QoS-aware Object Replication in Overlay Networks

Won J. Jeon*, Indrail Gupta†, and Klara Nahrstedt†
*Motorola PCS Design Center
Champaign, Illinois 61820
wonjeon@cs.uiuc.edu
†Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
{indy,klara}@cs.uiuc.edu

***Abstract*— Many emerging applications for peer to peer overlays may require nodes to satisfy strict *timing deadlines* to access a replica of a given object. This includes multimedia and hard real-time applications such as distributed gaming. We formulate the QoS-aware replication problem, the goal of which is to locate the minimum number of replicas to satisfy access time deadlines for all nodes while minimizing storage usage in the overlay. Existing replication schemes cannot be used to solve this problem since they are best-effort only. We show that finding a solution to the QoS-aware object replication in an arbitrary overlay topology is intractable (NP-complete). We then present simple centralized as well as decentralized heuristics for QoS-aware replication, and compare their performance experimentally. In addition, we investigate how these decentralized heuristics effectively works in a real network.** [1]

I. Introduction

Content distribution networks (CDNs) and peer-to-peer (P2P) overlay networks replicate data objects such as documents and multimedia sources in order to reduce access time, storage utilization, and bandwidth use. However, all replication schemes suggested for P2P systems so far are best-effort only [1][2][3], while many P2P applications are required to satisfy *a priori* specified *bounds* on access times. One example of such an application is a multimedia distributed gaming application where a client needs to locate details of a target object within a deadline [4]. Another example is a wide-area server farm that may require hard-real-time guarantees [5] in order to satisfy service level agreements (SLAs).

Such applications require an object to be replicated so that *every* node in the P2P overlay can access a replica within the a priori specified delay. At the same time, the replication scheme should create only the smallest number of replicas so that additional storage usage is minimized. We call this the *QoS(Quality of Service)-aware object replication problem* for overlay networks.

In this paper, we first show that even a simple problem formulation is NP-complete; this means that calculating the optimal object replication is prohibitively expensive, even in a static overlay with decision-making delegated to a centralized node. We then present several distributed and decentralized heuristics for object replication that are *conservative*, i.e., satisfy the specified access time deadlines, yet manage to achieve low storage overhead. We validate our approximations with experimental results that evaluate and compare the effectiveness of these distributed schemes.

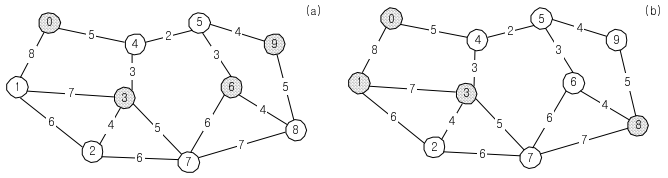

Fig. 1. Example of Object Replication in an Overlay Network: Nodes are associated with their unique nodeIDs, and links are associated with delays. Object replicas are stored at the shaded nodes. In solution (a), node 1 does not satisfy its deadline of 5 time units, while solution (b), all nodes satisfy their deadline of 5 time units, and storage usage is also minimized.

QoS-Aware Replication in Overlays: The problem is concretely defined as follows. Consider an arbitrary overlay graph $G = (V, E)$, with V the set of nodes and $E \subset V \times V$ the set of links between the nodes in the network[2]. Each edge e connecting node i and j in E is associated with a delay $d(i, j)$. Each node can access the object in question, and is also capable of storing a replica of the object.

We first consider the simplest specification of the problem. The QoS-aware replication problem is to assign to each node j a value of X_j either 1 or 0 (indicating whether the object is stored at node j or not), in such a way, to minimize the total number of replicas, $\sum_{i \in V} X_i$ for all nodes i, $\bar{d}(i, j) \leq \delta$, where $\bar{d}(i, j)$ represents the measured delay of the link from i to node j, which is the closest node to i that also has the object. δ is the deadline required by every node to access the object. This is different from typical P2P replication problems since our problem requires satisfaction of access deadline.

As an example, consider the overlay depicted in Figure 1. There are 10 nodes and 15 links, with each node identified by its nodeID, and the delay specified on each link. Suppose

[1]This research was done when the authors were at the University of Illinois at Urbana-Champaign and supported by the National Science Foundation under Grant No. CCR-0205638.

[2]Here we only consider the replication problem for a single object. An object is an independent unit data entity such as state information in distributed games, web pages or multimedia streams in servers and proxies. Any two objects are assumed to be semantically independent to each other and accessed independently.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

each node has a delay requirement $\delta = 5$ time units to retrieve an object from its neighbors. We show two different configurations of object replication in Figure 1(a) and Figure 1(b). Object replicas are stored at the shaded nodes. Both configurations have four replicas in the network, but node 1 in Figure 1(a) cannot retrieve the object within 5 time units, whereas all the nodes satisfy the deadline in Figure 1(b).

The rest of the paper is organized as follows. In Section II, we describe the related work. Then, in Section III, we give a proof of NP-completeness of this problem and the centralized approximation solution for this problem. We describe our distributed replication methods in Section IV. The simulation and experiment are shown in Section V and finally Section VI concludes our paper.

II. Related Work

Best-effort replication problems for overlays bear some similarity to the *facility location problem* studied in operations research [6]. Depending on the capacity constraint, this problem can be either *capacitated* or *un-capacitated*. The analogous problem in Computer Science is the *distributed paging* [7][8], which deals with the dynamic allocation of copies of files in a distributed network so as to minimize the total communication cost over a sequence of read and write requests.

Recent research has studied replica placement on the Internet for efficient content distribution. Replication techniques are divided into mirror (or proxy) replication [9][10][11] and content replication[12][13][14]. The former has been often discussed in the web caching context [3].

In order for efficient media delivery in overlay networks for streaming applications, most of work including [15] have focused on how to construct an overlay multicast to achieve the maximum throughput or the minimum delay. On the other hand, minimum or optimal replication problem has been discussed for file sharing applications [1] [2]. Even though some of them discussed a decentralized and self-organizing scheme, most of work only provides the optimal bound of replications based on a centralized approach. Recent work by Tang *et al.*[16] considered the QoS requirements in content delivery among content servers. Among their two replication scenarios, the replica-aware service looks similar to our configuration. However, they only provided a centralized greedy-based heuristic algorithm.

The QoS-aware replication problem discussed differs from the traditional content replication problem in two ways. Firstly, we try to guarantee the timed delivery of an object to every node in a network with the minimum number of replicas in the network. Solutions for facility location, distributed paging, and optimal replication algorithms, attempt to achieve the minimum cost of communication, not considering bounded cost requirement for nodes. Secondly, as overlay networks contain nodes that are managed by independent administrative authorities, a centralized replication protocol would be infeasible. The related work mentioned above only provides a centralized solution.

III. QoS-aware Object Replication

In this section, we show that the QoS-aware object replication problem is NP-complete. We give a centralized heuristic for finding an approximate solution.

A. Intractability of the Problem

We consider the QoS-aware replication problem outlined in Section I. All content size is assumed to be fixed, and storage capacity assumed to be infinite. The overlay graph may contain unidirectional and/or bidirectional links. [3]

We now precisely define QoS-aware object replication problem. Let $G = (V, E)$ be a graph with an associated delay $d(e)$ for each edge e in E and deadline δ for each node. Let V' be the set of nodes that contains a replica of the object. The delay for a path is the sum of the delays assigned to the edges in the path. Let $\bar{d}_i$ be the minimum delay for the node i to retrieve the object among all the possible retrieval paths from i to the replica set V'. If a node has the replica, then, of course, $\bar{d}_i = 0$. In this setting, we want to find a minimum-sized replica set $V' \subseteq V$ such that the retrieval of the object at each node in V meets the deadline, i.e., $\bar{d}_i < \delta$ for each i in V.

Theorem 1: The QoS-aware object replication problem is *NP-complete*.

Proof: We show that Minimum Dominating Set problem which is known to be a special instance of Minimum Set Cover problem is reduced to QoS-aware object replication problem. Let us define formally the solution of our replication problem, QAR as follows:

$$\{\langle G = (V, E), d, \delta, m\rangle \mid \text{There is } V' \subseteq V \text{ such that } \bar{d}_i \leq \delta_i \text{ and } |V'| \geq m \},$$

where G is a graph, d the delay of each link, δ the deadline for each node, $\bar{d}_i$ the minimum delay for node i to retrieve the replica, and m the size of the replica set. The solution of Minimum Dominating Set, MDS is the set

$$\{\langle G = (V, E), m\rangle \mid \text{There is } V' \subseteq V \text{ such that } V' \cap E = \emptyset \text{ and } |V'| \geq m\}$$

It is well-known that Minimum Dominating Set is NP-complete [18], [19].

Suppose that we are given $\langle G = (V, E), m\rangle$ as an instance of Minimum Dominating Set. We consider a delay d such that $d = 1$ for every e in E, and deadlines $\delta = 1$ for each node i in V. Suppose that V' is a replica set. Since the delay for each edge is 1 and deadline is 1, a node must have a neighbor in V'. Therefore, V' is also a minimum dominating set for G. On the other hand, if V' is a minimum dominating set for G, then clearly V' is a replica set for G with respect to the unit delay and the unit deadlines. Hence $\langle G, m\rangle \in MDS$ if and only if $\langle G, d, \delta\rangle \in QAR$, where $d = 1$ for every e in E, and deadlines $\delta = 1$ for every node in V. It is clear that this reduction is computable in polynomial time. ■

[3] The heuristics and distributed protocols we provide thereafter also work in systems where nodes have finite storage – this is achieved by eliminating such nodes from the considered candidates for replica location. The consistency of replicas when read and write operations are allowed has been discussed in many previous work [17], so we do not consider it in this paper.

1-4244-0357-X/06/$20.00 ©2006 IEEE

B. Centralized Solution

The above proof gives us a centralized heuristic for the QoS-aware object replication problem. The main idea is to modify the original QoS-aware object replication problem to the well-known Minimum Dominating Set problem.

1) Collect the link delay d_{ij} for every link e_{ij} $(i, j \in \{0, 2, ..., N-1\})$, where N is the total number of nodes in a network.
2) Construct the matrix B using the deadline constraint δ. B is a $N \times N$ matrix and each element b_{ij} of B equals δ.
3) Calculate the shortest paths from every node i to j $(i, j \in N)$ and construct it with the matrix A. A is a $N \times N$ symmetric matrix and each element a_{ij} of A represent the delay of shortest path between the node i and j.
4) Calculate the matrix A' by subtracting B from A. Then, construct the matrix C by marking every element in A' which is less than or equal to zero with 1, and with 0 otherwise. Each element c_{ij} of C represents the existence of edge between the node i and j. Note that C actually represents the graph with no link delay information, which is used to find a solution to Minimum Dominating Set problem.
5) Construct the graph from the matrix C. If the element c_{ij} equals 1, then draw an edge between the node i and j.
6) Find a solution of Minimum Dominating Set problem for this graph based on the approximation algorithm for Minimum Dominating Set problem.
7) Allocate a replica of the object to each node which is the element of the minimum dominating set.

We clarify the algorithm for our example of Figure 1. For that example, we obtain the matrix A representing all pairs shortest paths. Each element $a_{ij}(i, j \in [0, 9])$ represents the delay of the link from the node i to j.

$$A = \begin{bmatrix} 0 & 8 & 12 & 8 & 5 & 7 & 10 & 13 & 14 & 11 \\ 8 & 0 & 6 & 7 & 10 & 12 & 15 & 12 & 19 & 16 \\ 12 & 6 & 0 & 4 & 7 & 9 & 12 & 6 & 13 & 13 \\ 8 & 7 & 4 & 0 & 3 & 5 & 8 & 5 & 12 & 9 \\ 5 & 10 & 7 & 3 & 0 & 2 & 5 & 8 & 9 & 6 \\ 7 & 12 & 9 & 5 & 2 & 0 & 3 & 9 & 7 & 4 \\ 10 & 15 & 12 & 8 & 5 & 3 & 0 & 6 & 4 & 7 \\ 13 & 12 & 6 & 5 & 8 & 9 & 6 & 0 & 7 & 12 \\ 14 & 19 & 13 & 12 & 9 & 7 & 4 & 7 & 0 & 5 \\ 11 & 16 & 13 & 9 & 6 & 4 & 7 & 12 & 5 & 0 \end{bmatrix} \quad (1)$$

Since the overlay has bidirectional links, the matrix is symmetric. Then, construct the matrix B with each element b_{ij} of B being the delay requirement δ.

$$B = \begin{bmatrix} 5 & 5 & ... & 5 & 5 \\ & & ... & & \\ 5 & 5 & ... & 5 & 5 \end{bmatrix} \quad (2)$$

If we subtract the delay requirement $\delta = 5$ units of time from all the elements e_{ij}, we obtain the following matrix A'.

$$A' = \begin{bmatrix} -5 & 3 & 7 & 3 & 0 & 2 & 5 & 8 & 9 & 5 \\ 3 & -5 & 1 & 2 & 5 & 7 & 10 & 7 & 14 & 11 \\ 7 & 1 & -5 & -1 & 2 & 4 & 7 & 1 & 8 & 8 \\ 3 & 2 & -1 & -5 & -2 & 0 & 3 & 0 & 7 & 4 \\ 0 & 5 & 2 & -2 & -5 & -3 & 0 & 3 & 4 & 1 \\ 2 & 7 & 4 & 0 & -3 & -5 & -2 & 4 & 2 & -1 \\ 5 & 10 & 7 & 3 & 0 & -2 & -5 & 1 & -1 & 2 \\ 8 & 7 & 1 & 0 & 3 & 4 & 1 & -5 & 2 & 7 \\ 9 & 14 & 8 & 7 & 4 & 2 & -1 & 2 & -5 & 0 \\ 5 & 11 & 8 & 4 & 1 & -1 & 2 & 7 & 0 & -5 \end{bmatrix} \quad (3)$$

We mark all the elements equal to or less than zero with 1, and with 0 otherwise to transform our QoS-aware object replication problem to the dominating set covering problem.

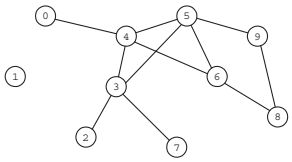

Fig. 2. Transformed graph from Fig. 1(a) without delay information on each edge

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (4)$$

Note that the matrix C corresponds to the following graph which does not have a delay information for each edge, which will be used to calculate a minimum dominating set later.

Now we apply any approximation algorithm to select the minimum dominating set problem to locate the replicas in the subset of the nodes in the graph. if we select the node 0, 1, 3, and 8, then the nodes that are covered by the nodes 0, 1, 3, and 8 are:

$$\begin{aligned} &SET_0 \cup SET_1 \cup SET_3 \cup SET_8 \\ &= \{0, 4\} \cup \{1\} \cup \{2, 3, 4, 5, 7\} \cup \{6, 8, 9\} \\ &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \end{aligned}$$

where SET_i represents the set of nodes that are covered by the node i. Therefore, if we allocate the object at the node 0, 1, 3, and 8, then all the nodes in the network are covered, which means that the delay requirements for all nodes are satisfied (in Figure 1(b)).

Calculating the minimum dominating set: The above centralized solution requires an algorithm for finding the minimum dominating set. The classical heuristic solution is a greedy one. Initially, all nodes are marked as not covered. Iteratively, find the node with the most (1 hop) neighbors not covered so far. Then, place a replica of the object at this node, and mark this node and its (1 hop) neighbors as covered . This method is known to be approximable within $1 + ln|V|$ where $|V|$ is the number of vertices in the given graph [20].

IV. Distributed Solutions

In large-scale P2P overlay networks where operational control is decentralized, and nodes are administrated by different authorities, calculating the minimum number of replicas of an object through the centralized algorithm is infeasible. Further, if link delays, the overlay links, or the set of nodes change over time (due to churn), periodically reporting such information to the central decision maker would be expensive. This motivates algorithms that are distributed and operate continuously. We present two flavors below.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

A. Core-selection Method

Each node i already knows its immediately neighboring nodes (in the overlay)[4], and the delays to reach any neighbor node j in the network through the $i \rightarrow j$ link. In addition, a node always maintains information of which of these immediate neighbors store replicas of the object, and the in-degree of each of these neighbors.

The node continuously monitors whether there is at least one object replica either at itself, or at one of its neighbor nodes reachable through with a delay $\leq \delta_i$ (delay constraint for the node i). If this condition ever becomes false, node i first checks if all its neighbor nodes j have an $i \rightarrow j$ delay $> \delta_i$; then, node i becomes a *core* i.e., it fetches and stores an object replica. Otherwise, node i selects that neighbor as core that has a delay $\leq \delta_i$ *and* has the maximum in-degree; this neighbor node is asked to fetch a replica of the object. This approach attempts to maximize the number of nodes that can access this new replica.

B. TTL (Time To Live)-based Method

The above approach has a node create replicas when it finds there is a risk that future requests *from* itself will violate the timing deadline. A diametrically opposite approach is the TTL-based algorithm, where a node decides when to create a replica at itself when future requests might come *to* itself from a different node. In this approach, a node storing an object replica maintains a list of *dependent* nodes (which may or may not be immediate neighbors) that are relying on it for accessing the object. If this list becomes empty, the node can delete the object replica.

Any node that does not have a local replica of the object periodically creates a *QoS-advert* message contains (a) its δ_i deadline value and (b) *depending-on*, the ID of the node that it is currently depending on (may be empty). The QoS-advert is broadcast to each of its immediate neighbors connected by a link with a delay smaller than δ_i. Each received QoS-advert is re-broadcast by the receiving node j, but only if the QoS value (δ_i) in the QoS-advert message is greater than or equal to the delay on the outgoing link.

If this condition evaluates to false for all neighbors of j, node j checks if the *depending-on* field in the message is either empty or specifies j. If either is true, node j replies to the QoS-advert, and this reply is reverse routed. Otherwise, node j does nothing. If *depending-on* field specified j, the original sender is added to the dependents list at j, and a local replica of the object is fetched unless one exists already.

When a node has sent out a QoS-advert with a non-empty *depending-on* field, it waits for the node specified therein to send a reply. If no such reply is received (determined by a timeout), another QoS-advert is sent immediately with the *depending-on* field empty. When a node has sent out a QoS-advert with an empty *depending-on* field, it waits to receive replies, makes a decision about whom to depend on, and sends out a QoS-advert with the *depending-on* field set.

[4]Usually a couple of random neighbor nodes are given to a newly participating node in its bootstrapping procedure.

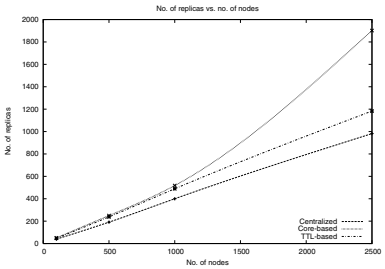

Fig. 3. No. of replications vs. no. of nodes

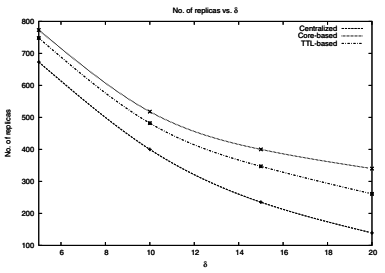

Fig. 4. No. of replications vs. δ (total no. of nodes = 1000)

Successive QoS-advert's by a node need to be uniquely identified by sequence numbers. The above algorithm is also amenable to aggregation of QoS-advert's and replies. The above algorithm can also be used when all nodes start out with a local replica, but delete this if no QoS-advert's are received for a while.

Node Failures: After a node or a link in the overlay fails, there could be a time window during which some requests for the object may not satisfy the deadline. No replication algorithm can eliminate this time window completely; however, good clock synchronization, and prudent values for the period and timeout can reduce the time windows sizes.

C. Initial Procedure

When a new node joins a network, at first it contacts the bootstrapping server to get the list of its neighboring nodes. Usually, the bootstrapping server gives a list of a few random nodes regardless of its location. Some of them might be very closely located to it and have smaller delays and the others might be located very far from it and have longer delays. Once the node populates the neighboring information, it filters out the nodes that have longer delay than the delay threshold value (specified by the system beforehand) and only keeps the nodes closely located as its neighbors.

V. Simulation Results

We compare all the three above approaches through simulation. We generated several P2P overlay network topologies with different numbers of nodes. We used BRITE network topology generator[21] to generate power-law based overlay

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

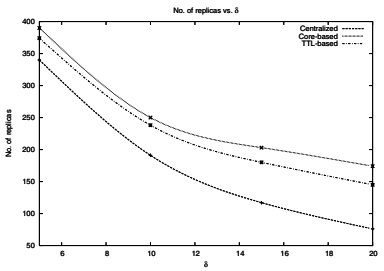

Fig. 5. No. of replications vs. δ (total no. of nodes = 5000)

network topologies. Four different network configurations consist of 100, 500, 1000, and 2500 nodes, respectively. The link delays on these topologies varied from 1 to 41, with an average around 11 to 12 for different topologies. For all the configurations, the QoS parameter(δ) was set to 10 for all nodes.

We compare the centralized and the distributed schemes against each other. The distributed schemes in the last section are for the continuous variety of the QoS-aware replication problem. However, in order to create a common ground for comparison, we evaluate only the *one-shot* version of the distributed algorithms, i.e., no overlay changes (node or link failures, or churn) occurs in the system. This means that the plots in Figure 3 show the behavior of the system after the algorithms quit. These plots show the performance of replication methods for the four different network configurations.

The centralized approximation algorithm performs better than the core-selection and TTL-based distributed methods in terms of number of replicas in the network. Among two distributed methods, the TTL-based solution performs better, and ends up with fewer replicas than the core-selection method. The overhead of replication of distributed methods compared with the centralized algorithm decreases as the number of nodes increases, and does not exceed 30% in our four configurations. The reason why the TTL-based method performs better than the core-selection one is that each node could have its depending node more than one hop away from it, so that the total number of depending nodes in the network could decrease.

Figure 4 and 5 show how different QoS parameters affects the number of replicas in the network. The TTL-based method continues to perform better than the core-selection method even as the QoS parameter(δ) value increases. This is because with increasing δ, in the TTL-based method, each node has its depending node a little far away.

VI. Conclusion

The emergence of successful killer applications for P2P and overlay networks depends on how well these substrates can satisfy application-specified requirements. In this paper, we have formulated one such problem – the QoS-aware replication problem – where each node specifies an upper bound on the time to access a given object. The problem is then to minimize the number of replicas inserted in order to satisfy these access deadlines. We have shown that this problem is intractable, since it is NP-complete. We have presented centralized as well as distributed solutions, and found that the distributed schemes perform almost as well as the centralized one, creating a small number of replicas at the right locations to satisfy all QoS requirements.

Acknowledgment

We thank Steven Ko for a discussion about the simulation methodology and the network topology generation and Sung-il Pae for the mathematical formulation.

References

[1] J. Kangasharju, K. W. Ross, and D. A. Turner, "Optimal content replication in p2p communities," *Manuscript*, 2002.
[2] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *ACM SIGCOMM '02*, Pittsburgh, Pennsylvania, August 2002.
[3] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *Proceedings of IEEE INFOCOM 2001*, 2001.
[4] B. K. et al., "Peer-to-peer support for massively multiplayer games," in *IEEE INFOCOM*, 2004.
[5] M. Aron, "Differentiated and predictable quality of service in web server systems," 2000, mohit Aron. Differentiated and Predictable Quality of Service in Web Server Systems. PhD thesis, Department of Computer Science, Rice University, October 2000.
[6] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Analysis of a local search heuristic for facility location problems," in *The Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, California, 1998, pp. 1–10.
[7] B. Awerbuch, Y. Bartal, and A. Fiat, "Distributed paging for general networks," in *The Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, Atlanta, Georgia, 1996, pp. 574–583.
[8] ——, "Competitive distributed file allocation," in *The Twenty-fifth Annual ACM Symposium on Theory of Computing (STOC '93)*, San Diego, California, 1993, pp. 164–173.
[9] J. Xu, B. Li, and D. L. Lee, "Placement problems for transparent data replication proxy services," *IEEE Journal of Selected Areas in Communications*, vol. 20, no. 7, September 2002.
[10] E. Cronin, S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," *IEEE Journal of Selected Areas in Communications*, vol. 20, no. 7, September 2002.
[11] B. J. Ko and D. Rubenstein, "Distributed server replication in large scale networks," in *Network and Operating System Support for Digital Audio and Video*, Ireland, June 2004.
[12] A. Jiang and J. Bruck, "Optimal content placement for en-route web caching," in *Second IEEE International Symposium on Network Computing and Applications*, April 2003.
[13] M. Karlson and C. Karamanolis, "Bounds on the replication cost for qos," *HP Technical Report HPL-2003-156*, July 2003.
[14] X. Tang and S. T. Chanson, "Coordinated en-route web caching," *IEEE Transactions on Computers*, vol. 51, no. 6, June 2002.
[15] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," in *IEEE INFOCOM 2003*, 2003.
[16] X. Tang and J. Xu, "On replica placement for qos-aware content distribution," in *IEEE INFOCOM 2004*, 2004.
[17] A. Tannenbaum, *Distributed Operating Systems*. Prentice Hall, 1995.
[18] R. M. Karp, *Reducibility Among Combinatorial Problems*. Plenum Press, NY, 1972.
[19] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
[20] D. S. Johnson, "Approximation algorithms for combinatorial problems," in *The fifth Annual ACM Symposium on Theory of Computing (STOC '73)*, 1973, pp. 38–49.
[21] "Brite: Boston university representative internet topology generator," http://www.cs.bu.edu/brite/.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.