A hybrid redundancy approach for data availability in structured P2P network systems

Guangping Xu, Gang Wang, Jing Liu

Information Technology Science College, Nankai Universitity, Tianjin, China

Email: xugp2008@yahoo.com.cn

Abstract:

For practical deployment of peer-to-peer (P2P) systems, it is one of the most important and challengeable aspects to achieve high data availability in structured P2P systems since the environment is much scalable and dynamic.

The paper utilizes the hybrid of two data redundancy schemes, namely replication and erasure coding, to improve system availability to deal with churn. To mask or hide the high churn from the short-lived but churn-frequent peers and permanent failure peers, we use replication among the nodes in a certain interval that can be considered as a virtual node. Then with an erasure-coded redundancy scheme, we consider that a set of virtual nodes that cooperatively provide guaranteed over the networks. The paper presents the hybrid redundancy prototype and protocol, analyzes the stochastic behaviors for data availability of single virtual node and cluster. The approach is effective through the evaluation with an empirical trace.

Keywords: data availability; structured P2P systems; replication; erasure coding; churn

Paper type: Regular paper

The corresponding author: Guangping Xu; *Email:* xugp2008@yahoo.com.cn

Phone: 86-22-81238812; *Fax:* 86-22-23504780

Mail Address: Computer Science and Technology School, No.263 Hongqi Nan Road, Nankai district, Tianjin City, China.(Zip: 300191)

1. Introduction

Structured peer-to-peer (P2P) overlay networks has being investigated to build decentralized, cooperative systems including content distribution, information retrieval and cooperative storage systems etc. in much scalable and dynamic environment. The applications usually employ Distributed Hash Tables (DHTs) [1, 2] as an infrastructure to provide an efficient way for storing and locating data. However, they do not offer preferable guarantees about data availability for network nodes may join, leave, and fail concurrently and frequently (i.e. churn). Due to the dynamic and scalable nature of the network, it is an important foundational and challengeable issue to deploy P2P applications. Thus, it remains unclear whether the dependable potential of P2P systems can be realized for data storage in practice.

Data items are the general notation that represents files or blocks of files stored in systems. Usually higher availability and better performance can be achieved by data redundancy to mask and cope with node failures in structured P2P overlay networks. Data redundancy is typically implemented in one of two approaches, namely replication and erasure coding. In replication approach, multiple copies of the same data item are distributed to different peers. Thus, an item can be accessed as long as at least one replica is available. By contrast, in erasure coding approaches an item is encoded using an erasure coding, and then divided into smaller fragments for storage in many different peers. As long as a certain minimum number of peers in which the coded fragments resided are available, the original data item can be recovered and accessed from the fragments.

The paper utilizes the hybrid of the two data redundancy schemes, namely replication and erasure coding, to improve system availability to deal with churn. First, to hide the high

churn from a portion of short-lived peers and the permanent failures of peers, we use replication among the nodes in a certain interval that can be considered as a virtual node. Then with erasure coding, we consider that a set of virtual nodes that cooperatively provide guaranteed over the networks. Finally, we try to quantitatively evaluate data availability of the hybrid approach with an empirical trace.

2. Backgrounds and related works

2.1 Node dynamics

In structured P2P systems, churn can have important effects on a P2P system. Peers may be unavailable due to network isolation, system crash etc. Many peers may become unavailable and then return to service without any data loss. Such churn leads the number of peers in systems to grow or shrink and changes the availability. Node failure churn exhibits both temporary and permanent characteristics [10]. A temporary failure occurs when a node departs the system for a period and then comes back. Any data stored on the node becomes unavailable during this period and is preserved across a transient failure, such as a reboot or temporary network disconnection, but is not permanently lost There are many empirical measurements. For example, in Overnet system [14], it has observed that each peer joined and left the system over 6 times per day on average. In contrast, a permanent failure, such as host crash or disk failure, results in loss of the data stored on the node. A permanent failure corresponds to permanent loss of data resided on the node. We consider that a node is available if its data can be retrieved across the network, and an item is available if it can be reconstructed from the data stored on currently available nodes. In summary, node failure churn leads to data unavailability as a result of a combination of temporary and permanent

failures.

2.2 Data redundancy

Data redundancy is typically implemented in one of two schemes, namely replication and erasure coding. P2P storage systems have employed replication. CFS splits every file in a number of blocks that are then replicated a fixed number of times based on [1]. PAST applies the same protocol, without chunking files into blocks based on [2]. Replication heavily relies on duplication data to related nodes, and its organization is simple and is mostly employed.

Erasure Coding divides an item into *m* fragments which are then used to generate *n* encoded fragments. It has the property that any *m* (or slightly more) out of the *n* encoded fragments suffice to recover the original *m* data fragments. The redundancy scheme yields much higher probabilities of recovery and offers lower storage costs compared to replication compared to replication schemes, unfortunately, it introduces higher computational complexity. Erasure coding needs multiple communication and combination among related nodes.. For example in OceanStore [4] and TotalRecall [5], data items are replicated by using erasure coding.

The availability relationship among the mean availability of peers, the number of replicas and the required data availability for the two redundancy schemes can be found in [6, 7]. The paper utilizes the hybrid of two data redundancy schemes, namely replication and erasure coding, to improve system availability to deal with churn.

3. Redundancy placement management

3.1 Redundancy placement

We categorize a variety of placement strategies into two types: Locally Neighboring

Placement (LNP) and Globally Rehashing Placement (GRP). The metric is based on the distribution fashion of redundancy objects (replicas with replication and fragments with erasure coding of a data item) among nodes in systems.

In LNP redundancy objects are placed on the neighboring nodes of the owner node, which is a node responsible for the objects according to consistent hashing. The neighbors can be successor nodes, predecessor nodes or entry nodes of routing table. For example, [1, 18] employs a successor-list scheme in which an item is replicated on the successive nodes with closest to the owner. In contrast, in GRP the responsible node of each replica is determined by a rehash function in the global space. The key of each replica can be obtained by re-hash function and then the corresponding node can be determined. For example, [19] provides a symmetric replication scheme in which the identifier space is partition into equivalence classes and an item is duplicated on each of the nodes in an equivalence class.

In the paper, we employ the two placement strategies to distribute redundancy objects (replicas within a virtual node by LNP and fragments of by GNP) in the virtual identifier space.

3.2 Cooperative management

In such dynamic environment, cooperation between nodes is a fundamental property in P2P networks. With sufficient redundancy with many peers, at any moment couple of peers will be in the system to make a given data item available with high probability cooperatively. When a peer is available, the data stored on it contributes to the overall degree of redundancy and increases data availability. Many peers contribute to store the related data and supply the repair t of node failures for the availability of data in systems.

In the optimizations of several DHTs [16, 17], the way of virtual node has been adopted. To pursue load balance, some work defines virtual nodes that each powerful physical peer pretends to be several distinct peers, i.e., virtual nodes per physical node. Many other optimizations [22, 23] take multiple physical nodes in an interval as a virtual node, to reduce routing latency for it can choose physical node from virtual node with lowest RTT, and to improve fault tolerance for only one node per interval needs to survive to allow routing through the interval.

In the paper, we take the latter virtual node notation as the part of infrastructure. The physical nodes in the system perform cooperatively data maintenance. To mask or hide the high churn from the little portion of short-lived nodes and temporary failure nodes, we use replication among the nodes in a certain interval that can be considered as a virtual node. Then with erasure coding scheme, we consider that a set of virtual nodes that cooperatively provide guaranteed over the scalable and dynamic networks.

4. Prototype

In the section, we present the hybrid approach for data availability including redundancy layers and data maintenance protocol. A P2P storage system has persistent state. Thus, a node failure may be transient in that the node recovers with useful, for example if the failure was a network outage or a reboot. We suppose that when a node returns after a transient failure, such as a network outage, it does not affect data stored on disk. Since most p2p properties of interest require a sufficiently evolved system, our analysis of data availability is in the equilibrium state.

4.1 System overview

The system prototype has three parts including message handler, storage manager and data repairer in each peer. The message handler copes with the routing and forwarding requests and if it recognizes a message that is handled locally, the storage manager will be triggered to work by the incoming messages. At the same time, the repairer monitors the amount of redundancy and if the amount falls below the maintenance threshold, data recover mechanism will be trigger to increase the redundancy.

4.2 Redundancy

In the prototype, the whole identifier space is split into intervals, called *virtual nodes*. A virtual node could be taken as the abstraction of multiple physical nodes in an equal-size interval in which the physical nodes can cooperatively accomplish routing and storage requests. Each interval is maintained by multiple nodes at the same time and each peer in every virtual node contributes storage space and cooperatively makes stored data persistent and available (figure 1).

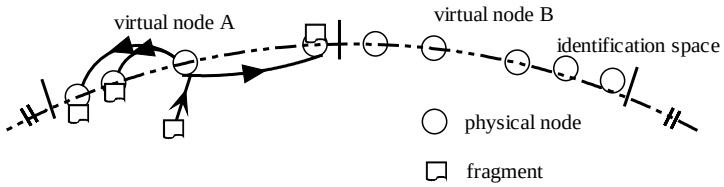


Figure 1. Each virtual node consists of multiple physical peers and each peer in every virtual node contributes storage space to cooperatively store data fragments. When a fragment is to store, synchronization copy is required in the virtual node.

Therefore, the prototype is organized into two layers for data storage: virtual node layer and physical node layer. We utilize the hybrid of two data redundancy schemes, namely replication and erasure coding, to distribute redundancy objects. On the whole, with erasure coding scheme each data item is divided into m fragments and recoded into n ($n>m$)

fragments which are stored separately at virtual node layer; we distribute m fragments into different virtual nodes according to GRP scheme. Within a virtual node, a fragment is replicated into its physical nodes. The replicas of each fragment are distributed by LRP scheme.

5. Protocol

In the protocol, the system should support the following types of operations: node joins, data requests, data maintenance etc.

5.1 Node joins

When a node joins an active virtual node, it announces itself to all other nodes responsible for the same interval. System should recognize two cases: fresh join and return. Data synchronization should be taken among the nodes in the interval in both cases. All data associated in the virtual node is replicated to the node if it is a fresh one; and if it is back after a temporary failure period, it should exchange data that each other does not have, that is, synchronize offline replicas after going online again. When any node leaves the system for the process is inactive, the standard stabilization routine will be performed. The predecessors and successor will be informed, and afterwards, inconsistent routing table entries are identified and updated using the periodic maintenance routine [1].

5.2 Data requests

Considering a new item is published into system, it is split into *n* fragments which are stored separately at different virtual nodes by *(m, n)* erasure coding. Notably, it requires that each fragment is independent to recover the original item from at least any m fragments in the distributed storage system. Fortunately, network coding [11-13] allow the creation of each

encoded fragment independently and are therefore useful for distributed storage systems, which need to create new fragments continuously as nodes join and leave the system. The codes guarantee the correctness of data storage in structured P2P networks. Every fragment is mapped into a virtual node by consistent hashing. It will be replicated and distributed by the owner node to all other nodes responsible in the same virtual node. The retrieve request for an item takes the reverse operations.

Data routing and location requests can be processed almost the same as the specification of DHT among the physical nodes in the identification space except little changes. Figure 2 presents that when a fragment is mapped to an identifier *k* belonging to virtual node *A* but the successor of the identifier is physical node *n* belonging to virtual node *B*. Thus, the routing and location for the fragment to store and access may be redirected to the predecessor node *m* by node *n*. Even thought in the procedure the redirected hop may be an incidental expense, it maybe save one more hops when routing and location for a fragment reach the corresponding virtual node rather than the owner physical node.

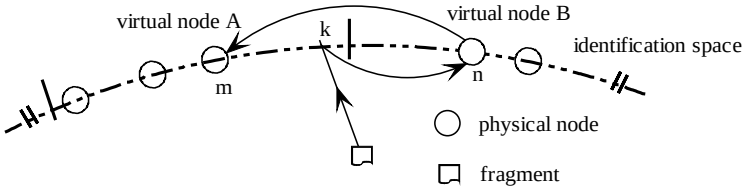


Figure 2. A fragment is mapped to an identifier *k* belonging to virtual node *A* but the successor of the identifier is physical node *n* belonging to virtual node *B*. In the case, the request message will be redirected to its predecessor.

The related procedures are executed whenever data operation messages arrive at the related nodes. As a variation of Chord, the routing table can be the same as finger table in the protocol. Indeed the protocol can combine into the other DHTs as well.

The data request message routing algorithm (figure 3) is crucial in the system design for

data operations depend on it. Given a message, the node first checks to see if the identifier falls within the range of the virtual node. If so, the node determines the action by resolving the message. Otherwise (i.e., the virtual node does not cover the identifier), then one of the two cases may happen. The message is redirected to the predecessor node *m* by node n in one case as shown in figure 2 (case 1). If the redirected node is not responsible for the fragment, the virtual node is unavailable and the message will be abandoned (case 2). In the other case, the routing table is used and the message is forwarded to a node that is the closest to the identifier (case 3).

```
//find the destination to execute the operation in msg by n according to the identifier k of the fragment in msg
n.find_destination(msg){
   if (msg.k ) ∈ n.interval()// message falls into the virtual node which n belongs to.
       return (function *)msg.oper;//invoke the operation of message by n
   if(msg.redirect ){//if the message is redirected, virtual node has failed.(case 2)
        error("could not routing the msg");
        return(-1);
   }
   if(msg.k < n.interval().lowbound && msg.k  ∈  (n.id, n.successor) ) {//case 1
       msg.redirect = TRUE;
      p = n.predecessor();
       forward(msg, p);//send msg to the predecessor of n.
       }
   else {//case 3
       n'= closest_preceding_node(msg.k);//search the closest node in routing table as in [1]
       forward(msg, n');
       }
}
```

Figure 3. The data request message routing

In a message, one of the operations is packaged which will be executed by the routing destination node according to the packaged identifier *k* in the message. These messages are produced by the item operations (PUT_ITEM and GET_ITEM), and data repair operation. If a peer decides to put an item that consists of *n* independent fragments, then the peer issues the PUT_FRAGMENT message for each fragment which identifier is *k*. Once at least *m* messages are acknowledged, the item is putted into the system. Otherwise, the peer tries threshold times. Similarly, if a peer tries to get an item, then it issues the GET_FRAGMENT message for each fragment. Once m responses are received, then item can be recovered. Otherwise, the peer tries threshold times. If the destination node is routed, then the operation will be performed and response will be acknowledged to the original node. For PUT_FRAGMENT, store the fragment in the virtual node; for GET_FRAGMENT, get the fragment if it exists in the virtual node.

5.3 Maintenance

In the long run, many peers are permanently leave the system and the data resided on them is lost. It follows that the redundancy of items would decrease and furthermore the availability of relevant items would be hurt. Therefore data maintenance mechanism is the necessary part and is integrated into the protocol for toleration the amount of peer departures for a long-term availability. The occasion to trigger data recovery lost data is by given a maintenance period and a redundancy threshold. The maintenance mechanism ensure data availability level on the occasion that the repairer discovers the redundancy degree is lower than the given threshold l ($m \leq l < n$).

In addition to the periodical maintenance, the repairer monitors data item requests issued

from each node and trigger a repair if the redundancy degree drops below the threshold. If the requested item could be reconstructed from any *m* fragment resided in different nodes but the response is lower than *l*, the repairer will repair the lost fragments to related virtual nodes while supplying the item for requestor. Otherwise, the repairer records the unavailable data item and periodically checks the degree of fragments. During the upper bound time, the check could repair the redundancy to *n*, otherwise if beyond the bound time, the item would be taken as failed one. The advantage of the maintenance is without having to perform frequent fragment repairs, i.e., it extends the maintenance period and thereby reduce communication overhead. The maintenance ensures the higher availability of the frequent accessed items. In the way, the repair policy belongs to the *lazy* repair according to [5].

6. Analysis

In the section, we present the analysis of the behaviors of single one virtual node and cluster of virtual nodes. According to the stochastic models of the behaviors, the data availability is presented based on one empirical trace dataset.

6.1 Behavior characteristics of single virtual node

The on-line/off-line alterations of an individual node can be modeled as an alternating renewal process, having an *on* process and an *off* process. We assume that physical nodes behave independently of each other and that each process $\{Z_i(t)\}$ is independent on each other.

The collective effect of multiple nodes in one virtual node can be also modeled as the superposition of multiple alternating renewal processes. Let l_i and d_i denote separately the mean lengths of an *on* and *off* period of each node. The asymptotic availability of each node *i* in the long time *t* is given by $a_i = \lim_{t\to\infty} P(Z_i(t) = 1) = l_i/(l_i + d_i)$. If we treat all *on* and *off*

processes as *i.i.d* sets of variables then the probability of a virtual node is *on*, that is, the availability of a virtual node is given by $v = 1 - \prod_i \frac{d_i}{l_i + d_i}$ in the long run. Considering the heterogeneous lifetimes of various nodes, the data availability could hardly be estimated.

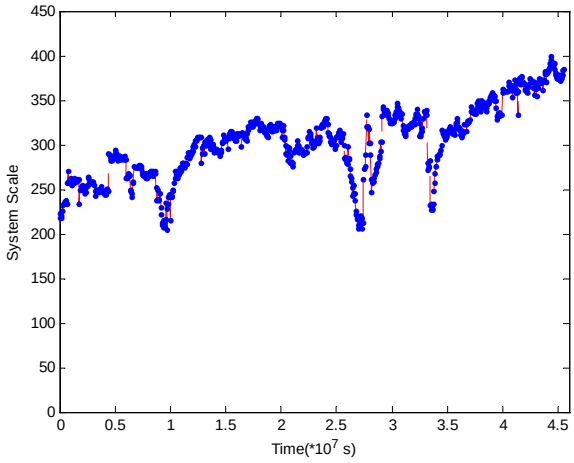


Figure 3(a). The evolution of system size *N(t)*.

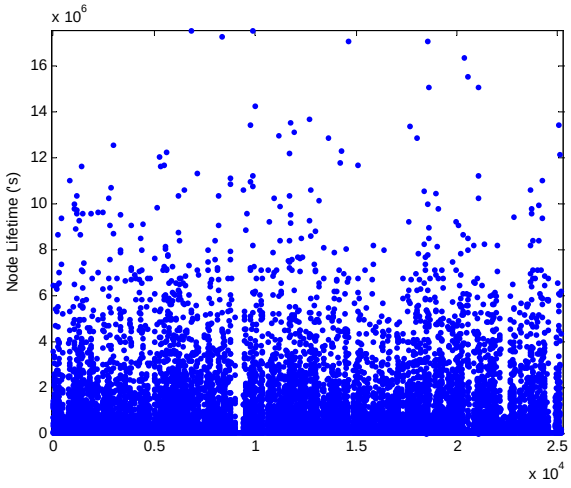
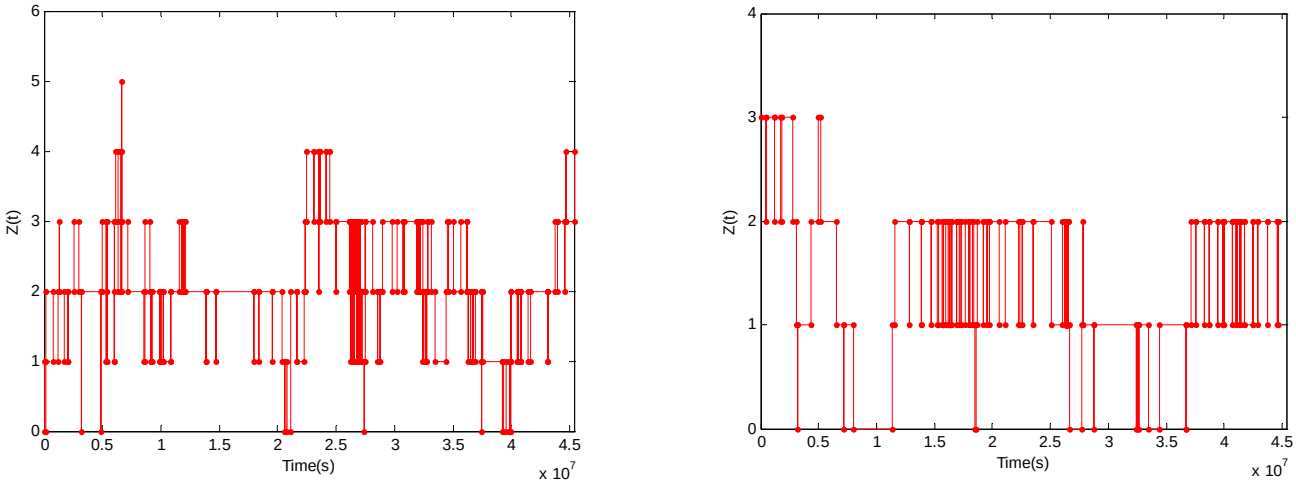


Figure 3(b). The sessions distribution in the trace.

According to the PlanetLab trace used in [20], this data set consists of pings sent every 15 minutes between all pairs of 200~400 PlanetLab nodes from January, 2004, to June, 2005 (i.e., $4.557*10^7$ s). A node is considered to be on in one 15-minute interval when at least half of the pings sent to it in that interval succeeded. Nearly all PlanetLab nodes were off in a number of periods due to planned system upgrades or measurement errors likely. In [8] the trace were removed each period of downtime when less than half the average number of nodes up.

As shown in figure 3(a) according to the cleaned trace, the evolution of system size *N(t)* is presented during the trace period. We consider the system in the stable state during the period. In the period, the system scale, i.e. the number of nodes amounts to 669 in the system. Figure 3(b) shows the session distribution in the trace (25130 sessions). The session distribution suggested in many studies is general, such as Pareto[14-15], Weilbull or lognormal distribution[9], which reveals the heterogeneous behaviors of nodes.

We are primarily interested in the metric *active replica degree (ard(t))* , which is defined as the number of active nodes in one virtual node (assume space size is *s*) at any time epoch *t*. The metric is a variation as nodes are turned *on* and *off*.



(a) s = 8 (b) s=6

Figure 4. The sample paths of a virtual node in the trace period.

Let *{Z(t)}* denote the evolving process of *ard(t)*, i.e, the number of active peers in system at time *t*, and the process is a regenerative process with the regeneration point defined at an arrival epoch. The degree should be at least 1, then the superposed process *{Z(t)}* is considered *on* if at least one of the node processes is *on*, and it is *off* if and only if all of the node processes are *off*. The sample paths are shown for the virtual node size *s=8 bits* in figure 4(a) and *s=6 bits* in figure 4(b). In the paths, a regeneration cycle contains an *up* period with at least one peer present and a *down* period with no one present. It can be observed that the *max{ard(t)}* is 5 in figure 4(a) and 3 in figure 4(b). As an important observation, the coverage behavior of each virtual node can mask failures effectively including temporary and permanent failures for the down periods are relatively little time, about 10^2~10^3s.

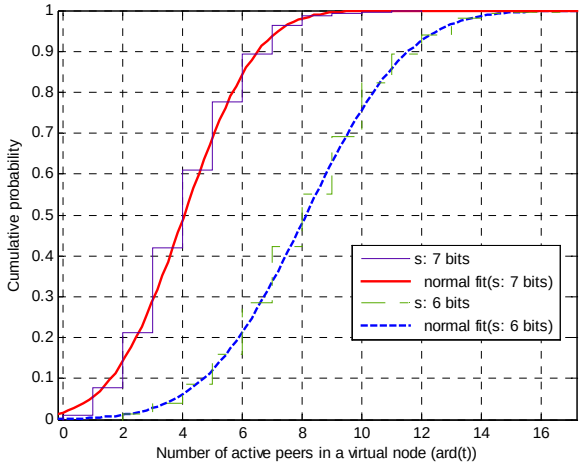


Figure 5. The number of active peers in a virtual node tends to a normal random variable observed in the stable. The relationship between the size of virtual node (*s*) and *ard(t)*.

Let T_k be the amount of time in a cycle with k peers present. Then we have $C = \sum_{i=0} T_i$. Let p_k denote the limiting probability that there are *k* peers in one virtual node at any time. In the count process {Z(t)} produced by the M/G/∞ queue model, which can be considered that peers arrive at an infinite server queue according to a Poisson process and the on-time of each peer can be general. From queuing theory, if the queue is stable, then *E[C]* is finite and $p_k = \lim_{t\to\infty} P\{Z(t) = k\} = E[T_k]/E[C]$. Thereby, the availability of fragment in one virtual node is given by $v = 1 - p_0$. The probability of p_k can be reduced to the number distribution of active peers in one virtual node. The number of active peers in a virtual node tends to a normal random variable observed as shown in figure 5 for different sizes of virtual node, s=7 and 6 respectively. The distribution parameters (mu, sigma) are (4.06, 1.91) for *s=7* and (8.13, 2.67) for *s=6*.

It was observed that while a little amount of short-lived peers join and leave the system at such a high rate that they constitute a relatively large portion of sessions at any point of time. Those short-lived and churn-frequent nodes will have much negative impact on data

availability. However, the impact can be reduced by the convergence behavior of nodes in a virtual node. Thereby the replication model could not be constrained to much heterogeneous nodes. That is, virtual node weakens even masks the behaviors of those churn-frequent nodes, i.e., those temporary failed nodes. As for the impact from those permanent failed nodes and fresh nodes, it could also be reduced to some extents. The on-time and off-time of one fragment replicated into one virtual node are shown as figure 6. In experiments, the size of virtual node is defined as the sum of physical nodes in the period, 3~7 nodes respectively. From the data as figure 6(a) and table 1, the convergence of heterogeneous nodes is much more effective as the size of virtual node increases. The quantity of off-times is similar as observed in figure 6(b).

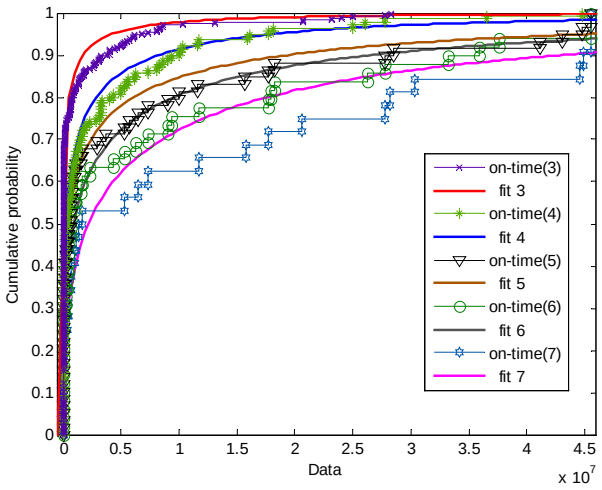


Figure 6(a). The on-time distribution of virtual node.

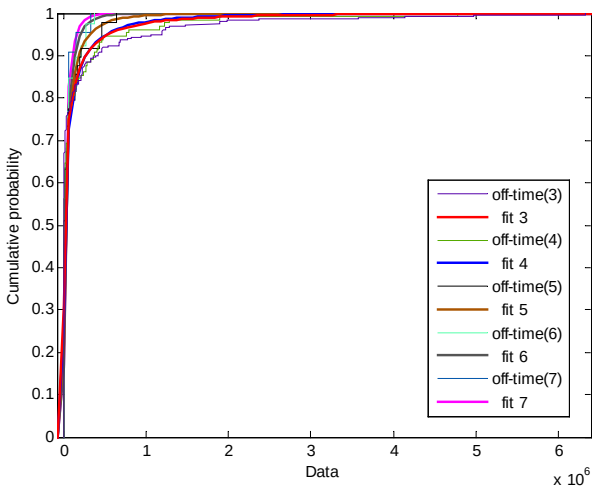


Figure 6(b). The off-time distribution of virtual node.

Table 1. The convergence behaviors for different size virtual nodes

Size	#on-time	Mean(e+006s)	Max(sec)	Weibull Fit(a, b)
3	309	1.2706	28288690	(92364.8, 0.286756)
4	140	3.1158	44748925	(693478,0.337441)
5	59	6.8204	45573054	(1.29659e+006,0.309017)
6	49	8.5168	45573054	(2.45977e+006,0.350952)
7	32	12.660	45573054	(5.37585e+006,0.401647)

As intended, rather than tangle with various behaviors of single peer, the availability of

each fragment is analyzed by the converge behavior of virtual node to mask the churn-frequent nodes. Further enhancement for data availability can be guaranteed by erasure coding among virtual nodes, as explained in next subsection to mask the little *off* time among virtual nodes as shown in figure 4.

6.2 Behavior characteristics of virtual node cluster

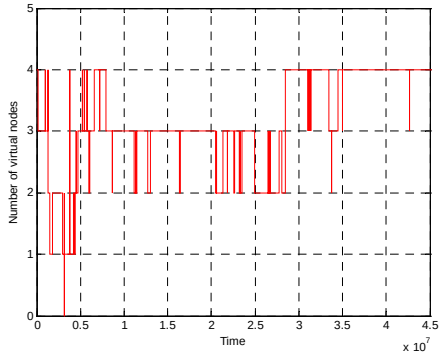
With the (m, n) erasure-coded redundancy scheme, an item will be encoded into *n* fragments and distributed in *n* different virtual nodes. There are e *n+1* states for each item, *0…n*. State *k* is the state that *k* virtual nodes are alive and *(n-k)* virtual nodes are unavailable. Let $A = \{m, ..., n\}$ and $\overline{A} = \{0, ..., m-1\}$ respectively denote the available states and the unavailable ones. The process is said to be *up* when in an available state and *down* when in an unavailable state. The change of a virtual node is independent and identical process with each other and data availability process changes states in accordance with a Markov chain having transition probabilities P_{ij}, $i, j \in \{0,1,...,n\}$. According to limiting probabilities for Markov chain [21], for $i \in A$ and $j \in \overline{A}$ the rate at which the process enters state j from state i is $\pi_i P_{ij}$. And so the rate at which the process enters state j from an available state i in A is $\sum_{i \in A} \pi_i P_{ij}$. Hence, the rate at which it enters any unavailable state from any available one (which is the rate at which breakdowns occur) is $\sum_{j \in \overline{A}} \sum_{i \in A} \pi_i P_{ij}$.

Now let u and d denote the average up-time and down-time separately in the process. Thereby, the transition frequency between any available state and unavailable one is $1/(u+d) = \sum_{j \in \overline{A}} \sum_{i \in A} \pi_i P_{ij}$. The average percentage of up-time in one cycle is $\sum_{i \in A} \pi_i$. Since the process is up on the average u out of every $(u+d)$ time units, the proportion of up-time is given by $u/(u+d) = \sum_{i \in A} \pi_i$. Therefore, the average up-time is $u = \sum_{i \in A} \pi_i / \sum_{j \in \overline{A}} \sum_{i \in A} \pi_i P_{ij}$

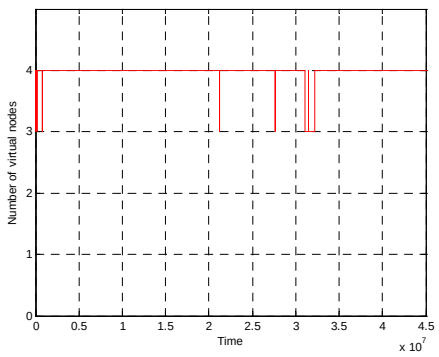
and down-time is $d = \sum_{i \in \bar{A}} \pi_i / \sum_{j \in \bar{A}} \sum_{i \in A} \pi_i P_{ij}$.

The up-time of an item is dependent on the parameters of the redundancy layers when it is put into the system which include the size of virtual node *s* and erasure coding parameters *n* and *m*. The tradeoff of parameters between the two redundancy layers is flexible and enviable. To achieve the same extents of available, the choices of the parameters are crucial. In one extreme, the size of virtual node *s* is to the greatest extent and erasure code parameters can be relaxed. In the other extreme, *n* is a larger number and m is close to 1, thereby the virtual node size s is relaxed. However, the former will consume much to maintenance synchronization and the latter will be degraded to replication that consumes much storage space.

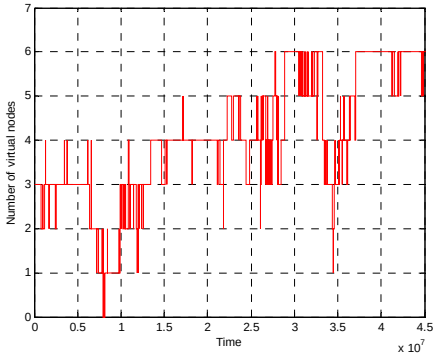
By exploring a large number of different settings with the parameters, we present the average availability of an item as shown in figure 7. In the experiments, the parameters are set as the following: s=3, 4 and m=4, 6. The choice of parameter n can be determined constrained by the requirements of the availability and related costs. According to the result of figure (a)~(d), the tradeoff values of *n* are setting. The system designers must choose the moderate values for (*s*, *m*, *n*) to make the data more available and reliable. Therefore, the result presents that the hybrid approach is effective.



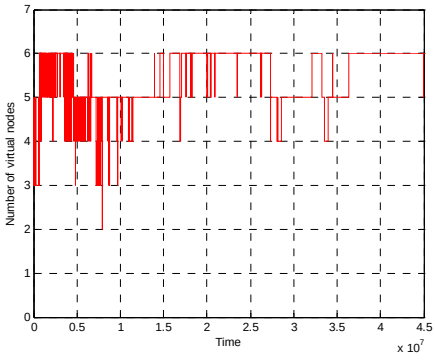
(a) s=3, m=4



(b) s=4, m =4



(c) s=3, m=6



(d) s=4, m =4

Figure 7. The availability results of cluster

7. Conclusion

The paper utilizes the hybrid of two data redundancy schemes to improve system availability to deal with churn. To mask or hide the high churn from the small portion of short-lived but frequent churn peers and permanent failure peers, we use replication among the nodes in a certain interval that can be considered as a virtual node. Then a set of virtual nodes that cooperatively provide guaranteed over the networks with erasure coding. The paper presents the prototype and protocol, and analyzes the behaviors of single one virtual node and cluster with an empirical trace.

References:

[1] I. Stoica et al. Chord: A scalable peer-to-peer lookup service for internet applications. In Proc. of ACM SIGCOMM, 2001.

[2] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proc. of IFIP/ACM International Conference on Distributed Systems Platforms, 2001.

[3] F. Dabek et al. Wide-area cooperative storage with CFS. In Proc. of SOSP, 2001.

[4] J. Kubiatowicz et al. Oceanstore: An architecture for global-scale persistent storage. In Proc. of ASPLOS, 2000.

[5] Ranjita Bhagwan et al. Total recall: System support for automated availability management. In

NSDI, 2004.

[6] R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In Proc. of IPTPS, 2003.

[7] Rodrigo Rodrigues, Barbara Liskov. High availability in DHTs: Erasure coding vs. replication. In Proc. of IPTPS, 2005.

[8] P. Brighten Godfrey, Scott Shenker, and Ion Stoica. Minimizing Churn in Distributed Systems. In Proc. of SIGCOMM, 2006.

[9] Daniel Stutzbach, Reza Rejaie. Understanding Churn in Peer-to-Peer Networks. In Proc. of IMC, 2006.

[10] K. Tati and G. M. Voelker. On object maintenance in peer-to-peer systems. In Proc. IPTPS, 2006.

[11] A. Shokrollahi. Raptor codes. IEEE Trans. on Information Theory, June 2006.

[12] M. Luby. Lt codes. Proc. IEEE Foundations of Computer Science (FOCS), 2002.

[13] A.G. Dimakis, P. Godfrey et al. Network Coding for Distributed Storage Systems. In Proc of INFOCOM, 2007.

[14] S. Saroiu et al. A measurement study of peer-to-peer file sharing systems. In Proc. of MMCN, 2002.

[15] F. E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In Proc. of International Workshop on Web Content Caching and Distribution, 2003.

[16] Ananth Rao et al. Load Balancing in Structured P2P Systems. In Proc. of IPTPS, 2003.

[17] David R. Karger and Matthias Ruhl.Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems. In Proc. of IPTPS, 2004.

[18] E. Brunskill. Building peer-to-peer systems with Chord, a distributed lookup service. In Proc. of the Eighth Workshop on Hot Topics in Operating Systems, 2001.

[19] A. Ghodsi et al. Symmetric replication for structured peer-to-peer systems. In Proc. of the 3rd International Workshop on Databases, Information Systems and Peer-to-Peer Computing, 2004.

[20] Jeremy Stribling. Planetlab all pairs ping. http://infospect.planet-lab.org/pings.

[21] Sheldon M. Ross. Introduction to probability models, 8th Edition. Elsevier, 2003.

[22] Kirsten Hildrum et al. Asymptotically Efficient Approaches to Fault-Tolerance in Peer-to-Peer Networks. In Proc. of the 17th International Symposium on Distributed Computing, 2004.

[23] Jared Saia et al. Dynamically Fault-Tolerant Content Addressable Networks. In Proc. of HOT-P2P, 2004.