Resource Management in a Shared Infrastructure Video CDN

Adrian J. Cahill and Cormac J. Sreenan

Mobile and Internet Systems Laboratory(MISL),
Department of Computer Science,
University College Cork,
Cork, Ireland
Email: {a.cahill,cjs}@cs.ucc.ie

ABSTRACT

In any large scale distribution architecture, considerable thought needs to be given to resource management, particularly in the case of high quality TV on-demand. This work presents a globally accessible network storage architecture operating over a shared infrastructure, termed Video Content Distribution Network (VCDN). The goal of which is to store all TV content broadcast over a period of time within the network and make it available to clients in an on-demand fashion. This paper evaluates a number of content placement approaches in terms of their ability to efficiently manage system resources. Due to the dynamic viewing patterns associated with TV viewing, the effectiveness of content placement is expected to change over time, therefore so too should the content placement. The placement of content within such a system is the single most influential factor in resource usage. Intuitively, the further content is placed from a requesting client, the higher the total bandwidth requirements are. Likewise, the more replicas of an object that are distributed throughout the network, the higher the storage costs will be. Ideally, the placement algorithm should consider both these resources when making placement decisions. Another desirable property of the placement algorithm, is that it should be able to converge on a placement solution quickly. A number of placement algorithms are examined, each with different properties, such as minimizing delivery path. There are a large number of variables in such a system, which are examined and their impact on the algorithms performance is shown.

1. INTRODUCTION

Digital video recorders (DVR) such as TiVo are changing the way TV is being viewed. The ability to intelligently record content and provide an easy to use play-back facility has meant that no longer is the viewer restricted to the TV broadcaster's schedule. The focus of this research is to extend the abilities of DVRs by removing the necessity for local storage, and replacing it with an *always-recording global storage*. In doing this, users of the system would no longer need to be sitting in front of their DVR to view a stored file, but instead would retrieve the object from a local video server over a high-speed Internet connection. Thereby providing a globally accessible TV on-demand facility.

There are a number of challenges that need to be overcome when designing a TV on-Demand (TVoD) system. Issues such as (i) Latency; clients will not be satisfied if the video does not begin streaming almost immediately, (ii) Reliability; the quality of the delivery stream should be consistently high, jitter and quality degradation are undesirable, (iii) Content Control; ensuring that Digital Rights Management infringements cannot occur, (iv) Resource Management; due to the size of these high-quality video files, operating and deployment costs for the system need to be monitored to ensure the system is operating as efficiently as possible. The primary focus of this research is the efficient management of resources when providing TVoD services. It is believed that resource management will become an important aspect of video delivery in the future, as the expected quality of video objects is always increasing to match improvements in client Internet connection speeds. If a TVoD system is to be deployed, then it should be capable of efficiently serving content to a similar sized user base as the existing terrestrial broadcasting model, while also maintaining a similar quality. System resources such as server storage space, disk and network I/O have always been an important factor in determining the scalability of multimedia distribution architectures. Optimizing the resources required to serve a number of clients involves optimizing (i) the content placement and (ii) the number of replicas of an object. But these parameters are influenced by the client request patterns, which may change over time; therefore it will be necessary to frequently re-evaluate the resource usage. Therefore, this work focuses on these issues. Although the process by which content is migrated between servers is an important one, it is beyond the scope of this work.

Multimedia Computing and Networking 2007, edited by
Roger Zimmermann, Carsten Griwodz, Proc. of SPIE-IS&T Electronic Imaging,
SPIE Vol. 6504, 65040D, © 2007 SPIE-IS&T · 0277-786X/07/$15

Previous work by the authors [1], proposed a new hybrid CDN-P2P architecture, termed Video Content Distribution Network (VCDN). VCDN is a content distribution infrastructure over which TVoD services can be provided. Unlike typical CDNs such as Akamai, VCDN does not operate over a private network, but rather over a shared network infrastructure. Resources such as distribution servers, storage space and bandwidth are leased from suitable service providers when client load increases, and when the load abates, the resources can be released again. This new CDN model removes the high start-up costs associated with CDNs, while also being dynamic in both size and resource usage. This is particularly important when considering TV distribution, as the expected user set, and hence resource requirements typically vary diurnally (less users in the morning than evening). In earlier work [1, 2], the authors carried out a preliminary evaluation of placement algorithms, looking at their performance under certain small scale conditions. That work acts as a precursor to the evaluation carried out in this paper. Currently, the VCDN architecture uses a computationally expensive placement algorithm, which is not useful for real world implementation due to its complexity. This work proposes and evaluates a number of heuristic placement algorithms, each of which aims to intelligently reduce the search space so that only potentially optimal proxies are considered for replica placement. In doing this the required execution time can be greatly reduced, while in some cases, suffering no loss of placement optimality. The algorithms proposed are based on a cost function, thereby facilitating their use in a generic environment, where for example the resource costs, or in fact the resources themselves can be different to that used in this work.

The remainder of this paper is organized as follows: Section 2 examines the plethora of related work that exists in this area. Section 3 gives an overview of the VCDN architecture and its new proposed extensions. Section 4 outlines the placement heuristic algorithms and their parameters. Section 5 describes the simulation environment and provides extensive evaluation of the proposed heuristic placement algorithms. Finally, Section 6 outlines the findings and conclusions of this work.

2. RELATED WORK

The decision over where content should be placed within a network can be modeled as a variation of the well known *Facility Location Problem* [3], which has been described as an NP-hard problem. The reason this problem is so difficult to solve is due to its inability to scale, as the number of possible outcomes to the problem is a function of the number of facilities and objects located at these facilities (proxies and replicas in the context of this paper). To overcome this issue a number of alternative approaches and heuristics have been proposed.

One proposed solution is to distribute the content within the network such that the overall distance between clients and their requested object is minimized. In [4] the authors proposed minimizing the number of Autonomous Systems traversed when clients request objects from the server. The authors compared four replicating strategies, and concluded that a single CDN server with knowledge of the entire system was capable of making the best decision as to the placement of proxies. They propose using combinatorial analysis to locate the optimal 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 frequently re-assess replica placement to ensure the system is always in a resource effective state. In [5] the authors only consider a tree topology network. By using a tree topology the authors have limited the path between any two nodes in the network, to a single path. The Internet is not configured in this fashion as there are a number of routes to a given destination. The authors also only deal with the instance where there is one web server which is cached at multiple locations. Qiu *et al* [6], model the replica placement problem as a K-median problem. In their work they assume that each replica site should contain a complete replica of the origin server. This is not desirable for TV content, as some media objects are expected to have higher request rate in certain areas of the network than others as would be the case with news programmes for example. Again the authors use a combinatorial algorithm which as previously mentioned is not scalable.

Probably the most relevant related work is that carried out by [7]. In their work the authors also propose using a shared network infrastructure as the basis for an overlay distribution network suitable for the distribution of large objects. The authors also propose a flexible network provisioning approach where resources such as storage and CPU can be leased as required. The authors provision their network based on a cost function and an expected *client demand* for a set of objects. As a result of provisioning their network in advance (based on this expected demand) the authors run the risk of over/under provisioning their network resulting in either resource starvation or waste. This would not be suitable for a TV distribution architecture as the variability in TV viewers is expected to be quite large between daytime hours and evening hours for an average weekday (due to potential viewers being at work/school). Therefore, particularly in a TV distribution network, content placement should be a frequently analyzed to ensure the network is operating in a cost effective manner. This could be achieved by dynamically adapting content placement with changing client loads.

P2P networks such as KaZaa and Gnutella have been used to distribute large objects such as feature length movies, but they typically perform this transfer without real-time constraints. Other issues also arise if P2P networks were to be considered as a suitable means to deliver high-quality TV objects, such as insufficient client uplink capacity, clients ability to depart from the network at will and also a lack of suitable digital rights management schemes. Also, although P2P networks have shown to be able to efficiently distribute popular objects, they may not be suitable for the distribution of unpopular content, as few requests will be made for the object and as a result, few replicas will exist. Some research exists which examines these issues such as [8, 9], but in general these problems still remain.

Xu *et al* [10] propose a novel distribution architecture to overcome some of the limitations in both CDN and P2P networks. A hybrid architecture uses a CDN to initially serve content to the requesting clients, with each participating client then forming a peer node in a P2P network. Over time the architecture changes from CDN-only to CDN-P2P and finally purely P2P with the CDN nodes being used for indexing purposes only. In doing this, the CDN can quickly seed a P2P network, and then remove the objects thus freeing the CDN servers for new content. Although this approach might work well for popular objects, less popular objects may reside on the CDN for long periods of time (as the P2P network will not be sufficiently seeded), as a result of this the CDN could become saturated with unpopular content thus reducing resources for newer popular objects. Also, the authors do not explain how content is initially distributed among the CDN servers, which would have a particularly large effect on resource usage within a distribution network for large objects such as high-quality video.

Finally, aside from P2P and CDNs, a number of broadcasting approaches have been proposed as a means of efficiently managing resources when distributing video objects to a large client base [11, 12]. Broadcasting approaches typically suffer from a lack of client control, as content is sent out over multicast channels from a server and periodic intervals, as a result of this, the delay between a client making a request for an object and the time when delivery begins may be quite high. Batching, patching and other improvements have been proposed in the past [13, 14, 15, 16], but these are all hindered by the lack of suitable multicast support within the Internet.

3. ARCHITECTURE DETAILS

This section provides the reader with an overview of the VCDN architecture and its core components, for a more detailed description of the architecture, the reader should look at [1]. In designing a suitable architecture for TV distribution a number of issues needed to be considered. TV viewing patterns change diurnally, i.e typically a higher percentage of the population view TV in the evening time than in the morning time. This can be attributed to users being in school or work. Previous work [1] has examined the suitability of both CDNs and P2P networks to TV distribution. They were both found to be lacking in a number of areas, such as lack of content control and reliability or dynamic expansion and running expense. To overcome these issues, the authors propose a new hybrid CDN-P2P architecture, termed VCDN. This architecture consists of leased service provider resources, such as storage space, CPU power and bandwidth. With such a design, this network does not incur a large setup cost, and can dynamically alter its size and resource requirements according to current system load. Under such an environment, minimizing the amount of resource usage is very important, as these resources are billed based on their usage. As a result of this, allowing objects to remain idle for long periods of time on a proxy is not desirable.

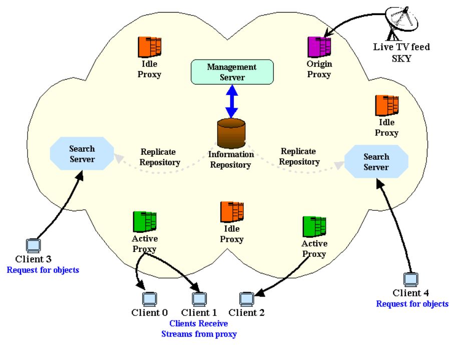


Figure 1. VCDN Component Layout.

3.1. VCDN Components

Fig. 1 shows a high level depiction of the VCDN components. Within the VCDN architecture, a number of VCDN-owned proxies would be deployed with TV record facilities. These proxies record live TV, digitize it and store the object along with suitable metadata. This content is now available for retrieval by system users. Clients can request objects using the search servers in either query or browse mode. Once a TV object is selected, the client is directed to the most suitable proxy which contains a replica. Content management aspects of the network, such as the number of replicas and their location within the network are maintained in the management server which can be queried and updated on-demand by the proxies. Details of this process are not discussed due to space constraints. The following is an overview of the relevant VCDN components and their role within the network.

Proxies

A proxy can exist in either an *Idle* or *Active* state. An *Idle* proxy is a proxy which can be leased from a service provider such as ISP or a Datacenter. When *Idle*, a proxy cannot directly partake in the distribution of objects. Under times of increased client activity, the Management Server may decide to dynamically lease one or more of these *Idle* proxies for use in distributing content, at which point the proxy will change state to *Active*.

Clients

A client consists of any set-top box or PC that is used to retrieve content from a proxy within the network.

Cluster

A cluster is described as a group of clients that view the same content and are located in close proximity within the network. Clustering is performed to reduce the complexity of content placement decisions, by assigning clients into clusters that are topologically close together we can infer that an optimal placement for the cluster is likely to be optimal for all the clients within the cluster (for a reasonably sized cluster). There have been a number of research papers such as [9] and [17] proposing clustering techniques and so will not be discussed herein.

3.2. System Resources

Current CDNs such as that provided by Akamai primarily focus on reducing the distance between a client and the requested object. This is achieved by replicating the requested object to a CDN server located in proximity to the client. CDN providers generally are not concerned with link costs and storage costs as they own all the required hardware. The approach outlined in this work assumes that the CDN operator does not *own* the resources, but rather that these resources can be leased on demand (for which a fee would be negotiated earlier). Therefore, when deciding on the number of replicas of an object to make, and where these replicas will be stored, in-depth analysis of the expected resource usage is required. This decision is made even more important by the large size of high-quality video files. This work captures the key resources involved in video distribution though the cost model could be extended to account for other resources quite easily.

Streamed Network Cost α_p

A cost associated with *streaming* video content from a video proxy 'p'. For the purposes of the cost function, this will be considered a cost *per byte / per hop*.

Bulk Transfer Network Cost $\beta_{s,p}$

A cost associated with the inter-proxy transfer of a video object between proxy 's' and proxy 'p', as occurs when a new replica is being created. For the purpose of this work, it will be considered a cost *per byte / per hop*. This is considered separately to *streamed network cost* to facilitate a cost model based on QoS requirements.

Storage Cost ϵ_p

A cost associated with storing content on a proxy 'p' . This is considered to be a cost *per byte / per unit time*.

Each cost outlined above could be set for each individual proxy, thereby allowing a placement algorithm to 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. In doing this, a competitive market could be formed, where ISPs lower their resource costs in a bid to entice more usage of their otherwise idle resources. Though the current resource costs are very biased in favor of minimizing bandwidth usage, the proposed cost function and placement algorithms allow for this work to be used in a very generic way. Thereby allowing it to be tailored to the current market environments and adapted to include other resources. The costs outlined above are all stored in a *information repository* and the content is *pushed* out to the proxies periodically. The repository can also be queried on-demand, in the event of a new proxy joining the CDN.

3.3. Assumptions

Currently, on-line storage businesses provide storage facilities by renting disk space in bulk. Our research looks at a different pricing model, 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 examine the effects of removing this assumption, and acquiring resources in bulk as is currently the case, for example renting server space from a Datacenter whereby they provide disk space in tens or hundred of gigabyte chunks.

3.4. Resource Management

Storage space and link bandwidth are the resources that are believed to most influence high-quality video distribution over a shared infrastructure. Due to the large nature of high-quality video files, their placement within the network can have immense impact on the resources required to deliver the objects. To this end, a cost function was proposed which calculates the resource requirements for delivering content from any proxy to any cluster. This cost function can then be used to quantify the relationship between two or more content placement layouts. This cost function and its parameters are described below.

The cost of delivering an object from a proxy to a cluster is comprised of three costs: (i) Replication Cost $RC_{s,d,m}$, which is the cost required to replicate movie m from P_s to P_d, and is given in Eqn. (1), (ii) Storage Cost $SC_{d,m}$, which is the cost of storing movie m on P_d and is a function of the length of time storage is required for, as shown in Eqn. (2) and (iii) Delivery Cost $DC_{n,C}$, which is the cost of streaming content from P_n to all C clusters, shown in Eqn. (3). The total cost of serving all C clusters from P_n is given in Eqn. (4), where the replication cost is only included if movie m is not already available on P_n .

Table 1. Parameters to Cost Function

Symbol	Definition
P_i	Proxy i
$S(m)$	File size of movie m (bytes)
B_C	Bytes remaining to be served in cluster *c*
α_p	The cost of streaming of 1 byte per hop from proxy p
$\beta_{src,dst}$	The cost of bulk-delivery between P_{src} and P_{dst} per byte, per hop
$\delta_{src,dst}$	Distance between *src, dst* in terms of network hops
ϵ_p	Cost of storing 1 byte on P_p (per unit time)
T_C	The largest time remaining in the delivery of a stream to cluster C, i,e time when the last client within the set of Clusters will reach the end of the video stream.

Replication Cost $RC_{s,d,m}$, which is the cost required to replicate movie m from proxy P_s to P_d, and is calculated as:

$$RC_{s,d,m} = \beta_{s,d} \cdot \delta_{s,d} \cdot S(m) \tag{1}$$

Storage Cost $SC_{p.m,c}$, which is the cost of storing movie m on P_p and is a function of the length of time storage is required for (largest time remaining for a client within cluster *c*), as given by:

$$SC_{p,m,C} = \epsilon_p \cdot S(m) \cdot T_C \tag{2}$$

Delivery Cost $DC_{s,d,C}$, which is the cost of streaming content from P_p to all C clusters, shown in Eqn. (3). The total cost of serving the set of clusters C from P_p is:

$$DC_{s,d,C} = \sum_{c=1}^{C}(\alpha_s \cdot \delta_{s,d} \cdot B_c) \tag{3}$$

The replication cost, denoted $RC_{s,p,m}$, is only included if movie m is not already available on proxy P_n . Therefore, the total cost of using proxy (P_p) to serve all clients within the set of clusters C, all viewing movie m is given by the following equation:

$$Cost_{s,p,m,C} = [\beta_{s,p} \cdot \delta_{s,p} \cdot S(m)] + \epsilon_p \cdot S(m) \cdot T_C + \sum_{c=1}^{C} (\alpha_{p,c} \cdot \delta_{p,c} \cdot B_c) \tag{4}$$

where B_c is the total bytes remaining to be served to cluster c, and T_C is the largest time remaining in the playout of the client streams within cluster C.

4. PLACEMENT ALGORITHMS

The previous section provided the reader with a brief overview of how to quantify the resource usage associated with serving a group of clusters from a given proxy. The goal of the placement algorithms outlined in this paper is to determine the resource requirements of serving a set of clusters from a set of proxies and choosing the arrangement yielding the lowest resource requirements. The placement algorithm is executed on each proxy autonomously whenever some requirement is met, such as a new client has joined this proxy. The placement algorithm selects all clusters 'C' viewing the requested movie 'm' and calculates the resource requirements to delivery the requested movie to each of these clusters from each proxy within the set of proxies 'P'. In the event that a placement is found that results in less resource costs than the current placement, a placement re-shuffle occurs. To prevent oscillation occurring (content is redirected back and forth between a set of proxies), a change in content placement only occurs if the resource cost associated with the new placement is less than the resource cost of the current placement.

This section outlines a number of placement algorithms which vary in algorithm complexity and information required. These algorithms will be evaluated in section 5 to determine their suitability to a TV on-demand architecture.

VCDN Placement Algorithm

The VCDN placement algorithm considers *all* proxy-cluster combinations to determine optimal placement, where the number of clusters involved in the calculation is a function of the number of clusters viewing a given movie on the proxy performing the placement evaluation. Therefore the number of calculations 'n' performed during a placement evaluation is:

$$n = P^C$$

where P is the total number of proxies (active and idle) within VCDN, and C is the number clusters viewing the requested movie on the current proxy.

This does not scale very well as values for both P and C could be quite large. Therefore a number of heuristics have been proposed to reduce the number of calculations required while also maintaining close to optimal resource management.

Best_X Placement Algorithm

The motivation behind this approach is to initially rank each potential proxy based on the cost function defined in Eqn. 4, and then only consider the top ranked proxies during the placement process. Only this reduced set of proxies is considered when determining a suitable placement for a given proxy. This algorithm takes a parameter 'X' which determines how many of the top ranked proxies to consider for each cluster, i.e. if $X = 2$ the top 2 proxies will be considered, therefore the number of calculations 'n' performed during a placement evaluation is:

$$n = X^C$$

where X is the parameter to the algorithm and C is the number of clusters involved in a placement decision. Intuitively, the ability of this heuristic to determine a resource efficient placement depends on the size of this set of proxies, i.e. the value of 'X'.

Closest Proxy Algorithm

The goal of the closest proxy algorithm is to place a replica of a requested object onto the proxy which is closest to the requesting client. When a client joins the network it determines its *closest* (in terms of network hops) proxy, all subsequent requests for objects must first be replicated onto this proxy, and then delivered to the clients. This algorithm was chosen to determine the level of resource optimization that can be gained by using intelligent placement algorithms.

$$n = 0$$

Greedy Placement Algorithm (One-Time Placement)

The greedy placement algorithm is defined as a static or one-time placement algorithm. This placement algorithm uses future information (i.e provided with advanced knowledge of the clients access pattern) to provision the network and place content appropriately. The decision over where content should be located is carried out in a greedy manner, whereby a replica is initially placed on a server which can most efficiently serve the largest cluster. This process is then repeated for the next largest cluster until all cluster requests have been satisfied. Because this algorithm has prior knowledge of the clients access pattern, it can determine the minimum number of servers required to serve the clients at peak load, but due to its static placement, it cannot dynamically alter the content placement or number of servers after setup.

$$n = C \cdot P \; (carried\; out\; offline)$$

The process of replicating content varies between placement algorithms. In all cases except the one-time placement approach, the source of a replica is the closest copy of the desired content. Therefore over time, the origin will be chosen as the replica source on fewer occasions. Under the one-time placement approach, where all replicas are created before simulation begins, only the origin server can be selected as the source of a replica. This can have adverse effects on the performance of the greedy placement algorithm as will be seen when examining the effects of increasing the length of time a replica is allowed to remain idle on a proxy.

5. EVALUATION

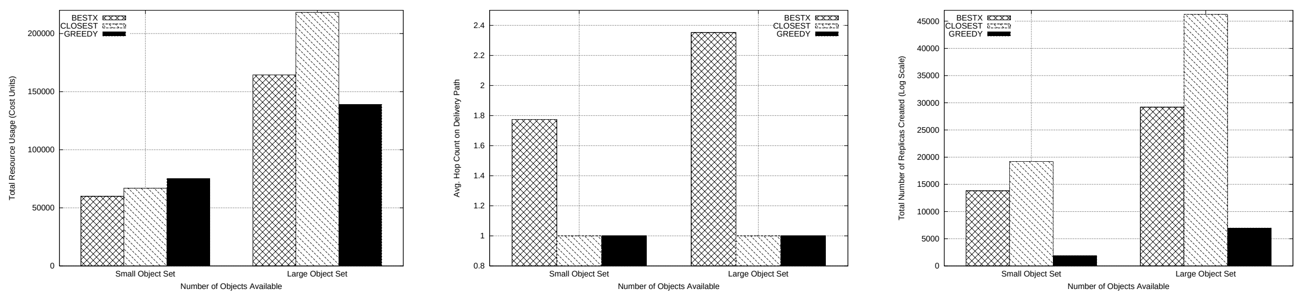
This section evaluates the proposed placement algorithms to determine which algorithm is most suited to a TVoD distribution architecture. The primary goal of a good placement algorithm is to be able to determine a placement strategy which can service a given set of client requests using the minimal amount of resources.

5.1. Simulation Environment

To determine the performance of the proposed placement algorithms, a number of simulations were carried out. Initial tests were carried out with the ns-2 simulator, but due to its unnecessary packet-level detail, ns-2 was too inefficient and cumbersome for the required experiments. Instead, a discrete event-driven simulator was developed in C++. The simulator is supplied with a number of parameter files such as a topology layout and a client workload configuration.

To evaluate these placement algorithms under a realistic environment, each test simulated a 24 hour period, with client arrival rates varying with according the the time of day (values obtained from Nielsen Media Research). The client viewing patterns are typical of what is expected, i.e high at peak times and very low at night time. For the purpose of these simulations, the client arrival rate was adjusted every 30 minutes to correspond with this viewing pattern. It would have been desirable to perform these simulations using actual viewing statistics provided by TV broadcasting companies, but this information was not available. The client arrival pattern was modeled as a Poisson distribution, with a parameter λ of 0.01(100 clients per minute as proposed by [18]). This represented the arrival rate during peak times. Though this is a very conservative figure, testing larger client arrival rates for a 24 hour period would not have been feasible. The distribution of client requests among the set of objects follows a Zipf distribution with parameter $\theta = 0.271$ as proposed by [19]. To accurately model the duration of client requests, a workload file was generated using the synthetic multimedia workload generator (MediSyn [20]).

The set of objects available for viewing by the clients was varied between 70 (objects stored for a $1\frac{1}{2}$days) and 400 (objects stored for 1 week), this will be used to determine how efficiently the algorithms can handle a wider selection of



(a) (b) (c)

Figure 2. Simulating 24 hrs of client requests using current market resource costs: (a) Total Resource Usage, (b) Avg. Delivery Path Length and (c) Avg. Total Number of Replicas Created.

client requests. Each object is 30 minutes in length, roughly the duration of a TV programme. Results were averaged over a number of simulation runs to ensure that a true representation of each algorithms performance was achieved. Test was carried out on both a small topology consisting of 30 proxies, and also on a larger 200 proxy topology. Due to space considerations, only results from the larger topology tests will be shown, though similar results were obtained under a smaller topology. A removal delay was attached to each proxy, the purpose of which is to remove any idle replicas after this time expires, as long as this is not the sole remaining replica of that object. This value was initially set at five minutes, though the effects of increasing this delay will be examined later in this section.

Finally, the costs assigned to each resource were initially based on the current market price for bandwidth and storage as given by [21] and [22], though the effects of varying these costs is examined. These costs are represented as a *cost per byte*. In the case of link usage, this cost is applied on a per hop basis, whereas in the case of storage, the cost is applied on a *per byte, per second* basis. It is worth noting that the cost per byte-hop for link usage is *four orders of magnitude* larger than the cost of storing 1 byte for a second. This outlines the industries bandwidth centric view of resource usage. These costs were then altered depending on the location of the resource within the network. Resources closer to the edge of the network are charged at a higher rate then those deeper in the network. This was to simulate value of service, such as the higher usage rate for resources maintained in large metropolitan areas, as opposed to those in sparse rural areas.

5.2. Results

The following tests examine the performance of the BestX, Closest and Greedy placement algorithms. The VCDN algorithm on the other-hand, could not be considered due to the time required to perform the required simulations. Previous simulations (not shown) indicate that the BestX algorithm where parameter $X = 3$, yields the same placement layouts as the VCDN algorithm. To ensure an unbiased comparison between placement algorithms, the results for the Greedy placement algorithm *include* the cost of creating the replicas.

Resource Management Capability using Current Market Resource Costs

The goal of the following experiments is to determine which algorithm results in the most efficient resource management, i.e uses the minimum resources while satisfying all client requests. Fig. 2 (a) compares the resource usage of each placement algorithm under a 200 proxy topology. It can clearly be seen that for a small set of objects, the BestX algorithm can satisfy all client requests using minimum resources, followed by the closest proxy algorithm and finally, the greedy placement algorithm. As expected, increasing the number of objects available for selection by the clients, results in increased resource usage, as the number of clients satisfied by each individual replica is reduced. Surprisingly though, the increase in resource usage by the greedy placement algorithm was significantly less than other placement algorithms, despite the increase in the client request distribution (client requests were distributed over 400 objects instead of 70). This can be attributed to the reduction in number of replicas required (Fig. 2 (c)) due to the static placement achieved under the greedy approach, as each replica created must remain for the duration of the simulation run. This reduction in number of replicas results in a lower overall resource usage brought about by a reduction in storage cost and replication cost.

Upon further examination of these results, it became clear that the length of time which a replica is permitted to remain on a proxy may influence the overall resource usage. Currently, both the BestX and closest proxy algorithms employ a replica removal delay of five minutes. This ensures that a replica does not remain idle for more than five minutes on a proxy, where it would be using up valuable storage space. Unfortunately, due to this small removal delay, many replicas are removed, which may again be requested in the near future. Before examining the effects of the removal delay, it is worth looking at the average length of the stream delivery path, as given by each placement algorithm. As expected, the closest proxy algorithm yields the shortest delivery path. The BestX algorithm results in a longer than required delivery path, as it tends to place replicas further back within the network where each replica can be of use to multiple clusters. A large delivery path is not a desirable property for streaming video applications, as larger delivery path lengths lead to greater variability in the overall QoS of the stream. Though the BestX algorithm does consider hop count during placement evaluation decisions, the addition of a weight to this parameter should reduce this effect.

Surprisingly, the static placement approach yields a similar delivery path to that achieved under the closest proxy approach. This is attributed to the low cost of storing a replica. Since the cost of storing a replica is significantly lower than the cost of streaming a replica (requires bandwidth, which is four orders of magnitude more expensive), this algorithm favors placing replicas close to the requesting clients. Also, the reason fewer replicas are created when using the static placement approach in comparison with the closest proxy approach, is due to the fact that static replicas remain for the full duration of the experiments, whereas replicas created under the closest proxy approach are removed if left idle for more than five minutes.

Replica Removal Delay

In the tests carried out thus far, the replica removal delay had been set at five minutes. Fig. 3 shows the effects of increasing the replica removal delay for each algorithm under a large topology network. From this graph it can be seen that increasing the length of time a replica is allowed to remain idle on a proxy reduces overall resource usage. This is akin to caching concepts, where content is left on a server in the hope that further requests will arrive in the future for that object. A number of interesting points to arise from this graph.

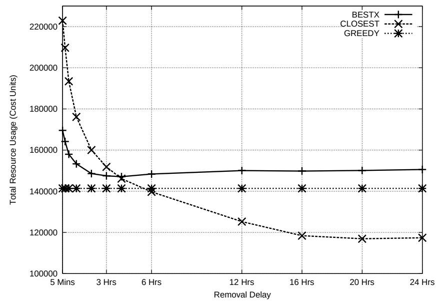


Figure 3. Effects of increasing the replica removal delay.

Firstly, the benefits of caching under the BestX algorithm diminish after the 4 hour mark, this corresponds exactly with the cost of replicating an object. This can also be shown mathematically, as being the time after which it would be cheaper to remove an idle replica and later replace it. For example, if a replica is idle for more than 4 hours (based on the current costs) it would have been more cost effective to remove the replica when it became idle, and make a replica when the clients later requested the file. Therefore, the earliest point at which maintaining an idle replica becomes more expensive than replicating in the future can be formulated as a tradeoff between storage and bandwidth costs:

$$T = \frac{\beta_{s,d} \cdot \delta_{s,d}}{\epsilon_p} \tag{5}$$

When the resource costs are plugged into Eqn. 5, and δ is set to *one hop*, four hours is the maximum removal duration, which corresponds to the results attained. Similar results are experienced for the closest proxy algorithm, except that the maximum removal delay is much higher (around the 20 hours mark) due to the higher number of hops (δ) on the replication path.

Secondly, it can also be seen that the cost benefit gained by increasing the replica removal delay was *insufficient* to make the BestX algorithm competitive with the static placement layout yielded by the greedy placement algorithm. To fully understand the reasons for this, it is necessary to examine how the BestX algorithm tends to place replicas. Typically, the algorithm refrains from placing replicas at edge nodes in the distribution network, but instead opts for placing replicas within the network to maximize the number of clusters which can be served. If the load from a cluster becomes large enough, then a dedicated replica would be placed closer to the cluster, but in general it is more resource efficient to strategically place a replica which can benefit multiple clusters. This has adverse effects if the removal delay is too large, as many idle replicas remain within the network. This case does not occur with the greedy placement or closest proxy algorithm, as replicas are typically created at edge nodes. Again, this can be seen by examining Fig. 2 (b)&(c) which show both the average length of the delivery path and the average number of replicas created by each placement algorithm.

Finally, one of the more surprising results obtained from this graph is that increasing the removal delay in the closest proxy algorithm beyond 5 hours results in improved performance over *all* other placement algorithms. There are two reasons for this. To simplify the explanation, a comparison of both the greedy and closest algorithms will be performed when both have a removal delay of 24 hours. At this point, though the removal delays are the same, the length of time that a replica remains within the network is not. The reason for this is because the static placement used in the greedy algorithm requires that all replicas be placed at time $t = 0$ (before simulation begins), whereas in the closest proxy algorithm, the replicas are only being made as the clients join the network, which can be any time within 24 hours, therefore the storage time for each replica under the greedy algorithm is 24 hours, whereas in the closest proxy algorithm this time ranges from 0-24 hours for each replica. The second reason for this saving stems from an implementation deficiency. Under the greedy based approach, each replica placed on a proxy is initially transferred from the origin proxy to the new replica position, whereas in all other placement schemes, replicas are created from the closest copy available. As a result of this, there is a higher cost associated with replicating in the greedy approach than in all other algorithms.

It should be noted, that the goal of the placement algorithm was to minimize resource usage, which when compared with the closest proxy algorithm in Fig. 2 (a) shows considerable improvement. Unfortunately, this is only the case when the algorithms are operating under certain conditions, most notably, when the replica removal delay is kept relatively low. In all cases, the BestX algorithm yields higher resource usage than a greedy (static placement) approach. It is believed that this is primarily due to the significant difference in resource costs, and will therefore be examined in the next section.

Resource Costs

The previous two sections have shown the impact that both client request concentration and the replica removal delay has on the performance of each placement algorithm. This section examines the costs used within these experiments and the impact of using alternative costs. For the following tests, the optimal replica removal delay was chosen based on the results obtained in the previous experiment, in the case of the BestX algorithm it was set at four hours, whereas for the closest proxy algorithm, this value was set at 20 hours.

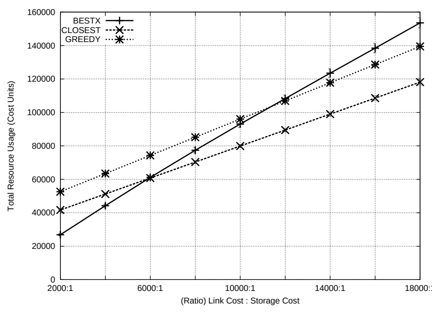


Figure 4. Tradeoff of link and storage costs under each placement algorithm.

The costs charged by service providers for resources can vary substantially, and are constantly evolving. Therefore, it is important for an operator considering the deployment of a resource conscious placement algorithm to be able to determine which placement approach most suits the current market situation. The goal of these simulations is to determine what cost ratio is suitable for each placement algorithm. Currently, link cost is in the region of 16000 larger than storage cost when broken down to a per byte per second cost. Fig. 4 shows the impact of changing this cost ratio.

As expected, if the cost of storing an object is comparable with the cost of streaming an object then an algorithm which considers both costs and dynamically adapts to changing client request patterns should perform well. But, if the cost of storing an object is negligible in comparison with link usage cost, the a static algorithm which determines a placement layout a priori based on the expected client load should perform optimally. When the ratio of link cost to storage cost is low (the difference between resource costs is low) then the dynamically adapting BestX algorithm performs optimally. But, as the ratio grows, indicating a larger gap between the costs of these resources, the performance benefits of the BestX algorithm diminish, in favor of replicating content closer to clients (reduce the length of the delivery path and hence overall link usage). Similarly, as the cost of link usage continues to rise, a one-time placement strategy which requires minimal replication of content and which has future knowledge of client requests, yields best results.

Hop Count

The previous tests have focused on the resource management aspects of each of these placement algorithms. Although this is the primary focus of this work, it is worth looking at the resultant delivery path length, as this is an important factor in the experienced QoS to the clients. The more hops on the delivery path, the more variability in link delay and congestion due to contention at the routers. Though, the cost function proposed in Eqn. 4 does consider the number of hops on the delivery path, it can be seen from Fig. 5 that in both a small and large topology, the delivery path length is considerably larger than that achievable by both the closest proxy and the greedy algorithm. Since the closest proxy algorithm always places content on the nearest proxy to the requesting client, the best that can be achieved from the BestX algorithm is to equal this performance, but it is evident that this is not the case. As expected, when the topology size increases, so too does the average delivery path length. The reason for this significant increase in delivery path is due to the average (total) number of replicas created under each scenario. Though not shown due to space reasons, it was determined that the BestX algorithm results in between 35%-40% fewer replicas being created, each of these replicas are placed deeper within the network where they can serve a greater number of clusters. The addition of a weight applied to the streaming cost in Eqn. 3 can reduce the length of the delivery path, but results in an increased number of replicas being created, and higher overall resource usage. Obviously, there is a trade-off between these two cost, which is the focus of our current work.

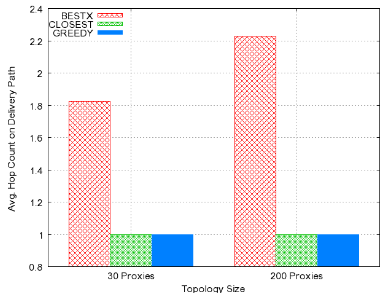


Figure 5. Length of delivery path.

6. CONCLUSIONS

This work set out to evaluate the performance of each of the placement algorithms to determine which is the most appropriate to use in shared infrastructure TVoD distribution network. This evaluation essentially compares the performance of three types of placement algorithms. Firstly, a static placement algorithm where content is distributed throughout the network using a greedy based algorithm with future knowledge of client requests. Under this approach, the content placement layout is determined before simulation began and the layout is not altered after the test begins. Secondly, an algorithm termed closest proxy algorithm was evaluated, the goal of which is to minimize latency (hops). Under this approach, all clients are served by their closest proxy. If a replica does not exist on the proxy, then one is created. Finally, a resource conscious algorithm is evaluated. This algorithm considers both link and storage usage when determining a suitable content placement layout.

The following is a list of the major findings:

- A static placement algorithm cannot efficiently manage resources in a changing environment such as TV distribution.
- Limited caching benefits can be achieved by leaving an idle replica on a proxy, though the benefits are far greater in an algorithm which places replicas on edge nodes, rather than deeper within the network.
- A placement algorithm which minimizes hop count yields better resource management in the current resource market, and this trend is likely to continue into the future, until the service providers make the resource costs more competitive.
- Although a resource conscious algorithm can adapt its resource usage to accommodate changes in client request patterns, in its current form it does not put enough emphasis on reducing the length of the delivery path. This can be overcome with the addition of a weight to the stream delivery cost.

A final influencing factor, which was not included in this paper due to space constraints, is that of client request concentration. This is closely linked with the formation of client clusters. Recall, that a cluster is a group of clients, all requesting the same object from a similar region of the network. If these regions are small, then for a given number of clients, there is likely to be more clusters, and as such an increased number of locations where replicas will be required. In such a situation, a static placement or closest proxy approach may be adversely affected, as more replicas will be required for a given number of clients. This does not impact the BestX algorithm as much, as replicas are placed in such a manner as to maximize the number of clients (clusters) which it can serve.

Future work will examine the effect of using a coarse grained cost model, where resources cannot be purchased on a *per byte* scale.

7. ACKNOWLEDGMENTS

The authors wish to thank AT&T Labs Research (USA) for part-sponsoring this project. In addition, support from the EU FP6 E-NEXT Network of Excellence contributed greatly to this work, by enabling close interaction with other E-NEXT partners working on related research. The authors would also like to thank the Boole Centre for Research and Informatics for use of their cluster computer. The authors would also like to thank Dr. Utz Roedig for interesting discussions and feedback.

REFERENCES

1. A. J. Cahill and C. J. Sreenan, "An efficient cdn placement algorithm for high-quality tv content," in *Proc. from Internet and Multimedia Systems and Applications - EuroIMSA*, (Grindelwald, Switzerland), February 2005.
2. A. J. Cahill and C. J. Sreenan, "An efficient resource management system for a streaming media distribution network," *Interactive Technology & Smart Education (ITSE)* **3**(1), pp. 31–44, 2006.
3. P. B. Mirchandani and R. L. Francis, *Discrete Location Theory*, ch. The Uncapacitated Facility Location Problem, pp. 120–168. John Wiley & Son Inc, New York, USA, 1989.
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. X. Jia, D. Li, H. Du, and J. Cao, "On optimal replication of data object at hierarchical and transparent web proxies," *IEEE Trans. Parallel Distrib. Syst.* **16**(8), pp. 673–685, 2005.
6. L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *Proc. from IEEE INFOCOM*, pp. 1587–1596, (Anchorage, Alaska), April 2001.
7. T. V. Nguyen, C. T. Chou, and P. Boustead, "Provisioning content distribution networks over shared infrastructure," in *Proc. from 11th IEEE Internation Conference On Networks (ICON)*, (Sydney, Australia), Sept 2003.
8. D. Choon-Hoong, S. Nutanong, and R. Buyya, *Peer-to-Peer Computing: Evolution of a Disruptive Technology*, ch. Peer-to-Peer Networks for Content Sharing, pp. 28–65. Idea Group Publisher, Hershey, PA, USA, 2005.
9. M. M. Hefeeda, B. K. Bhargava, and D. K. Y. Yau, "A hybrid architecture for cost-effective on-demand media streaming," *Comput. Networks* **44**(3), pp. 353–382, 2004.
10. D. Xu, S. Kulkarni, C. Rosenberg, and H.-K. Chai, "Analysis of a cdn-p2p hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems Journal* **11**, pp. 383–399, April 2006.
11. K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proc. from ACM SIGCOMM*, pp. 89–100, (Cannes, France), 1997.
12. A. Hu, "Video-on-demand broadcasting protocols: a comprehensive study," in *INFOCOM*, pp. 508–517, 2001.
13. A. Dan and D. Sitaram, "A generalized interval caching policy for mixed interactive and long video environments," in *Proc. from IS & T SPIE Multimedia Computing and Networking Conference*, (San Jose, CA), January 1996.
14. D. Eager, M. Vernon, and J. Zahorjan, "Bandwidth skimming: A technique for cost-effective video-on-demand," in *Proc. IS&T/SPIE Conf. on Multimedia Computing and Networking*, (San Jose, CA, USA), 2000.
15. K. Hue, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proc. of 6th ACM International Multimedia Conference*, ACM Multimedia '98, (Bristol, UK), September 1998.
16. C. Griwodz, "The use of stream merging mechanisms in a hierarchical cdn," in *Proc. from IS&TSPIE Multimedia Computing and Networking (MMCN)*, (Santa Clara, California), January 2004.
17. L. Amini, *Models and Algorithms for Resource Management in Distributed Computing Cooperatives*. PhD thesis, "Columbia University", 2004.
18. B. Wang, S. Sen, M. Adler, and D. Towsley, ""optimal proxy cache allocation for efficient streaming media distribution"," *IEEE Transaction on Multimedia, Special Issue on Streaming Media* **6**, April 2004.
19. C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On optimal batching policies for video-on-demand storage server," in *Proc. from the International Conference on Multimedia Computing and Systems*, pp. 253–258, (Hiroshima, Japan), 1996.
20. W. Tang, Y. Fu, L. Cherkasova, 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*, NOSSDAV, (Monterey, California, USA), June 2003.
21. HEAnet, "Heanet," 2006. http://www.heanet.ie.
22. IBackup, "Ibackup," 2006. http://www.ibackup.com.