

# 國立交通大學

## 多媒體工程研究所

### 碩士論文 (初稿)



基於基因演算法在點對點即時串流系統下之動態最佳化

Genetic Algorithm Based Dynamic Optimization in Peer-to-Peer Live  
Streaming System

研究生：吳彥暉

指導教授：蕭旭峯 教授

中華民國 一〇一年一月

# 基於基因演算法在點對點即時串流系統下之動態最佳化

研究生: 吳彥暉

指導教授: 蕭旭峯

國立交通大學多媒體工程研究所

## 摘要

在點對點網路架構下節點可以從多個來源接收視訊片段，節點如何選擇具有足夠能力的父節點來滿足自身的需求是一項重要問題。過去的演算法大多使用單一的評定方法來挑選父節點群，而這篇論文提出一個可以同時衡量多個因素，並且針對節點的目標需求與評估當時的網路環境，配合基因演算法實施動態最佳化策略。模擬的結果顯示在符合使用者的期望下，其他次要目標也能夠維持一定水準。



# Genetic Algorithm Based Dynamic Optimization in Peer-to-Peer Live Streaming System

Student: Yan-Hui Wu

Advisor: Hsu-Feng Hsiao

Institute of Multimedia Engineering

National Chiao Tung University

## **Abstract**

In Peer-to-Peer network structure, a peer can receive video fragments from several source nodes. It is an important issue that how a peer pick up group of parents to fulfill its demand. Many algorithms in the literature adopt some monotonic criterion to choose parents before. In this paper, we propose an algorithm that integrates genetic algorithm for dynamic optimization with multiple criteria and multiple objectives. The simulations show the proposed algorithm can satisfy the demand/objectives defined by any individual peer.

# Acknowledgement

感謝老師教導與指引讓我能順利完成論文內容，雖然過程中難免遇到不少挫折，但仍不厭其煩地提醒、教誨我應該要注意的方向和求學應當有的正確態度，使我獲益良多。同時也要感謝實驗室的夥伴，在我遇到瓶頸的時候總能適時的給予支持。最後也要感謝家人在背後默默的付出，讓我生活不置匱乏，求學無後顧之憂。謹以此論文獻給所有幫助過我的人。



# Table of Contents

摘要 .....	1
Abstract .....	2
Acknowledgement.....	3
Table of Contents .....	4
List of Tables .....	5
List of Tables .....	6
<b>Chapter 1 Introduction .....</b>	<b>7</b>
1.1 Preface .....	7
1.2 Motivation .....	7
1.3 Research Objectives .....	8
1.4 Outline of the thesis .....	9
<b>Chapter 2 Related Works.....</b>	<b>10</b>
2.1 Preface .....	10
2.2 Optimization .....	10
2.2.1 Single Objective Optimization .....	11
2.2.2 Multiple Objective Optimizations .....	13
2.3 Evaluation of a Peer's Ability in P2P Live Streaming System.....	14
2.4 Bloom Filter.....	15
2.5 Genetic Algorithm .....	19
<b>Chapter 3 Proposed Method .....</b>	<b>20</b>
3.1 Preface .....	20
3.2 Parent Selection .....	20
3.2.1 Weightings of Criteria.....	20
3.2.2 Weightings of Objective .....	24
3.3 Join and Update Scheme.....	27
3.3.1 Join Protocol .....	27
3.3.2 Update Protocol .....	30
<b>Chapter 4 Simulation Result.....</b>	<b>34</b>
4.1 Simulation Surroundings and setup.....	34
4.2 The Same Objective - Emphasize on PABW .....	36
4.3 The Branch of Objectives .....	38
<b>Chapter 5 Conclusions .....</b>	<b>41</b>
<b>Reference .....</b>	<b>42</b>

# List of Table

Figure 2-1: The optimization process that consists of finding solution $x_0$ which provides the maximum of function [13].	11
Figure 2-2: Trying to find the optimum [13].	12
Figure 2-3: Global optimum and local optima [13].	13
Figure 2-4: Bloom Filter and Its Usage.	16
Figure 2-5: False positive happens if an element does not belong to set and its footprint are 1.	16
Figure 2-6: False positive rate of a bloom filter with respect to $m/n$ .	17
Figure 2-7: False positive rate of a bloom filter with respect to $k$ .	18
Figure 3-1: A peer receives bit array from its own parents, insert its own parents in bit array and then combine into one.	23
Figure 3-2: Flow Chat	24
Figure 3-3: Initialize and get list from patch server.	28
Figure 3-4: Send join Message and Receive Response.	29
Figure 3-5: Measure candidate peers, Send join handshake and Receive Response.	30
Figure 3-6: Evaluate objective score, create new criteria and get list from patch server and parents	31
Figure 3-7: Send update Message and Receive Response	32
Figure 3-8: Measure candidate peers, Send update handshake and Receive response.	32
Figure 4-1: Network topology	34
Figure 4-2: Continuous index	36
Figure 4-3: Throughput and Goodput	37
Figure 4-4: Section difference	37
Figure 4-5: Continuous index between difference objectives.	38
Figure 4-6: Section difference between difference objectives.	39
Figure 4-7: Duplicate bit rate between difference objectives.	40

# List of Tables

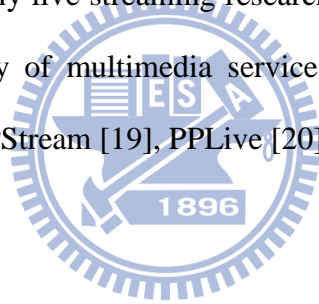
Table 3-1: To convert decimal to binary, and keep the number after binary point. ....	26
Table 3-2: Pick up two chromosomes and do crossover process. ....	26
Table 3-3: Mutation.....	27
Table 4-1: Parameter setup.....	35
Table 4-2: Genetic Algorithm Setup .....	35
Table 4-3: Bloom Filter Setup.....	35



# Chapter 1 Introduction

## 1.1 Preface

With the rapid development of broadband networks, many popular Internet applications have been created [18]. Traditional Client-Server architecture [19] for today's demand has obvious weakness. In order to accommodate the rapidly growing network traffic, it's replaced by Peer-to-Peer in architecture that a peer plays the role of the client and the server simultaneously. It will become more and more important to take advantage of p2p architectures. In recent years, many live streaming researches have appeared one by one [16] [17], and there have been plenty of multimedia services based on P2P technologies; for example: Cool Streaming [13], PPStream [19], PPLive [20], and QQLive [21]...etc.



## 1.2 Motivation

In a P2P live streaming system, the user experience is highly correlated with smoothly playing video in real time. Users may pay more attention to different targets, like Short-delay playback, lower bandwidth consumption or others.

The receiver chooses several peers in the upstream by different methods according to distinct individual experiences, and it further affects whole peer-to-peer network environment. In other words, it is necessary to dynamically change peer's selection policy so as to adapt for fluctuant network.



With this conclusion, we hope we could present a method which can solve the peer selection problem.

## 1.3 Research Objectives

The objectives of our framework can be summarized as the following two parts:

- **Satisfy several competing objectives based on user's expectation**

In the world, a lot of problems concern about several objectives. The peer-to-peer system developed the same condition. Some peer-to-peer systems evaluate a peer's ability according to only one parameter such as available bandwidth, locality, or something else. It possibly reduces the effectiveness by losing other information.

We merge several criteria parameters into a single whole by adding weight to each parameter, the more important target is, the higher weighting it has.

- **Peers track their upstream peer's capacity and adjust its upstream selection policy**

In addition to satisfy many objectives on demand, it should also track the changing optimum over time due to variation in the dynamic environment. However, it is not an easy way to solve dynamic optimization problems directly for large scale system. The reason is that peer-to-peer live streaming system should always works until source terminates and the situation changes all the time. The old solution was found by peer's old selection policy may not suitable for now situation.

## 1.4 Outline of the thesis

The rest of this paper is organized as follows. In Chapter 2 we introduce the related works. The proposed algorithm for peer's join and update scheme, bloom filter and genetic algorithm are presented in Chapter 3. In Chapter 4, we evaluate the performance of our proposed algorithm. Finally, conclusions are drawn in Chapter 5.

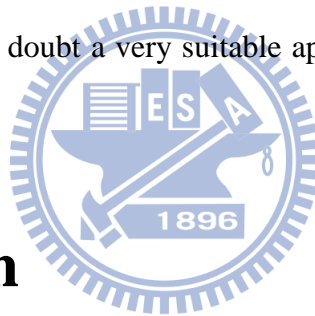


# Chapter 2 Related Works

## 2.1 Preface

All P2P live streaming systems should deal with highly dynamic environments, whose optimal solution may change over time as a result of the changing of objective function, design variable or constraints.

Multi-objective optimization, also known as multi-criteria or multi-attribute optimization, is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. It is without a doubt a very suitable approach for dynamic environments. We introduce them below.



## 2.2 Optimization

An optimization problem is the problem of finding the best solution in all feasible solutions which has the minimum or maximum value of the objective function.

Some problems can be found with a definite answer. In the field of approximation algorithms, the inadequate definitions of problem lead to many acceptable solutions that are generated.

## 2.2.1 Single Objective Optimization

In the simplest case, it means solving problems in which one seeks to minimize or maximize a real function by systematically choosing the values of variables from within a solution space, also known as the search space. The function  $f(x)$  is using a scalar, real-valued objective function, and we want it to find the value  $x$  to yield to the maximum or minimum value of  $f(x)$ . As shown in Figure 2-1.

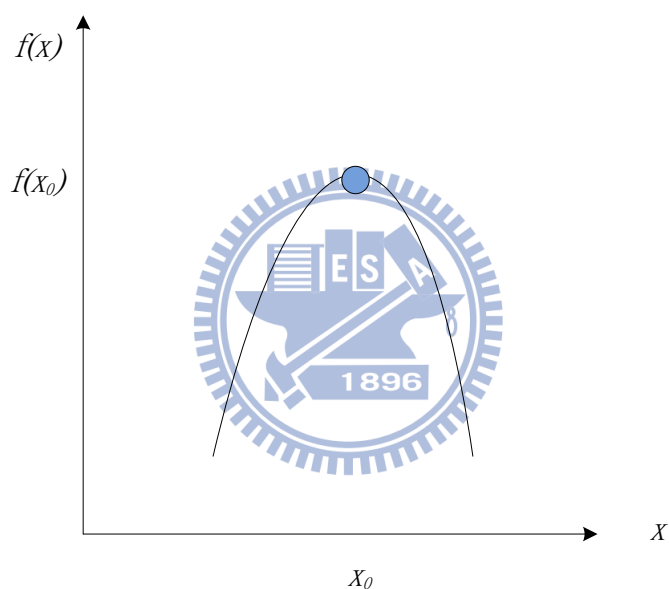


Figure 2-1: The optimization process that consists of finding solution  $x_0$  which provides the maximum of function [13].

Exact and stochastic methods are two types of methods to find the optimum of problem. Exact method guarantees to find optimal value of problem, but it is not applicable when search spaces are large.

Stochastic method [13] does not ensure to find the optimum, but it provides good approximations in a reasonable time. Currently, the most popular algorithms are the

metaheuristic. The idea of metaheuristic is to take a solution and obtain new one by disturbing the current solution somehow. If the new solution is better than current solution, then new one replace old one as current solution. This process repeats until a stopping condition, or the maximum has been found. We can obtain a set of solutions which approximate progressively to the goal, as shown in figure 2-2.

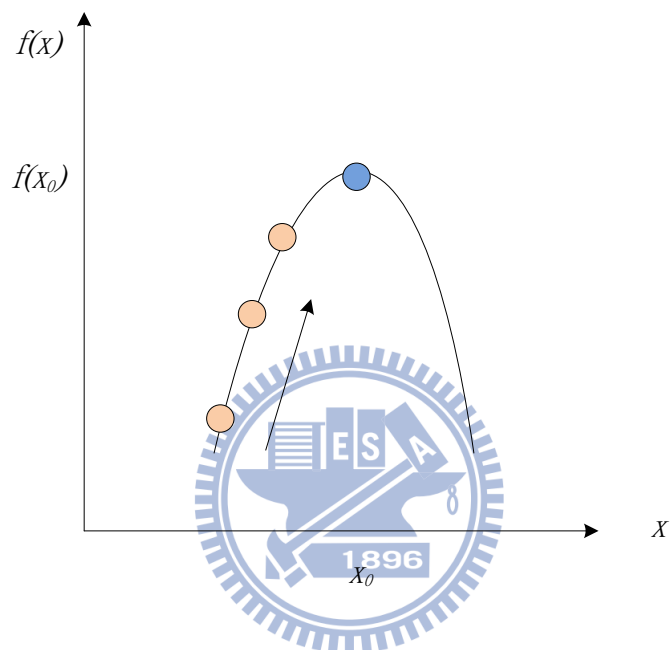


Figure 2-2: Trying to find the optimum [13].

Unfortunately, the function can be formed as the one in figure 2-3.

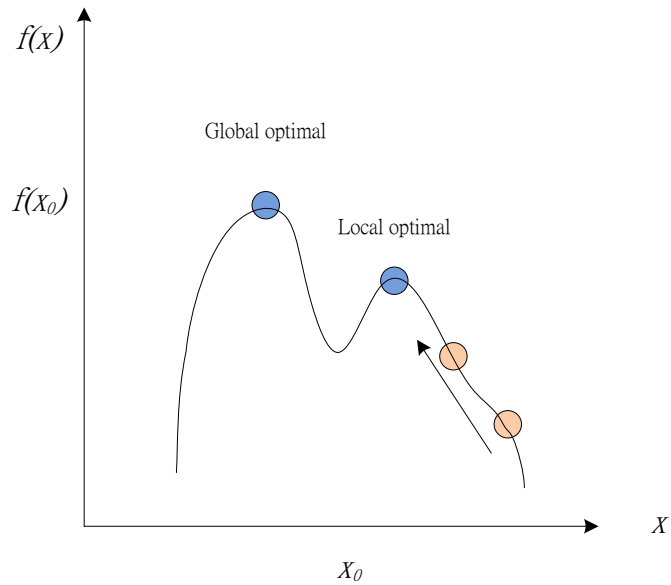


Figure 2-3: Global optimum and local optima [13].

In this case, the function has many local optima. There are many chances to be trapped in local optima. This is a typical condition that could be solved by metaheuristic. There are many metaheuristics, including Evolutionary Algorithm (EA), Particle Swarm Optimization (PSO) [5] [6], and Tabu Search (TS)...etc. The most popular metaheuristics are Genetic Algorithms (GA). Genetic algorithm is a subclass of EA.

Generally speaking, it is rare for any problem to concern only a single objective in the world around us.

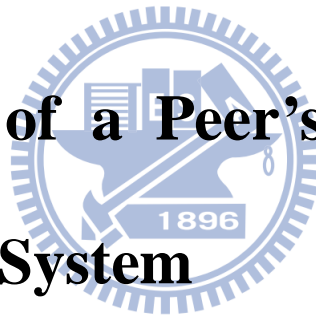
## 2.2.2 Multiple Objective Optimizations

Multi-objective optimization, also known as multi-criteria or multi-attribute optimization, is the process of satisfying more than one function at the same time. Those objectives of functions are in conflict with each other.

A multi-objective problem should not have only one solution that simultaneously minimizes each objective. Instead there is a multitude of alternative solutions, sometime so many as infinite, different balances of the various objectives that all have the same global fitness. We need to find and quantify this solution is much better than many other solutions, is the goal when setting up and solving a multi-objective optimization problem.

There are many methods to finding a solution to a multi-objective optimization problem. According to definition of problem, you even can integrate many algorithms, or cooperate to work each other. In [1], it combines genetic algorithm with goal programming to establish a model for solving the network topology design problem with multiple objectives and multiple criteria.

## **2.3 Evaluation of a Peer's Ability in P2P Live Streaming System**



There are many researches in literature of p2p live streaming systems presented in recent years. They evaluate ability of a peer by different methods, such as below

In Dynamic and Resilient Peer-to-Peer Architecture for Live [11], its multiple factors of parent selection algorithm based on peer lifetime distribution can choose reliable parents. Also it use LT codes [9], one of rateless codes, to reduce data coordinate message efficiently. But the weightings of evaluation process are set for constant value that is not flexibility in dealing with user's demand. Our proposed method is trying to improve this drawback by dynamic optimization.

Anysee [15] introduces an attribute called *LastDelay*, which is the minimal of all source-to-end delays from the current node to the streaming source on different paths. Source writes timestamp into the media block's header. A peer computes the difference of the initial timestamp and arriving timestamp when it receives the media block from source. The minimal difference value will be *LastDelay*, and a peer can join or leave the topology according to *LastDelay*.

In rStream [12], the receiver selects the upstream peer close to the source and the streaming rate is satisfied. It formulates two factors as an objective function to guarantee bandwidth availability and minimize end-to-end latency.

CoolStreaming [13] has partnership manager for establishing and maintaining connection. Every node has a unique identifier, and keeps other nodes' identifier in its membership cache (mCache). A new node contacts the original node first to get random deputy list from original node's mCache, then the new node can obtain a list of partner candidates from the deputy, and contacts these candidates to establish its partners in the overlay. A node calculates a score for its partner by using function  $\max\{\bar{s}_{i,j}, \bar{s}_{j,i}\}$ , where  $\bar{s}_{i,j}$  is the average number of segments that node  $i$  retrieved from node  $j$  per unit time. The new node keeps those nodes with high score as partners.

## 2.4 Bloom Filter

Bloom filter is a hash-based data structure. It can represent a set  $A = \{a_1, a_2, \dots, a_n\}$  of  $n$  elements that supports for element mapping and membership query.

An initial Bloom filter is a bit array (BA) of  $m$  bits, all set to 0. There is also being defined  $k$  independent hash functions,  $H = \{h_1, h_2, \dots, h_k\}$  each range  $\{0, 1, \dots, m - 1\}$  with



property of uniform distribution. For element  $a_i \in A$  sets  $k$  bits to 1 at positions  $\{h_1(a_i), h_2(a_i), \dots, h_k(a_i)\}$  in the bit array. For the convenience, we call the set  $\{h_1(a_i), h_2(a_i), \dots, h_k(a_i)\}$  as  $a_i$ 's footprint [2]. If we want to check an unknown element  $x$ 's membership, we check its footprints in the bit array. If there is any 0 bit in its footprints,  $x \notin A$ . If all  $k$  bits set to 1,  $x \in A$ .

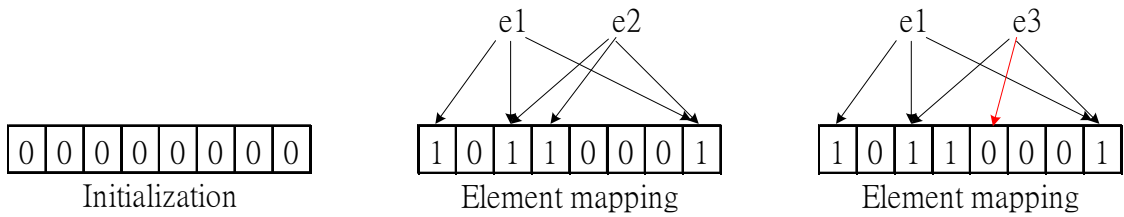


Figure 2-4: Bloom Filter and Its Usage.

Bloom filter can indicate the membership in  $O(k)$  time and has good efficiency in respect of memory space utilization. However, there is a certain probability that an element  $x \notin A$ , if its footprints are 1, called false positive. As shown in Figure 2-5.

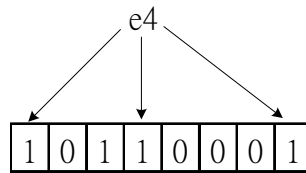


Figure 2-5: False positive happens if an element does not belong to set and its footprint are 1.

The probability of a false positive has been dissected by Border *et al* [3]. The equation is  $P_{fp} = (1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-\frac{kn}{m}})$ , where  $k$  is number of hash functions,  $n$  is the number of inserted elements, and  $m$  is the number of bits in the array. From this equation, we

know the probability is a tradeoff between the number of bits per element  $\frac{m}{n}$  and the number of hash function in use  $k$ .

Figure 2-6 and figure 2-7 show the influence of  $\frac{m}{n}$  and  $k$  on false positive rate; we use two large word lists  $M$  and  $Q$  for mapping and querying respectively. There are approximately 18000 words in  $M$  and  $Q$ , and  $M$  and  $Q$  are totally different. After mapping 2000 elements  $e \in M$  into bit array, we count false positive rate right away. If element's footprint is 1 when we do query procedure, it must be false positive due to the reason that  $Q$  is different from  $M$ .

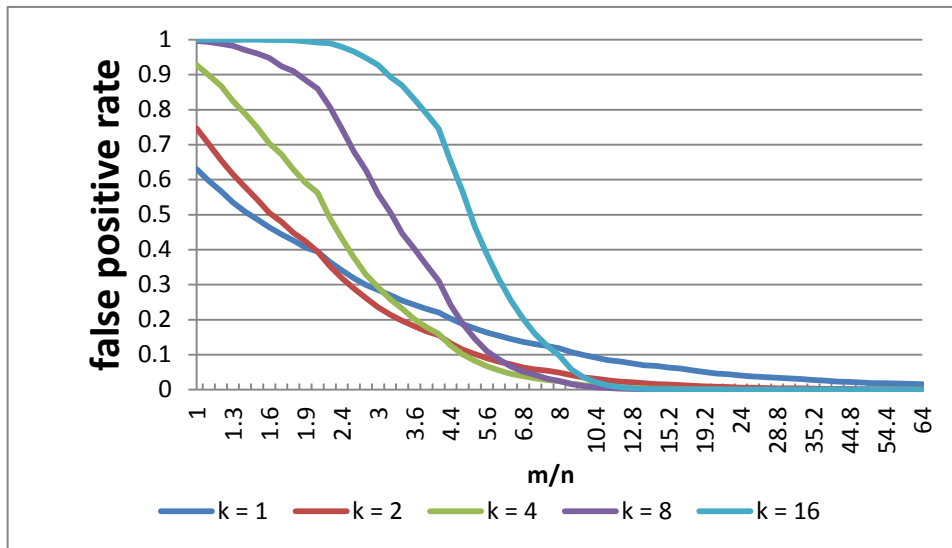


Figure 2-6: False positive rate of a bloom filter with respect to  $m/n$ .

Figure 2-6 shows relationship between false positive rate and  $m/n$ . For any  $k$ , the false positive rate decreases as bit array size becomes bigger. Assume every hash function selects each array position with the same probability  $p = \frac{1}{m}$ . Then the array position which is not selected to set to 1 with probability  $(1 - p) = (1 - \frac{1}{m})$ . There are  $k$  hash functions. The probability of array position which is not set by any hash function is  $(1 - \frac{1}{m})^k$ . If we insert  $n$  elements, the probability of array position being still 0 is  $(1 - \frac{1}{m})^{kn}$ . Therefore, the

probability of array position which is set to 1 is  $1 - (1 - \frac{1}{m})^{kn}$ . To query membership of an element, we use hash functions which are the same as what mapping procedure uses. False positive happened when an element is not in the set, but its footprint all are 1. The false positive rate can be given as  $(1 - (1 - \frac{1}{m})^{kn})^k$ . Also, it is approximate  $(1 - e^{-\frac{kn}{m}})^k$  due to  $e = \lim_{m \rightarrow \infty} (1 + \frac{1}{m})^m$ .

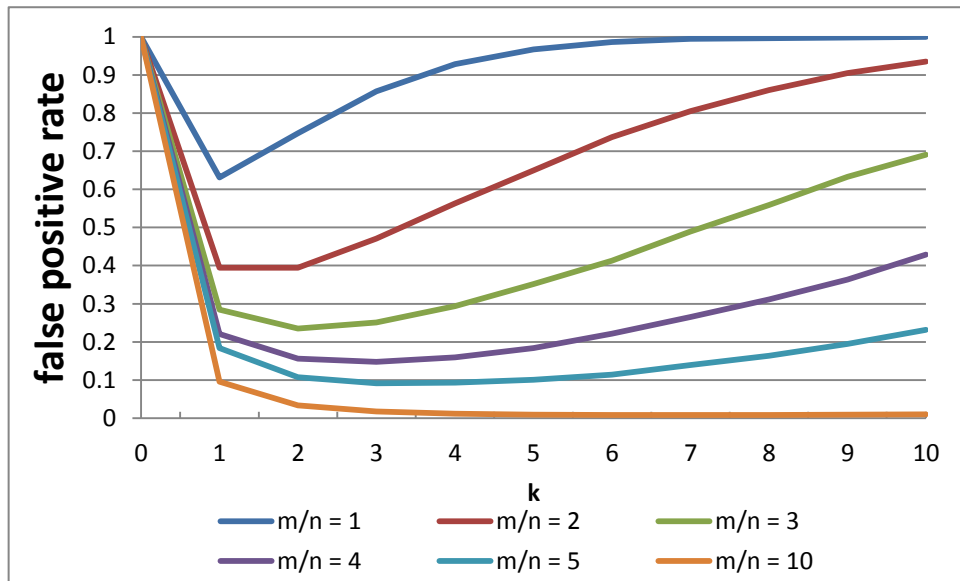


Figure 2-7: False positive rate of a bloom filter with respect to  $k$ .

For a given  $m/n$ , figure 2-7 shows different  $k$  impact on the false positive rate. It is more complex, but there is an optimal value of  $k$  for a given  $m/n$ . When  $k = \ln 2 \times \frac{m}{n}$ , the false positive rate reaches its minimum at  $0.6185^{\frac{m}{n}}$  [3].

Bloom filter offers god space efficiency with the cost of reduced accuracy, and has found numerous applications [2] [4] in a wide range, especially where a large amount of data is involved.

## 2.5 Genetic Algorithm

In 1970s, John Holland developed the genetic algorithm (GA), which is a stochastic searching method for optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [7] [8].

In the genetic algorithm, the candidate solutions are called chromosome (or individual) which are encoded as a population of strings. Traditionally, all possible solutions, called chromosomes, are represented in binary as strings of 0's and 1's in the search space; also, it can be expressed as different form. Each chromosome has an associated objective function value, fitness. A good chromosome has a high or low fitness value depending upon the type of problems, and fitness represents how good potential of being carried to the next generation.

The four main steps in GA are:

- Selection/reproduction: The process chose a good chromosome base on its probability from the current generation to be carried to the next generation, and the probability is proportional to the fitness.
- Crossover: The process chooses two random chromosomes to generate new offspring.
- Mutation: It is an important part of the genetic algorithm that prevents the chromosome was trapped in any local optima.
- Evaluation: The process computes fitness value by objective function.

# Chapter 3 Proposed Method

## 3.1 Preface

In [11], peers exchange message with others, and calculate the lost probability of their parents. But it cannot dynamically change their parent selection policy in time.

In this chapter, we will describe our proposed method in detail. In section 3.2, we explain bloom filter and genetic algorithm first that we use in the method. The detail of our algorithm will be presented in the following section. It contain peer's join and update scheme.

According to user's objectives, we aim to pick up a group of parents based on different network condition so as to make good impression.



## 3.2 Parent Selection

### 3.2.1 Weightings of Criteria

In [11], it uses three factors to measure a peer's ability. The three factors are including PABW (Path Available Bandwidth), ER (Effective Ratio), and FL (Fresh Level). Here we take PABW and FL into criteria of parent selection according to our requirement, where a peer can get candidate's information immediately without gathering. For example, ER is the ratio of the effective data bit rate from a parent to the request bit rate toward the parent. Obviously, it needs time to collect data and then calculates the ratio.

In fact, the distribution of a peer's lifetime in peer-to-peer system trends to be heavy-tailed distribution. It means that peers have already survived in the system for some time will likely remain online for longer periods of time than arriving peers. There are many researches about peer lifetime distribution. In order to fit for discrete event simulator NS2 which is targeted at networking research, [11] modifies continuous probability distribution into discrete probability distribution.  $L_j(t_j)$  represents the probability that peer  $j$  will keep alive until peer  $i$  leaves the system.  $D_i(t_i)$  represents the probability that peer  $i$  survives next  $s$  seconds after it has lived  $t_i$  seconds.

#### **A. Path Available Bandwidth (PABW)**

Path Available Bandwidth (PABW) [11] factor is minimum supply bandwidth of a peer in the future. If peer  $i$  wants to calculate PABW value itself, it can use the equations below.

$$SPABW(i) = \sum_{j \in i's \text{ parent}} (PABW(j) \times L_j(t_j)) \quad (1)$$

$$PABW(i) = \min(SPABW(i), ABW(i) \times D_i(t_i)) \quad (2)$$

$SPABW(i)$  is the total bandwidth that a peer  $i$  will receive from its parents in the future.

$ABW(i)$  is available bandwidth of peer  $i$ . Then, peer  $i$  chooses the minimum between  $SPABW(i)$  and  $ABW(i) \times D_i(t_i)$  as its PABW.

#### **B. Fresh Level (FL)**

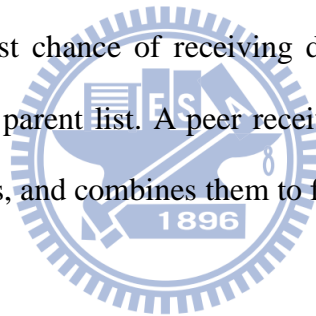
Fresh level (FL) is the delay relative to the source node of a peer. In [11], fresh level value can be calculated by using the time stamp from source. Peer records fresh level of all parents, and chooses the maximal one as its fresh level. It does not adapt to LT codes because LT codes is one of rateless fountain codes. Not only source can generate coded data, but also

other peers which already restore original message. It will make incorrect fresh level value if coded data and time stamp are made by peers.

Now we calculate fresh level value by difference of sections of coded data between peer and source. It is clearer and easier than the old one, and they all represent the delay relative to source node of a peer.

### ***C. Hamming Distance Ratio (HDR)***

Hamming distance ratio represents the dissimilarity of path among peers. A peer receives data from its group of parents. If peers have the same group of parents, they get the same data with high probability. In a similar way, a peer receives the data packets that pass through the common paths; it may have worst chance of receiving different data packets. A peer uses bloom filter for keeping track of parent list. A peer receives other bit arrays (BA) resulting from bloom filters from its parents, and combines them to form a new one.



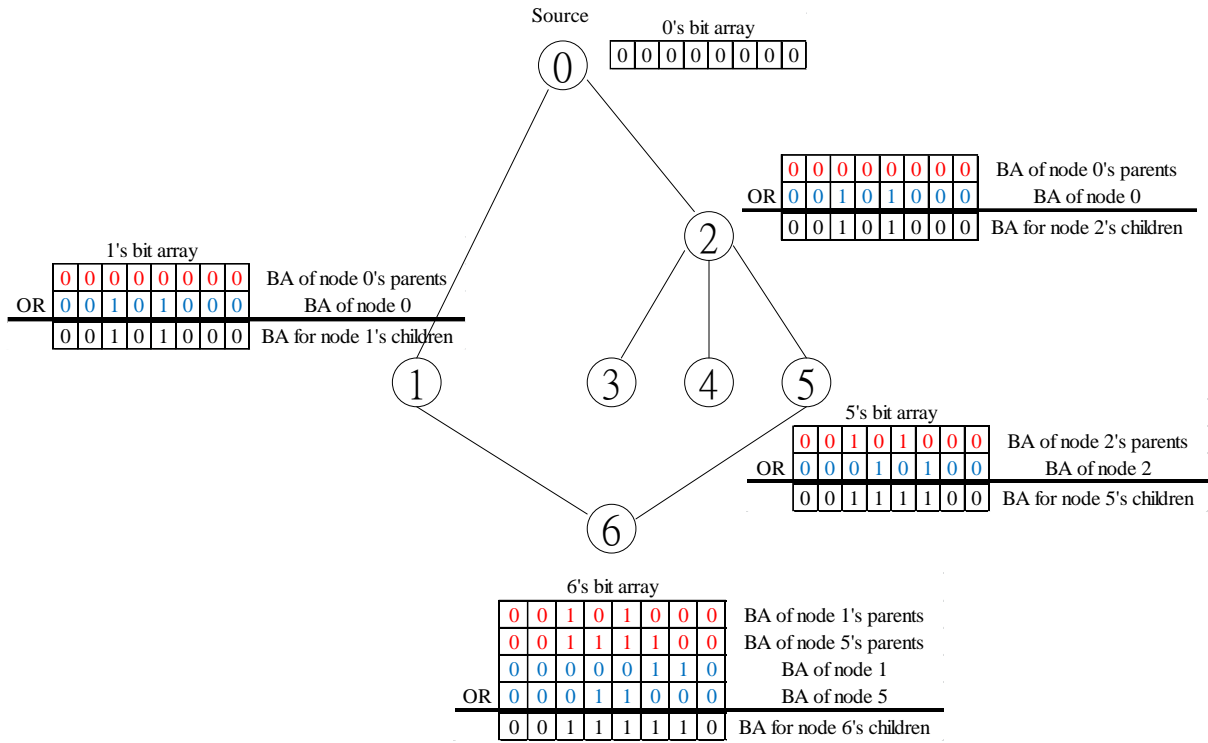


Figure 3-1: A peer receives bit array from its own parents, insert its own parents in bit array and then combine into one.

This new bloom filter contains all paths where the packets could come from. So the equation of hamming distance ratio is

$$1 - \frac{\sum_{j=1}^L A_j}{L} \quad \text{If a peer doesn't have any parent} \quad (1)$$

$$\frac{\sum_{j=1}^L (A_j \oplus B_j)}{L}, \text{ other else} \quad (2)$$

$A(B)$  is bloom filter, and  $A_j(B_j)$  is positions  $j$  in  $A(B)$ .  $L$  is length of bloom filter. When peer chooses first parent, there is no member in bloom filter, so we take into accounting candidate parent as near as source.



The criteria score (CS) of a parent candidate  $P_i$  can be expressed as:

$$CS(i) = w_1 * PABW(i) + w_2 * FL(i) + w_3 * HDR(i) \quad (3)$$

, where  $w_1, w_2$  and  $w_3$  are weighting factors, and  $w_1 + w_2 + w_3 = 1$ .

### 3.2.2 Weightings of Objective

When a peer joins live streaming system, it can set its weightings of criteria according to its own objective. Path available bandwidth can reflect a peer's stability of data supply. Fresh level factor can reflect data latency. Hamming distance ratio can reflect dissimilarity of path and decrease the duplicate of data.

As time goes by the environment of live streaming system will change, and old weightings of criteria may not adapt to new circumstance. So we need to adjust weightings of criteria dynamically by genetic algorithm.

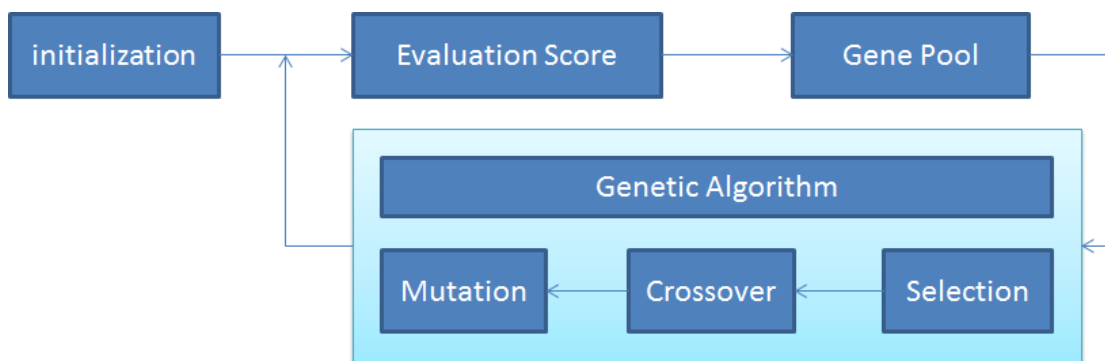


Figure 3-2: Flow Chat

For the purpose of evaluating performance of parents, we use weightings of objective and two factors, Continuous Index (CI) and Section Delay (SD).

#### ***D. Conscious Index (CI)***

Continuous index is trying to show how smooth on streaming playback. It is the ratio of success for playback in a period. In user's opinion, a good experience means smooth and uninterrupted playback.

#### ***E. Section Delay (SD)***

Section delay represents the average delay relative to the parent of a peer. A peer keeps the difference value of section id between the peer and its parent, and also keeps maximal difference value of section id. A peer decides the start section after join or update procedure. The start section is according to the section that most parents receive. As similar as criteria score, objective score (OS) of parent can be expressed as:

$$OS(i) = u_1 * CI(i) + u_2 * SD(i) \quad (4)$$

, where  $u_1$  and  $u_2$  are weighting factors, and  $u_1 + u_2 = 1$ .

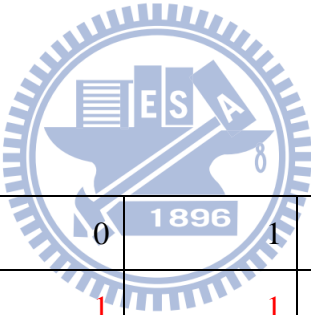
The weightings of objective are set according to user's objective. Every once in a while a peer needs to gather statistics about continuous index and section delay; then gets objective score and puts it into gene pool with weightings of objective. For convenience, we call weightings in the gene pool chromosome.

Before creating new chromosome, weightings of criteria were converted to binary string first. The number to the right of decimal point is represented in binary as string of 0s and 1s. Let length of chromosome is L, if length of binary string is less than L, the remainder of string are set as 0s. If length of binary of binary string is larger than L, the part of over length is deserted.

	$w_1$	$w_2$	$w_3$
Decimal	0.625	0.125	0.25
Binary	$0.10100000_2$	$0.00100000_2$	$0.01000000_2$
Chromosome	10100000	00100000	01000000

Table 3-1: To convert decimal to binary, and keep the number after binary point.

Now we want to create new chromosome for measuring parent candidate next time; we choose two chromosomes from gene pool based on their objective score and have them crossovered. If chromosome has higher objective score, it will remain more genes in the offspring.



						OS
Chromosome 1	0	0	1	1	1	0.6
Chromosome 2	1	1	1	0	0	0.8
Child	1	1	1	0	1	

Table 3-2: Pick up two chromosomes and do crossover process.

The new chromosome still has one step to do, called mutation. To prevent chromosome may trap into local optimum zone, every genes have to mutate based on probability correspond to its position, and the most important position will have lower probability of mutation. The first two figures of string have the same probability, so that total probability is equal to 1.

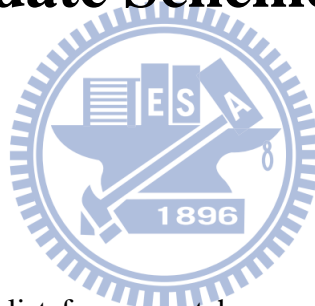
Probability	1/16	1/16	1/8	1/4	1/2
Child	1	1	1	0	1
Mutated child	1	0	1	1	1

Table 3-3: Mutation

After mutating, the new chromosome will be new weightings of criteria; a peer uses it to measure ability of parent candidate. A peer will repeat those steps until leaving system.

## 3.3 Join and Update Scheme

### 3.3.1 Join Protocol



1. Peer X asks a random peer list from a patch server when joining into the streaming system. The patch server records partial active peers in the system, and it replies random list of active peers if it receives a request from a peer.

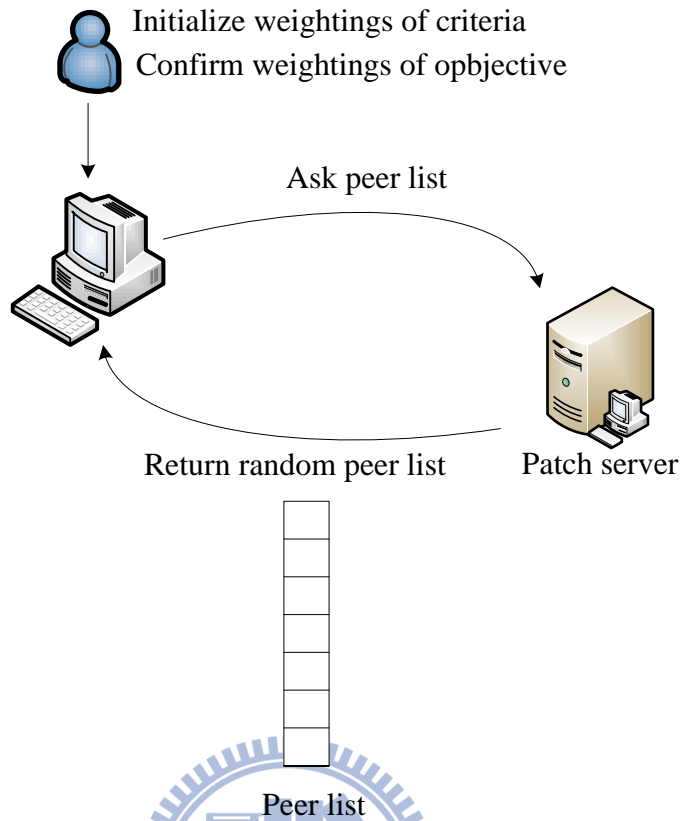


Figure 3-3: Initialize and get list from patch server.

2. After getting list of peers, peer X sends out join message to those peers on the list. If those peers receive a message from peer X, they will reply their evaluation factors and peer list which they connect with. Peer X acquires information in a period of time.

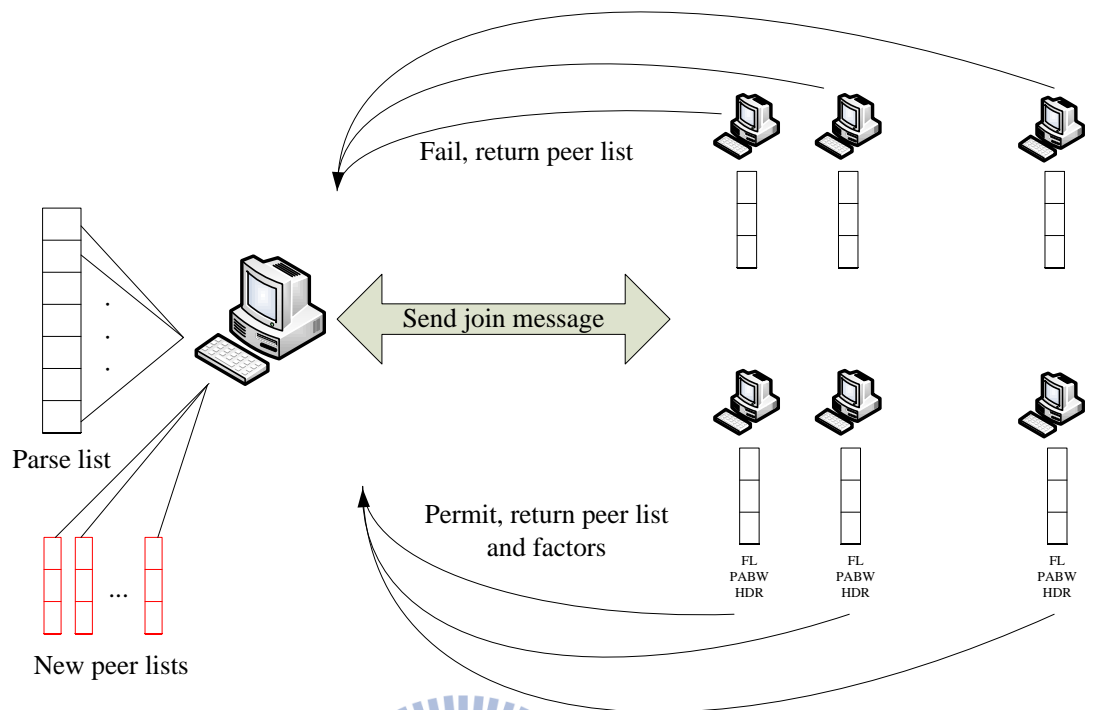


Figure 3-4: Send join Message and Receive Response.

3. Peer X does criteria score process which we introduce above. It ranks the scores and decides a candidate peer, then peer X's bit array combines with candidate peer's bit array to form new one to replace old peer X's bit array. It sends out join handshake message to the candidate peer.
4. Peer X receives accept/refuse message from candidate peers. If receiving refuse message, peer X needs to update its bit array. When summation of criteria scores of parents exceeds the threshold, peer X stops sending out join handshake message and finishes the join process.

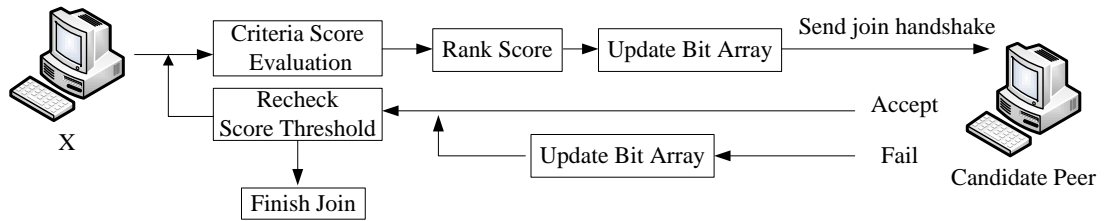


Figure 3-5: Measure candidate peers, Send join handshake and Receive Response.

### 3.3.2 Update Protocol

Peers should do update procedure to reduce the impact if its parents leave or supply lower quality of data. A peer checks the status of its parents in a fixed period of time, and replaces some of parents that do not fit into the requirement of the peer.

1. Peer X calculates objective score, creates new weightings of criteria, and asks a random peer list from patch server and updates its parent list about their evaluation factors and peer list which they connected with.

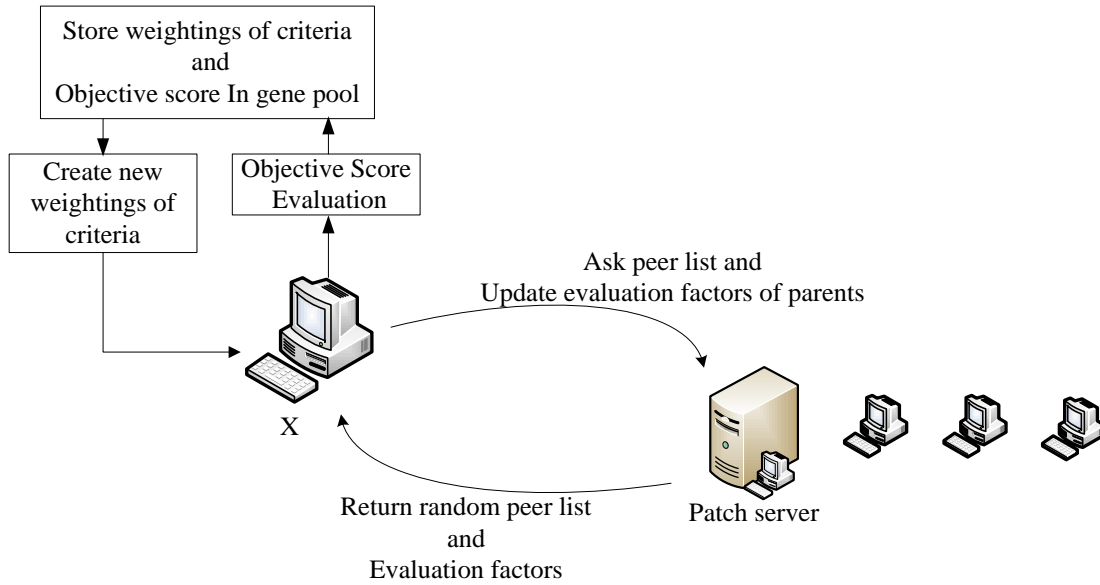


Figure 3-6: Evaluate objective score, create new criteria and get list from patch server and parents

2. After getting list of peers, peer X sends out update message to those peers on the list. If those peers receive a message from peer X, they will reply their evaluation factors and peer list which they connected with. Peer X acquires information in a period of time.



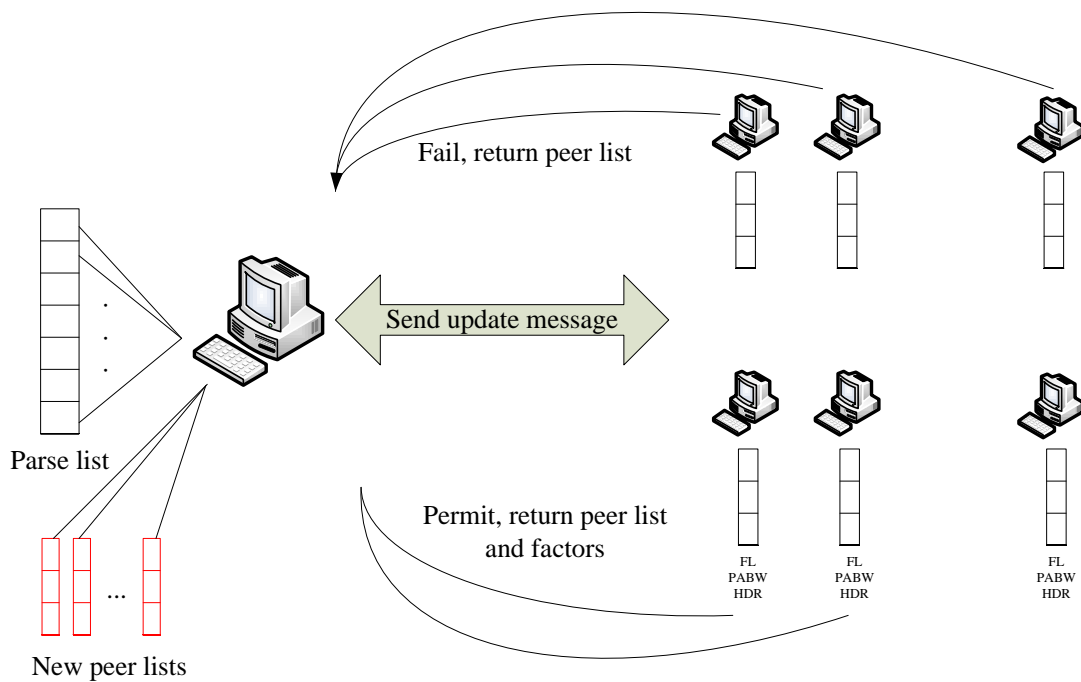


Figure 3-7: Send update Message and Receive Response

- Peer X does criteria score process which we introduce above. It ranks the score and decides a candidate peer, then peer X's bit array is combined with candidate peer's bit array to form new one to replace old peer X's bit array. It sends out update handshake message to the candidate peer.

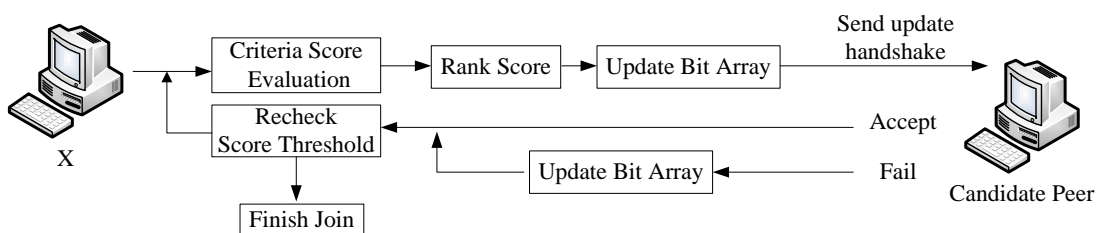


Figure 3-8: Measure candidate peers, Send update handshake and Receive response.

- Peer X receives accept/refuse message from candidate peers. If receiving refuse message, peer X needs to update its bit array. When summation of criteria scores of parents exceed

the threshold, peer X stops sending out update handshake message and finishes the update process.



# Chapter 4 Simulation Result

## 4.1 Simulation Surroundings and setup

We will present results of the experiment in this chapter. Our simulated platform is NS-2

2.34. Network topology is generated by BRITE topology generator.

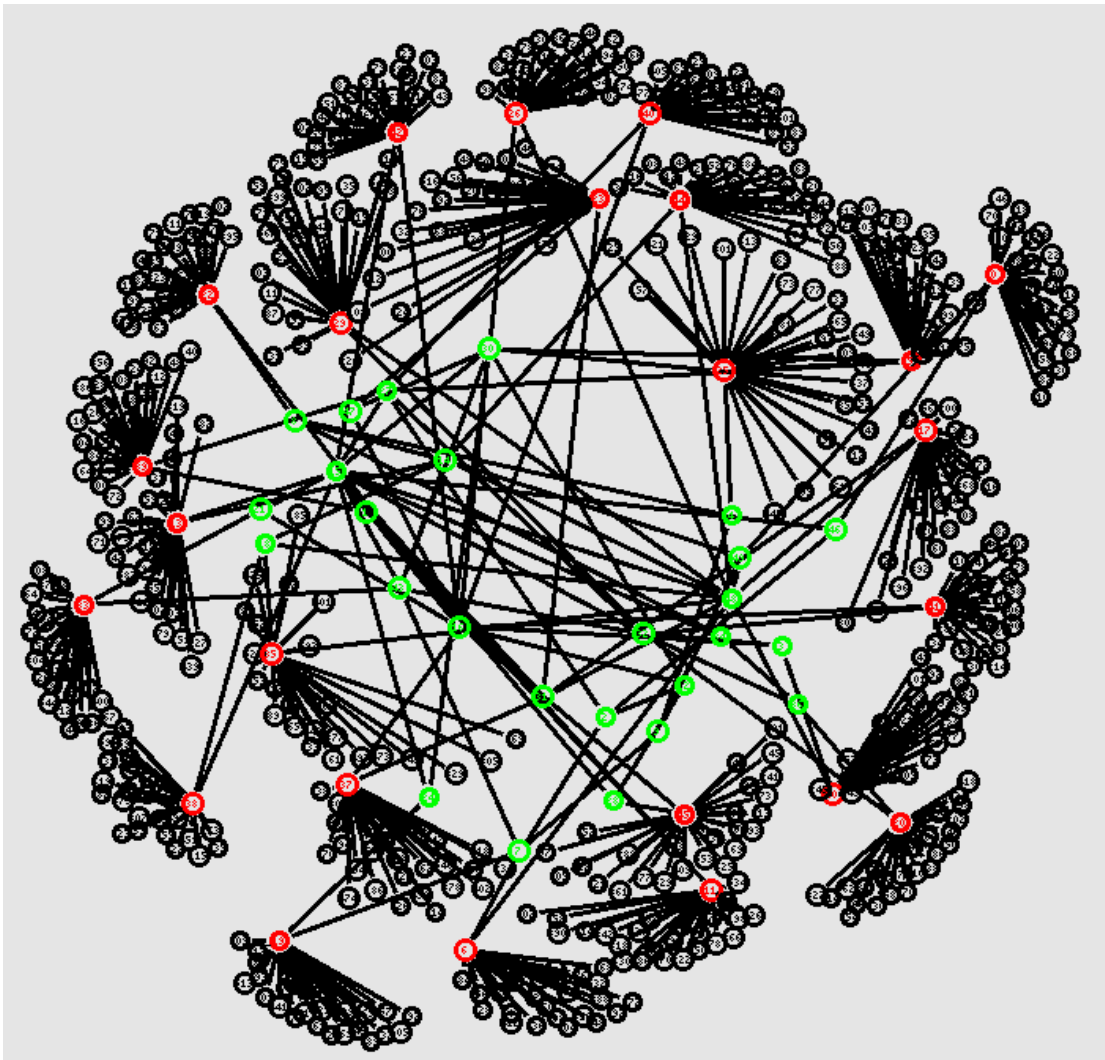


Figure 4-1: Network topology

As figure 4-1 shows, the green nodes and the red nodes are generated by BRITE, and we put ours nodes on the red nodes.

Pareto Sampling (s)	2 seconds
Peer upload/download capacity	1024 Kbps
Source upload capacity	10 Mbps
Join/Update Probing Time	1 second
Score Threshold	8
Update time period	5 seconds
LT block size	1024 Bytes
LT section size	128 K Bytes
Target bit rate	512 Kbps
Buffer size	2 sections
Simulation Time	800 seconds

Table 4-1: Parameter setup

Gene pool size	10
Length per chromosome	10
Force update time period	20 seconds

Table 4-2: Genetic Algorithm Setup

Bloom filter size	7992 bits
Number of hash functions	9

Table 4-3: Bloom Filter Setup

## 4.2 The Same Objective - Emphasize on PABW

In order to check up the difference between our algorithm and [11], we let all peers have the same objective by set  $(u_1, u_2)$  as  $(0.5, 0.5)$ , and  $(w_1, w_2, w_3)$  as  $(1/6, 2/3, 1/6)$ . In [11], it gets the better simulation results by setting its weighting as  $(1/6, 2/3, 1/6)$  based on its formula.

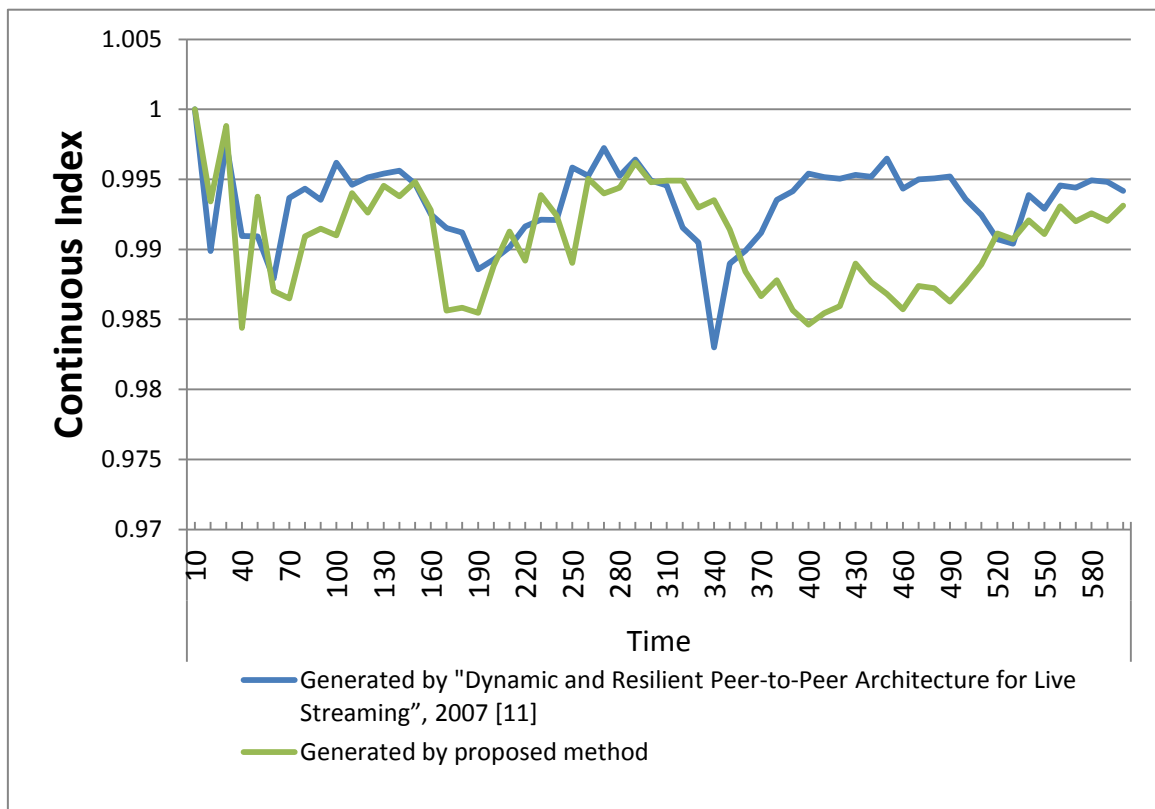


Figure 4-2: Continuous index

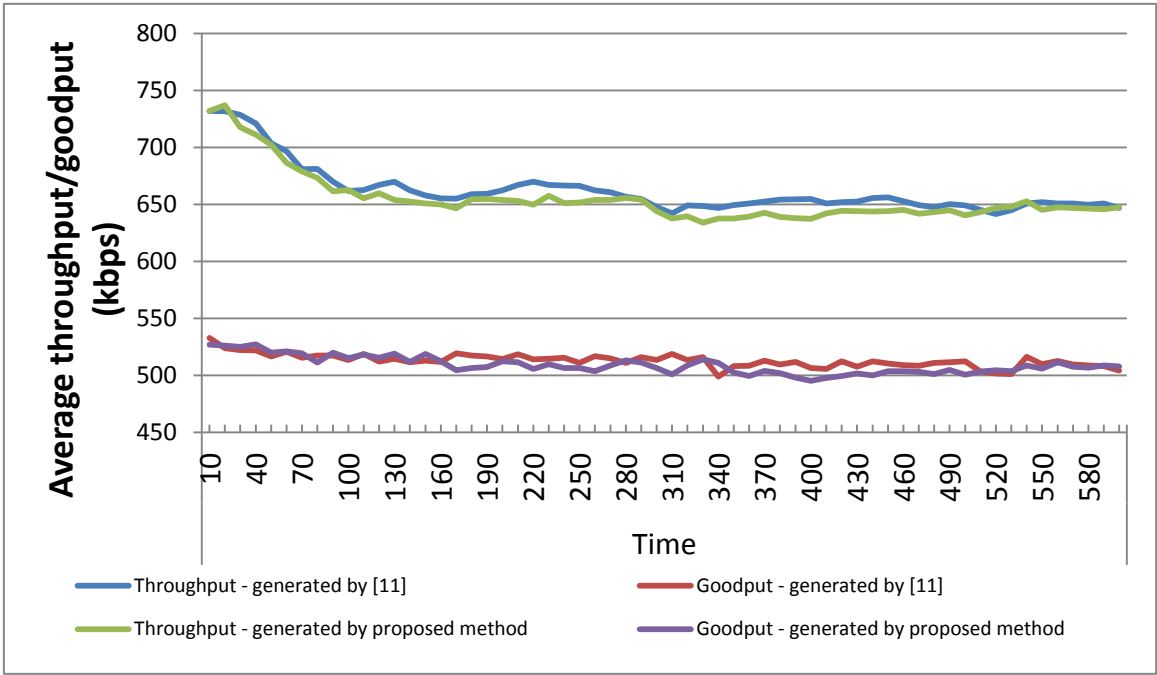


Figure 4-3: Throughput and Goodput

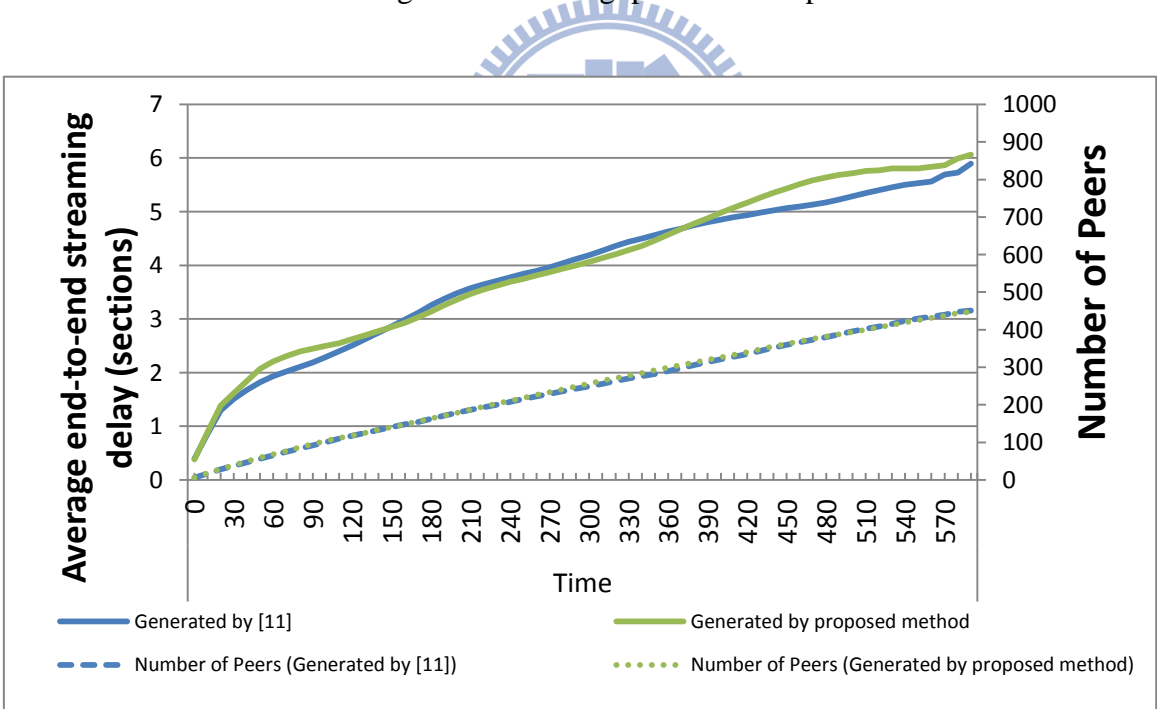


Figure 4-4: Section difference

Figure 4-5, 4-6 and 4-7 show the system performance. Our algorithm makes the effective throughput and the decoding rates are close to [11].

## 4.3 The Branch of Objectives

We decide two objectives to compare, “Proposed – Continuous Index” totally emphasizes on smooth playback, and “Proposed – Synchronization of Peers” pays attention to synchronization between a peer and its parents.

In order to get better convergence efficiency, we initialize several sets of weightings of criteria in a peer’s gene pool when it joins system. According to formula (3), “Proposed – Continuous Index” basically sets  $(w_1, w_2, w_3)$  as  $(3/2, 1/6, 1/6)$ , and “Proposed – Synchronization of Peers” basically sets  $(w_1, w_2, w_3)$  as  $(1/6, 2/3, 1/6)$ . After setting, we do mutation process as table 3-3 three times and then store them in gene pool, so that we shall get many sets of weighting of criteria.

We group 300 nodes into “Proposed – Continuous Index”, and group other 300 nodes into “Proposed – Synchronization of Peers” every simulation.

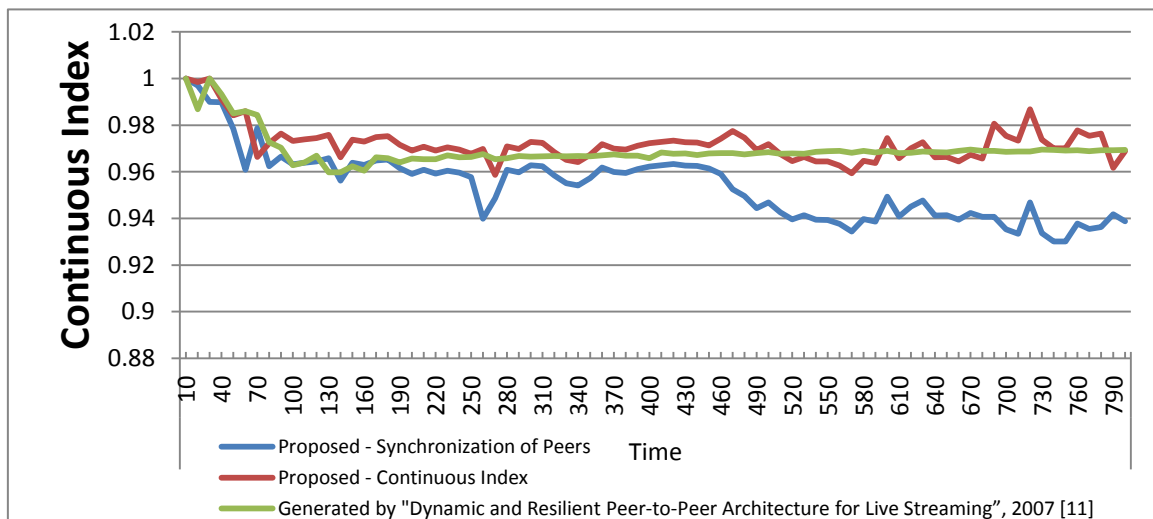


Figure 4-5: Continuous index between difference objectives.

The figure 4-2 is the simulation result of continuous index. It shows peers work smoothly than others if they emphasize with playback smooth. Because they have many chances to receive stable throughput from parents.

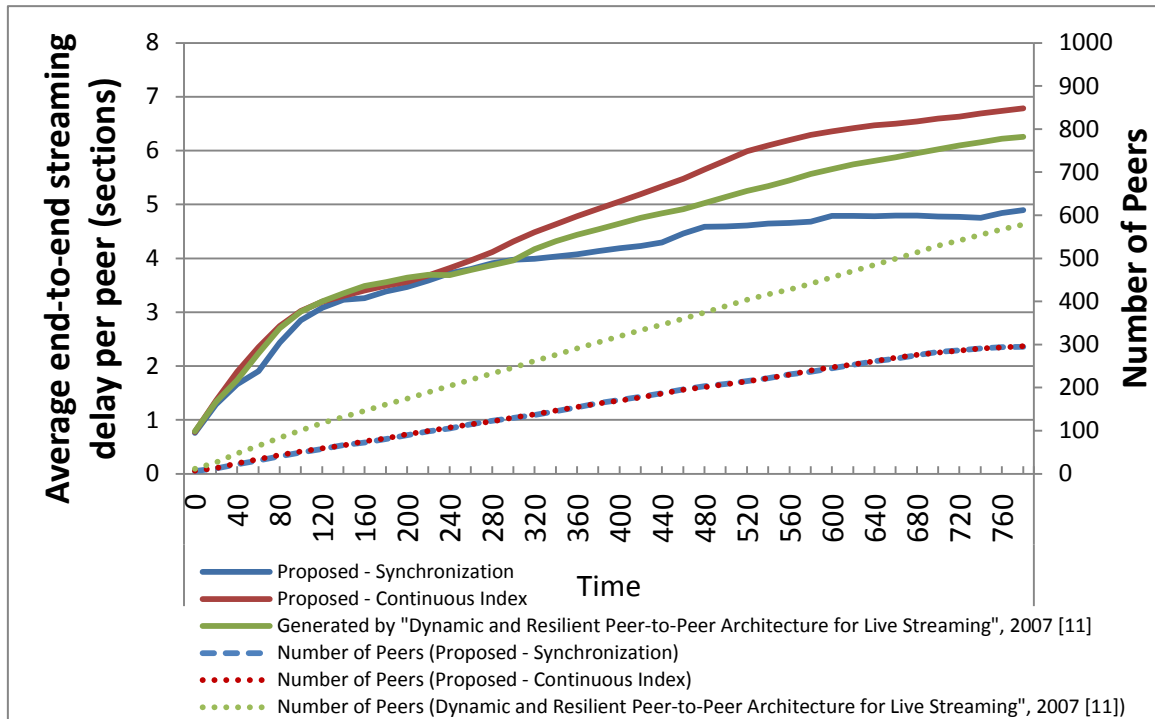


Figure 4-6: Section difference between difference objectives.



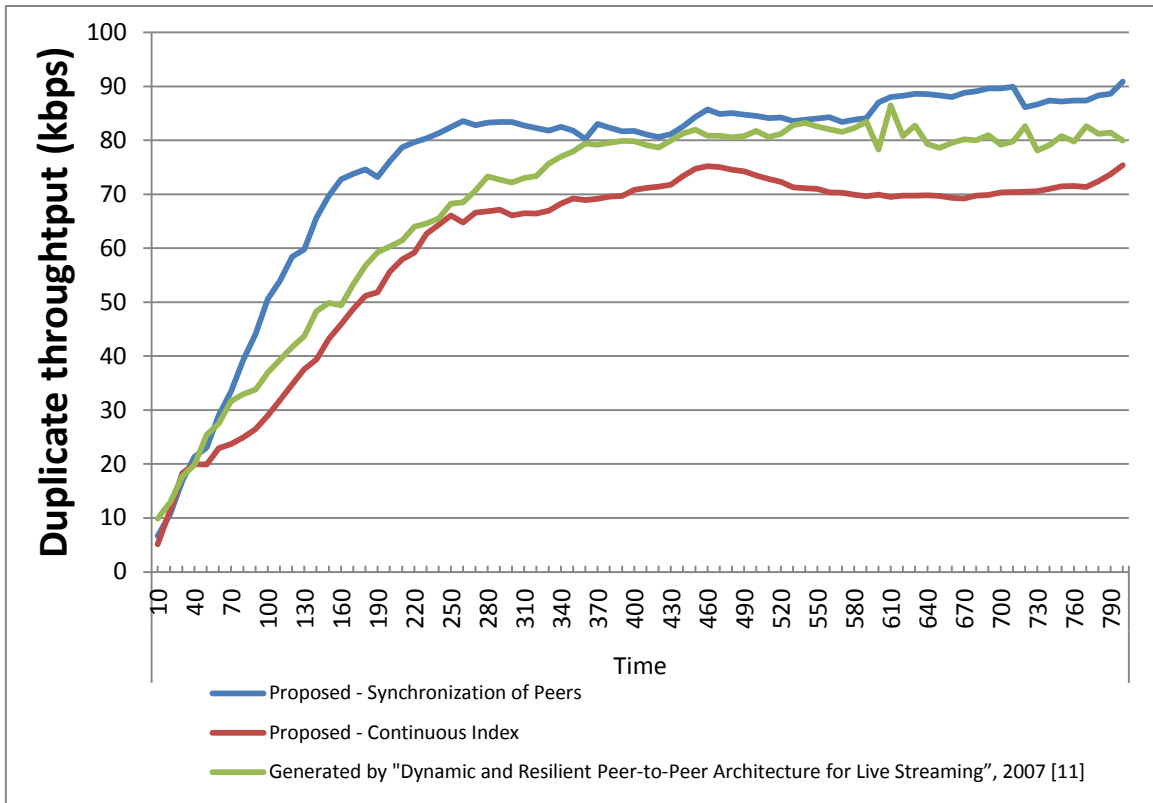


Figure 4-7: Duplicate bit rate between difference objectives.

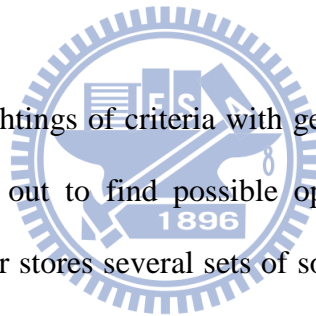
The figure 4-3 it's the simulation result of average end-to-end streaming delay. It is obviously "Proposed – Synchronization of Peers" can make smaller section delay than others. Of course, it cost a little bit worse of playback, their parents have section data as close as source has but may not much in their transfer buffers, so that peers receive duplicate more often than not. Even though they are in unfavorable conditions, their average of continuous index still can keep approximately 0.9. It still can be tolerated.

# Chapter 5 Conclusions

Our algorithm allows user to decide their objective and keeps good performance with other objectives that they compete each other. At the same time, a peer cooperate update process with genetic algorithm and counts upstream peer's ability by formula of objective as well.

We have three factors to measure a peer. The path available bandwidth factor can promise data bit rate of receivers on aggregate. The fresh level factor makes select latest data. The hamming distance ratio factor chooses dissimilar path that can reduce duplicate data bit rate of receivers.

To dynamically change weightings of criteria with genetic algorithm is a good practical policy that is helpful for trying out to find possible optimal weightings, and preventing trapping into local optimal. A peer stores several sets of solutions of multiple criteria in gene pool that it used before. Gene pool can help direct a peer to create a new solution without going the wrong way.



# Reference

- [1] C. S. Wang and C. T. Chang, "Integrated genetic algorithm and goal programming for network topology design problem with multiple objectives and multiple criteria", *IEEE/ACM Trans. on Networking*, vol. 16, no. 3, Jun. 2008.
- [2] H. Cai, P. Ge and J. Wang, "Applications of Bloom Filters in Peer-to-peer Systems: Issues and Questions", *International Conf. on Networking, Architecture, and Storage*, pp. 97-103, 2008.
- [3] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey", *Internet Math.*, vol. 1, no. 4, pp. 485-509, 2003.
- [4] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes and R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data", *ACM Trans. on Computer Systems (TOCS)*, vol. 26 , no. 2, Jun. 2008.
- [5] M. Clerc and J. Kennedy, "The particle swarm: Explosion, stability and convergence in a multidimensional complex space", *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.
- [6] R. Poli, J. Kennedy and T. Blackwell, "Particle swarm optimization An overview", *Springer Science + Business Media, LLC*, vol. 1, no. 1, pp. 33-57, 2007.
- [7] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Reading, MA: Addison-Wesley, 1989.
- [8] C. A. Coello, "An Updated Survey of GA-Based Multiobjective Optimization Techniques", *ACM Computing Surveys*, vol. 32, no. 2, pp.109-143, Jun. 2000.
- [9] M. Luby, "LT Codes", in *Proc. 43rd Symp. Foundations of Computer Science*, pp.271-280, 2002.

- [10] D. Leonard, V. Rai and D. Loguinov, "On Lifetime-Based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks", *IEEE/ACM Trans. on Networking*, vol. 15, no. 3, pp.644, Jun. 2007.
- [11] C. W. Fan-Chiang and H. F. Hsiao, "Dynamic and Resilient Peer-to-Peer Architecture for Live Streaming", 2009
- [12] C. Wu and B. Li, "rStream: Resilient and Optimal Peer-to-Peer Streaming with Rateless Codes", *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, pp. 77-92, 2008.
- [13] 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 *Proc. IEEE INFOCOM 2005. 24th Annual Joint Conf. of the IEEE Computer and Communications Societies*, vol. 3, pp. 2102-2111, Mar. 2005
- [14] Principles of heuristic optimization: <http://www.mm.helsinki.fi/kurssi/marv/MSUU14/Heuristic.pdf>
- [15] X. Liao, H. Jin, Y. Liu, L. M. Ni and D. Deng, "AnySee: Peer-to-Peer Live Streaming", in *Proc. INFOCOM 2006. 25th IEEE International Conference on Computer Communications*, pp. 1-10, Apr. 2006.
- [16] F. Pianese, D. Perino, J. Keller and E. W. Biersack, "PULSE: An Adaptive, Incentive-Based, Unstructured P2P Live Streaming System", *IEEE Trans. on Multimedia*, vol. 9, pp. 1645-1660, 2007.
- [17] X. Jiang, Y. Dong, D. Xu, Bhargava, B., "GnuStream: a P2P media streaming system prototype", in *Proc. 2003 International Conference on Multimedia and Expo, 2003. ICME '03*, vol. 2, pp. 2, Jul. 2003.
- [18] X. Hei, C. Liang, J. Liang, Y. Liu, Ross and K.W., "A Measurement Study of a Large-Scale P2P IPTV System", *IEEE Trans. on Multimedia*, vol. 9, no. 8, Dec. 2007.

- [19] R. Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications", in 2001. *Proc. First International Conf. on Peer-to-Peer Computing*, Aug. 2001.
- [20] PPStream : <http://www.ppstream.com/>
- [21] PPLive : <http://www.pptv.com/>
- [22] QQLive : <http://live.qq.com/>
- [23] Open Bloom Filter Library: <http://www.partow.net>

