Behaviour of the fast consistency algorithm in the set of replicas with multiple zones with high demand

Jesús Acosta-Elias, Leandro Navarro-Moldes

Polytechnic University of Catalonia, Spain
{jacosta, leandro}@ac.upc.es

Abstract. In this work we have investigated the behaviour of the fast consistency algorithm for the propagation of changes in a set of replicas with multiple zones with high read demand. We demonstrate that the algorithm does not perform well in regions with a more than one valley of high demand surrounded by lower demand. The situation is characterized and a potential solution to change the distribution of demand equivalent to a single valley of demand where this algorithm provides high performance.

Keywords: Weak consistency, consistency algorithms, replication, distributed system.

1 Introduction

There is a growing interest on Internet scale distributed systems where many potential clients may contact a single host to request a given service at almost the same time from several locations. The presence of replica servers may help to improve the situation because clients will be able to contact the nearest replica. A Replica is a host who provides exactly the same services as the principal host. In this paper we will use the terms server and replica in the same sense.

Content replication between servers in a distributed system is justified by the need to reduce delay, to provide availability and to be scalable [13], to tolerate failure in the links, and also to withstand segmentation. The algorithms currently available for replica updating can be broadly classified into two groups, according to their consistency:

- Strong consistency, and
- Weak consistency

Strong consistency algorithms are costly, non-scalable on wide area networks; generate considerable latency and a great deal of traffic. They are suitable for systems with a small number of replicas, in which it must be

guaranteed that all the replicas are in a consistent state (i.e. all the replicas possess exactly the same content) before any transaction can be carried out (synchronous systems) [4, 16].

However, weak consistency algorithms [8, 15, 3] generate very little traffic, low latency, and are more scalable. They do not sacrifice either availability or reply time in order to guarantee strong consistency, but only need to ensure that the replicas eventually converge to a consistent state in a finite, but not bounded, period of time. They are very useful in systems where it is necessary for all the replicas to be totally consistent in order for transactions to be carried out (systems that withstand a certain degree of asynchrony). This is the case of Usenet news, or in computer-supported cooperative work systems.

With the weak consistency algorithm [8], each server (replica) from time to time chooses a neighbour to start an update session. In an update session two servers mutually exchange summary vectors then they exchange some data to mutually update their contents. At the end of the session both servers will have the same mutually consistent content. These are called anti-entropy sessions: It is called an anti-entropy session because in each session between replicas, the total entropy in the system is reduced. In this paper it will be referred to simply as a "session". The metric principle to be employed is how many sessions are necessary for a change brought about in a replica to be propagated to all the others.

Golding [8] demonstrated that the neighbouring server's random choice has the best performance (fewest number of sessions) for maintaining the consistency of the replicas in a peer-to-peer network. This gives rise to the fact that all the replicas are updated, no matter how much demand (number of requests per unit of time) they have, in such a way that a replica with low demand can be updated first, before another with much greater demand.

In most distributed applications, some replicas tend to have more demand than others due to different factors, such as:

- Geographical distribution,
- Number of clients,
- Number of requests arising from more intense work among clients.

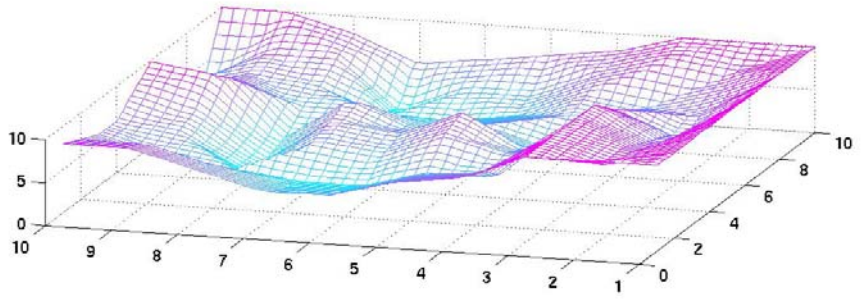

Fig. 1. Example of demand in a large distributed system

Thus if we draw a graph of the distribution of the replicas in the X-Y plane, and their demand along the Z axis, we will have an image of hills and valleys in which the valleys, in our case, are the areas of greater demand and the replicas with least demand are on the hills (see Fig. 1)

For this reasons, in [1] we propose the "Fast consistency algorithm", which prioritizes replicas with most demand.

2 Behaviour of the fast consistency algorithm in one zone with high demand

In order to investigate the behaviour of the fast consistency algorithm in one zone with high demand, a simulator based on NS[14] has been built. The topology chosen is a ring. It is a simple topology where every node has exactly two neighbours. This avoids the boundary effects of the edges of linear networks (nodes with different number of links on the edges than on the rest of nodes), and all nodes see the same diameter of the network. Our fast consistency algorithm has been tested in [1] using a generator of random representative Internet topologies [11, 12, 7], but in this paper, the simpler ring topology is preferred, since otherwise it would be extremely difficult to view and interpret the results.

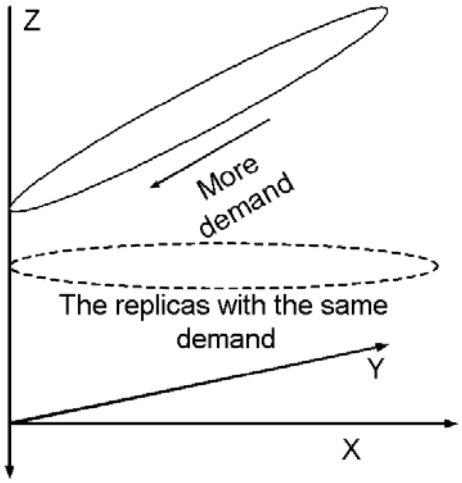

Fig. 2. Simulated situations

The simulation is performed on a network of 100 connected replicas in a ring topology. Two different cases are simulated: one in which all the replicas have the same demand, and the other in which the replicas in the region of the ring have more demand than those in the opposite region (Ring with

continuous line in fig. 2). In this way we have two zones outlined according to demand. One zone simulates a valley (that of greater demand), and the other a mountain (lesser demand). Thus we obtain the appropriate conditions with which to investigate the behaviour of the fast consistency algorithm in a zone with a mountain and in another zone consisting of valley and mountain. The first case, where all the replicas have the same demand (Ring with dotted line in fig. 2), is for the purposes of comparison. 5000 simulations (This number has been selected experimentally as a good compromise between random bias and heavy simulations time) have been carried out for each case, and the mean results used to draw the graphs.

3. Results

Figures 3 and 4 have been drawn up using the results obtained from the simulations. In Fig. 3 the results are shown in the shape of a ring (non-polar). This shape is precisely the simulated physical topology, and therefore the behaviour of the fast consistency algorithm can be viewed clearly. One may see how the number of sessions necessary for each replica to reach a consistent state changes. The nodes are uniformly distributed in the ring, the sessions are in the radius of the circle, and in the centre is the zero sessions point.

In Figure 4, only the way of showing the results has been altered, the ring is opened up at one end and becomes a line. The nodes are on the X axis, and on the Y axis we see the sessions necessary for each node to receive the changes occasioned in each one of the remaining nodes.

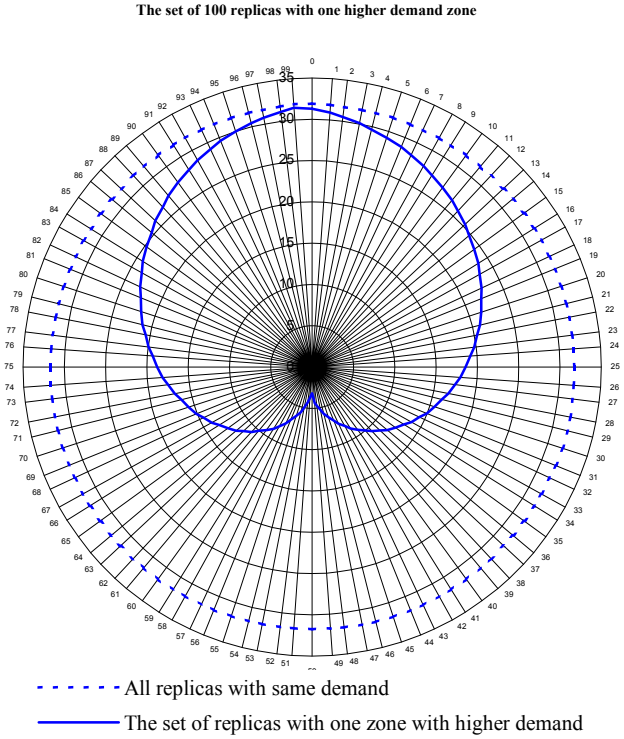

Fig. 3. Average number of sessions necessary for each change produced in each one of the replicas to reach the other replicas.

The dotted line represents the mean number of sessions necessary for each change produced in each replica to reach all the other replicas. This is the case in which all the replicas have the same demand.

The continuous line represents the mean number of sessions necessary for each replica to receive a change brought about in each one of the other replicas. This is the case in which there is a zone of greater demand and an opposite zone of less demand.

When the replicas all have the same demand, the average sessions are the same for all replicas; for 100 replicas in the ring topology, the average obtained was 31.69 (dotted line). On the other hand, where there is a zone of greater demand and one of lesser demand, for sessions are required for the update to arrive at the replicas with more demand, and this tendency is gradual. The replica with more demand is the one that receives the updates fastest. Updates take longer and longer to reach replicas as their demand decreases, until arrival time is the same for those replicas having the same demand. The worst behaviour of the fast consistency algorithm is similar to

the normal behaviour of the weak consistency algorithm, and this performance is obtained at the peak of the mountain.

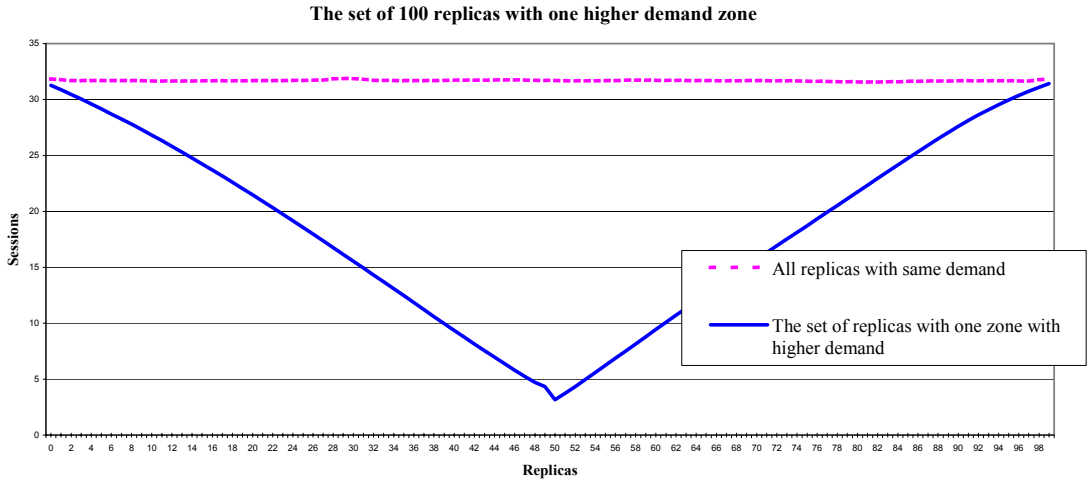

Fig. 4. Mean number of sessions required for each replica to receive changes

The results obtained from the simulations can be summarized by stating that the updates drop towards the bottom of the valley with increasing linear velocity, and rise to the zones of less demand with decreasing linear velocity.

4 Simulation in a set of replicas with two high demand zones

To find out the behaviour of the fast consistency algorithm in the presence of multiple high demand zones separated by zones of low demand, we use the same NS[14] based simulator. To avoid the effects of the ends on the linear networks, the topology chosen is once again a ring with 100 replicas.
Demand on the replicas is distributed in such a way that two areas of the ring having greater demand separated by an area of low demand. Thus we have three zones outlined according to demand. Two zones simulate valleys (greater demand), and the other zone simulates a mountain (lesser demand). In these conditions we are able to investigate the behaviour of the fast consistency algorithm in a zone with two valleys separated by a region with low demand (mountain).

The simulator is fed with this topology, and to each replica is assigned a change, which in the process of consistency must be propagated to all the replicas in the group. 5000 simulations are carried out.

5. Results in the two high demand zones

Figures 5 and 6 are drawn using the mean results of the simulation. In Figure 5 the results are shown in the form of a ring (non polar). In this figure one sees the number of sessions necessary for each replica to reach the consistent state; that is, for each replica to receive all the changes produced in each replica in the group. The nodes are uniformly distributed in the ring, the sessions in the radius of the circle, and in the centre is the zero sessions point.

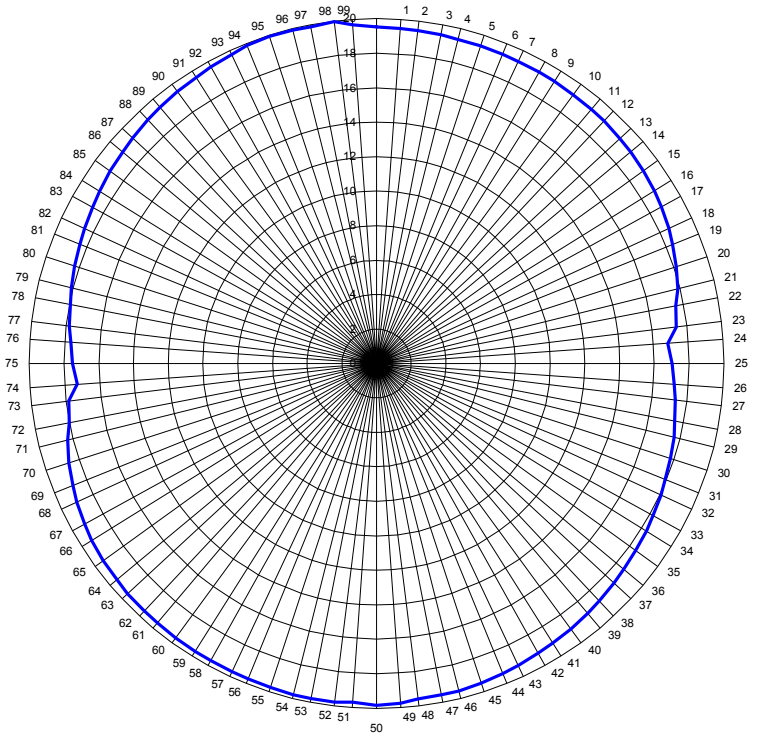

Fig 5. Set of replica with two demand zones

In Figure 6, only the way the results are shown has been changed; the ring is opened up at one end and becomes a line. The nodes are on the X axis, and on the Y axis we find the sessions necessary for each node to receive the changes produced in each one of the other nodes.

The continuous line represents the mean number of sessions necessary for each replica to receive a change occasioned in each of the other replicas.

As one can observe in Figures 5 and 6, the replicas in the high demand zones take on average 16.79 sessions to reach a consistent state. This number is much greater than that obtained in the case of a high demand zone (Fig. 4),

where the average is only 3.18 sessions. In other words, the performance of the fast consistency algorithm is much less in multiple zones with high demand. In the following section we seek to explain what occurs in these cases with the fast consistency algorithm.

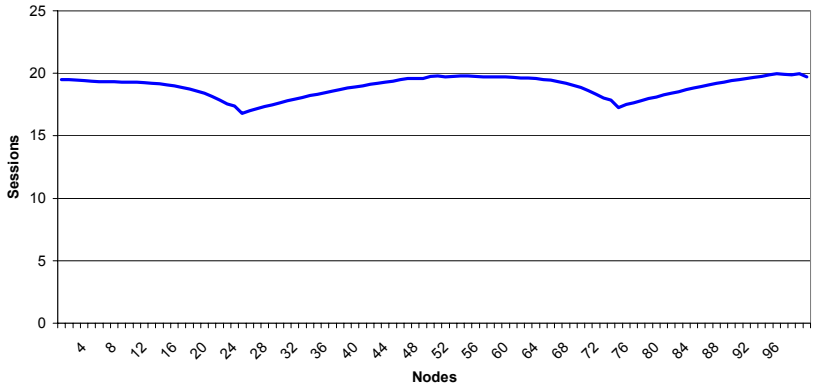

Fig. 6. Number of sessions required for replicas with two higher demand zones

The fast consistency algorithm causes the updates to move at a greater speed to the areas of greater request; in other words, the zones of high demand are also the zones with the most consistent replicas. This is the desired effect. However, it also causes the zones of little or no demand to have less consistent content, as can be seen in Zone II of Fig. 7.

The existence of these zones (II), whose content is not so up-to-date, is also the desired effect of the fast consistency algorithm, since the zones of low demand are updated at the normal speed of the weak consistency algorithm.

The existence of low demand regions, such as that we can see in (II), and in which the updates arrive at a relatively low speed, gives rise to areas that the fast consistency algorithm finds difficult to overcome. It also causes the high demand regions (I) to be isolated one from the other, and therefore their content undergoes delay before being mutually consistent.

These highly consistent regions surrounded by low consistency zones appear as islands formed by locally consistent replicas.

A change produced in A, for example, quickly reaches B, but rises until C at a relatively slow speed, then falls rapidly to D. Ideally, the change produced in A should reach D just as rapidly as B, but in fact this is not the case since Zone II stands in the way.

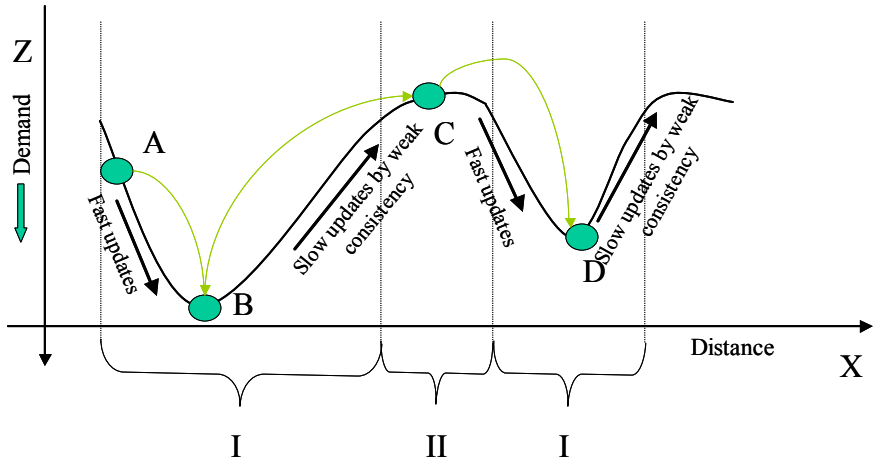

Fig. 7. Shows a cross section of the distribution of nodes. The demand is on the Z axis, and on the X axis we have the distribution in the X-Y plane, where Y = k. The nodes are shown in such a way that those with most demand are towards the bottom, and those with least demand are found towards the top. Two different kinds of regions can be seen, valleys (I) and mountains (II).

6 Related work

There are some proposals that set forth the necessity for the replicas to be adapted to customer requirements, so in this section we briefly describe some of the work dealing with this area of study.

Fluid Replication [5] is a reactive mechanism whose purpose is the automatic creation of replicas whenever and wherever they are necessary. In fluid replication, clients monitor their observed performance in interacting with the service. When performance becomes poor through increased network load or client mobility, a replica is created to reduce the dependence upon the poor network path.

Sudha [9] proposes an algorithm, based on the historic performance of a set of replicas, which is capable of selecting deterministically replicas having all the service quality requirements a customer might request.

Richard Lenz [10] proposes ASPECT (Application Oriented Specification of Consistency Terms), which enables us to specify weak consistency requirements from the point of view of the application according to the "need to know" principle, which states that "data only have to be made available where they are needed and only as current and consistent as required by the applications that access these data". It proposes the use of two dimensions to

specify the replica consistency requirements, one spatial and one temporal; the spatial dimension describes the degree of consistency or quality of data of a particular copy. The temporal dimension specifies when this quality of data is required. As we may observe, this proposition is only a specification.

Petersen et al. in [15] describe the Bayou anti-entropy protocol, which facilitates the propagation of replicas by means of weak consistency. They pose, furthermore, the possibility of providing a variety of policies for where and when the updating may be propagated.

The question is also raised of furnishing four types of policy in the anti-entropy protocol: Policies for when to reconcile; policies for selecting with which replicas to reconcile, policies for deciding how aggressively to truncate the write-log, and policies for selecting a server from which to create new replicas.

- Policies for when to carry out reconciliation: Potential policies could be periodic reconciliation, manually initiated reconciliation, or those initiated by the system.

- Policies for selecting with which server the anti-entropy session can be started may depend on many factors: what other replicas are attainable, for example; characteristics of the network connections using these replicas.

- Policies for selecting how aggressively to truncate the write-log enable us to negotiate storage and network resources necessary in an anti-entropy session. Truncating the write-log very aggressively can give rise to very long anti-entropy sessions among some servers due to the need to transfer complete databases.

- When various servers are available for creating a new replica, quantities to be considered must be identified, in addition to other characteristics of these replicas, how out of time they are, band width of connections, and how complete their write-logs are.

This proposition enumerates some interesting policies. However, it is no more than this an outline of possible policies, none of which have been researched in detail, neither from the point of view of implantation nor performance.

Brun-Cotan and Makpangou [6]: This article presents an architecture and a run-time for "adaptable replicated objects". Different applications require different contracts concerning the consistency of replicas, or a different negotiation between performance and consistency. The architecture proposed structures a replicated object into three component classes: access objects, replicas, and consistency managers. Together with the access object, the consistency manager are the components responsible for maintaining consistency in each object, depending on application needs; that is to say, the application for a particular activity may require strong consistency, and in other cases weak consistency.

This architecture is conceived for the design of distributed applications, and its advantage is the existence of a contract that specifies the consistency of

the objects replicated, in which the degree of required consistency is defined, whether it be strong or weak.

7 Conclusions and Future Work

The fast consistency algorithm performs very well in replica groups in which there is only one high demand zone. On the other hand, when the replica groups have multiple high demand zones separated by low demand zones, the updates take fewer sessions to reach the replicas with most demand. The improvement obtained, however, is quite poor when compared with the case in which there is only one high demand region. The low demand replicas act like barriers, preventing the fast consistency algorithm from functioning effectively. This demonstrates that the algorithm itself is not generalizable to regions with multiple high demand zones, since the speed with which the updates reach the high demand replicas is not substantially improved. In the high demand zones, however, local changes within the zone, if they move at high speed, form these "islands" of replicas with highly updated content at a local level.

In order to make the best use of this algorithm, mechanisms enabling high demand zones to be interconnected are required, perhaps by creating a hierarchical type of logical topology [2]. Other approach perhaps is a leader election in each high demand zone and interconnect them. In both cases, multiple zones of high demand can become equivalent to one zone only. Work under way is evaluating the behaviour and performance of these approaches.

8 References

1. Jesús Acosta Elias, Leandro Navarro Moldes, "A Demand Based Algorithm for Rapid Updating of Replicas ", IEEE Workshop on Resource Sharing in Massively Distributed Systems(RESH'02), July 2002.
2. N. Adly, M. Nagi and J. Bacon. 'A Hierarchical Asynchronous Replication Protocol for Large Scale Systems'. Proceedings of the IEEE Workshop on Parallel and Distributed Systems, Princeton, New Jersey, October 1993, pp.152-157.
3. A. Adya, "Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions", PhD thesis Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, March 1999.
4. K. P. Birman, "The process group approach to reliable distributed computing", Communications of ACM, Decembre 1993/Vol. 36, No. 12
5. Cox, L. P. and Noble, B. D. 2001."Fast Reconciliations in Fluid Replications", Int. Conf. On Dist. Comp. Syst. (ICDCS) (April 2001). http://mobility.eecs.umich.edu/papers/icdcs01.pdf
6. G. Brun-Cota, M. Makpangou, "Adaptable Replicated Objects in Distributed Enviroments", Research Report 2593, INRIA, May 1995.

7. M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-Law Relationships of the Internet Topology", ACM SIGCOMM, Cambridge, MA, September 1999
8. R. A. Golding, "Weak-Consistency Group Communication and Membership", PhD thesis, University of California, Santa Cruz, Computer and Information Sciences Technical Report UCSC-CRL-92-52, December 1992.
9. S. Krishnamurtyhy, W. Sanders, and M. Cukier, "A Dynamic Replica Selection Algorithm for Tolerating Timing Faults", In Proc. Of the The International Conference on Dependable Systems and Networks, pages 107-116, July 2001.
10. R. Lenz, "Adaptive distributed data management with weak consistent replicated data", ACM Symposium on Applied Computing, February 1996
11. A. Medina, I. Matta, and J. Byers, "On the Origin of Power Laws in Internet Topologies", ACM Computer Communication Review, pages 160-163, April 2000.
12. A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Universal Topology Generation from a User's Perspective",
13. B. C. Neuman, "Scale in Distributed Systems. In Readings in Distributed Computing Systems", IEEE Computer Society Press, 1994
14. The Network Simulator: http://www.isi.edu/nsnam/ns/
15. K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and Demers, "Flexible Update Propagation for Weakly Consistent Replication", Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP-16), Saint Malo, France, October 5-8, 1997, pages 288-301.
16. V. Duvvuri, P. Shenoy and R. Tewari, "Adaptative Leases: A Strong Consistency Mechanism for the World Wide Web", IEEE INFOCOM 2000, pages 834-843.