IEICE TRANS. COMMUN., VOL.E91–B, NO.12 DECEMBER 2008

LETTER

Lifetime-Aware Replication for Data Durability in P2P Storage Network

Kyungbaek KIM[†a)], ***Member***

SUMMARY Many p2p based wide-area storage networks have been proposed to provide scalable storage services by combining the idle resources of many unreliable nodes. These storage networks can also provide highly available and reliable storage services, by replicating each data on several nodes. The popular approach is availability based replication which uses individual node availability. However, some replicas leave within a short time under high churn in p2p networks. This results in heavy and bursty data traffic, and sometimes some data are lost. This paper presents the lifetime-aware replication which uses the lifetime of each node to prevent the bursty failures and the data loss. It keeps a primary replica which has enough time to replace a lost redundancy. It also spreads replicas on the timeline to reduce the overlapped replicas as best as it can. Results from event-driven simulations show that the lifetime-aware replication keeps high data durability with less data traffic.

key words: *peer-to-peer, replication, lifetime-aware, durability*

1. Introduction

In these days, P2P has become the most popular method in storage intensive applications. One of important parameters is *data durability*, which means that the data which a user has put into a P2P network are not lost due to node failures, and most P2P storage networks replicate data in order to keep high durability. Each participant monitors its data durability and replaces lost redundancy on others in reaction to failures [2], [4], [5]. However, P2P networks suffer from high node churn and replication methods in these networks should consider the efficiency which means minimizing data traffic overhead.

The popular approach in recent studies [1], [6] is *availability based replication* which calculates data durability by exploiting the node availability of each replica like the following model: $A_d = 1 - (1 - A_1)(1 - A_2) \ldots (1 - A_n)$ where n is the number of replicas and $A_1, A_2 \ldots A_n$ are the node availability of each replica. Each node manages the information of its replicas such as the number of replicas and the node availability of each replica by communicating periodically with its *neighbor nodes*. The neighbor nodes are systemically closer nodes such as nodes on a successor list of CHORD [4], leaf nodes of PASTRY [3] and immediate neighbors of Gnutella, communicating with each other periodically to ensure the correctness of p2p networks. Most P2P storage networks pick replicas among these neighbor nodes [1], [4]–[6]. When a node detects any changes of its neighbor nodes which affect its replicas, it checks whether its data durability is below a target value or not. If it is, this node selects a new replica which is the most available node among its neighbor nodes until the new data durability becomes above the target value. More available nodes are mainly selected as replicas and the number of replication operations decreases. According to these, this approach reduces data traffic.

Manuscript received May 12, 2008.
Manuscript revised August 9, 2008.

[†]The author is with the Department of Computer Science, University of California, Irvine, USA.
a) E-mail: kyungbak@uci.edu

DOI: 10.1093/ietcom/e91–b.12.4020

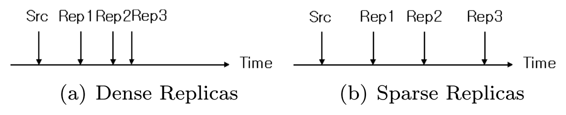


Fig. 1 Timing effect of replicas. (Src: a responsible node for data. R1, R2 and R3: replicas for src. Each arrow: leave time of each node.) Sparse replicas are more efficient than dense replicas.

However, in P2P networks, most nodes have the high probability of a failure and sometimes some replicas for data leave or fail within a short time like Fig. 1(a). The join/leave time of a node is basically independent of both of its availability and its reliability. Though more available nodes are selected as replicas, this situation may occur occasionally. The bursty failures of these *dense replicas* need many replication operations for a short time exhausting much traffic bandwidth instantly. That is, some meaningless overlapped replicas can not help keeping high data durability and just waste bandwidth. If we can spread out replicas more like Fig. 1(b), they are distributed with some time intervals and each replica becomes more meaningful. These *sparse replicas* can reduce the number of replication operations and the needed data traffic decreases.

2. Lifetime-Aware Replication

We suggest the lifetime-aware replication which uses node availability as well as the information of replicas' lifetime to achieve high data durability with less data copy traffic. The lifetime-aware replication is a kind of availability based replication and uses the described data durability model for checking whether it needs more replicas or not. Whenever the calculated data durability is below a target value, it selects new replicas until the new data durability becomes over the target value. The major concern of the lifetime-aware replication is which nodes are suited for the sparse replicas. The lifetime-aware replication selects new replicas among the neighbor nodes by exploiting both of the node availability and the lifetime information of each node. At first, it tries to get a *primary replica* which is mainly responsible for carrying out a replication operation reliably. To do this, the node which has enough lifetime to create a new replica is selected among the neighbor nodes including the existing replicas as the primary replica. After that, it selects more available nodes which are not overlapped with other replicas. That is, if possible, it separates replicas on a timeline. This behavior can diminish the number of meaningless replicas and reduces the number of the bursty leaves of replicas. If we know the lifetimes of all nodes exactly, we could get the best performance of the lifetime-aware replication, but this is almost impossible. Instead of the exact lifetime, each node measures its *MTTF* (Mean Time To Failure) and broadcasts it to neighbor nodes.

The lifetime-aware replication uses the node availability as well as the lifetime information. The node availability is inherited by the natural characteristics of a node, but the lifetime information should be measured by each node and be shared with each other. The lifetime information is composed of current $JoinTime$ and *MTTF*. Remembering the previous lifetime, each node computes *MTTF* right after joining the P2P network. Both of the calculated *MTTF* and the current $JoinTime$ are informed to its neighbor nodes by piggybacking them to the periodic keep-alive message. Eventually each node stores both of the node availability and the lifetime information of each neighbor node. Based on this lifetime information, when a node L initiates a replication operation, it calculates its EET (Estimated End Time) and $EETs$ of its neighbor nodes with the following equation: $EET_L = JoinTime_L + MTTF_L$. Using this EET, node L calculates RT (Residual Time) of each neighbor node which is the time difference between the initiated node and each neighbor node like this equation: $RT_{LR} = EET_R - EET_L$. This RT is used to select separate replicas.

At first, the lifetime-aware replication manages the primary replica which is mainly responsible for performing a replication operation reliably. A transaction of a replication operation is composed of fault detection, checking data durability, requesting data and copying data and we call the required time for a transaction FRT (Fault Recovery Time). If there is at least one replica whose RT is greater than FRT during performing a transaction, this transaction is successfully accomplished. To do this, RT of a primary replica should be longer than $\alpha * FRT, \alpha > 1$. When α is big, the reliability of a primary replica increases but it is hard to find the primary replica satisfying the condition. The detailed result about α is presented on Sect. 3. Because of this timing condition ensuring that a primary replica has enough time for performing a replication operation, we can consider that a primary replica is highly available for its responsible data even if it has low node availability. According to this, during calculating data durability, the node availability of a primary replica has additional node availability value up to 0.9.

Checking the existence of a primary replica among the pre-existing replicas is performed during calculating data durability just before starting replication operations. If there is no primary replica, the lifetime-aware replication chooses a new candidate node among the unselected neighbor nodes according to the timing condition for a primary replica like line 2–3 in Fig. 2. In the case where there is already a primary replica or there is no candidate node which satisfied the timing condition for a primary replica, the lifetime-aware replication selects the node whose availability is biggest among the unselected neighbor nodes as a candidate node like line 4–5.

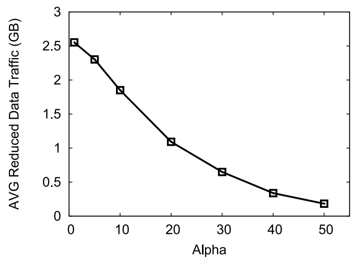
```
 1: While( data durability < target durability )
 2:   If( ! primary replica )
 3:     find a new candidate node whose RT is longer than α*FRT
 4:   if( ! new candidate node )
 5:     find a new candidate node whose node availability is biggest
 6:   If( new candidate node )
 7:     If( ! Check_timespan_overlap ( new candidate node) )
 8:       add_pending_list( new candidate node )
 9:     Else    select this candidate node as a new replica
10:   Else    select a replica which is most available among the pending nodes.

11: Check_timespan_overlap( node )
12:   if( RT of node < 0 ) return 0;
13:   for each replicas Ri
14:     if( | RT of node – RT of Ri | < β*FRT ) return 0;
15:   return 1;
```

Fig. 2 Basic operation of lifetime-aware replication.

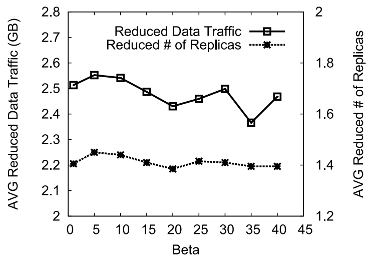
After selecting a candidate node, the function *check_timespan_overlap* inspects whether it is proper to a new replica or not. At first, like line 12 if RT of the candidate node is below 0, we can estimate that this node will leave soon regardless its node availability and this node is not proper to a new replica. Then, at line 13–14, RT of the candidate node is compared with RT of each replica. If any one of RT differences is below $\beta * FRT, (\beta > 0)$, this node is overlapped other replicas and this function returns 0. β is the adjustment parameter for separate replicas. When β is big, each replica is apart more but it is hard to find separate replicas. According to this function, selecting new replicas which are apart from other replicas, the lifetime-aware replication forms the sparse replicas. Being not proper to a new replica, this candidate node is added to the pending list which is used when there is no more candidate node. When the lifetime-aware replication uses the pending list, it selects the most available node among the list like line 10.

3. Evaluation

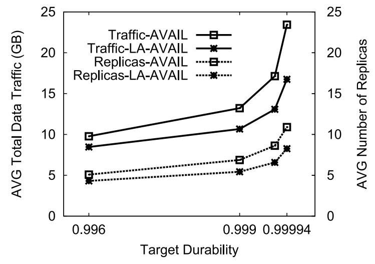
We carried out event-driven simulations to evaluate the lifetime-aware replication and the results show that it keeps high data durability with less data traffic. We made a P2P simulator which emulates behavior of nodes on the application layer with a DHT based P2P algorithm, Pastry, whose number of leaf nodes is 16 [3]. We use 20000 nodes whose characteristics are preset by the Poisson distribution and the on/off duration of each node is assigned by using the exponential distribution whose mean is inherited from the Poisson distribution. Each node manages averagely 1024 objects whose average size is 20 KB and target data durability is variable from 0.996 to 0.99994 which is same to the number



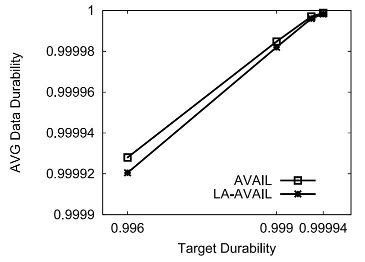
(a) Preliminary : variable α, $\beta = 5$



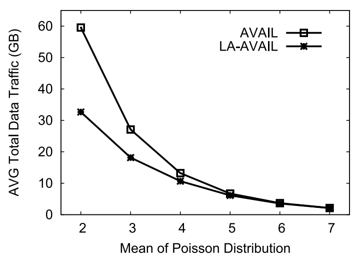
(b) Preliminary : variable β, $\alpha = 1$



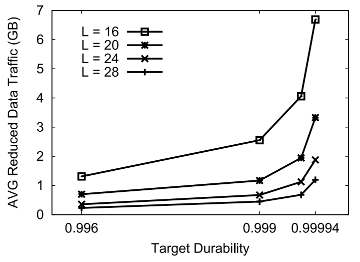
(c) Comparison of various target durability



(d) Durability



(e) Comparison of various node behaviors, target durability = 0.999



(f) Comparison of various range of neighbor nodes

Fig. 3 Evaluation results.

of replicas from 8 to 14 with 0.5 node availability. The total simulation time is 15000 ticks and the average *MTTF* of a node is about 300 ticks. If the average *MTTF* of a node is 1 hour, 1 tick means 12 seconds and about 60 nodes join/leave during 1 tick. *FRT* is set to 15 ticks which is same to the transaction of a replication operation for 20 MB data over 150 KB links. We compare the lifetime-aware replication with availability based replication.

Figure 3(a) and Fig. 3(b) shows the preliminary results in order to select values for α and β. The average reduced data traffic means how much traffic the lifetime-aware replication reduces against availability based replication. When this value gets bigger, data traffic is reduced more. In Fig. 3(a), as α increases, the performance of the lifetime-aware replication decreases and saturates to 0. Even if a primary replica is more reliable with big α, it is hard to get a suitable primary replica which has additional availability bonus. In Fig. 3(b), the reduced traffic fluctuates with β between 2.35 GB and 2.55 GB and has the peak when β is 5. For more details, we also measure the average reduced number of replicas. It looks like a damped oscillating function whose amplitude decays away to 1.395. The peak value of this function is 1.45 when β is 5. If β is too small, the meaningless replicas can still exist, otherwise if β is too big, the lifetime-aware replication can not select the separated replicas well. According to these results, we use 1 as α and 5 as β.

Figure 3(c) represents the comparison of average data traffic per a node with various target durability. “AVAIL” means availability based replication and “LA-AVAIL” means the lifetime-aware replication. The lifetime-aware replication reduces more total data traffic in between 13.4% and 28.6% than availability based replication. To find out the detailed effect of our new replication, we measure the number of replicas for each target durability too. This is similar to the function of data traffic and the lifetime-aware replication reduces the number of replicas in between 14% and 23.8%. The number of replicas decreases by using a primary replica and sparse replicas, and the number of replication operations also reduces. That is, total data copy traffic decreases in proportion to the reduced number of replicas. We note that even though the number of replicas is only reduced by 23.8% for 0.99994, the data traffic decreases in 28.6%. That is, the lifetime-aware replication can be used more efficiently with higher target durability. Moreover, because the replicas of the updated data should be updated for data integrity, the reduction of replicas also mitigates data update traffic.

To find out how the lifetime-aware replication affects data durability, we measured data durability during the simulation. When a node leaves right now and there is no leftover replica whose *RT* is longer than *FRT*, we treat it as a data loss. Figure 3(d) shows that the lifetime-aware replication keeps up high data durability as same as availability based replication. The lifetime-aware replication manages the primary replica which has enough time to replace the lost redundancy and spreads replicas on a timeline to reduce the overlapped replicas. Though the lifetime-aware replication has less replicas, these properties maintain the probability of having a replica which has enough time for a replication operation. According to this, the lifetime-aware replication achieves high data durability with less data copy traffic.

Figure 3(e) represents the comparison of data traffic with various node behavior by changing mean of Poisson distribution. When the mean decreases, more nodes are short-lived. Oppositely, when the mean increases, more nodes are long-lived. As we expected, when the mean increases, data traffic decreases significantly. We note that as the mean becomes smaller the lifetime-aware replication reduces more data traffic. Especially, when the mean is 2, it saves about 45.2% data traffic. That is, when more nodes are short lived churning more severely in the P2P network, the probability of forming the dense replicas increases more and the lifetime-base replication takes full advantage of its behavior such as preventing from overlapping each replica. According to this, the lifetime-based replication is good for very dynamic P2P storage networks.

Figure 3(f) shows that changing the range of neighbor nodes affects the performance of the lifetime-aware replicaion. In this figure, "L" means the range of neighbor nodes. Because neighbor nodes communicate with each other periodically and this can result in traffic flooding, the range of neighbor nodes is typically small (In PASTRY, the number of leaf nodes is 16 [3]). If the range of neighbor nodes increases with a risk of flooding, the probability of finding highly available nodes which are not overlapped increases, and the required data traffic of availability based replication decreases. However, even if an enhancement of availability based replication is achieved, the lifetime-aware replication still reduces more data traffic than availability based replication like Fig. 3(f).

4. Related Works

The commercial P2P file sharing networks leave data replications up to the popularity of data. In this case, the popular data have very high data durability but the unpopular data do not. To make a P2P storage network durable, smart data replication methods are needed. Additionally each inserted data is available for any time and has the similar data durability. [2], [4] and [5] use the quick replication which replaces lost redundancy. However, this replication should be efficient. That is, the required data copy traffic should be minimized. To address this problem, [1] and [6] exploit node availability to calculate the data durability. Moreover, many approaches [7], [8] try to improve the performance of this availability based replication. [7] proposes that node availability changes according to the number of total nodes. [8] suggests that node availability means the steady-state availability and is the summation of the transient-state availabilities. However, these approaches missed the timing effect between replicas which we consider mainly.

5. Conclusion

We propose and show that the lifetime of replicas positively affects both of data copy traffic and data durability. Our lifetime-aware replication exploits the lifetime information of replicas, especially measured *MTTF* and prevents the overlapped replicas on the timeline. These sparse replicas with primary replica keep the same high data durability as availability based replication, but reduce data copy traffic by 28.6%. Moreover, simulation results with various settings show that the lifetime-aware replication is more efficient in highly dynamic p2p storage networks with very high target durability.

References

[1] K. Kim and D. Park, "Reducing replication overhead for data durability in DHT based P2P system," IEICE Trans. Inf. & Syst., vol.E90-D, no.9, pp.1452–1455, Sept. 2007.

[2] K. Kim and D. Park, "Efficient and scalable client clustering for Web proxy cache," IEICE Trans. Inf. & Syst., vol.E86-D, no.9, pp.1577–1585, Sept. 2003.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," Proc. Middleware, pp.329–350, Nov. 2001.

[4] R. Bhagwan, K. Tati, Y.C. Cheng, S. Savage, and G.M. Voelker, "Total recall: System support for automated availability management," Proc. NSDI, pp.227–350, March 2004.

[5] B.G. Chun, F. Dabaek, A. Haeberlen, E. Sit, H. Weatherspoon, M.F. Kaashoek, J. Kubiatowicz, and R. Morris, "Efficient replica maintenance for distributed storage systems," Proc. NSDI, pp.45–58, March 2006.

[6] R. Bhagwan, S. Savage, and G. Voelker, "Replication strategies for highly available peer-to-peer storage systems," Proc. FuDiCo, pp.40–49, June 2002.

[7] R.J. Dunn, J. Zahorjan, S.D. Gribble, and H.M. Levy, "Presence-based availability and P2P systems," Proc. P2P, pp.209–216, Sept. 2005.

[8] J. Tian, Z. Yang, and Y. Dai, "A data placement scheme with time-related model for P2P storages," Proc. P2P, pp.151–158, Sept. 2007.