

Peer to peer size estimation in large and dynamic networks: A comparative study

Erwan Le Merrer, Anne-Marie Kermarrec, Laurent Massoulié

▶ **To cite this version:**

Erwan Le Merrer, Anne-Marie Kermarrec, Laurent Massoulié. Peer to peer size estimation in large and dynamic networks: A comparative study. [Research Report] RR-5842, INRIA. 2006, pp.21. inria-00070184

HAL Id: inria-00070184
https://inria.hal.science/inria-00070184

Submitted on 19 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Peer to peer size estimation in large and dynamic networks: A comparative study

Erwan Le Merrer , Anne-Marie Kermarrec and Laurent Massoulié

N°5842

Février 2006

(hors thème INRIA)



ISSN 0249-6399 ISRN INRIA/RR--5842--FR+ENG



Peer to peer size estimation in large and dynamic networks: A comparative study

Erwan Le Merrer [*] , Anne-Marie Kermarrec [†] and Laurent Massoulié [‡]

(hors thème INRIA)
Projet Paris

Rapport de recherche n° 5842 — Février 2006 — 21 pages

Abstract: As the size of distributed systems keeps growing, the peer to peer communication paradigm has been identified as the key to scalability. Peer to peer overlay networks are characterized by their self-organizing capabilities, resilience to failure and fully decentralized control. In a peer to peer overlay, no entity has a global knowledge of the system. As much as this property is essential to ensure the scalability, monitoring the system under such circumstances is a complex task. Yet, estimating the size of the system is a core functionality for many distributed applications to parameter setting or monitoring purposes. In this paper, we propose a comparative study between three algorithms that estimate in a fully decentralized way the size of a peer to peer overlay. Candidate approaches are generally applicable irrespective of the underlying structure of the peer to peer overlay. The paper reports the head to head comparison of estimation system size algorithms. The simulations have been conducted using the same simulation framework and inputs and highlight the differences in cost and accuracy of the estimation between the algorithms both in static and dynamic settings.

Key-words: peer to peer, counting, size estimation, comparison

(Résumé : tsvp)

[*] erwan.lemerrer@rd.francetelecom.com
[†] Anne-Marie.Kermarrec@irisa.fr
[‡] lmassoul@microsoft.com

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Peer to peer size estimation in large and dynamic networks: A comparative study

Résumé : Alors que la taille des systèmes distribués ne cesse d'augmenter, le paradigme du pair à pair est considéré comme la clé du passage à l'échelle. Les réseaux pair à pair sont caractérisés par leur capacités d'auto organisation, de résistance aux défaillances et par leur aspect totalement décentralisé. Dans ce cadre d'une couche logique pair à pair, aucune entité ne possède de connaissance globale du système; alors que cette propriété est essentielle pour assurer le passage à l'échelle du système, la supervision de celui ci sous certaines circonstances devient une tâche complexe. L'estimation de la taille du système est pourtant une nécessité pour de nombreuses applications distribuées, pour l'ajustement des paramètres ou pour des besoins de supervisions par exemple. Dans ce papier, nous proposons une étude comparative entre trois algorithmes représentatifs de trois classes de solutions préalablement proposées. Ceux ci estiment, d'une façon totalement distribuée, le nombre de pairs au sein d'une couche réseau pair à pair. Les approches candidates sont applicables indépendamment de la couche sous-jacente. Ce papier fournit une comparaison de ces trois algorithmes, au sein du même simulateur comportant les mêmes paramètres, et ceci afin d'apporter une critique la plus objective possible. Les simulations soulignent des différences notables de précision d'estimation et de coût de ces algorithmes, dans des contextes de réseaux statiques et dynamiques.

Mots-clé : pair à pair, comptage, estimation de taille, comparaison

1 Introduction

The past decade has been marked by a tremendous shift in the scale of distributed systems. One of the most striking consequences is that most of traditional algorithms are no longer relevant in such a context. To cope with an increasing number of participants, the peer to peer communication paradigm has been imposed as a key to fill this gap and many algorithms relies on such a model both in industry and academia.

Peer to peer systems are scalable, self-organizing and resilient to failures. They provide an attractive support for many applications from file sharing systems (e.g. Kazaa, Gnutella) to video over IP applications (e.g. CoolStreeming), or voice on IP applications (e.g. Skype). A peer to peer overlay network connects peers in a logical manner on top of IP so that each peer is aware of a small subset of the network. Yet, the sum of individual decisions based on a restricted knowledge makes the system converging towards global properties.

As much as the fully decentralized nature of a peer to peer system is essential to the scalability, it makes the system monitoring much more complex. Obtaining global statistics on the system becomes a hard issue. However, such statistics might be crucial to adjust the parameters of the system as the structure and characteristics change over time. System size estimation is one of the most crucial statistic to be collected on a system since it can be used for many purposes such as parameter value setting or network monitoring. For example, some peer to peer protocols require this estimation as an input parameter: the constant degree of the Viceroy network [10] requires this information to choose a *level* for an incoming peer. Some gossip-based protocols use this information to compute the number of gossip targets of a message to reach given propagation probabilities [5],[3].

Whereas the number of participants can easily be tracked by a set of servers in server-based systems (e.g. Naspter), it is no longer obvious in a fully unstructured network where each node is connected to a set of random neighbors (e.g. Gnutella, Freenet). In the context of structured overlay networks, several algorithms relying on the actual structure of the network have been proposed ([15, 9, 11, 12]). Such algorithms exploit the fact that node identifiers are uniformly assigned at random. The size estimation may then be directly inferred from the observation of the density of identifiers that fall into a given subset of the global identifier space.

While those methods provide good approximation of the system size, their applicability is strictly limited to those identifier-based overlay networks. Instead, in this paper, we consider generally applicable solutions irrespective of the underlying network topology. The main motivation of this paper is to compare several generally applicable algorithms to estimate the system size in both static and dynamic contexts in an attempt to help application developers to choose the best strategy for a given setting/cost/accuracy. To this end, we have carefully chosen three candidate algorithms, each representative of a class of approaches, for the head to head comparison. The first algorithm relies on random walks, using the inverted birthday paradox, the second candidate is based on probabilistic polling while the last one relies on epidemic-based information aggregation. To the best of our knowledge such a comparison is the first of its kind. The only paper we are aware of comparing different fully distributed

algorithms is [15] comparing the Hops Sampling method, discussed in this paper, with an interval density based approach.

We implemented the three candidates algorithms and simulated their behavior against the same inputs for a fair comparison. In this paper, we report the simulation results and compare the methods along their accuracy, overhead and reactivity to changes in the environment.

The rest of the paper is structured as follow. In the next section, we provide some background of generic solution to estimate in a decentralized way the size of a large system. In Section 3, we successively describe the three candidate algorithms. These methods are then evaluated and compared and the results are reported in Section 4. We discuss the pros and cons of the compared candidates based on the simulation results in Section 5 and conclude in Section 6.

2 Background on generic counting algorithms

In this section, we provide some background on the three main generic distributed counting approaches we have identified.

The algorithms belonging to the first class rely on called *probabilistic polling* or *randomized report* techniques. In such approach, the basic idea is to probe the network in a probabilistic way and to infer the size of the system based on the set of replies. To this end, the initiator node broadcasts a message to all nodes in the overlay, and waits for responses. Upon receipt of a broadcast message, the nodes send back a response with a probability depending on the probability parameter set in the broadcast message [1, 4], or based on their distance to the initiator node [9] for example. In this paper, we chose the last one as a candidate for comparison; the fact of using node's distance to the initiator node could lower message overhead compared to simple probabilistic response, as fewer "far nodes" should reply with messages that will cross an important part of the overlay. This algorithm has also been evaluated in [9] and will permit us by results comparison to validate our simulator approach.

The second class of approaches relies on epidemic algorithms to permanently gather information on the system by constantly exchanging information between peers, mostly in a random manner [7, 6, 17]. Epidemic algorithms have recently received a lot of attention due to their great resilience to failure; they also have been used for aggregation applications [16]. The algorithm presented in [7] is based on a *push-pull anti-entropy epidemic* protocol. In this protocol, each peer periodically selects another peer at random to exchange information with. During this epidemic propagation, an *aggregation* of values will gradually take place, hosted by each cooperating overlay node, thus leading to a global value aggregate representing the targeted system size. This method has proven to be accurate and to converge pretty quickly. We chose this algorithim as a candidate.

Finally, the last class gathers the approaches relying on random walks. For example, [1] proposes a random *increasing* walk on the topology, in an overlay were each node owns an identifier. Starting from the node with the smallest *id*, the message is forwarded toward

higher *id* nodes. This method however suffers from the same drawbacks as density-based approaches since it relies on identifier assignment. [13] first introduces the *Random Tour* method, based on an emulation of the return time of a random walk to the initiating node. The second method proposes [13], the *Sample&Collide* algorithm, uses random walks to produce an unbiased sampling (uniformly at random) of overlay nodes. Those samples are then used to compute the system size according to the *inverted birthday paradox* introduced in [1]. Given the results provided in [13], the overhead of the *Sample&Collide* algorithm is much lower than the one of Random Tour. Therefore, we chose the *Sample&Collide* algorithm as the candidate of this class.

3 Candidate algorithms overviews

3.1 Sample&Collide

The *Sample and collide* method [13] is inspired from the "Inverted Birthday Paradox" introduced in [1]. The "birthday paradox" states that the probability of having two people in a room that have the same birthday is at least $1/2$, for a group of 23 peoples. Sample&Collide builds upon this technique to estimate the system size. The estimation relies on drawing a peer sample uniformly at random and to use the sample to approximate the system size. This is based on the number of samples observed between two identical samples.

The algorithm works as follows: the initiator samples nodes uniformly at random until a sample returns a node that already has been selected. X, the expected number of samples is $\sqrt{2n}$; the system size is thus estimated to $\widehat{N} = X^2/2$.

This algorithm heavily relies on the correctness of the sampling method used. *Sample&Collide* approach improves upon the approach presented in [1] in two ways: first the sample method provides some unbiased results by emulating a continuous random walk (as opposed to discrete time random walks) and second, the way the sample is used is refined.

The uniform peer sampling technique which produces unbiased samples is implemented as follows: the initiator node sets a predefined value $T > 0$. This value is then sent to a neighbor chosen uniformly at random. Each node receiving the message first picks a random number U, uniformly distributed on $[0, 1]$; it then simply decrements T by $\log(U)/d_i$ (d_i is the degree of the current node), and forwards the message to a neighbor, if $T > 0$. Otherwise the current node is the sample node, and it returns its *id* to the initiator.

The accuracy/overhead tradeoff of this algorithm relies on the control parameter l which determines the number of newly sampled nodes that have already been observed. The greater l, the more precise is the estimation, but the higher is the generated overhead.

The details of the algorithm, the analysis and simulation results are available in [13] as well as a comparison with the *Random tour* approach. Results show that the *Sample&Collide* approach improves upon the *Random Tour* and justify the choice of the former in this comparative study.

3.2 HopsSampling

The second candidate approach we chose is called the *HopsSampling* and represents in this comparative study the class of probabilistic polling approaches. Two versions of the Hops Sampling algorithm are presented in [15] and [9]: the *gossipSample* and *minHopsReporting* heuristics.

As a first step to choose the candidate approach for the comparison, we tried to reproduce the results obtained in [15] and [9]. We were able to reproduce the results obtained using the minHopsReporting heuristic. The results we obtained by implementing the pseudo-code in [15] (*gossipSample* heuristic) somehow led to less accurate results. Therefore for the sake of fairness, and after discussions with the authors [14], we decided to choose the system size estimation algorithm relying on the *minHopsReporting* heuristic.

This algorithm is based on the probabilistic polling technique where an initiator spreads messages in the network and estimates the system size based on the replies it gets back. These messages are send back in a probabilistic manner. The HopsSampling algorithm, evaluated in this paper works as follows: an initiator node spreads by gossip a message across the network. Initially this message contains *hopCount* set to 0. At each traversed node, this value is incremented; the lowest *hopCount* value received by a node is remembered and represents its distance from the count initiator. Then, depending on each node's distance (in order to avoid massive flood towards the initiator), there is a probabilistic response message sent to the initiator node: (i) if $hopCount < minHopsReporting$, a response is set with probability 1, else (ii) the response is sent with probability $\frac{1}{gossipTo^{hopCount-minHopsReporting}}$. For each message count received from nodes at a certain distance, the initiator needs to multiply it by the percentage of peers in the network they represents. For example, if $minHopsReporting = 2$, only 25% of nodes with distance 4 will report back. The number of collected responses will thus be multiplied by $\frac{1}{0,25}$, *i.e.* 4. The system size is then extrapolated from theses collected returns.

This algorithm works with a priori fixed parameters values discussed in [15, 9]. We have also talked about those parameters value in [14] to ensure a fair comparison.

3.3 Gossip-based Aggregation

The last chosen approach for comparison is representative of the class of epidemic-based approaches. Epidemic-based protocols have recently received a lot of attention given their scalability properties. In an epidemic protocol, each peer periodically exchanges information with one of its neighbor picked at random. In [7], a gossip-based aggregation protocol is presented and the results, that we were able to reproduce, show that this approach provides accurate results.

This approach [7] relies on the following statement: if exactly one node of the system holds a value equal to 1, and all the other values are equal to 0, the average is $1/N$. The system size could thus be directly computed. To run this algorithm, an initiator should take the value equal to 1, and start gossiping; the reached nodes participate to the process by setting their value to 0. At each predefined cycle, each node in the network chooses one of

its neighbor at random and swaps its estimation parameter. The contacted node does the same (push/pull heuristic of [7]). Both nodes then recompute their estimation as follows:

$$Estimation \leftarrow \frac{Estimation + Neighbor's_Estimation}{2}$$

To provide correct estimations, this algorithm needs to wait a certain number of *rounds* to elapse before computing the size estimation; this period is the required time for the gossip to propagates in the whole overlay and for the values to converge.

This method converges toward exact system size in a stable system. More details may be found in [7].

3.4 Summary

We have chosen three candidate algorithms, that we believe are representative of three classes of generic solutions to estimate the size of a large-scale peer to peer system in a fully decentralized way. They are applicable to any peer to peer overlay in which each node is connected to a set of *random* neighbors. Most of structured overlays, such as e.g. Pastry, fall into this category.

The goal of the comparative study is then to run simulations in the same framework in order to compare those methods not only in terms of accuracy but overhead and resilience to dynamic changes.

4 Evaluation

4.1 Experimental Setup

One of the main metrics along which we compare the different algorithms is their scalability in terms of overhead, as well as their accuracy as the system size varies. Given the considered scale, which could not realistically be reached in a lab testing environment, we evaluated them using a discrete event simulator, able to simulate static and dynamic network configurations. The simulator counts the messages over the network. It does not model the physical network topology nor the queuing delays and packet losses.

The first step of the simulation, as mentioned in the previous section, was to reproduce the results obtained by the authors of each algorithm. This step has been successfully reached for the three considered algorithms and we now compare them in the same simulation environment.

The simulations were run on unstructured peer to peer networks. To this end, we built the peer to peer overlay as a random graph, where each node was provided with a set of k neighbors, chosen uniformly at random among the whole set of peers. We do not consider in this paper the actual construction of such graphs but several approaches exist to build such peer to peer overlay in practice [8]. In order to be as close as possible from real world implementations, the graphs we used in the evaluation where not fully homogeneous graphs where

each node had the same number of neighbors. Instead, each node owns a random number of neighbors, below a given threshold. This reflects for instance heterogeneity between nodes in terms of number of incoming and outgoing links capacities. We also run some tests in the context of homogeneous graphs. This parameter consistently improved all algorithms. Therefore, we chose the worst case setting and present the results of experiments conducted on heterogeneous networks.

In all test configurations, each node has a number of neighbors varying between $log(N)$ and $2*log(N)$, N being the actual size of the system. This value ensures graph connectivity. This represents 10 neighbors max for a 100,000 node overlay, and 8 max for 1,000,000[1].

The links between nodes are assumed to be bidirectional (whenever a node contacts another one, the reached node also has knowledge of communication initiator's existence and keeps a link back to the contact node), leading to an undirected graph. We ensured that overlays used in our experiments are always connected. We verified this property by first initiating a tree-based count technique [1]: if each node is reachable by the count initiator, then the final count is equal to the targeted size.

We scaled up to 1,000,000 node graphs, but dynamic environment was created on 100,000 node graphs for practical considerations. Node removals may introduce a loss of connectivity for the remaining nodes; in this simulation, the nodes that have lost one or several neighbors do not create new links with other nodes.

The rest of the section is organized as follows: We first describe the evaluation criteria. In Section 4.3, we report the results obtained in static contexts and in Section 4.4, we introduce some dynamics in the system to evaluate the various algorithms capabilities to provide reactive estimations. Section 4.5 provides some overhead results.

4.2 Evaluation criteria

We used several metrics to evaluate the different approaches to estimate in a distributed fashion the size of a large-scale dynamic system.

Accuracy One of the most important criterion is the quality of the estimate and the speed to which this accuracy is reached. This is an important parameter for the use of the system size information. For some applications, a quick approximation might be more appropriate than a more accurate one which would take much longer to compute.

Reactivity to changes A second characteristic we want to compare the approaches against is their ability to react to changes in the environment. To this end, we compute the time to react to a growth or increase of the number of peers in the system.

Overhead The third important criterion is the overhead of each approach. In this paper, we compute the overhead as the number of messages required to compute the system

[1] We used 8 neighbors per node for 1,000,000 node graph due to limitations of our computing capacities

size. This information could be extrapolated by forthcoming designers to approximate the bandwidth and the computational resources that will be needed by the application.

4.3 Simulation results in static contexts

In the following set of experiments, the algorithms run on an already constructed overlay where peers are connected to a random graph, which size remains static throughout the experiments. On the following figures, the system size is normalized to 100 to enable us to express the quality of the estimation in terms of percentage.

Sample&collide The quality of *Sample&Collide*'s estimations depends upon the value of the T and l parameters. In the rest of the experiments we set T to 10; this value is sufficient for an accurate sampling [13]. Figures 1 and 2 depict the quality (on the y axis) of an estimation respectively in a 100,000 and 1,000,000 node configuration depending on the number of estimations computed over time (x axis). The dashed curve represents a *oneShot* estimation which is a single non averaged estimation while the continuous curve, *Last10runs*, is the average of the 10 last estimations. In these experiments we set l to 200, we will consider other values later in the evaluation.

Our first observation is that the results show that the *oneShot* curve remains most of the time in a 10% precision window, with some peaks at 20%. The *last10runs* one remains within 3 or 4% of the exact value. Obviously *oneShot* has a very low overhead as compared to the *last10run* one. Therefore, in the rest of the paper, the *Sample&Collide approach* will be related to the *oneShot* configuration.

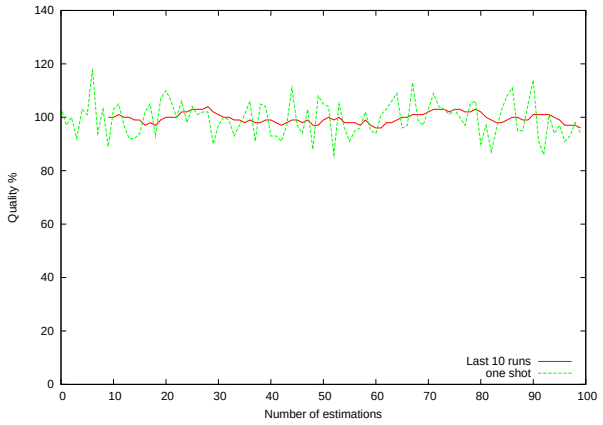


Figure 1: *Sample&collide*: Oneshot and last10runs heuristic with l=200, 100,000 node network, static environment

Hops Sampling We evaluated this algorithm first in small size networks in order to reproduce the results presented in [9], and observed similar results. We used in our implementation

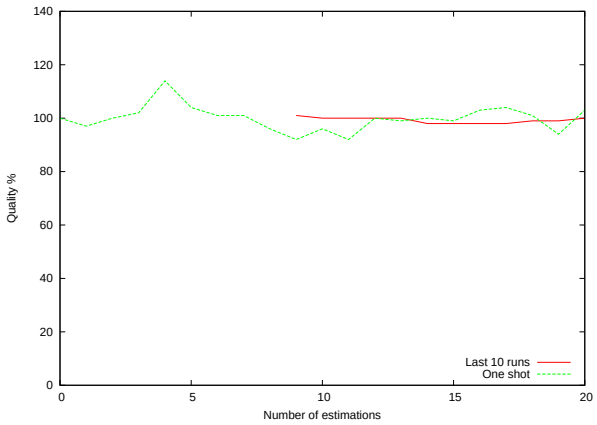


Figure 2: *Sample&collide*: Oneshot and last10runs heuristic with l=200, 1,000,000 node network

the following values for the parameters: $gossipTo = 2$, $gossipFor = 1$, $gossipUntil = 1$, $minHopsReporting = 5$.

Figures 3 and 4 depict the quality of the estimation (y axis), depending on the number of runs (x axis), respectively in a 100,000 and 1,000,000 node configuration. As in *Sample&Collide*, we report the one shot and average over the last 10 runs results. We observe that the algorithm scales well. We also observe not surprisingly that the *last10runs* heuristic provides a less noisy curve. All observed results for *last10runs* remain in a 20% precision range, while *oneShot* has some peaks over 50% of error. Both have a consistent tendency for under estimation.

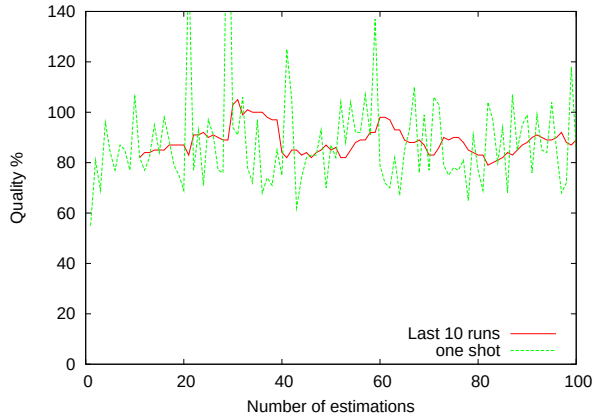


Figure 3: *HopsSampling*: Oneshot and last10runs heuristics, 100,000 node network

Aggregation Finally, we implemented the epidemic-based aggregation protocol [7] and conducted the same set of experiments. The results match the ones presented in the original

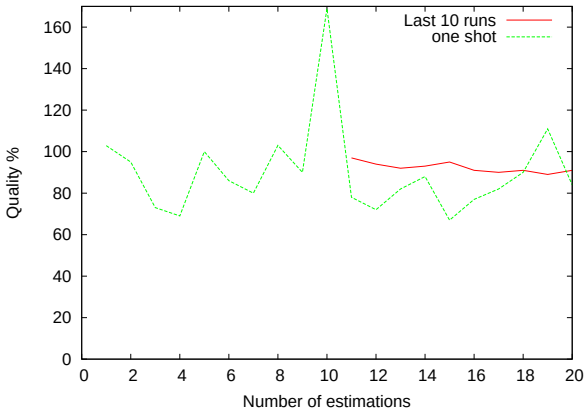


Figure 4: *HopsSampling*: Oneshot and last10runs heuristics, 1,000,000 node network

paper. One initiator node was chosen at random and initiated a gossip-based algorithm by exchanging information with a neighbor picked up at random and averaging the new value.

The results are reported on Figures 5 and 6 for respectively a 100,000 and 1,000,000 node network and give the quality of the estimation given the number of rounds. Experiments show that the size estimation naturally converges towards 100% precision around 40 rounds for 100,000 nodes and around 55 for 1,000,000 node network. The results confirm the accuracy of the algorithm, the quality and scalability of the convergence speed in static settings.

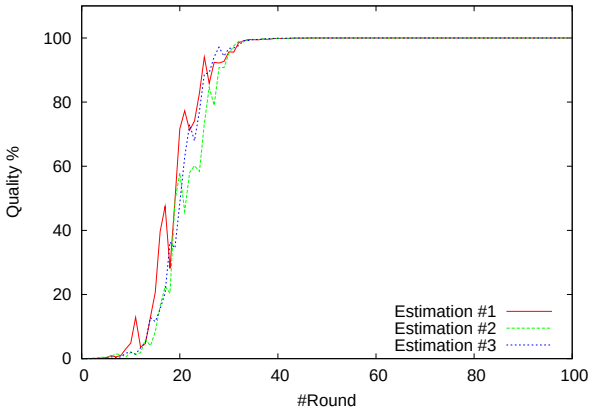


Figure 5: *Aggregation*: 100,000 node network

Static setting, summary Our primary conclusion is that the *Aggregation* algorithm provides accurate, and eventually exact, results in a reasonable number of cycles. This algorithm may leverage the gossip-based underlying protocol if any. In this paper, we do

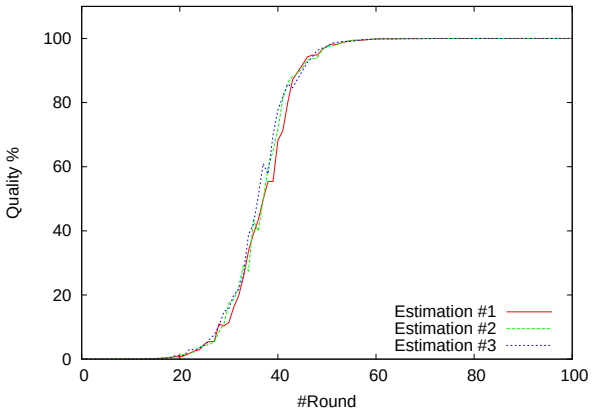


Figure 6: *Aggregation*: 1,000,000 node network

not take into account this underlying protocol. *Aggregation* is a sound approach for static settings, although no formal analysis is provided along with the algorithm.

HopsSampling and *Sample&Collide* produce more noisy curves although the accuracy window remains reasonable. We observed a tendency for *HopsSampling* to produce little underestimated results; this matches experiments in [15, 9]. The fact that the algorithm should not be biased makes us think that this surprising result is due to the fact that not all nodes are reached during the process message spread, or because the distances from the node initiator are not precise enough, probably due to non optimal parameter setting.

4.4 Simulation results in dynamic context

Peer to peer overlay systems are subject to dramatic size variations over time given the versatility of the participating entities. A distributed estimation size algorithm should be able to react to such dramatic changes.

In order to simulate a changing environment, we applied constant nodes arrivals and departures (+/-50%) as well as catastrophic failures (-25%) and evaluated the accuracy of the size estimation in such contexts.

As the number of nodes in the system now keeps varying, the value on the y-axis of the figures is no longer normalized but represents the actual network size.

Sample&Collide The algorithm has to be executed perpetually in order to track size variations; the monitoring process should sample continuously the system in order to provide periodical estimations. Figures 7, 8 and 9 report the evolution of the estimation in a 100,000 node network in three scenarii: a catastrophic failure scenario, a growing scenario and a shrinking one.

All results show that the algorithm reacts very well to changes, even brutal, of sizes in the network and is therefore able to handle effectively network dynamics.

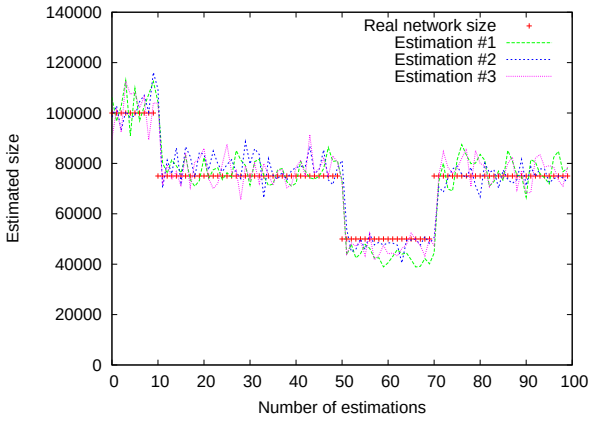


Figure 7: *Sample&collide*: oneShot heuristic, 100,000 node network, catastrophic failures

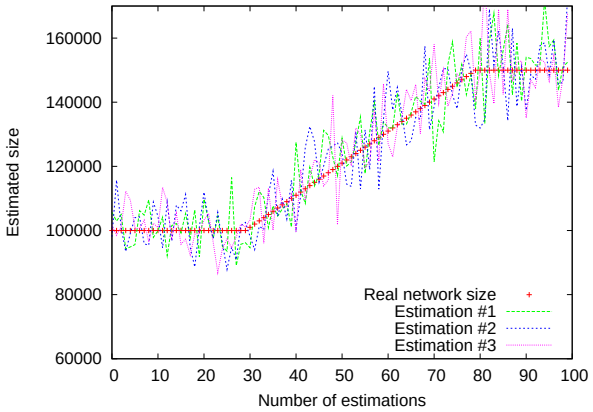


Figure 8: *Sample&collide*: oneShot, 100,000 node network, growing network

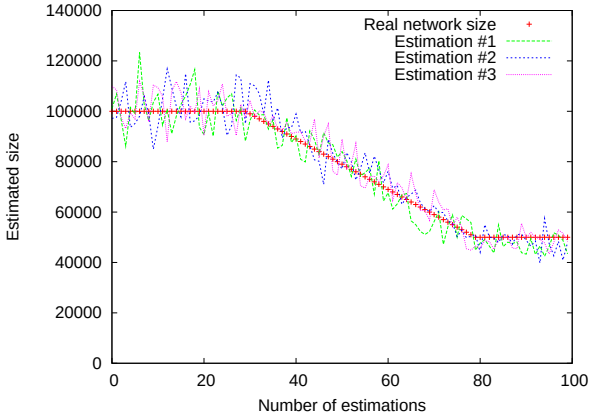


Figure 9: *Sample&collide*: oneShot, 100,000 node network, shrinking network

HopsSampling To cope with changing scenario, the algorithm needs to be restarted periodically, there is no gradual adaptation of the estimation as new polls must be gossiped for each estimation.

Figures 10, 12 and 11 report the evolution of the estimation in a 100,000 node network, implementing the *last10runs* heuristic in the three considered scenarii. The results show a good algorithm behavior facing dynamic network, without any impact due to the overlay degradation or increase. However, we can observe that results remain slightly under estimated and show a higher variation around the real size than in *Sample&Collide*.

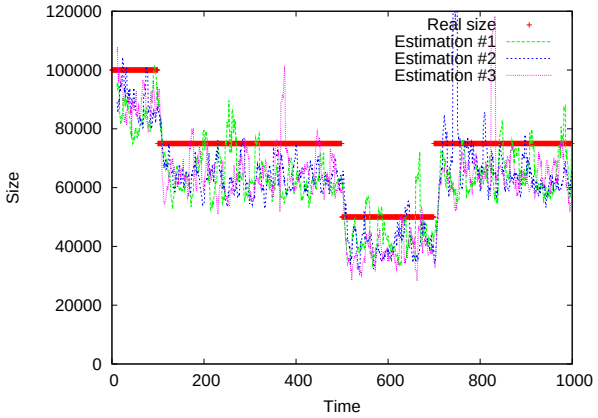


Figure 10: *HopsSampling*: Last10runs heuristic, 100,000 node network, catastrophic failures

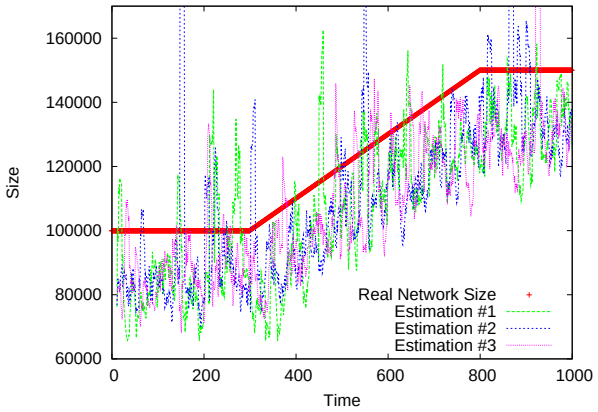


Figure 11: *HopsSampling*: Last10runs heuristic, 100,000 node network, growing network

Aggregation In a dynamic context, the aggregation algorithm is unable, and this is intrinsic to the way the algorithm works due to the value setting, to take node's arrivals/departures into account once an evaluation process is launched. This is due to the fact that during the aggregation process, a node sends its information to a random *alive* neighbor. Thus, there

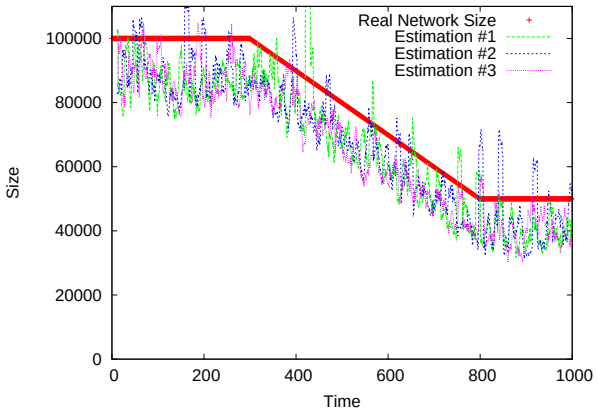


Figure 12: *HopsSampling*: Last10runs heuristic, 100,000 node network, shrinking network

is a conservative effect, as removed nodes no longer participate and as new nodes do not get synchronized information. The overlay size estimated is only accurate at the time where the process was started.

To track size variations, the solution is to reinitialize an aggregation process at regular time intervals. By using tags (unique identifiers) on each new counting process, the algorithm can be reinitialized on demand: a node which is reached by a counting message with a new tag will create a 0 initial value and will starts to participate to the active process. In our simulations, we set this time interval as the number of rounds needed by the gossip process to provide an accurate result. We observed on 5 and 6 that this value is approximately 40 for 100,000 nodes overlay, and 55 for 1,000,000, for a 99% convergence. In order not to make any hypothesis on the targeted system size, we took 50 in the following simulations. This value represents the best possible algorithm's reactivity, in terms of latency, for an accurate estimation.

Figure 13, 14 and 15 report the estimation results in the three considered scenario. We observe that the algorithm does not cope well with the decrease of the network size as observed in the shrinking scenario and the catastrophic scenario. Actually, Figures 15 and 13 show a reasonable algorithm behavior until a certain threshold of nodes departures (around 30% for 100,000 nodes, with 50 rounds per estimation) is reached. We believe that this is due to the loss of connectivity of the overlay. This prevents the propagation of the estimation information across the network. The solution to correct this point is to let a larger number of gossip rounds to elapse before each estimation; this will permit the information to spread better despite low graph connectivity, but will introduce more overhead.

However, we observe that the algorithm provides a fairly good adaptation to a growing network.

Dynamic settings, summary We observe that *Sample&Collide* and *HopsSampling* provide fairly good results when facing brutal as well as gradual changes in the size of the

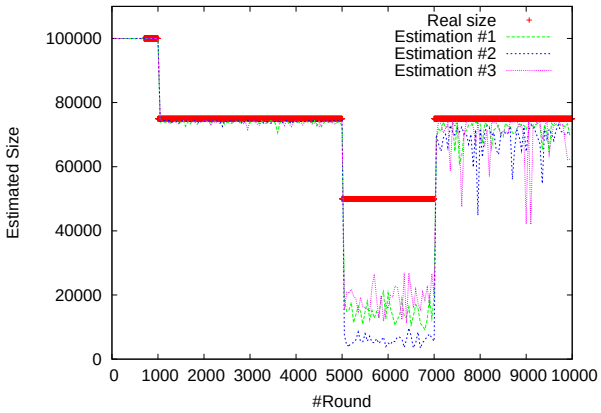


Figure 13: *Aggregation*: Reaction under failures, 100,000 nodes at beginning, -25% of nodes at 100 and 500, +25000 nodes at 700

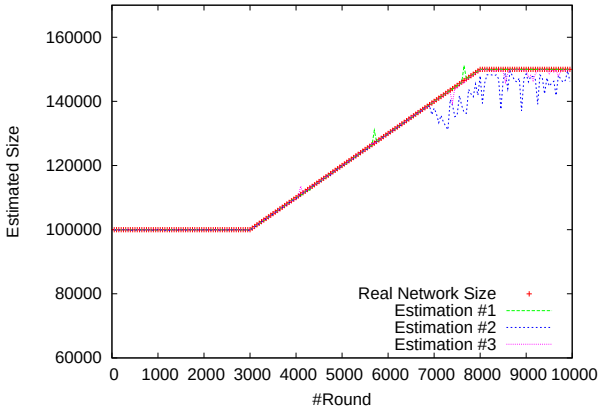


Figure 14: *Aggregation*: Growing network, 100,000 node network

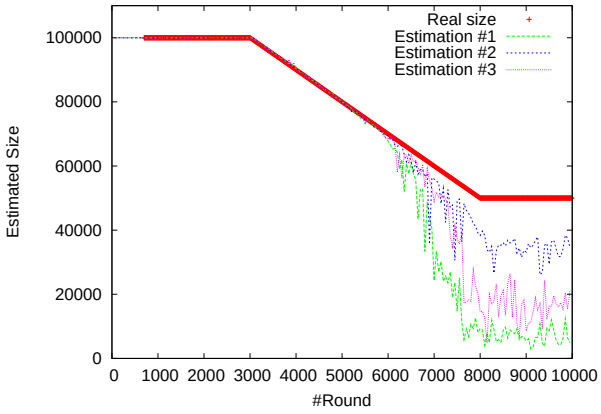


Figure 15: *Aggregation*: Shrinking network, 100,000 node network

system. However, *HopsSampling* as well as *Aggregation* would require an estimation process to be launched periodically.

Sample&Collide provides really reactive results; this could be explained by the one shot heuristic as the algorithm does not keep any memory, while for *HopsSampling* with its last 10 runs average, there is a little convergence time to elapse to produce an accurate estimation facing a brutal topology changes.

Given that some parameters are well tuned, each of those algorithms would be able to provide accurate results. this is then a matter of acceptable overhead as discussed in the next section.

4.5 Algorithms overheads

Besides the accuracy of the estimation, an important parameter is the overhead induced by each method. We measure the overhead of the different algorithms as the total number of messages sent to produce the estimation. This includes spreading messages for *Aggregation* and for *HopsSampling*, return messages for *HopsSampling*, the message associated to the random walk for *Sample&Collide* as well as each sampled node's return. For space reason we do not provide the graphs related to the overhead.

- Sample&Collide: Analysis of costs incurred in the same simulator are provided in [13]. More specifically, we observe good scalability properties as the algorithm with $l = 100$ incurs a cost which is only 3.27 times the one incured for $l = 10$. In order to have a more accurate result, and to compute the associated overhead, we ran some experiments with $l = 200$. The measured overhead of an estimation is around 480,000 messages, which represents 1.40 times the one incurred for $l = 100$.
- Hops Sampling: With current parameters, a single shot estimation consumes $O(2N)$; *last10runs* thus uses $O(10 * 2N)$ messages for an estimation.
- Aggregation: The overhead of an estimation for *Aggregation* algorithm is fairly simple to compute: $Overhead = number_of_nodes * number_of_rounds * 2$. 2 is due to the push/pull heuristic[2].

The overhead associated to each algorithm is fairly simple to compute based on the algorithm characteristics. Table 1 summarizes these results and gives a overhead comparison between the three algorithms, considering the average precision of the estimation. We observe that *Aggregation* provides a nearly perfect estimation with a high overhead compared to the other algorithms. This is the good candidate for really stringent application needs. Note that *Sample&Collide* results, when averaged with the last 10 runs heuristic and with $l = 200$, are at only few percents of the exact system size for a half overhead compared to *Aggregation*. *Sample&Collide* outperforms *HopsSampling* both in terms of the quality of the estimations and the overhead.

[2] the push/pull heuristic means that when two nodes communicates, they both receive the information of the other and re-compute the current estimation

Algorithm	Sample&Collide (l=200)	HopsSampling	Sample&Collide (l=200)	Aggregation
Parameters	One shot	last 10 runs	last 10 runs	50 rounds
Accuracy	+/- 10%	- 20%	+/- 4%	- 1%
Overhead	0,5M	2,5M	5M	10M

Table 1: Example of algorithm's overhead for an estimation (in millions of messages) on a 100,000 node overlay

One advantage of *Sample&Collide* and *HopsSampling* is that they provide an opportunity to limit the overhead at the price of a less accurate estimation. On the contrary, the *Aggregation* method does not provide this flexibility and trying to obtain a rough idea of the system while keeping the overhead low could lead to really bad results.

5 Comparisons and tradeoffs

With respect to tradeoffs, we observed that *Sample&Collide* provides the most flexible algorithm and offers a wide range of possible configurations depending on the accuracy of the estimate and associated overhead. For example, a really accurate result might not always be important and could furthermore uselessly waste resources. In this case, *Sample&Collide* with a smaller l parameter ($l = 10$ for example) could be a cheap solution for a relatively good result, as depicted on Figure 16 where on average only 100,000 messages are used for an estimation. Our experiments show that for *HopsSampling*, using a lower $minHopsReporting$ parameter does not significantly reduce the overhead, while degrading accuracy. *HopsSampling*, to a lower extent also provides some flexibility. However *Aggregation* does not, the only option is a high precision associated to a fairly large overhead. As detailed in [13], *Sample&Collide* could as well produced really precise results by increasing l parameter, and hence could compete with aggregation. A strength of this algorithm is thus to adapt to the application performance needs by simply modifying one parameter.

With respect to dynamic networks, we observed that *Sample&Collide* provides reliable results despite the degradation of the overlay connectivity. On the other hand, gossip-based algorithms need more time to complete a run due to the increase of the epidemic propagation time. The HopsSampling algorithm does not suffer from the same problem as it sends more messages each round, hence increasing the propagation efficiency. *Aggregation* and *HopsSampling* imply that each node in the overlay participates to the current counting process, which systematically entails resources consumption on each node at each process round.

In terms of accuracy, *Aggregation* outperforms the other algorithms and provides in a reasonable number of rounds an almost exact estimate. This approach should be used for applications with the most stringent needs. Another interesting characteristic of the *Aggregation* is that eventually the size estimation is available at each node of the network, as every overlay node possesses the local information to compute the estimation. There is no need for a broadcast of the size estimation as in the two other algorithms where only the initiator gets to compute the size estimation.

HopsSampling has the drawback of creating a message flood towards the initiator during the *HopCount* collection process; this produces a sensitive point in the network, and may overload the initiator's neighbors. *Sample&Collide*, which provides less overhead and more accurate results, might be preferred by the user. However, *HopsSampling* probably outperforms the other algorithms in terms of delay, which we haven't measured in this comparison due to the fact that physical network topology was not modelled in our simulator. A gossip based broadcast and an immediate ACK response from the nodes that succeeds the probabilistic test is very likely to be much shorter than the 50 rounds of *Aggregation* or the 200 equivalent samples of *Sample&Collide*.

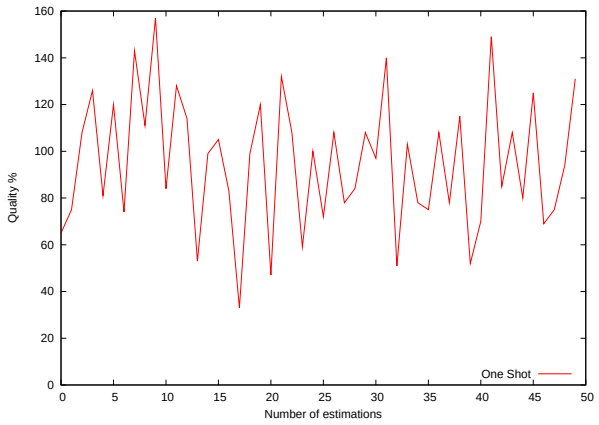


Figure 16: Sample&Collide with l=10, 100,000 node network

6 Conclusion

Monitoring a peer to peer system is a core functionality of distributed system and might be used in many applicative contexts. More specifically, the system size is very often needed to set the parameter values or monitor a system behaviour and this can be used for a wide spectrum of applications ranging from sensor networks to Grid computing.

In this paper, we provide the first comparison of some solutions computing the size of a large-scale system in a fully decentralized manner, in peer to peer unstructured overlays where peers have only a restricted knowledge of the system. We evaluated three different algorithms, representative of the probabilistic polling, random walk-based and gossip-based approaches and compared them along their accuracy, convergence, capability to handle network size changes and the associated overhead.

We observe that all three algorithms have passed the scalability and dynamicity tests but significant differences arised. Not surprisingly, as in many areas of computer science, there is a natural tradeoff between the quality of the estimate and the associated overhead, here bandwidth and computational resources. With respect to this tradeoff, *Sample&Collide* seems to be the most flexible algorithm.

As part of future work, the physical network modelling would be an interesting goal and might provide new insights on the comparison.

References

[1] M. Bawa, H. Garcia-Molina, A. Gionis and R. Motwani. Estimating aggregates on a peer-to-peer network. *Technical Report, Dept. of computer science, Stanford University*, 2003.

[2] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer overlay networks. *OSDI 02*, December 2002.

[3] P. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Transaction on Computer Systems, 21(4)*, November 2003.

[4] T. Friedman and D. Towsley. Multicast session membership size estimation. *IEEE INFOCOMM '99*, 1999.

[5] A. J. Ganesh, A.M. Kermarrec and L. Massoulié Peer-to-peer membership management for gossip-based protocols. *IEEE transactions on computers, vol.52, no.2, February 2003*, 2003.

[6] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations. *Middleware 2004*, October 2004.

[7] M. Jelasity and A. Montresor. Epidemic-style proactive aggregation in large overlay networks. *ICDCS 2004*, 2004.

[8] M. Jelasity, S. Voulgaris, R. Guerraoui, A-M. Kermarrec. Gossip-based peer sampling. *Submitted.*

[9] D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers. Decentralized schemes for size estimation in large and dynamic groups. *IEEE NCA '05*, 2005.

[10] D. Malkhi, M. Naor and D. Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. *PODC 2002*, July 2002.

[11] G. S. Manku. Routing networks for distributed hash tables. *PODC 2003*, June 2003.

[12] G. S. Manku, M. Bawa and P. Raghavan, Symphony: Distributed Hashing in a Small World *USITS 2003*, 2003.

[13] L. Massoulié, A.-M. Kermarrec, E. Le Merrer and A.J. Ganesh. Peer counting and sampling in overlay networks: random walk methods. *Technical report MSR-TR-2005-156*, 2005.

[14] Dimitrios Psaltoulis. *Private communication.*, July 2005.

[15] D. Psaltoulis, D. Kostoulas, I. Gupta, K. Birman, and A. Demers. Practical algorithms for size estimation in large and dynamic groups. *PODC 2004*, 2004.

[16] R. van Renesse, K. Birman and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems (TOCS), Vol. 21 pp. 164-206*, May 2003.

[17] S. Voulgaris, D. Gavidia, M. van Steen. CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays. *Journal of Network and Systems Management, Vol. 13, No. 2*, June 2005.



Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399