Multimedia Streaming over P2P networks using Markov Decision Processes

Ester Gutiérrez Zulaica

Assistant: Hyunggon Park

Supervisor: Prof. Pascal Frossard

May 2010
## Contents

1 Introduction .................................................. 8  
  1.1 Motivation .................................................. 8  
  1.2 Goal ......................................................... 9  
  1.3 Related work ............................................... 10  
  1.4 Overview ................................................... 12  

2 Preliminaries .................................................. 13  
  2.1 Terminology ................................................ 13  
  2.2 BitTorrent systems ......................................... 14  
  2.3 Foresighted Resource Reciprocation  
      Overview ................................................... 15  
  2.4 FRR limitations for multimedia streaming ............... 18
3 Foresighted Streaming Resource Reciprocation Strategy

3.1 Position Dependent Priority Functions
  3.1.1 Square shape priority function
  3.1.2 Exponential shape priority function

3.2 Segment-Type Dependent Priority Functions

4 Protocol

4.1 Learning Process

4.2 Decision Process
  4.2.1 Tit-for-Tat phase
  4.2.2 Foresighted Phase

4.3 Policy Finding Process

4.4 Downloading process
  4.4.1 Rarest First Search
  4.4.2 BiTos Piece Selection Algorithm
  4.4.3 New approach for the Piece Selection Algorithm

4.5 Seeder and Leecher state
5 Simulation Results

5.1 Simulation settings ........................................ 37
5.2 Utility ......................................................... 38
5.3 Selecting the priority function ............................. 39
  5.3.1 Square shape priority function simulations ............. 39
  5.3.2 Exponential shape priority function simulations .......... 41
5.4 Piece Selection Algorithm ..................................... 42
5.5 Comparison with other approaches .......................... 43
5.6 Video results .................................................. 45
  5.6.1 PSNR comparison ........................................ 46
  5.6.2 PSNR achieved for different number of rechoke periods 47
  5.6.3 Additional video results ................................... 48

6 Conclusions ...................................................... 51

A Video experiments .............................................. 52
  A.1 H.264 Video coding ........................................ 52
  A.2 Video sequences ............................................ 55
    A.2.1 Foreman sequence ..................................... 55
List of Figures

2.1 An illustration of resource reciprocation of peer $j$ associating with three peers at time $t$ and $t+1$. .......................... 15

2.2 Example of MDP scheme .................................. 16

3.1 An illustrative example of time-ordered data segments in an ideal decoding buffer. ............................... 21

3.2 Example of square shape priority function ................. 23

3.3 Example of exponential shape priorities functions ......... 24

4.1 The Main Processes in Protocol Design ..................... 26

4.2 Time line ..................................................... 27

5.1 Network topology ............................................ 38

5.2 Utility depending on distance $d$ ............................ 40

5.3 Results of the exponential function ........................ 41
5.4 Example of utilities achieved for different $p$ values . . . . . . . . 43
5.5 An illustrative example of the packet loss rate . . . . . . . . . . . . . 44
5.6 Example of utilities achieved . . . . . . . . . . . . . . . . . . . . . . . 45
5.7 Example of video quality received . . . . . . . . . . . . . . . . . . . . 46
5.8 PSNR achieved for different number of rechoke periods . . . . . . . 47
5.9 Frame 52 with different strategies . . . . . . . . . . . . . . . . . . . . 48
5.10 Frame 234 with different strategies . . . . . . . . . . . . . . . . . . . . 49
5.11 Frame 235 with different strategies . . . . . . . . . . . . . . . . . . . . 49
5.12 Frame 240 with different strategies . . . . . . . . . . . . . . . . . . . . 50
A.1 Encoding structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
A.2 Frames size for encoded sequence . . . . . . . . . . . . . . . . . . . . . . 54
List of Tables

4.1 Example of State transition table ..................................... 28

5.1 Average PSNR ................................................................. 46

A.1 H.264 encoding parameters for encoding structure ............... 54
Chapter 1

Introduction

1.1 Motivation

In recent years, multimedia streaming services are growing rapidly and becoming pervasive applications over the Internet. Traditional client server approaches allocate a dedicated stream from the server for each client request. This makes them expensive and does not scale well. Therefore, peer to peer (P2P) technologies have been proposed in order to overcome these limitations. In P2P networks, there is no dedicated infrastructure. Rather, the peers in the networks share their resources (e.g., content, bandwidth, etc.) by receiving them from other peers or contributing them to the other peers. One of the fundamental advantages of using P2P networks for multimedia streaming applications is to leverage peer upload capacities to minimize the bandwidth costs on dedicated streaming servers. Thus, P2P networks provide superior reliability and scalability compared to classic client-server type approaches. However, existing solutions for content distribution or content share cannot efficiently support the quality of service (QoS) in P2P multi-
media streaming applications.

Since Napster appeared in 1999, peer-to-peer (P2P) networks have experienced a great evolution. In 2003, P2P became the most important Web application and at the end of 2004, P2P protocols represented the 60% of the total traffic in Internet [13]. In this project, we consider data-driven P2P systems which adopt pull-based techniques (for example, CoolStreaming [15]). In these systems, multimedia streams are divided into chunks of uniform length, and are distributed over the P2P network. Each peer possesses several chunks which are shared among interested peers. Information about the availability of the chunks is also periodically exchanged among the associated peers. Using this information, peers continuously associate themselves with other peers (i.e., make coalitions) in order to exchange their chunks. Specifically in this work we adopt a very popular P2P network called BitTorrent [1,7,8] as the environment for our experiments.

P2P networks provide a cost effective framework for disseminating multimedia content. However, in P2P streaming systems a critical requirement is to operate the media distribution continuously in order to guarantee the Quality of Service (QoS). In order to play multimedia content successfully users should be able to start playback immediately after requesting the media and have uninterrupted playback during the download. Thus, schedule the order in which pieces of the desired media are downloaded becomes a critical requirement.

1.2 Goal

The aim of this project is to overcome the limitations of Foresighted Resource Reciprocation strategy (FRR) [11] for distributing multimedia content while providing a better performance for resource reciprocation than other solu-
In this project, we build on the Foresighted Resource Reciprocity strategy by explicitly considering the timing constraints for continuous display of the multimedia data and the importance of each multimedia data segment for the multimedia quality. In particular, we incorporate data priority functions into the reward function in order to adapt to the specific characteristics of media streaming applications. As a result, the peers exchange video packets with a strategy that ensures that the most important packets have a higher probability to reach the decoder on time for proper decoding.

1.3 Related work

In order to address the problem of reproduce multimedia content in this type of networks, several solutions have been proposed, [10,14,15].

For live media streaming a data-centric design is presented in [15]. The public Internet-based implementation called Coolstreaming v.0.9 has been used to lively broadcast sports programs. This implementations is used for live media streaming to a large population of users. However, our work is focused in both live and stored media streaming and thus the conditions for the users are not the same. In live media streaming all users are interested in the same pieces since they are all watching the same point of the content. On the other hand, in stored media streaming scenarios peers are interested in different pieces depending on its current playback point of the file. In [10,14,15] for efficient support of multimedia streaming, they deploy several peer or piece selection algorithms. For example, in [15], supplying peers having the highest bandwidth are selected and time allocation slots and data segments are selected depending on the number of potential suppliers In [5], however, data segments in the sliding window are randomly selected.
The most popular P2P protocol that is currently deployed in file sharing is BitTorrent [7, 8]. However, as discussed, the focus of this protocol is on efficient content distribution over P2P networks, without considering the timing constraints. Hence, this protocol can only provide a limited performance. Some changes in BitTorrent mechanisms are proposed in [14] to support streaming. This system, called BiToS, consist in three main components: the Received Pieces, the High priority Set and the remaining Pieces Set. The Received Pieces contains all the download pieces of the video stream. The High priority Set contains the pieces that are not been downloaded yet and are close to be reproduce while the Remaining Pieces Set contains the remain pieces that are not downloaded and are not in the High Priority Set.

While these solutions lead to an improved support for multimedia streaming, the resource reciprocation strategy does not consider the interactions of self-interested and heterogeneous peers. Moreover the peers determine their actions in order to maximize their utilities myopically without taking into account the projection of its actions in their future utilities.

In [11], the resource reciprocation among the interested peer is modeled as a stochastic game. Within this game, peers decide their resource distribution in a way that maximizes not only their immediate reward but also their cumulative future rewards. Using Markov Decision Processes (MDP) the peers estimate the associated peers time-varying behaviors relying on long term-history. Based on that information peers are able to find an optimal resource reciprocation strategy. As a result, this approach lead to a higher efficiency in P2P environment. However, this solution does not consider the time constrains that are critical for supporting streaming applications.

In this project a new approach for Foresighted Streaming Resource Reciprocation is presented in order to improve the streaming capability of the FRR strategy. While introducing a new priority function the scheduling algorithm will consider the time conditions of multimedia streaming.
1.4 Overview

This report is organized as follows: first, we briefly overview the foresighted resource reciprocation strategy and discuss its limitations for media streaming applications in chapter 2. Then, the proposed solution for its streaming capabilities limitations are discussed in chapter 3. The implementation of the new system is described in chapter 4 and the simulation results are presented in chapter 5. Finally the conclusions are drawn in chapter 6.
Chapter 2

Preliminaries

2.1 Terminology

In the peer-to-peer networks the terminology used is not standardized. For the sake of clarity, in this section the terms used in this project are defined.

A peer has two states, the leecher and the seed state. In leecher state the peer is still downloading content because it does not have all the pieces of the content. On the other hand, in seed state peer has already download all the pieces. Peers that download pieces and never upload are called free riders.

Each peer has a list called associated peer set. Peer $j$ can potentially send pieces to the other peers in its peer set, called $C_j$. If peer $j$ decides to share its contents with peer $i$, we say that peer $j$ unchokes peer $i$. Otherwise, peer $j$ chokes peer $i$. Each peer knows which pieces each peer has in its peer set. This enables the peer to focus its actions depending on its interest in other peer’s pieces.
2.2 BitTorrent systems

BitTorrent [8] is a successful peer to peer protocol, focused on efficient content delivery. A specificity of BitTorrent is the notion of torrent, which defines a session of transfer of a single content to a set of peers. A torrent is alive as long as there is at least one seed in the torrent. A user joins an existing torrent by downloading a .torrent file which contains information about the file to be download, such as the number of pieces in the file and the IP address of the tracker. The tracker has the IP addresses of the peers involved in the file distribution. Thus, the peer ask the tracker for a list of IP addresses of other peers to download the content and the addresses received formed the associated peer set.

The files transferred in a BitTorrent network are divided in pieces, and only complete pieces can be shared by a peer. Each piece is divided in blocks, which are the transmission unit on the network, but the protocol only accounts for transferred pieces.

Peers share pieces among peers using two core algorithms: the choke algorithm and the rarest first algorithm. The goal of the choke algorithm is to provide a fair reciprocation and penalize the free riders. The choke algorithm, called Tit-for-tat, is different for a peer in leecher state than for a seeder, a peer in seed state. In leecher state the algorithm is called every ten seconds and it is explained in detail in section 4.2.1. The rarest first algorithm is used to ensure a uniform dissemination of the file pieces and prevent peers from waiting too long to find the last missing pieces. Thus, the peer maintains the number of copies in its peer set of each content piece. It uses this information to define the rarest pieces, and they are updated each time a copy of a piece is added to the peer set. This mechanism is explained in more detail in section 4.4.1.
2.3 Foresighted Resource Reciprocation

Overview

In [11] resource reciprocation games, played by peers interested in other peers contents, are modeled as a stochastic game. In this game, peers decides its optimal actions based on the past history of resource reciprocation and the other peers behaviors. To successfully find this optimal actions, the resource reciprocation games are formulated as a Markov Decision Process (MDP) [4]. In order to understand this approach, an overview of the Foresighted Resource Reciprocation (FRR) is presented next.

\[
\begin{align*}
    &s_j(t) = \{s_{1,j}, s_{2,j}, s_{3,j}\} \\
    &s'_j(t+1) = \{s'_{1,j}, s'_{2,j}, s'_{3,j}\}
\end{align*}
\]

Figure 2.1: An illustration of resource reciprocation of peer \( j \) associating with three peers at time \( t \) and \( t + 1 \).

A Markov Decision Processes (MDP) contains a set of possible actions \( A \), a set of possible states \( S \), a real valued reward function \( R(s, a) \) for \( s \in S \) \( a \in A \) and a description \( P \) of each actions effect in each state. \( P \) is defined as the probability of states transition. Thus, for a peer \( j \) an MDP is a tuple \( < S_j, A_j, P_j, R_j > \). In Figure 2.1 is shown an example of the resource reciprocation game. This example shows how the state of peer \( j \) evolves from...
$s_j$ to $s'_j$ depending on action $a_j$ at time $t$, and the reactions of its associated peers $x_j$ at time $t + 1$.

![Diagram showing MDP scheme](image)

Figure 2.2: Example of MDP scheme

The details are explained as follows:

1. **State space** $S_j$: A state space of peer $j$ represents a set of resources received from the associated peers in the peer set at time $t$. In this paper, we consider that each of a state $s$ can have two values 0 and 1, i.e.,

   $$s_{i,j} = \begin{cases} 1, & \text{if } r_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (2.1)$$

   where $r_{i,j}$ represent peer $j$ resources received from peer $i$. Then, we can denote the state space of peer $j$ in its peer set as

   $$S_j = \{(s_{1,j}(t), \ldots, s_{N,j}(t))|s_{i,j} \in \{0,1\}\}.$$ 

2. **Action space** $A_j$: An action space is defined as a set of actions that peer $j$ can take in its peer set. We consider that an action of peer $j$ to peer $i$ can be either 0 or 1, which represents whether peer $j$ chokes or unchokes peer $i$, respectively. If peer $j$ unchokes peer $i$, this means
that peer $j$ shares its contents with peer $i$. Otherwise, peer $j$ chokes $i$ by sharing no content with peer $i$.

$$a_{i,j} = \begin{cases} 0, & \text{if peer } j \text{ chokes peer } i \\ 1, & \text{otherwise} \end{cases} \quad (2.2)$$

A set of actions that peer $j$ can take to its associated peers can thus be expressed as

$$A_j = \{(a_{1,j}, \ldots, a_{N,j})|a_{i,j} \in \{0, 1\}\}.$$

3. **State Transition Probability** $P_{a_j}(s_j, s_j')$: The state transition probability represents the probability that peer $j$ in state $s_j$ transit into a new state $s_j'$ by taking an action $a_j$. Thus, $P_{a_j}(s_j, s_j')$ maps each state into a future state taking an action $a_j$ as: $P_j : S_j \times A_j \times S_j \rightarrow [0, 1]$. We assume that the state transition probabilities are independent, thus, the state transition probability can be defined as:

$$P_{a_j(t)}(s_j(t), s_j(t+1)) = \prod_{i=1}^{N} P_{a_{ji}}(s_{ji}(t), s_{ji}(t+1)). \quad (2.3)$$

where $N$ is the number of peers associated to peer $j$. In Figure 2.2 an example of diagram of states, states’ transition probabilities and actions is shown.

4. **Reward** $R_j$: Reward $R_j(s_j(t))$ for a peer $j$ in state $s_j(t) \in S_j$ represents received resources from its state $s_j(t)$, expressed as

$$R_j(t)(s_j(t)) = \sum_{i=1}^{N} r_{i,j}(s_{i,j}) \quad (2.4)$$

where $r_{i,j}(s_{i,j})$ denotes the received resources from $s_{i,j}$.

5. **Reciprocation Policy** $\pi^*_j$: A reciprocation policy provides optimal actions $a_j(t) \in A_j$ from states $s_j(t) \in S_j$, i.e.,

$$\pi^*_j(s_j) = a_j.$$
This policy can be obtained from a solution to the MDP and peer $j$ can make foresighted decisions from all of its states. The foresighted actions enable peer $j$ to achieve a maximum cumulative discounted rewards (i.e., the immediate expected reward and the discounted future rewards) \[11\]. The cumulative discounted expected reward at time $t$ can be expressed as

$$R_j^{\text{fore}}(t)(s_j(t)) \triangleq \sum_{t=t'+1}^{\infty} \gamma_j^{(t-(t'+1))} E[R_j(t)(s_j(t))] \quad (2.5)$$

where $\gamma$ denotes a discount factor. The resource reciprocation policy can be obtained based on well-known methods such as value iteration and policy iteration \[4\]. Note that solving MDP may require high computational complexity, which exponentially increases as more peers are considered in a peer set.

Thus, it is important to only consider appropriate peers in each peer set. An illustrative implementation is discussed in \[12\].

### 2.4 FRR limitations for multimedia streaming

The FRR strategy for P2P networks is implemented in \[12\] in order to replace previous choke algorithms used in BitTorrent like the *tit-for-tat* strategy. Finding the optimal policy, peers decides its actions in order to maximize not only the immediate expected reward, but also the cumulative future expected rewards. Thus, this strategy improves the performance of the previous strategies where peers decides its actions myopically. Nevertheless the focus is only on maximize the received resources (i.e, download rates) and thus it provides a limited performance for multimedia streaming applications, as it does not consider multimedia characteristics especially timing constrains of
multimedia data. This may result in undesirable interruptions of continuous playback of multimedia streaming.

Moreover, in FRR, peers decides which piece has to be requested using the same strategy than BitTorrent, the rarest first search. As a consequence, peers will request the rarest pieces of the associated peers in order to increase the entropy of the system. This mechanism is explained in more detail in section 4.4.1. This strategy does not consider the time constrains of the segments which are critical for multimedia streaming. In time sensitive traffic, each piece should be received within a certain time limit. After this time, the piece is not useful and it is marked as missed and thus it can not be reproduced by the player. This factor is not considered in FRR, since pieces are requested based on their rareness in spite of being requested based on their deadline.

In streaming applications, such as video streaming, each peer needs to explicitly consider the orderings of the data segments, while downloading them as fast as possible. Specifically, each peer needs to receive the data before their decoding deadlines. Furthermore, in video streaming, there is different types of frames depending on the importance of each frame for decoding. Each data packet may have different quality impact depending on the encoding structure and the type of picture in multimedia streaming applications (e.g., video streaming). Hence, in order to successfully display multimedia content using the FRR strategy we are going to introduce a new priority function that differentiates between packets depending on their decoding deadlines and quality impact.
Chapter 3

Foresighted Streaming Resource Reciprocation Strategy

As discussed before, the Foresighted Resource Reciprocation Strategy has to be reformulated in order to overcome the limitations for multimedia streaming. Thus, a solution that provides streaming capability is presented next.

In this chapter we are going to define a new concept of reward, which both includes the priority of the packets and the resources received from the associated peers. The new reward definition will incorporate a new priority function called $\rho$. Thus, the reward obtained by peer $j$ in state $s_j$ is defined as follows:

$$R_j(s_j) = \sum_{i=1}^{N} \rho_{j,i}(t)s_{j,i}r_{j,i}$$

(3.1)

where $N$ is the number of associated peers and $\rho_{j,i}$ represents the preference...
that peer $j$ has on segment of peer $i$ at time $t$.

The preference not only depends on the time constrains of that segment, but also on the importance of that segment for decoding. Thus, we are going to define two different $\rho(t)$ functions:

$$\rho_{j,i}(t) = \rho^T(t)\rho^D$$

(3.2)

where $\rho^T(t)$ depends on the timing constraints (i.e, relative position in the ideal decoding buffer) and $\rho^D$ depends on the quality impact of each segment $x$.

### 3.1 Position Dependent Priority Functions

As discussed before, we need to consider the time constrains for real time streaming applications. Thus, peers should deploy a specific strategy to determine the order of the segment requested.

Figure 3.1: An illustrative example of time-ordered data segments in an ideal decoding buffer.
In Figure 3.1 there is an example of a multimedia buffer. The packets are ordered in the buffer as they are going to be displayed. Colored packets represent the packets that have been downloaded already, and the rest of them are the packets that are scheduled for downloading. The dotted line represents a sliding window, which is moving as the packets are being decoded by the peer. Thus, the packets that have not been downloaded before they reach the playback point are considered as lost packets. As a consequence, peer experiments a poor video quality as the packet loss increases.

To avoid loosing these packets, the pieces that are not downloaded and are close to the playback point must have higher priority to be requested. We define the priority in this case as a function that depends on the piece’s position in the buffer structure. The main characteristics of these functions are:

1. $\rho^T(t)$ has to be a decreasing function of time. This will guarantee the higher priority for the first pieces, and the low value of priority for those that are going to be displayed in the future.

2. $\rho^T(t)$ has to be updated each time the peer downloads a new piece in order to request only the pieces that are not downloaded yet.

In this project, different shapes for the time dependent priority functions will be discussed. In next sections two illustrative priority function shapes will be considered: the square shape and the exponential shape priority functions.

### 3.1.1 Square shape priority function

The motivation of square shape priority function is to divide the segments into two groups - groups with high priority and with low priority. In both
groups all the packets have the same value for the priority function. This can be easily extended to several levels of priority by defining stair shape priority functions.

The corresponding priority function \( \rho_{T_i,j}(x) \) can be defined as

\[
\rho_{T_i,j}(x) = \begin{cases} 
\alpha_H & \text{if } x < d \\
\alpha_L & \text{if } x > d 
\end{cases}
\]  

(3.3)

where \( \alpha_H > \alpha_L \) and \( d \) is a threshold dividing the two groups. The data segments in the group of high priority are closer to the playback point than those in the group of low priority. The value of \( d \) can be determined as a percentage for the part of the total number of data segments in a file. In Figure 3.2 an example of square shape priority function is shown.

![Figure 3.2: Example of square shape priority function](image)

The distance \( d \) can be represented as a percentage of the total number of packets in a file. Several simulations with different \( d \) values have been developed and the results are shown in section 5.3.1.
3.1.2 Exponential shape priority function

Another different shape for the priority function is proposed in order to evaluate different approaches. Unlike the square shape priority function, which imposes the same priority on the data segments in a group, exponential shape priority function can impose individually different priorities to data segments. This type of function also enables the peers to easily control the variation of priorities among the segments, leading to maximum performance.

The exponential shape priority function can be expressed as

$$\rho_{i,j}(x) = e^{-\beta x} \quad (3.4)$$

where $\beta$ determines how fast the priorities are decaying.

![Exponential Shape Priority Function](image)

Figure 3.3: Example of exponential shape priorities functions

The parameter $\beta$ has been studied, similar to the distance $d$ for the square shape priority function. The results are shown in section 5.3.2. In Figure 3.3 an example of exponential shape functions with different $\beta$ values are shown.
3.2 Segment-Type Dependent Priority Functions

A sequence of compressed video frames may have different types, such as I-frame, P-frame, and B-frame in MPEG or H.264 standards [6, 9]. Thus, the corresponding data segments can also be characterized by the type of the frame they represent. For the sake of simplicity, we consider that the priority function based on the segment types is determined by the quality impact of the types. In this project three different types of frames are considered as it is shown in Figure 3.1. In the buffer structure it is shown three different frames called H, M and L which represents the High, Medium and Low priority frames in the framework of video coding. Thus, for decoding the video packets, the H frames have higher priority than the M frames or the L frames. This division has been made based on the different types of frames for videos, i.e.: I, P and B. More information about video coding structures can be found in Appendix A.1. As a result, the corresponding priority function \( \rho_{i,j}^D \) is expressed as:

\[
\rho_{i,j}^D(x) = \begin{cases} 
\gamma_H, & \text{if } x \in \mathcal{H} \\
\gamma_M, & \text{if } x \in \mathcal{M} \\
\gamma_L, & \text{if } x \in \mathcal{L}
\end{cases}
\]  

(3.5)

where \( \mathcal{H}, \mathcal{M}, \) and \( \mathcal{L} \) denote the set of data segments marked with H, M, and L, respectively. The priority function \( \rho_{i,j}^D(x) \) can provide different levels of priority on each data segments based on its quality impact. Finally, the streaming reward function will be calculated as

\[
R_{j}^{fore}(t)(s_j(t)) = \sum_{t=t'+1}^{\infty} \gamma_j^{(t-(t'+1))} \sum_{i=1}^{N} \rho_{i,j}(x)r_{i,j}(s_{i,j})
\]  

(3.6)
Chapter 4

Protocol

In this section we are going to present our system’s implementation as well as the implementation of the other systems in order to compare the result obtained using different solutions. Moreover, we are going to explain in detail how the implementation of the leecher and seeder state has been developed.

Figure 4.1: The Main Processes in Protocol Design
For the implementation of the system each peer has 4 different processes (see Figure 4.1): the Learning Process, the Decision Process, the Policy finding Process and the Downloading Process. Each peer uses the 4 processes in parallel, which are called every X seconds depending on their period. For example, the Decision Process is called every rechoke period (10 seconds, as in BitTorrent systems) and the Policy Finding Process is called every 3 rechoke periods. In this implementation the Learning process is called more often in order to store the state’s transitions due to the other peers actions. Finally, the downloading process is called every transmission unit on the network. In Figure 4.2 there is an example of the different processes’ periods.

![Figure 4.2: Time line](image)

In next sections we briefly review these processes and introduce the new piece selection mechanism.

### 4.1 Learning Process

The aim of the *Learning Process* is to provide the information of peer set behaviors. Thus, the Learning process is designed as a thread that stores the result of the actions taken by the associated peers in order to provide this information to calculate the optimal policy. This information is stored in a table called *State transition table*. This table $T_j$ (see example in Table 4.1) is divided in 8 rows, each one of them represents the probability $P_{s_i,j}$ →
s'_{j,i}(a_{j,i}) and the number of columns depends on the number of associated peers. Each time the learning process detects any change in peers current state it recalculate the probability that each peer in the associated peer set decides to choke or unchoke peer \( j \) depending on the action taken by peer \( j \). The probabilities are calculated using empirical frequency as it is proposed in [11].

The probabilities are calculated using empirical frequency as it is proposed in [11].

<table>
<thead>
<tr>
<th>State transition Probability</th>
<th>Peer 1</th>
<th>Peer 2</th>
<th>\ldots</th>
<th>Peer N</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_{s_{j,i}=0 \rightarrow s'<em>{j,i}=0} ) ((a</em>{j,i} = 0))</td>
<td>0.5</td>
<td>0.5</td>
<td>\ldots</td>
<td>0.5</td>
</tr>
<tr>
<td>( P_{s_{j,i}=0 \rightarrow s'<em>{j,i}=1} ) ((a</em>{j,i} = 0))</td>
<td>0.5</td>
<td>0.5</td>
<td>\ldots</td>
<td>0.5</td>
</tr>
<tr>
<td>( P_{s_{j,i}=1 \rightarrow s'<em>{j,i}=0} ) ((a</em>{j,i} = 0))</td>
<td>0.5</td>
<td>0.5</td>
<td>\ldots</td>
<td>0.5</td>
</tr>
<tr>
<td>( P_{s_{j,i}=1 \rightarrow s'<em>{j,i}=1} ) ((a</em>{j,i} = 0))</td>
<td>0.5</td>
<td>0.5</td>
<td>\ldots</td>
<td>0.5</td>
</tr>
<tr>
<td>( P_{s_{j,i}=0 \rightarrow s'<em>{j,i}=0} ) ((a</em>{j,i} = 1))</td>
<td>0.5</td>
<td>0.5</td>
<td>\ldots</td>
<td>0.5</td>
</tr>
<tr>
<td>( P_{s_{j,i}=0 \rightarrow s'<em>{j,i}=1} ) ((a</em>{j,i} = 1))</td>
<td>0.5</td>
<td>0.5</td>
<td>\ldots</td>
<td>0.5</td>
</tr>
<tr>
<td>( P_{s_{j,i}=1 \rightarrow s'<em>{j,i}=0} ) ((a</em>{j,i} = 1))</td>
<td>0.5</td>
<td>0.5</td>
<td>\ldots</td>
<td>0.5</td>
</tr>
<tr>
<td>( P_{s_{j,i}=1 \rightarrow s'<em>{j,i}=1} ) ((a</em>{j,i} = 1))</td>
<td>0.5</td>
<td>0.5</td>
<td>\ldots</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 4.1: Example of State transition table

A State Transition Probability represents the probability that by taking an action, a peer will transit into a new state. The state transition probability is calculated using the previous equation (2.3). Note that the above equation is only valid if we assume that state transition probabilities of associated peers are independent. Peer \( j \) has 2 possible states respecting peer \( i \): \( s_0 \) \( s_1 \), and 2 possible actions to peer \( i \): \( a_0 \) and \( a_1 \). Thus, there is 4 possible state transitions for each action, and as a consequence we need to store the information for all this 4 \( \times \) 2 transitions. Hence, using the State transition table with size 8 x Number of peers the learning process can calculate the peer’s state transition probabilities as required. While the empirical frequency method is deployed, other algorithms can also be used, such as reciprocation models, discussed in [11].
Learning Process also stores the information about the upload rates received from the associated peers. Every new round starts the estimated Upload rate of peer $j$ to peer $i$ is updated as follows:

$$\hat{U}_{s_i,j} = \alpha \hat{U}_{s_i,j} + (1 - \alpha)r_{i,j}. \quad (4.1)$$

where $\alpha$ represents the belief of peer $j$ in the past received resources from peer $i$. Consequently, the fluctuation of the network dynamics is reduced. Moreover, peer $j$ estimate optimistically the upload rate of a peer that newly join the network. Thus, for peers without resource reciprocation history, peer $j$ assumes that they are going to reciprocate their resources with high probability and high upload rate. This helps new peers to get their first pieces and encourage peers to discover new partners.

4.2 Decision Process

The purpose of the Decision Process is to determine the optimal actions identified by the optimal policy $\pi_j^*$. Therefore, it can generate the actions that peer $j$ will take in its current state, i.e.,

$$\pi_j^*(s_j(t)) = a_j(t)^*.$$

However, in the initial phase where the peer $j$ first joins the system, it does not have enough information about its associated peers to calculate the policy. For this reason, in our implementation, the decision process begins with applying the tit-for-tat strategy until the number of reciprocations is sufficient for capturing the behaviors of other peers. The two phases of the Decision Process are explained in detail next.
4.2.1 Tit-for-Tat phase

In this phase the decision process deploy the Tit-for-Tat strategy before the program starts to calculate the Foresighted Resource Reciprocation policy. The Tit-for-Tat strategy is the most common choking algorithm used by BitTorrent. When in leecher state, the choke algorithm is called every ten seconds and the following steps are executed:

1. At the beginning of every 3 choked decisions, a random choked peer is selected to be unchoked. This is called the “optimistic unchoke” and enables the peer to find new partners and also gives an opportunity to those peers who do not have any piece to share by giving them their first pieces.

2. Secondly the algorithm choose another 3 peers based on their current download rate to peer I. Thus, the three fastest peers are unchoked.

3. Finally, if the optimistic unchoke peer is one of the three fastest peers, the algorithm chooses another random peer to be the optimistic unchoked peer.

This phase enables our algorithm to start learning the associated peers behaviors before having enough information to begin the foresighted phase. Once the learning process determines that it has enough information to process the foresighted Resource Reciprocation policy, the decision process will start executing the second phase. This transition is set as default after 10 state transitions.
4.2.2 Foresighted Phase

In this phase the choke algorithm is based on the calculated optimal policy given by the Policy finding Process. This process computes the strategy to follow in order to maximize the future rewards and share this set of actions with the decision process. Thus, the decision process becomes the responsible to carry out the actions considered in the policy finding process. The decision process checks periodically the result of the policy finding process. This enables the thread to update the policy as needed. After that, the process checks the current state of the peer and applies the optimal action that matches with that state. This procedure is repeated every 10 seconds as in the Tit-for-Tat phase.

4.3 Policy Finding Process

The Policy Finding Process is divided in two main processes: the peer set reduction and the policy calculation. As explained before, peer set reduction is needed in order to reduce the computing complexity of calculating the foresighted policy. The main purpose of reducing the number of peers is to reduce the action and state space while keeping the peers that reciprocate their resources with higher probability and higher upload rate. The peer set reduction algorithm used is the one introduced in [12].

After finding the peer set reduction, the peer is prepared to compute the foresighted policy as follows:

1. For all possible states and for all possible actions, peer j calculates de value, $V$, of the rewards received taking the action $a_i$ in the state $s_i$. The value $V$ is computed as the sum of the cumulative discounted
expected reward, expressed as.

\[
R_{j}(s_j(t)) = \sum_{s_j(tc+1)\in S_j} P_{a_j(tc+1)}(s_j(tc), s_j(tc+1))R_j(s_j(tc+1))
\]

\[
\inf_{t'=tc+1} \sum_{s_j(t'+1)\in S_j} \gamma P_{a_j(t'+1)}(s_j(t), s_j(t'+1))R_j(s_j(t'+1))
\]

\[ (4.2) \]

2. The action selected for that state is the one that maximizes \( V \). Thus, we obtain a policy for all states applying the value iteration algorithm for each peer’s possible state.

The optimal policy can be obtained using well-known methods such as value iteration and policy iteration [4]. In this project the value iteration algorithm has been implemented as follows:

**Algorithm 1** Value iteration algorithm

- Initialize \( V(s) \) arbitrarily

- while \( |V(s) - V'| < \epsilon \) do

- \( V' = V(s) \)

- for \( s \in S \) do

- for \( a \in A \) do

- \( Q(s, a) = R(s, a) + \gamma \sum_{s'\in S} P(s, a, s')R(s', a) \)

- \( V(s) = \max_{a} Q(s, a) \)

This algorithm is repeated until the difference between two successive value functions is less than \( \epsilon \). The discount factor used in (4.2) is needed to make the value iteration method converge to 0.
4.4 Downloading process

This process is responsible for choosing the pieces that are going to be downloaded from the associated peers. The Download Process runs in parallel with the other 3 main processes. In order to simulate the packets size, the time to download a piece is chosen depending on the current download rate received from the associated peers and the packet’s number of bytes. For the sake of simplicity, in our simulation the propagation time and the network dynamics are not taken into account. Thus, only the download rate received is needed to calculate the velocity for download a packet. The minimum time for download a packet is the minimum block size divided into the maximum upload rate per peer. A block is the unit of data transfer. Once the minimum time, \( t_{\text{min}} \) is calculated, the downloading process is called every \( t_{\text{min}} \) seconds to check if the download is completed or not. Once the peer completes the download of a piece, the Piece Selection algorithm chooses the next piece to be downloaded and the downloading process continues.

However, not all the packets have the same size since it depends on the type of frame encoded. More details respect the encoding and the type of frames could be found in Appendix A.

The Piece Selection algorithms used in BitTorrent, BiTos and the one introduced for our approach are explained as follows.

4.4.1 Rarest First Search

The goal of the Rarest First Search algorithm is to maximize the entropy, or the diversity, of the packets in the network. The pieces are selected based on its rareness, thus the rarest pieces become duplicate faster. As a result, this algorithm does a good job at attracting missing pieces in a peer set.
Moreover, the rarest first algorithm is presented in order to avoid the problem of the last pieces [5]. We say that there is a last pieces problem when the downloading speed decrease for the last pieces. Thus, as these pieces become more difficult to be founded, the rarest first algorithm improves the behavior of the choke algorithm in BitTorrent systems.

Although this mechanism is very efficient in maximizing the pieces entropy, it fails in case of sensitive time data. Rarest First Search mechanism is adequate only for distributing a entire file because all parts of the file need to be download previously before it can be displayed.

4.4.2 BiTos Piece Selection Algorithm

In BiTos [14] the mechanism for selecting a piece is developed in order to provide streaming capability to the BitTorrent regular protocol. Thus, pieces are divided in two groups, the High priority group and the Remaining pieces set. The first group contains the pieces that have not been downloaded and are close to the playback point. The peer chooses with probability $p$ to download a piece in the High priority group and with probability $1 - p$ a piece in the Remaining pieces set. The mechanism used to choose a piece within the first or the second group is the Rarest First search. However, they also include a minor change in the rarest first mechanism which is that, if two or more pieces has the same rareness, the piece closest to the playback point is selected.

However, the value of the probability $p$ highly depends on the dynamics of the scenario, and thus for their scenario they chose $p = 0.8$ as they obtained better results.
4.4.3 New approach for the Piece Selection Algorithm

For our approach, the Piece Selection Algorithm has been modified in order to consider both the rarest first search and the priority of the packets. We implement our algorithm similar to the BiTos piece selection algorithm. The peer chooses to download a piece depending on its priority with probability $p$ and using the rarest first search algorithm with probability $1 - p$. This provides the system with the advantages of using the rarest first search algorithm as the entropy of the pieces increase. Moreover, peers decide to download pieces depending on their priority which enables the peers to obtain the first pieces to start reproducing the multimedia content. The probability $p$ highly depends on the dynamics of the network, similar as in BiTos system, thus for our scenario we choose $p = 0.8$. However, an adaptive mechanism for $p$ could be an interesting field for future research.

4.5 Seeder and Leecher state

The processes explained before are only valid if the peer is in leecher state. Otherwise, if the peer is in seed state, the decision process change and the Downloading, Policy Finding and Learning process stops as they are not needed anymore. Thus, when the peer finish to download all the packets it changes its state into seeder. This is only valid if we are not considering the free riders. The implementation of the Decision Process in seed state has been programmed based on [8]. The process is called every 10 seconds, and only the associated peers are considered. The implementation is explained as follows:

1. The peers are ordered based on their upload rate and the time they were last unchoked, giving priority to tho the highest upload rate and
most recently unchoked peers.

2. During three re choke periods, the first 3 peers are unchoked and another random peer is unchoked.

The previous versions of BitTorrent mainline client only ordered the peers based on their upload rate, thus, a single peer could monopolize the resources of the seeder. Using this new method, peers in the active set are changed frequently and thus, a single peer can not be always unchoked by the seeder.
Chapter 5

Simulation Results

In this chapter we are going to present the results of the simulations. This chapter is organized as follows. First, we are going to describe the simulation setting. Secondly, we are going to present the results about the different proposed priority functions in order to evaluate the effect in the utility achieved by the peers in section 5.3. The Piece selection algorithm will be also studied in 5.4. Then, we are going to present the results obtained by comparing our approach with the other solutions for peer to peer networks, such as BitTorrent, BiTos and the Foresighted Resource Reciprocation strategy in section 5.5. Finally, we are going to show the results using a real video file in section 5.6.

5.1 Simulation settings

The implementation of the system, described in chapter 4, has been programmed using Java environment. The concurrent programming methods for Java allow to simulate each peer as an independent thread. Thus, 15 dif-
different peers have been programmed as threads, 2 as seeders and the rest of them as leechers. An example of the network scheme is shown in Figure 5.1. Leechers and seeders has a buffer which represents the pieces that have been already downloaded (shadowed) and pieces that are going to be requested. In our simulation each leecher starts with different random pieces in order to simulate different network conditions.

![Network topology](image)

Figure 5.1: Network topology

The total upload bandwidth available for each peer is 128Kbps. In the simulations we assume that the download bandwidth is large enough to only consider the upload bandwidth of the supplying peers to calculate the time of download a piece. Simulations are performed based on the assumption of static network condition, i.e., a given network topology with a fixed number of peers.

### 5.2 Utility

In order to quantify the impact of the priority function on the average performance of the proposed algorithm, we define a measure $U_j(t)$ for peer $j$,
which is referred to as a utility in this project, as

\[ U_j(t) = \sum_{x \in N_p} p_{i,j}^T(x) \cdot (1 - P_L(x)) \quad (5.1) \]

where \( N_p \) represents the total number of data segments and \( P_L(x) \) represents the packet loss probability or equivalently the probability for a data segment located at position \( x \) not to be received by the playback deadline.

### 5.3 Selecting the priority function

As explained before in chapter 3, a new priority function has been defined in order to provide streaming capability to the previous implementation of the FRR strategy. In this section the Position Depending Priority Function is studied in order to compare the utilities achieved for different function’s shape.

#### 5.3.1 Square shape priority function simulations

The square shape priority function, defined in 3.1.1, has a parameter \( d \) that represents the width of the High Priority group. In order to select a value for distance \( d \) a set of simulations have been developed. The distance \( d \) is defined as a percentage of the entire file. The simulations are presented in Figure 5.2.

We can observe that higher utilities can be achieved for lower values of \( d \), while the achieved utilities decrease as the values of \( d \) increases. If
the values of $d$ for each peer become high (e.g., $d > 80\%$), the role of the priority functions is minimized, leading to the similar effect that there is no priority function. However, if lower values of $d$ (e.g., $d < 10\%$) are used for each peer, then all the peers focus on downloading only a small number of data segments with the high priorities, leading to a small availability of data segment replicas.

Note that as the distance $d$ is increased the results obtained are similar to the results obtained with the same conditions using the Foresighted Resource Reciprocation strategy. The FRR strategy does not consider the priority of the packets and thus, the policy is calculated only depending in the resources received. As a consequence, the results of using a high value for distance $d$, approximatively to 100\%, are logically the same as using the FRR strategy.
5.3.2 Exponential shape priority function simulations

This function has a parameter $\beta$ that could be adjusted to maximize the utility. The results for different values of $\beta$ are shown in Figure 5.3.

The results are represented for different values of $\beta$ logarithm. The cause of these results are similar to those obtained with the previous function. For large values of $\beta$, exponential function is similar to a narrow square function. Thus, a deterioration of the utility is produced because only the first packet closest to the playback point is considered. Also for very small values of beta utility decreases as we increase the number of packets to be considered more urgent. In this example, $\beta = 1$ leads to the highest utility. This function provides better performance due to the decreasing shape that provides different values for all the pieces depending on its position. As a result, the time priority of the packets is always considered and the policy found is focused in download the pieces in order. Thus, the granularity of the exponential function provide better performance for our simulations.
5.4 Piece Selection Algorithm

The Piece Selection Algorithm proposed in section 4.4.3 is a mixed algorithm between \textit{Rarest First Search} and a selection based on the priority of the pieces. The parameter that modules the behavior of this algorithm is the probability $p$. In this section we are going to study the impact of $p$ value on system’s utility.

The results presented have been obtained using the same number of re-choke periods. The only parameter that has been modified during the simulations is $p$ from 0.01 to 1. The results for different values of $p$ as an percentage are shown in Figure 5.4. This results clearly shows that the utility achieved increases as the percentage of $p$ increases. Large values of $p$ means that the piece’s selection highly depends on the piece’s priority rather than its rareness, and thus, the systems utility increases since more priority pieces are downloaded. However, for values of $p$ closest to 100\% or in other words $p = 1$, the utility decreases. Since $p$ closer to 1 means that the rareness of a piece is never considered, the entropy of the packets decreases. Finally, we can say that the probability $p$ is a balance between receiving the most priority pieces on time and replicate the rarest pieces.
5.5 Comparison with other approaches

In this section we are going to show the results of the comparison between our approach with other strategies such as BitTorrent, BiTos and the FRR strategy. The simulations of all the systems have been done using the same time conditions and the same initial conditions in order to evaluate with fairness the results obtained. Thus, all the simulations have been run with the same number of rechoke periods, the same file and packets size and the same bandwidth conditions.

In order to calculate the optimal policy successfully the number of peers in the peer set after applying the peer set reduction algorithm is 6. We assume that the number of slots for downloading is 4, and as a consequence, the number of possible actions is reduced from $2^6 = 64$ to 57. The discount factor is $\gamma = 0.8$. In BiTos implementation, the value of $p$ is set to 0.8 and for the tit-for-tat strategy the number of slots for downloading is also 4. For
our approach, the time depended priority function defined in 3.4 is used as it provides better performance than the square shape priority function. We compare the 4 algorithms after 50 rechoke periods to display the results in terms of Packet Loss Rate and Utility.

In Figure 5.5 an example of Packet Loss Rate is shown. The pieces has been grouped based on its position in the buffer and the lower number of priority group represents the pieces closest to the playback point. The results clearly shows that our approach provides a better performance in terms of packet loss. Also worth noting that our solution not only reduces the loss rate of urgent packets, but also improves long-term performance. This is due to the purpose of the Foresighted Resource Reciprocation strategy of maximizing long-term rewards.

In Figure 5.6 there is an example of the Utilities achieved. The state index represent the different initial states of a peer. The results shows clearly that the proposed solution improves the Utility for all the proposed initial conditions.

Figure 5.5: An illustrative example of the packet loss rate
In this section we are going to present the results for real video sequences. More information about video coding, sequences and additional results can be find in section A. For this simulation we have encoded a video sequence and then we have used our proposed algorithm as well as the other approaches studied to send the video packets. The simulations have been programmed in order to obtain results after certain number of rechoke periods. This will provide results for different playback delays since the number of rechoke periods is directly proportional to the time as it was shown in Figure 4.2. The results presented next have been obtained using the same number of rechoke periods for each peer. After the simulations, the video sequence is decoded and the PSNR is calculated on average for all the frames received and decoded.
5.6.1 PSNR comparison

For this simulations peers share a video sequence called Foreman sequence, with 300 frames, CIF quality and size 17MB. The frame rate is 30frames/second. The results are shown in Figure 5.7.

Results shows that our proposed algorithm improves the video quality received compared to the other strategies implemented. The state index represent the different initial states of a peer. In table 5.1 we can see the average quality received for all cases.

<table>
<thead>
<tr>
<th>Proposed Algorithm</th>
<th>35.641</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiTos</td>
<td>32.1678</td>
</tr>
<tr>
<td>FRR strategy</td>
<td>22.008</td>
</tr>
<tr>
<td>TFT strategy</td>
<td>16.6062</td>
</tr>
</tbody>
</table>

Table 5.1: Average PSNR

On average, the proposed algorithm improves 3.47 dB over the BiToS
system, 13.63 dB over FRR strategy and 19.03 dB over the BitTorrent-like TFT strategy across all the different state indices using the same conditions.

5.6.2 PSNR achieved for different number of rechoke periods

In this section we are going to compare the PSNR achieved for a different number of rechoke periods. In this simulations, the 15 peers download the Salesman video sequence of 450 frames, with QCIF quality and a frame rate of 30 frames/second. The 15 seconds of video is sent 8 times. We compare the PSNR achieved after a different number of rechoke periods. Results are shown in Figure 5.8.

![Figure 5.8: PSNR achieved for different number of rechoke periods](image)

Results clearly show that the proposed algorithm outperforms the other algorithms in terms of video quality. Alternatively, we can observe that the proposed approach requires less number of rechoke periods to achieve a certain level of PSNR, which implies that the proposed algorithm needs less
time for downloading.

5.6.3 Additional video results

The following pictures 5.9 show an example of video sequences received. These first figures show the same frame for the video sequence called Foreman with CIF resolution and the frame rate of 30 frames/second.

The pictures show that using the Foresighted Resource Reciprocation strategy the received video quality is better than the quality received using the tit-for-tat strategy. However, the quality received is very poor for multimedia streaming and in most of the cases the video cannot be reproduced.
continuously. As it is shown in the same figure, the quality of the video sequence received using BiTos and our proposed algorithm is the higher possible quality for that sequence. Nevertheless, our approach gives better performance all over the time and thus we are going to show the results for the last frames of the video sequences.

The pictures at left side show the frame received using our approach and decoded. On the other hand pictures at right side show the frames received using BiTos.

![Figure 5.10: Frame 234 with different strategies](image)

![Figure 5.11: Frame 235 with different strategies](image)

In this case the quality achieved is similar, however for the lasts frames
the video quality achieved using our proposed algorithm is better than the one achieved using BiTos. The reason is that this video sequence has the last frames very similar, as the motion of the last frames is almost fixed. In order to display better the difference between the quality achieved another sequence is used. In fig 5.6.3 the comparison between BiTos and our proposed algorithm is shown for news sequence.

![Frame 240 with different strategies](image)

(a) BiTos  (b) Our proposed algorithm  (c) Original frame

Figure 5.12: Frame 240 with different strategies

The quality of the last frames is now more obvious than the previous one with the other sequence. Moreover, the number of frames decoded in the first case is higher than the second case. The reason is that using BiTos the importance of the I type frames or P frames are not consider and thus, if this last pieces are lost the decoder can not decode the lasts GOP’s.
Chapter 6

Conclusions

In this project we have demonstrate that multimedia streaming over P2P networks can be implemented in the framework of the Foresighted Resource Reciprocation strategy. With minimal changes in the previous implementation [12], we have developed a new strategy that both considers the time priority of the pieces and the importance of the pieces for decoding. Simulation results confirm that our approach outperforms several existing algorithms such as tit-for-tat and BiToS in P2P networks, in terms of packet loss rates and utility. Moreover we have tested our approach with real video files and the results clearly shows that the Received Video Quality (PSNR) is improved for given playback delays.
Appendix A

Video experiments

In this appendix the information about the video experiments is provided. First the video encoding options are explained and then more information about the sequences is presented. Finally additional results obtained using different video sequences are presented in section 5.6.3.

A.1 H.264 Video coding

The results presented in chapter 5 and in the following sections were obtained using the video coding standard H.264, also called MPEG-4 Advanced Video Coding or H.264/AVC. A reference software implementation has been used, which is freely available in [2].

A Group of pictures, GOP, generally use a encoding structure with 3 types of frames. The first picture of the group is an I frame which encoded using intra-prediction only. It is the reference picture, which represents a fixed image and which is independent of other picture types. Each GOP begins
with this type of picture. In general, I frames produce a much larger bit rate. Then, P frames are encoded after forming a motion compensated prediction based on the preceding frame. A P frame contains motion-compensated difference information from the preceding I- or P-frame. B frames are bi-directionally predicted frames, they depend on the neighboring P frames or I frames. This encoding structure is periodically repeated to encode the entire video sequence.

The I-frames contain the full image and do not require any additional information to reconstruct it. Therefore any errors within the GOP structure will propagate until the next I-frame is successfully decoded, since this type of picture does not depend on previously encoded pictures.

The more I-frames the video stream has, the more editable it is. However, having more I-frames increases the stream size. In order to save bandwidth and disk space, videos prepared for internet broadcast often have only one I-frame per GOP. Several encoding structures can be used for multimedia video streaming. The encoding structure used in our experiments is shown in Figure A.1. The I-frame period and GOP length is 32.

The H.264 coding parameters are shown in Table A.1 and the frames’ size of the encoded sequence are shown in Figure A.2.

In Figure A.2 is shown the different size of encoded frames. This has been calculated in order to provide the information to the system and divide the frames in rtp packets with the size of the transmission unit on the network.
Figure A.2: Frames size for encoded sequence

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUV format</td>
<td>4:2:0</td>
</tr>
<tr>
<td>Frames encoded</td>
<td>300</td>
</tr>
<tr>
<td>Image format</td>
<td>352x288</td>
</tr>
<tr>
<td>Search Range</td>
<td>32</td>
</tr>
<tr>
<td>Number of reference frames</td>
<td>1</td>
</tr>
<tr>
<td>Sequence type</td>
<td>I-B-B-B-P</td>
</tr>
<tr>
<td>QP</td>
<td>I 28, P 28, B 30</td>
</tr>
<tr>
<td>Entropy coding method</td>
<td>CABAC</td>
</tr>
</tbody>
</table>

Table A.1: H.264 encoding parameters for encoding structure

Once the sequences are encoded, they are sent using a P2P network with the different algorithms.

After the simulations, the results of packets received by each peer are decoded. As losses cannot be avoided, error concealment is used at the decoder. The technique used on this project is frame copy. This is a very simple technique based on frame repetition which requires no additional computation but does not avoid visual artifacts. Finally, the PSNR values are calculated between the decoded sequence and the original one.
A.2 Video sequences

In this section we are going to provide information about the sequences used in this project. In order to compare the results of different type of video sequences three sequences has been encoded using the parameters described in section A.1.

A.2.1 Foreman sequence

(a) Example picture of Foreman sequence

(b) Frame size for encoded sequence

- **Description:** Sequence captured with a hand-held camera. It shows a man talking and a construction site. In the first part of the sequence motion is due changes in facial expression and camera shaking. In the second part of the video panning is included.

- **Resolution:** CIF $288 \times 352$, 30 frames per second

- **Number of frames:** 297 frames
A.2.2 News sequence

(c) Example picture of news sequence

(d) Frame size for encoded sequence

- **Description:** typical news sequences with a fixed camera. Most of the activity of the sequence is due to a screen in the background displaying a ballet with camera pan.

- **Resolution:** CIF $288 \times 352$, 30 frames per second

- **Number of frames:** 297 frames
A.2.3 Salesman sequence

(e) Example picture of Salesman sequence

(f) Frame size for encoded sequence

- **Description**: Sequence with fixed background that shows a man talking. All the motion is due to the face and hands movement.

- **Resolution**: QCIF $176 \times 188$, 30 frames per second

- **Number of frames**: 498 frames
Bibliography


