Object Replication Strategies in Content Distribution Networks

Jussi Kangasharju
Institut Eurécom
Sophia Antipolis
France

James Roberts
France Télécom R & D
Issy-les-Moulineaux
France

Keith W. Ross
Institut Eurécom
Sophia Antipolis
France

Abstract

Recently the Internet has witnessed the emergence of content distribution networks (CDNs). In this paper we study the problem of optimally replicating objects in CDN servers. In our model, each Internet Autonomous System (AS) is a node with finite storage capacity for replicating objects. The optimization problem is to replicate objects so that when clients fetch objects from the nearest CDN server with the requested object, the average number of ASs traversed is minimized. We formulate this problem as a combinatorial optimization problem. We show that this optimization problem is NP complete. We develop four natural heuristics and compare them numerically using real Internet topology data. We find that the best results are obtained with heuristics that have all the CDN servers cooperating in making the replication decisions. We also develop a model for studying the benefits of cooperation between nodes, which provides insight into peer-to-peer content distribution.

Keywords: Optimal replication strategies, Content Distribution Networks, Peer-to-peer networks, Cooperation

1 Introduction

Recently the Internet has witnessed the emergence of content distribution networks (CDNs). CDNs are targeted for speeding up the delivery of normal Web content and reduce the load on the origin servers and the network. CDNs, such as Akamai [1] or Digital Island [4], distribute content by placing it on content servers which are located near the users. A content provider can sign up for the service and have its content placed on the content servers. The content is replicated either on-demand when users request it, or it can be replicated beforehand, by pushing the content on the content servers.

In this paper we study the problem of optimally replicating objects in CDN servers. We consider each AS as a node in a graph with one CDN server with finite storage capacity for replicating objects. The optimization problem is to replicate objects so that the average number of ASs traversed is minimized when clients fetch objects from the nearest CDN server containing the requested object. We formulate this problem as a combinatorial optimization problem and show that this optimization problem is NP complete. We develop four natural heuristics and compare them numerically using real Internet topology data. Our results show that the best results are obtained with heuristics that have all the CDN servers cooperating in making the replication decisions. We also develop a model for studying the benefits of cooperation between nodes in a peer-to-peer networking context.

This paper is organized as follows. Section 2 presents the network topologies we used. In Section 3 we develop our cost model. Section 4 presents the replication heuristics we have developed and Section 5 presents our evaluation methods and results. Section 6 presents peer-to-peer content distribution and develops and evaluates a model for cooperation in peer-to-peer networks. Section 7 discusses related work. Finally, Section 8 concludes the paper and presents directions for future work.

2 Network Model

Our network model is based on the actual Internet AS topology. To construct the topology, we will use data provided by NLANR [12]. This data represents summaries of Internet routing data collected in the Route Views Project [15] from 1997 to beginning of 2000. These summaries provide information about which ASs are connected to each other. From these summaries we constructed a graph where the nodes are the ASs and the edges are the inter-AS connections. We then calculated the shortest paths between all the node pairs, and used this data to form a distance matrix for the network.

As discussed in [5], this method is slightly inaccurate because we cannot infer that all the connections are valid. For example, consider a small ISP that buys

connectivity from two bigger ISPs. For each of the providers, our graph will show an edge between the small ISP and the provider, but in reality the small ISP would not route traffic from one of its providers to another. This is not likely to be a problem, though, since in most cases the two providers in this case either have a direct link between them or are able to connect via a common provider.

A CDN tries to put its content servers as close to users as possible. By placing the clients and the content servers (storage nodes in our network) in the same nodes, we can simulate the ideal redirection where all clients are always redirected to the closest server (the redirection in this case is static). By studying different replication strategies we can determine which strategies a CDN should use when deciding which objects to replicate. Because a CDN has complete knowledge of its network, we can use strategies which require global information or cooperation.

2.1 Network Topologies

We downloaded several different files from NLANR [12], each describing the AS topology on a different day. Table 1 shows the different topologies we used, the number of ASs in each one, the number of leaf ASs, the average length of the shortest path between any two nodes, and the average distance of the shortest path between any two leaf nodes.

From topologies in Table 1 and other topologies we downloaded (not shown), we see that the average distance between nodes tends to increase as the number of nodes in the network increases. It is also important to note that these topologies *do not* contain all of the ASs in the Internet, but only a subset of them. Therefore, the full Internet AS topology would likely have a higher average distance than any of the topologies we used.

3 Cost Model

We consider a network where the nodes are the autonomous systems (ASs). For simplicity, we assume that one AS is the same as one ISP. We have I ASs in the network. AS i, $i \in 1, 2, \ldots, I$, has S_i bytes of storage capacity and has clients that request objects at aggregate rate λ_i.

We have J objects. Object j has a size of b_j, $j \in \{1, 2, \ldots, J\}$ and a request probability p_j which is the probability that a client will request this object. We assume that client request patterns are homogeneous, i.e., the p_j's are the same for all ASs. But this model can be extended to include other request patterns by using p_{ij}, which would be the request probability for object j from AS i.

We have the following variables:

$$x_{ij} = \begin{cases} 1 & \text{if object } j \text{ is stored at AS } i, \\ 0 & \text{otherwise.} \end{cases}$$

The storage is constrained by the space available at AS i, that is

$$\sum_{j=1}^{J} b_j x_{ij} \leq S_i \qquad i = 1, \ldots, I$$

The goal is to choose the x_{ij}'s so that a given performance metric is minimized. In this paper our goal is to minimize the average number of inter-AS hops that a request must traverse. This reflects the download time of an object to some degree and can thus be used as an indicator of the user perceived latency.

We denote the matrix of all x_{ij}'s by $\mathbf{x}$. Furthermore, we assume that each object j is initially placed on an origin server; we denote by O_j the AS that contains this origin server. We assume that all of the objects are always available in their origin servers, regardless of the placement $\mathbf{x}$. We denote the placement of objects to origin servers as $\mathbf{x}_o$. Note that this origin server placement does not count against the storage capacity S_{O_j}.

The average number of hops that a request must traverse from AS i is

$$C_i(\mathbf{x}) = \sum_{j=1}^{J} p_j d_{ij}(\mathbf{x}) \tag{1}$$

where $d_{ij}(\mathbf{x})$ is the shortest distance to a copy of object j from AS i under the placement $\mathbf{x}$. This nearest copy is either in the origin AS O_j, or in another AS where the object has been replicated. We assume that the client is always redirected to the nearest copy. In this paper we do not consider the mechanisms used to redirect clients, but instead assume that such a mechanism is in place.

Let $\Lambda = \sum_i \lambda_i$ be the total request rate of all ASs. The average number of hops from all ASs is then

$$\begin{aligned} C(\mathbf{x}) &= \frac{1}{\Lambda} \sum_{i=1}^{I} \lambda_i C_i(\mathbf{x}) \\ &= \frac{1}{\Lambda} \sum_{i=1}^{I} \sum_{j=1}^{J} \lambda_i p_j d_{ij}(\mathbf{x}) \\ &= \sum_{i=1}^{I} \sum_{j=1}^{J} s_{ij} d_{ij}(\mathbf{x}) \end{aligned} \tag{2}$$

Date	Number of nodes	Number of leaf nodes	Average distance	Avg. leaf distance
97/12/21	3184	1423	3.76	4.34
99/01/11	549	136	3.40	4.18
99/12/08	767	222	3.03	3.38
99/12/11	1477	502	3.45	4.10

Table 1: AS topologies

where $s_{ij} = \lambda_i p_j / \Lambda$. The placement $\mathbf{x}$ is subject to

$$\sum_{j=1}^{J} b_j x_{ij} \leq S_i \qquad i = 1, \ldots, I$$

This cost function represents the long term average cost. For a large number of objects and ASs, it is not feasible to solve this problem optimally; in fact, as we show in the next section, this problem is NP-complete.

3.1 Proving NP-Completeness

In order to prove that our optimization problem is NP-complete, we first formulate the problem as a decision problem. Given a target number of hops T, we ask is there a placement $\mathbf{x}$ such that

$$\sum_{i=1}^{I} \sum_{j=1}^{J} s_{ij} d_{ij}(\mathbf{x}) \leq T$$

subject to

$$\sum_{j=1}^{J} b_j x_{ij} \leq S_i \qquad i = 1, \ldots, I$$

We prove the NP-completeness of this problem by showing that it belongs in NP and then we reduce the knapsack problem to a special case of our problem. This proves the NP-completeness.

The problem is easily seen to be in NP. Given a placement $\mathbf{x}$ and number of hops T, we can verify in polynomial time whether the placement results in an average cost of less than T hops.

Next, we consider the special case where $S_1 = S$, $S_i = 0$, $i = 2, \ldots, I$, $\lambda_i = \lambda$, $i = 1, \ldots, I$, $p_j = p$, $j = 1, \ldots, J$, i.e., we have only one AS on which to place objects, all ASs have the same request rate, and all objects are equally popular.

Recall that each object j is always available at the origin AS O_j. The cost of getting object j for a client in AS i is $d_{ij}(\mathbf{x}_o)$. Because all clients always go to the nearest copy, placing copies on the only available AS can only decrease the cost for any client, i.e., $d_{ij}(\mathbf{x}) \leq d_{ij}(\mathbf{x}_o)$ for all i and j. We define the utility of placing object j on AS i as $u(j) = \sum_i [d_{ij}(\mathbf{x}_o) - d_{ij}(\mathbf{x})]$, i.e., the decrease in number of hops we would obtain if we placed object j in the AS.

Given target decrease in number of hops T' we now ask if there is a set of objects J' such that

$$\sum_{j \in J'} b_j \leq S \quad \text{and} \quad \sum_{j \in J'} u(j) \geq T'$$

This problem is identical to the well-known NP-complete knapsack problem [6]. Given that our placement problem belongs in NP and that the knapsack problem reduces to it, we know that our placement problem is NP-complete.

4 Replication Heuristics

Because our optimization problem is NP-complete, finding the optimal solution is not feasible. Therefore we have designed several heuristics that use the available information in different ways in order to get the best results.

In our simulations we use the following heuristics.

1. **Random**. Assigns objects to storage nodes randomly subject to the storage constraints. We pick one object with uniform probability and one node with uniform probability, and we store the object in that node. If the node already stores that object, we pick a new object and a new node. As a result, an object can be assigned to several nodes, but a node will have at maximum one copy of an object.

2. **Popularity**. Each node stores the most popular objects among its clients. The node sorts the objects in decreasing order of popularity and stores as many objects in this order as the storage constraint allows. The node can estimate the popularities by observing the requests it receives form its clients. This heuristic does not require the node to get any information from outside of the node. Note that in our case, the object popularities p_j are the same across all nodes, hence all the

nodes will store the objects in the same order but subject to different storage constraints.

3. **Greedy-Single**. Each node i calculates $C_{ij} = p_j d_{ij}(\mathbf{x}_o)$ for each object j. This represents the contribution of an individual object to (1) under the initial placement. The node then sorts the objects in decreasing order of C_{ij} and stores as many objects in this order as the storage constraint allows. The popularities are obtained as in the *Popularity* heuristics, but the CDN also needs information about the network topology in order to estimate the d_{ij}'s. Note that the C_{ij}'s are calculated only once under the placement $\mathbf{x}_o$ and are not adjusted when objects are stored. This means that every node stores objects independently of all the other nodes and no cooperation between nodes is required.

4. **Greedy-Global** The CDN first calculates $C_{ij} = \lambda_i p_j d_{ij}(\mathbf{x}_o)$ for all nodes i and objects j. Then the CDN picks the node-object-pair which has the highest C_{ij} and stores that object in that node. This results in a new placement $\mathbf{x}_1$. Then the CDN re-calculates the costs C_{ij} under the new placement and pick the node-object-pair that has the highest cost. We store that object in that node and obtain a new placement $\mathbf{x}_2$. We iterate this until all the storage nodes have been filled.

We do not consider any variants of these base heuristics, such as using popularity divided by object size, but evaluate only the performance of the base heuristics. Different variants of these heuristics have been studied in the context of web caching (see [8] and references therein) and any such improvements could be used directly to enhance the performance of our heuristics.

5 Evaluation of Heuristics

We evaluated the performances of our heuristics using the topologies from Section 2.1. We ran each heuristic on each topology using different parameters for object popularity and storage capacity in the nodes.

We assigned the popularities to the objects from a Zipf-like distribution where we varied the parameter from 0.6 to 1.0. Values between 0.6 and 0.8 have been typically observed in Web proxy traffic [2]. Much higher values, up to 1.4, have also been discovered in the context of popular Web servers [13]. Object sizes were randomly drawn from a uniform distribution. The storage capacity at each node was fixed to some percentage of the total size of the set of objects. We varied this percentage in the course of the experiments. All the client nodes had the same request rate λ.

We placed all the content servers at the leafs of the network, i.e., for all non-leaf ASs we set $S_i = 0$. Even though this assignment of storage capacity is artificial, it allows us to study the performances of our heuristics better. By placing all the storage capacity and objects at the edges, the average number of hops needed to obtain an object is higher than with a more realistic assignment. This makes it more important to replicate the right objects and lets us see more clearly the differences between our heuristics.

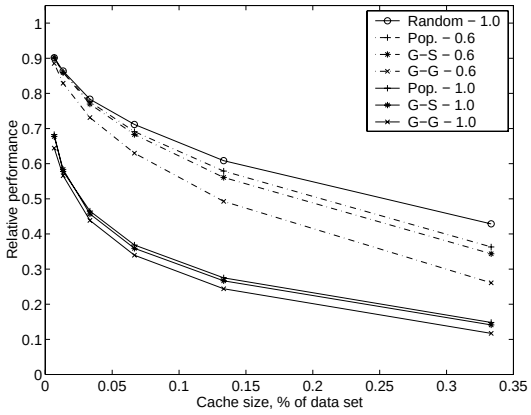
The baseline for our experiments was the initial placement $\mathbf{x}_o$ which we obtained by randomly assigning objects to storage nodes. We compared the performance of each of the heuristics to this baseline and report the relative performance obtained with each heuristic. Because the memory requirements for the experiments grow with the product IJ, we were not able to run all experiments for all the topologies.

Figure 1 shows the results from experiments with 1000 objects. We only show results for two topologies, but we observed that the results for the remaining two were similar to the two shown here. On the x-axis we plot the amount of storage at a node as percentage of the total size of the objects. On the y-axis we plot the performance relative to the baseline. In each graph, we plot different curves for different heuristics and different values of the Zipf-parameter (0.6, and 1.0). Note that we only plot the *Random* heuristic with Zipf-parameter 1.0. The performance of the *Random* heuristic was similar for the other Zipf-values. The curves of the other heuristics for Zipf-values between 0.6 and 1.0 was between the curves plotted.

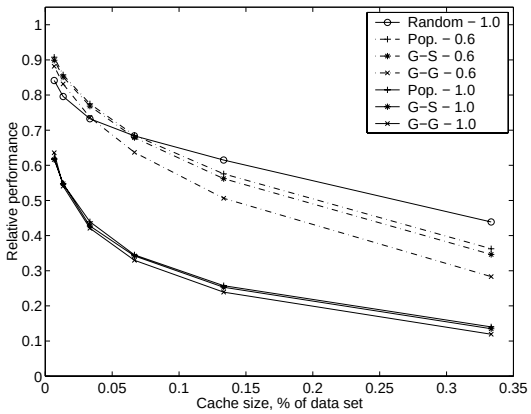
From Figure 1 we can see that *Greedy-Global* is the best performing heuristic. The second best is *Greedy-Single*, followed closely by *Popularity*. *Random* heuristic is consistently the worst and does not achieve substantial reductions in number of hops, even for large storage capacities. As we mentioned, the performance of *Random* did not change much with different Zipf-parameter values.

As Figure 1 shows, the gains increase logarithmically with increased storage capacity. The main determining factor is the Zipf-parameter value. The larger this value is, the smaller is the number of objects generating a large amount of requests. Thus, it is easy to significantly reduce the cost if only a small number of objects is very popular. To reduce the number of hops by 50%, we need only a small amount of storage for parameter 1.0 and up to 25% of total data set for parameter value 0.6.

In Figure 2 we plot the results from experiments

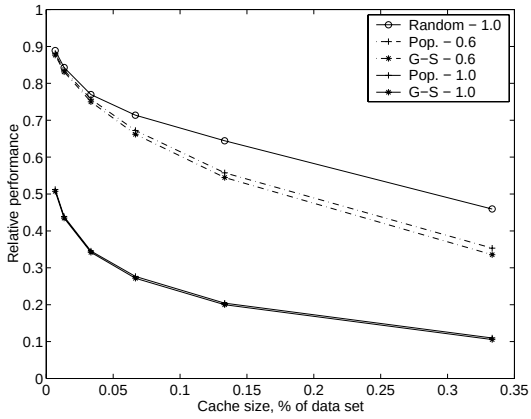


(a) 99/01/11

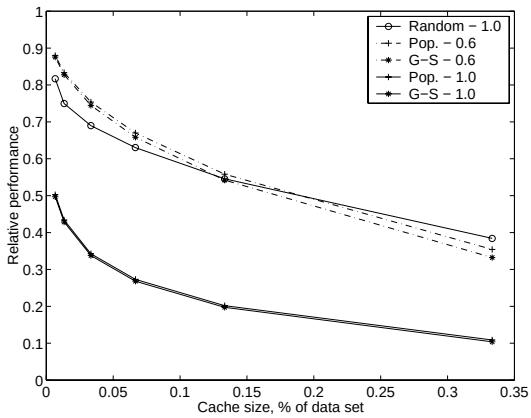


(b) 99/12/08

Figure 1: Experiments with 1,000 objects



(a) 99/12/08



(b) 99/12/11

Figure 2: Experiments with 10,000 objects

with 10,000 objects. Due to memory limitations, we were not able to run *Greedy-Global* for all the topologies, therefore it is not shown on the plots.

The results are very similar to the results from the previous experiment. *Random* is still the worst in terms of performance, but the difference between *Popularity* and *Greedy-Single* has decreased. This is because as the number of objects grows, the individual object popularities will get smaller. Therefore, the contribution of an individual object's retrieval cost to the global cost (2) will get proportionally smaller. Hence, the popularity of the object becomes more important in determining the cost. The product of popularity and distance used by *Greedy-Single* is still a slightly better indicator, but the difference is minimal.

From our results we can conclude that the best performance is obtained when the object replication is coordinated by a single source, in our case, this would be the CDN. The difference in performance between *Greedy-Global* and the other two heuristics is quite significant, especially for small Zipf-parameter values. In Figure 1 we can see that the largest improvements in performance are up to 24%. This result shows that *a CDN should use a coordinated replication strategy* and not let the CDN servers act on their own.

6 Peer-to-Peer Content Distribution

We now introduce a new form of content distribution, namely peer-to-peer content distribution. Peer-to-peer

networks have recently emerged as a new form of content distribution and are mainly used to share individual files between users. In peer-to-peer networks, such as Napster [11], or Gnutella [7], individual users decide to share files with others. With the help of a directory service, users can determine where different files can be downloaded from. While this new model does not directly compete with the traditional Web browsing model, it is establishing itself as a means for distributing larger files, such as applications or music, between users. Because peer-to-peer networks are made up of individual users, we cannot use strategies which require global information and coordination of nodes as with CDNs. Instead we must restrict ourselves to strategies that need only locally available information; one example of such strategy is the *Popularity* heuristic.

As before, we assume that one AS in our network corresponds to one ISP. We can view the storage capacity in an AS as the aggregation of the peer-to-peer storage offered by the users in that AS. In a similar vein, $x_{ij} = 1$ means that at least one user in AS i has a copy of object j. We assume that the cost of retrieving objects from other users in the same AS is negligible; this is consistent with our definition of retrieval cost in Section 3. In this section we will investigate the benefits of cooperation between the users in different ASs. Our goal is to see if the users in a peer-to-peer network could gain anything from cooperating with other nearby users.

We will now develop a cooperation model for peer-to-peer content distribution under the popularity heuristic. For simplicity and ease of notation, we shall assume throughout this section that all objects have the same size, i.e., $b_j = b$ for $j = 1, \ldots, J$.

Consider two ASs, A and B. Denote the shortest path between them by D_{AB}. Let K be the number of objects that each AS can store. We do not assume anything about the relationship between the two ASs, except that the distance between them is D_{AB}. In particular, we do not assume that one is the access provider of the other one. We assume that $\lambda_A = \lambda_B$.

If both ASs act independently, they would both cache the K most popular objects and the average number of hops for requests from A and B would be

$$\begin{aligned}&\frac{\lambda_A}{\lambda_A+\lambda_B}\sum_{j=K+1}^{J} p_j d_{Aj} + \frac{\lambda_A}{\lambda_A+\lambda_B}\sum_{j=K+1}^{J} p_j d_{Bj}\\ =&\frac{1}{2}\sum_{j=K+1}^{J} p_j d_{Aj} + \frac{1}{2}\sum_{j=K+1}^{J} p_j d_{Bj} \qquad (3)\end{aligned}$$

We now consider the case where the two ASs cooperate and do not necessarily both store the same objects. We assume that both of them store a copy of the L most popular objects ($L \le K$), and that in addition A stores objects $(L+1), \ldots, K$, and B stores objects $(K+1), \ldots, (2K-L)$. Because the request rates are identical, it does not matter how the objects $(L+1), \ldots, (2K-L)$ are shared between A and B. Note that because of the same reason, we only need to consider replicating objects in A and B and not in the intermediate nodes.

The average number of hops for requests from A and B under this scheme is

$$\begin{aligned}&\frac{1}{2}\sum_{j=K+1}^{2K-L} p_j D_{AB} + \frac{1}{2}\sum_{j=2K-L+1}^{J} p_j d_{Aj}\\ &\quad+\frac{1}{2}\sum_{j=L+1}^{K} p_j D_{AB} + \frac{1}{2}\sum_{j=2K-L+1}^{J} p_j d_{Bj}\\ =&\frac{1}{2}\sum_{j=L+1}^{2K-L} p_j D_{AB} + \frac{1}{2}\sum_{j=2K-L+1}^{J} (p_j d_{Aj} + p_j d_{Bj}) \qquad (4)\end{aligned}$$

The difference between (3) and (4) is

$$\frac{1}{2}\sum_{j=K+1}^{2K-L} (p_j d_{Aj} + p_j d_{Bj}) - \frac{1}{2}\sum_{j=L+1}^{2K-L} p_j D_{AB} \qquad (5)$$

If (5) is greater than zero, then it is better for A and B to cooperate. By assuming $d_{Aj} = d_{Bj} = d_{avg}$, we can calculate the value of d_{avg} where cooperation becomes the preferred strategy. The equation becomes

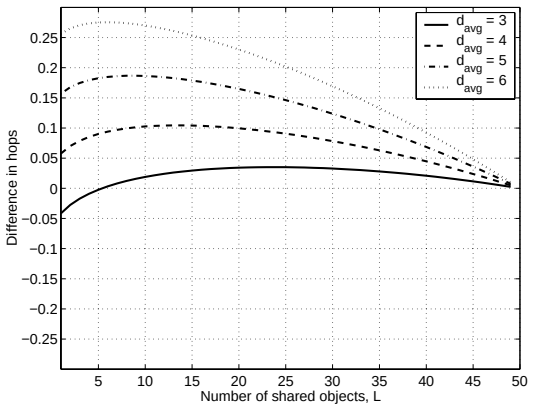
$$\frac{1}{2}\sum_{j=K+1}^{2K-L} (p_j d_{avg} + p_j d_{avg}) - \frac{1}{2}\sum_{j=L+1}^{2K-L} p_j D_{AB} > 0 \qquad (6)$$

Solving for d_{avg}, we get

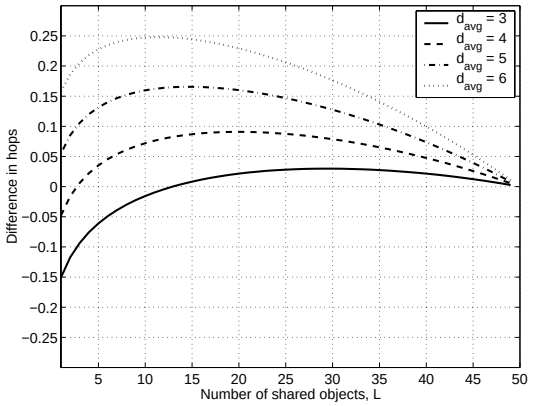
$$d_{avg} > \frac{D_{AB}}{2}\frac{\sum_{j=L+1}^{2K-L} p_j}{\sum_{j=K+1}^{2K-L} p_j} \qquad (7)$$

If (7) holds, then it is better for A and B to cooperate. Because d_{avg} is likely to be reasonably stable over long intervals and because p_j's are known to both parties, A and B can use (7) as a quick test to see whether they could gain anything by cooperating. For the test, A and B either need to specify the value of L or verify equation (7) over several values of L.

In the following, we will consider two values for D_{AB}, namely 1 and 2. If D_{AB} is equal to 1, then A and B are neighbors. We plotted (5) for several Zipf-parameter values, average distances d_{avg}, and values of K, and in all cases, cooperation is almost always superior to a non-cooperating strategy. In some instances, however, the difference between the two is small.



(a) Zipf 0.6



(b) Zipf 0.8

Figure 3: Gain from cooperation for $K = 50$, 1000 objects

In Figure 3 we show typical plots of (5). In these plots, we have 1,000 objects available and both nodes have capacity to store 50 objects, or 5% of the data set. We show two plots for two different values of the Zipf-distribution parameter. On the x-axis we plot the value of L, i.e., the number of objects stored at both A and B, and on the y-axis we show the value of (5), that is, the difference between individual and cooperative strategies in hops. If the difference is positive, then cooperation is better.

As we can see from the plots in Figure 3, there are always some values of L for which cooperation gives better results, but in some cases the gains are small. We can see that as d_{avg} gets smaller, the gains become smaller. This is no surprise since a small d_{avg} means that the requested objects are typically already very close and cooperation would not help much. We also see that as d_{avg} increases, the potential gains increase significantly.

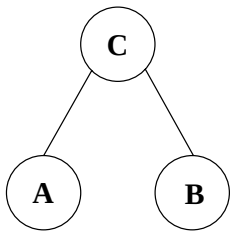


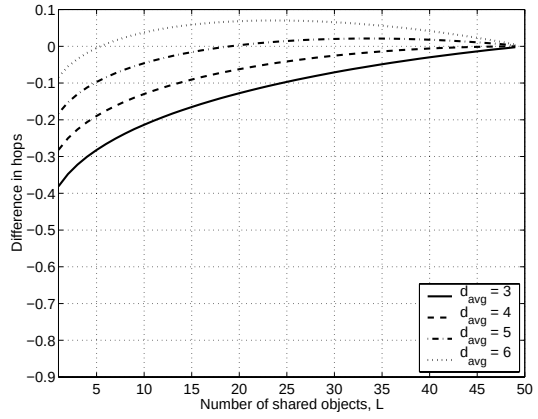
Figure 4: 3-way cooperation

Even though a gain of 0.2 hops may not seem much, it is important to note that the difference between *Popularity* heuristic and *Greedy-Single* was typically less than 0.05 hops. The difference between *Popularity* and *Greedy-Global* was 0.1–0.15 hops. Therefore, two cooperating nodes using the *Popularity* heuristic would obtain significantly better performance than they could hope to obtain by acting independently. We can conclude that *cooperation is much more efficient at reducing the cost* than changing a heuristic from a popularity based heuristic into a greedy one.

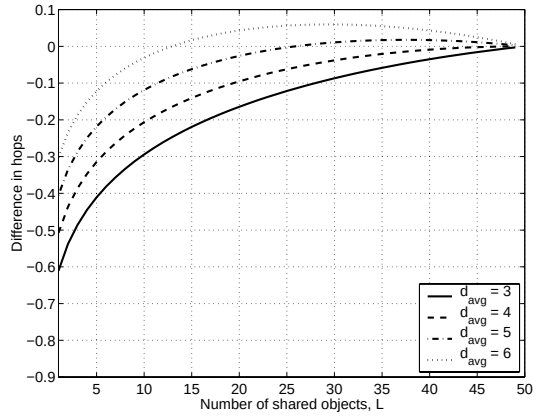
If D_{AB} is equal to 2, then A and B are separated by one AS. One example of this case is shown in Figure 4 where we have three nodes, one parent and two children. In this case, the cooperation would happen between the children. Figure 5 shows the graphs for this case.

Comparing Figures 3 and 5 we can see that the gains get smaller as D_{AB} increases. When D_{AB} increases even further, cooperation will no longer be beneficial to A and B. This is to be expected, since in the network topologies we used, the average distance between leaf nodes was around 4 and therefore the distance between A and B would be roughly the same as the distance to the origin server.

We also investigated cases where there are a very large number of objects in the network. In these cases, the general form of the gains matches those in Figures 3 and 5, but the actual gain is slightly lower. This is because the storage nodes can only hold a very small fraction of the objects, and even two nodes cooperating cannot hold enough objects to reduce the average number of hops significantly. However, even in these cases, cooperation yielded a smaller average number of hops than not cooperating. For example, with 1 million objects and K equal to 2000, the maximum gain was around 0.2 hops as opposed to 0.27 with 1000 objects and $K = 50$ (for $d_{avg} = 6$).



(a) Zipf 0.6



(b) Zipf 0.8

Figure 5: Gain from cooperation for $D_{AB} = 2$, $K = 50$, 1000 objects

7 Related Work

Related work on replicating content has mostly concentrated on the problem of placing the replica servers for *one origin server*. In this paper we consider a more global case where we have content from several origin servers and we decide which objects to replicate on the replica servers.

In both [10] and [3], the authors consider the problem of placing replicas for one origin server on a tree topology. While the real Internet topology is not a tree, this simpler approach allows the authors to develop optimal algorithms. The tree-approach may not generalize, because it would require that the trees for different origin servers overlap at the replica sites which cannot be guaranteed without a manual selection of the replica sites. Also, in [10], the algorithm is of a high computational complexity $O(N^3M^2)$.

In [14] the authors present their algorithms for placing server replicas in a CDN. They assume that the replicas are complete replicas and they do not study replicating individual objects; in our work we make replication decisions on a per-object granularity. They formulate the problem as the NP-complete K-median problem, develop heuristics and evaluate their performance. They only consider placing replicas for a single origin server; our heuristics replicate objects from all the origin servers. All of their heuristics require information about the network topology as well as client request loads. Also, in their work, all the replicas act independently and they do not study any cooperating schemes.

In [9] the authors consider the placement of intercepting proxies inside the network to reduce the download time. They present optimal solutions for simple topologies, such as line and ring, and consider the case of placing proxies for a single server in a tree topology. Relying on intercepting proxies requires that routing is stable during the lifetime of the connection.

8 Conclusion

In this paper we have studied the problem of optimally replicating objects in CDN servers. We treat each AS as a node with finite capacity for storing objects. Our optimization problem is to replicate objects so that when clients fetch objects from the nearest CDN server, the average number of ASs traversed is minimized. We have formulated this problem as a combinatorial optimization problem and have shown it to be NP complete.

We have developed four natural heuristics and compared them numerically using real Internet topology data. Our results show that the best performing heuristic is *Greedy-Global* which has all the CDN servers cooperating. The difference in performance between *Greedy-Global* and the simpler heuristics was up to 24%.

We have also studied peer-to-peer content distribution and developed a model for studying the benefits of cooperation between nodes. Our evaluation of the cooperation model shows that nodes using simple heuristics and intelligent cooperation can get significant performance gains.

The field of content distribution and peer-to-peer networks has significant potential for future research. Our future work will look more closely into inter-node cooperation and investigate the optimization problem

more closely in order to establish lower bounds on the achievable performance. We also plan to extend our cost model to include other important factors, such as network traffic or server load.

Acknowledgements

The routing data summaries used in constructing the network topologies were provided by National Science Foundation Cooperative Agreement No. ANI-9807479, and the National Laboratory for Applied Network Research. We would also like to thank Anat Bremler-Barr of Tel Aviv University for the discussions about the topology models.

References

[1] Akamai. <http://www.akamai.com>.

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of IEEE Infocom*, New York, NY, March 21–25, 1999.

[3] I. Cidon, S. Kutten, and R. Soffer. Optimal allocation of electronic content. In *Proceedings of IEEE Infocom*, Anchorage, AK, April 22–26, 2001.

[4] Digital Island Inc. <http://www.digisle.net>.

[5] L. Gao. On inferring autonomous system relationships in the internet. In *Proceedings of IEEE Global Internet*, San Francisco, CA, November 28–30, 2000.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[7] Gnutella web site. <http://www.gnutella.co.uk/>.

[8] S. Jin and A. Bestavros. GreedyDual* web caching algorithm: Exploiting the two sources of temporal locality in web request streams. In *Proceedings of 5th Web Caching and Content Distribution Workshop*, Lisbon, Portugal, May 22–24, 2000.

[9] P. Krishnan, D. Raz, and Y. Shavitt. The cache location problem. *IEEE/ACM Transactions on Networking*, 8(5):568–582, October 2000.

[10] B. Li, M. J. Golin, G. F. Italiano, and X. Deng. On the optimal placement of web proxies in the internet. In *Proceedings of IEEE Infocom*, New York, NY, March 21–25, 1999.

[11] Napster. <http://www.napster.com>.

[12] NLANR measurement and operations analysis team. <http://moat.nlanr.net>.

[13] V. N. Padmanabhan and L. Qiu. The content and access dynamics of a busy web site: Findings and implications. In *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, August 28 – September 1, 2000.

[14] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *Proceedings of IEEE Infocom*, Anchorage, AK, April 22–26, 2001.

[15] Route views project homepage. <http://www.antc.uoregon.edu/route-views/>.