Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019

Internet Content Replication: A Solution from Game Theory

Samee Ullah Khan and Ishfaq Ahmad
{sakhan,iahmad}@cse.uta.edu

Technical Report CSE-2004-5
July 2004

Internet Content Replication: A Solution from Game Theory

Samee Ullah Khan and Ishfaq Ahmad

Department of Computer Science and Engineering

University of Texas at Arlington, Arlington, Texas, USA

{*sakhan*,*iahmad*}*@cse.uta.edu*

Abstract: This paper addresses the fine-grained data replication problem on the Internet and proposes a novel solution using Game Theory. The proposed approach abstracts a large distributed system as an agent-based model wherein mobile agents continuously bid for allocation and re-allocation of system resources for their sites. The agents attempt to maximize certain performance measures such as the user access time and communication costs. The resource allocation mechanism is designed around the classical English and Dutch auction and bidding techniques using non-cooperative games that allow these agents to effectively compete for resources in an extremely fast manner. Optimal strategies for the agents to competitively acquire data are derived that yield Nash equilibrium in both cases. The results prove that if Nash equilibrium exists in either of the auctions systems, it conforms to an optimal solution towards the web content replication problem. Experimental results using various important parameters, accompanied by theoretical analysis, distinctly show the performance and advantages of the proposed scheme. Comparisons are also made with eight related algorithms, where the proposed techniques identify solutions within 5% of the optimal solution and reduce the execution times by several folds.

1. Introduction

Caching was traditionally applied to distributed file systems such as the AFS [35]. Although it is a well-studied problem, its application on the Internet gave rise to new problems, e.g., where to place a cache, how to make sure cached contents are valid, and how to handle dynamic pages. Replication, in contrast, has been commonly used in distributed systems to increase availability and fault tolerance. Both techniques play complementary roles in the Internet environment. Replication can be coarse-grained (replication of an entire site or server) or fine-grained (replication of individual data items). The majority of the existing work is based on the former approach. Caching attempts to store the most commonly accessed objects as close to the

clients as possible, while replication distributes a site's contents across multiple mirror servers. Replication accounts for improved end-to-end response by allowing clients to download from their closest mirror server. Caching can be viewed as a special case of replication when mirror servers store only parts of a site's contents [1]. This analogy leads to some interesting comparisons. For instance, cache replacement algorithms are examples of on-line, distributed, locally greedy algorithms for data allocation in replicated systems. Furthermore, caches do not have full server capabilities and thus can be viewed as a replicated system that sends requests for specific object types (e.g., dynamic pages) to a single server. Essentially, every major aspect of a caching scheme has its equivalent in replicated systems, but not vice versa. Replication, as a side effect, leads to load balancing and increases client-server proximity. Sites are commonly mirrored, and several schemes address the issue of redirecting [47]. For fault-tolerant and highly dependable systems, replication is essential, as demonstrated in a real example of OceanStore [49]. A number of bibliographies and reading materials for web caching are also available online, e.g., [16]. Replication and its challenges are provided in [35] and [47]. The two popular problem formulations for the data replication are as follows:

Coarse-Grained Replication Model: Similarity to the celebrated distributed file allocation problem [52], has moved the researches to address the problem of data replication on similar lines: "*Given* N *files, identify* M *out of them for a site(s) such that certain constraints are minimized*." These constraints could be memory, reduction in latency, communication cost, etc. The majority of the initial work assumed the coarse replication model [26], [27], [33], [46]. Both simulation [32] and real data from Internet log files [46] have been used to validate heuristic approaches such as greedy [46], dynamic programming [33], randomized and hot spot [46]. However, in the context of high-performance systems, coarse-grained allocation is a too simplistic approach.

Fine-Grained Replication Model: Fine-grained replication allows the allocation of certain objects and not the entire site. This approach has many advantages, such as; it saves the server memory capacity by only moving those object that are actually required to be moved, the problem definition is highly scalable and robust. The generalized fine grain replication problem is known to be NP-complete not only for general graphs [22], but also for partitioned graphs [28]. Several techniques such as the Greedy, Evolutionary, Adaptive Genetic, Linear Integer Programming, etc. have been utilized [28], [35].

There are various system related issues of the data replication problem that shall not be discussed in this article, but are identified in Figure 1. In this article, our focus is on the algorithmic aspects of data replication; therefore, we are more interested in the placement of the replicas. Replica placement determines where and how many replicas to be placed, so as to maximize the system performance and minimize latency, communication cost, etc. [47]. This, in turn, leads to the reallocation problem of continuously migrating objects that are necessary for generating dynamic pages, resulting in high transfer cost. Myriad theoretical approaches are proposed that we classify into the following six categories:

Facility Location: The simplest equivalent problem is the facility location problem, which can be defined as follows [22]: "*Find a location that minimizes a weighted sum of distances to each of the several locations."* This NP-complete [22] problem was first considered by Fermat in the 17th century and solved with an approximate algorithm [27]. The facility location problem does not fully capture the concept of replicating a single object/site over a fixed number of hosts [27], [34].

File Allocation: File allocation has been a popular line of research in: distributed computing [38], distributed databases [4], multimedia databases [30], paging algorithms [20], and video server systems [52]. The generalized file allocation problem for multiple objects has been proven to be NP-complete [13], [19]. We can formally state the file allocation problem in the context of the replication problem as [35]: *"For a network of* M *sites each with different storage capacity, replicate* N *files such that it satisfies the storage constraint and also optimizes some performance parameters, e.g., network flow and/or reduce download speed*." The problem was studied under the assumption of unlimited capacity [31], where the authors provided a guaranteed optimal result but of little practical use.

Minimum k-Median: The celebrated NP-complete minimum *k*-median problem [22], which captures coarse-grained replication well, is formally defined as follows [35]: "*Given a graph G*(*V*,*E*) *with weights on the nodes representing the number of requests and lengths on the edges, satisfy a request such that it minimizes the network cost of traversal and the path from the origin node and a server(s)."* The problem is solved in context of data replication with heuristics [28], [33] and approximation algorithms [6], [53].

Capacity-constrained Optimization: The capacity-free (unlimited storage) version has a better worst-case performance than the capacity-constrained version [11]. In [46], the authors use the capacity-constrained version of the minimum *k*-median problem, and guarantee stable performance. However, such results are possible only with very conservative assumptions [26], [32] and can not handle dynamics of the system [35].

Bin Packing: Widely studied in the field of on-line algorithms [53], the bin packing problem is known to be NP-hard [22]. We can formally state the bin packing problem with context to replication as [35]: "*Given* N *various objects of different sizes, partition them into the minimum number of disjoint sets such that the cumulative storage at each set does not exceed a certain threshold*." This approach is studied in [47] and [55] showing good results when the network under consideration is small.

Knapsack: To reduce network latency, a proactive web server can decide where to place the copies of the objects in a distributed web server. To achieve better load balancing partial replication is employed [34]. In [10] and [38] the authors have primarily adopted the knapsack problem approach. The knapsack problem in the context of the data replication problem is formulated as follows [35]: *"Given a network of* M *nodes with distinguishable capacities and* N *objects, find a subset of objects whose total size is bounded by the capacities for a site, and the total profit is maximized*."

Replication (*Both fine- and coarse-grained*)	Replica Placement	Facility Location	[27].
		File Allocation	[4], [13], [19], [20], [30], [38], [52].
		Minimum k-median	[6], [28], **[33]**, [53].
		Capacity-constrained Optimization	[11], [26], [32], **[46]**.
		Bin Packing	[47], [55].
		Knapsack	[10], **[34]**, [38].
		Game Theoretical	*This article.*
	System Issues	Server Identification	[2], [24], [27], [28], [44], [48].
		Replica Ranking	[3], [9], [12], [51].
		Redirection Service	[14], [15], [18], [47].
		Replica Consistency	[3], [8], [23].
		Concurrent Retrieval/Access	[36], [50].

Figure 1: Some system related issues and the major work reported in the field of replica placements. (Entries in bold represent that the work will be compared against in this article)

In this article we address the data replication problem as a *fine-grained capacity-constrained optimization* problem with the additional constraint of maintaining valid replicas. The article introduces new replica placement techniques based on Game Theory. To the authors best knowledge this is the very first work that address the data replication problem on the Internet using such techniques. A game theoretic

resource allocation mechanism is introduced that guarantees Nash equilibrium among a set of autonomous agents that compete in a non-cooperative environment to acquire data for replication in order to reduce the download time, add robustness and load balance the distributed system (Internet). The equilibrium is achieved when the agents are allowed to compete in either English or Dutch auctions. These agents evaluate their goals under system constraints, and compete to acquire objects for replication onto their respective site(s). Each agent evaluates the current state of the system, and if desired, modifies its strategy. This formulation enables us to conceptually solve the problem of data replication without any human interaction, since these agents if fully implemented, can be left to wander around over the Internet. Moreover, the agents continuously play these games, virtually making the system in a constant self-repairing mode. Based on the needs and desires of their respective clients which are associated with them by the virtue that they access the sites for which the agents are responsible, they can decided: when, where, how many and what kind of objects to replicate.

This article is organized as follows. In section 2, we give a brief introduction of Game Theory. Section 3 formally describes the data replication problem. In section 4 we proceed with our proposed resource allocation mechanism. Sections 5 and 6 are dedicated to the experimental results and concluding remarks.

2. Game Theoretical Background

Game Theory is widely thought to have originated in the early twentieth century, when von Neumann proved the min-max theorem [56]. Although, it was the first formally stated major work in this field, the roots of Game Theory can be traced back to the ancient Babylonian Talmud. The Talmud is a compilation of the ancient laws set forth in the first five centuries A.D. Its traces can be found in various religions and the modern civil and criminal laws. One related problem discussed in the Talmud is the marriage contract problem: A man has three wives. Their marriage contracts specify that in the case of the husband's death, the wives receive 1:2:3 of his property. The Talmud in all mystery gives self-contradictory recommendations. It states that: if the man dies leaving an estate of only 100, there should be an equal division. If the estate is worth 300 it recommends proportional division (50,100,150), while for an estate of 200, it recommends (50,75,75). In 1985, Aumann and Maschler [7] reported that the marriage contract problem and its weird

solution discussed in the Talmud are only justifiable via cooperative game theoretical analysis. The foundation of the famous min-max theorem is credited to Waldegrave, who on November 13, 1713 wrote a letter to de Montmort describing a card game *le Her* and his solution [30]. It would take two centuries for Waldegrave's result to be formally acknowledged. Some of the most pioneering results were reported within a year, when Nobel Laureate John Nash made seminal contributions to both *cooperative* and *non-cooperative* games. In [41] and [42], Nash proved the existence of a strategic equilibrium for non-cooperative games (Nash Equilibrium). He also proposed that cooperative games were reducible to non-cooperative games. In the next two papers [42], [43] he eventually accomplished that and founded the axiomatic bargaining theory and proved the existence of the Nash Bargaining Solution for cooperative games (a notion similar to the Nash Equilibrium). The beauty of Game Theory is in its abstractly defined mathematics and notions of optimality. In no other branch of Sciences, do we find so many understandable definitions and levels of optimality [45]. In computer science, a few applications of Game Theory applied to job scheduling and networking have been documented [17].

A special branch of Game Theory deals with *Biddings* and *Auctions*, which have long been an important part of the market mechanisms. Auctions allow the buyers to opt for prices often lesser than the original market prices, but they have to compete and in doing so they have to realize their needs and constraints. Analysis of such games began with the pioneering work of Vickery [54]. Recently, auction theory is being recognized as the emerging solution for problems in microeconomics, computational resource allocation and agent-based systems [29]. There are four standard types of auctions: (a) First price sealed bid, e.g., procurements; (b) Second price sealed bid, e.g., FCC bandwidth; (c) English auction (open and ascending), e.g., antiques; (d) Dutch auction (open and descending), e.g., fish markets. Many variations of the four basic types of auctions exist, depending on the kind of entity to be auctioned, kinds of participating bidders, and the purpose of the auction. Some examples of the modified versions of the basic types of auctions include: Ausubel Auctions, Vikrey Auctions, Discriminatory Auctions, Anglo-Dutch, etc. A comprehensive survey on the theory and the various types of auctions are reported in [37]. In the *first price* sealed bid auction each player submits its bid in a sealed envelope without the knowledge of what the other players have bid [29].

The winner is the player with the highest bid. The *second price* bid auction is exactly the same as the first price auction, but now the winner pays the second highest bid [37]. In *ascending* bid auction, the price of the entity is raised until only one bidder remains, and that bidder wins the entity [17]. In *descending* bid auction, the auctioneer starts the bidding at a very high price and continuously lowers the price until one of the players accepts the currently announced price. In this article we will not consider the first two types of auctions since the simplified versions of both the ascending and descending auctions are strategically equivalent to the second price sealed bid auction and the first price sealed bid auction respectively [54],[57].

Table 1: Notations and their meanings.

Symbol	Meaning
M	Total number of sites in the network.
N	Total number of objects to be replicated.
O_k	k-th object.
o_k	Size of object k.
S_i	i-th site.
s_i	Size of site i.
r_k^i	Number of reads for object k from site i.
R_k^i	Aggregate read cost of r_k^i.
w_k^i	Number of writes for object k from site i.
W_k^i	Aggregate write cost of w_k^i.
NN_k^i	Nearest neighbor of site i holding object k.
$c(i,j)$	Communication cost between sites i and j.
P_k	Primary site of the k-th object.
R_k	Replication schema of object k.
$C_{overall}$	Total overall data transfer cost.
B_i	A bidder i in our resource allocation mechanism.
$\tilde{B}_i$	Denotes a bid from a bidder B_i.
β_i	The bidding strategy of a bidder B_i.
$S(i)$	Set of sites for which a bidder (agent) B_i is responsible for.
S^j	Extraction function that identifies the size of a particular site from the set of sites $S(i)$.
v_i	Evaluation function that takes the bidding strategy β_i as input and projects a specific value in the interval G_i for a bidder B_i.
SGRG	Self Generated Random Graphs
GT-ITM PR	Georgia Tech Internetwork Topology Models Pure Random.
GT-ITM W	Georgia Tech Internetwork Topology Models Waxman.
SGFCGUD	Self Generated Fully Connected Graphs with Uniform Distribution.
SGFCGRD	Self Generated Fully Connected Graphs with Random Distribution.
SGRGLND	Self Generated Random Graphs with Lognormal Distribution.
DRP	Data replication problem.
RC	Replication cost (network communication cost).
MSV	Multi-item single vendor.

3. The Problem Description

Consider a distributed system comprising of M sites, with each site having its own processing power, memory (primary storage) and media (secondary storage). Let S_i and s_i be the name and the total storage capacity, respectively, of site i where $1 \leq i \leq M$. The M sites of the system are connected by a communication network. A link between two sites S_i and S_j (if it exists) has an integer $c(i,j) \geq 0$ associated with it, giving the communication cost for transferring a data unit between sites S_i and S_j. If the two sites are not directly connected by a communication link then the above cost is given by the sum of the costs of all the links in a chosen path from site S_i to the site S_j. Without the loss of generality we assume that $c(i,j) = c(j,i)$. This is a very common assumptions e.g. see [26], [34], [46], etc. Let there be N objects, each identifiable by a unique name O_k and size in simple data unites o_k where $1 \leq k \leq N$. Let r_k^i and w_k^i be the total number of reads and writes, respectively, initiated from S_i for O_k during a certain time period. Our replication policy assumes the existence of one primary copy for each object in the network. Let P_k, be the site which holds the primary copy of O_k, *i.e.*, the only copy in the network that cannot be de-allocated, hence referred to as primary site of the k-th object. Each primary site P_k, contains information about the whole replication scheme R_k of O_k. This can be done by maintaining a list of the sites where the k-th object is replicated at, called from now on the *replicators* of O_k. Moreover, every site S_i stores a two-field record for each object. The first field is its primary site P_k and the second the nearest neighborhood site NN_k^i of site S_i which holds a replica of object k. In other words, NN_k^i is the site for which the reads from S_i for O_k, if served there, would incur the minimum possible communication cost. It is possible that $NN_k^i = S_i$, if S_i is a *replicator* or the primary site of O_k. Another possibility is that $NN_k^i = P_k$, if the primary site is the closest one holding a replica of O_k. When a site S_i reads an object, it does so by addressing the request to the corresponding NN_k^i. For the updates we assume that every site can update every object. Updates of an object O_k are performed by sending the updated version to its primary site P_k, which afterwards broadcasts it to every site in its replication scheme R_k.

For the data replication problem (DRP) under consideration, we are interested in minimizing the total replication cost (RC) (or the total network communication cost) due to object movement. There are two

components affecting RC. First, is the RC created from the read requests. Let R_k^i denote the total RC, due to *Si*s' reading requests for object O_k, addressed to the nearest site NN_k^i. This cost is given by:

$$R_k^i = r_k^i o_k c(i, NN_k^i), \tag{1}$$

where $NN_k^i = \{Site\ \ j \mid j \in R_k \wedge \min\ \ c(i,j)\}$. The second component of RC is the cost arising due to the writes. Let W_k^i be the total RC, due to *Si*s' writing requests for object O_k, addressed to the primary site P_k. This cost is given by the following equation:

$$W_k^i = w_k^i o_k (c(i, P_k) + \sum_{\forall (j \in R_k), j \neq i} c(NN_k^i, j)). \tag{2}$$

Here, we made the indirect assumption that in order to perform a write we need to ship the whole updated version of the object. This of course is not always the case, as we can move only the updated parts of it (modeling such policies can also be done using our framework). The cumulative RC, denoted as $C_{overall}$, due to reads and writes is given by:

$$C_{overall} = \sum_{i=1}^{M} \sum_{k=1}^{N} (R_k^i + W_k^i) \tag{3}$$

Let X_{ik}=1 if S_i holds a replica of object O_k, and 0 otherwise. X_{ik}s define an $M \times N$ replication matrix, named X, with boolean elements. Equation 3 is now refined to:

$$X = \sum_{i=1}^{M} \sum_{k=1}^{N} (1 - X_{ik})[r_k^i o_k \min\{c(i,j) \mid X_{jk} = 1\} + w_k^i o_k c(i, P_k)] + X_{ik} (\sum_{x=1}^{M} w_k^x) o_k c(i, P_k) \tag{4}$$

Sites which are not the *replicators* of object O_k create RC equal to the communication cost of their reads from the nearest *replicator*, plus that of sending their writes to the primary site of O_k. Sites belonging to the replication scheme of O_k, are associated with the cost of sending/receiving all the updated versions of it. Using the above formulation, the data replication problem (DRP) can be defined as:

Find the assignment of 0, 1 values in the X matrix that minimizes $C_{overall}$, *subject to the storage capacity constraint*: $\sum_{k=1}^{N} X_{ik} o_k \leq s_i, \forall (1 \leq i \leq M)$, *and subject to the primary copies policy*: $X_{P_k k} = 1, \forall (1 \leq k \leq N)$.

The minimization of $C_{overall}$ will have two impacts on the distributed system under consideration: First, it ensures that the object replication is done in such a way that it minimizes the maximum distance between the replicas and their respective primary objects. Second, it ensures that the maximum distance between an

object *k* and the user(s) accessing that object is also minimized. Thus, the solution aims for reducing the overall object transfer cost of the system. In the generalized case, DRP is essentially a constraint optimization problem, reducible to the *Knapsack* problem, and without the storage constraints, to the *minimum k-median* problem [34].

4. The Proposed Resource Allocation Mechanism

In order to solve the above problem, we will first describe our agent-based model to represent the system. We then define the players involved and introduce the resource allocation accomplished through the bidding games played by the agents.

4.1 The Agent-Based Model

Agent-based models due to their flexibility are becoming increasingly popular [25]. In our system, an agent is a mobile task that is responsible for a certain number of sites (servers). It aims to optimize some profit for its set of sites. Figure 2(a) shows a network with 8 sites. The number on a communication link indicates the time (milliseconds) taken to transfer 1K Bytes of data, which is the sum of the latency and transmission time. The figure shows each agent with a unique ID and the set of its sites. For example, Agent *A* is responsible for sites 2, 5, 6, and 7, while Agent *B* is responsible for site 3, and Agent *C* is responsible for sites 1, 4, and 8. The number of agents to be unleashed in the system would depend on many factors, such as, the implementation overhead, network traffic, and how aggressive the participating sites are, etc. Although the determinacy of the number of agents in the system is altogether a different line of research, yet we provide experimental results that give an insight on the number of agents necessary in a system (Appendix F).

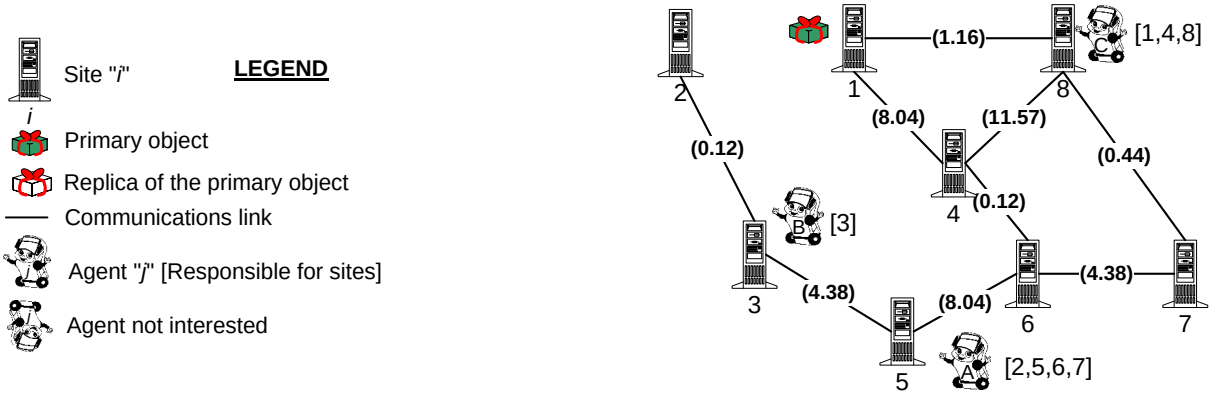

Figure 2(a): Agent initiation.

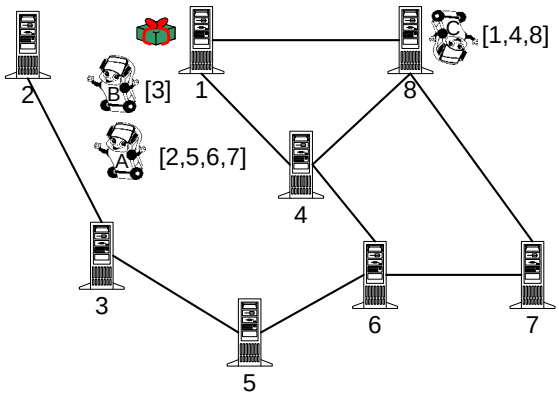

Figure 2(b): Agents move to compete for data.

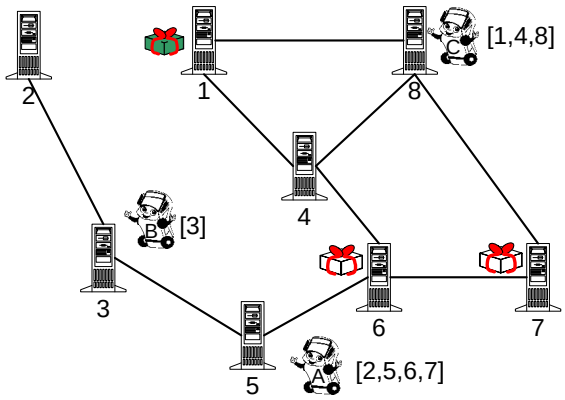

Figure 2(c): Agents move back to replicate.

We assume that there is only one primary object residing on site 1. Based on the information gathered from the user read/write requests, object sizes, and where the primary objects are located, etc., an agent builds its own partial replication cost index for the replication of an object of interest. When multiple agents desire the same object, they compete for it through bidding. Bidding takes place by converging the agents to the location of the primary object. This is done so as to reduce the overhead of control messages. To allow fair play, a bulletin board service announces that an object is available for auction. In Figure 2(b) Agents *A* and *B* convene at Site 1, while Agent *C* is not interested in the auction for reasons explained later. Here, the agents use various game theoretical strategies and optimization techniques to maximize their respective profits. The next section discusses various rules of the games, strategies, and bidding protocols. After acquiring their data, the agents move back to their respective regions to replicate the acquired data, and get ready to compete for the next object (see Figure 2(c)). From here onwards, we will refer to these agents as bidders.

4.2 The Players

Vendor(s): A vendor is the owner (site) of the primary object P_i. When a vendor puts a P_i up for bidding, it may use a certain criterion to allow bidders to win a maximum number of copies of P_i.

Bidder(s): A bidder is the agent of certain site(s). Let a bidder B_i own a set of uniquely identifiable sites $S(i)$. If a bidder B_i *wins* a certain number of objects, the replication of the objects can be done internally without any further cost. The decision where to replicate internally is up to the bidder, but in this article the internal replication is based on the ascending replication cost for that object per site(s) (bin packing).

Auctioneer: An auctioneer is responsible for conducting (announcing opening and closing of sessions) an auction. It is also responsible for overseeing the rules and regulations abiding the auction session as well as regulating a fair business among the bidders and the vendors. An auctioneer conducts an open bidding session for a set of sites, even though, theoretically, no refereeing body to overlook the game is required since, by definition, mathematical games are played among a set of *rational* players. However, the introduction of auctioneers is for practical reasons: It limits the scope of a bidder to its region, and allows scalability, as a large system may be partition into regions, each governed by an auctioneer. An auctioneer may be a separate entity but for now we assume a vendor acts as an auctioneer for its own primary object.

4.3 Gaming Mechanism

Let a vendor in a system of M sites bring an object j for auction to n bidders. The bidder B_i's bids are represented by a set, b_i, distributed over an interval $G_i = [\min(b_i), \max(b_i)] \in R^n$, where the bounds are defined as: $\min(b_i) = \min_{k \in S(i)-P_i} (R_j^k + W_j^k) \forall k \in S(i)$ and $\max(b_i) = \max_{k \in S(i)-P_i} (R_j^k + W_j^k) \forall k \in S(i)$. The lower bound, $\min(b_i)$, and the upper bound, $\max(b_i)$, respectively, represent the minimum and maximum read plus write costs for object j at a site belonging to the set of sites, $S(i)$. The entire game can be represented as a Cartesian product of the evaluations of all the bidders as $G = \times \sum_{i=1}^{n} G_i = G_1 \times G_2 \times \cdots \times G_n$ [39] (Appendix G).

Let us consider the bidding mechanism for the system shown in Figure 2. Here, the three agents try to bid for an object of size 1k Bytes. The set $S(C)$ for bidder C is {1, 4, 8}. Site 1 hosts the primary object and, thus, cannot be considered for replication, while Sites 4 and 8 have no available storage space. Therefore, bidder C is not interested. Table 2 shows some system parameters to be considered by bidders A and B. Bidder A calculates the read and write costs for replication of the object at each site as well as each combination of sites from its set $S(A)$ = {2, 5, 6, 7}. The possible sites and combinations of sites for replication as well as the replication costs by bidder A are shown in Table 3. For example, if the object is replicated at site 5, the read costs for accessing the objects from sites 2, 5, 6, and 7 are tabulated, as given by Equation 1. For instance, the read cost from site 2 to access the object from site 5 is given by the product of the read frequency, object size, and the transfer cost through the shortest path. The transfer cost for the local object is equal to the

latency (assumed to be 2.8×10^8 m/s), making the read cost from site 5 to be almost zero. Sites 6 and 7 access the same object, but from their nearest neighbor which is site 1 instead of site 5. The write cost (Equation 2) is the sum of the cost to send the update to the primary location and the broadcast cost to the set of sites ($S(A)$) holding the replica. Bidder B calculates these costs for its set of sites ($S(B)$) in a similar fashion.

Table 2: Some system parameters.

Bidder	Site	Shortest path from P_i	Transfer cost	Storage	Read frequency	Write frequency
A	2	[1,8,7,6,5,3,2]	18.51	Not available	10	20
	5	[1,8,7,6,5]	14.01	Available	5	17
	6	[1,8,7,6]	5.97	Available	30	15
	7	[1,8,7]	1.60	Available	10	1
B	3	[1,8,7,6,5,3]	18.40	Available	40	30

Table 3: Calculations of the read, write, and total costs by bidders A and B.

Bidder	Repl. site	Read cost [k, repl. site or nearest neighbor]				Write cost [primary, (repl. sites)]	Total Cost (sec.)
A	5	[2,5]=45000	[5,5]≈0	[6,1]= 179100	[7,1]=16000	[1,(5)]=238170	478.270
	6	[2,6]=125400	[5,6]=40200	[6,6] ≈0	[7,1]=16000	[1,(6)]=89550	185.950
	7	[2,7]=169200	[5,7]=62100	[6,1]=179100	[7,7] ≈0	[1,(7)]=1600	412.000
	5,6	[2,5]=45000	[5,5]≈0	[6,6] ≈0	[7,1]=16000	[1,(5,6)]=639300	700.300
	5,7	[2,5]=45000	[5,5]≈0	[6,1]=179100	[7,1]=16000	[1,(5,7)]=280980	521.080
	6,7	[2,6]=125400	[5,6]=40200	[6,6] ≈0	[7,7] ≈0	[1, (6,7)]=141120	306.720
	5,6,7	[2,5]=45000	[5,5]≈0	[6,6] ≈0	[7,7] ≈0	[1, (5,6,7)]=712140	748.140
B	3	[3,3]≈0				[1,(3)]=55200	552.000

The game now is for all the bidders to choose their optimal bids, and the highest bid wins the object. Bidders may use various strategies (Figure 3). For example, the bids may be based on a greedy approach, *i.e.*, they project the exact cost of replicating the object to the site where it costs less. In this case, bidder A wins object j for Site 6. A random choice of bids could result in arbitrary winners. A bid interval may be based on a certain distribution. For example, bidders may multiply their greedy bids by 3/2, so as to increase their probability of winning [37]. In this case bidder A wins the object j, but pays more than the actual worth. Strategies may be complex or may be simplified to reduce the complexity of the search space.

Site 2	Site 5	Site 6	Site 7
Nil	478	185	412

Sites 5,6	Sites 5,7	Sites 6,7	Sites 5,6,7
700	521	306	748

Site 3
552

Site 1	Site 4	Site 8
N/A	Nil	Nil

G_A=[185-748]

G_B=[552-552]

G_C=[*empty*]

Bidders	Types of bids (b_i)		
	Greedy	Random	Uniform
A	185	735	277
B	552	525	828

Figure 3: The values of bids under various strategies.

Fine-grained replication would require multiple item auction games. Such auctions (termed as *combinatorial auctions*) suffer from large amount of data as the bidders have a natural choice of combinations of items required by them [29]. Therefore, such auctions are not only hard to evaluate, but are also difficult to implement [54]. One solution for such auctions is the iterative identical multi-item single vendor game (MSV) [29], that is, at each iteration a vendor puts a certain number of identical items for auction, and the bidders have the flexibility to bid for the number of items that they wish to obtain if desired [17]. The formulation of MSV requires the concepts of revenue and incentives. An MSV may be conducted with or without an auctioneer; for instance, the primary object site may serve as the auctioneer. Each bidder has to its disposal certain amount of money. This money directly represents objects' popularity captured by the number of read and write frequencies at the bidder's set of sites. For simplicity, assume that each read and write at the replicated site brings \$10 each to the bidder's account, which is the due to the saving of not having to go to the primary site. In MSV each bidder uses the money in its bank account to obtain the rights to make copies of a primary object. The price paid by a bidder to copy an object *j* to a site should be close to the object's replication cost. Each bidder's bid is optimized to win the maximum number of copies as possible, while paying the minimum amount of money (*i.e.*, the cost). The general bidding process is shown in Figure 4. Next, we describe two important auction mechanisms and show how we cast our problem as such and find the solution.

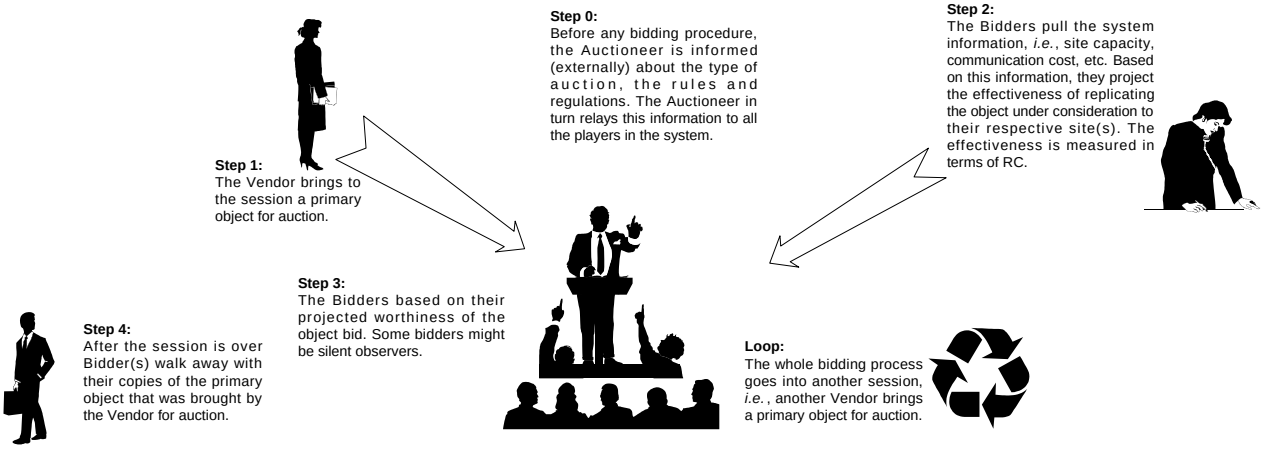

Figure 4: The bidding process in the context of the replication problem from a birds eye view.

4.4 Iterative Dutch Auction

In the traditional dutch auction the auctioneer begins with a high asking price which is lowered until some participant is willing to accept the auctioneer's price. That participant pays the last announced price. This type of auction is convenient when it is important to auction goods quickly, since a sale never requires more than one bid [57]. In no case does the auctioneer reveal any of the bids submitted to him, and no information is shared between the bidders. Figures 5(a)-(b) show an illustrative example of MSV using the Iterative Dutch auction. Assume that the bidders have acquired both the money in their account and their respective local replication costs. An object j is brought for auction at its primary site. The auctioneer makes a rough estimate on the value of the object by making a guess on the read and write frequencies of that object at all the sites and the savings in transfer costs if the object was replicated. The auctioneer's guess is based on its local read and write frequencies and an estimated average transfer cost. Suppose the numbers of reads and writes are 35, and 25, respectively, and that the average transfer cost is 10. Thus, the starting price is $(35+25)\times10=600$ expressed in dollars, with \$50 decrement (an external parameter). It is to be noted that the choice of decrement has no bearing on any auction procedure [17], [29], [39]. It is shown that in order for a bidder to have a probabilistically superior bid than n-1 other bids, a bidder should evaluate the asking price by dividing it by n [29], [37] (Appendix A and B). Since the asking price is \$600, bidder A finds a bid that is the closest to \$300. Thus, v_a=306 is the closest bid to replicate the object at Sites 6 and 7. But since v_a is less than the next possible asking price, *i.e.*, \$550, bidder A remains silent. On the other hand, bidder B has only

one choice, v_b=552, which lies between the current and next asking price. Thus, bidder *B* immediately bids. At this moment, a single object is awarded to bidder *B*. Contrary to the original Dutch auction, in order to sell more copies, we allow the auctioneer to reduce the asking price by the decrement of \$50 until one bidder is left. The final reduced price is reserved with one addition decrement for the last bidder. In this example, bidder *A*, immediately becomes the last player. Therefore, the auctioneer reserves the asking price after decrementing the current bid, *i.e.*, \$(600-50) = \$550. Bidder A has no option, then to comply. The whole process allows the bidders to make (2+1) 3 copies of the object *j* at Sites 3, 6 and 7.

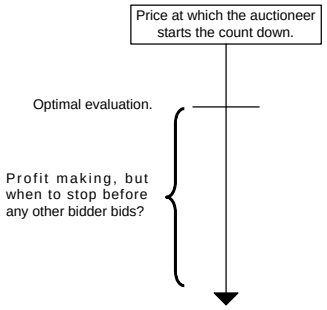

Figure 5(a): The Dutch auction procedure.

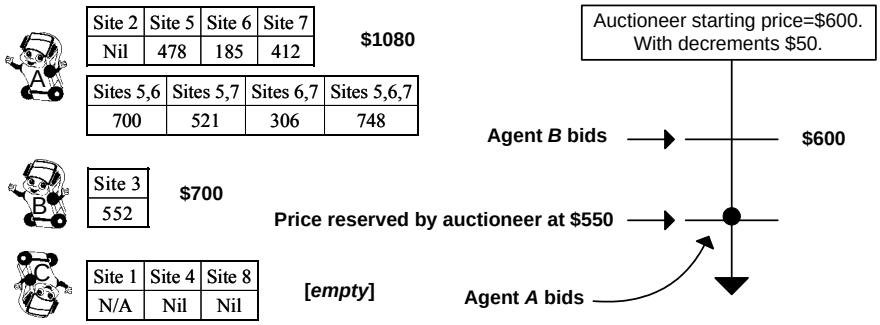

Figure 5(b): An example illustrating the replication process via Dutch auction in 2 iterations.

4.5 Iterative English Auction

In this type of auction, the participants bid openly against one another, with each bid being higher than the previous bid. The auction ends when no participant is willing to bid further [37]. During the auction when an auctioneer receives a bid higher than the currently submitted bids, he announces the bid value so that other bidders (if needed) can revise their currently submitted bids. In [29] and [37] the discussion on English auction reveals that the optimal strategy for a bidder *i* is to bid a value which is directly derived from his evaluation (Appendix C and D). Figures 6(a)-(b) describe the replica assignment process using the English auction. An object *j* is brought for auction. The auctioneer decides on the minimum bid increment (external parameter), which is \$50 in this case. Bidder *A* submits the first bid at \$185, *i.e.*, the minimum acceptable replication cost savings. Bidder *B* immediately replies with a bid of \$235, since the current bid plus the minimum bid increment is less than his evaluation of \$552. Bidder *A* responds to this by increasing his offer to \$285, since this offer plus an increment is less than his new evaluation of \$306. Now when bidder *B*

increases the offer to \$335, a response by bidder *A* of \$385 would mean that his new evaluation would become \$412. But this evaluation gains him nothing since with evaluation of \$306 he can obtain two replicas as appose to one with \$412. Therefore, it is in his favor to remain silent. With no response from bidder *A*, bidder *B* is declared a winner. On similar lines of the Iterative Dutch auction, in the Iterative English auction when a winner is determined, the auctioneer in order to sell more copies, reserves an asking price equal to the winner's bid. Therefore, all the *n*-1 bidders (expect the winner) have to re-compute their evaluations. It is to be noted that not all the *n*-1 bidders would comply with the reserve price. In this example, bidder *A* complies (willingly) since the reserved price is not more than its last offer plus the minimum bid increment. At the end of the session the bidders are allowed to make (2+1) 3 copies of the object *j* at Sites 3, 6 and 7.

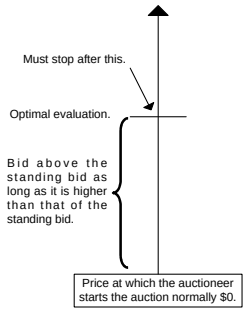

Figure 6(a): The English auction procedure.

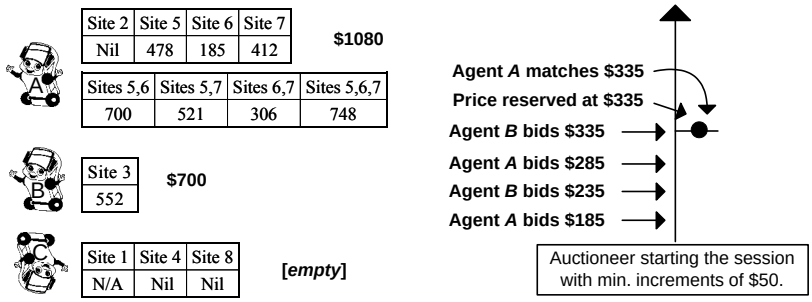

Figure 6(b): An example illustrating the replication process via English auction in 5 iterations.

4.6 Optimality of the Proposed Techniques

The question now is how good the above mechanism is. Is it optimal? If not, how far is it from the optimal? In order to find the answers to these questions, one must understand the notion of optimality in the context of game theory. Auctions fall in the category of non-cooperative games, that is, the players play independently with the objective of optimizing their *own* profits. In such an environment, the optimization of self-profits leads to a bottleneck where no other strategy can result in a better individual profit unless other players also change their strategies. This bottleneck point is explained by Nash [41]: *There is a set of strategies with the property that no player can benefit by changing its strategy while the other players keep their strategies unchanged, then that set of strategies and the corresponding payoffs constitute the (Nash) equilibrium.*

Thus, if every player plays its own game optimally, then the combination of all the games would constitute to a steady (equilibrium) state where no further benefit to any of the player is possible. Thus, every player is optimizing not only for itself but also (unknowingly) for the whole system.

In order to proceed with the claim that both the English and Dutch auctions provide optimal solutions towards the data replication problem, we first have to define the measure of optimality. In our resource allocation mechanism, there are two major players, *i.e.*, the bidders and the vendors. From the vendors' point of view, they seek to populate the system with the copies of their respective primary objects. From the bidders' point of view, they seek to obtain the copies in such a way that it gives them the exact worth of their evaluations. It is to be noted that the bidder's evaluations are directly extracted from the system information. Thus, if all the bidders optimize their evaluations, they indirectly optimize the replication cost. Optimization of the replication cost also indirectly optimizes the vendors' objective. Thus, all we have to show is that the expected revenue of the vendor is exactly the same as the bidder's expected evaluation (Appendix E).

5. Experimental Setup and the Discussion of Results

We performed experiments on a 440MHz Ultra 10 machine with 512MB memory. The experimental evaluations were targeted to benchmark the placement policies. The solution quality in all cases, is measured according to the RC percentage that is saved under the replication scheme found by the algorithms, compared to the initial one, *i.e.*, when only primary copies exist. The resource allocation mechanism was implemented using IBM Pthreads. Each player has its own thread except the vendor. The vendor thread is eliminated to reduce the total turn-around time. The network architecture is generated as follows. First, the number of sites M and objects N are defined. To establish diversity in our experimental setups, the network connectively is changed considerably. In this article, we limit or network to a maximum of 500 sites (nodes). We used existing topology generator toolkits and also self generated networks. Table 2 summarizes the various techniques used to gather forty-five various topologies for the 100 node networks. It is to be noted that the parameters vary for network with lesser/larger number of nodes. All the results reported, represent the average performance over all the topologies.

Table 4: Parameter interval variance characterization for topologies with 100 nodes.

Topology Name	Mathematical Representation	Parameter Interval Variance
SGRG (12 topologies)	Randomized layout with node degree (d*) and Euclidian distance (d) between nodes as parameters.	d={5,10,15,20}, d*={10,15,20}.
GT-ITM PR [9] (5 topologies)	Randomized layout with edges added between the randomly located vertices with a probability (p).	p={0.4,0.5,0.6,0.7,0.8}.
GT-ITM W [9] (9 topologies)	$P(u,v)=\alpha e^{-d/(\beta L)}$	α={0.1,0.15,0.2,0.25}, β={0.2,0.3,0.4}.
SGFCGUD (5 topologies)	Fully connected graph with uniform link distances (d).	d_1=[1,10], d_2=[1,20], d_3=[1,50], d_4=[10,20], d_5=[20,50].
SGFCGRD (5 topologies)	Fully connected graph with random link distances (d).	d_1=[1,10], d_2=[1,20], d_3=[1,50], d_4=[10,20], d_5=[20,50].
SGRGLND (9 topologies)	Random layout with link distance having a lognormal distribution [21].	μ={8.455,9.345,9.564}, σ={1.278,1.305,1.378}.

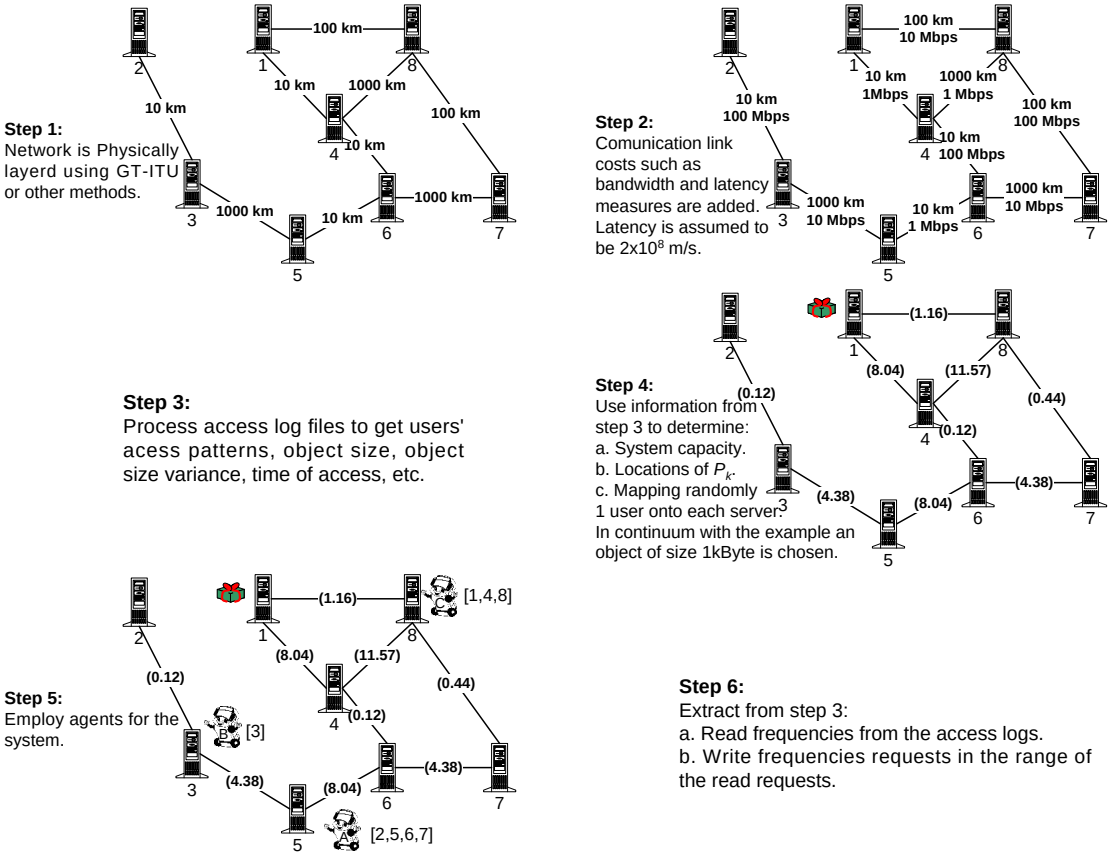

Figure 7: A walk through the necessary steps involved in an experimental setup.

To evaluate our proposed replication techniques on realistic traffic patterns, we used the access logs collected at the Soccer World Cup 1998 website [5]. Each experimental setup was evaluated thirteen times, *i.e.*, only the Friday (24 hours) logs from May 1, 1998 to July 24, 1998. Thus, each experimental setup in fact represents an average of the 585 (13×45) data set points. To process the logs, we wrote a script that returned: only those objects which were present in all the logs (2000 in our case), the total number of requests from a particular client for an object, the average and the variance of the object size. From this log we choose the

top five hundred clients (maximum experimental setup), which were randomly mapped to one of the nodes of the topologies. The primary replicas' original site was mimicked by choosing random locations. The capacities of the sites *C*% are generated randomly with range from *Total Primary Object Sizes/2* to *1.5×Total Primary Object Sizes*. The variance in the object size collected from the access logs helps to instill enough diversity to benchmark object updates (writes). The updates are randomly pushed onto different sites, and the total system update load is measured in terms of the percentage update requests *U*% compared that to the initial network with no updates. The experimental setup is elaborated in Figure 7.

5.1 Comparative Analysis

For comparison, we have selected eight various types of replica placement techniques. To provide a fair comparison, the assumptions and system parameters are the same in all the approaches. We propose six new and use two existing techniques. They include: efficient branch-and-bound (A-star) technique, sub-optimal A-star based heuristics (SA1, SA2, SA3), bin packing heuristics using local (LMM) and global (GMM) knowledge. For fine-grained replication, the algorithms proposed in [33], [34], and [46] are the only ones that address the problem domain similar to ours. We select from [46] the greedy approach (Greedy) for comparison because it is shown to be the best compared with 4 other approaches (including the proposed technique in [33]); thus, we indirectly compare with 4 additional approaches as well.

Performance metric: The solution quality is measured in terms of network communication cost (RC percentage) that is saved under the replication scheme found by the algorithms, compared to the initial one, *i.e.*, when only primary copies exists.

A-star based technique: The modified A-star based searching technique for the replication problem starts from an assignment *P*, and explores all the *potential* options of assigning an object to a site. We use the following heuristic for the A-star searching technique: Let O_i and S^k represent the set of objects and sites in the system. Let *U* be the set of unassigned objects and *t* be the global minimum of all the replication costs associated with an object. Thus, we define the minimum of such cost as a set: $T=min_{0\le j\le N-1}(t(O_i, S^k))$, $\forall O_i \in U$. For a node *n*, let *mmk*(*n*) define the maximum element of set *T* (the max-min replication cost). *mmk*(*n*) then represents the best possible replica allocation without the unrealistic assumption that every object in *U* can be

replicated to a site in M without a conflict. Thus we gave our heuristic for the replication cost as follows: $h_{cost}(n)=max(0,[mmk(n)-g(n)])$.

A-star based Heuristics: Three heuristics (sub-optimal A-star) algorithms, refereed to hereafter as SA1, SA2, SA3 are proposed. The name SA comes from Sub-optimal Assignments. The basic idea behind these algorithms is that when the search process reaches a certain depth in the search tree, some search path(s) can be avoided (some tree nodes can be discarded) without moving far from the optimal solution. In SA1, when the algorithm selects a node that belongs to level R or below, it generates only the best successors (lowest expansion cost) of it. All the other successors except the best one are discarded. In SA2, when the depth level R is reached for the very first time, all the successors except the minimum cost are discarded among all the nodes marked for expansion. In SA3, the discarding is done similar to SA2 except that now the nodes are removed from the expansion tree (ET). For instance, if n nodes are generated, then all of them are inserted in the ET, and the $n-1$ high cost nodes are discarded. These techniques will not suffer from memory overflow, since at level R, for every node taken out of the ET for expansion, only one node is inserted. Also the running time is reduced by many folds since the algorithm expands/explores less number of nodes when it reaches R. In all the experiments the bound on $R=\lfloor d/2 \rfloor$, where d represents the depth of the tree (number of objects).

Greedy based technique: We modify the greedy approach reported in [46], to fit our problem formulation. The greedy algorithm works in an iterative fashion. In the first iteration, all the M sites are investigated to find the replica location(s) of the first among a total of N objects. Consider that we choose an object i for replication. The algorithm recursively makes calculations based on the assumption that all the users in the system request for object i. Thus, we have to pick a site that yields the lowest cost of replication for the object i. In the second iteration, the location for the second site is considered. Based on the choice of object i, the algorithm now would identify the second site for replication, which, in conjunction with the site already picked, yields the lowest replication cost. Observe here that this assignment may or may not be for the same object i. The algorithm progresses forward till either one of the DRP constraints are violated.

Local Min-Min: Let O_k and S_i represent the set of objects and sites in the system. Let U be the set of unassigned objects to a site S_i . The set U is sorted in the ascending order of the replication cost of the objects

to be assigned to a particular site. Iteratively we choose the head of the list for assignment. If there is a tie among two objects, then the tie is broken by the minimum object size, hence the name Min-Min. Since we do the assignment iteratively for every object and do not consider the effects of the choice of an object to a site with respect to other sites, we call it Local Min-Min (LMM).

Global Min-Min: Let O_k and S_i represent the set of objects and sites in the system. Let U be the set of unassigned objects and k be the global minimum of all the replication costs associated with an object. The minimum of such cost as a set $T=min_{0\leq j\leq N-1}(k(O_k,S_i),\forall O_k \in U$. If during the assignment, the minimum replication cost of an object is the same for two different sites, the tie is broken by the minimum object size. For a node n let $mink(n)$ define the minimum element of set T. Thus $mink(n)$ represents the best minimum replication cost that would occur if object O_k is replicated to a site S_i, *i.e.*, Global Min-Min (GMM).

GRA: In [34], the authors have proposed a genetic algorithm based heuristic called GRA. GRA provides good solution quality, but suffers from slow termination time. This algorithm is chosen since it was the first work that realistically addressed the fine-grained replication on the same problem formulation as taken in this article. Extensive evaluations in [34] strengthen their results.

5.2 Comparative Performance Analysis

We study the behavior of the placement techniques when the number of sites increases (Figure 8), by setting the number of objects to 2000 and the capacity to 15%, while in Figure 9, we study the behavior when the number of objects increase, by setting the number of sites to 200 and the capacity to 20%. In both the experiments we fixed U=10%. We should note here that the space limitations restricted us to include various other scenarios with varying capacity and update ratio. The plot trends were similar to the ones reported in this article. The first observation is that EA, DA and Greedy outperform other techniques by considerable amounts. Second, all branch and bound based techniques and GMM failed to converge to a solution with certain problem instance, depicted by "●" in the plots. Before its failure, A-star showed very high performance even up to 78%. Other notables were SA3 and GRA which gave approximately the same amount of savings. Some interesting observations were also recorded. First, LMM and GMM showed high gain with the initial number of site increase in the system, as much as 23% gain was recorded in case of

GMM with only a 100 site increase. Second, All the A-star based heuristics showed an initial loss in savings. A 19% loss was recorded for SA2 with the initial site increase of 100. Both of the above observations are very much in line of the employed techniques. LMM and GMM show high initial gain since with the increase in the number of sites, the combinations of bins increase, but with later increase the effect is not so observable as all the essential objects are already replicated. A-star heuristics on the other hand can only gain from such a phenomenon when the increase of sites are of high capacity as they rely on the bound *R* of the explored tree. It can be seen from the plot that all the A-star based techniques gain back their lost savings.

To observe the effect of increase in the number of objects in the system, we intentionally chose a smaller setup. The intention was to observe the trends for all the algorithms as much as possible as some techniques fail to yield results as observable from Figure 8. The increase of objects has diverse effects on the system as new read/write patterns emerge and also the capacity of the system increases. An effective algorithm should incorporate both the opposing trends. From the plot, we can observe that the bin packing techniques perform the worst with a loss of nearly 35% in case of LMM. SA3 showed a linear decrease, while the most surprising result came from GRA. It dropped its savings from 58% to 13%. The most stable performance was observable in the case of EA which retained its RC savings throughout. A-star which failed after the first few problem instances in fact showed a gain in savings.

An increase in storage capacities means that a large number of objects can be replicated. Since objects are not equally read intensive, increase in storage capacity has great impact at the beginning, but has little effect after a certain point where the most beneficial ones are already replicated. This is observable in Figure 10, which shows the performance of the algorithms. LMM and GMM once again performed the worst. The gap between all other approaches was reduced to within 10% of each other. EA and DA with an immediate increase (the point where further replicating objects is inefficient) in its RC savings afterward showed constant performance. Further experiments with various update ratios (5%, 10%, and 20%) showed similar plot trends. It is also noteworthy that the increase in capacity from 10% to 17%, resulted in 4 times (on average) more replicas for all the algorithms.

Next, we keep the number of objects and sites constant to observe the effects of the increase in the number of read and update patterns. Increase in the number of reads in the system would mean that there is a need to replicate as many object as possible (closer to the users). A clear classification can be made between the algorithms. EA, DA and Greedy incorporate the increase in the number of reads by replicating more objects and thus savings increase up to 88%. LMM gains the least with a maximum savings of 38%. The rest of the algorithms gave mediocre performance as can be seen from Figure 11. The performance of LMM, GMM and SA2 decreased exponentially to the update ratio (Figure 12). A sub-exponential decrease was also observable in the cases of SA1, SA3 and GRA. EA, DA and Greedy on the other hand showed extreme robustness and retained their initial savings. It is to be noted that the increase in the update ratio, forces the algorithms to replicate objects as close as possible to the primary object site, while decreasing the aggregate distance from the users accessing the objects.

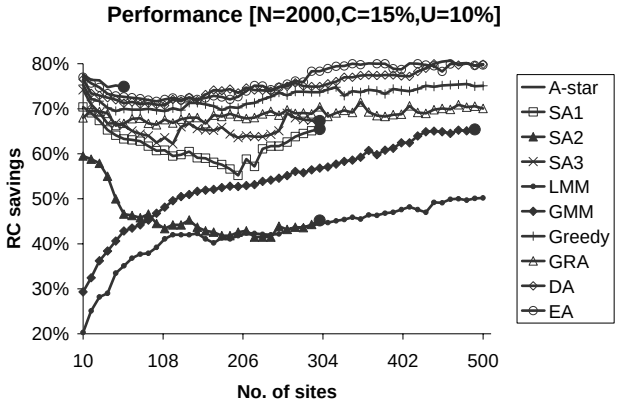

Figure 8: RC savings versus number of sites.

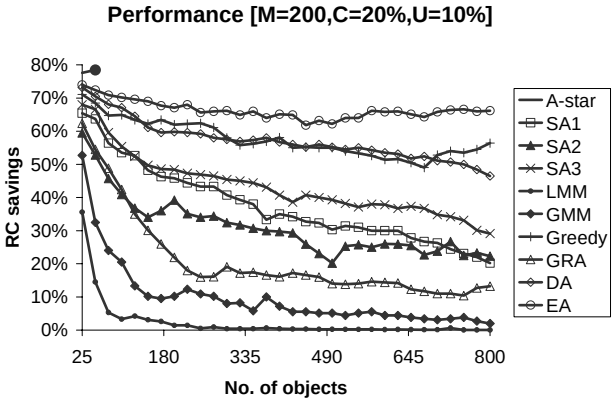

Figure 9: RC savings versus number of objects.

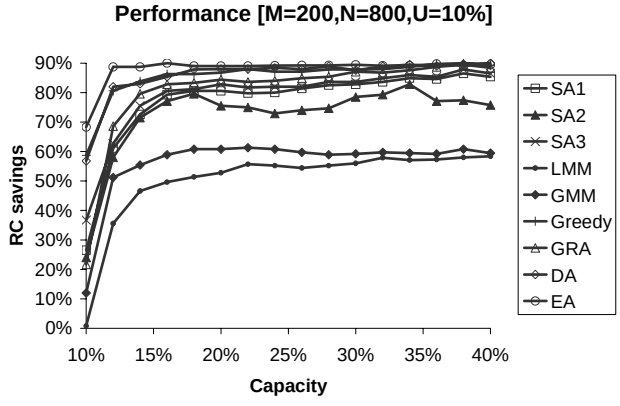

Figure 10: RC savings versus capacity.

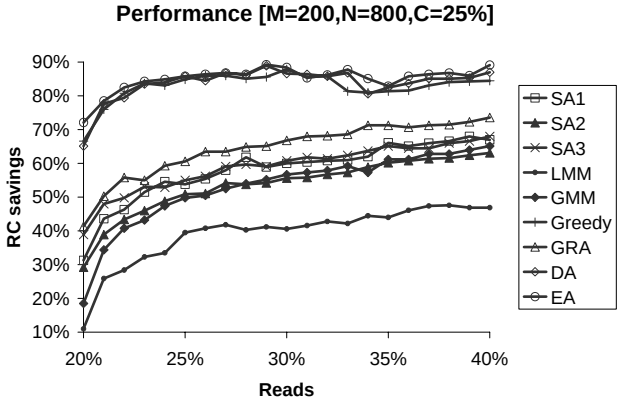

Figure 11: RC savings versus reads.

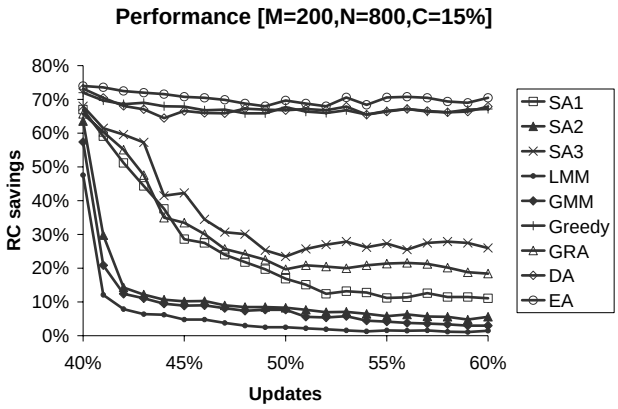

Figure 12: RC savings versus updates.

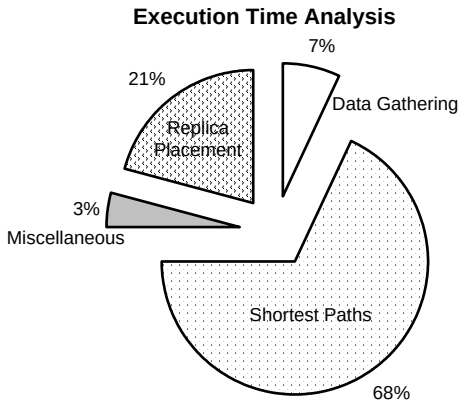

Figure 13: Components determining running time.

Table 5: Average time (seconds) taken to converge to solution under some problem instances.

Problem Size	A-star	SA1	SA2	SA3	LMM	GMM	Greedy	GRA	DA	EA
N=150,M=20	350.13	620.34	420.44	456.23	95.25	88.32	78.26	301.25	**32.44**	**54.57**
N=200,M=50	1356.35	1260.55	1377.31	1389.25	320.51	372.04	156.26	423.51	**62.19**	**118.69**
N=300,M=50	2200.45	1430.56	1808.34	1845.52	360.32	360.29	158.21	526.12	**72.93**	**98.85**
N=300,M=60	2650.32	1670.43	1788.95	1925.21	442.34	442.95	168.26	548.25	**75.43**	**110.91**
N=400,M=100	X	3800.56	3013.53	3125.56	594.18	695.95	198.26	602.45	**111.73**	**138.03**
N=500,M=100	X	4025.56	3526.25	3785.58	693.25	896.54	206.44	625.21	**121.94**	**173.78**
N=800,M=200	X	5002.23	4526.58	4985.26	725.26	989.26	289.26	726.26	**132.91**	**190.54**
N=1000,M=300	X	X	X	X	752.25	1002.26	326.21	892.15	**201.26**	**234.42**
N=1500,M=400	X	X	X	X	958.26	1366.26	541.26	948.42	**208.92**	**245.47**
N=2000,M=500	X	X	X	X	1123.33	X	795.21	1263.56	**312.68**	**345.56**

Lastly, we compare the termination time of the algorithms. Before we proceed, we would like to clarify our measurement of algorithm termination timings. The approach we took was to see if these algorithms can be used in dynamic scenarios. Thus, we gather and process data as if it was a dynamic system. The average breakdown of the execution time of all the algorithms combined is depicted in Figure 13. There 68% of all the algorithm termination time was taken by the repeated calculation of the shortest paths. Data gathering and dispersion, such as reading the read frequencies from our processed log, etc. takes 7% of the total time. Other miscellaneous operations including input/output were recorded to carry 3% of the total execution time. From the plot it is clear that a totally static setup would take no less that 21% of the time depicted in Table 5. Various problem instances were recorded with U=25% and C=20%. Each problem instance represents the average recorded time over all the 45 topologies and 13 various access logs. The entries in bold represent the fastest and the second fastest time recorded over the problem instance. Entries marked with "X" means the algorithm could not converge. It is clear that DA which is the fastest of all the algorithms is 80% faster than

Greedy which outperform all the other non game theoretic techniques. If a static environment was considered, DA with the maximum problem instance would have terminated approximately in 65.67 seconds.

Table 6: Average RC (%) savings under some problem instances.

Problem Size	A-star	SA1	SA2	SA3	LMM	GMM	Greedy	GRA	DA	EA
N=150,M=20	**75.43**	72.26	63.26	75.26	49.21	59.11	69.74	70.46	70.15	73.15
N=200,M=50	**79.43**	75.13	72.36	76.45	45.02	60.29	70.18	73.94	72.66	77.41
N=300,M=50	**70.23**	70.22	68.70	70.01	48.36	62.53	71.29	70.01	68.23	70.11
N=300,M=60	**72.12**	69.80	67.28	70.56	47.31	64.72	69.94	71.66	70.22	71.23
N=400,M=100	X	68.29	65.14	70.26	42.56	64.64	66.07	67.40	69.46	**70.55**
N=500,M=100	X	68.92	66.43	70.10	48.11	66.47	67.62	66.15	70.21	**71.12**
N=800,M=200	X	69.16	68.56	69.25	49.21	65.17	65.91	67.46	69.29	**70.61**
N=1000,M=300	X	X	X	X	50.47	64.98	64.08	69.10	70.16	**71.29**
N=1500,M=400	X	X	X	X	54.95	63.81	70.49	63.59	**72.77**	72.61
N=2000,M=500	X	X	X	X	56.12	X	65.37	67.03	68.63	**69.24**

5.3 Summary of Results

Table 6 shows the quality of the solution in terms of RC percentage for 10 problem instances, each being a combination of various numbers of sites and objects. For each row, the best result is indicated in bold. For all the problem instances certain parameters were kept constant, *i.e.*, C=20%, U=25%. Clearly, the proposed DA and EA approaches outperform other approaches or perform close to the optimal A-star algorithm, which has a very high complexity. In terms of execution time (Table 5), DA is about 80% faster than the Greedy approach that mostly outperforms the rest of the approaches. Several other approaches fail to yield results for larger problem but the DA and EA perform consistently both in terms of the solution quality and the execution time, indicating their scalability. It is to be noted that the performance of DA is slightly lower than EA due to its silent gaming nature, *i.e.*, unless a bid is made the bidders have no knowledge about any entity.

6. Concluding Remarks

This article proposed game theoretical techniques to obtain fast and high quality of solution towards fine-grained web content replication. A resource allocation mechanism was derived that exhibited Nash equilibrium for non-cooperative bidders. The proposed techniques were extensively evaluated against eight other techniques based mainly on branch and bound, bin packing, greed and genetic algorithms. The comparative performance shows that the proposed techniques outperform all the other techniques by considerable amounts both in terms of quality of solution and algorithm termination timings.

References

[1] G. Abdulla, "Analysis and Modeling of World Wide Web Traffic," PhD thesis, Virginia Polytechnic Institute and State University, 1998.

[2] Adero, Available at: http://www.adero.com/.

[3] Y. Amir, "Replication Using Group Communication Over a Partitioned Network," PhD thesis, Hebrew University, Jerusalem, Israel, 1995.

[4] P. Apers, "Data Allocation in Distributed Database Systems," *ACM Trans. Database Systems*, vol. 13, no. 3, Sept. 1988, pp. 263-304.

[5] M. Arlitt and T. Jin, "Workload characterization of the 1998 World Cup Web Site," Tech. report, Hewlett Packard Lab, Palo Alto, HPL-1999-35(R.1), 1999.

[6] V. Arya, N. Garg, R. Khandekar, K. Munagala and V. Pandit, "Local Search Heuristic for k-median and Facility Location Problems*,"* in *ACM Symposium on Theory of Computing*, 2001, pages 21-29.

[7] R. Aumann and M. Maschler, "Game theoretic analysis of a bankruptcy problem from the talmud," *Journal of Economic Theory* vol. 36, pp. 195-213, 1985.

[8] F. Bry and R. Manthey, "Checking consistency of Database Constraints: a Logical Basis," in *Proc. of the 12th International Conference on Very Large Data Bases*, Kyoto, Japan, August 1986, pp. 13-20.

[9] K. Calvert, M. Doar, E. Zegura, "Modeling Internet topology," *IEEE Communications*, vol. 35, no. 6, pp. 160-163, 1997.

[10] C. Ceri, G. Pelagatti, and G. Martella, "Optimal File Allocation in a Computer Network: A Solution based on Knapsack Problem," *Computer Networks*, vol. 6, pp. 345-357, 1982.

[11] M. Charikar, S. Guha, E. Tardos and D. Shmoys, "A Constant-Factor Approximation Algorithm for the K-Median Problem," in *ACM Symposium on Theory of Computing*, 1999, pp. 1-10.

[12] M. Crovella and R. Carter, "Dynamic Server Selection in the Internet," in *Proc. of the 3rd Workshop on the Architecture and Implementation of High Performance Communication Subsystems*, 1995.

[13] W. Chu, "Optimal File Allocation in a Multiple Computer System," *IEEE Trans. Computers*, vol. C-18, no. 10, 1969.

[14] Cisco Systems Inc., Distributed Director, White paper, http://www.cisco.com/warp/public/cc/pd/cxsr/dd/tech/dd_wp.pdf.

[15] K. Dasgupta and K. Kalpakis, "Maintaining Replicated Redirection Services in Web-based Information Systems," in *Proc. of the IEEE Workshop on Internet Applications*, July 2001.

[16] B. Davison, "A Survey of Proxy Cache Evaluation Techniques," in *Proc. of the 4th International Web Caching Workshop*, 1999.

[17] G. Demange, D. Gale and M. Sotomayor, "Multi-item auctions," *Journal of Political Economy* , no. 94, 1986, pp. 836-872.

[18] D. Dias, W. Kish, R. Mukherjee and R. Tewari, "A Scalable and Highly Available Web Server," *in COMPCON*, 1996, pp. 85-92.

[19] K. Eswaran, "Placement of Records in a File and File Allocation in a Computer Network," *Information Processing*, 1974, pp. 304-307.

[20] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator and N. Young, "Competitive paging algorithms," *Journal of Algorithms*, no. 12, vol. 4, 1991 pp. 685-699.

[21] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Trans. Networking*, vol. 9, no. 4, pp. 253-285, 2001.

[22] M. Garey and D. Johnson, *Computers and Intractability*, W.H. Freeman and Co., 1979.

[23] A. Goel, C. Pu and G. Popek, "View Consistency for Optimistic Replication," in *Symposium on Reliable Distributed Systems*, 1998, pp. 36-42.

[24] A. Guyton and M. Schwartz, "Locating Nearby Copies of Replicated Internet Servers," *in Proc. of the ACM SIGCOMM'95 Conf.*, Cambridge, MA, pp. 288-298, Aug. 1995.

[25] A. Hayzelden and J. Bigham, *Software Agents for Future Communications*, Springer Verlag, 2000.

[26] S. Jamin, C. Jin, Y. Jin, D. Riaz, Y. Shavitt and L. Zhang, "On the Placement of Internet Instrumentation," in *Proc. of the IEEE INFOCOM*, 2000.

[27] S. Jamin, C. Jin, T. Kurc, D. Raz and Y. Shavitt, "Constrained Mirror Placement on the Internet," in *Proc. of the IEEE INFOCOM*, 2001.

[28] J. Kangasharju, J. Roberts and K. Ross, "Object Replication Strategies in Content Distribution Networks," in *Proc. of Web Caching and Content Distribution Workshop*, 2001, pp. 455-456.

[29] V. Krishna. *Auction Theory*, Academic Press, San Diego, U.S.A., 2002.

[30] H. Kuhn, "Excerpt from Montmort's Letter to Nicholas Bernoulli," in *Precursors in Mathematical Economics: An Anthology*, ser. Reprints of Scarce Works on Political Economy, W. Baumol and S. Goldfeld, eds., vol. 19, pp. 3-6, 1968.

[31] Y. Kwok, K. Karlapalem, I. Ahmad and N. Pun, "Design and Evaluation of Data Allocation Algorithms for Distributed Database Systems," *IEEE Journal on Selected areas in Communication (Special Issue on Distributed Multimedia Systems)*, vol. 14, no. 7, pp. 1332-1348, Sept. 1996.

[32] B. Lee and J. Weissman, "Dynamic Replica Management in the Service Grid," in Proc. *of IEEE International Symposium on High Performance Distributed Computing*, Oct. 2001.

[33] B. Li, M. Golin, G. Italiano and X. Deng, "On the Optimal Placement of Web Proxies in the Internet," in *Proc. of the IEEE INFOCOM*, March 2000.

[34] T. Loukopoulos, and I. Ahmad, "Static and Adaptive Distributed Data Replication using Genetic Algorithms," Accepted to appear in *Journal of Parallel and Distributed Computing.*

[35] T. Loukopoulos, I. Ahmad, and D. Papadias, "An Overview of Data Replication on the Internet," in *International Symposium on Parallel Architectures, Algorithms and Networks*, 2002, pp. 31-36.

[36] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," in *16th ACM International Conference on Supercomputing*, New York, NJ, June 2002.

[37] R. MacAfee amd J. MacMillan, "Auctions and Bidding," *Journal of Economic Literature,* no. 25, pp. 699-738, 1987.

[38] S. March and S. Rho, "Allocating Data and Operations to Nodes in Distributed Database Design," *IEEE Trans. Knowledge and Data Engineering*, vol. 7, no. 2, November 1995, pp.305-317.

[39] R. Myerson, "Optimal Auction Design," *Mathematics of Operations Research*, no. 6, 1981, pp. 58-73.

[40] J. Nash, "The Bargaining Problem," *Econometrica*, vol. 18, pp. 155–162, 1950.

[41] J. Nash, "Non-Cooperative Games," *Annals of Mathematics*, vol. 54, pp. 286-295, 1951.

[42] J. Nash, "Equilibrium Points in N-person Games," in *Proc. of the National Academy of Sciences*, vol. 36, pp. 48-49, 1950.

[43] J. Nash and L. Shapley, "A Simple Three-person Poker Game," *Annals of Mathematical Studies*, vol. 24, pp. 105-106, 1950.

[44] Project Nice, Available at: http://www.cs.umd.edu/projects/nice/.

[45] M. J. Osborne and A. Rubistein, *A Course in Game Theory*, MIT Press, Massachusetts, U.S.A., 1994

[46] L. Qiu, V. Padmanabhan and G. Voelker, "On the Placement of Web Server Replicas," in *Proc. of the IEEE INFOCOM*, Anchorage, AK, USA April 2001.

[47] M. Rabinovich, "Issues in Web Content Replication," *Data Engineering Bulletin,* vol. 21 no.4, 1998.

[48] M. Rabinovich and A. Aggarwal, "RaDaR: A Scalable Architecture for a Global Web Hosting Service," *Computer Networks*, vol. 31, no. 11-16, 1999, pp. 1545-1561.

[49] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, J. Kubiatowicz, "Maintenance-free Global Storage," IEEE Internet Computing, vol. 5, no. 5, pp. 40-49, 2001.

[50] P. Rodriguez and E. Biersack, "Dynamic Parallel-Access to Replicated Content in the Internet," *IEEE/ACM Trans. Networking*, vol. 10, no.4, August 2002.

[51] M. Sayal, Y. Breitbart, P. Scheuermann and R. Vingralek, "Selection Algorithms for Replicated Web Servers," in *Proc. of the 1998 Workshop on Internet Server Performance*, June 1998.

[52] S. So, I. Ahmad and K. Karlapalem, "Response Time Driven Multimedia Data Objects Allocation for Browsing Documents in Distributed Environments," *IEEE Trans. Knowledge and Data Engineering*, vol. 11, no.3, 1999, pp. 386-405.

[53] V. Vazirani, *Approximation Algorithms*, Springer-Verlag, Berlin, Germany, 2001.

[54] W. Vickery, "Counter-speculation, auctions and competitive sealed tenders," *Journal of Finance*, pp. 8-37, 1961

[55] A. Vigneron, L. Gao, M. Golin, G. Italiano and B. Li, "An Algorithm for Finding a K-Median in a Directed Tree," *Information Processing Letters*, vol. 74 , pp. 81-88, 2000.

[56] J. von Neumann, "Zur Theorie der Gesellschaftsspiele," *Mathematische Annalen*, vol. 100, pp. 295-

320, 1928.

[57] W. Yuen, C. Sung and W. Wong, "Optimal Price Decrement Strategy for Dutch Auctions," *Communications in Information and Systems*, vol. 2, no. 4, 2002.

Appendix A

Let τ_i be a portion of a bidder B_i's evaluation v_i. For understanding τ_i can be viewed as a fine tuning knob. We can then represent the bids in the form of: $\hat{B}_i = (\tau_i \times v_i)$. Let y_i be the probability that a bidder B_i wins the auction. Let $U_i \in \Re_i^n$ represent the utility function (expected payoff), and let t_i be the amount of monies paid to the vendor. Thus, we can write the utility function as: $U_i(y_i, \hat{v}_i) = \hat{v}_i \times y_i - t_i$. Thus, the outcome in terms of probabilities can be defined as: 0 if $\hat{B}_i < Max(\hat{B}) \forall j$, 1 if $\hat{B}_i > \hat{B}_j \forall i \neq j$, and 0.5 if $\hat{B}_i = \hat{B}_j$. Thus, from the utility functions and the outcome probabilities, any player's strategy (bid) would be to maximize the expected gain as: $\hat{B}_i = \max_{\tau_i}(v_i - \hat{B}_i) \times p_i(\hat{B}_i > \sum_{i=2}^{n} \hat{B}_i) \times v_i$. With simple algebra we can get: $v_i(1-\tau_i) \times \tau_i \big/ \sum_{j=1}^{n} \tau_j$, $j \neq i$. ■

Appendix B

Assume for the time being a two player iterative Dutch auction. Then, the game conforms to the strategy $\hat{\beta} = (\hat{B}_1, \hat{B}_2, U_1, U_1)$. Without the loss of generality we assume that $U_i \in R^2$. Thus for a two player game the choice of utility for player 1 would be:

$$U_1(B_1, B_2) = \begin{cases} \hat{v}_1 - \hat{v}_1(\hat{B}_1) & \hat{B}_1 \geq \hat{B}_2 \\ 0 & \hat{B}_1 < \hat{B}_2 \end{cases} \quad \text{(B1)}$$

, and the choice of utility for player 2 would be:

$$U_2(\hat{B}_2, \hat{B}_1) = \begin{cases} \hat{v}_2 - \hat{v}_2(\hat{B}_2) & \hat{B}_2 > \hat{B}_1 \\ 0 & \hat{B}_2 \leq \hat{B}_1 \end{cases} \quad \text{(B2)}$$

Let $(\hat{B}_1, \hat{B}_2)$ represent Nash equilibrium. If so, then the player 1 would win the entity. Assume that the player 1 does not win the entity, then this would imply that $\hat{B}_1 < \hat{B}_2$, and the payoff from (B1) would be zero. If $\hat{B}_1 < \hat{B}_2$, then $\hat{B}_2 < \hat{v}_1$, this tells us that the bid $\hat{B}_2$ is better then bid $\hat{B}_1$. If this is the case then the best response of player 1 against any bid from player two would be to bid $\hat{B}_2$, but from (B2) that would give a payoff of zero, and there already exists a positive payoff by bidding $\hat{B}_1$, a contradiction. For the second player if $\hat{B}_1 > \hat{B}_2$ does not conform to equilibrium, then player 1 would in order to increase its payoff bid $\hat{v}_2 - \hat{v}_2(\hat{B}_2)$ and not $\hat{v}_1 - \hat{v}_1(\hat{B}_1)$, but that is a contradiction since it is the payoff of player 2 and not of player 1. Thus, by best response towards player 2's bid $\hat{B}_1 \geq \hat{B}_2$ should hold and the equilibrium would be when $\hat{B}_1 = \hat{B}_2$. Extending this to n bidders, a bidder B_i's optimal strategy would be to increase its corresponding bid, *i.e.*, $\hat{B}_i \geq \max\{\hat{B}_j\} \forall j \neq i$, a Nash equilibrium. ■

Appendix C

The probability that a bid $\hat{B}_i$ with evaluation v_i would be the highest among all the n bids is: $\int_{v_i}^{\hat{B}_i} f(v_i) dv_i$ [37]. Solving the integral we get: $\bar{B}_i - v_i / b_i - v_i$. Thus for a bidder to maximize its chances to win the bid would be: $(v_i - \bar{B}_i)(\bar{B}_i - v_i / b_i - v_i)^{n-1}$. Solving for $\partial / \partial \bar{B}_i = 0$, we obtain: $\bar{B}_i = v_i + v_i(n-1)/n$. ■

Appendix D

We examine the iterative English auction as a set of sequential games. Let $\mu(t)$ represent a single iteration at a given instance t. From RAM, we know that $\mu(t)$ would contain $\hat{\beta}$, where $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, ..., \hat{\beta}_n)$. Let $p_i(\beta_i(t))$, represent the winning probability of a bidding strategy by a bidder B_i at time t. Thus, if we are given the probability of winning and the strategy of the bid, we can easily extract the evaluation as: $\hat{v}_i(\beta_i(t), p(\beta_i(t)))$.

Lemma 1 suggests that there exists an optimal strategy for a bidder B_i at a given instance of time t. Without the loss of generality we assume that in order to fulfill the optimal strategy, B_i submits a bid $\widehat{B}_i$ at time t, as the best possible response to the rest of the bidders. Therefore, if a bidder B_i bids the winning bid, then the winning bid would be: $\hat{v}_i(\hat{\beta}_i(t), p(\hat{\beta}_i(t)))$, *i.e.*,

$\hat{v}_i(\hat{\beta}_i(t), p_i(\hat{\beta}_i)) \geq \hat{v}_{i-1}(\hat{\beta}_{i-1}(t), p_{i-1}(\hat{\beta}_{i-1}(t))) \geq .... \geq \hat{v}_0(\hat{\beta}_0(t), p_0(\hat{\beta}_0(t)))$, a Nash equilibrium. ■

Appendix E

In our proposed technique the expected revenue for the vendor from [39] is: $E[R] = \sum_{i \in n} E[\ell_i(G_i)]$ (E1)

On the other hand, the expected *ex ante* payment by the bidder will be: $E[\ell_i(G_i)] = \int_0^{b_i} \ell_i(x_i) f_i(x_i) dx_i$ (E2)

Note that in Equation E2, the lower bound of the integral starts from 0. This bound ensures that only positive revenue is evaluated, it is also true in our problem definition, since the replication cost will always be positive. We also want to clarify that x_i is nothing more than an arbitrary value in the interval G_i. From the *Revenue Equivalence* relationship [37], $\forall\, i$ and $x_i \in G_i$, the factor $\ell_i(x_i)$ is equivalent to :

$$\ell_i(0) + \kappa_i(x_i) x_i - \int_0^{x_i} \kappa_i(t_i) dt_i \,. \tag{E3}$$

Expanding Equation A3 and substituting it in Equation E2, we get:

$$E[\ell_i(G_i)] = \ell_i(0) + \int \kappa_i(x_i) x_i f_i(x_i) dx_i - \int_0^{b_i} \int_0^{x_i} \kappa_i(t_i) f_i(x_i) dt_i dx_i \tag{E4}$$

Thus, the expected payment of the bidder reduces to:

$$E[\ell_i(G_i)] = \ell_i(0) + \int_{G_i} \left(x_i - 1 - F_i(x_i)/f_i(x_i)\right) \kappa_i(x_i) f(x_i) dx_i \tag{E5}$$

In Equation E5 we define: $\pi_i(x_i) \equiv x_i - 1 - F_i(x_i)/f_i(x_i)$. Then, $\forall\, i$, we have to evaluate $E[\pi_i(x_i)] = 0$. According to the arguments in [29], the true value of x_i reduces to: $\pi_i(x_i) \equiv x_i - 1/\varsigma_i(x_i)$, where $\varsigma_i(x_i) = f/1 - F_i$. Therefore, the vendor in order to get the maximum revenue should maximize:

$$\sum_{i \in n} m_i(0) + \int_G \left(\sum_{i \in n} \varsigma_i(x_i) G_i(x_i)\right) f(x) dx \tag{E6}$$

If $\varpi = \sum_{i \in n} \varsigma_i(x_i) G_i(x_i)$ then, from Equation E6 ϖ is equivalent to our proposed resource allocation mechanism, since it optimizes nothing more than the evaluations of the bidders (bidder s' point of view of optimality). We can thus extract the following two facts:

1. κ_i would be in favor of the bidder if and only if $\varsigma_i(x_i) = \max_{j \in n} \varsigma_i(x_j)$, which implies: $\kappa_i(x) > 0$, (E7)

2. and the amount that the bidder would dispose off would be: $\ell_i(x) = \kappa_i(x) x_i - \int_0^{x_i} \kappa_i(z_i, x_{i-1}) dz_i$. (E8)

Therefore all we have to do now is to prove that Equations A7 and A8 do indeed do indeed conform to the optimal solution. The basic idea is to have a non-decreasing function of ς_i. We prove this by contradiction assumes that $z_i < x_{i-1}$ then by Equation E7, we have $\varsigma_i(z_i) < \varsigma_i(x_i)$. This would make for all x_{-i} the allocation $\kappa_i(z_i, x_{i-1}) < \kappa_i(x_i, x_{i-1})$, which by Equation E8 is not true. From Equation E7, it is also clear that $\kappa_i(0, x_{i-1}) = 0$. Thus, for all x_{-i}, $m_i(0) = 0$.

Once we have established the fact that our resource allocation mechanism provides a fair ground for the bidders to optimize their strategies, we now proceed to prove that this optimization also works in the two proposed techniques. Assume that EA does not exhibit Nash equilibrium. If so, then there would be a bidder B_i that has a bidding strategy β_i, which is always blocked by n-1 bidder(s). If this is the case then the bids by other n-1 bidders would also be blocked (domino effect). If not, then there would be some bidder B_j who wins the session and thus his evaluation is optimal. This implies that there is Nash equilibrium. Assuming that we have established the fact that the equilibrium exists then this would only hold if there exists an optimal strategy for each bidder B_i. But from Appendix A the optimal strategy is no more then to minimize the replication cost of the objects won by a bidder. Similar arguments hold for DA.

Appendix F

The determinance of the optimal number of agents in a system is only possible through experimental results. The basic idea is to choose the number of agents in a system such that the algorithm converges to the solution quickly with high quality of solution (vertical line in Figure A1 for a problem size of N=2000,

M=500, C=20%, U=25%). It is also interesting to note that the running time of the algorithm increases with increasing number of agents. Table A1 shows the number of agents used in our experimental results. It is to be noted that this determinance is only possible using EA and not DA which is a silent auction [29].

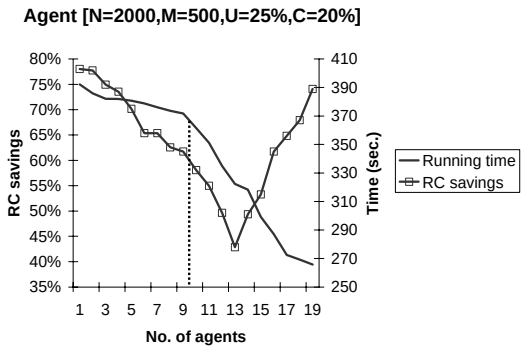

Figure A1: Determinance of the no. of agents for a problem instance.

Table A1: No. of Agents used.

Problem Size	No. of Agents
N=150,M=20	2
N=200,M=50	2
N=300,M=50	2
N=300,M=60	3
N=400,M=100	3
N=500,M=100	4
N=800,M=200	6
N=1000,M=300	6
N=1500,M=400	8
N=2000,M=500	10

Appendix G

Our resource allocation mechanism is defined as a game of incomplete information. Assume that a vendor brings a set of identical objects for auction. Let there be n risk-neutral [37] bidders B_n, where $n < M$. The bidder B_i's evaluations $\hat{v}$ are assumed to be private and distributed over an interval $G_i = [-w_i, +w_i] \in R^n$ according to some distribution function $f(\bar{v})$. In this paper the distribution function is assumed to be uniform. To capture the core of the game, the product of the set of bidders has to be defined. The core is necessary since it holds all the possible strategies from all the players and is the heart of the resource allocation mechanism. Based on the guidelines on how to design optimal auction schemas [39], we define the core as: $G = \times \sum_{i=1}^{n} G_i$. Thus, if we want to extract a specific bidder's set of values, we can do so by taking a projection as: $G_{-i} = \times_{i \neq j} G_j$.

Based on the above information, we can define our auction mechanism as a triplet (B, δ, γ). The first attribute is the set of all the bids $\bar{B}_i$ for each bidder B_i. The second attribute is the allocation rule, which is in fact a mapping of the set of bids and the probability distribution (ψ) of the bidders B_i ($\delta : B \rightarrow \psi$). Thus, given a bid $\bar{B}_i$, the function $\delta_i(\bar{B}_i)$ would project the winning probability for the bidder B_i. The third

attribute defines the payment rule(s). All such rules (γ) are pre-agreed before the start of a session or can be negotiated for the entire auction. We define γ as a mapping of the bids B to a hyper structure ($\gamma : B \to R^n$). Each face of this hyper structure represents the mapping of a particular bidder to its payment options. Then, given a bid $\bar{B}_i$, $\gamma_i(\bar{B}_i)$ identifies the exact amount payment that is to be made by the bidder B_i .

For simplicity, assume that for all bidder's i , the possible bids B_i are auto mapped to G_i . This mapping now allows δ to be defined as $\delta_i(\bar{B}_i) = 1$ (win), if $\bar{B}_i > \max_{j \neq i} \bar{B}_j$ and $\delta_i(\bar{B}_j) = 0$ (loss) for all $j \neq i$. If a tie occurs, then it can be broken by awarding the set of items to the bid $\bar{B}_i, i < j$.

If there exists an equilibrium for such a mechanism, then the set of strategies of each bidder B_i would form an n-tuple $\beta_i : [-w_i, +w_i] \to B_i$. Given the strategies β_{-i} of all the other $n-1$ bidders competing in a non-cooperative environment, an equilibrium will exist for i , if $\beta_i(\hat{v}_i)$ is maximized. Maximizing $\beta_i(\hat{v}_i)$ would maximize i 's expected payoff (a natural equilibrium [39]).

The above described mechanism conforms to a game that is complicated for the simple reason that there is no underlying assumption on the bids and the bidding style. We will now simplify this mechanism to a class of bids, where the set of bids $\bar{B}_i$ are the same as the set of values G_i , i.e. $\bar{B}_i = G_i$. This simplification reduces the complexity of the game by considerable amounts. Now, each bidder would only report a two-tuple entity (κ, ℓ) , where $\kappa : G \to \psi$, and $\ell : G \to R^n$. From this two-tuple entity, we can now extract the probability of winning as: $\kappa_i(\hat{v}_i)$, and the amount payable as: $\ell_i(\hat{v}_i)$. It is easy to reduce Lemma 1 to conform to the simpler form of the game. What we are really interested is in knowing that if all the bidders use this two-tuple entity and forms an equilibrium, this equilibrium would reveal the bidder's true value [39].

For the define resource allocation mechanism and its equilibrium, we can rewrite the equilibrium as: $\kappa : G \to \psi$ and $\ell : G \to R^n$. The two forms of game, i.e. the triplet and the two-tuple entity games can be equated by revisiting their definitions as: $\kappa_i(\hat{v}_i) = \delta_i(\hat{B}_i)$ and $\ell_i(\hat{v}_i) = \gamma_i(\hat{B}_i)$.

We are now ready to define our multi-item resource allocation mechanism as an auction game played between n bidders and P vendors.