Efficiency of BitTorrent-like Peer-to-Peer Live Streaming Systems

Ali Issaei

A Thesis

In

The Department

Of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

January 2010

Your file Votre référence
ISBN: 978-0-494-67281-5
Our file Notre référence
ISBN: 978-0-494-67281-5

ABSTRACT

Efficiency of BitTorrent-like Peer-to-Peer Live Streaming Systems

Ali Issaei

From the past few years, peer-to-peer (P2P) applications have becoming well-liked magnificently. BitTorrent (BT) has one of the most effective mechanisms for P2P content distribution among all P2P applications. In P2P applications, each node plays a role both as a client and a server simultaneously, comparing to the traditional client-server systems, where every node has only one responsibility to act as either a client or a server. Therefore, in P2P applications, the upload bandwidth of each peer can be counted as a significant resource of the network. Although BT was created for file-sharing purposes at the beginning, which is a time-insensitive distribution, after a while, it has attracted the attentions to be use for video/audio streaming purposes too, which are time-sensitive. The importance of this capability is that peers now, can be able to watch, or listen to, their favorite live streaming content concurrently. Motivated by this fact, a stochastic model for a BT-based P2P live streaming system is proposed and numerically solved and based on what we gain, it is also shown how the performance of the system can be affected by different parameters of the system (such as the number of neighbors, delay time, size of the buffer, etc.). Moreover, we also try to apply some minor changes needed in the BitTorrent's mechanisms, in order to support the video/audio streaming more efficiently.

To my parents

For their love and support

ACKNOWLEDGEMENTS

This thesis would not have been possible without the supervision of Dr. Dongyu Qiu, whose patience, encouragements, guidance, understanding and supports, from the beginning to the final level of my master study at Concordia University, taught me to believe myself, have a better understanding of my surrounds and develop an understanding of the subject.

I would also like to show my gratefulness to the faculty and staff member of Concordia University, who provided me assistance during my studies there.

Lastly, but not least, I offer my regards and blessings to Ms. Nadia Secreto for her truthful support and valuable remarks throughout this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
BT	BitTorrent
DHT	Distributed Hash Table
FCFS	First-Come-First-Served
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
P2P	Peer-to-Peer
SMTP	Simple Mail Transfer Protocol
Snap	Structured overlay Networks Application Platform
UDP	User Datagram Protocol
URL	Uniform Resource Locator

Chapter 1

INTRODUCTION

1.1 LITERATURE REVIEW

The development of the Internet in the duration of the last decades was absolutely fast. According to statistics that have been done by the Internet World Stats on September 30, 2009, estimated number of Internet users over the world is 1,733,993,741 people as it is shown in Figure 1.1 and 1.2 [16]. The Internet is one of the most essential and useful tools used in today's society which can be a significant impact in people's life style. Sometimes it can be used as an informative resource, which John Allen Paulos [1] said that: "The Internet is the world's largest library. It's just that all the books are on the floor", while some other times it will be used as an entertaining or a communicational tool.

Peer-to-Peer (P2P) applications are one of the most interesting applications which have been proposed a few years ago and have been attracting more and more attentions every day. A peer-to-peer (P2P) network includes some participants who share a part of their resources (such as network bandwidth, disk storage, etc) with the other participants in that network and act simultaneously both as clients and servers in the network and as a

result of this characteristic they would not need to use any central coordination instances (stable hosts or servers) to interact with each other [4].

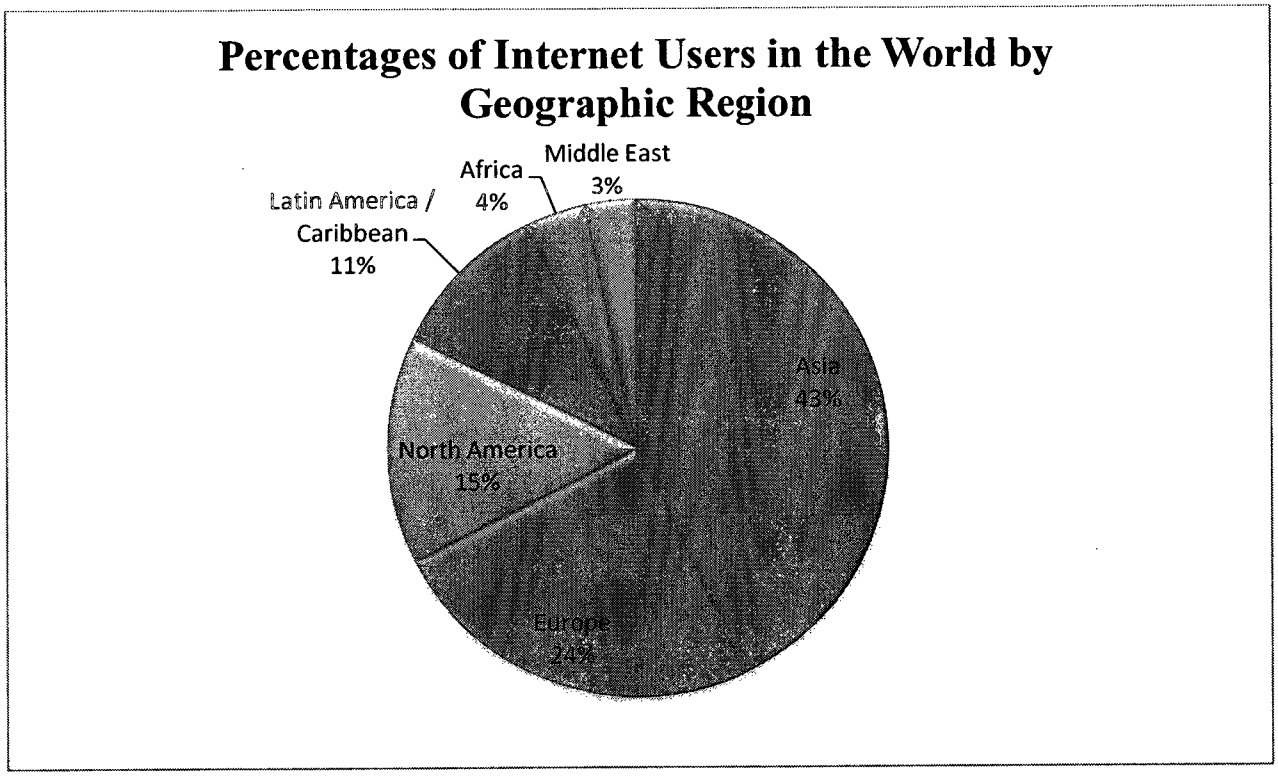


Figure 1.1: Percentages of Internet Users in the World by Geographic Region

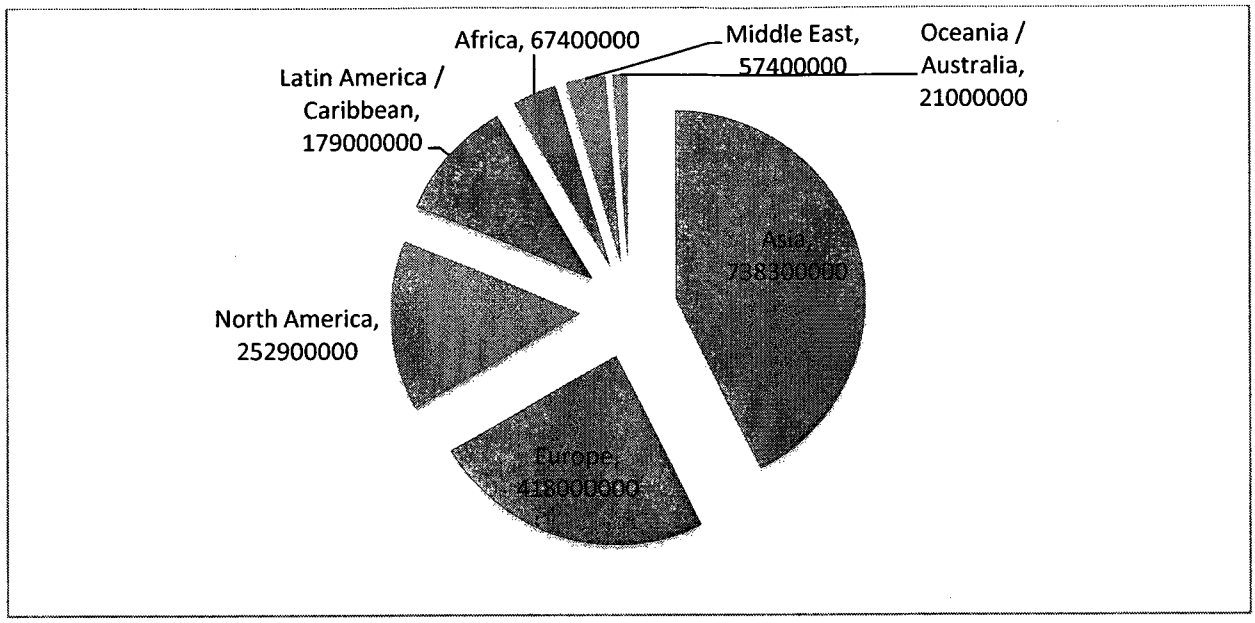


Figure 1.2: Number of Internet Users in the World by Geographic Region

1.2 CLIENT-SERVER MODEL

Definitions:

A Client-Server network will be defined as distributed network that consists of one higher performance system which is generally called the server, and several lower performance systems which are called the clients.

A server is a central registering unit and it is the only provider (source) of a specific content and service. A network server is a computer (a group of computers) which is (are) designed to process the requests of the clients in that network and deliver data to them over the Internet or a local network[4] & [5].

On the other hand, a client would only be able to request the content(s) or execute the service(s) which has (have) been provided by the server without having the ability of sharing any portion of its own contents (resources) with other clients in that network [4].

The Client-Server architecture is a network-architecture in which each any process or any computer on that network could play a role either as a client or as a server.

A Server normally is a process dedicated to printers (print servers), manage disk drives (file servers), network traffic (network servers) or a powerful computer.

Clients usually are personal computers or workstations on which users run their applications. As it seems, the clients depend on servers for resources such as devices, files and processing power [6].

Client-Server model in a network provides an easy way to link the programs, which are effectively distributed across different locations. A common Client-Server model could be the computer transactions. For example, when we want to check our bank account online over the Internet with our computers, a client program in our computer will forward our request to a server program in our bank. This program might forward the request to its own client program that sends a request to a database server at another bank computer (probably to the central bank computer) to retrieve our account balance. The balance will be returned back to the bank data client, which in turn serves it back to the client in our personal computer. Hence, it provides the information that we are looking for [7].

Architecture:

One of the most popular models in constructing distributed systems is the Client-Server model. In this design, clients demand for contents or required services from a server. The

client and server require a known set of customs before they could be able to have interaction with each other. This set of customs contains a protocol, which should be carried out at both ends of a connection. As some examples, we can mention the TELNET protocol that has been used in the Internet for remote terminal imitation, the File Transfer Protocol (FTP) and Hypertext Transfer Protocol (HTTP).

Server:

A server as a service provider must determine responses to the requests and then by using an appropriate protocol, it should return the results. A server also can be run either on the same device that the client is running on, or on another obtainable device in that network.

Performance:

In some specific conditions, the clients could be deficient devices. In this case the calculation would be done on a high-performance server. The usability of this approach is really rare nowadays; but in some application, such as virtual reality computations for film scenes, they still use this method.

Central data management:

The Client-Server models are very common and some of services provided by a server are as follows:

File server:

One server provides multiple clients with a file system. Access control and transaction control are some of the tasks that these kinds of servers will do. It is also important to mention that only one client may have access to the file with write permission at any time.

Web server:

A Web server provides various clients with information. And the information could be either static on a Web server or dynamic, which would be generated by different service applications.

Client:

A client generally is a device or a process that uses the service(s) of one or more servers. Clients normally act like interfaces among servers and the users (people). They have been designed for some purposes like the information input and visualization of information. In the past, clients had only a few numbers of resources with a really low functionality. However, today we see that most of the clients are personal computers with much higher performances required to the resources and functionalities. Earlier clients displayed only the running application on the server and forwarded the input(s) of the user(s) to the server. [9]

A thin client (sometimes also called a lean or slim client) has very limited local resources in both hardware and software dimensions, which means that it relies on other computer (server) to do its traditional computational roles.[8]

Thin clients, share their computations with the same server and it shows that they act as the components of a broader computer infrastructure and so theirs infrastructures can be seen as the depreciation of some computing service across several user-interfaces. [11]

It also applicably needs the processing time. One of the most common usages of the modern thin clients is the low-end microcomputer that merely focuses on providing a graphical user interface (GUI) to the end-user(s). (10)

On the other hand, a thick client, also called a fat client or rich client, is rich in terms of hardware and software. They can store, execute their own independently and provide rich functionality without any need of the central server [12]. They usually refer to personal computer(s) [9].

Client-Server model's Advantages and Disadvantages:

Some of the advantages and disadvantages of the Client-Server architecture are listed below:

Advantages:

- Data management is very easy because of the centralized location for the files and data. This property causes the fast backups and low errors; it consists of different permission levels that will prevent users (clients) from making the files and data damaged. All the processes take place on the server and only the final outcomes would be returned to the users (clients). This procedure by lessening the volume of the traffic in the network among the server and the clients will increase the total performance in the network.
- Since all the data, file and applications are on the server, if there is any defective client in the network, thin clients can replace it very efficiently.

Therefore, we can categorize the client-server advantages and disadvantages as the following [13]:

- **Centralized Resources:**
  - It means that all data is stored in one central location.
  - It is easier to backup files and data.
  - Accessibility to files and data are easier.

- **Good Efficiency:**
  - In most of the cases, when the load of the network is light or medium, the traditional client-server models have good efficiencies (Software and hardware are optimized for multiple users).
- **High Security:**
  - In traditional client-server networks, because there is only one central login, they have high securities generally.
- **Good Scalability:**
  - When the load of the traditional client-server networks is light or medium, they generally have good scalability.

Disadvantages:

- Client-Server systems are costly and require a high maintenance.
- In Client-Server systems, the servers are extremely vulnerable and if a problem happens to them, their failures would probably cause heavy delays or sometimes a complete system breakdown. Resulting in blocking several clients from working with their applications or other data.

As a summary we can categorize the client-server disadvantages as the following [13]:

- If the server goes down, it will bring a part or the whole network down.
- When the number of users increases, the client-server's scalability decreases and sometimes it could result in the partial/complete breakdown in the server.
- It is expensive to install and upgrading the Hardware(s) is costly.
- It needs to be maintained and supervised by some professional IT staff.

1.3 P2P MODEL

The P2P (peer-to-peer) paradigm was proposed based on the principle that every single component of the system is responsible for acting both as a client and as a server concurrently, in order to overcome the inadequacy of the traditional client-server model. As a result of this fabulous property in P2P model, each user in the network would share a portion of its resources (e.g. bandwidth) and as a result, the total bandwidth of network would be improved as the number of clients increases. In other words, the total bandwidth of the network is equal to the sum of all the users' bandwidths in that network. Nowadays, several applications have been designed based on the P2P model, such as Instant Messaging (IM), Online games, Collaborative Community, IP Telephony, High Performance Computing, File sharing and Streaming (Live Streaming and Recorded Streaming) [14].

According to an analysis of traffic demographics in North-American broadband networks that has been done by Sandvine Intelligent Broadband Networks in May 2008, it shows that P2P users account for around 43.5% of the Internet traffic, as shown in Figure 1.3 [15].

As mentioned earlier, the client-server model is a traditional model which is used in many of the Internet protocols such as File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), Hypertext Transfer Protocol (HTTP), etc [17].

Having a centralization feature in client-server models, allows them to have some great advantages (such as service assurance, satisfactory security, easy server maintenance and

ease of updating the resources). On the other hand, this characteristic also brings them some disadvantages as we discussed earlier.

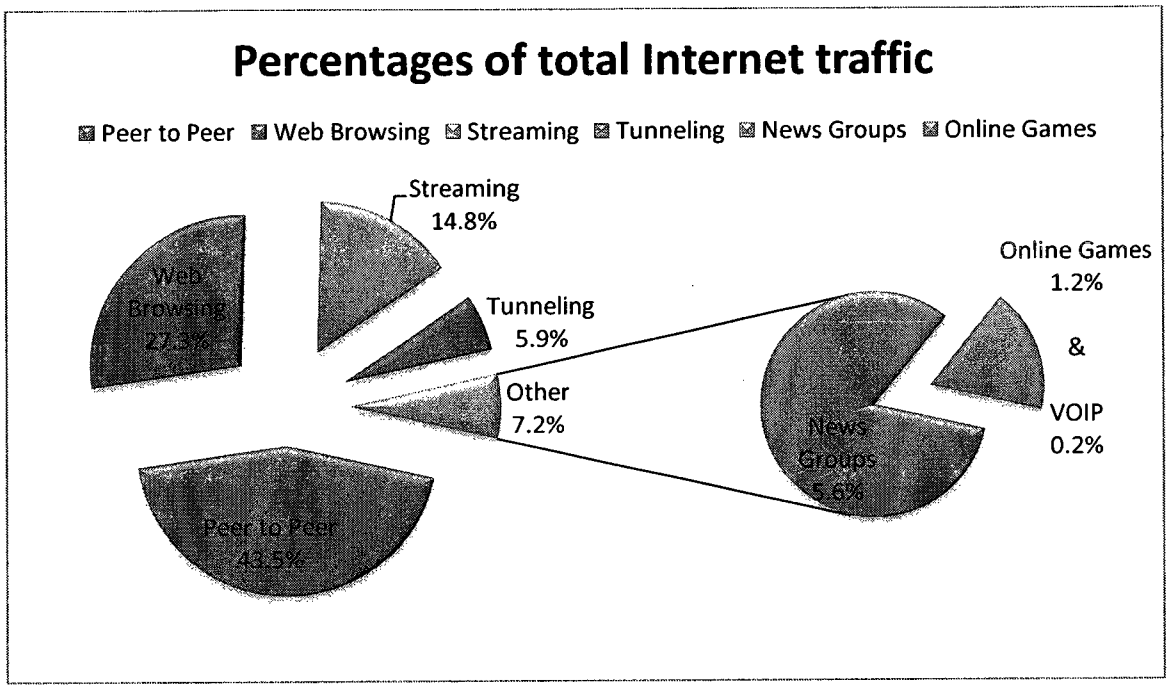


Figure 1.3: Percentages of total Internet traffic

Among those disadvantages, one of the biggest issues that the traditional client-server models face is the scalability problem. When the load of the service request is light, they can work properly; but, the problem will come up when the service load is not light anymore. Imagine a very popular web server such as yahoo.com, google.com, cnn.com, etc; there may be large number of service requests at any time while the server's capacity is limited. When the load of the system (service requests) increases, it will result in the

increase of the response time crucially and relatively. Users probably must wait from a few minutes to several minutes to be able to view a simple web page.

Although there are some solutions, such as employing expensive servers with high capacities, in order to face this problem, it seems that it is not wise enough and the scalability is still vulnerable.

In order to overcome over the scalability issue, the P2P models has been needed to be proposed, in which each node (user) works both as a client and a server simultaneously.

In general, P2P models have been developed in a distributed manner where they do not depend on a single centralized server in the system. Generally speaking, a centralized server is a must for the traditional client-server network.

The difference between a P2P and a client-server model has been shown in Figure 1.4.

P2P models can be seen as a substitution for the client-server models. Therefore, they can be added as a complementary component to the client-server networks to improve their performances.

In some applications, a combination of a P2P model and a client-server model has been employed in order to achieve a better performance. As an example, the Structured overlay Networks Application Platform (SNAP) uses this combination to get to a better performance [18].

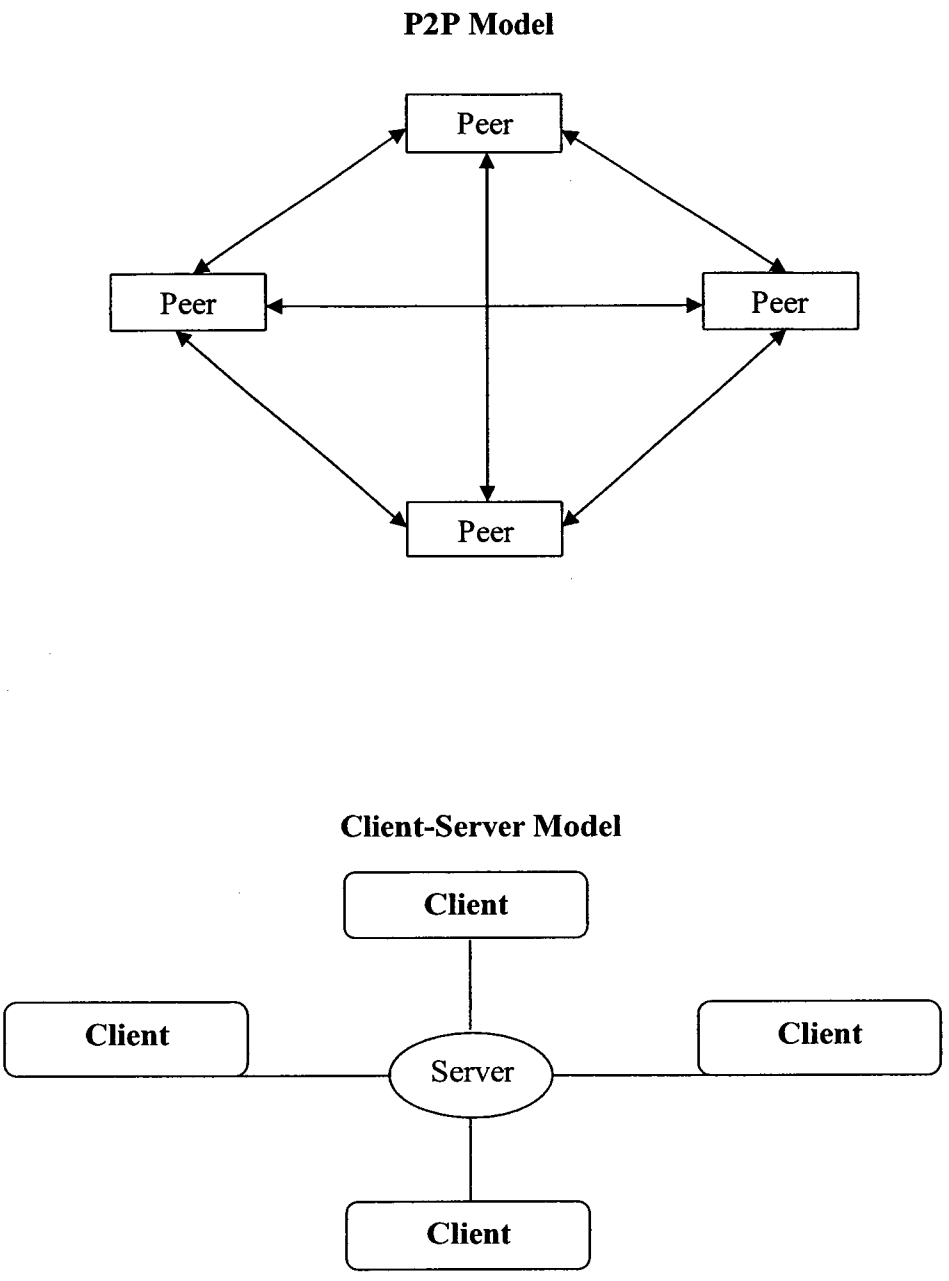


Figure 1.4: P2P model versus Client-Server model

Therefore, a P2P model has the following advantages in brief:

- **Scalability :**
  - In a P2P system as we mentioned earlier, each node (peer) concurrently acts as both a client and a server. Meaning, in a well-designed P2P model, the more peers are in the system, the better performance the network would have.
- **Anonymity:**
  - Nodes (peers) in a P2P network can interact with each other anonymously.
- **Ease of Sharing:**
  - Peers can choose the best neighbors to communicate. Moreover, P2P models do not need any publication step, which means that they neither need to create web pages nor upload information to a server.
- **Low cost:**
  - In a P2P network, every single peer (node) in that network shares a portion of its resources (e.g. bandwidth, storage space, etc) in that network. Therefore, there is no need to costly high capacity servers and also facing any special management or administrative issues.
- **Robustness and Reliability (Fault Resilience):**
  - Because of the P2P models' architecture, dependency on the single central servers is diminished and moreover, the system breakdown which could be due to the single point failure is preventable.

Although, P2P applications have lots of advantages as mentioned above, they also have some disadvantages that are coming in the following [19]:

❖ **Virus infection:**

- In a traditional Server-Client model, all the resources (files/data) are stored on the server side only, the server is maintained by professional administrators and because of this, the resources are safe and have no virus, unless the files have been replaced by hackers with illegal hacking activities.

  On the other hand, in the P2P architectures, every single peer (node) in the network acts both as a client and a server simultaneously. From this characteristic the files/data which they share with other peers in the network might be infected by the viruses and cannot be completely reliable.

❖ **Junk Information:**

- Sometimes, we see that the files that have been shared by a peer or a group of peers are not totally related to the content that they have been claimed for. Such threats, in most of the times are the tricks rather than the malice. This would lead the peers to waste their time and bandwidth in the network.

❖ **Privacy Invasion:**

- Important information such as personal information, military secrets or commercial secrets released by the P2P applications is a disruptive threat, which can be caused either by inappropriate configuration in P2P applications or by virus infection(s).

❖ **Storage Damage:**

- In a traditional Client-Server system, a client can only download the resources (data/file) from the server. In a P2P system, each node (peer) play a role both as a

server and a client concurrently, which means that it uploads data while it is also downloading. According to BitJourney's test report [19], in order to download a 7MB MP3 file with Foxy (a famous P2P application in Taiwan), it needs to do the writing action 4986 times on average and in the meantime, it requires doing the reading action 2472 times on average. These numbers lead us to the conclusion that the file has been shared while the downloading has not been yet completed. Even when the file has been downloaded completely, it has been on the process of getting shared on the internet till the file had been removed or the P2P application has been shut down completely. However, the reading actions are not essential in the traditional Client-Server models. By doing a simple calculation, it would be revealed that the damage has been increased approximately about 49.6%.

1.4 INTRODUCTION TO P2P FILE SHARING SYSTEMS

In a P2P file sharing system as we described earlier, the resources (files or data) have been stored that the users, which are called "peers" or "nodes", and they are normally personal computers, and the network generally does not have a centralized control unit.

Because of the P2P models' infrastructure, each peer at the same time works both as a client and a server; which means that during the time which it is connected to the network, it might make a request for a specific file or data to other peers in the network and simultaneously it also might reply to the other peers' requests in the system.

Before P2P file sharing model has been introduced, the client-server model by using some protocols such as File Transfer Protocol (FTP) or Hypertext Transfer Protocol (HTTP) was responsible for the most of the file-sharing on the Internet. We also should mention that they are still in use. Nowadays as we see them in some file-hosting sites proposed a few years ago, such as rapidshare.com, megaupload.com, etc. For example regarding the Rapidshare file-hosting website, we should say that it is a German-owned one-click hosting pay and free-service (the free service comes with some limitations on the download speed and the number of downloads per day) website which operates from Switzerland[20]. The original Rapidshare's website address is rapidShare.de, which uses the German top-level domain ".de" [21]. On October 20, 2006, Rapidshare announced that "Unfortunately all drives of RapidShare.de are full right now" [22]. We see that how a huge file-sharing site like rapidshare.de with having a lot of super expensive and gigantic equipments has been gotten full and could not be able to handle any other requests. This has made its founders established a new website (rapidshare.com), which is now one of the largest file-hosting sites in the world and claims that its clients have uploaded more than 10 Petabytes (10^{16} bytes) of files onto its servers and can fulfill concurrently, up to three million requests. Later on, when the number of clients increases, this will lead to the increase in the load of the system and the reduction of the performance of the network. Again they will need to buy larger equipments and pay a lot of money to be able to handle the volume of the requests at that time. As a result, this method could not be a realistic and wise solution for this problem. On the other hand, P2P file sharing systems show their excellent scalability in these

situations. Even in a well designed P2P model, when the number of users (peers) increases, the performance does not get decreased and sometimes it can be increased surprisingly, if the system was designed well. No wonder that the P2P file sharing applications have been being become popular.

The P2P models, based on the connection protocol which they use, can be classified into three main categories which are structured, unstructured and hybrid models.

Structured P2P systems has a strict structure to connect the peers to each other whereas in unstructured systems, each peer will be connected to a fixed number of other peers arbitrarily where the files' locations have not been reported.

Each of structured and unstructured approaches has its own advantages and disadvantages. Hybrid approaches have been introduced to use a combination of these two approaches that overcome each of their disadvantages while keeping their benefits.

1.4.1 FIRST GENERATION OF P2P FILE-SHARING SYSTEMS

In the first generation of P2P file sharing applications, both of the P2P and client-server models are combined together. For instance, we can mention the Napster [23, 24, 25, 26], that has a centralized server, which is generally utilized as the centralized file index server. The centralized system controls the traffic among the peers in the network; and the server keeps the information, regarding the shared file/data; and the peers upload the specific resource in order to provide them to the other users that are willing to download that shared resource in the network. In this model, whenever a peer joins the network or sends a request for a specific file/data which is in the network, the network will send him

a list of peers who have that specific resource. It then could be able to start communicating with those peers and download the resource from them.

These kinds of P2P systems have single centralized servers, pretty similar to the client-server model, they have a single failure point at their central servers; which means that if their single central servers stopped working properly, all of the network would be affected and might not work properly to any further extent.

1.4.2 SECOND GENERATION OF P2P FILE-SHARING SYSTEMS

The second generation of the P2P file sharing network, started by arrival of the Gnutella [32]. In this model of P2P file sharing networks, there would be no need to any centralized server as in the first generation. In other words, this generation of P2P file sharing networks is a decentralized P2P model which makes higher-capacity peers act as index server and the lower ones ramify from them.

Distributed hash tables (DHTs) [27], were introduced with the appearance of the second generation of P2P file sharing systems. By selecting a range of nodes (peers) to index certain hashes, we see the searching efficiency has been improved. As some examples for this class of P2P applications, we can mention some such as Kazaa, Gnutella, eDonkey2000 and Emule [25, 29, 26, 31, 32].

1.4.3 THIRD GENERATION OF P2P FILE-SHARING SYSTEMS

The third generation of P2P network is an indirect and encrypted P2P network [29, 24].

There is no doubt, that in fact, the BitTorrent (BT) merits the name of the third generation file sharing systems. BitTorrent is named the third generation P2P technology which was invented by an American software engineer, Bram Cohen, in 2001[33].

The third generation of the P2P file sharing network, is an indirect and encrypted P2P network. It interconnects uploaders (seeders) to downloaders (leechers) by using a tracking file (.torrent), while it does not use a decentralized server to search sharing files. By using this method, it would get rid of the searching issues for sharing files across the network. However, existence of online forums or some communities to exchange .torrent files is vital [34].

1.4.4 FOURTH GENERATION OF P2P FILE-SHARING SYSTEMS

There are services which send streams instead of files over a P2P network. Therefore, we can listen to a radio channel or watch a television channel without using any server.

The streaming media is distributed over a P2P network and it would be really helpful that instead of using a treelike network structure or any traditional client-server model. A P2P model gets employed for this aim [35].

We also need to bring it into attention that in this category of P2P networks, it is the streams which get exchanged rather than the files as in the other previous classes of P2P models. For example, some of the popular streaming sharing systems are PPlive, PPstream, Peercast, Miro, Cybersky and Demo TV [36].

1.5 BITTORRENT

1.5.1 INTRODUCTION

BitTorrent has the highest popularity among the P2P file sharing applications. According to a recent study done by Ipoque in 2009 [37], a leading European providers of deep packet inspection (DPI) solutions for Internet Traffic Management and analysis, BitTorrent is still the most common file sharing protocol in the world. It is responsible for almost 45-78% of all P2P traffic [38].

Ipoque statistics show that more than 90% of the P2P traffics is generated by two protocols, which are Emule (eDonkey) and BitTorrent. Gnutella only stands for nearly 2% of the P2P traffic, and Kazaa is almost nothing.

BitTorrent traffic iconsists of movie downloads, while Edonkey/Emule traffic seems miscellaneous and mischievous. Ipoque addressed that, "By looking at the number of shared files it becomes apparent, that small-volume content types such as e-books are massively shared, too. Music, movies, pornography and TV-series are the most often shared content types for BitTorrent. eDonkey's relative number of pornographic files is at 30% about twice as high as for BitTorrent."[39].

In a BitTorrent network, peers (nodes) are divided into two main categories which are called downloaders (leechers) and uploaders (seeds). A downloader would be defined as a peer which only has a portion (or none) of the file or data that is interested in. On the other hand, a seed would be defined as a peer that has the whole file and provides its upload bandwidth to serve other downloaders.

1.5.2 HOW BITTORRENT WORKS?

In earlier versions of BitTorrent, when there were some central servers, when a peer joined the network, it would first acquire a torrent file. A torrent file is a small size file which has the Uniform Resource Locator (URL) of a tracker. A tracker would be defined as an Internet server that arranges communications among peers. Also torrent files generally can be found on file sharing websites or through the search engines. When a peer gets the specific torrent file that it was looking for, it would be connected to the tracker. The most important function of a tracker is returning the list of peers, which have the desired file requested by the newly joined peer.

After the newly joined peer obtains the torrent file, and due to that the other peers' information, it would start communicating with them, the downloading process would begin and in the meanwhile, they also communicate with the tracker and update the information recurrently. If for any reason the tracker fails, the downloading process for peers that already began the file exchange would not be interrupted but no more peers could be able to join the network and start interacting with other peers.

Therefore, the tracker is vulnerable and could be seen as a single point of failure. Fortunately, this problem has been resolved in the latest versions of the BitTorrent and the centralized tracker has been has been replaced by the multi-tracker torrents in order to elevate the reliability when the tracker stops working. With this method, when one of the trackers fails, the other ones would support the file transfer [40]; and the tracker-less torrents when every single peer (node) in the network can also play the role of a tracker and as a result, having a single tracker server would be avoidable.

Generally speaking, the tracker-less torrents are divided into two main categories; Azureus Distributed Database (DDB) and Distributed hash table (DHT) implementations [40, 41, 42].

In the P2P file sharing applications, files would be divided into smaller and fixed size pieces. Typically, in the BitTorrent application, the default size of a piece is 256KB. And each of these pieces also would be divided into smaller and fixed size sub-pieces, which a default size of 16KB and regularly requests for five sub-pieces would be sent all at once. Normally, during the downloading process each peer has different pieces from the others in that network. It is also obvious that when the number of pieces that a peer has increases, the number of requests which it also receives would increase and vice versa, the less number of pieces that a peer holds, the less number of requests it would receive. BitTorrent uses four main piece selection strategies; namely, random first, rarest first, strict priority and the endgame mode, in order to keep its performance acceptable.

In the following, each of these four piece selection strategies will be described in brief:

- **Random first:** A newly joined peer to the network with no pieces, will select a piece to download at random. This strategy guarantees that the new peer downloads the first piece as soon as possible and starts serving the other peers in the system.
- **Rarest first:** It simply says that the peer should first download the rarest piece in the network and the random first is just an exception to the rarest first. For instance, when there is only one copy of a piece available in the network and the peer holding that, leaves the network, then we can find no peer in the network which has that piece. In

order to reduce the possibility of such situations, having the rarest first strategy is significantly beneficial.

- **Strict priority:** It means that when a peer makes a request for a sub-piece of specific piece, the remaining sub-pieces of that particular piece would be requested before any other sub-pieces, which belongs to other pieces.
- **Endgame mode:** Consider a situation when a peer has already downloaded all the pieces except one piece which is left. In this condition, the peer sends the request for the left piece to all of its neighbors (all of the peers that it is in communicating with) and it could probably result in the last piece duplication; and due to that some of the bandwidth would be wasted [24, 34].

1.6 AN INTRODUCTION TO P2P STREAMING SYSTEMS

Before that audio or video stream could be played, streaming media players have to buffer segments of that stream. After a segment of that stream gets buffered, the media player would play it and simultaneously, the media player is also dealing with the buffering of the next segment. In an ideal manner, the media stream (video or audio) which is playing should not be stopped [45]. Regrettably in real world, the internet servers and internet channels normally get overloaded and due to that, users who are watching (or listening to) a specific media will be encountered with a frozen frame and because of that, they have to wait from a few seconds to several minutes to be able to watch (listen to) the rest of the media stream. P2P streaming model, which is categorized as the fourth generation of P2P models, is the key to improve the Internet streaming.

Some examples of the P2P streaming applications are PPLive, PPStream, Coolstreaming, QQLive, SopCast, Feidian and TV Ants. These applications, by employing a similar technology to that of BitTorrent (BT), allow users to watch (or listen to) their favorite media stream (such as TV channel or radio channel) [45, 46]. By using the BitTorrent technology in these P2P streaming application, every single peer (user) simultaneously views (downloads) a media stream in the network and also uploads the resources (media streams that it has already downloaded) to the other interested users that are looking for those specific pieces in the network. It is obvious that in this kind of networks, every single user (viewer) act both as a downloader (viewer or a client) and also as an uploader (broadcaster or a server) simultaneously.

Therefore, in a well design P2P streaming application, when the number of users (peers) increases, the performance of the system improves and the possibility of having discontinuity (freezing) during watching or listening to a media stream perhaps will be reduced. In this generation of P2P applications, each node (peer) automatically would find different connection nodes and would download its favorite media stream from the closest peers (nodes with the closest distances from it). We also need to mention this good point that new P2P streaming applications have been designed in a way in which the media stream (data) is buffered in the memory (cache), instead of the hard disk which used to be in the other generations of P2P applications [45].

P2P streaming Internet TV/radio applications create a stream on a local host and then, that stream would be played by Windows Media Player, Real Player or other media

players. Again, because of the incentive mechanisms that have been applied into the P2P streaming applications, when a peer wants to receive the media stream pieces (segments), it has to send (upload) data to other peers which are interested in its pieces in return.

In conclusion, the more users share their segments with other peers, the smoother play of the media stream they would have. Thus, it would make it better for each peer that before starting time of a preferred event, it establishes a connection as soon as it can in order to increase the possibility of having a smoother play of that stream, when it is watching or listening to its media stream [45, 46].

1.7 RELATED WORKS AND RESEARCH OBJECTIVES

During the last decade, BitTorrent has drawn a significant amount of research interest. Lots of works have been done on BitTorrent-like file sharing systems. Early works have been focused on the traffic measurements and system designs for BitTorrent-like file sharing systems, [47, 48, 49]. In [50], a closed queuing system is used to model a general P2P file sharing system. In [51, 52], they studied how efficient the incentive mechanism works and they improved the collaboration among peers, by proposing some new simple mechanisms. In [53], the authors implemented a BitTorrent client and by collecting statistics, information and messages, which have been shared among the clients, they provided some insights on how BitTorrent file sharing systems works. They showed that it is an efficient and cheap solution to the traditional client-server systems. In [54], one of

the greatest works has been done on the performance and scalability of BitTorrent-like P2P file sharing systems using a simple fluid model.

However, during the past few years, a lot of research attentions have been inclined towards using P2P architecture for streaming purposes. In [55], it has been provided a streaming service by using a hybrid server/P2P streaming system approach. The clients (peers) obtain the stream from a streaming server while at the same time, they share pieces using BitTorrent. In this case, the BitTorrent protocol remains unchanged except that the clients cannot download any data prior to the current playback time. They also mention that the BitTorrent-like systems are not suitable for streaming purposes; because the peers will have only sequential pieces of the stream and as a result, the Tit-for-Tat strategy fails. However in our work we question the accuracy of this statement. We show that if BitTorrent-like systems are used for the live streaming, this statement cannot be right. Moreover, we also show that by applying some modifications to the BitTorrent-like file sharing, we would be able to reach to a satisfying QOS (quality of service). In [56], the other interesting work that has been done is CoolStreaming. CoolStreaming, uses a data-centric design of an overlay network. Its core operations are same as in BitTorrent. Similar to the Piece Selection mechanism found in BitTorrent, in Coolstreaming, based on a heuristic scheduling algorithm, the decision of requesting a particular piece will be made. However, it does not explain the trade-offs involved in selecting the target number of peers or the buffer length.

Another interesting work is Chainsaw [57], which uses BT concept but also uses gossip and pushed-based approaches. The BiTos [58], uses BitTorrent for streaming purpose.

Therefore, it is similar to our work here somewhat. Unfortunately, it only discusses the mechanisms that should be applied to BitTorrent, the importance of selecting an appropriate buffer size and the piece selection strategies; and no analytical model has been proposed in this paper.

As we already mentioned, Qiu *et al.* in [54], obtained a formula to calculate the efficiency of P2P file sharing. This formula has been applied directly to P2P streaming by Tewari *at al.* in [59], but it is incorrect, because of the significant differences between P2P streaming and P2P file-sharing.

In [2], some insights into the BitTorrent-based live video streaming is provided and a general probability model and an analytical analysis of the BitTorrent-based live video streaming under some certain assumptions is proposed. Although it is similar in spirit to our work, we propose a better and more realistic probability model for the BitTorrent-like live streaming systems. We propose a stochastic model which provides some great insights. We determine how different parameters of the system can affect the performance of the system. Moreover, we go further and propose changes to the infrastructure of our BitTorrent-like live streaming system and prove that the performance of the system can even go higher.

1.8 THESIS ORGANIZATION

This thesis will be presented as follows:

Chapter 2

- **Efficiency of BitTorrent-like P2P live streaming in peers with non-prioritized and limited size buffer**

  - In this chapter, the efficiency of the BitTorrent-like P2P live streaming systems will be studied in a network that peers has non-prioritized and limited size buffer. In section 2.1, the motivation would be described and it will be followed by section 2.2, which would present assumptions of the system and the proposed model requests will be presented and analyzed. In section 2.3, numerical results will be shown that some important insights on how different parameters could affect the performance of a BitTorrent-like P2P live streaming system. And finally, it will be followed by section 2.4, which is a brief conclusion on chapter 2.

Chapter 3

- **Efficiency of BitTorrent-like P2P live streaming systems with peers which have limited size buffers and high priority only in their first slots**
  - In this Chapter, the efficiency of this type of systems will be studied. In section 3.1, the motivation would be described and the rational for the changes applied to

the architecture of the previous model will be explained. In section 3.2 assumptions of our new model will be presented. In section 3.3, through the numerical methods, we obtain some important insights which confirm the advantages of our new model. And at the end, this chapter will be followed by section 3.4, which is a brief conclusion on chapter 3.

Chapter 4

- **Conclusions and future work**
  - In section 4.1, the conclusions of this thesis are summarized and in section 4.2, some of the possible future works are discussed.

Chapter 2

EFFICIENCY OF BITTORRENT-LIKE P2P LIVE STREAMING IN PEERS WITH NON-PRIORITIZED AND LIMITED SIZE BUFFERS

2.1 MOTIVATION

As we explained in chapter one, in P2P BitTorrent-like live streaming systems, a certain file/data would be divided into many small little pieces, where the size of each piece could vary from a few hundred kilobytes up to some megabytes. A little while after the arrival of a new peer (viewer) into that network, it starts interacting with other peers in the network and the downloading (viewing) procedure begins. As discussed in chapter one, in a well designed P2P application, when the number of peers increase, the network gets larger. Surprisingly, the performance of the system would be degraded and

sometimes it even can get better; because every single peer in these kinds of networks collaborates simultaneously both as a downloader (viewer) and as a uploader (server).

Therefore, if resources of such these P2P networks are used efficiently, then they will have very impressive performances. A handful number of researches have been done in this field. For example, we consider the number of pieces that a given peer has at any point of time, could be counted as one of the most important factors that have an effect on the upload utilization. For instance, we consider a case when a peer would just join the network. At the beginning of its arrival, it has no piece and as a result, it cannot be able to upload any pieces. Therefore, its upload utilization would be equal to zero. On the other hand, the more number of pieces that a peer could have at any point of time, with higher probability it can upload its pieces to other peers and the higher upload utilization it can have.

Although P2P live streaming and P2P file sharing applications have almost the same principles and infrastructure, they also have some considerable differences that would be listed in the following:

- **Content size (file size) in P2P live streaming is unknown**
  - Unlike to P2P file sharing systems, the content size is not known in advance. When we talk about sharing a static file (media or data), we already know all the information about the file size; however, when it comes to a live streaming content, like a live soccer match, we cannot find its size. (Because it is still running and who knows when exactly that would be done?!)

- **Unlike P2P file sharing models, contents (data/media) is stored in the buffer temporarily, where normally the buffer size is limited and it is much shorter in size compared to the total length of the content(s), of a peer instead of getting stored on the hard disk.**
  - In a P2P live streaming application, content (data or media) would be stored on a very fixed and limited buffer, a memory area, instead of getting stored on hard disk. When the buffer size is finite. it means that a random peer not only cannot be able to receive much older pieces (because it does not need them anymore) but also cannot receive too new (fresh) pieces as well (because the buffer size is limited and the peer is looking only for just closer pieces).
- **P2P live streaming is very time sensitive**
  - Because the buffer size in P2P live streaming systems is fixed and has a very limited size, data/media-file should be delivered to it before the playback time.

As already mentioned, in this work we would consider P2P live streaming system with a limited size buffer. We propose a stochastic model to study the peer distribution regarding to the number of pieces that a peer could have at any point in time.

We define $\{P_i\}$ as the probability that a random peer has i useful pieces, where $0 \leq i \leq L$ and L is the size of the buffer. We also assume that all the peers in the network have the same buffer size.

2.2 STOCASTIC MODEL

2.2.1 MODEL ASSUMPTIONS

In this section, our assumptions for the stochastic model, which we proposed, will be described:

- **Total number of Pieces in the network N**
  - As mentioned earlier, in BitTorrent-like P2P applications, a random file will be divided into several small pieces. Here, we assume that the total number of the pieces of a specific media stream/data is N pieces.
- **Peer distribution P_i**
  - Peer distribution P_i, is the probability that a random peer in the network has i useful pieces in its buffer of size L when it is in steady state, where $0 \leq i \leq L$.
- μ_i
  - In our analysis, μ_i would be described as the probability that a randomly selected peer in the network, which has i useful pieces in its buffer in a specific time slot, would be able to play a piece in this time slot.
- **Useful pieces**
  - As we already discussed, any random peer in the network has a limited buffer size. Here, a useful piece for a random peer will be defined as a piece which that peer has already downloaded it but has not played it out yet.

❖ **Number of neighbours (*H*)**

- When a random peer joins to a network, as we already described about the BitTorrent-like P2P models, the first thing that it does is getting the peers' list from the tracker; These peers would be defined as its neighbours.

  In our analysis, we assume that the number of neighbours of each peer is same (with size *H)*.

❖ **Download and upload bandwidth:**

- For simplicity of our analysis, I assumed that the download bandwidths of all the peers in the network will be assumed to be unlimited and all the peers have the same upload bandwidth in that network. In order to be able to have a smooth play on every peer's side, it is also important to assume that the upload rates of the peers are greater or equal to the bit rate of the specific streamed media which is going to be streamed in the network.

❖ **Source play time (t_s)**

- In a P2P live streaming, as we already discussed, there are one or more sources which broadcast live streamed media content(s). We define t_s as the play time of that streamed media which is broadcasting.

❖ **Maximum delay (*T*)**

- It would be define as the maximum possible time difference between the play time of the source and the play time a peer. Based on our assumption, we will have $t_s + 1 \leq t_{playtime\ of\ a\ random\ peer\ in\ the\ network} \leq t_s + T$ and $T \geq 1$, where

the playtime of any randomly picked peer in the network is uniformly distributed in this interval.

❖ **Playback pointer:**

At any point in time, a randomly picked peer in the network with the buffer size L has a playback pointer on its buffer that indicates which piece has to be played at that time slot. The time that the playback pointer indicates at any point of time is called the play time for that peer at that time slot and it will be shown with t_A.

As an example, a randomly picked peer (e.g. Peer A), with buffer size L=10 and the playtime t_A=15, has been shown in the following Figure 2.1.

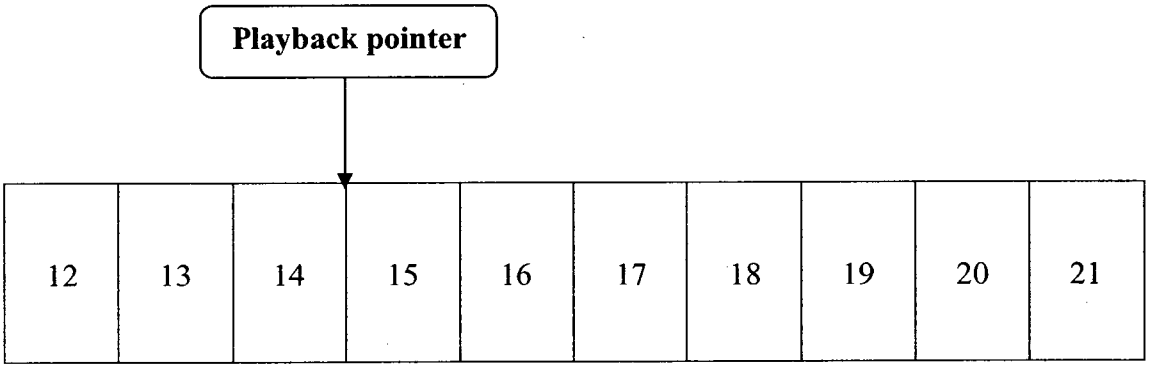


Figure 2.1: Peer A with buffer size L=10 and playtime of t_A= 15

❖ **Playing time duration of a piece**

➢ In the analysis we assumed that the fastest time that would be possible for a peer to play a piece in its buffer is equal to one time slot. In other words, a randomly

picked peer can play one piece at most during a slot time. For example, in the best situation, if a peer has a piece in its buffer and that piece is ready to be played at that time (it is the playtime of that piece), at the beginning of a time slot, then it will take one time slot of its time to play that piece.

- **Steady State**

  Based on the studies and the verified measurements that have been done, any BitTorrent-like P2P file sharing/streaming network has three stages, namely, growing stage, stabilizing stage (steady state) and decaying stage. The stabilizing phase is the stage in which most of downloads occur and due to this fact, the performance of the system can be established solely based on this stage.

  In our model, the steady state would be satisfied under the condition that $N - L \geq t_s \geq L$, when N is much greater than L ($N>>L$).

- **Number of peers in the network (M)**

  - We assume that we the network is in the steady state, and the total number of peers in the network is equal to M, where $M \geq H$.

- **Piece distribution**

  In P2P live streaming models, every single peer (viewer) has a limited buffer size (L) to store the data temporarily in it. For example, for a random peer like peer A with buffer size L and play time t_A, if we assume that at a specific time it has i useful pieces, where $0 \leq i \leq L$, then at this time, it is looking for (L-i) other pieces to download. As we already expressed, because all peers have limited buffer sizes, they only look for those pieces which are very close to their playtimes. It means that they cannot

hold old pieces for a long time in their buffers (because they want to replace them with new pieces that they would be interested in) and they also are not interested in too fresh (new) pieces to download, because they reserve the available spaces for those new pieces which are closer.

Before we start explaining how what happens inside a randomly peer's buffer, we first need to describe some terms, which are useful pieces, old pieces, new pieces, too new pieces and too old pieces, in brief:

- **Useful piece:**
  - A useful piece in a random peer's buffer would be described as a piece which has not been played yet by that peer.
- **Old piece:**
  - An old piece in a randomly picked peer would be described as a piece that has been already played by that peer but has not been replaced (rewritten) by a new piece yet.
- **New piece:**
  - A new piece would be described as a piece that a random peer is interested in downloading that piece, which means that peer has at least one old piece in its buffer to replace that with the new piece, but has not downloaded it yet.
- **Too new piece:**
  - Too new piece is a piece that a randomly picked peer is not interested in it currently, but it would be interested in future.
- **Too old piece:**

- Too old piece is a piece which a randomly picked peer has already played it and probably is not interested in it anymore.

As we mentioned earlier, every single peer in this P2P live streaming model that we proposed, has a limited size buffer, where it is much smaller than the total number of pieces of a specific streamed media. Therefore, when a useful piece in a randomly picked peer in that network plays, it would become an old piece and afterwards, that peer would be able to send a request for a new specific piece to its neighbours. Afterwards, if its request gets approved, then it will substitute the new piece with that old one in its buffer.

In order to understand the above, let us explain it intuitively with an example:

As we see, each peer has two pointers, namely, write pointer and playback pointer which track the playback process and write process correspondingly. In order to have playback continuity, the write pointer should always be ahead of the playback pointer. At whatever time that the latter gets to the former, the playback must be stopped till the write pointer leads the playback pointer to some extent. This process is called the rebuffering process. It is also important to mention that once the playback pointer of a randomly picked peer catches the end of its buffer, it has to roll back to the beginning of it in order to be able to play next useful pieces which have been already downloaded and stored in the buffer.

To better understand the processes that occurs in the following buffer of a randomly picked peer (e.g. peer A), we will provide an example. first, we assume that peer A with the buffer size L=10 has initially 10 useful pieces (i =10), pieces #12 - #21, in its buffer. Hence, the playback pointer is at the beginning of its buffer and the write pointer is

located at the end of its buffer as we see in Figure 2.2. Note that the green color represents the useful piece(s) and the red color shows the old piece(s) in the buffer.

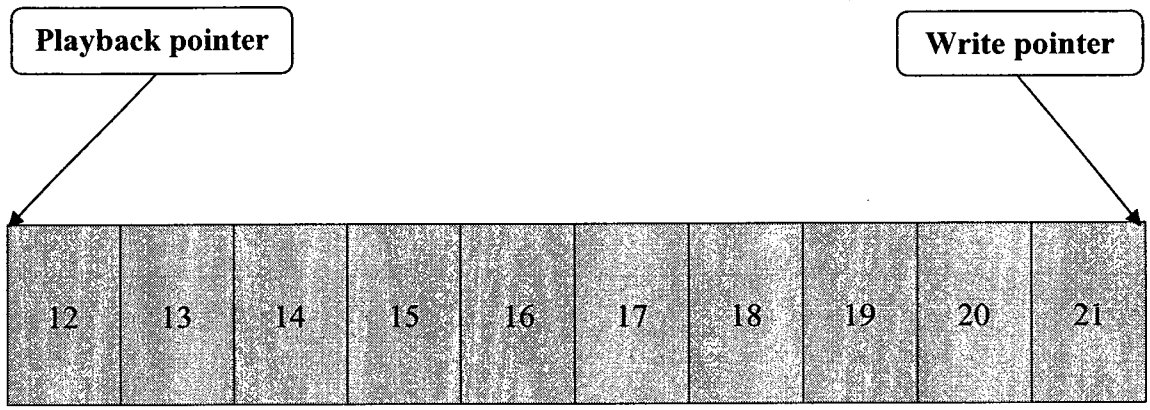


Figure 2.2: Peer A with buffer size L=10 when there are 10 useful pieces in its buffer (Pieces #12 ~ #21)

At this time, the write pointer will be frozen until the playback point goes forward (peer A starts watching the streamed media). As it is shown in Figure 2.3, a few time slots after that, then there would be some old pieces in the buffer and now peer A can look for some new pieces of its interest in its neighbours, send requests to the desired ones, and if the neighbours approve them, download them simultaneously, the write pointer moves forward (in this condition it will roll back to the beginning of the buffer) and replace the old pieces with those which is getting downloaded in its buffer. This procedure will be continued till the streamed media gets done.

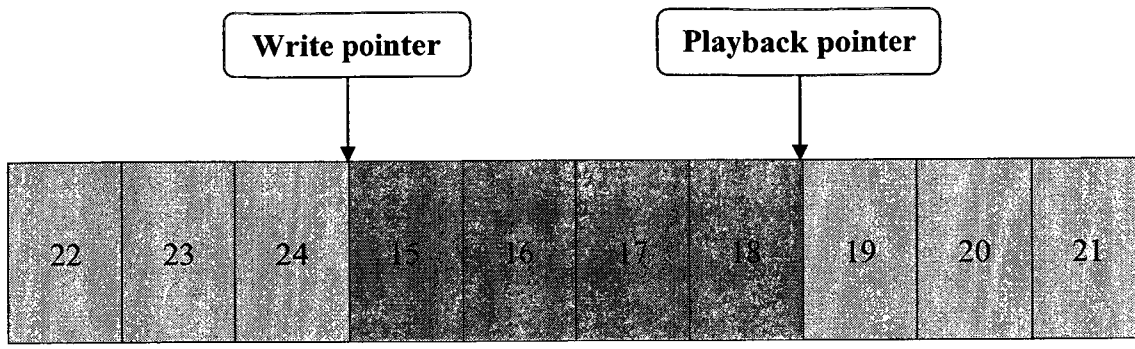


Figure 2.3: Peer A with buffer size L=10 when some of its pieces old pieces (pieces #12 ~ #14 that have been already played) have been replaced (overwritten) by #22 ~ #24 (new) pieces.

2.2.2 MODEL ANALYSIS

Before proposing the model, it is better to explain one more aspect regarding to the mechanisms which happen in a randomly picked peer in the network. Primarily, it is better to understand how a randomly picked peer in a network (e.g. peer A) with buffer size L ($L \geq 1$), i useful pieces ($0 \leq i \leq L$) and playtime t_A, will look for next piece(s) to download?

When peer A has i useful pieces in his buffer at a specific time slot when it is in the steady state condition, it looks for $(L-i)$ pieces to download in that time slot. How it would choose its pieces is an important parameter which affects its efficiency. In our analysis, we divide a peer's buffer into two sections. The first section is called real buffer and the second one is called virtual buffer. The real buffer consists of two parts, namely, useful part and old part. Useful part has those useful pieces (i pieces) which peer A has in

its buffer and old part involves pieces ((*L-i)* pieces) that have been already played by peer A and they still exist in its buffer (old pieces).

The virtual buffer does not exist in reality. At any time slot, we only imagine that there exists a virtual buffer with size L where it begins with the playtime piece and it consists of i useful pieces in the buffer, where $(L-i)$ slots in that buffer are empty. Based on our assumptions, at the beginning of any time slot, at first, that peer will take a look at its virtual buffer to make a list of missing pieces. Afterwards, it will send requests for those missing ones to its neighbours.

For example, we assume that there is a randomly picked peer (e.g. peer A) with buffer size L=10 in the network that at a beginning of a specific time slot, it has also four useful pieces (i=4) in its buffer which are pieces #18 ~ #21 and its play time is t_A=18 as it is shown in Figure 2.4. Based on our assumptions, we see that there exists $L-i$ = 6 old pieces in the real buffer.

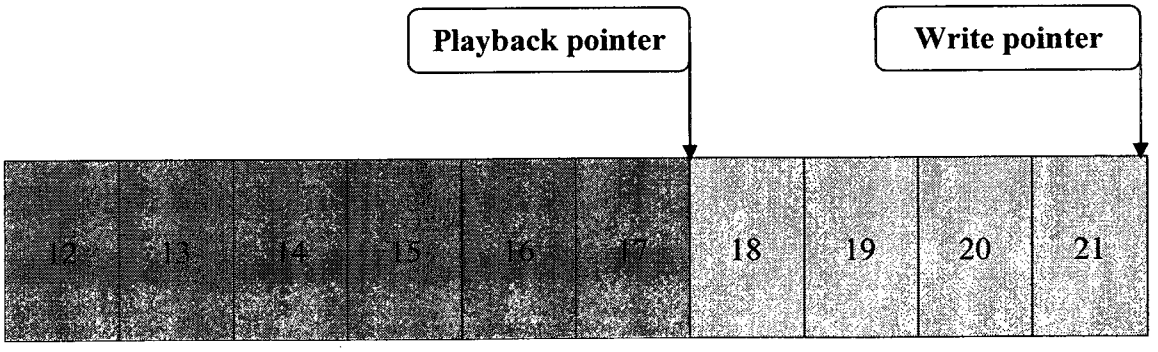


Figure 2.4: Real buffer (when Peer A with buffer size L=10 has four useful pieces (pieces #18 ~ #21) its playtime is t_A=18 at a specific time slot.)

Now we can consider our virtual buffer, where it starts with the playback piece number (t_A), has a length size of L (here we assume that L=10) and has i (here we assume that i=4) useful pieces and its buffer where (L-i) other slots of its buffer are left empty, as we see in Figure 2.5. We define green, red and purple blocks representing those slots which contain useful pieces, old pieces (those pieces that have been already played and still exist in the buffer) and those pieces which are scheduled to be downloaded in that specific time slot, which have not been downloaded yet, respectively.

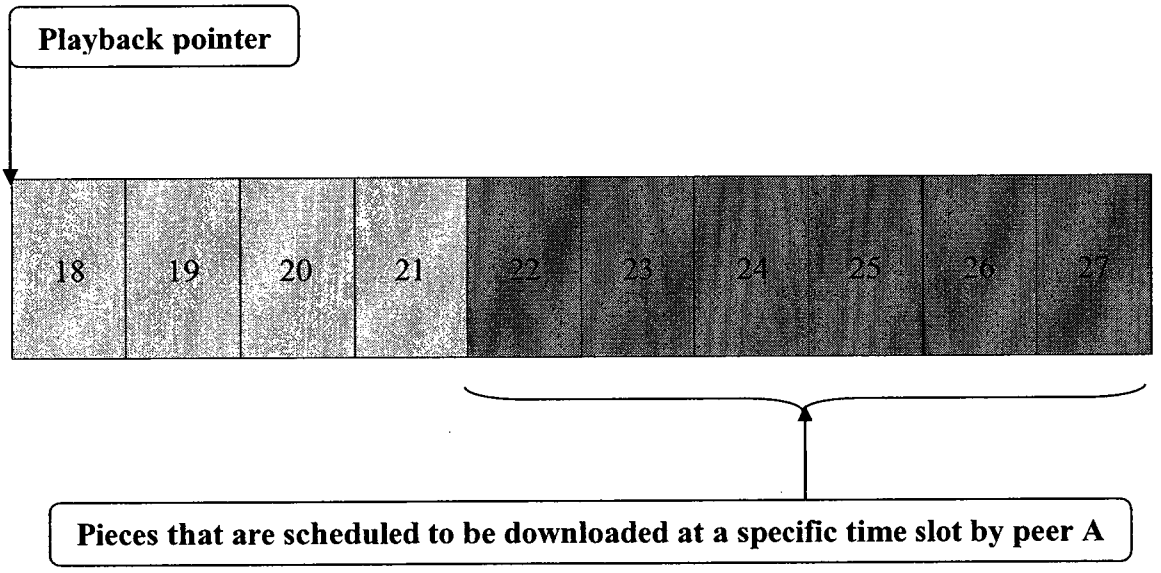


Figure 2.5: Virtual buffer (when Peer A with buffer size L=10 has four useful pieces (pieces #18 ~ #21) its playtime is t_A=18 at a specific time slot.)

So, if we merge the real buffer and the virtual buffer together, as it is shown in Figure 2.6 for peer A, we can have a better perception about the events that take place at any time slot inside any randomly picked peer in the network.

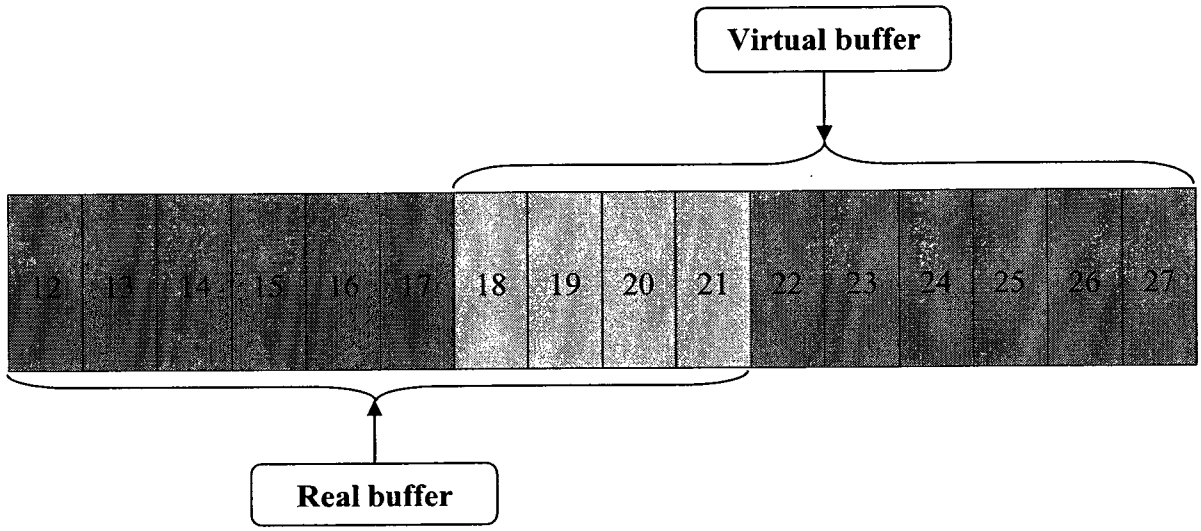


Figure 2.6: An insight into the peer A's buffer, when its buffer is the product of merging its real and virtual buffers together.

With the assumption and explanations that have been described, the stochastic model can be shown as Figure 2.7.

At any time slot, a randomly picked peer that has a fixed and limited buffer length of size L, will be in one of those $(L+1)$ states, which have been shown in Figure 2.7. The value which is written in each of the above states (circles) indicates the number of useful pieces that exist in the buffer of that peer at that time slot. In other words, it is obvious that a peer with buffer size L would have zero piece, one piece, two pieces,...,up to L pieces at

most in its buffer at any time slot and hence, it would be fitted in one of the above L states.

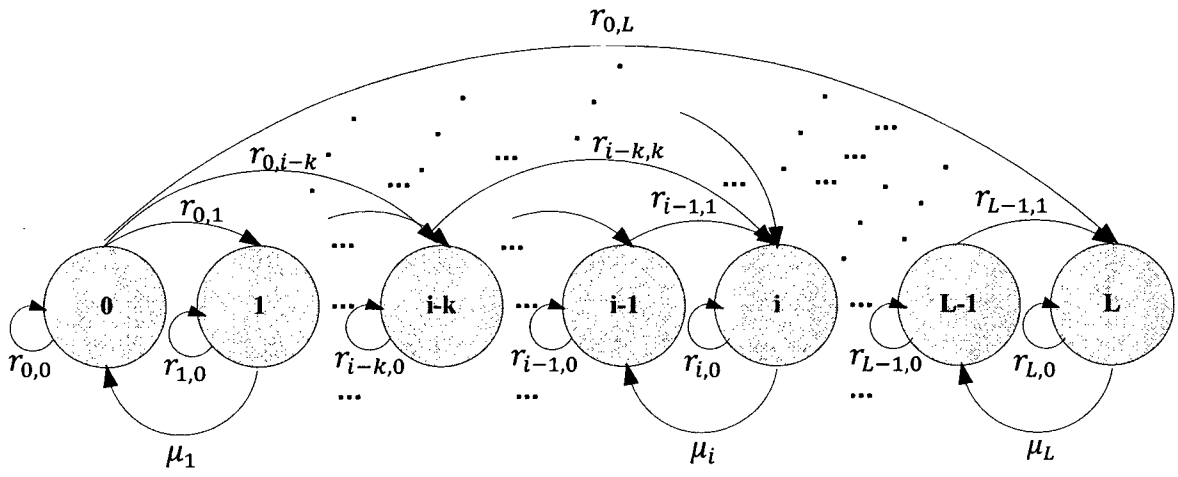


Figure 2.7: A discrete-time stochastic model for a BitTorrent-like P2P live streaming system when peers have limited buffer length of size L

In BitTorrent-like systems, whenever a peer receives a piece or some pieces, it will update its recent information with its neighbours. Likewise, whenever its neighbours obtain pieces, they will also inform it with their latest information about pieces that they have.

In our BitTorrent-like P2P live streaming model, at the beginning of any time slot, a randomly picked peer in that network will send requests for those pieces of its interest to those neighbours which have those ones. And because any peer in these networks acts simultaneously both as a client (downloader) and a server (uploader), therefore that peer would probably receive one or some requests from its neighbours too.

If that peer receives more than one request at any time slot, it will approve one of the requests, fulfills that request, and rejects the other ones. How and based on what condition(s) that peer will choose one of these requests and refuse the rest of them is out of scope of this work and could be a part of the future work. In a nutshell, we would just mention that this decision would be made by a built-in incentive mechanism designed in BitTorrent-like P2P applications.

In the next step, we continue our work by analyzing the interactions of two peers, which are neighbours. We assume that peer A has *i* useful pieces and peer B has *j* useful pieces in their buffers at a random time slot.

We define *F(i,j)* as the probability that peer A with *i* useful pieces in its buffer would not be interested in peer B, when it has *j* useful pieces in its buffer. It is obvious that peer A would not be interested in peer B's pieces. If peer A contains all the pieces that peer B has in its own buffer at that time.

In order to derive *F(i,j)*, we should consider four mutually exclusive cases depending on the situation of peer B's play time (t_B) to peer A's play time (t_A):

Case 1- When $t_s + 1 \leq t_B \leq t_A - L$

$$P_1' = \sum_{t_A=t_s+1}^{t_s+T} \sum_{t_B=t_s+1}^{t_A-L} \left(\frac{1}{T^2}\right) \quad (1)$$

Case 2- When $t_A - L + 1 \leq t_B \leq t_A$

$$P_2' = \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=Max(t_S+1,t_A-L+1)}^{t_A} \sum_{x=Min\,(t_A-t_B,i)}^{Min(L-(t_A-t_B),i)} \frac{\binom{L-t_A-t_B}{x} * \binom{t_A-t_B}{i-x} * \binom{t_A-t_B+x}{j}}{\binom{L}{i} * \binom{L}{j}} \left(\frac{1}{T^2}\right) \quad (2)$$

Case 3- When $t_A + 1 \leq t_B \leq t_A + 2L - j - 1$

$$P_3' = \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=Min(t_S+T,t_A+1)}^{Min(t_A+2L-j-1,t_S+T)} \sum_{x=i-Min\,(t_A-t_B,i)}^{Min(L-(t_B-t_A),i-Min(t_B-t_A,L-j))} \frac{\binom{t_B-t_A}{i-x} * \binom{L-t_A-t_B}{x}}{\binom{L}{i} * \binom{L}{j}} *$$

$$* \frac{\binom{t_B-t_A+x}{j} * \binom{t_B-t_A-x}{i-x-Min(t_B-t_A,L-j)}}{\binom{t_B-t_A}{i-x}} * \left(\frac{1}{T^2}\right) \quad (3)$$

Case 4- When $t_A + 2L - j \leq t_B \leq t_s + T$

$$P_4' = \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=t_A+2L-j}^{t_S+T} \left(\frac{1}{T^2}\right) \quad (4)$$

Therefore, *F(i,j)* can be expressed as follows:

$$F(i,j) = P_1' + P_2' + P_3' + P_4' \quad (5)$$

At any given time slot, if peer B has a pieces (or some pieces) that peer A is interested in, peer A will send a request to peer B. In this work, we assume that any random peer in a network has H neighbours, and it is a constant. We also assume that at any time slot if any randomly picked peer (e.g. peer B) receives more than one request, only one of them will be fulfilled by peer B randomly.

Based on the infrastructures which have been designed in P2P file sharing BitTottent-like systems, when a randomly picked peer in the network has i pieces, then the maximum number of requests which can be sent by that peer to its neighbours is equal to d, where $d=min\ (H,(L-i))$ and H is the number of neighbours which that peer has and L is the length of the buffer.

In this work, that our model is a P2P live streaming BitTorrent-like system with the limited buffer length of size L. When a randomly selected peer in the network has i useful pieces at a specific time slot, then the maximum number of requests that can be sent by that peer to its neighbours would be $d=min(H,(L-i))$ requests, where $(L-i)$ is number of pieces which that peer wants to download at that specific time slot.

We also need to find U_i, as the probability that a randomly selected peer which has i useful pieces in a specific time slot in its buffer in the network (e.g. peer A), would be interested in having a piece from one of its randomly selected neighbours. Therefore, it can be expressed by the following equation:

$$U_i = \sum_{j=0}^{L} \big(1 - F(i,j)\big) * P_j \qquad (6)$$

In the next step, we need to find the probability that a randomly selected peer in the network (e.g. peer A) sends k requests to its neighbours.

We define $F(H,i',k)$, as the probability that a randomly selected peer which has i useful pieces ,in a specific time slot, in its buffer and it is looking for i', where $i'=L-i$, pieces in that time slot, sends k requests to its neighbours. It is obvious that $0 \leq k \leq min\ (i',\ H)$.

Where $F(m,i',k)$ is a recursive function and it can be calculated by using the following equation (7), as follows:

$$F(m,i',k)=U_i*F(m-1,i'-1,k-1)+(1-U_i)*F(m,i',k), \quad (7)$$

Where $H \geq m \geq 0$, i=0, 1, ... , L and i'=0, ..., L-i and k=0, ..., min(m,i').

Using equation (7) and the initial conditions, which are shown as follows, will lead us to get to $F(H,i',k)$.

1-$F(m,i',k) = 1$; m=0 , k=0 , i=0, 1, ..., L *and* i'=0, ... , min(m, i').

2-$F(m,i',k)=0$; $H \geq m \geq 1$, i=0, 1, ..., L , i'=0, ... , min(m, i') and $i'*k$=0.

It can be seen that we can derive $F(H,\ i',k)$ by solving the following H equations, as they are shown in below :

Equation 1) $F(0,\ i',k) = 0$

Equation 2) $F(1,\ i',k) = U_i * F(0,\ i'-1,k-1) + (1 - U_i) * (1-F(0,\ i',k))$

Equation 3) $F(2,\ i',k) = U_i * F(1,\ i'-1,k-1) + (1-U_i) * (F(1,\ i',k))$

Equation H-1) $F(H\text{-}1, i', k) = U_i * F(H\text{-}2, i' - 1, k - 1) + (1 - U_i) * F(H\text{-}2, i', k)$

Equation H) $F(H, i', k) = U_i * F(H\text{-}1, i' - 1, k - 1) + (1 - U_i) * F(H\text{-}1, i', k)$ (8)

Next, we define $\overline{K_{i'}}$, as the average number of requests that a randomly picked peer, which has i useful pieces and is looking for $i' = (L - i)$ pieces in a specific time slot, will send to its neighbours. And it can be derived by calculating the expected value of $F(H, i', k)$, as shown below:

$\overline{K_{i'}} = Expected\text{-}value\ [F(H, i', k)] = \sum_{k=0}^{Min(i', H)} k * F(H, i', k),$ (9)

Where i=0, 1, 2, ... , L , i'=0, ... , L-i and k=0, ... , min(H, i').

At any given time slot, we define $\bar{k}$, as the average number of requests that any randomly picked peer sends to its neighbours at any given time slot; and it can be expressed by,

$$\bar{k} = \sum_{i'=0}^{L} (P_{i'} * \overline{K_{i'}}) \quad (10)$$

We also define $\bar{X}$, as the average number of requests that a randomly selected peer (e.g. peer B) receives from its neighbours besides peer A's request, as it s shown as below:

$$\bar{X} = \frac{(H-1) * \bar{k}}{H} \tag{11}$$

Now, we introduce Q, as the probability that a randomly selected peer (e.g. peer B) fulfils a specific request (e.g. peer A's request) among all the requests that it has received. And it can be expressed as follows:

$$Q = \frac{1}{1+\bar{X}} \tag{12}$$

Where i=0, 1, ... , L and i'=0, ... , L-i.

Next, we define $r_{i,n}$, as the probability that a peer, which has i useful pieces at a given time slot, downloads n pieces in this time slot. Again, as we discussed earlier, the *maximum* number of requests that can be sent by this peer will be *d=min(L-i,H).*

It is also obvious that the number of pieces which can be downloaded by this peer at any given time slot is a Binomial random variable with parameters k and Q. And it can be expressed as follows:

$$r_{i,n} = \sum_{k=n}^{d} \left(F(H, L-i, k) * \binom{k}{n} * (Q)^n * (1-Q)^{k-n} \right), \tag{13}$$

Where i=0, ... , L, i'=0, ... , L-i , d=min(H, i') and n=0, ... ,d.

In our analysis, we assume that in a specific time slot, in a randomly picked peer's buffer in the network, desired pieces, which are scheduled to be downloaded in this time slot, gets downloaded and stored in the buffer by that peer is uniformly distributed.

As we have discussed before, μ_i, the probability that a randomly picked peer in the network with the buffer length of size L and i useful pieces, which are uniformly distributed, in its buffer in a specific time slot, can play a piece in this time slot, would be expressed as follows:

$$\mu_i = \frac{i}{L}, \tag{14}$$

Where i = 0, 1, ... , L.

As we know, since the system is in the steady state, the peer distribution does not change during time. Therefore, based on the Figure 2.7, we will have the following equation:

$$(\mu_{i+1} * P_{i+1}) + \left(\sum_{k=0}^{Min(H,i)} (r_{i-k,k} * P_{i-k}) \right) - \left(\sum_{k=1}^{L-i} (r_{i,k} * P_i) \right) - (\mu_i * P_i) = 0, \tag{15}$$

Where i=0, 1, ... , L and $(\sum_{i=0}^{L} P_i) = 1$.

Solving the Eq. (15) will lead us to obtain the peer distribution $\{P_i\}$, where $i = 0, 1, \ldots, L$.

Although it is very complicated to find a closed form for the peer distribution, it can be seen that it is numerically solvable.

Then, we define $P_{Continuity}$, as the probability that a randomly picked peer in the network that is watching or listening to a certain live streaming content in a specific time slot, would be able to play its desired piece at this time slot. It can be expressed as follows:

$$P_{Continuity} = \sum_{i=0}^{L} P_i * \mu_i \quad (16)$$

We define d_i, as the average download rate of a peer that has i useful pieces in its buffer of size L in a specific time slot. It can be shown as follows:

$$d_i = \sum_{k=0}^{Min(H,L-i)} (K * r_{i,k}), \quad (17)$$

Where i=0, 1, ... , L.

2.3 NUMERICAL RESULTS

In this section, we show how different parameters such as number of neighbours *(H)*, delay time *(T)* and buffer length *(L)*, can affect the system performance (which is the probability of continuity ($P_{Continuity}$)) and the download rate (d_i), where i is the number of useful pieces that a randomly picked peer in the network has in its buffer in a given time slot, $0 \leq i \leq L$).

❖ The Effect of buffer length

In this part, we do the numerical results under the following assumptions:

- ✓ ***N=1000***

  In the numerical results, our streaming media has 1000 pieces in total.

- ✓ $\mathbf{2 \leq L \leq 20}$

  As we already explained, when peers are in the steady state, the length of the buffer is much smaller compared to the size of N ($N>>L$). Here we do our numerical results for different values for L, between 1 and 20.

- ✓ $\boldsymbol{H}$**=40**

  Each peer has forty neighbours.

Here, we will consider the effect of the buffer length (L) on the probability of continuity ($P_{Continuity}$) and the download rate (d_i), as they are shown in below:

❖ The effect of the buffer length on the probability of continuity ($P_{Continuity}$)

As we already said, we did our numerical results for different values of L between 2 and 20 in different values for T, T=1, 5, 10, 15 and 20.

As we see in the Figure 2.8, in order to get to an almost satisfying value for the $P_{Continuity}$, the size of the buffer (L) should have a size of at least 4 or greater. Although increasing the buffer length can be helpful, it is found out that if the size of the buffer gets increased more than a certain amount, the performance of the system drops. For example, if we consider one of following cases above, when the delay time $T=20$, we can see when $L=2$, probability of continuity is equal to 66.27% ($P_{Continuity} = 66.27\%$) and by increasing the buffer size to L=7, probability of continuity gets to 79.46% ($P_{Continuity} = 79.46\%$). However, if the buffer size gets larger the performance of the

system gets worse; where as we see, when L=20, probability of continuity drops very significantly and gets to the value of 56.72% ($P_{Continuity} = 56.72\%$), which is very unpleasant.

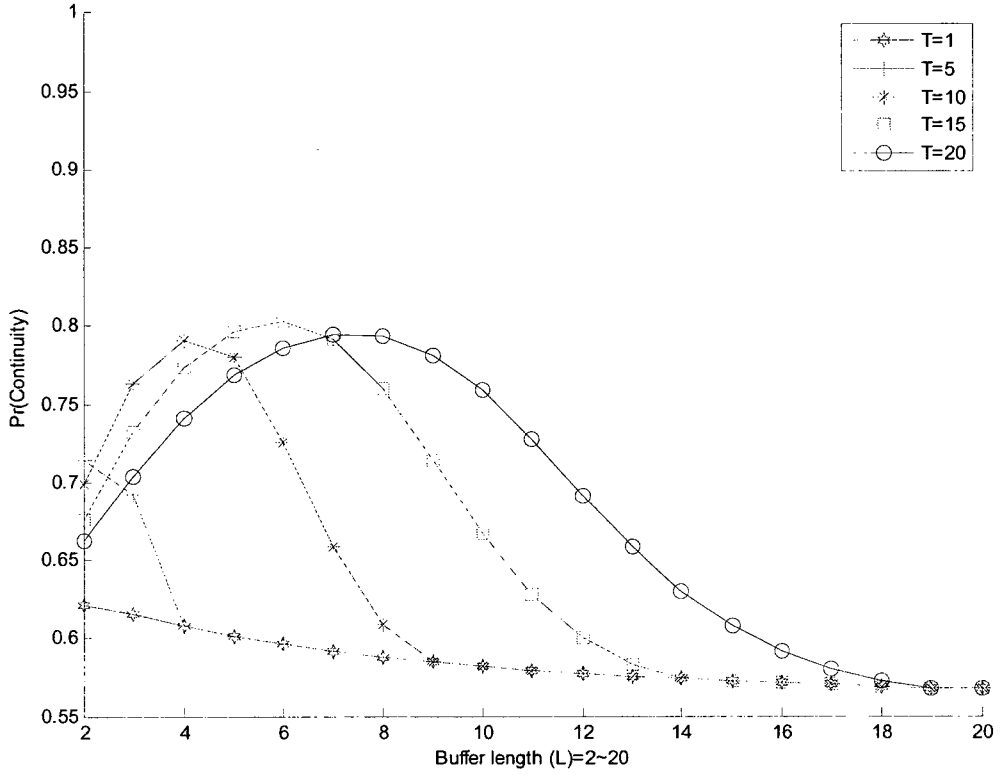


Figure 2.8: The effect of the buffer length on the probability of continuity $(P_{Continuity})$, where $20 \geq L \geq 2$ and T=1, 5, 10, 15 and 20.

❖ The Effect of Delay time

In this part, we do the numerical results under the following assumptions:

- ✓ ***N=1000***

  In the numerical results, our streaming media has 1000 pieces in total.

- ✓ $\mathbf{1 \leq T \leq 20}$

  Here we do our numerical results for different values for *T*, between 1 and 20.

- ✓ ***H*=40**

  Each peer has forty neighbours.

Here, we will consider the effect of the delay time (*T*) on the probability of continuity ($P_{Continuity}$) and the download rate (d_i).

Therefore, we divide our numerical results in this part into three main categories, which are expressed in below:

❖ The effect of delay time on the probability of continuity ($P_{Continuity}$)

We divide our numerical results in this part into five categories, as they are expressed in below:

As we already described, we did our numerical results for different values of *T* between 1 and 20 in different values for *L*, where *L*=2, 5, 10, 15 and 20. By taking a look at the Figure 2.9, we find out that there exists a relation between the size of the buffer (*L*) and the delay time (*T*); we see, in most of the cases, when some delay time (*T*) gets added to the system, the performance of the system gets increased. However, if the amount of delay time gets more than sufficient, the performance of the system drops.

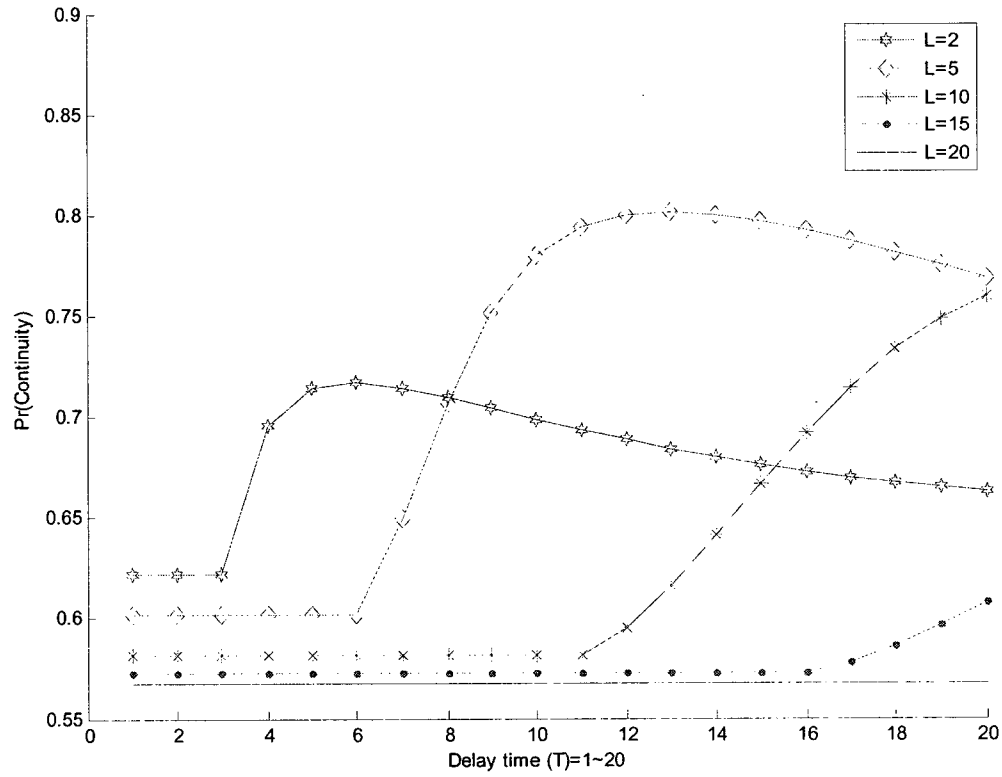


Figure 2.9: The effect of the delay time on the probability of continuity $(P_{Continuity})$, where $20 \geq T \geq 1$ and L=2, 5, 10, 15 and 20.

As an example, if we consider one of the following case in Figure 2.9 for the case that L=5, we see that the probability of continuity is around 60% ($P_{Continuity} \approx 60\%$), where $1 \leq T \leq 6$. It also can be seen that for the delay time (T), between 7 and 12, the higher amount of T brings the higher probability of continuity for a peer, when at T=12, the probability of continuity will be around 80% ($P_{Continuity} \approx 80\%$). However, it can be seen when T goes beyond 12, the larger T results in the lower probability of continuity.

❖ The effect of delay time and the buffer length on the download rate (d_i)

We divide our numerical results in this part into five categories, as they are expressed in below:

- **The effect of the delay time and the buffer length on the download rate (d_i), when $20 \geq L \geq 2$ and $T = 1$**

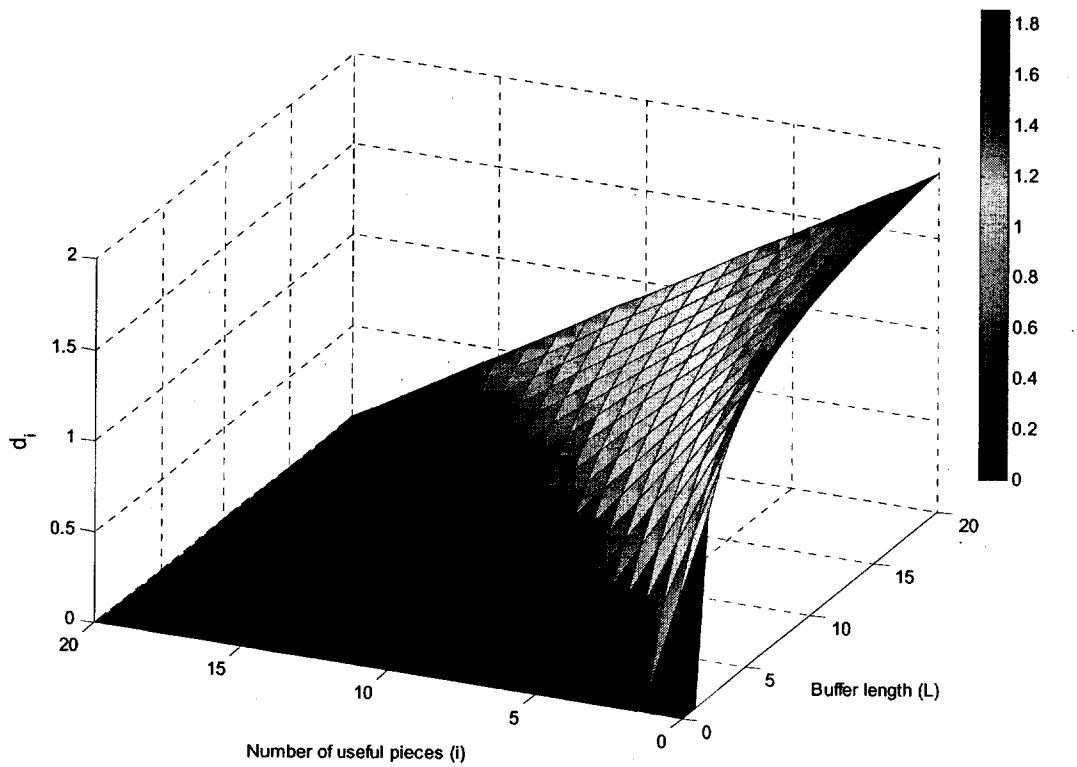


Figure 2.10: The effect of the delay time and the buffer length on the download rate (d_i), where $20 \geq L \geq 2$ and $T = 1$.

- **The effect of the delay time on the download rate (d_i), when $20 \geq L \geq 2$ and $T = 5$**

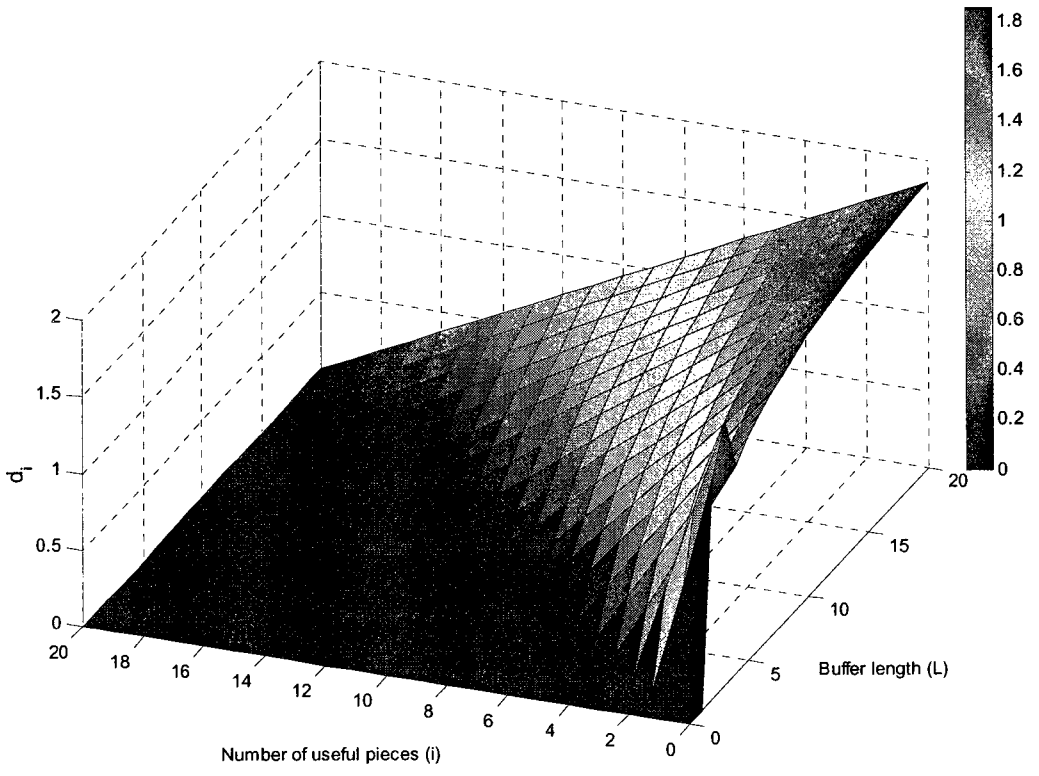


Figure 2.11: The effect of the delay time and the buffer length on the download rate (d_i), where $20 \geq L \geq 2$ and $T = 5$.

- **The effect of the delay time on the download rate (d_i), when $20 \geq L \geq 2$ and $T = 10$**

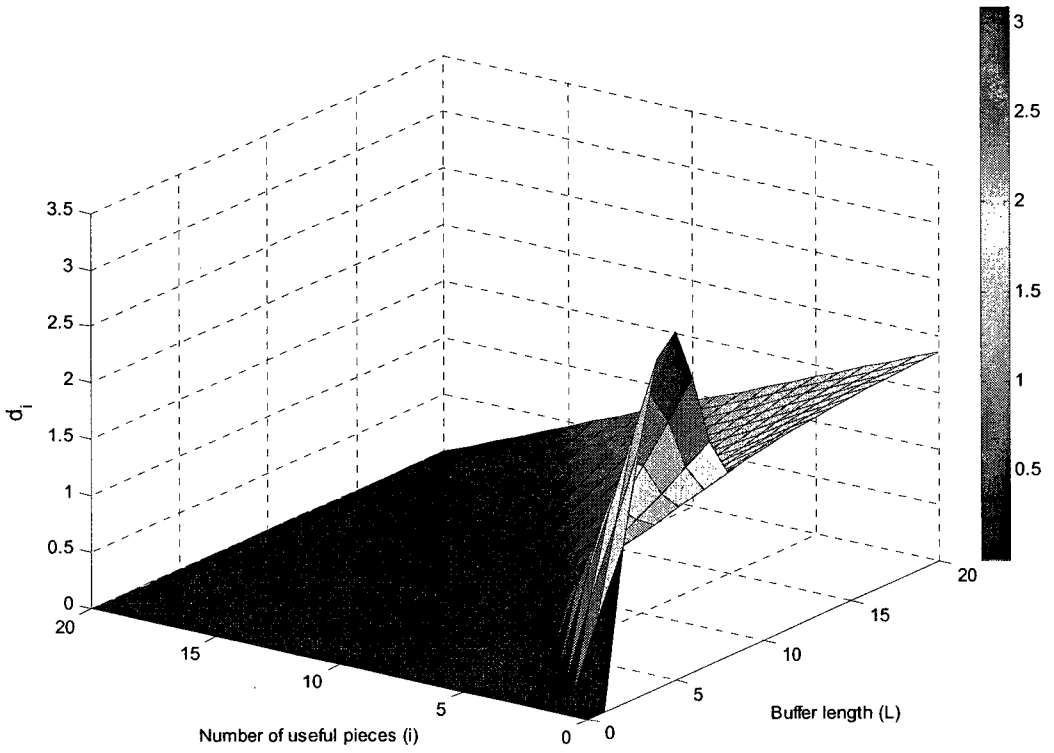


Figure 2.12: The effect of the delay time and the buffer length on the download rate (d_i), where $20 \geq L \geq 2$ and $T = 10$.

- **The effect of the delay time on the download rate (d_i), when $20 \geq L \geq 2$ and $T = 15$**

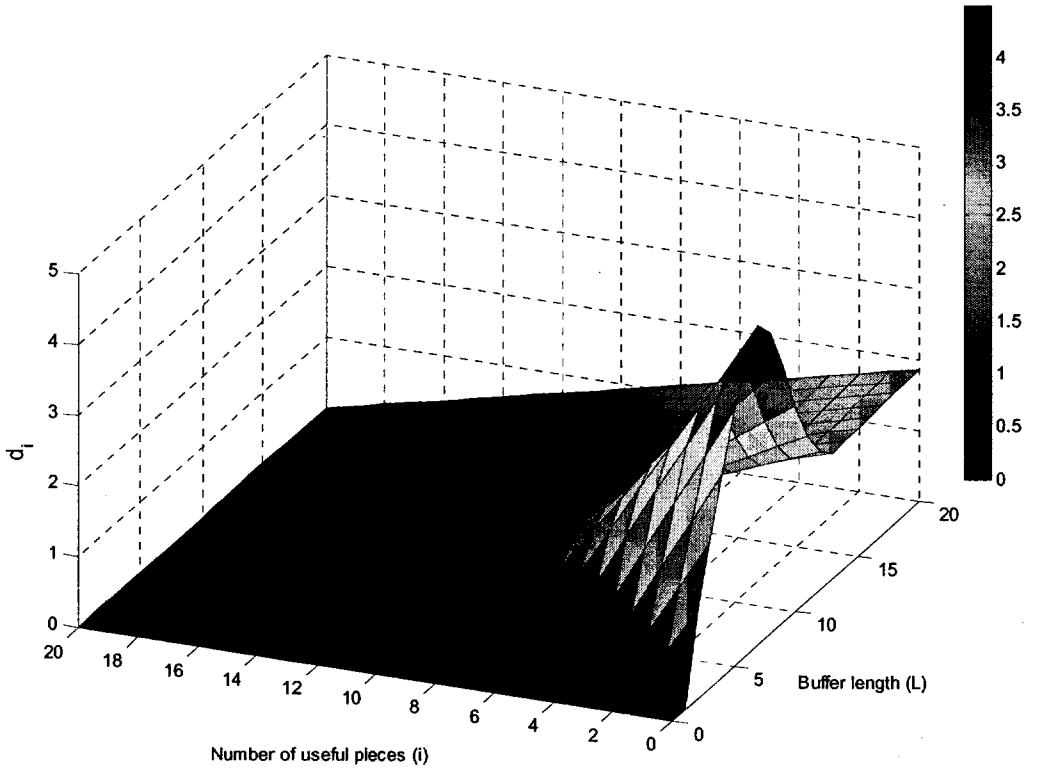


Figure 2.13: The effect of the delay time and the buffer length on the download rate (d_i), where $20 \geq L \geq 2$ and $T = 15$.

- **The effect of the delay time on the download rate (d_i), when $20 \geq L \geq 2$ and $T = 20$**

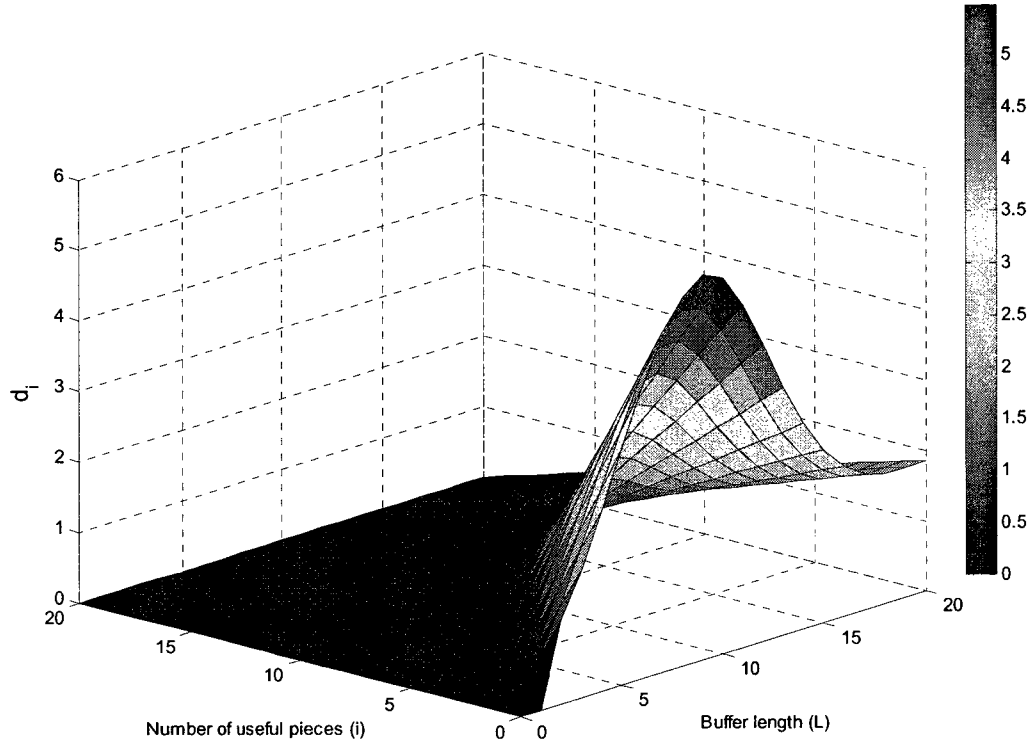


Figure 2.14: The effect of the delay time and the buffer length on the download rate (d_i), where $20 \geq L \geq 2$ and $T = 20$.

As we see in Figures, Figures 2.10, 2.11, 2.12, 2.13 and 2.14, for small values for *i*, we will have the highest download rates. When the delay time (T) gets increased in the system, the download rate will be increased. However, it does not necessarily mean that the probability of continuity for our system also gets increased. When the delay time increases in the system, peers play-times will me more diverse. This means that the probability that any given two neighbour peers could have more

different pieces compared to each other and hence, they will upload more to each other and their download rates gets increased.

❖ The effect of the buffer length and the delay time on the probability of continuity ($P_{Continuity}$)

And finally, if we want to see the $P_{Continuity}$ versus the buffer length (L) and the delay time (T) all together, where $H = 40, 20 \geq L \geq 2$ and $20 \geq T \geq 1$, it can be shown as follows:

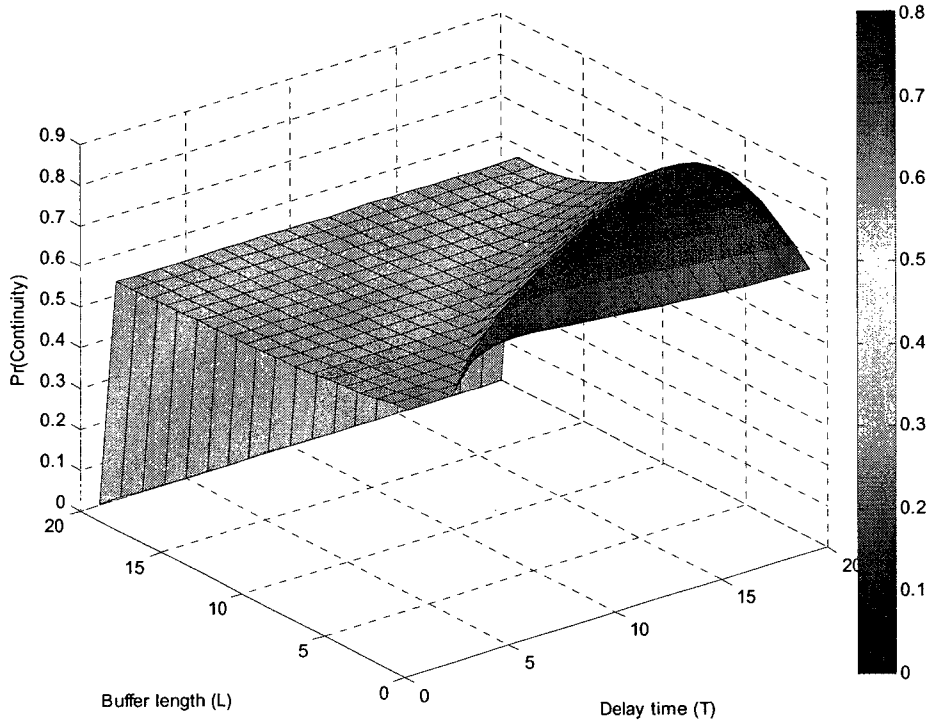


Figure 2.15: The effect of the delay time and the buffer length on the probability of continuity ($P_{Continuity}$), where $H = 40, 20 \geq L \geq 2$ and $20 \geq T \geq 1$.

❖ The effect of neighbour number on the probability of continuity $(P_{Continuity})$

In this case, we do the numerical results for some random cases (three cases) as an example, to see how the number of neighbours could affect the probability of continuity $(P_{Continuity})$.

- **The effect of neighbour number on the probability of continuity $(P_{Continuity})$, when $L = 4$ and $T = 10$ and $180 \geq H \geq 10$**

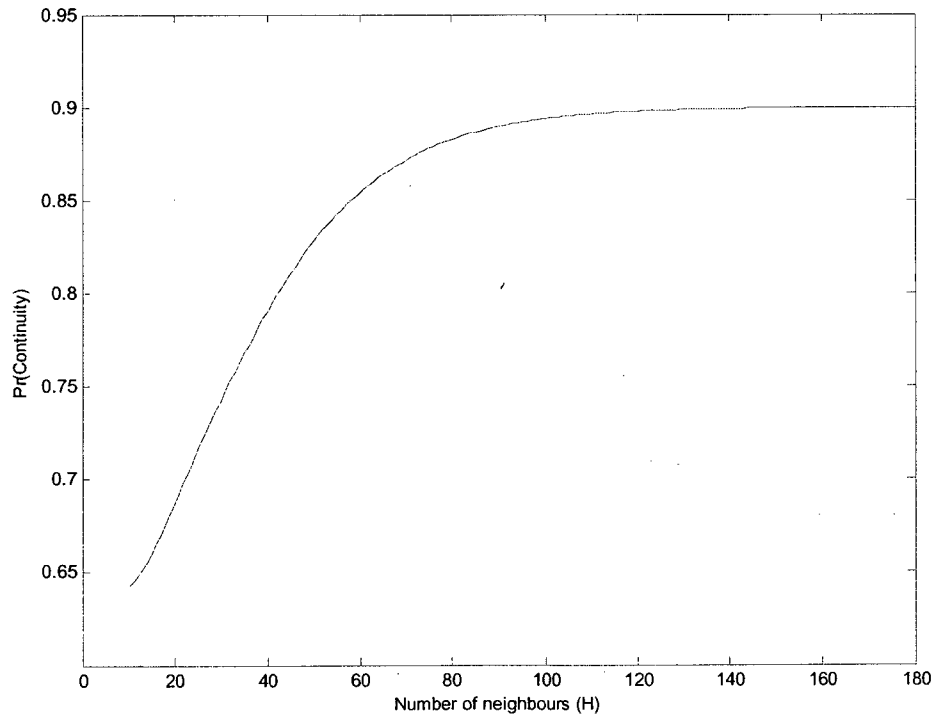


Figure 2.16: The effect of neighbour number on the probability of continuity $(P_{Continuity})$, where $L = 4, T = 10$ and $180 \geq H \geq 10$.

- **The effect of neighbour number on the probability of continuity $(P_{Continuity})$, when $L = 6$ and $T = 15$ and $180 \geq H \geq 10$**

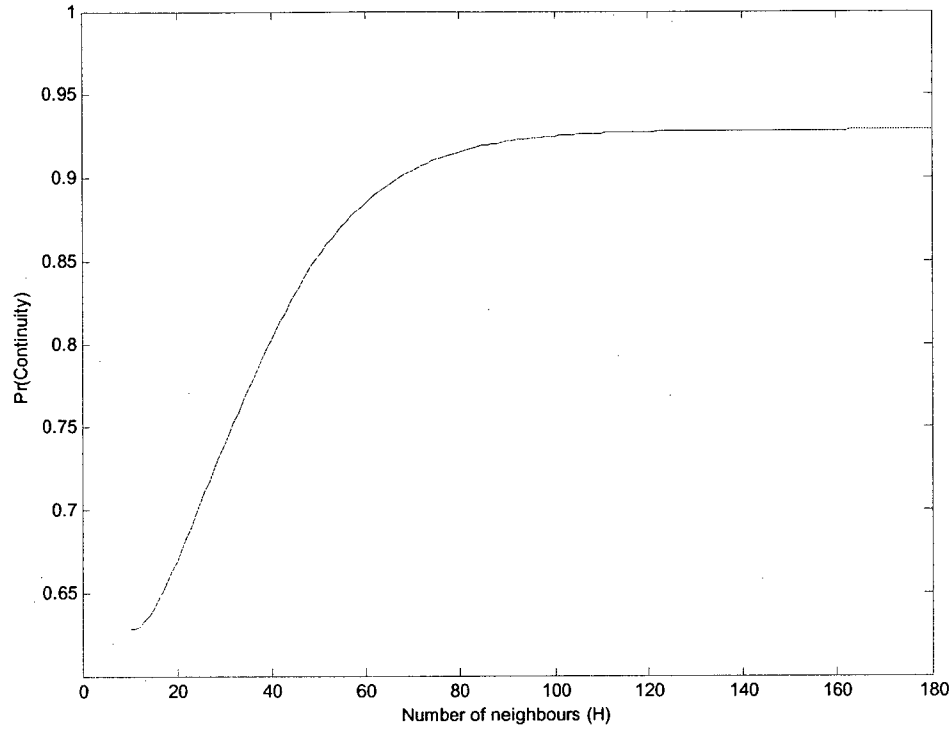


Figure 2.17: The effect of neighbour number on the probability of continuity $(P_{Continuity})$, where $L = 6, T = 15$ and $180 \geq H \geq 10$.

- **The effect of neighbour number on the probability of continuity $(P_{Continuity})$, when $L = 7$ and $T = 20$ and $180 \geq H \geq 10$**

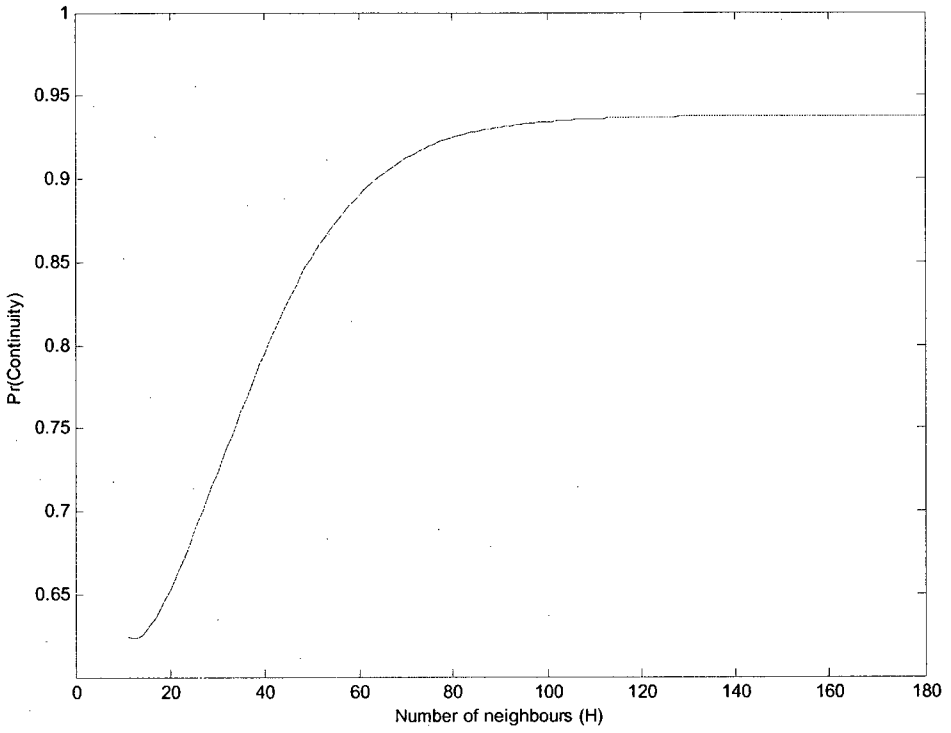


Figure 2.18: The effect of neighbour number on the probability of continuity $(P_{Continuity})$, where $L = 7, T = 20$ and $180 \geq H \geq 10$.

As we see, in Figures 2.16, 2.17 and 2.18, when the number of neighbours is too small, the probability that a peer uploads to its neighbours is small and because of that, the performance of the systems (probability of continuity) is very low. When the number of neighbours gets increased, the probability of continuity (performance of the system) will be increased. However, when the number of neighbours gets more than certain values,

because the more number of neighbours causes the more competition among peers for downloading their desired pieces, the performance of the system will not be changed significantly.

2.4 CONCLUSION

In this chapter, we proposed a stochastic model for the BitTorrent-like P2P live streaming protocols, with limited size buffer. We have analyzed the efficiency of such these systems, by solving them numerically, and could be able to get some significant insights regarding to the effect of different parameters of the system on its performance.

Chapter 3

EFFICIENCY OF BITTORRENT-LIKE P2P LIVE STREAMING SYSTEMS WITH PEERS WHICH HAVE LIMITED SIZE BUFFERS AND HIGH PRIORITIES ONLY IN THEIR FIRST SLOTS

3.1 MOTIVATION

As we already discussed before, we know that the P2P BitTorrent-like live streaming systems are time sensitive comparing to the BitTorrent-like file-sharing systems. By doing numerical results on the previous model that have been proposed in the last chapter, we come to the conclusion that employing a pure file-sharing BitTorrent-like

model would not be able to satisfy those requirements which are needed in a live-streaming context solely. These numerical results tell us that the probability of discontinuity is generally high in such these systems and as a result, their efficiencies are lower than what a peer (user) is looking for in such these systems and it is not satisfactory most of the times. This will lead us to this conclusion that in order to achieve higher efficiency and better performance in these systems, we need to apply some modifications and changes to the structure of BitTorrent-like P2P file sharing systems and make them function better in supporting content delivery with time constrains.

3.2 STOCASTIC MODEL

3.2.1 MODEL ASSUMPTIONS

In this section, our assumptions for the stochastic model will be described as follows:

- **Total number of Pieces in the network (N)**
  - As we said earlier, in BitTorrent-like P2P applications, a random file will be divided into several small pieces. Here, we assume that the total number of the pieces of a specific media stream/data is N pieces.
- **Peer distribution $P_{(i_1,i_2)}$**

- In this model, we put the high priority on the first slot of a randomly selected peer's buffer in the network and a non-prioritized priority on the rest of its slots on that buffer. Peer distribution $P_{(i_1,i_2)}$, is the probability that a randomly picked peer, which has a buffer with size L, in the network has $i_1$useful piece in the first slot of its buffer, where $0 \leq i_1 \leq 1$, and i_2 useful pieces in the remaining slots in the buffer, where $0 \leq i_2 \leq (L-1)$.

❖ **Useful pieces**

- As we already discussed, any random peer in the network has a limited buffer size. Here, a useful piece for a random peer will be defined as a piece which that peer has already downloaded it but has not played it yet.

❖ **Number of neighbours (H)**

- When a random peer joins to a network, as we already described about the BitTorrent-like P2P models, the first thing that does is getting the peers' list from the tracker. These peers would be defined as its neighbours.

  In our analysis, we assume that the number of neighbours that a peers has is the same (with size H).

❖ **Download and upload bandwidth:**

- For the simplicity of the analysis, assuming that the download bandwidths of all the peers in the network are unlimited and all the peers have the same upload bandwidth in that network. In order to be able to have a smooth play on every peer's side, it is also important to assume that the upload rates of the peers are

greater or equal to the bit rate of the specific streamed media which is going to be streamed in that network.

- **Play time of the source (t_s)**
  - In a P2P live streaming, as we already discussed, there are one or more sources which broadcast live streamed media content(s). We define t_s as the play time of that streamed media which is broadcasting.
- **Maximum delay (T)**
  - When peers in the network want to receive a piece or some pieces from the source(s), the transmission(s) will be happened with some delay (equal to T). Assuming that any randomly picked peer's play time is always between the transmission time (t_s) and the transmission time plus the propagation delay time, where $t_s + 1 \leq t_{playtime\ of\ a\ random\ peer\ in\ the\ network} \leq t_s + T$ and $T \geq 1$.
- **Playback pointer:**

  At any point of time, a randomly picked peer in the network with the buffer size L has a playback pointer on its buffer that indicates which piece has to be played at that time slot. The time that the playback pointer indicates at any point of time is called the play time for that peer at that time slot and that slot will be shown with t_A.

  For example, a randomly picked peer, called Peer A, with buffer size L=10 with the playtime t_A=18 has been shown in the following Figure 3.1.

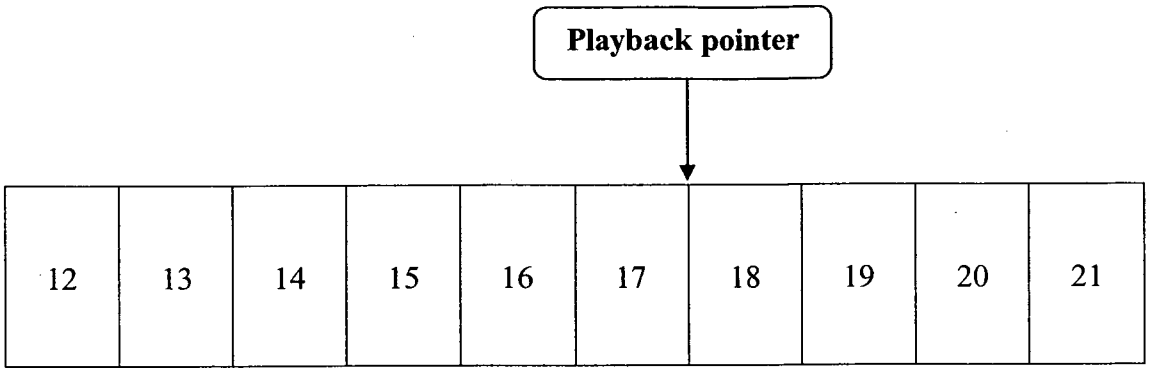


Figure 3.1: Peer A with buffer size L=10 and playtime of t_A= 18

❖ **Write pointers**

In this proposed model, at any point of time, a randomly picked peer in the network with the buffer size L and the playback pointer t_A has two write pointers. The first write pointer depending on the availability of the piece in the first slot of that peer's buffer has different responsibilities and the responsibility of the second write pointer depends on how the first write pointer acts at any given time slot. How each of these write pointers works will be discussed further.

❖ **Playing time duration of a piece**

➢ The fastest time that would be possible for a peer to play a piece in its buffer is equal to one time slot. In other words, a randomly picked peer can play one piece at most during any slot time. For example, in the best situation if a peer has a piece in its buffer and that piece is ready to be played at that time (it is the

playtime piece), at the beginning of a time slot, then it will take one time slot of its time to play that piece.

❖ **Steady State**

Based on the studies and the real measurements that have been done, any BitTorrent-like P2P file sharing/streaming network has three stages, namely, growing stage, stabilizing stage (steady state) and decaying stage.

The stabilizing phase mostly is the stage in which most of downloads occur and due to this fact, the performance of the system can be established solely based on this stage.

In our model, the steady state would be satisfied under the condition that $(N - L) \geq t_s \geq L$, when N is much greater than L ($N>>L$).

❖ **Piece distribution**

In P2P live streaming models, every single peer (viewer) has a limited buffer size (L) to store the data temporarily on it. For example, for a random peer like peer A with buffer size L and play time t_A, if we assume that at a specific time it has i_1 useful pieces in its first slot, where $0 \leq i_1 \leq 1$, and i_2 useful pieces in the remaining ($L-1$) slots, where $0 \leq i_2 \leq (L-1)$, then at this time, it is looking for $1-i_1$ piece in its first slot and ($L-1-i_2$) useful pieces in its next ($L-1$) slots.

As been denoted, because any peers have a limited buffer size, they only look for those pieces which are very close to their playtimes. It means that they cannot hold old pieces for a long time in their buffers (because they want to replace them with new pieces that they would be interested in) and they also are not interested in too

fresh (new) pieces to download, because they reserve the available spaces for those new pieces which are closer.

Before I start explaining what happens inside a randomly peer's buffer, we first need to describe some terms, which are useful pieces, old pieces, new pieces, too new pieces and too old pieces, in brief:

- **Useful piece:**
  - A useful piece in a random peer's buffer would be described as a piece which has not been played yet by that peer.
- **Old piece:**
  - An old piece in a randomly picked peer would be described as a piece that has been already played by that peer but has not been replaced (rewritten) by a new piece yet.
- **New piece:**
  - A new piece would be described as a piece that a random peer is interested in downloading that piece, which means that peer has at least one old piece in its buffer to replace that with the new piece, but it has not downloaded it yet.
- **Too new piece:**
  - Too new piece is a piece that a randomly picked peer is not interested in it currently, but it would be interested in future.
- **Too old piece:**
  - Too old piece is a piece which a randomly picked peer has already played it probably and is not interested in it anymore.

For a randomly selected peer (e.g. peer A) in a specific time slot when its playback pointer is pointing on t_A and the peer has a buffer length of size L in the network, one of the following cases could happen:

- **When peer A does not have a useful piece in its first slot ($i_1 = 0$) and has i_2 pieces in the rest of its *L-1* slots, where $0 \leq i_2 \leq (L-1)$, in this time slot.**
  - In this case, peer A is interested in the piece number t_A to download and replace it with the old piece, which exists in its first slot. Simultaneously, it is also looking for (L-1-i_2) pieces to download and replace them with those (L-1-i_2) old pieces that are located in its next (L-1) slots.

    Therefore, in this situation, we put the high priority only on the first slot and the first write pointer and the playback pointer are both located at the beginning of the first slot (piece #t_A).

    Let's assume that a randomly picked peer (e.g. peer A), with a buffer length of size L=10, at a given time slot it has its playback pointer at $t_A = 25$ and does not have a useful piece in its first slot (i_1=0) and old piece #16 still exists in this slot) and in the next 9 slots, there exists 4 ($i_2 = 4$) slots which hold 4 useful pieces (pieces #27 ~ #30) and 5 slots that hold 5 old pieces (pieces #21 ~ #25), as it can be seen in Figure 3.2. At this time slot, it is obvious that peer A is looking for the piece number $t_A = 26$ to download and replace with piece #16 and at the same time, it is also looking for pieces #31 ~ #35 to download and replace them with pieces #21 ~ #25, as it is shown in Figure 3.3.

Green, red and purple blocks show those slots which contain useful pieces, old pieces and missing pieces (those pieces that peer A is interested in them to download at this specific time slot) respectively.

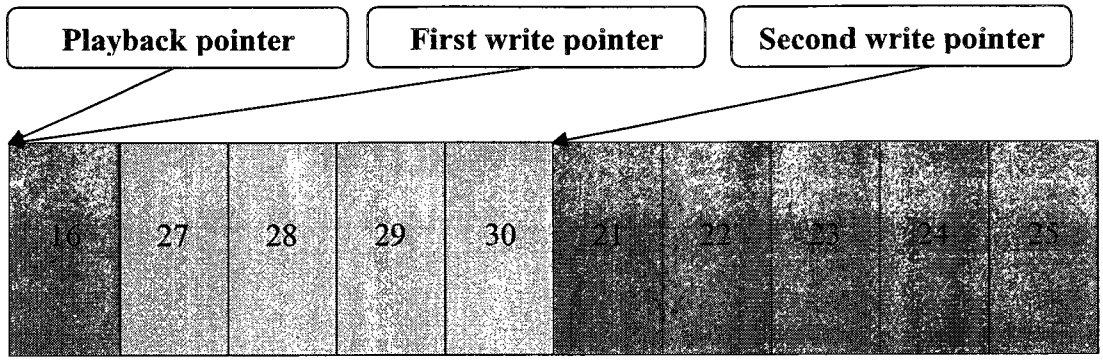


Figure 3.2: Peer A with buffer size L=10, containing old piece #16, old pieces #21 ~ #25 and useful pieces #27 ~ #30 in its buffer, in a specific time slot.

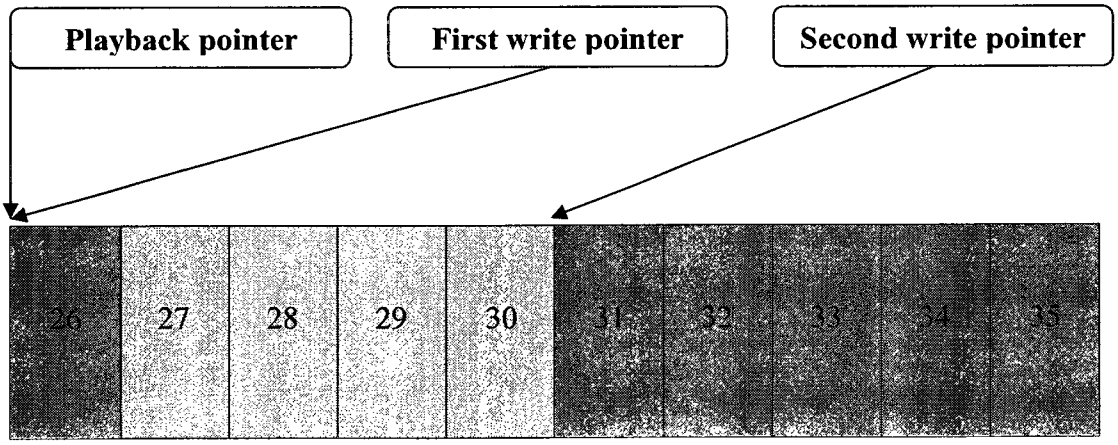


Figure 3.3: Peer A with buffer size L=10 with 4 useful pieces (pieces #27 ~ #30) in its buffer, when it is missing (and looking for) piece #26 and pieces #31 ~ 35 in a specific time slot.

In this case, when there is not a useful piece in the first slot ($i_1 = 0$), in a specific time, we see that the first write pointer is only responsible for the first slot in the buffer and the second write pointer is responsible for the remaining slots in that buffer.

By taking a look at Figure 3.2, we see that in this case, at the beginning of this time slot, the first write pointer and the playback pointer are both located at the beginning of the first slot and the second write pointer is located at the end of the last piece that has been downloaded and written in the buffer in the previous time slot.

- **When peer A has a useful piece in its first slot ($i_1 = 1$), does not have a useful piece in its next slot and has (L-2-i_2) pieces in the rest of its (L-2) slots, where $0 \leq i_2 \leq (L-2)$, in this time slot.**
  - In this case, we put the high priority only on the second slot ($t_A + 1$) and the first write pointer located at the beginning of the second slot and the playback pointer is located at the beginning of the first slot (piece $\#t_A$).

    As an example, let's assume that a randomly picked peer (e.g. peer A), with a buffer length of size L=10, at a given time slot it has its playback pointer at piece $\#t_A$ (piece #26) and has its useful piece (piece #26) in its first slot (i_1=1) and is missing the piece #27 (which should be in its second slot); and in its next 8 slots, there exists 3 slots which hold 3 useful pieces (pieces #28 ~ #30) and 5 slots that hold 5 old pieces (pieces #21 ~ #25), as it can be seen in Figure 3.4.

At this time slot, it is obvious that peer A is looking for the piece number $\#(t_A + 1)$ (piece #27) to download and replace with piece #16 and at the same time, it is also looking for pieces #31 ~ #35 to download and replace them with pieces #21 ~ #25, as it is shown in Figure 3.5. Again, Green, red and purple blocks representing those slots which contain useful pieces, old pieces and missing pieces (those pieces that peer A is interested in them to download at this specific time slot) respectively.

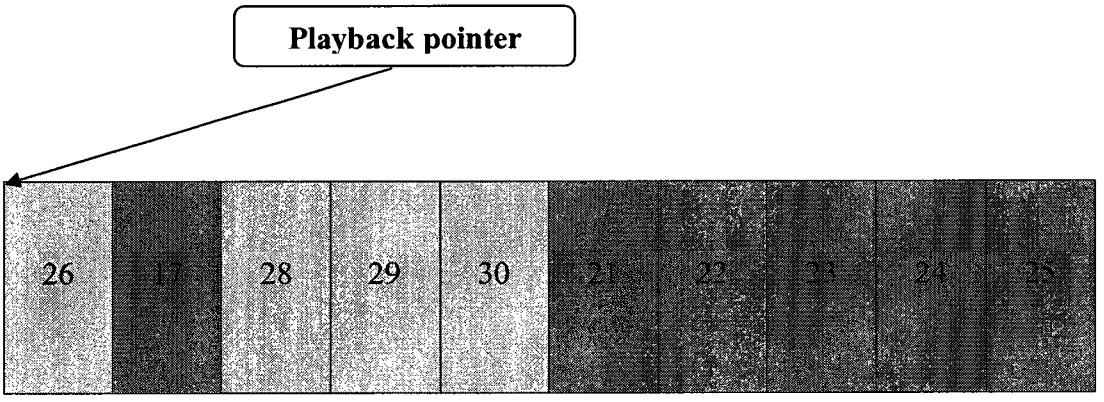


Figure 3.4: Peer A with buffer size L=10, containing old piece #17, old pieces #21 ~ #25, useful piece #26 and useful pieces #28 ~ #30 in its buffer in a specific time slot.

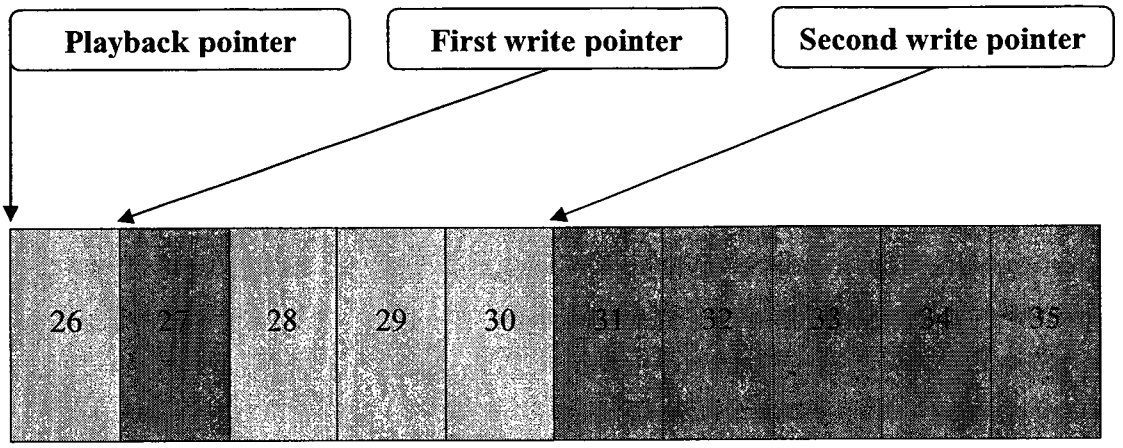


Figure 3.5: Peer A with buffer size L=10 with 4 useful pieces (piece #26 and pieces #28 ~ #30) in its buffer , when it is missing (and looking for) piece #27 and pieces #31 ~ 35 in a specific time slot.

3.2.2 ANALYSIS OF THE MODEL

Before proposing the new model, it is better to explain those events that happen for a randomly picked peer in the network. At first, it is better to understand how a randomly picked peer in a network (e.g. peer A) with a buffer size L ($N>>L$) and play time t_A, downloads its required pieces that have been scheduled to be downloaded in this time slot.

As mentioned before, in this model, the buffer is split into two regions. The first region (high prioritized region) has a high priority compare to the other pieces in the buffer and it only includes one slot at any given time slot. The second region (non-prioritized region), includes the remaining slots (*(L-1)* slots) in that buffer at any given time slot.

The first region of the buffer, includes only one slot in that buffer and depending on the availability of a useful piece in the first slot (piece $\#t_A$) the second slot (piece $\#(t_A + 1)$), it can be region containing the first or the second slot. In other words, if in a specific time slot, the playtime piece (piece $\#t_A$) does not exist, then the high prioritized region will be dedicated to the first slot, and seeks for the piece $\#t_A$ among the neighbours to download it at that time slot. If in a specific time slot, the playtime piece (piece $\#t_A$) exists and the next piece (piece $\#(t_A + 1)$) does not exist, then the first region will be belonged to the second slot, and searches for the piece $\#(t_A + 1)$ to download at this time slot.

The second part of the buffer (non-prioritized region), depending on each of the above situations that could be happened in the first region, includes the remaining *(L-1)* or *(L-2)* other slots in that buffer. It is obvious that in the non-prioritized region, none of the missing pieces, which have been scheduled to be downloaded in this time slot in the second part of the buffer, have priority over the other ones.

With the assumption and explanations that have been described, the stochastic model can be shown as Figure 3.6, in below.

At any time slot, a randomly picked peer that has a finite buffer length of size L will be in one of these $2L$ states, which has been shown in Figure 3.6.

Inside each of the above states (circles), there is a (i_1, i_2) pair, where $0 \leq i_1 \leq 1$ and $0 \leq i_2 \leq (L - 1)$. i_1 represents the availability of the playtime piece (the piece that should be played at that specific time slot); when $i_1 = 1$, it means that piece is available and when $i_1 = 0$, it means that it does not exist in the buffer in this specific time slot.

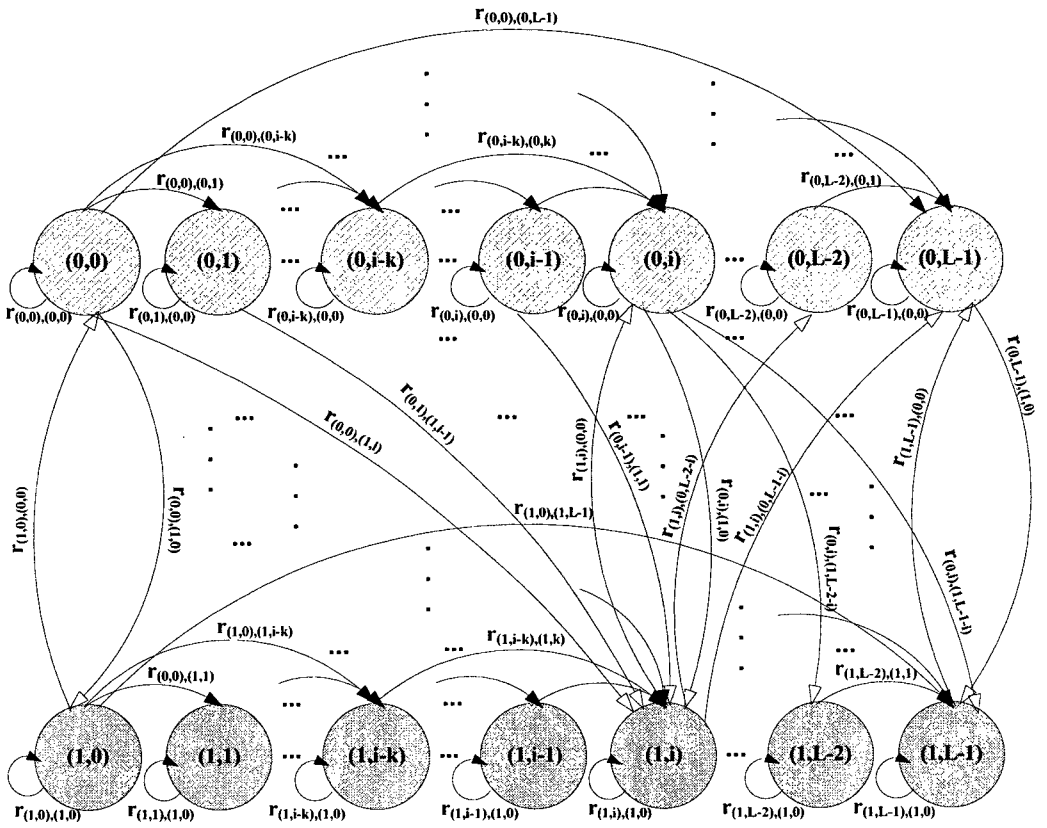


Figure 3.6: A discrete-time stochastic model for a BitTorrent-like P2P live streaming system for peers that have limited buffer length of size L and there is high priority on the first slot.

At any time slot, a randomly picked peer that has a fixed and limited buffer length of size L will be in one of these $2L$ states, which has been shown in Figure 3.6.

Inside each of the above states (circles), there is a (i_1, i_2) pair, where $0 \leq i_1 \leq 1$ and $0 \leq i_2 \leq (L-1)$. i_1 represents the availability of the playtime piece (the piece that should be played at that specific time slot); when $i_1 = 1$, it means that piece is available and when $i_1 = 0$, it means that it does not exist in the buffer in this specific time slot.

In BitTorrent-like systems, whenever a peer receives a piece or some pieces, it will update its recent information with its neighbours and likewise, whenever its neighbours obtain pieces, they will also inform it with their latest information about pieces that they have.

In our proposed BitTorrent-like P2P live streaming model, at the beginning of any time slot, a randomly picked peer in that network will send requests for those pieces of its interest to those neighbours which have those ones. Any peer in these networks acts simultaneously both as a client (downloader) and a server (uploader), therefore that peer would probably receive one or some requests from its neighbours too.

If that peer receives more than one request at any time slot, it will choose (approve) one of the requests and fulfills that request and rejects the other ones. How and based on what condition(s) that peer will choose one of the request and refuses the rest of them is out of scope of this work and could be a part of the future work.

In the next step, we continue our work by analyzing the interactions between two randomly selected peer (e.g. peer A and peer B), in a specific time slot. Before that, in all of our analyses here, we assume that all the peers in this network have fixed and limited buffer length of size L; the play times of peer A and peer B are t_A and t_B respectively; and peer A has i_{1_A} useful piece in its first slot and i_{2_A}useful pieces in its remaining slots, in this time slot, where $0 \leq i_{1_A} \leq 1$ and $0 \leq i_{2_A} \leq (L-1)$. And peer B has i_{1_B}useful piece in its first slot and $i_{2_B} = j$ useful pieces in its remaining slots in this time slot, where

$0 \leq i_{1_B} \leq 1$ and $0 \leq j \leq (L-1)$. For this reason, it can be classified into two main categories, as follows:

❖ **When $i_{1_A} = 0$ and $i_{2_A} = i$, where $0 \leq i \leq (L-1)$:**

➢ In this case, the high priority region will be dedicated to the playtime piece (piece $\#t_A$). Therefore at this time slot, peer A is looking for piece $\#t_A$ with the high priority and simultaneously, it is looking for *(L-1-i)* other useful pieces in its remaining non-prioritized slots.

We define P_{t_A}, as the probability that peer A, which does not have piece $\#t_A$ and is looking for this piece in this time slot, is not interested in peer B.

In order to derive P_{t_A}, we should consider four cases depending on the situation of Peer B's play time (t_B) comparing to the position of the peer A's play time (t_A):

Case 1- When $t_s + 1 \leq t_B \leq t_A - L$

$$P_1' = \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=t_S+1}^{t_A-L} \left(\frac{1}{T^2}\right) \quad (1)$$

Case 2- When $t_A - L + 1 \leq t_B \leq t_A - 1$

$$P_2' = \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min\,(t_S+1,t_A-L+1)}^{t_A-1} \sum_{i_{1_B=0}}^{1} \sum_{j=0}^{L-1} \left(1 - \frac{j}{L-1}\right) * \left(\frac{1}{T^2}\right) * P\left(i_{1_B}, j\right) \quad (2)$$

Case 3- When $t_B = t_A$

$$P_3' = \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=t_A}^{t_A} \sum_{j=0}^{L-1} \left(\frac{1}{T^2}\right) * P(0,j) \tag{3}$$

Case 4- When $t_S + T \geq t_B > t_A$

$$P_4' = \left(\left(\sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min\,(t_S+T,t_A+1)}^{t_S+T} \sum_{j=L-t_B+t_A+1}^{L-1} \left(\frac{1}{T^2}\right) * P(0,j) \right) + \left(\sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min\,(t_S+T,t_A+1)}^{t_S+T} \sum_{j=L-t_B+t_A}^{L-1} \left(\frac{1}{T^2}\right) * P(1,j) \right) \right) \tag{4}$$

Therefore, P_{t_A} can be expressed as follows:

$$P_{t_A} = P_1' + P_2' + P_3' + P_4', \tag{5}$$

Next, we define this probability $P_{L-1_{(i)}}$, as the probability that peer A, which has $i_{2_A} = i$ useful pieces, where $0 \leq i \leq (L-1)$, in its non-prioritized buffer region of size $(L-1)$, is not interested in peer B. We also assume that peer B has i_{1_B} useful piece in its first slot, where $0 \leq i_{1_B} \leq 1$, and $i_{2_B} = j$ useful pieces in the rest of its buffer, where $0 \leq j \leq (L-1)$.

In order to derive $P_{L-1_{(i)}}$, we should consider four cases depending on the situation of Peer B's play time (t_B) comparing to the position of the next peer A's play time ($t_A + 1$):

Case 1- When $t_s + 1 \leq t_B \leq t_A - L + 1$

$$P_{L-1_{(i)}}{}'_{\mathbf{1}} = \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=t_S+1}^{t_A-L+1} \left(\frac{1}{T^2}\right) \tag{6}$$

Case 2- When $t_A - L + 2 \leq t_B \leq t_A$

$$P_{L-1_{(i)}}{}'_{\mathbf{2}}$$

$$= \sum_{j=0}^{l-1} \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=Max(t_S+1,t_A-L+2)}^{min(t_A,t_S+T)} \sum_{x=i-Min\,(t_A-t_B,i)}^{Min(L-1-(t_A-t_B+1),i)} \frac{\binom{L-1-(t_A-t_B)}{x} * \binom{t_A-t_B}{i-x}}{\binom{L-1}{i}}$$

$$* \frac{\binom{t_A-t_B+x}{j}}{\binom{L-1}{j}} * (P(0,j) + P(1,j))$$

$$* \left(\frac{1}{T^2}\right) \tag{7}$$

Case 3- When $t_A + 1 \leq t_B \leq t_A + L - 1$

$$P_{L-1_{(i)}}{}'_3$$

$$= \sum_{j=0}^{l-1} \Bigg(\sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min(t_S+T,t_A+1)}^{min(t_A+L-1,t_S+T)} \sum_{y=i-min(t_B-(t_A+1),i)}^{min(t_B-(t_A+1),i-min(t_B-(t_A+1),L-j))} \left(\frac{i}{L-1}\right)$$

$$* \frac{\binom{t_B-(t_A+1)}{i-y-1} * \binom{L-1-(t_B-t_A)}{y}}{\binom{L-2}{i-1} * \binom{L-1}{j}} * \frac{\binom{t_B-t_A+y}{j} * \binom{t_B-(t_A+1)-min(t_B-(t_A+1),L-j)}{i-y-1-min(t_B-(t_A+1),L-j)}}{\binom{t_B-(t_A+1)}{i-y-1}}$$

$$* \left(\frac{1}{T^2}\right) * P(0,j)$$

$$+ \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min(t_S+T,t_A+1)}^{min(t_A+L-1,t_S+T)} \sum_{y=i-min(t_B-(t_A+1),i)}^{min(t_B-(t_A+1),i-min(t_B-(t_A+1),L-j))} \left(1-\frac{i}{L-1}\right)$$

$$* \frac{\binom{t_B-(t_A+1)}{i-y} * \binom{L-1-(t_B-t_A)}{y}}{\binom{L-2}{i-1} * \binom{L-1}{j}} * \frac{\binom{t_B-t_A+y}{j} * \binom{t_B-(t_A+1)-min(t_B-(t_A+1),L-j)}{i-y-min(t_B-(t_A+1),L-j)}}{\binom{t_B-(t_A+1)}{i-y-1}}$$

$$* \left(\frac{1}{T^2}\right) * P(0,j)$$

$$+ \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min(t_S+T,t_A+1)}^{min(t_A+L-1,t_S+T)} \sum_{y=i-min(t_B-(t_A+1),i)}^{min(t_B-(t_A+1),i-min(t_B-(t_A+1),L-j-1))} \left(\frac{i}{L-1}\right)$$

$$* \frac{\binom{t_B-(t_A+1)}{i-y-1} * \binom{L-1-(t_B-t_A)}{y}}{\binom{L-2}{i-1} * \binom{L-1}{j}} * \frac{\binom{t_B-t_A+y}{j} * \binom{t_B-(t_A+1)-min(t_B-(t_A+1),L-j-1)}{i-y-1-min(t_B-(t_A+1),L-j-1)}}{\binom{t_B-(t_A+1)}{i-y-1}}$$

$$* \left(\frac{1}{T^2}\right)$$

$$* P(1,j)) \qquad (8)$$

Case 4- When $t_B \geq t_A + L$

$$P_{L-1_{(i)}}{}'_{\mathbf{4}}$$

$$= \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min\,(t_A+L,t_S+T)}^{t_S+T} \sum_{j=0}^{L-1} \sum_{i_{1_B}=0}^{1} \frac{\binom{L-1-t_A+L-(t_B-min\,(t_B-(t_A+1),L-i_{1_B}-j))}{i-t_A+L-(t_B-min\,(t_B-(t_A+1),L-i_{1_B}-j))}}{\binom{L-1}{i}}$$

$$* \left(\frac{1}{T^2}\right) * P(i_{1_B}, j) \qquad (9)$$

Therefore, $P_{L-1_{(i)}}$ can be expressed as follows:

$$P_{L-1_{(i)}} = P_{L-1_{(i)}}{}'_{\mathbf{1}} + P_{L-1_{(i)}}{}'_{\mathbf{2}} + P_{L-1_{(i)}}{}'_{\mathbf{3}} + P_{L-1_{(i)}}{}'_{\mathbf{4}}, \qquad (10)$$

Where $0 \leq i \leq (L-1)$.

- **When $i_{1_A} = 1$ (piece $\#t_A$ exists in the buffer) , but piece $\#(\mathbf{t_A + 1})$ is not available in the buffer and therefore, $i_{2_A} = i$, where $0 \leq i \leq (L-2)$:**
  - In this case, as we explained it earlier, the high priority region will be dedicated to the piece $\#(t_A + 1)$, Therefore at this time slot, peer A is looking for piece $\#(t_A + 1)$ with the high priority and simultaneously, it is looking for *(L-1-i)* other useful pieces in its remaining non-prioritized slots.

    We define $P_{t_{A+1}}$, as the probability that peer A has piece $\#t_A$ and does not have piece $\#(t_A + 1)$, and therefore, it is looking for this piece in this time slot, would not be interested in peer B.

In order to derive $P_{t_{A+1}}$, we should consider four cases again, depending on the situation of Peer B's play time (t_B) comparing to the position of the next peer A's play time ($t_A + 1$):

Case 1- When $t_s + 1 \leq t_B \leq (t_A - L + 1)$

$$P_1'' = \sum_{t_A=t_s+1}^{t_s+T} \sum_{t_B=t_s+1}^{t_A-L+1} \left(\frac{1}{T^2}\right) \tag{11}$$

Case 2- When $t_A - L + 2 \leq t_B \leq t_A$

$$P_2'' = \sum_{t_A=t_s+1}^{t_s+T} \sum_{t_B=min\,(t_s+1,t_A-L+2)}^{t_A} \sum_{i_{1_B}=0}^{1} \sum_{j=0}^{L-1} \left(1 - \frac{j}{L-1}\right) * \left(\frac{1}{T^2}\right) * P\left(i_{1_B} j\right) \tag{12}$$

Case 3- When $t_B = t_A + 1$

$$P_3'' = \sum_{t_A=t_s+1}^{t_s+T} \sum_{t_B=t_A+1}^{t_A+1} \sum_{j=0}^{L-1} \left(\frac{1}{T^2}\right) * P(0,j) \tag{13}$$

Case 4- When $t_s + T \geq t_B > t_A + 1$

$$P_4'' = \left(\left(\sum_{t_A=t_s+1}^{t_s+T} \sum_{t_B=min\,(t_s+T,t_A+2)}^{t_s+T} \sum_{j=L-t_B+t_A+2}^{L-1} \left(\frac{1}{T^2}\right) * P(0,j) \right) + \left(\sum_{t_A=t_s+1}^{t_s+T} \sum_{t_B=min\,(t_s+T,t_A+2)}^{t_s+T} \sum_{j=L-t_B+t_A+1}^{L-1} \left(\frac{1}{T^2}\right) * P(1,j) \right) \right) \quad (14)$$

Therefore, $P_{t_{A+1}}$ can be expressed as follows:

$$P_{t_{A+1}} = P_1'' + P_2'' + P_3'' + P_4'' \quad (15)$$

Next, we define this probability $P_{L-2_{(i)}}$, as the probability that peer A, which has $i_{2_A} = i$ useful pieces, where *0* $\leq i \leq$ *(L-2)*, in its non-prioritized buffer region of size *L-1*, is not interested in peer B. We also assume that peer B has i_{1_B} useful piece in its first slot, where $0 \leq i_{1_B} \leq 1$, and $i_{2_B} = j$ useful pieces in the rest of its buffer, where $0 \leq j \leq (L-1)$. In order to derive $P_{L-2_{(i)}}$, we should consider four cases depending on the situation of Peer B's play time (t_B) comparing to the position of the next peer A's play time ($t_A + 1$):

Case 1- When $t_s + 1 \leq t_B \leq t_A - L + 2$

$$P_{L-2_{(i)}}{}'_{\mathbf{1}} = \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=t_S+1}^{t_A-L+2} \left(\frac{1}{T^2}\right) \tag{16}$$

Case 2- When $t_A - L + 3 \leq t_B \leq t_A + 1$

$$P_{L-2_{(i)}}{}'_{\mathbf{2}}$$

$$= \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=Max(t_S+1,t_A-L+3)}^{min(t_A+1,t_S+T)} \sum_{x=i-Min\,(t_A-t_B+1,i)}^{Min(L-2-(t_A-t_B),i)} \frac{\binom{L-2-(t_A-t_B)}{x} * \binom{t_A-t_B}{i-x}}{\binom{L-2}{i}}$$

$$* \frac{\binom{t_A-t_B+1+x}{j}}{\binom{L-1}{j}} * (P(0,j) + P(1,j)) * \left(\frac{1}{T^2}\right) \tag{17}$$

Case 3- When $t_A + 2 \leq t_B \leq t_A + L - 1$

$$P_{L-2_{(i)}}{}'_{\mathbf{3}}$$

$$= \sum_{j=0}^{l-1} \Bigg(\sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min(t_S+T,t_A+2)}^{min(t_A+L-1,t_S+T)} \sum_{y=i-min(t_B-(t_A+2),i)}^{min(L-1-(t_B-t_A),i-min(t_B-(t_A+2),L-j))} \left(\frac{i}{L-2}\right)$$

$$* \frac{\binom{t_B-(t_A+2)}{i-y-1} * \binom{L-1-(t_B-t_A)}{y}}{\binom{L-3}{i-1} * \binom{L-1}{j}} * \frac{\binom{t_B-t_A+y}{j} * \binom{t_B-(t_A+2)-min(t_B-(t_A+2),L-j)}{i-y-1-min(t_B-(t_A+2),L-j)}}{\binom{t_B-(t_A+2)}{i-y-1}}$$

$$* \left(\frac{1}{T^2}\right) * P(0,j)$$

$$+ \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min(t_S+T,t_A+2)}^{min(t_A+L-1,t_S+T)} \sum_{y=i-min(t_B-(t_A+2),i)}^{min(L-1-(t_B-t_A),i-min(t_B-(t_A+2),L-j))} \left(1-\frac{i}{L-2}\right)$$

$$* \frac{\binom{t_B-(t_A+2)}{i-y} * \binom{L-1-(t_B-t_A)}{y}}{\binom{L-3}{i-1} * \binom{L-1}{j}} * \frac{\binom{t_B-t_A+y}{j} * \binom{t_B-(t_A+2)-min(t_B-(t_A+2),L-j)}{i-y-min(t_B-(t_A+2),L-j)}}{\binom{t_B-(t_A+2)}{i-y}}$$

$$* \left(\frac{1}{T^2}\right) * P(0,j)$$

$$+ \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min(t_S+T,t_A+2)}^{min(t_A+L-1,t_S+T)} \sum_{y=i-min(t_B-(t_A+2),i)}^{min(L-1-(t_B-t_A),i-min(t_B-(t_A+2),L-j))} \left(\frac{i}{L-2}\right)$$

$$* \frac{\binom{t_B-(t_A+2)}{i-y-1} * \binom{L-1-(t_B-t_A)}{y}}{\binom{L-3}{i-1} * \binom{L-1}{j}} * \frac{\binom{t_B-t_A+y}{j} * \binom{t_B-(t_A+2)-min(t_B-(t_A+2),L-j-1)}{i-y-1-min(t_B-(t_A+2),L-j-1)}}{\binom{t_B-(t_A+2)}{i-y-1}}$$

$$* \left(\frac{1}{T^2}\right)$$

$$* P(1,j)) \qquad (18)$$

Case 4- When $t_B \geq t_A + L$

$$P_{L-2_{(i)}}{}'_{4}$$

$$= \sum_{t_A=t_S+1}^{t_S+T} \sum_{t_B=min\,(t_A+L,t_S+T)}^{t_S+T} \sum_{j=0}^{L-1} \sum_{i_{1_B}=0}^{1} \frac{\binom{L-2-t_A+L-(t_B-min\,(t_B-(t_A+1),L-i_{1_B}-j))}{i-t_A+L-(t_B-min\,(t_B-(t_A+1),L-i_{1_B}-j))}}{\binom{L-2}{i}}$$

$$* \left(\frac{1}{T^2}\right) * P(i_{1_B}, j) \tag{19}$$

Therefore, $P_{L-2_{(i)}}$ can be expressed as follows:

$$P_{L-2_{(i)}} = P_{L-2_{(i)}}{}'_{\mathbf{1}} + P_{L-2_{(i)}}{}'_{\mathbf{2}} + P_{L-2_{(i)}}{}'_{\mathbf{3}} + P_{L-2_{(i)}}{}'_{\mathbf{4}}, \tag{20}$$

Where $0 \leq i \leq (L-2)$.

Now we define, $U_{\alpha_{i_{w_L}}}$ as the probability that a randomly selected peer with the buffer length of size L, that has i useful pieces in its non-prioritized buffer part of size w (and it is looking for downloading $j=(w-i)$ other useful pieces in that specific time slot), is interested in a randomly picked neighbour in this time slot. In this work, in order to find this probability, we have to consider two different conditions, depending on the size of the non-prioritized buffer ($w=(L-1)$ or $w=(L-2)$) of a randomly selected peer (e.g. peer A), where the total length of the buffer is L.

❖ **When is *w=L-1***

➢ In this case, by using Eq. (10), $U_{\alpha i_{L-1_L}}$ can be expressed as follows:

$$U_{\alpha i_{L-1_L}} = \left(1 - P_{L-1_{(i)}}\right) \tag{21}$$

❖ **When is *w=L-2***

➢ In this case, by using Eq. (20), $U_{\alpha i_{L-2_L}}$ can be expressed as follows:

$$U_{\alpha i_{L-2_L}} = \left(1 - P_{L-2_{(i)}}\right) \tag{22}$$

Next, we define *F(V, j, k ,w)*, as the probability which a randomly selected peer ,which has a total buffer length of size L and a non-prioritized buffer of size w and, is looking for j useful pieces for its non-prioritized buffer area in a specific time slot and sends k requests to its neighbours of size *V* in this time slot, where $L \geq w$, $0 \leq j \leq w$ and $0 \leq k \leq$ *min (V,j)*.

F (m, j, k, w) would be defined as a recursive function and it can be calculated by using the following equation:

$$F(m,j,k,w) = U_{\alpha w-j_{w_L}} * F(m-1, j-1, k-1, w) + (1 - U_{\alpha w-j_{w_L}}) * F(m,j,k,L), \tag{23}$$

Where $V \geq m \geq 0$, $j=0, 1, \ldots, L$ and $k=0, \ldots, min(m, j)$.

In order to find *F(V, j, k, w)*, we also need an initial condition, which is expressed in below:

$$F(m, j, k, w) = 1 \quad ; \quad m=0, k=0, j=0. \tag{24}$$

It can be seen that now, we are able to derive *F (V, j, k, w)* by solving the following V equations, as they are expressed in below:

Equation 1) $F(0, j, k, w) = 0$

Equation 2) $F(1, j, k, w) = U_{\alpha_{w-j_{w_L}}} * F(0, j-1, k-1, w) + (1 - U_{\alpha_{w-j_{w_L}}}) * (1-F(0, j, k, w)$

Equation 3) $F(2, j, k, w) = U_{\alpha_{w-j_{w_L}}} * F(1, j-1, k-1, w) + (1 - U_{\alpha_{w-j_{w_L}}}) * (F(1, j, k, w)$

.

.

.

Equation V-1) $F(V\text{-}1, j, k, w) = U_{\alpha_{w-j_{w_L}}} * F(V\text{-}2, j-1, k-1, w) + \left(1 - U_{\alpha_{w-j_{w_L}}}\right) * F(V\text{-}2, j, k, w)$

Equation V) $F(V, j, k, w) = U_{\alpha_{w-j_{w_L}}} * F(V-1, j-1, k-1, w) + \left(1 - U_{\alpha_{w-j_{w_L}}}\right) *$

$F(V-1, j, k, w)$ (25)

In our analysis for this chapter, we could encounter different buffer lengths and different values for the number of neighbours in a random time slot for a peer. Therefore, we categorize them into four different classes depending on the values of w and V, in below:

- **When $V=H$ and $w=L-1$**
  - In this case, we want to find F (H, j, k, L-1). And by employing Eq. (21) , (24) and (25), it can be calculated, as it is expressed in below:

    $F(H, j, k, L-1) = (U_{\alpha_{i_{L-1_L}}}) * F(H-1, j-1, k-1, L-1) + \left(1 - U_{\alpha_{i_{L-1_L}}}\right) * F(H-1, j, k, L-1),$ (26)

    Where, $0 \leq j \leq L-1$ and $k = 0, 1, \ldots, min(H,j)$

- **When $V=H-1$ and $w=L-1$**
  - In this case, we want to find F (H, j, k, L-1). And by employing Eq. (21) , (24) and (25), it will be calculated, as it is expressed in below:

    $F(H-1, j, k, L-1) = (U_{\alpha_{i_{L-1_L}}}) * F(H-2, j-1, k-1, L-1) + \left(1 - U_{\alpha_{i_{L-1_L}}}\right) * F(H-2, j, k, L-1),$ (27)

    Where, $0 \leq j \leq L-1$ and $k = 0, 1, \ldots, min(H-1,j)$

- **When $V=H$ and $w=L-2$**

- In this case, we want to find *F (H, j, k, L-2)*; and by employing Eq. (22), (24) and (25), it will be calculated, as it is expressed in below:

$$F(H, j, k, L-2) = (U_{\alpha_{i_{L-2_L}}}) * F(H-1, j-1,\ k-1,\ L-2) + \left(1 - U_{\alpha_{i_{L-2_L}}}\right) * F(H-1, j,\ k,\ L-2), \quad (28)$$

Where, $0 \leq j \leq L-2$ and $k = 0, 1, \ldots, min\ (H,j)$

❖ **When *V=H-1* and *w=L-2***

- In this case, we want to find *F (H-1, j, k, L-2)*; and by employing Eq. (22), (24) and (25), it will be calculated, as it is expressed in below:

$$F(H-1, j,\ k,\ L-2) = (U_{\alpha_{i_{L-2_L}}}) * F(H-2, j-1,\ k-1,\ L-2) + \left(1 - U_{\alpha_{i_{L-2_L}}}\right) * F(H-2, j,\ k,\ L-2), \quad (29)$$

Where, $0 \leq j \leq L-2$ and $k = 0, 1, \ldots, min\ (H-1,j)$

Next, we can introduce $E[K]$ as the average number of requests that a peer sends to its neighbours in a given time slot. In order to derive it, we assume that in a specific time slot, we have a randomly picked peer (e.g. peer A) with the buffer length of size L, which has i_0 piece in its high prioritized buffer region (e.g. piece $\#t_A$) and has i_1 useful pieces in its non-prioritized buffer part and is looking for $j=(L-1-i_1)$ other useful pieces for this area, in this time slot, where $0 \leq i_1 \leq (L-1)$ and $0 \leq i_0 \leq 1$; then we divide our calculations here into two parts. In the first one, peer A does not have a useful piece in its high-prioritized slot; and in the second one, it has a useful piece in it, as they are explained in below:

❖ **When peer A does not have the piece $\#t_A$ in its buffer ($i_0 = 0$).**

➢ In this case, we define $E[K]_0$, as the number of requests that peer A send in any given time slot on average, when it does not have a useful piece in its high prioritized slot. In this case, depending on the availability of this high prioritized piece among the peer A's neighbours, one of the following two possible events could be happened:

- **When none of peer A's neighbours have the piece $\#t_A$.**

  In this case, we define $E[K]_{0_0}$ the average number of requests that peer A sends in any given time slot will be expressed as follows:

$$E[K]_{0_0} = \left[\sum_{i_1=0}^{L-1} \sum_{k_1=0}^{min(H,L-1-i_1)} \begin{matrix} k_1 * P(0,i_1) * \\ \left(P_{t_A}{}^{H}\right) * F\,(H, L-1-i_1, k_1, L-1) \end{matrix}\right] \quad (30)$$

- **When at least of peer A's neighbours has the piece $\#t_A$ in this time slot.**

  In this case, we define $E[K]_{0_1}$ the average number of requests that peer A sends in any given time slot will be expressed as follows:

$$E[K]_{0_1} = \left[\sum_{i_1=0}^{L-1} \sum_{k_1=0}^{min(H-1,L-1-i_1)} \begin{matrix} (k_1+1) * P(0,i_1) * \\ * \left(1 - P_{t_A}{}^{H}\right) \\ * F\,(H-1, L-1-i_1, k_1, L-1) \end{matrix}\right] \quad (31)$$

Therefore, $E[K]_0$ would be described as the sum of Eq. (27) and (28) ($E[K]_0 = E[K]_{0_0} + E[K]_{0_1}$). As we already mentioned, it will lead us to the average number of

requests that is sent by a randomly selected peer in the network, when it does not have a useful piece in its high prioritized buffer. And it will be expressed as follows:

$$E[K]_0 = \left[\sum_{i_1=0}^{L-1} \sum_{k_1=0}^{min(H,L-1-i_1)} \begin{matrix} k_1 * P(0,i_1) * \\ \left(P_{t_A}{}^H\right) * F\,(H, L-1-i_1, k_1, L-1) \end{matrix}\right]$$

$$+ \left[\sum_{i_1=0}^{L-1} \sum_{k_1=0}^{min(H-1,L-1-i_1)} \begin{matrix} (k_1+1) * P(0,i_1) * \\ * \left(1 - P_{t_A}{}^H\right) \\ * F\,(H-1, L-1-i_1, k_1, L-1) \end{matrix}\right] \quad (32)$$

❖ **When peer A has the piece $\#t_A$ in its buffer ($i_0 = 1$).**

➢ In this case, we define $E[K]_1$, as the number of requests that peer A send in any given time slot on average, when it has a useful piece in its high prioritized slot (piece $\#t_A$ is available in its buffer).

Depending on the availability of the next piece (piece $\#(t_A + 1)$) in peer A's buffer and also peer A's neighbours, one of the following three possible events could be happened:

▪ **When peer A does not have the piece $\#(t_A + 1)$ in its buffer at the beginning of this time slot and none of its neighbours have this piece in their buffers either.**

In this case, we define $E[K]_{1_0}$ the average number of requests that peer A sends in any given time slot will be expressed as follows:

$$E[K]_{1_0} = \left[\sum_{i_1=0}^{L-2} \sum_{k_1=0}^{min(H,L-1-i_1)} k_1 * \left(1 - \frac{i_1}{L-1}\right) * P(1,i_1) * (P_{t_{A+1}}{}^{H}) * F\,(H, L-1-i_1, k_1, L-2)\right] \quad (33)$$

- **When peer A does not have the piece $\#(t_A + 1)$ in its buffer at the beginning of this time slot and at least one of its neighbours has this piece in its buffer.**

  In this case, we define $E[K]_{1_1}$ the average number of requests that peer A sends in any given time slot will be expressed as follows:

$$E[K]_{1_1} = \left[\sum_{i_1=0}^{L-2} \sum_{k_1=0}^{min(H-1,L-1-i_1)} (1+k_1) * \left(1 - \frac{i_1}{L-1}\right) * P(1,i_1) * F\,(H-1, L-1-i_1, k_1, L-2) * (1 - P_{t_{A+1}}{}^{H})\right] \quad (34)$$

- **When peer A has piece $\#(t_A + 1)$ in its buffer at the beginning of this time slot.**

  In this case, we define $E[K]_{1_2}$ the average number of requests that peer A sends in any given time slot will be expressed as follows:

$$E[K]_{1_2} = \left[\sum_{i_1=0}^{L-2} \sum_{k_1=0}^{min(H,L-1-i_1)} (k_1) * \left(\frac{i_1}{L-1}\right) * P(1,i_1) * F\,(H, L-1-i_1, k_1, L-2)\right] \quad (35)$$

Therefore, $E[K]_1$ would be described as the sum of Eq. (33), (34) and (35) ($E[K]_1 = E[K]_{1_0} + E[K]_{1_1} + E[K]_{1_2}$). This will lead us to the average number of requests that is

sent by a randomly selected peer in the network, when it has a useful piece in its high prioritized buffer. This will be expressed as follows:

$$E[K]_1 = \left[\sum_{i_1=0}^{L-2} \sum_{k_1=0}^{min(H,L-1-i_1)} k_1 * \left(1 - \frac{i_1}{L-1}\right) * P(1,i_1) * (P_{t_{A+1}}{}^{H}) * F\,(H, L-1-i_1, k_1, L-2)\right]$$

$$+ \left[\sum_{i_1=0}^{L-2} \sum_{k_1=0}^{min(H-1,L-1-i_1)} (1+k_1) * \left(1 - \frac{i_1}{L-1}\right) * P(1,i_1) * F\,(H-1, L-1-i_1, k_1, L-2) * (1 - P_{t_{A+1}}{}^{H})\right]$$

$$+ \left[\sum_{i_1=0}^{L-2} \sum_{k_1=0}^{min(H,L-1-i_1)} (k_1) * \left(\frac{i_1}{L-1}\right) * P(1,i_1) * F\,(H, L-1-i_1, k_1, L-2)\right] \quad (36)$$

Finally, by integrating Eq. (32) and (36) together, we can reach to the number of requests that a randomly picked peer in the network sends in any given time slot on average. This can be expressed as follows:

$$E[K] = E[K]_0 + E[K]_1 \quad (37)$$

Next, we assume that we have two randomly selected neighbours, namely, peers A and B. Then we introduce $\bar{X}$, as the average number of requests that peer B receives in any given time slot beside peer A's request, where each peer in the network has H neighbours. Therefore, by using Eq. (37), $\bar{X}$ can be expressed as follows:

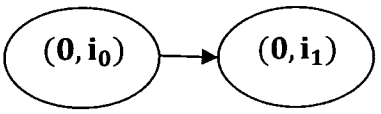
$$\bar{X} = \frac{(H-1)*E[K]}{H} \tag{38}$$

Now, we introduce Q, as the probability that a randomly selected peer (e.g. peer B) fulfils a specific request (e.g. peer A's request) among all other requests that it receives from other peers in a specific time slot, where each peer in the network has H neighbours. Therefore, by using Eq. (38), Q can be expressed as follows:

$$Q = \frac{1}{1+\bar{X}} = \frac{1}{\left(1+\frac{(H-1)*E[K]}{H}\right)} \tag{39}$$

Next, we continue our work by analyzing the possible jumps that a randomly selected peer (e.g. peer A), where play time of peer A is t_A , neighbour size is H and the buffer size is L, could have in a specific time slot. For this reason, the possible jumps can be classified into four different classes, as follow:

- **When peer A Jumps from state $(0, i_0)$ to state $(0, i_1)$, where $L-1 \geq i_1 \geq i_0 \geq 0$**



  - In this situation, peer A can have this jump if one of these two possible ways happens for it:

    1- **When none of peer A's neighbours have piece $\#t_A$ and peer A obtains $(i_1 - i_0)$ pieces from its neighbours (H neighbours) for its non-prioritized buffer part of size (L-1).**

We name this probability as $r_{((0,i_0),(0,i_1))_1}$, and it can be expressed as follows:

$r_{((0,i_0),(0,i_1))_1}$ = *Probability [Peer A's neighbours do not have piece* $\#t_A$*]* $*$ *Probability [Peer A downloads* $(i_1 - i_0)$ *pieces from its H neighbours for its non-prioritized buffer area of size (L-1) in this time slot]* (40)

Or, it can be expressed as follows:

$$r_{((0,i_0),(0,i_1))_1} = (P_{t_A})^H * \sum_{k=(i_1-i_0)}^{min(H,i_1-i_0)} \begin{bmatrix} F(H, i_1 - i_0, k, L-1) \\ * \begin{bmatrix} k \\ i_1 - i_0 \end{bmatrix} \\ * (Q)^{k-(i_1-i_0)} \\ * (1-Q)^{i_1-i_0} \end{bmatrix} \quad (41)$$

2- **When at least one of peer A's neighbours has piece** $\#t_A$ **but peer A's request for is piece gets rejected and besides that, peer A obtains** $(i_1 - i_0)$ **pieces from the rest of its neighbours ((*H-1*) other neighbours) for its non-prioritized buffer part of size (*L-1*).**

We name this probability as $r_{((0,i_0),(0,i_1))_2}$, and it can be expressed as follows:

$r_{((0,i_0),(0,i_1))_2}$ =*Probability [At least one of peer A's neighbours has piece* $\#t_A$*]* $*$ *Probability[Peer A's request for piece* $\#t_A$ *gets rejected]* $*$ *Probability[Peer A downloads* $(i_1 - i_0)$ *pieces from the remaining (H-1) other neighbours for its non-prioritized buffer area of size (L-1) in this time slot]* (42)

Where Probability [Peer A's request for piece $\#t_A$ gets rejected] =(*1-Q*)

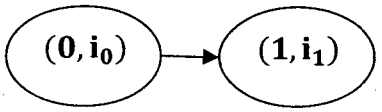
Or, it can be expressed as follows:

$$r_{((0,i_0),(0;i_1))_2} = \sum_{k=(i_1-i_0)}^{min(H,i_1-i_0)} \left[\begin{bmatrix} F(H, i_1 - i_0, k, L-1) * \\ \begin{bmatrix} k \\ i_1 - i_0 \end{bmatrix} * \\ (Q)^{k-(i_1-i_0)} * \end{bmatrix} \begin{bmatrix} (1-Q)^{i_1-i_0} * \left(1 - {P_{t_A}}^H\right) * (1-Q) \end{bmatrix} \right] \quad (43)$$

Therefore, the jumping probability, defined as $r_{((0,i_0),(0,i_1))}$, would be the sum of $r_{((0,i_0),(0,i_1))_0}$ and $r_{((0,i_0),(0,i_1))_1}$, Eq. (41) and (43), and it is expressed in below:

$$r_{((0,i_0),(0,i_1))} = r_{((0,i_0),(0,i_1))_1} + r_{((0,i_0),(0,i_1))_2} \quad (44)$$

❖ **When peer A Jumps from state $(0, i_0)$ to state $(1, i_1)$, where $L - 1 \geq i_1 \geq i_0 \geq 0$**



➢ In this situation, peer A can have this jump if it downloads piece $\#t_A$ from one of its neighbours; and at the same time, it also downloads $(i_1 - i_0)$ other pieces for its non-prioritized buffer area of size (*L-1*) in this time slot.

Therefore, the jumping probability, defined as $r_{((0,i_0),(1,i_1))}$, can be expressed by:

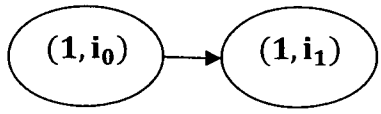
$r_{((0,i_0),(1,i_1))}$ = *Probability [at least one of peer A's neighbours has piece* $\#t_A$ * and Peer A obtains piece* $\#t_A$ *from one of them]* * *Probability [Peer A downloads* $(i_1 - i_0)$ *pieces from the rest of its neighbours ((H-1) other neighbours) for its*

non-prioritized buffer area of size (L-1) in this time slot]. (45)

Or, it can be expressed as follows:

$$r_{((0,i_0),(1,i_1))} = \sum_{k=(i_1-i_0)}^{min(H-1,i_1-i_0)} \begin{bmatrix} F(H-1,i_1-i_0,k,L-1) * \\ \begin{bmatrix} k \\ i_1-i_0 \end{bmatrix} * \\ (Q)^{k-(i_1-i_0)} * \\ (1-Q)^{i_1-i_0} * \left(1-{P_{t_A}}^H\right) * (Q) \end{bmatrix} \quad (46)$$

❖ **When peer A Jumps from state $(1, i_0)$ to state $(1, i_1)$, where $L-1 \geq i_1 \geq i_0 \geq 0$**



➢ In this situation, peer A can have this jump if one of these two possible ways happens for it:

1- When peer A that has piece $\#t_A$ and also piece $\#(t_A+1)$, with the probability of $(\frac{i_0}{L-1})$, and there are i_0 pieces in peer A's non-prioritized buffer of size $(L-1)$; and Peer A obtains (i_1-i_0) pieces from its neighbours (H neighbours) for its non-prioritized buffer part of size $(L-2)$.

As discussed before, those pieces that a randomly picked peer in the system has in its non-prioritized buffer area, they are uniformly distributed.

Therefore, for example in this case, when peer A has i_0 pieces in its non-prioritized buffer part, the probability that peer A could have the first piece (piece $\#(t_A+1)$) in that part of its buffer would be $(\frac{i_0}{L-1})$.

We name the probability of this part as $r_{((1,i_0),(1,i_1))_1}$, and it can be expressed as follows:

$r_{((1,i_0),(1,i_1))_1}$ = *Probability [Peer A has piece* $\#(t_A+1)$ *in its buffer at the beginning of this time slot]* * *Probability [Peer A downloads* (i_1-i_0) *pieces from its H neighbours for the rest of its non-prioritized buffer area of size* $(L-2)$ *in this time slot].* (47)

Or, it can be expressed as follows:

$$r_{((1,i_0),(1,i_1))_1} = \left(\frac{i_0}{L-1}\right) * \sum_{k=(i_1-i_0)}^{min(H,i_1-i_0)} \begin{bmatrix} F(H,i_1-i_0,k,L-2) * \\ \begin{bmatrix} k \\ i_1-i_0 \end{bmatrix} * \\ (Q)^{k-(i_1-i_0)} * \\ (1-Q)^{i_1-i_0} \end{bmatrix} \quad (48)$$

2- **When peer A that has piece $\#t_A$, does not have $\#(t_A+1)$, with the probability of $(1-\frac{i_0}{L-1})$, when it has i_0 pieces in its non-prioritized buffer of size (*L-1*) ; and it obtains piece $\#(t_A+1)$ in this time slot and (i_1-i_0-1) pieces from the rest of its neighbours ((*H-1*) other neighbours) for its non-prioritized buffer part of size (*L-2*).**

Therefore, for example in this case, when peer A has i_0 pieces in its non-prioritized buffer part, the probability that peer A could not have the first piece (piece $\#(t_A+1)$) in this part of its buffer would be $(1-\frac{i_0}{L-1})$.

We name this probability as $r_{((1,i_0),(1,i_1))_2}$. It can be expressed as follows:

$r_{((1,i_0),(1,i_1))_2}$ = *Probability [Peer A, which does not have piece* $\#(t_A+1)$*, downloads piece* $\#(t_A+1)$ *in this time slot, given at least one of its neighbours has this piece]* $*$ *Probability [Peer A downloads* (i_1-i_0-1) *pieces from the rest of its neighbours ((H-1) other neighbours) for the rest of its non-prioritized buffer area of size (L-2) in this time slot]* (49)

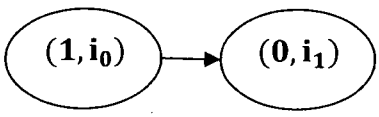
Or, it can be expressed as follows:

$$r_{((1,i_0),(1,i_1))_2} = \sum_{k=(i_1-i_0-1)}^{min(H-1,i_1-i_0-1)} \begin{bmatrix} F(H-1,i_1-i_0-1,k,L-2) * \\ \begin{bmatrix} k \\ i_1-i_0-1 \end{bmatrix} * \\ (Q)^{k-(i_1-i_0-1)} * \\ (1-Q)^{i_1-i_0-1} * \\ \left(1-\frac{i_0}{L-1}\right) * \left(1-(P_{t_{A+1}})^H\right) * Q \end{bmatrix} \quad (50)$$

Therefore, the jumping probability, defined as $r_{((1,i_0),(1,i_1))}$, in this case can be expressed by:

$$r_{((0,i_0),(0,i_1))} = r_{((1,i_0),(1,i_1))_1} + r_{((1,i_0),(1,i_1))_2} \quad (51)$$

❖ **When peer A jumps from state $(1, i_0)$ to state $(0, i_1)$, where $L-1 \geq i_1 \geq i_0 \geq 0$**



➢ In this situation, peer A can have this jump if one of these two possible ways happens for it:

1- **When peer A has already downloaded piece $\#t_A$, but is did not download piece $\#(t_A + 1)$, by the beginning of the current slot time, and it does not obtain this piece in this time slot; because its request for this piece gets refused by one of its neighbours which the request for this specific piece has been sent to it, when this neighbour has this piece in its buffer; and at the same time, peer A obtains $(i_1 - i_0)$ pieces from the rest of its neighbours ((*H-1*) neighbours) for its non-prioritized buffer part of size (*L-2*), when it has i_0 pieces in its non-prioritized buffer of size (*L-1*).**

As we already discussed, those pieces that a randomly picked peer in the system has in its non-prioritized buffer area, they are uniformly distributed.

Hence, for example in this case, when peer A has i_0 pieces in its non-prioritized buffer part, the probability that peer A does not have the piece $\#(t_A + 1)$ in this part of its buffer would be $(1 - \frac{i_0}{L-1})$.

We name the probability of this part as $r_{((1,i_0),(0,i_1))_1}$, and it can be expressed as follows:

$r_{((1,i_0),(o,i_1))_1}$ = *Probability [Peer A does not have piece* $\#(t_A + 1)$ *in its buffer] *Probability [At least of peer A's neighbours has piece* $\#(t_A + 1)$ *and peer A's request for this piece gets declined]* $*$ *Probability [Peer A downloads* $(i_1 - i_0)$ *pieces from the rest of its neighbours ((H-1) neighbours) for the rest of its non-prioritized buffer area of size (L-2)]* [52]

Where Probability [Peer A's request for piece $\#(t_A + 1)$ gets rejected]=(1- Q)

Or it can be expressed as follows:

$$r_{((1,i_0),(0,i_1))_1} = \sum_{k=(i_1-i_0)}^{\min(H-1,i_1-i_0)} \left[\begin{array}{c} F(H-1, i_1-i_0, k, L-2) \\ * \begin{bmatrix} k \\ i_1-i_0 \end{bmatrix} * (Q)^{k-(i_1-i_0)} \\ * ((1-Q)^{i_1-i_0}) * \left(1-\frac{i_0}{L-1}\right) \\ * \left(1-P_{t_{A+1}}{}^{H}\right) * (1-Q) \end{array} \right] \quad (53)$$

2- **When peer A has already downloaded piece $\#t_A$, but it did not download piece $\#(t_A+1)$, by the beginning of this current slot time, and it does not obtain this piece in this time slot (Because none of its neighbours have this piece in their buffers); and concurrently, it obtains (i_1-i_0) pieces from its neighbours (H neighbours) for its non-prioritized buffer part of size (L-2).**

Therefore, for example in this case, when peer A has i_0 pieces in its non-prioritized buffer part, the probability that peer A could not have the first piece (piece $\#(t_A+1)$) in this part of its buffer would be $(1-\frac{i_0}{L-1})$.

We name this probability as $r_{((1,i_0),(0,i_1))_2}$, and it can be expressed as follows:

$r_{((1,i_0),(0,i_1))_2} = \left(1-\frac{i_0}{L-1}\right) * Probability\ [None\ of\ peer\ A's\ neighbours\ have\ piece\ \#(t_A+1)] * [Peer\ A\ downloads\ (i_1-i_0)\ pieces\ from\ its\ neighbours\ (H\ neighbours)\ for\ the\ rest\ of\ its\ non\text{-}prioritized\ buffer\ area\ of\ size\ (L\text{-}2)]$. (54)

Or it can be expressed as follows:

$$r_{((1,i_0),(0,i_1))_2} = \left(\sum_{k=(i_1-i_0)}^{\min(H,i_1-i_0)} \left[\begin{array}{c} F(H, i_1 - i_0, k, L-2) * \\ \binom{k}{i_1 - i_0} * (Q)^{k-(i_1-i_0)} \\ * (1-Q)^{i_1-i_0} * \left(1 - \frac{i_0}{L-1}\right) \\ * \left((P_{t_{A+1}})^H\right) \end{array} \right] \right. \quad (55)$$

Therefore, the jumping probability, defined as $r_{((1,i_0),(1,i_1))}$, in this case can be expressed by:

$$r_{((1,i_0),(0,i_1))} = r_{((1,i_0),(0,i_1))_1} + r_{((1,i_0),(0,i_1))_2} \quad (56)$$

Next, we define $P_{Continuity}$, as the probability that a randomly picked peer in the network that is watching or listening to a certain live streaming content in a specific time slot, would be able to play its desired play time piece at this time slot. If we assume that the size of the buffers for the peers is L, then $P_{Continuity}$ can be expressed as follows:

$$P_{Continuity} = \sum_{i_1=0}^{i_1=L-1} P(1, i_1) \quad (57)$$

Next, we define d_i, as the average download rate of a peer with the total buffer size of L that has i_0 piece in its high prioritized buffer and i_1 useful pieces in its non-prioritized

buffer in a specific time slot, where $0 \leq i_0 \leq 1$ and $0 \leq i_1 \leq L-1$. And it can be shown as follows:

$$d_{(i_0,i_1)} = \sum_{k_1=0}^{min(H-k_0,L-1-i_1)} \sum_{k_0=0}^{min(H,1-i_0)} (k_0 + k_1) * r_{(i_0,i_1),(i_0+k_0,i_1+k_1)}, \quad (58)$$

Where $1 \geq i_0 \geq 0$ and $(L-1) \geq i_1 \geq 0$.

Since the system is in the steady state, the peer distribution does not change during the time and therefore, base on the Figure 3.6, when peers have buffers of size L, we will have the following two equations for a randomly picked peer (e.g. peer A) in the network:

- **When peer A has a useful piece in its high prioritized buffer ($i_0 = 1$) and has $i_1 = i$ useful pieces in its non-prioritized buffer of size L-1, where $0 \leq i \leq L-1$.**
  - In this case, peer A will be in the state of $(1, i_1)$; and therefore, we would have the following equilibrium equation:

$$P_{(1,i)} = (\sum_{k=0}^{i} (r_{(1,k),(1,i)} * P_{(1,k)}) + (\sum_{k=0}^{i} (r_{(0,k),(1,i)} * P_{(0,k)}) - (\sum_{k=1}^{L-1-i} (r_{(1,i),(1,i+k)} * P_{(1,i)}) - (\sum_{k=0}^{L-1-i} (r_{(1,i),(0,i+k)} * P_{(1,i)}), \quad (59)$$

Where i=0, 1, ... , L-1.

❖ **When peer A does not have a useful piece in its high prioritized buffer ($i_0 = 0$) and has $i_1 = i$ useful pieces in its non-prioritized buffer of size L-1, where $0 \leq i \leq (L-1)$.**

➢ In this case, peer A will be in the state of $(0, i_1)$; and therefore, we would have the following equilibrium equation:

$$P_{(0,i)} = \left(\sum_{k=0}^{i} (r_{(0,k),(0,i)} * P_{(0,k)})\right) + \left(\sum_{k=0}^{i} (r_{(1,k),(0,i)} * P_{(1,k)})\right) - \left(\sum_{k=1}^{L-1-i} (r_{(0,i),(0,i+k)} * P_{(0,i)})\right) - \left(\sum_{k=0}^{L-1-i} (r_{(0,i),(1,i+k)} * P_{(0,i)})\right), \quad (60)$$

Where i=0, 1, ... , L-1.

Solving the Eq. (59) and (60), will lead us to obtain the peer distribution $\{P_{(i_0,i_1)}\}$, where $1 \geq i_0 \geq 0$ and $(L-1) \geq i_1 \geq 0$.

Although it is very complicated to find a closed form for the peer distribution, it can be seen that it would be solved numerically.

3.3 NUMERICAL RESULTS

As it has been already explained, the reason that we proposed a new model in this chapter is that we try to achieve a better system performance (probability of continuity), compared to the previous model that has been proposed in the previous chapter. Here, we inspect how different parameters such as number of neighbours (*H*), delay time (*T*) and buffer length (*L*), can affect the performance of the system and verify our goal, which is achieving a better system performance, by comparing our new results to the results in previous chapter.

❖ The Effect of buffer length of the probability of continuity

In this part, we do the numerical results under the following assumptions:

- ✓ ***N=1000***

  In the numerical results, our streaming media has 1000 pieces in total.

- ✓ $\mathbf{2 \leq L \leq 20}$

  As we already explained, when peers are in the steady state, the length of the buffer is much smaller comparing to the size of N ($N>>L$). Here we do our numerical results for different values for *L*, between 1 and 20.

- ✓ ***H*=40**

  Each peer has forty neighbours.

Here, we will consider the effect of the buffer length (L) on the probability of continuity ($P_{Continuity}$), as it is shown in below:

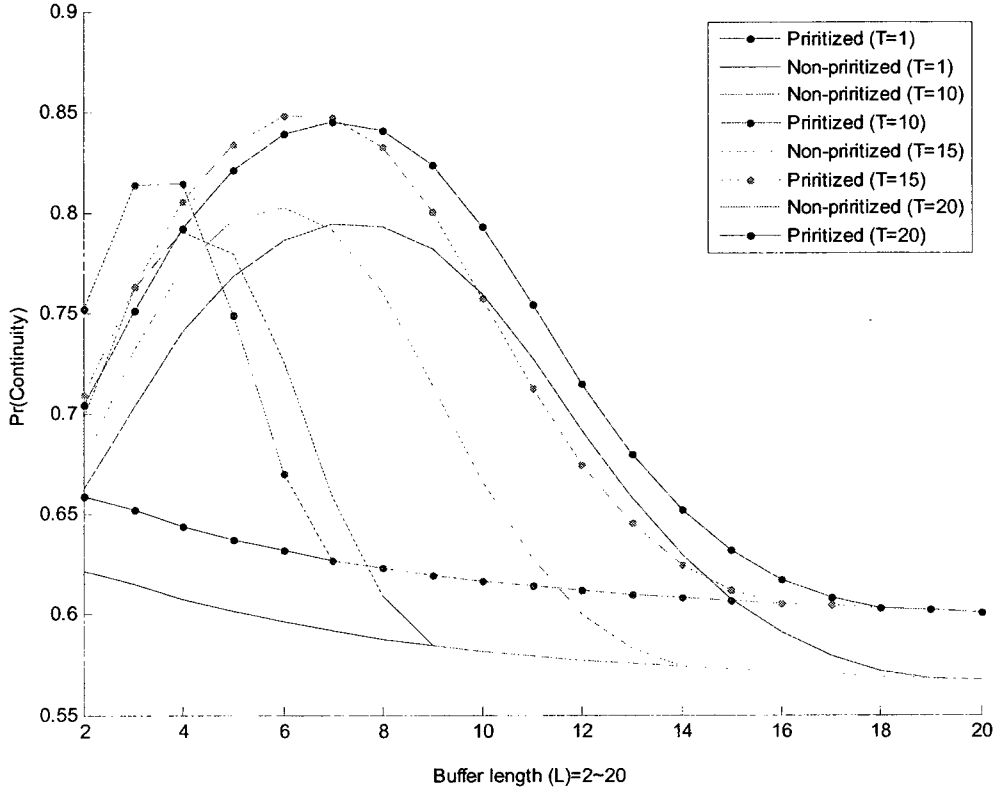


Figure 3.7: The effect of the buffer length on the probability of continuity ($P_{Continuity}$), where $20 \geq L \geq 2$ and T=1, 5, 10, 15 and 20.

In Figure 3.7, in most cases, we see that the performance of the system gets increased by increasing the length of the buffer somewhat. However, it can be seen when the buffer length goes beyond certain value, in any of the given cases, the performance of the system drops. The reason could be based on not having given priorities to the rest of the buffer.

It is also can be seen that by using our new proposed model we can achieve higher performance in our system.

❖ The Effect of Delay time

In this part, we do the numerical results under the following assumptions:

- ✓ ***N=1000***

  In the numerical results, our streaming media has 1000 pieces in total.

- ✓ $\mathbf{1 \leq T \leq 20}$

  Here we do our numerical results for different values for *T,* between 1 and 20.

- ✓ ***H*=40**

  Each peer has forty neighbours.

Here, we will consider the effect of the delay time (T) on the probability of continuity ($P_{Continuity}$), as it is shown in below:

As we see in Figure 3.8, adding some delay to our system, to some extent, can be helpful and elevates the performance of the systems. However, if the delay goes beyond some certain values, it is detrimental and decreases the system performance. It is because when the delay time increases in the system, the probability that different peers have different playtimes gets increased and as a result, the probability that any randomly selected peer in the system could be interested to its neighbours' pieces will be lesser and hence, the performance of the system gets reduced.

It also can be seen in Figure 3.8 that we can get to a better system performance in our new proposed model compared with the previous model that has been discussed in already; and gain we see that it verifies our goal in this chapter, which was achieving higher performance in our systems.

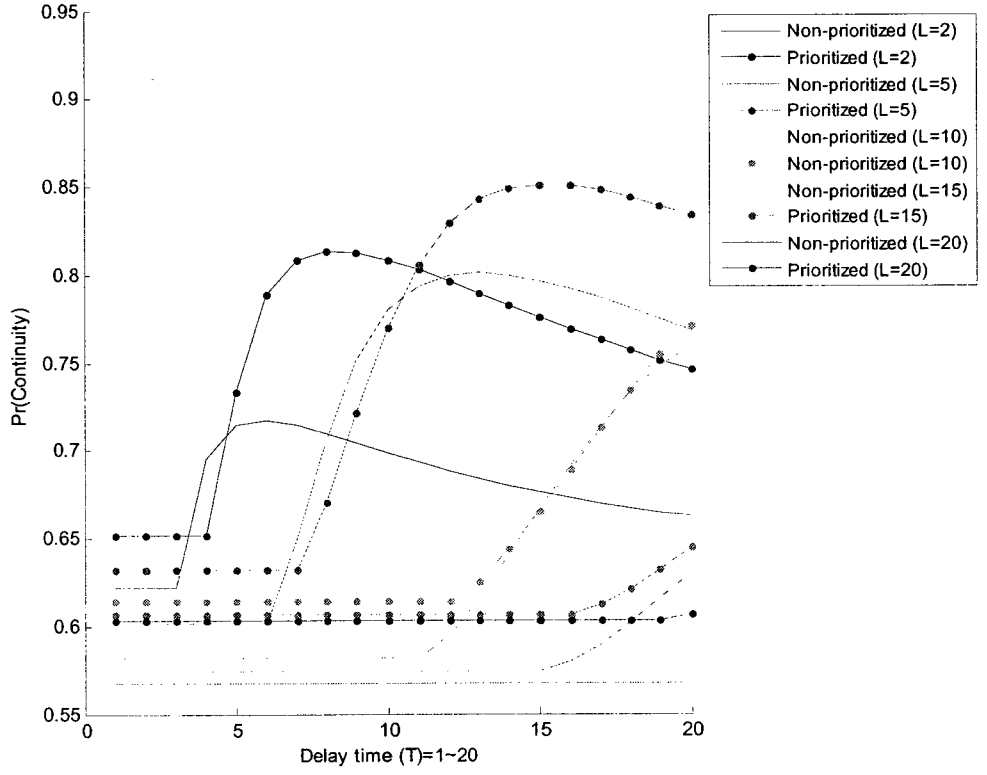


Figure 3.8: The effect of the delay time on the probability of continuity $(P_{Continuity})$, where $20 \geq T \geq 1$ and L=2, 5, 10, 15 and 20.

❖ The effect of neighbour number on the probability of continuity $(P_{Continuity})$

In this case, we do the numerical results for some random cases (three cases) as an example, to see how the number of neighbours can affect the probability of continuity $(P_{Continuity})$, as it is shown in below:

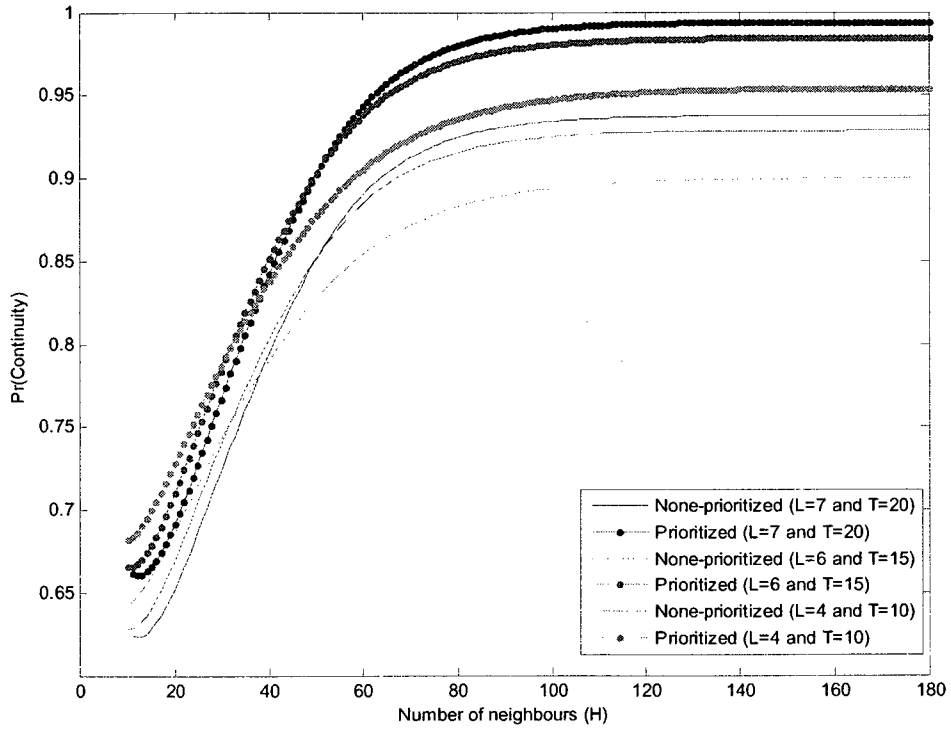


Figure 3.9: The effect of neighbour number on the probability of continuity $(P_{Continuity})$, where $180 \geq H \geq 10$.

When the number of neighbours is too small, the probability that a peer uploads to its neighbours is small and as a result, the performance of the systems (probability of continuity) is very low. When the number of neighbours gets increased, the probability of Continuity (performance of the system) will be increased. However, when the number of neighbours gets more than certain values, because the more number of neighbours causes the more competition among peers for downloading their desired pieces, the performance of the system will not be changed significantly. We also see that with our new proposed model, as it can be seen in the above Figure, performance of the system has been gotten better.

Chapter 4

CONCLUSION AND FUTURE WORK

4.1 CONCLUSIONS

During the last decade, peer-to-peer model have drawn so many attentions and a lot of new internet applications have been being designed based on this model.

P2P applications have been designed for the purpose of file-sharing substantially. However, in recent years, a significant amount of attention has been inclined towards using P2P file-sharing model, which is a time-non-sensitive model, for audio/video streaming purposes, which are time-sensitive.

In this thesis, the efficiency of BitTorrent-based P2P live streaming systems has been studied. Additionally, I proposed some minor changes to BitTorrent's mechanisms, in order to make BitTorrent support live streaming better.

In chapter 2, we proposed a stochastic model and under the assumption that our system is in the steady state, we analyzed the system and derived some significant analytical equations for different parameters of our system. Next, we solved the equations

numerically and then, we could be able to get to some great insights on how the different parameters of our system, such as the delay time, size of the buffer, number of neighbours, etc., can affect the probability of continuity and the download rate of a peer in this system. We also found that in different situations, each of these influential parameters can act differently on the performance of the system. Sometimes increasing/decreasing of each of these parameters can improve the performance of the system significantly; sometimes if the amounts of these changes go beyond more than they are needed, they can drop the performance of the system; and sometimes they may have not some significant influence on the performance of the system. Therefore, based on these achievements that we gained, we will be able to design an optimum P2P live streaming network.

As we already mentioned in previous chapters, an ideal P2P live streaming network can be defined as a system in which there is no discontinuity for any randomly picked peer in that network when it is watching, or listening to, its favourite streaming content.

In chapter 3, I tried to apply some minimal changes to the BitTorrent's mechanisms in order to have a BitTorrent-like P2P live streaming system more efficiently.

In this approach, for a randomly selected peer in the network, I give a higher priority to a piece that is supposed to be played in a given time slot, if this piece does not exist in the buffer at that time slot. Otherwise, the higher priority will be given to the next piece number, which should be played in the next time slot.

Then, I have proposed a new stochastic model for this case; and again I studied the system under the assumption that the system is in the steady state mode.

I have derived a set of equations again in order to describe the system. These equations have been numerically solved and we could be able to get to the peer distribution.

Then, I have shown that in our new design BitTorrent-like P2P live streaming system, we can be able to have a smoother playtime, a higher probability of continuity, comparing to our previous model.

4.2 FUTURE WORKS

Although our proposed models in this thesis contains almost all the important characteristics that a BitTorrent-base P2P live streaming system can have, it can be seen that we have assumed lots of assumptions in order to make our analysis simpler. To make it clear, we explain some of them in below:

1. We assumed that all peers in the network have unlimited download bandwidth or they have the same upload bandwidth, which are not true in real networks. In real networks, peers have limited download bandwidth and different upload bandwidth normally. Even a peer can also have different download/upload bandwidth in different times.
2. We also assumed that the number of requests that any randomly selected peer in the network sends in any given time slot is a fixed number and based on that we

have approximated the probability that a request of a randomly picked peer in the network, which has been sent to one its neighbour, gets fulfilled. However, we know that in real networks these assumptions are not true and we just made them to simplify our analysis.

3. In chapter 3, we tried to improve the performance of the system by putting higher priority only to closest piece that is close to be reproduced by the player. We went through several huge analytical formulas and finally, by solving the equations numerically and comparing the results, we showed that this was a successful approach somehow. One of the other future works could be an extension of our approach. Instead of giving priority only to one piece, we extend it and put different priorities on all the pieces. Depending on how close those pieces are, to be reproduced by the player, we give them different priorities. The closer each of them is to be reproduced by the player, the higher priority it will get.
4. Finally, we attained all the numerical results by doing the numerical results and unfortunately, we have not been able to perform our experiments in a real BitTorrent-like P2P live streaming network to verify out results, due to the complexity of the experiments. Therefore, scrutinizing our results in a real network is also another work that can be done in the future.

BIBLIOGRAPHY

[1] John Allen Paulos, available at *http://www.math.temple.edu/~paulos/* (January 2010).

[2] Hao Liu and George Riley, "How Efficient Peer-to-Peer Video Streaming Could Be?," in *proceeding of IEEE in Consumer Communications and Networking Conference (CCNC)*, 2009.

[3] World Internet Users and Population Stats, available at *http://www.internetworldstats.com/stats.htm/* (January 2010).

[4] Schollmeier, R., "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-PeerArchitectures and Applications," in *Proceedings of the First International Conference on Peer-to-Peer Computing P2P,* January 2001.

[5] Server, available at *http://compnetworking.about.com/od/basicnetworkingconcepts/g/network_servers.htm* (January 2010).

[6] Client/server architecture, available at *http://www.webopedia.com/TERM/C/client_server_architecture.html* (January 2010).

[7] Client/server, available at *http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci211796,00.html* (January 2010).

[8] Thin client, available at *http://www.webopedia.com/TERM/t/thin_client.html* (January 2010).

[9] ETHOS: Client Servers, available at *http://www.ethoseurope.org/ethos/Techterm.nsf/All/CLIENT+SERVERS* (January 2010).

[10] Graphical user interface, available at *http://en.wikipedia.org/wiki/Graphical_user_interface* (January 2010).

[11] Thin client, available at *http://en.wikipedia.org/wiki/Thin_client* (January 2010).

[12] Fat client, available at *http://en.wikipedia.org/wiki/Fat_client* (January 2010).

[13] What are the advantages and disadvantages of client/server?, available at *http://www.faqs.org/qa/qa-17360.html* (January 2010).

[14] P2P: Introduction and Real World Applications, available at *http://www.readwriteweb.com/archives/p2p_introduction_real_world_applications.php* (January 2010).

[15] 2008 Analysis of Traffic Demographics in North-American Broadband Networks, available at *http://www.sandvine.com/downloads/documents/Traffic_Demographics_NA_Broadband_Networks.pdf* (January 2010).

[16] World Internet Users and Population Stats, available at *http://www.internetworldstats.com/stats.htm* (January 2010).

[17] Computer Protocols- TCP/IP, POP, SMTP, HTTP, FTP and More, available at *http://vlaurie.com/computers2/Articles/protocol.htm* (January 2010).

[18] Carles Pairot Gavalda, Pedro Garcia Lopez, and Ruben Mondejar Andreu, "Deploying Wide-Area Applications Is a Snap", *IEEE Internet Computing,* Vol. 11, No. 2, pp. 72-79, March 2007.

[19] What is P2P (Peer-to-Peer) file downloading?, available at *http://www.bitjourney.com.tw/en_us/Prologue_IR_en.htm* (January 2010).

[20] RapidShare, available at *http://en.wikipedia.org/wiki/RapidShare* (January 2010).

[21] RapidShare is Appealing, available at *http://www.reuters.com/article/idUS198247+01-Feb-2008+PRN20080201* (January 2010).

[22] Archived front page of RapidShare.de, available at *http://web.archive.org/web/20061025014424/http://www.rapidshare.de/* (October 2006).

[23] Green Matthew. Napster opens pandora's box: Examining how file-sharing services threaten the enforcement of copyright on the Internet. Page 63:799. Ohio State Law Journal, 2002.

[24] File sharing, available at *http://en.wikipedia.org/wiki/File_sharing* (January 2010).

[25] eMule, available at *http://en.wikipedia.org/wiki/EMule* (January 2010)

[26] eDonkey / Overnet P2P File Sharing Client, available at *http://compnetworking.about.com/od/p2ppeertopeer/p/overnetedonkey.htm* (January 2010).

[27] Distributed_hash_table, available at *http://en.wikipedia.org/wiki/Distributed_hash_table* (January 2010).

[28] eMule, available at *http://en.wikipedia.org/wiki/EMule* (January 2010)

[29] Peer-to-peer, available at *http://en.wikipedia.org/wiki/Peer-to-peer* (January 2010).

[30] eDonkey / Overnet P2P File Sharing Client, available at *http://compnetworking.about.com/od/p2ppeertopeer/p/overnetedonkey.htm* (January 2010).

[31] Kazaa site becomes legal service, available at *http://news.bbc.co.uk/2/hi/5220406.stm* (January 2010).

[32] Filesharing history, available at *http://filesharingz.com/guides/filesharing-history.php* (January 2010).

[33] Bram Cohen, available at *http://en.wikipedia.org/wiki/Bram_Cohen* (January 2010).

[34] From P2P to BT, File-Sharing Software Upgrades with Lawsuits, available at *http://www.chinaipmagazine.com/en/journal-show.asp?id=327* (January 2010).

[35] The fourth P2P-Generation: Streams over P2P, available at *http://cmogen.com/sc_blog/?p=7#more-7* (January 2010).

[36] Ultimate guide for download and use P2PTV software to watch television online for free, available at *http://p2ptv.yourglobaltv.com/* (January 2010).

[37] Ipoque :: Profile, available at *http://www.ipoque.com/company/profile* (January 2010).

[38] BitTorrent Still King of P2P Traffic, available at *http://torrentfreak.com/bittorrent-still-king-of-p2p-traffic-090218/* (January 2010).

[39] P2P traffic on the rise in Germany, available at *http://www.p2p-blog.com/index.php?blogid=1&archive=2006-10* (January 2010).

[40] BitTorrent tracker, available at *http://en.wikipedia.org/wiki/BitTorrent_tracker* (January 2010).

[41] All about BitTorrent, available at *http://www.onlytorrents.com/what-is-torrent/* (January 2010).

[42] Distributed database, available at *http://www.azureuswiki.com/index.php/Distributed_database* (January 2010).

[43] BitComet, available at *http://en.wikipedia.org/wiki/BitComet* (January 2010).

[44] Incentives Build Robustness in BitTorrent, available at *http://www.bittorrent.org/bittorrentecon.pdf* (January 2010).

[45] p2p media streaming: Peer-to-peer streaming Internet TV, available at *http://all-streaming-media.com/peer-to-peer-TV/p2p-media-streaming-peer-to-peer-streaming-Internet-TV.htm* (January 2010).

[46] P2PTV, available at *http://en.wikipedia.org/wiki/P2PTV* (January 2010).

[47] T.S. E. Ng, Y.-H. Chu, S. G. Rao, K. Sripanidkulachai, and H. Zhang, "Measurement-based Optimization Techniques for Bandwidth-Demanding P22r-to-Peer Systems", *IEEE INFOCOM*, Vol. 3, pp. 2199-2209, April 2003.

[48] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design", *IEEE internet computing Journal*, Vol. 6, No. 1, 2002.

[49] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *Proceeding of International Conference on Peer-to-peer Computing,* August 2001.

[50] Z.Ge, D. R. Figueiredo, S. Jaiswal, J.Kurose, and D. Towsley, "Modeling peer-to-peer file sharing systems", *IEEE INFOCOM*, Vol. 3, pp. 2188-2198, 2003.

[51] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu, " Influences on cooperation in bittorrent communities, " in *Proceedings of ACM Sigcomm*, Philadelphia, PA, Aug 2005.

[52] S. Jun and M. Ahamad, "Incentives in bittorrent induce free riding," in *Proceedings of ACM Sigcomm*, Philadelphia, PA, Aug 2005.

[53] A. Legout and G. Urvoy-Kellerand P. Michiardi, "Understanding bittorrent: An experimental perspective," *Technical report*, Sophia Antipolis, France, 2005.

[54] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," in *Proceedings of ACM Sigcomm*, vol. 51, 2004, p. 61801.

[55] C. Dana, D. Li, D. Harrison, and C. Chuah, "Bass: Bittorrent assisted streaming system for video-on-demand," in *International Workshop on Multimedia Signal Processing (MMsP) IEEE Press*, 2005.

[56] X. Zhang, J. Liu, B. Li, and T.P. Yum, "Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming," in *Proceedings of IEEE/INFOCOM*, Miami, March 2005.

[57] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A.E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *Proceedings of IPTPS*, Ithaca, New York, Feb 2005.

[58] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for Supporting Streaming Applications", in *Global Internet Workshop in conjunction with IEEE INFOCOM 2006*, April 2006.

[59] S. Tewari and L. Kleinrock, "Analytical model for BitTorrent-based live video streaming," in *Proceedings of IEEE NIME Workshop*, 2007.