Coding and Scheduling Considerations for Peer-to-Peer Storage Backup Systems

Yitzhak Birk
Electrical Engineering Department
Technion, Haifa, Israel
birk@ee.technion.ac.il

Tomer Kol
Google Haifa Engineering Center
Haifa, Israel
tomer.kol@gmail.com

Abstract

Peer-to-peer storage- and in particular backup-system architectures have recently attracted much interest due to their use of "free" resources, with disk spindles and communication bandwidth being at least as important as storage space. This paper complements most of the works on this topic, whose focus was on metadata, security, locating the stored data, etc., by focusing on the data itself. It offers important design considerations and insights pertaining to the composition of erasure-correction code (ECC) groups, their size and the level of redundancy. Dynamic issues such as the co-scheduling of the concurrent reconstruction of multiple ECC groups are also explored. Finally, we identify an interesting natural match between asymmetric communication bandwidth (e.g., ADSL) and a hierarchical reconstruction architecture aimed at alleviating bottlenecks at the reconstructing node.

1 Introduction

The abundance of underutilized storage capacity in networked computers and the cost of centralized storage and backup solutions have resulted in a growing interest in peer-to-peer architectures: participating nodes contribute some of their resources, namely storage space and access capacity, processing power and network bandwidth, in return for some incentive such as data protection offered to group members, faster retrieval of information, etc.

Some distributed storage system architectures have been suggested and even implemented. Examples of such systems include OceanStore [11, 16], Chord and CFS [10, 17], Freenet [7] and Farsite [6]. Examples of P2P backup systems include pStore [3], DIBS [13], iDIBS [14] and Pastiche [9] to name a few, all trying to harness resources contributed by peers to create a fault tolerant system.

These and other studies focused on issues such as location management, meta-data, privacy and anonymity, as well as on fairness of resource contribution by the users. The parameter selection of the erasure correcting code (ECC) used was not treated beyond the need of R redundancy blocks in order to tolerate R failures. Also common to those systems is the treatment of a recovery process with some given set of failed (e.g., using a fail-stop model to represent disk crashes) or in some cases Byzantine nodes. They then studied the probability that, at the time of a failure, the available subset of blocks would suffice for recovery. Lillibridge et. al. [12] commented that the reconstructing process can retry retrievals in the hope of gathering enough blocks.

Another class of Peer-to-peer systems is file sharing systems such as eMule [1] and bit-torrent [8]. Here, the performance and availability of data depends on its popularity, and there is no mechanism to guarantee availability or even try to ensure the availability of data in the event that the original "master" copy is unavailable.

This work complements prior art by focusing on the data itself: its placement and ECC parameter selection, as well as data transfer and scheduling issues when recovering multiple objects. The focus is on providing an extremely robust system for longer-term storage and retrieval of data and on more uniform data protection, unlike some common P2P systems that are geared towards dissemination of popular data.

The remainder of the paper is organized as follows. Section 2 presents the problem characterization and the models used; Section 3 explores static design considerations related to the ECC used and ECC group composition; Section 4 discuss dynamic issues such as the co-scheduling reconstruction of multiple ECC groups and availability enhancement and Section 5 offers concluding remarks.

2 Problem characterization and models

2.1 Rethinking the P2P storage/backup model

A simple, somewhat simplistic characterization of the backup problem is that when the local/primary copy is destroyed, the system should be able to recover it[1]. However,

[1]For an ECC computed over a set of K blocks and constructing R "redundant" blocks, at least K blocks should be available when a failure

0-7695-3097-4/08 $25.00 © 2008 IEEE
DOI 10.1109/SNAPI.2007.19

characteristics such as the node failure model and limited communication and storage bandwidth can have a profound effect on the relative merits of different system designs.

The fail-stop model, commonly assumed in analysis of backup systems, is important and is often a good match to local failures. However, we observe that unavailability of remote nodes is usually transient. For example, a node may have communication problems, be turned off for the day, or may be unwilling to participate in the recovery process due to heavy load.

A second observation is that limited communication bandwidth usually prevents reconstruction from being instantaneous or even nearly so. As an extreme example, fully recovering a 100GB disk via a 5Mb/s network connection will take well over a day.

Based on these observations, the traditional measure, namely the probability that a sufficient number of blocks is available *at the time of failure*, is thus too harsh because more nodes may become available in time to be useful; yet, it does not guarantee anything because available nodes may become unavailable before the data they hold is retrieved.

In this work, we relax the requirement for instant availability at the time of the client's request. Instead, for any given transient failure random process, we examine the duration of the recovery process beyond the minimum dictated by the recovering node's communication link. We explore design trade-offs involving ECC group size and degree of redundancy as well as ECC group composition. Dynamic policies such as the joint scheduling of the concurrent reconstruction of multiple ECC groups are also explored.

The characteristics, or dimensions that span the problem space can be grouped as follows.

- **Reconstruction environment:** node availability and communication characteristics (e.g., bandwidth and whether it is symmetric).
- **Client requests:** amount of data; type of consumption (bulk, stream) and performance goal (expected mean latency, (probabilistic) latency guarantees, etc.)

We next discuss these issues in some detail.

2.2 The reconstruction environment

A ***node*** may be physically down due to a malfunction or because it has been switched off. It may be disconnected due to communication problems, which is equivalent to being down. A node that is physically up and connected may nonetheless refuse to grant a peer-to-peer service request as a means of controlling its level of participation.

The transitions among node states can be modeled using one of (or a combination of) several models such as:

- **Random.** Block availability is modeled as an i.i.d. Bernoulli process, whereby the aggregate number of willing nodes is Binomially distributed. While the physical availability of a node is often not random, a node can use randomization to control the amount of resources contributed to the peer backup activities. For convenience, we model time as a sequence of fixed-size time slots. Note that while node availability is independent from slot to slot, the state of the entire group *is not*. If n blocks have already been retrieved, there are only $(K + R - n)$ remaining relevant blocks.
- **Markovian birth-death process.** The rates of birth/death depend only on the numbers of blocks in each category. Holding times in each state are modeled as an exponential random variable. Note that residual holding time is memoryless (exponential RV), but the probability distribution of available nodes is state-dependent.
- **Periodic.** Very short periods may express a restricted participation policy, while longer ones may reflect business hours after which computers are shut down or during which P2P participation is refused. Availability times may furthermore vary, reflecting different time zones.

In our analysis, we focus on the first two models unless noted otherwise.

Like block and node availability, the network resources available to the P2P storage/backup system are determined by a combination of physical and policy issues. The uplink and downlink bandwidths may be equal, as in a LAN, or "asymmetric" as in ADSL. Asymmetry can stem from the physical connection or reflect a policy. This work focuses on point-to-point connections and transfers. Other network types, such as broadcast networks, may influence the achievable rates and may require more complex algorithms. Some examples of information dissemination algorithms over a broadcast channel can be found in [2, 4].

Two models for block transfer time are used in this work: exponential service time and fixed data rate (transfer time proportional to block size). Our simulation study shows that both models yield essentially the same expected latencies, and exhibit the same relative behavior under various scheduling policies. One obvious difference is that the exponential model yields a larger variance in the results.

2.3 Client request (target data set)

Data recovery may vary between retrieval of a small object and reconstruction of an entire failed hard disk. We will consider the following representative cases: a single block; one ECC group (stripe); multiple ECC groups; and a stream.

The reconstruction of less than an entire ECC group may nonetheless require the retrieval of K blocks. One can eliminate the extra transfers by using small blocks so an application's data block constitutes an entire ECC group. How-

occurs.

ever, this would increase the communication-management overhead for larger reconstructions, would not reduce the number of disk seeks required for a single block recovery, and would increase this number for larger recoveries.

The major consumption modes of the data include **on-demand** (reading data in degraded mode), **bulk reconstruction** (e.g., a failed disk) and **streaming** (sequential consumption of the data). We next discuss design considerations, beginning with static (off-line) design.

3 Static design considerations

The static design considerations include the composition of an ECC group (whose blocks it comprises) and the parameters of the (K, R) ECC, considering implications of both the relative and absolute values of K and R.

3.1 Composition of ECC groups

Data owners. When selecting data blocks to compose an ECC group, the question arises whether to select all blocks from the same client or create a conglomerate with blocks coming from different clients.

Data distribution. The selection of nodes that will hold the blocks of a given ECC group, whether determined statically during the system design phase or determined dynamically during the creation of a new ECC group, should consider the possibility of correlated failures (as studied by Nath et. al. [15] in the context of centralized systems). As an extreme case, co-locating multiple blocks on the same machine practically guarantees that they will present a common mode failure pattern. Some correlated unavailabilities, such as those reflecting business hours, can be predicted. When time zones are involved, it may even be possible to create anti-dependencies, with one subset of nodes likely to be available when the other is not.

3.2 ECC parameter selection

An Erasure-Correcting-Code (ECC) can be used to disperse data and redundancy. The characterizing parameters of a (K, R) ECC are K - the number of data blocks in the ECC group, and R - the number of *redundancy* blocks. A *systematic ECC* is one that includes the information in its original form and simply adds R redundant blocks computed from it. A client may thus be able to directly retrieve exactly the blocks it needs without retrieving K blocks. However, this may be impossible or impractical due to unavailability or heavy load at the node holding a requested block, in which case the client will have to resort to retrieving K blocks. In the remainder of the paper, we will focus on the case wherein the retrieval of K blocks is required. For a system wherein a verbatim copy may be available, our results should be combined with the probability and expected latency of retrieving a verbatim copy of the data.

During reconstruction, the state of an ECC group is described by: N_r - the residual number of required blocks; N_a - the number of available relevant blocks (node is up, transfer hasn't started yet) and N_t - the number of in-transfer blocks (transfer started but has not completed yet).

Initially, $N_r = K$, $N_t = 0$; reconstruction ends when $N_r = 0$, $N_t = 0$. Unless stated otherwise, we assume that all block transfers complete successfully, so N_r will exclude blocks whose transfers have commenced.

The values of K and R affect several aspects of the system: the ratio R/K determines the **storage overhead** of the erasure-correcting code; also, for a given R/K ratio, increasing K often requires increasing the buffer space used to hold the information and calculate the ECC. For stream consumption, the use of a larger K turns the uniform consumption rate typical of streaming into bursts of stripe retrievals.

The **retrieval overhead** of using a (K, R) ECC is up to a K-fold increase in disk accesses whenever the verbatim copy of the original block is not available (a single-block request). A special case worth mentioning is $K = 1$, or *replication*. In this case redundancy blocks are copies of the original data, and there is no overhead when reading, regardless of block size. The down side is reduced protection for the same level of redundancy. As shown in [5], when recovering K data blocks, the minimum expected reconstruction time for a given redundancy level is achieved when these blocks constitute a single ECC group.

ECC group size. With larger ECC groups, there is less uncertainty regarding the expected reconstruction time as stated formally in lemma 1 (see [5] for details).

Lemma 1. *For blocks with i.i.d. availability distributions, increasing $K + R$ decreases the probability that $T_{reconstruction} > E\{T_{reconstruction}\} + c\,\sigma\{T_{reconstruction}\}$ for any $c > 0$.* □

An implication of the reduced uncertainty is that as the required guarantee of meeting a certain reconstruction latency bound becomes stronger, it may be beneficial to increase K. In fact, even if requests are always for a fixed known number of blocks, the optimal value of K may be greater than this number despite the increased reconstruction overhead.

Fig. 1 depicts the results of a simulation study of reconstruction time for a Markovian availability model, deterministic transfer times and no mid-transfer failures. The curves are for the expected (bottom), 90% confidence and 99% confidence (top). In this example, even if the number of required blocks in a reconstruction request is below 10, increasing K (up to 10), while increasing expected latency, reduces the latency that can be achieved with high probabil-

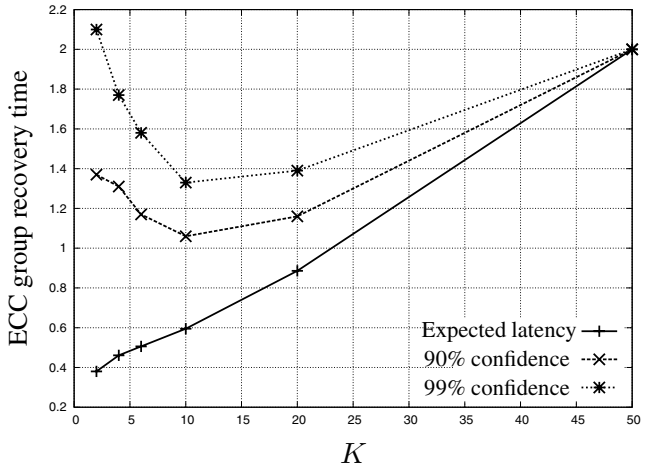


Figure 1. ECC group recovery times: expected, 90th percentile and 99th percentile. $(R = K/2)$**, Markovian availability with** $\lambda = \mu = 1$**, deterministic block transfer time** $t_{transfer} = 0.04$**.**

ity.

Finally, while different client request patterns and node availability may motivate the tailoring of different ECC parameters and block sizes to different parts of the system, it is important to bear in mind that there are many advantages to uniformity.

3.3 ECC group recovery process

The reconstruction process was analyzed and simulated for several availability models, including the commonly used death-birth process and slotted random availability (representing, for example, clients that use randomization for load balancing and control). As an example, Fig. 2 depicts the state diagram of an ECC group, with block availability modeled as a Markovian birth-death process of nodes coming up and going down, respectively. A state is defined by the three values of (N_r, N_a, N_t). Note that as the number of required blocks decreases (descending a row), the range of reachable states shrinks.

Let λ (μ) represent the birth (death) rate. The residual reconstruction time is

$$T(N_r, N_a, N_t) =$$

$$= \begin{cases} 0 & N_r = 0, N_t = 0 \\ T(N_r - 1, N_a - 1, N_t + 1) & N_t < N_t^{max}, N_a > 0 \\ \frac{1}{(N_r+R-N_a)\lambda + N_a\mu + N_tC}(1+ & \\ \quad (N_r + R - N_a)\lambda\, T(N_r, N_a + 1, N_t) & \text{otherwise} \\ \quad +N_a\mu\, T(N_r, N_a - 1, N_t) & \\ \quad +N_tC\, T(N_r - 1, N_a - 1, N_t - 1)) & \end{cases}$$

Note: for $N_t = 0$, the term $N_tCT(N_r{-}1, N_a{-}1, N_t{-}1) = 0$ so we do not define $T(*, *, -1)$. Likewise, for $N_a = 0$, the term $N_a\mu T(N_r, N_a - 1, N_t) = 0$. In the general case, $N_t = O(N_t^{max})$.

The system has a somewhat simpler form when $N_t^{max} \geq K$ or $N_t^{max} = 1$. For facility of exposition and analysis, we assume that the transfer rate of any given block is independent of the number of blocks being transferred concurrently to the reconstructing client. This, along with the limitation $N_t < N_t^{max}$, represents the case of asymmetric bandwidth (e.g., ADSL) wherein the downlink of the reconstructing client is several times faster than the uplinks of its peers. In other situations, this is imprecise but is nonetheless a close approximation in view of the assumed large ratio between node availability time constants and the transfer time of a block.

The above equations, as well as the ones for the slotted random availability model, can be solved progressively from $N_r = 0$ to $N_r = K$. Finally, combining with the probability distribution of N_a when the recovery starts, we obtain the expected recovery latency. The 90% and 99% results were obtained by simulation.

Other models, including periodic availability, are also possible. A real-world case is likely to contain elements of the above "pure" availability patterns as well additional ones. When designing a system, the architect should consider the relevance of the various models and assess the related observations and insights.

One result, to be expected based on the foregoing insights, is that even if the expected number of concurrently active nodes does not suffice for the retrieval of all required blocks, reconstruction time may be very close to or even equal to the minimum that is determined by the limited communication bandwidth. This is because the only requirement for this is that whenever the reconstructing client is ready to fetch the next block, at least one relevant block is available.

As R/K (the redundancy level) grows, two phenomena take place: the expected number of available nodes increases, and its variance becomes smaller. As a result, the percentile curves converge at smaller values of K.

As mentioned in Section 3.2, we assume that in order to recover even a single block, K blocks should be retrieved. If small requests are the common scenario, this favors small values of K for reasons of expected transfer latency as well as easier computation and smaller buffers.

A system architect should also bear in mind the implications of increasing K, as depicted in Fig. 1. If the design goal is providing a high probability of not exceeding some limit rather than merely minimizing the expected reconstruction time, it is sometime better to increase K in order to reduce the variance of the recovery time. As can be seen from the above figures, this phenomenon is most pronounced for small K (less than 10). Finally, increasing K should be balanced with the increase in the expected values and related issues such as a larger required buffer space.

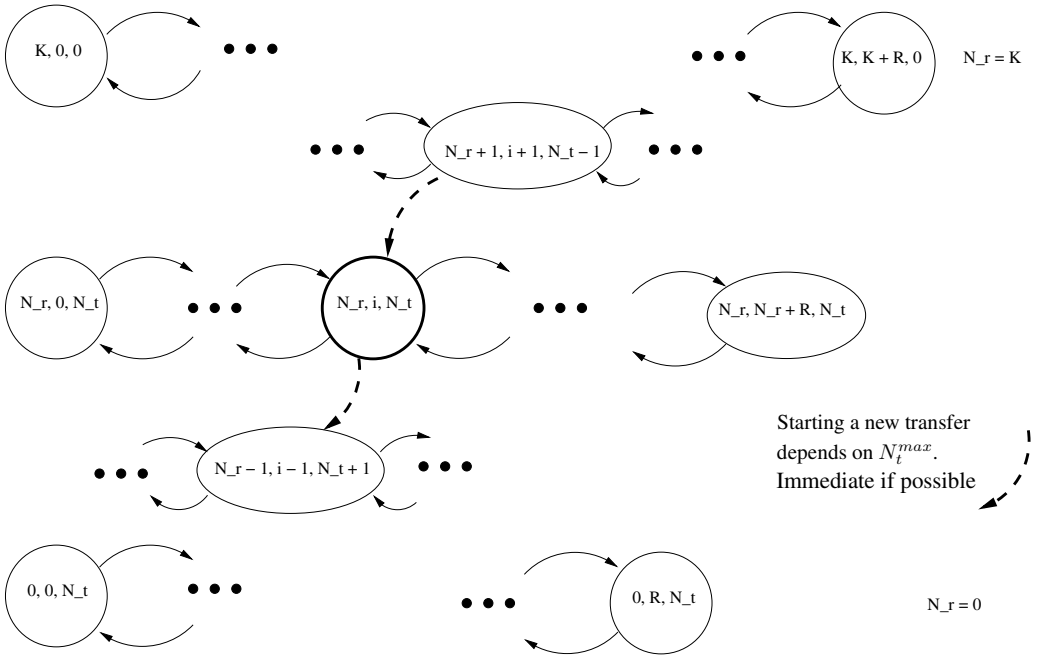


Figure 2. ECC group state diagram for Markovian availability

4 Dynamic design considerations

4.1 Co-scheduling the recovery of multiple ECC groups

The need to concurrently recover multiple ECC groups arises, for example, whenever more than K blocks need to be recovered. With bandwidth or some other factor limiting the number of concurrent block transfers, the reconstructor may have to choose among available blocks of different ECC groups. We refer to the selection of the next block to transfer as the recovery scheduling policy.

Region of interest. In the extreme cases of availability, namely when blocks are always available and when there is at most one candidate, the scheduling policy is obvious. In the latter case, a work-conserving policy, namely no unforced idle periods, should be used [5]. Between these extremes, there may be multiple available blocks as well as idle periods, so the optimal scheduling policy is not obvious.

Scheduling policies

Below are several observations and insights pertaining to the selection of the data transfer scheduling policy. We refer to on-line policies with no specific knowledge of future availability of nodes. (See [5] for further discussion and proofs.)

Observation 2. *For minimization of the expected make-span of the recovery of multiple ECC groups with similar independent availability patterns, there is an advantage to favoring blocks of the ECC group that has the largest N_r (residual number of required blocks).*

Rationale: whenever the transfer of a block is completed, the number of relevant blocks drops by one – the just-utilized block. However, once K blocks have been retrieved, the group's recovery is complete and the R remaining relevant blocks become irrelevant regardless of their availability. Keeping more groups "alive" therefore maximizes the number of relevant nodes (blocks), thereby reducing the probability of (temporary) starvation.

Definition 3. *Given ECC groups G_i and G_j being recovered, a* Most Remaining First (MRF) *scheduling policy is one that prefers to retrieve from G_i iff $N_r(i) > N_r(j)$.*

Theorem 4. *Given two i.i.d. Bernoulli slotted random availability (SRA) ECC groups and $N_t^{max} = 1$, MRF is the optimal online policy for minimizing the make-span of recovering both ECC groups [5].* □

This is not true for offline policies.

In general, while hard to quantify analytically, the influence of selecting the ECC groups with the most required blocks is greater toward the end of the reconstruction process, as the number of relevant blocks decreases and the probability of starvation increases.

Observation 5. *For minimization of the make-span of a multiple-ECC-group recovery with node availability characteristics that are not memoryless, there is an advantage to preferring to retrieve blocks from the ECC group with the smallest number of available relevant blocks.*

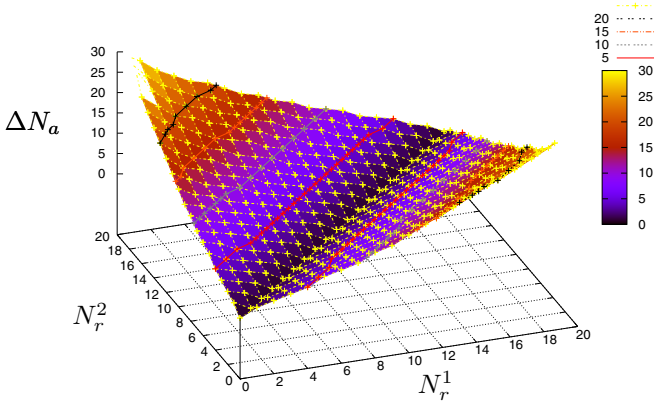


Figure 3. Relative influence of ΔN_r and ΔN_a

Motivating example. Consider the case of a node availability pattern that it is not memoryless, and two ECC groups (G_1 and G_2) with the following states: $N_r^1 = 1$, $N_a^1 = 1$, $N_r^2 = 2$ and $N_a^2 = 100$. Transferring from G_2 first, the probability that N_a^1 will drop to zero and we will have to wait for some block of G_1 to become available is higher than the probability that, if we first transfer the block from G_1, all 100 available blocks will become unavailable.

Whenever node availability is not memoryless, the two aforementioned observations may advocate contradictory policies, as is the case in the motivating example. In such cases, the optimal policy depends on the states of the two groups in a non-trivial way.

As an example, Fig. 3 depicts the relative influence of the two observations on N_r and N_a for two ECC groups, with $K = 20$, $R = 10$, per-node birth and death rates of 1, and an exponential transfer completion rate of 500. For each (N_r^1, N_r^2) combination, the z-axis value shows the difference between N_a^1 and N_a^2 above which one should best favor the group with the *smaller* N_r, namely the difference above which the influence of observation 5 outweighs that of observation 2. Employing different scheduling policies can yield a make-span difference of up to tens of percents for certain parameter values and states. However, when factoring in the probability to reach such a state, the influence of policy selection is usually at most a few percents.

Summarizing, we have found that the performance of multi-ECC group reconstruction is usually not very sensitive to the scheduling policy, though there may be substantial differences in some specific system states. Overall, preferring to transfer from the group with the largest N_r, thereby keeping more groups "alive", is simple to implement and performs close to the optimal online policy for the availability patterns studied. (A possible drawback of this scheme is that as more groups are under construction, additional memory buffers may be required.)

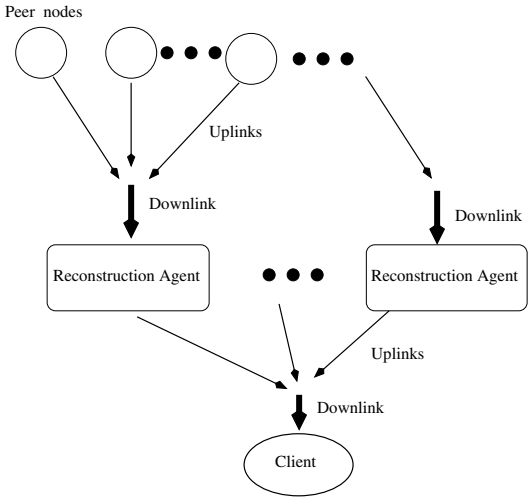


Figure 4. Distributed reconstruction scheme.

4.2 Backup and reconstruction schemes

The nature of the various resources (e.g., communication bandwidth) that can be harnessed for a P2P storage/backup system influences the relative merits of possible schemes. One interesting example is asymmetric data rates with up to an order of magnitude difference between a node's downlink and uplink speeds, e.g., ADSL or cable modems. We next illustrate this issue.

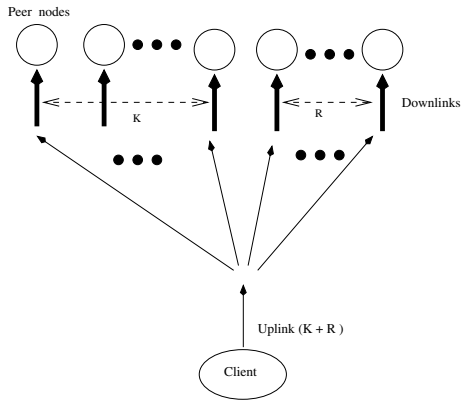
Distributed recovery

The general load across the network due to a recovery process should not be too high, as the load is distributed across multiple peers. The recovering node, however, is an activity hot spot. As such, it may become a bottleneck for both transmission and computation. To mitigate this, a client may use other nodes as *reconstruction agents* (Fig. 4).

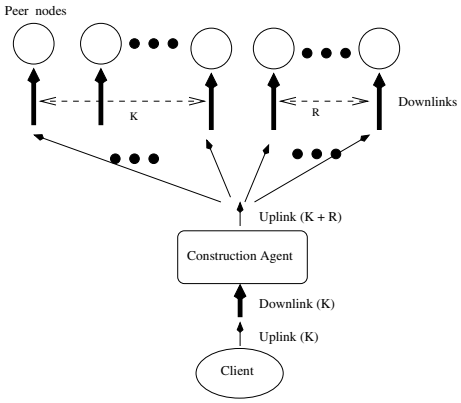
If the client requires only part of the data in the ECC group, the recovery agents can filter and send to the client only the parts that it needs, thereby also alleviating the communication bottleneck. In this case, when combined with asymmetric bandwidth, the resource utilization maps nicely onto the available bandwidth: a storage peer only sends one block over its slow uplink; a reconstruction agent receives multiple data blocks over its fast downlink, and only sends the single required block over its slow uplink; the client, in turn, receives the multiple data blocks (one from each reconstruction agent) over its own fast downlink.

Distributed creation of backup ECC groups

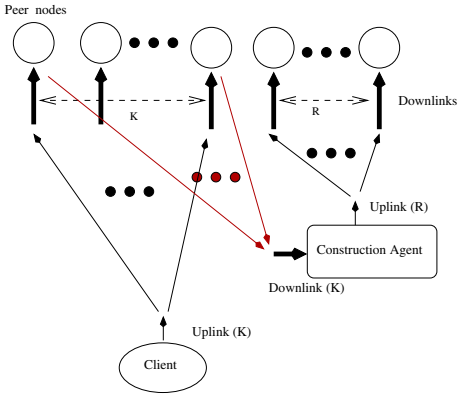
Creating a backup using Reed-Solomon or Lubby Transform codes requires that some node take K blocks and create additional R redundancy blocks, where R is often much smaller than K.



(a) Basic scheme



(b) Using a construction agent



(c) Distributing transfers

Figure 5. Distributed ECC group deployment schemes.

In the basic scheme (Fig. 5a), with all data blocks in an ECC group coming from the same client, this client has to compute the additional R redundancy blocks and send the $K + R$ blocks to peer nodes over its slow uplink.

Instead, the scheme depicted in Fig 5b offloads the computation of ECC groups from the client, and is also suitable for ECC groups consisting of data blocks from multiple clients. With this scheme, any given set of K data blocks are sent to a *backup agent*. The data owner needs to upload up to K data blocks (in the case of a single source). The backup agent generates R redundancy blocks, and sends the $K + R$ code blocks via its uplink to the peer nodes that will hold them. Each peer node thus receives a single block over its downlink.

Table 1. Communication during ECC group creation, assuming a single data origin node.

	Baseline scheme (Fig. 5a)		Improved scheme (Fig. 5c)	
	Uplink [blocks]	Downlink [blocks]	Uplink [blocks]	Downlink [blocks]
Source node	K	0	K	0
Backup gent	K+R	K	R	K
Peer holding data block (one of K)	0	1	1	1
Peer holding redundancy block (one of R)	0	1	0	1

An improved scheme, depicted in Fig 5c, demonstrates that it is possible to reduce the load on uplinks by combining the use of a systematic ECC and intelligent routing. The client(s) owning the original blocks sends them directly to K destination peers. A *backup agent* can then retrieve the K blocks from the peers, calculate the ECC and then send the newly created R redundancy blocks to their destinations – the peers that will store them.

Note that while the backup agent must still receive the original K blocks, it only has to send only R blocks. The uplink bandwidth savings can easily be a factor of 3 or more. If the network connection is asymmetric, the uplink bandwidth is a much scarcer resource. The chore of sending the K data blocks (that are part of the systematic ECC group) to the peers is carried out by the owner(s) of the data blocks. The total communication counted in messages is not reduced, as the K block transmissions avoided by the backup agent are replaced with transmissions by the peers holding them to the backup agent. The transmission load, however, is distributed more evenly, with some of it moving from the backup agent to the peers holding blocks. As usually $K > R$, the reduction in the backup agent's uplink traffic is substantial, and the overall (up and downlinks) traffic to/from the backup agent is almost halved.

Whenever all K blocks originate from a single node, that node's uplink traffic is still $O(K)$. Additionally, when compared with the basic scheme whereby the client generates the redundancy blocks by itself, in the third (as well as in the second) scheme the original client saves the need to upload R blocks, thereby alleviating load on its slow uplink.

Table 1 summarizes the communication (associated with data transfer) required for constructing an ECC group.

4.3 Dynamic availability enhancement.

The transfer time of a single block is expected to be short relative to changes in the network and the transferring node's state (and thus its policy based decisions). Hence we have so far assumed that once a particular block transfer was started, it will complete successfully with probability tending to one. This assumption is referred to as *"mini-ratchet"*.

The *full-ratchet* model goes a step further: any required block that becomes available is assumed to stay available for the purpose of the reconstruction process that is under way. The rationale behind this model is twofold: (a) It is reasonable to expect/demand that a node that commits to supplying a block will not change that decision, and (b) the implementation can contain proactive mechanisms to reduce the likelihood of blocks becoming unavailable.

Our results indicate that while using a ratchet scheme presents advantages over no ratchet, the difference between mini and full ratchet is small whenever the availability patterns are smooth over time. The importance of ratchets increases when nodes are likely to go down for long times.

5 Conclusion

Our focus in this work has been on the data itself, its organization and transfer rather than on the metadata and management related issues. We presented a clear characterization for the reconstruction process of data dispersed in a peer-to-peer system using an erasure correcting code. Realizing that availability is often transient and that reconstruction is not instantaneous, we advocate the measure of data reconstruction latency and an associated confidence level rather than the traditional measure of the probability that a sufficient number of blocks is available at the time of failure.

The set of observations, design considerations and insights presented in this work can help architect better and more efficient P2P storage/backup systems. Finally, the interplay between the data related aspects presented in this work and the metadata related aspects that have been the focus of much past work is a topic for future research.

References

[1] eMule: A peer-to-peer file sharing system. `http://www.emule-project.net/`.

[2] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol. Index coding with side information. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 197–206, Washington, DC, USA, 2006.

[3] C. Batten, K. Barr, A. Saraf, and S. Trepetin. pStore: A secure peer-to-peer backup system. Technical Memo MIT-LCS-TM-632, MIT Laboratory for Computer Science, October 2002.

[4] Y. Birk and T. Kol. Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients. *IEEE Transactions on Information Theory*, 52(6), 2006.

[5] Y. Birk and T. Kol. Peer-to-peer storage backup systems — coding and scheduling considerations. CCIT Technical Report 637, Electrical Engineering Department, Technion, August 2007.

[6] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. *SIGMETRICS Perform. Eval. Rev.*, 28(1):34–43, 2000.

[7] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[8] B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[9] L. P. Cox, C. D. Murray, and B. D. Noble. Pastiche: Making backup cheap and easy. In *OSDI*, 2002.

[10] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP'01)*, Chateau Lake Louise, Banff, Canada, October 2001.

[11] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'00)*. ACM, November 2000.

[12] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard. A cooperative internet backup scheme. In *Proceedings of the USENIX Annual Technical Conference 2003 on USENIX Annual Technical Conference (ATEC'03)*, 2003.

[13] E. Martinian. DIBS: Distributed Internet Backup System. `http://web.mit.edu/~emin/www/source_code/dibs/`, 2004.

[14] F. Morcos, T. Chantem, P. Little, T. Gasiba, and D. Thain. iDIBS: An improved distributed backup system. In *Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06)*, pages 58–67, 2006.

[15] S. Nath, H. Yu, P. B. Gibbons, and S. Seshan. Subtleties in Tolerating Correlated Failures. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation NSDI'06*, May 2006.

[16] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: The OceanStore prototype. In *Proceedings of the Conference on File and Storage Technologies (FAST'03)*, pages 1–14, San Francisco, Apr 2003.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM'01 Conference*, San Diego, California, August 2001.