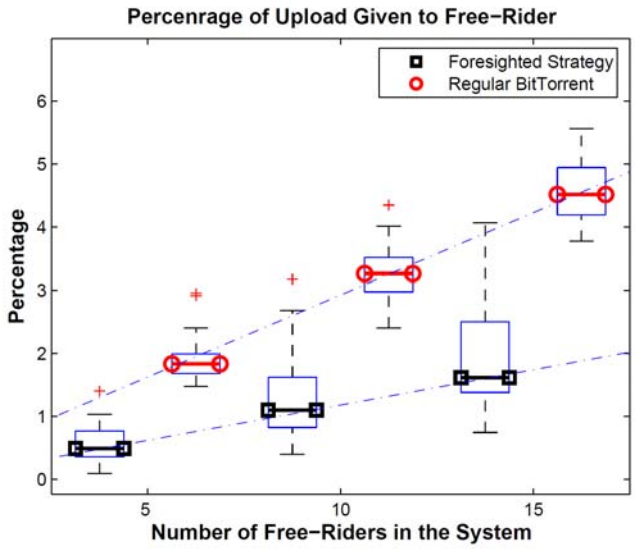

Figure 9. Percentage of free-riders' download from contributing leechers

When leechers adopt the foresighted strategy, they can efficiently capture the selfish behavior of the free-riders. Thus, they can unchoke the free-riders with a significantly low probability. Hence, the free-riders can download their content mainly from seeds not from the leechers. The results shown in Fig. 9 also confirm that the leechers in the regular BitTorrent system upload approximately 2.8-3.7 times more data to the free-riders compared to the leechers in the system where foresighted strategy is adopted. This also shows that the P2P networks consist of leechers adopting the foresighted strategy are more robust against the selfish behaviors of peers than the networks operating using the regular BitTorrent protocol.

In summary, the experiment results confirm that the foresighted strategy provides more incentives for leechers to maximize their upload rate by improving fairness, enables the leechers to discourage non-cooperative behaviors such as free-riding, and enhances the robustness of the network.

6. NEW DIRECTIONS FOR GAME-THEORETIC APPROACHES TO INCENTIVE DESIGN IN P2P NETWORKS

P2P networks are widely used to share user-generated content such as photos, videos, news, and customer reviews. In the case of user-generated content, peers play the roles of producers, suppliers, and consumers at the same time. Most of existing work on incentives in P2P networks focuses on incentives for peers to share content that is already produced, thereby ignoring incentives to produce content. In (Park & van der Schaar, 2010), a game theoretic formulation is proposed to jointly analyze incentives for peers to produce, share, and consume content in P2P networks. The interaction among peers that are connected in a P2P network and interested in content on the same item is modeled as a three-stage game. In stage one (production stage), each peer produces content, while its production decision is not known to other peers. In stage two (sharing stage), each peer makes a portion of its produced content available in the P2P network. In stage three (transfer and consumption stage), peers transfer content available in the P2P network and consume content they have after transfer, which is the sum of produced and downloaded content.

In the one-shot non-cooperative outcome of the content production and sharing game, peers do not share any content at all while producing and consuming an autarkic optimal amount. As upload incurs costs to the uploader while it does not benefit the uploader in the absence of an incentive mechanism, peers do not have an incentive to share their content in the P2P network. On the contrary, social optimum requires full sharing of produced content, where the optimal amount of total production is chosen to equate the marginal benefit of production to peers under full sharing and the marginal cost of production and transfer. As content sharing in the P2P network reduces the overall cost of obtaining content, peers consume more content at social optimum than at one-shot non-cooperative equilibrium. An alternative scenario where the P2P network can enforce full sharing of produced content but cannot enforce the production decisions of peers is also considered. In such a scenario, each peer faces a higher effective marginal cost of production since it has to upload produced content to other peers. As a result, each peer produces a smaller amount than the autarkic optimal amount, and the welfare implication of enforced full sharing is ambiguous.

In (Park & van der Schaar, 2010), two pricing schemes are proposed to achieve social optimum among non-cooperative peers. A marginal product (MP) pricing scheme determines payments to peers based on their sharing decisions in stage two. Using the main idea of the VCG mechanism, an MP pricing scheme provides incentives for users to maximize social welfare. A linear pricing scheme compensates peers for their upload and charges peers for their download at predetermined upload and download prices, respectively. Depending on the system objective, the upload and download prices can be chosen to obtain social optimum as a non-cooperative outcome or to maximize the profit of the P2P network. The model of (Park & van der Schaar, 2010) offers a basic framework which can be extended to analyze more complex situations. For example, a dynamic extension of the model can be formulated to address the incentives of peers interacting over time when each peer can share not only content it produced but also content it downloaded in the past.

7. CONCLUSIONS AND FUTURE CHALLENGES

In this chapter, we discussed P2P systems that have been deployed in file sharing and real-time media streaming. We overviewed various P2P system structures and discussed their advantages and disadvantages for different illustrative implementations. Then, we investigated existing P2P-

based file sharing and media streaming applications in detail, and discussed the limitations of their implementations. One of the drawbacks of the existing implementation is induced by a lack of optimal resource reciprocation strategies among self-interested peers. While BitTorrent systems deploy incentive strategies based on TFT strategy, the myopic resource reciprocations based on this strategy provides only a suboptimal performance. More advanced resource reciprocation strategies are discussed, where the resource reciprocation among the interested peers as a stochastic game, and thus, peers can make foresighted decisions on their resource distribution in a way that maximizes their cumulative utilities. This is a desirable property of resource reciprocation strategy, as peers generally are involved in long-term and repeated interactions. Finally, we investigated the impact of the bounded rationality of peers on their resource reciprocation and the corresponding performance.

While preliminary results achieved by the foresighted resource reciprocation strategies show that they are promising for file sharing and media streaming, additional modules that can support robust and efficient media streaming should be designed and developed. Then, the foresighted resource reciprocation strategies in conjunction with the supporting modules can ultimately improve the media streaming over P2P networks. Moreover, novel algorithms that can reduce the computational complexity required for deploying the foresighted strategies are still to be developed for real-time media streaming.

KEY TERMS AND DEFINITIONS

- torrent (or swarm): a collection of end hosts (or peers) participating in the download of content, where content may refer to one or multiple files.
- tracker**:** a server that coordinates and assists the peers in the swarm.
- leecher and seed: a peer in the leecher state is still downloading pieces of a content, while a peer in the seed state has a complete set of pieces and is sharing them with other peers.
- interested: peer A is interested in peer B if peer B has pieces that peer A does not have and would like to have.
- choked: peer A is choked by peer B if peer B decided to provide no pieces to peer A
- resource reciprocation: resource reciprocation among peers is a set of resources that they have exchanged.

ACKNOWLEDGEMENT

The authors would like to thank Mr. Nicholas Mastronarde for his valuable comments and corrections, which helped us to clarify this chapter. We also would like to thank Dr. Jaeok Park for providing Section 6, which discussed new directions for game-theoretic approaches to incentive design in P2P networks. Finally, this work was supported in part by the Swiss National Science Foundation grants 200021-118230.

REFERENCES

Adar, E. & Huberman, B. A. (2000). Free Riding on Gnutella. *First Monday* 5 (10).

Ali, S.; Mathur, A. & Zhang, H. (2006). Measurement of commercial peer-to-peer live video streaming. *Workshop on Recent Advances in P2P Streaming*, Waterloo, ON, Canada.

Anagnostakis, K. G. & Greenwald, M. B. (2004). Exchange-based Incentive Mechanisms for Peer-to-Peer File Sharing. *Proceeding of International Conference on Distributed Computing Systems,* Philadelphia, PA, USA.

Banerjee, S.; Bhattacharjee, B. & Kommareddy, C. (2002). Scalable application layer multicast. *Proceeding of ACM Special Interest Group on Data Communication*, Pittsburgh, PA, USA.

Bertsekas, D. P. (1976), *Dynamic Programming and Stochastic Control*, New York: Academic, 1976.

Bharambe, A. R.; Herley, C. & Padmanabhan, V. N. (2006). Analyzing and Improving BitTorrent Performance. *Proceedings of IEEE Conference on Computer Communications*, Barcelona, Catalunya, Spain.

Buchegger, S. & Le Boudec, J.-Y. (2004). A Robust Reputation System for P2P and Mobile Ad-hoc Networks. *Economics of P2P Systems*, Cambridge, MA, USA.

Buragohain, C.; Agrawal, D.; Suri, S. (2003). A Game Theoretic Framework for Incentives in P2P Systems. *Proceedings of IEEE International Conference on Peer-to-Peer Computing*. Linköpings, Sweden.

Castro, M.; Druschel, P.; Kermarrec, A.-M.; Nandi, A.; Rowstron A. & Singh, A. (2003). SplitStream: High-bandwidth multicast in cooperative environments. *Proceedings of ACM Symposium on Operating Systems Principles,* New York, USA.

Chen, L.; Low, S. H. & Doyle, J. C. (2007). Contention Control: A Game-Theoretic Approach. *Proceedings of IEEE Conference on Decision and Control,* New Orleans, LA, USA.

Chow, A. L.; Golubchik, L. & Misra, V. (2008). Improving BitTorrent: A Simple Approach. *Proceedings of International Workshop on Peer-to-Peer Systems*, Tampa Bay, FL, USA.

Chu, Y. –H.; Rao, S. G. & Zhang, H. (2000). A Case for End System Multicast. *Proceedings of ACM* I*nternational Conference on Measurement and Modeling of Computer Systems,* Santa Clara, CA, USA.

Chu, Y. –H; Chuang, J. & Zhang, H. (2004). A Case for Taxation in Peer-to-Peer Streaming Broadcast. *Proceeding of ACM Special Interest Group on Data Communication,* Portland, OR, USA .

Cohen, B. (2003). Incentives build robustness in BitTorrent. *Proceedings of P2P Economics Workshop*, Berkeley, CA, USA.

Despotovic, Z & Aberer, K. (2005). P2P reputation management: probabilistic estimation vs. social networks. *Computer Networks*, 50 (2006).

Deshpande, H.; Bawa, M. & Garcia-Molina, H. (2001). Streaming live media over peers. Technical Report 2001-31, Computer Science Department, Stanford University.

Eugster, P.; Guerraoui, R.; Kermarrec, A.-M. & Massoulié, L. (2004). From epidemics to distributed computing. *IEEE Computer*, 37(5), 60-67.

Fan, B.; Chiu, D.-M. & Lui, J. C. (2006). The Delicate Tradeoffs in BitTorrent like File Sharing Protocol Design. *Proceedings of IEEE International Conference on Network Protocols*, Santa Barbara, CA, USA.

Feldman, M; Lai, K; Stoica, I. & Chuang, J. (2004). Robust Incentive Techniques for Peer-to-Peer Networks. *Proceedings of ACM Conference on Electronic Commerce,* New York, NY, USA.

Fudenberg. D & Tirole, J. (1991) *Game Theory*. Cambridge, MA: MIT Press.

Gkantsidis, C; Mihail, M. & Saberi, S. (2004). Random Walks in Peer-to-Peer Networks. *Proceedings of IEEE Conference on Computer Communications*, Hong Kong.

Guo, L; Chen, S.; Ren, S.; Chen, X. & Jiang, S. (2004). PROP: a scalable and reliable P2P assisted proxy streaming system. *Proceedings of International Conference on Distributed Computing Systems*, Tokyo, Japan.

Gkantsidis, C & Rodriguez, P. R. (2005). Network coding for large scale content distribution. *Proceedings of IEEE Conference on Computer Communications*, Miami, FL, USA.

Golle, P.; Leyton-Brown, L.; Mironov, I. & Lillibridge, M. (2001). Incentives for Sharing in Peer-to-Peer Networks. *Lecture Notes in Computer Science*, 2322, 75-87.

Goyal V. K (2001) "Multiple Description Coding: Compression meets the network," *IEEE Signal Processing Magazine*, 74-93.

Guo, L.; Chen, S.; Xiao, Z.; Tan, E.; Ding, X. & Zhang, X. (2005). Measurements, Analysis, and Modeling of BitTorrent-like Systems. *Internet Measurement Conference*, Berkeley, CA, USA.

Gupta, M; Judge, P. & Ammar, M. (2003). A Reputation System for Peer-to-Peer Networks. *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video,* Monterey, CA, USA

Haruvy, E.; Stahl, D. O. & Wilson, P. W. (1999). Evidence for optimistic and pessimistic behavior in normal-form games. *Economics Letters*, 63, 255–259.

Heffeeda, M.; Habib, A.; Botev, B.; Xu, D. & Bhargava, B. (2003). PROMISE: Peer-to-peer media streaming using CollectCast. *Proceedings of ACM Multimedia*, Berkeley, CA, USA.

Hefeeda, M.; Bhargava, B. & Yau, D. K.-Y. (2004). A hybrid architecture for cost-effective on-demand media streaming. *Computer Networks*, 44(3).

Izal, M.; Urvoy-Keller, G.; Biersack, E. W.; Felber, P.A.; Al Hamra, A. & Garc´es-Erice, L. (2004). Dissecting BitTorrent: Five Months in a Torrent's Lifetime," *Passive and Active Measurement Workshop,* Antibes Juan-les-Pins, France

Izhak-Ratzin, R. (2009). Collaboration in BitTorrent systems. *Networking,* 5550/2009, 338-351.

Izhak-Ratzin, R.; Liogkas, N. & Majumdar, R. (2009). Team incentives in BitTorrent systems. *Proceedings of International Conference on Computer Communications and Networks*, San Francisco, CA, USA.

Jaramillo, J. J & Srikant, R. (2007). DARWIN: Distributed and Adaptive Reputation mechanism for WIreless ad-hoc Networks. *Proceedings of International Conference on Mobile Computing and Networking*, Montreal, Quebec, Canada.

Jain, K; Lovász, L. & Chou, P. A. (2007). Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding. *Journal on Distributed Computing*, 19(4), 301–311.

Jun, S. & Ahamad, M. (2005). Incentives in BitTorrent Induce Free Riding. *Workshop on Economics of Peer-to-Peer Systems*, Philadelphia, PA, USA.

Kamvar, S. D; Schlosser, M. T & Garcia-Molina, H. (2003). The EigenTrust Algorithm for Reputation Management in P2P Networks. *Proceedings of International World Wide Web Conference,* Budapest, Hungary.

Kostic, D.; Rodriguez, A.; Albrecht, J. & Vahdat, A (2003). Bullet: High bandwidth data dissemination using an overlay mesh. *Proceedings of ACM Symposium on Operating Systems Principle,* Bolton Landin, NY, USA.

Lai, K; Feldman, M; Stoica, I. & Chuang, J. (2003). Incentives for Cooperation in Peer-to-Peer Networks. *Workshop on Economics of Peer-to-Peer Systems*, Cambridge, MA, USA.

Legout, A.; Urvoy-Keller, G. & Michiardi, P. (2006). Rarest first and choke algorithms are enough. *Internet Measurement Conference,* Rio de Janeiro, Brazil.

Lee, S. I.; Park, H. & van der Schaar, M. (2009). Foresighted Joint Resource Reciprocation and Scheduling Strategies for Real-time Video Streaming over Peer-to-Peer Networks. *International Packet Video Workshop,* Seattle, WA, USA.

Legout, A.; Liogkas, A.; Kohler, E. & Zhang, L. (2007). Clustering and sharing incentives in BitTorrent systems. *SIGMETRICS Performance Evaluation Review*, 35(1), 301–312.

Levin, D.; LaCurts, K.; Spring, N. & Bhattacharjee, B. (2008). BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives. *Proceedings of ACM Special Interest Group on Data Communication,* Seattle, WA, USA.

Li, J.; Chou, P. A. & Zhang, C. (2004). Mutualcast: An Efficient Mechanism for One-To-Many Content Distribution. *Proceedings of ACM Special Interest Group on Data Communication*, Beijing, China.

Liang, C.; Guo, Y. & Liu, Y. (2009). Investigating the Scheduling Sensitivity of P2P Video Streaming: An Experimental Study. *IEEE Transactions on Multimedia*, 11(3), 348-360.

Lin, W.S.; Zhao, H. V.; Liu, K. J. R. (2009). Incentive Cooperation Strategies for Peer-to-Peer Live Multimedia Streaming Social Networks. *IEEE Transactions on Multimedia*, 11(3), 396-412.

Lian, Q.; Peng, Y.; Yang, M.; Zhang, Z.; Dai, Y. & Li, X. (2006). Robust incentives via multi-level tit-for-tat. *Proceedings of International Workshop on Peer-to-Peer Systems,* Santa Barbara, CA, USA.

Liogkas, N.; Nelson, R.; Kohler, E. & Zhang, L. (2006). Exploiting BitTorrent For Fun (But Not Profit). *Proceedings of International Workshop on Peer-to-Peer Systems*, Santa Barbara, CA, USA.

Liogkas, N.; Nelson, R.; Kohler, E. & Zhang, E. (2007). In Exploring the Robustness of BitTorrent Peer-to-Peer Systems. *Concurrency and Computation: Practice and Experience*, 10(2), 179-189.

Liu, J.; Li, B. & Zhang, Y.-Q. (2003). Adaptive video multicast over the Internet. *IEEE Multimedia*, 10(1), 22-31.

Locher, T.; Moor, P.; Schmid, S. & Wattenhofer, R. (2006). Free Riding in BitTorrent is Cheap. *Workshop on Hot Topics in Networks*, Irvine, CA, USA.

Locher, T.; Schmid, S. & Wattenhofer, R. (2007). Rescuing Tit-for-Tat with Source Coding. *International Peer-to-Peer conference*, Galway, Ireland.

Massoulié, L. & VojnoviĆ, M. (2005). Coupon replication systems. *Proceedings of the International Conference on Measurements and Modeling of Computer Systems*, Banff, Alberta, Canada.

Neglia, G.; Presti, G. L.; Zhang, H. & Towsley, D. (2007). A network formation game approach to study BitTorrent tit-for-tat. *Proceedings of Network Control and Optimization*, Avignon, France.

Nguyen, K; Nguyen, T. & Cheung, S. -C (2007). Peer-to-Peer streaming with hierarchical network coding. *Proceedings of IEEE International Conference on Multimedia & Expo*. Beijing, China.

Pai, V.; Kumar, K.; Tamilmani, K.; Sambamurthy, V & Mohr, A. E. (2005). Chainsaw: Eliminating trees from overlay multicast. *Proceedings of International Workshop on Peer-to-Peer Systems*, Ithaca, NY, USA.

Park, H. & van der Schaar, M. (2007). Bargaining strategies for networked multimedia resource management. *IEEE Transactions on Signal Processing*, 55(7), 3496-3511.

Park, H. & van der Schaar, M. (2009). A framework for foresighted resource reciprocation in P2P networks. *IEEE Transactions on Multimedia*, 11(1), 101–116.

Park, H. & van der Schaar, M. (2009b). On the impact of bounded rationality in peer-to-peer networks. *IEEE Signal Processing Letters*, 16(8), 675-678.

Park, H. & van der Schaar, M. (2009c). Evolution of resource reciprocation strategies in P2P networks. *IEEE Transactions on Signal Processing*, 58(3).

Park, H. & van der Schaar, M. (2009d). Quality-based resource brokerage for autonomous networked multimedia applications. *IEEE Transactions on Circuits and Systems for Video Technology,* 19(12), 1781-1792.

Park, J. & van der Schaar, M. (2010). Pricing and Incentives in Peer-to-Peer Networks. *Proceedings of IEEE Conference on Computer Communications,* San Diego, CA, USA.

Padmanabhan, V. N. & Sripanidkulchai, K. (2002). The case for cooperative networking. *Proceedings of International Workshop on Peer-to-Peer Systems*, Cambridge, MA, USA.

Padmanabhan, V. N.; Wang H. J; Chou, P. A. &and Sripanidkulchai, K. (2002b). Distributing streaming media content using cooperative networking. *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video*, FL, USA.

Padmanabhan, V. N.; Wang, H. J. & Chou, P. A. (2003). Resilient peer-to-peer streaming. *Proceedings of IEEE International Conference on Network Protocols*, Atlanta, GA, USA.

Piatek, M.; Isdal, T.; Anderson, T.; Krishnamurthy, A. & Venkataramani, A. (2007). Do incentives build robustness in BitTorrent? *Symposium on Networked Systems Design and Implementation,* Cambridge, MA, USA.

Piatek, M.; Isdal, T.; Krishnamurthy, A. & Anderson T. (2008). One hop reputations for peer to peer file sharing workloads. *Symposium on Networked Systems Design and Implementation,* San Francisco, CA, USA.

Pouwelse, J. A.; Garbacki, P.; Epema, D. H. J. & Sips, H. J. (2005). The BitTorrent P2P file-sharing system: Measurements and Analysis. *Proceedings of International Workshop on Peer-to-Peer Systems*, Ithaca, NY, USA.

Qiu, D. & Srikant, R. (2004). Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. *Proceedings of ACM Special Interest Group on Data Communication*, Portland, OR, USA.

Ratnasamy, S.; Francis, P.; Handley, M.; Karp, R. & Shenker, S. A. (2001). Scalable Content-Addressable Network. *Proceedings of ACM Special Interest Group on Data Communication*, San Diego, CA, USA.

Rowstron, A. & Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large scale peer to peer systems. *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany.

Simon, H. A. (1955). A behavioral model of rational choice. *Quarterly Journal of Economics* 59, 99–118.

Shi, S. & Turner, J (2002). Routing in overlay multicast networks. *Proceedings of IEEE Conference on Computer Communications*, New York, NY, USA.

Shneidman, J.; Parkes, D. C. & Massoulié, L. (2004). Faithfulness in Internet Algorithms. *Practice and Theory of Incentives and Game Theory in Networked Systems*, Portland, OR, USA.

Sirivianos, M.; Park, J. H.; Chen, R. & Yang, X. (2007). Free-riding in BitTorrent Networks with the Large View Exploit. *Proceedings of International Workshop on Peer-to-Peer Systems*, Bellevue, WA, USA.

Stoica, I.; Morris, R.; Karger, D.; FransKaashoek, M.; Dabek, M. & Balakrishnan, H. (2001). Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. *Proceedings of ACM Special Interest Group on Data Communication,* San Diego, CA, USA.

Tian, R.; Zhang, Q. ; Xiang, Z. ; Xiong, Y. ; Li, X. & Zhu, W. (2005). Robust and efficient path diversity in application-layer multicast for video streaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(8), 961-972.

Tran, D. A.; Hua, K. A. & Do, T. T (2004). A Peer-to-Peer Architecture for Media Streaming *IEEE Journal on Selected Areas in Communications*, 22(1), 121-133.

Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4), 279–292.

van der Schaar, M & Chou, P. A. (Ed.). (2007). *Multimedia over IP and Wireless Networks*, New York: Academic.

Venkataraman, V.; Francis, P. & Calandrino, J. (2006). ChunkySpread: Multitree unstructured peer-to-peer multicast. *Proceedings of International Workshop on Peer-to-Peer Systems*, Santa Barbara, CA, USA.

Vlavianos, A.; Iliofotou, M. & Faloutsos, M (2006), "BiToS: Enhancing BitTorrent for supporting streaming applications. *Proceedings of IEEE Conference on Computer Communications*, Barcelona, Catalunya, Spain.

Walsh, K & Sirer, E. G (2006) "Experience with an Object Reputation System for Peer-to-Peer Filesharing," in *Proceedings of Symposium on Networked Systems Design and Implementation*, San Jose, CA, USA.

Xiong, L. & Liu, L. (2004) "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Transactions on Knowledge and Data Engineering*, 16(7), 843-857.

Yang, M.; Zhang, Z.; Li, X. & Dai, Y. (2005) "An Empirical Study of Free-Riding Behavior in the Maze P2P File-Sharing System. *Proceedings of International Workshop on Peer-to-Peer Systems*, Ithaca, NY, USA.

Wang, M. & Li, B. (2007) "R^2 : Random push with random network coding in live peer-to-peer streaming," *IEEE Journal on Selected Areas in Communications*, 25(9), 1-12.

Wang, M. & Li, B. (2007b) "Lava: A reality check of network coding in peer-to-peer live streaming. *Proceedings of IEEE Conference on Computer Communications,* Anchorage, Alaska, USA.

Zhang, X.; Liu, J.; Li, B. & Yum, T. S. P. (2005). CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming. *Proceedings of IEEE Conference on Computer Communications*, Miami, FL, USA.

Zhang, M.; Luo, J.-G.; Zhao, L. &Yang, S.-Q. (2005b). A peer-to-peer network for live media streaming using a push-pull approach. *Proceedings of ACM Multimedia*, Hilton, Singapore.

Zhang, Q; Xue, H.–F. & Kou, X.–D. (2007). An Evolutionary Game Model of Resources-sharing Mechanism in P2P Networks. *Proceedings of the Workshop on Intelligent Information Technology Application*, Zhang Jiajie, China.

Zhao, B. Y.; Huang, L.; Stribling, J.; Rhea, S. C.; Joseph, A.D. & Kubiatowicz, J. D. (2004). Tapestry: A resilient global scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), 41–53.

Zhuang, S. Q.; Zhao, B. Y. & Joseph, A. D. (2001). Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video*, New York, USA.