AN EFFICIENT CDN PLACEMENT ALGORITHM FOR HIGH-QUALITY TV CONTENT

Adrian J. Cahill
Department of Computer Science
University College Cork
Cork, Ireland
email: a.cahill@cs.ucc.ie

Cormac J. Sreenan
Department of Computer Science
University College Cork
Cork, Ireland
email: cjs@cs.ucc.ie

ABSTRACT
Content Distribution Networks (CDNs) are used extensively on the WWW to relieve hot spots and reduce client latency when delivering web documents, but recently their role has been extended to the delivery of streaming multimedia content. Efficient placement of replicas in a CDN is very important, particularly when the content is large in size as is the case with high-quality TV content. This paper outlines a scalable and efficient algorithm for the placement of video content in a Video CDN (VCDN). This paper evaluates the effectiveness of the placement algorithm, detailing the important parameters associated with the algorithm. The VCDN placement algorithm is also compared with the well known *closest-proxy* algorithm to evaluate its effectiveness.

KEY WORDS
Content Placement, Video Distribution, CDN Management

1 Introduction

This paper describes an architecture to store and deliver high-quality TV content over the Internet. This new approach to TV delivery is expected to overcome the reliance on TV broadcast schedules, remove the need for local storage (VCR & PVR) while improving the client viewing experience by providing features such as a networked archive of all TV content broadcast over a period of time.

There are a number of key issues which must be examined before such a system can be designed. The resources required to deliver a video stream can quickly become exhausted if a large number of clients request an object over a short period of time, to overcome this issue scalable video servers such as TigerShark [1] could be used, but these are very expensive and a number of these would be required. Another solution is to use the services of Content Distribution Networks (CDNs) such as Akamai [2]. CDNs replicate popular content on a number of strategically placed servers within the Internet. This provides multiple access points to content resulting in lower latencies experienced by clients while also distributing the load among the surrogate servers. The efficiency of a CDN can be affected immensely by the replica placement algorithm in use, particularly when the content to be replicated is very large such as high-quality video objects. This paper describes a Video Content Distribution Network (VCDN) and replica placement algorithm used to deliver high quality TV content efficiently. The replication algorithm is designed to automatically replicate or move content according to changes in client access patterns, resulting in efficient resource usage.

The remainder of this paper is organized as follows. Section 2 outlines some of the related work examined. Section 3 presents the system architecture and cost model. Section 4 describes the placement algorithm and cost functions used. Section 5 outlines the simulation procedure, explaining the algorithms, their parameters and other factors which influence the efficiency of VCDN. Section 6 explains the simulation results and finally section 7 shows the authors' conclusions and future plans for this project.

2 Related Work

The replica placement problem can be modelled as the well known *Facility Location Problem* [3], which has been described as an NP-hard problem.

In [4] the authors tried to solve this problem by minimizing the number of Autonomous Systems traversed when clients request objects from the server. They propose using combinatorial analysis to locate the optimal placement layout, which can be computationally expensive in large scale networks such as those expected in the VCDN architecture. This problem would be amplified by the need to constantly assess the current replica placement to ensure the system is always in a resource effective state. In [5] Qiu *et al* look at modelling the replica placement problem as a K-median problem. Here, the authors use a combinatorial algorithm which as previously mentioned is not scalable. Li *et al* [6] also carried out research in this area, but the authors only considered a tree topology network which is unrealistic in the current Internet.

Other related work includes the use of Peer-To-Peer (p2p) systems such as [7] to distribute streaming media, though these suffer from a number of problems such as client bandwidth requirements. Content management could

also become an issue as there would be very little control over the dissemination of content, this would not be suitable to TV content as content creators are unlikely to allow the distribution of their content without some control over its viewing.

3 Architecture Details

In this section, the architecture behind the VCDN is described, while also showing how this system can provide an improvement on the current continuous media delivery system. By placing video proxies/servers throughout the Internet, each with the ability to store and replay content, a network storage facility could in theory be provided for all video content which is being broadcast throughout the world. Clients would then be permitted to retrieve content from the most suitable video proxy which contains the desired content. In doing this, the following goals can be achieved:

- Eliminating the necessity for clients to store content and hence reducing the need for client disk-space.
- Minimize the possibility of clients sharing content, without the content creators consent.
- Reduce the need for clients to explicitly or implicitly schedule a recording for a program which he/she will not be able to view.
- Provide a far greater choice of content to the clients, as any content which is stored within the network could be available to the clients as long as certain requirements have been met (e.g. some content may be pay per view).

This approach is expected to remove the restriction on content transmission such as TV broadcast range, and deliver the content to all clients connected to the Internet. It is assumed that clients wishing to view *live* content can do so without having to burden the video servers, but rather by joining the relevant multicast group and viewing the content as it is streamed to their set-top box. Clients who wish to view this content at a later stage, may do so by contacting the relevant proxies in an on-demand fashion. Fig: 1 depicts a possible layout of the VCDN architecture, including the steps involved in requesting a media object. The details of this VCDN are described in more detail in a previous paper by the authors [8].

3.1 Assumptions

Currently, on-line storage businesses provide storage facilities by renting disk space in bulk. Our research looks at a different pricing scheme, where storage can be purchased on a per-byte basis. It is believed that by purchasing storage in this fine-grained model we could provide a more cost effective solution for video distribution. Future work will look at comparing our placement algorithm when storage can be bought in bulk and on a per bytes basis.

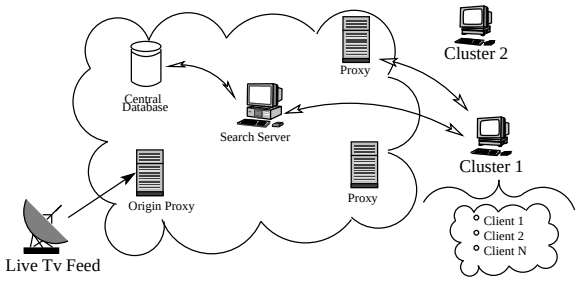

Figure 1. VCDN components

3.2 System Elements

Proxy A proxy is a server located within the Internet which can store and deliver video content to clients. Proxies can be in any of two states, active or inactive. A proxy can store and deliver content only when in the active state. A proxy in an inactive state is assumed to use no resources.

Resource Proxies, storage space and network traffic are all quantifiable elements which can affect the performance of the system, and thus are labelled resources.

Client A client is any set-top box or PC, which is used to retrieve content from a proxy within the network.

Cluster Clusters are described as groups of clients which are all viewing the same content, have similar network properties and located in close proximity within the network.

Cluster Load The cluster load is the sum of the load generated by all clients which are members of that cluster.

Central Database The central database is used to store information about the content currently available within the network and also information regarding the current state of the VCDN.

Search Server The search server is the client interface to the system. Clients query the search server with various parameters describing a media object and a link is returned to the most suitable object replica that is available.

3.3 Costs Involved in the CDN

To be able to quantify the state of the VCDN it is necessary to assign costs to specific elements of the VCDN architecture which could affect the overall efficiency of the network. These costs can later be weighted in accordance with

what resources the VCDN owner choose to minimize, for example if the VCDN owner was not concerned with storage costs (due to the low cost of hard disks) then they could reduce the weight on the storage cost, thus making other costs more influential when deciding on content placement. The following section outlines the elements which are believed to be of influence during proxy placement.

Streamed Network Cost β_r A cost associated with *streaming* video content from video proxy *r*. This is considered to be a cost unit per byte, per hop.

Bulk Transfer Network Cost $\alpha_{s,d}$ A cost associated with the transfer of a video object from video proxy *s* to proxy *d*. For the purpose of our research we consider this to be a cost unit, per byte, per hop. This is considered separately to *streamed network cost* as this provides a fine grained cost model which facilitates a different pricing depending on the network requirements of the connection, for example QoS requirements.

Storage Cost λ_d A cost associated with storing content on proxy *d*. This is considered to be a cost unit per byte, per unit time.

Each cost outlined above could be set for each individual proxy, thus allowing a placement algorithm take advantage of a fine-grained cost model. This would allow certain proxies to be considered more valuable than others, for example a proxy in a large city could have a larger *storage cost* than that of a proxy in a rural area. The costs outlined above are all stored in a *central database* and the content is *pushed* out to the proxies periodically. This database can also be queried on-demand, in the event of a new proxy joining the CDN.

4 VCDN Placement Algorithm

The placement of content within any CDN is of great importance to the overall efficiency of the network. Content should be placed in a location which yields the lowest resource cost when delivering the content to its clients. One of the largest costs in such a system is that of *network cost*, as this influences many aspects of content delivery including latency experienced by the client. For this reason, the VCDN placement algorithm has been designed to automatically react with changing client trends, and dynamically move or replicate content accordingly. Using the cost variables mentioned in the previous section, a cost function was designed as a means of locating the optimal proxy (yields lowest resource cost) to deliver content to a set of clusters.

4.1 Cost Function

We have developed a cost function which factors in the resources required to serve a given client request and determines which server requires the least resources to serve the client. The work presented in this paper focuses on clusters of clients rather than individual clients defined as follows: **A cluster is a group of clients located within the same region of the network, who are all viewing the same object**

P_i	Proxy *i*
$S(m)$	Filesize of movie *m* (bytes)
B_c	Bytes remaining to be served in cluster *c*
$\delta_{src,dst}$	Num of network hops between *src* and *dst*
λ_d	Cost of storing 1 byte on Server *d* (per unit time)
T_{max}	Time required to serve the client earliest in the stream for this cluster
$RC_{s,d,m}$	Cost of replicating movie *m* from server *s* to server *d*
$SC_{d,m}$	Cost of storing movie *m* on server *d*
$Cost_{s,d,m,N}$	Total cost of serving movie *m* to all *N* clients from proxy *d*, including replciation cost from proxy *s*

Figure 2. Parameters to Cost Function

The cost of using proxy (P_d) for serving *N* clients watching movie *m* is given by the following equations:

$$RC_{s,d,m} = \alpha_{s,d} * \delta_{s,d} * S(m) \quad (1)$$

$$SC_{d,m} = \lambda_d * S(m) * T_{max} \quad (2)$$

$$Cost_{s,d,m,N} = [RC_{s,d,m}] + SC_{d,m} + \sum_{c=1}^{C} (\beta_r * \delta_{d,c} * B_c) \quad (3)$$

where B_n can be calculated as follows, if the client has just started then B_n = S(m), otherwise B_n = S(m) - *bytes already served for that cluster*.

By periodically executing this cost function at a proxy and altering content placement based on its results, we can ensure that the system always stays in a cost effective state.

4.2 Dynamic Content Placement

The previous section outlines how two proxies can be compared to determine which proxy would consume the

least amount of resources while delivering content to a set of clusters. A more interesting problem is providing a fine-grained replication service, whereby a subset of objects are replicated on a number of different proxies which is expected to yield optimal resource efficiency.

A combinatorial search of each proxy and cluster pair is required to determine the optimal replica placement, for each combination the previous equation must be calculated to determine the effect of replication. This process needs to be repeated regularly (function of client join times) to ensure that the VCDN remains in a cost effective state. In an effort to make this process scalable, it is proposed that the static nature of some of the values in the equation be exploited and pre-calculated, the results of which form a cost table (fig: 3). This process is described in more detail in previous work by the authors [8].

	$Proxy1$	$Proxy2$	$Proxy3$	...
Region 1	$\beta_1 * \delta_{1,1}$	$\beta_2 * \delta_{2,1}$	$\beta_3 * \delta_{3,1}$	...
Region 2	$\beta_1 * \delta_{1,2}$	$\beta_2 * \delta_{2,2}$	$\beta_3 * \delta_{3,2}$	...
...				

Figure 3. Example cost table showing the costs of serving 1 MB of data from each proxy to each region of the network, where region is any location where a cluster can form (eg autonomous systems or a subset of IP addresses)

In the previous sections of this paper we have looked at the replica placement problem, which is an integral part of a Video Content Distribution Network. The remainder of this paper evaluates the VCDN placement algorithms along with another well known placement algorithm. We also look at some parameters for the VCDN algorithm and their effects on resource management.

5 Evaluation Methodolgy

In this section we evaluate the efficiency of both the *VCDN* and *closest-proxy* placement algorithms at delivering high-quality media objects to a number of clients. As part of the evaluation, client viewing patterns must be considered, particularly the likelihood of a TV viewer of *channel-hopping*. For the remainder of this paper, the term *channel-hopping* refers to a client requesting a movie object and terminating the stream within a short period of time. Firstly, we describe how the placement algorithms operate:

Closest Proxy Algorithm (CP) For the remainder of the paper we will refer to this algorithm as *CP*. This algorithm makes replication decisions based solely on network hop count (always tries to minimize hop count, regardless of other resource costs). As each cluster joins the network, it connects to its closest proxy, and requests a media object. In the event that that object is not currently stored on this proxy, a replica is requested. The replication process is carried out using a write-though protocol (similar to that employed in caching techniques). This ensures that a the cluster is always receiving content from its closest proxy.

Video Content Distribution Network (VCDN) This algorithm makes replication decisions based on resource costs (*currently: link cost and storage cost*). When a client joins a cluster, the proxy which is currently serving the cluster initiates a cost evaluation procedure, whereby the proxy checks to see if any of its existing clusters (viewing a particular object) would be better served by a different proxy. This is evaluated using the cost function described in section 4.1. The evaluation procedure will determine if the object should be replicated and if so, where to replicate the object.

VCDN Evaluation Time The VCDN algorithm also includes a parameter which is used to delay the evaluation procedure which would be executed when a client joins. It is hoped that by delaying the evaluation by a predetermined amount of time, the placement algorithm will not make replication decisions that may be adversely affected by the *channel-hopping* effect. The experiments were carried out with this parameter varied between 0, 1 and 5 minutes, as can be seen in legend at each graph as (EV_0, EV_1, EV_5).

5.1 Simulation Setup

To examine the performance of these algorithms, a number of experiments were carried out. Initial experiments were performed with a small number of clients and small topology to easily verify the placement decisions of the algorithms. Later experiments were carried out with a larger client set and more realistic workloads. Each experiment was carried out a number of times with different weights, where the weight was varied between 0 and 1. This weight was applied to the *storage cost* and inversely applied to the *link cost*. A number of clients were selected (9 clients for the initial experiments, 100 clients for the realistic workload set experiments). Each experiment was carried out until all client requests had been fully satisfied, which varied from 4 minutes (all clients departed after 4 minutes of the stream) to 92 minutes (the last client from the realistic workload finished). The frequency of re-evaluation depends on the algorithm being executed. The CP algorithm, identifies the closest proxy when the first client joins, and sticks with this proxy (as this will always be the closest proxy for this cluster). On the otherhand, the VCDN algorithm re-evaluates the placement whenever a new client joins the system.

The following depicts a lists of the interesting parameters which have a great influence over the results of the experiments:

Resource costs In the initial experiments a weight is assigned to the storage and inversely to link costs, and these were varied between 0 and 1. As expected, during times when link cost is low, few or no replicas are made under the VCDN algorithm, and when storage cost is low replication is more prominent. This cost has no effect on the placement decision of the CP algorithm, as decisions are based solely on hop count.

Viewing Patterns Tang *et al* [9] examined the trace logs of *HPLabs Media server* over a 21 month period and created *MediSyn* which can be used to create synthetic media workload files. We used this tool to develop a number of workloads with varying parameters for the client viewing duration, which the authors claim follows an *exponential* distribution for the first 5 minutes and a *normal* distribution thereafter. The authors state that the initial portion of the client viewing time distribution follows a exponential distribution (up to a period of time, 5 min's for example) after which clients tended to watch the entire stream, this is as you would expect with TV viewing. We varied the parameters to these distributions to explore the effects of users mostly viewing the complete object and users mostly viewing a small portion of the object.

Topology For the purpose of these experiments we used small topologies (6 interconnected proxies, with varying hop counts) to show some of the interesting aspects of how these algorithms compared. Future work will look at extending the scale of these simulations.

6 Simulation Results

In this section we evaluate the performance of both algorithms under a variety of client workloads. In the initial tests we used the following setup: 9 clients, 3 clusters and 1 movie. The goal of these tests was to show the mechanics of the VCDN algorithm. In later tests (Figs. 5,6), the experiments were carried out on 100 clients. A number of realistic client workloads were generated using the *MediSyn* tool, each showing the effects of different client viewing durations (simulating the severity of *channel-hopping*), we have included a subset of these results.

6.1 Small Client Load

Fig. 4 shows the performance in terms of overall system cost of delivering 1 movie to each of the 3 clusters, with 3 clients in each cluster. As can be seen, when *storage cost=1* and *link cost=0*, the VCDN algorithms opts to use the origin server to serve all clusters, whereas CP replicates and thus yields a poorer resource cost. On the other hand, when *storage cost=0* and *link cost=1* the VCDN algorithm with delayed evaluation performs poorest due to the delay in the replication decision. But the VCDN with no evaluation delay performs better, as it only creates a single replica of the

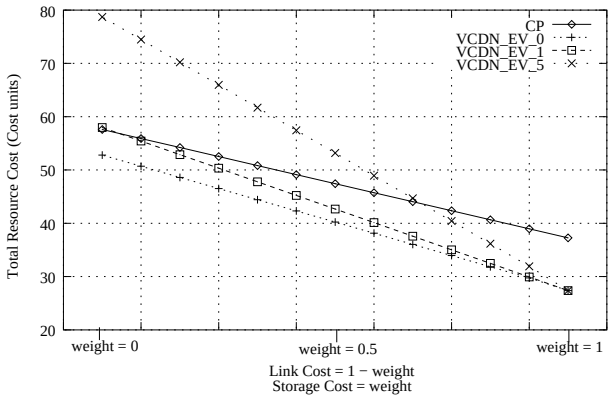

Figure 4. Shows 3 clients per cluster, all viewing movie: 1 to completion.

movie on a common proxy close to all requesting clients. This is also evident for the intermediate point of the graph also, whereby CP creates an individual replica per cluster, VCDN tries to maximize resources, thereby limiting the replicas where possible. This experiment was repeated, with different client-cluster configurations, each yielding similar results.

6.2 Realistic Client Load

The following experiments (Figs. 5,6) were carried out with 100 clients, each randomly distributed among 6 clusters. The client viewing patterns used in these experiments were obtained from the *MediSyn* media workload generator. The client viewing duration was altered from 16% of clients viewing the complete object to 50% of clients viewing the complete object. The storage and link weight was also varied between 0 and 1.

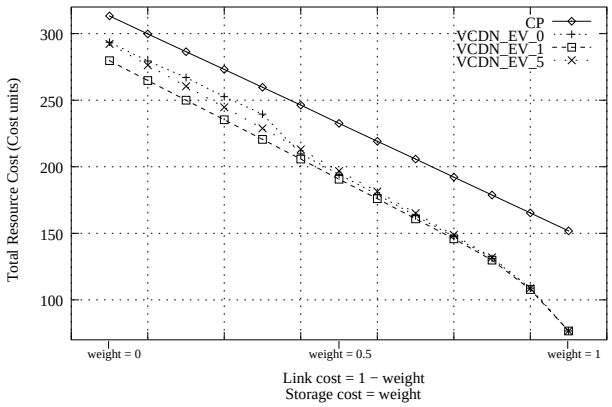

Figure 5. Realistic workload, 50% view full movie, 30% dropout within 60 seconds.

Fig. 5 shows that the VCDN algorithms with a marginal evaluation delay perform best, as this ignores clients requests for a movie with duration $<$ 60 seconds

which accounts for between 30% - 45% of the requests. It is also for this reason that the VCDN algorithm with no evaluation delay didn't perform well, though it still performed better than the CP algorithm. Again, the CP algorithm performed poorly, as replicas were being placed on proxies for short periods of time and then being removed when there was no requests. We also ran the CP algorithm with a built in movie removal delay of 5 minutes, in the hope that this would significantly improve the results, but on average this performed as poorly, and worse in some cases. The sudden drop in cost for the VCDN algorithm with 0 delay (vcdn_ev_0, when the weight = 0.4) can be attributed to the lowered impact of the link cost in the cost function, this results in less harsh decisions being made by the placement algorithm.

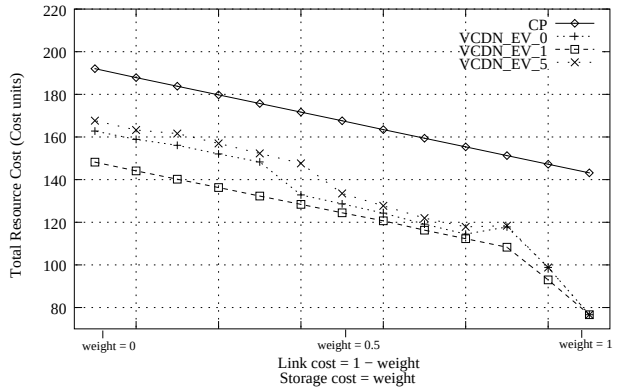

Figure 6. Realistic workload, 16% view full movie, 45% dropout within 60 seconds.

In Fig. 6, the VCDN placement algorithms with a high evaluation delay surprisingly perform poorly even though they ignore many of the channel hopping clients, they also penalize the clients whose viewing time $>$ 5 minutes. When the weight is low, resulting in a high link cost, the algorithm creates many replicas, but at the point when the *weight=0.5*, the algorithm improves in performance considerably. This is due to the reduction in the number of replicas created as a balance between link cost and storage cost is found. The costs continue to be higher than the other VCDN algorithms, but this is due to the extra delay needed before the algorithms decides to make a replica. Again, the VCDN algorithms exhibit rapid drops in their resource usage, this is due to the effects of the assigned weight to the link/storage costs. The steps represent the algorithm reducing the number of replicas it uses.
On average VCDN with a low evaluation delay can perform upto 30% better than the CP algorithm. This can be attributed to the manner in which CP makes placement decisions. Although, the current simulations were carried out on a small topology, a large topology would only further increase the resources used by the CP algorithm as there would be far more proxies and clusters to replicate on.

7 Conclusion and Future Work

In [8] the authors propose a replication placement algorithm for video content based on minimizing the resources required to serve the object. The proposed solution involved performing combinatorial searched whose search space was dependant on the number of proxies and the number of clusters viewing the object. This paper evaluates the efficiency of the VCDN placement algorithm under a variety of parameters, and concludes that an efficient placement algorithm should include a small evaluation delay to ignore channel-hoppers, while also looking at placing a single replica to serve multiple clusters.
Future work will look at adding hierarchy to the placement process, this should reduce the search space while keeping resource usage close to optimal. Further evaluation of the VCDN placement algorithm is needed, such as comparing it with other well known *replica placement algorithms*.

8 Acknowledgements

The authors would like to thank AT&T Labs-Research for supporting this project.

References

[1] R. L. Haskin, "Tiger Shark – A scalable fi le system for multimedia," *IBM Journal of Research and Development*, vol. 42, no. 2, pp. 185–197, 1998.

[2] "Akamai." http://www.akamai.com.

[3] V. Arya, N. Garg, R. Khanderekar, K. Munagala, and V. Pandit, "Local search heuristic for k-median and facility location problems," in *Proc. from ACM Symposium on Theory of Computing*, (Crete, Greece), pp. 21–29, July 2001.

[4] J. Kangasharju, J. Roberts, and K. Ross, "Object replication strategies in content distribution networks," in *Proc. of WCW'01: Web Caching and Content Distribution Workshop*, (Boston, MA, USA), June 2001.

[5] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *Proc. from IEEE INFOCOM*, (Anchorage, Alaska), pp. 1587–1596, April 2001.

[6] B. Li, M. Golin, F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the internet," in *Proc. of INFOCOM*, (New York, NY, USA), March 1999.

[7] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. of Workshop on Network and Operating System Support for Digital Audio and Video*, (Miami, FL, USA), NOSSDAV, May 2002.

[8] A. J. Cahill and C. J. Sreenan, "Vcdn: A content distribution network for high quality video distribution," in *Proc. from Information Technology & Telecommunications*, (Letterkenny, Ireland), October 2003.

[9] L. C. W. Tang, Y. Fu and A. Vahdat, "Medisyn: A synthetic streaming media service workload generator," in *Proc. of Workshop on Network and Operating System Support for Digital Audio and Video*, (Monterey, California, USA), NOSSDAV, June 2003.