Improving Availability with Adaptive Roaming Replicas in Presence of Determined DoS Attacks

Chin-Tser Huang, Prasanth Kalakota, Alexander B. Alexandrov

Department of Computer Science and Engineering
University of South Carolina
{huangct, kalakota, alexand2}@cse.sc.edu

***Abstract*— Static replicas have been proven useful in providing fault tolerance and load balancing, but they may not provide enough assurance on the continuous availability of mission-critical data in face of a determined denial-of-service (DoS) attacker. A roaming replica scheme can provide higher availability assurance, but the overhead associated with replica movement and lookup is high. In this paper, we propose ARRP, an adaptive roaming replication protocol in which static replicas are used normally but if a certain percentage of static replicas has already been shut down, then a small number of roaming replicas will be added and stored in randomly selected hosts that are changed periodically. In particular, we analyze the appropriate threshold when the roaming replica scheme should be enabled by empirically investigating the tradeoff between availability, performance, and overhead. Simulation results show that ARRP can effectively mitigate the impacts of DoS attacks and host failures to ensure continuous availability of critical data, with better performance and reasonable overhead compared to only using static replicas.**

***Index Terms*—Data Replication, Assurance, Availability, Denial-of-Service Attacks.**

I. Introduction

Data replication on the Internet is becoming more and more common. By placing replicas at multiple locations people can access the data more quickly and reliably. By replicating data single point of failure can be avoided, as this is a particularly desirable property for mission-critical applications.

Consider a military operation scenario where the soldiers are the clients, base station serves as the server with battlefield information being the critical data exchanged between them. Assume that the soldiers periodically contact the base station and get critical data. If the base station is shut down by a DoS attack or due to some other problem, the soldiers can no longer access this data. However, if the same data were replicated at different places, then the soldiers can access the critical data via the replicas.

There are mainly three objectives with such a data replication strategy. First, as explained in the previous example, if the main server is down due to host failures or attacks, the clients can still have access to the critical data from the replicas. Second, replicas may provide faster access to clients. Third, by allowing clients to access different replicas, Load balancing can be achieved.

One of the common ways to replicate data is by using static copies. In this approach the server designates some nodes as replicas and transfers data to them. To maintain the consistency of replicated data, the server periodically updates the replicas. This approach achieves the second and third objectives. One problem with static replicas is that if an adversary locates all the static replicas, they can launch targeted DoS attacks as discussed in [9] to shut down these replicas. For greater assurance more replicas can be stored in the network, but this may incur too much overhead as all replicas have to be updated when the original copy is updated.

To mitigate this problem, a roaming replica method is proposed in our previous paper [4]. In this approach, instead of using a large number of static replicas, only a small number of roaming replicas are used. The server periodically moves the replicas to different hosts, and the clients use a discovery protocol to find the location of replicas in a secure manner. It is demonstrated in [4] that by moving data periodically greater assurance on data availability can be achieved.

The main problem with the roaming replica approach is the necessity to move replicas periodically and to discover the current location of a replica. Even during periods when there are no attacks or attack level is low, data still needs to be moved periodically. Clients have to run the discovery protocol every time to find the location of replicas. In this paper, we propose an Adaptive Roaming Replication Protocol (ARRP) which aims to integrate both schemes of static replicas and roaming replicas. In this approach roaming replicas are included in addition to static replicas only if the attack level is above the threshold, otherwise only static replicas are used. In this way the survivability of replicas is increased and the overhead of roaming replicas is minimized. The main question that remains to be determined is: what is the *most appropriate* threshold needed to enable the roaming replica scheme.

In an attempt to answer this question, we analyze the tradeoff relationship between *Availability*, *Performance*, and *Overhead* in the presence of attacks. Availability is a measure of how survivable the replicas are in face of DoS attacks. If either the server of at least one of the replicas is available, then the critical data is available. Performance is a measure of how long it would take for a client to get the critical data from the server or an available replica. If more replicas are deployed, there is a better chance that a client can quickly find a replica close to it, thereby improving the performance. On the other hand, in the presence of DoS attacks, as it takes more time to find an available replica performance decreases. Overhead is composed of four components, namely movement overhead, discovery overhead, storage overhead, and update overhead. Movement overhead is about message transmissions needed to

move roaming replicas to a different location. Discovery overhead is about message transmissions needed to find an available replica. Storage and update overheads are proportional to the number of replicas used in the system. In the simulation and evaluation, we create a random network topology with 300 nodes and try to find the performance and overhead for various client loads in presence of various degrees of replica failures.

The remainder of this paper is organized as follows. In Section II, we discuss the related work. In Sections III and IV, roaming data redundancy scheme and ARRP scheme are discussed. In Section V we describe the experimental setup and simulation and discuss the evaluation on the tradeoff between availability, performance, and overhead. Finally, we conclude and discuss future work in Section VI.

II. Related Works

Most of the work related to replica placement focuses on providing Quality of Service (QoS) to clients. A method using Tapestry method was proposed by Chen et al. [3]. In this work the network is treated as a dissemination tree and the method addresses the placement of replicas in order to achieve load balancing. In [1], Bartolini et al. propose a method for placing replicas based on the traffic estimations and current replica location. In [11], Szymaniak et al. propose HotZone, an algorithm to place replicas in wide-area network such that the client to replica latency is minimized. This algorithm work very well in terms of performance, but they do not address the issue about the availability of critical data.

In [12], Tang and Xu discuss the problem of placing replicas of an object in content distributed systems to meet QoS requirements while minimizing the replicating cost. The authors used replica-aware model and replica-blind model for different problem specifications. In the replica-aware model the server knows the location of the replicas and the user and redirects the user request to the replica which is close to the user, which is NP-complete problem. In the replica-blind model, the server randomly selects one replica and sends the user request to that replica, which can be solved in polynomial time. Still, this work does not address replica availability.

In [5], Khattab et al. propose a method in which *n* out of *m* servers are selected to be active servers, rendering the remaining *m* - *n* servers acting as honeypots. To mitigate DoS attacks a different set of *n* servers are randomly selected to be active after some time. In [6], the same authors propose a method in which the server roams among a pool of servers. This method requires modifications to TCP connection state as servers are moved physically. Only legitimate clients can follow the location of the server. Although this approach achieves fault tolerance and increases availability, the overhead of moving servers can be high. Their results show an increase of about 14% in average response time when there are no attacks. In contrast, our approach adds a small number of roaming replicas only when replica failure rate is high.

In [9], Srivatsa and Liu discuss the targeted file attacks and propose a LocationGuard scheme to counter the attacks. In this approach, each client uses a lookup guard which takes a stored location key to securely calculate the location of the target file or its replica in an overlay network. The adversary is hidden from the target location because it does not know the corresponding location key. However, this scheme may not tolerate a determined attacker who launches multiple DoS attacks by guessing and gradually taking down more replicas.

As for the discovery of available replicas, there have been several distributed hash table (DHT) based replica lookup protocols, e.g. Chord [10], CAN [7], Pastry [8], and Tapestry [3]. These schemes allow for lookup in a small and bounded number of hops. However, in presence of determined DoS attacks, these schemes will also require a lot of retries.

III. Roaming Data Redundancy

The Roaming Data Redundancy Scheme was proposed in our previous paper [4]. This scheme basically consists of two protocols, a Redundant Data Moving Protocol (RDMP) and a Redundant Data Discovery Protocol (RDDP). RDMP allows host to move replicas of the critical data periodically to different hosts. RDDP allows roaming replicas of critical data to be discovered by clients. Both protocols are designed to incorporate multiple types of critical data, with each type of critical data maintained by a different host.

A. Assumptions

Before presenting the scheme, we discuss the assumptions that we make about the critical data service and the adversary. We assume that there are multiple types of critical data present in the network. All legitimate clients are aware of which host is the main server of which type of critical data. To protect the privacy and integrity of critical data and location of redundant copies we assume all the communication is encrypted. For broadcast messages we assume that the messages are encrypted with the shared key between the main server and client hosts, whereas all the unicast messages are encrypted using public key encryption. And also we assume that all the hosts in the network can be trusted. They do not collude with adversary by leaking the private keys.

Even though adversary cannot decrypt the messages, we assume that the adversary can do traffic analysis and can also perform replay attacks. We also assume that the adversary is aware of the location of all the hosts in the network. The adversary can also attack the *m* hosts simultaneously and can shut them down all at once.

B. Redundant Data Moving Protocol

Redundant Data Moving Protocol consists of *n* processes *rdm*[0..*n*-1]. Each host participating in the protocol has an input *cd*, which represents the critical data maintained by the host, which is the owner of the critical data of the host. The owner has the authority to manage the roaming replicas of the data. Each *rdm*[*i*] also maintains an array *rd*[0..*n*-1] which represents the replicas of the other hosts' critical data currently kept by this host. Each host also maintains an array *sq*[0..*n*-1] that represents the next sequence number to be used by each process to send the next request message to move the critical data. Periodically *rdm*[*i*] selects the next keeper of its critical data, broadcasts the *dlt* message, to notify the keeper to delete the outdated message and sends unicast message to transfer the critical data to the next keeper. If process *rdm*[*j*] keeps a roaming replica, then *rdm*[*j*] sends a *dltack* message to *rdm*[*i*]

to acknowledge the deletion. If *rdm*[*j*] is the next keeper of the roaming replica then *rdm*[*j*] sends a *movack* message to acknowledge the reception of the critical data.

C. Redundant Data Discovery Protocol

The Redundant Data Discovery Protocol consists of n processes rdd[0..n-1]. Each process rdd[i] maintains an input array rd[0..n-1] which is provided by rdm[i] in the redundant data moving protocol and represents the replicas of the other hosts' critical data currently kept by this host. Each process rdd[i] also maintains an array sq[0..n-1] that represents the next sequence number to be used by each process to send the next query. Each process *rdd*[*i*] in the RDDP can send to every other process a *drqst*(*sq*[*i*], *tgt*, *i*) request message, where *sq*[*i*] is the sequence number of the *drqst* message sent by *rdd*[*i*], *tgt* is the index of the target critical data and *i* is the index of *rdd*[*i*]. Every time *rdd*[*i*] sends out a *drqst* message, *sq*[*i*] needs to be incremented by 1 in every process in order to keep consistency. If process *rdd*[*j*] currently keeps a roaming replica, then *rdd*[*j*] will send a *drply*(*sq*[*i*], *tgt*, *j*) message to *rdd*[*i*], where *sq*[*i*] is the corresponding sequence number of *rdd*[i], *tgt* is the index of the target critical data, and *j* is the index of *rdd*[*j*]. The other processes that do not keep track of the critical data will discard the message. Figure 1 illustrates the basic operations in RDMP and RDDP.

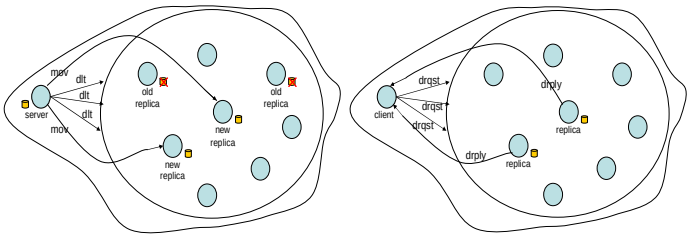

Figure 1: Basic operations of Redundant Data Moving Protocol (RDMP) and Redundant Data Discovery Protocol (RDDP).

IV. Adaptive Roaming Replication Protocol

It has been shown in [4] that the roaming replica scheme effectively mitigates the impacts of DoS attacks and host failures and provides higher assurance on the continuous availability of critical data. However, the main disadvantage of the pure roaming replica scheme is about its overheads due to the movement and discovery of roaming replicas. These overheads remain even when the level of DoS attack is very low, and as a result, they may cancel the benefit of availability guarantee and discourage the adoption of this scheme.

To address this problem, we propose the Adaptive Roaming Replication Protocol that integrates both schemes of static and roaming replicas. The static replica scheme is still used at all times. Each client caches the addresses of a few static replicas that are close to it, such that the client can access the closest available replica first and shorten the latency. If none of the cached addresses of static replicas is reachable, then the client will use the RDDP protocol to find an available replica, either static or roaming. The server is able to derive an estimate of the percentage of failed static replicas when it periodically updates the replicas. When the percentage of failed replicas exceeds a certain threshold *th*, the roaming replica scheme is enabled to add a small number of roaming replicas to the network. Since there are still some available static replicas in the network, the client can attempt to access the cached static replica positions to see if they are still available. If so, then the client will access the data from the available static replica. Only when these attempts fail the client will resort to RDDP protocol to find a roaming replica. Later if some failed replicas recover and the server detects that the percentage of failed replicas falls below *th*, it turns off the roaming replica scheme. This is easily achieved by requesting the current holders of a roaming replica to remove it from their storage without designating the next roaming replica holders.

V. Experimental Setup and Results

A. Simulation Model

We have developed a simplified model of our roaming replica scheme in C++ and conducted a number of experiments to study the effect of different parameters. We first created 300 node network topology using BRITE [13]. To evaluate the effectiveness of our simulation we tested our results on different topological networks sparse, medium and dense networks whose nodes have average degrees of 2, 5 and 8 respectively. We assume that these three different topologies cover various network densities. In addition we assume that all links in the network are 10Mbps unless specified otherwise.

Initially, the server selects 20 nodes as static replicas and transfers critical data to them. In each time unit clients try to contact their closest available replica and gets critical data. Replicas can go down due to the presence of DoS attacks and node failures. To implement the node failures and DoS attacks on replicas, we used a probabilistic method in which a random number is generated to determine whether a replica is up or down. We ran the simulations under replica failure probabilities of 25%, 50% and 75% respectively.

In addition to the static replicas we used 3 replicas as roaming replicas. In each time period the server selects 3 new replicas and uses RDMP protocol to send critical data to them. Even if the attacker is able to find a subset of the roaming replicas, the attack is successful only during the time period because after each time period the roaming replicas change their location.

We assume that the clients cache the location of some closest static replicas. The client first checks the cached locations before applying RDDP protocol to find an available replica. In the simulation we assume that the client caches 1, 2 or 3 closest static replicas respectively and the results are compared with the basic static replica placement method where a legitimate client knows the location of all static replicas (e.g. by looking up some public directory).

We ran the simulation for 100 time units with server updating the replica position for every time unit. The simulation is run for 20 times and each point in the graph represents the average over 20 runs. All the client requests are distributed uniformly throughout the time period.

B. Evaluation

We use the above model to conduct various simulations in order to evaluate the availability, performance, and overhead

of the ARRP scheme. In particular, we analyze the tradeoff between the three aspects.

1) Availability

We first analyze how survivable the roaming replicas are against a determined attacker. In theory, if the number of hosts in the service network is *n*, with *r* roaming replicas among the *n* hosts, and the attacker is able to launch *an* attack in parallel, then the probability that the attacker hits all the *r* roaming replicas at the same time is given by

$$\binom{n-r}{a-r} \Big/ \binom{n}{a}$$

With the parameters used in our simulation (300 nodes, 3 roaming replicas, and 30 attacks at the same time), the probability is just 0.09%, which is very low.

In order to quantify the availability more concretely, we design the following experiment. In the roaming replica scheme, every time unit we let the original source server randomly choose 1, 2, 3, or 4 servers out of 300 total servers to keep the roaming replicas. Every time unit the adversary randomly chooses 30 servers to attack simultaneously. If all the current roaming redundant copies are hit by the DoS attacks, then the attack is regarded successful and we measure the time elapsed in time units. Otherwise, in the next time unit the original source server again randomly chooses 1, 2, or 3 servers and the adversary again randomly chooses 30 servers to attack. The longer the elapsed time before the attack succeeds the better the availability is, since the network proves to be more survivable to the attack. The results are compared with the static replica scheme in which 10, 20, and 30 static replicas are stored in a total of 300 servers. At the beginning of the simulation the source chooses 10, 20, or 30 servers to keep the redundant copies and the attacker randomly selects 30 servers to attack. If the adversary hits a server that keeps a redundant copy, the adversary shuts it down. The adversary uses its remaining attacks to keep attacking until it locates and shuts down all the servers that keep a redundant copy, and we measure the time elapsed in time units.

Figure 2 shows the statistics of 1000 runs for our roaming data redundancy model – 1, 2, 3, and 4 roaming copies in 300 total servers under 30 attacks. As we increase the number of roaming copies, the time needed for the adversary to succeed increases exponentially. Therefore by increasing the number of roaming copies by just one, we can achieve exponential increase in the difficulty for the adversary.

Figure 3 shows the statistics for our comparison model – 10, 20, 30 static copies distributed in 300 total servers. Note that the number of simultaneous attacks is 30 so that the adversary is able to successfully shut down all the static replicas. While the increase of the number of static replicas increases the time necessary for a successful attack, the increase is smooth and the average time needed for a successful attack is apparently shorter than when 2 or 3 roaming copies are used.

From the figures it is clear that the ARRP scheme provides higher availability than using only static replicas, and Figure 3 shows that the benefits of using our approach increase when the number of roaming copies increases, as the average time needed for the attack to succeed increases by around 10 times

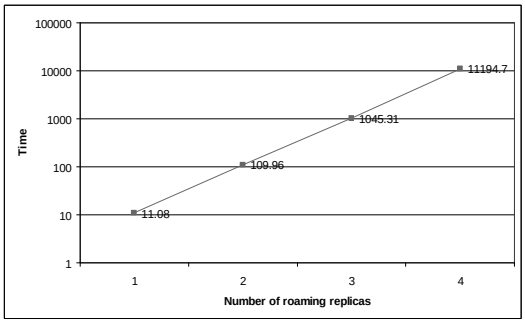

Figure 2: Time for successful DoS attacks, with 300 nodes, 30 simultaneous attacks, and 1, 2, 3, 4 roaming replicas respectively.

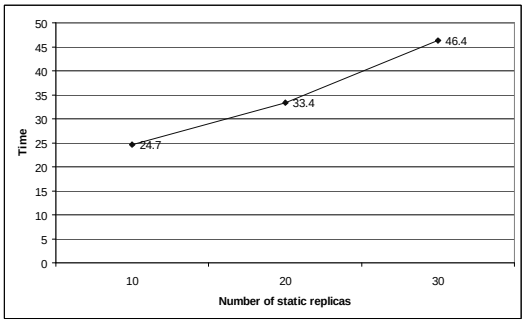

Figure 3: Time for successful DoS attacks, with 300 nodes, 30 simultaneous attacks, and 10, 20, 30 static replicas respectively.

with every additional roaming copy. The results also indicate that in this specific setup adding 3 roaming replicas to the service network can already achieve high availability; adding more replicas will just add to the overhead.

2) Performance

We evaluate the performance by calculating the average amount of time it takes to for clients to get data from replicas. Due to space limit we only show the simulation results for medium networks (with average node degree of 5). Figures 4 and 5 show the amount of time it takes to get data from replicas under different replica failure rates (25%, 50%, 75%) in case of medium (with 50 clients) and high loads (with 100 clients) respectively. The client requests are distributed uniformly over the time period.

In the base case in which there is no roaming replica and the clients know the location of all static replicas, the clients will check the availability of replicas one by one without applying RDDP protocol. However as the replica failure increases there is a lower chance to find an available replica. So it takes more time to find the closest available replicas. In the other three cases the clients cache 1, 2, and 3 closest static replica locations respectively. Using RDDP protocol it is possible to find the closest available replica fast by sending broadcast request, because it does not require a lot of retries when the replica failure rate is high.

From the figures we can see that as the replica failure increases, the average amount of time to get data also increases. Moreover, as the load increases the time taken to process the requests also increases, thereby each client request has to wait for longer time on average. This explains the increase in average amount of time to get data as the load increases.

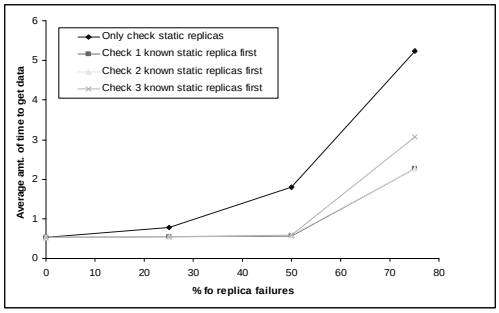

Figure 4: Performance against the percentage of replica failures for 50 clients in medium network.

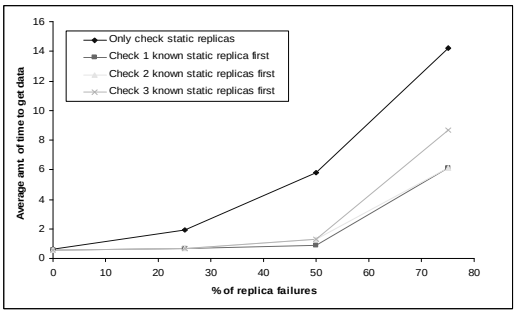

Figure 5: Performance against the percentage of replica failures for 100 clients in medium network.

3) Overhead

We want to determine how much overhead the proposed approach incurs when different threshold is used. As discussed in Section I, the overhead is composed of movement overhead, discovery overhead, storage overhead and update overhead. Among these four components, movement overhead, storage overhead and update overhead are proportional to the number of roaming replicas, and we have shown that only a very small number of roaming replicas are needed to achieve high availability. The discovery overhead, however, is dependent on how many accesses end up using RDDP to find an available replica. To estimate this value, we use the same simulation model to develop a random sequence of 2000 client accesses and experiment it with different percentage of static replica failures. The number of cached static replica addresses is assumed to be 3.

The results are shown in Figure 6. When the percentage of replica failure is below 65%, less than 10% of all accesses will use RDDP. This figure provides us two insights. First, the discovery overhead due to RDDP is small when the replica failure percentage is low. Second, the appropriate threshold should be higher than 65%, because when the replica failure percentage is less than 65% most clients can still find an available replica close to it without resorting to RDDP.

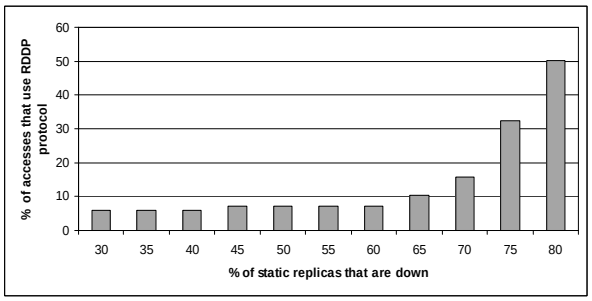

Figure 6: The percentage of total accesses that use RDDP protocol to find a replica under different percentage of static replica failure.

VI. Concluding Remarks

In this paper, we point out the need for greater assurance of the continuous availability of critical data services, and show that current solutions are not sufficient to provide the desired level of assurance under determined DoS attacks. We then introduce a novel adaptive roaming replica scheme called ARRP that aims to ensure constant availability of critical data by adding a small number of roaming replicas when the percentage of static replica failure is higher than a threshold. Simulation results show that ARRP can effectively mitigate the impacts of DoS attacks and host failures to ensure continuous availability of critical data, with better performance and reasonable overhead compared to only using static replicas.

In the future work, we will implement a prototype of ARRP and evaluate it with client access sequences recorded from a real service network and synthetic attack traffic data. Moreover, we will investigate how frequently the roaming replicas should be moved so that they can survive the attacks with less overhead. Furthermore, we will study the impacts that the topology of the network and the routing algorithm has on the overall performance and overhead of ARRP.

References

[1] N. Bartolini, F. L. Presti and C. Petrioli, “Dynamic Replica Placement and user request Redirection in Content Delivery Networks,” IEEE International Conference on Communications, ICC 2005.

[2] Y. Chen, A. Bargteil, D. Bindel, R. Katz, J. Kubiatowicz, “Quantifying Network Denial of Service: A Location Service Case Study,” Proceedings of Third International Conference on Information and Communications Security (ICICS 2001), November 2001.

[3] Y. Chen, R. H. Katz, J. D. Kubiatowicz, “Dynamic Replica Placement for Scalable Content Delivery,” Proceedings of First International Workshop on Peer-to-Peer Systems (IPTPS 2002), Cambridge, MA, March 2002.

[4] C.-T. Huang, A. B. Alexandrov, P. Kalakota, “Roaming Data Redundancy for Assurance in Critical Data Services,” Proceedings of 2006 High Availability and Performance Computing Workshop (HAPCW 2006), October 2006.

[5] S. M. Khattab, C. Sangpachatanaruk, D. Mossé, R. Melhem, T. Znati, “Roaming Honeypots for Mitigating Service-Level Denial-of-Service Attacks,” Proceedings of 24th International Conference on Distributed Computing Systems, March 2004.

[6] S. M. Khattab, C. Sangpachatanaruk, D. Mossé, T. Znati, “Proactive Server Roaming for Mitigating Denial-of-Service Attacks”, Annual Simulation Symposium, 2003.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, “A Scalable Content-Addressable Network”, Proceedings of ACM SIGCOMM Conference, August 2001.

[8] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems”, Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), November 2001.

[9] M. Srivatsa, L. Liu, “Countering Targeted File Attacks using LocationGuard,” Proceedings of 14th USENIX Security Symposium (USENIX Security 2005).

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, Proceedings of ACM SIGCOMM Conference, August 2001.

[11] M. Szymaniak, G. Pierre, M. V. Steen, “Latency-Driven Replica Placement,” Proceedings of the 2005 IEEE International Symposium on Applications and the Internet, February 2005.

[12] X. Tang and J. Xu, “On replica placement for QoS-aware content distribution,” Proceedings of IEEE INFOCOM’2004, March 2004.

[13] BRITE. Boston University Representative Internet Topology Generator. Available at http://www.cs.bu.edu/brite/