Robust CDN Replica Placement Techniques

Samee Ullah Khan
Department of Electrical and Computer Engineering
North Dakota State University
Fargo, ND 58108
samee.khan@ndsu.edu

Anthony A. Maciejewski[1] and Howard Jay Siegel[1,2]
[1]Department of Electrical and Computer Engineering
[2]Department of Computer Science
Colorado State University
Fort Collins, CO 80523
{aam, hj}@colostate.edu

***Abstract*—Creating replicas of frequently accessed data objects across a read-intensive Content Delivery Network (CDN) can result in reduced user response time. Because CDNs often operate under volatile conditions, it is of the utmost importance to study replica placement techniques that can cope with uncertainties in the system parameters. We propose four CDN replica placement heuristics that guarantee a robust performance under the uncertainty of arbitrary CDN server failures. By robust performance we mean the solution quality that a heuristic guarantees given the uncertainties in system parameters. The simulation results reveal interesting characteristics of the studied heuristics. We report these characteristics with a detailed discussion on which heuristics to utilize for robust CDN data replication given a specific scenario.**

***Index Terms*—robustness; data replication; content delivery networks; resource allocation**

I. Introduction

More efficient content delivery over the Web has become an important element of improving Web performance. Content Delivery Networks (CDNs) have been proposed to improve accessibility and reliability through content replication [23]. With CDNs, content is replicated to servers located close to clients, resulting in fast and reliable applications and Web services for the clients. Because of these qualities, CDNs are becoming an increasingly popular solution for data sharing [3].

A number of static and dynamic CDN replica placement techniques have been proposed over the last decade (e.g., [16], [17]). However, the techniques operate on the assumption that the CDN infrastructure is fault-free [3]. The performance that is delivered by a (static or dynamic) CDN replica placement technique may degrade due to the circumstances that change unpredictably, such as server failures; therefore, robust replica placement techniques must be developed that can guarantee a desired level of system performance despite certain fluctuations in a set of specified parameters.

A resource allocation is defined to be robust against perturbations in specified system parameters if degradation in the performance feature is limited when the perturbations occur within a certain range [1], [2], [18]–[21]. For instance, if a replica placement has been declared to be robust with respect to satisfying a maximum allowable CDN traffic requirement against unpredictable CDN server failures, then the CDN should continue to operate without violating the maximum allowable traffic requirement when CDN servers fail. A natural question at this point would be: what is the degree of robustness associated with a given replica placement? That is, for the example given above, how many server failures can occur before the CDN traffic requirement is violated.

The major goal of this paper is to propose static robust replica placement techniques for CDNs to support collaborative decision making applications in unpredictable environments. For this purpose, we propose four different heuristics, each with different characteristics. The performance of the studied heuristics were recorded and analyzed using a simulator that mimicked real-world CDN infrastructure.

The remainder of this paper is organized as follows. In Section II, we give a brief description of the related works and discuss how our approach is different from previous work in this field. We describe the system model and derive a mathematical model for robust replica placements in Section III. Section IV details the four robust replica placement techniques, followed by experimental evaluations in Section V. Finally, in Section VI, we summarize this paper.

II. Related Work

Replica placement literature is full of procedures for "reliable" replica placement techniques, e.g. [6], [22]. For a comprehensive survey on reliable replica placement techniques, see [3] and [12]. However, we do not find a concrete definition of robustness or a performance metric that can quantitatively measure robustness for a replica scheme. This is primarily because the published research has followed the paradigm that the system either ensures reliability or it does not [10].

This research is perhaps closest in philosophy to the study in [22] that attempts to provide some level of performance guarantee for replica placements. The guarantee is made in terms of the responsiveness of the system, i.e., how fast the clients can access the requested information. Ref. [22] studies the problem of placing replicas of data objects in CDNs to meet the responsiveness requirements of clients with the objective of minimizing the replication cost. The replication cost is measured in terms of storage, consistency management, or a combination of both. Our proposed work differs from [22] in the following ways.

1) We deliver a performance guarantee in the presence of CDN server failures, whereas the work reported in [22]

978-1-4244-3750-4/09/$25.00 ©2009 IEEE

considers a fault-free system.

2) In [22] the authors rely on complete knowledge about the system parameters to deliver a performance guarantee. Our work delivers a performance guarantee in the presence of inaccuracies in system parameters (e.g., uncertainty in server failure rate).

All of the above mentioned techniques do not provide any performance guarantee except that clients will still be able to access data when components fail in the system. In this research, we replace the traditional notions of reliable replica placement techniques with a quantification of what it really means for a system to be robust. Performance guarantees are important because when replicas are lost, clients experience increased latency. In some environments, e.g., homeland security, where timely information is extremely important, delays may lead to critical security breaches.

III. System Model and Problem Description

A. CDN Replica Placement

In this section, we derive a mathematical formulation for the CDN replica placement problem. Our problem formulation is one way to represent the optimization problem. There are many other possible ways to represent the same problem. Consider a generic CDN infrastructure consisting of N geographically distributed servers (see Figure 1). The servers include both surrogate servers and ISPs. Surrogate servers store the replicated data and ISPs do not. The N servers of the system are interconnected through a communication network. Let there be M different data objects, named $\{O_1, O_2, \cdots, O_M\}$, that have subscribed to the CDN provider's hosting service. Let S^i and $|S^i|$ be the name and the total available storage capacity for replication (in bytes) of server i, respectively, for $1 \leq i \leq N$, where the storage capacity for an ISP is assumed to be null, i.e., $S^i = 0$. The size of data object O_k, denoted by $|O_k|$, is also measured in bytes. The communication time $C(S^i, S^j)$ between two servers S^i and S^j is the cumulative communication time of the shortest path between the two servers S^i and S^j per byte basis. We assume that the values of $C(S^i, S^j)$ are known *a priori*. We also assume that $C(S^i, S^j) = C(S^j, S^i)$. These assumptions are often undertaken in formulating a replica placement problem (e.g., [22]). In certain situations, the assumption that $C(S^i, S^j) = C(S^j, S^i)$ may be relaxed when the upstream and downstream communication times vary for a path. However, current commercial CDNs do not allow varying upstream and downstream communication times for a path [3].

The replication policy assumes the existence of one primary copy for each data object. Let P_k be the server that holds the primary copy of O_k, hence, referred to as the *primary server* of O_k. P_k is not a part of the CDN network and thus, the primary copy cannot be lost due to a CDN server failure. The notion of the primary server is analogous to the subscriber's personal server in a real-world commercial CDN. In the real-world commercial CDNs, the clients are not allowed to access the primary server via the CDN because of possible security breaches [11]. Hence, we do not allow clients to access the primary server by considering it to be outside of the CDN.

Each primary server P_k contains information about the entire *replication scheme* of O_k, i.e., the location of all of the replicas of O_k. This can be done by maintaining a list of the servers where the k-th data object is replicated. The primary server holds the information about the entire replica scheme because eventually the primary server may update the data object [17]. However, the typical update frequency of a data object in a commercial CDN has been observed to be very low. Therefore, the network traffic generated due to updates is negligible compared to the network traffic generated due to reading data objects. Because of this, none of the replica placement techniques in the literature consider optimizing the network traffic due to updates [22].

Every ISP stores a multi-field record for each data object O_k. The first field is the nearest server (in terms of communication times) SN_k^i of ISP S^i, that holds a replica of O_k. The remainder of the fields store a list of the subsequent nearest servers that hold replicas of O_k, ordered by the communication times.

Finally, we assume that whenever a client issues an `http` request for one of the M data objects, the Domain Name Server (DNS) resolver at the client's ISP will reply with the IP address of the nearest, in terms of communication time, server (step (2) in Figure 1). We will call this a *first-hop CDN server*. The first-hop CDN server will act essentially as a proxy server, and if the requested data object is not replicated at the first-hop CDN server, it will redirect the request to the appropriate server (i.e., the corresponding SN_k^i). The `http` reply will be sent back to the first-hop CDN server that in turn will forward it to the client via ISP.

Let r_k^i be the number of requests for O_k, initiated from an ISP S^i during a certain given time period. Because the CDN servers cannot initiate requests for any data objects, they can only service requests, r_k^i for any data object from any CDN server will be equal to zero. Our objective is to minimize the total network traffic within the CDN, due to data object transfers to satisfy access requests.

To realize our objective, we must judicially store replicated data on the available CDN servers. Because a CDN is composed of N servers and M data objects, the entire replication scheme can be represented by an $N \times M$ matrix, named X. An element $X_{ik} \in X$ will be equal to 1 if a replica of O_k is stored at S^i, and 0 otherwise. Without the storage constraint, the most obvious solution to minimize the total network traffic would be to replicate every data object on every CDN server. However, each CDN server has a bounded storage capacity. Hence, the total size of the stored replicated data on a CDN server must not exceed the CDN server's bounded storage capacity. This constraint can be represented as

$$\sum_{k=1}^{M} (X_{ik} \cdot |O_k|) \leq |S^i|, \quad 1 \leq i \leq N. \tag{1}$$

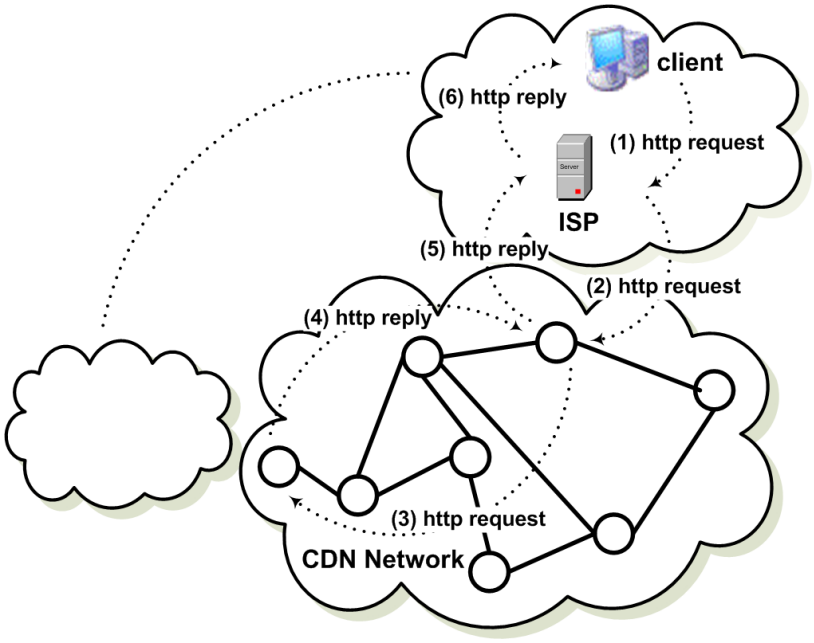


Fig. 1. A generic CDN architecture. When a data object is requested by a client, the following procedure is invoked. Step (1): client sends an `http` request to the ISP. Step (2): the ISP forwards the `http` request to the nearest (in terms of communication times) CDN server called the first-hop CDN server. Step (3): the first-hop CDN server resolves the DNS entry and identifies the nearest server that holds the replica of the requested data object. The `http` request is forwarded to that server. Step (4): the data object is sent to the first-hop CDN server. Step (5): the first-hop CDN server forwards the data object to the ISP. Step (6): the data object is sent to the client from the ISP.

Let R_k^i denote the network traffic due to ISP S^i's requests for data object O_k, addressed to the nearest server in terms of communication time, which is given by

$$R_k^i = (1 - X_{ik}) \cdot r_k^i \cdot |O_k| \cdot \min_j \{C(S^i, S^j) | X_{jk} = 1\}. \quad (2)$$

Given our objective is to minimize the total network traffic, (2) must be accumulated over all of the CDN servers and data objects. The cumulative network traffic within the CDN, denoted by D, can be represented as

$$D = \sum_{i=1}^{N} \sum_{k=1}^{M} R_k^i. \quad (3)$$

This performance metric is often used in the literature (e.g., [5], [17]). We can formally state the CDN replica placement problem as: "Find the assignment of 0 or 1 values in the X matrix that minimizes (3) subject to the storage constraint in (1)."

Because of the close resemblance of the CDN replica placement problem with the well-known NP-complete knapsack problem [14], we suspect that the CDN replica placement also must be classified in NP. Therefore, heuristics must be deigned that can efficiently and effectively minimize the overall network traffic by selectively storing replicated data on CDN servers. The minimization of the overall network traffic will reduce the average end-user perceived access time, i.e., the smaller the value of D the better experience for the end-users. However, when anomalies, such as server failures occur, the value of D, in general, will increase because: (a) when servers storing replicated data fail, requests must traverse further to retrieve data, and (b) when communication paths are lost as a consequence of server failures, request must traverse through potentially slower paths. Therefore, one must design heuristics that can cope with such anomalies by proactively provisioning resources such that when servers fail, the total network traffic is within an acceptable range. This is the topic of the subsequent section.

B. Robust CDN Replica Placement

In this paper, our goal is to develop robust replica placement techniques for systems that are fraught with uncertainties of when and how many CDN servers fail. To derive a deterministic robustness metric for the static CDN replica placement problem formulation, we make use of the four-step FePIA (**Fe**atures, **P**erturbations, **I**mpact, **A**nalysis) procedure [2].

Step 1: Describe the performance feature and requirements on that feature for the system to be robust.
Let D_O be the amount of network traffic when the CDN does not have any replicas and the CDN servers do not fail during the life-span of the CDN. Let D_f denote the amount of network traffic when the CDN does have replicated data and the CDN servers can fail during the life-span of the CDN. Because our target system is a CDN that may experience arbitrary server failures, with each (or a combination of) server failure(s) the total network traffic may increase. To quantify the impact of server failures on the total CDN network traffic,

we consider D_f to be the performance feature. With each server failure D_f is bound to increase. However, we must restrict the increase in D_f to an acceptable bound. In CDNs, this bound may be represented by D_O. Therefore, a CDN replica placement technique is considered to be robust if it can guarantee that the total cumulative network traffic within a given CDN exhibits the following

$$D_f \leq D_O. \tag{4}$$

Step 2: Identify the system and environmental perturbation parameters (a) that are uncertain, and (b) whose impact on the system performance features selected is to be evaluated in the given robustness analysis.
The perturbation parameter that we select is the uncertainty of which and how many servers fail within a CDN. We evaluate the impact on the total CDN network traffic due to server failures. (We assume that once a CDN server fails, it does not recover. The failure is permanent.)

Step 3: Identify the impact of perturbation parameters on the system performance features.
Server failures will increase the network traffic within a CDN.

Step 4: Analysis to determine the smallest collective variation in the values of perturbation parameters that will cause any of the performance features to violate its acceptable variation.
We must determine the largest number of arbitrary server failures that a given CDN can tolerate before the robustness constraint given in step **1** (Equation 4) is violated.

IV. Robust Replica Placement Techniques

In our simulations, we have used four heuristics described below. The basic idea is to have a heuristic place at least K replicas of each object in the CDN. This criteria is enforced to make a replica scheme tolerate $K-1$ arbitrary replica (or CDN server) failures, in the worst-case scenario.

1. Random-Popularity (RP). Randomized algorithms may exhibit a "good" average-case performance with fast execution times [14]. Based on these characteristics, we proceed with the development of a randomized algorithm for the CDN replica placement problem.

RP assigns the most popular data objects to CDN servers randomly subject to the storage constraints. We pick the most popular data object $O_k, \text{where } k = \text{argmax}_k(\sum_{i=1}^{N} r_k^i, \ \forall \ 1 \leq k \leq M)$ and then pick one CDN server with uniform probability, and we store the data object in that CDN server. If the CDN server is the closest possible to an ISP that is accessing the replicated data from the CDN server, then the popularity of the data objects is adjusted by subtracting the number of read requests generated by the ISP's request for the data object. Because by definition, if a replica is placed at the closest possible CDN server, then the cost of retrieval will be the minimal. Hence, no further improvement in solution quality is possible. The popularity of the data object O_k is not adjusted if a replica is not placed at the closest possible CDN server to the ISP that is requesting O_k. If the CDN server is the closest possible to multiple ISPs that are accessing the replicated data from the CDN server, then the popularity of the data object is adjusted by subtracting the number of read requests generated by all of the ISPs' request for the data object. The RP heuristic also maintains the number of replicas of each data object and ensures that at least K replicas of each data object are present within the CDN. This can be done by maintaining a counter for each data object that stores the number of replicas in the system. Once the counter reaches K that particular data object is not chosen for further replication unless all data objects have K number of replicas within the CDN.

2. Tree Based Bottom Up (TBBU). The Internet topology can be considered as a tree, the root being the top most CDN server and the leaves being the ISPs [9]. If there are a multiple number of roots, then a virtual root with zero storage capacity can be added to complete the topology. Because our simulations are based on a hierarchical Internet topological model, a simple depth-first traversal can be used to convert the graph topology into a tree topology. The general topology of the tree is such that other than the leaf nodes all other nodes are potential CDN servers. Intuitively, if replicated data is stored within a close vicinity (a few levels above the leaf) of the ISPs, then the total network traffic within the CDN should be minimized. However, when CDN servers fail, the total network traffic may drastically increase because an ISP must traverse higher up the tree to retrieve replicated data.

TBBU performs a bottom-up traversal of the tree. Because the leaf nodes are the ISPs, the TBBU heuristic must move one level up (using breath-first traversal) the tree to identify potential CDN servers. The first CDN server that the breath-first traversal explores, replicates data objects that are requested by the leaf nodes of the subtree of which the current CDN server is the parent. The CDN server replicates as many data objects as its storage permits. If the current node (CDN server) is the closest CDN server to any of the ISPs, then the popularities of the data objects are adjusted similar to the RP heuristic. After all of the current level nodes (CDN servers) have replicated the appropriate data objects, the breath-first traversal moves one-level up the tree and replicates data objects as done previously. This process is repeated until the root of the tree is reached. Similar to the RP heuristic, it is ensured that in the TBBU heuristic (a) no data object is further replicated until all of the data objects have at least K replicas within the CDN by maintaining appropriate counters and (b) at least K copies of all of the data objects are always replicated within the CDN by the repeated runs of the TBBU heuristic, if necessary.

3. Greedy Median (GM). When it is known that anomalies, such as server failure may exist within a CDN, one must

develop heuristics that consider the relative communication delay between the replicas of a data object. The reason is that when replicas are lost, requests must be serviced from replicas in close vicinity of the lost replica. If a heuristic optimizes the relative communication delay between the replicated data, then it may exhibit a smaller increase in the total network traffic when CDN servers fail.

The CDN replica placement problem is closely related to the K-median problem [14], which must place K facilities on the nodes of a graph such that the sum of the distances from each node to its nearest facility is minimized. Based on this close resemblance, the GM heuristic sorts the data objects based on their popularities. The most popular data object, say O_k, is placed at a CDN server that would incur in the minimum cumulative communication delay for O_k when accessed by all of the ISPs. If the located CDN server is the closest possible to an ISP that is requesting O_k, then the popularity of O_k is adjusted similar to the RP heuristic. In each iteration, the GM heuristic places only one data object. Note that any two consecutive iterations of the GM heuristic may result in the replication of the same data object. If a CDN server cannot accommodate a replica due to storage constraint, then the next best CDN server is chosen. The GM heuristic also keeps a count of the total number of replicas of each data object. If a data object already has K replicas within the CDN it is not replicated further until all of the data objects have at least K replicas within the CDN.

4. Greedy Median Robust (GMR). The optimization of the relative communication delay between replicas of a data object can be a very effective method to reduce the total network traffic. However, storage capacity constraints may not warrant an optimized placement of replicas. Therefore, one must also consider the worst-case impact of a server failure on the increase in the total network traffic. The basic idea behind the GMR heuristic is to apply the placement of the GM heuristic to potential worst-case server failures. This guarantees that we always improve the impact of the worst-case server failures.

First, the data objects are sorted in the decreasing order of popularities. Then the data object that is the most popular is replicated at each of the CDN servers where there is enough storage space. The GMR heuristic, then identifies a replica that if dropped from the current replica scheme will result in a minimal increase in network traffic. The GMR heuristic, drops this replica and the process continues until only K replicas of the data object remain within the CDN. Then, the GMR heuristic picks the next most popular data object and repeats the process until all of the data objects have at least K replicas within the CDN. After placing K replicas for each data object, the GMR heuristic further replicates those data objects that can potentially have an adverse affect on the network traffic due to server failures. The replica (say R) that increases the most network traffic if lost due to a server failure is replicated within the vicinity of R, i.e., the heuristic places a replica of R on the closest CDN server from the server replicating R that has enough storage capacity. This process of placing additional replicas continues until the storage capacity of the CDN servers is exhausted.

Different variants of these heuristics are possible and any improvements could be used directly to enhance the performance of our proposed heuristics. However, due to the large number of possible combinations of improvements, we will defer these extensions for future work.

V. Results and Discussion

Here, we present the results of our simulations. Before proceeding to the discussion of results, we describe our simulation setup.

The network topologies we used in the simulations were generated using the Inet topology generator toolkit. The toolkit is publicly available on the Internet (http://topology.eecs.umich.edu/inet/). The toolkit produces random network topologies following the characteristics of the Internet topology reported in [13]. (The Inet toolkit has been successfully used in recently published data replication research to simulate large-scale Internet topologies, e.g., [15], [17].) Each simulated network had a total of 20,000 nodes. Out of the 20,000 nodes, we selected at random: (a) 400 to be the ISPs and (b) 100 to be the primary locations of the 30,000 data objects. The Inet designated 1500 to the CDN server nodes and the rest of the 18,000 nodes to be routing nodes.

Based on a literature survey, we allow the distribution of object sizes to follow a heavy-tailed characterization [10] that consists of a body and a tail. The body is a lognormal distribution with mean and standard deviation equal to 8.5 and 1.318, respectively [5], [15]. The tail of the object size distribution is a Pareto distribution that has a large variation (i.e., heavy-tailed) with *Pareto index* and the *necessarily positive minimum possible value of the distribution* equal to 1.1 and 133Kb, respectively [15], [24]. By using this setting, more than 93 percent of objects fall into the body distribution and the mean size of objects is about 11Kb. Statistics have shown that the mean object size within the Internet is 10–12Kb [17].

The read frequency to data object O_k follows the Zipf-like distribution. During the simulations, the rank parameter was set to 0.75 because that was the average number obtained from the popular web sites studied in [8] and subsequently in [16], [17]. All of these parameters were used by the SURGE Internet traffic generator toolkit to produce a list of data objects with unique identification, sizes, and access frequencies. The SURGE toolkit is publicly available on the Internet (http://cs-www.bu.edu/faculty/crovella/surge_1.00a.tar.gz).

Let TS denote the total size of the data objects. The capacity of a CDN server was generated using a uniform distribution from $(\frac{1}{2}TS)C$ and $(\frac{3}{2}TS)C$, where $0 \leq C \leq 1$ is a parameter that reflects the storage capacities of the CDN servers. For example, when $TS = 100$Mb and $C = 0.30$ the capacities of the CDN servers are uniformly distributed between $(\frac{1}{2}\times 100\times 0.30 =)$ 15Mb and $(\frac{3}{2}\times 100\times 0.30 =)$ 45Mb. For our simulations, the CDN servers' capacities were varied by setting the parameter C as $0.30, 0.35, 0.40, 0.45$, and 0.50 [16], [17]. To have enough confidence in our simulation results, the

Inet topologies, the locations of the ISPs, the locations of the primary servers, and the CDN server storage capacities were altered five times each per simulated network—a total of 625 runs per simulated network. The minimum number of required replicas, i.e. K, for each data object was set to 20.

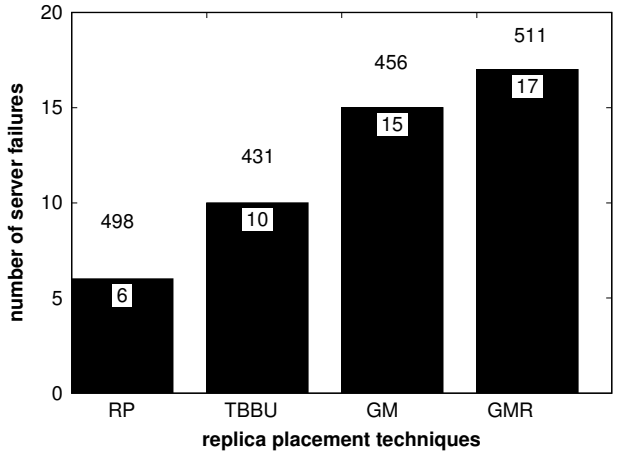
Each of the plots shown in the subsequent text, reports the performance of the heuristics that is an average over the 625 individual runs. (The average performance is a widely used measurement in data replication research [5].) Because the locations of the original servers also vary during each topological variance, the value for D_O also varies. In all of the results, the value for D_O is to be interpreted as the average over all of the simulations.

The robustness criteria (as described in Section III-B) advocates that we must find the largest number of arbitrary CDN server failures before the constraint given in Equation 4 is violated. To achieve that we must exhaustively search for all possible combinations of server failures before we can ensure an accurate measure of the worst-case performance. However, it takes exponential time to exhaustively check for all possible combinations. The results reported here only show an exhaustive combinatorial search for up to five CDN servers failures. (On average for a single run of identifying exhaustively the top five CDN server combination failures took the simulation package fifteen minutes.) For the higher number of server failure combinations, we took the result of the top five CDN server combination failures and extended it for each additional CDN server failure by selecting from the remaining CDN servers the one that produced the highest impact on network traffic.

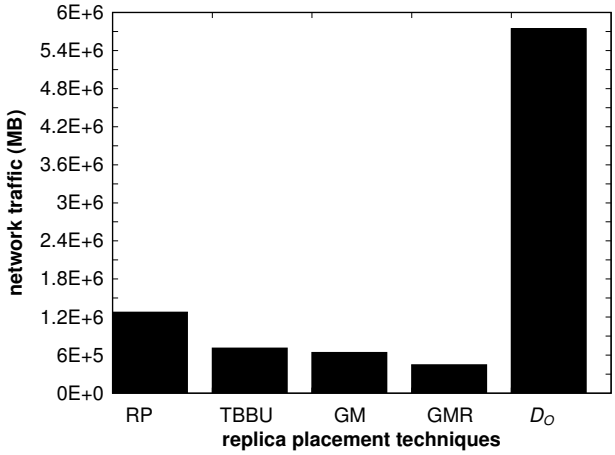
To give an overview of the results, Figure 2(a) shows the performance of all of the heuristics averaged over all of the simulated runs. We can see that the GMR heuristic tolerates the most server failures (seventeen to be exact) before it violates the robustness criteria. This robust performance can be attributed to the fact that the heuristic circumvents the worst-case impact on network traffic of a failing CDN server by judicially replicating more data objects within a close vicinity. Within the plot, above each bar, one also can see the average number of replicas placed (per data object) by each of the compared heuristics. The average number of replicas placed has a profound impact on the total network traffic within the CDN. The more the replicas, in general, the lesser would be the total network traffic. However, the placement of replicas also considerably affects the total network traffic. This phenomenon is evident in Figure 2 (b) that shows the solution quality of the heuristics when the CDN servers do not fail during the life-span of the system. Observe that the solution quality of the RP heuristic is the lowest among all of the simulated heuristics. (Higher the network traffic, lower is the solution quality of a heuristic.) This is due to the fact that RP heuristic randomly replicates data objects, even though the RP heuristic places the second largest number of replicas within the system. The TBBU heuristic places the least number of replicas within the CDN. However, greedily replicating data objects (as is the case in the TBBU heuristic) also may not be such a good idea because storage quickly depletes before other popular data objects are replicated.

As suggested by several researchers (e.g., [5], [16], [17], [24]), we also must consider measuring the increase in network traffic with the increase in the number of CDN server failures to get an idea about the performance of a heuristic. In the next set of simulations, we study the effect of variance in the locations of the CDN server storage capacities on the solution quality of the heuristics. First, we restrict the simulated network to have 50% of the entire available storage capacity at the CDN servers that are at most four-hops away from the ISPs. The rest of the storage capacity is allocated at random to the remaining CDN servers. The reason for limiting the range to a four-hop neighborhood is that most of the current commercial CDNs resolve requests to data objects within a four-hop neighboorhood of the client requesting the data objects [4]. This suggests that most of the data objects are replicated on CDN servers that are within four-hop distance from the ISPs and that is only possible when there is enough provision for storage on those CDN servers. The results for this simulated network are reported in Figure 2(c). Note that the D_O line represents the robustness criteria (as described in Equation 4), i.e., how many server failures are allowed for each of the replica placement technique (e.g., the TBBU heuristic allows twelve server failures before the robustness criteria is violated and the RP heuristic allows seven server failures). The TBBU heuristic clearly outperforms the RP and GM heuristics. This is because the TBBU heuristic by construction prefers to replicate data objects as close as possible to the ISPs. Thereby, reducing the impact on the network traffic when replicas are lost due to CDN servers failures. Another interesting result is that the GM heuristic is less robust to this simulated network compared to its average performance as shown in Figure 2(a). However, from Figure 2(c) we can observe that the GM and GMR heuristics gradually increase the network traffic compared to the RP heuristic that drastically increase the network traffic as the number of failed CDN servers increase within the CDN. Figure 2(c) also clearly illustrates that the TBBU heuristic (due to its construction) is the least sensitive to the CDN server failures given the simulated network when half of the storage is located around the ISPs.

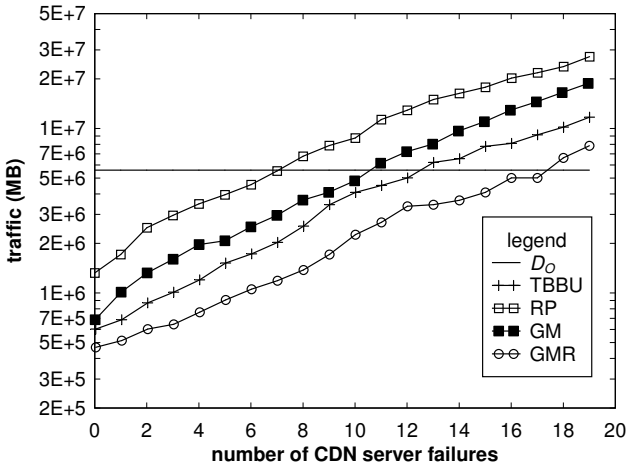
We now compare the performance of the heuristics when we restrict the simulated network to have 50% of the entire available storage capacity at the CDN servers that are at most four-hops away from the primary servers. Figure 2(d) shows the results for this simulated network. Because majority of the storage is now located at the CDN servers that are further away from the ISPs, the TBBU heuristic is the most sensitive to such a change in the simulated network. Compared to the previous result, the TBBU heuristic is robust against only ten CDN server failures. The GM and the GMR heurisitcs are again moderately sensitive to the simulated network. Figure 2(d) clearly indicates that the TBBU heuristic is the most sensitive to this simulated network. The heuristic has the largest gradiant compared to other heuristics. All heuristics maintain their relative rankings compared to the previous set



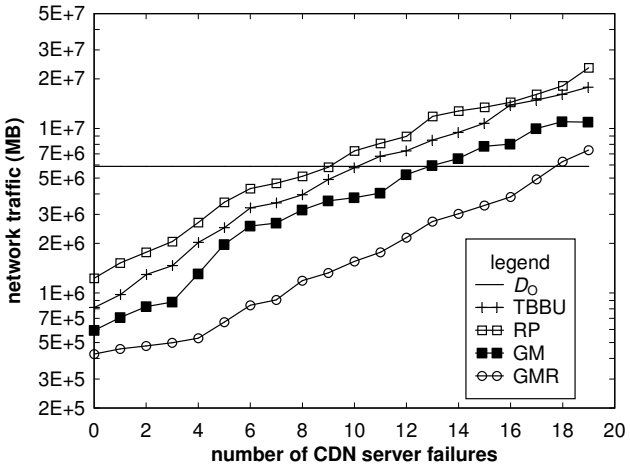
(a) Robustness of the replica placement heuristics and the number of replicas per data object (average over all simulation runs).



(b) The network traffic under the replica schema identified by the replica placement heuristics when the CDN servers do not fail within the CDN.



(c) Increase in network traffic as the number of server failures increase within the CDN when 50% of the entire storage capacity of the CDN is within the vacinity of the ISPs.



(d) Increase in network traffic as the number of server failures increase within the CDN when 50% of the entire storage capacity of the CDN is within the vacinity of the primary servers.

Fig. 2. Simulation results.

of simulations.

One also may suspect that altering the locations of the primary servers within a CDN will impact the performance of the heuristics. However, we found that such a variance only alters the baseline performance, i.e., it only affects the value of D_O. Moreover, we also found that this change in the baseline performance was somewhat arbitrary—mainly because the locations of the primary servers are not accessible by the ISPs. Hence, altering the the locations of the primary servers only moves the baseline bar up and down (see either Figure 2(c) or 2(d) to visualize the vertical movement of the baseline performance bar) and it has no correlation with the sensitivity of a heuristic. Furthermore, we also observed that such a change does not drastically change the relative rankings of the heuristics, i.e. if we ranked the heuristics according to the robustness reported in Figure 2(a), then the results obtained with the altered locations of the primary servers would be ranked similarly.

The proposed heuristics have different characteristics and all use the information available to them in a different fashion. Overall the RP heuristic is the least sensitive to the changes in the CDN server capacities, while the TBBU heuristic is the most sensitive. The TBBU heuristic may be a good option when the underlying CDN has most of its storage located within a close vicinity of the ISPs. The GM and GMR heuristics gave a very balanced performance for all of the topological variances of the simulated network. In essence, the GMR heuristic is the most robust compared to the rest of the heuristics.

VI. Concluding Remarks

In this paper, we proposed four robust replica placement heuristics for CDNs, where CDN servers could fail arbitrarily. Such techniques are meaningful to develop when a certain performance guarantee is demanded by the end users. There are many ways in which this current work can be extended. One of them is to relax the assumption that accesses to data objects are independent of each other. In reality, a collection of data objects may be accessed simultaneously or they can have interdependent access patterns—as is the case when accessing a web page. Therefore, it may be effective to replicate interdependent data objects on the same server or a close by server. Investigating such scenarios will require modeling real-world data sets. That would be considerably different than existing approaches that have always assumed independent data object accesses.

Acknowledgments

The authors would like to thank Luis Diego Briceño and Jay Smith for their valuable comments. This research was supported by the NSF under grant CNS-0615170, by the Colorado State University George T. Abell Endowment, and by the North Dakota State University Research, Creative Activities & Technology Transfer grant.

References

[1] S. Ali, A. A. Maciejewski, and H. J. Siegel, "Perspectives on robust resource allocation for heterogeneous parallel systems," in *Handbook of Parallel Computing: Models, Algorithms, and Applications*, S. Rajasekaran and J. Reif, Eds., Chapman & Hall/CRC Press, Boca Raton, FL, 2008, pp. 41-1–41-30.

[2] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, July 2004, pp. 630-641.

[3] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, Dec. 2004, pp. 335-371.

[4] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "Enhancing the web's infrastructure: From caching to replication," *IEEE Internet Computing*, vol. 1, no. 2, pp. 18–27, Mar./Apr. 1997.

[5] S. Bakiras and T. Loukopoulos, "Combining replica placement and caching techniques in content distribution networks," *Computer Communications*, vol. 28, no. 9, Sep. 2005, pp. 1062–1073.

[6] D. Barbara and H. Garcia-Molina, "Mutual exclusion in partitioned distributed systems," *Distributed Computing*, vol. 3, no. 1, Jan. 1986, pp. 119–132.

[7] P. Barford and M. Crovella, "Generating representative Web workloads for network and server performance evaluation," *International Conference on Measurement and Modeling of Computer Systems*, June 1998, pp. 151–160.

[8] L. Breslau, P. Cao, L. Fan, G. Philips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," *IEEE INFOCOM*, Apr. 1999, pp. 126-134.

[9] J. Bolot and P. Hoschka, "Performance engineering of the world wide web: Application to dimensioning and cache design," *International World Wide Web Conference*, May 1996, pp. 23–35.

[10] S. Ceri and G. Pelagatti, *Distributed Databases Principles and Systems,* New York, NY, McGraw-Hill, 1984.

[11] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Computing*, vol. 13, no. 7, Sep./Oct. 2002, pp. 50–58.

[12] L. W. Dowdy and D. V. Foster, "Comparative models of the file assignment problem," *ACM Computing Surveys*, vol. 14, no. 2, June 1982, pp. 38-56.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," *ACM SIGCOMM*, Aug. 1999, pp. 251–262.

[14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman & Co., New York, NY, USA, 1990.

[15] X. Jia, D. Li, H. Du, and J. Cao, "On optimal replication of data object at hierarchical and transparent web proxies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 8, Aug. 2005, pp. 673–685.

[16] S. U. Khan and I. Ahmad, "Discriminatory algorithmic mechanism design based WWW content replication," *Informatica*, vol. 31, no. 1, Jan. 2007, pp. 105–119.

[17] S. U. Khan and I. Ahmad, "Comparison and analysis of ten static heuristics–based Internet data replication techniques," *Journal of Parallel and Distributed Computing*, vol. 68, no. 2, Feb. 2008, pp. 113–136.

[18] A. M. Mehta, J. Smith, H. J. Siegel, A. A. Maciejewski, A. Jayaseelan, and B. Ye, "Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness," *Journal of Supercomputing*, Oct. 2007, pp. 33-58.

[19] V. Shestak, J. Smith, A. A. Maciejewski, and H. J. Siegel, "Stochastic robustness metric and its use for static resource allocations," *Journal of Parallel and Distributed Computing*, vol. 68, no. 8, Aug. 2008, pp. 1157–1173.

[20] J. Smith, H. J. Siegel, and A. A. Maciejewski, "Robust resource allocation in heterogeneous parallel and distributed computing systems," in *Wiley Encyclopedia of Computing*, B. Wah, Ed., John Wiley & Sons, New York, NY, vol. 4, 2009, pp. 2461–2470.

[21] P. Sugavanam, H. J. Siegel, A. A. Maciejewski, M. Oltikar, A. Mehta, R. Pichel, A. Horiuchi, V. Shestak, M. Al-Otaibi, Y. Krishnamurthy, S. Ali, J. Zhang, M. Aydin, P. Lee, K. Guru, M. Raskey, and A. Pippin, "Robust static allocation of resources for independent tasks under makespan and dollar cost constraints," *Journal of Parallel and Distributed Computing*, vol. 67, no. 4, Apr. 2007, pp. 400–416.

[22] M. X. Tang and M. J. Xu, "QoS-aware replica placement for content distribution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 10, Oct. 2005, pp. 921–932.

[23] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Computer Magazine*, vol. 7, no. 6, Nov./Dec. 2003, pp. 68-74.

[24] M. Zhong, K. Shen, and J. Seiferas, "Replication degree customization for high availability," *SIGOPS Operating System Review*, vol. 42, no. 4, June 2008, pp. 55–68.