University of Warsaw
Faculty of Mathematics, Informatics and Mechanics
Vrije Universiteit
Department of Computer Science

Piotr Powałowski

Student number: 209403

Local view modeling for ETA in BitTorrent

Master's thesis
in COMPUTER SCIENCE

Supervisors:
dr Guillaume Pierre
dr Paolo Costa
Vrije Universiteit
dr Janina Mincer-Daszkiewicz
University of Warsaw

August 2008

Abstract

Most of BitTorrent systems implement an algorithm predicting the download time of the torrent. Commonly used Simple ETA algorithm is known for its vulnerability on download rate fluctuations and lack of precision. Many authors address the topic of torrent evolution prediction, but majority of created models focus on torrent's global parameters or take unrealistic assumptions and therefore are unusable in ETA forecast. In this thesis we analyze the conditions influencing the torrent download and propose the local–view model for ETA prediction. We analyze the model qualities and validate it against the real–life torrent traces. We show that created model improves ETA prediction quality when compared to Simple ETA algorithm.

Contents

Chapter 1

Introduction

In the past years P2P data sharing has become increasingly popular among the Internet. One of the most important P2P transfer protocols is now BitTorrent [1], which allows to distribute content among users without the continuous contribution from the original distributor. Content is divided into pieces that are sent to and exchanged between downloaders, effectively distributing the workload and increasing performance. The rules of that exchange are described in the BitTorrent specification, which states the basic mechanisms and concepts of file distribution [2]. The good performance of this protocol resulted in its great popularity and in many implementations of BitTorrent clients.

Even though BitTorrent is commonly used in today's world and it has proven to be useful, there is still little understanding about the properties of BitTorrent data sharing. Taking into account the popularity and importance of this protocol, it is clear that for future developments, improvements and usage in real-world applications, a deeper understanding of its properties and behavior is needed.

Especially essential in understanding BitTorent is the characterization of patterns and policies of data exchange between peers, which could result in deeper insight into the download process. Unfortunately BitTorrent data exchanges are not easy to model. Although all BitTorrent clients comply to the specification, it is very difficult to describe global properties of the content download for a given torrent. The first problem is that different clients tend to use different strategies of cooperation with other users, trying to maximize the benefits of exchange and download speed. Second, in real life situations torrent behavior is very complex to monitor globally. A torrent is composed of many users that may join and leave at any time from all over the world, so there is no possibility of obtaining at any time the global knowledge about the current state of the whole torrent. Because of that uncertainty, assessing properties of a torrent or making accurate predictions about its future states is very difficult, if not impossible.

Because of the difficulty of describing the global state, many authors focus their work on a simplified global model. They often assume a total symmetry between the downloading users and full knowledge of the initial state. This approach gives the advantage of simplifying the mathematical models to describe global properties and predict future state, but these predictions are rarely applicable to real life situations. What is more, these global models
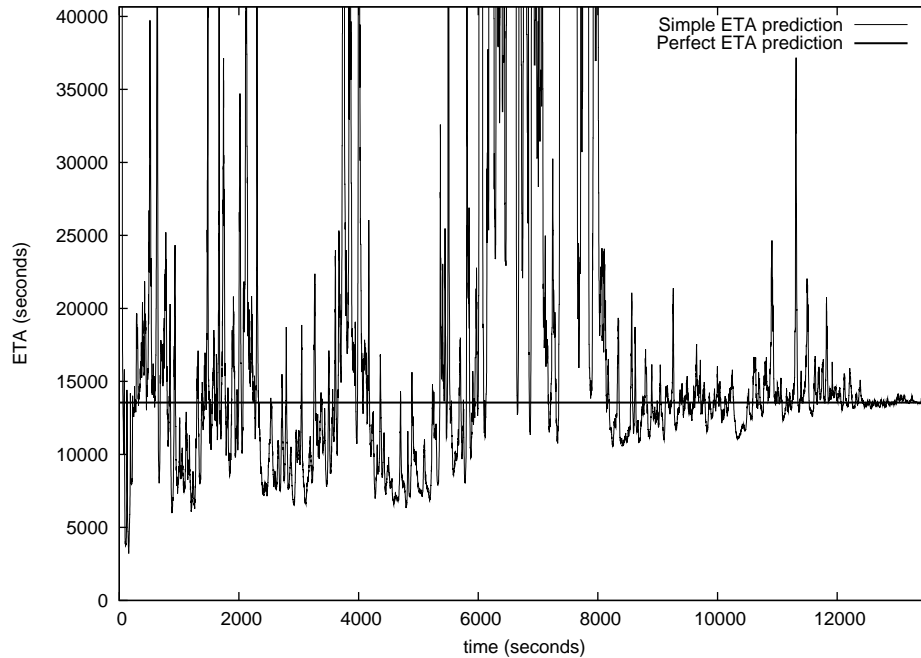
Figure 1.1: Simple ETA prediction in time for a given torrent.

give little insight into the peer to peer data exchange characteristics.

Due to the limited information available in practice about the current state of the torrent, it is clear that a local view analysis would be more applicable to this problem. Such approach focuses on the information that can be obtained by one selected peer, that describes its partial view of the torrent. Based on that data we want to calculate parameters of peer's interactions with others and perform predictions of its state. This idea partially solves the problem of obtaining global state information, because we limit ourselves to the part of the torrent that is directly visible to the selected peer. Because of that, any results obtained by this approach will be easily applicable to real world situations and implementable in BitTorrent clients.

This thesis addresses a specific performance modeling problem based on a local view of a torrent: ETA prediction. ETA stands for Estimated Time of Arrival and is the problem of predicting from the perspective of a given peer how much time it will take for the torrent content to be fully downloaded. Algorithms predicting ETA are implemented in almost all BitTorrent clients. The most commonly used ETA prediction algorithm is Simple ETA, which states that ETA is equal to the remaining size of the content to be downloaded divided by the current transfer speed. However, this prediction is based on the assumption that the download speed of a torrent remains constant for the whole download period, which every BitTorrent user knows is wrong. This ETA prediction is therefore known for its lack of precision. Figure 1.1 shows the Simple ETA predictions for each moment of a given torrent download. The calculated Simple ETA value (thin line) is compared to the perfect ETA prediction – actual time of torrent download (thick horizontal line). We can see on the graph that the prediction given by Simple ETA algorithm is far from perfect, rapidly changing and being strongly influenced by unsteady current

6

download speed. Because of that there is room for improvement and a need to find a more accurate and stable prediction algorithm.

This thesis addresses the problem of BitTorrent ETA prediction by focusing on a peer's local view of the torrent and the characterization of patterns and policies of data exchange between peers. The main idea is to create a model of peer data exchange interactions with the rest of the torrent, by separately analyzing the communication with each of the connected peers. Based on history and patterns found in each of these 'sessions' we can to a certain extent forecast the future data exchange, therefore giving a prediction of peer's state evolution in time as combined prediction of its interactions with connected peers.

Session analysis is done in multiple steps. This thesis propose a state model for peer–to–peer interactions. We introduce an algorithm that determines and separates different phases of communication between peers by classifying them in different states. We use this algorithm and the model to calculate for each interaction the history of state evolution, based on current and past data about the local view of the torrent. By using statistic methods we analyze the state history of each interaction and predict the future behavior and data transfer. Finally we give a prediction for the BitTorrent client's general interaction with the local view of the torrent, with the estimation of the time of the download.

It is important to mention that the quality of ETA prediction is a good marker of the quality of the created model of peer's data exchange interactions with the rest of the torrent. ETA prediction is entirely dependent on a peer's local view of the torrent and its accurate prediction would involve both peer–to–peer future data exchange patterns forecast and estimation of the peers download progress. Furthermore, ETA is easy to implement and makes the comparison of different models straightforward, therefore being a precise analysis tool. Finally, results from this paper, especially concerning the local view model with good ETA prediction algorithm, are easily applicable in practical usage.

The contributions of this work are three-fold. First I determine conditions occurring in real–life traces that influence the torrent download and classify data exchange characteristics between connected peers. Second, I analyze the Simple ETA prediction algorithm as a statistical estimator and determine its qualities (with mathematical proofs). I propose a local view state model for peer–to–peer interaction and heuristic estimators used for ETA predictions. Finally I evaluate the quality of predictions by collecting real–life torrent traces and analyzing them with implementation of the proposed system. Torrent traces are collected by specially instrumented popular open–source BitTorrent client.

The remainder of this thesis is structured as follows. Chapter 2 presents general discussion about BitTorrent based on specification and related works. Chapter 3 contains the analysis of BitTorrent behavior and its influences obtained in experiments. Chapter 4 introduces the local view state model and the algorithm for ETA prediction. Chapter 5 describes the experimental results and evaluation of the model ETA prediction quality. Finally in chapter 6 future works are proposed.

Chapter 2

Related work

This chapter contains information about work explicitly related to the topic of this paper. At first we will give definitions of terms, features and protocols related to BitTorrent, then we will briefly discuss papers of other authors addressing the topic of torrent modeling.

2.1 BitTorrent

In this section we give introduce BitTorrent, concepts related to it and its specification. We will give definitions of *local view of a torrent* and the *ETA*.

2.1.1 BitTorrent basics

BitTorrent is a peer–to–peer file sharing communications protocol [1]. The main goal of BitTorrent is to distribute content to users among Internet in an efficient way. The content consists of any number of files and/or folders.

Data distribution is done by using a BitTorrent client – a program that implements BitTorrent protocol and is required to exchange data. There are many different implementations of BitTorrent clients [3]. Every BitTorrent client can participate in download/upload of any content that is distributed through the BitTorrent protocol.

Users downloading given content are refereed to as *leechers*. Users who already obtained the whole content and distribute it are called *seeds*. All users – both seeds and leechers – are called peers. The group of peers cooperating together in distribution of a given content is called a *swarm*. All peers in a swarm distribute the content they have already downloaded to other peers. Seeds, since they have already the full content, send it to connected leechers. Leechers send content they have to other leechers and download the remaining content from the other peers (both seeds and leechers). After downloading the whole content, a leecher becomes a seed.

A *tracker* is a server that ties the swarm together and assists in the communication between all peers in distributing given content. Each BitTorrent client can obtain from the tracker a list of selected peers in the swarm to connect to.

To join the swarm a new user usually has to download a file from a web page. This file is refereed to as a *torrent file* and contains information about files to download and the tracker that is responsible of the swarm distributing

that content. After download, user's BitTorrent client reads in information from the torrent file and connects to the tracker.

The name *torrent* depending on the context can have different meaning. It can be used to describe the content of a given the torrent file. It also can be used in reference to the state of all peers in swarm downloading content described in given torrent file and to the proccess of downloading the content. In this thesis we will use this term mostly in the last two of its meanings.

2.1.2 Local view of a torrent

BitTorrent content is distributed in a number of identically–sized pieces, typically between 64 kB and 4 MB each. Every peer in a swarm keeps information about the list of pieces it has, is currently downloading or does not have.

Each peer is connected only to some of the other peers in the swarm. It can connect to additional peers only if contacted by them or by asking the tracker for new peers. Connected peers can exchange content, but do not have to. A peer interested in specific piece of a content sends request for download to one of connected peers that has it. The policies of data exchange between peers will be discussed in the next section.

Connected peers exchange information about themselves and their current download progress. This includes the list of all pieces and state of their download, piece requests, name of the BitTorrent client and IP address. This is the only information about the state of the torrent that can be directly obtained by a peer. For every given peer, total information about the torrent obtained from peers connected to it is refereed to as *local information* or *local view* of the torrent.

2.1.3 Data exchange policy

A peer can only control the amount of data it uploads to any connected peer, having no direct control over the download transfer. Each peer decides if it will send content to connected peers based both on its local view of the torrent and a history of past content exchange. The decision policies are partially described in the BitTorrent protocol specification, but the details differ in client implementations. Different polices apply to seeds and leechers.

Seeds send requested data piece to connected leechers, giving preference to these with high–speed connection.

A Leecher sends data to other leechers requesting some of its downloaded pieces based on *Tit-for-tat* exchange policy. This policy states that a leecher A will upload the content to a connected leecher B as long as the following conditions are met:

- Leecher B has pieces that leecher A does not possesses (leecher A is interested in leecher B's data).

- Leecher B uploads interesting data to leecher A.

- Rate of data transfer from Leecher B to A meets given quality conditions. These conditions usually state that for a given period of time average data transfer was high enough in comparison to the transfer from other peers. The details vary in different BitTorrent client implementations.

During content exchange between peers A and B, if according to data exchange policy of peer A the download transfer from peer B does not meet the quality conditions, the peer A can (besides stopping the upload to peer B) *choke* peer B. Choking means explicitly stating that peer A refuses to upload content to peer B. This information is available to B during the exchange the current download progress information between peers A and B (discussed in 2.1.2).

Every peer periodically performs *optimistic unchoking* – a process that allows to undo the choking and test peers that have newly connected. For optimistic unchoking a given peer A selects one of the connected choked peers that are interested in its content (peer B). For a given period of time A uploads the interesting content to peer B. This way A tests the quality of data exchange with B and later decides based on rules described above whether or not to continue exchange and remove the choke flag.

2.1.4 ETA definition

ETA stands for Estimated Time of Arrival. ETA predicting is the problem of estimating the time at which the torrent content will be fully downloaded. This prediction is based from the perspective of a given peer, for a given moment in time of the download. From now on, the peer for which the prediction of ETA is given will be referred to as *selected peer*.

In the paper we will refer to ETA estimation as a function $ETA(t)$:

$$ETA : [0, T] \rightarrow [t, \infty)$$

where T is the actual time of download (and also the estimated value).

At any time t the ETA prediction can only be based on the past states of the download process. Many factors influencing the torrent evolution can be described as random, so in time t the actual time of download T is a random variable. It is desired for ETA estimation in given time t and given download history $History(0, t)$ that the $ETA(t)$ value is equal to the expected actual time of download T:

$$\mathbb{E}(T|History(0, t)) = ETA(t) \tag{2.1}$$

However, since in real-world torrents it would be very difficult to restart given torrent many times from any given time $0 < t < T$, this condition can not be used in practice to measure the quality of ETA prediction.

In this paper we determine the quality of ETA prediction for a given torrent based on the difference between $ETA(t)$ and T for every $t \in [0, T]$ and on its stability. This quality measurement will be discussed later in Section 3.4.

2.1.5 Local view ETA estimation algorithms

A local view ETA estimation algorithm is an algorithm that computes $ETA(t)$ values based only on information available to a selected peer in time between 0 and t. This information includes:

- full information about data owned by the selected peer at any moment between 0 and t (bytes downloaded, pieces downloaded or active, peers connected, etc.)

11

- data exchange characteristic with every connected peer between 0 and t (upload/download transfer, pieces requested, choked information)

- information about the state of every connected peer that have been exchanged between 0 and t (pieces possessed/requested by a connected peer).

Since all of this data are continuous functions of time, storing them would be very difficult. Therefore, in this thesis we limit ourselves to collecting data at discrete moments of time - every one second.

The most common and simplest local view ETA estimation algorithm is Simple ETA algorithm. This algorithm assumes that selected peer's total torrent download transfer $s(t)$ is constant troughout the download time, not influenced by network nor evolution of the torrent, connected peers and selected peer itself. In the basic version the prediction is based on following formula:

$$Simple\_ETA(t) = t + \frac{fileSize - sizeDownloaded(t)}{s(t)}$$

As shown, this algorithm is based only on the current value of the total download transfer of a given peer, so it is in fact local view ETA estimation algorithm. A benefit of this algorithm is the easy implementation.

Based on assumptions above it can be shown that Simple ETA algorithm prediction $ETA(t)$ is equal to T. Unfortunately, these assumptions are rarely met in real-time situations. Usually transfer rate varies intensively and these variations have a huge impact on the Simple ETA prediction, making it imprecise. Futher study of Simple ETA algorithm will be given in Section 3.3.

2.2   Modeling BitTorrent

The topic of analyzing evolution of the torrent and making prediction for its future state has been addressed by many authors. In this section we discuss existing work about modeling the torrent evolution, details of created evolution models, their properties and resulting information about torrent. We distinguish works taking global and local view approach.

2.2.1   Global view approach

The most common approach to the problem of torrent evolution is the global view analysis. It focuses on the evolution of all peers in the swarm, based on total information about the swarm, knowledge about state and connections of all peers. Global view approaches usually assume symmetry between peers in the swarm, continuous values of a number of swarm parameters and do not focus on pieces possession or each peer's interests. The global view approach focuses on aggregate behavior of the torrent system. It simplifies the interactions between peers to get better insight into swarm characteristics, such as distribution of download time among peers or seed–to–peer ratio evolution, and impact of torrent parameters on these values.

One of the most important global view models is the fluid model [4]. Authors consider torrent evolution as a continuous function of the number of

seeds and leechers in time. It is assumed that the rate at which peers join, leave, upload and download are constant in time. A total symmetry between peers is assumed and all peers stay connected long enough to download the content. Interactions between leechers are considered without details about pieces distribution, by given 'file sharing effectiveness' which indicates how effectively leechers distribute content compared to seeds.

Based on these assumptions authors present a fluid model based on simple deterministic differential evolution equations. They give a steady–state solution of these equations and discuss it. This leads to estimation of the 'file sharing effectiveness' parameter value and discussion about the local stability around the steady state.

The simple fluid model, despite its many assumptions is positively validated by real–life experiments. Authors show that it can give accurate prediction of the changes of number of leechers and seeds in torrent in time. On the other hand, the model is very simple and does not takes many factors into account, like for example the long–term death of a torrent. Because of that, the prediction is only covering the growth of seed/leecher number to the saturation point and therefore it does not give a huge insight into torrent evolution.

This paper also presents a simplified BitTorrent peer selection algorithm and study its properties and effect on network performance [4]. An important result is a proven, that in certain conditions a Nash equilibrium exists, under which each peer chooses its actual uploading bandwidth to be equal to the physical uploading bandwidth. This means that under taken assumptions the BitTorrent protocol in some situations enforces peers in a swarm to upload with maximal bandwidth possible, thus maximizing the total data exchange efficiency.

A simpler approach to global view torrent evolution prediction is taken in [5]. In this paper authors discuss different torrent evolution models, based on which they show BitTorrents quality issues, like poor service availability, fluctuating downloading performance and unfair services to peers. All models assume total symmetry between peers: every peer is connected to every other peer, all peers have the same upload/download speed. The number of peers and seeds in the torrent and a peer's download progress are given as a continuous function of time.

The first model considered makes the assumption that the probability distribution of a new peer joining the swarm is exponential. Based on that authors give estimation of total torrent lifespan and the number of peers in torrent. Finally they calculate downloading failure ratio which is a percent measure of how many peers connected to the swarm fail to download the content before the last seed disconnects. Even though the taken assumptions are strong, this is one of few models concern about the long–term prediction of torrent evolution and gives good results when validated by experiments.

Next presented in the paper approach is a fluid model of seed and leecher number evolution in time, based on paper [4]. Authors made an additional simplifying assumption, that peer's download speed is far greater than upload speed. They validated this model against real-world traces and used it for service fairness study.

Finally, authors presented a study of inter-torrent relations. This approach

is rare among papers about BitTorrent, but it is important to notice that in real–life situation most users download more than one torrent at a time, so the interaction between torrents have large impact on evolution of each of them. Further discussion of that interference is presented in 3.1.8 In the model of a multiple torrent evolution, shown in [5], the interactions of peers within one torrent are based on fluid model results, while each peer is allowed to participate in multiple torrents. Detailed calculations show that this affects the time that seeds stay connected and therefore extend torrent lifespan. This model assumes, beside fluid model assumptions, total symmetry between all torrents and a constant torrent creation rate.

An advanced fluid model is presented in [6]. This work rely strongly on [4], but has deeper mathematical background. Authors of this paper describe torrent evolution in terms of stochastic differential equation, which are more realistic than the fluid model in [4]. This model considers three different types of peers: seeds, leechers with few pieces and leechers with most pieces. Authors address the problem of pieces distribution among leechers by assuming its uniformity and then estimating probability of interest and data exchange between all three types of peers. Similar to [4], authors assume that there is a total symmetry between peers of given type, all peers are connected with each other and stay connected long enough to download the content.

Based on results and assumptions above, authors give stochastic torrent evolution equations. This is a large improvement over a deterministic differential evolution of a simple fluid model shown in [4]. Authors of [6] give study of a steady–state performance of their model and provide equations for average numbers of peers of each type as functions of time. They derive a formula for total average download time in steady–state and discuss the impact on its value by connection bandwidth, peer arrival and seed departure rate.

In paper's conclusion authors validate the stochastic model against real–life data. They show that their model not only gives fair prediction of torrent evolution, but also it is more accurate than the simple fluid model from [4].

There have been written many papers about global view models. We have discussed the details of some of them. Even though, they do give insight into torrent evolution, they focus only on major swarm parameters and are useless in ETA prediction. Global view models can forecast future number of leechers and seeds in the swarm, but since they treat peers cumulatively, they can not predict the future evolution of a selected peer in a given state of download.

2.2.2 Local view approach

As opposed to global view, local view approach to torrent evolution prediction focuses only on torrent changes in time observed by a selected peer. In this approach we are restricted only to the local view information that is available to the peer by BitTorrent specification (see Section 2.1.2) and aim to predict future state in time of a selected peer.

The greatest advantage of local view modeling is that the obtained results can be easily applied to real–world situations, since they use only information available to the selected peer by the BitTorrent client. The downfall is that we do not have the access to most of information about the swarm, thus making estimation of interesting torrent parameters less accurate and more complicated than in global view approach.

An example of local view model is presented in [7]. Authors model evolution of a given peer P in time as a Markov chain in state space (n, b, p), where n is the number of active connections, b is the number of downloaded pieces, and p is the number of peers that are interested in and are interesting for selected peer P. They present an equation for transition probabilities as a function of multiple torrent parameters and assume a total symmetry between connected peers. This model is limited only to strict 'tit–for–tat' data exchange strategy, so it is useful only for leecher–leecher interaction. The authors show that this model predicts occurrence of three different phases during the download process, although one might argue that this result is implicitly placed among model assumptions.

Although the model is successfully validated against torrent simulation and real–world traces, it has limited use for real–life torrent evolution prediction due to many strong assumptions that in general are not met. In example this model takes into account only leecher–leecher interactions, despite the fact that in many cases a majority of the data transferred in the swarm is sent by seeds (see Section 3.1.2). As a result this model is ineffective in ETA prediction.

Even though this model itself has limited real–world applications, some results based on it can be used in prediction of torrent evolution. Relying on the proposed model, the authors study the impact of the maximum permissible number of simultaneous connections on the efficiency of the system, followed by the discussion of the stability of the BitTorrent protocol. Additionally they discuss the problem of retrieving the last piece of the torrent files (*Last Piece Problem*) and impact of the effects of seeding on the model.

2.2.3 Discussion

The ETA prediction is a topic that have not been addressed directly in the literature. There are many global view models created that give fair prediction of the swarm evolution, but are too simple to result in an algorithm that would forecast the download time. On the other hand, few local view models have been created. The only one model that focuses on the evolution of a selected peer is too abstract for real–life implementation, as it takes many assumptions that are not met in real world torrents. Therefore to address directly the problem of ETA prediction it is needed to build a new model. It has to be specialized in local view data exchange between selected peer and peers connected to it and result in a algorithm that will be implementable in real–life situations.

Chapter 3

BitTorrent experimental analysis

To build a good local–view model of BitTorrent, we first need to observe the variety of behaviors typical of such networks.

This chapter discusses different causes that can influence the download of the torrent and therefore can have an impact on the ETA prediction. Based on traces of real–life torrents we present and describe different data exchange patterns between peers. Next, we define and analyze the "*Simple ETA Prediction*" algorithm that is most commonly used in BitTorrent clients. Finally, we discuss the problem of measuring the ETA prediction quality.

3.1 Conditions influencing torrent download

Many different conditions can have an influence on a torrent's download time. Some of them are explicitly related to the BitTorrent specification, while others are a result of protocol's extensions or are built on top of it. Finally, some of them are caused by user interference. All of them may occur in real–life torrent download and have impact on ETA prediction.

Here we list several of these conditions that will be discussed in the thesis and referenced during the creation of the prediction model. We give a brief description of each of them and its impact on torrent evolution and ETA prediction.

3.1.1 Connected peers exchange

A peer connects to many other peers during the download time. Since it has a limited upload capacity, it tries to connect to peers that will grant the most profitable exchange of data. This is done mostly with the *tit–for–tat* algorithm, but details vary in different BitTorrent clients. If a connected peer does not meet selected peer data exchange quality requirements, the selected peer stops sending to it data, often choke it and finally disconnects. In search for connecting peers with better upload transfer selected peer uses *unchoke* and *optimistic unchoking* algorithms.

The connection and disconnection to remote peers is an important part of BitTorrent specification. Each peer continuously changes the set of the peers it is connected to in order to maximize its own download speed. The total
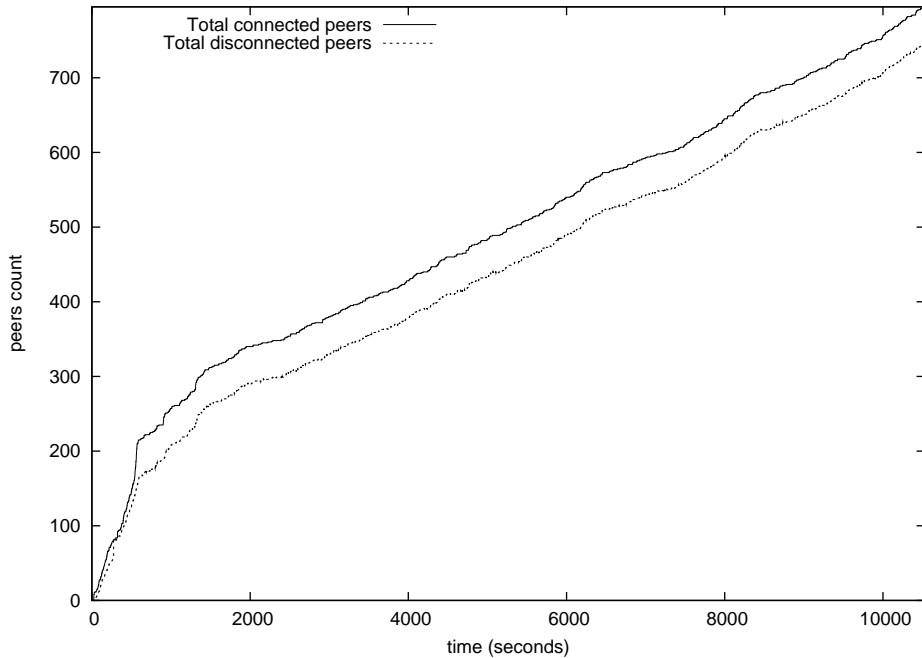
Figure 3.1: Count of total peers connected and disconnected in time for a given real–life torrent. It can be noticed that the difference between these – a number of peers currently connected is constant in time.

number of connected peers is relatively constant in time, so the connection and disconnection rates are similar. The speed of peer exchange tends to be stable during the main part of the download time.

An example can be seen in Figure 3.1, showing the total count of peers connected and disconnected during the time of torrent download. We observe that after some start–up time the difference between these two, which is the number of currently connected peers, is constant in time. Additionally, starting from time $t = 1000s$ to the end of download the connection and disconnection speed is relatively constant.

It is important to mention that the peer exchange rate varies between torrents. Small torrents or torrents with high seed–to–leecher ratio usually have low peer exchange rate (about 1 connection per minute). Large torrents with low seed–to–leecher ratio can have exchange rates of 5 connections per minute or higher.

The peer exchange rate has a direct impact on the download process – high rates result in shorter connections with peers, thus less stable download transfer. This complicates the ETA prediction.

3.1.2 Power seeds

In most torrents, the connection speeds of different peers in the swarm are not uniform – they depend on many factors and can have extreme values (from hundreds of bytes per second to tens of megabytes per second). Because of that, the majority of torrent data is often downloaded from a minority of connected peers. In some examples one connected peer sends even up to 50%
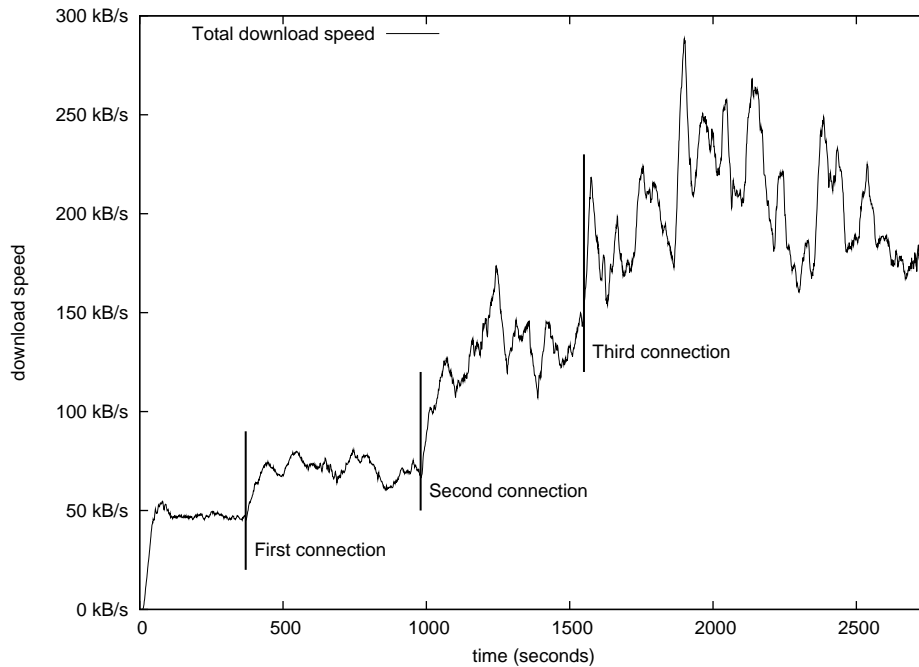
Figure 3.2: An example of torrent download speed in time. On the graph we have marked three connections by *power seeds*.

of the total data. This is particularly true for seeds, as they tend to keep more stable connections than leechers. A connected seed with high data upload speed that sends large part of the data to selected peer will be referred to as *power seed*.

When a peer changes its peer connections, it has a chance to connect to a power seed. This is a rare event whose probability varies between torrents. In some observed examples during the download time selected peer connected to as many as three power seeds.

A connection to a power seed has a huge impact on the torrent download process. It strongly increases the download rate, in some cases even twice or more. This is a important problem in ETA prediction, as connections to power seeds are rare and therefore difficult to predict.

To illustrate the impact of power peer connections on download process we give an example of a torrent trace. Its download speed as a function of time is shown in Figure 3.2. On the graph we have marked the moments of three power peer connections. As we can see after each connection the download speed increased significantly.

3.1.3 Tracker updates

Every peer in the swarm may contact the tracker at periodic intervals to obtain a list of peers it can connect to. The received information is useful to the peer as it helps in efficient peer exchange (see 3.1.1). The update periods are often 10–20 minutes long and this information is easily accessible from the BitTorrent client.

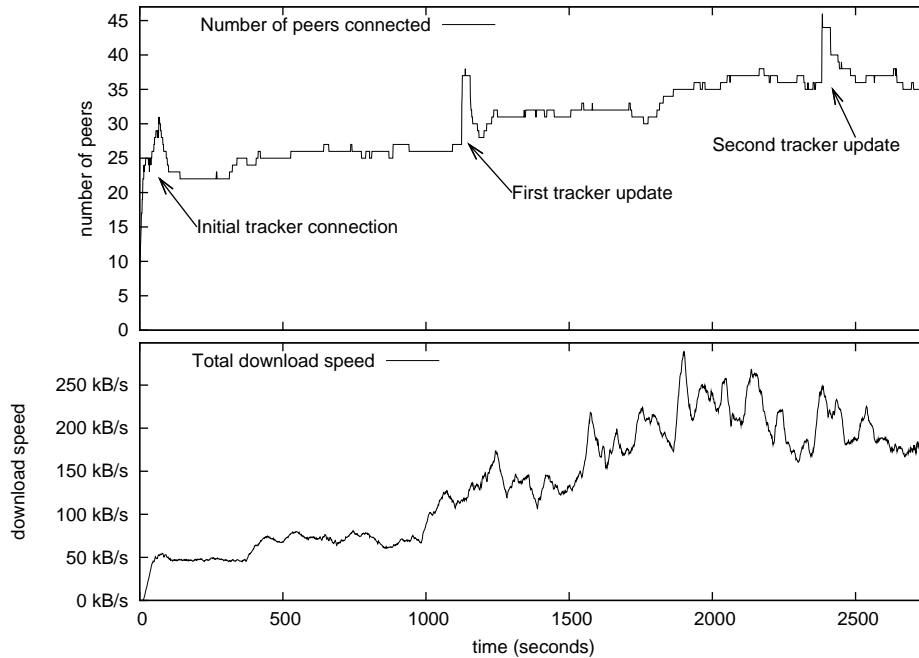The impact of tracker updates can be seen in some torrent traces, as in

Figure 3.3: Top graph presents number of currently connected peers in time for a given torrent. We can see tracker updates temporarily increasing the number of connected peers. This compared with bottom graph show no noticeable impact of tracker updates on download speed.

periods of 10–20 minutes in the beginning of the torrent a temporary increase in the number of connected peers can be found. Tracker updates help in maximizing the selected peer download speed, but their impact on the download speed is small.

An example of impact of tracker updates on download process is shown in Figure 3.3. We see on the top graph that the number of connected peers is slowly growing in time with exception of three moments in which it increases and decreases rapidly. The first peak occurs at the beginning of download, which can be interpreted as initial contact with the tracker. Two next peak occur in interval of 20 minutes and are caused by tracker updates. By comparing top graph with the bottom one showing the torrent download speed in time we see that there is no noticeable impact of tracker updates on download rate.

3.1.4   Content prioritizing

BitTorrent clients can choose to download pieces of the torrent file in any order, but it is believed that the *rarest first* strategy is the most efficient [8]. The *Rarest first* download strategy suggests for a selected peer to choose to first download the piece that is the rarest among the connected peers, therefore increasing the number of peers interested in selected peer and resulting in uniform pieces distribution among the peers in swarm.

Most of the BitTorrent clients have implemented the feature of prioritizing the content download. This feature enables the user downloading multi–file

Private Tracker			Public Tracker		
Torrent #	Seeds	Leechers	Torrent #	Seeds	Leechers
1	15	0	1	815	2339
2	15	1	2	540	1619
3	8	1	3	426	1041
4	7	1	4	879	2119
5	2	0	5	10660	21549
6	8	0	6	343	1329
7	16	1	7	653	911
8	5	0	8	473	716
9	10	1	9	403	593
10	13	0	10	496	785

	Private	Public
Average leecher–to–seed ratio	0.05	2.26
Standard deviation of ratio	0.06	0.82

Table 3.1: Comparison of private and public tracker parameters.

torrent to set priorities for the download of separate files. The most common settings include priorities: high, normal and do not download. Setting the priorities is explicitly conflicting with the rarest first strategy, as it forces BitTorrent client to make piece choices that are not optimal for the download progress. This usually has a negative impact on the download time.

3.1.5 Private trackers

A private tracker is a tracker which restricts who can use it, often by requiring registration of a user account. Private trackers usually gather statistics of how much the users uploads and downloads and may enforce a minimum upload–to–download ratio and/or seeding period after torrent download.

The download process of a torrent from a private tracker is in most cases much different than from public one. As a result of the restricted access and statistics keeping, the number of leechers is much smaller than in public tracker torrent. On the other hand, since most users are interested in having high upload–to–download ratio, there are many seeds distributing the torrent long after the download of the files. Therefore when downloading the torrent the transfer rate is much higher than in public tracker torrent and is mostly due to seed upload.

An comparison of public and private tracker is shown in table 3.1. There we present number of leechers and seeds of ten random real–life examples of torrents that have a similar age (three days old) and category (movies) for a private and a public tracker. We see that for a private tracker the average swarm size is much smaller than for public one and the number of leechers is minimal. Public tracker on the other hand is overloaded with leechers. In comparison of average leecher–to–seed ratio we get that the public tracker ratio is far higher than private one.

3.1.6 Extensions to the BitTorrent protocol

Since the birth of BitTorrent many different extensions have been introduced. Some of them are official and therefore implemented in most of BitTorrent clients, while others are still unofficial, usually introduced in one of the clients and therefore are not commonly supported. All of them are intended to increase the efficiency of the download process.

As an example of possible extensions, two can be mentioned. 'Fast Peers Extensions' allow a peer to more quickly bootstrap into a swarm by giving it a specific set of pieces which it will be allowed to download regardless of its choked status [9]. 'BitTorrent Location-aware Protocol 1.0' is an unofficial algorithm that increases the download performance of selected peer by considering peers geographical location in connection process [10].

Extensions to BitTorrent add complexity into the download process and each of them is supported only by a subset of all BitTorrent clients. Extensions have a reasonable impact on torrent download time, but because of the client dependency they cannot be included into the general ETA prediction.

A list of most important extensions with their description can be found in [3].

3.1.7 Global torrent evolution

The global torrent state understood as a state of all peers in swarms changes over time. As a torrent becomes popular new leechers join the swarm, they download more and more of the content, finally becoming seeds and leaving the swarm. This evolution of the torrent has been discussed and modeled by many authors [4, 6]. As the experimental results show, the fluid model is useful in making prediction about the future number of seeds and leechers in swarm.

In ETA prediction we focus on a local view of the torrent and download time of torrent files for a selected peer only. On the other hand this is largely affected by global torrent properties as the number of seeds and leechers in the swarm. In many cases the download time of selected peer is relatively short compared to torrent lifetime and therefore changes in the global torrent state does not have a noticeable impact on the download process. However, in some cases, mostly for unpopular torrents with small swarm size, above statement is not true and the impact of the torrent evolution on the download process is important. This is particularly true for both very new and very old torrents. In the first case the leecher–to–seed ratio and number of peers change rapidly, while in the second case the decreasing number of peers has a huge impact on download speed of a selected peer. Since the global torrent evolution has an impact on the download process, it also affects the download time and should be considered in ETA prediction.

3.1.8 User related influences

Peer connection quality   One of the factors that has the biggest affect on the download process of a selected peer is the quality of its Internet connection. Different connection technologies (Dial–up, Broadband, ISDN, wireless) offer different transfer rates and connection stability. This has a threefold

impact on the download process. First, it affects the selected peer download rate limit, therefore peers with high–speed connection can download torrent faster. Second, higher upload speed makes selected peer more likely to meet other peer's connection quality standards and therefore keep the data exchange longer. Larger upload capacity is a resource that peer can trade for additional download rate. Finally, unstable connections have a huge impact on selected peer's download process as it changes in time the current download and upload capacity. This can be observed in torrent traces as periods of decreased download rate. Those changes have an important influence on ETA prediction and cannot be accurately predicted by local view algorithm, as they are not related with BitTorrent.

Other connection parameters, like firewalls or NAT, also affect the download process as they interfere with selected peer being connected to by other peers and limit download transfer.

Multiple torrents download   In most real–life situations a peer is involved in several torrents at the same time. All these torrents are downloaded or seeded simultaneously and influence each other. Since they share the same resources (upload/download capacity) an increase in data exchange in one torrent is very likely to result in a decrease in other ones. On top of that most BitTorrent clients have implemented algorithms that in time activate/pause downloading torrents to optimize the total download rate. This choice is usually based on gathered statistics for every of the torrents, e.g. the number of connected peers and average download speed.

Studies indicate that, besides these negative interferences, multiple torrent download can also have positive effects by increasing the average seeding time in the swarm [5].

Other network activity   It is common that a user besides running Bit-Torrent client to download torrents also uses its computer for other Internet–related purposes. This includes HTTP downloads, FTP data transfer, web browsing, Internet telephony (VoIP) and instant messaging. Since all these actions require Internet connection, they drain peers' upload/download capacity and therefore can have a negative effect on the torrent download process. This is a serious issue, not only because of the impact on the ETA, but especially because information about non–BitTorrent network activity cannot be accessed from BitTorrent client, therefore complicating the ETA prediction.

BitTorrent client events   Most of the BitTorrent clients give the user a large set of options for control over the download process. Among many things discussed above (3.1.4 and 3.1.8), a user can at any time pause/unpause the torrent download, turn off or on the BitTorrent client and change the download settings. While not related to BitTorrent protocol, this has a direct impact on the download performance.

Some of the settings that are available to the user, besides the control of the download process, change the rules of peer connecting. This is especially true for setting limits to download rate of the torrent as it interferes with *tit–for–tat* policy.

Peer activity patterns   We have discussed above different user–related processes that affect the torrent download. Since user every day activities follow regular patterns (sleep patterns, weekly patterns), the processes shown in 3.1.8 are also expected to have periodic behavior. An example that shows such a behavior could be a torrent in which most of the peers turn their computers off every day for the night.

This in a direct way affects the download process. What is more, the periodic behavior could be a subject of prognosis, therefore improving the ETA prediction.

3.2   Data exchange characteristics

In local view of a torrent an important thing to understand are the interactions of the selected peer with peers connected to him. In particular, for ETA prediction, it is important to have an insight into specifics of data exchange with connected peers. That insight can be used to forecast the future data download rate from every connected peer, which adds to selected peer total download speed that translates to ETA.

In this section we will show and classify different data exchange patterns between the selected peer and peers connected to him. The categorization will be done on exemplary real–life torrent data, we will focus on the changes of the download rate in time. The results of this classification will be used when constructing the model in chapter 4.

From now on we will use the term *session* to describe all the interactions between the selected peer and one remote peer connected to it. We will consider a session to last for the total download time, disregarding the time when the remote peer connected or was connected, and whether or not it disconnected at any time of the download.

3.2.1   Connection and disconnection

Once connected, a remote peer does not have to continuously send torrent data to the selected peer. In general during one session there are only few periods in time when data is being transferred. On average, depending on the torrent, the number of periods is about three per session. This number can vary significantly. The size of periods changes between the sessions, although it have been observed that during one session periods tend to have similar length.

The lack of download transfer from remote peer can have two different reasons. The first one is that the remote peer have stopped sending data due to loss of interest in the selected peer or poor data exchange quality. In this case the selected peer is still connected, has access to remote peer's info (see 2.1.2) and absence of download transfer is more likely to be temporary as in time the remote peer can reconsider its decision. The second reason is a network disconnection resulting in lack of any current information about remote peer, which can be more permanent.

On the example shown in Figure 3.4 we have a part of a session with a seed. We see that there are eight distinctive periods in time when the download rate is above zero. There are also two periods (marked with arrows) when the
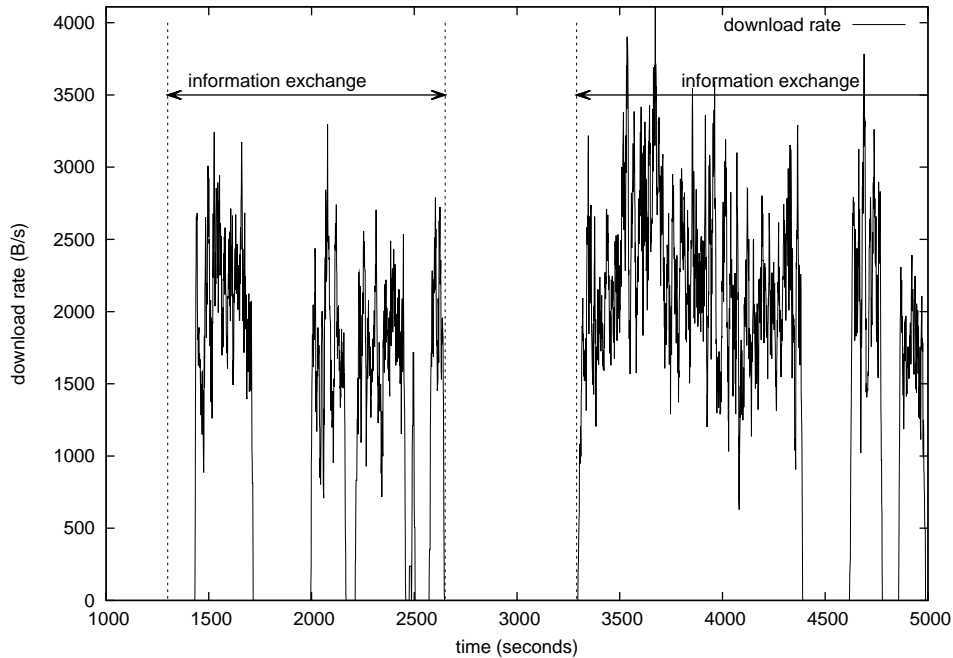
24

Figure 3.4: Example of a part of a seed session with multiple connections and disconnections. Periods when the seed was exchanging the information about itself have been marked.

information about the seed is available to the BitTorrent client – the seed is connected. As we see, the fact of connection does not imply that there is a download in progress. What is more, during the first connection period the download rate several times falls to zero without the disconnection (possibly due to choking by the seed). Around $t = 2650s$ the seed disconnects and the download transfer falls to zero. After almost $700s$ it reconnects again and continues to send data.

3.2.2 Choking and unchoking

When maximizing its download rate, a peer browses through remote peers in a swarm in search for a better data exchange. One of the methods of finding a connected remote peer that provides a good download speed is *optimistic unchoking* (see 2.1.3 for details). For optimistic unchoking, at any one time there is a remote peer which is unchoked regardless of its upload rate. Which peer is optimistically unchoked rotates every given period of time (usually 30 seconds) [2]. Newly connected peers are three times as likely to start as the current optimistic unchoke as anywhere else in the rotation.

The *optimistic unchoking* can be observed on the session download rate graph as a group of 'spikes' – transfer periods that have the same height and occur in relatively constant, short periods of time. Since unchoking can be unsuccessful, it does not have to lead to actual transfer. In some cases remote peer's whole session can be made of one or few unchoking spikes.

It is important to mention, that in some rare cases a choked remote peer can still send data to selected peer. This can be true among other reasons
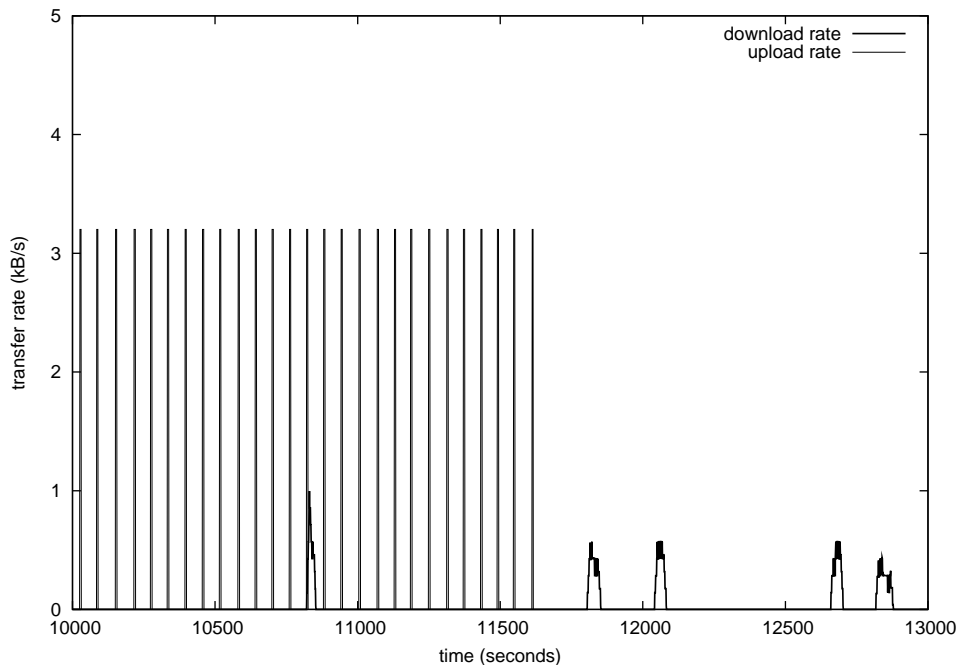
25

Figure 3.5: Example of a part of a session with a leecher. We see clearly visible unchoking spikes.

as a part of unchoking, optimistic unchoking, delayed data transfer or one of BitTorrent extensions (e.g. 'Fast Peers Extensions' – see 3.1.6 for reference).

On the Figure 3.5 we have an example of a session with a leecher when the process of optimistic unchoking takes place. We see that the selected peer is repeatedly unchoking the connected leecher trying to start the data exchange. In regular periods it sends data to the leecher in hope that it will return the transfer. At time $t = 10800s$ the leecher briefly responds, but we see that no long–term data exchange takes place.

3.2.3   Comparison of seeds and leechers sessions

The sessions of selected peer with leechers and with seeds are similar in terms of their composition from transfer periods. However, seed sessions tend to be more stable, disconnection periods are shorter and connection periods are longer. It is also more likely for the connection with seed to last up to the end of the torrent download time.

It is important to mention, that since a remote peer can change from a leecher to seed during download time, one session can contain both data exchange with a leecher and a seed (that is in fact the same peer). An example of that kind of session is shown in Figure 3.6. On the graph we see the download progress of a connected peer. At time $t = 5085s$ it finishes downloading the torrent content and becomes a seed. The impact of that change on the download/upload rate can be clearly noticed. After the change the upload to the peer stops, as it stopped downloading the torrent. The download rate from this peer after the change increases and becomes more stable. What is more, we see that the number of connections/disconnections decreases.
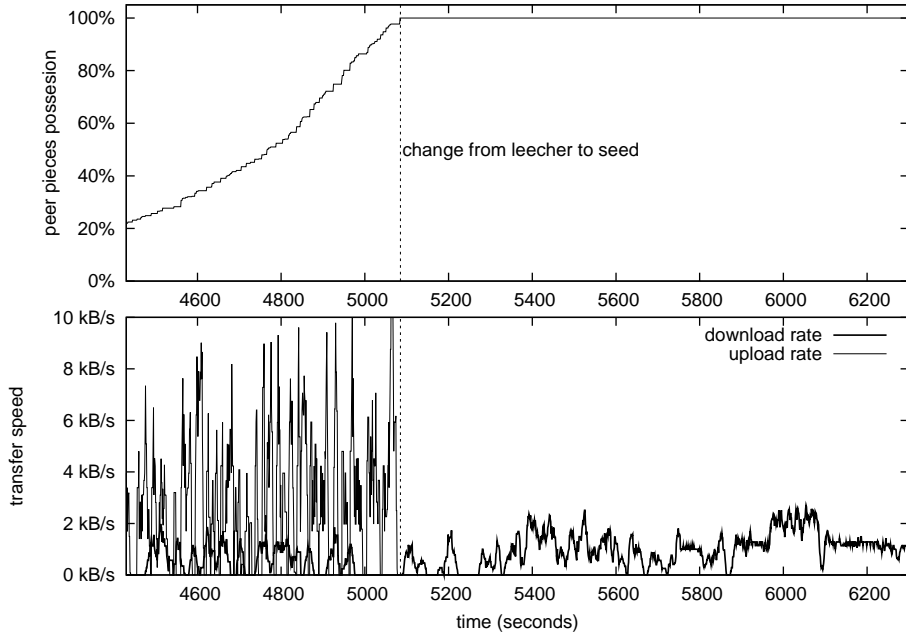
Figure 3.6: Example of a session where a leecher changes to a seed.

3.3 Simple ETA Prediction

The Simple ETA prediction is the most commonly implemented algorithm for estimation of the torrent download time. In Section 2.1.5 we have shown a basic version of this algorithm. In this section we show its full version and discussion of its quality depending on taken assumptions about the download speed. Finally, we show the main properties and problems with Simple ETA prediction.

3.3.1 Definition

Let $s(t)$ be the torrent download speed in time t and $sizeDownloaded(t)$ the total size of data downloaded until the time t. Let the total torrent data size be equal to $fileSize$. We assume that time is quantified in seconds, so t is an integer value. We define Simple ETA in time t as a prediction based on the following formula:

$$Simple\_ETA(N,t) = t + \frac{fileSize - sizeDownloaded(t)}{Average(N,t,s)}$$

The N value is an integer parameter of the prediction algorithm ($N > 0$). The $Average(N,t,s)$ function returns the average of the latest N values of function $s()$ in the last N seconds, starting from the value in time t.

$$Average(N,t,s) = \frac{\sum_{i=max(0,t-N+1)}^{t} s(i)}{min(t+1,N)}$$

We see that when $N = 1$ the function $Average(N,t,s)$ is equal to $s(t)$ and we get the basic form of Simple ETA prediction shown in Section 2.1.5.

27

3.3.2 Accuracy

We will verify the accuracy of Simple ETA prediction in two steps by making more and more realistic assumptions about the download speed $s(t)$.

At first let us assume that the download speed $s(t) = s$ is constant during the full download time. Based on that simple calculations show that for every value of N we get the accurate prediction of the download time.

$$
\begin{aligned}
downloadTime &= min(t \in \mathbb{N} : \sum_{0}^{t-1} s(t) * 1s \geq fileSize) = \frac{fileSize}{s} = \\
&= t + \frac{fileSize - s * t}{s} = t + \frac{fileSize - sizeDownloaded(t)}{Average(N, t, s)} = \\
&= Simple\_ETA(N, t)
\end{aligned}
$$

Even though this is a very good result, it is important to notice that the taken assumption is very strong and in real–life situations download rate is far from being constant.

In second step instead of $s(t) = const.$ we take a weaker assumption that $s(t) > 0$ is a random value with a probability distribution f that is constant in time. What is more we assume that the standard deviation of f is finite and that $s(t_1)$ and $s(t_2)$ are independent when $t_1 \neq t_2$. Those assumptions give us the opportunity to make a more realistic analysis of Simple ETA algorithm, while still having the possibility of working with mathematical model.

Based on these assumptions we can show that $Simple\_ETA()$ prediction is biased and the predicted ETA is on average higher than the actual download time.

$$\mathbb{E}(Simple\_ETA(N, t)) > downloadTime$$

The difference between average given prediction and the actual download time is proportional to the amount of data do be downloaded ($fileSize - sizeDownloaded(t)$). What is more the difference increases with growing standard deviation of $Average(N, t, s)$.

$$\mathbb{D}^2 Average(N, t, s) \propto \mathbb{E}(Simple\_ETA(N, t)) - downloadTime$$

Detailed calculations are in the Appendix A.

Since Simple ETA prediction is biased, it is unlikely to give accurate download time. It is important to notice, that $\mathbb{D}^2 Average(N, t, s)$ is proportional to $1/N$ so by increasing the value of N we can improve the accuracy of the prediction. What is more, standard deviation of the ETA prediction decreases with growing N and becomes more stable, unaffected by fluctuation of download speed. Even though both of these results require taken assumptions, they can be observed in real–world data.

3.3.3 Stability and responsiveness

In this section of analysis we will focus only on real–life situations. We will focus on the stability of Simple ETA prediction and its relation with ability to work in changing download conditions – its *responsiveness*.

There are multiple processes influencing the stability of Simple ETA prediction. The most important is fluctuating download transfer between selected
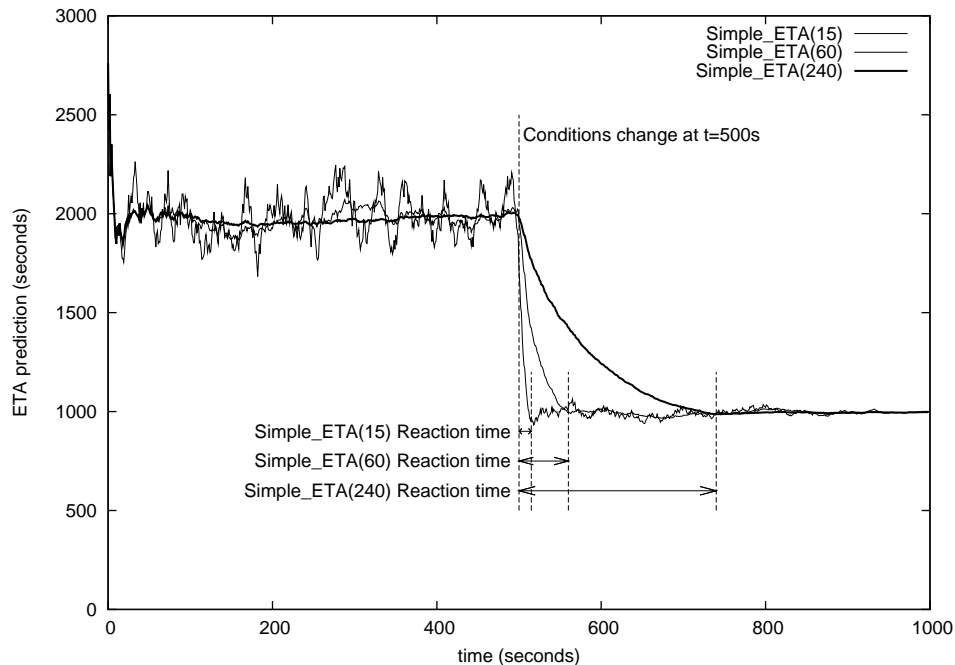
Figure 3.7: An example of relation between stability and responsiveness for Simple ETA algorithm based on artificial data.

peer and connected peers. Additionally many processes listed in Section 3.1 have their impact, for example the connected peer exchange (3.1.1) and unstable network connection (3.1.8). They influence the download speed $s(t)$ by increasing its standard deviation. What is more, they allow the download transfer to fall to zero and therefore resulting in prediction of infinite ETA.

Simple ETA algorithm address the problem of stability by using average transfer for the prediction. As stated before, when average is taken over a larger period of time (N increases) the prediction becomes more stable.

A problem arises when we focus on ETA prediction in fast changing download conditions. An example of these are connections from power seeds (3.1.2) and global torrent evolution when swarm is small (3.1.7). In these situations it is important for an ETA prediction algorithm to quickly react and adjust to the new conditions. Again, the Simple ETA algorithm does not address this problem directly and its responsiveness can be increased only by decreasing the N parameter.

We see that for Simple ETA algorithm stability and responsiveness are two opposing processes and that increase of one can be made only by decreasing the other. Since in different torrents and for different selected peer the importance of these two processes changes, it is impossible for Simple ETA prediction to be accurate in all these situations.

An example of relation between stability and responsiveness for Simple ETA algorithm is shown on figure 3.7. In this example we artificially simulate change in download conditions. For the first 500 seconds download speed is equal to a constant value with 50% uniform random deviation. At $t = 500s$ the download conditions change and download rate increases by factor of three. On the graph we show three Simple ETA predictions with different N parameter

29

($N = 15s$, $N = 60s$ and $N = 240s$). We see that the $Simple\_ETA(240)$ prediction is the most stable, the least influenced by transfer fluctuations. $Simple\_ETA(15)$ prediction varies most intensively, but after the change of conditions it is the fastest to react and adjust to new download speed.

In implementations of Simple ETA algorithm the value of parameter N is chosen to balance the stability and responsiveness of the ETA prediction. Since the connection conditions during the real–world torrent download process have a tendency to change very often, a greater emphasis is placed on the responsiveness. Because of that the value of N is often set to be around 30 seconds.

3.3.4 Conclusion

The advantage of Simple ETA algorithm is its simplicity. It is easy implementable in real–life situations and thanks to lack of complex structure it is easy to analyze and adjust. Based on the discussion above we see that Simple ETA algorithm in most situations does not give an accurate prediction, it is biased and unstable. Because of relation between stability and responsiveness it is difficult to set it to work in many different situations that are expected to occur in real–life download processes.

3.4 ETA prediction quality measurement

One of the main goals of this thesis is to create an ETA prediction model. To evaluate the performance of any created model we need to have a measurement algorithm. Here we discuss the desired qualities of a ETA prediction that are important from the point of view of the practical usage and that should be included in the metric. Finally we explain in detail how for the purpose of this thesis the quality of a given ETA prediction will be measured.

Based on discussion of quality of Simple ETA algorithm (see Section 3.3), we consider the following properties important as a measure of quality of a ETA prediction:

- *Accuracy* – as defined by equation 2.1 in Section 2.1.4, the average distance between the actual download time and predicted ETA for every moment during the download time.

- *Stability* – the steadiness of $ETA(t)$ value in time of download.

- *Responsiveness* – ability to react and adjust the $ETA(t)$ prediction to the changing download conditions.

This thesis measures the quality of an ETA algorithm by analyzing different real–life torrent traces. We try to provide traces that are representative for the most of torrent usages. Additionally we study border–case situation traces. For each trace we give the evaluation of prediction quality by discussing its accuracy, stability and responsiveness. Since definitions of these properties are impractical in real–life application, when analyzing these properties against torrent traces we use the following measurement methods:

- *Accuracy* – The definition given by equation 2.1 is impractical, as it is very difficult to repeat the same global state of the real–life torrent in

order to calculate the expected values. For real–life torrent traces we instead measure accuracy of the prediction in two ways. First we compare the $ETA(t)$ value for every t with the actual time of the download T. This is one of the most important properties, as T is an exact value that is the subject of estimation. The disadvantage of that method is that for given time t it takes into account download condition changes after time t that were a random value before it. Therefore this method does not reflect exactly the definition from the equation 2.1. To address this problem, as the second measurement method we compare the $ETA(t)$ given by the algorithm with the $Simple\_ETA(30, t)$ prediction. This way we will get insight into torrent download conditions in $[0, t]$ interval and therefore the actual accuracy of $ETA(t)$.

- *Stability* – The fluctuations of ETA prediction in time t could be measured as $Average(N, t, ETA(t)')$. It is important to notice, that these fluctuations should decrease in time since they are proportional to the amount of data that is to be downloaded. Because of that we refrain from using the formula above and instead will measure stability of ETA prediction by observing the changes of $ETA(t)$ and having in mind the relation between stability and data to download described above.

- *Responsiveness* – To measure responsiveness of ETA prediction we collect the information about major changes in download conditions (like connections/disconnections with power seeds) and changes in total download speed. We compare timing of these events with changes in ETA prediction. Depending on the behavior of $ETA(t)$ after the event we will discuss how much time is needed for the prediction to take it into account.

Chapter 4

Model

This chapter presents the mathematical model of a selected peer's interaction with a connected peers that results in a local view ETA prediction algorithm. We start with stating taken assumptions, then we give the concept of the ETA prediction as based on analysis of separate sessions. We explain in detail the model of session analysis and introduce the prediction algorithm for the future data download rate within the session. Finally, we discuss an extension to the model that takes into account peer exchange process described in 3.1.1.

4.1 Model basics

The first step in creating an ETA prediction model is to define its domain of application. In this section we describe the global properties of the model. We state assumptions about the torrent and download process that will be needed by the model. We reference the influences of the download process presented in Section 3.1 and describe their impact on the model. Finally, we give the main model details and the concept of ETA prediction based on session analysis.

4.1.1 Assumptions and limitations

Among many influences of the download process stated in Section 3.1, there are many situations that are caused directly or indirectly by users. Those influences are not interesting from the point of this thesis as they do not give insight into BitTorrent. Many of them can be prevented or their impact on download process can be decreased. To exclude these issues from ETA predicting we restrict our analysis to the situations where their influence is reduced to a minimum. We define a *clean torrent trace* as a torrent trace that have been gathered in special conditions preventing unwanted user related influences. Hereafter for the purposes of the model definition and its evaluation we will work only with clean torrent traces.

Definition 4.1 (Clean Torrent Trace) *A torrent trace is considered to be a 'clean torrent trace' if during its collection the following conditions have been met:*

- Public tracker – *downloaded torrent is announced by a public tracker on a public web page.*

- No multiple torrent download – *during the total time of collecting traces there was only one BitTorrent client working on the machine and it was downloading only one torrent.*

- Minimal other network activity – *during the total time of collecting traces all network activity of the machine unrelated to BitTorrent is decreased to minimum. This means that there are no other user application that consume important network resources, like web browser, VoIP or instant messaging applications.*

- Internet connection quality – *the user's Internet connection provides stable and constant download and upload rates, that do not influence the download process.*

- No BitTorrent client user events – *during the total trace collection the user does not interfere with the download process by changing the Bit-Torrent client settings, torrent download settings or pausing/unpausing the torrent.*

- BitTorrent settings – *the BitTorrent client downloads full torrent content, without setting download priorities and upload/download rate limits.*

- No user interference – *during the total time of collecting traces the user does not interact with the machine in a way that can influence the download process, in particular he does not hibernate or restart the computer and does not run other applications that result in major system resources consumption.*

The most important assumption about the model is that it has to be based on a local view of the torrent, as described in Section 2.1.2. This requirement is a main idea of this thesis as it enables to create an implementable ETA prediction algorithm based on the model. Therefore the model is based only on information that can be obtained by a any peer by means of the BitTorrent Protocol.

There are many different implementations of BitTorrent clients that use different data exchange quality policies [3, 11]. In this thesis our goal is to create an ETA prediction algorithm that is independent of the BitTorrent client and can be implemented in any of them. Therefore in the model we assume to have no client–specific knowledge about the download process. In particular we assume that we do not know the policy used for choke/unchoke mechanism.

In this thesis we address the local view ETA prediction problem by splitting the download process into single sessions. The fundamental assumption taken during the model creation is that interactions between the selected peer and a several other peers are independent from each other. This assumption is not met in general, as multiple sessions compete for the same resources and a comparison between them is used by choke/unchoke algorithm. However, in many real–life situations it is a close approximation of the real behavior. In most examples the maximal usage of download and upload capacity is not reached, correlation between sessions is relatively small and this assumption

should not affect the download time prediction. What is most important, taking that assumption gives much simplification to the model and the ETA prediction algorithm based on it.

The goal of this thesis is to create a model that will to work in any real–life situation, therefore it has to give ETA prediction in all situations that can occur. However, to evaluate its performance properly we need to minimize the interference of non–BitTorrent events and collect the traces in the clean torrent environment.

We acknowledge that a selected peer can be connected with peers using different BitTorrent clients and do not download torrents in clean trace environments. We allow a selected peer and its connected peers to use any BitTorrent extension possible (see Section 3.1.6), but since they are not our main interest we assume that their impact on download process is small and can be ignored. By the same token we will assume that the behavior of connected peers is constant in a way that it does not show any periodic behavior such as described in Section 3.1.8. Additionally, since we focus on a local view ETA prediction, we will assume that global torrent evolution occurs, but is slow as observed during the download time and it therefore does not have a noticeable impact on it.

4.1.2 ETA prediction algorithm

The idea behind the ETA prediction presented in this thesis is to analyze the history of the download process in terms of sessions. At every time $0 \leq \tau \leq T$ we predict the future download transfer rate $s_\tau(t)$ of a torrent by summing up predictions given for separate sessions. We take into account all sessions that have been started before time τ regardless of the current state of connection. For each session i, based on the data exchange history with a connected peer, we use the model to estimate the future data download speed as a function of time $s_{i,\tau}(t)$.

$$
\begin{aligned}
s_{i,\tau} : [0, 1, \ldots, \infty] &\rightarrow [0, MaxTransfer] \\
s_\tau(t) &= \sum_i s_{i,\tau}(t)
\end{aligned}
$$

$MaxTransfer$ is a maximal download speed limit. The value $s_{i,\tau}(t)$ represents the predicted transfer rate within session i in t seconds in the future based on prediction calculated in time τ.

To predict the download time of the torrent T we calculate the moment in time when, based on predicted download speed $s_\tau(t)$, the total torrent content will be downloaded.

$$
\begin{aligned}
ETA(\tau) &= \tau + \min\{t \in \mathbb{N} : \sum_{j=0}^{t-1} s_\tau(j) \geq fileSize\} \\
&= \tau + \min\{t \in \mathbb{N} : \sum_i \sum_{j=0}^{t-1} s_{i,\tau}(j) \geq fileSize\}
\end{aligned}
$$

Since functions $s_{i,\tau}(t)$ are returned by the model, the minimum can be easily calculated with a simple bisection algorithm [12].

It is important to notice that the predicted download rate $s_{i,\tau}(t)$, even though it is calculated based on session i, does not only represent the future data exchange within this session. It is possible that in future the connection with a remote peer responsible for session i will be choked and replaced with a new session. Therefore in prediction $s_{i,\tau}(t)$ we try to include future data exchange within session i and sessions that will occur in its replacement.

In the following sections we explain the details of the session model for calculating the $s_{i,\tau}(t)$ download predictions.

4.2 Session model

Previously in Section 4.1.2 we have presented the algorithm of ETA prediction based on separate session analysis. In this section we give details about the model used to calculate the predicted download rate $s_{i,\tau}(t)$ for any given session.

4.2.1 Session states

In Section 3.2 we have discussed the details of session data exchange. Based on the obtained results during the download process in each session we can distinguish several different intervals in time that represent specific data exchange behavior. The first step in prediction of $s_{i,\tau}(t)$ is to classify different behaviors and give rules for such classification that can be implemented for real–time torrents.

In this thesis we distinguish four different data exchange behaviors which are represented as four *states*. For each session at every moment of time t we will assign a state $state(t)$ indicating the current behavior. We consider that a session at any time t is in one of the following states:

- TRANSFER – the session has a continuous non–zero download rate. This state represents the stable download periods.

- UNCHOKED – the selected peer currently receives data from its remote peer, but the download rate recently was zero. This state represents unstable download periods, like being unchoked by the remote peer.

- CHOKED – the download rate is zero, but recently was higher. This state represents unstable, temporary disconnection periods, like a brief disconnection or choking by the remote peer.

- DISCONNECTED – the download rate has been zero for a given period of time. This state represents stable, permanent lack of download from remote peer.

We see that the presented states can be divided according to their stability and data download (Table 4.1).

	Download	No download
Stable	TRANSFER	DISCONNECTED
Unstable	UNCHOKED	CHOKED

Table 4.1: Classification of the states according to their stability and download rate.

4.2.2 State calculation

During the torrent download time, at every time t we calculate the $state_i(t)$ for each started session i. The calculation value of $state(t)$ is done only at time t, according to data exchange history in period $[0, t]$ and never recalculated again.

To calculate current session state at time t we focus on the recent download rate within the session and analyze its stability and current speed. For example, following the given definition we want the algorithm to compute $state(t) = TRANSFER$ if there is a current download transfer at time t and in the recent past it has been stable.

Before we give exact rules for calculating $state_i(t)$, we define following functions based on the download rate $s_i(t)$:

$$
\begin{aligned}
downloadTime(t) &= \max\{j \in \mathbb{N} : \forall t' \in (t-j, \ldots, t] s_i(t') > 0)\} \\
disconnectionTime(t) &= \max\{j \in \mathbb{N} : \forall t' \in (t-j, \ldots, t] s_i(t') = 0)\} \\
connectionMade(t) &\iff \exists t' \in [0, t] s_i(t') > 0
\end{aligned}
$$

The $downloadTime(t)$ (resp. $disconnectionTime(t)$) is equal to the length of the current download (resp. disconnection) period. Function $connectionMade(t)$ is true only if up to time t there has been a moment of non–zero download rate. It is important to notice that at any time t if $connectionMade(t) = true$ we have that one and only one function between $downloadTime(t)$ and $disconnectionTime(t)$ has a non–zero value.

$$
\begin{aligned}
connectionMade(t) &\implies \\
\implies \quad &((downloadTime(t) > 0 \wedge disconnectionTime(t) = 0) \vee \\
\vee \quad &(downloadTime(t) = 0 \wedge disconnectionTime(t) > 0)) \qquad (4.1)
\end{aligned}
$$

The model calculating $state_i(t)$ takes two parameters. First TT is the minimal duration for stable transfer period. Second, DT is minimum duration with no transfer to be considered as a stable disconnection. To calculate at time t the current state of session i we use the following algorithm:

- If $connectionMade(t) = false$, then we still have not started the download process and we return $state_i(t) = DISCONNECTED$.

- If $connectionMade(t) = true$ and $downloadTime(t) > 0$, then there is a non–zero download rate. If $downloadTime(t) > TT$ we consider to have stable download transfer and return $state_i(t) = TRANSFER$, otherwise $state_i(t) = UNCHOKED$.

- If $connectionMade(t) = true$ and $disconnectionTime(t) > 0$ than the current download rate is zero. If $disconnectionTime(t) > DT$ we believe

37

that the lack of download transfer is permanent and therefore return $state_i(t) = DISCONNECTED$, otherwise we set the current session to $CHOKED$.

Thanks to relation 4.1 we see that this algorithm returns one and only one value at any time t.

An example of results generated on a real–life session by the presented algorithm is shown in Figure 4.1. We see that during the presented session there was no download rate up to time $t \sim 12400$ and therefore calculated state for that period is $DISCONNECTED$. At this moment the connected peer started sending data and that resulted in state change to $UNCHOKED$ for a period of $TT = 30$ seconds. Because transfer continued after that time the state has been changed to $TRANSFER$. Due to next changes in download rate the calculated state in time changed between $CHOKED$ and $UNCHOKED$. When the download rate finally stabilized after $t = 12900$ there occurred another period of $TRANSFER$ state. Around time $t = 13800$ the download rate decreased to zero and stayed that way until the end. We see that after this event there for a period of time the session is in $CHOKED$ state, which after time $DT = 600s$ changes into stable $DISCONNECTION$ state.

By analyzing this algorithm we see that there is a relation between value of $state_i(t)$ and a possible value of $state_i(t+1)$. For example, if the session is in state $TRANSFER$ in successive moments of time it can stay in it longer or change only into state $CHOKED$. The possible transitions between the states are shown in Figure 4.2. It is worth mentioning that according to the presented algorithm a session can stay in stable states infinitely, while in states $UNCHOKED$ and $CHOKED$ it can stay at most for TT and DT respectively.

4.2.3 Download rate prediction

At time τ we calculate the predicted download transfer $s_{i,\tau}(t)$ within session i based on the state history of the session in time period $[0, \ldots, \tau]$. By analysing the state history we can calculate how long the session has remained in any given state and what the average download transfer rate has been. The main idea behind the prediction is to use these data to estimate how long the session will stay in its current state. What is more, by knowing the possible changes of states (Figure 4.2) in some situations we can predict what the next session states will be. Since we also have the knowledge of the average transfer rate per session, we have the grounds to give download transfer prognosis.

The prediction of download transfer $s_{i,\tau}(t)$ is divided into two parts.

- *Short–term prediction* – this prediction is based on current session state and recent download transfer rate. We assume that the current download conditions will continue in the near future, later followed by state changes according to Figure 4.2. The short–term prediction consists of a list of time periods and expected average download speed for each of them.

- *Long–term prediction* – this is the predicted download rate that will start after the period of short–term prediction, and last to infinity. It is based on long–term session download rate analysis and not upon current state
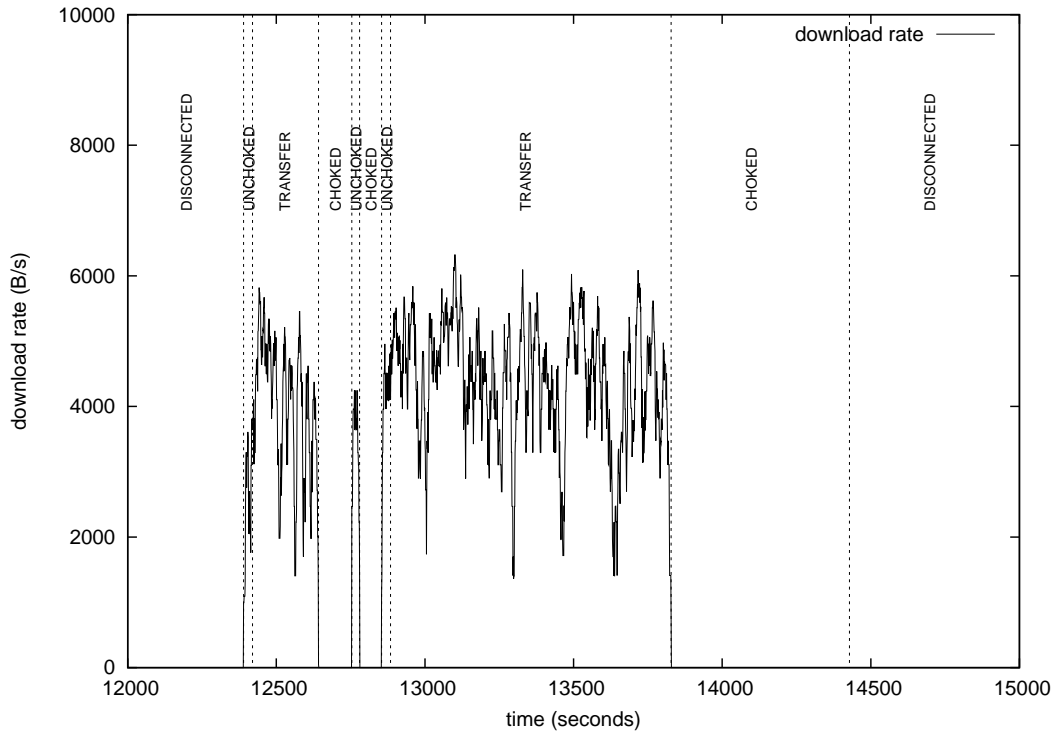
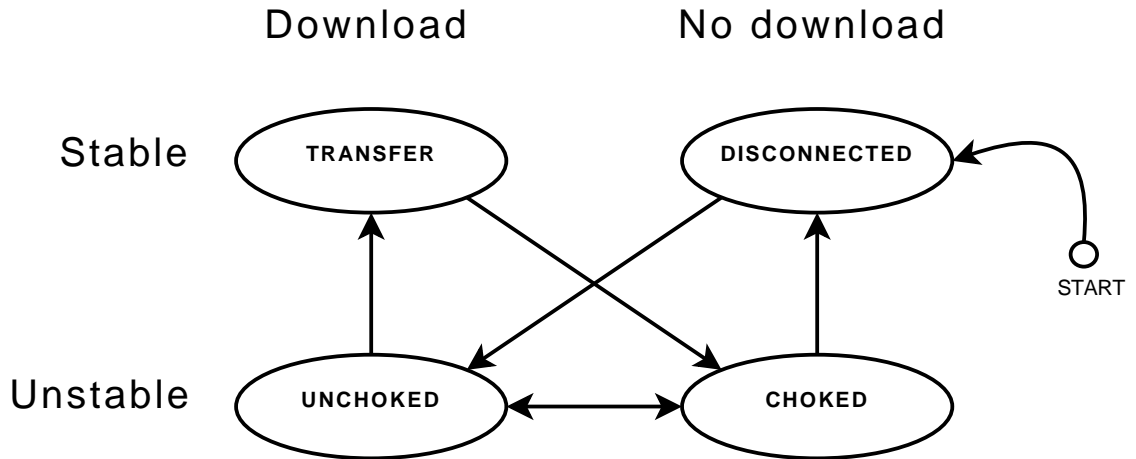Figure 4.1: An example of real–life session and the calculated *state*(*t*).

Figure 4.2: Diagram presenting possible changes in state of a given session.

or state history. Since it is a prediction for the 'far future', predicting the timing of any future changes is pointless. Therefore long–term prediction is a constant download speed value.

The total prediction will be given by combining the long and short–term predictions. If $I_1, \ldots, I_n$ are the short–term intervals lengths, $S_1, \ldots, S_n$ are average transfer rates respectively and S is long–term prediction then $s_\tau(t)$ is computed as following:

$$
s_{i,\tau}(t) = \begin{cases} S_i & \text{if } \sum_{j=1}^{i-1} I_j \geq t > \sum_{j=1}^{i} I_j \\ S & \text{if } t \geq \sum_{1}^{n} I_j \end{cases}
$$

An example of possible prediction $s_{i,\tau}(t)$ is shown in Figure 4.3. We see the prediction given in time τ which has two short–term period lasting I_1 and I_2. The predicted download speed in these periods is S_1 and S_2 respectively. After time $\tau + I_1 + I_2$ a long–term download prediction occurs with download rate S.

4.2.4 Prediction algorithm

In this section we give detailed algorithm for calculating the short and long–term prediction for a given session. We start by introducing following functions necessary for the algorithm.

- $averagePeriodInState_\tau(state)$ – the average length of each period that session spent in a given $state$ between time 0 and τ.

- $averageTransferInState_\tau(state)$ – the average value of the download rate when session was in a given $state$. The average is calculated on last AST moments in time when session was in a given $state$.

- $averageTransfer_\tau$ – the average value of the session download rate for last ADT moments in time (regardless of state) since the start of download process (since time t when $connectionMade(t) = true$).

The values of AST and ADT are parameters of the model.

First, we give the algorithm for calculating the $s_\tau(t)$ prediction when the connected peer is a seed. The prediction given in moment τ depends on the value of $state(\tau)$ as following:

- $state(\tau) = TRANSFER$:
  The session is in a stable transfer state. According to Figure 4.2 we see that this state is always followed by $CHOKED$ state. Therefore we give the following prediction with two short–term periods:

$$
\begin{aligned}
I_1 &= averagePeriodInState_\tau(TRANSFER) \\
I_2 &= averagePeriodInState_\tau(CHOKED) \\
S_1 &= averageTransferInState_\tau(TRANSFER) \\
S_2 &= averageTransferInState_\tau(CHOKED) \\
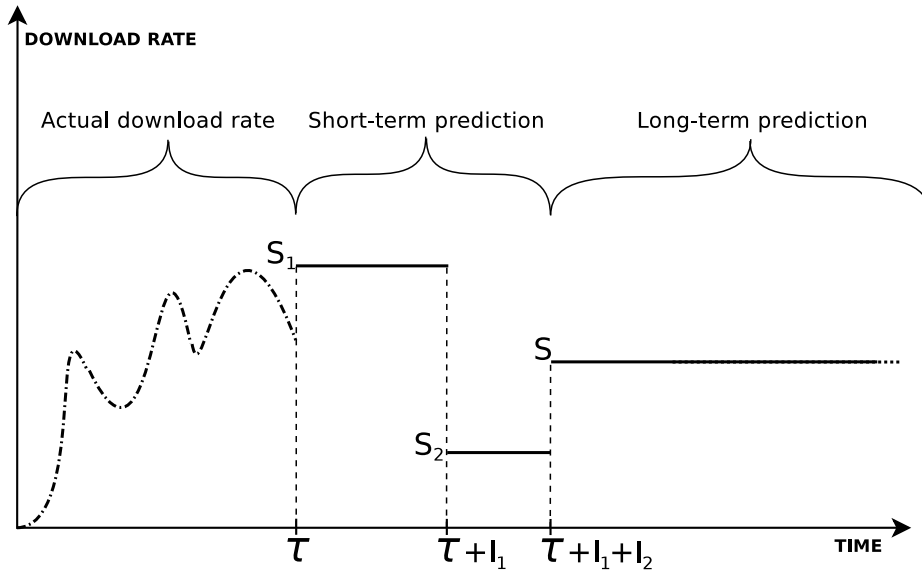S &= averageTransfer_\tau
\end{aligned}
$$

40

Figure 4.3: An example of a possible download rate prediction.

- $state(\tau) = UNCHOKED$:
  The session is in an unstable transfer state. Its next state could be either $TRANSFER$ or $CHOKED$. We give the following prediction with one short–term period:

$$
\begin{aligned}
I_1 &= averagePeriodInState_\tau(UNCHOKED) \\
S_1 &= averageTransferInState_\tau(UNCHOKED) \\
S &= averageTransfer_\tau
\end{aligned}
$$

- $state(\tau) = CHOKED$:
  The session is currently not receiving any data. We assume that the no transfer period is only temporary, but it is ambiguous what the next state will be. We give the following prediction with one short–term period:

$$
\begin{aligned}
I_1 &= averagePeriodInState_\tau(CHOKED) \\
S_1 &= averageTransferInState_\tau(CHOKED) = 0 \\
S &= averageTransfer_\tau
\end{aligned}
$$

- $state(\tau) = DISCONNECTED$:
  The session is in a stable disconnection state. We believe that the lack of download activity is permanent, therefore we give a prediction with no short–term period:

$$
S = 0
$$

One of the most important difference between leecher and seed from the point of view of predicting the future download rate is that the download rate from a leecher is dependent upon whether there is mutual interest in possessed pieces. Unlike a seed, when sending data to the selected peer a transfer from

41

a leecher can stop because it have lost interest or stopped to be interesting for the selected peer.

Since from the local view of the torrent we have the information about the pieces possessed by any connected peer, we can predict when the leecher will loose the interest in selected peer (or the other way around) by following in time the number of pieces the selected peer possesses that are interesting to the leecher.

The prediction for a leecher is very similar to the one for a seed. The only difference is in calculating the value of I_1 when the session $state(\tau) = TRANSFER$. By following both the number of pieces possessed by selected peer that are interesting for the leecher and the number of pieces possessed by the leecher that are interesting for the selected peer we can predict the time at which the data exchange will stop due to lost of interest by the selected peer or the leecher.

To include that into the model at time of prediction τ we analyze the past history focusing on piece possession. We calculate following values:

- $TIME\_TO\_CHOKE\_ME(\tau)$ – predicted time it will take for the connected peer to lose interest in selected peer's pieces. This is calculated by analyzing in time the number of pieces that connected peer was interested in and using linear regression to give the future forecast. The value of $TIME\_TO\_CHOKE\_ME(\tau)$ is equal to the duration of the predicted period when number of interesting pieces is greater than zero.

- $TIME\_BEFORE\_I\_CHOKE(\tau)$ – predicted time it will take for the selected peer to lose interest in connected peer's pieces. Similar to above, this is calculated by analyzing in time the number of pieces that selected peer was interested in and using linear regression to give the future forecast. The value of $TIME\_BEFORE\_I\_CHOKE(\tau)$ is equal to the duration of the predicted period when number of interesting pieces is greater than zero.

Linear regression in both cases is calculated based on a latest 120 values.

We have now three values that predict the timing of the end of transfer state. First (as for seed) is the $averagePeriodInState_\tau(TRANSFER)$ that stands for disconnection based on regular peer activity. Other two are defined above $TIME\_TO\_CHOKE\_ME(\tau)$ and $TIME\_BEFORE\_I\_CHOKE(\tau)$ that stand for the disconnection due to lost of interest in pieces. The value of the duration of the transfer state I_1 for a leecher in $TRANSFER$ state is set as a minimal value of these three:

$$
\begin{aligned}
I_1 \;=\; min \quad & (TIME\_TO\_CHOKE\_ME(\tau), \\
& TIME\_BEFORE\_I\_CHOKE(\tau), \\
& averagePeriodInState_\tau(TRANSFER))
\end{aligned}
$$

To illustrate the algorithm we show in Figure 4.4 an example of a download rate prediction for a real–life session. As we see in the moment of prediction there was a download in process from the connected peer. Since the download rate above zero lasted for more than $TT = 30s$ in the moment of prediction the calculated state was $TRANSFER$. As we see the given prediction consists of
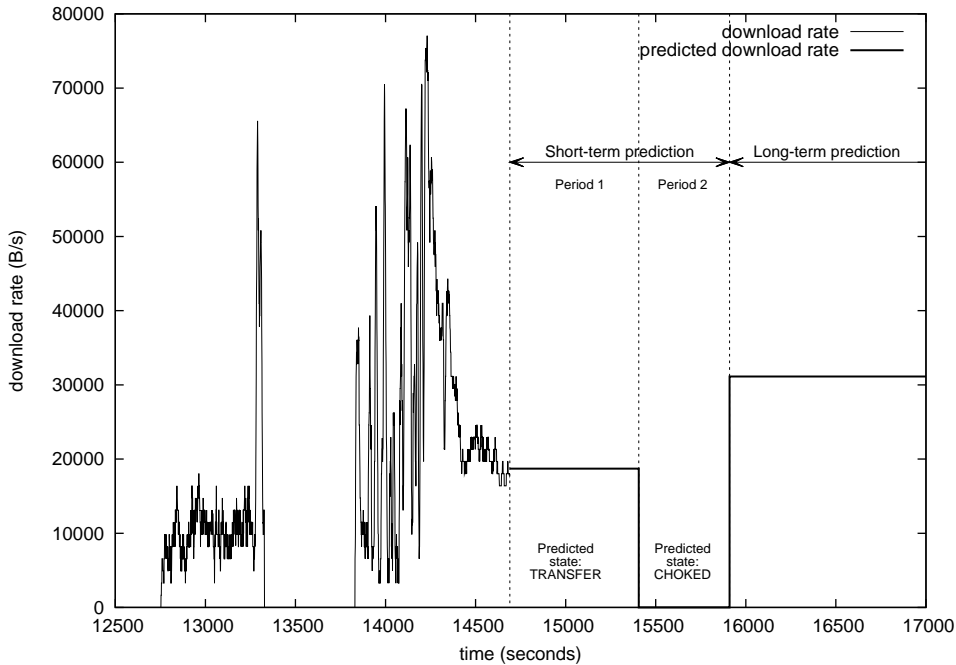
Figure 4.4: An example a download rate prediction given for a real–life session.

two short–term periods for states $TRANSFER$ and $CHOKED$ respectively and an infinite long–term prediction. Both of the short–term predictions are calculated based on the state history of the session. We see that the length of predicted state $TRANSFER$ interval is almost equal to the average period of positive download rate in the past. The length of $CHOKED$ period is predicted based on the average $CHOKED$ periods in the past. Since there was only one period of that kind (between $t = 13300s$ and $t = 13800s$) the length of the predicted interval is around $500s$. An interesting thing to notice is that in this example the predicted download rate S_1 in $TRANSFER$ short–term period is smaller than the long–term download rate S. This is because both values are calculated at the moment of prediction and at that time the average transfer in last period of $ADT = 600s$ used for long–term prediction was higher than the average transfer in last period of $AST = 120s$.

4.3 'Peer exchange' extension

The presented basic version of the model gives fair predictions in most of the real–life torrent examples. However, further analysis showed that in some cases the download time is underestimated. This happens mostly for torrents with high peer exchange rate (see Section 3.1.1). The cause of this problem is the value of model's parameter DT – disconnection time. When selected peer exchange the connected peers, their sessions are still considered by the model and remain in $CHOKED$. The predicted download rate for that state slowly decreases in time. After time DT these sessions states are changed to $DISCONNECTED$ and then the prediction for them is finally zero. This way for torrents with high peer exchange rate the model overestimates the future download rate and therefore gives decreased ETA.

To prevent the underestimation of the ETA we introduce an algorithm that calculates whether the torrent is rapidly exchanging connected peers and if that is true, it decreases the value of DT parameter. What is more, since the exchange rate of seeds and leechers are different, the algorithm calculates DT separately for them.

The value of DT is calculated based on the average seed (leecher) disconnection rate calculated on latest $DT\_INTERVAL$ period of time. If it is above the set limit DT_LIMIT, that we consider the torrent to be in fast peer exchange state and calculate the value of DT. If the torrent is below the limit, we return a constant value $DT_{default}$. We use the following formulas:

$$
dr(t, peerType) = averageDisconnectionRate(t, peerType, DT\_INTERVAL)
$$

$$
DT_{leecher}(t) = \begin{cases} DT_{default} & \text{if } dr(t, leecher) \leq DT\_LIMIT \\ DT_{default}\frac{DT\_LIMIT}{dr(t,leecher)} & \text{if } dr(t, leecher) > DT\_LIMIT \end{cases}
$$

$$
DT_{seed}(t) = \begin{cases} DT_{default} & \text{if } dr(t, seed) \leq DT\_LIMIT \\ DT_{default}\frac{DT\_LIMIT}{dr(t,seed)} & \text{if } dr(t, seed) > DT\_LIMIT \end{cases}
$$

We see the algorithm above to be an extension of the model presented in this chapter. During the model evaluation we will consider it to be optional and show its impact on some of the presented examples.

It is worth mentioning that this extension by setting the dependency between the value of DT and accumulated behavior of different sessions (disconnection speed) lifts the basic model assumption taken in 4.1.1 about the independence of the sessions.

Chapter 5

Evaluation

This chapter presents an evaluation of the model discussed in Chapter 4. The evaluation is done by focusing on a some real–life torrent traces and analyzing the ETA prediction given by the model. First, we explain how the traces have been collected and processed. Then we give a list of torrent examples that are the subject of further analysis. We conclude by discussing obtained results.

5.1 Implementation

In this section we focus on the details of implementation both the BitTorrent client collecting the traces and the application analyzing them.

5.1.1 Gathering data

To collect torrent traces we have instrumented a BitTorrent client. We have chosen Azureus for that purpose – a popular open–source client written in Java [13]. We have instrumented version 3.0.4.3_CVS of Azureus and deployed it in Linux environment to collect the traces. Every second the altered Azureus saves the data about the global torrent parameters and information about each active session into a trace file.

The global torrent parameters are the following:

- TIME – current time of the download in seconds, starting from zero.
- AZUREUS_ETA – current prediction given by Azureus' Simple_ETA(30) algorithm.
- REMAINING_SIZE – size of content left to be downloaded (in bytes).
- TOTAL_SIZE – total size of the torrent content.
- TOTAL_PIECES – total number of pieces that the torrent content is divided into.
- PIECES_DOWNLOADED – number of pieces that have already been completely downloaded.
- CONNECTED_LEECHERS_COUNT – number of currently connected leechers.
- CONNECTED_SEEDS_COUNT – number of currently connected seeds.
- TOTAL_LEECHERS_COUNT – total number of leechers in the swarm.

- TOTAL_SEEDS_COUNT – total number of seeds in the swarm.

For each session that Azureus keeps track of, we log the following data:

- IP – the IP address of the connected peer.
- IS_SEED – boolean information whether the connected peer is a seed.
- PIECES_ONLY_PEER_HAS – number of pieces that the connected peer possesses but the selected peer does not (the number of interesting pieces for the selected peer).
- PIECES_ONLY_WE_HAVE – number of pieces that the selected peer possesses but the connected peer does not (the number of interesting pieces for the connected peer).
- PIECES_WE_BOTH_HAVE – number of pieces that both the selected and the connected peer possess.
- RECEIVE_FROM_PEER_RATE – download transfer rate from the connected peer.
- SEND_TO_PEER_RATE – upload transfer rate to the connected peer.
- IS_CHOKING_ME – boolean information whether the connected peer is choking the selected peer.
- IS_CHOKED_BY_ME – boolean information whether the selected peer chokes the connected peer.

5.1.2 Analysing data

The collected traces are analyzed by a stand–alone Java application that implements the described local view model. Different values of model parameters can be set for every analysis. Additionally Perl scripts have been created to extract the detailed information from the traces about each session and the number of leecher and seed connection.

5.2 Experiment

This section describes the model settings and torrent traces used in the experimental model evaluation.

5.2.1 Model settings

In the experimental model evaluation we have analyzed the collected traces with the application implementing the model. As stated in chapter 4 the model has several parameters. During the analysis process, unless stated otherwise, the default settings have been used (Table 5.1).

In some torrent examples we show the impact of these values on the prediction given by the model.

5.2.2 Selection of torrents

Table 5.2 shows the list of the torrents that have been chosen for the evaluation of the model. They differ in size, time of download and swarm size. The

Parameter	Value
$DT_{default}$	600 s (10 min)
TT	30 s
AST	120 s (2 min)
ADT	600 s (10 min)
DT_LIMIT	5 disconnections / min
$DT\_INTERVAL$	1800 s (30 min)

Table 5.1: Default settings of the model's parameters.

#	Torrent Name	Duration	Size	Sessions	Leechers	Seeds
1	One Power Seed	18799s	315.19 MB	461	82	152
2	Multiple Power Seeds	156368s	528.97 MB	39	3.7	2.5
3	Stable Peer Exchange	8225s	1.36 GB	582	793	208
4	Unstable Peer Exchange	17681s	1.56 GB	1274	2327	2.7
5	Very Stable Peer Exchange	80195s	702.19MB	270	47.7	9.0

Table 5.2: List of torrent traces that have been used for the model evaluation.

examples below are a good representation of the real–life torrents. We analyze each of them in detail.

In the table above *Sessions* is the total number of all sessions that have started during the torrent download time, including these where no data nor information were received. The values of *leechers* and *seeds* are equal respectively to the average number of leechers and seeds in the swarm during the download time.

5.3   Results

5.3.1   Torrent 1 – One Power Seed

The first analyzed example is a medium size torrent. It can be observed from the swarm size and leecher–to–seed ratio that the torrent is quite popular and its global evolution was balanced. During the download time the torrent was relatively stable – the number of leechers increased from 58 to 95, while the number of seeds oscillated between 145 and 160. During the torrent download 92.5% of the data have been retrieved from seeds.

This example is a typical torrent trace. Its swarm is stabilized and consists of many seeds. During the download time the selected peer is connected to similar number of seeds and leechers, but the vast majority of the downloaded content come from stable sessions with seeds. Those qualities are commonly observed in downloaded torrents, therefore making the analysis of this example very important.

The comparison of model ETA and Azureus Simple_ETA(30) is shown in Figure 5.1. In this example model ETA gives better ETA forecast than the Simple ETA used by Azureus. On the graph we can see that the prediction given by the model is much more stable than the one given by Azureus. This is clearly visible after time $t = 4000s$ when the Azureus ETA highly oscillates
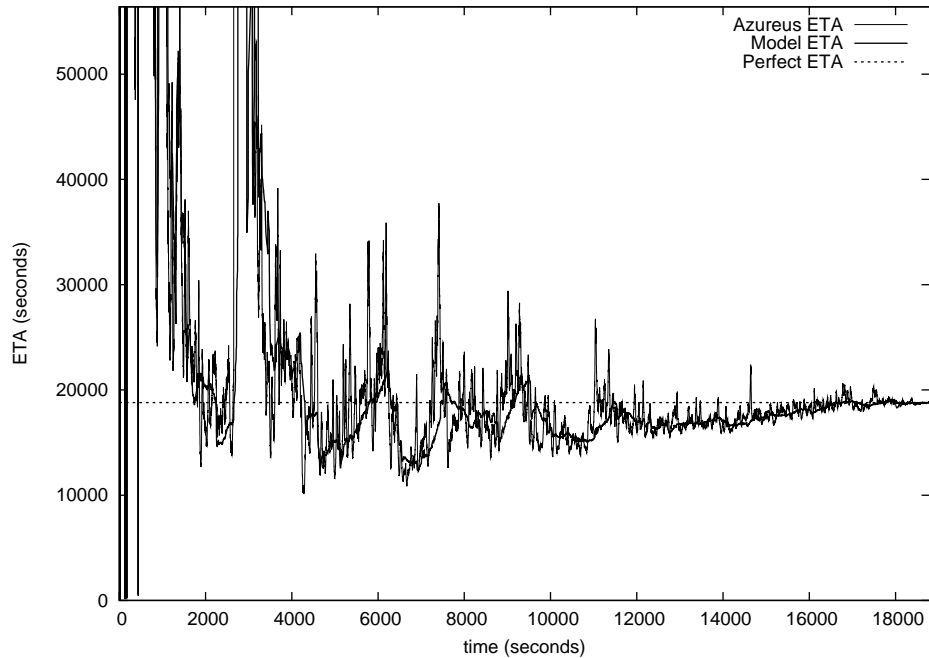
Figure 5.1: Comparison of model ETA and Azureus Simple_ETA(30) for Torrent 1 (One Power Seed).

around the values given by model ETA. What is more, model ETA is more accurate.

It is interesting to focus on the distribution of the downloaded data among the sessions in this torrent. Even though there were more than 400 sessions, only 88 of them downloaded more than 0.1% of the total torrent content. Most of the content was downloaded from seeds and there was one Power Seed that sends more than 25% of the content. The second best peer was also a seed and sent 6.9%. The third result belonged to a seed with 4.9%. The download speed of these three seeds over time is shown in Figure 5.2.

Another thing important for this torrent is a network disconnection that occurs around $t = 2600s$. We can see in Figure 5.1 that both predictions increase rapidly at that time. This can be also observed in Figure 5.3, where at time $t = 2554s$ the number of connected peers rapidly falls to zero. This shows that even in a controlled environment it is possible to get external non–BitTorrent influences.

To discuss the responsiveness of the model we will select few events (Table 5.3) that have happened during the torrent download and compare the reaction of both the model ETA and Azureus Simple_ETA(30). The chosen events are marked in Figure 5.4. The first event is a network disconnection. We see that the Simple ETA reacted very quickly, while model ETA took almost four times longer to give an almost infinite prediction. This is because Simple ETA needs 30 seconds to adjust to zero download transfer, while the model gradually decreases the session prediction and changes the sessions state from CHOKED to DISCONNECTION.

Events 2 and 5 are an increase in torrent download speed due to peers starting sending data to selected peer. In the first one the power seed that
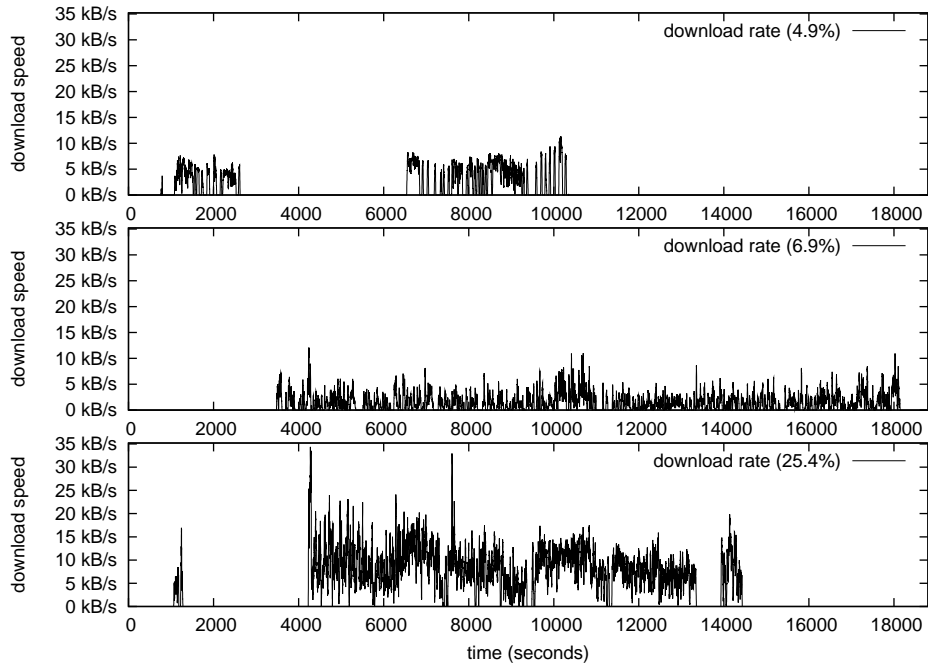
48

Figure 5.2: The download speed as a function of time for three sessions with the largest downloaded data size.
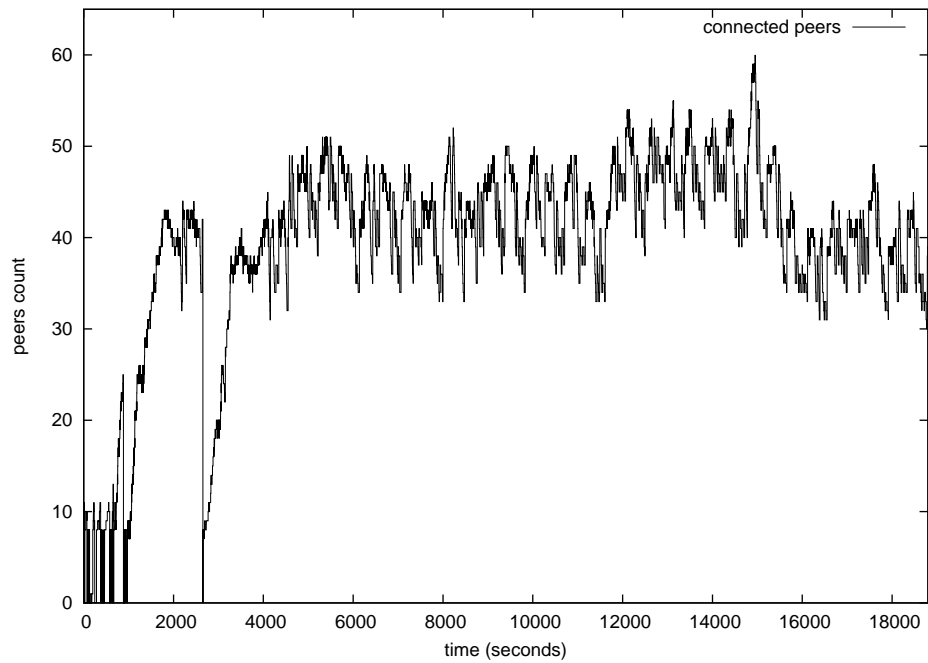

Figure 5.3: The number of connected peers during the time of download.

#	Event	Time	Comment
1	Network disconnection	2624s	The download rate drops to zero.
2	Seed reconnection	4225s	The 25.4% seed after a period of zero download rate started transferring data.
5	Seed connection	6459s	A seed have connected the selected peer and started sending data with $6kB/s$.
3	Speed drop	7409s	The 25.4% seed download during the oscillation have fallen to zero.
4	Speed maximum	7607s	The 25.4% seed download during the oscillation have rapidly increased up to $33.7kB/s$.

Table 5.3: List of events from torrent 1 - One Power Seed.

	Connection rate:	Disconnection rate:
Total:	13.45 / min	13.33 / min
Seeds:	1.62 / min	1.53 / min
Leechers:	11.82 / min	11.79 / min

Table 5.4: Connection and disconnection rates for Torrent 1 - One Power Seed.

transferred more than 25% of data during its session again starts to send data, the second event is a seed connecting for the first time. Both of these events can be observed in Figure 5.4 as a sudden drop in Simple ETA prediction. We see that model ETA reacts slightly slower than Simple ETA. The reaction on the event 5 is stronger than on event 2 as the first one is a new connection and therefore the long–term prediction for this session is not decreased by a period of silence that we have in power seed session. The reaction on event 5 is almost as fast as Simple ETA's. It is interesting to notice that while Simple ETA reacts with a rapid drop and after time restores the balance, the model ETA's reaction on these events is more smooth.

Finally, events 3 and 4 are fluctuations in power seed download rate. We see that Simple ETA reaction on these events is rapid. The model ETA prediction seems almost unaffected by these events. It is interesting to compare the reaction on event 4 and 5 for both of the predictions. Although these two events seem similar when comparing the Simple ETA reaction (both of them are sudden increase in download rate), the reaction of the model ETA is completely different. We see that the model rapidly adjusted to the new situation after event 4, predicting that the new session as a result of the connection will last, but it almost ignored the event 5 on the grounds that this is just a fluctuation in power seed download rate.

The average connection and disconnection rates are shown in Table 5.4. We see that the seed disconnection speed is lower that DT_LIMIT, while
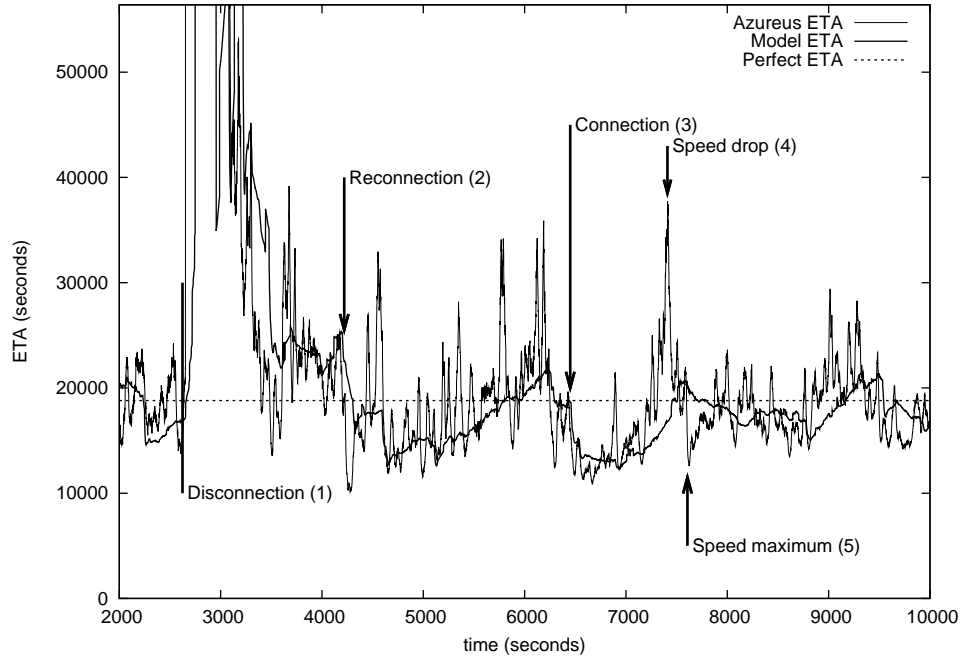
Figure 5.4: Model ETA and Azureus Simple_ETA(30) for Torrent 1 (One Power Seed) for a limited period of time with marked events.

leecher disconnect more than twice faster than DT_LIMIT and therefore the $DT_{leecher}$ is changed by peer exchange extension. However, since majority of data is downloaded from seed, it does not have noticeable impact on the given ETA prediction.

In conclusion, we have that in example of torrent 1 (One Power Seed) the prediction given by the model is better than the one given by Simple ETA. The model's prediction is slightly more accurate and far more stable. Its responsiveness, even though weaker than Simple ETA's, is still very good.

5.3.2 Torrent 2 – Multiple Power Seeds

Unlike the first example, this torrent's swarm size is very small. The number of seeds and leechers varies between one and eight. The download time is very long (almost two days) which is ten times longer than for the first analyzed torrent.

The percentage of data downloaded from seeds is 64.5%, which less than in torrent 1 but still high. The distribution of the downloaded data among the sessions is more even than for the previous torrent, but still the majority of the content is downloaded from a minority of the peers. The top five peers according to the downloaded data size are shown in Table 5.5. We see that almost half of the content has been received from these five power seeds, therefore we refer to this trace as a 'Multiple Power Seeds' trace.

Before we analyze the ETA prediction, let us focus on the changes in time in the number of connected peers (Figure 5.5). We see that this value changes rapidly in time. Because the swarm size is small, during the download time on average Azureus was connected to only seven peers.

51

#	data received	session type
1	14.1%	seed
2	10.1%	seed
3	8.6%	leecher changing to seed
4	8.0%	seed
5	7.1%	seed
TOTAL:	47.9%	

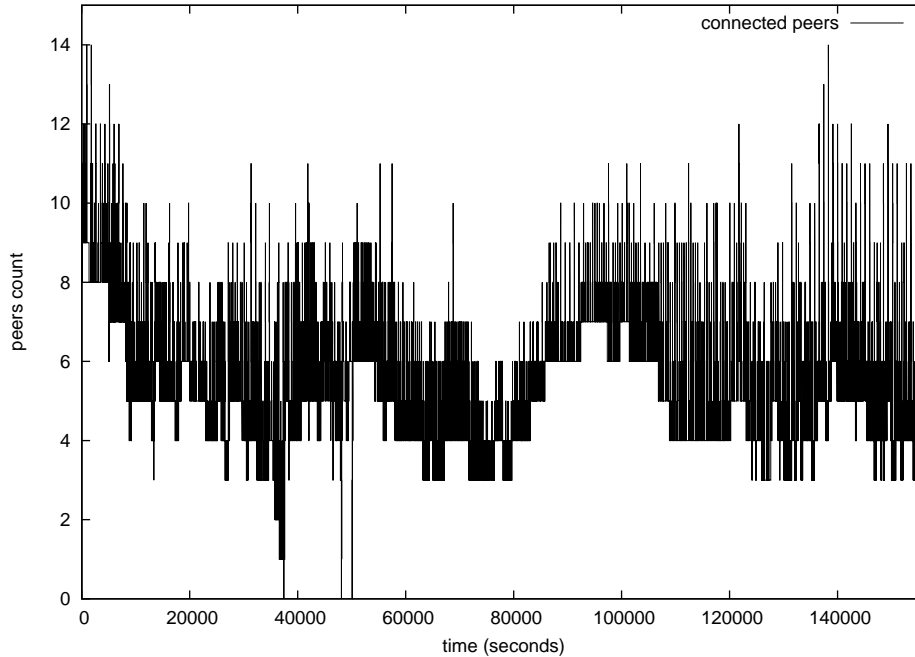Table 5.5: List of top five peers according to the downloaded size for Torrent 2 - (Multiple Power Seeds).

Figure 5.5: Number of connected peers in time for Torrent 2 (Multiple Power Seeds).

Since the number of peers is small, every connection has a important contribution into the download rate. Because of that, every change in number of connections has a huge impact on the total download speed. This can be seen in the Simple_ETA(30) prediction in Figure 5.6, where the calculated ETA rapidly changes by the orders of magnitude, making the download time forecast extremely unstable and therefore unusable.

In comparison the prediction given by the model is far more stable. Although there are periods where in short time its value changes by a ratio of 2 (in period $65000s - 80000s$), in general the model ETA is resistant to wide download rate fluctuations (e.g. periods $90000 - 110000$ and $135000 - 150000$). What is more, as we see by comparison of the Figures 5.6 and 5.7, model ETA is more accurate than the Simple_ETA(30). Similar to torrent 1 example, in this trace we observe a very good responsiveness of the model ETA to the changes in download conditions.
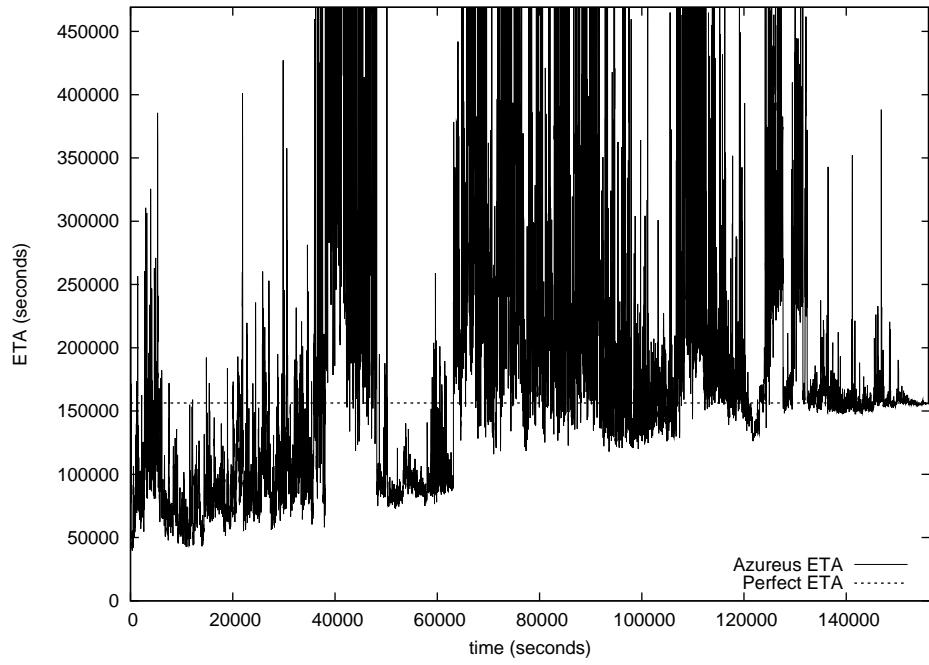
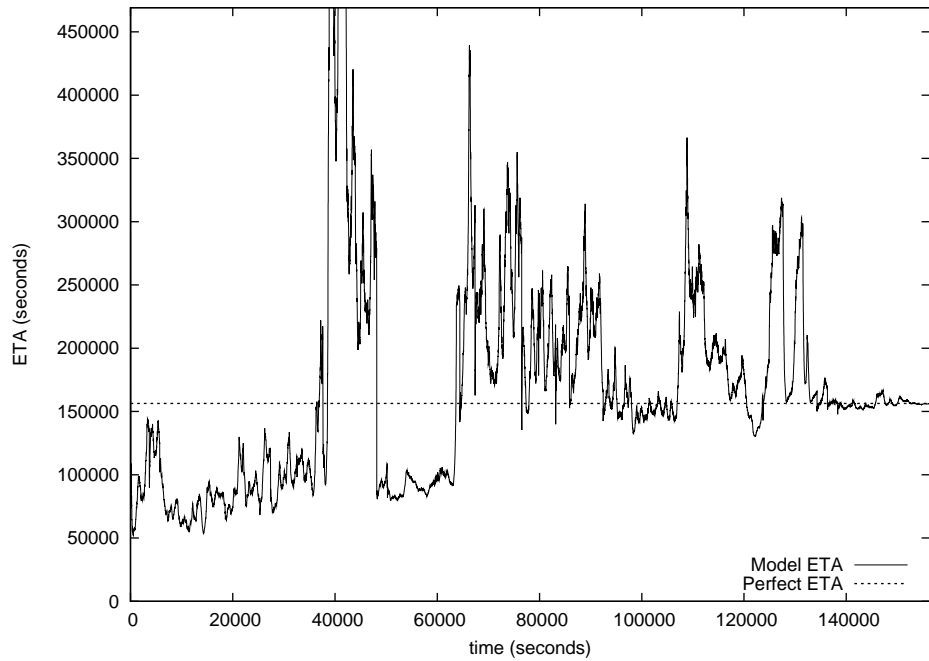Figure 5.6: Azureus Simple ETA(30) calculated for Torrent 2 (Multiple Power Seeds).

Figure 5.7: Model ETA calculated for Torrent 2 (Multiple Power Seeds).

53

	Connection rate:	Disconnection rate:
Total:	1.44 / min	1.44 / min
Seeds:	0.02 / min	0.02 / min
Leechers:	1.42 / min	1.42 / min

Table 5.6: Connection and disconnection rates for torrent 2 - One Power Seed.

The average connection and disconnection rates for this trace are shown in Table 5.6. We see that disconnection rates both for seeds and leechers are much smaller than the disconnection limit constant $DT\_LIMIT = 5.0/min$. Based on these values we see that the sessions during this torrent were relatively stable. Because of that the peer exchange extension to the model had no impact on the ETA prediction during the total download time.

The torrent 2 example shows that again the model ETA prediction performance is better than the Simple_ETA(30). This is caused mostly because of the length of the torrent download time and the size of the swarm. Thanks to these factors the number of different sessions is small and the model has a lot of time to measure them. This results in far better stability and accuracy of the prediction. Simple_ETA(30) algorithm is not capable to benefit from the advantage of measuring the sessions and thus every change download speed (caused by disconnection or fluctuation) has an impact on its prediction, making it extremely unstable.

5.3.3   Torrent 3 – Stable Peer Exchange

Compared to the previous two examples, this torrent trace has the shortest download time, the largest size and the largest average swarm size. In that short period of time there were almost six hundred sessions. The swarm size was stable and had on average 793 leechers and 208 seeds.

Almost 50 percent of the content was downloaded from the seeds. 34.5% of the torrent were downloaded from one peer – a leecher that turned into a seed 10 minutes after the connection. Only 51 out of 582 sessions resulted in more than 0.1% of the content download.

Before we focus on the ETA prediction, let us discuss the number of connected peers in time for this torrent shown in Figure 5.8. We see that just after few minutes from the start of the download the number of connections reaches the Azureus limit of 50. With brief fluctuation this value remains until the end of download. This suggest that in this trace we have an occurrence of peer exchange. Unlike the regular tit–for–tat changes in the connected peer set where the number of connections varies in time, in peer exchange every disconnected (choked) peer is instantly replaced by a new one and the total number of connections reaches the maximal limit. This behavior happens in torrents with large swarm size, especially with high leecher–to–seed ratio, where the selected peer in its continuous search for optimal set of connections quickly browses trough a large number of leechers with poor data exchange quality.

Other parameters indicating the peer exchange are the connection and disconnection rates shown in Table 5.7. We see, in comparison with the values for torrent 2 in table 5.6, that these rates are almost thirty times larger and

	Connection rate:	Disconnection rate:
Total:	32.4 / min	32.4 / min
Seeds:	4.4 / min	4.3 / min
Leechers:	28.1 / min	28.1 / min

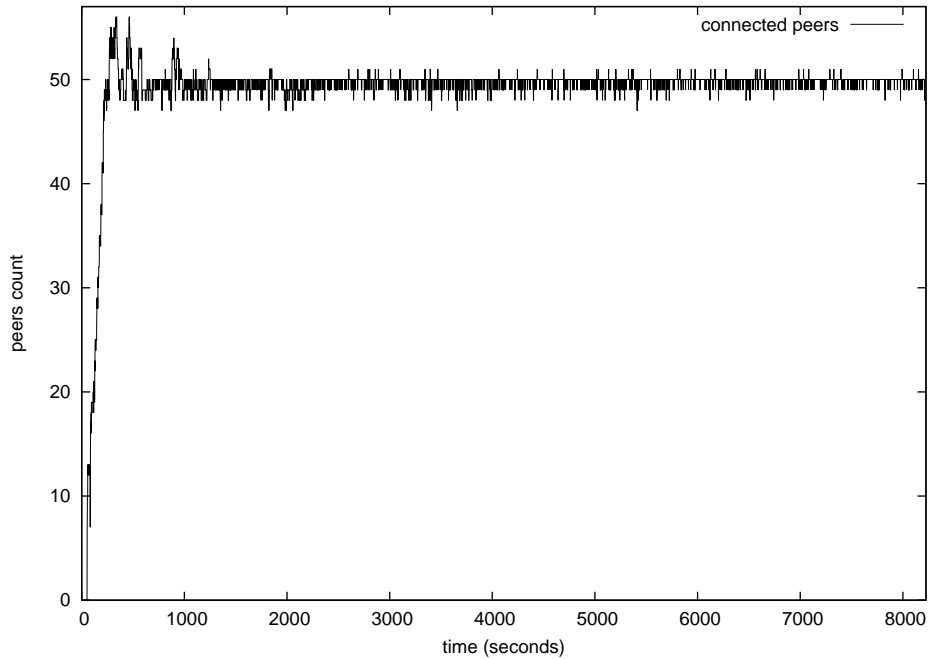Table 5.7: Connection and disconnection rates for torrent 3 - Small Peer Exchange.

Figure 5.8: Number of connected peers in time for Torrent 3 (Small Peer Exchange).

indicate that during the download time the selected peer kept connecting to new peers at a very fast pace.

Because the disconnection rate for leechers is way above the DT_LIMIT value, it is expected that the peer exchange extension will have an impact on the ETA prediction. On the other hand, since the 50% of downloaded data comes from stable sessions with seeds (34.5% from the largest session), the influence of exchanging leechers into the torrent download rate and therefore ETA is expected to be limited.

In Figure 5.9 we see the ETA prediction calculated by the Simple_ETA(30) algorithm compared with the prediction given by the model (without the peer exchange extension). We observe that the Simple ETA forecast is far less stable than the one given by the model. On the other hand the accuracy of these two is similar, especially after $t = 4000s$. It is important to notice in model prediction for this trace that the calculated ETA is on average below the Simple_ETA's. This underestimation is also one of the factors indicating peer exchange that have been discussed in detail in Section 4.3.

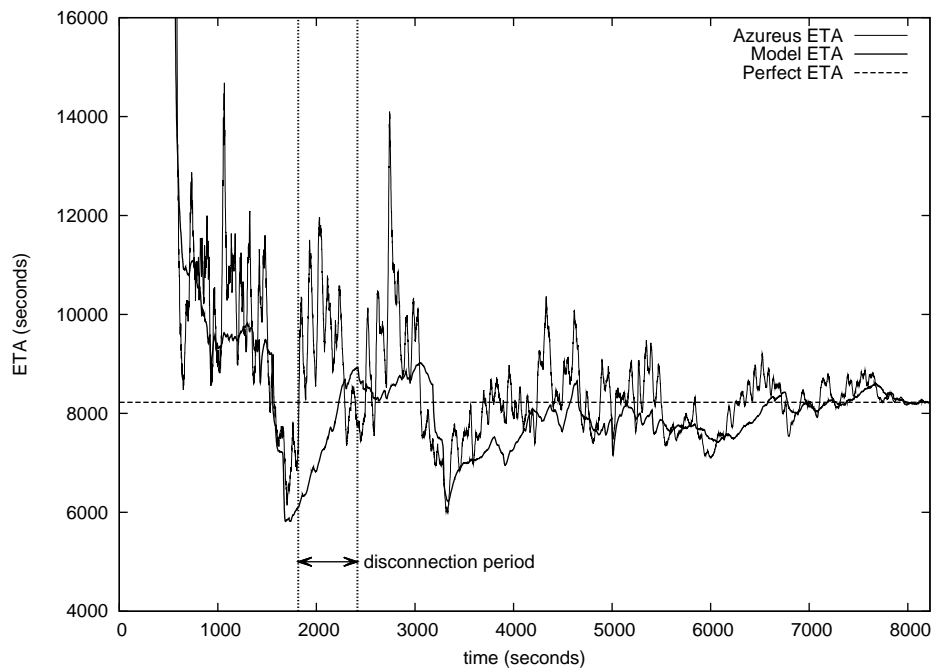To discuss the responsiveness of the model prediction, lets first focus on

Figure 5.9: Comparison of model ETA with the Azureus Simple_ETA(30) for Torrent 3 (Small Peer Exchange).
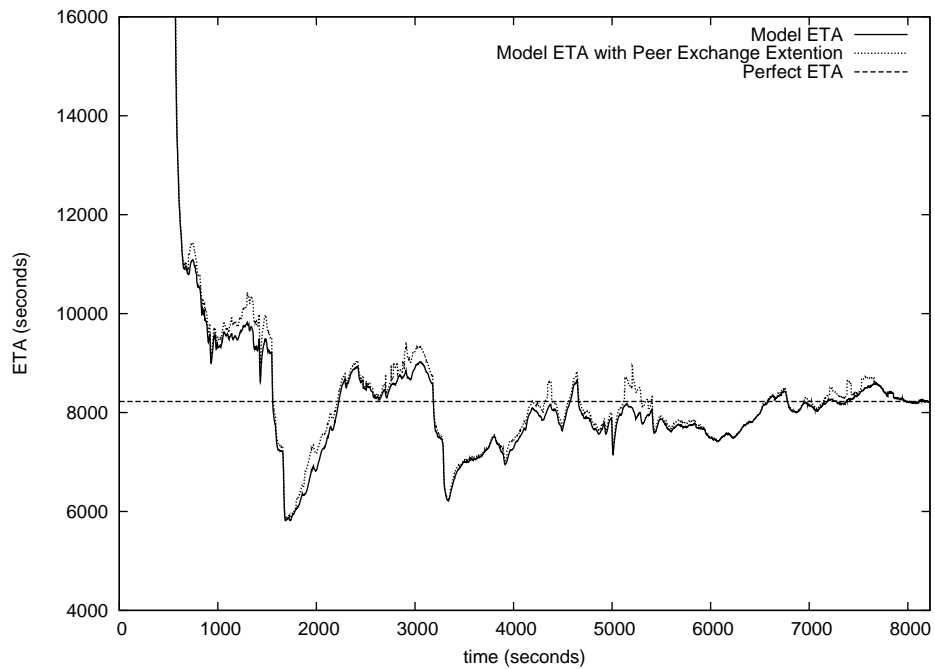

Figure 5.10: Comparison of model ETA with and without the peer exchange extension for Torrent 3 (Small Peer Exchange).
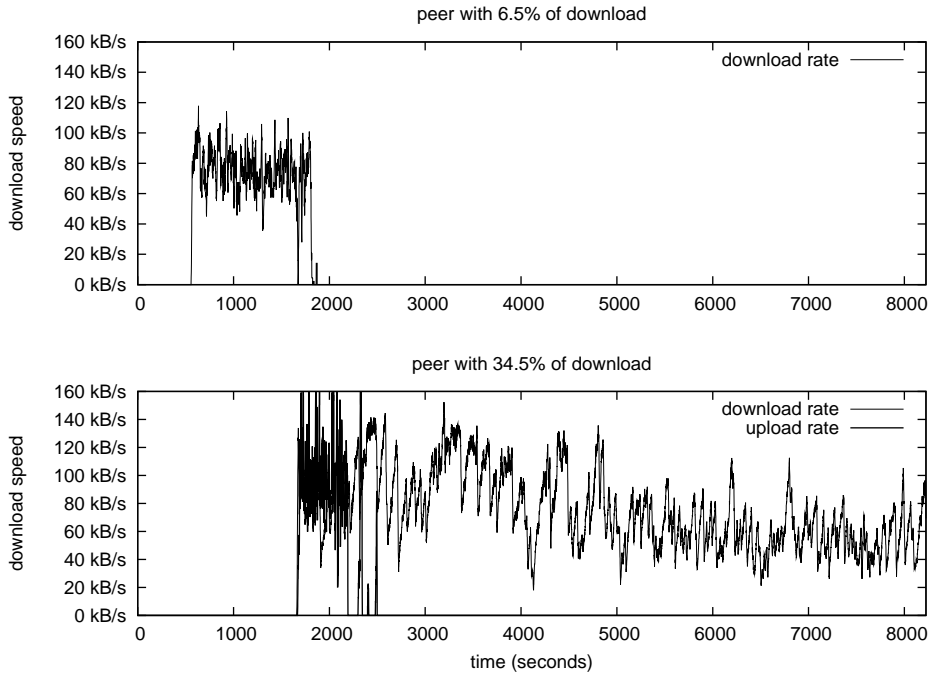
Figure 5.11: Two of the sessions in Torrent 3 (Small Peer Exchange).

two sessions presented in Figure 5.11. We see that during the period $1500s - 2000s$ the data exchange with the first one ended and with the second one started. There was a short period in which the download from both of them overlapped – this can be seen in Simple_ETA prediction in Figure 5.9 as a temporary decrease in predicted ETA around $t = 1700s$ which is immediately corrected after 100 seconds. The reaction of the model on these events was more complex. It immediately reacts to the event of the connection with a new peer – we see that the drop of model ETA value around $t = 1500s$ is as fast as the Simple_ETA's. The reaction on the disconnection was longer. Unlike Simple_ETA algorithm, when the download rate of the second session falls to zero, during the disconnection time (DT) the model had gradually decreases the prediction of the download rate. Because of that and the influence of both of these sessions on the total download rate of the torrent, the model ETA prediction is far below the Simple_ETA value during that disconnection period. The disconnection period has been marked in Figure 5.9.

We see that the model's responsiveness is as good as Simple_ETA's for the connections with new peers, but is much worse in disconnections. This is a trade–off for the model stability and in most cases its negative effect on the responsiveness is reduced, as many sessions restart sending the data in the future. What is more, usually the download conditions change more due to power seeds connecting than disconnecting the selected peer, so the lowered responsiveness for disconnection is not an issue.

Another thing worth focusing on is the influence of the peer exchange extension on the ETA prediction, as this is the first discussed torrent trace in which this extension has an impact on the predicted ETA. As explained in Section 4.3, the peer exchange extension in some cases reduces the disconnection time DT used in model ETA prediction. Since that has a negative

	Connection rate:	Disconnection rate:
Total:	42.0 / min	42.0 / min
Seeds:	0.07 / min	0.04 / min
Leechers:	42.0 / min	42.0 / min

Table 5.8: Connection and disconnection rates for torrent 4 - Large Peer Exchange.

influence on the predicted download rate, the value of ETA calculated with the extension should be higher than the value calculated without it. This can be observed when comparing model predictions for torrent 3 in Figure 5.10.

We see that the extension decreases the underestimation caused by the peer exchange. However, in this example the stability of the prediction is slightly decreased and due to the large amount of data downloaded from seeds the effect on the accuracy is minimal.

To conclude, in this example all presented ETA predictions had similar accuracy. The predictions given by the model are more stable, but their responsiveness on the peer disconnection is lowered. The effect of the peer exchange algorithm is noticeable, but minor.

5.3.4 Torrent 4 – Unstable Peer Exchange

In this example we analyze a very young torrent – at the time the download process starts there is only one (initial) seed and more than 700 leechers. The global evolution of this torrent is rapid. At the end of the download time there is already more than 4600 leechers and eight seeds. 99.84% of the total torrent size have been downloaded from the leechers, there have been more than 1200 started sessions.

Just like the previous torrent, this is an example of a very fast peer exchange. The connection and disconnection rates are shown in Table 5.8. We see that the leecher disconnection rate is very high, much higher than the limit for the peer exchange ($DT\_LIMIT = 5.0/min$). What is more, the number of connections in time shown in Figure 5.12 is constantly equal to the Azureus limit.

The comparison of the Simple_ETA with the model prediction and the comparison of model predictions with and without the peer exchange extension are shown respectively in Figures 5.13 and 5.14. We see that during the download time there were many changes in the download conditions, which resulted in very unstable predictions. The Simple_ETA algorithm as in previous examples were most vulnerable to download speed fluctuations. Slightly more stable is the ETA forecast of the model with the extension. The most stable prediction was given by the model without the peer exchange extension.

Based on the Figures 5.13 and 5.14 we observe that the responsiveness of the Simple_ETA and model with peer exchange algorithm are very similar. Unlike these two, we observe that the prediction given by the plain model is far less responsive. This is clearly visible at $t \sim 6000s$ when the sudden download rate drop occurs and again at $t = 11000s$ during the temporary increase in download speed.

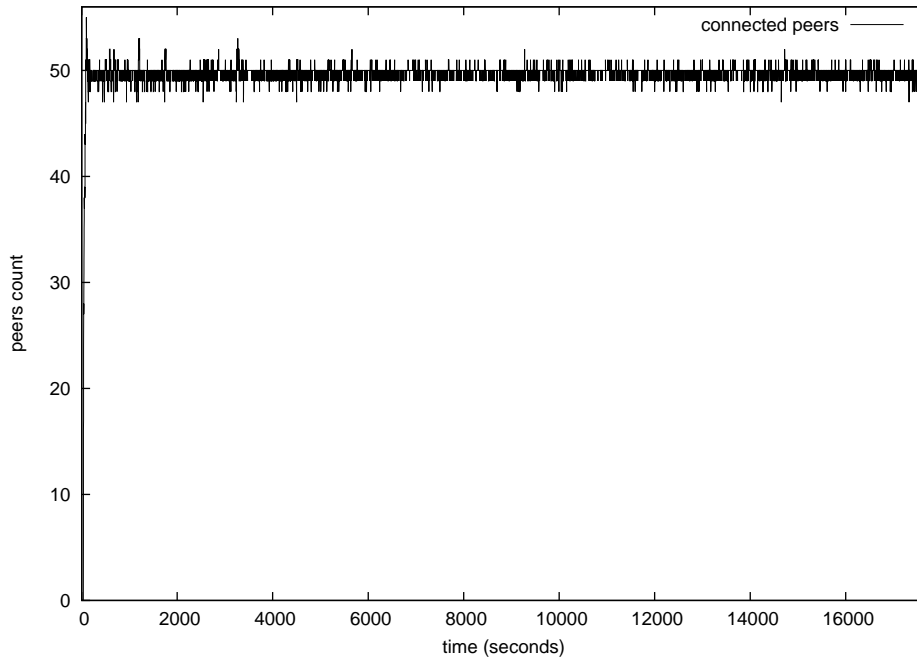As in the previous example the ETA values predicted by the model without

Figure 5.12: Number of connected peers in time for Torrent 4 (Unstable Peer Exchange).

the extension are far smaller than these given by the Simple ETA algorithm. Again, we see that the usage of the peer exchange algorithm decreases the underestimation, but in this example it has a very negative effect on the stability of the prediction.

It is difficult to calculate the accuracy of the presented predictions – this is mostly because of the instability of the download process. The model prediction without the extension gives by far the most accurate forecast of the ETA when compared with the final download time, while the other two algorithms are equally accurate. However, it is important to realize that in the last 15 minutes of the download there have been a significant increase in the download rate, that was a result of a random connection. If by any chance this would not happen, the prediction given by the plain model would give much underestimated ETA value. Prediction of the model with the peer exchange extension would be then as accurate as Simple ETA with the advantage of higher stability.

In conclusion we see that in the example of the torrent 4 the most accurate and stable prediction was given by the model without the peer exchange extension. On the other hand it is difficult to generalize that result, as it strongly depends on a random change of the download speed in the last minutes of the download process. The model ETA with peer exchange algorithm gave a prediction that was as responsive and accurate as Simple ETA, but more stable.
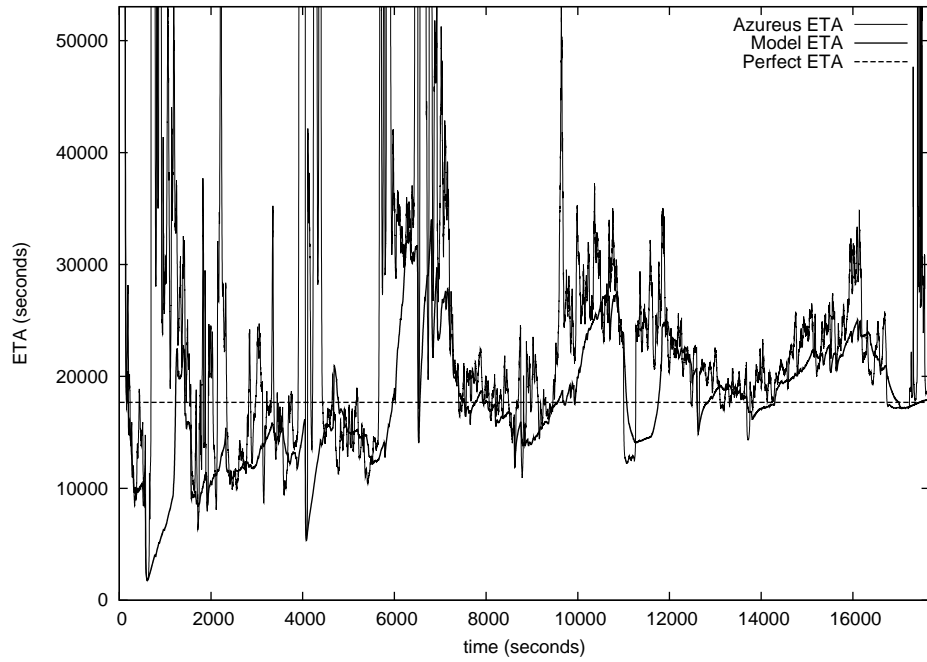
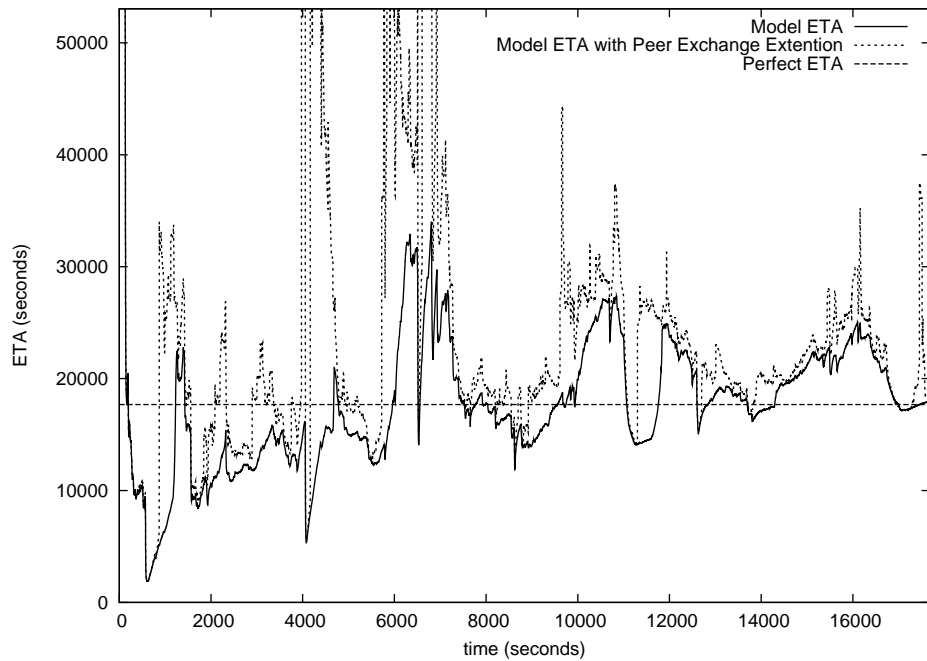Figure 5.13: Comparison of model ETA with the Azureus Simple_ETA(30) for Torrent 4 (Unstable Peer Exchange).

Figure 5.14: Comparison of model ETA with and without peer exchange extension for Torrent 4 (Unstable Peer Exchange).

	Connection rate:	Disconnection rate:
Total:	8.84 / min	8.82 / min
Seeds:	0.55 / min	0.54 / min
Leechers:	8.29 / min	8.28 / min

Table 5.9: Connection and disconnection rates for torrent 5 - Very Stable Peer Exchange.

5.3.5   Torrent 5 – Very Stable Peer Exchange

The last torrent trace example that is analyzed is one of the rarest. Here we encounter a torrent that can be classified as a fast peer exchanging and due to a coincidence of different factors can clearly benefit on a peer exchange extension to the model.

This trace is one of the longest – more than 80000 seconds of download. It is a new torrent, at the beginning there are only two seeds in the swarm and 23 leechers. During the download both of these values grow to 15 and 48 respectively.

The main factor influencing the effect of the peer extension is the percentage of the content downloaded from seeds, which is in this trace less than 38%. Thanks to that the under estimation created by the fast leecher exchange can be visible. What is also very important the total download speed in trace is very stable. This results in a stable prediction and therefore easier analysis of the peer exchange effect. Finally the peer exchange rates for this trace are shown in Table 5.9. We see that the average disconnection rate for leechers is barely above the disconnection limit DT_LIMIT, but it is high enough to decrease the DT value by means of peer exchange algorithm.

In figure 5.15 and 5.16 we can see the comparison of three ETA predictions: Simple_ETA(30), model without and with peer exchange extension. As in the previous torrent examples we see that both of the model predictions are more stable than Simple ETA algorithm. In this trace we see that the download speed of the torrent during the download time had almost constant random fluctuations. This resulted in very high stability of both model predictions. Because of almost no changes in download conditions it is hard to determine and compare the responsiveness of these three predictions, but this resulted in a high accuracy of the model predictions.

Another effect of no conditions change is the impact of peer exchange extension on the model prediction. In Figure 5.16 we see comparison of both model predictions – after $t = 50000s$ we can clearly see that the simple model prediction under estimates the download time. The effect of peer exchange algorithm is clearly visible and reducing the under estimation (Figure 5.17). This can be compared with torrent 4 example (Unstable Peer Exchange) where the leecher disconnection speed was much higher than DT_LIMIT and the effect of the peer exchange was larger, but due to high instability of the download speed there were no clear increase in accuracy of the prediction.

In conclusion the 'Very Stable Peer Exchange' torrent trace is a relatively rare example of a trace. Due to coincidence of very high download rate stability, fast leecher exchange rate and large percent of content downloaded from leechers we can observe the under estimation of the model ETA and the in-
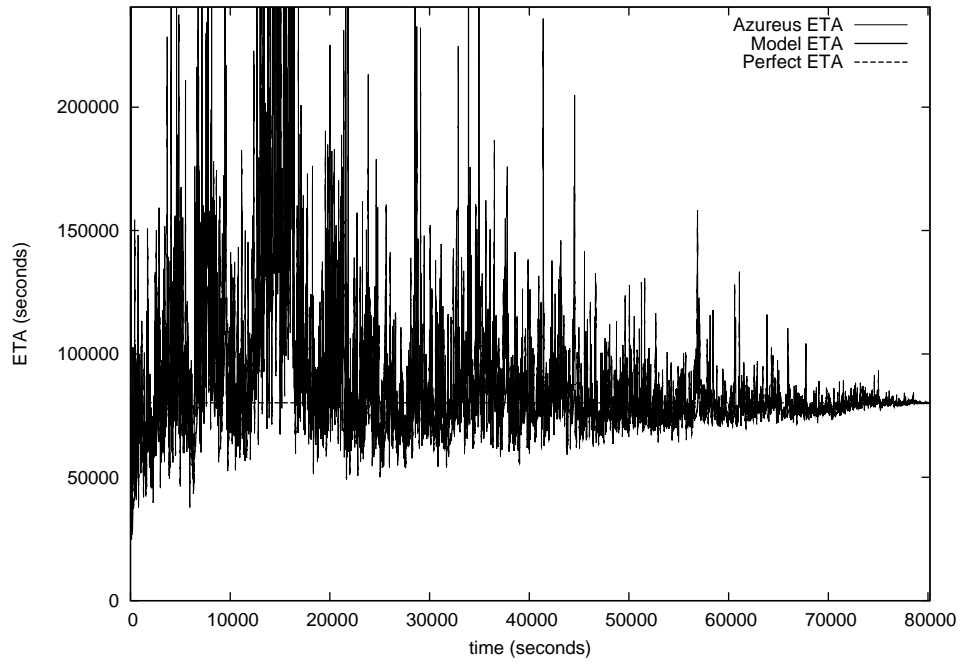
Figure 5.15: Comparison of model ETA without peer exchange extension with the Azureus Simple_ETA(30) for Torrent 5 (Very Stable Peer Exchange).
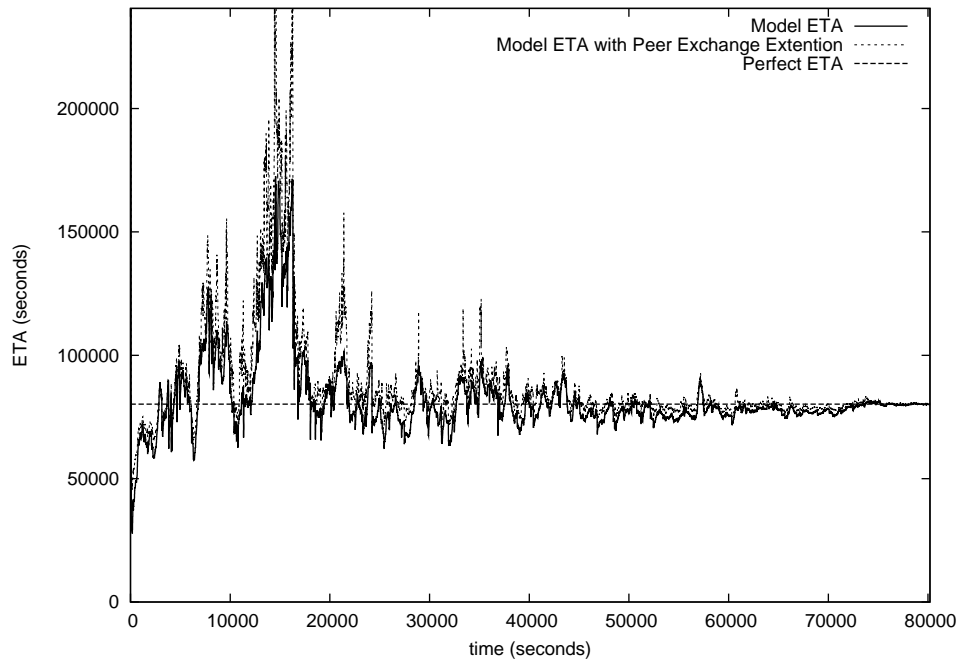

Figure 5.16: Comparison of model ETA with and without peer exchange extension for Torrent 5 (Very Stable Peer Exchange).
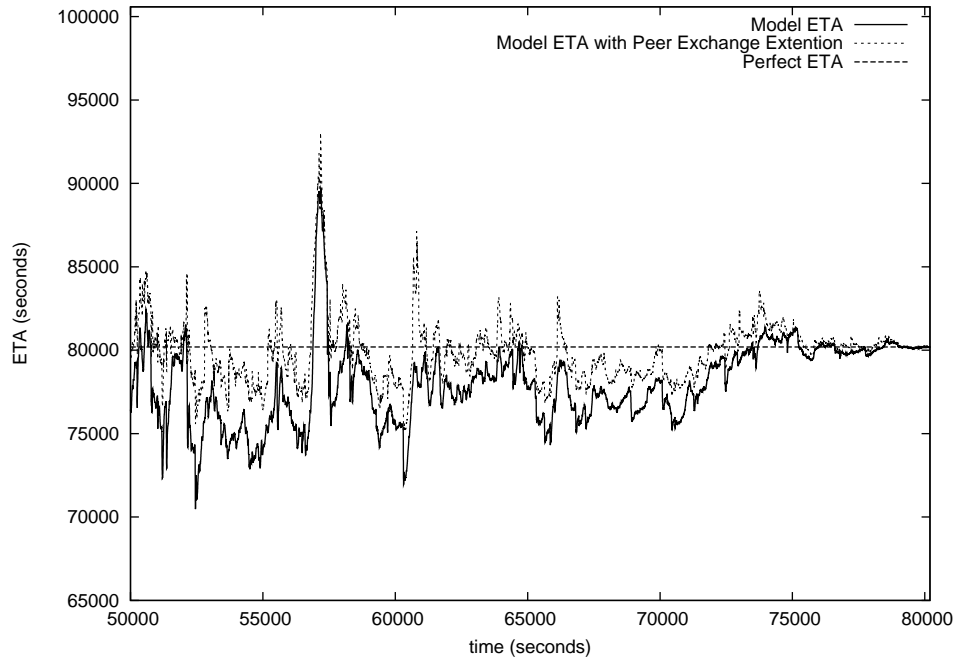
Figure 5.17: Magnification of Figure 5.16. We see the positive effect of the peer exchange extension on the accuracy of the prediction.

crease in accuracy as an influence of the peer exchange extension on the model prediction. Both of the model predictions are much more accurate and stable than $Simple\_ETA$ algorithm.

5.4   Analysis

Evaluation presented above shows that the prediction given by the model has many qualities that are better or equally good as the ones presented by the Simple ETA algorithm. In section 3.3 we have shown that in the Simple ETA algorithm there is a trade–off between the stability and responsiveness of the prediction. The main advantage of the model's algorithm is a clearly higher ETA forecast stability with only slight loss of the responsiveness, therefore improving the relation between these two. An accuracy of the model's prediction on average shows an improvement over Simple ETA, but this varies in different torrent trace examples.

Thanks to the model's separate session analysis the model can distinguish between the importance of different changes in download conditions and react accordingly. As shown in torrent trace example 1 (One Power Seed) the model strongly reacts on connection and disconnection with remote peer, while reactions on download rate fluctuations are being minimized. It is worth mentioning that responsiveness on the connection is similar to the Simple ETA's, while response to the disconnection is slower.

The analysis shows that the peer exchange extension gives fine results when the connection conditions are stable, the disconnection rate is high and large percent of data is downloaded from leechers. In these cases it has a positive effect on accuracy by reducing the ETA underestimation. In other cases it

has a negative impact on the stability of the prediction and does not improve accuracy.

We have seen that in most of the torrent traces a connection to the power seed has a high impact on the download conditions. This is the most important problem in the ETA forecast. Since the number of connections of that kind is very small, there are no reliable means of predicting them. Therefore the main goal of the ETA forecast is to determine their occurrence and react appropriately. Thanks to the model high responsiveness to connections this type of change has an immediate impact on the ETA prediction, giving an advantage over the Simple ETA algorithm.

When comparing the Simple ETA and the model algorithm we observe that in traces with long download time the calculation of the accuracy of their prediction is often complicated. Both the Simple ETA and model prediction show vulnerability on long–term changes in download conditions. The cause of that effect can be due to the global torrent swarm evolution or peer activity patterns. Those influences could be a subject of an analysis and improve the ETA forecast, but their prediction have not been included in model algorithm.

Another observation is that the quality of both ETA predictions depends on the swarm size and leecher–to–seed ratio. In torrents with small swarm size we see that random changes in set of connected peers have a higher impact on the download process. This problem complicates the ETA prediction and results in decreased stability. Torrents with high seed–to–leecher ratio increased connection rate stability, but present the far greater chance of connection with power seed which drastically changes the download conditions. An advantage of the model prediction over Simple ETA is the responsiveness on the power seed connections which make it work better in traces with high leecher–to–seed ratio. Finally we can mention that we see both of the predictions to give very bad results during the beginning of the download process.

It is important to say that even though the measurements and analysis shown above have been done in the clean–trace environment, the obtained results are also applicable to non-clean torrent traces. Most of the non–clean influences are well tolerated by the model. The only exception is the setting of the user limit on the download rate. This influence changes the rules of peer disconnection and have a very negative effect on the stability of the model ETA prediction.

During the analysis we have observed that torrents downloaded on one machine in the similar download condition tend to present similar behavior (download rate stability, peer exchange rates). On the other hand torrents traces collected on different machines are usually downloaded in very different conditions, therefore we have observed large differences in torrent behavior among different machines.

Chapter 6

Conclusions

In this thesis we have discussed the problem of ETA prediction in BitTorrent. This topic directly have been addressed before in BitTorrent clients with Simple ETA algorithm. It bases its prediction only on the current total download speed and its results are consider unstable. Different authors discussed the topic of the torrent evolution, but focused in their research on global torrent properties or unrealistic model assumptions, therefore making their results inapplicable in ETA predicting. We have created a new torrent download model that gives improvement over Simple ETA and benefits from information about BitTorrent specification.

The created model is based on the analysis of the BitTorrent specification, data exchange characteristics between peers and real–life influences on the download process. Its main improvement over the global torrent evolution models is that it bases the prediction algorithm only on the local–view data available to the BitTorrent client, therefore making it implementable in real–life situations.

The model's ETA prediction concept is based on splitting the torrent download process into separate sessions with each of the connected peers. Every session is analyzed and its future download rate prediction is given. The total ETA forecast is based on cumulative download rate prediction from all started sessions. Session analysis is based on a state model. The benefit of this approach is the selective responsiveness of the model on the different types of events occurring during torrent download. This results both in improved prediction stability and fast adjustment to change in download conditions.

During the analysis of the Simple ETA algorithm we have determined that the desired qualities of the ETA prediction are accuracy, stability and responsiveness. Based on that we have evaluated the performance of the created model on real–life traces and compared it with the Simple ETA algorithm. The analysis have shown that the model prediction is much more stable and its responsiveness is only slightly lowered as compared to Simple ETA. On average model forecast is more accurate, but this varies in different torrent traces and download conditions.

The evaluation has shown that a number of influences has an impact on the torrent ETA prediction. The most important problem is the connection to a power seed, which is extremely difficult to predict and causes huge change in download conditions. Percentage of download from seeds and leecher–to–seed ratio are factors that strongly influence the stability of the torrent download

rate.

In some traces we have seen an impact of long–term influences to the torrent download conditions, that complicates the ETA prediction. The most important of these are global torrent evolution and peer activity patterns. They are not directly addressed by the model and their impact can be seen mostly in newly created torrents or when the download time is long. The forecast of these influences is vital in ETA prediction and should be a subject of future works.

Bibliography

[1] B. Cohen. *Incentives build robustness in BiTtorrent*, In proceedings of Workshop on Economics of Peer–to–Peer Systems, 2003.

[2] BitTorrent specification: *http://wiki.theory.org/BitTorrentSpecification*

[3] BitTorrent client list: *http://en.wikipedia.org/wiki/BitTorrent_client*

[4] D. Qiu and R. Srikant. *Modeling and performance analysis of bittorrent-like peer-to-peer networks* In proceedings of the ACM SIGCOMM, 2004.

[5] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding and X. Zhang. *Measurements, analysis, and modeling of bitTorrent-like systems* In proceedings of the Internet Measurement Conference, 2005.

[6] B. Fan, D. Chiu and J. Lui. *Stochastic Differential Equation Approach to Model BitTorrent-like P2P Systems* In proceedings of IEEE International Conference on Communications, 2006.

[7] V. Rai, S. Sivasubramanian, S. Bhulai, P. Garbacki and M. van Steen. *A Multi Phased Approach for Modeling and Analysis of the Bittorrent Protocol* In proceedings of 27th international conference on distributed computing systems, 2007.

[8] A. Legout, G. Urvoy-Keller, P. Michiardi. *Rarest first and choke algorithms are enough.* In proceedings of the ACM SIGCOMM, 2006.

[9] 'Fast Peers Extensions' specification: *http://bittorrent.org/beps/bep_0006.html*

[10] 'BitTorrent Location-aware Protocol 1.0' extension specification: *http://wiki.theory.org/BitTorrent_Location-aware_Protocol_1.0_Specification*

[11] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy and A. Venkataramani. *Do incentives build robustness in BitTorrent?* In proceedings of the USENIX Symposium on Networked Systems Design and Implementation, 2007.

[12] R. Burden and J. D. Faires *Numerical Analysis (7th ed.)*, Brooks/Cole, 2000

[13] Azureus (now Vuze) homepage: *http://azureus.sourceforge.net/*

Appendix A

Randomized simple ETA

A.1 Introduction

This is a mathematical sketch of a proof that Simple ETA algorithm's prediction is biased. By taking few assumptions about the download process that Simple ETA prediction over–estimates the download time value.

At first we define the model and terminology that will be used, based on these we prove that the Simple ETA algorithm's expected result is biased and higher than the actual time of torrent download. We conclude with an example of artificial data showing the over–estimation.

A.2 Model

The Simple ETA algorithm gives accurate result assuming that the total torrent download speed is constant in time. Since that assumption is not met in real–life situations we want to create download model that will be more realistic and include the random download speed fluctuations. In general we could consider a model in which the download speed for every time $t \in \mathbb{R}$ is a random value with a given distribution, but then $\mathbb{E}s(t)$ could not be calculated as as $s(t)$ is not a measurable function. To preserve the concept of a random value of $s(t)$ at every measurement and satisfy the needed mathematical correctness we propose the following model:

- V be the size of a torrent file.

- There is a time period value given $0 < \Delta t \ll 1s$

- Time is divided into Δt periods: $[0, \Delta t), [\Delta t, 2\Delta t), \ldots$

- Speed of download is constant for every period.

- For every period speed of download s is a random value with a distribution $S : t \rightarrow (0, s_{MAX})$, where s_{MAX} is a maximal theoretically possible transfer value. This definition enforces that $s(t) > 0$ and also that $\bar{s} = \mathbb{E}s(t) < \infty$ and $\rho^2 = \mathbb{D}^2 s(t) < \infty$

- Values $s(t_1)$ and $s(t_2)$, if t_1 and t_2 belong to different periods, are independent.

Time of download is the soonest time when the downloaded content will be greater or equal V. This can be defined as:

$$T = \min t : \sum_{i=0}^{i\Delta t < t} s(i\Delta t)\Delta t \geq V$$

Let us assume that $\bar{s} \ll \frac{V}{\Delta t}$, so it will take many Δt periods to download the file. Then, based on CLT theorem we get that the data downloaded in N first Δt periods has a Gaussian distribution:

$$\sum_{i=0}^{N-1} s(i\Delta t)\Delta t \sim \mathcal{N}(N\bar{s}\Delta t, N\rho^2\Delta t^2)$$

For any given value of $\bar{T} = N\Delta t$ we calculate the probability that the download time T will be less than $\bar{T}$:

$$P(T < \bar{T}) = P(T < N\Delta t) = P(\sum_{i=0}^{N-1} s(i\Delta t)\Delta t > V) \simeq$$

$$\simeq 1 - \Phi(\frac{V - N\bar{s}\Delta t}{\rho\Delta t\sqrt{N}}) = \Phi(\frac{\bar{T} - \frac{V}{\bar{s}}}{\frac{\rho\sqrt{\Delta t}}{\bar{s}}\sqrt{\bar{T}}})$$

The function Φ is the cumulative Gaussian distribution. Based on the equation above we get that for every given value $\delta > 0$, if the Δt is small enough, we have:

$$P(\frac{V}{\bar{s}} - \delta < T < \frac{V}{\bar{s}} + \delta) \sim 1$$

From now on we assume that Δt is small enough that this formula is true for $\delta \ll 0, 1s$. Since for the purpose of this thesis we measure the download time with one second granularity, the following formula is valid:

$$T = \frac{V}{\bar{s}}$$

We see that in the presented download rate model we have all requested qualities. At every measurement point we have the value of $s(t)$ to be random. What is more we have a fixed download time T that is based only on download rate distribution and is constant (not random). During the ETA prediction we do not have the knowledge of the value of T nor download speed distribution $s(t)$.

This model is useful for testing ETA prediction algorithms that are based only on current torrent download speed. An example of that kind of an algorithm is the Simple ETA prediction algorithm.

A.3   Analysis

In this section we show how the more realistic download rate model influences the results of the Simple ETA prediction. As stated previously in the thesis we use the following Simple ETA estimator:

$$ETA(t) = t + \frac{sizeToDownload(t)}{s(t)}$$

70

Based on model definition we have that T is not random from our point of view. Therefore $sizeToDownload(t)$ is also not random and is equal to:

$$sizeToDownload(t) = \frac{fileSize * (T - t)}{T} = \frac{(T - t) * T * \bar{s}}{T} = (T - t)\bar{s}$$

We show that $Simple\_ETA$ estimator is biased[1] and overestimates the download time value. This is equivalent to $\mathbb{E}(Simple\_ETA(t)) - T > 0$. The proof of that relation uses a lemma based on Holder's inequality. It have been proven[2] that:

$$\mathbb{E}|XY| \leq (\mathbb{E}|X|^p)^{1/p}(\mathbb{E}|X|^q)^{1/q}$$

where X and Y are random variables and $\frac{1}{p} + \frac{1}{q} = 1$. Since $s(t) > 0$, we can take $X = \sqrt{s(t)}$, $Y = \frac{1}{\sqrt{s(t)}}$, $p = 2$ and $q = 2$. We get:

$$1 = \mathbb{E}1 \leq \mathbb{E}(\sqrt{s(t)}\frac{1}{\sqrt{s(t)}}) = (\mathbb{E}\sqrt{s(t)}^2)^{1/2}(\mathbb{E}\frac{1}{\sqrt{s(t)}^2})^{1/2} = (\mathbb{E}s(t))^{1/2}(\mathbb{E}\frac{1}{s(t)})^{1/2}$$

this results in:

$$\mathbb{E}\frac{1}{s(t)} \geq \frac{1}{\mathbb{E}s(t)}$$

Based on that we prove the over-estimation of the Simple ETA algorithm:

$$
\begin{aligned}
\mathbb{E}(Simple\_ETA(t)) &= t + sizeToDownload(t)\mathbb{E}\frac{1}{s(t)} \geq \\
&\geq sizeToDownload(t)\frac{1}{\mathbb{E}s(t)} = \\
&= t + (T - t)\mathbb{E}s(t)\frac{1}{\mathbb{E}s(t)} = T
\end{aligned}
$$

We see that the expected value of Simple ETA prediction is higher than the actual download time and therefore Simple ETA as an mathematical estimator of T is biased. It also can be shown that the difference between the expected value and T is growing with $\rho^2 = \mathbb{D}^2 s(t)$.

Another interesting fact is that if according to the distribution of the download speed there is only one possible value of $s(t)$ (with the probability of one) the inequality above will change into equality (as $\mathbb{D}s(t) = 0$). In that situation the presented model can be reduced to the constant download speed model. As a result we have that in that when the download speed is constant the Simple ETA algorithm is accurate, what is consistent with the analysis presented before.

A.4   Example

We present an example of Simple ETA prediction for artificial data. We simulate the download rate of a torrent with content size equal to $5250kB$. The used download speed distribution is uniform on a interval $(0.5kB, 10kB)$. The

---

[1]See: http://en.wikipedia.org/wiki/Estimator#Point_estimators (point 5)

[2]See: http://en.wikipedia.org/wiki/H%C3%B6lder's_inequality#Notable_special_cases (point 4)
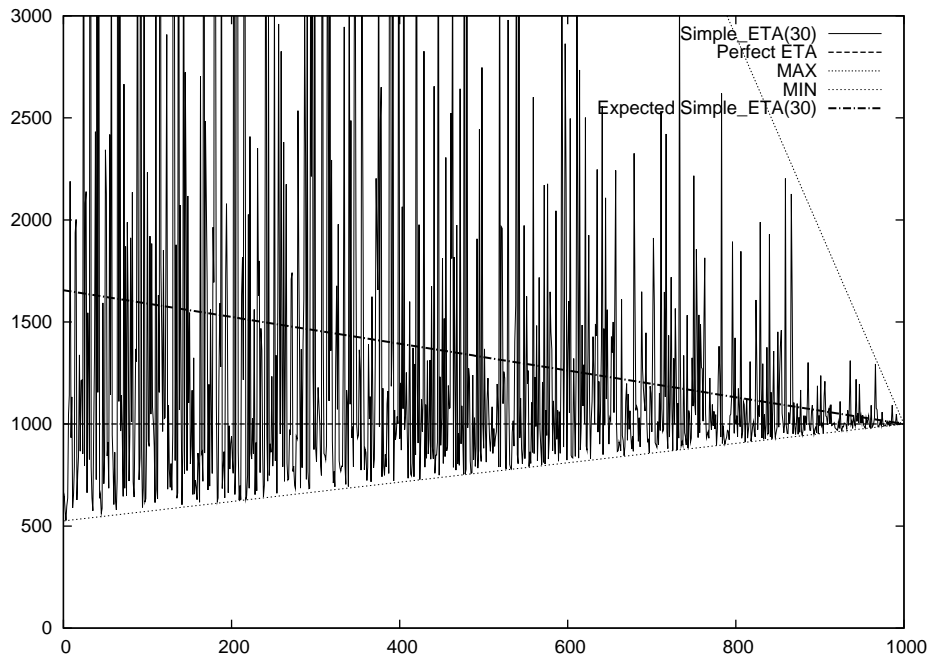
Figure A.1: An example of Simple ETA prediction for artificial torrent download data.

average download speed for that distribution is $5.25kB/s$ and therefore we get that the download time T according to the model is equal to $T = 1000s$.

In Figure A.1 we see the $Simple\_ETA(30, t)$ prediction calculated for the generated data. This can be compared both to the actual download time and to the expected value of $Simple\_ETA(30, t)$. We see that the expected value is higher than the download time T. In time $t = 0$ we have:

$$\mathbb{E}(Simple\_ETA(t = 0)) = 1656s$$

what is more than 60% of over-estimation. Lines labeled as 'MIN' and 'MAX' give boundaries to Simple ETA estimation (these boundaries are an effect of distribution's interval boundaries). We see that the average value of Simple ETA is always above the download time T.