LETTER

Relative MTTF-based incentive scheme for availability-based replication in P2P systems

Kyungbaek KIM[†a)], ***Member***

SUMMARY

When P2P systems are used for data sensitive systems, the data availability has become an important issue. The availability-based replication using individual node availability is the most popular method keeping high data availability efficiently. However, since the individual node availability is derived by the individual lifetime information of each node, the availability-based replication may select useless replicas. In this paper, we explore the *relative MTTF(Mean Time To Failure)-based incentive scheme* for the more efficient availability-based replication. The relative $MTTF$ is used to classify the *guaranteed replicas* which can get the *incentive node availability*, and these replicas help reduce the data traffic and the number of replicas without losing the target data availability. Results from trace-driven simulations show that the replication using our relative $MTTF$-based incentive scheme achieves the same target data availability with 41% less data traffic and 24% less replicas.

key words: *P2P, Availability, Replication, MTTF, Incentive*

1. Introduction

In these days, the P2P data sharing concept has been the most promising method for large scale distributed storage systems such as file sharing systems, distributed wiki systems, content distribution systems, and so on. One of the important issues in the P2P data sharing, especially supporting a persistent data storage service, is the data availability, because the service entities are unreliable and selfish. As the scale of the distributed storage systems increases, the data availability issue becomes more important [1].

Many P2P systems achieve the desired data availability by increasing the redundancy of data which is realized by creating data replicas among its neighbor nodes[3][9]. The data replicated on multiple replicas are lost only if all the replicas fail within a short time interval and we call this interval as a *critical time interval*. As the number of replicas increases, the probability that data are lost decreases and the system can achieve high data availability. However, in such an unreliable environment, many improper replicas may incur too much traffic to keep the required redundancy of data and to manage the consistency of the data between its replicas. According to this, we should consider the efficient replication scheme which can sort out more reliable replicas in order to keep the high data availability with less data traffic and fewer replicas.

The most popular and efficient replication scheme is the *availability-based replication* [4][5][6]. The node availability represents the likelihood that a node is available, and it is obtained by observing the past individual behavior of a node. Conventionally, the fraction of time a node is online, $\frac{MTTF}{MTTF+MTTR}$ where $MTTF$ means *Mean Time To Failure* and $MTTR$ means *Mean Time To Recover*, is used for the node availability whose value lies in (0,1). The data availability is calculated by the following equation: $A_D = 1 - \Pi_j(1 - A_j)$, where A_j is the node availability of the replica j. Whenever the data availability drops below the given target data availability, the node which has the highest node availability among the candidate nodes is selected as a new replica, and this process continues until the data availability increases over the given target data availability.

It would seem the availability-based replication works well in general, but there is still room for improvement by considering the relativity of replicas. The actual purpose of the redundancy of data is not keeping the high value of the data availability in any time, but minimizing the probability that all the replicas fail within the critical time interval which may represent the essential time duration of creating a new replica. That is, if it is possible that a replica R of a node i guarantees that it has enough time to create another replica after the node i leaves, we do not need to maintain replicas eagerly in order to sustain the *high value* of the data availability in any time. We call the replica R as a *guaranteed replica*. In this paper, we explore how to find guaranteed replicas and how to enhance the availability-based replication with the guaranteed replicas in terms of data traffic and the number of replicas.

2. Relative MTTF-based incentive node availability

Before describing details of our incentive scheme, we present how to get the *relative* $MTTF$ which is the main parameter to sort out the guaranteed replicas. When a node i joins a P2P system, it sets its join time (T_{J_i}) to the current time (T_{C_i}) and updates its individual node availability(A_i) by using the updated $MTTF_i$ and the updated $MTTR_i$ like the paper [6]. Then it initializes its *absolute* $MTTF$ ($aMTTF_i$), which is the sum of $MTTF_i$ and T_{J_i}. In turn, it exchanges both of its A_i and $aMTTF_i$ with its neighbor nodes by piggybacking them on the periodic keep-alive messages. After exchanging A_i and $aMTTF_i$, a node calculates relative MTTF of its all neighbor nodes. The node j's

Manuscript received June 23, 2010.

[†]The author is with the Department of Computer Science, University of California, Irvine, USA

a) E-mail: kyungbak@uci.edu;kyungbaekkim@gmail.com

Algorithm 1 Calculating data availability of node i using relative $MTTF$-based incentive node availability

Require: $\Delta MTTF_{R_{ij}}, \forall j \in U$ {U : set of replicas}
1: $A_F \leftarrow 1$
2: **for all** $j \in U$ **do**
3: **if** $\Delta MTTF_{R_{ij}} > T_{Thres}$ {T_{Thres} : time threshold} **then**
4: **if** $A_{INC} > A_j$ {A_{INC} : incentive availability} **then**
5: $A_F \leftarrow A_F * (1 - A_{INC})$
6: **else**
7: $A_F \leftarrow A_F * (1 - A_j)$
8: **else**
9: $A_F \leftarrow A_F * (1 - A_j)$
10: $A_D = 1 - A_F$ {A_D : Data availability}

relative $MTTF$ calculated by node i ($\Delta MTTF_{R_{ji}}$) sets to the time difference between $aMTTF_j$ and node i's base time, T_{Bi}. That is, $\Delta MTTF_{R_{ji}} = aMTTF_j - T_{B_i}$. T_{B_i} basically sets to $aMTTF_i$, but if node i's current time, T_{C_i}, has already passed $aMTTF_i$, T_{B_i} sets to T_{C_i}. As $\Delta MTTF_{R_{ji}}$ becomes bigger, the probability that node j will be online after node i leaves increases. Otherwise, if $\Delta MTTF_{R_{ji}}$ has a negative value, most likely node j will leave before node i leaves.

The relative $MTTF$ is mainly used to find more proper replicas. Whenever a node i calculates data availability or selects new replicas, node i firstly sorts out *guaranteed replicas* whose relative $MTTF$ is greater than the given time threshold (T_{Thres}) like line 3 in algorithm 1 and line 2 in algorithm 2. The guaranteed replicas most likely ensure that they have enough time to create a new replica after the node i leaves the P2P system regardless of their individual node availability. The required time for creating a new replica is called *Failure Recovery Time*(FRT). As FRT increases, the required T_{Thres} should increase to find out guaranteed replicas correctly. A guaranteed replica can use the *incentive node availability* (A_{INC}) rather than its individual node availability. A_{INC} represents how much a replication scheme believes the reliability of the selected guaranteed replicas. As A_{INC} increases, the replication scheme can exploit guaranteed replicas more aggressively to reduce data traffic/the number of replicas, but it may cause data loss. We will show the detailed tradeoff in section 3.

Both T_{Thres} and A_{INC} are tunable parameters. Based on the properties of a P2P system, the proper values for these parameters can be determined. That is, we can adjust these parameters by monitoring the P2P system, specifically the data loss. At first, T_{Thres} is set to very big value (80000 sec) and A_{INC} is set to very low value (0.8). In turn, if no data loss is detected for a period of time, we can decrease T_{Thres} and increase A_{INC} to reduce the data traffic and the number of replicas. Otherwise, if significant data losses are detected, T_{Thres} should increase and A_{INC} must decrease to prevent the unintended data loss. According to this feedback process, we can adjust both T_{Thres} and A_{INC} for a P2P system to achieve high data availability efficiently without any unintended data loss.

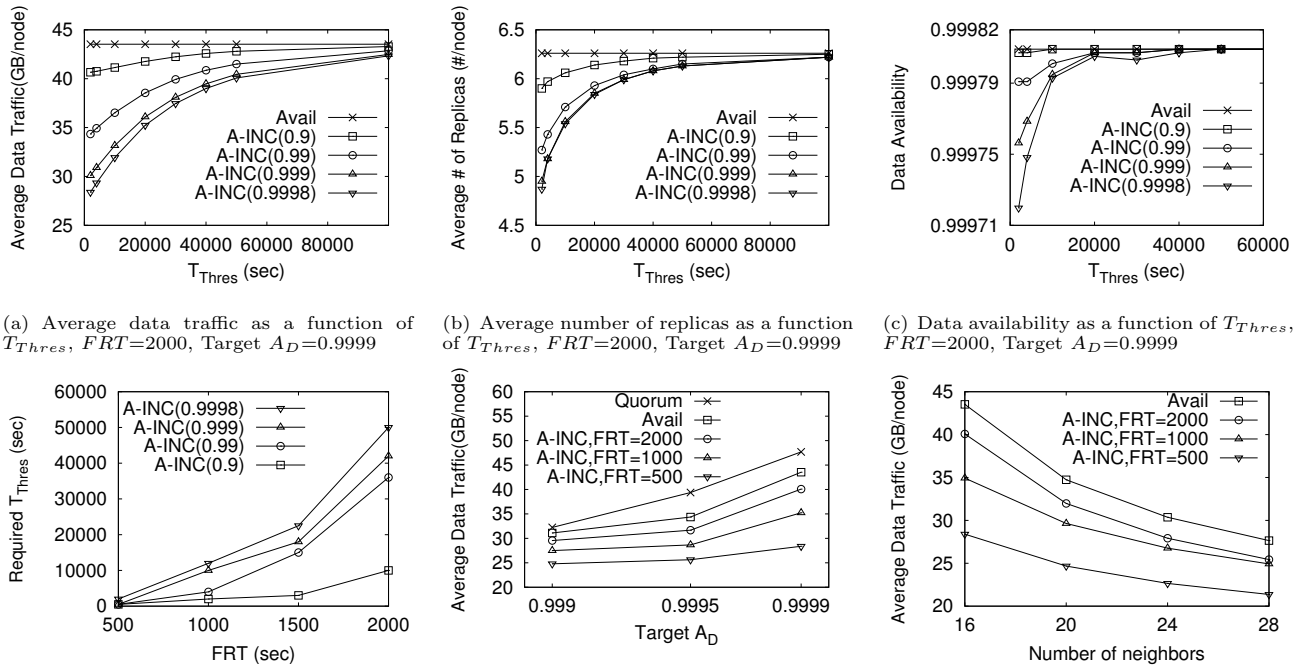
A node i calculates its data availability like algorithm 1 whenever it perceives any change of its replicas

Algorithm 2 Replication operation for node i using relative $MTTF$-based incentive node availability

Require: $\Delta MTTF_{R_{ij}}, \forall j \in N$ {N : set of neighbors exclude replicas}
1: **for all** $j \in N$ **do**
2: **if** $\Delta MTTF_{R_{ij}} > T_{Thres}$ **then**
3: add j to I {I : set of candidates of guaranteed replicas}
4: remove j from N
5: **while** $A_D <$ Target A_D **do**
6: $A_F \leftarrow (1 - A_D)$
7: **if** $I \neq NULL$ **then**
8: pick a new node j among I {condition : $\Delta MTTF_{R_{ij}}$ is longest among I }
9: **if** $A_{INC} > A_j$ **then**
10: $A_F = A_F * (1 - A_{INC})$
11: **else**
12: $A_F = A_F * (1 - A_j)$
13: remove j from I
14: **else**
15: **if** $N \neq NULL$ **then**
16: pick a new node j among N {condition : A_j is biggest among N}
17: **if** $|\Delta MTTF_{R_{ij}}| < T_{Thres}$ **then**
18: add j to P {P : a set of pending nodes}
19: **else**
20: $A_F = A_F * (1 - A_j)$
21: remove j from N
22: **else if** $P \neq NULL$ **then**
23: pick a new node j among P {condition : A_j is biggest among P}
24: $A_F = A_F * (1 - A_j)$
25: remove j from P
26: **else**
27: break
28: $A_D \leftarrow (1 - A_F)$

by exchanging periodic keep-alive messages with neighbors or getting a join/leave notification. If the calculated data availability is lower than the target data availability, the node i starts to select a new replica like algorithm 2. At first, it finds the candidates of guaranteed replicas, then it picks the node j having the longest $\Delta MTTF_{R_{ij}}$ among the candidates (line 8). Even though there can be many possible guaranteed replicas, we should pick the most proper guaranteed replica to maximize the effectiveness of the incentive scheme. After selecting a new replica, the data availability is calculated with the incentive availability value (A_{INC}), unless the individual node availability of the newly selected replica is bigger than A_{INC} (line 9-12).

If the node i needs more replicas but there are no more candidates of guaranteed replicas, it selects the node j whose individual node availability is biggest among the set of normal neighbor nodes like the traditional availability-based replication. However, if the absolute value of $\Delta MTTF_{R_{ij}}$ is smaller than the threshold (T_{Thres}), the node j is not selected as a new replica but added to the pending list (line 17-18). With this policy we can find the replica which has lived for long time before being selected. That is, this policy relies on that the node being online for long time will be online for much longer time, which is revealed with many previous researches. If there is no possible node for a new replica among the neighbor nodes except the nodes on the pending list, a new replica is selected among



(a) Average data traffic as a function of T_{Thres}, FRT=2000, Target A_D=0.9999

(b) Average number of replicas as a function of T_{Thres}, FRT=2000, Target A_D=0.9999

(c) Data availability as a function of T_{Thres}, FRT=2000, Target A_D=0.9999

(d) Required T_{Thres} as a function of FRT, Target A_D=0.9999

(e) Average data traffic as a function of target A_D without unintended data loss

(f) Average data traffic as a function of the number of neighbors, Target A_D=0.9999

Fig. 1 Performance comparison of the availability based replication using relative MTTF-based incentive scheme (A-INC) with the traditional availability-based replication (Avail). In A-INC(N), N means the incentive node availability(A_{INC}) for guaranteed replicas. T_{Thres} is the given time threshold value for sorting out guaranteed replicas. FRT means *Failure Recovery Time*. Target A_D is the target data availability.

the pending list based on the individual node availability (line 23). This replication operation selecting new replicas performs until the newly calculated data availability is above the target data availability (line 5) or until there is no possible neighbor node (line 26-27).

3. Evaluation

In this section, we evaluate the effectiveness of the relative $MTTF$-based incentive node availability for the availability-based replication in P2P systems in terms of data traffic, number of replicas, and data availability, through the trace driven evaluation using the real trace. We implemented PASTRY [3] as a P2P routing protocol. In PASTRY, nodes and data are mapped into a sequence of digits with base 2^b and the choice of b involves the performance of routing. Each node has L number of leaf nodes which are the nodes with the numerically closest nodeIds relative to the present node's nodeId. The leaf nodes are used to the neighbor nodes among which the replication scheme picks up the required replicas. 160 bit nodeId space is used to identify nodes, and b and L is set to 4 and 16, respectively.

To mimic the behavior of P2P users, we use the trace from paper [2] which includes over 90K peer lifespan measured in the Gnutella network between March 1st and 8th, 2003. Each node joins/leaves the P2P system at the specific time given by the trace. According to the trace, the number of online nodes fluctuates between 10K and 14K in the stable state, and our results are measured after 10K nodes join the system.

We assume that the P2P storage system attempts to guarantee the target data availability of the stored data, even though the creator of the data is offline. Each node creates its own data up to 200MBytes. The data is stored at the node whose nodeId is closest to the data key, and also replicated by using the given replication scheme with the target data availability, which varies from 0.999 to 0.9999 in this evaluation.

Firstly, in figure 1(a), 1(b) and 1(c), we present the effectiveness of the relative $MTTF$-based incentive scheme under various setting to classify the guaranteed replicas. While "Avail" represents a constant line, "A-INC" shows a logarithmic line along with T_{Thres}. That is, "Avail" is equal to "A-INC(N)" when $T_{Thres} = \infty$. As T_{Thres} decreases more, the data traffic caused by replication operations and the number of replicas to keep the target A_D decreases exponentially. This is because the less T_{Thres} allows that more nodes can be classified as guaranteed replicas. Moreover, as the incentive node availability (A_{INC}) for a guaranteed replica increases more, "A-INC" can save more traffic and use fewer replicas.

However, when the guaranteed replicas classified by the small T_{Thres} have the high A_{INC}, they are most likely overestimated and some unintended data losses occur like figure 1(c). To measure the data availability, we check the data availability of the specific data whenever the node being responsible for the data leaves the P2P system. Since FRT is the minimally required

time for completing a replication operation, if there is at least one replica who will live longer than FRT from the starting time of the data replication, the data is supposed to be available. According to figure 1(c), we note that "A-INC" requires an adequate T_{Thres} to prevent the unintended data losses. That is, as A_{INC} increases, the required T_{Thres} should increases. Consequently, there is a tradeoff between enhancing performance (less data traffic/less number of replicas) and achieving high data availability, and the required T_{Thres} represents the breakpoint of the tradeoff under a given A_{INC}.

The important parameter to determine the level of the data availability is FRT. The longer FRT means that a node may handle more data and a replication operation takes more time. According to this, the required T_{Thres}, which is the breakpoint between the performance and the data availability, is also affected by FRT. In figure 1(d), we observed that as FRT increases, the required T_{Thres} increases dramatically except the case where A_{INC} is small such as 0.9. That is, in order to give the high A_{INC} to the guaranteed replicas, we need very big T_{Thres}. Fortunately, if FRT is short, we can aggressively exploit the guaranteed replicas with very high A_{INC}.

In figure 1(e), we show the effectiveness of our relative $MTTF$-based incentive scheme for various target A_D. In here, "Quorum" means the quorum-based replication algorithm which always attempts to keep the given number of replicas [6]. The number of replicas for "Quorum" is set to the average number of replicas of "Avail" for each target A_D. "A-INC,FRT=M" means the replication using relative $MTTF$-based incentive scheme which enhances the performance maximally without the unintended data losses under FRT=M. As the target A_D increases and FRT decreases, "A-INC" reduces more data traffic. Especially, when FRT=500 and target A_D=0.9999, we can reduce around 41% data traffic.

We also show the effect of the size of neighbors in figure 1(f). Though "Avail" can reduce data traffic along with the size of neighbors, the probability sorting out guaranteed replicas also increases along with it, and our relative $MTTF$-based incentive scheme still works well.

4. Related works

There are several papers to improve the performance of the availability-based replication [7][8]. But these approaches miss the relativity between replicas which we mainly considered. Some researches [2][14] explore the usage of nodes' lifetime to enhance the performance of P2P system, but their target domains (reducing the connection breakdown, proofing the resilience of P2P system) are different to ours. Some efforts solve the data availability problem by using erasure coding [10][11][12] or exploiting proactive replication [13], but they still do not consider the relativity between replicas. The coding-based replication is out of the scope of this paper, but we mainly focus on how to enhance the performance of the availability-based replication by considering the relativity between nodes.

5. Conclusion

We propose the relative $MTTF$-based incentive scheme for the availability-based replication to achieve very high data availability with less data traffic and fewer replicas. By tuning the time threshold for classifying the guaranteed replicas and their incentive node availability, our incentive scheme is adjustable to various kinds of P2P storage systems whose purposes are different to each other. Moreover, the primitive design of the relative $MTTF$-based incentive scheme can be extended to a run-time scheme which adjusts the tunable parameters (T_{Thres} and A_{INC}) by monitoring the nodes' behavior and the loss of data.

References

[1] R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding Availability," Proc. IPTPS, pp.256-267, Feb. 2003.
[2] F.E. Bustamante and Y. Qiao, "Friendships that last: Peer lifespan and its role in P2P protocols," Proc. IWCD, pp.233-246, 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.337-350, March 2004.
[5] B. G. Chun, F. Dabaek, A. Haeberlen, E. Sit, H. Weatherspoon, M.F. Kaashoek, J. Kubiatowicz, and R. Morris, "Effcient replica maintenance for distributed storage systems," Proc. NSDI, pp.45-58, May 2006.
[6] 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, Sep. 2007.
[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.
[9] 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, Sep. 2003.
[10] W. K. Lin, D. M. Chiu, and Y. B. Lee, "Erasure Code Replication Revisited," Proc. P2P, pp.90-97, Aug. 2004.
[11] R. Rodrigues and B. Liskov, "High Availability in DHTs: Erasure Coding vs. Replication," Proc. IPTPS, pp.226-239, Feb. 2005.
[12] G. Chen, T. Qiu, F. Wu, "Insight into redundancy schemes in DHTs," Journal of Supercomputing, Vol.43,pp.183-198, Feb. 2008.
[13] E. Sit, A. Haeberlen, F. Dabek, B. G. Chun, H. Weatherspoon, R. Morris, M. F. Kaashoek, and J. Kubiatowicz, "Proactive replication for data durability," Proc. IPTPS, Feb. 2006.
[14] D. Leonard, Z. Yao, V. Rai and D. Loguinov, "On Lifetime-Based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks," IEEE Trans. Networking, Vol.15,Issue.3,pp.644-655, June 2007.