Dynamic Replica Placement in Content Delivery Network

Abstract
The Content Delivery Networks (CDN) paradigm is based on the idea to move third-party content closer to the users transparently. More specifically, content is replicated on servers closer to the users, and users requests are redirected to the best replica in a transparent way, so that the user perceives better content access service. In this paper we address the problem of *dynamic* replica placement and user requests redirection jointly. Our approach accounts for users demand variability and server constraints, and minimizes the costs paid by a CDN provider without degrading the quality of the user perceived access service. A non-linear integer programming formulation is given for the replica placement and user request redirection problems. The output of the model is the periodic determination of the new location of replicas based on traffic estimates and the current replica locations. The actual solution is obtained by mapping the non-linear integer problem into a series of mixed integer linear problems obtained by linearizing the non-linear constraints of the original problem. Preliminary numerical results show that the proposed solution is capable of effectively limiting the percentage of unsatisfied requests without over-replicating the contents over the CDN servers.
Keywords
Dynamic Replica Placement, Content Distribution Networks

AUTHORS:
Novella Bartolini
Dipartimento di Informatica
Università degli Studi di Roma “La Sapienza”
Via Salaria, 113
00198 Roma, ITALY
Tel. +39 06 4991 8357, Fax: +39 06 8541 842
E-mail: novella@dsi.uniroma1.it

Francesco Lo Presti
Dipartimento di Informatica
Università dell’Aquila
Via Vetoio (Coppito 1)
67010 Coppito, AQ, ITALY
Tel. +39 0862 433165, Fax: +39 0862 433180
E-mail: lopresti@di.univaq.it

Chiara Petrioli
Dipartimento di Informatica
Università degli Studi di Roma “La Sapienza”
Via Salaria, 113
00198 Roma, ITALY
Tel. +39 06 4991 8536, Fax: +39 06 8541 842
E-mail: petrioli@di.uniroma1.it

Dynamic Replica Placement and User Request Redirection in Content Delivery Networks

Novella Bartolini,* Francesco Lo Presti† and Chiara Petrioli*
* Dipartimento di Informatica
Università di Roma "La Sapienza"
E-mail: {novella,petrioli}@di.uniroma1.it
† Dipartimento di Informatica
Università dell'Aquila
E-mail: lopresti@di.univaq.it

***Abstract*— The Content Delivery Networks (CDN) paradigm is based on the idea to move third-party content closer to the users transparently. More specifically, content is replicated on servers closer to the users, and users requests are redirected to the best replica in a transparent way, so that the user perceives better content access service. In this paper we address the problem of *dynamic* replica placement and user requests redirection jointly. Our approach accounts for users demand variability and server constraints, and minimizes the costs paid by a CDN provider without degrading the quality of the user perceived access service. A non-linear integer programming formulation is given for the replica placement and user request redirection problems. The output of the model is the periodic determination of the new location of replicas based on traffic estimates and the current replica locations. The actual solution is obtained by mapping the non-linear integer problem into a series of mixed integer linear problems obtained by linearizing the non-linear constraints of the original problem. Preliminary numerical results show that the proposed solution is capable of effectively limiting the percentage of unsatisfied requests without over-replicating the contents over the CDN servers.**

I. Introduction

Content Delivery Networks (CDNs) are one of the answers to the challenges posed by the remarkable commercial success of the Internet in the very recent years. Replicating third-party content on servers closer to the final users, and redirecting transparently their requests to the "best replica" (e.g., the closest replica in terms of distance, latency, etc.) CDN providers are able to offer improved content access service.

Solutions for CDN require addressing a number of technical problems, which include the selection of the kind of content that should be hosted (if any) at a given CDN server (replica placement), what is the best replica for a given customer, and which mechanisms should be used to redirect the user to such replica.

This paper concerns the joint optimization of replica placement and user request redirection. Our solution is dynamic in the sense that replicas are added and removed from CDN servers according to the dynamically changing user request traffic. The redirection mechanism is also changed accordingly.

Previous solutions have mostly addressed these problems separately. Of the solutions proposed for replica placement, most concern the static case. Basically, given the topology of the network, the set of CDN servers as well as the request traffic pattern, replicas are placed so that some objective function is optimized while meeting constraints on the system resources (server storage, server sustainable load, etc.). Typical problem formulations aim at either maximizing the user perceived quality given an upper bound on the number of replicas, or minimizing the cost of the CDN infrastructure while meeting constraints on the user perceived quality (e.g., latency) [1].

For the static case, simple efficient greedy solutions have been proposed in [2], [3] and [4]. In [2] Qiu et al. formulate the static replica placement problem as a minimum K median problem, in which K replicas have to be selected so that the sum of the distances between the users and their best replica is minimized. A simple greedy heuristic is shown to have performance within 50% of the optimal strategy. An evaluation of such greedy scheme to assess the impact of K on the user satisfaction (closeness to the replica), for different traffic patterns, is presented in [5]. Qui et al. have also proposed "hot spot," a solution for placing replicas on nodes that along with their neighbors generate the greatest load [2]. In [6] "hot zone" is presented as an evolution of the "hot spot" algorithm. The idea is to first identify network regions made of nodes whose latency to each other is low. Regions are then ranked according to the content request load that they generate and replicas are placed in the regions high in the ordering. In [3] and [4] Jamin et al. and Radoslavov et al. propose fan-out based heuristics in which replicas are placed at the nodes with the highest fan-out irrespective of the actual cost function. The rationale is that such nodes are likely to be in strategic places, closest (on average) to all other nodes, and therefore suitable for replica location. In [4] a performance evaluation based on real-world router-level topologies shows that the fan-out based heuristic has trends close to the greedy heuristic in terms of the average client latency.

A major limit of all these solutions is that they neglect to consider the natural dynamics in the user requests traffic. When things change that would make a different placement less costly or more satisfying for the users, the only possible solution is to re-apply the placement algorithm from scratch. This approach has a couple of problems. First of all, it may react slowly to the system changes, so that the new placement of the replicas is not the best one for the current user request traffic. Moreover, the replica placement happens every time from scratch, i.e., without considering where replicas are currently placed. This could possibly lead to non-negligible reconfiguration costs.

A few papers (e.g., [7] and [8]) have addressed the problem of dynamic replica placement. However, the proposed schemes are embedded in specific architectures for performing requests redirection and computing the best replicas. No framework is provided for identifying the optimal strategy, and for quantifying the solutions performance with respect to the optimum. In RaDar [7] a threshold based heuristic is proposed to replicate, migrate and delete replicas in response to system dynamics. The overall proposed solution combines dynamic replica allocation with servers load-aware redirection to the best replica to achieve low average users latency while empirically balancing the load among the CDN servers. No limits on the servers storage and on the maximum users latency are explicitly enforced. In [8] two schemes designed for the Tapestry

architecture [9] are presented. The idea is that upon a content request the neighborhood of the user access point in the overlay network is searched. If there is a server hosting a replica of the requested content within a maximum distance from the user, and such server is not overloaded, the request will be redirected to this server (or to the closest server if multiple servers meet such constraints). Otherwise a new replica is added to meet the user request. Two variants are introduced depending on the neighborhood of the overlay network which is searched for replicas, and on the scheme used to select the best location for the new replica. Although the ideas presented in the paper appear promising they are tightly coupled with the Tapestry architecture, and the approach does not explicitly account for neither the costs of reconfiguration nor for possible servers storage limits. Finally, no information is provided in [8] on the rule to remove replicas, making it hard to compare with our approach. Recently, the authors have presented a framework for dynamic replica placement in [10]. By assuming the users requests dynamics to obey to a Markovian model the problem of optimal dynamic replica placement has been described as a semi-Markov decision process accounting for the traffic, the user level of satisfaction as well as the costs paid to add, maintain or remove a replica from CDN servers. Although this model allows us to achieve optimality and provides insights to the dynamic replica placement problem, it is not scalable. Moreover, the paper concerns only replica placement, without addressing user request redirection.

In this paper we address the joint optimization of dynamic replica placement and users requests redirection to the best replica. We assume that users access the CDN networks through one among $|V_A|$ access points, requesting access to one of C possible contents. Replicas of the C contents can be stored in one or more among a number $|V_R| \geq 0$ of CDN servers. The user request satisfaction is modeled by a $0,1$ variable. A weight is associated to the routes from a user to a replica. The weight indicates the user perceived quality of accessing that replica. A user is said to be satisfied when the weight of the route to the best replica is below a given threshold d_{max}. The aim of our model is to limit the CDN infrastructure cost while guaranteeing that over a given percentage $1-\epsilon$ of the users (say, 99%) are satisfied.

Our scheme relies on the idea of periodically taking decisions on which replicas to add/remove, and on the best replica to which a given user request should be redirected. The decision on how the system should evolve is the outcome of a non-linear integer programming formulation of the problem. The inputs are the CDN topology, the current replica allocation, the estimated users requests traffic over the next period of time, the CDN servers resource constraints (available storage and maximum load), d_{max} and ϵ.

The actual solution is obtained by mapping the non-linear integer problem into a series of mixed integer linear problems obtained by linearizing the non-linear constraints of the original problem. The series of mixed integer linear problems is then solved numerically leading to a solution of the replica placement and user requests redirection problem for the next time interval. Our solution is proactive in the sense that it takes into account the user requests dynamics over the upcoming new time interval (of length T). This is obtained by using RLS (Recursive Least Square prediction) to design an adaptive filter. This allows us to estimate, based on current and past traffic, the future user requests traffic process. Despite the fact our RLS-prediction is affected by the users traffic dynamics and by the length of the interval T (the shorter T is the more accurate is the estimate), preliminary results show that the proposed solution is capable of effectively limit the percentage of unsatisfied requests. This is achieved without over-replicating the contents over the CDN servers, and hence with low cost for the CDN infrastructure. In the considered scenarios the placed replica are used, on average, at around 90% of their maximum load, showing the scheme effectiveness in placing new replicas only when needed.

The paper is organized as follows. In Section II we introduce mathematical preliminaries needed for describing our scheme. In Section III the dynamic replica placement and user requests redirection problems are described: the system constraints and costs are introduced and discussed here. Sections IV and V describe the non-linear integer programming formulation as well as the linearization techniques used to obtain solutions on how the CDN system should evolve. Section VI describes the results of a preliminary performance evaluation which assesses the effectiveness of the proposed solution. Finally conclusions end the paper in Section VII.

II. Mathematical Preliminaries

A. Autoregressive Process Prediction

An Autoregressive (AR) process x of order m is a stationary Gaussian process which takes the form

$$x(n) = \theta_0 + \theta_1 x(n-1) + \ldots + \theta_m x(n-m) + a(n) \qquad (1)$$

where $\theta = (\theta_0, \theta_1, \ldots, \theta_m)$ is a set of weights and a is a "white noise" Gaussian process with zero mean and variance σ_a^2.

Prediction for AR processes is straightforward. The "best" predictors (in least mean square error sense) of the future values $x(n+1), x(n+2), \ldots, x(n+L)$, given the past $x(n), x(n-1), \ldots$ are obtained by setting to zero the future values of the white noise a. Thus, the best predictor $\widehat{x}(n+j)$ of $x(n+j)$ can be recursively computed for $j = 1, \ldots, L$, as follows

$$\widehat{x}(n+j) = \theta_0 + \theta_1 \widehat{x}(n+j-1) + \theta_2 \widehat{x}(n+j-2) + \ldots + \theta_m \widehat{x}(n+j-m) \qquad (2)$$

where we set $\widehat{x}(k) = x(k)$ for $k \leq n$.

Denote $e(l) = \widehat{x}(n+l) - x(n+l)$ the l-ahead forecast error. $e(l)$ captures how $x(n+l)$ deviates from the predicted value $\widehat{x}(n+l)$. $e(l)$ is normally distributed with zero mean, and its variance is $\sigma_{e(l)}^2 = \sigma_a^2 \cdot (\sum_{i=0}^{l} \psi^2(k)) \leq \sigma_x^2$, where $\psi(.)$ is the impulse response of the Infinite Impulse Response filter with parameter θ[1]. The error variance increases with l and converges to the variance of the process σ_x^2 as l grows to infinity.

It is important to observe that the equations above are nothing but determining the conditional distribution of the future values of the process x given knowledge of the past values. Note that this conditional distribution has mean $\widehat{x}(n+l)$, which in general differs from $\mathsf{E}[x]$, and variance $\sigma_{e(l)}^2 \leq \sigma_x^2$. This inequality implies that past knowledge reduces our uncertainty on the future. Nevertheless, as l increases, the variance converges to σ_x^2 while the mean converges to $\mathsf{E}[x]$. Intuitively, knowledge of the past becomes less and less useful as we consider more distant future values.

In the following, to stress that the predicted values $\widehat{x}^{(n+l)}$ and the error variance $\sigma_{e(l)}^2$ are the mean and the variance of the (Gaussian) conditional distribution of the future values, we will denote them by $\mu(l)$ and $\sigma^2(l)$, respectively. We will also denote as $\tilde{x}(l) \sim \mathcal{N}(\mu(l), \sigma^2(l))$ a Gaussian random variable with mean $\mu(l)$ and variance $\sigma^2(l)$.

[1] In other words, $\psi(0), \psi(1), \ldots$ can be computed via (1) using the "impulse" input: $a(0) = 1$ and $a(k) = 0, k > 0$.

B. Recursive Least Square Process Prediction

In this paper we use Recursive Least Square (RLS) based process prediction. The idea behind the RLS-based prediction amounts to: 1) regard/model a process as a (time varying parameter) AR process; 2) use this model to predict future behavior, and 3) adopt a recursive form for the estimation of the model parameter to reduce computational complexity.

To model the user request we need to: 1) choose the model order m; (2) estimate the unknown parameter θ and the variance σ_a^2 of the white noise. The RLS approach proceeds recursively as follows. Hereafter, for simplicity, we will assume the model order m as given.

Let $\mathbf{x}(n) = (x(n-1), \ldots, x(n-m+1))$ be the most recent m observed values of the process (excluding the current value $x(n)$), and $\theta(n) = (\theta_1(n), \ldots, \theta_m(n))$ the current estimate of θ. The RLS estimation of θ is then recursively expressed as

$$\begin{aligned} \mathbf{k}(n) &= \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^T\mathbf{P}(n-1)\mathbf{x}(n)} && (3)\\ \theta(n) &= \theta(n-1)+\mathbf{k}(n)(x(n)-\theta(n-1)\mathbf{x}(n)) && (4)\\ \mathbf{P} &= \lambda^{-1}\mathbf{P}(n-1)-\lambda^{-1}\mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1) && (5) \end{aligned}$$

where λ is a forgetting factor, $\mathbf{P}(n)$ denotes the inverse of the input correlation matrix, and $\mathbf{k}(n)$ is a gain vector.

Given $\theta(n)$, the estimation of the future values of the process $x(n+1), x(n+2), \ldots$, is carried out as in (2) with θ replaced by $\theta(n)$, i.e.,

$$\begin{aligned} \widehat{x}(n+j) &= \theta_0(n)+\theta_1(n)\widehat{x}(n+j-1)+ \\ &\quad \theta_2(n)\widehat{x}(n+j-2)+\ldots+\theta_m(n)\widehat{x}(n+j-m) \end{aligned} \quad (6)$$

where we set $\widehat{x}(k) = x(k)$ for $k \le n$.

For $l = 1, 2, \ldots$, the estimate the variance $\widehat{\sigma}^2_{e(l)}(n)$ of the error is

$$\widehat{\sigma}^2_{e(l)}(n) = \sum_{k=0}^{l} \psi^2(k;n)\widehat{\sigma}^2_a(n) \quad (7)$$

where $\psi(\cdot\,;n)$ is the impulse response of the filter $\theta(n)$, and

$$\widehat{\sigma}^2_a(n) = \frac{1}{n-2m-1}\sum_{j=m+1}^{n} e^2(j)$$

is the current estimate of the white noise process variance.

As before, we observe that the predicted values and the error variances characterize the conditional distribution of the future values of the process. We remark that these conditional distributions converge to the actual conditional distribution only in the case of Gaussian processes. In all other case these are conditional distribution are only approximations.

III. Problem statement

We model the Internet network topology as a weighted undirected graph $G = \{V, E\}$. The vertex set V is the set of network nodes, while each edge in the set E represents a physical network link and is labeled with some kind of additive metric, e.g. the number of hops between the endpoints. We identify two subsets V_A and V_R of the set of network nodes V. V_A is the set of CDN access nodes where the requests generated by the users enter the core CDN network. V_R is the set of nodes in the core network where one or more content replica servers can be placed (called sites in the following). Figure 1 shows an example with a 40 nodes hierarchical transit stub network topology obtained by running the `gt-itm` topology generator [11]. The white circles represent the access nodes, the gray big circles the sites that can host replicas, and the small black circles nodes only used for sake of routing. Thin and thick links reflect the low or high bandwidth of the links.

We assume that C content providers exploit the hosting service of the CDN. Customers entering the CDN through an access node in V_A can therefore issue requests for one of C sets of contents, and replicas of some of the C contents can be placed in each site in V_R. Requests entering the CDN are measured in units of aggregate requests. No more than V_A^{MAX} units of aggregate requests can be generated by an access node (to model the limited access link bandwidth). Requests for a given content are served by a suitable replica. To model user satisfaction, we assume that user requests cannot be served by a replica at a distance above a given threshold d_{max}. We will denote by $R(i) \subseteq V_R$ the set of sites within distance d_{max} of an access node $i \in V_A$. Users requests from access node i are redirected to one of the replicas in $R(i)$. This can be accomplished by several means, *i.e.*, anycast. We assume that each replica can serve up to K units of aggregate request for that content (replica service capacity limit). No more than V_R^{MAX} replicas can be hosted at a given site (site resource limit).

We model the user requests at node $i \in V_A$ for content $c \in C$ as a discrete time stochastic process $x_{i,c}(n)$, $n = 0, 1, \ldots$. We do not make any particular assumption on these processes. User requests are redirected to available replicas. We denote by $\alpha_{ij,c}$ the fraction of requests for content c, originating from node i and redirected to node j. Clearly, $\sum_{j\in R(i)} \alpha_{ij,c} = 1$.

For a site $j \in V_R$, the aggregate demand of content c is $x_{j,c}(n) = \sum_{i\in S(j)} \alpha_{ij,c}x_{i,c}(n)$, where $S(j) = \{i \in V_A | j \in R(i)\}$ is the set of access nodes which can be served by replicas in site j. Denote by $r_{j,c}(n)$ the number of content c replicas at node j at time n. The requests can be fully served if $x_{j,c}(n) \le K r_{j,c}(n)$, i.e., if the aggregate demand does not exceed the replica capacity for that content; otherwise, users that were redirected to that replica suffer some level of service degradation.

We describe a given configuration of requests and replicas by means of a state vector $s = (x, r)$ with $x = (x_{i,c})_{i\in V_A, c\in C}$ and $r = (r_{j,c})_{j\in V_R, c\in C}$ in which the variable $x_{i,c}$ represents the number of requests for content $c \in C$ originated at node $i \in V_A$, and $r_{j,c}$ is the number of replicas of content $c \in C$ placed at site $j \in V_R$.

We assume that replica placement and redirection decisions are taken at regular intervals of time T. At each interval, a decision is made on which replicas to place (and where to place) and/or which replica to remove (and from where to remove it). We can denote such a decision by a vector $d = (d_{j,c+}, d_{j,c-})_{j\in V_R, c\in C} \in \mathcal{D}$, where

$$\begin{aligned} \mathcal{D} = \{\, & d | d_{j,cx} \in \mathbb{N}, x = +/-, d_{j,c+}d_{j,c-} = 0 \\ & d_{j,c+} - d_{j,c-} + r_{j,c} \ge 0, \\ & 0 \le \sum_{c\in C} d_{j,c+} - d_{j,c-} + r_{j,c} \le V_R^{\mathsf{MAX}}, j \in V_R, c \in C\} \end{aligned}$$

The variable $d_{j,c+}$ denotes how many replicas of content c are added while $d_{j,c-}$ how many removed. For convenience, we will often use the variables $d_{j,c} = d_{j,c+} - d_{j,c-}$ which denote the relative changes in the number of replicas. Replicas can be added up to site saturation or removed up to site depletion.

At the same time also a decision is taken on how to redirect requests in the next interval. We can denote such a decision by a vector $\alpha \in cA$, where $\alpha = (\alpha_{ij,c})_{i\in V_A, j\in R(i), c\in C}$ and

$$\mathcal{A} = \{\alpha | \sum_{j\in R(i), c\in C} \alpha_{ij,c} = 1, i \in V_A, \alpha_{ij,c} \ge 0, i \in V_A, j \in R(i), c \in C\}$$

Redirection is accomplished by splitting users' requests in each access node, for each content, according to the vector α, that is, for $i \in V_A$, $j \in V_R$ and $c \in C$, α_{ij}^c is the fraction of requests from node i for content c that are redirected to node j.

We associate a cost to each state and decision. We associate to each state a cost - paid per unit of time - which is the sum of a cost derived from the users perceived quality (users to replica distance, number of unsatisfied requests) and of the CDN infrastructure costs for hosting and maintaining replicas.

We measure users perceived quality by the sum over all users requests of the distance between the access node where the request is originated and the replica serving it, i.e., by $\sum_{i \in V_A, j \in V_R, c \in C} \alpha_{ij,c} x_{i,c} d_{ij}$ where d_{ij} is the distance between node i and j. A replica maintenance cost is used to model the costs of hosting replicas and keeping them up to date. We use a simple proportional cost model $C_{Maint} \sum_{j \in V_R, c \in C} r_{j,c} + d_{j,c}$, where C_{Maint} is a suitable constant.

Two other costs C^+ and C^- are paid by the CDN provider when dynamically adjusting the number of replicated servers, and are associated to the decision to add or remove a replica respectively.

The overall cost for each each state and decision is thus

$$\begin{aligned} g(x, r, d, \alpha) \quad = \quad & \sum_{i \in V_A, j \in V_R, c \in C} \alpha_{ij,c} x_{i,c} d_{ij} + \\ & C_{Maint} \sum_{j \in V_R, c \in C} (r_{j,c} + d_{j,c}) + \\ & \sum_{j \in V_R, c \in C} (C^+ d_{j,c+} - C^- d_{j,c-}) \end{aligned} \tag{8}$$

The minimization of the long run average costs described above enables a decision making criterion that can be used to formulate dynamic replica placement strategies. Given a state, a cost function associated to it and the costs of replicating and deleting replicas, the goal is to identify a strategy which dynamically allocates and deallocates replicas in response to users demand variations so that the overall cost is minimized while meeting the constraints on the replica service capacity and site resources.

IV. Replica Placement Algorithm

If we model the user requests to obey to a Markov process, we can formulate the optimal strategy as Markov Decision Process. This approach, though, poses serious problems in practice since determining the optimal strategy is not feasible in practice. Here we consider the simpler myopic strategy which consists in minimizing, step by step, the current cost.

We formulate the replica placement algorithm as an optimization problem. At each decision interval, we take the action $d \in \mathcal{D}$ and adopt the redirection strategy α which solves the following optimization problem:

$$\begin{array}{ll} \text{Minimize} & g(x, r, d, \alpha) \\ \text{subject to} & \end{array} \tag{9}$$

$$\begin{aligned} \mathsf{P}_u^f &\leq \epsilon \\ d &\in \mathcal{D} \\ \alpha &\in \mathcal{A} \end{aligned} \tag{10}$$

where P_u^f is (an estimate) of the probability of not being able to satisfy all requests in the future interval. Here ϵ is a small constant, e.g., 0.01, which determines the desired level of statistical guarantee to provide. We will derive an expression for the bound in the next section.

A. User Request Prediction

Our optimization revolves around the constraint $\mathsf{P}_u^f \leq \epsilon$. The goal is to keep the probability of not satisfying requests below a given threshold during the next interval. To this end, we predict the user content request dynamics. At each decision point jT, $j = 0, 1, \ldots$, for each access node $i \in V_A$ and content $c \in C$, the algorithm predicts the future requests dynamics based on past history $x_{i,c}(jT), x_{i,c}(jT-1), \ldots$. Prediction is carried out by means of the RLS-based algorithm presented in Section II-B. The prediction phase yields an estimate of the future requests dynamics. For each process $x_{i,c}$, this amounts to a set of Gaussian random variables $(\tilde{x}_{i,c}(l))_{l=1,\ldots,T}$, with $\tilde{x}_{i,c}(l) \sim \mathcal{N}(\mu_{i,c}(l), \sigma_{i,c}^2(l))$, $l = 1, \ldots, T$.

Denote by $\tilde{x}_{ij,c}(l)$ the predicted l-step ahead user requests for content c originated at node i and redirected to node j. Then, $\tilde{x}_{ij,c}(l) \sim \mathcal{N}(\alpha_{ij,c}\mu_{i,c}(l), \alpha_{ij,c}^2 \sigma_{\tilde{x}_{i,c}}^2(l))$.

Finally, denote by $\tilde{x}_{j,c}(l) = \sum_{i \in S(j)} x_{ij,c}(l)$, the aggregate predicted l-step ahead demand for content c at node j. Then, $\tilde{x}_{j,c}(l) \sim \mathcal{N}(\mu_{j,c}(l), \sigma_{j,c}^2(l))$, where $\mu_{c,j}(l) = \sum_{i \in V_A} \alpha_{ij,c}\mu_{i,c}(l)$ and $\sigma_{j,c}^2(l) = \sum_{i \in V_A} \alpha_{ij,c}^2 \sigma_{ij,c}^2(l)$.

Requests can be fully served if and only if

$$\tilde{x}_{j,c}(l) \leq K\left(r_{j,c} + d_{j,c}\right), j \in V_R, c \in C, l = 1, \ldots, T \tag{11}$$

i.e., if in the following interval of length T, for all contents the aggregate demands do not exceed the site replica capacity; otherwise, users requests suffer some level of service degradation.

Let $A_{j,c}(l)$ be the event that that (11) holds. Then

$$\mathsf{P}[\overline{A_{j,c}(l)}] = \Psi\left(\frac{(r_{j,c} + d_{j,c})K - \mu_{j,c}(l)}{\sigma_{j,c}(l)}\right) \tag{12}$$

where $\Psi(\cdot)$ is the ccdf of the Normal standard distribution.

We can express P_u^f in terms of the probabilities above by using the union bound

$$\mathsf{P}_u^f = \mathsf{P}[\cup_{j \in V_R, c \in C, l=1,\ldots,T} \overline{A_{j,c}(l)}] \leq \sum_{j \in V_R, c \in C, l=1,\ldots,T} \mathsf{P}[\overline{A_{j,c}(l)}]. \tag{13}$$

For sake of simplicity, in the following we will approximate the sum above with its dominant term. In other words, we will use the following approximation

$$\mathsf{P}_u^f \leq \sum_{j \in V_R, c \in C, l=1,\ldots,T} \mathsf{P}[\overline{A_{j,c}(l)}] \approx \max_{j \in V_R, c \in C, l=1,\ldots,T} \mathsf{P}[\overline{A_{j,c}(l)}]. \tag{14}$$

We will thus replace (10), $\mathsf{P}_u^f \leq \epsilon$, by

$$\mathsf{P}[\overline{A_{j,c}(l)}] \leq \epsilon \quad j \in V_R,\ c \in C, l = 1, \ldots, T. \tag{15}$$

which, along with (12), becomes

$$\mu_{j,c}(l) + z_\epsilon \sigma_{j,c}(l) \leq K(r_{j,c} + d_{j,c}) \quad j \in V_R,\ k \in C,\ l = 1, \ldots, T \tag{16}$$

where z_ϵ is the ϵ-percentile of the standard normal distribution.

We can now express our replica placement optimization problem, REP for short, as:

$$\begin{array}{ll} \textsc{Problem} & \mathsf{REP} \\ & \\ \text{Minimize} & g(x, r, d, \alpha) \\ \text{subject to} & \end{array} \tag{17}$$

$$
\begin{aligned}
z_\epsilon \sqrt{\sum_{i \in s(j)} \alpha_{ij,c}^2 \sigma_{i,c}^2(l)} &\leq K(r_{j,c} + d_{j,c}) - \sum_{i \in V_A} \alpha_{ij,c} \mu_{i,c} \\
& \qquad j \in V_R,\ c \in C,\ l = 1, \ldots, T \quad (18) \\
d_{j,c} + r_{j,c} &\geq 0 \quad j \in V_R,\ c \in C \\
0 \leq \sum_{c=1}^{C} d_{j,c} + r_{j,c} &\leq V_R^{\mathsf{MAX}} \quad j \in V_R \\
\sum_{j \in R(i), c \in C} \alpha_{ij,c} &= 1 \quad i \in V_A \\
\alpha_{ij,c} &\geq 0 \quad i \in V_A,\ j \in V_R,\ c \in C \\
d_{j,c+}, d_{j,c-} &\in \mathbb{N} \quad j \in V_R,\ c \in C
\end{aligned}
$$

V. A MIP Formulation

The optimization problem (17) poses serious challenges because of the presence of integer variables and nonlinear constraints. Here we propose the following approach which consists in considering a (series of) mixed integer linear problem (MIP) obtained by linearizing the non-linear constraints (18) as follows.

First of all, we eliminate the square root—which is not differentiable in zero—by taking the square of both sides of

(18). This yields

$$
\begin{aligned}
z_\epsilon^2 \sum_{i \in S(j)} \alpha_{ij,c}^2 \sigma_{i,c}^2(l) &\leq \left(K(r_{j,c} + d_{j,c}) - \sum_{i \in V_A} \alpha_{ij,c} \mu_{i,c}(l) \right)^2 \quad (19) \\
0 &\leq K(r_{j,c} + d_{j,c}) - \sum_{i \in V_A} \alpha_{ij,c} \mu_{i,c}(l) \quad (20)
\end{aligned}
$$

for $j \in V_R,\ c \in C,\ l = 1, \ldots, T$, with the second inequality expressing the non-negativity of the right hand side of (18). (19) is a concave quadratic constraint which transforms the problem to a quadratic constrained problem with concave constraints for which no general algorithm is known.

We now linearize (19). For the RHS, let $f_{j,c,l}((d,\alpha)) = (K(r_{j,c} + d_{j,c}) - \sum_{i \in S(j)} \alpha_{ij,c} \mu_{i,c}(l))^2$. The linear term of the Taylor expansion of $f_{j,c,l}(d,\alpha)$ around a point $(d',\alpha') \in \mathcal{D} \times \mathcal{A}$ is

$$
\begin{aligned}
& L_{j,c,l}((d,\alpha);(d',\alpha')) = \\
& 2\gamma_{j,c,l} \left(K(r_{j,c} + d_{j,c}) - \textstyle\sum_{i \in S(j)} \alpha_{ij,c} \mu_{i,c}(l) - \gamma_{j,c,l}^2 \right) \quad (21)
\end{aligned}
$$

where $\gamma_{j,c,l} = (K(r_{j,c} + d'_{j,c}) - \sum_{i \in S(j)} \alpha'_{ij,c} \mu_{i,c}(l))$.

For the LHS, we simply replace $\alpha_{ij,c}^2$ by $\alpha_{ij,c}$, obtaining $z_\epsilon^2 \sum_{i \in R(j)} \alpha_{ij,c} \sigma_{i,c}^2(l)$.

Given $(d',\alpha') \in \mathcal{D} \times \mathcal{A}$, we call the Replica Placement Mixed Integer Programming formulation ($\mathsf{REP-MIP}((d',\alpha'))$ for short) the following optimization problem obtained from REP by replacing the LHS and RHS of (19) by $L_{j,c,l}((d,\alpha);(d',\alpha'))$ and $\sum_{i \in S(j)} \alpha_{ij,c} \sigma_{i,c}^2(l)$, respectively:

$$
\begin{aligned}
&\text{PROBLEM} \quad \mathsf{REP-MIP}((d',\alpha') \\
&\text{Minimize} \quad g(x,r,d,\alpha) \\
&\text{subject to} \qquad\qquad\qquad\qquad\qquad\qquad (22) \\
&\textstyle\sum_{i \in S(j)} \left(2\gamma_{j,c,l} \mu_{i,c}(l) + z_\epsilon^2 \sigma_{i,c}^2(l) \right) \alpha_{ij,c} - 2\gamma_{j,c} K d_{j,c} \\
&\qquad \leq 2\gamma_{j,c} K r_{i,c} - \gamma_{j,c}^2 \\
&\qquad j \in V_R,\ c \in C,\ l = 1, \ldots, C \\
&d_{j,c} + r_{j,c} \geq 0, \quad j \in V_R,\ c \in C \\
&0 \leq \sum_{c=1}^{C} d_{j,c} + r_{j,c} \leq V_R^{\mathsf{MAX}} \quad j \in V_R \\
&\sum_{j \in R(i), c \in C} \alpha_{ij,c} = 1 \quad i \in V_A \\
&\alpha_{ij,c} \geq 0 \quad i \in V_A, j \in V_R, c \in C \\
&d_{j,c+}, d_{j,c-} \in \mathbb{N} \quad j \in V_R,\ c \in C
\end{aligned}
$$

It is easy to realize that $\mathsf{REP-MIP}(d',\alpha')$ has been obtained by upper bounding and lower bounding by means of suitable linear functions the left and right hand side of (19), respectively. Thus , $\mathsf{REP-MIP}(d',\alpha')$ has a smaller feasible region then REP; hence, if $\mathsf{REP-MIP}(d',\alpha')$ is feasible its optimal solution is also a solution - not necessarily an optimal one - for REP.

We formalize this in the following lemma which ensures that the optimal solution of $\mathsf{REP-MIP}(d',\alpha')$ is also a solution - not necessarily the optimal one - of REP.

Lemma 1: Let (d,α) be a feasible solution of $\mathsf{REP-MIP}((d',\alpha'))$. Then, (d,α) is also a solution of REP.

Proof: We have only to show that (19), i.e, $z_\epsilon^2 \sum_{i \in S(j)} \alpha_{ij,c}^2 \sigma_{i,c}^2(l) \leq f_{j,c,l}(d,\alpha)$, holds for (d,α). To this end, observe that: 1) $0 \leq \alpha_{ij,c} \leq 1$ implies that $z_\epsilon^2 \sum_{i \in V_A} \alpha_{ij,c}^2 \sigma_{i,c}^2(l) \leq z_\epsilon^2 \sum_{i \in V_A} \alpha_{ij,c} \sigma_{i,c}^2(l)$; 2) $f_{j,c,l}(d,\alpha)$ is a convex function of (d,α) (it is a composition of a convex increasing function - the square - with a linear (affine) function). Convexity implies that $f_{j,c,l}((d,\alpha)) \geq L_{j,c,l}((d,\alpha);(d',\alpha'))$, $(d',\alpha') \in \mathcal{D} \times \mathcal{A}$. Thus,

$$
\begin{aligned}
z_\epsilon^2 \sum_{i \in S(j)} \alpha_{ij,c}^2 \sigma_{i,c}^2(l) &\leq z_\epsilon^2 \sum_{i \in S(j)} \alpha_{ij,c} \sigma_{i,c}^2(l) \quad (24) \\
&\leq L_{j,c,l}((d,\alpha);(d',\alpha')) \quad (25) \\
&\leq f_{j,c,l}(d,\alpha) \quad (26)
\end{aligned}
$$

■

A question that arises is how close to optimal the quality of the solution of $\mathsf{REP-MIP}$ is. It is easy to realize that it is the choice of the linearization point (d',α') which mostly affects the quality of the bounds and hence of the solution. Intuitively, the closer the optimizer (d,α) of $\mathsf{REP-MIP}(d',\alpha')$ (d',α') the better, because $L_{j,c,l}((d,\alpha);(d',\alpha'))$ gets close to $f_{i,j,l}(d.\alpha)$ in the neighbor of (d',α'). Hence, better results are expected when the linearization point (d',α') is close to the optimizer itself. This suggests the following iterative approach to solve for the replica placement, whereby we consider a sequence of $\mathsf{REP-MIP}$ where the optimizer of the current problem is used as the linearization point of the next.

1) *Initialization:* Choose an initial value for (d,α), $(d^{(0)},\alpha^{(0)})$.
2) $\mathsf{REP-MIP}$ *solution.* Given $(d^{(\ell-1)},\alpha^{(\ell-1)})$, solve the problem $\mathsf{REP-MIP}(d^{(\ell-1)}\alpha^{(\ell-1)})$. Denote by $(d^{(\ell)},\alpha^{(\ell)})$ the optimal solution.
3) *Iteration:* Iterate step 2 until

$$|g(x,r,d^{(\ell)},\alpha^{(\ell)}) - g(x,r,d^{(\ell-1)},\alpha^{(\ell-1)})| \leq \delta$$

where δ is a small constant.

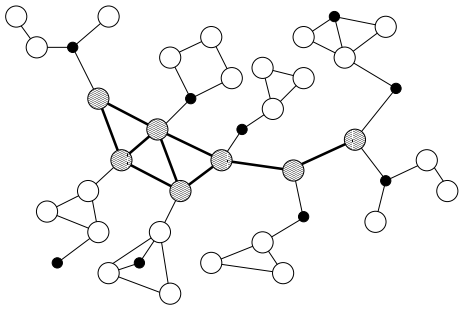

Fig. 1. NETWORK TOPOLOGY USED IN THE EXAMPLE.

The following result ensures that each new iteration yields a better solution and that the iterations terminate.

Theorem 1: Consider a pair $(d^{(0)}, \alpha^{(0)}) \in \mathcal{D} \times \mathcal{A}$ and assume that $\mathsf{REP-MIP}(\mathsf{d}^{(0)}, \alpha^{(0)})$ has a feasible solution. The following holds:

(i) the problems $\mathsf{REP-MIP}(\mathsf{d}^{(\ell)}, \alpha^{(\ell)})$, $\ell = 1, 2, \ldots$, where $(d^{(\ell)}, \alpha^{(\ell)})$ is the maximizer of $\mathsf{REP-MIP}(\mathsf{d}^{(\ell-1)}, \alpha^{(\ell-1)})$ have a feasible solution;

(ii) The sequence $g(x, r, d^{(\ell)}, \alpha^{(\ell)})$ is nonincreasing and converges to a value g^*;

Proof: The proofs are by induction. (i) We show that $(d^{(\ell)}, \alpha^{(\ell)})$ satisfies the constraints of $\mathsf{REP-MIP}(d^{(\ell)}, \alpha^{(\ell)})$. It suffices to show that (24) holds, i.e., that $z_\epsilon^2 \sum_{i \in V_A} \alpha_{ij,c}^{(\ell)} \sigma_{i,c}^2(l) \leq L_{j,c,l}((d^{(\ell)}, \alpha^{(\ell)}); (d^{(\ell)}, \alpha^{(\ell)}))$. Since $(d^{(\ell)}, \alpha^{(\ell)})$ is a feasible solution of $\mathsf{REP-MIP}(d^{(\ell-1)}, \alpha^{(\ell-1)})$ and $f_{j,c,l}$ is convex we have

$$z_\epsilon^2 \sum_{i \in V_A} \alpha_{ij,c}^{\ell} \sigma_{i,c}^2(l) \leq L_{j,c,l}((d^{(\ell)}, \alpha^{(\ell)}); (d^{(\ell-1)}, \alpha^{(\ell-1)})) \quad (27)$$

$$\leq f_{j,c,l}(d^{(\ell)}, \alpha^{(\ell)}) \quad (28)$$

$$= L_{j,c,l}((d^{(\ell)}, \alpha^{(\ell)}); (d^{(\ell)}, \alpha^{(\ell-1)})). \quad (29)$$

Hence (24) holds. (ii) We have just shown that $(d^{(\ell)}, \alpha^{(\ell)})$ is a solution - not necessarily the optimal one - of $\mathsf{REP-MIP}(d^{(\ell)}, \alpha^{(\ell)})$. The optimizer $(d^{(\ell+1)}, \alpha^{(\ell+1)})$ of $\mathsf{REP-MIP}(d^{(\ell)}, \alpha^{(\ell)})$ then satisfies $g(x, r, d^{(\ell+1)}, \alpha^{(\ell+1)}) \leq g(d^{(\ell)}, \alpha^{(\ell)})$. Hence, the sequence $g(x, r, d^{(\ell)}, \alpha^{(\ell)})$, $\ell = 1, 2, \ldots$ is non-increasing. Convergence to a value g^* is then ensured by the fact that $\mathsf{REP-MIP}$ has no unbounded solution. ■

We observe that the above results, while providing an algorithm to improve the solution, do not say anything on how close this solution is to the optimizer of REP. Our numerical results, nevertheless, suggest that our solution might be close to the actual optimizer in most cases, yet we not have a proof for that. This will be subject of further research.

VI. NUMERICAL RESULTS

In this section, we provide numerical examples to illustrate the dynamic replica placement algorithm behavior. Due to the limited space here we concentrate on the simple topology in Figure 1 with 24 access nodes and 7 service nodes (sites). The thin lines denote slower links (with a weight of 2), the thick ones faster links (with a weight of 1). We assume users issue requests for two types of content. The aggregate requests at site $i \in V_A$ for content c are modeled as independent Markovian birth-death process with birth and death rate equal to 0.005. We set $K = 1$, $V_A^{\mathsf{MAX}} = 10$, $V_R^{\mathsf{MAX}} = 30$, $d_{\mathsf{max}} = \infty$. For the replica placement, we set $\epsilon = 0.05$, $C_{maint} = 10$, $C^+ = C^- = 0$ (thus in this example we ignore the cost of adding and removing replicas). The value of parameter T was varied in the different set of experiments. Finally, for the RLS prediction we use $m = 3$ and a forgetting factor $\lambda = 0.99$.

TABLE I
SIMULATIONS RESULTS.

T	% of time with not served requests	% not served	av. utilization
20	3.64 ± 3.27	0.55 ± 1.15	0.89
50	4.9 ± 4.7	1.12 ± 1.84	0.88
100	7.8 ± 6.27	1.76 ± 2.81	0.88

We summarize the results, in table I, where we report: (1) the fraction of time when there were unsatisfied requests; (2) the fraction of requests that could not be served, i.e.,

$$\frac{\sum_{n=1}^{t} \sum_{j \in V_R, c \in C} (x_{j,c}(n) - r_{j,c}(n))^+}{\sum_{l=1}^{n} \sum_{j \in V_R, c \in C} x_{j,c}(n)}$$

where $(x)^+ = \max\{x, 0\}$; and (3), the average utilization, computed as

$$\frac{\sum_{n=1}^{t} \sum_{j \in V_R, c \in C} \min\{x_{j,c}(n), r_{j,c}(n))}{\sum_{l=1}^{n} \sum_{j \in V_R, c \in C} (r_{j,c}(n))};$$

Simulation results include 99% confidence interval computed over 100 independent simulations. Each simulation was $t = 10000$ time unit long (after having removed the initial transient period).

Not surprisingly, the algorithms perform better for smaller decision intervals. The causes of the degradation for the larger intervals are twofold. First of all, the performance of the RLS estimator degrades as the intervals length grows. Second and most of all, use of the approximation $\mathsf{P}_u^f \approx \max_{j \in V_R, c \in C, l=1,\ldots,T} \mathsf{P}[\overline{A_{j,c}(l)}]$ becomes more and more inaccurate as T grows. As a consequence we do observe that the fraction of time in which requests are not satisfied exceeds ϵ, especially for $T = 100$. We note that, nonetheless, the fraction of requests that cannot be served, which well captures the user perceived quality of service, attains very small values, well below the 5% value set for ϵ for all values of T. At the same time, the resource utilization is quite high and close to 90% for all values of T. In this example, the algorithm achieves good quality of service while attaining at the same time very high utilization, showing its capability of satisfying the users requests while limiting the number of replicas, thus the infrastructure costs.

To illustrate the dynamic behavior of the replica placement strategy for one simulation and $T = 20$ we plot in Figure 2, separately for each of the two contents, the number of requests redirected to (dotted line) and the number of requests that can be served by all the hosted replicas (continuous line) at each of the seven sites as function of time (x-axis). The allocation behavior is easily understood by observing that in order to maximize statistical multiplexing gain, the best strategy lies in aggregating as much requests as possible. In this scenario, where each replica can serve all nodes, the algorithm keeps, for each content, two sites filled with replicas of that content (the first and the third sites with content 1 replicas, the second and the fourth with content 2 replicas); additional replicas are allocated where space is available in the remaining sites. We observe that replica placement closely tracks user requests dynamics. This has been observed in all simulations and accounts for the high level of utilization observed.

VII. CONCLUSIONS

In this paper we have tackled with the problem of jointly optimizing the problems of dynamic replica placement and users requests

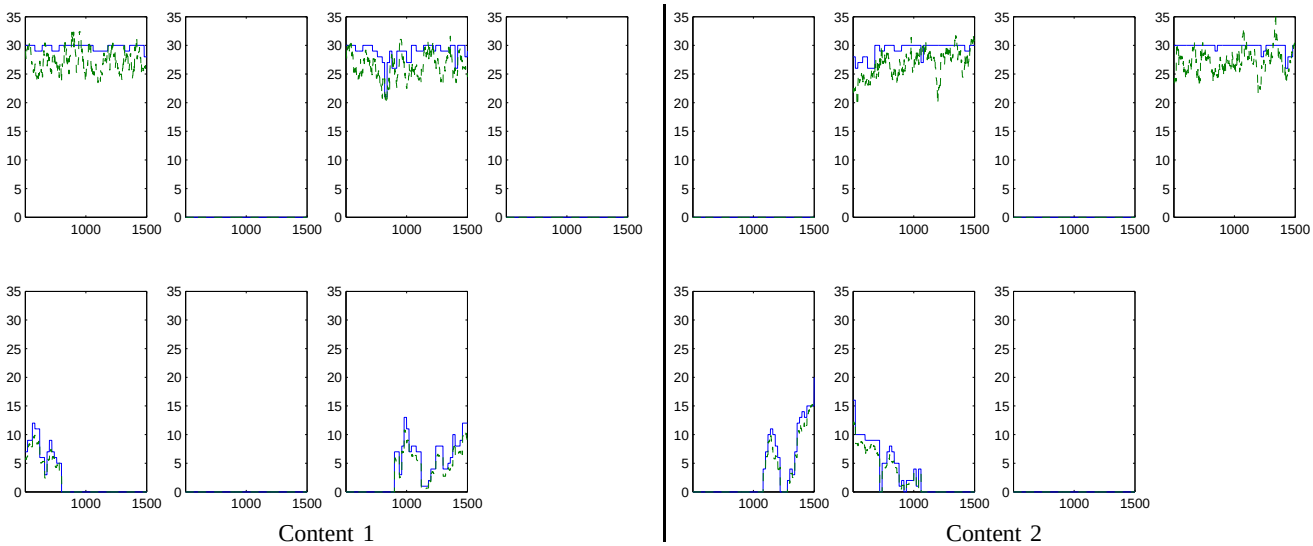

Fig. 2. SIMULATION: CONTENT 1 (LEFT PLOTS) AND CONTENT 2 (RIGHT PLOTS). NUMBER OF REQUESTS REDIRECTED TO (DOTTED LINE) AND NUMBER OF REQUESTS SERVED BY THE HOSTED REPLICAS (CONTINUOUS LINE) AT EACH OF THE SEVEN CDN SITES AS FUNCTION OF TIME.

redirection to the best replica. Differently from previous approaches, our solution does not rely on specific architectures and accounts for all the major relevant constraints of the problem. Aim of our formulation is to limit the CDN infrastructure costs (e.g., the number of replicas, the cost for their installation, maintenance and removal) while guaranteeing that the percentage of users requests which do not receive a satisfactory access service is bounded by a small value. Multiple contents and realistic constraints on the CDN servers resources (storage and maximum load) are also accounted for. Our results are twofold. First, we have modeled the problem by non-linear integer programming, and then solved it by a series of mixed integer linear programming problems obtained by linearizing the non-linear constraints of the original problem. Secondly, results are shown which assess the effectiveness of the proposed solution in limiting the percentage of unsatisfied users requests while avoiding to over-replicate contents in the CDN networks. Such results show that all the replica hosted by CDN servers are highly utilized (on average at around 90% of their maximum load), thus confirming the scheme capability of introducing new replicas only when needed.

REFERENCES

[1] M. Karlsson, C. Karamanolis, and M. Mahalingam, "A unified framework for evaluating replica placement algorithms," *Technical Report HPL-2002, Hewlett Packard Laboratories*, 2002.

[2] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *Proceedings if IEEE INFOCOM 2001*, Anchorage, AK, April 22–26 2001, pp. 1587–1596.

[3] S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," in *Proceedings if IEEE INFOCOM 2001*, Anchorage, AK, April 22–26 2001, pp. 31–40.

[4] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed internet replica placement," *Proceedings of WCW'01: Web Caching and Content Distribution Workshop*, June 20–22 2001.

[5] Y. Li and M. T. Liu, "Optimization of performance gain in content distrubution networks with server replicas," in *Proceedings of the 2003 Symposium on Applications and the Internet, SAINT 2003*, Orlando, FL, January 27–31 2003, pp. 182–189.

[6] M. Szymaniak, G. Pierre, and M. van Steen, "Latency-driven replica placement," Submitted for publication, May 2004, http://www.globule.org/publi/LDRP_draft.html.

[7] M. Rabinovich and A. Aggarwal, "RaDaR: a scalable architecture for a global Web hosting service," *Elsevier Computer Networks*, vol. 31, no. 11–16, pp. 1545–1561, 1999.

[8] Y. Chen, R. Katz, and J. Kubiatowicz, "Dynamic replica placement for scalable content delivery," in *International Workshop on Peer-to-Peer Systems, IPTPS 2002*, Cambridge, MA, March 7–8 2002.

[9] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communication*, vol. 22, no. 1, January 2004.

[10] N. Bartolini, F. Lo Presti, and C. Petrioli, "Optimal dynamic replica placement in Content Delivery Networks," in *Proceedings of the* 11*th IEEE International Conference on Networks, ICON 2003*, Sydney, Australia, September 28–October 1 2003, pp. 125–130.

[11] GT-ITM: Georgia Tech Internetwork Topology Models, http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html.