A Powerful Direct Mechanism for Optimal WWW Content Replication

Samee Ullah Khan and Ishfaq Ahmad
Department of Computer Science and Engineering
University of Texas at Arlington, Arlington, TX-76019, U.S.A.
{sakhan, iahmad}@cse.uta.edu

Abstract

This paper addresses the problem of fine-grained data replication in large distributed systems, such as the Internet, so as to minimize the user access delays. With fine-grained data replication, certain data objects, as opposed to a complete site, are duplicated at multiple servers. In this paper, we abstract the distributed system as an agent-based model wherein mobile agents on behalf of their nodes continuously compete for allocation and reallocation of data objects. However, since these agents do not have a global view of the system, the optimization process becomes highly local. This localization may encourage these selfish agents to alter the output of the resource allocation mechanism in their favor by misreporting critical data such as the objects' popularity. This paper proposes a game theoretical resource allocation mechanism involving selfish agents. The mechanism ensures that the agents do not misreport, always follow the rules, and that a global optima is achieved. The mechanism is extensively evaluated against some well-known algorithms, such as: greedy, branch and bound, game theoretical auctions and genetic algorithms. The experimental results reveal that the mechanism provides excellent solution quality, while maintaining fast execution time.

1. Introduction

Data replication across a read intensive network can potentially reduce the network traffic, which, in turn, can lower the response times experienced by end-users. On the other hand, with rapid updates, maintaining a large number of replicas can incur a prohibitively high overhead [14]. Therefore, efficient and effective replication schemas strongly depend on how many replicas to be placed in the system, and more importantly where.

We abstract the distributed computing system (Internet) as an agent-based model wherein mobile agents continuously compete for allocation and reallocation of the system resources (data objects in our case). An agent is a computational entity that is capable of autonomous behaviour in the sense of being aware of the options available to it when faced with a decision making task related to its domain of interest [2]. Numerous applications exist for such agents often in various fields of electronic commerce, network management, intelligent user interfaces, etc. In agent-based distributed computing systems, an agent is seen as part of a community of similar though heterogeneous agents that are designed to compete for scarce resources. Motivated by their self interests and the fact that the agents do not have a global view of the distributed system, they optimize their individual interests, such as, minimize communication costs, latencies, etc. In such systems there is no motivation for cooperation and the agents may manipulate the outcome of the resource allocation mechanism by misreporting their capabilities leading to severe performance degradation. To cope with these *selfish* agents, new resource allocation mechanisms are to be conceived. The goal of a mechanism should be to force the agents not to misreport and always follow the rules.

In this paper, we will use game theoretical techniques to identify a mechanism that encapsulates the selfishness of the agents, while having a controlling hand over them. This work is inspired from the work reported in [17] and [20]. In essence, game theory is the study of what happens when independent agents act selfishly. A mechanism asks how one can design systems so that agents' selfish behavior results in the desired system-wide goals.

The major results of this paper are as follows:

1. We derive a generalized resource allocation mechanism. This mechanism allows selfish agents to compete in a non-cooperative environment.
2. We investigate this mechanism in detail by identifying some useful properties and the necessary

conditions of optimality.
3. As an application we employ the derived mechanism to the fine-grained data replication problem (DRP). We perform extensive experimental comparisons against some well-known techniques, such as: greedy, branch and bound, genetic and game theoretical auctions.

The remainder of this paper is organized as follows. Section 2 describes the resource allocation mechanism. Section 3 formulates the DRP. Section 4 concentrates on modeling the resource allocation mechanism for the DRP. The experimental results, related work and concluding remarks are provided in Sections 5, 6 and 7, respectively.

2. The Resource Allocation Mechanism

We describe the resource allocation mechanism as:

The Environment: The environment can be described as a triplet (M,Ω,Θ). The first element of the triplet M is the list of agents (or potential participating agents) in the mechanism. The second element Ω, is the set of possible outcomes over which the agents and the mechanism have preferences. The third element is a highly abstract one: $\Theta = \Theta^1 \times \ldots \times \Theta^M$ is the set of type profiles $t = (t^1,\ldots,t^M)$, which includes a type for each agent. Agent i's type t^i indexes the agent's preferences. The type profile and the outcome combine to determine individual payoffs: u^i: $\Omega \times \Theta \rightarrow \mathbf{R}$. Thus, $u^i(\Phi,t)$ denotes the payoff that agent i gets when the outcome is $\Phi \in \Omega$ and the type profile is t. It is sometimes convenient to write a type profile as $t = (t^i,t^{-i})$, where t^{-i} lists the types of the agents other than i, *i.e.*, $t^{-i} = (t^1,\ldots t^{i-1},t^{i+1},\ldots,t^M)$.

The Setup: Let there be M agents. Let X denote the set of possible decisions with typical element x. An outcome is a pair (x,p) describing a decision x and a vector of positive or negative payments $p = (p^1,\ldots p^M)$ by the agents. For instance, in auctions, the decision is a vector where $x^i = 1$ if agent i gets the object and 0 otherwise. The associated vector of payments is p, where $p^i = bid^i$ if i bids bid^i and wins, and in that case $p^j = 0$ for the other agents.

Utility: Each agent i values outcome (*utility*) according to $u^i((x,p),t) = v^i(x,t^i) - p^i$, that is, i's payoff corresponding to outcome (x,p) is i's value $v^i(x,t^i)$ of the decision x, which depends only on i's own type t^i, minus the payment that i must make in order to acquire the object.

Performance: The performance of any mechanism can be described in two parts [20]: a) the decision performance maps types t into decisions x, whereas b) the transfer performance maps types t into payments p. When the decision x allocates objects, we call this x, the *allocation performance*. The mechanism attempts to achieve efficient performance subject to the constraint that payments add up to zero. Given the assumptions described above, a decision x is efficient if it maximizes the total value $\sum_{i \in M} v^i(x,t^i)$. For instance, a final allocation is efficient if it awards the object to the agent who values it most. In our proposed mechanism, by construction, net payments always add up to zero, because the mechanism receives any sums that the agents pay.

Incentive Compatibility: This means that a) $S = \Theta$ and that b) the strategy profile $(\sigma^i(t^i) = t^i)$, $i \in M$ is an equilibrium, where S is the strategy set and σ^i is any strategy based on the type profile of agent i. In simple words, the first condition means that each agent is required to report a type to the mechanism. In literature the direct mechanism is usually referred as being pairs (x,p), leaving the strategy set implicit. The second condition, incentive compatibility means that reporting ones' type truthfully is equilibrium according to whatever solution concept is chosen. In this paper, we focus on dominant strategy implementation, so the relevant solution concept is that each agent plays a dominant strategy.

Objective: The mechanism uses the reported types to compute the maximum total value $V(X,M,t)$ and a corresponding total value maximizing decision $\hat{o}(X,M,t)$ as follows:

$$V(X,M,t) = \max_{x \in X} \sum_{i \in M} v^i(x,t^i), \quad (1)$$

$$\hat{o}(X,M,t) = \arg\max_{x \in X} \sum_{i \in M} v^i(x,t^i). \quad (2)$$

One might think that such a direct approach would be doomed to failure, because each agent seems to have an incentive to misrepresent its preferences to influence the decision in its favor. However, the agent's incentives depend not only on the decision but also on the payments, which is the clever and surprising part of this mechanism.

Misreporting and Payments: The mechanism eliminates incentives for misreporting by imposing on each agent the cost of any distortion it causes. The payment for agent i is set so that i's report cannot effect the total payoff to the set of other agents (excluding agent i), M-i. With this principle in mind, let us derive a formula for the payments. To capture the effect of i's report on the outcome, we introduce a hypothetical *null report*, which corresponds to agent i reporting that it is indifferent among the possible decisions and cares only about payments. When i makes the null report, the mechanism optimally chooses the decision $\hat{o}(X,M\text{-}i,t^{-i})$. The resulting total

value of the decision for the set of agents M-i would be $V(X,M\text{-}i,t^{-i})$, and the mechanism might also collect a payment $h^i(t^{-i})$ from agent i. Thus, if i makes a null report, the total payoff to the agents in set M-i is $V(X,M\text{-}i,t^{-i}) - h^i(t^{-i})$.

Discussion– The mechanism is constructed so that this $(V(X,M\text{-}i,t^{-i}) - h^i(t^{-i}))$ same amount is the total payoff to those agents regardless of i's report. Thus, suppose that when the reported type is t, i's payment is $p^i(X,M,t) + h^i(t^{-i})$, so that $p^i(X,M,t)$ is i's additional payment over what i would pay if it made the null report. The decision $\hat{o}(X,M,t)$ generally depends on i's report, and the total payoff to members of M-i is then $\sum_{i\in M-i} v^i(\hat{o}(X,M,t)),t^i) + p^i(X,M,t) + h^i(t^{-i})$. We equate this total value with the corresponding total value when i makes the null report:

$$\sum_{i\in M-i} v^i\left(\hat{o}\left(X,M,t\right),t^i\right) + p^i\left(X,M,t\right) + h^i\left(t^{-i}\right) = h^i\left(t^{-i}\right) + V\left(X,M-i,t^{-i}\right). \quad (3)$$

Using Eq. 1, we solve for the extra payment as:

$$p^i\left(X,M,t\right) = V\left(X,M-i,t^{-i}\right) - \sum_{i\in M-i} v^i\left(\hat{o}\left(X,M,t\right),t^i\right), \quad (4)$$

$$= \sum_{i\in M-i} v^i\left(\hat{o}\left(X,M-i,t\right),t^i\right) - \sum_{i\in M-i} v^i\left(\hat{o}\left(X,M,t\right),t^i\right). \quad (5)$$

According to Eq. 4, if agent i's report leads to a change in the decision x, then i's extra payment $p^i(X,M,t)$ is specified to compensate the members of M-i for the total losses they suffer on the account.

Thus, we arrive at the formal definition of mechanism, and we state:

Definition 1: *The mechanism* $(\Theta,(\hat{o},p+h))$
1. *is a direct mechanism in which* $\hat{o}$ *satisfies Eq.* 2, *p satisfies Eq.* 4 (*for all* M,X,t *and* $i\in M$), *and payments are determined by* $p^i(X,M,t) + h^i(t^{-i})$.
2. *An agent is pivotal if* $\hat{o}(X,M,t) \neq \hat{o}(X,M\text{-}i,t^{-i})$.
3. *The pivot mechanism is the resource allocation mechanism in which* $h^i = 0$ *for all* $i\in M$.

Mechanism Optimality: The derived mechanism ensures that it is always optimal for the agents to report truthfully, regardless of the reports made by others. We also demonstrate that reporting truthfully is a dominating strategy, that is, it is the only strategy that is always optimal. Let t^i' represent a type which agent i assumes that it can bring more incentive then reporting t^i. We formulize these claims using the following definition.

Definition 2: *Truthful reporting is always an optimal strategy if condition i) below holds, and it is a dominant strategy if, in addition, condition ii) holds*:
i) $t^i \in \text{argmax}_{ti}\{v^i(\hat{o}(X,M,t^i',t^{-i}),t^i) - p^i(X,M,t^i',t^{-i})\}$.
ii) *if* $t^{-i} \neq t^i$, *then for some* t^{-i}, $t^{-i} \notin \text{argmax}_{ti'}\{v^i(\hat{o}(X,M,t^i',t^{-i}),t^i) - p^i(X,M,t^i',t^{-i})\}$.

The above optimality conditions still have loop holes. For instance, it does not cater for the condition that all reports should be potentially pivotal. Thus, we insert the following condition:

Condition iii): *For all* $i\in M$, t^i, $t^i'\in\Theta^i$, *there exists* $t^{-i}\in\Theta^i$, *such that* $\sum_{i\in M} v^i(x(X,M,t^i',t^{-i}),t^i) < V(X,M,t)$.

We now arrive at our main result, which we state as follows:

Theorem 1: *1) Truthful reporting is always an optimal strategy. 2) If all reports are potentially pivotal, then truthful reporting is a dominant strategy.*

Proof: To show that truthful reporting is always optimal, fix profile t of actual types. When agent i reports t^i', the decision chosen is $x(X,M,t^i',t^{-i})$. so, given the formula for i's payment, its payoff is $u^i((x,p), t^i') = v^i(\hat{o}(X,M,t^i',t^{-i}),t^i) - p^i(X,M,t^i',t^{-i}) - h^i(t^{-i})$. Using Equation 4, the gain that i enjoys from the deviation is therefore:

$$\begin{aligned}
&u^i\left((x,p),t^i'\right) - u^i\left((x,p),t^i\right) \\
&= \left[v^i\left(\hat{o}\left(X,M,t^i',t^{-i}\right),t^i\right) - p^i\left(X,M,t^i',t^{-i}\right) - h^i\left(t^{-i}\right)\right] - \\
&\left[v^i\left(\hat{o}\left(X,M,t\right),t^i\right) - p^i\left(X,M,t\right) - h^i\left(t^{-i}\right)\right] \\
&= \sum_{i\in M} v^i\left(\hat{o}\left(X,M,t^i',t^{-i}\right),t^i\right) - \sum_{i\in M} v^i\left(\hat{o}\left(X,M,t\right),t^i\right) \\
&= \sum_{i\in M} v^i\left(\hat{o}\left(X,M,t^i',t^{-i}\right),t^i\right) - V\left(X,M,t\right) \le 0 .
\end{aligned}$$

This proves that truthful reporting is always optimal. By the assumption that all reports are potentially pivotal, for all $t^i' \neq t^i$, there exists t^{-i} such that: $u^i((x,p), t^i') - u^i((x,p), t^i) = \sum_{i\in M} v^i(\hat{o}(X,M,t^i',t^{-i}),t^i) - V(X,M,t) \le 0$. Hence, by definition, truthful reporting is a dominant strategy. ■

Discussion– The derived resource allocation is in its most general form. A careful observation would reveal that its special cases include every possible payment procedure. The most famous of them all is the Vickrey payment. We will reveal how our mechanism's payment schema is equivalent to Vickrey payments. An agent's value for any decision depends only on the objects that the agent acquires, and not on the objects acquired by other agents: $v^i(x,t^i) = v^i(x^i,t^i)$, where $x^i = 1$ if the agent acquires the object and $x^i = 0$ otherwise. The value of not acquiring the object is normalized to zero: $v^i(0,t^i) = 0$. Since the loosing bidders are not pivotal (because their presence does not affect the allocation x), they pay zero in our mechanism. For simplicity let us write v^i for $v^i(1,t^i)$. According to Eq. 4, the price a winning agent pays in the (derived) mechanism is equal to the difference between the two numbers. The first number is the maximum total value to the other agents, when i does not participate in the allocation process, which is $\max_{j\neq i} v^j$. The second number is the total value to the other agents when i

wins, which is zero. Thus, when i wins, it pays $\max_{j \neq i} v^j$, which is equal to the second highest valuation. This is exactly the Vickrey payment. To keep things simple, we will focus on applying the mechanism using Vickrey payments.

3. 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 [10], [14], [18], 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.

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:

$$R_k^i = r_k^i o_k c\left(i, NN_k^i\right), \tag{6}$$

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:

$$W_k^i = w_k^i o_k \left(c\left(i, P_k\right) + \sum_{\forall j \in R_k, j \neq i} c\left(NN_k^i, j\right) \right) \tag{7}$$

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:

$$C_{overall} = \sum_{i=1}^{M} \sum_{k=1}^{N} \left(R_k^i + W_k^i \right). \tag{8}$$

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:

$$X = \sum_{i=1}^{M} \sum_{k=1}^{N} \left[\begin{array}{l} \left(1 - X_{ik}\right) \left[r_k^i o_k \min\left\{ c\left(i,j\right) \mid X_{jk} = 1 \right\} \right. \\ \left. + w_k^i o_k c\left(i, P_k\right) \right] + X_{ik} \left(\sum_{x=1}^{M} w_k^x \right) o_k c\left(i, P_k\right) \end{array} \right] \tag{9}$$

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, 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*: $X_{P_k k} = 1 \quad \forall (1 \leq k \leq N)$.

In the generalized case, the DRP has been proven to be NP-complete [14].

4. Mechanism Applied to the DRP

We follow the same pattern as in Section 2.

The Setup: The distributed system described in Section 3 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 b^i, and b) the replication cost ($RC_k^i = R_k^i + W_k^i$) of an object k to the agent's site i. There are three possible cases:

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

Valuation: The agents in the mechanism value an object k for its benefit that it brings to the agent's site i. This benefit is equivalent to the savings that the object k brings in the total object transfer cost (OTC) if the object k is replicated at site i. This benefit is given as: $B_k^i = RC_k^i - \sum_{x=1}^{M} w_k^x o_k c(i, P_k)$.

Communications: The agents calculate the benefit of every potential object and relay to the mechanism their best possible response, *i.e.*, the object that has the most benefit to them, *i.e.*, $t^i = \text{argmax}_{k \in N} B_k^i$.

Payments: Each agent after acquiring the right to replicate an object onto its site makes a Vickery payment. This right is granted by the mechanism to the agent who projects the highest true type. The mechanism then informs the agent to make a payment equivalent to the second highest benefit projected by an agent in the set M-i. This form of payment was discussed at the end of Section 2, and is termed as Vickrey payments.

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 b^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 08) to the mechanism. The mechanism receives all the corresponding entries, and then chooses the best 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 21 (NN^i_{OMAX}). The object is replicated and payments made to the agent. The mechanism progresses forward till there are no more agents interested in acquiring any data for replication.

The Mechanism

Initialize:
LS, L^i, T_k^i, M, MT

```
01 WHILE LS ≠ NULL DO
02   OMAX = NULL; MT = NULL; P^i = NULL;
03     PARFOR each S^i ∈ LS DO
04       FOR each O_k ∈ L^i DO
05         T_k^i = compute (B_k^i);  /*valuation*/
06       ENDFOR
07     t^i = argmax_k(T_k^i);
08     SEND t^i to M; RECEIVE at M t^i in MT;
09     ENDPARFOR
10 OMAX = argmax_k(MT);   /*Choose the global dominate valuation*/
11 DELETE k from MT;
12 P^i = argmax_k(MT);          /*Calculate the payment*/
13 BROADCAST OMAX;
14 SEND P^i to S^i; RECEIVE at S^i  /*The winning agent pays this amount*/
15 SEND P^i to M; RECEIVE at M      /*Send the payment*/
16 Replicate O_OMAX;
17 b^i=b^i - o_k;                  /*Update capacity*/
18 L^i = L^i - O_k;                /*Update the list*/
19 IF L^i = NULL THEN SEND info to M to update LS = LS - S^i;
20     PARFOR each S^i ∈ LS DO
21       Update NN^i_OMAX
22     ENDPARFOR                  /*Get ready for the next round*/
23 ENDWHILE
```

Figure 1: The Mechanism.

Theorem 2: *The mechanism takes* $O(MN^2)$ *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 20). 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 $O(MN^2)$.■

5. Experiments and 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 resource allocation mechanism was implemented using IBM Pthreads.

Table 1: Parameter interval variance.

Topology	Mathematical Representation	Parameter Interval Variance
SGRG [10] (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 [10] (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 [10] (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 [10] (9 topologies)	Random layout with link distance having a lognormal distribution [7].	M={8.455,9.345,9.564}, σ={1.278,1.305,1.378}.

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 topologies the distance of the link between nodes was equivalent to the communication cost. Table 1 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 replica 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.*, 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 install 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.

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. We select from [18] the greedy approach (Greedy) for comparison because it is shown to be the best compared with four other approaches (including the proposed technique in [13]); thus, we indirectly compare with four additional approaches as well. Algorithms reported in [10] (branch and bound (Aε-Star)), [11] (Dutch (DA) and English auctions (EA)) and [14] (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 [10] 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 cost node by a factor greater than 1+ε. The technique works similar to A-Star, with the exception that the node selection 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 [18], 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.

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, the agent's valuation should be divided by *n* [11].

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 [11] 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 [14], 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.

5.1. Comparative Analysis

We study the behavior of the placement techniques when the number of sites increases (Figure 2), by setting the number of objects to 2000, while in Figure 3, 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 extreme cases. The first observation is that the derived resource allocation mechanism (MECH) 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 MECH) 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 [10] and [14].

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 [14]. But there the authors had used a uniformly distributed link cost topology, and their traffic was based on the Zipf distribution [21], while the traffic access logs of the World Cup 1998 are more or less double-Pareto in nature. 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 MECH is observable. Both the techniques showed high performance, with a slight edge in favor of MECH.

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 not equally read intensive, increase in the storage capacity

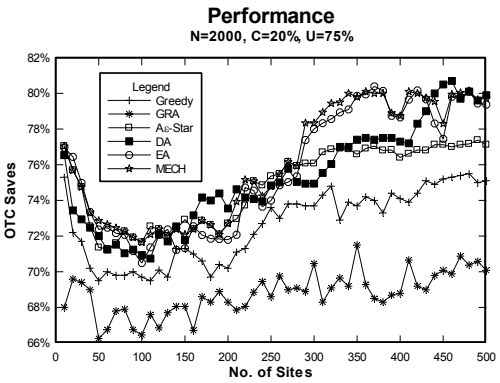

Figure 2: OTC savings versus number of sites.

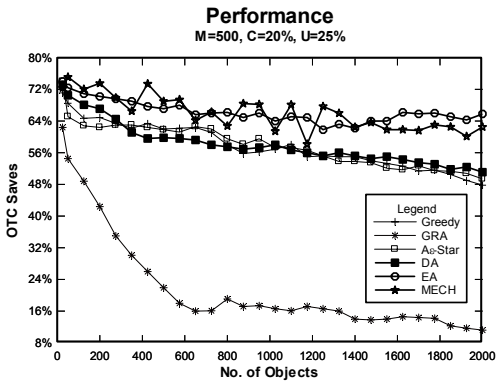

Figure 3: OTC savings versus number of objects.

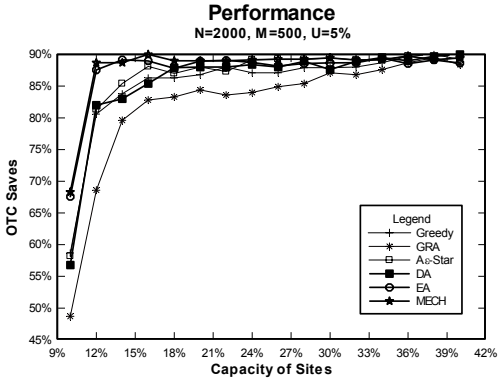

Figure 4: OTC savings versus site capacity.

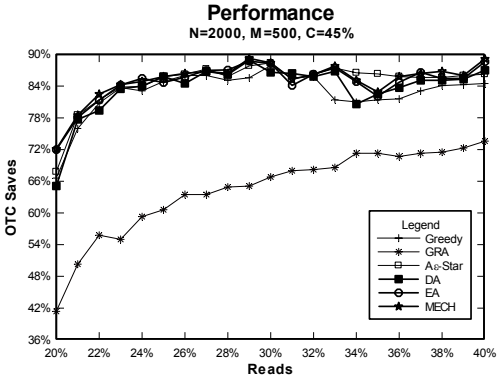

Figure 5: OTC savings versus reads.

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

Now, 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. The plots in Figures 5 and 6 show the results of read and update frequencies, respectively. A clear classification can be made between the algorithms. Aε-Star, DA, EA, Greedy and MECH 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 [14]). 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 MECH never tend to deviate from their global view of the problem domain.

Next, we compare the termination time of the algorithms. Various problem instances were recorded

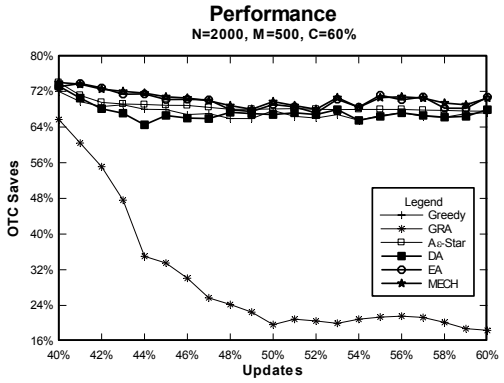

Figure 6: OTC savings versus updates.

Table 3: Running time (sec.) [*C*=20%, *U*=45%].

Problem Size	Greedy	GRA	Aε-Star	DA	EA	MECH
M=20, *N*=50	70.11	92.66	97.18	**25.16**	39.36	30.86
M=20, *N*=100	76.59	96.40	102.97	**27.71**	41.21	36.12
M=20, *N*=150	78.26	101.01	113.85	**32.44**	54.57	43.21
M=30, *N*=50	95.24	126.92	140.78	**38.45**	59.25	49.01
M=30, *N*=100	109.17	125.04	148.83	**39.21**	63.14	54.33
M=30, *N*=150	135.21	148.59	179.74	**45.96**	68.20	59.15
M=40, *N*=50	126.40	154.13	198.77	**42.66**	76.27	63.39
M=40, *N*=100	134.65	168.48	236.67	**43.62**	77.16	73.56

Table 2: Running time (sec.) [*C*=45%, *U*=15%].

Problem Size	Greedy	GRA	Aε-Star	DA	EA	MECH
M=300, *N*=1400	206.26	326.82	279.45	**95.64**	178.90	126.32
M=300, *N*=1450	236.61	379.01	310.12	**115.19**	185.15	134.65
M=300, *N*=1500	258.45	409.17	333.03	**127.10**	191.24	157.46
M=300, *N*=1550	275.63	469.38	368.89	**143.94**	197.93	171.21
M=300, *N*=2000	298.12	475.02	387.94	**158.45**	204.29	194.29
M=400, *N*=1400	348.53	536.96	368.03	**187.26**	223.56	220.06
M=400, *N*=1450	366.38	541.12	396.96	**192.41**	221.10	236.35
M=400, *N*=1500	376.85	559.74	412.17	**208.92**	245.47	238.63
M=400, *N*=1550	389.71	605.63	415.55	**215.24**	269.31	241.93
M=400, *N*=2000	391.55	659.39	447.97	**224.18**	274.24	253.21
M=500, *N*=1400	478.10	689.44	511.69	**257.96**	301.72	279.42
M=500, *N*=1450	485.34	705.07	582.71	**269.45**	315.13	293.24
M=500, *N*=1500	511.06	736.43	628.23	**278.15**	324.26	310.55
M=500, *N*=1550	525.33	753.50	645.26	**289.64**	331.57	325.32
M=500, *N*=2000	539.15	776.99	735.36	**312.68**	345.94	334.90

Table 4: Average OTC (%) savings.

Problem Size	Greedy	GRA	Aε-Star	DA	EA	MECH
N=150, *M*=20 [*C*=20%,*U*=25%]	70.46	69.74	74.62	70.15	73.15	**74.73**
N=200, *M*=50 [*C*=20%,*U*=20%]	73.94	70.18	77.42	72.66	77.41	**78.16**
N=300, *M*=60 [*C*=35%,*U*=5%]	71.66	65.94	72.01	70.22	71.23	**73.45**
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	**73.35**
N=2000, *M*=500 [*C*=10%,*U*=60%]	67.03	63.37	72.16	68.63	69.24	**73.28**

with *C*=20%, 45% and *U*=45%, 15%. 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 DA and MECH terminated faster than all the other techniques, followed by EA, Greedy, Aε-Star and GRA. MECH with the maximum problem setup terminated approximately in 334.90 seconds (Table 2 last entry). Tables 2 and 3 summarize the results.

Lastly, Table 4 shows the quality of the solution in terms of OTC percentage for eight 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 MECH 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 MECH.

6. Related Work

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

1. Facility Location: In [8], 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 [8] does not fully capture the concept of replicating a single object/site over a fixed number of hosts [15].

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], [12], [16]. 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 [12], provided a guaranteed optimal result for Internet data replication, but has little practical use [15], 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 [13] 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 [18]. There the authors proposed a greedy algorithm that outperformed other techniques including the work reported in [13].

4. Capacity-constrained Optimization: In [9], 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 [8]; therefore, they can not handle the dynamics of the system [15].

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 [16]. 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 [10].

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 [15]. Some of the significance work in this line of pursuit is reported in [5] and [14].

A number of bibliographies and reading materials for web content replication are also available online, e.g., [6]. A brief overview of replication and its challenges are provided in [15] and [19].

7. 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-known techniques, such as: branch and bound, greedy, game theoretical auctions, and genetic algorithms revealed that the proposed mechanism exhibited 5%-10% improvement in the solution quality and incurred fast execution time.

References

[1] P. Apers, "Data Allocation in Distributed Database Systems," *ACM Trans. Database Systems*, 13(3), pp. 263-304, 1988.

[2] D. Appleby, and S. Steward, "Mobile Software Agents for Control in Telecommunications Networks," *BT Technology Journal*, 12(2), pp. 104-113, 1994.

[3] M. Arlitt and T. Jin, "Workload characterization of the 1998 World Cup Web Site," Tech. report, HP 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] B. Davison, "A Survey of Proxy Cache Evaluation Techniques," in *Proc. of the 4th International Web Caching Workshop*, 1999, pp. 67-77.

[7] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM Trans. Networking*, 9(4), pp. 253-285, 2001.

[8] S. Jamin, C. Jin, T. Kurc, D. Raz and Y. Shavitt, "Constrained Mirror Placement on the Internet," in *Proc. of the IEEE INFOCOM*, 2001, pp. 31-40.

[9] 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. 455-466.

[10] S. Khan and I. Ahmad, "Heuristic-based Replication Schemas for Fast Information Retrevial over the Internet," in *Proc. of 17th International Conference on Parallel and Distributed Computing Systems*, San Fransisco, U.S.A., 2004.

[11] S. Khan and I. Ahmad, "Internet Content Replication: A Solution from Game Theory," Tech. report, CSE-2004-05, 2004.

[12] 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.

[13] 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, pp. 1282-1290.

[14] T. Loukopoulos, and I. Ahmad, "Static and Adaptive Distributed Data Replication using Genetic Algorithms," *Journal of Parallel and Distributed Computing*, 64(11), pp. 1270-1285, 2004.

[15] T. Loukopoulos, I. Ahmad, and D. Papadias, "An Overview of Data Replication on the Internet," in *Proc. of ISPAN*, 2002, pp. 31-36.

[16] 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.

[17] N. Nisan and A. Ronen, "Algorithimic Mechanism Design," in *Proc. of ACM STOC*, 1999, pp. 129-140.

[18] L. Qiu, V. Padmanabhan and G. Voelker, "On the Placement of Web Server Replicas," in *Proc. of the IEEE INFOCOM*, 2001, pp. 1587-1596.

[19] M. Rabinovich, "Issues in Web Content Replication," *Data Engineering Bulletin,* 21(4), pp. 21-29, 1998.

[20] W. Vickrey, "Counterspeculation, Auctions and Competitive Sealed Tenders," *Journal of Finance*, pp. 8-37, 1961.

[21] G. Zipf, Human Behavior and the Principle of Least-Effort, Addison-Wesley, 1949.