# **Resource Optimization for Content Distribution Networks in Shared Infrastructure** Environment

Thanh Vinh Nguyen, Chun Tung Chou, Paul Boustead

Telecommunications and Information Technology Research Institute (TITR) University of Wollongong, Australia Email: {vinh, ctchou, paul}@titr.uow.edu.au

*Abstract*— Current Content Distribution Networks (CDN) deployment requires heavy infrastructure investment since a large number of servers have to be deployed over a wide area. This paper proposes a new paradigm where future CDNs are to be deployed over a leased server and network infrastructure. While this paradigm shift introduces a new dimension of flexibility, it requires the resource provisioning and object placement problems to be jointly considered. This paper formulates this new optimization problem, and presents solution to it based on Lagrangian relaxation and greedy heuristics.

*Index Terms*—Content distribution networks, shared infrastructure, provisioning, replica placement, greedy heuristic.

# I. INTRODUCTION

Content Distribution Networks (CDN) are networks of surrogate servers spanning the Internet, aiming to overcome network bottlenecks and improve user experience by bringing contents to network edge. In recent years, with the appearance of commercial CDNs, operated by companies like Akamai [1], Digital Island [2] and Speedera [3], this area has been receiving a great deal of attention from research community. While recognizing that Content Distribution Service is a useful service, we believe that there exists the needs for new innovative CDN deployment approaches. Deploying a CDN currently involves building an Internet-wide network of servers to host replicated contents. As an example, Akamai currently has more than 12000 servers located in 66 countries [1]. Such scale of infrastructure is obviously challenging in financial, technical and administrative terms, both for deployment and operation of the service.

We believe that in the future, the deployment of content networks will follow one of two possible paths: 1) One is the formation of large, Internet-wide CDNs by inter-networking among regional CDN operators/carriers. This content-peering approach is being investigated by Internet Engineering Task Force content distribution internetworking working group [4]. 2) The second alternative is to deploy CDNs using hired resources, including bandwidth, server storage space and processing power. Our paper aims to explore and investigate this "resource-hiring" paradigm.

Inspired by works such as [5], [6], [7] (and references therein), we envisage a networking environment where Network Providers (NP) manage the physical server and network infrastructure, over which services can be delivered using leased server and network resources. The Service Providers (SP) lease these resources in the form of overlay network topologies, including virtual servers at server farms/server clusters and virtual links, to deliver their services.

By using a common shared infrastructure, each service is provided with the full advantage of a dedicated network at lower initial investment and resource commitment. The approach also introduces a whole new dimension of flexibility into service networks - the ability to dynamically adjust resource allocation, including changing the overlay topology, when necessary. Considering that more and more services will need distributed infrastructure for scalability and quality of service, we believe that *shared infrastructure* is the logical next step for network service delivery. Content Distribution Service, with its massive distributed infrastructure requirement, is a typical application that we believe would benefit from this paradigm shift.

However, such a fundamental shift in deploying and running CDNs will introduce new requirements and challenges, especially resource provisioning and content replication that have yet been addressed in literature. For example, the question of where to place contents and how requests are directed - ie. replica placement (RP) problem - and where and how much resources should be allocated - i.e. provisioning problem now become closely coupled and must be addressed simultaneously. Besides, any resource usage, e.g. storage space or processing power, incurs charges on the CDN provider, thus a RP algorithm that fills up all potential resources in search of the best performance possible ([8], [9]) is no longer appropriate. Rather, we would expect providers to strive to achieve an acceptable performance at the least cost possible. The introduction of the "resource costs" concept also makes the problem much more complex by itself, since the total usage must be optimized across multiple resource types, each with different costs at different locations.

In this paper, we will first describe a general framework for provisioning service overlays over shared infrastructure, which aims to define the components a overlay CDN, as well as the functions and interaction between NPs and SPs regarding service provisioning. Based on this framework, we will then develop a CDN resource provisioning model that deals with resource allocation and object placement problems jointly. The model is formulated as a mixed integer programming problem, which we have been able to solved efficiently using a greedy search heuristic.

The organization of this paper is as follows: Sections II describes a general provisioning framework for share infrastructure, Section III then introduces parameters and formulation for our CDN provisioning model. Section IV outlines the heuristics we have used to solve this model, namely Lagrangian relaxation and greedy search heuristic. Section V describes and discuss our implementation and numerical results. Section VI then provides a brief summary of other related works. Finally Section VII sums up the main contribution of this work and outlines the future directions that we will take.

# **II. FRAMEWORK**

This section aims to give an overview of the CDN provisioning process over a shared infrastructure. The aim of the provisioing process is to determine the amount of resources to be hired from the network provider. The resources required to deploy a CDN include computation power and memory space at servers, and network bandwidth. The resource requirement can be communicated to the NP by using a "CDN topology" information structure that includes the following components:

# • Virtual CDN servers:

These virtual CDN servers are logical partitions of physical servers or server clusters. Each virtual server is specified through the following attributes:

- Server location
- Computation power
- Storage space
- Delivery bandwidth

Virtual servers are required to be accessible to end users, to whom contents are to be delivered. A server's "delivery bandwidth" specifies the aggregated bandwidth used to deliver contents from that server. As content requests may come from any part of the network, the SP is only interested in and provision for the aggregated bandwidth at a particular server. This is similar to the "hose" bandwidth provisioning concept in VPN literature [7].

# Virtual links:

A virtual link is a connection between two CDN servers. These links form the backbone of the CDN, and are used for software distribution, content replication and updates, and back-end communications (e.g. accessing a vendor's back-end databases in an online transaction). A virtual link is specified by its end points and bandwidth.

A key issue for the CDN operator is to determine the best CDN topology. In the "resource-hiring" paradigm, topology decisions will have to tradeoff between service performance and resource cost. An optimal topology would deliver the required quality of service at the minimum cost. Topology design, therefore, would be an optimization process taking into account the following:

- Potential server locations
- Hiring cost of various resources and their capacity limits
- User demands and their spatial distribution
- Service performance requirement

The NP is expected to provide the SP with the first two pieces of information listed above. We envisage the optimization process to be implemented in a provisioning tool used by the SP, which gathers the necessary information, computes the optimal "CDN topology", and passes this information onto the NP. The NP will then create the requested topology provided that it passes an admission control test. An important element of the resource hiring framework is that the SP can adjust its resource usage according to the current level of demand by performing the above optimization process at appropriate time intervals. Although we have illustrated the provisioning framework using CDN as an example, it can also be applied to other services.

Based on this framework, in the next section we will formulate a joint CDN provisioning-replica placement optimization problem, reflecting our vision on how CDNs will most likely work in this new innovative environment.

#### **III. CDN PROVISIONING MODEL**

#### A. Assumptions

Since for a CDN the bandwidth consumed by content delivery is expected to be much higher than for distribution and updates, we have taken into account only the delivery bandwidth provisioning in our model, i.e. we have assumed a simplified approach where the more significant factors - CDN servers and delivery bandwidth - are determined before the less important connecting "pipes" are provisioned.

We also assumed that the server resource consumption exhibits a constant per-request average computation and bandwidth usage. In other words, computation power and bandwidth usage, and hence their cost, at a server is linearly dependent on the amount of requests handled.

# B. Network and cost models

We model the network as a graph  $G = \langle V, E \rangle$ , where V is the set of nodes and E is the set of edges. Two subsets of nodes are defined - customer nodes  $V_c$  and potential server nodes  $V_s$ , where  $V = V_c \cup V_s$ ,  $M = |V_c|$  and  $N = |V_s|$ . A customer node is where requests for contents are generated, which in real life may be an ISP access network. A potential server node represents a location that resources can be hired and a surrogate server can be established, e.g. a server farm. To model server capacities, each potential server  $i \in V_s$  has a limit  $C_i$  on the amount of requests it can handle per unit time.

Each edge  $e \in E$  has a weight that represent distances between two nodes. The distances  $d_{ij}$  between any node  $i \in V_s$  and  $j \in V_c$  is assumed to be known and is preprocessed using shortest path algorithms on graph G.

A content population of K objects is used, with object k having size  $s_k$ . The amount of requests generated by customer node j for object k per unit time is represented by  $\lambda_{jk}$ .

A decision by the CDN owner to use potential node i for content replication and delivery is assumed to incur the following costs:

- Site start up cost  $\gamma_i$  a once-off charge that models the costs involved when an new server is established (e.g. resource allocation, shipping software, configuration...)
- Storage cost assumed to be proportional to the amount of contents stored at the site, with unit cost being  $\alpha_i$ .
- Serving cost a combined cost for processing power and delivery bandwidth. The cost per unit request is  $\beta_i$ .

An "acceptable QoS threshold" is represented by a requirement of average customer-content distance for each object under T. The provisioning problem can then be formulated as an optimization problem whose objective is to find a configuration (i.e. which locations are used, which objects are replicated, and how much serving power should be used at each location) that satisfies the QoS constraint, and at the same time minimizes the total cost incurred on the CDN provider.

## C. Problem Formulation

To formulate this optimization problem we used the following binary and continuous decision variables:

$$y_i = \begin{cases} 1 & \text{if site i is used} \\ 0 & \text{otherwise} \end{cases}$$
(1)

$$x_{ik} = \begin{cases} 1 & \text{if object k is replicated at site i} \\ 0 & \text{otherwise} \end{cases}$$
(2)

$$r_{ij}^k \in [0, \lambda_{jk}] \tag{3}$$

The continuous variable  $r_{ij}^k$  is used to indicate the fraction of requests for object k from customer j that should be directed to site i. Based on these variables, the optimization can be formulated as the following mixed integer linear programming model:

$$(\mathbf{P}) \quad Min \underbrace{\sum_{i=1}^{N} \sum_{k=1}^{K} x_{ik} \alpha_{i} s_{k}}_{Storage \ cost} + \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{K} r_{ij}^{k} \beta_{i}}_{Serving \ cost} + \underbrace{\sum_{i=1}^{N} y_{i} \gamma_{i}}_{Startup \ cost}$$

$$(4)$$

Subject to:

$$\sum_{j=1}^{M} \sum_{k=1}^{K} r_{ij}^k \le C_i, \forall i$$
(5a)

$$\sum_{i=1}^{N} r_{ij}^{k} = \lambda_{jk}, \forall j, k$$
(5b)

$$\frac{\sum_{i=1}^{N} \sum_{j=1}^{M} r_{ij}^{k} d_{ij}}{\sum_{i=1}^{M} \lambda_{ik}} \le T, \forall k$$
(5c)

$$\sum_{i=1}^{N} y_i C_i \ge D_t \tag{5d}$$

$$r_{ij}^k \le \lambda_{jk} x_{ik}, \forall i, j, k \tag{5e}$$

$$x_{ik} < y_i, \forall i, k \tag{5f}$$

$$x_{ik}, y_i \in \{0, 1\}$$
(5g)

$$r_{ij}^k \in [0, \lambda_{jk}] \tag{5h}$$

where  $D_t = \sum_{j=1}^{M} \sum_{k=1}^{K} \lambda_{jk}$  is the total demand from all customers. In this formulation, constraint (5a) enforces the serving power limit at each potential site, (5b) means that all requests must be served, while constraint (5c) maintains the QoS threshold for each object. Constraint (5e) means that a site can only serve request for an object if that object is replicated there, and (5f) means a site must be in use for objects to be replicated there. According to constraint (5d), the total capacity of opened sites must not be smaller than the total demand. Note that this

is actually a redundant constraint, which can be deduced from (5a), (5e) and (5f). Although redundant in the original model, it has been added to strengthen a Lagrangian relaxation approach that we will be using in Section IV. Similar techniques have often been used in literature [10].

Also note that in our model there is an underlying assumption requests from customers can be split and directed to a suitable server. The splitting ratios are indicated by  $r_{ij}^k$ . This is necessary and useful considering that each customer node represents a customer base, such as an ISP network, not an end user.

We will show below that this mixed integer linear programming formulation is NP-hard, and thus we need to develop heuristics to solve the problem for realistic networks. In the Section IV we will describe a number of proposed heuristics.

# D. Proving NP-hardness

We will prove the NP-hardness of this model by reducing it to a well-known NP-hard problem - the facility location problem [11], [10]. Consider our model with K = 1,  $\gamma_i = 0$ ,  $\forall i$ ,  $T = \infty$  and denote  $v_{ij} = \frac{r_{ij}}{\lambda_j}$ ,  $v_{ij} \in [0, 1]$ . Note that subscript for k has been dropped since there is only one object. In this particular case, the start up cost and performance constraint (5c) are eliminated, and the problem can be rewritten as:

$$Min\sum_{i=1}^{N}\sum_{j=1}^{M}v_{ij}\lambda_{j}\beta_{i} + \sum_{i=1}^{N}x_{i}\alpha_{i}s$$

Subject to:

$$\sum_{j=1}^{M} v_{ij} \lambda_j \leq C_i, \forall i$$
$$\sum_{i=1}^{N} v_{ij} = 1, \forall j$$
$$0 \leq v_{ij} \leq x_i \leq 1, \forall i, j; x_{ik} \in \{0, 1\}$$

which is a capacitated facility location problem. Thus our model is NP-hard  $\Box$ 

#### **IV. SOLUTION PROCEDURES**

#### A. Lagrangian Heuristic

One of our first heuristic attempt is a Lagrangian relaxation is approach. We relax constraint (5e) and add the following term to the objective function:

$$\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{K} \mu_{ij}^{k}(r_{ij}^{k} - \lambda_{jk}x_{ik}), \mu_{ij}^{k} \ge 0$$

where  $\mu_{ij}^k$  is the Lagrangian multiplier. By rearranging the relaxed objective function, the problem can be decomposed into two sub-problems:

(P1) 
$$Min \sum_{i=1}^{N} \sum_{k=1}^{K} x_{ik} (\alpha_i s_k - \sum_{j=1}^{M} \lambda_{jk} \mu_{ij}^k) + \sum_{i=1}^{N} y_i \gamma_i$$
 (6)

Subject to: (5d), (5f), (5g), and:

(P2) 
$$Min \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{K} r_{ij}^{k} (\beta_{i} + \mu_{ij}^{k})$$
 (7)

Subject to: (5a), (5b), (5c), (5h)

Note that we have been able to separate binary variables into problem P1, and the continuous variable into P2. For each value of the Lagrangian multiplier vector  $\mu_{ij}^k$ , the optimal solution to the relaxed problem can be obtained by solving P1 and P2 separately.

1) Solving subproblems: To find the optimal solution to problem P1, we first leave aside constraint (5d) and further decompose the problem into one subproblem for each location i:

$$(P1_i) Min \sum_{k=1}^{K} x_{ik} \underbrace{(\alpha_i s_k - \sum_{j=1}^{M} \lambda_{jk} \mu_{ij}^k)}_{c_{ik}} + y_i \gamma_i \qquad (8)$$

Subject to: (5f).

For each of these N problems, we consider the following cases:

- 1) If  $y_i = 0$  (server *i* closed) then:
- x<sub>ik</sub> = 0, ∀k ∈ 1, 2, ..., K, due to constraint (5f).
  2) If y<sub>i</sub> = 1 (server i opened) then:
  - for each  $k \in 1, 2, \dots, K$ 
    - If  $c_{ik} \ge 0$  then  $x_{ik} = 0$
    - If  $c_{ik} < 0$  then  $x_{ik} = 1$
    - where  $c_{ik}$  is defined in (8).

• 
$$P1_i = \sum_{k=1}^{K} x_{ik} c_{ik} + y_i \gamma_i$$

Thus server *i* incurs a cost of  $P1_i$  if opened, and 0 otherwise. Note that if  $P1_i \leq 0$ , server *i* is opened in the optimal solution to problem P1, i.e.  $y_i = 1, \forall i \in V_s^- = \{i | P1_i \leq 0\}$ . If  $\sum_{i \in V_s^-} C_i \geq D_t$ , then constraint (5d) is satisfied, and  $V_s^-$  is optimal set of servers to be opened. Otherwise (5f) can be enforced by solving the following problem, which determines which additional servers should also be opened.

$$Min\sum_{i\in V_s\setminus V_s^-}y_iP1$$

Subject to:

i

$$\sum_{\in V_s \setminus V_s^-} y_i C_i \ge D_t - \sum_{i \in V_s^-} C_i, y_i \in \{0, 1\}$$

which is a binary knapsack problem of at most size N. There exist exact algorithms to solve this problem, for example dynamic programming algorithms [12], [13]. Note that although the addition of constraint (5d) has made solving this subproblem more complicated, our experiments showed a significant improvement in solution bounds with its inclusion.

Problem P2, on the other hand, is a constrained multicommodity minimum cost flow problem and is a pure linear programming problem. In our current implementation, the Cplex optimization package [14] is used to solve it. 2) Subgradient heuristic: We used the popular subgradient method to adjust Lagrangian multiplier  $\mu_{ij}^k$  and generate solution bounds. Instead of solving the original mixed integer problem, we solve the relaxed problem for a series of different multiplier values iteratively. In each iteration, solution to the relaxed problem gives a lower bound of the original problem. We then attempt to reconstruct a feasible solution using the following procedure:

- Set  $\bar{x}_{ik} = 0$  and  $\bar{y}_i = 0, \forall i, k$
- Consider the solution  $\{r_{ij}^k\}$  to subproblem P2
- For each  $r_{ij}^k$
- set  $\bar{x}_{ik}$  and  $\bar{y}_i$  to 1 if  $r_{ij}^k > 0$

The set  $(r_{ij}^k, \bar{x}_{ik}, \bar{y}_i)$  created by this procedure is a feasible solution to the original model, and thus gives us upper bound. The Lagrangian multiplier is adjusted in a way to improve the gap between these bounds, and the best feasible solution found is retained as heuristic solution. Due to space restriction, details of the subgradient method cannot be explained here. Interested readers are referred to [5].

# B. Greedy heuristics

1) Analysis: In the original optimization (4), if  $y_i$  were determined, i.e. if the topology were fixed, the objective function would become:

$$(P_T) Min \sum_{i=1}^{N} \sum_{k=1}^{K} x_{ik} \alpha_i s_k + \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{K} r_{ij}^k \beta_i + C_{sites}$$
(9)

where the total start up costs  $C_{sites}$  is fixed and can be removed from the objective. Note that the objective function now can be broken up across k (objects), and that in the constraint set, only constraint (5a) binds the objects together. Thus we can approximate  $P_T$  by ranking the objects (e.g. according to total demand or priority) and solving the following problem for each object sequentially. The resulted problem for object k is below, with subscript k dropped for convenience.

$$(\boldsymbol{P_{Tk}}) \quad Min \sum_{i=1}^{N} x_i \alpha_i s + \sum_{i=1}^{N} \sum_{j=1}^{M} r_{ij} \beta_i$$
(10)

Subject to:

$$\sum_{j=1}^{M} r_{ij} \le C_i^{residual,k}, \forall i$$
 (11a)

$$\sum_{i=1}^{N} r_{ij} = \lambda_j, \forall j \tag{11b}$$

$$\frac{\sum_{i=1}^{N} \sum_{j=1}^{M} r_{ij} d_{ij}}{\sum_{j=1}^{M} \lambda_j} \le T,$$
(11c)

$$r_{ij} \le \lambda_j x_i, \forall i, j$$
 (11d)

$$x_i \le y_i, \forall i \tag{11e}$$

$$x_i, y_i \in \{0, 1\}; r_{ij} \in [0, \lambda_j]$$
 (11f)

Note that the residual resource has to be updated after the optimization is done for each object:  $C_i^{residual,k} =$ 

 $C_i^{residual,k-1} - \sum_{j=1}^M r_{ij}^{k-1}$ .  $P_{Tk}$  is similar to a facility location problem (FLP) with an extra performance constraint. If the set  $x_i$  (i.e. object replication) were fixed, the problem is reduced to a constrained transport problem, where optimal  $r_{ij}$  is deterministic and can be solved efficiently.

2) Heuristic outline: Based on the above observations, we have designed a two-level greedy search heuristic, with the first level attempts to search for candidate topology and the second level searches for the best replication within each topology. The following outline describes our current implementation, using a Drop procedure [10] at both levels.

Topology search:

- 1) Initialize: Start with all server locations opened, calculate the cost of current topology.
- 2) For each node *i* in the current topology, find the total cost of the resulting topology if server at *i* is removed.
- 3) If there are possible cost savings, close server *i* that offers the maximum saving.
- 4) Go back to (2) and continue till no improvements can be found

The cost for each candidate topology is found using the following procedure:

- 1) Rank the objects in decreasing order of total demand.
- 2) For each object k
  - Start by placing a replica of k at every server and calculate current cost by solving a constrained transport problem.
  - For each replica, re-optimize the problem to find the cost saving if it were removed.
  - Drop the replica that offers the maximum saving.
  - Continue drop attempts till no further improvements can be found.
- 3) Update the residual resource remained at each server and proceed to next object.

Thus instead of solving the original problem, we solve a large number of constrained transport problems of size at most M.N. The number of transport problems can be shown to be  $O(N^2K)$ . In our implementation, Cplex is used to solve the transport problems. Note that although this outline uses only Drop heuristic for both topology and replication search, we are currently implementing other variations of greedy search (i.e. add and interchange [10]), which can also be fitted within the same framework.

#### V. NUMERICAL RESULTS

The proposed heuristics have been implemented and tested on a number of different network topologies generated by the GT-ITM topology generator [15]. In each topology, a number of nodes are randomly selected to be potential server nodes, and the others are assumed to be customer nodes. Other parameters, including server capacities, request rates and resource costs are randomly generated. For each topology, we experimented with different cost combinations in which resource costs are varied across a wide range.

Table I shows the the results obtained from experiments with a these generated topologies. Some problems of small sizes were

used so that we could obtain an exact (optimal) solution with the mixed integer programming solver provided by Cplex optimization package, which enabled a comparison between heuristic and optimal solutions.

In this table, each row show the results from a different sample problem. The first three columns show problem size parameters, namely N, M and K. Greedy heuristic results are given as  $C_q$ , together with the gaps between these solutions and optimal solutions (if available) or lower bound provided by the Lagrangian solution (not included in the table) – denoted  $gap_a$ - and associated cpu time  $t_g$ . Lagrangian solutions are given as  $C_l$ , together with the gap  $gap_l$  between solution and lower bound LB,  $gap_l = \frac{C_l - LB}{LB}$ , cpu time  $t_l$ , and the fractional cost contributed by each type of resources – denoted  $c_{\gamma}$ ,  $c_{\alpha}$ ,  $c_{\beta}$  for start up, storage and serving costs, respectively. Exact solutions, if obtainable, are displayed in the *Cplex* column.

Generally, the Lagrangean heuristic provided results with a  $gap_l$  of under 37% for all of our experiments. It has also been observed that this gap is small (under 10%) if serving or start up costs are large compared to storage cost, and tends get larger (over 10%) if storage cost is set to dominate the final total cost. However, at small problem sizes, where we were were able to compare this with optimal solution, it appears that even in those cases, this heuristic result is actually still very close (under 5%) to the optimal solution. This suggests that although the heuristic does not always provide a good warrantee on solution quality at large storage costs, it does give a good indication of where requests should be directed, which, in turn, translates into a good feasible solution with our solution recovery procedure. The main draw back of this Lagrangian heuristic is scalability, with cpu times exceed 20 hours as the problem size approaches a total of 30 nodes and 1000 objects. This is due to the fact that subproblem (P2) is has a large number of variables (NMK), thus although it is solvable, performance decreases quickly as problem size increases.

The greedy heuristic, on the other hand, achieved a comparable accuracy to the Lagrangian heuristic. Although most of the times if provides a worse solution, the difference is not significant as the gap between a greedy solution and the Lagrangian lower bound is seen to be consistently under 40%. The advantage of this approach is performance. For example, it can be seen from Table I cpu time  $t_q$  is under 2% of cpu time  $t_l$  for problems of size  $15 \times 15 \times 1000$ .

# VI. RELATED WORKS

There have been a significant amount of research in the field of replica placement algorithms for CDNs over the past few years. Some examples are [9], [8], [16], [17]. There are also works in other related areas that were not intended for CDNs, but can be relevant and useful in terms of formulation and algorithms, such as proxy locations [18], cable network content replication [19], or placement of network instrumentations [20]. Work in [21] provided a comprehensive review of these and other related publications.

Though different in details, these existing RP models share significant characteristics. Server capacities are usually assumed unlimited and content-customer distances are then optimized

Problem size			Greedy heuristic			Lagrangean heuristic						Cplex
Ν	Μ	K	$C_g$	$gap_g(\%)$	$t_g$	$C_l$	$gap_l(\%)$	$c_{\gamma}(\%)$	$c_{\alpha}(\%)$	$c_{\beta}(\%)$	$t_l$	
10	10	10	91781	5	2s	90308	9	16	28	56	27s	86004
10	10	10	78394	9	2s	76223	15	40	14	46	50s	72030
10	10	10	45362	51	2s	31665	12	87	7	6	68s	30010
10	10	10	545970	1	2s	548382	3	5	2	93	10s	542234
15	15	100	568313	17	6m	549558	13	3	68	29	50m	-
15	15	100	106594	5	7m	115976	15	2.	56	42	50m	-
15	15	100	222781	23	7m	215880	2	66	80	19	48m	-
15	15	100	142850	29	7m	150579	36	0.1	99	0.9	56m	-
15	15	1000	798656	11	18m	887503	23	3	80	13	13h	-
15	15	1000	470674	38	16m	462129	36	42	40	18	26h	-
15	15	1000	565316	28	16m	493909	12	12	53	38	21h	-
54	54	1000	230500	-	5h	-	-	-	-	-	-	-

TABLE I Numerical results.

without considering server loads [9], [16], [20]. Besides, the optimization objectives have often followed a "fill-it-up" trend - looking for best performance possible using all available resources. As pointed out in Section I, such approach would not be suitable in a shared infrastructure environment. Unlike these existing works, we aim to create a model suitable for this new environment that captures all the key constraints.

More recently, work in [22] used content clustering to reduce the number of objects that an RP algorithm has to consider. The authors showed that with clustering, the number of objects can be reduced significantly and comparable RP results can then be achieved at even as low as 1 - 2% of cpu time. We consider this work complementary to ours and intend to leverage it to improve our heuristic's scalability.

# VII. CONCLUSION AND FUTURE WORK

We believe that service provisioning over shared infrastructure represents a promising future direction for the Content Distribution Network service; however, such paradigm shift implies new requirements for provisioning and replica placement algorithms that have yet to be fully addressed in current literature. In this paper we have formulated an optimization model to address the CDN provisioning and replica placement problem that arises in this new environment, and have been able to solve it with a two level greedy search heuristic.

We are currently expanding this heuristic implementation to include other greedy search variants (including add and interchange procedures) and intend to investigate the use of content clustering to improve heuristic scalability.

#### VIII. ACKNOWLEDGMENT

The support of the Cooperative Research Centre for Smart Internet Technology (CRC-SIT, www.smartinternet.com.au) for this work and permission to publish this paper is hereby acknowledged.

#### REFERENCES

- [1] Akamai. www.akamai.com.
- [2] Digital Island. www.digitalisland.net.
- [3] Speedera. www.speedera.com.
- [4] IETF Content Distribution Internetworking Working Group. www.ietf.org/html.charters/cdi-charter.html.
- [5] F. Safaei, I. Ouveysi, M. Zukerman, and R. Pattie. Carrier-scale programmable networks: Wholesaler platform and resource optimization. *IEEE Journal on Selected Areas in Communications*, 19(3), March 2001.
- [6] T. Brunner and R. Stadler. Service management in multiparty active networks. *IEEE Communications Magazine*, 38(3):144–151, 2000.
- [7] Service Overlay Networks: SLAs, QoS and Bandwidth Provisioning, Paris, France, November 2002.
- [8] J. Kangasharju, J. Roberts, and K. Ross. Object replication strategies in content distribution networks. In *In Proc. WCW'01: Web Caching and Content Distribution Workshop*, Boston, MA, June 2001.
- [9] L. Qiu, V. Padmanabham, and G. Voelker. On the placement of web server replicas. In *In Proc. 20th IEEE INFOCOM*, 2001.
- [10] R. Sridharan. The capacitated plant location problem. European Journal of Operational Research, 87:203–213, 1995.
- [11] D. B. Shmoys, Éva Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract. pages 265–274, 1997.
- [12] S. Martello and P. Toth. Knapsack problems. Wiley, 1990.
- [13] G. L. Nemhauser and L. A. Wolsey. Integer and Combinatorial Optimization. Wiley, 1999.
- [14] ILOG Cplex Optimization Suite. www.ilog.com/products/cplex.
- [15] K. Calvert, M. Doar, and E. W. Zegura. Modeling internet topology. IEEE Communications Magazine, June 1997.
- [16] S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the internet. In *Proceedings of INFOCOM*, pages 31–40, 2001.
- [17] A. Venkataramani, M. Dahlin, and P. Weidmann. Bandwidth constrained placement in a wan. ACM Principles of Distributed Computing, Aug 2001.
- [18] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the internet. In *Proceedings of INFOCOM*, 1999.
- [19] I. Cidon, S. Kutten, and R. Soffer. Optimal allocation of electronic content. In *Proceedings of INFOCOM*, pages 1773–1780, 2001.
- [20] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of internet instrumentation. In *Proceedings of INFOCOM*, pages 295–304, 2000.
- [21] M. Karlsson and M. Mahalingam. Do we need replica placement algorithms in content delivery networks. In WCW'01. IEEE International Web Content Caching and Distribution Workshop, August 2002.
- [22] Yan Chen, Lili Qiu, Weiyu Chen, Luan Nguyen, and Randy H. Katz. Efficient and adaptive web replication using content clustering. *IEEE Journal on Selected Areas in Communications*, 21(Y), 2003.