Core Persistence in Peer-to-Peer Systems: Relating Size to Lifetime

Vincent Gramoli, Anne-Marie Kermarrec, Achour Mostefaoui, Michel Raynal, Bruno Sericola

Core Persistence in Peer-to-Peer Systems: Relating Size to Lifetime

Vincent Gramoli, Anne-Marie Kermarrec
Achour Mostefaoui, Michel Raynal, and Bruno Sericola

IRISA (Université de Rennes 1 and INRIA)
Campus de Beaulieu, 35042 Rennes, France.
{vgramoli,akermarr,achour,raynal,bsericol}@irisa.fr

Abstract. Distributed systems are now both very large and highly dynamic. Peer to peer overlay networks have been proved efficient to cope with this new deal that traditional approaches can no longer accommodate. While the challenge of organizing peers in an overlay network has generated a lot of interest leading to a large number of solutions, maintaining critical data in such a network remains an open issue. In this paper, we are interested in defining the portion of nodes and frequency one has to probe, given the churn observed in the system, in order to achieve a given probability of maintaining the persistence of some critical data. More specifically, we provide a clear result relating the size and the frequency of the probing set along with its proof as well as an analysis of the way of leveraging such an information in a large scale dynamic distributed system.

Keywords: Churn, Core, Dynamic system, Peer to peer system, Persistence, Probabilistic guarantee, Quality of service, Survivability.

1 Introduction

Context of the paper. Persistence of critical data in distributed applications is a crucial problem. Although static systems have experienced many solutions, mostly relying on defining the right degree of replication, this remains an open issue in the context of dynamic systems.

Recently, peer to peer (P2P) systems became popular as they have been proved efficient to cope with the scale shift observed in distributed systems. A P2P system is a dynamic system that allow peers (nodes) to join or leave the system. In the meantime, a natural tendency to trade strong deterministic guarantees for probabilistic ones aimed at coping with both scale and dynamism. Yet, quantifying bounds of guarantee that can be achieved probabilistically is very important for the deployment of applications.

More specifically, a typical issue is to ensure that despite dynamism some critical data is not lost. The set of nodes owning a copy of the critical data is called a *core* (distinct cores can possibly co-exist, each associated with a particular data).

Provided that core nodes remain long enough in the system, a “data/state transfer” protocol can transmit the critical data from nodes to nodes. This ensures that a new core of nodes in the system will keep track of the data. Hence, such protocols provide

data persistence despite the uncertainty of the system state involved by the dynamic evolution of its members.

There is however an inherent tradeoff in the use of such protocols. If the policy that is used is too conservative, the data transfer protocol might be executed too often, thereby consuming resources and increasing the whole system overhead. Conversely, if the protocol is executed too rarely, all nodes owning a copy of the data may leave (or crash) before a new protocol execution, and the data would be lost. This fundamental tradeoff is the main problem addressed in this paper.

Content of the paper. Considering the previous context, we are interested in providing some probabilistic guarantees of maintaining a *core* in the system. More precisely, given the churn observed in the system, we aim at maintaining the persistence of some critical data. To this end, we are interested in defining the portion of nodes that must be probed, as well as the frequency to which this probe must occur to achieve this result with a given probability. This boils down to relating the size and the frequency of the probing set according to a target probability of success and the churn observed in the system.

The investigation of the previous tradeoff relies on critical parameters. One of them is naturally the size of the core. Two other parameters are the percentage of nodes that enter/leave the system per time unit, and the duration during which we observe the system. We first assume that, per time unit, the number of entering nodes is the same as the number of leaving nodes. In other words, the number of nodes remains constant.

Let S be the system at some time τ. It is composed of n nodes including a subset of q nodes defining a core Q for a given critical data. Let S' be the system at time $\tau + \delta$. Because of the system evolution, some nodes owning a copy of the critical data at time τ might have left the system at time $\tau + \delta$ (those nodes are in S and not in S'). So, an important question is the following: "Given a set Q' of q' nodes of S', what is the probability that Q and Q' do intersect?" We derive an explicit expression of this probability as a function of the parameters characterizing the dynamic system. This allows us to compute some of them when other ones are fixed. This provides distributed applications with the opportunity to set a tradeoff between a probabilistic guarantee of achieving a core and the overhead involved computed either as the number of nodes probed or the frequency at which the probing set needs to be refreshed.

Related work. As mentioned above, P2P systems have received a great deal of attention both in academia and industry for the past five years. More specifically, a lot of approaches have been proposed to create whether they are structured, such as Chord [18], CAN [15] or Pastry [16], or unstructured [5,6,9]. Maintenance of such overlay networks in the presence of high churns has also been studied as one of the major goal of P2P overlay networks [10]. The parameters impacting on connectivity and routing capabilities in P2P overlay networks are now well understood.

In structured P2P networks, routing tables contain critical information and refreshment must occur with some frequency depending on the churn observed in the network [2] to achieve routing capabilities. For instance in Pastry, the size of the leaf set (set of nodes whose identities are numerically the closest to current node identity) and its maintenance protocol can be tuned to achieve the routing within reasonable delay stretch and low overhead. Finally, there has been approaches evaluating the number of

locations to which a data has to be replicated in the system in order to be successfully searched by flooding-based or random walk-based algorithms [4]. These approaches do not consider specifically churn in their analysis. In this paper churn is a primary concern. The result of this work can be applied to any P2P network, regardless of its structure, in order to maintain critical data by refreshment at sufficiently many locations.

The use of a base core to extend protocols designed for static systems to dynamic systems has been investigated in [14]. Persistent cores share some features with quorums (i.e., mutually intersecting sets). Quorums originated a long time ago with majority voting systems [7,19] introduced to ensure data consistency. More recently, quorum reconfiguration [11,3] have been proposed to face system dynamism while guaranteeing atomic consistency: this application outlines the strength of such dynamic quorums. Quorum-based protocols for searching objects in P2P systems are proposed in [13]. Probabilistic quorum systems have been introduced in [12]. They use randomization to relax the strict intersection property to a probabilistic one. They have been extended to dynamic systems in [1].

Roadmap. The paper is organized as follows. Section 2 defines the system model. Section 3 describes our dynamic system analysis and our probabilistic results. Section 4 interprets the previous formulas and shows how to use them to control the uncertainty of the key parameters of P2P applications. Finally, Section 5 concludes the paper.

2 System model

The system model, sketched in the introduction is simple. The system consists of n nodes. It is dynamic in the following sense. For the sake of simplicity, let n be the size of the system. Every time unit, cn nodes leave the system and cn nodes enter the system, where c is the percentage of nodes that enter/leave the system per time unit; this can be seen as new nodes "replacing" leaving nodes. Although monitoring the leave and join rates of a large-scale dynamic system remains an open issue, it is reasonable to assume join and leave are tightly correlated in P2P systems. A more realistic model would take in account variation of the system size depending for instance, on night-time and day-time as observed in [17].

A node leaves the system either voluntarily or because it crashes. A node that leaves the system does not enter it later. (Practically, this means that, to re-enter the system, a node that has left must be considered as a new node; all its previous knowledge of the system state is lost.) For instance, initially (at time τ), assume there are n nodes (identified from 1 to n; let us take $n = 5$ to simplify). Let $c = 0.2$, which means that, every time unit, $nc = 1$ node changes (a node disappears and a new node replaces it). That is, at time $\tau + 1$, one node leaves the system and another one joins. From now on, observe that next leaving nodes are either nodes that were initially in the system or nodes that joined after time τ.

3 Relating the key parameters of the dynamic system

This section answers the question posed in the introduction, namely, given a set $Q(\tau)$ of nodes at time τ (the core), and a set $Q(\tau')$ of nodes at time $\tau' = \tau + \delta$, what is the

probability of the event "$Q(\tau) \cap Q(\tau') \neq \emptyset$". In the remaining of this paper, we assume that both $Q(\tau)$ and $Q(\tau')$ contain q nodes, since an interesting goal is to minimize both the number of nodes where the data is replicated and the number of nodes one has to probe to find the data. Let an *initial* node be a node that belongs to the system at time τ. Moreover, without loss of generality, let $\tau = 0$ (hence, $\tau' = \delta$).

Lemma 1. *Let C be the ratio of initial nodes that are replaced after δ time units. We have $C = 1 - (1 - c)^{\delta}$.*

Proof We claim that the number of initial nodes that are still in the system after δ time units is $n(1-c)^{\delta}$. The proof is by induction on the time instants. Let us remind that c is the percentage of nodes that are replaced in one time unit. For the Base case, at time 1, $n - nc = n(1 - c)$ nodes have not been replaced. For the induction case, let us assume that at time $\delta - 1$, the number of initial nodes that have not been replaced is $n(1-c)^{\delta-1}$. Let us consider the time instant δ. The number of initial nodes that are not replaced after δ time units is $n(1 - c)^{\delta-1} - n(1 - c)^{\delta-1}c$, i.e., $n(1 - c)^{\delta}$, which proves the claim. It follows from the previous claim that the number of initial nodes that are replaced during δ time units is $n - n(1 - c)^{\delta}$. Hence, $C = (n - n(1 - c)^{\delta})/n = 1 - (1 - c)^{\delta}$.

$\square_{Lemma\ 1}$

Given a core of q nodes at time τ (each having a copy of the critical data), the following theorem gives the probability that, at time $\tau' = \tau + \delta$, an arbitrary node cannot obtain the data when it queries q nodes arbitrarily chosen.

For this purpose, using result of Lemma 1 we take the number of elements that have left the system during the period δ as $\alpha = \lceil Cn \rceil = \lceil (1 - (1 - c)^{\delta})n \rceil$. This number allows us to evaluate the aforementioned probability.

Theorem 1. *Let $x_1, ..., x_q$ be any node in the system at time $\tau' = \tau + \delta$. The probability that none of these nodes belong to the initial core is*

$$\frac{\sum_{k=a}^{b} \left[\binom{n+k-q}{q} \binom{q}{k} \binom{n-q}{\alpha-k} \right]}{\binom{n}{q} \binom{n}{\alpha}},$$

where $\alpha = \lceil (1 - (1 - c)^{\delta})n \rceil$, $a = \max(0, \alpha - n + q)$, and $b = \min(\alpha, q)$.

Proof The problem we have to solve can be represented in the following way:

The system is an urn containing n balls (nodes), such that, initially, q balls are green (they represent the initial core $Q(\tau)$ and are represented by the set $\mathcal{Q}$ in Figure 1), while the $n - q$ remaining balls are black.

We randomly draw $\alpha = \lceil Cn \rceil$ balls from the urn (according to a uniform distribution), and paint them red. These α balls represent the initial nodes that are replaced by new nodes after δ units of time (each of these balls was initially green or black). After it has been colored red, each of these balls is put back in the urn (so, the urn contains again n balls).

We then obtain the system as described in the right part of Figure 1 (which represents the system state at time $\tau' = \tau + \delta$). The set $\mathcal{A}$ is the set of balls that have been

painted red. $\mathcal{Q}'$ is the core set $\mathcal{Q}$ after some of its balls have been painted red (these balls represent the nodes of the core that have left the system). This means the set $\mathcal{Q}' \setminus \mathcal{A}$, that we denote by $\mathcal{E}$, contains all the green balls and only them.

We denote by β the number of balls in the set $\mathcal{Q}' \cap \mathcal{A}$. It is well-known that β has a hypergeometric distribution, i.e., for $a \leq k \leq b$ where $a = \max(0, \alpha - n + q)$ and $b = \min(\alpha, q)$, we have

$$\Pr[\beta = k] = \frac{\binom{q}{k}\binom{n-q}{\alpha-k}}{\binom{n}{\alpha}}. \tag{1}$$

We finally draw randomly and successively q balls $x_1, ..., x_q$ from the urn (system at time τ') without replacing them. The problem consists in computing the probability of the event {none of the selected balls $x_1, ..., x_q$ are green}, which can be written as $\Pr[x_1 \notin \mathcal{E}, ..., x_q \notin \mathcal{E}]$.

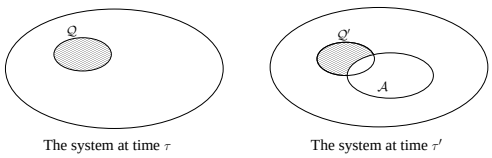

Fig. 1. The system at times τ and $\tau' = \tau + \delta$

As $\{x \in \mathcal{E}\} \Leftrightarrow \{x \in \mathcal{Q}'\} \cap \{x \notin \mathcal{Q}' \cap \mathcal{A}\}$, we have (taking the contrapositive) $\{x \notin \mathcal{E}\} \Leftrightarrow \{x \notin \mathcal{Q}'\} \cup \{x \in \mathcal{Q}' \cap \mathcal{A}\}$, from which we can conclude $\Pr[x \notin \mathcal{E}] = \Pr[\{x \notin \mathcal{Q}'\} \cup \{x \in \mathcal{Q}' \cap \mathcal{A}\}]$. As the events $\{x \notin \mathcal{Q}'\}$ and $\{x \in \mathcal{Q}' \cap \mathcal{A}\}$ are disjoints, we obtain $\Pr[x \notin \mathcal{E}] = \Pr[x \notin \mathcal{Q}'] + \Pr[x \in \mathcal{Q}' \cap \mathcal{A}]$. The system contains n balls. The number of balls in $\mathcal{Q}'$, $\mathcal{A}$ and $\mathcal{Q}' \cap \mathcal{A}$ is equal to q, α and β, respectively. Since there is no replacement, we get,

$$\Pr[x_1 \notin \mathcal{E}, ..., x_q \notin \mathcal{E} \,/\, \beta = k] = \sum_{k=a}^{b} \prod_{i=1}^{q} \left(1 - \tfrac{q-k}{n-i+1}\right) = \sum_{k=a}^{b} \frac{\binom{n-q+k}{q}}{\binom{n}{q}}. \tag{2}$$

To uncondition the aforementioned result (2), we simply multiply it by (1), leading to

$$\Pr[x_1 \notin \mathcal{E}, ..., x_q \notin \mathcal{E}] = \frac{\sum_{k=a}^{b} \left[\binom{n+k-q}{q}\binom{q}{k}\binom{n-q}{\alpha-k}\right]}{\binom{n}{q}\binom{n}{\alpha}}.$$

$\square_{Theorem\ 1}$

4 From formulas to parameter tuning

In the previous section, we have provided a set of formulas that can be leveraged and exploited by distributed applications in many ways. Typically, in a P2P system, the churn rate is not negotiated but observed[1]. Nevertheless, applications deployed on P2P overlays may need to choose the probabilistic guarantees that a node of the initial core is probed. Given such a probability, the application may fix either the size of the probing set of nodes or the frequency at which the core needs to be re-established from the current set of nodes (with the help of an appropriate data transfer protocol).

This section exploits the previous formula to relate these various elements. More precisely, we provide the various relations existing between the three factors that can be tuned by an application designer: the size of the probing set q, the frequency of the probing δ, and the probability of achieving a core characterized by $p = 1 - \epsilon$. (For the sake of clarity all along this section, a ratio C, or c, is sometimes expressed as a percentage. Floating point numbers on the y-axis are represented in their mantissa and exponent numbers.)

Relation linking c and δ. The first parameter that we formalized is C, that can be interpreted as the rate of dynamism in the system. C depends both on the churn rate (c) observed in the system and the probing frequency ($1/\delta$). More specifically, we foresee here a scenario in which an application designer would consider tolerating a churn C in order to define the size of a core and thus ensure the persistence of some critical data. For example, an application may need to tolerate a churn rate of 10% in the system, meaning that the persistence of some critical data should be ensured as long as up to 10% of the nodes in the system change over time. Therefore, depending on the churn observed and monitored in the system, we are able to define the longest period δ before which the core should be re-instantiated on a set of the current nodes. One of the main interest of linking c and δ is that if c varies over time, δ can be adapted accordingly without compromising the initial requirements of the application.

More formally, Lemma 1 provides an explicit value of C (the ratio of initial nodes that are replaced) as a function of c (the replacement ratio per time unit) and δ (the number of time units). Figure 2 represents this function for several values of C. More explicitly, it depicts on a logarithmic scale the curve $c = 1 - \sqrt[\delta]{1 - C}$ (or equivalently, the curve $\delta = \frac{\log(1-C)}{\log(1-c)}$). As an example, the curve associated with $C = 10\%$ indicates that 10% of the initial nodes have been replaced after $\delta = 105$ time units (point A, Figure 2), when the replacement ratio is $c = 10^{-3}$ per time unit. Similarly, the same replacement ratio per time unit entails the replacement of 30% of the initial nodes when the duration we consider is $\delta = 356$ time units (point B, Figure 2). The system designer can benefit from these values to better appreciate the way the system evolves according to the assumed replacement ratio per time unit. To summarize, this result can be used as follows. In a system, aiming at tolerating a churn of X% of the nodes, our goal is to provide an application with the corresponding value of δ, knowing the churn c observed in the system. This gives the opportunity to adjust δ if c changes over time.

[1] Monitoring the churn rate of a system, although very interesting, is out of the scope of this paper.

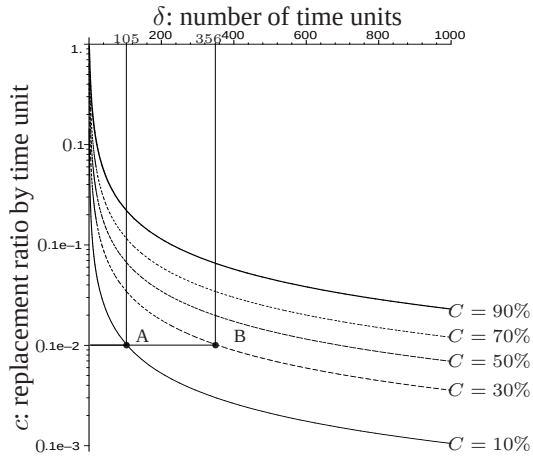

Fig. 2. Evolution of the pair (c, δ) for given values of C

Relation linking the core size q and ϵ Now, given a value C set by an application developer, there are still two parameters that may influence either the overhead of maintaining a core in the system, or the probabilistic guarantee of having such a core. The overhead may be measured in a straightforward manner in this context as the number of nodes that need to be probed, namely q. Intuitively, for a given C, as q increases, the probability of probing a node of the initial core increases. In this section, we define how much these parameters are related.

Let us consider the value ϵ determined by Theorem 1. That value can be interpreted the following way: $p = 1 - \epsilon$ is the probability that, at time $\tau' = \tau + \delta$, one of the q queries issued (randomly) by a node hits a node of the core. An important question is then the following: How are ϵ and q related? Or equivalently, how increasing the size of q enable to decrease ϵ? This relation is depicted in Figure 3(a) where several curves are represented for $n = 10,000$ nodes.

Each curve corresponds to a percentage of the initial nodes that have been replaced. (As an example, the curve 30% corresponds to the case where $C = 30\%$ of the initial nodes have left the system; the way C, δ and c are related has been seen previously.) Let us consider $\epsilon = 10^{-3}$. The curves show that $q = 274$ is a sufficient core size for not bypassing that value of ϵ when up to 10% of the nodes are replaced (point A, Figure 3(a)). Differently, $q = 274$ is not sufficient when up to 50% of the nodes are replaced; in that case, the size $q = 369$ is required (point B, Figure 3(a)).

The curves of both Figure 2 and Figure 3(a) provide the system designer with realistic hints to set the value of δ (deadline before which a data transfer protocol establishing a new core has to be executed). Figure 3(b) is a zoom of Figure 3(a) focusing on the small values of ϵ. It shows that, when $10^{-3} \leq \epsilon \leq 10^{-2}$, the probability $p = 1 - \epsilon$ increases very rapidly towards 1, though the size of the core increases only very slightly. As an example, let us consider the curve associated with $C = 10\%$ in Figure 3(b). It shows that a core of $q = 224$ nodes ensures an intersection probability $= 1 - \epsilon = 0.99$, and a core of $q = 274$ nodes ensures an intersection probability $= 1 - \epsilon = 0.999$.

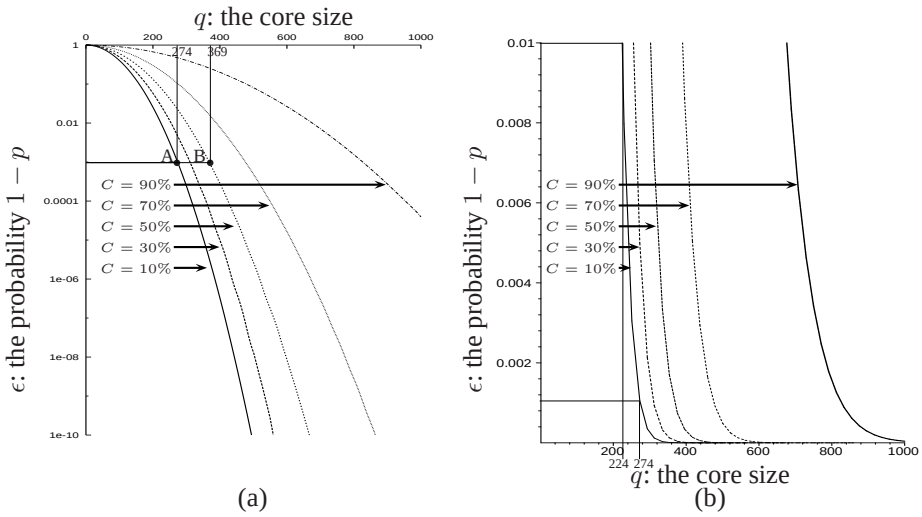

Fig. 3. Non-intersection probability over the core size

Interestingly, this phenomenon is similar to the *birthday paradox*[2] [8] that can be roughly summarized as follows. How many persons must be present in a room for two of them to have the same birthday with probability $p = 1-\epsilon$? Actually, for that probability to be greater than $1/2$, it is sufficient that the number of persons in the room be equal (only) to 23! When, there are 50 persons in the room, the probability becomes 97%, and increases to 99.9996% for 100 persons. In our case, we observe a similar phenomenon: the probability $p = 1 - \epsilon$ increases very rapidly despite the fact that the frequency of the core size q increases slightly.

In our case, this means that the system designer can choose to slightly increase the size of the probing set q (and therefore only slightly increase the associated overhead) while significantly increasing the probability to access a node of the core.

Relation linking q and δ So far, we have considered that an application may need to fix C and then define the size of the probing set to achieve a given probability p of success. There is another remaining trade-off that an application designer might want to decide upon: trading the size of the probing set with the probing frequency while fixing the probability $p = 1 - \epsilon$ of intersecting the initial core. This is precisely defined by relating q to δ for a fixed ϵ.

In the following we investigate the way the size and lifetime of the core are related when the required intersection probability is 99% or 99.9%. We chose these values to better illustrate our purpose, as we believe they reflect what could be expected by an application designer. For both probabilities we present two different figures summarizing the required values of q.

Figure 4 focuses on the core size that is required in a static system and in a dynamic system (according to various values of the ratio C). The static system implies that no nodes leave or join the system while the dynamic system contains nodes that join and leave the system depending on several churn values. For the sake of clarity we omit

[2] The paradox is with respect to intuition, not with respect to logics.

Intersection probability	Churn $C = 1-(1-c)^{\delta}$	Core size $n = 10^3$	$n = 10^4$	$n = 10^5$
99%	static	66	213	677 *
	10%	70	224	714
	30%	79	255	809
	60%	105	337	1071
	80%	143	478	1516
99.9%	static	80 *	260	828 *
	10%	85	274	873 *
	30%	96	311	990 *
	60%	128	413 *	1311
	80%	182	584	1855

Fig. 4. The core size depending on the system sizes and the churn rate.

values of δ and simply present C taking several values from 10% to 80%. The analysis of the results depicted in the figure leads to two interesting observations.

First, when δ is big enough for 10% of the system nodes to be replaced, then the core size required is amazingly close to the static case (873 versus 828 when $n = 10^5$ and the probability is 0.999). Moreover, q has to be equal to 990 only when C increases up to 30%. Second, even when δ is sufficiently large to let 80% of the system nodes be replaced, the minimal number of nodes to probe remains low with respect to the system size. For instance, if δ is sufficiently large to let $6,000$ nodes be replaced in a system with $10,000$ nodes, then only 413 nodes must be randomly probed to obtain an intersection with probability $p = 0.999$.

To conclude, these results clearly show that a critical data in a highly dynamic system can persist in a scalable way: even though the delay between core re-establishments is reasonably large while the size of the core remains relatively low.

5 Conclusion

Maintenance of critical data in large-scale dynamic systems where nodes may join and leave dynamically is a critical issue. In this paper, we define the notion of persistent core of nodes that can maintain such critical data with a high probability regardless of the structure of the underlying P2P network. More specifically, we relate the parameters that can be tuned to achieve a high probability of defining a core, namely the size of the core, the frequency at which it has to be re-established, and the churn rate of the system.

Our results provide application designers with a set of guidelines to tune the system parameters depending on the expected guarantees and the churn rate variation. An interesting outcome of this paper is to show that slightly increasing the size of the core result in a significant probability increase of the guarantee.

This work opens up a number of very interesting research directions. An interesting question is related to the design and evaluation of efficient probing protocols, defining such a core in the system applicable to a large spectrum of peer to peer overlay networks. Monitoring the system in order to estimate the churn rate is another interesting issue.

References

1. I. Abraham and D. Malkhi. Probabilistic quorum systems for dynamic systems. *Distributed Systems*, 18(2):113–124, 2005.
2. M. Castro, M. Costa, and A. Rowstron. Performance and dependability of structured peer-to-peer overlays. In *Proc. Int'l IEEE Conference on Dependable Systems and Networks (DSN)*, pages 9–18, Florence (Italy), 2004.
3. G. Chockler, S. Gilbert, V. Gramoli, P. M. Musial, and A. A. Shvartsman. Reconfigurable distributed storage for dynamic networks. In *Proceedings of 9th International Conference on Principles of Distributed Systems (OPODIS)*, Dec. 2005.
4. E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proc. of ACM SIGCOMM*, pages 177–190. ACM Press, 2002.
5. P. T. Eugster, R. Guerraoui, S. B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Trans. on Comp. Systems*, 21(4):341–374, 2003.
6. A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Trans. on Computers*, 52(2):139–149, February 2003.
7. D. K. Gifford. Weighted voting for replicated data. In *Proc. 7th Int'l ACM Symposium on Operating Systems Principles (SOSP)*, pages 150–162. ACM Press, 1979.
8. R. Isaac. *The pleasure of probabilities*. Springer Verlag, Reading in Mathematics, 1995.
9. M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In H.-A. Jacobsen, editor, *Middleware 2004*, volume 3231 of *LNCS*, pages 79–98. Springer-Verlag, 2004.
10. D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing (PODC)*, pages 233–242, New York, NY, USA, 2002. ACM Press.
11. N. Lynch and A. Shvartsman. RAMBO: A reconfigurable atomic memory service for dynamic networks. In *Proc. of 16th International Symposium on Distributed Computing*, pages 173–190, 2002.
12. D. Malkhi, M. Reiter, A. Wool, and R. Wright. Probabilistic quorum systems. *Information and Computation*, 170(2):184–206, 2001.
13. K. Miura, T. Tagawa, and H. Kakugawa. A quorum-based protocol for searching objects in peer-to-peer networks. *IEEE Trans. on Parallel and Distributed Systems*, 17(1):25–37, January 2006.
14. A. Mostefaoui, M. Raynal, C. Travers, S. Peterson, A. E. Abbadi, and D. Agrawal. From static distributed systems to dynamic systems. In *Proc. 24th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 109–119, 2005.
15. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM*, pages 161–172, San Diego, CA, 2001. ACM Press.
16. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In R. Guerraoui, editor, *Middleware 2001*, volume 2218 of *LNCS*, pages 329–350. Springer-Verlag, 2001.
17. S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *In Proceedings of Multimedia Computing and Networking*, 2002.
18. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM*, pages 149–160, San Diego (CA), 2001. ACM Press.
19. R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. on Database Systems*, 4(2):180–209, 1979.