Proactive Repairs and Incentives for Content Availability in P2P Overlay Networks

Octavio Herrera-Ruiz[†] and Taieb Znati[*,†]
[†]Graduate Telecommunications Program, School of Information Science
[*]Computer Science Department, School of Arts and Sciences
University of Pittsburgh, USA

Summary
Peer-to-Peer (P2P) networks enable access to shared distributed resources across the Internet. However, the availability of these resources is hindered by the members' transient participation (i.e., *churn*) and uncooperative behaviors (e.g., *free-riding*). Content redundancy using the idle storage space of nodes can be used to improve content availability. Maintain the scalability and self-organization properties of a P2P system, however, requires (*i*) minimizing the redundancy repair traffic (caused by *churn*), (*ii*) self-organizing mechanisms to balance the load and (*iii*) mechanisms to promote cooperation and enforce fair exchange of resources. In this paper, we take a holistic approach to content availability and propose a framework centered on efficient content redundancy, low-overhead maintenance and repair and incentives to mitigate the impact of churn. To this end, we propose a redundancy scheme that requires reduced repair bandwidth to improve content availability. The scheme is augmented with an efficient redundancy maintenance process to automate repairs. We also introduce a novel incentive-based mechanism to ensure a sustained and fair participation of peers and fair content sharing. The proposed redundancy scheme, referred to as Proactive Repair (PR), is studied analytically. The analysis shows that its repair bandwidth outperforms that of erasure coding and exact-MBR network coding. The proposed algorithms and mechanisms are implemented in an experimental testbed to evaluate their performance. The results indicate that our proposed solution is feasible and that it can improve content availability in P2P networks significantly.
Key words:
P2P, Availability, Redundancy, Churn, Incentives.

1. Introduction

Peer-to-Peer (P2P) technology has emerged as an important alternative to the traditional client server communication paradigm to build large-scale distributed systems. P2P enables the creation, dissemination and access to information at low cost and without the need of dedicated coordinating entities. However, existing P2P systems fail to provide high-levels of content availability, which limit their applicability and adoption [1]. This paper presents a holistic approach to device mechanisms to improve content availability in large-scale P2P systems.

Content availability in P2P networks can be impacted by hardware failures and churn. Hardware failures, in the form of disk or node failures, render information inaccessible. Churn, an inherent property of P2P networks, is the collective effect of the users' uncoordinated behaviour, which occurs when a large percentage of nodes join and leave frequently. Such a behaviour reduces content availability significantly. Mitigating the combined effect of hardware failures and churn on content availability in P2P networks requires new and innovative solutions that go beyond those applied in existing distributed systems. To addresses this challenge, we propose two complementary, low cost mechanisms, whereby nodes self-organize to overcome failures and improve content availability. The first mechanism is a low complexity and highly flexible hybrid redundancy scheme, referred to as Proactive Repair (PR). The second mechanism is an incentive-based scheme that promotes cooperation and enforces fair exchange of resources among peers. These mechanisms provide the basis for the development of distributed self-organizing algorithms to automate PR and, through incentives, maximize their effectiveness in realistic P2P environments.

Our proposed solution is evaluated using a combination of analytical and experimental methods. The analytical models are developed to determine the availability and repair cost properties of PR. The results indicate that PR's repair cost outperforms other redundancy schemes. The experimental analysis was carried out using simulation and the implementation of a testbed. The simulation results confirm that PR improves content availability in P2P. The proposed mechanisms are implemented and tested using a DHT-based P2P application environment. The experimental results indicate that the incentive-based mechanism can promote fair exchange of resources and

Manuscript received April 5, 2012
Manuscript revised April 20, 2012

limits the impact of uncooperative behaviors such as "free-riding".

Given the technology trend that disk space grows much faster (and cheaper) than access bandwidth, we assume the arguments presented in [2] stating that bandwidth, rather than storage space is the limiting factor for distributed storage in P2P networks under *churn*. Thus, P2P nodes are now, and in the future, more likely to have unused storage space rather than spare access bandwidth.

The contributions of our research include:

- A new low complexity and highly flexible redundancy scheme with small repair bandwidth requirements.
- Analytical models to evaluate fragment availability and redundancy repair cost for PR to derive its performance under diverse availability conditions.
- A self-tuning distributed redundancy maintenance process to automate PR in DHT-based P2P networks.
- A novel incentives-based mechanism that promotes collaboration, to achieve content availability, in exchange for download bandwidth (i.e., it is a bartering mechanism of storage versus transmission bandwidth).
- The experimental evaluation of a prototype implementation of our proposed mechanisms showing that PR+Incentives can improve content availability in P2P networks significantly.

The remainder of this paper is organized as follows. In Section 2, we present an overview of related work. In Section 3 introduces our redundancy scheme and our analytical models for fragment availability, file availability and redundancy maintenance cost. Section 4 describes the operation of the algorithms of our redundancy maintenance process. Section 5 describes our incentive-based mechanism. Section 6 describes our experimental work, and in Section 7 we present our conclusions and describe opportunities for future work.

2. Related Work

The use of redundancy for distributed storage has been broadly discussed in the research literature. Most initiatives discuss the use of replication and erasure coding redundancy as a dichotomy. The authors of [3], [4] and [5] confirm that erasure systems offer substantial storage savings vs replicated systems. Nonetheless, the authors of [5] point out that these savings might not be worth the added system complexity. Furthermore, with respect to redundancy repair cost, the authors of [2] and [5] conclude that distributed storage is not feasible for highly dynamic node environments (P2P networks) and for environments with highly available nodes (PlanetLab) replication should be preferred. Furthermore, [5] recognize the disadvantage of erasure coding when it comes to redundancy repair: to recreate any lost fragment we might need to download up to k elements. However, in their analysis, they assume the existence of a complete copy of the file to avoid this costly overhead.

Table 1. Redundancy Notation

A_F	File availability
ϕ	Fragment type availability
q^m	Availability of node m
N	Total number of unique fragments ($N=S*k$)
k	Reception efficiency
S	Coding gain
r	Replication degree (PR only)
d	Repair degree (exact-MBR only)
$f_{i,j}$	Fragment type i, replica j.
δ	Maintenance epoch
a_n^δ	Node availability, using maintenance epoch δ
a_d^δ	Disk availability, using maintenance epoch δ
α	Fragment size (e.g. $\|F\|/k$)
β	Repair block size (PR and EC: $\beta=\alpha$, exact-MBR: $\beta=\alpha/d$)
L_F^δ	Fragments lost during maintenance epoch δ
Ω_F^δ	Redundancy repair cost using maintenance epoch δ

Analytical expressions to determine the optimal storage overhead (S) for erasure coding and replication are presented in [5] and the optimal number of blocks (k) to divide a file in erasure coding is obtained by the authors of [6]. However, these optimal values do not translate into a better or worst overhead maintenance and repair, which is our main concern in P2P networks. The authors of [6] also argue that the product of the storage overhead (S) times average node availability (a) determines which redundancy method should be used. For $S*a<1$ replication is better and for $S*a>1$ erasure coding should be used. However, for most P2P settings (given a target file availability) the required storage overhead implies that the $S*a$ product will be larger than one. Therefore, erasure coding is always preferred, but once again, the authors do not consider the repair bandwidth aspects of the problem.

The authors of [7] present a hybrid redundancy mechanism with a combination of erasure coding and replication. However, their focus is not minimizing maintenance bandwidth consumption. Instead, when the file availability obtained with replication is insufficient, they boost it adding erasure coding.

An important aspect in the analysis of redundancy mechanisms is to determine the proper settings to evaluate a system. With respect to the model for node's availability (a), authors have taken different approaches. In [5] a is taken from traces of three different distributed systems

(Overnet, corporate PCs and PanetLab) and it is expressed as a function of the membership timeout. [8] follows a similar approach, using four different system's traces (PanetLab, Microsfot PCs, Skype and Gnutella), but they use the mean value of *a*, calculated as the fraction of nodes available out of those not considered as permanently failed. In [3] *a* is calculated using the probability distribution of disk lifetimes and the redundancy maintenance interval. In our work, we employ this last approach, but we use the distribution of session lengths instead of the distribution of disk lifetimes.

A promising new approach to redundancy in distributed storage systems is *Network Coding*. Dimakis, et. al., present a recent survey of the field in [8]. The authors explain that network codes can be constructed to minimize the amount of storage or the amount of bandwidth needed for repairs (i.e., recover lost blocks). In network coding, there is an optimality tradeoff curve between repair bandwidth and the amount of storage at each node. Our approach is different in the sense that instead of tackling the repair bandwidth issue with a single redundancy mechanism, we employ two mechanisms; achieving both simplicity and flexibility. Furthermore, the redundancy mechanism we are presenting in this paper could be used in conjunction with network coding (instead of erasure coding), but for simplicity we are analyzing the performance of our mechanism with the former exclusively.

The success of P2P system is deeply rooted in the level of cooperation among peers [9]. In that regard, the use of incentive mechanisms to influence nodes behaviors to help increase the overall system utility, rather than just their own, has been an extensive field of research. Zhang, et. al., present a review of incentive mechanisms in P2P networks in [10] and suggest five design requirements for their construction: *decentralization*, *service diversity*, *incentive*, *penalty*, *adaptability and lightweight*. However, none of the existing systems satisfies all of these requirements.

The authors of [11] present an incentive mechanism to control the minimum amount of time that nodes should participate in the system, as well as the minimum number of files that they should share throughout that time. The authors of [1] on the other hand, analyze the use of bundling to improve content availability, in BitTorrent in particular, improving even the download time experienced by peers when publishers exhibit high unavailability.

Our research differs from previous work in one or more of the following aspects: *i*) we explore the system design tradeoffs in the context of DHT-based P2P systems since this routing architecture presents highly desirable properties for the deployment of user-generated content sharing networks, such as: scalability, anonymity and resistance to censorship; *ii*) we take a novel approach to redundancy by pairing existing mechanisms to complement each other vs confronting their properties, as in [3], [4] or [5]; *iii*) our redundancy mechanism can be used in conjunction with incentive mechanisms to address fairness by aligning the level of service that nodes receive with their level of (storage) contribution to the network, and *iv*) our main objective is improving content availability using the least amount of repair traffic possible and without forcing peers to change their churn behavior, as it is the case in [1] and [11].

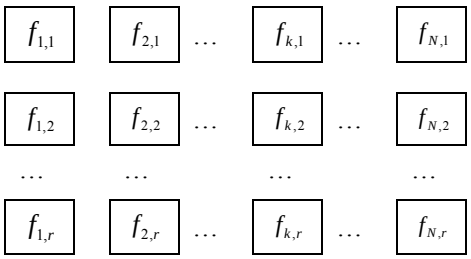

Figure 1. PR Redundancy Data Structure.

3. Proactive Repair Redundancy

Achieving a desired level of content availability when the peers storing the file have moderate to low availability can be accomplished using redundancy. We present a hybrid redundancy scheme to overcome the challenges of the P2P application environment.

Figure 1 presents the data structure of PR redundancy, which is built in two stages. First, PR redundancy encodes file *F* to create *N* unique fragments. The coding scheme is such that any *k*-out-of-*N* fragments suffice to reconstruct the original data (i. e., this stage uses maximum distance separable erasure coding (EC)). Second, PR creates *r* copies of each unique fragment and stores them at different nodes in the network. Retrieving a file from the network requires nodes to gather *k*-out-of-*N* unique fragments to reconstruct the original data.

File availability for PR redundancy can be obtained as follows:

$$A_{\mathrm{F}} = \sum_{i=k}^{N} \binom{N}{i} \cdot \phi^{i} \cdot (1-\phi)^{N-i} \tag{1}$$

where

ϕ = Pr[at least one fragment type *i* is available]

= 1 – Pr[no fragment type *i* is available]

$$= 1 - \prod_{j=1}^{r} (1 - q^{m} \chi_{i,j}^{m}) \tag{2}$$

and

q^m is the availability of node m,

$x_{i,j}^m$ is an indicator function such that

$$x_{i,j}^m = \begin{cases} 1 & \text{if } f_{i,j} \text{ is hosted in node m} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

When r=1, the formulation above corresponds to the binomial distribution, which applies for file availability of EC and exact-MBR redundancy.

Given a desired level of file availability and assuming nodes availabilities to be homogeneous and stationary, we can use the normal approximation to the binomial distribution to derive the optimal (i.e., minimum) value for ϕ:

$$\phi = \left(\frac{\sigma_\varepsilon \sqrt{S/k} + \sqrt{\sigma_\varepsilon^2 (S/k) + 4(S + \sigma_\varepsilon \sqrt{S/k}}}{2(S + \sqrt{S/k})} \right)^2 \quad (4)$$

Where S is the coding gain of the EC scheme used within PR (i.e. N/k), k is the reception efficiency and σ_ε is the number of standard deviations for the required level of file availability. For example, for a target file availability of two nines (0.99), k=8 and N=12, ϕ=0.8142. Furthermore, once this value is obtained, (2) it can be used to derive the optimal (i.e., minimum) value for the replication component in PR, r:

$$\begin{aligned} 1 - \prod_{j=1}^{r} (1-a) &\geq \phi \\ 1 - (1-a)^r &\geq \phi \end{aligned} \quad (5)$$

Where a is the average fragment availability, such that:

$$r \geq \frac{\log(1-\phi)}{\log(1-a)} \quad (6)$$

Notice that we have used the term fragment availability rather than the traditional term node availability. The reason being that to accurately capture the availability of content in P2P networks we have built a model capable to reflect not only the transient connectivity of nodes (i.e., traditional node availability) but other sources of content errors (such as disk failures).

4.1. Fragment Availability Model

For our fragment availability model, we derive expressions for the availability of fragments as a function of the frequency at which redundancy is evaluated (and eventually repaired) in the network. That is, we model fragment availability, denoted by a_f, as a function of the system maintenance epochs, denoted by δ. We derive our model in this way with the aim of devising design guidelines for the construction of a redundancy-maintenance process for content availability.

We define fragment availability as the product of the probability of finding a node still available after a maintenance epoch and the probability that no hardware errors prevent access to the data stored in the node. That is, fragment availability is the product of node availability, a_n^δ, disk availability, a_d^δ, and the probabilities of disk-read errors, $(1-b_r)^\alpha$, and failed data transmissions $(1-b_t)^\beta$.

Our starting point to obtain a_n^δ is the analytical work presented in [3], from which we have:

$$\begin{aligned} a_n^\delta &= \int_\delta^\infty \frac{t \cdot f(t)}{E[f(t)]} \frac{t-\delta}{t} dt \\ &= \frac{1}{E[f(t)]} \int_\delta^\infty f(t)(t-\delta) dt \end{aligned} \quad (7)$$

Where δ is the length of the maintenance epoch, $f(t)$ is the probability distribution of session lengths and $(t - \delta)/t$ reflects the probability of storing a fragment on a node early enough during its session so that it still will be online at the next maintenance epoch. $E[f(t)]$ is the expected value of the distribution $f(t)$ (i.e., mean value).

For a_d^δ we use the same formulation, so that:

$$a_d^\delta = \int_\delta^\infty \frac{t \cdot f_d(x)}{\mu_d} \cdot \frac{t-\delta}{t} dt = \frac{1}{\mu_d} \int_\delta^\infty f_d(t) \cdot (t-\delta) dt \quad (8)$$

Where $f_d(x)$ is the probability distribution of disk lifetimes and μ_d is its expected value.

For the disk read errors, p_r, and failed data transmissions, p_t, we assume a simple binomial model where bits fail independently. Using b_r to denote the non-recoverable read error rate and b_t to denote the bit transmission error rate, we have the following expression for fragments read and transmitted:

$$p_r = (1-b_r)^\alpha \quad (9)$$

$$p_t = (1-b_t)^\beta \quad (10)$$

Where α is the size of fragments and β is the amount of information transferred between nodes during a repair. For EC and PR redundancy these values are identical, but in the case of exact-MBR network coding $\beta = a/d$, where d defines the number of nodes needed to reconstruct a lost fragment (see [8] for a detailed description of fragments' repair process in network coding).

From a system design point of view, δ can be selected to obtain the most convenient results in (7) and (8), while the parameters of the redundancy scheme (e.g., EC or PR) can determine the results in (9) and (10). For the remaining factors in these equations, we assume that we have no control over them. Consequently, our design guidelines to

construct a redundancy maintenance system to improve content availability have to be focused on the selection of *i*) δ and *ii*) a redundancy scheme and its parameters.

4.2. Redundancy Maintenance Cost

Using the expressions above, we can determine the number of fragments lost in the network for a single file during a maintenance epoch:

$$E\left[L_{\mathrm{F}}^{\delta}\right]=\sum_{i=1}^{m} l_i \cdot \Pr[L_{\mathrm{F}}^{\delta}=l_i] \\ =\sum_{i=1}^{m} i\cdot\binom{m}{i}\cdot(a_f^{\delta})^{m-i}\cdot\left(1-a_f^{\delta}\right)^{i}=m\cdot(1-a_f^{\delta}) \tag{11}$$

Where m is the total number of nodes used to store a single file. For EC and exact-MBR m is equal to N (in equation (1)) and for PR it is $N*r$.

Given a maintenance epoch δ, the cost of restoring the redundancy lost in the system is given by the product (size)*(frequency). For short maintenance epochs, δ, cost is dominated by frequency and for long maintenance epochs cost is dominated by size. The average overhead to reconstruct the redundancy lost is given by:

$$\Omega_{\mathrm{F}}^{\delta}=\begin{cases}(L_{\mathrm{F}}^{\delta}+k)\cdot R(\tau)\cdot\beta_{EC}\cdot\dfrac{T}{\delta^2} & EC\\ (L_{\mathrm{F}}^{\delta}\cdot d)\cdot R(\tau)\cdot\beta_{MBR}\cdot\dfrac{T}{\delta^2} & exact\text{-}MBR\\ (L_{\mathrm{F}}^{\delta}+\gamma\cdot k)\cdot R(\tau)\cdot\beta_{PR}\cdot\dfrac{T}{\delta^2} & PR\end{cases} \tag{12}$$

Where T denotes a long period of time (e.g., 10 hrs) during which repairs are performed regularly (T/δ times in average). $R(\tau)$ is the average number of transmission attempts needed before a successful repair. τ is the time required to transfer a repair block of size β. Assuming an average transmission bandwidth for repairs equal to B, $\tau=\beta/B$. $R(\tau)$ is obtained using a similar approach to the formulation of a_f^{δ}, with the exception that in this case, two nodes must remain active:

$$R(\tau)^{-1}=(a_n^{\delta})^2 * a_d^{\delta} * (1-b_r)^{\alpha} * (1-b_t)^{\beta} \tag{13}$$

In addition, $\beta_{EC}=\beta_{PR}$, which is approximately equal to the file's size divided by k (i.e., $|F|/k$) and for β_{MBR} we assume the exact-MBR construction presented in [12] where $\beta = \alpha/d$ and $\alpha = 2*d*|F|/k(2*d-k+1)$. Finally, γ for PR's redundancy repair cost, is the probability that all r replicas of a fragment type are lost simultaneously, which is given by:

$$\gamma=\binom{L_{\mathrm{F}}^{\delta}}{r}\Big/\binom{N\cdot(r-1)}{r} \tag{14}$$

Figure 2 presents the redundancy repair cost for EC, PR, exact-MBR and an ideal case of EC where repairs can be done transferring only a single fragment (e.g., from a node with a complete copy of the file). It is clear that our proposed redundancy scheme consumes less repair bandwidth than exact-MBR and EC. The redundancy repair cost decreases for all schemes at longer maintenance epochs. However, it is not practical to set up a system with large maintenance epochs because these imply a larger number of nodes needed to store a file (i.e., larger coding gain to satisfy $A_F \geq 0.99$ in (1)) and the likelihood of loosing all replicas of a fragment type simultaneously increases.

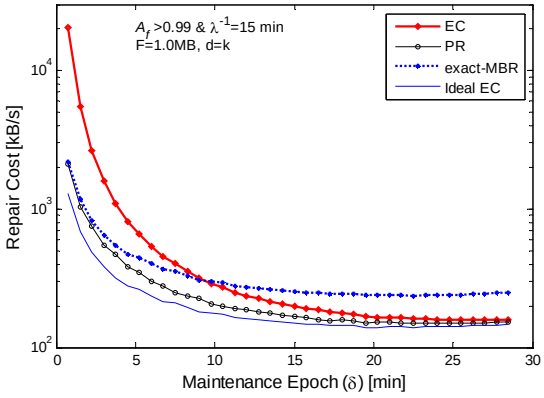

Figure 2. Maintenance Cost for Different Redundancy Schemes.

To obtain Figure 2, we obtain the optimal parameters for each redundancy mechanism. Given $A_F \geq 0.99$ and k=8 for all schemes, equation (1) is evaluated numerically to determine the optimal value of N at each maintenance epoch evaluated. Then, (12) is computed for each redundancy scheme assuming: *i*) distribution of session lengths in the network follows an exponential distribution with mean λ^{-1}=15 minutes, *ii*) disk lifetimes follows an exponential distribution, with a mean taken from the annual failure rate reported in [13], $\lambda_d^{-1} = 9.82\text{x}10^{-6}$, *iii*) the probability of bit read errors is equal to the disk specification of a consumer-class SATA hard-drive [14], $1.10\text{x}10^{-14}$ and *iv*) for the transmission bit error rate, we use the value reported in [15], $1\text{x}10^{-13}$.

In addition to consume less repair bandwidth than exact-MBR, PR redundancy has the advantage of being less complex. File reconstruction still requires decoding fragments, but the repair process itself does not involve decoding, as in EC and exact-MBR. Furthermore, PR always performs better than EC, as opposed to exact-MBR, which for maintenance epoch longer than 9 minutes consumes more repair bandwidth than EC (i.e., when the average fragment availability is below 0.55). This result is consistent with the results presented by Dimakis, et. al., in [16], where the authors indicate that as the network becomes less stable, the performance of network coding can be "very slightly worse" than erasure coding. Lastly,

PR is more flexible since its parameter *r* can be adjusted to accommodate different networking conditions as well as different availability requirement for different files. The code-only structure of EC and exact-MBR does not allow this flexibility.

In summary, PR redundancy is a novel redundancy mechanism that combines the storage-availability efficiency of EC with the repair simplicity of replication. Together, these redundancy schemes constitute a flexible and efficient hybrid redundancy scheme that outperforms other redundancy schemes.

4. Redundancy Maintenance Process

We adopt PR redundancy as our fundamental building block to construct a system to improve content availability in P2P networks. In the analysis presented above, we assume the presence of an oracle entity with perfect knowledge about the state of the system. This oracle is able to decide unequivocally when to perform repairs and which nodes should be involved. In practice, such entity does not exist. Furthermore, any centralized approach to manage such functionality is likely to become a scalability and security concern. Thus, we decided to define a distributed redundancy maintenance process capable to adapt to the particularities of the P2P application environment.

We device an automatic distributed redundancy maintenance process based on the fundamental content location functionality in DHT-based P2P architectures; that is, lookups.

Assuming the existence of a DHT-based P2P routing architecture, we adopt the following model. Nodes can play three roles in the redundancy maintenance process: *holder*, *index* and *target*. *Holder* nodes host complete or partial (i.e., fragment) copies of items; *index* nodes are in charge of evaluating the availability of these items and they trigger repairs as needed; *targets*, are nodes willing to host new fragment replicas (i.e., repair fragments) for files they are not currently storing any information.

For each file in the system, nodes collectively maintain a list of *holder* nodes. This list is referred to as the index entry of file f_i. It is maintained at f_i's i*ndex* node, where a file's *index* node is determined using the DHT routing mechanism. First, f_i's file-id is mapped into the DHT key-naming space; then, the peer with node-id numerically closer to this value becomes f_i's *index* node.

In our redundancy maintenance process, each *index* node knows the current redundancy level for the files with ids mapping to its portion of the key-naming space. Thus, *Index* nodes can instruct these *holder* nodes when a repair process has to be initiated.

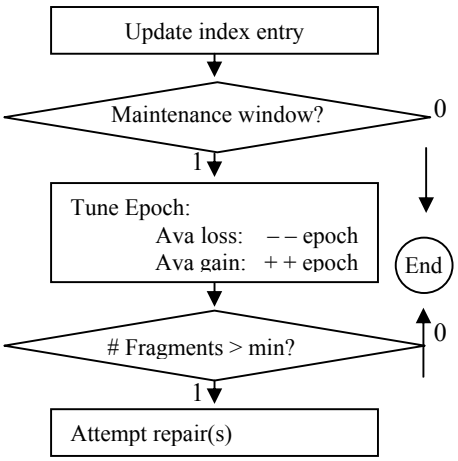

Figure 3. Redundancy Maintenance Process.

The redundancy maintenance process is performed in two stages: decision and transfer. The decision phase is responsibility of the *index* nodes, exclusively. The transfer of content on the other hand, is managed by at least two nodes. Data transfers are performed among content *holders* and *target* node(s). The *target* nodes for a particular repair are selected at random. Thus, in our redundancy maintenance process *targets* are not accountable for the fragments they host.

In addition to *holder*, *index* and *target*, P2P nodes can play an additional fourth role, a *requestor*. These are nodes that want to download a file from the network. When a *requestor* node searches for a file, it generates a lookup for the file id in the DHT's key-naming space. The result is the file's root node (i.e., *index* node). The *index* node respond with a list containing all the peers having a complete copy or fragments of the requested file (i.e., a list of *holder* nodes). Then, the requestor can contact these nodes directly to start a data transfer.

Figure 3 outlines the algorithm executed by *index* nodes every time they receive a registration message from *holder* nodes. The first block in this diagram indicates that *index* nodes update the respective index entry after every registration message. This procedure updates the last-seen timer for the peer sending the message and removes the entries of defunct *holder* nodes (i.e., with outdated last-seen values). The next stages of the redundancy maintenance process are executed only when the registration request arrives within the maintenance window (e.g., 30 seconds) at the beginning of a maintenance epoch period (e.g., 3 minutes). The actual length of the maintenance epoch is adjusted dynamically to match the average fragment availability of each individual file stored in the network. This is a key feature in our maintenance algorithm that allows us to handle efficiently the heterogeneous connectivity patterns of deployed P2P networks. The dynamic adjustment of the

maintenance epoch allows the system to tune itself at a predefined repair cost/reliability tradeoff. We omit the details of the four alternative algorithms we developed with this purpose due to space constrains. Nonetheless, we can describe the basic approach as follows: when file availability increases, maintenance epochs can be extended, and when file availability deteriorates, maintenance epochs should be reduced. This procedure is presented in the diagram above as tune epoch. After this adjustment of the maintenance epoch, *index* nodes are ready to trigger all required repairs. To accomplish this task, *index* nodes send a repair requests to selected *holder* nodes. Within these requests, *index* nodes include a list of *target* nodes that could host new fragment replicas. *Index* nodes verify that the total number of unique fragments present in the network is above a predefined minimum before generating any repair request. This is done to avoid repairs for files that can no longer be retrieved from the network (i.e., there are fewer than k unique fragments available).

5. Incentives

Incentives are mechanisms embedded into the operation of a system to regulate the exchange of resources among participants. Incentives are used to guide the behavior of individual nodes towards a common goal. This guidance can be expressed as a reward, given the cooperation of a node, or as a penalty, in the absence of it.

In our case, the behavior we want to promote is participation in the redundancy maintenance process of PR (i.e., accept/generate new replicas upon request). It is assumed that the inclusion of incentives does not hinder the content availability gains obtained with redundancy. On the contrary, it makes these gains more robust by defining system rules that promote the inclusion of nodes that otherwise would, acting selfishly, avoid participation. To achieve this goal, we define both a reward and a penalty metric. We base both of these metrics on the fundamental content availability component of PR. That is, coded file fragments.

Let c_m denote the contribution of node m:

$$c_m = F(\chi^m) \cdot G(\chi^m, t, s, p) \tag{15}$$

where:

$$F(\chi^m) = \sum_{\forall i} \sum_{\forall j} \chi_{i,j}^m = n_m \tag{16}$$

We name this function *fragment count* and χ^m is an indicator function:

$$x_{i,j}^m = \begin{cases} 1 & \text{if } f_{i,j} \text{ is hosted in node } m \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

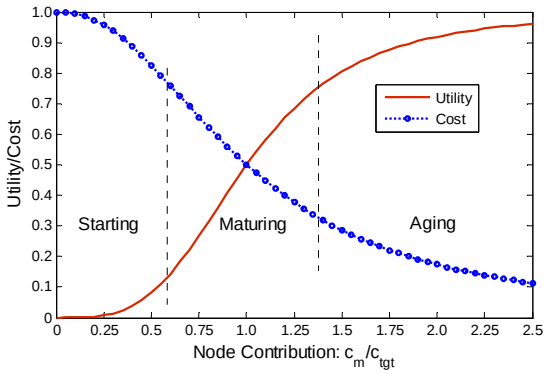

Figure 4. Sigmoid Functions.

and $G(\chi^m, t, s, p)$ is named *contribution gain* function, which is a non-decreasing function of fragments' age (i.e., time), size and popularity. For this function, we consider two variants:

$$G(\chi^m, t, s, p) = \begin{cases} 1 & \textit{Unitary} \\ H(t) \cdot K(s) \cdot L(p) & \textit{Enhanced} \end{cases} \tag{18}$$

where $H(t)$, $K(s)$ and $L(p)$ are staircase functions of a node's average fragment age, size and popularity, respectively.

For simplicity, we assume the *unitary contribution gain* case while describing the structure and implementation of our incentive mechanism. Further details of the *enhanced* case are presented in the experimental portion of this paper.

We model the reward and penalty components of the incentive mechanism using variants of a sigmoid function. A sigmoid has a S-shaped curve that can be used to model the life cycles of different natural and man-made systems [18]. A sigmoid has the following form:

$$S(c_m) = \frac{(c_m / c_{tgt})^\sigma}{1 + (c_m / c_{tgt})^\sigma} \tag{19}$$

where σ>1.0 is the shape parameter and c_{tgt} is the average (or target) content contribution in the system.

Figure 4 presents two sigmoid functions and indicates three different stages in the content contribution of nodes: *starting*, *maturing* and *aging*. The figure illustrates the different cost/utility tradeoffs that nodes can achieve as they increase their content contribution.

For our incentive mechanism we define two sigmoid variants named: replication probability or cost, *Cost*(c_m) and transmission bandwidth or utility, *Util*(c_m). *Cost* is used to regulate whether nodes should accept new replicas. Thus, it is a decreasing function of c_m (e.g., 1-$S(c_m)$). *Util*, defines the maximum amount of bandwidth nodes can receive when they initiate a data transfer. Consequently, it is an increasing function or c_m.

$$Cost(c_m, \vartheta) = 1 - \frac{(\vartheta \cdot c_m / c_{tgt})^{\sigma_c}}{1 + (\vartheta \cdot c_m / c_{tgt})^{\sigma_c}} \quad (20)$$

$$Util(c_m, \xi) = \xi + \frac{(c_m / c_{tgt})^{\sigma_u}}{1 + (c_m / c_{tgt})^{\sigma_u}} \quad (21)$$

where ϑ and ξ are used to adjust the shape of the sigmoid function as follows. The role of ϑ in (20) is to shift the sigmoid to the left, so that once nodes reach a content contribution equal to c_{tgt}, their probability of accepting additional replicas approaches zero (versus 0.5 in the regular sigmoid function (5)). In (21), the role of ξ is to define a minimum transmission bandwidth so that nodes do not starve when they first join the system (i.e., without a contribution).

Our incentives mechanism is different to others initiatives (see [10] for a review) in two aspects. First, we adopt a completely distributed architecture. Second, our incentive mechanism employs two dissimilar system properties, storage and bandwidth. Storage is a long term property while bandwidth is short term (i.e., it is realized during "short" intervals). The incentive-based mechanism we propose barters availability (i.e., storage) for performance (i.e., bandwidth) as opposed to traditional approaches that use a single system metric as the mean to influence the behaviour of participants (either storage, bandwidth or other unidimentional property). To minimize overhead, our incentive mechanism is embedded into PR's redundancy maintenance process.

Figure 5 presents a sequence diagram for the redundancy maintenance process of PR with incentives. In the figure, the *Index* node evaluates the availability of the file upon receiving a registration message. Repair requests are created as needed (but the diagram only shows the message for *Holder 1*). Upon receiving a repair request, *Holder* 1 attempts to push a f_i replica to a new node. Each *Target* contacted accepts the replica with probability $Cost(c_{A|B})$ (i.e., using their own contribution). In case of accepting, *Target* sets the maximum achievable bandwidth for this transfer using $Util(c_{H1})$*MAX_BW (i.e., using *Holder 1*'s contribution). The list of possible *Targets* is provided by *Index* node. This list, is sorted by contribution, so that the first node to be contacted has the highest probability of accepting.

The structure of the redundancy maintenance process with incentives allows nodes to define their own utility/cost tradeoff. Individual nodes can define unilaterally the level of contribution that works best for them (according to their resources). Still, the exchange of information among peers would be fair. Nodes with larger contributions will receive better transmission bandwidths, without precluding nodes with poor connectivity (i.e., slow transmission bandwidth) from participating in the system.

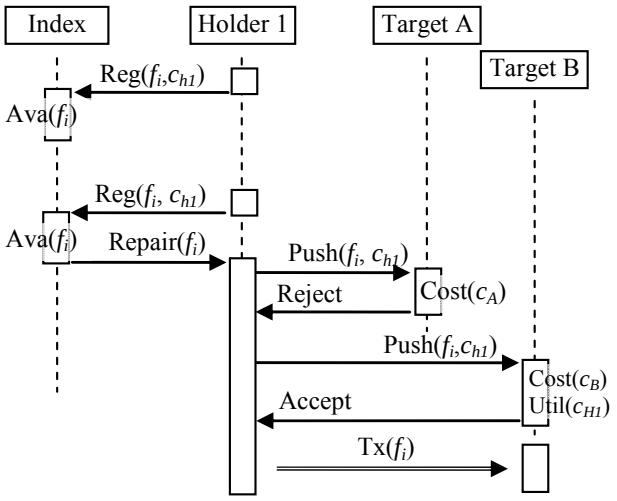

Figure 5. Redundancy Maintenance for PR+Incentives.

6. Experimental Work

6.1. Simulation Setup

We have implemented a redundancy-maintenance system using Bamboo [17] and PR with incentives. Bamboo is an open source DHT-based P2P application substrate written in Java. We employ Modelnet [19], to emulate a wide area networking environment in a cluster of machines interconnected with a Gigabit Ethernet LAN.

The emulated underlying Internet topology in our experiments consists on 1,344 edge nodes distributed across 836 distinct AS-level stub networks in a 4,000 node wide area network. Peers are mapped to one of these nodes randomly. To allow modelnet's routing model to scale, each node executes up to three instances of Bamboo, depending on the host machine capacity. A total of twelve host Linux machines are used to simulate 1,840 peers with heterogeneous online and offline intervals, which are defined based on the model presented in [20]. The average node availability is 0.28 and the median session length is 18.7 minutes.

The system manages a total of 500 unique files. 60% are audio, 10% video and the rest are Web-like. Complete files and fragments are assigned to nodes at random before the start of the simulation, with 10% of the nodes receiving a complete file. Our proposed redundancy scheme, PR, uses N=18, k=6 and r=2[1].

[1] Except for PR+Inc1, which uses r=3.

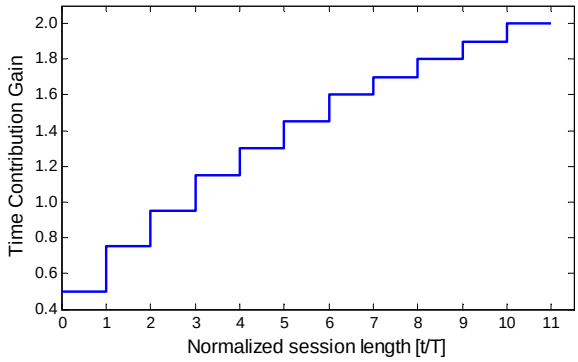

Figure 6. Time Contribution Gain.

Individual experiments last 4.5 hours and are replicated to obtain confidence intervals. Nodes continuously perform the redundancy maintenance process outlined earlier (see Figures 3 and 5). In addition, nodes attempt file retrievals (i.e., lookups and a subsequent data transfer) at exponential distributed intervals with a mean of 5 minutes. File retrievals are considered successful when nodes complete the download of k unique fragments from *holder* nodes.

File's popularities follow a Zipf distribution and fragments inherit their popularity from their originating files. Popularity values, p_x, are maintained in cumulative form within the simulation That is, files have a popularity value (between zero and one) that corresponds to the cumulative mass function, CMF, of the file's rank in the system's Zipf distribution.

We present results for three variants of PR with incentives: PR+Inc1, PR+Inc2 and PR+Inc3.

PR+Inc1 uses a unitary *contribution gain* (i.e., $G(\chi^m, t, s, p)=1$).

PR+Inc2 uses a size gain $K(s)=1.5$ when the average size of the fragments stored by nodes is larger than 682 kB (average audio fragment size) and 1.0 otherwise. The popularity gain $L(p)=1.25$ when the average popularity of the items stored is above 0.8 and 1.0 otherwise (i.e., when nodes are storing rare items mainly). Last, for the age gain we use the staircase function presented in Figure 6, with a normalization value of T=2 minutes. For sessions larger than 20 minutes, the gain remains constant at $H(t)=2.0$.

PR+Inc3 uses the same $H(t)$ function as Pr+Inc2 and integrates $K(s)$ and $L(p)$ in a single metric:

$$S^* = \frac{1}{k \cdot n} \cdot \left(\sum_{\forall i} \sum_{\forall j} \chi_{i,j} \cdot s_i^{p+p_0} \right) \qquad (22)$$

where p is the item's (CMF) popularity and $p_0=0.8$. This metric is then normalized with the average fragment size to define a staircase function with a max gain value of 2.0. The purpose of this metric is to give a larger weight to the content availability contributions for rare items.

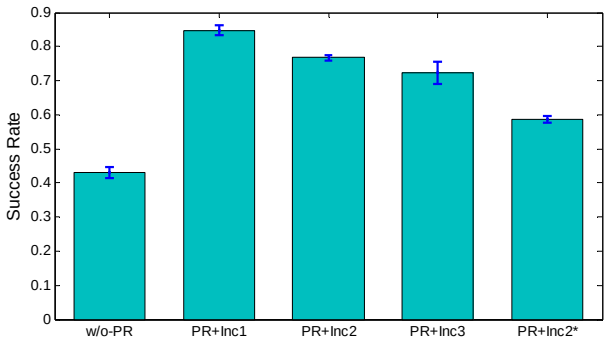

Figure 7. Content Retrieval Success Rate.

6.2. Results

The first property we corroborate is that the performance of the system improves when using PR redundancy with incentives. Figure 7 shows that the content retrieval success rate of the system without PR (first bar) is inferior that any of the variants of our mechanism. The last bar in the figure uses PR+Inc2, but with 30% of the nodes acting as "free-riders" (i.e., these nodes do not participate in the redundancy maintenance process). Even under these circumstances, the performance of the system is better than without PR, by almost 50%.

Figure 8 presents a scatter diagram for the average transmission bandwidths nodes receive to download files. The trend is clear, larger content contributions secure better performance for nodes. In other words, the incentives mechanism defines a system that distributes resources fairly. Figure 9 presents a complementary view of the evolution of the system. It shows how the average cost function decays as nodes increase their content contribution during the network's lifetime. At the beginning of the simulation, most nodes have a low contribution and the redundancy maintenance process triggers a large number of repairs. As nodes incorporate more and more fragments to their storage space, the average cost function in the system decays. After a

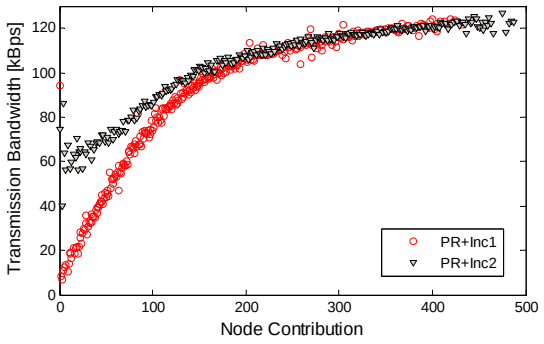

Figure 8. Util Function for PR+Inc1.

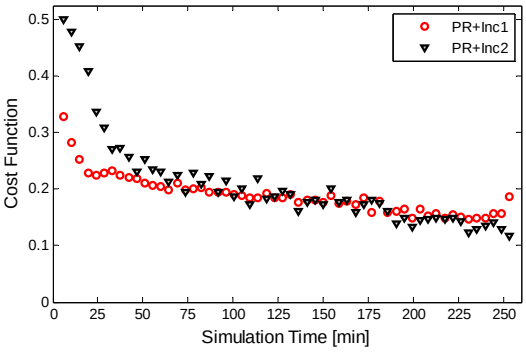

Figure 9. Replication Probability for Pr+Inc2.

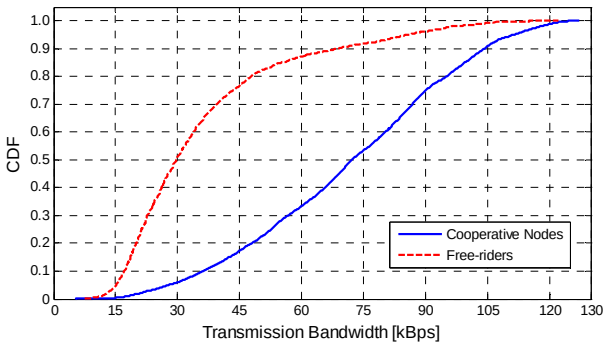

Figure 10. Transmission Bandwidth CDF.

transient period of approximately one hour, the average replication probability function reaches a slow decay rate.

For the results presented in the next plot, Figure 9, we let 30% of the node population act as *free-riders*. That is, these nodes do not participate in the redundancy maintenance process. Still, they perform basic peer functionalities truthfully (e.g. forward lookup requests and even uploading the few items they hold, upon request). The figure shows that the median transmission bandwidth of the cooperative nodes is more than twice the median of the *free-riders*. The last bar in Figure 9 presents the average content retrieval rate for this system setting (i.e., 0.59). If we disaggregate the content retrieval success rate of cooperative and *free-riders* it is clear than the nodes participating in the redundancy maintenance process get a good return on their investment (i.e., fragments stored). Cooperating nodes have a content retrieval rate of 0.73 while *free-riders* only achieve 0.26.

7. Conclusions and Future Work

In this paper we introduced two mechanisms to improve content availability in P2P networks: a hybrid redundancy scheme, referred to as Proactive Repair (PR), and an incentive-based scheme. We show analytically that PR redundancy outperforms other methods, including exact-MBR network coding with respect to its repair bandwidth requirements. This is a fundamental scalability concern in P2P networks; thus, our redundancy scheme has better scalability prospects than EC and exact-MBR redundancy. Experimentally, we show that a distributed redundancy maintenance process for PR augmented with incentives is feasible and effective in improving content availability. Our prototype implementation of these mechanisms is capable to improve the content retrieval of the system from 43% to 85%.

The scalability and performance of P2P networks is rooted in the cooperation of its members. By enforcing fair exchange of resources among nodes, the incentives mechanism we introduce in this paper foster nodes cooperation. Thus, we speculate that it makes the system more robust and scalable. The incentives mechanism grants low transmission bandwidths to nodes with small content contributions and for nodes with large content contributions the incentives mechanism allow them to achieve maximum transmission bandwidths. Even in the presence of a large fraction (30%) of non-compliant nodes (i.e., free-riders) the content availability and fairness properties of the system are preserved.

For future work, we envision the implementation of security features to prevent nodes from cheating. In particular, we believe that some form of self-certifying data and audits are good candidates to implement such features within the distributed architecture of our system. In addition, modeling different adversarial models for non-compliant nodes could prove important to test the robustness of the system. Lastly, we believe that gamification [21] could be employed in our system to influence user behaviors. Properly designed, the content contributions of nodes can be used as the basis for a game-inspired user interface to promote users cooperation.

References

[1] G Daniel S. Menasce, Antonio A. Rocha, Bin Li, et.al., Content Availability and Bundling in Swarming Systems, CoNeXT'09, Rome

[2] Charles Blake, Rodrigo Rodriguez, High Availability, Scalable Storage, dynamic Peer Networks: Pick Two, HotOS IX, 2003

[3] Hakim Weatherspoon and John D. Kubiatowicz, Erasure Coding vs Replication: A Quantitative Comparison, IPTPS'02

[4] Ranjita Bhagwan, Stefan Savage and Geofrey M. Voelker, Replication Strategies for Highly Available Peer-to-Peer Storage Systems, Tech Report, UC San Diego, 2002

[5] Rodrigo Rodriguez and Barbara Liskov, High Availability in DHTs: Erasure Coding vs Replication, IPTPS'05

[6] W.K. Lin, D.M. Chiu, Y. B. Lee, Erasure Code Replication Revisited, P2P'04

[7] Fan Wu, Tongqing Qiu Yuequan Chen, and Guihai Chen, Redundancy Schemes for High Availability in DHTs, IPSA 2005, LNCS 3758

[8] AG Dimakis, K Ramchandran, Y Wu and C Suh, A Survey on Network Codes for Distributed Storage, Proceedings of the IEEE, March 2011

[9] Zghaibeh, M. and K.G. Anagnostakis, On the Impact of P2P Incentive Mechanisms On User Behavior, in NetEcon+IBC. 2007: San Diego.

[10] Kan Zhang, Nick Antonopoulos, and Z. Mahmood, A Review of Incentive Mechanisms in Peer-to-Peer Systems, in First International Conference on Advances in P2P Systems. 2009, IEEE Computer Society.

[11] Panayotis Antoniadis, Costas Courcoubetis and Ben Strulo, Incentives for Content Availability in Memory-less Peer-to-Peer File Sharing Systems, ACM SIG on Ecommerce, 2005

[12] K. V. Rashmi, et al., Optimal Exact-Regeneration Codes for Distributed Storage at the MSR and MBR Points via Product-Matrix Construction. IEEE Trans. Information Theory, 2010

[13] Pinheiro, E., W.-D. Weber, and L.A.e. Barroso, Failure Trends in a Large Disk Drive Population. USENIX Conference on File and Storage Technologies, 2007.

[14] Hard drive manufacturer specifications. 2011 Available from: http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-701220.pdf

[15] Lehpamer, H., Transmission Networks Fundamentals, in Microwave Transmission Networks, Second Edition. 2010, McGrawHill. p. 15

[16] Dimakis, A.G., P. B. Godfrey, and Y. Wu. Network Coding for Distributed Storage Systems. in Information Theory. 2010.

[17] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz, Handling Churn in a DHT, USENIX '04, June 2004

[18] Xin Bai A , Dan C. Marinescu A , and et. al., A Macroeconomic Model for Resource Allocation in Large-Scale Distributed Systems. Parallel and Distributed Computing, 2008.

[19] Modelnet. Available from: http://modelnet.ucsd.edu.

[20] Z. Yao, et al. Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks. in IEEE ICNP. 2006.

[21] Deterding, S. Meaningful Play: Getting Gamification Right Google Tech Talk [Webcast] Jan 24, 201

Octavio Herrera-Ruiz received the B.E. in Electrical Engineering from the National Autonomous University of Mexico in 1997, where he was head of the network operations department from 1997-1999. Received a M.S. and a Ph.D. in Telecommunications from the University of Pittsburgh in 2001 and 2012 respectively.

Taieb Znati did his undergraduate studies in Tunisia and Paris, received a master degree from Purdue University in 1983, and finished his doctorate at Michigan State University in 1988. He is Professor at the department of Computer Science at the University of Pittsburgh, with a join appointment in the department of Telecommunications. He has served as Senior Program Director of Advanced Networking Research at NSF (2001-2005) and later as director of the Division of Computer and Networked Systems (2007-2009).