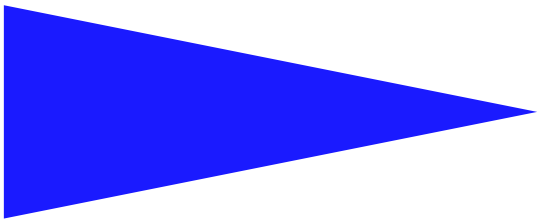
Core Persistence in Peer-to-Peer Systems: Relating Size to Lifetime

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

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1799 version 2



CORE PERSISTENCE IN PEER-TO-PEER SYSTEMS: RELATING SIZE TO LIFETIME

V. GRAMOLI A-M. KERMARREC
A. MOSTEFAOUI M. RAYNAL B. SERICOLA

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

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

V. Gramoli A-M. Kermarrec
A. Mostefaoui M. Raynal B. Sericola

Systèmes communicants
Projet ASAP

Publication interne n ° 1799 version 2 — Version initiale Avril 2006
Version révisée Juin 2006 * — 16 pages

Abstract: Distributed systems are now both very large and highly dynamic. Peer to peer overlay networks have been proven efficient to cope with this new deal that traditional approaches can no longer accommodate. While organizing peers in an overlay network have 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.

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

(Résumé : tsvp)

* Révision à partir du Lemme 2: Le nombre de noeuds quittant le noyau est une variable aléatoire discrète.

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes
Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Persistence d’un noyau dans un système pair-à-pair

Résumé : De nos jours, les systèmes dynamiques sont à la fois très vastes et fortement dynamiques. Les couches de communication dans les réseaux pair-à-pair sont efficaces pour ce type de systèmes. Bien que l’organisation de pairs en couches de communication connait de nombreuses solutions, maintenir une donnée critique dans un tel système reste un problème ouvert. Dans cet article nous nous interessons au nombre de nœuds à contacter ainsi qu’à la fréquence avec laquelle ce contact se fait pour maintenir la donnée critique avec une probabilité donnée et en dépit du roulement. Autrement dit, nous proposons un résultat sur la relation entre la taille et la fréquence de l’ensemble á contacter, sa preuve, et une analyse sur la manière d’utiliser cette information dans un système dynamique à grande échelle.

Mots clés : Roulement, Noyau, Système dynamique, Système pair à pair, Persistance, Garantie probabiliste, Qualité de service, Survie.

1 Introduction

Context of the paper Maintaining the persistence of some critical data in a distributed system is crucial for many distributed applications. While a range of solutions have been proposed in static contexts, mostly relying on defining the right degree of replication, this remains an open issue in the context of dynamic systems.

Peer to peer (P2P) systems recently received a lot of attention as they have been proven efficient to cope with the recent scale shift observed in distributed systems. A P2P system is a dynamic system that allow peers (nodes) to dynamically join or leave the system. P2P systems get automatically reorganized to cope with the high churn rate of those systems. At the same time, in an attempt to cope with both scale and dynamism, there has been a natural tendency to trade strong deterministic guarantees for probabilistic ones in such systems. Yet, quantifying bounds of guarantee that can be achieved probabilistically is very important for the deployment of applications.

More specifically, a crucial issue in such a context consists in ensuring that, despite new arrivals and departures, some critical data not to be lost. The set of the nodes that have a copy of the critical data is sometimes 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, the critical data can be passed from nodes to nodes by executing a "data/state transfer" protocol that, from the nodes currently in the system, establishes a new core for the critical data. Such protocols are consequently capable to provide data persistence despite the uncertainty on the system state created by the dynamic evolution of its members.

There is nevertheless an inherent tradeoff in the use of such data transfer protocols. If the policy that is used is too conservative, it is possible that the data transfer protocol is executed too often, thereby consuming resources and increasing the whole system overhead. On the opposite, if the data transfer protocol is executed too rarely, it is possible that data becomes lost because all the nodes that have a copy of it leave (or crash) before a new protocol execution is launched. This crucial 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 specifically, we are interested in defining the portion of nodes that have to be probed, as well as the frequency to which this probing has to take place, given the churn observed in the system in order to achieve a given probability of maintaining the persistence of some critical data. This translates into 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. In the paper we first assume that, per time unit, the number of nodes that enter the system is the same as the number of nodes that leave the system (the more general case is dealt with later). This means that, despite the system evolution, 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 that defines a core Q for a given critical data. Let S' be the system at time $\tau + \delta$. According to the evolution of the system, some nodes that have 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 after δ time units". We derive in this paper an explicit expression of this probability as a function of the parameters characterizing the dynamic system. This allows us to compute some of them, as soon as the other ones have been 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 previously mentioned, peer to peer 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 [20], CAN [18] or Pastry [19], or unstructured [5, 7, 10]. Maintenance of such overlay networks in the presence of high churns has also been extensively studied as one of the major goal of P2P overlay networks. The parameters involved in maintaining connectivity and/or routing capabilities in P2P overlay networks is now well understood both in structured and unstructured overlay networks.

In structured P2P networks, this translates into the construction of the routing table and the frequency to which it has to be refreshed to achieve routing capabilities depending on the churn observed in the network [2]. As an example, in Pastry, the leaf set of a given node (a set of k nodes whose identities are numerically the closest to current node identity) enables to ensure that the routing can always take place as long as this set is aggressively maintained. The size of the leaf set and the associated maintenance protocol can be tuned to achieve the routing within reasonable delay stretch and low overhead. Structured overlay networks provide a natural support to implement distributed hash table (DHT) capabilities. They map keys to nodes and ensure that a message routed to a given key will always reach the associated node. Therefore, by sufficiently replicating an object on the closest nodes (in the numerical space of the identities), an application can tolerate nodes joining and leaving the network. Nevertheless, the degree of replication is directly connected to the maximum number of tolerated "failures" (a failure is either a failure or a leave). A data replicated $k + 1$ times may survive k failures.

The connectivity property of unstructured P2P networks is mostly guaranteed depending on the number of neighbors that each node has to maintain and the way a node chooses its neighbors [10]. Also there has been a number of approaches evaluating the degree to which a data has to be replicated in the system in order to be successfully searched by flooding-based or random walks-based algorithms [4]. These approaches do not consider churn specifically in their analysis.

The results presented in this paper, where churn is a primary concern, may be applied to any P2P overlay network, regardless of its structure.

The use of a base core to extend protocols designed for static systems to dynamic systems has been investigated in [17]. Persistent cores share some features with quorums. Quorums originated a long time ago with majority voting systems [8, 21] introduced to ensure data consistency. A quorum system is a set of pairwise intersecting sets. Quorum-based protocols for searching objects in P2P

systems are proposed in [16]. Probabilistic quorum systems have been introduced in [13]. They use randomization to relax the strict intersection property to a probabilistic one. They have been extended to dynamic systems in [1].

Roadmap The rest of the paper is organized as follows. Section 2 defines the system model. Then, Section 3 describes our analysis of the dynamic system and presents the probabilistic results. Section 4 provides an interpretation of the previous formulas and shows how they may be exploited to control the uncertainty associated with the key parameters of P2P applications. Finally, Section 5 discusses the approach and Section 6 concludes the paper.

2 Computation model

The computation model, sketched in the introduction, is very simple. The system consists of n nodes. It is dynamic in the following sense. 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. (The case where the size of the system is not known and varies is considered in Section 5.1.)

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 is considered as a new node; all its previous knowledge of the system state is lost.)

Figure 1 describes a possible system evolution. Initially (time τ), 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). Then, at time $\tau + 1$ the node 2 is replaced by the node $2'$. Let $\delta = 4$. At time $\tau + 4$, we see that the nodes 3 and n have been replaced by the nodes $3'$ and n', respectively, while the new node $2'$ has in turn been replaced by the node $2''$ and the nodes 1 and 4 still belong to the system. The important point here is that a new node can in turn be replaced at a later time.

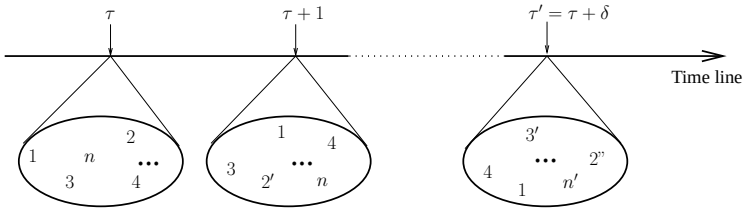


Figure 1: System evolution

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 q nodes at time τ (the core), and a set $Q(\tau')$ of q nodes at time $\tau' = \tau + \delta$, which is the probability of the event "$Q(\tau) \cap Q(\tau') \neq \emptyset$".

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.

- Base case. At time 1, $n - nc = n(1 - c)$ nodes have not been replaced.
- 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 2 *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 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 2), 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 2 (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

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

- 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 $\Pr[x_1 \notin \mathcal{E}, ..., x_q \notin \mathcal{E}]$.

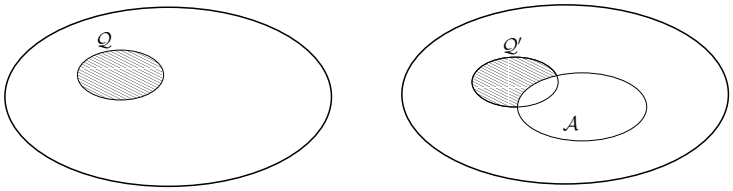


Figure 2: The system at times τ and $\tau' = \tau + \delta$

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. Therefore, we have, from Figure 2,

$$\Pr[x_1 \notin \mathcal{E} \mid \beta = k] = 1 - \frac{q-k}{n}.$$

Since there is no replacement, we easily get in the same way,

$$\Pr[x_1 \notin \mathcal{E}, x_2 \notin \mathcal{E} \mid \beta = k] = \left(1 - \frac{q-k}{n}\right)\left(1 - \frac{q-k}{n-1}\right),$$

and

$$\begin{aligned}
\Pr[x_1 \notin \mathcal{E}, ..., x_q \notin \mathcal{E} \mid \beta = k] &= \sum_{k=a}^{b} \prod_{i=1}^{q} \left(1 - \frac{q-k}{n-i+1}\right), \\
&= \sum_{k=a}^{b} \prod_{i=1}^{q} \left(1 - \frac{n-i-q+k}{n-i+1}\right), \\
&= \sum_{k=a}^{b} \frac{\binom{n-q+k}{q}}{\binom{n}{q}}.
\end{aligned}$$

We obtain the result by unconditionning, i.e.,

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

$\square_{Theorem\ 2}$

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

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

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 ϵ. (For the sake of clarity all along this section, a ratio C,or c, is sometimes expressed as a percentage. Moreover, e^{-z} is used to denote 10^{-z}.)

4.1 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.

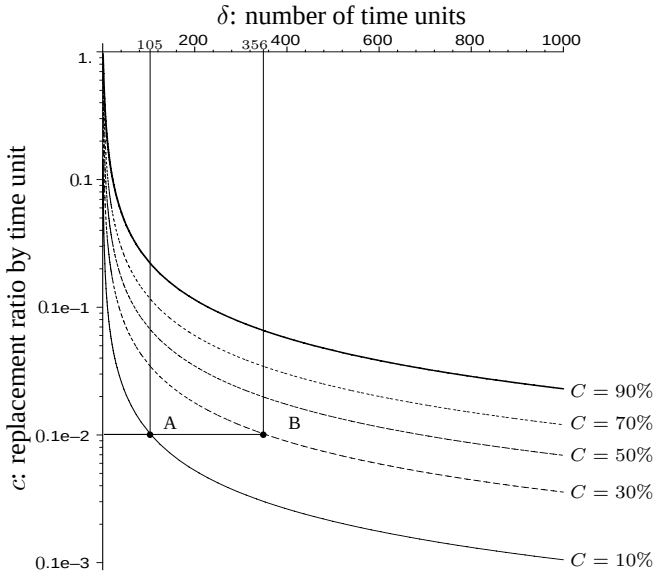


Figure 3: Evolution of the pair (c, δ) for given values of C

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 3 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 3), 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 3). 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.

4.2 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 to what extent these parameters are related.

Let us consider the value ϵ determined by Theorem 2. 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 allows decreasing ϵ? This relation is depicted in Figure 4 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 4). 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 4).

The curves of both Figure 3 and Figure 4 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 5 is a zoom of Figure 4 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 in a very slight way. As an example, let us consider the curve associated with $C = 10\%$ in Figure 5. 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 $= 1 - \epsilon = 0.999$.

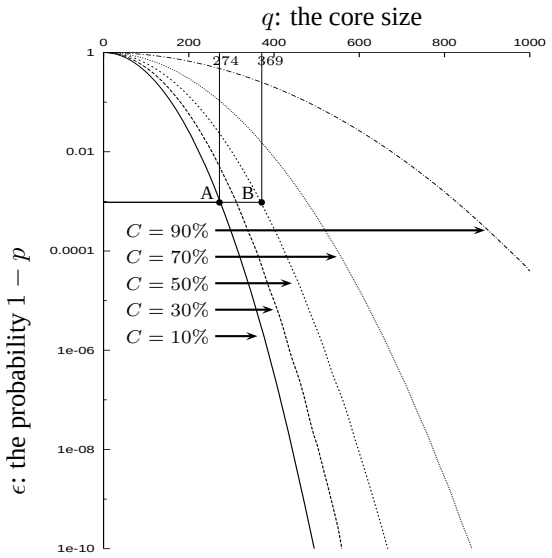


Figure 4: Core size for the intersection probability $p = 1 - \epsilon$

Interestingly, this phenomenon is similar to the *birthday paradox*[2] [9] 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.

4.3 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.

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

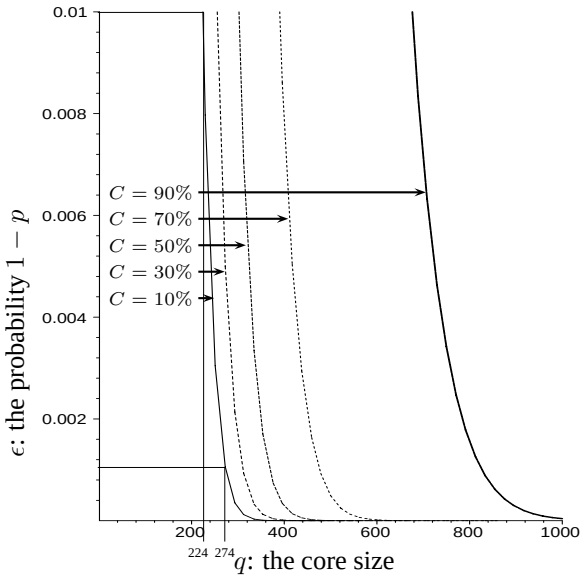


Figure 5: Core size as the probability $p = 1 - \epsilon$ approaches 1

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 6 presents the required core size in case the system is static (i.e., no node leaves the system), while Figure 7 presents the corresponding core size in a dynamic system. (The starred values are the values used in the explanations that follow.)

$1 - \epsilon$	$1 - (1 - c)^{\delta}$	$q(n = 10^3)$	$q(n = 10^4)$	$q(n = 10^5)$
99%	0	66	213	677 *
99.9%	0	80 *	260	828 *

Figure 6: Minimal core size in the static case

In Figure 6, the ratio $C = 1 - (1 - c)^{\delta}$ is equal to 0 (the system is static). As an example, the figure shows that a core of size $q = 80$ is sufficient for achieving an intersection probability $p = 99.9\%$ in a system of $n = 10^3$ nodes, and a core of size $q = 677$ is sufficient for achieving an intersection probability $p = 99\%$ in a system of $n = 10^5$ nodes.

Figure 7 focuses on the core size that is required according to various values of the ratio C. Recall that c is the ratio of node replacement by unit of time as observed in the system (and consequently,

$1-\epsilon$	$1-(1-c)^{\delta}$	$q(n=10^3)$	$q(n=10^4)$	$q(n=10^5)$
99%	10%	70	224	714
	30%	79	255	809
	60%	105	337	1071
	80%	143	478	1516
99.9%	10%	85	274	873 *
	30%	96	311	990 *
	60%	128	413 *	1311
	80%	182	584	1855

Figure 7: Minimal core size in the dynamic case

tuning δ is equivalent to tuning C). For the sake of clarity we omit 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 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 regarding to the system size. For instance, if δ is sufficiently large to let $6,000$ nodes being 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 Discussion

5.1 Dealing with approximate system size

The major contribution of this paper is to relate the various parameters involved in the establishment of a persistent core in a large-scale dynamic system. The number of nodes in the system, n is explicitly needed to correctly set the parameters, although the system size is not globally known in a large-scale peer to peer network.

However, this is not an issue as an approximation of n is sufficient to correctly set the parameters involved in the definition of a persistent core. Several approaches to dynamically, and in a fully decentralized way, estimate th system size exist in the literature. Three main approaches to distributed counting approaches can be distinguished. The first one rely on probabilistic polling approaches. The basic idea of such approaches is to probe the network in a probabilistic way and to infer the size

of the systems based on the replies [6, 12]. The second approach relies on epidemic algorithms [11] and provides very accurate information. The last class of approaches rely on random walks such as the *Sample and collide* algorithm [14]. A comparison of the efficiency and accuracy of candidates approaches from these three classes can be found in [15].

5.2 Applications

The data persistence is a major requirement for numerous applications and an important issue in P2P systems because of the nodes unpredictable volatility. Consider resource allocation applications. The information (whether or not a node is in the critical section) must persist. Otherwise, either every node might wait for entering the critical section forever, or a node may enter the critical section while another is already in. Dedicating a re-establishing core to register if a node is currently in the critical section would help implementing such an application.

Likewise, dynamic distributed shared memory requires critical data to be persistent [3]. Assuming that nodes testifying of the last value written leave the system, further read operations would return a stale value, thus, violating consistency. A solution would be in associating the last value written to a core. Then, forcing the system to execute every δ-period a write operation would re-establish the core sufficiently to ensure atomic consistency. This means that the core re-establishment mechanism presented here is a fundamental building block for largely used applications.

6 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 defined the notion of a persistent core of nodes that can maintain such critical data regardless of the structure of the underlying peer to peer network with a high probability. More specifically, we related 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 set the system parameters depending on the expected guarantees and to adjust them as the churn rate varies in the system over time. An interesting outcome of this paper is that we show that slightly increasing the size of the core result in a significant increase in the probabilistic guarantee that can be achieved.

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 network. Monitoring the system in order to estimate the churn rate is another interesting issue.

References

[1] Ittai Abraham and Dahlia Malkhi. Probabilistic quorum systems for dynamic systems. *Distributed Systems*, 18(2):113–124, 2005.

[2] Miguel Castro, Manuel Costa, and Antony Rowstron. Performance and dependability of structured peer-to-peer overlays. In *Proc. Int'l IEEE Conference on Dependable Systems and Networks (DSN(04)*, pages 9–18, Florence (Italy), 2004.

[3] Gregory Chockler, Seth Gilbert, Vincent Gramoli, Peter M. Musial, and Alex A. Shvartsman. Reconfigurable distributed storage for dynamic networks. In *Proceedings of 9th International Conference on Principles of Distributed Systems (OPODIS'05)*. Springer-Verlag, Dec. 2005. to appear.

[4] Edith Cohen and Scott Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proc. Int'l ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'02)*, pages 177–190. ACM Press, 2002.

[5] Patrick Th. Eugster, Rachid Guerraoui, Sidath B. Handurukande, Anne-Marie Kermarrec, and Petr Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003.

[6] Ted Friedman and Don Towsley. Multicast session membership size estimation. In *Proceedings of the 12th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'93)*, 1999.

[7] Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, February 2003.

[8] David K. Gifford. Weighted voting for replicated data. In *Proc. 7th Int'l ACM Symposium on Operating Systems Principles (SOSP'79)*, pages 150–162. ACM Press, 1979.

[9] Richard Isaac. *The pleasure of probabilities*. Springer Verlag, Reading in Mathematics, 1995.

[10] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In Hans-Arno Jacobsen, editor, *Middleware 2004*, volume 3231 of *LNCS*, pages 79–98. Springer-Verlag, 2004.

[11] Mark Jelasity and Alberto Montresor. Epidemic-style proactive aggregation in large overlay networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 102–109, 2004.

[12] Dionysios Kostoulas, Dimitrios Psaltoulis, Indranil Gupta, Ken Birman, and Alan demers. Decentralized schemes for size estimation in large and dynamic groups. In *Proceedings of 4th IEEE International Symposium on Network Computing and Applications (NCA'05)*, pages 41–48, July 2005.

[13] Dahlia Malkhi, Michael Reiter, Avishai Wool, and Rebecca Wright. Probabilistic quorum systems. *Information and Computation*, 170(2):184–206, 2001.

[14] Laurent Massoulié, Erwan Le Merrer, Anne-Marie Kermarrec, and Ayalvadi Ganesh. Peer counting and sampling in overlay networks: random walk methods. In *Twenty-Fifth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2006)*, Denver (CO), 2006. to appear.

[15] Erwan Le Merrer, Anne-Marie Kermarrec, and Laurent Massoulié. Peer to peer size estimation in large and dynamic networks: A comparative study. In *15th International Symposium on High performance Distributed Computing (HPDC)*, Paris, France, 2006. to appear.

[16] Ken Miura, Taro Tagawa, and Hirotsugu Kakugawa. A quorum-based protocol for searching objects in peer-to-peer networks. *IEEE Transactions on Parallel and distributed Systems*, 17(1):25–37, January 2006.

[17] Achour Mostefaoui, Michel Raynal, Corentin Travers, Stacy Peterson, Amr El Abbadi, and Divyakant Agrawal. From static distributed systems to dynamic systems. In *Proc. 24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 109–119. IEEE Computer Society Press, 2005.

[18] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 161–172, San Diego, CA, 2001. ACM Press.

[19] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Rachid Guerraoui, editor, *Middleware 2001*, volume 2218 of *LNCS*, pages 329–350. Springer-Verlag, 2001.

[20] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. Int'l ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'01)*, pages 149–160, San Diego (CA), 2001. ACM Press.

[21] Robert H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4(2):180–209, 1979.