World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:6, No:12, 2012

A Frugal Bidding Procedure for Replicating WWW Content

Samee Ullah Khan and Cemal Ardil

***Abstract*—**Fine-grained data replication over the Internet allows duplication of frequently accessed data objects, as opposed to entire sites, to certain locations so as to improve the performance of large-scale content distribution systems. In a distributed system, agents representing their sites try to maximize their own benefit since they are driven by different goals such as to minimize their communication costs, latency, etc. In this paper, we will use game theoretical techniques and in particular auctions to identify a bidding mechanism that encapsulates the selfishness of the agents, while having a controlling hand over them. In essence, the proposed game theory based mechanism is the study of what happens when independent agents act selfishly and how to control them to maximize the overall performance. A bidding mechanism asks how one can design systems so that agents' selfish behavior results in the desired system-wide goals. Experimental results reveal that this mechanism provides excellent solution quality, while maintaining fast execution time. The comparisons are recorded against some well known techniques such as greedy, branch and bound, game theoretical auctions and genetic algorithms.

***Keywords*—**Internet, data content replication, static allocation, mechanism design, equilibrium.

I. Introduction

IN the Internet a magnitude of heterogeneous entities (e.g. providers, servers, commercial services, etc.) offer, use, and even compete with each other for resources. The Internet is emerging as a new platform for distributed computing and brings with it problems never seen before. New solutions should take into account the various new concepts derived from multi-agent systems in which the agents cannot be assumed to act in accordance to the deployed algorithm. In a heterogeneous system such as the Internet entitles act selfishly. This is obvious since they are driven by different goals such as to minimize their communication costs, latency, etc. Thus, one cannot assume that agents would follow the protocol or the algorithm; though they respond to incentives (e.g. payments received for compensation).

In this paper, we will use game theoretical techniques and in particular auctions to identify a bidding mechanism that encapsulates the selfishness of the agents, while having a controlling hand over them. This work is inspired from the work reported in [46] and [54]. In essence, game theory is the study of what happens when independent agents act selfishly. A bidding mechanism asks how one can design systems so that agents' selfish behavior results in the desired system-wide goals.

In this paper, we will apply the derived mechanism to the fine grained data replication problem (DRP) over the Internet. This problem strongly conforms to the selfish agents' notion and has a wide range of applications. Replication is widely used to improve the performance of large-scale content distribution systems such as the CDNs [50]. Replicating the data over geographically dispersed locations reduces access latency, network traffic, and in turn adds reliability, robustness and fault-tolerance to the system. Discussions in [14], [17], [39], [40], [47], etc. reveal that client(s) experience reduced access latencies provided that data is replicated within their close proximity. However, this is applicable in cases when only read accesses are considered. If updates of the contents are also under focus, then the locations of the replicas have to be: 1) in close proximity to the client(s), and 2) in close proximity to the primary (assuming a broadcast update model) copy. For fault-tolerant and highly dependable systems, replication is essential, as demonstrated in a real world example of OceanStore [50]. Therefore, efficient and effective replication schemas strongly depend on how many replicas to be placed in the system, and more importantly where. Needless to say that our work differs form the existing techniques in the usage of game theoretical techniques. To the best of the authors' knowledge this is the very first work that addresses the problem using such techniques.

The major results of this paper are as follows:

1. We derive a specialized auction mechanism. This mechanism allows selfish agents to compete in a non-cooperative environment.
2. We investigate this auction mechanism, provide some useful properties and identify the necessary conditions of optimality.
3. As an application we employ the derived mechanism to the DRP. We perform extensive experimental comparisons against some well known techniques, such as greedy, branch and bound, genetic and game theoretical auctions.

S. U. Khan is with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58102, USA (phone: 701-231-7615; fax: 701-231-8677; e-mail: samee.khan@ ndsu.edu).

C. Ardil is with the National Academy of Aviation, Baku, Azerbaijan, (e-mail: cemalardil@gmail.com).

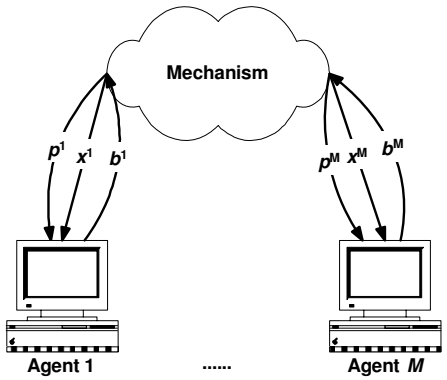

Fig. 1 Frugal bidding mechanism

TABLE I
NOTATIONS AND THEIR MEANINGS

Symbols	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
SGRG	Self Generate 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 Uniform Distribution
SGFCGRD	Self Generated Fully Connected Graphs Random Distribution
SGRGLND	Self Generated Random Graphs Lognormal Distribution
DRP	Data replication problem
OTC	Object transfer cost (network communication cost)

The remainder of this paper is organized as follows. Section II describes the resource allocation mechanism. Section III formulates the DRP. Section IV concentrates on modeling the resource allocation mechanism for the DRP, followed by theoretical proofs in Section V. The experimental results, related work and concluding remarks are provided in Sections VI, VII and VIII, respectively.

II. The Frugal Bidding Procedure

We approach the mechanism (Fig. 1) design in a stepwise fashion.

The Basics: The mechanism contains M agents. Each agent i has some private data $t^i \in \mathbf{R}$. This data is termed as the agent's *true data* or *true type*. Only agent i has knowledge of t^i. Everything else in the mechanism is public knowledge. Let t denote the vector of all the true types $t = (t^1 \ldots t^M)$.

Communications: Since the agents are selfish in nature, they do not communicate to the mechanism the value t^i. The only information that is relayed to the mechanism is the corresponding bid b^i. Let b denote the vector of all the bids ($(b = (b^1 \ldots b^M))$), and let b^{-i} denote the vector of bids, not including agent i, *i.e.*, $b^{-i} = (b^1 \ldots b^{i-1}, b^{i+1}, \ldots b^M)$. It is to be understood that we can also write $b = (b^{-i}, b^i)$.

Components: The mechanism has two components 1) the algorithmic output $x(\cdot)$, and 2) the payment mapping function $p(\cdot)$.

Algorithmic output: The mechanism allows a set of outputs X, based on the output function which takes in as the argument, the bidding vector, *i.e.*, $x(b) = \{x^1(b), \ldots, x^M(b)\}$, where $x(b) \in X$. This output function relays a unique output given a vector b. That is, when $x(\cdot)$ receives b, it generates an output which is of the form of allocations $x^i(b)$. Intuitively it would mean that the algorithm takes in the vector bid b and then relays to each agent its allocation. Many sophisticated allocation outputs can be constructed, but in this paper we choose to have a simple, intuitive and an optimal allocation.

Monetary cost: Each agent i incurs some monetary cost

$c^i(t^i,x)$, *i.e.*, the cost to accommodate the allocation $x^i(b)$. This cost is dependent upon the output and the agent's private data.

Payments: To offset c^i, the mechanism makes a payment $p^i(b)$ to agent i. An agent i always attempts to maximize its profit (utility) $u^i(t^i,b) = p^i(b) - c^i(t^i,x)$. Each agent i cares about the other agents' bid only insofar as they influence the outcome and the payment. While t^i is only known to agent i, the function c^i is public.

Bids: Each agent i is interested in reporting a bid b^i such that it maximizes its profit, regardless of what the other agents bid, *i.e.*, $u^i(t^i,(b^{-i},t^i)) \geq u^i(t^i,(b^{-i},b^i))$ for all b^{-i} and b^i.

A closer look at the bids reveal that the agents are posting bids that are dominant in nature. We state the following lemma from literature which says that truth-telling are dominant strategies, *i.e.*, if all the agents report the exact worth of an object, it conforms to dominating strategies.

Lemma 1: *Truth-telling is a dominant strategy.*

Proof: [46].

The Mechanism: We now put all the pieces together. A mechanism m consists of a pair $m = (x(b),p(b))$, where $x(\cdot)$ is the output function and $p(\cdot)$ is the payment mapping function. The objective of the mechanism is to select an output x, that optimizes a given objective function $g(b,x)$.

Below we identify the desired characteristics of the mechanism.

Truthfulness: We say that an output function admits truthful payments if there exists a payment mapping function $p(\cdot)$ such that the mechanism m is truthful. Using Lemma 1, this would transform to: a mechanism m that is implemented using dominant strategies ($m = (x(t),p(t))$).

Voluntary participation: A mechanism is characterized as a voluntary participation mechanism if for every agent i, $u^i(t^i,(b^{-i},t^i)) \geq 0$, *i.e.*, no agent incurs a net loss.

In recent times, such mechanisms have been applied to the scheduling problems [2], [13], [46], etc. The only practical work that can be found in the literature is reported in [13]. However, that work is the exact implementation of the work reported in [46]. Moreover, the authors in [13] have failed to reason why the implementation works to begin with. It is to be noted that a truthful mechanism strongly relies on the payment function, *i.e.*, the agents are forced to tell the truth because telling a lie gives them no greater profit. Therefore, they are better off telling the truth. Our work differs from others in that we give a concrete application of the truthful mechanism, and concentrate on the payment function. We prove that our approach results in payments that are the exact representations of the monetary costs, *i.e.*, the payments are **frugal**.

Objective: The mechanism defined above leaves us with the following two optimization problems:

1. Identify a strategy that is dominant to each agent i.
2. Identify a payment mapping function that is truthful, and frugal.

III. Description of the Data Replication Problem

Consider a distributed system comprising 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 (in simple data units e.g. blocks), 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 a positive integer $c(i,j)$ 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 common assumption (e.g. see [14], [17], [40], [47], 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 DRP under consideration, we are interested in minimizing the total network transfer cost due to object movement, *i.e.* the Object Transfer Cost (OTC). The communication cost of the control messages has minor impact to the overall performance of the system, therefore, we do not consider it in the transfer cost model, but it is to be noted that incorporation of such a cost would be a trivial exercise. There are two components affecting OTC. The first component of OTC is due to the read requests. Let R_k^i denote the total OTC, due to S^is' reading requests for object O_k, addressed to the nearest site NN_k^i. This cost is given by the following equation:

$$R_k^i = r_k^i o_k c(i, NN_k^i), \quad (1)$$

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

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

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 OTC, denoted as $C_{overall}$, due to reads and writes is given by:

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

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. Sites which are not the *replicators* of object O_k create OTC 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 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*: $X_{P_k k} = 1 \quad \forall (1 \leq k \leq N)$.

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 OTC of the system. In the generalized case, the DRP has been proven to be NP-complete [40].

IV. Mechanism Applied to the DRP

We follow the same pattern as discussed in Section II.

The Basics: The distributed system described in Section III is considered, where each site is represented by an agent, *i.e.*, the mechanism contains M agents. In the context of the DRP, an agent holds two key elements of data a) the available site capacity ac^i, and b) the cost to replicate ($RC_k^i = R_k^i + W_k^i$) an object k to the agent's site i. There are three possible cases:

1. DRP [π]: where each agent i holds the cost to replicate $RC_k^i = t^i$ associated with each object k, where as the available site capacity and everything else is public knowledge.
2. DRP [σ]: where each agent i holds the available site capacity $ac^i = t^i$, where as the cost to replicate and everything else is public knowledge.
3. DRP [π,σ]: where each agent i holds both the cost to replicate and the site capacity $\{RC_k^i, ac^i\} = t^i$, where as everything else is public knowledge.

Intuitively, if agents know the available site capacities of other agents, that gives them no advantage whatsoever. However, if they come about to know their replication cost then they can modify their valuations and alter the algorithmic output. It is to be noted that an agent can only calculate the replication cost via the frequencies of reads and writes. Everything else such as the network topology, latency on communication lines, and even the site capacities can be public knowledge. Therefore, DRP[π] is the only natural choice.

Communications: The agents in the mechanism are assumed to be selfish and therefore, they project a bid b^i to the mechanism. In reality the amount of communications made are immense. This fact was not realized in [13], where the authors assume superfluous assumptions on the implementation. In the later text we will reveal how to cope with this dilemma.

Components: The mechanism has two components 1) the algorithmic output $x(\cdot)$, and 2) the payment mapping function $p(\cdot)$.

Algorithmic output: In the context of the DRP, the algorithm accepts bids from all the agents, and outputs the maximum beneficial bid, *i.e.*, the bid that incurs the minimum replication cost overall (Equation 3). We will give a detailed description of the algorithm in the later text.

Monetary cost: When an object is allocated (for replication) to an agent i, the agent becomes responsible to entertain (read and write) requests to that object. For example, assume object k is replicated to agent i. Then the amount of traffic that the agent has to entertain due to the replication of object k is exactly equivalent to the replication cost, i.e., $c^i = RC_k^i$. This fact is easily deducible from Equation 4.

Payments: To offset c^i, the mechanism makes a payment $p^i(b)$ to agent i. This payment is equivalent to the cost it incurs to replicate the object, i.e., $p^i(b) = c^i$. The readers would immediately note that in such a payment agent i can never get a profit greater than 0. This is exactly what we want. In a selfish environment, it is possible that the agents bid higher than the true value, the mechanism creates an illusion to negate that. By compensating the agents with the exact amount of what the cost occurs, it leaves no room for the agents to overbid or underbid (in the later text we will rigorously prove the above argument). Therefore, the voluntary characteristic of the mechanism now becomes a strongly voluntary and we quote from the literature the following definition.

Definition 1: *A mechanism is characterized as a strongly*

Frugal Auction (FA) Mechanism
Initialize:
01 LS, L^i, T_k^i, M, MT
02 WHILE $LS \neq$ NULL **DO**
03 $OMAX$ = NULL; MT = NULL; P^i = NULL;
04 **PARFOR** each $S^i \in$ LS **DO**
05 **FOR** each $O_k \in L^i$ **DO**
06 T_k^i = compute (t^i); /*compute the valuation corresponding to the desired object*/
07 **ENDFOR**
08 $t^i = \text{argmax}_k(T_k^i)$;
09 **SEND** t^i to M; **RECEIV**E at M t^i in MT;
10 **ENDPARFOR**
11 OMAX = $\text{argmax}_k(MT)$; /*Choose the global dominate valuation*/
12 P^i = 1/OMAX; /*Calculate the payment*/
13 **BROADCAST** $OMAX$;
14 **SEND** P^i to S^i; /*Send payments to the agent who is allocate the object OMAX*/
15 Replicate O_{OMAX};
16 $ac^i = ac^i - o_k$; /*Update capacity*/
17 $L^i = L^i - O_k$; /*Update the list*/
18 **IF** L^i = NULL **THEN SEND** info to M to update $LS = LS - S^i$; /*Update mechanism players*/
19 **PARFOR** each $S^i \in$ LS **DO**
20 Update NN^i_{OMAX} /*Update the nearest neighbor list*/
21 **ENDPARFOR** /*Get ready for the next round*/
22 ENDWHILE

Fig. 2 Frugal Auction (FA) Mechanism

voluntary participation mechanism if for every agent i, $u^i(t^i,(b^{-i},t^i)) = 0$ [54].

We want to emphasis that each agent's incentive is to replicate objects so that queries can be answered locally, for the sake of users that access the agent's site. If the replicas are made available elsewhere, the agent may lose the users, as they might divert their accesses to other sites.

Bids: Each agent i reports a bid that is the direct representation of the true data that it holds. Therefore, a bid b^i is equivalent to $1/RC_k^i$. That is, the lower the replication cost the higher is the bid and the higher are the chances for the bid b^i to win.

In essence, the mechanism $m(x(b),p(b))$, takes in the vector of bids b from all the agents, and selects the highest bid. The highest bidder is allocated the object k which is added to its allocation set x^i. The mechanism then pays the bidder p^i. This payment is equivalent to the cost incurred due to entertain requests from object k by users. The mechanism is given in Fig. 2.

Description of Algorithm: We maintain a list L^i at each server. This list contains all the objects that can be replicated by agent i onto site S^i. We can obtain this list by examining the two constraints of the DRP. List L^i would contain all the objects that have their size less then the total available space ac^i. Moreover, if site S^i is the primary host of some object k', then k' should not be in L^i. We also maintain a list LS containing all sites that can replicate an object, *i.e.*, $S^i \in LS$ if $L^i \neq$NULL. The algorithm works iteratively. In each step the mechanism asks all the agents to send their preferences (first **PARFOR** loop). Each agent i recursively calculates the true data of every object in list L^i. Each agent then reports the dominant true data (line 09) to the mechanism. The mechanism receives all the corresponding entries, and then chooses the globally dominant true data. This is broadcasted to all the agents, so that they can update their nearest neighbor table NN_k^i, which is shown in Line 20 (NN^i_{OMAX}). The object is replicated and the payment is made to the agent. The mechanism progresses forward till there are no more agents interested in acquiring any data for replication (Line 18).

The above discussion allows us to deduce the following two results about the mechanism.

Theorem 1: *FA requires* $O(MN^2)$ *time.*

Proof: The worst case scenario is when each site has sufficient capacity to store all objects. In that case, the while loop (Line 02) performs MN iterations. The time complexity for each iteration is governed by the two **PARFOR** loops (Lines 04 and 19). The first loop uses at most N iterations, while the send loop performs the update in constant time. Hence, we conclude that the worst case running time of the mechanism is $O(MN^2)$. ■

Theorem 2: *In the worst case FA uses* $O(M^3N)$ *messages.*

Proof: First we calculate the number of messages in a single iteration. First, each agent sends its true data to the mechanism, which constitutes M messages. Second, the mechanism broadcasts information about the object being allocated, this constitutes M messages. Third, the mechanism sends a single message about payment to the agent to whom the replica was assigned, which we can ignore since it has little impact on the total number of messages. The total number of messages required in a single iteration as of the order of M^2. From Theorem 1, we can conclude that in the

worst case the mechanism requires $O(M^3N)$ messages. ■

It is evident from Theorem 2 that the mechanism requires tremendous amounts of communications. This might be reduced by using some sophisticated network protocols. In the future generation distributed computing systems, the participating agents might actually be mobile, *i.e.*, they can travel from node to node in the network. In such a scenario the agents can converge to a specific site and execute the mechanism. In that case the number of messages will be reduced astronomically. In our experimentations we mimic this scenario and use IBM Pthreads to implement the mechanism. Other possible improvements are left as future research issues.

V. Supplementary Results

Here, we present some results that strengthen our claim on the optimality of the derived bidding mechanism. We begin by making the following observations.

Assume that the mechanism $m = (x(b),p(b))$ is truthful and each payment $p^i(b^{-i},b^i)$ and allocation $x^i(b^{-i},b^i)$ is twice differentiable with respect to b^i, for all the values of b^{-i}. We fix some agent i and derive a formula for p^i, allocation x^i, and profit to be the functions of just agent i's bid b^i. Since agent i's profit is always maximized by bidding truthfully (Lemma 1), the derivative is zero and the second derivative is non-positive at t^i. Since this holds no matter what the value of t^i is, we can integrate to obtain an expression for p^i. We state: $p^i(b^i) = p^i(0)+b^ix^i(b^i)-\int_0^{bi}x^i(u)du$. This is now the basis of our extended theoretical results. Literature survey revealed the following two important characteristics of a frugal payment mechanism. We state them below.

Definition 2: *With the other agents' bid b^{-i} fixed, consider $x^i(b^{-i},b^i)$ as a single variable function of b^i. We call this the allocation curve or the allocation profile of agent i. We say the output function x is decreasing if each of the associated allocation curves is decreasing, i.e., $x^i(b^{-i},b^i)$ is a decreasing function of b^i, for all i and b^{-i}.*

Based on the above definition, we can state the following theorem.

Theorem 3: *A mechanism is truthful if its output function $x(b)$ is decreasing.*

Proof: We prove this for the DRP mechanism. For simplicity we fix all bids b^{-i}, and focus on $x^i(b^{-i},b^i)$ as a single variable function of b^i, *i.e.*, the allocation x^i would only change if b^i is altered. We now consider two bids b^i and $b^{i'}$, such that $b^{i'} > b^i$. In terms of the true data t^i, this conforms to $RC_k^{i'} > RC_k^i$. Let x^i and $x^{i'}$ be the allocations made to the agent i when it bids b^i and $b^{i'}$, respectively. For a given allocation, the total replication cost associated can be represented as $C^i=\sum_{k\in xi}RC_k^i$. The proof of the theorem reduces to proving that $x^{i'} < x^i$, *i.e.*, the allocation computed by the algorithmic output is decreasing in b^i. The proof is simple by contradiction. Assume that that $x^{i'} \geq x^i$. This implies that $1/(C^i\text{-}RC_k^i) < 1/(C^{i'}\text{-}RC_k^i) \leq 1/(C^i\text{-}RC_k^{i'})$. This means that there must be an agent $-i$ who has a bid that supersedes $b^{i'}$. But that is not possible as we began with the assumption that all other bids are fixed so there can be no other agent $-i$. If $i = -i$, then that is also not possible since we assumed that $b^{i'} > b^i$. ■

We now extend the result obtained in Theorem 3 and state:

Theorem 4: A *decreasing output function admits a truthful payment scheme satisfying voluntary participation if and only if $\int_0^\infty x^i(b^{-i},u)du<\infty$ for all i, b^{-i}. In this case we can take the payments to be: $p^i(b^{-i},b^i)=b^ix^i(b^{-i},b^i)+\int_0^\infty x^i(b^{-i},u)du$.*

Proof: The first term $b^ix^i(b^{-i},b^i)$ compensates the cost incurred by agent i to host the allocation x^i. The second term $\int_0^\infty x^i(b^{-i},u)du$ represents the expected profit of agent i. If agent i bids its true value t^i, then its profit is

$$= u^i(t^i,(b^{-i},t^i))$$

$$= t^ix^i(b^{-i},t^i)+\int_{t^i}^\infty x^i(b^{-i},x)dx-t^ix^i(b^{-i},t^i)$$

$$= \int_{t^i}^\infty x^i(b^{-i},x)dx$$

If agent i bids its true value, then the expected profit is greater than in the case it bids other values. We explain this as follows: If agent i bids higher ($b^{i'}>t^i$), then the expected profit is

$$= u^i(t^i,(b^{-i},b^{i'}))$$

$$= b^{i'}x^i(b^{-i},b^{i'})+\int_{b^{i'}}^\infty x^i(b^{-i},x)dx-t^ix^i(b^{-i},b^{i'})$$

$$= (b^{i'}-t^i)x^i(b^{-i},b^{i'})+\int_{b^{i'}}^\infty x^i(b^{-i},x)dx\ .$$

Because $\int_{b^{i'}}^\infty x^i(b^{-i},x)dx<\infty$ and $b^{i'} > t^i$, we can express the profit when agent i bids the true value as follows: $\int_{t^i}^{b^{i'}} x^i(b^{-i},x)dx+\int_{b^{i'}}^\infty x^i(b^{-i},x)dx$. This is because x^i is decreasing in b^i and $b^{i'} > t^i$, we have the following equation: $(b^{i'}-t^i)x^i(b^{-i},b^{i'})<\int_{t^i}^{b^{i'}} x^i(b^{-i},x)dx$. From this relation, it can be seen that the profit with overbidding is lower then the profit with bidding the true data. Similar arguments can be used for underbidding. ■

VI. Experimental Setup and Discussion of Results

A. Setup

We performed experiments on a 440MHz Ultra 10 machine with 512MB memory. The experimental evaluations were targeted to benchmark the placement policies. The resource allocation mechanism was implemented using IBM Pthreads.

To establish diversity in our experimental setups, the network connectively was changed considerably. In this paper, we only present the results that were obtained using a maximum of 500 sites (nodes). We used existing topology generator toolkits and also self generated networks. In all the

TABLE II
PARAMETER INTERVAL VARIANCE CHARACTERIZATION FOR TOPOLOGIES WITH 100 NODES

Topology	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 [4] (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 [4] (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 [9].	μ={8.455,9.345,9.564}, σ={1.278,1.305,1.378}.

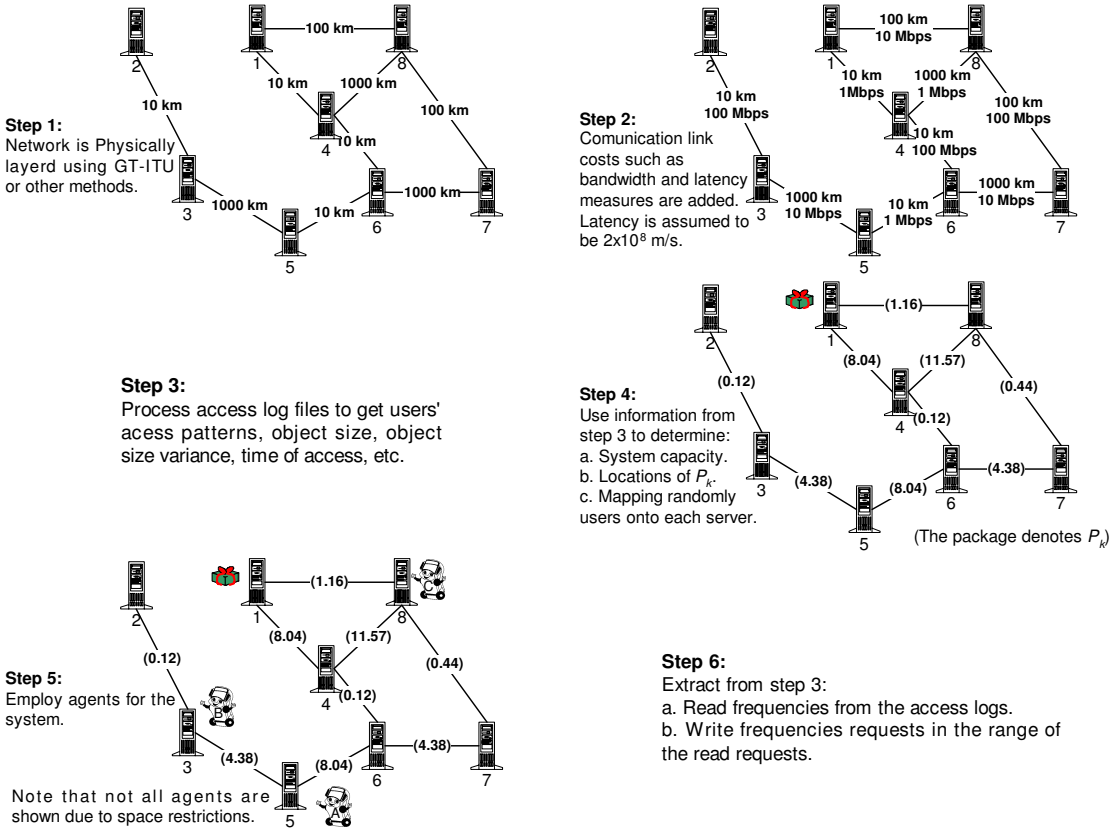

Fig. 3 A walk through the necessary steps involved in an experimental setup

topologies, the distance of the link between nodes was equivalent to the communication cost. Table II summarizes the various techniques used to gather forty-five various topologies for networks with 100 nodes. It is to be noted that the parameters vary for networks with lesser/larger number of nodes.

To evaluate the chosen replication placement techniques on realistic traffic patterns, we used the access logs collected at the Soccer World Cup 1998 website [3]. 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 chose the top five hundred clients (maximum experimental setup). A random mapping was then performed of the clients to the nodes of the topologies. Note that this mapping is not 1-1, rather 1-*M*. This gave us enough skewed workload to mimic real world scenarios. It is also worthwhile to mention that the total amount of requests entertained for each problem instance was in the range of 1-2 million. The primary replicas' original site was mimicked by choosing random locations. The capacities of the sites *C%* were 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 helped to instill enough diversity to benchmark object updates. The updates were randomly pushed onto different sites, and the total system update load was measured in terms of the percentage update requests *U%* compared that to the initial network with no updates. A brief overview on how the experimental setups are obtained is depicted in Fig. 3.

B. Comparative Algorithms

For comparison, we selected five various types of replica placement techniques. To provide a fair comparison, the assumptions and system parameters were kept the same in all the approaches. The techniques studied include efficient branch-and-bound based technique (Aε-Star [17]). For fine-grained replication, the algorithms proposed in [39], [40], and [47] are the only ones that address the problem domain similar to ours. We select from [47] the greedy approach (Greedy) for comparison because it is shown to be the best compared with 4 other approaches (including the proposed technique in [39]); thus, we indirectly compare with 4 additional approaches as well. Algorithms reported in [21] (Dutch (DA) and English auctions (EA)) and [40] (Genetic based algorithm (GRA)) are also among the chosen techniques for comparisons. Due to space limitations we will only give a brief overview of the comparative techniques. Details for a specific technique can be obtained from the referenced papers.

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

1. Aε-Star: In [17] the authors proposed a 1+ε admissible A-Star based technique called Aε-Star. This technique uses two lists: OPEN and FOCAL. The FOCAL list is the sub-list of OPEN, and only contains those nodes that do not deviate from the lowest f node by a factor greater than 1+ε. The technique works similar to A-Star, with the exception that the node selection (lowest h) is done not from the OPEN but from the FOCAL list. It is easy to see that this approach will never run into the problem of memory overflow, moreover, the FOCAL list always ensures that only the candidate solutions within a bound of 1+ε of the A-Star are expanded.

2. Greedy based technique: We modify the greedy approach reported in [47], 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. The readers will immediately realize that the bidding mechanism reported in this paper works similar to the Greedy algorithm. This is true; however, the Greedy approach does not guarantee optimality even if the algorithm is run on the very same problem instance. Recall that Greedy relies on making combinations of object assignments and therefore, suffers from the initial choice of object selection (which is done randomly). This is never the case in the derived bidding mechanism, which identifies optimal allocations in every case.

3. Dutch auction: The auctioneer begins with a high asking price which is lowered until some agent is willing to accept the auctioneer's price. That agent pays the last announced price. This type of auction is convenient when it is important to auction objects quickly, since a sale never requires more than one bid. In no case does the auctioneer reveal any of the bids submitted to him, and no information is shared between the agents. It is shown that for an agent to have a probabilistically superior bid than n-1 other bids; an agent should have the valuation divided by n.

4. English auction: In this type of auction, the agents bid openly against one another, with each bid being higher than the previous bid. The auction ends when no agent is willing to bid further. During the auction when an auctioneer receives a bid higher than the currently submitted bids, he announces the bid value so that other agents (if needed) can revise their currently submitted bids. In [44] the discussion on EA reveals that the optimal strategy for a bidder i is to bid a value which is directly derived from his valuation.

5. GRA: In [40], the authors proposed a genetic algorithm based heuristic called GRA. GRA provides good solution quality, but suffers from slow termination time. This algorithm was selected since it realistically addressed the fine-grained data replication using the same problem formulation as undertaken in this article.

C. Comparative Analysis

We study the behavior of the placement techniques when the number of sites increases (Fig. 3), by setting the number of objects to 2000, while in Fig. 4, we study the behavior when the number of objects increase, by setting the number of sites to 500. 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. For the first experiment we fixed C=20% and U=75%. We intentionally chose a high workload so as to see if the techniques studied successfully handled the

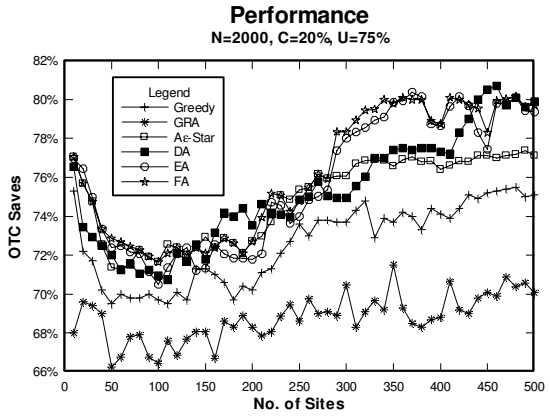

Fig. 4 OTC savings versus number of sites

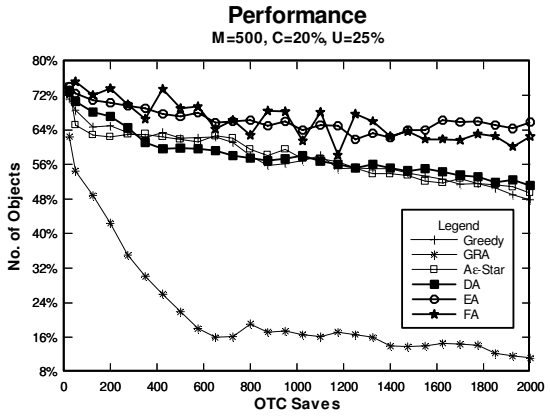

Fig. 5 OTC savings versus number of objects

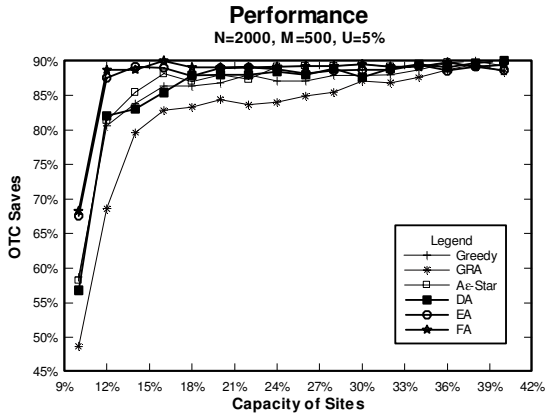

Fig. 6 OTC savings versus capacity

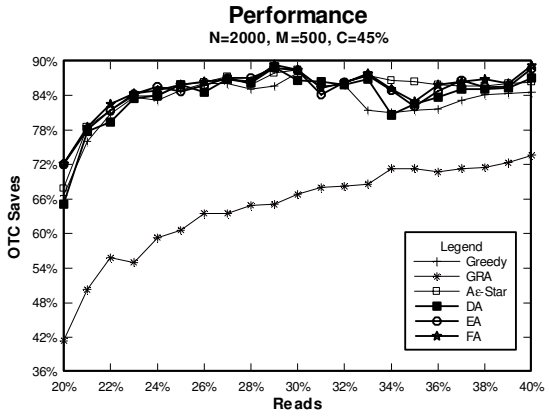

Fig. 7 OTC savings versus reads

extreme cases. The first observation is that FA and EA outperformed other techniques by considerable amounts. Second, DA converged to a better solution quality under certain problem instances. Some interesting observations were also recorded, such as, all but GRA showed initial loss in OTC savings with the initial number of site increase in the system, as much as 7% loss was recorded in case of Greedy with only a 40 site increase. GRA showed an initial gain since with the increase in the number of sites, the population permutations increase exponentially, but with the further increase in the number of sites this phenomenon is not so observable as all the essential objects are already replicated. The top performing techniques (DA, EA, Aε-Star and FA) showed an almost constant performance increase (after the initial loss in OTC savings). This is because by adding a site (server) in the network, we introduce additional traffic (local requests), together with more storage capacity available for replication. All four equally cater for the two diverse effects. GRA also showed a similar trend but maintained lower OTC savings. This was in line with the claims presented in [17] and [40].

To observe the effect of increase in the number of objects in the system, we chose a softer workload with C=20% and U=25%. The intention was to observe the trends for all the algorithms under various workloads. The increase in the number of objects has diverse effects on the system as new read/write patterns (users are offered more choices) emerge, and also the increase in the strain on the overall capacity of the system (increase in the number of replicas). An effective algorithm should incorporate both the opposing trends. From the plot, the most surprising result came from GRA. It dropped its savings from 58% to 13%. This was contradictory to what was reported in [40]. But there the authors had used a uniformly distributed link cost topology, and their traffic was based on the Zipf distribution [55]. While the traffic access logs of the World Cup 1998 are more or less double-Pareto in nature. In either case the exploits and limitations of the technique under discussion are obvious. The plot also shows a near identical performance by Aε-Star, DA and Greedy. The relative difference among the three techniques is less than 2%. However, Aε-Star did maintain its domination. From the plots the supremacy of EA and FA is observable. Both the techniques showed high performance, with a slight edge in favor of FA.

Next, we observe the effects of system capacity increase. An increase in the storage capacity means that a large number of objects can be replicated. Replicating an object that is already extensively replicated, is unlikely to result in significant traffic savings as only a small portion of the servers will be affected overall. Moreover, since objects are

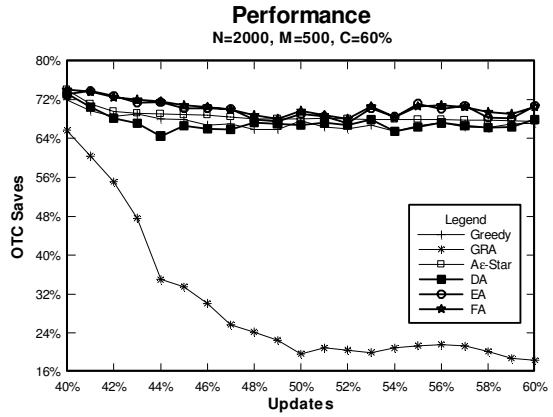

Fig. 8 OTC savings versus updates

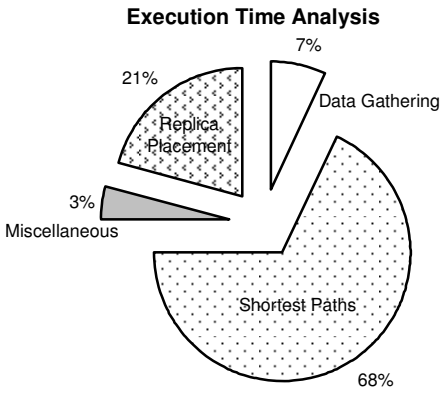

Fig. 9 Execution time components

TABLE III
RUNNING TIME IN SECONDS [C=20%, U=45%] (SMALL PROBLEM INSTANCES)

Problem Size	Greedy	GRA	Aε-Star	DA	EA	FA
M=20, N=50	70.11	92.66	97.18	**25.16**	39.36	26.14
M=20, N=100	76.59	96.40	102.97	27.71	41.21	**27.01**
M=20, N=150	78.26	101.01	113.85	**32.44**	54.57	36.26
M=30, N=50	95.24	126.92	140.78	**38.45**	59.25	39.11
M=30, N=100	109.17	125.04	148.83	**39.21**	63.14	40.10
M=30, N=150	135.21	148.59	179.74	45.96	68.20	**42.09**
M=40, N=50	126.40	154.13	198.77	**42.66**	76.27	45.41
M=40, N=100	134.65	168.48	236.67	**43.62**	77.16	46.97
M=40, N=150	141.08	204.43	270.63	**47.52**	82.53	48.83

TABLE IV
RUNNING TIME IN SECONDS [C=45%, U=15%] (LARGE PROBLEM INSTANCES)

Problem Size	Greedy	GRA	Aε-Star	DA	EA	FA
M=300, N=1350	190.01	242.12	247.66	**87.92**	164.15	93.26
M=300, N=1400	206.26	326.82	279.45	**95.64**	178.90	97.98
M=300, N=1450	236.61	379.01	310.12	115.19	185.15	**113.65**
M=300, N=1500	258.45	409.17	333.03	127.10	191.24	**124.73**
M=300, N=1550	275.63	469.38	368.89	**143.94**	197.93	147.16
M=300, N=2000	298.12	475.02	387.94	**158.45**	204.29	159.12
M=400, N=1350	321.60	492.10	353.08	**176.51**	218.15	176.90
M=400, N=1400	348.53	536.96	368.03	**187.26**	223.56	195.41
M=400, N=1450	366.38	541.12	396.96	**192.41**	221.10	214.55
M=400, N=1500	376.85	559.74	412.17	**208.92**	245.47	218.73
M=400, N=1550	389.71	605.63	415.55	**215.24**	269.31	223.92
M=400, N=2000	391.55	659.39	447.97	**224.18**	274.24	235.17
M=500, N=1350	402.20	660.86	460.44	**246.43**	284.63	259.56
M=500, N=1400	478.10	689.44	511.69	**257.96**	301.72	266.42
M=500, N=1450	485.34	705.07	582.71	**269.45**	315.13	272.68
M=500, N=1500	511.06	736.43	628.23	**278.15**	324.26	291.83
M=500, N=1550	525.33	753.50	645.26	**289.64**	331.57	304.47
M=500, N=2000	539.15	776.99	735.36	**312.68**	345.94	317.60

not equally read intensive, increase in the storage capacity would have a great impact at the beginning (initial increase in capacity), but has little effect after a certain point, where the most beneficial ones are already replicated. This is observable in Fig. 5, which shows the performance of the algorithms. GRA once again performed the worst. The gap between all other approaches was reduced to within 7% of each other. DA and FA showed an immediate initial increase (the point after which further replicating objects is inefficient) in its OTC savings, but afterward showed a near constant performance. GRA although performed the worst, but observably gained the most OTC savings (35%) followed by Greedy with 29%. Further experiments with various update ratios (5%, 10%, and 20%) showed similar plot trends. It is

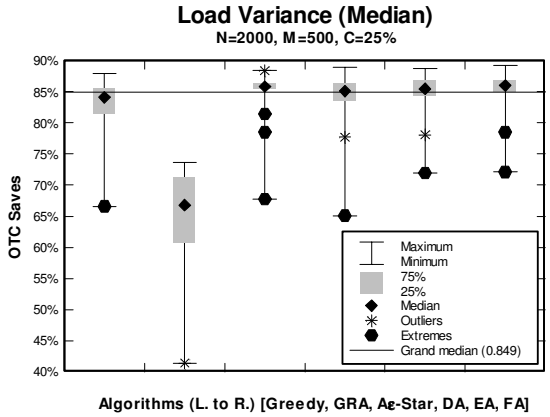

Fig. 10 Comparative load variance (Median)

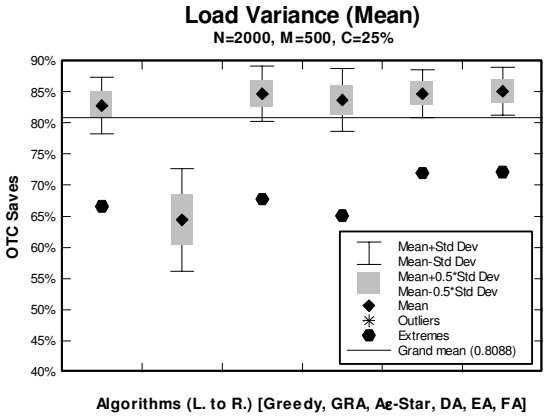

Fig.11 Comparative load variance (Mean)

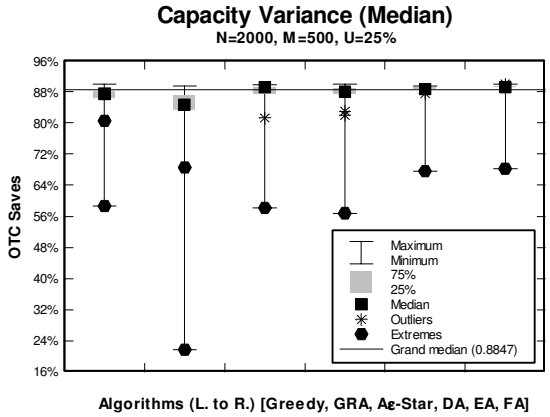

Fig. 12 Comparative capacity variance (Median)

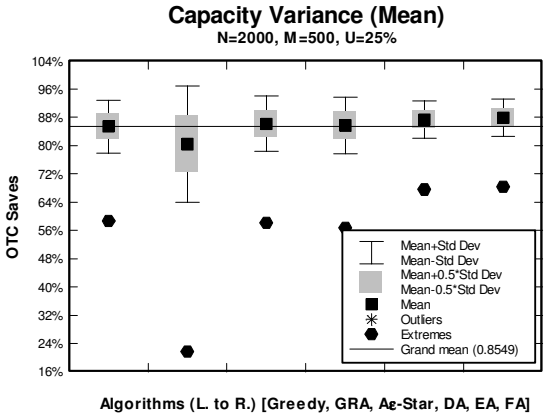

Fig. 13 Comparative capacity variance (Mean)

TABLE V
AVERAGE OTC (%) SAVINGS UNDER SOME PROBLEM INSTANCES

Problem Size	Greedy	GRA	Aε-Star	DA	EA	FA
N=150, *M*=20 [*C*=20%,*U*=25%]	70.46	69.74	**74.62**	70.15	73.15	74.24
N=200, *M*=50 [*C*=20%,*U*=20%]	73.94	70.18	**77.42**	72.66	77.41	75.16
N=300, *M*=50 [*C*=25%,*U*=5%]	70.01	64.29	**70.33**	68.23	70.11	69.53
N=300, *M*=60 [*C*=35%,*U*=5%]	71.66	65.94	72.01	70.22	71.23	**73.45**
N=400, *M*=100 [*C*=25%,*U*=25%]	67.40	62.07	71.26	69.46	70.55	**72.81**
N=500, *M*=100 [*C*=30%,*U*=35%]	66.15	61.62	71.50	70.21	71.12	**72.04**
N=800, *M*=200 [*C*=25%,*U*=15%]	67.46	65.91	70.15	69.29	70.61	**72.19**
N=1000, *M*=300 [*C*=25%,*U*=35%]	69.10	64.08	70.01	70.16	71.29	**71.95**
N=1500, *M*=400 [*C*=35%,*U*=50%]	70.59	63.49	70.51	**72.77**	72.61	72.35
N=2000, *M*=500 [*C*=10%,*U*=60%]	67.03	63.37	72.16	68.63	69.24	**73.25**

also noteworthy (plots not shown in this paper due to space restrictions) that the increase in capacity from 10% to 17%, resulted in 4 times (on average) more replicas for all the algorithms.

Next, we observe the effects of increase in the read and update (write) frequencies. Since these two parameters are complementary to each other, we describe them together. In both the setups the number of sites and objects were kept constant. 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). However, the increase in the number of updates in the system requires the replicas be placed as close as to the primary site as possible (to reduce the update broadcast). This phenomenon is also interrelated with the system capacity, as the update ratio sets an upper bound on the possible traffic reduction through replication. Thus, if we consider a system with unlimited capacity, the "replicate everywhere anything" policy is strictly inadequate. The read and update parameters indeed help in drawing a line between good and marginal algorithms. The plots in Figs. 6 and 7 show the results of read and update frequencies, respectively. A clear classification can be made between the algorithms.

Aε-Star, DA, EA, Greedy and FA incorporate the increase in the number of reads by replicating more objects and thus savings increase up to 88%. GRA gained the least of the OTC savings of up to 67%. To understand why there is such a gap in the performance between the algorithms, we should recall that GRA specifically depend on the initial population (for details see [40]). Moreover, GRA maintains a localized network perception. Increase in updates result in objects having decreased local significance (unless the vicinity is in close proximity to the primary location). On the other hand, Aε-Star, DA, EA, Greedy and FA never tend to deviate from their global view of the problem search space.

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 Fig. 8. There 68% of all the algorithm termination time was taken by the repeated calculations of the shortest paths. Data gathering and dispersion, such as reading the access frequencies from the processed log, etc. took 7% of the total time. Other miscellaneous operations including I/O 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 Tables III and IV.

Various problem instances were recorded with C=20%, 45% and U=15%, 45%. 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 time recorded over the problem instance. It is observable that FA and DA terminated faster than all the other techniques, followed by EA, Greedy, Aε-Star and GRA. If a static environment was considered, FA with the maximum problem instance would have terminated approximately in 66.69 seconds (21% of the algorithm termination time).

In summary, based on the solution quality alone, the algorithms can be classified into four categories: 1) The very high performance algorithms that include EA and FA, 2) the high performance algorithms of Greedy and DA, 3) the medium-high performance Aε-Star, and finally 4) the mediocre performance algorithm of GRA. While considering the termination timings, FA and DA did extremely well, followed by EA, Greedy, Aε-Star, and GRA.

D.Supplementary Analysis

Here, we present some supplementary results that strengthen our comparative analysis provided in Section VI.C. We show the relative performance of the techniques with load and storage capacity variance. The plots in Figs. 10, 11, 12 and 13 show the recorded performances. All the plots summarize the measured performance with varying parameters (most of which could not be included in this paper due to space limitations). We are mostly interested in measuring the median and mean performances of the algorithms. With load variance FA edges over Aε-Star with a savings of 87%. The plot also shows that nearly every algorithm performed well with grand median on 84.9%. The graphs are self explanatory and also capture the outliners and extreme points. The basic exercise in plotting these results is to see which algorithms perform consistently. GRA for example, records the lowest extremes, and hardly any outliners. On the other hand the proposed FA's performance is captured in a small interval, with high median and mean OTC savings.

Table V shows the quality of the solution in terms of OTC percentage for 10 problem instances (randomly chosen), each being a combination of various numbers of sites and objects, with varying storage capacity and update ratio. For each row, the best result is indicated in bold. The proposed FA algorithm steals the show in the context of solution quality, but Aε-Star, EA and DA do indeed give a good competition, with a savings within a range of 5%-10% of FA.

VII. Related Work

Myriad theoretical approaches are proposed that we classify into the following six categories:

1. Facility Location: In [15], the authors employed several techniques to address the Internet data replication problem similar to that of the classical facility location problem. The techniques reported are very tedious and have superfluous assumptions. Thus, the problem definition in [15] does not fully capture the concept of replicating a single object/site over a fixed number of hosts [41].

2. File Allocation: File allocation has been a popular line of research in: distributed computing, distributed databases, multimedia databases, paging algorithms, and video server systems [1], [8], [37], [42]. All the above referenced articles incorporate data replication onto a set of distributed locations (distributed system), which can easily be modified to its equivalent problem in the context of Internet. Under the assumption of unlimited server memory the authors in [37], provided a guaranteed optimal result for Internet data replication, but has little practical use [41], since the replica placements are based on the belief that the access patterns remain unchanged.

3. Minimum k-Median: The celebrated NP-complete minimum k-median problem captures the coarse-grained replication well, as it can tackle with the problem of distributing a single replica over a fixed number of hosts. In [39] the authors studied the problem of placing M proxies at N nodes when the topology of the network is a tree and proposed an $O(N^3M^2)$ algorithm. A more generalized solution was presented in [47]. There the authors proposed a greedy algorithm that outperformed other techniques including the work reported in [39].

4. Capacity-constrained Optimization: In [16], the

authors use the capacity-constrained version of the minimum *k*-median problem, and guarantee a stable performance. However, such results are possible only with very conservative assumptions as addressed in [14] and [38], therefore, they can not handle the dynamics of the system [41].

5. Bin Packing: The bin packing based problem formulation is commonly used to model load balancing problems. The problem of distributing documents in a cluster of web servers in order to perform load balancing was reported in [45]. However, the goodness of the results only holds when the network under consideration was small. A more extensive evaluation using bin packing techniques is performed in [17].

6. Knapsack: To achieve better load balancing partial replication can be employed. The idea of partial replication is analogous to the classical 0-1 knapsack problem [41]. Some of the significance work in this line of pursuit is reported in [5], [40], and [52].

A number of bibliographies and reading materials for web content replication are also available online, e.g., [7]. A brief overview of replication and its challenges are provided in [41] and [48]. Moreover, we also must make the reader aware of a number of research papers on resource and replica allocation using game theory [17]-[35].

VIII. Conclusion

This paper proposed a game theoretical resource allocation mechanism that effectively addressed the fine-grained data replication problem with selfish players. The experimental results which were recorded against some well know techniques such as branch and bound, greedy, game theoretical auctions, and genetic algorithms revealed that the proposed mechanism exhibited 5%-10% better solution quality and incurred fast execution time.

References

[1] P. Apers, "Data Allocation in Distributed Database Systems," ACM Transactions on Database Systems, 13(3), pp. 263-304, 1988.
[2] A. Archer and E. Tardos, "Truthful Mechanism for One-parameter Agents," in Proc. Of 42nd IEEE FOCS, pp. 482-491, 2001.
[3] 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.
[4] K. Calvert, M. Doar, E. Zegura, "Modeling Internet Topology," IEEE Communications, 35(6), pp. 160-163, 1997.
[5] 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.
[6] M. Charikar, S. Guha, E. Tardos and D. Shmoys, "A Constant-Factor Approximation Algorithm for the K-Median Problem," in ACM STOC, pp. 1-10, 1999.
[7] B. Davison, "A Survey of Proxy Cache Evaluation Techniques," in Proc. of the 4th International Web Caching Workshop, 1999.
[8] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator and N. Young, "Competitive Paging Algorithms," Journal of Algorithms, 12(4), pp. 685-699, 1991.
[9] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," IEEE/ACM Trans. Networking, 9(4), pp. 253-285, 2001.
[10] M. Garey and D. Johnson, Computers and Intractability, W.H. Freeman and Co., 1979.
[11] J. Green and J. Laffont, "Characterization of Satisfactory Mechnisms for the revelation of Preferences for Public Goods," Econometrica, pp. 427-438, 1977.
[12] T. Groves, "Incentives in Teams," Econometrica, pp. 617-631, 1973.
[13] D. Grosu and A. Chronopoulos, "Algorithmic Mechnism Design for Load Balancing in Distributed Systems," IEEE Trans. Systems, Man and Cybernatics B, 34(1), pp. 77-84, 2004.
[14] 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.
[15] S. Jamin, C. Jin, T. Kurc, D. Raz and Y. Shavitt, "Constrained Mirror Placement on the Internet," in Proc. of the IEEE INFOCOM, 2001.
[16] J. Kangasharju, J. Roberts and K. Ross, "Object Replication Strategies in Content Distribution Networks," in Proc. of Web Caching and Content Distribution Workshop, pp. 455-456, 2001.
[17] S. U. Khan and I. Ahmad, "Heuristics-based Replication Schemas for Fast Information Retrieval over the Internet," in 17th International Conference on Parallel and Distributed Computing Systems (PDCS), San Francisco, CA, USA, September 2004, pp. 278-283.
[18] S. U. Khan and I. Ahmad, "A Powerful Direct Mechanism for Optimal WWW Content Replication," in 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Denver, CO, USA, April 2005.
[19] S. U. Khan and I. Ahmad, "A Game Theoretical Extended Vickery Auction Mechanism for Replicating Data in Large-scale Distributed Computing Systems," in International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, NV, USA, June 2005, pp. 904-910.
[20] S. U. Khan and I. Ahmad, "RAMM: A Game Theoretical Replica Allocation and Management Mechanism," in 8th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), Las Vegas, NV, USA, December 2005, pp. 160-165.
[21] S. U. Khan and I. Ahmad, "A Powerful Direct Mechanism for Optimal WWW Content Replication," in 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Denver, CO, USA, April 2005.
[22] S. U. Khan and I. Ahmad, "A Game Theoretical Extended Vickery Auction Mechanism for Replicating Data in Large-scale Distributed Computing Systems," in International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, NV, USA, June 2005, pp. 904-910.
[23] S. U. Khan and I. Ahmad, "RAMM: A Game Theoretical Replica Allocation and Management Mechanism," in 8th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), Las Vegas, NV, USA, December 2005, pp. 160-165.
[24] S. U. Khan and I. Ahmad, "Discriminatory Algorithmic Mechanism Design Based WWW Content Replication," Informatica, vol. 31, no. 1, pp. 105-119, 2007.
[25] S. U. Khan and I. Ahmad, "Game Theoretical Solutions for Data Replication in Distributed Computing Systems," in Handbook of Parallel Computing: Models, Algorithms, and Applications, S. Rajasekaran and J. Reif, Eds., Chapman & Hall/CRC Press, Boca Raton, FL, USA, 2007, ISBN 1-584-88623-4, Chapter 45.
[26] S. U. Khan and I. Ahmad, "A Semi-Distributed Axiomatic Game Theoretical Mechanism for Replicating Data Objects in Large Distributed Computing Systems," in 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS), Long Beach, CA, USA, March 2007.
[27] B. Khargharia, S. Hariri, F. Szidarovszky, M. Houri, H. El-Rewini, S. U. Khan, I. Ahmad, and M. S. Yousif, "Autonomic Power and Performance Management for Large-Scale Data Centers," in 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS), Long Beach, CA, USA, March 2007.
[28] S. U. Khan, "Game Theoretical Techniques for Designing Counter-Terrorism Systems," in 5th International Symposium on Defense and Security, vol. 6560 of SPIE (Society of Photo-Optical Instrumentation Engineers), Orlando, FL, USA, April 2007, pp. 74-82.
[29] S. U. Khan and I. Ahmad, "A Cooperative Game Theoretical Replica Placement Technique," in 13th International Conference on Parallel and Distributed Systems (ICPADS), Hsinchu, Taiwan, December 2007.
[30] S. U. Khan and I. Ahmad, "Comparison and Analysis of Ten Static Heuristics-based Internet Data Replication Techniques," Journal of Parallel and Distributed Computing, vol. 68, no. 2, pp. 113-136, 2008.

[31] S. U. Khan, A. A. Maciejewski, H. J. Siegel, and I. Ahmad, "A Game Theoretical Data Replication Technique for Mobile Ad Hoc Networks," in 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS), Miami, FL, USA, April 2008.
[32] I. Ahmad, S. Ranka, and S. U. Khan, "Using Game Theory for Scheduling Tasks on Multi-core Processors for Simultaneous Optimization of Performance and Energy," in 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS), Miami, FL, USA, April 2008.
[33] S. U. Khan and I. Ahmad, "A Pure Nash Equilibrium based Game Theoretical Method for Data Replication across Multiple Servers," IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 3, pp. 346-360, 2009.
[34] S. U. Khan and I. Ahmad, " Cooperative Game Theoretical Technique for Joint Optimization of Energy Consumption and Response Time in Computational Grids," IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 4, pp. 537-553, 2009.
[35] S. U. Khan and C. Ardil, "A Weighted Sum Technique for the Joint Optimization of Performance and Power Consumption in Data Centers," International Journal of Electrical, Computer, and Systems Engineering, vol. 3, no. 1, pp. 35-40, 2009.
[36] V. Krishna, Auction Theory, Academic Press, 2002.
[37] 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, 14(7), pp. 1332-1348, 1996.
[38] B. Lee and J. Weissman, "Dynamic Replica Management in the Service Grid," in Proc. of IEEE International Symposium on High Performance Distributed Computing, 2001.
[39] 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, 2000.
[40] T. Loukopoulos, and I. Ahmad, "Static and Adaptive Distributed Data Replication using Genetic Algorithms," Accepted to appear in Journal of Parallel and Distributed Computing.
[41] T. Loukopoulos, I. Ahmad, and D. Papadias, "An Overview of Data Replication on the Internet," in Proc. of ISPAN, pp. 31-36, 2002.
[42] S. March and S. Rho, "Allocating Data and Operations to Nodes in Distributed Database Design," IEEE Trans. Knowledge and Data Engineering, 7(2), pp.305-317, 1995.
[43] S. Martello and P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, 1990.
[44] A. Mas-Collel, W. Whinston and J. Green, Microeconomic Theory, Oxford University Press, 1995.
[45] B. Narebdran, S. Rangarajan and S. Yajnik, "Data Distribution Algorithms for Load Balancing Fault-Tolerant Web Access," in Proc. of the 16th Symposium on Reliable Distributed Systems, 1997.
[46] N. Nisan and A. Ronen, "Algorithimic Mechanism Design," in Proc. of 31st ACM STOC, pp. 129-140, 1999.
[47] L. Qiu, V. Padmanabhan and G. Voelker, "On the Placement of Web Server Replicas," in Proc. of the IEEE INFOCOM, 2001.
[48] M. Rabinovich, "Issues in Web Content Replication," Data Engineering Bulletin, 21(4), 1998.
[49] M. Rabanovich and O. Spatscheck, Web Caching and Replication, Addison-Wesley, 2002.
[50] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, J. Kubiatowicz, "Maintenance-free Global Storage," IEEE Internet Computing, 5(5), pp. 40-49, 2001.
[51] S. Saurabh and D. Parkes, "Hard-to-Manupilate VCG-Based Auctions," Avaialable at: http://www.eecs.harvard.edu/econcs/pubs/hard_to_manipulate.pdf
[52] M. Sayal, Y. Breitbart, P. Scheuermann and R. Vingralek, "Selection Algorithms for Replicated Web Servers," in Proc. of the Workshop on Internet Server Performance, 1998.
[53] S. So, I. Ahmad and K. Karlapalem, "Response Time Driven Multimedia Data Objects Allocation for Browsing Documents in Distributed Environments," IEEE Transactions on Knowledge and Data Engineering, 11(3), pp. 386-405, 1999.
[54] W. Vickrey, "Counterspeculation, 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] G. Zipf, Human Behavior and the Principle of Least-Effort, Addison-Wesley, 1949.