# When FEC Speed up Data Access in P2P Networks

J. Lacan, L. Lancérica, L. Dairaine

ENSICA,
1, Place Emile Blouin
31052 Toulouse cedex 5, France
{Jerome.Lacan, Laurent.Lancerica, Laurent.Dairaine}@ensica.fr

**Abstract.** Peer to peer (P2P) Network is a high level logical network architecture build over end-user nodes interconnected by a physical network infrastructure. The performance in data access over the P2P networks is one of the main issues because the downloading of data is often the longest phase of P2P usage by end-user. In the present effort, we propose an approach based on Forward Error Correction (FEC) to speed-up data access into P2P networks. The main idea is to use FEC to dilute the information over the peers. This dilution allows a greater and flexible choice among downloadable disseminated parts of the FEC-encoded data and then enhances the speed up of the transfer. A case study illustrates the technique and a performance evaluation shows that for the same amount of information disseminated into the P2P network, this solution is more efficient in terms of data access performance compared to classical file replications.

## 1 Introduction

The popularity of P2P file sharing applications offers new prospects to Internet end-users. In this context, users can move from the simple consumer state to the active state of publisher, sharing their contents through the network. Peer to peer (P2P) Network is a high level logical network architecture build over end-user nodes interconnected by a physical network infrastructure. Main P2P systems services are storage, finding and download of data. The main characteristic of P2P network is to avoid any centralized point, allowing the building of new distributed services that are not built on classical models such as master-slave, consumer-supplier or client-server. The main benefits of P2P networks are to enhance the utilization of information, bandwidth, and computing resources [7]. The classical client-server model reduces the information resource usage because it makes difficult to find all the information from centralised servers. Allowing peers to collaborate in such a way to find the information is a decentralised solution avoiding those difficulties. The centralisation of information onto servers that attract so many requests from clients introduces bottleneck at network and system level, leading to the building of large web farms and broadband network resources. The dissemination of information over a set of peers allows distributing the load over the physical network and the end-systems. Bandwidth resources can be also enhanced because the P2P systems may introduce

high level routing function allowing speed up data access by load balancing or other approaches. Finally, computing resources are shared over the P2P network allowing e.g. a file to be replicated over a big number of peers.
As a result, numerous research level issues are related to the building of such a system. Scalability, fault tolerance, authentification, routing or data finding are examples of hot topics associated to this domain. The performance in data access over the P2P networks is also a crucial aspect because the downloading of data is often the longest phase of P2P usage by end-user. The contribution of this paper concerns this last issue.
A classical approach in this domain consists in enhancing the localization of replicated copies [1] [11] and downloading one them from the closest server. In the present effort, we propose a complementary approach consisting in using these optimized searching techniques over a set of Forward Error Correction (FEC) encoded blocks, to speed-up the data access into the P2P network. The FEC usage allows the information to be diluted over a number of different data blocks. Those blocks being disseminated over the P2P network, the FEC properties induce that only subparts of the total set of blocks are needed to reconstitute the original information. This allows the client to better choose the closest blocks among the multiple FEC blocks copies in such a way to reconstitute the original information. We show in this paper that for the same amount of information disseminated into the P2P network, this solution is more efficient in terms of data access performance compared to classical file replications.
The paper is structured as follows. The next section proposes a related work onto the FEC usage into P2P networks. Forward Error encoding is firstly described, and main applications are given in the context of P2P networks. The next section explains how the FEC are used into the P2P network. The technique is illustrated into a specific case study. Section 4 presents a performance evaluation, based both on analytical and simulation study. Section 5 proposes concluding remarks.

## 2 Related works

### 2.1 Forward Error Correction (FEC)

The error correcting codes or Forward Error Correction (FEC) techniques are a classical mechanism to protect the information against errors or losses in transmissions. The principle of FEC is to add redundancy to the transmitted information, so that it is still possible for receivers to recover the whole transmitted data, even after experiencing error transmissions.
The main class of FEC used in network domain is the block codes. These codes consider a group of k packets and compute n-k redundancy packets. In an erasure channel, *i.e.* a channel where the positions of the errors are known, the fundamental property of FEC permits to the receiver to reconstruct the k initial packets as soon as it receives k packets among the n emitted ones. Note that this property holds only for maximum distance separable (MDS) codes, *e.g.* Reed-Solomon codes [12][13]. Other

classical codes used in erasure channels, e.g. Tornado codes [2], need generally more than k received packets.
FEC are used in several kinds of applications: wireless transmissions (e.g., satellite, cellular phone) data storage (e.g., CD-ROM, DVD.) or computer memory (e.g., RDRAM). In networking context, FEC are classically used at lower layers (hardware) in detection/correction mode. In higher layers, similar software techniques such as CRC or checksums are used to detect corrupted packets. Since few years, with the improvements of the performance of personal computers, it is possible to implement encoding and decoding of the FEC in software at user level (i.e., transport or application layer). For example, most of reliable multicast transport protocols use FEC [16]. In data storage area, FEC are used to protect data against the failures of storage devices. Small scratches on CD-ROM are corrected by FEC directly implemented on the encoded information. Failures of hard disks can be protected by FEC-based systems such as RAID technology [10].

### 2.2 Use of FEC in the context of distributed storage

A classical concept on fault-tolerant systems consists in replicating data on several servers. FEC can improve the reliability of these systems by encoding the data, splitting up the encoded data and distributing the fragments over various servers [4]. The basic idea is that a redundant fragment situated on a server can compensate for the loss of any other fragment (of same size) due to the failure of another server. From a fault-tolerance point of view, the use of FEC reduces “mean time of failures by many orders of magnitude compared to replication systems with similar storage and bandwidth requirements” [17].
In the Internet context, the replication is used to speedup data access in mirror sites. A FEC-based alternative is proposed in [3] with a parallel access scheme for open-loop multicast distributions. The data are encoded on each mirror site with Tornado codes and are distributed in multicast mode. A user receives FEC-encoded parts of the data from several servers and is able to reconstruct the data as soon as it obtains a sufficient number of different packets. However, this solution can only be applied in particular contexts (such as software distribution) and presents several drawbacks such as, for example, the ending of the connections or the management of the network congestions.

## 3 Using FEC to enhance downloading performances

### 3.1 Context

P2P files sharing systems have very specific characteristics compared to classical distributed storage systems. An important point concerns the potential number of peers (e.g., up to 1.5 million users online at any time for Kazaa System in May 2002 [8]). Moreover, these peers strongly differ from storage system concerning the lifetime in the P2P network, the available throughput and the quantity of stored data.

These parameters can vary between three and five orders of magnitude across the peers in the systems [15]. In this context, the design of decentralized systems which takes in account all these properties is a difficult problem.
Our proposal is particularly well suited in this heterogeneous and moving context. Indeed, the usage of FEC allows enhancing the dissemination of the (parts of) files in the P2P networks. In addition to an enhancement of the global reliability of the data (like in distributed storage area [17]), this dissemination amazingly permits to improve the downloading performance of the users.
This section defines the use of FEC in such system and presents on a case study the basic principle which permits to speed up the data access.

### 3.2 Principle

A P2P system uses a peering architecture that offers the support for various P2P services such as file sharing between the various peers. Examples of such system are Napster, Gnutella, Kazaa or Edonkey [5][6][8][9]. Using these systems, each node is able to determine its peers and to localize data over the peer network. The peering algorithms allow furthermore determining the effective cost, in terms of bandwidth, for transferring data directly between two considered peers ([8][9]).
Our approach is based on the following method. Before the publication, the shared file is cut into n blocks. These various blocks are then FEC-encoded (see Fig. 1).

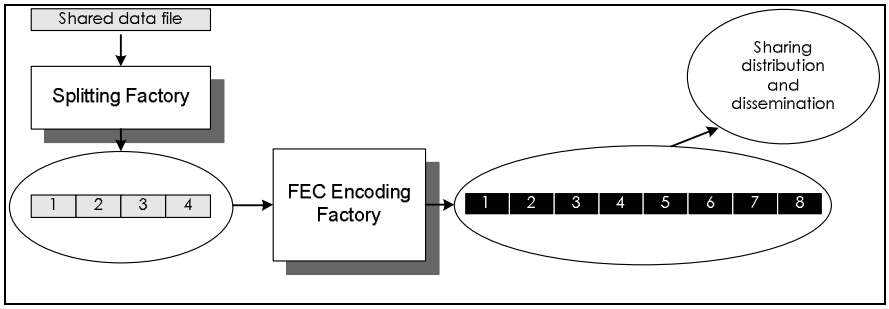


**Fig. 1.** Encoding and sharing mechanism of the data.

The shared files then become a sets of n+m blocks. Depending on FEC encoding technique, the n first blocks can or not be the n original blocks. In first case, they just result from the original file splitting. The m next blocks integrate the redundancy introduced by the FEC encoding. In the second case, the original information contained into the n blocks is diluted into the n+m blocks. The last phase of publication is the distribution of the various blocks over the P2P network. This distribution is ensured using a native service of the P2P architecture. This dissemination algorithm can be based either on the natural dissemination due to the user downloads between the peers or on a more specific dissemination algorithm of the P2P system. However, we are aware that this second solution, which is more efficient, can induce additional bandwidth usage.

Considering the various blocks disseminated into the network, downloading a complete file is equivalent to download any of n distinct blocks among the n+m ones. As there are greater choices between the different blocks to get, the availability and the robustness of the system is increased. When these blocks are disseminated over the P2P network, the searching service helps to determine the closest ones by considering a certain cost function (e.g., the biggest bandwidth). Then, the n closest blocks are downloaded in a classical way. The original data file can be finally obtained from the decoding of those n blocks. The next section proposes a concrete illustration of the performance gain obtained using such a technique.

### 3.3 Case study

To illustrate the concept and the benefits resulting from a P2P file sharing system using FEC mechanism, we present a case study based on a (very simple) P2P network. In this case study, the file transfer between two peers in ensured by a direct connection with a given cost. For sake of simplicity, we assume that the P2P structure is organized such as this cost is proportional to the total number of P2P hops between the two nodes in the peer tree. This cost does not take into account the time due to the localization algorithm. Moreover this inferred time is usually negligible compared to the downloading time. The network structure is fixed and different blocks or files distributions schemes are compared according to their downloading cost (see Fig. 2).

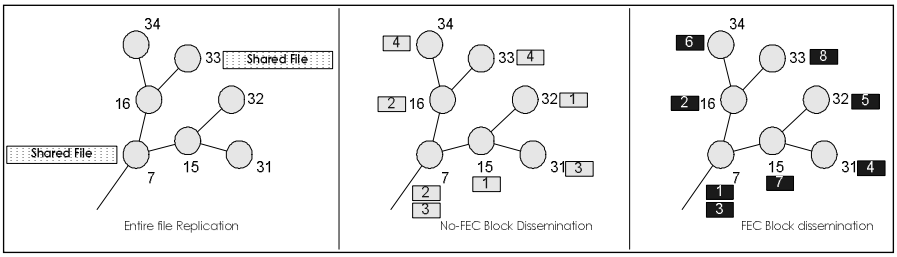


**Fig. 2**. Simple example to compute cost to recover entire file

We suppose the node 31 wants to recover the file.
The first approach, named *Entire File Replication*, consists in keeping the entire file which is simply replicated into 2 random nodes of the network (e.g., node 33 and node 7). When split into blocks (see next approach), the considered file is supposed to be cut in 4 blocks of the same size. Then the cost to download the file is equivalent to download 4 blocks situated at 2 hops (i.e., from node 7 to node 31), then $C = 4 * 2 = 8$.
This approach can be compared to the *No-FEC Block Dissemination* that considers the same file split up into 4 blocks. In this case, the 4 different blocks are duplicated 2 times so that the total load is always 8 blocks. In this case, 1block is at 0 hop, 1 block

is at 1 hop, 1 block is at 2 hops, and 1 block is at 4 hops then $C = 1*0+1*1+1*2+1*4=7$.

In the last approach, the *FEC Block Dissemination* there is still 8 blocks disseminated over the P2P network, but these blocks are FEC encoded. This encoding permits to choose any 4 over the 8 blocks of data, allowing a better choice of Peers. *As a result, the FEC Block Dissemination* cost is: 1block is at 0 hop, 1 block is at 1 hop, and 2 blocks are at 2 hops then $C = 1*0+1*1+2*2=5$

The results obtained in the example of Fig. 2 illustrates that the use of FEC can reduce the downloading cost for downloading a file. We are conscious that some P2P systems allow parallel connections to download the various blocks. In this case, the file downloading cost is now greatly influenced by the block the most expensive to download. In a perfect parallel context (bottleneck-disjoint), the *no-FEC Block dissemination* cost would be equal to 4. This cost must be compared to the *FEC Block Dissemination* cost equal to 2. Note that in the remaining of this article, we only consider serial (no parallel) downloading.

In order to generalize these results, we then consider the topology illustrated in Fig. 3.

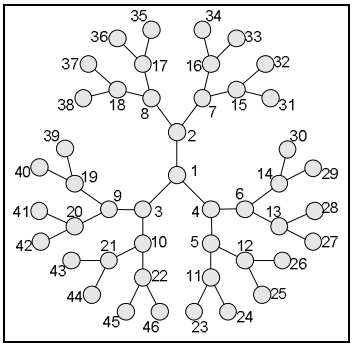


**Fig. 3.** The simple network topology used in the case study

Given this topology and cost computation technique, let us compare the downloading cost of a file for the different approaches. We consider that the file is split up into 10 blocks and that the rate of the FEC code is ½. Then, each encoded file is a set of 20 blocks. One will consider furthermore that each of these blocks is duplicated 5 times to constitute a set of 100 blocks to be spread in the network. This replication is operated randomly among all nodes. This cost is compared to two other approaches. 10,000 tests are processed to obtain a qualitative average.

- *FEC Block Dissemination*: **File downloading average cost per peer : 13.99**
  - 10 blocks file, FEC encoding rate ½, each block duplicate 5 times : total 100 blocks
- *No-FEC Block Dissemination* : **File downloading average cost per peer: 17.03**
  - 10 blocks file, no FEC encoding, each block duplicate 10 times : total 100 blocks
- *Entire file Replication* : **File downloading average cost per peer : 16.85**
  - 10 blocks file, no FEC encoding, each block duplicate 10 times : total 100 blocks

These early results show that on average, the use of a system using FEC increases (up to 17%) the access speed to the data shared in a P2P network with regard to classical approaches.

Furthermore the use of the FEC increases the presence probability of a file in the network by offering besides an improved reliability. The following section proposes a more general and realistic model to quantify the performance obtained by a P2P system implementing such a FEC technique.

# 4 Data access performance evaluation

## 4.1 Problem Modeling

The main objective of the current section is to quantify the improvement of using the technique presented in the previous paragraph. We propose a simple model allowing computing the cost to get an entire file over the three dissemination approaches (a) *Entire File Replication*, (b) *no-FEC Block Dissemination* and (c) *FEC-block Dissemination.* This cost refers to the downloading time of the file. Note that the time needed for the localization of the data is not taken into account because it is usually much lower than the downloading time. However, it must be remarked that the localization of the different blocks (approaches (b) and (c)) will take generally more time than the localization of a single file.
The reference cost concerns the entire file replication cost. In this approach, the file is entirely replicated r times over the network. The cost C then can be computed from the following expression.

$$C = \min_{j=1..r}(C_j) \tag{1}$$

Where $C_j$ is the cost of getting the $j^{th}$ replication of the entire file.

The *no-FEC Block Dissemination* supposes the file to be split up into $n$ blocks $b_1, b_2, ...b_n$; each block is supposed to have the same size. In this experience, the cost of getting the entire file consists in summing the costs to download each of the $n$ blocks. If $r$ replicas of the blocks are disseminated into the peer network, C becomes:

$$C = \sum_{i=1}^{n} \min_{j=1..r}(c_{i,j}) \tag{2}$$

Where $c_{i,j}$ is the cost of download of the $j^{th}$ replica of the $i^{th}$ block.

We finally consider the *FEC-Block Dissemination.* The $n$ original blocks constituting the file become now $n+m$ blocks due to FEC redundancy. Getting the file consists now in downloading the $n$ distinct blocks of minimum costs over the $n+m$ blocks. If the $n+m$ blocks are replicated $r'$ times over the network, if we define

$m_i = \min_{j=1..r'}(c_{i,j})$, for $i = 1,...,m+n$, and the function $f : \{1,...,n+m\} \to \{1,...,n+m\}$ such as $m_{f(1)} < m_{f(2)} < ... < m_{f(n+m)}$, then the cost to get the entire file is given:

$$C = \sum_{i=1}^{n} m_{f(i)} \tag{3}$$

To be fair with the previous approaches in term of data storage load over the P2P network, we assume the total number of blocks disseminated into the network to be constant, then:

$$r' = \frac{(n * r)}{(n+m)} \tag{4}$$

### 4.2 Simulation study

**Implementation.** The simple model given in the previous section has been implemented. The main idea is to consider 3 arrays of size $r$ for *entire file dissemination* and of size $n.r$ and $n.m.r'$ for respectively *no-FEC block dissemination* and *FEC-block dissemination.* The arrays are filled by random numbers following a certain probability law. Each value of the arrays represents the cost of getting the corresponding block (or file for the first array). Equations (1) and (2) allow an easy and quick computing of the download cost for the entire file. Each experience is done 100.000 times and gain percentages compared of the two last approaches are compared to the first one.

**Probability Law.** The characteristics of the probability law are a central issue for the last model. Indeed, depending on the probability law, the model previously presented can be adapted to any kinds of networks. In order to build a probability law as close to the reality as possible, we make two assumptions:

- the delay of the transmission between two peers is proportional to their distance;
- the distribution of the peers in the world is uniform.

It follows that the number of peers contained in a given surface is proportional to the area of this surface. By considering a particular peer, the number of peers which are at a distance greater than d and less than d+1 corresponds to the area of the corresponding ring (see Fig. 4).

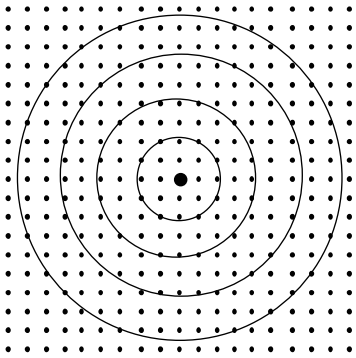

**Fig. 4.** Distribution of the peers

It may be observed that the area of the ring bounded by r and r+1, denoted by $A(R_{r,r+1})$ grows linearly, i.e, $A(R_{r,r+1}) - A(R_{r-1,r}) = 2\Pi$, for every r. On Fig. 4, the number of points included in $R_{0,1}$, $R_{1,2}$, $R_{2,3}$, $R_{3,4}$ is respectively 13, 44, 72 and 104. On this example, it can be verified that the number of points is increased of roughly 30 at each step.

Therefore, in order to randomly generate peers, we fix a maximum distance $d$ between the central peer and the other peers (this distance corresponds to the radius of the greatest disk) and we consider that the discrete random variable X used in the simulations is such that $\Pr(X = x)$ corresponds to the probability that a peer randomly chosen is in $R_{x-1,x}$, i.e. $\Pr(X = x) = (2x-1)/d^2$, for $x = 1,...,d$.

This distribution is used to fill the costs arrays of the model implementation.

*Impact on number of blocks onto the performance gain.* Model implementation is used to evaluate the gain of splitting up the file into a variable number of blocks using or not FEC (from 1 to 100 blocks). For each simulation, we compute the cost of getting an entire file replicated 10 times. The mean cost is computed over 100.000 simulations.

The first curve results from the cost comparison between getting the entire file when it is entirely replicated 10 times in the network (*entire file replication)*, and when it is split up into a number of blocks, those blocks being replicated 10 times (*No FEC block dissemination*).

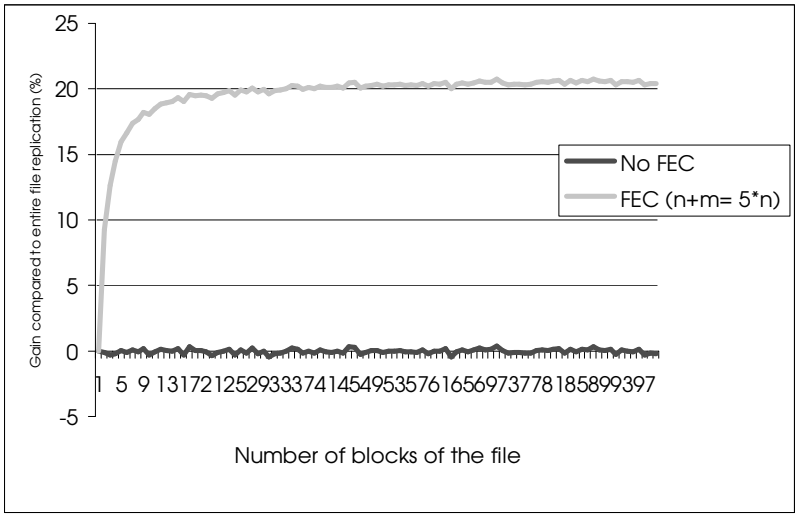


**Fig. 5.** Performance gain according to the number of blocks of the file.

We first notice in Fig. 5 that the *No FEC* curve shows that no performance gain is obtained when using a simple No-FEC Block dissemination. This result is not amazing and the equivalence of the two approaches can be mathematically proved (if parallel connections are not used).
The second curve presents the cost gain when using Bloc-FEC dissemination technique versus File Replication. The FEC redundancy rate used in this simulation is 5 FEC blocks for 1 original block (n+m=5*n). Due to this redundancy, each of the resulting blocks is only replicated 2 times (see Equation (4)). In this case, we can see that the gain quickly reaches approx. 20% when the file is split up into a minimum number of blocks (approx. 15 blocks). The cost of splitting a file into several blocks is the control information needed to localize the different blocks. The curve also shows that no more gain is obtained when the file is split up into a number of blocks larger than 30.

*Impact of FEC redundancy on performance gain.* The original file is now supposed to be split up into 20 blocks replicated 40 times. The total number of blocks is then fixed to 800. The cost of getting the entire file represents the reference cost. From this basis, we compute the gain for different values of FEC redundancy. The abscissa represents the redundancy rate R of the FEC, and is given by $R = (n+m)/n$. The redundancy rate is then computed so that Equation (4) is verified. The next Figure presents the obtained gain according to FEC redundancy rate.

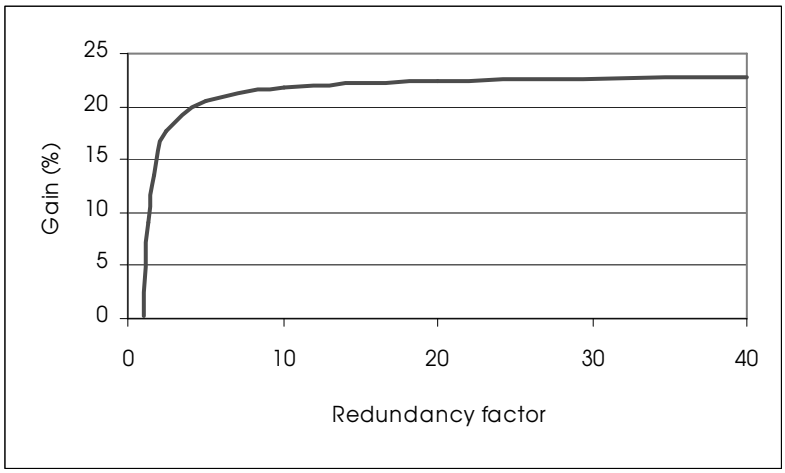


**Fig. 6.** Gain according to FEC redundancy rate.

The curve shows that up to a FEC rate of 5, the gain progresses quickly from 0 to approx. 22%. Then the gain is stable, and never goes beyond 24%. The recommended value for FEC redundancy factor is about 5 in this scenario.

An important point is that all these results are closely related to the probability law. For example, by using a uniform probability law (i.e. $\Pr(X = x_i) = \Pr(X = x_j),\ \forall i, j$), the gain can easily reach up to 30%. Unfortunately, the Uniform law does not seem very realistic.

## 5 Concluding Remarks

In this paper, we have presented a P2P file sharing system using a FEC scheme to disseminate information over the peers. Profits were estimated by comparing our system to classical approaches of various P2P systems.
Several lessons emerged from the obtained results. First, the use of FEC allows increasing the average availability of data and decreasing the access time. Secondly, although the obtained results depend on the probability law of date distribution over the peer network, there is a set of values optimizing the peering architecture by using a number of blocks and a FEC redundancy rate couple. Finally, the use of such a FEC-based system enhances system robustness, data availability and data access time reduction, while keeping constant the storage capacity of every peer.
In the sights of the obtained results, future works concern studies of different realistic probability laws of peers distribution to obtain optimum benefits. We plan to study in more details the behavior of our technique in the case of parallel downloadings (early simulations show a gain up to 25% of *FEC block dissemination* compared to *FEC block dissemination*). A study of dissemination and localization algorithms well-

adapted to FEC-encoded blocks management is also envisaged. A mathematical proof of the interest of the FEC based on the presented model is also under study.

## 6 References


1. K. Aberer, M. Punceva, M. Hauswirth, R. Schmidt, “Improving Data Access in P2P Systems”, IEEE Internet Computing 6, 1, pp. 58-67, Jan./Feb. 2002.
2. J. W. Byers, M. Luby, M. Mitzenmacher, A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data”, ACM SIGCOMM '98, September 2-4, 1998.
3. J. W. Byers, M. Luby, M. Mitzenmacher, “Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads”, in INFOCOM 99, 1999.
4. Y. Chen, J. Edler, A. Goldberg, A. Goettlieb, S. Sobti, P. Yianilos “Prototype Implementation of Archival Intermemory”, in *Proc. of IEEE ICDE*, February 1996.
5. Edonkey2000 Peer-to-peer system, www.edonkey2000.com, May 2002,
6. Gnutella Peer-to-peer system, www.gnutella.com, May 2002,
7. Li Gong, “Peer-to-Peer Networks in Action”, IEEE Internet Computing, January-February 2002
8. Kazaa Peer-to-peer system, www.kazaa.com, May 2002,
9. Napster Peer-to-peer system, www.napster.com, May 2002,
10. D. Patterson, G. Gibson, R. Katz, “A case for redundant arrays of inexpensive disks (RAID).,”, *Proceedings of ACM SIGMOD '88*, 1988.
11. C. G. Plaxton, R. Rajaraman, A. W. Richa, “Accessing Nearby Copies of Replicated Objects in Distributed Environment”, Proc. ACM Symp. Parallel Algorithms and Architectures, ACM Press, New York, June 1997.
12. I. S. Reed, G. Solomon, “Polynomial Codes Over Certain Finite Field”, J. SIAM, vol. 8, pp. 300-304, 1960.
13. L. Rizzo, “Effective Erasure Codes for Reliable Computer Communication Protocols”, In *Computer Communication Review*, April 1997.
14. P. Rodriguez, A. Kirpal, E. W. Biersack, “Parallel-Access for Mirror Sites in the Internet”, *In Proceedings of IEEE/Infocom'2000*, Tel-Aviv, Israel. March 2000.
15. S. Saroiu, P. Gummadi, S. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems”, Technical Report, UW-CSE-01-06-02, University of Washington, Department of Computer Science and Engineering, Seattle, WA 98195-2350, July 2001
16. T. Speakman, D. Farinacci, J. Crowcroft, J. Gemmell, S. Lin, A. Tweedly, D. Leshchiner, M. Luby, N. Bhaskar, R. Edmonstone, K. Morse Johnson, T. Montgomery, L. Rizzo, R. Sumanasekera, L. Vicisano, “PGM Reliable Transport Protocol”, April 2000
17. H.Weatherspoon, J. D. Kubiatowicz, “Erasure Coding vs. Replication : A Quantitative Comparison”, *inProc. of IPTPS '02*, March 2002.