# Dynamic Data Replication with Churn Prediction in P2P Network

Final Report

*COEN 241 - Cloud Computing Summer 2016*

Gaurang Garg    Bo Li    Yun Qin

## Table of Contents

# 1. Introduction

## 1.1 Objective

Our objective is to implement a stable Peer to Peer (P2P) Network by providing dynamic replication of data using dynamic churn prediction.

## 1.2 What is the problem

P2P systems are deployed in networks with millions of nodes. The rate that these nodes leave or join the network can range from static to dynamic. When nodes leave the network, there is data loss if the data on the leaving nodes is not replicated. Minimizing data loss and increasing stability in a P2P system is a challenging endeavor.

## 1.3 Why this is a project related to this class

Our project focuses on the following topics related to the class:

- P2P systems
- Distributed Hash Table
- Data Replication
- Churn Prediction

### 1.4 Why other approach is no good

Most distributed hash tables have a fixed replication factor (RF) when deployed. This rigidity often leads to inefficiency in the P2P system when the system is stable and churn rate is low. Furthermore, the system can become unstable if the churn rate is high if the number of replicas in the system is small.

### 1.5 Why you think your approach is better

Our approach prevents data loss while being more efficient than existing methods for data replication in P2P systems. We accomplish this by using an algorithm called dynamic replication mechanism. Dynamic RM computes a dynamic replication factor that changes as the churn rate in the system changes. Since the system predicts what the churn rate should be, it can adapt the replication factor so the system will be fault tolerant when churn is high and efficient when churn is low.

### 1.6 Area or scope of investigation

Our goal in this project is to implement a distributed hash table that will be fault tolerant and efficient in a P2P deployment. We will accomplish this by providing an algorithm to predict the churn rate and an algorithm to determine the replication factor for the system once churn rate is predicted.

# 2. Theoretical bases and literature review

### 2.1 Definition of the problem

To provide dynamic replication of data in a P2P system by using churn prediction.

### 2.2 Theoretical background of the problem

In the case of churn in a distributed P2P system, redundancy of resources and data can help compensate for the loss of data when nodes leave the network. The amount of redundancy in the system depends on the churn rate of the network. In a local cluster environment, redundancy requirement is much less compared to a P2P file sharing network with millions of nodes. Thus, to determine the replication factor (the number of replicas to create for the system), one must first know the churn rate in the system.

## 2.3 Related research to solve the problem

Many authors have attempted to tackle the issue of churn by devising replication algorithms. Some of the more popular replication mechanisms are Symmetric Replication[3], Successor-List Replication[4], ID-Replication[4], and Multiple Identity Replication.

The symmetric replication[3] put each node in a group in which this node will store all other group members’ items such that any other node’s items can be found by put a request get to this node. In successor-list replication [4] the items of a node are replicated to its immediate k successors and in RelaxDHT[5] the items are replicated to a set of leaves. ID-replication [4] and [6] use the idea to assign an identifier to a group and node identifiers are only unique inside a group such that a group is responsible for a key range and not a single node. All nodes of a group replicate every key in the group’s responsibility.

## 2.4 Advantage/disadvantage of those research

The symmetric replication[3] is super stable, it works even each group has been reduced to just one node. But it suffers from the replication overhead problem because every node has to save all the information of other group members. Successor-list replication and RelaxDHT are easy to implemented but have the similar issue. ID-replication [4] and [6] eliminate the drawback of a successor-list replication which assign an identifier to every node but they suffer from two issues. The first issue of this design is that they are still using fixed RF. Also, the control overhead due to group merge and split need to be handled.

We would like to emphasize the common issues with the above replication algorithms are they don’t follow an adaptive replication scheme and they don’t predict churn rate using local churn observations.

## 2.5 Your solution to solve this problem

Our solution revolves around using churn prediction where nodes make local churn observations. To estimate churn, moving averages are used for short term prediction. There are many models for computing moving averages such as Simple

Moving Average, Simple Weighted Moving Average, Exponential Moving Average, Dynamic Exponential Moving Average.

In addition, in our conditional probability matrix (CPM) method, we try to identify the characteristics the the sequence of the number of departing nodes as a random process. In our PID Feedback Estimator (PIDFE), we try to combine the advantage of moving average with feedback mechanism which enable our prediction to quickly respond to the change of churn rate.

### 2.6 Where your solution is different from others

Our solution is different since we use a dynamic replication factor instead of a static replication factor. This makes our system more efficient when the network is stable as well as fault tolerant during heavy churn. Furthermore, when predicting churn, nodes make local observations that are not dependent on other nodes in the system which makes the system more robust. Also, we try to take into account the possibility that the correlation of the number of departing neighbor nodes over time is more complicated than linear correlation.

### 2.7 Why your solution is better

The systems we design perform better than The Dynamic Exponential Moving Average (DEMA) when the number of neighborhood leaving for each node is according to a random distribution rather than based on the assumption that the predicted number of departing nodes in future can be treated as a simple linear combination of previous estimation or observation.

## 3. Hypothesis (or goals)

Our goal is to design and implement a dynamic replication mechanism. By using churn prediction, the system would automatically adjust its replication configuration.

For churn prediction, we prepare three algorithms. We would compare these algorithms based on their prediction result. Then we can find out which algorithm is most suitable for churn prediction.

By periodically calculating the churn and replication factor, the system could automatically adjust its replication configuration so that to satisfy the desired reliability.

# 4. Methodology

## 4.1 How to generate/collect input data

There would be a data set generated for simulating churn. We'd simulate a network with 500 nodes, then using random number between 0 to 500 every 60 seconds to get the swarm size. We assume at each observation time a uniformly distributed percentage of nodes is removed and added to the system. So this would cover the most critical case, in which almost every nodes leave at the same time in a turn.

## 4.2 How to solve the problem

### 4.2.1 Prediction Algorithm design

Churn prediction is the basis of dynamically calculating the replication factor (RF). In P2P network any node might have local information of neighbour joining or leaving. Based on these information every node can make churn prediction. We would use different prediction models and compare them to see how accurate these models would be in predicting churn. Those Models are described as following:

- **Algorithm 1: Moving Average**

A moving average [1] is commonly used with time series data to smooth out short-term prediction and show longer-term trends or cycles. Moving average is often applied in stock market prediction or forecasting sales.

If an observation x is made during a time interval ∆t, e.g. 20 minutes. The number of observations used to calculate the prediction m is called Observation Length (OL) and denoted k.

The **Simple Moving Average (SMA)**[1] is calculated as the regular average except that only the k last observations x of churn at time t are considered instead of all observations made. Therefore, SMA changes with every new observation. The SMA is defined by the formula:

$$sma_t = \frac{x_{t-k+1} + x_{t-k+2} + \ldots + x_t}{k}$$

The **Exponential Moving Average (EMA)**[1] reacts on recent changing behaviors while a certain smoothing effect on the churn prediction will be maintained. Another advantage of EMA is that it does not require all observations to be stored, since it can be calculated from the last EMA and the new observation $x_t$ at time t, as shown in the formula:

$$ema_t = ema_{t-1} + \alpha(x_t - ema_{t-1}), \quad t > 1$$

The smoothing factor α can be calculated by the formula:

$$\alpha = \frac{2}{k+1}$$

The **Dynamic Exponential Moving Average (DEMA)**[1] uses linear regression analysis to determine the OL , which gives the best linear fit. For each possible value of k, the regression line with the least squares and the coefficient of determination, denoted $R^2$, is calculated. The closer $R^2$ is to 1, the better the regression line fits. The OL which gives the highest value for R2 is used for k .

In moving average model, the Observation Length(OL) would influence the smoothness of the prediction curve. More changes of the churn rate will lead to a lot of overhead due to too frequent changing of replication factor. So a proper Observation Length needs to be determined.

Then we come up with two algorithms that we would like to apply for the prediction of the next number of departing neighbors for each node. The first idea is we enable each node to maintain some knowledge about the possibilities of the number of departing neighbors in future based on the previous observation. With this knowledge, each node will predict the coming two observations and this knowledge will be updated based on new observations. The second idea is to borrow the concept of PID controller in control system. In this prediction method, we consider the integral part by using the simple moving average to track the long-time tendency. We use proportional term to rectify the error between the observation and the prediction. Then, we try to apply the third derivative term to settle the possible overhead by the previous two parts.

- **Algorithm 2: PID Feedback Estimator (PIDFE)**

In the paper [a], the author proposed several moving average methods, here we propose one method in which we leverage the basic concept of PID controller in system control. Suppose each node has their observation O(1), O(2), … O(t-1), O(t). And they have their prediction as P(0), P(1), … P(t-2), P(t-1). OL indicates observation length which is the time interval between two observation. Then their next prediction could be

$$\begin{aligned} P(t) = {} & K1 * (O(t - k + 1) + \ldots + O(t))/k \\ & + K2 * (O(t) - P( t- 1)) \\ & + K3 * ((O(t) - P(t - 1) - (O(t - 1) - P(t - 2))) / OL) \end{aligned}$$

in which, K1, K2, and K3 can be adjusted based on the network characteristics. If we set K1 as 1 and K2 and K3 as 0, then this algorithm reduces as simple moving average method. If we set K3 as 0 and replace the term with K1 with P(t-1), then it reduces to the Exponential Moving Average (EMA) method. The K3 term is used to catch a fast change of the churn by predicting the change tendency of the departing of neighbor nodes and thus improves settling time and stability of the system. However, how to adjust the three parameters are still a problem for us. Also, in a high churn circumstance, the K3-term might not be applicable because it is too sensitive to the rapid change.

- **Algorithm 3: Conditional Probability Matrix (CPM)**

In this algorithm, each node will maintain a conditional probability matrix X of size m by m where m is the total number of its neighbors. The entry X(O(t+1) = i | O(t) = j) (denoted as X( i | j ) for short) in row i and column j of the matrix X means the conditional probability of the next number of departing nodes is i at time t+1 given the fact that j nodes depart from this node at current time t. Basically, X acts as a "knowledge" for the node. When an observation i comes based on the previous observation j, we update column j of matrix X by using the following equation:

$$X( o | j ) = X( o | j ) /(1 + X( o | j )) \text{ where } o \neq i$$
$$X( i | j ) = 1 - X( 1 | j ) - X( 2 | j ) - \ldots X( i - 1 | j ) - X( i + 1 | j ) - \ldots X( m | j )$$

In this way, each X( o | j ) will be decreased if o != i and the entry with high probability will get high penalty (decrease more). The conditional probability X( i | j ) will be increased based on the current observation i and the previous observation j. We are still trying to figure a smarter way to update this matrix. Each node will also maintain two vector V_next and V_next2 of size m by 1. V_next indicates the probability prediction of next observation of departing nodes, suppose our current observation is O(t) = j, after we update X by using the observation, we can set V_next as a weighted sum of X's column j and the V_next2 vector.

V_next2 is 2-hop prediction of the number of departing nodes. When current observation is O(t) = j, after we update V_next, we will update V_next2 by

V_next2 = X * V_next.

In this way, once my next observation O(t+1) comes, V_next2 is actually a prediction of the coming departing nodes based on my previous knowledge before this observation O(t+1).

Then each time, we could predict the number of coming departing nodes as

$$E(t+1) = [1\ 2\ 3\ \dots\ m]' * V\_next$$

### 4.2.2 Replication Factor

For calculation of the replication factor(RF) part, we use the same method as mentioned in [1], in which we calculate RF based on the prediction number m and calculate it by using

$$p = m(m-1)...(m-RF+1)/n(n-1)...(n-RF+1)$$

in which, n is the number of neighbor nodes; m is the number of nodes will leave the system; r is the probability that a record is kept safe; p is the probability of RF nodes will leave. Then we recursively increase RF and recalculate p until

$$1 - p >= r.$$

### 4.2.3 Language and tool used

We use Java as our programming language.

For P2P network simulation, we use TomP2P, which is a open source P2P framework providing an advanced DHT. We would use TomP2P as underlying framework and the prediction model is built on top of it. TomP2P itself has two kind of replication mechanisms available. In both replication types, the  replication factor is automatically set by the dynamic RM.

## 4.3 How to generate output

For prediction part, we use generated data set to calculate the prediction of different models. By comparing the result, we'd get the most suitable model. After that, the experiment based on churn prediction would be run on TomP2P to test if dynamic replication mechanism works well.

## 4.4 How to test against hypothesis

The data loss experiment would show how the system react towards the changing churn rate and if the replication factor is set dynamically. As the desired reliability is increased, the data loss decrease because of the dynamically re-calculated RF.

# 5. Implementation

## 5.1 Programming Requirements

### 5.1.1 Use Cases

**Use Case 1**: Generate random churn
**Primary Actor**: churn generator
**Scope**: Evaluation for prediction and replication
**Brief**:
A random churn will be generated based on the previous churn and a fluctuation rate.
**Preconditions:**
Initial churn rate and churn fluctuation rate.
**Minimal Guarantees:**
A random churn is generated at each time based on the previous churn and some random distribution.
**Success Guarantees:**
The random churn sequence almost matches the real-life churn.
**Triggers:**
When we start the prediction or replication evaluation process, the system starts to generate random churns.
**Basic flow:**
1. A random churn fluctuation for current time is generated based on some random distribution.
2. A new churn is generated by adding the fluctuation to the previous churn rate.

**Use Case 2**: Evaluate prediction performance
**Primary Actor**:
prediction submodule (see 5.2.1 for modules)
prediction evaluation submodule
**Scope**: Evaluation for prediction
**Brief**:
Generate the churn prediction data. Calculate RF based on prediction and real RF based on observation. Compare real churn with prediction. Compare calculated RF with real RF.
**Preconditions:**
use case 1
**Triggers:**

A new churn is generated and input to the prediction submodule.

**Basic flow:**

1. Generate a new churn.
2. Generate the churn prediction.
2. Calculate RF based on the prediction.
3. Calculate real RF based on the next observation.
4. Record all data above including the real current churn to a data file.
5. Plotting.

**Use Case 3**: Evaluate replication performance

**Primary Actor**:

dynamic replication module (including prediction and replication parts)

prediction evaluation submodule

**Scope**: Evaluation for replication(data loss)

**Brief**:

The DHT see the new churn to drop peer nodes. The replication system use the new observation to do prediction and replication. The system check all keys and record the data in a data file. Create data plot.

**Preconditions:**

use case 1, use case 2

**Triggers:**

A new churn is generated and input to the prediction submodule and the P2P DHT.

**Basic flow:**

1. Generate a new churn.
2. Drop nodes based on the churn.
3. Generate the churn prediction and calculate the RF.
4. Do Replication based on RF.
3. Traverse all key in DHT to figure out loss of keys.
4. Record all data above.
5. Plotting and do comparison.

## 5.2 Design

### 5.2.1 Modules

Based on our requirements, we modularize the main system into 4 main parts.

The first module is the dynamic replication system, which should include a prediction submodule based on the observation and prediction data, and a replication submodule which will calculate the RF based on the prediction from prediction submodule and replicates data.

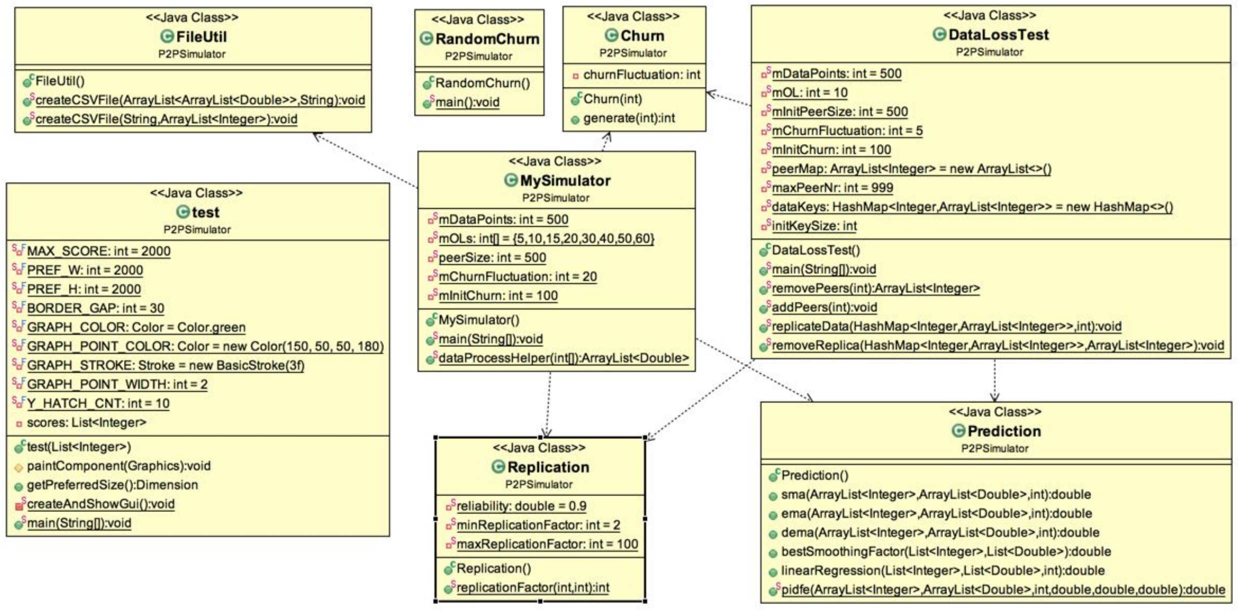
The second module is the performance comparison environment, which includes 2 submodules. We can set up different testing environment and this module generate all the experiment data for the 4 different algorithm for our future analysis. The first submodule is used to evaluate the churn prediction accuracy and the RF win rate for different algorithms. The second submodule is used for the evaluation of data loss performance of the 4 algorithms.

Besides, we also need the third module for analyzing data. This include a submodule to handle file I/O tasks to save our experiment data from the second module. Here we export the experiment data to a small Python plotting program for our analysis.

Also, we need to implement a random churn generator.

### 5.2.2 Class Diagram

Our main class diagram is shown in Figure 1.



<<Java Class>>
FileUtil
P2PSimulator
FileUtil()
createCSVFile(ArrayList<ArrayList<Double>>,String):void
createCSVFile(String,ArrayList<Integer>):void
<<Java Class>>
RandomChurn
P2PSimulator
RandomChurn()
main():void
<<Java Class>>
Churn
P2PSimulator
churnFluctuation: int
Churn(int)
generate(int):int
<<Java Class>>
DataLossTest
P2PSimulator
mDataPoints: int = 500
mOL: int = 10
mInitPeerSize: int = 500
mChurnFluctuation: int = 5
mInitChurn: int = 100
peerMap: ArrayList<Integer> = new ArrayList<>()
maxPeerNr: int = 999
dataKeys: HashMap<Integer,ArrayList<Integer>> = new HashMap<>()
initKeySize: int
DataLossTest()
main(String[]):void
removePeers(int):ArrayList<Integer>
addPeers(int):void
replicateData(HashMap<Integer,ArrayList<Integer>>,int):void
removeReplica(HashMap<Integer,ArrayList<Integer>>,ArrayList<Integer>):void
<<Java Class>>
MySimulator
P2PSimulator
mDataPoints: int = 500
mOLs: int[] = {5,10,15,20,30,40,50,60}
peerSize: int = 500
mChurnFluctuation: int = 20
mInitChurn: int = 100
MySimulator()
main(String[]):void
dataProcessHelper(int[]):ArrayList<Double>
<<Java Class>>
test
P2PSimulator
MAX_SCORE: int = 2000
PREF_W: int = 2000
PREF_H: int = 2000
BORDER_GAP: int = 30
GRAPH_COLOR: Color = Color.green
GRAPH_POINT_COLOR: Color = new Color(150, 50, 50, 180)
GRAPH_STROKE: Stroke = new BasicStroke(3f)
GRAPH_POINT_WIDTH: int = 2
Y_HATCH_CNT: int = 10
scores: List<Integer>
test(List<Integer>)
paintComponent(Graphics):void
getPreferredSize():Dimension
createAndShowGui():void
main(String[]):void
<<Java Class>>
Replication
P2PSimulator
reliability: double = 0.9
minReplicationFactor: int = 2
maxReplicationFactor: int = 100
Replication()
replicationFactor(int,int):int
<<Java Class>>
Prediction
P2PSimulator
Prediction()
sma(ArrayList<Integer>,ArrayList<Double>,int):double
ema(ArrayList<Integer>,ArrayList<Double>,int):double
dema(ArrayList<Integer>,ArrayList<Double>,int):double
bestSmoothingFactor(List<Integer>,List<Double>):double
linearRegression(List<Integer>,List<Double>,int):double
pidfe(ArrayList<Integer>,ArrayList<Double>,int,double,double,double):double

*Figure 1 Implementation Class Diagram*

Class Churn is used to create the random churn rate sequence, which will call on the random churn generator class RandomChurn. Class Prediction provides our churn prediction algorithm and SMA, EMA and DEMA as the benchmark algorithms. Class Replication will replicate data based on the predicted churn rate and the calculated RF. Class Mysimulator focuses on the prediction part. The performance of the 4 algorithms are compared based on their win rate of churn prediction and accuracy of RF calculation. The experiment data is written to some record files for future analysis. Class DataLossTest carries out the task of evaluating the data loss performance of the 4 algorithm. Class test is also implemented as a temporary visual data analysis tool and later on we switched to Python to get a more precise data plotting for analysis.

## 5.3 Data Flow

### 5.3.1 Prediction Evaluation

The data flow chart for prediction evaluation is shown in Figure 2. At each time step, a random churn will be generated and fed into the churn prediction system. The churn prediction system will predict the next churn and calculate the current RF based on its prediction. Also, based on the real churn rate, a correct RF will be calculated. The real churn rate of current time step, the churn prediction for next time, the calculated RF and the real RF that should be applied for current time will be written into a data document. Then the data will be used to draw plots for future analysis.

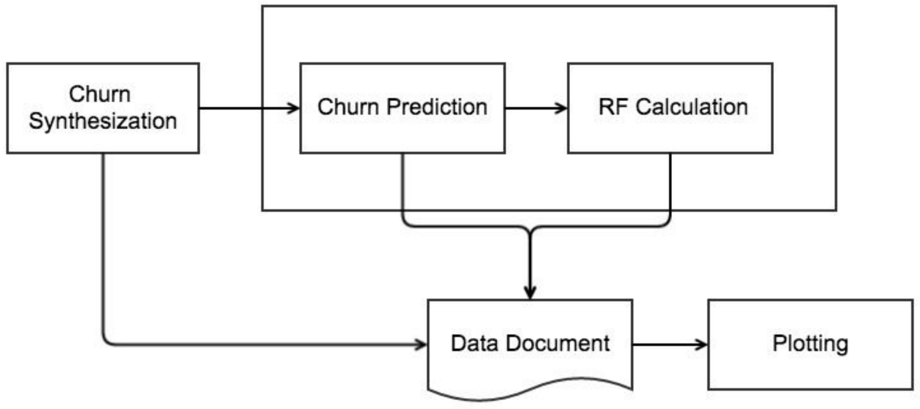


*Figure 2  Prediction Data Flow*

### 5.3.2 Data Loss Evaluation

The data flow chart for prediction evaluation is shown in Figure 3. At each time step, a random churn will be generated and fed into both the churn prediction system and the P2P DHT. The churn prediction/replication system will predict churn, calculate RF and do data replication for the P2P DHT. The P2P DHT will randomly remove and add nodes based on the random data it get from Churn Synthesization part. The system will check the available keys at each time and record it in a data document. Then the data will be used to draw plots for future analysis.

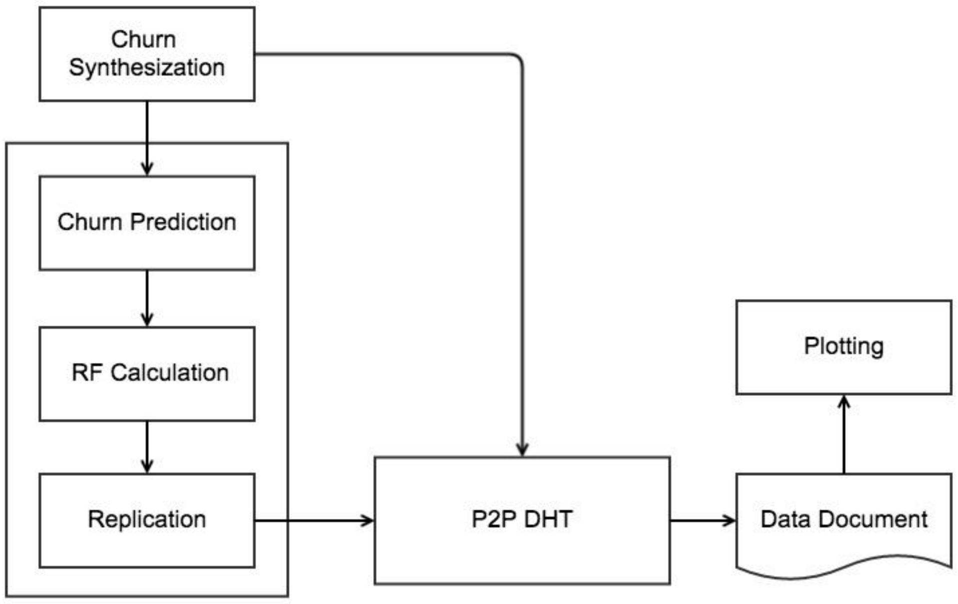


*Figure 3 Data Loss Test Data Flow*

## 6. Data Analysis and Discussion

### 6.1 Output Generation

First, we evaluate the performance of prediction models by feeding them the synthetic churn data set we generate, calculating their predictions, and comparing with the original data set. Second, after finding the best model, a data loss test is run for validation while data loss is measured.

Figure 4 is the example of our synthetic data set, which is generated with initial peer size 1000, initial churn 150, and churn fluctuation 10%.

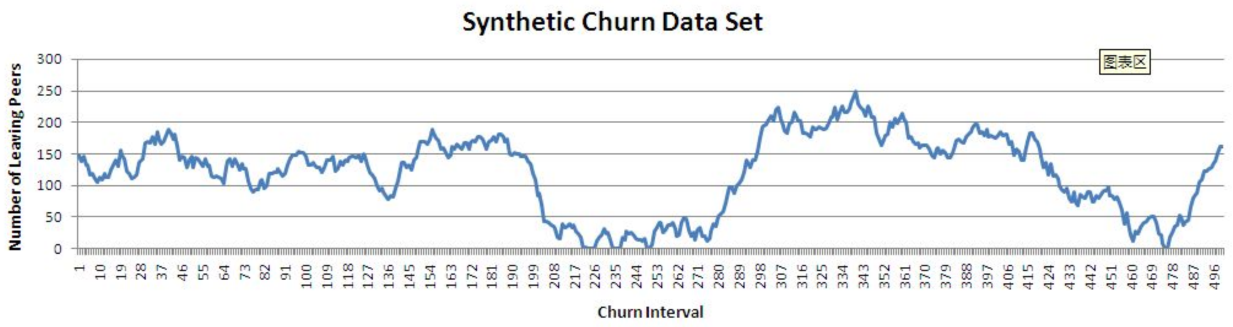


*Fig.4 Synthetic Churn Data*

## 6.2 Output Analysis

### 6.2.1 Comparison of Prediction Model

In our implementation, four prediction models are evaluated. The first model is **SMA**, a very simple and basic moving average approach. The second one is **EMA**, which gives more responsive react on recent changes. The third model is **DEMA**, considered the best fitting one in the paper[1]. It uses linear regression to determine the smoothing factor in the prediction. Last one **PIDFE** is the one we proposed, which explicitly takes prediction history into account.

Figure 5-8 show the predictions made by different models for OL10,20,40 and 60.

The result of  DEMA gives the most fitting curve comparing with others. It is proved that DEMA is the most suitable algorithm for prediction. As for PIDEF, it seems like it works not bad, slightly better than EMA, when OL is small. But for large OL, it’s more like SMA.

Figure 9-10 gives the comparison between 4 models for OL10 and OL20. Obviously, the lower OL results in a more responsive curve.

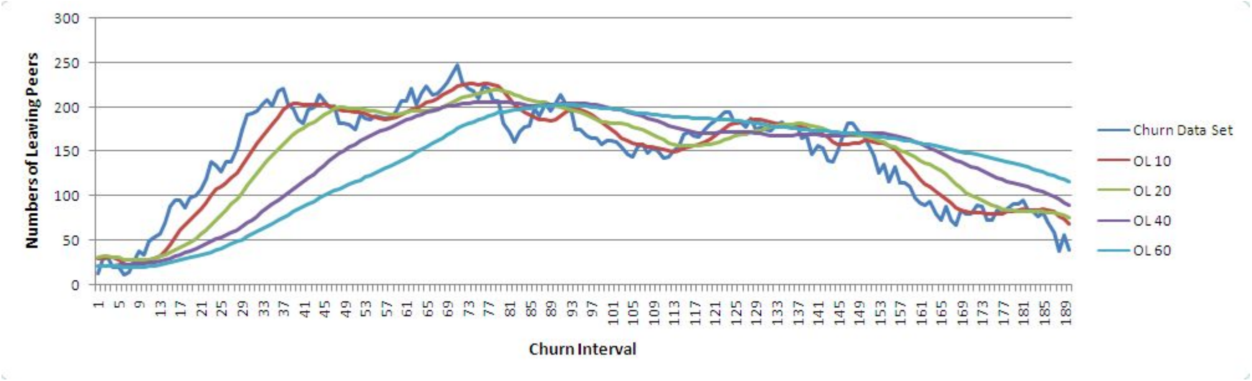


*Fig.5 Prediction of SMA model*

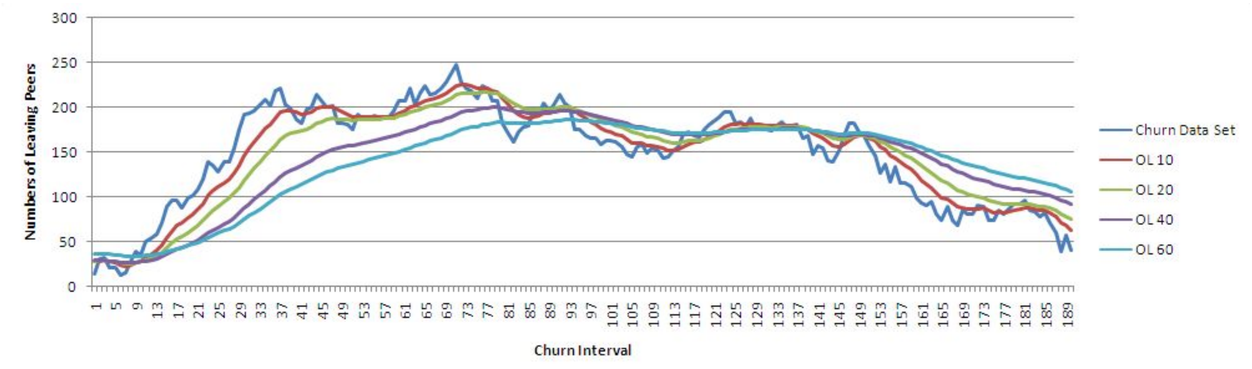


*Fig.6 Prediction of EMA model*

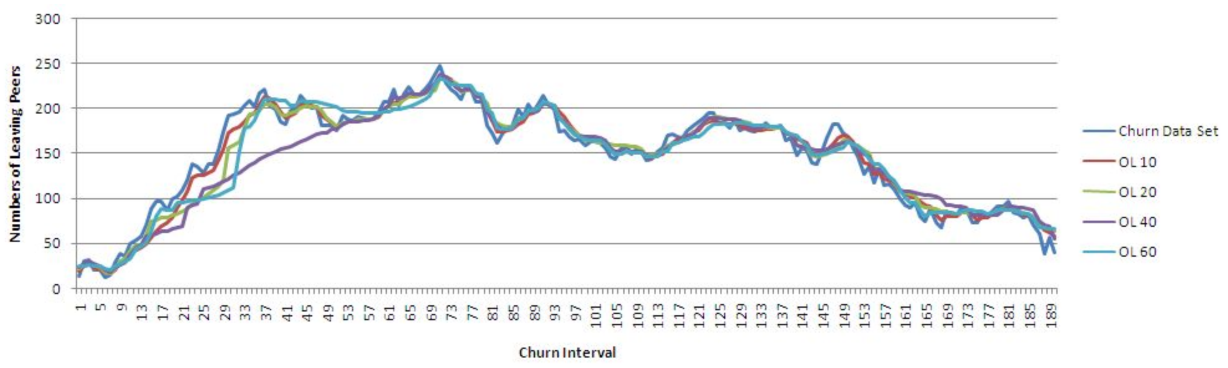


*Fig.7 Prediction of DEMA model*

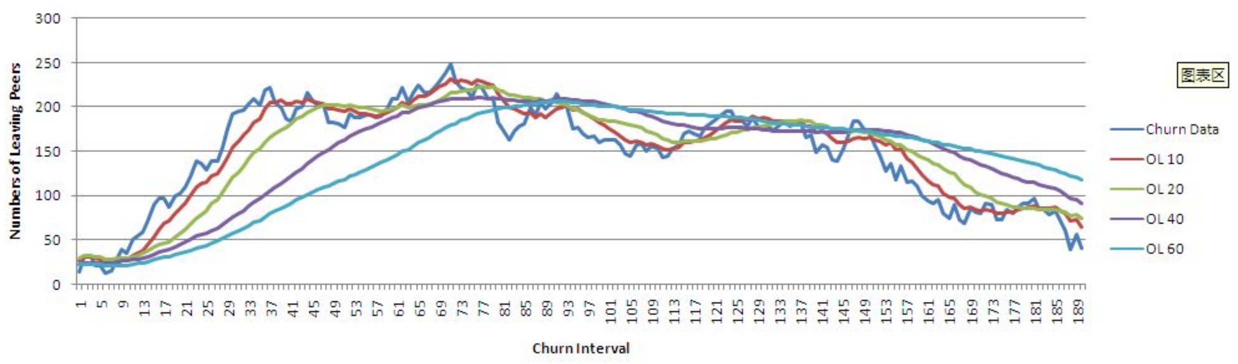


*Fig.8 Prediction of PIDFE model*

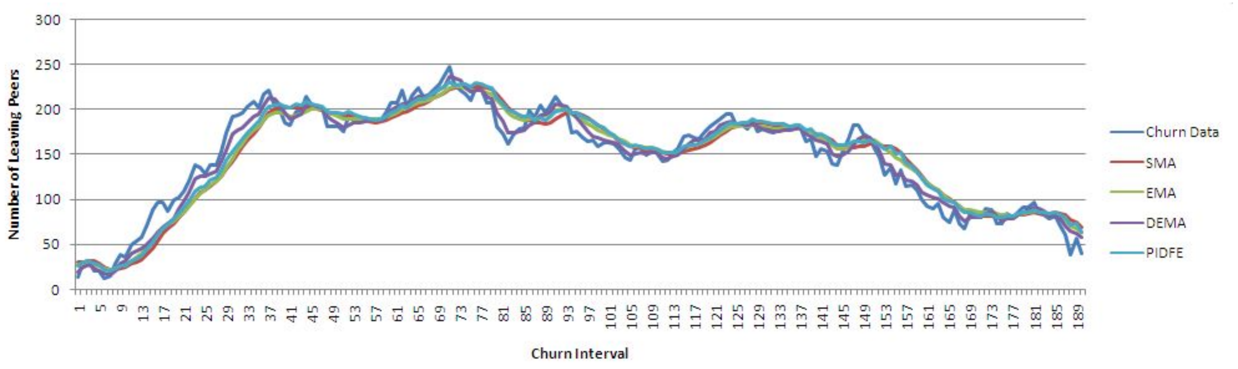


*Fig.9 Comparison of 4 Models for OL10*

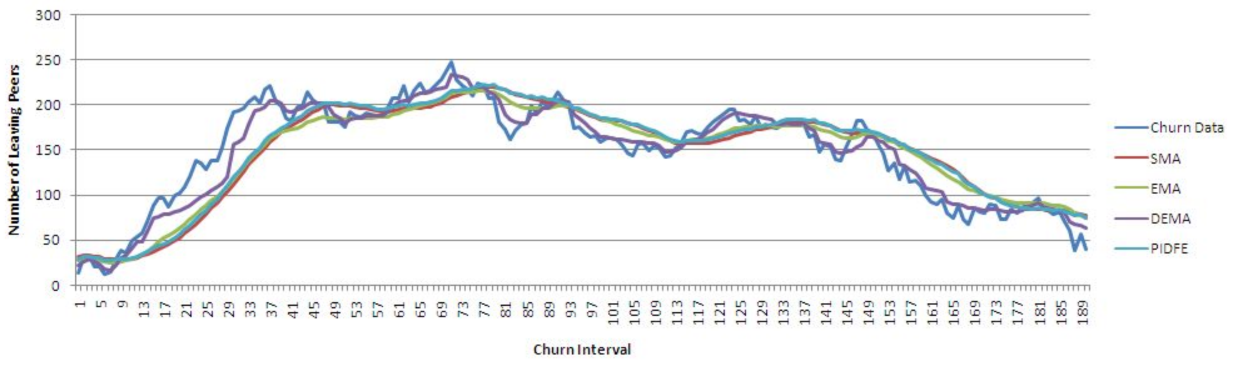


*Fig.10 Comparison of 4 Models for OL20*

Now we know the optimal Observation Length, which is OL10. We also need to identify the model with most accurate prediction. So the convergence of the prediction to the ideal model is analyzed. The result shows in Figure 11 and DEMA wins.

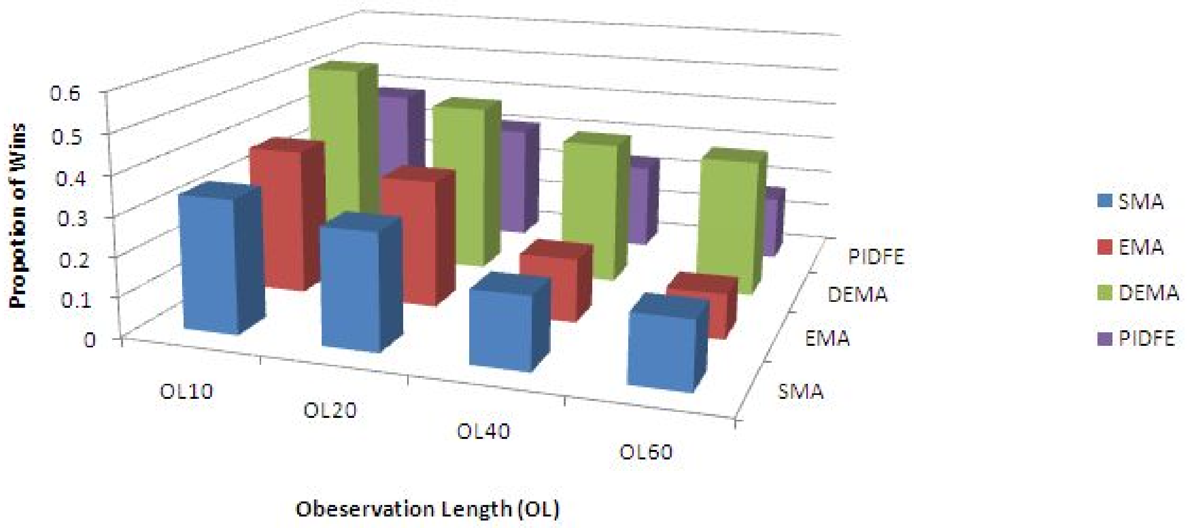


*Fig.11 Convergence of 4  Prediction Models*

As we mentioned before, the prediction is the basis of RF, to better evaluate the prediction, we also verify the accuracy of predicted RF. The ideal Replication Factor RF is calculated from the churn data set. The model-specific predRF_model is calculated for 4 models with different OLs respectively. For each time step, RF is compared to predRF_model. The predRF_model would be considered to be accurate when RF<=predRF_model<=RF+3[1], which means the higher-than-necessary RF is to be tolerated but not a lower one. Figure 12 shows the convergence result of 4 models with different OLs.

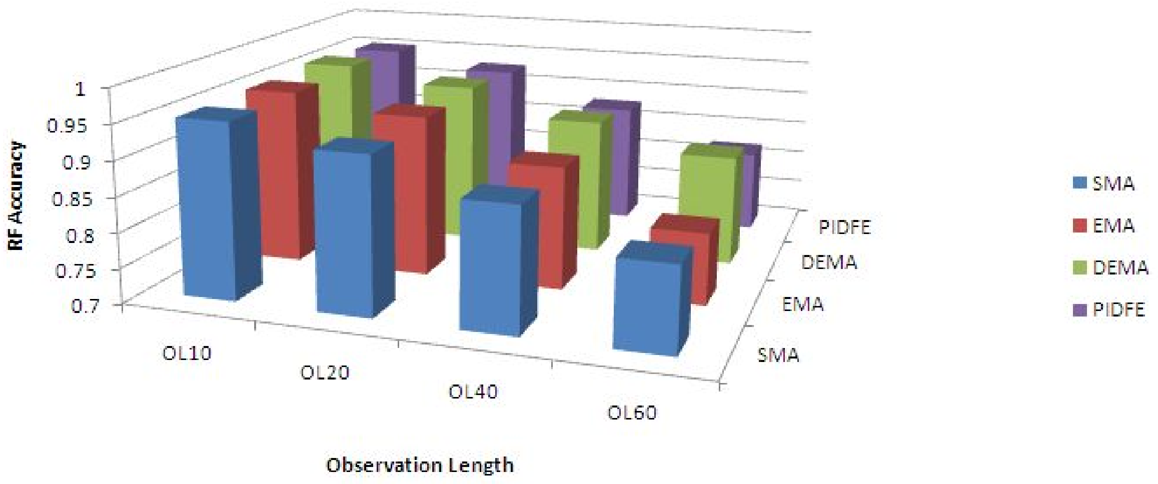
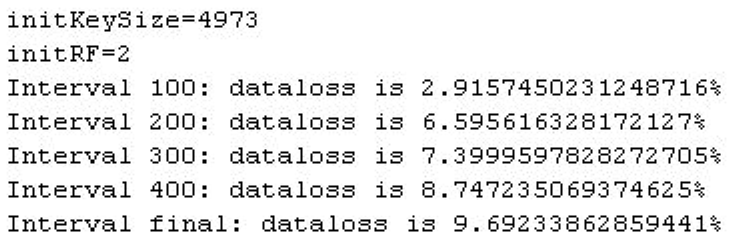


*Fig.12 Replication Factor Accuracy between 4 models and different OL for reliability 0.9*

Therefore, we conclude that the DEMA model with OL=10 can be considered the generally most accurate prediction model in our evaluation. And this would be used in the data loss test.

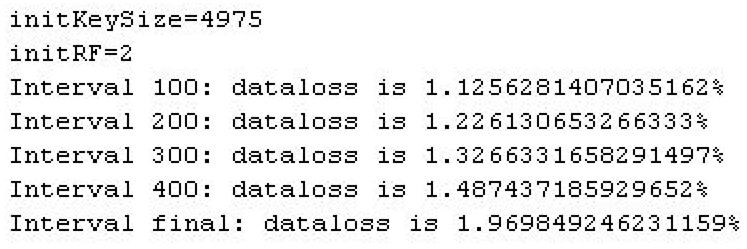
### 6.2.2 Data Loss Test

To evaluate how much data would be lost when using dynamic replication mechanism, we design the synthetic data with initial peer map size 5000 and initial hashkey size around 5000, which represents the data. We use prediction model DEMA with OL=10. Figure 13 shows the percentage of data loss for reliability value of 0.9, and Figure 14 shows the percentage of data loss for reliability value of 0.99.



```
initKeySize=4973
initRF=2
Interval 100: dataloss is 2.9157450231248716%
Interval 200: dataloss is 6.595616328172127%
Interval 300: dataloss is 7.3999597828272705%
Interval 400: dataloss is 8.747235069374625%
Interval final: dataloss is 9.69233862859441%
```

*Fig.13 Data Loss Result for reliability 0.9*



```
initKeySize=4975
initRF=2
Interval 100: dataloss is 1.1256281407035162%
Interval 200: dataloss is 1.226130653266333%
Interval 300: dataloss is 1.3266331658291497%
Interval 400: dataloss is 1.487437185929652%
Interval final: dataloss is 1.969849246231159%
```

*Fig.14 Data Loss Result for reliability 0.99*

Both the figures show that data loss behaves as expected, the more churn the more loss and the higher the reliability value the less the data loss.

## 6.3 Compare against Hypothesis

We implemented our PIDFE algorithm and it works for churn prediction and the system is able to adjust its replication configuration to satisfy the desired reliability.

Compared to the three algorithm, in prediction, our algorithm is better than SMA when there is a frequent churn fluctuation as it responds fast to the change of churn and is better than EMA in a less fluctuating churn environment as it has less vibration motivated by the fluctuation of the churn change. However, the performance of our system largely depends on how we select the parameters in the algorithm as OL, k1, k2 and k3 and this is still an open question for future study. DEMA is still the best among all the algorithm since it dynamically change the observation length OL. The data loss experiment provides similar results as in the prediction parts since this part is based on the prediction experiment.

## 6.4 abnormal case explanation

### 6.4.1 Small/Large Peer Size

In the data loss experiment, if we pick up a too small initial peer size, then the churn rate is relatively too large compared to the peer size, then no matter what algorithm we pick up, the data loss is very large and increase very fast to 100%. The reason is that each time there are too many peer leaving the network. The real RF is bounded between 2 and 6 (TomP2P), so the peer nodes cannot make enough copies for its data to response to this large churn rate. If the initial peer size is too large, in most case of the experiment, the peer size will be always large. Then the churn rate is relatively small with a fixed initial churn and a small fluctuation. This will lead to a result that all algorithms perform well since there is no enough churn to affect the network's data loss. Therefore, we need to select the peer size appropriately.

### 6.4.2 Small/Large Observation Size

Basically, for every algorithm, if the OL is set to be small, the algorithm more relies on the recent observation of churn while if the OL is large, the algorithm more relies on the history. For our algorithm, the selection of OL affects the performance of the system a lot since it will affect both the moving averaging part (how many previous observation should we use) and the second part (by affecting the weight parameter k2). A very small OL makes the prediction of our system more fluctuate. A very large OL will make our system too slow to respond to the change of churn.

### 6.4.3 Change of Peer Size

Since we use a uniformly distribution to simulate the process of leaving and adding peer nodes for the data loss experiment. It is possible that for a long period the number of leaving node is always greater than the number of adding nodes, which might lead to the result that the peer size become smaller and smaller and finally close to 0, or reversely, if the number of leaving node is always less than the number of adding nodes, the peer size will be too large. Both of this two cases will fall into the abnormal case 6.4.1 we just talked about. This could happen because we use random functions to create these numbers. In the future, more restrictions might be introduced based on the real network environment. Here we just simulate the experiment and disregard such extreme cases.

## 6.5 Discussion

Both the setup of the prediction and replication module of the P2P network and the setup of the testing environment is critically important for us to detect the behaviors of the replication mechanism based on each algorithm. As we talked in the analysis abnormal case explanation, how to select initial peer size, average churn rate, churn fluctuation will affect if the network environment will work normally. Only if the environment of the P2P network itself is good, then we can apply our replication design and see its performance. The test environment should be set up to be as close to the real life as possible. Also, it’s very important to determine the parameters of the algorithm, different algorithm with different configuration only works if the algorithm is able to be applied to that test environment and the configuration is appropriate.

# 7. Conclusion

## 7.1 Summary and Conclusion

In this project, we study 3 available prediction algorithms in P2P network and then propose our PIDFE algorithm to address the churn issue for data loss in P2P network. Then we abstract and simulate the replication overlay of P2P DHT. We implement 4 different algorithms to do the simulation and compare their performance based on different observation length(OL) and other parameters. Based on the generated experimental data, we compare the performance of each algorithm and the impact of different parameters on our PIDFE algorithm.

Based on our study and research, churn is a serious issue in P2P network. DEMA is the current best algorithm for the prediction of the churn and the replication of data. Our algorithm is better than SMA in a more fluctuating churn environment and has less vibration than EMA in a less fluctuating churn environment. But how to determine and adjust the parameter OL, k1, k2, and k3 precisely is difficult and still an open question.

## 7.2 Recommendations for Future Studies

### 7.2.1 Real P2P Network Simulation

In this project, we abstract and simulate the replication overlay of a P2P DHT, and apply 4 different algorithms to this overlay and compare their performance. In the future, the algorithm can be applied to a real P2P network to test its performance in a real-life network. In additional, we generate the random churn sequence by using a Normal Distribution, however, in a real life, the next churn might be generated based on a different distribution depending on the actual environment. In order to get a more precise analysis about the performance of different algorithm, the real-life churn data could be sampled from some real P2P network such as  BitTorrent.

### 7.2.2 Conditional Probability Matrix

One could also implement the second algorithm - conditional probability matrix. This algorithm expects to explore each peer’s individual ability to distinguish the characteristics of the random distribution according to which churn rate is changing. Therefore, each node tries to understand the churn process better and better.  For example, the current work assumes that the churn is generated based on a certain Normal Distribution. With conditional probability matrix algorithm, the peer node could slowly figure out the average and the standard deviation of the distribution and therefore the prediction of each peer might be more precise.

# 8. Bibliography

[1] Andri Lareida, Thomas Bocek, Maxat Pernebayev, Burkhard Stiller, “Automatic Network Configuration with Dynamic Churn Prediction,” IFIP/IEEE International

Symposium on Integrated Network Management (IM2015), 2015 May, pp. 363 - 370.

[2] TomP2P, a P2P-based key-value pair storage library, http://tomp2p.net/, retrieved on Aug 9th.

[3] A. Ghodsi, L. Alima, and S. Haridi, “Symmetric Replication for Structured Peer-to-Peer Systems,” in Databases, Information Systems, and Peer-to-Peer Computing, ser. Lecture Notes in Computer Science, G. Moro, S. Bergamaschi, S. Joseph, J.-H. Morin, and A. Ouksel, Eds. Springer Berlin Heidelberg, 2007, vol. 4125, pp. 74–85.

[4] T. Shafaat, B. Ahmad, and S. Haridi, “ID-Replication for Structured Peer-to-Peer Systems,” in Euro-Par 2012 Parallel Processing, ser. Lecture Notes in Computer Science, C. Kaklamanis, T. Papatheodorou, and P. Spirakis, Eds. Springer Berlin Heidelberg, August 2012, vol. 7484, pp. 364–376.

[5] S. Legtchenko, S. Monnet, P. Sens, and G. Muller, “RelaxDHT: A Churn-resilient Replication Strategy for Peer-to-peer Distributed Hash- tables,” ACM Transactions on Autonomous and Adaptive Systems, vol. 7, no. 2, pp. 28:1–28:18, July 2012.

[6] Z. Li, J. Wu, J. Xie, T. Zhang, G. Chen, and Y. Dai, "Stability-optimal grouping strategy of peer-to-peer systems," in Proceedings of IEEE Transactions on Parallel and Distributed Systems, 2011, pp. 2079-2087.

# 9. Appendices

Please find the source code attached.