# A Pure Nash Equilibrium Guaranteeing Game Theoretical Replica Allocation Method for Reducing Web Access Time

Samee Ullah Khan and Ishfaq Ahmad
*Department of Computer Science and Engineering*
*University of Texas, Arlington, Texas, USA*
*{sakhan, iahmad}@cse.uta.edu*

## Abstract

*This paper proposes a **non-cooperative** game theoretical **r**eplica allocation technique (NCOR) to reduce user perceived Web access delays. NCOR uses distributed agents that because of their local knowledge act in a self-interested manner in order to enhance the performance of the servers that they represent. This can lead to some performance gains for some servers but has the potential to negatively impact the overall system's performance. NCOR uses an effective cost model to guarantee the overall system performance gain despite the self-interested actions of these agents. With spontaneous and non-deterministic strategies, the system can exhibit Nash equilibrium. However, that may or may not guaranteed system-wide performance at a given time. Furthermore, their can be multiple Nash equilibria, making it difficult to decide which one is the best. Instead, we use the notion of pure Nash equilibrium, which if achieved is guaranteed to ensure stable optimal performance. Pure Nash equilibrium can be only achieved by deterministic strategies. In general, the existence of a pure Nash equilibrium is remarkably hard to achieve; however, we prove the existence of such an equilibrium in NCOR. Experimental comparisons with several non-game theoretical techniques reveal that NCOR maintains superior solution quality, in terms of lower communication cost and reduced execution time.*

## 1. Introduction

A number of techniques for object-based Web content replication have been proposed with the underlying assumption that servers cooperate with one another in order to layout a replica schema that optimizes the overall system performance. For instance, almost all content distribution networks (CDNs) related replica allocation methods ([5], [6]) rely on a centralized decision making body which optimizes a given objective (such as to reduce the communication cost) regardless of the costs incurred by each server [7]. In reality, servers aim at maximizing their own benefits, possibly at the expense of the global optimal [3].

To study this self-interested behavior, we make use of game theoretical techniques and abstract the Web (or large scale distributed computing system) as an agent based model. Each server in the system is represented by an agent which is a computational entity that is capable of autonomous behavior 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 [9]. These agents are motivated by their individual interests and compete in a **non-co**operative **r**eplica allocation game (*NCOR*). In *NCOR* each agent has two possible actions for each object. If an access is made to an object that is located at a nearby server, then the agent is better off redirecting the request to that server. On the other hand if the object is located at a far off server, then the agent is better off replicating that object.

The goal of this paper is to see whether these self-interested agents in *NCOR*, can layout replica schemas that converge to global optimum solution(s) targeted towards reducing the communication cost induced by accessing the objects. Using game theory, we show that in the worst-case scenario, the system as a whole resides in a social optimum domain, *i.e.*, the solution quality can never be worse than a pareto-optimal solution. This social optimum domain is used as the basis to prove that *NCOR* indeed converges to global optimum solution(s) that conform to pure Nash equilibrium(s) when the self-interested agents play deterministic strategies according to *NCOR*'s cost model. Pure Nash equilibrium is different from the classical Nash equilibrium in the sense that the former results when the strategies played are deterministic, while the later results when the strategies played are

non-deterministic. Also, a system will achieve a global optimum solution throughout the lifespan of the system once such a pure Nash equilibrium is achieved. This is certainly not the case when a system exhibits a classical Nash equilibrium, for the simple reason that there could be multiple Nash equilibria, making it difficult to decide which one is the best. An elaborate discussion on these two types of Nash equilibria, their properties and differences can be found in [4] and [19].

The proposed *NCOR* is experimentally compared against: 1) branch and bound [8], 2) greedy [18] and 3) genetic algorithm [15] using GT-ITM [20] and Inet [2] network topology generators and Soccer World Cup 1998 traffic logs [1]. The experimental results reveal that *NCOR* maintains superior solution quality with reduced execution time.

The remainder of this paper is organized as follows. In Section 2 we provide a formal description of the data replication problem. Section 3 concentrates on modeling the data replication problem as a non-cooperative replica allocation game. Comparative experimental results, related work and final concluding remarks in Sections 4, 5 and 6, respectively.

## 2. The data replication problem

Consider a distributed system comprising $M$ servers, with each server 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 server $i$ where $1 \le i \le M$. The $M$ servers of the system are connected by a communication network. A link between two servers $S_i$ 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 servers $S_i$ and $S_j$. If the two servers 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 server $S_i$ to the server $S_j$. Without the loss of generality we assume that $c(i,j) = c(j,i)$. Let there be $N$ objects, each identifiable by a unique name $O_k$ and size in simple data unites $o_k$ where $1 \le k \le N$. 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 server 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 server of the $k$-th object. Each primary server $P_k$, contains information about the whole replication scheme $R_k$ of $O_k$. This can be done by maintaining a list of the servers where the $k$-th object is replicated at, called from now on the *replicators* of $O_k$. Moreover, every server $S_i$ stores a two-field record for each object. The first field is its primary server $P_k$ and the second the nearest neighborhood server $NN_k^i$ of server $S_i$ which holds a replica of object $k$. In other words, $NN_k^i$ is the server 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 server of $O_k$. Another possibility is that $NN_k^i = P_k$, if the primary server is the closest one holding a replica of $O_k$. When a server $S_i$ reads an object, it does so by addressing the request to the corresponding $NN_k^i$. For the updates we assume that every server can update every object. Updates of an object $O_k$ are performed by sending the updated version to its primary server $P_k$, which afterwards broadcasts it to every server in $R_k$.

For the DRP under consideration, we are interested in minimizing the total Object Transfer Cost (OTC) due to object movement, since the communication cost of control messages has minor impact to the overall performance of the system. 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 $S_i$s' reading requests for object $O_k$, addressed to the nearest server $NN_k^i$. This cost is given by:

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

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

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

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:

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

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

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

Servers 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 server of $O_k$ . Servers 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* $C_{overall}$, s*ubject to the storage capacity constraint*: $\sum_{k=1}^{N} X_{ik} o_k \le s_i \ \forall (1 \le i \le M)$, *and subject to the primary copies policy:* $X_{P_k k} = 1 \ \forall (1 \le k \le N)$."

## 3. Non-cooperative replica allocation game

Before we discuss the exact game structure of *NCOR* (the non-cooperative replica allocation game), it is essential to lay down the basis of *NCOR*. We start by defining:

**Definition 1 (Feasible Strategies):** *An agent i's strategy is termed feasible, $\varphi_i$, when the two constraints of the data replication problem (storage and no de-allocation of the primary copy) are met before a decision to replicate an object $O_k$ can be undertaken.*

Of all these possibly infinity many feasible strategies, let $\varsigma_i \in \varphi_i$ be a strategy chosen by an agent *i*, where $\varsigma_i$=1 means object is replicated and $\varsigma_i$=0 means it is not. (Note that $\varsigma_i$ only focus on a specific object $O_k$. Therefore it is not necessary to write $\varsigma_i$ as $\varsigma_{i,k}$ or any other notation that would differentiate any two objects.) Since each agent chooses $\varsigma_i \in \varphi_i$ independently (keeping both the constraints at par), we can look at the replication of each object $O_k$ as a separate game, and combine the pure Nash equilibrium of these games to obtain a pure Nash equilibrium of the multi-object game, *NCOR*. (This argument would become clearer when Definition 3 and Lemma 1 are reviewed.)

**Definition 2 (Strategy Profile)** [16]**:** *A strategy profile $\varsigma=(\varsigma_{1,\dots,}\varsigma_M)$ is a set of strategies for each agent which fully specifies all of its actions. A strategy profile must include one and only one strategy for every agent.*

For convenience we can also write $\varsigma$ as $(\varsigma_i,\varsigma_{-i})$, where $\varsigma_i$ is the strategy of agent *i*, and $\varsigma_{-i}$ is the set of strategies of all other agents in *NCOR* excluding agent *i*. Given $\varsigma$ one can easily find out which agents have opted to replicate $O_k$.

**Definition 3 (Pure Nash Equilibrium)** [19]**:** *A situation in a non-cooperative game in which agents play using a set of deterministic strategies whereby no agent can improve its benefit by changing its strategy unilaterally.*

**Lemma 1 (Combining Pure Nash Equilibriums)** [4]**:** *If two games are known to have pure Nash equilibriums, then the union of the games is also guaranteed to have a pure Nash equilibrium.* ■

Thus, if we are able to prove that a given $\varsigma$ conforms to a pure Nash equilibrium, then $\cup\varsigma_i$ also conforms to a pure Nash equilibrium. Conversely, if $\chi(\varsigma)$ is the cost function associated with $\varsigma$, then $\Sigma\chi(\varsigma_i)$ over all *N* objects is the cost associated with $\cup\varsigma_i$. Based on this, we can give a formal mathematical definition of pure Nash equilibrium as:

**Definition 4 (Pure Nash Equilibrium (Mathematically)):** *Let $(\varsigma,\chi)$ be a game, where $\varsigma$ is the set of strategy profiles and $\chi$ is the cost function. When each agent i plays $\varsigma_i$ then agent i incurs a cost $\chi_i(\varsigma)=\chi_i(\varsigma_{1,\dots,}\varsigma_M)$. $\varsigma^*$ is pure Nash equilibrium if for any deviation $\varsigma_i$ by an agent i is not beneficial, that is $\chi_i(\varsigma_i,\varsigma_{-i}^*) \le \chi_i(\varsigma_i^*,\varsigma_{-i}^*)$.*

**Definition 5 (Stability of a Pure Nash Equilibrium)** [19]**:** *Equilibrium is stable if an infinitesimal small change in the strategy of one agent leads to a situation where the following hold:*

*(a): The agent who did not change has no better strategy in the new circumstance.*

*(b): The agent who did change is now playing with a strictly worse strategy.*

It is also important that we accentuate on the difference between $\varsigma$ and $R_k$ (replica schema of $O_k$). If we are given $R_k$, we only know which severs hold a copy of $O_k$, but if $\varsigma$ is given, we can also find out which agents have not opted to replicate $O_k$ along with their corresponding cost functions.

**Definition 6 (Replica Schema):** *A replica schema, $R_k$, for object $O_k$ is the set of servers that replicates $O_k$.*

We now proceed with describing the game structure of *NCOR*.

**The Setup:** The Web (or large scale distributed computing system) described in Section 2 is considered, where each server is represented by an agent, *i.e.*, *NCOR* contains *M* agents. Although *NCOR* is non-cooperative in nature, yet there is no information hiding. That is, the network topology, the size of the object and location of replicas are all public knowledge. The only information that is private to each agent is the frequency of reads and writes for each object from its server.

**Cost model:** We first concentrate on deriving the cost model for a single object. This will be expanded later on to fully encapsulate the multi-object data replication problem as described in Section 2.

Let $\varphi_i$ be the set of feasible strategies for an agent *i*. For $O_k$, the agent chooses a strategy $\varsigma_i \in \varphi_i$ that describes its desire to replicate or otherwise. Thus, given a strategy profile $\varsigma$, we say that an agent *i* incurs

a cost $\chi_i(\varsigma)$ if it considers replicating object $O_k$. This cost is given as:

$$\chi_i(\varsigma) = \sum_{k\in\varsigma_i}\left[w_i^k o_k\left(\sum_{\forall j\in R_k, i\neq j} c(P_k, j)\right)\right], \\ +\sum_{k\notin\varsigma_i}\left[r_i^k o_k c\left(i, NN_i^k\right) + w_i^k o_k c\left(i, P_k\right)\right] \quad (5)$$

which implies that if an agent replicates $O_k$, then the cost incurred due to reads is $0 = r_i^k o_k c(i,NN_i^k)$ since $NN_i^k = i$. The cost incurred due to local writes (or updates) is equal to zero since the copy resides locally, but whenever $O_k$ is updated anywhere in the network, agent $i$ has to continuously update $O_k$'s contents locally as well. Therefore, the aggregate cost of writes is equivalent to $w_i^k o_k \Sigma_{\forall (j\in Rk),\, i\neq j}\, c(P_k,j)$. On the other hand if an agent does not replicate $O_k$, then the cost incurred due to reads is equal to $r_i^k o_k c(i,NN_i^k)$, and the cost incurred due to writes is equal to $w_i^k o_k c(i,P_k)$ since it only has to send the update to the primary server which then broadcasts the update based on $R_k$ to the agents who have replicated the object.

Equation 5 captures the dilemma faced by an agent $i$ when considering replicating $O_k$. If agent $i$ replicates $O_k$ then it brings down the read cost to zero, but now it has to keep the contents of $O_k$ up to date. If agent $i$ does not replicate $O_k$, then it reduces the overhead of keeping the contents up to date, but now it has to redirect the read requests to the nearest neighborhood server which holds a copy of $O_k$. Keeping these cost considerations in mind, for an object $O_k$ each agent $i$ has two strategies: (0) not to replicate or (1) to replicate; allowing us to rewrite Equation 5 in a visually appealing form:

$$\chi_i(\varsigma) = \varsigma_i\left[w_i^k o_k\left(\sum_{\forall j\in R_k, i\neq j} c(P_k, j)\right)\right] \\ +(1-\varsigma_i)\left[r_i^k o_k c\left(i, NN_i^k\right) + w_i^k o_k c\left(i, P_k\right)\right] \quad (6)$$

**Discussion on Cost Model:** Each agent $i$'s cost to replicate an $O_k$ (or otherwise) sturdily relies on the access (both read and write) frequencies, the replica locations, and the size of $O_k$ ($o_k$). Essentially, *NCOR* starts with a given (possibly a random) replica schema, and evolves it into a replica schema that exhibits pure Nash equilibrium as each agent alters its strategy so as to minimize its cost. That is, a pure Nash equilibrium $(\varsigma^*_i,\varsigma^*_{-i})$ for *NCOR* identifies a replica schema $R_k$ such that $\forall i\in M$, $i\in R_k$ if and only if $\varsigma^*_i=1$. Recall that there can be infinitely many feasible strategies, which in turn means that there can be infinitely many replica schemas that are identifiable by a pure Nash equilibrium. Let $\epsilon$ represent the set of all possible pure Nash equilibrium replica schemas and we say:

**Definition 7 (Pure Nash Equilibrium Replica Schema):** *A replica schema belongs to the set of pure Nash equilibrium replica schemas $R_k\in\epsilon$ if and only if each agent $i\in M$ chooses a feasible strategy $\varsigma_i\in\varphi_i$ such that when each agent $i$ plays $\varsigma_i$, it cannot improve its cost by changing its strategy unilaterally, that is $\chi_i(\varsigma_i,\varsigma_{-i}^*)\leq\chi_i(\varsigma_i^*,\varsigma_{-i}^*)$, where $\varsigma^*$ is a pure Nash equilibrium.*

Keeping Definition 7 in mind, for *NCOR* we can straightforwardly deduce the following:

$R_k\in\epsilon$ if and only if:

(a) $\forall i\in M$, $\exists j\in R_k$ such that $c(i,j)$ is minimum. (7)

(b) $\forall j\in R_k$, $\nexists\, j'\in R_k$ such that $c(j,j')<c(i,j)$. (8)

We observe that for an object $O_k$'s replica schema to be in a state of pure Nash equilibrium, each agent $i$ has placed $O_k$'s replica at a server that incurs minimum possible communication cost from $S^i$. (That is, if the replica is not placed at $i$, then it is replicated at a server $j$ which has the minimum cost of communication from $S^i$, compared to any other server in the system.) On the other hand if agent $i$, has already replicated object $O_k$, then there is no benefit for agent $i$ to drop the replica since the location incurs a minimal communication cost to at least one server (which holds the replica). Note that what we have just discussed above (Equations 7 and 8) is equivalent to the two conditions ((a) and (b), respectively) of equilibrium stability as stated in Definition 5. With this said, we now expand this single object replica allocation cost model to the multi-object data replication problem. First, let us see what is the cost incurred by the society (all $M$ agents) as a whole.

**Definition 8 (Social Optimum)** [16]**:** *The maximum net benefits for everybody in society, regardless of who enjoys the benefits.*

Social optimum is the analogous concept of optimum resource allocation [19]. In this paper since the resources are replicas, we can say the social optimum is equivalent to the optimum replica allocation, and we note that:

**Definition 9 (Pareto Optimum)** [16]**:** *A pareto optimum is a situation in which it is not possible to make any one agent better off without making some other agent worse off.*

**Lemma 2 (A Condition for Pareto Optimum)** [16]**:** *A pareto optimum is not possible unless the net benefits for every agent in society are maximized.* ■

In an ideal price based competitive economy, achieving a social (or pareto) optimum is no big deal [16]. Every agent maximizes its private benefit, but since every agent pays for any benefits it receives, and bears only the corresponding costs, the result of this private benefit maximization is that social net benefits are maximized. However, when pricing is not involved (as is the case in *NCOR*), it is no longer trivial to guarantee social optimum. We write the social cost for *NCOR* as:

$$\chi(\varsigma) = \sum_{i=1}^{M} \chi_i(\varsigma) . \quad (9)$$

Refining Equation 9 using the definition of social optimum we say:

$$\chi(\varsigma_{\min}) = \min_{\varsigma} \chi(\varsigma) . \quad (10)$$

Equation 10 encapsulates the notion of cooperation among all agents to layout a replica schema that incurs minimum communication cost. But the agents are self-interested, hence we use $\chi(\varsigma_{\min})$ as an important measure for the solution quality. What the agents are trying to achieve in conjunction to $\chi(\varsigma_{\min})$ is:

$$\chi(\varsigma) = \text{minimize} \sum_{i=1}^{M} \chi_i(\varsigma) . \quad (11)$$

Using Lemma 1 we say that:

$$\chi(\varsigma) = \text{minimize} \sum_{k=1}^{N} \sum_{i=1}^{M} \chi_i(\varsigma) . \quad (12)$$

Expanding Equation 12 using Equation 5 we obtain:

$$\chi(\varsigma) = \text{minimize} \sum_{k=1}^{N} \sum_{i=1}^{M} \left[ \begin{array}{l} \left[ w_i^k o_k \left( \sum_{\forall j \in R_k, i \neq j} c(P_k, j) \right) \right] + \\ \left[ r_i^k o_k c\left(i, NN_i^k\right) + w_i^k o_k c\left(i, P_k\right) \right] \end{array} \right] . \quad (13)$$

Thus, the pure Nash equilibrium in *NCOR may* exist when over the set of all objects $N$; all $M$ agents maximize their benefits (by minimizing the communication costs). A closer look at Equation 13 reveals that it is nothing more than the minimization problem described by Equation 3. Hence, the following holds:

**Theorem 1 (Equivalence):** *The data replication problem and the non-cooperative replica allocation game are equivalent and have the same objective.* ■

Based on the above discussion we describe the procedure for *NCOR* in Figure 1.

**Description of *NCOR* Procedure:** We maintain a list $L^i$ at each server. This list contains all the objects that can be replicated by agent $i$ onto server $S^i$. (In other words $L^i$ represents the set of feasible strategies.) We can obtain this list by examining the two constraints of the data replication problem. List $L^i$ would contain all the objects that have their size less then the total available space $b^i$. Moreover, if server $S^i$ is the primary host of some object $O_k$, then $O_k$ would not be in $L^i$. We also maintain a list $LS$ containing all servers that can replicate an object, *i.e.*, $S^i \in LS$ if $L^i \neq$NULL. The algorithm works iteratively. In each step the servers calculate the cost of replicating an object $O_k$ using Equation 6 (Line 04). This cost is compared to the current cost incurred by the server. If this new cost is less or equal to the current cost, then the server opts to replicate that object. After a decision of replication is taken, each server updates the server storage capacity and the nearest neighbor list (Lines 8 and 9). Servers also evict the object from the list $L^i$ since a decision on it has already been undertaken (Line 12). This procedure continues till the list $L^i$ becomes empty. When $L^i$ becomes empty, a message is sent to the moderator $M$ (which is a control thread) to evict the server from the game since it is no longer able to undertake any further decisions (Line 14). We would like to clarify that the list $L^i$ is dynamically update in accordance to with the changes of server capacity. For example, if by replicating an object a server exhausts all of its storage capacity, then $M$ dynamically adjusts $L^i$ to empty.

**The *NCOR* Procedure**

**Initialize:**
$LS$, $L^i$, $\chi^*_i(\varsigma)=\infty$, $M$, $\varsigma$=NULL

```
01 WHILE LS ≠ NULL DO
02   PARFOR each S^i∈LS DO
03         FOR each O_k∈L^i DO
04           Compute χ_i(ς)=min{χ_i(ς)|ς_i=1, χ_i(ς)|ς_i=0}; /* Eq. 6 */
05             IF χ_i(ς)≤χ*_i(ς) THEN
06                χ*_i(ς)=χ_i(ς);   /* Update current best cost */
07                ς_i=1;            /* Replicate object O_k */
08                b^i=b^i - o_k;    /* Update capacity */
09                Update NN_k^i; /* Update the nearest neighbors */
10             ELSE
11                ς_i=0;          /* Do not replicate object O_k */
12           L^i = L^i - O_k;          /* Update the list*/
13           IF L^i = NULL THEN
14                SEND info to M to update LS = LS - S^i;
15         ENDFOR
16   ENDPARFOR       /*Social cost achieved Equation 9 */
17 ENDWHILE /* Pure Nash equilibrium achieved Th. 2 and 3 */
```

**Figure 1: The Pseudo-code for *NCOR* Procedure.**

**Theorem 2 (Existence of Pure Nash Equilibrium):** *A pure Nash equilibrium exists for the self-interested agents, if they play according to the cost model of single object NCOR.*

**Proof:** Let $M$ denoted the set of agents in the system, where each agent represents a server. Let $c(i,NN_i^k)$ represent the cost of assessing object $O_k$ from server $S^i$ to replicated at the nearest server from $NN_i^k$. Let $M'$ represent the set of servers for which a server $S^i$ incurs the minimum communication cost for all servers $m \in M'$, *i.e.*, $M' = \{m | c(i,m) \leq c(i,NN_i^k)\}$. Essentially, *NCOR* chooses a server $m \in M'$ such that $c(i,m) \leq c(i,NN_i^k)$ $\forall i \in M$ to hold the replica. After allocating a replica at $m$, it is removed along with all servers $m \in M'$ from $M$. This is done because no servers ($m$) has incentive to replicate $O_k$ since it can access m's replica at a lower or equal cost than $NN_i^k$'s replica. *NCOR* iteratively chooses a server m till $M=\varnothing$. Again, since at each iteration $m$ is the remaining server with minimum $c(i,m)$, no other server can be selected to replicate $O_k$ such that $c(i,NN_i^k) \leq c(i,m)$. Hence, no agent can gain benefit by unilaterally opting to replicate an object without disturbing the equilibrium. ■

**Theorem 3 (*NCOR* Pure Nash Equilibrium):** *A pure Nash equilibrium exists for the multi-object NCOR.*
**Proof:** Follows from Lemma 1 and Theorem 2. ■

## 4. Experimental results and discussions

We performed experiments on a 440MHz Ultra 10 machine with 512MB memory. The experimental evaluations were targeted to benchmark the placement policies. *NCOR* was implemented using Ada and Ada GNAT's distributed systems annex GLADE [17].

To establish diversity in our experimental setups, the network connectively was changed considerably. We used GT-ITM [20] for the network topologies, the procedure for which we explain below.

A random graph $G(M,P(\text{edge} = p))$ with $0 \leq p \leq 1$ contains all graphs with nodes (servers) $M$ in which the edges are chosen independently and with a probability $p$. The pure random topologies were obtained with $p = \{0.4, 0.5, 0.6, 0.7, 0.8\}$. In each of these topologies the distance between two serves was reversed mapped to the communication cost of transmitting a 1kB of data and the latency on a link was assumed to be equivalent to that of a copper wire, *i.e.*, $2.8\times10^{-8}$ m/s.

To evaluate the replica allocation methods under realistic traffic patterns, we used the access logs collected at the Soccer World Cup 1998 web server [1]. Each experimental setup was evaluated thirteen times, *i.e.*, only the Friday (24 hours) logs from May 1, 1998 to July 24, 1998. (The Friday logs have the heaviest traffic compared to any other day of the week.) To process the logs, we wrote a script that returned: only those objects which were present in all the logs (25,000 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 server was mimicked by choosing random locations. The capacities of the servers $C$% 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 miscellanies to benchmark object updates. The updates were randomly pushed onto different servers, 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.

Since the access logs are of the year 1998, we first used Inet [2] topology generator to estimate the number of nodes in the network. This number came up to be 3718, *i.e.*, there were 3718 AS-level nodes in the Internet at the time when the Soccer World Cup 1998 was being played. Therefore, we set the upper bound on the number of servers in the system at $M = 3718$.

**Comparative algorithms:** For comparisons, we chose three types of replica allocation methods. To provide a fair comparison, the assumptions and system parameters were kept the same in all the methods. For the data replication problem, the non-game theoretical techniques proposed in [8], [13], [15] and [18] are the only ones that address the problem domain similar to ours. We select from [18] the greedy approach (Greedy) for comparison because it is shown to be the best compared with 4 other approaches (including the proposed technique in [13]); thus, we indirectly compare with 4 additional approaches as well. Algorithms reported in [8] (the efficient branch and bound based technique Aε-Star) and [15] (the genetic algorithm based method GRA) are also among the chosen techniques for comparisons. We encourage the readers to obtain an insight on the comparative techniques from the referenced papers.

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

**Comparative analysis:** First, we observe the effects of increase in storage capacity. 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 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 2, which shows the performance of the algorithms. The performance between all approaches except GRA was within 15% of each other. *NCOR* and Greedy 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 performed the worst, but observably gained the most OTC savings (49%) followed by Greedy with 44%. Further experiments

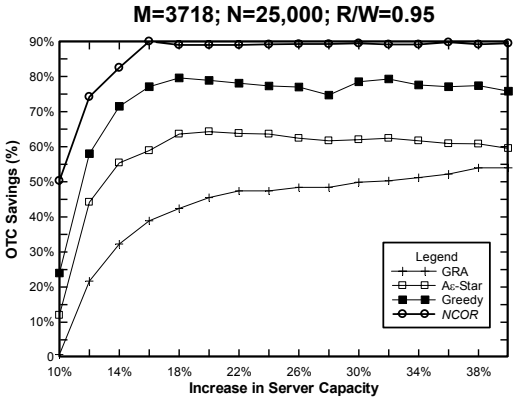


**Figure 2: OTC savings versus capacity.**

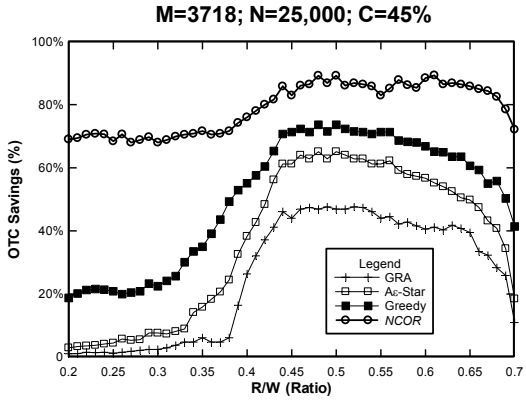


**Figure 3: OTC savings versus read/write ratio.**

**Table 1: Running time of the replica placement methods in seconds.**

**(a): Small problem instances [*C*=20%, *R/W*=0.45]**

| Problem Size | Greedy | GRA | Aε-Star | *NCOR* |
|---|---|---|---|---|
| *M*=200, *N*=500 | 84.13 | 111.19 | 116.61 | **37.03** |
| *M*=200, *N*=1000 | 91.90 | 115.68 | 123.56 | **43.34** |
| *M*=200, *N*=1500 | 93.91 | 121.21 | 136.62 | **51.85** |
| *M*=300, *N*=500 | 114.28 | 152.30 | 168.93 | **58.81** |
| *M*=300, *N*=1000 | 131.00 | 150.04 | 178.59 | **65.19** |
| *M*=300, *N*=1500 | 162.25 | 178.30 | 215.68 | **70.98** |
| *M*=400, *N*=500 | 151.68 | 184.95 | 238.52 | **76.06** |
| *M*=400, *N*=1000 | 161.58 | 202.17 | 284.00 | **88.27** |
| *M*=400, *N*=1500 | 169.29 | 245.31 | 324.75 | **95.55** |

**(b): Large problem instances [*C*=45%, *R/W*=0.85]**

| Problem Size | Greedy | GRA | Aε-Star | *NCOR* |
|---|---|---|---|---|
| *M*=2500, *N*=15,000 | 310.14 | 491.00 | 399.63 | **188.95** |
| *M*=2500, *N*=20,000 | 330.75 | 563.25 | 442.66 | **205.45** |
| *M*=2500, *N*=25,000 | 357.74 | 570.02 | 465.52 | **233.14** |
| *M*=3000, *N*=15,000 | 452.22 | 671.68 | 494.60 | **286.35** |
| *M*=3000, *N*=20,000 | 467.65 | 726.75 | 498.66 | **290.31** |
| *M*=3000, *N*=25,000 | 469.86 | 791.26 | 537.56 | **303.85** |
| *M*=3718, *N*=15,000 | 613.27 | 883.71 | 753.87 | **372.66** |
| *M*=3718, *N*=20,000 | 630.39 | 904.20 | 774.31 | **390.38** |
| *M*=3718, *N*=25,000 | 646.98 | 932.38 | 882.43 | **401.88** |

with various read/write ratios (0.90, 0.80, and 0.70) 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 18%, resulted in 4 times more replicas for all the algorithms.

Next, we observe the effects of increase in the read and write frequencies. Since these two parameters are complementary to each other, we describe them together. To observe the system utilization with varying read/write frequencies, we kept the number of servers and objects 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 server 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 plot in Figure 3 shows the results of read/write ratio against the OTC savings. A clear classification can be made between the algorithms. *NCOR*, Aε-Star and Greedy incorporate the increase in the number of reads by replicating more objects and thus savings increased up to 88%, while GRA gained the least of the OTC savings of up to 42%. To understand why there is such a gap in the performance between the algorithms, we should recall that GRA specifically depends on the initial selection of gene population (for details see [15]). 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, *NCOR*, Aε-Star and Greedy never tend to deviate from their global view of the problem.

Lastly, we compare the termination time of the algorithms. Various problem instances were recorded with *C*=20%, 45% and *R/W*=0.45, 0.85. The entries in Tables 1(a) and 1(b) made bold represent the fastest time recorded over the problem instance. *NCOR* terminated faster than all the other techniques, followed by Greedy, Aε-Star and GRA.

## 5. Related work

In the context of data replication, game theoretical techniques have not received much attention. We are aware of only five published articles which directly or indirectly deal with the data replication problem using game theoretical techniques. The first work [3] is mainly on caching and uses an empirical model to

derive Nash equilibrium. The second set of works [9], [10], and [11] focuses on mechanism design issues and derives various incentive compatible auctions for replicating data on the Web. The last work [12] deals with identifying Nash strategies derived from synthetic utility functions. Our work differs from all the game theoretical techniques in: 1) identifying a non-cooperative non-priced based replica allocation method to tackle the data replication problem, 2) using game theoretical techniques to study an environment where the agents behave in a self-interested manner, and 3) deriving pure Nash equilibrium and pure strategies for the agents. Readers are encouraged to see [14] for a survey on the non-game theoretical techniques used for the data replication problem.

## 6. Concluding remarks

A detailed discussion revealed that in a realistic system, agents have no incentive to cooperate and to resolve the problem as a community; rather they act in a self-interested fashion and optimize their own benefits. To this end, we proposed a non-cooperative replica allocation game (*NCOR*), in which agents competed to host the replicas of different objects in a self-interested manner. We showed that *NCOR* exhibited a pure Nash equilibrium, and the system as a whole resided in a social optimal domain.

## References


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

[2] H. Chang, R. Govindan, S. Jamin and S. Shenker, "Towards Capturing Representative AS-Level Internet Topologies," *Computer Networks Journal*, vol. 44, no. 6, pp 737-755, 2004.

[3] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. Papadimitriou and J. Kubiatowicz, "Selfish Caching in Distributed Systems: A Game-Theoretic Analysis," in *Proc. of 23rd ACM PoDC*, 2004, pp. 21-30.

[4] A. Fabrikant, C. Papadimitriou and K. Talwar, "The Complexity of Pure Nash Equilibria," in *Proc. of 36th ACM SToC*, 2004, pp. 604-612.

[5] S. Hakimi, "Optimum Location of Switching Centers and the Absolute Centers and Medians of a Graph," *Operations Research*, vol. 12, pp. 450–459, 1964.

[6] 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, pp. 295-304.

[7] M. Karlsson and M. Mahalingam, "Do We Need Replica Placement Algorithms in Content Delivery Networks?" in *Proc. of WCCD Workshop*, 2002, pp. 117-128.

[8] S. Khan and I. Ahmad, "Heuristic-based Replication Schemas for Fast Information Retrieval over the Internet," in *Proc. of 17th ISCA PDCS*, 2004, pp. 278-283.

[9] S. Khan and I. Ahmad, "A Powerful Direct Mechanism for Optimal WWW Content Replication," in *Proc. of 19th IEEE IPDPS*, 2005.

[10] S.U. Khan and I. Ahmad, "A Game Theoretical Extended Vickery Auction Mechanism for Replicating Data in Large-scale Distributed Computing Systems," *in Proc. of PDPTA*, 2005, pp. 904-910.

[11] S.U. Khan and I. Ahmad, "RAMM: A Game Theoretical Replica Allocation and Management Mechanism," in *Proc. of 8th I-SPAN*, 2005, pp. 160-165.

[12] N. Laoutaris, O. Telelis, V. Zissimopoulos and I. Stavrakakis, "Local Utility Aware Content Replication," in *Proc. of IFIP Networking Conference*, 2005, pp. 455-468.

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

[15] 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.

[16] M. Osborne and A. Rubinstein, *A Course in Game Theory*, MIT Press, 1994.

[17] L. Pautet and S. Tardieu, "GLADE: A Framework for Building Large Object-Oriented Real-Time Distributed Systems," in *3rd International Symposium on Object-Oriented Real-Time Distributed Systems*, 2000, pp. 244-251.

[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] E. van Damme, *Stability and Perfection of Nash Equilibia*, Springer-Verlag, 1996.

[20] E. Zegura, K. Calvert and M. Donahoo, "A Quantitative Comparison of Graph-based Models for Internet Topology," *IEEE/ACM Trans. On Networking*, 5(6), pp. 770-783, 1997.