Content Delivery using Replicated Digital Fountains

Niklas Carlsson
University of Calgary
Calgary, CA, Canada
ncarlsso@cpsc.ucalgary.ca

Derek L. Eager
University of Saskatchewan
Saskatoon, SK, Canada
eager@cs.usask.ca

***Abstract*—With a majority of Internet traffic being predicted to be caused by content delivery, it is clear that content delivery applications will consume much of the resources on the Internet. This paper considers the problem of cost-efficient content delivery, in which the application incurs both a network delivery cost (e.g., from cross ISP traffic or, more generally, operation/energy costs at Internet routers) and costs at the servers (e.g., due to cost of ownership, energy, or disk bandwidth). While the cost objective and the absolute cost tradeoff may be different from case to case, we argue that an architecture with distributed servers, each using digital fountain delivery, may be an attractive candidate architecture when considering the total content delivery cost. Within the context of a simple system model, we then determine optimal server selection policies for such an architecture, and derive analytic expressions for their associated delivery costs. A readily-implementable heuristic policy is proposed that is found to achieve within 10% of the minimal cost. Finally, we show how our results for content download can also be applied to streaming video delivery.**

I. Introduction

Content delivery applications consume a large fraction of the total Internet bandwidth [1], both at the servers and throughout the network. The problem of cost-efficient content delivery, in which the application endures both a network delivery cost and costs at the servers, is therefore becoming increasingly important. Network delivery costs (e.g., from cross ISP traffic or, more generally, operation/energy costs at Internet routers) have traditionally been reduced through *replication* techniques (e.g., [2]–[5]), in which multiple servers (often geographically distributed as in a CDN, for example) share the load of processing client requests and may enable delivery from a nearby server. To reduce costs at the servers (e.g., due to cost of ownership, energy, or disk bandwidth) service *aggregation* techniques (e.g., [6]–[10]) have been proposed that allow multiple client requests to be served together more efficiently than if served individually.

Although service aggregation can yield high efficiency on its own, a potential drawback of serving requests for particular content at a common server is the cost of delivering the content to a client far away from this server. In general, the content delivery solution that minimizes the cost may entail use of both replication of the content at geographically distributed servers, with each of these servers using some service aggregation technique. This paper focuses on server-based approaches that use both replication and service aggregation. In particular, we advocate a content delivery system with multiple servers, each using digital fountain delivery. With digital fountains typically erasure coded [6], [8], [11] content is cyclically delivered whenever there is at least one client that has requested the file and has not yet received all blocks.

In systems with multiple servers, each using digital fountain delivery, a policy is needed for selecting the server(s) from which each client receives service. Note that there is a tradeoff between locality of service, and efficiency of service aggregation. In particular, the network cost of serving a request is minimized by a policy that selects the nearest server, while server cost is minimized by directing all closely-spaced requests for the same content to a common server. An optimal policy for any particular system might be one of these extremes, or an intermediate policy, depending on workload parameters as well as on the precise characteristics of the network and server costs associated with the system under consideration.

In this paper, we formulate a simple system model that enables analytic comparison of server selection policies that are optimal within their respective class, as well as determination of a tight lower bound on content delivery cost. Our results provide insights both with regards to the cost tradeoffs and the performance that are achievable with different policy classes and what policy complexity is needed to achieve close to minimum delivery cost. Based on the insights from the operation of the optimal (lower bound) policy, a readily-implementable policy is proposed that is found to achieve within 10% of this lower bound. Finally, we show how our results for content download can also be applied to streaming video delivery, through use of the Reliable Periodic Broadcast (RPB) protocols [12]. These protocols divide the video file into segments of increasing lengths, each of which is delivered using a digital fountain. In systems in which the segments are replicated at multiple servers, a server selection decision is needed for each segment.

The remainder of the paper is organized as follows. Sections II and III present a case for using distributed fountains and our system model, respectively. Section IV presents an optimal (tight) lower bound policy that achieves the minimum possible delivery cost and compares this lower bound to the costs with three simple baseline policies. Motivated by the cost differences between the baseline policies and the lower bound, improved server selection policies are described and evaluated in Section V. Section VI applies our results to streaming video delivery. Related work is discussed in Section VII. Finally, conclusions are presented in Section VIII.

II. The Case for Distributed Digital Fountains

We are interested in the scenario where clients are to receive service without being subjected to any additional batching delay. For this case, the most efficient service aggregation technique in the absence of packet loss uses cyclic delivery. Consider a large file stored on disk, to be delivered over a network supporting only unicast communication, by a single server. With cyclic delivery, while there is at least one client that has requested the file and has not yet received all blocks, the server retrieves the next file block (wrapping around if at the end) from disk, and sends it to all such clients. In this manner, server costs such as disk accesses can be amortized over many requests. If multicast (IP or application level) is available and employed this approach reduces network costs as well.

The digital fountain technique is similar in that whenever a block is transmitted, it is sent to all clients that have requested the file, but with this technique the file is erasure coded [6], [8], [11]. Erasure coding allows clients to recreate the original content after having received as many (or slightly more, depending on the erasure-coding technique) unique blocks as there are blocks in the uncoded file. Thus, when packet loss occurs, the respective client does not need to wait until the server cycles around and transmits the lost packet(s) again.

For simplicity, in the remainder of this paper, we use the term "digital fountain" to refer to either true digital fountain, or to cyclic delivery of the uncoded file, with the choice among these techniques dependent on the packet loss characteristics.

In addition to replication and service aggregation, we note that peer-to-peer (e.g., [13]) techniques also provide a scalable distribution solution. However, overall, server-based approaches may enable greater control of the content, and allow the use of service aggregation techniques for hot content, as in the digital fountain technique, which can dramatically reduce the total work required for servicing a request stream. (Note that peer-to-peer techniques transfer work from servers to peers, but may not reduce the total amount of work performed [14].) Should reducing the total work become increasingly important (e.g., due to energy consumption considerations) we expect that server-based solutions will become increasingly attractive compared to peer-to-peer solutions [15], at least for the hot content for which service aggregation is applicable.

Finally, with much recent attention being on sustainability and energy efficiency, we note that a major issue in content delivery is the energy cost of the required server resources. With the emergence of "green" energy and "green" data centers (e.g., zero-carbon data centers) located close to (or at the site of) renewable energy sources, however, server-based approaches have the advantage that they potentially can use "greener" energy than that consumed by the peers in peer-to-peer approaches. Further, there is considerable current work on making power usage in servers and data centers more "proportional" [16], [17]. For these reasons, as the totality and the form of energy usage become more important, we expect that server-based solutions will become increasingly attractive.

TABLE I: Notation

Symbol	Definition
λ_i	File request rate from the clients of group i; groups indexed so that $\lambda_i \geq \lambda_j$, for $i \geq j$
λ	Total request rate, summed over all client groups
L	Amount of data that a client needs to receive to successfully reconstruct the requested file
M	Number of client groups
N	Number of servers
C_i	Server i file delivery cost
C	Total file delivery cost
c	Remote access cost per unit of data received for all client groups i and servers j, $i \neq j$
g_i	Average fraction of the file data that a group i client receives from a remote server
b	Fountain data rate

III. System Model

We use here a simple model that allows us to focus on the impact of the server selection policy, and in particular on the tradeoff between choosing the nearest server or a common server at which multiple requests may be served concurrently.

We consider delivery of a single representative file that is replicated at N homogeneous servers[1], each of which uses digital fountain delivery. Clients are divided according to network location into M groups, such that all of the clients in a group can be considered to be at approximately the same network distance from each server. For simplicity, in the following it is assumed that $M = N$. Given this assumption, the client groups and servers are indexed such that server i is the nearest (the "local") server for group i clients. Each server j for $j \neq i$ is called a "remote" server for this group.

Requests for the representative file from the clients of each group i are assumed to be Poisson at rate λ_i, with the servers/groups indexed from 1 to N in non-increasing order of their request rates. Poisson arrivals can be expected for independent requests from large numbers of clients, but have also been observed in measurement studies of content delivery systems with smaller numbers of clients (e.g., [20]). Note that request burstiness is beneficial in systems using service aggregation; so, for a fixed average request rate, the delivery cost in systems with burstier request arrival processes typically is lower than predicted by models assuming Poisson requests.

Denoting the fountain data rate by b, and assuming, for simplicity, that clients need to receive a fixed amount of data L to reconstruct the file, the file download time is equal to L/b. The system metric of interest is the total file delivery cost required to achieve this client performance. In the simple model we use here, the costs are assumed to consist of only the following three kinds:

- System costs that do not depend on the server selection policy (for example, are fixed, or are dependent only on

[1]The model could be easily extended to reflect differences among the servers with respect to relative "cost", as might result from pricing differences [18], [19] or carbon cost.

the client request rates and file size), and can therefore be neglected in our model.

- Server costs (associated with CPU, disk, and other components) that increase linearly with the time a digital fountain is active, and the data rate of the fountain, independently of how many clients are receiving the fountain transmissions.
- Network costs that represent the increased cost associated with receiving data from a remote server, assumed linearly proportional to the amount of such data.

Given this assumption, the total file delivery cost C is defined as the total average rate at which cost is incurred at the servers, plus the total average rate at which *remote access cost* is incurred, neglecting any cost component that does not depend on the server selection policy. The average rate at which cost is incurred at server i (C_i) is given (using normalized units) by the fountain data rate b times the proportion of time the digital fountain at server i is active. The remote access cost incurred when a client from group i receives a fraction g of its data from a server j is given by $c_{ij}gL$, where the constant c_{ij} gives the network cost per unit of data received when server j delivers data to a client from group i. Since we model only the increased cost associated with receiving data from a remote server, $c_{ii} = 0$. For simplicity, in the following it is assumed that the c_{ij} for all $i \neq j$ are identical, equal to some constant c. It is also assumed that $0 < c < 1$; if $c \geq 1$, then it is always optimal to simply choose the local server for each request.

Using the notation defined in Table I, this yields

$$C = \sum_{i=1}^{N} C_i + (\sum_{i=1}^{N} \lambda_i g_i) cL, \tag{1}$$

where the first term corresponds to the policy-dependent server costs, and the second term corresponds to the policy-dependent network costs.

While the precise characterization of the various network and server components of the total cost is complicated by many factors (including the sharing of components with other applications, splitting of costs among multiple organizations, and different energy sources being more or less environmentally friendly and/or costly), we believe that the above model captures most first-order effects. Furthermore, we note that the model has been found to capture the performance and cost tradeoffs among different server selection policy classes observed using more detailed network topology models [21].

IV. Baseline Policies and Lower Bound

A. An Optimal Tight Lower Bound Policy

In the context of the model of Section III, an optimal server selection policy (i.e., one that achieves the minimum possible file delivery cost) can be determined if complete knowledge of the system state is assumed, as well as negligible overhead for activating/deactivating fountains or switching service from one server to another. Analysis of the delivery cost with such a policy then yields a lower bound on the delivery cost that would be possible with a realistic policy.

Consider some arbitrary point in time t, and denote the number of group i clients that have requested the file but not yet fully received it at time t by n_i. If all of the n_i are zero, no fountain is active at time t; otherwise, there must be at least one server with an active fountain. Suppose that the fountain at some server j is active, and that $n_i > 0$ for some $i \neq j$. Cost is minimized if the group i clients receive service locally (requiring the fountain at server i to be active), rather than remotely (at server j, for example), if and only if $n_i > 1/c$.

Based on the above necessary conditions of an optimal policy, it can be shown that the following policy achieves the minimum file delivery cost. At each time t, and for each i, the fountain at server i is active if and only if either $n_i > 1/c$, or $n_i = \max_j n_j \geq 1$ and there is no $k < i$ such that $n_k = \max_j n_j$. Group i clients receive service from server i if its fountain is active, and otherwise receive service from any remote server with an active fountain.

An optimality proof easily follows from the fact that the remote access cost incurred by serving group i clients remotely is greater than the cost of serving these requests locally (by activating the local fountain) whenever $n_i > 1/c$, and less than the cost of serving these requests locally whenever $n_i < 1/c$.

The total file delivery cost with the above policy, and thus a tight lower bound on the delivery cost achievable with any server selection policy, is given by

$$b \sum_{i=1}^{N} \Bigg[(1 - p(n_i \leq 1/c)) + \sum_{k=1}^{\lfloor 1/c \rfloor} p(n_i = k) \Bigg[kc + (1 - kc) \prod_{j=1}^{i-1} p(n_j < k) \prod_{j=i+1}^{N} p(n_j \leq k) \Bigg] \Bigg], \tag{2}$$

where $p(n_i = m) = \frac{(\lambda_i L/b)^m}{m!} e^{-\lambda_i L/b}$. Here the first term within the outer sum gives the server cost of the fountain at server i being active, owing to there being more than $1/c$ active group i clients. The first term within the inner big brackets gives the remote access cost incurred by group i clients receiving service remotely, when there are no more than $1/c$ active group i clients. The second term within the inner big brackets accounts for the case in which there are no more than $1/c$ active group i clients, and yet the fountain at server i is nonetheless active (implying that server cost is incurred at server i, but no remote access cost owing to the active group i clients).

B. Baseline Policies

We now consider three very simple baseline policies, and compare the file delivery costs with these policies to that of the lower bound given by expression (2).

- **Individual delivery at local server:** Perhaps the simplest policy is to serve each request individually at the closest server. Since there is no remote access with this policy and each client receives data individually at rate b for duration L/b from its local server, the total file delivery cost is given by λL.
- **Fountain delivery at local server:** Similarly to the previous policy, each request is served at the local server,

but now the servers use digital fountain delivery. Since the download time for each client is L/b, the probability of there being at least one active group i client (and therefore of the fountain at server i being active) is $1 - e^{-\lambda_i L/b}$, and the total file delivery cost is given by $b(N - \sum_{i=1}^{N} e^{-\lambda_i L/b})$.

- **Fountain delivery at single server:** An approach that minimizes the server cost, at the expense of potentially high remote access cost, is to use a single server for all requests. It is most efficient to pick the server whose local client group has the highest request rate, which is server 1 with our indexing. This policy has a total file delivery cost of $b(1 - e^{-\lambda L/b}) + (\lambda - \lambda_1)cL$.

C. Performance Comparisons

In the following comparisons, without loss of generality the unit of cost is chosen to be the server delivery cost for a fountain delivering data at rate b for time L/b (the client download time), and the unit of time is chosen to be L/b. With these choices of units, $b = 1$ and $L = 1$. A request rate of 2 with these units, for example, implies that there are two requests for the file in the time it takes to download the file. We will often show costs divided by the total request rate, giving the average cost per request. If the total server delivery cost per request, with our chosen units, is 0.5, for example, then on average each active fountain is serving 2 clients concurrently.

Figures 1(a) and (b) show the average server cost and remote access cost per request, respectively, as functions of the file request rate from each client group, for the baseline policies and the lower bound policy. These figures use our default parameter settings of N = 16 servers (and client groups), a remote access cost per unit of data received of c = 0.5, and an equal request rate from all client groups ($\alpha = 0$, where α is the Zipf parameter that we will use for cases of heterogeneous request rates). Comparing fountain delivery at the local server to individual delivery at the local server, note that fountain delivery begins to yield appreciable cost benefits at a request rate from each client group of about 0.1. With the lower bound policy, clients receive service from their local server when the request rate is either very low or very high; at intermediate request rates, some fraction of clients receive service at a remote server so as to make better use of service aggregation.

Figure 1(c) shows the percentage increase in the total file delivery cost with each of the baseline policies, in comparison to with the lower bound policy, as functions of the request rate from each client group. (Note that this corresponds to comparing the relative differences of the combined costs shown in Figures 1(a) and 1(b).) Note that both individual delivery at the local server, and fountain delivery at a single server, become highly inefficient for high request rates, owing to high server costs and high remote access costs, respectively. Fountain delivery at the local server is efficient for both low and high request rates, but is significantly sub-optimal for intermediate request rates.

Figures 2(a), (b), and (c) show the impact of alternative parameter settings. Figures 2(a) and (b) show results for scenarios with an equal request rate from all client groups ($\lambda_i = 1$), and varying remote access cost and number of servers (and client groups), respectively. Figure 2(c) shows results for scenarios in which the client group request rates are Zipf distributed, with the request rate λ_i from client group i equal to Ω/i^{α}, where the normalization constant $\Omega = N/\sum_{j=1}^{N} \frac{1}{j^{\alpha}}$. Note that with $\alpha = 0$, all client groups have the same request rate; for $\alpha = 4$, in contrast, 92% of the requests are from client group 1.

As shown in Figures 1 and 2, for each of the baseline policies there are substantial regions of the parameter space in which the policy yields a delivery cost much greater than optimal. Although the lower bound policy can achieve substantially lower cost than the baseline policies, it is also substantially more complex, requiring full knowledge of how many clients are active from each client group at each point in time, and re-evaluation of server selection decisions whenever the population of active clients changes. Motivated by these results, the following section proposes improved policies and evaluates their associated file delivery costs.

V. Improved Policies

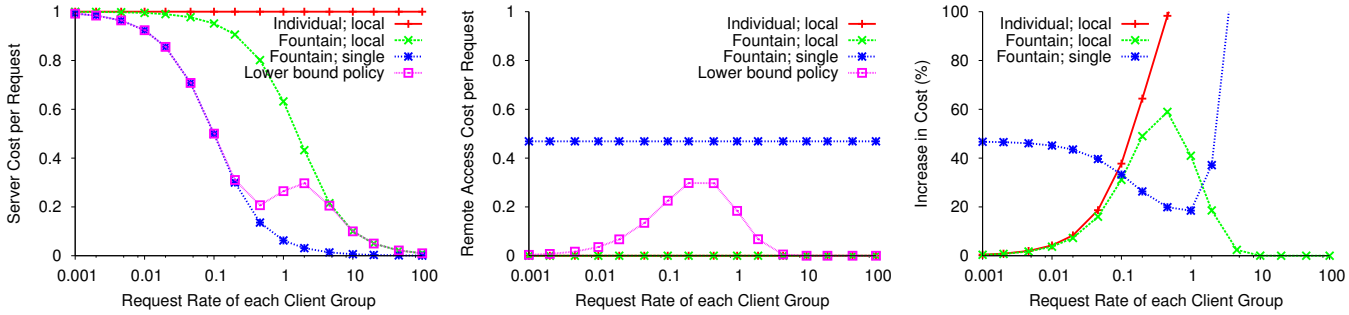
In this section we consider three policies for which the associated delivery costs are expected to be lower than with the baseline policies, and that also have substantially lower overheads than the lower bound policy.

- **Optimal static policy:** Only knowledge of the client group request rates λ_i (together with basic system parameters such as the number N of client groups and servers, and the remote access cost per unit of data c) are used to determine the server that will serve the requests from each client group. The optimal static policy achieves the minimum cost among all policies that make server selection decisions independent of the system state.
- **Optimal at-arrival policy:** Immediately when a request is made, it is decided which server the respective client will receive service from, at each point during the client's download time of L/b. The optimal at-arrival policy achieves the minimum cost among all such "at-arrival" policies. Note that this policy uses knowledge of the current global system state, as well as the request rates.

Owing to the information it requires for its server selection decisions, the optimal at-arrival policy may be infeasible. It is also inflexible once server selection decisions have been made. A decision to use a remote server, for example, can not be changed if there is a sudden burst of requests from the same client group that would warrant activating the fountain at the local server.

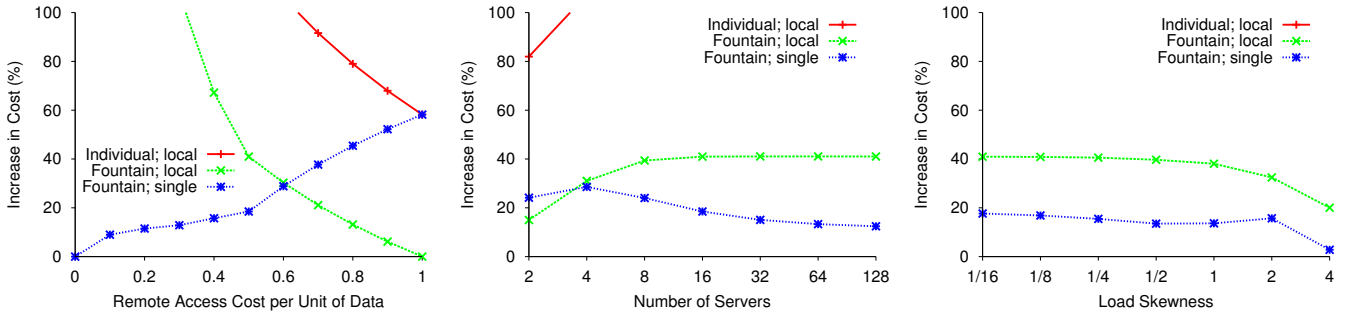
- **Dynamic policy:** The dynamic policy that we consider allows server selection decisions to be changed dynamically, as in the lower bound policy, but much less frequently, and using considerably less state information.

The first two policies delimit the performance achievable with their respective classes of policies. The dynamic policy attempts to achieve the benefits of the more flexible approach



(a) Server cost (b) Remote access cost (c) Relative difference

Fig. 1: Performance of baseline policies ($c = 0.5, N = 16, \alpha = 0$).



(a) Remote access cost per data unit (c) (b) Number of servers (N) (c) Load skewness (α)

Fig. 2: Impact of system parameters on baseline policies (default parameters: $c = 0.5, N = 16, \lambda = N, \alpha = 0$).

to server selection used in the lower bound policy, while reducing the state information needed.

The following three subsections describe in detail each of the above policies, and present analyses of the first two. Cost comparisons are provided in Section V-D.

A. Optimal Static Policy

Since static policies do not use information concerning the system state, there is no advantage in such policies to switching servers during a client's download time. Thus, in the optimal static policy, each request is served at a single server. Further, in a static policy, either all requests from a given client group are served by the same server, or server selection is probabilistic. With Poisson requests from each client group, request arrivals at each server will also be Poisson, and therefore (with fountain delivery) the server cost C_i is a monotonically increasing, concave function of the request arrival rate at server i.

Since we assume $c_{ij} = c$ for all $i \neq j$, the concave increasing form of C_i implies that it is optimal to serve all requests that receive remote service at the server with the highest rate of requests from its local client group (i.e., server 1, given our assumed indexing). Also, it is optimal to serve either all requests from a client group at server 1, or none, and the former case can hold only if all requests from client groups with equal or lower request rate are also served remotely.

Based on the above, in our optimal static policy there is an index $k (1 \leq k \leq N)$, such that all requests from group i clients, for $i \leq k$, are served at the local server, while all requests from group j clients, for $j > k$, are served at server 1. The total file delivery cost with this policy is given by:

$$\min_{k=1,2,\ldots,N} \Bigg[b\left(1 - e^{-(\lambda_1 + \sum_{i=k+1}^{N} \lambda_i) L/b}\right) + \sum_{i=2}^{k} b\left(1 - e^{-\lambda L/b}\right) + c \sum_{i=k+1}^{N} \lambda_i L \Bigg]. \tag{3}$$

It is interesting to note that the optimal static policy has some similarities with the lower bound policy. In particular, the server whose local client group has the highest request rate (optimal static policy) or highest number of currently active clients (lower bound policy) serves (at least) its local clients, as does any server whose local client group has a request rate (optimal static policy) or number of currently active clients (lower bound policy) greater than a threshold value.

B. Optimal At-arrival Policy

With an at-arrival server selection policy, at the time of a client's request it must be determined which server the client will receive data from at each point during the client's download time. Denoting the time since the previous request arrival by t_a, it is necessary to determine: (1) which server's fountain should be scheduled to be active for the last $\min[t_a, L/b]$ of the download time, and (2) for each time instant at which the fountain of the local server is not scheduled during the initial $\max[0, L/b - t_a]$ of the download time, whether this fountain

should now be scheduled for that time, or whether the new request should receive service from a remote server whose fountain is already scheduled at that time (such a server must exist owing to the previous request).

1) Homogeneous Case: Consider first the case in which all client groups have the same request rate λ/N. In this case, it is clearly optimal to schedule the fountain of the local server for the portion of the download time during which no server's fountain is already scheduled (the last $\min[t_a, L/b]$). For each time offset t at which the local server's fountain is not scheduled, within the initial $\max[0, L/b - t_a]$ of the download time, it is optimal to schedule the fountain of the local server, rather than to have the new request receive data from a (remote) server whose fountain is already scheduled, if and only if $b \leq bc(1+(\lambda/N)t)$, or equivalently $t \geq N(1-c)/(c\lambda)$. Denoting the threshold value $\min[N(1-c)/(c\lambda), L/b]$ by T, the above observations yield the following total file delivery cost with the optimal at-arrival policy:

$$b\Bigg[N(1-e^{-(\lambda/N)(L/b-T)}) + (e^{-(\lambda/N)(L/b-T)} - e^{-\lambda(L/b-T)})c\lambda T$$

$$+e^{-\lambda(L/b-t)}((1-e^{-\lambda T}) + c(\lambda T - (1-e^{-\lambda T}))\frac{N-1}{N})\Bigg]. \quad (4)$$

This expression is derived as follows. Consider the state of the system at an arbitrary point in time under the operation of the optimal at-arrival policy. If there was at least one request from client group i at an offset prior to the current time in the interval $[-L/b, -T]$, the fountain at server i will currently be active, yielding the first term within the outer parentheses. If there were no requests from group i but at least one request from some other group j in $[-L/b, -T]$ (and thus the fountain at server j is currently active), any group i requests that were made in $[-T, 0]$ will currently be receiving service from a remote server, yielding the second term within the outer parentheses. Finally, if there were no requests from any group in $[-L/b, -T]$, one server's fountain will currently be active if and only if at least one request arrived in the interval $[-T, 0]$, and all requests that arrived in the interval $[-T, 0]$ from client groups other than that from which the first such request arrived, will currently be receiving service from a remote server.

2) Heterogeneous Case: For the general case in which client groups may have differing request rates, the optimal choice between receiving service from a remote server whose fountain is already scheduled, or scheduling the local server's fountain, is determined according to the time offset t from the beginning of the download time as in the case of homogeneous client groups. For $t \geq T_i = \min[(1-c)/(c\lambda_i), L/b]$, it is optimal to schedule the local server's fountain; otherwise, it is optimal to receive service from the remote server.

Unlike in the case of equal request rates, for new requests from other than group 1 (that with the highest request rate) it may not be optimal to schedule the local server's fountain for the portion of the client's download time during which no server's fountain is already scheduled. Consider a newly-arriving request from other than group 1, and a time offset $t > T_1$ from the beginning of the download time, at which no server's fountain is already scheduled. In this case, it is optimal to schedule the local server's fountain to be active at time t if and only if $b(1-e^{-\lambda_1(t-T_1)}) + e^{-\lambda_1(t-T_1)}bc\lambda_1 T_1 \leq bc(1+\lambda_i t)$; otherwise, server 1's fountain should be scheduled instead. The left-hand side of this relation corresponds to the expected cost associated with scheduling the local server's fountain for time t. This cost takes into account the possibility that even if the local server's fountain is scheduled to be active at time t, server 1's fountain may at the time of a subsequent group 1 request arrival also be scheduled to be active at time t. Consider now $t \leq T_1$. It is optimal to schedule the local server's fountain to be active at time t if and only if $bc\lambda_1 t \leq bc(1+\lambda_i t)$; otherwise, it is optimal to schedule server 1's fountain instead. Combining these two cases yields the condition

$$(1-e^{\lambda_1(t-\min[t,T_1])}) + e^{-\lambda_1(t-\min[t,T_1])}c\lambda_1 \min[t, T_1] \leq c(1+\lambda_i t). \quad (5)$$

It is straightforward to verify that this condition divides the interval $[0, L/b]$ into (at most) three regions: an initial region in which the condition holds, a second region that may or may not be present and in which the condition does not hold, and (should the second region be present) a third region that may or may not be present in which the condition again holds. Define T_i' and T_i'' to be the boundary values separating the regions, should all three exist, with $T_i'' < T_i'$; note that we must have $T_i' < T_i$ in this case. If just the first two regions exist, we must have $T_i = L/b$. In this case define T_i'' as the boundary value separating these regions and define $T_i' = T_i$ $(= L/b)$. Finally, if only the first region exists, define $T_i'' = T_i' = T_i$. Then it is optimal to schedule the local server's fountain to be active at time t for $t \leq T_i''$ and for $t \geq T_i'$, and server 1's fountain for $T_i'' < t < T_i'$.

Defining f_i as the fraction of requests that are from client group i, analysis of the above optimal at-arrival policy yields the following expression for the total file delivery cost, for the general case of heterogeneous client group request rates:

$$\begin{aligned}
&b\textstyle\sum_{i=2}^{N}\{(1-e^{-\lambda_i(L/b-T_i)}) + (e^{-\lambda_i(L/b-T_i)} - e^{-\lambda_i(L/b-T_i)})c\lambda_i T_i \\
&\quad + e^{-\lambda(L/b-T_i)}(1-f_i)c(\lambda T_i - (1-e^{\lambda T_i}))f_i \\
&\quad + ((e^{-\lambda(L/b-T_i)} - e^{-\lambda(L/b-T_i')}) + (e^{-\lambda(L/b-T_i'')} - e^{-\lambda L/b}))f_i \\
&\quad + e^{-\lambda(L/b-T_i')}f_i c(\lambda_i(T_i' - T_i'') + (1-e^{\lambda(T_i'-T_i'')})(1-f_i)) \\
&\quad + (e^{-\lambda(L/b-T_i')} - e^{-\lambda(L/b-T_i'')})f_i c\lambda_i T_i'' \\
&\quad + e^{-\lambda(L/b-T_i')}e^{\lambda_1(T_i'-T_1)}(1-e^{(\lambda-\lambda_1)(T_i'-\max[T_1,T_i''])})\tfrac{\lambda_i(1-c\lambda_1 T_1)}{\lambda-\lambda_1} \\
&\quad + (e^{-\lambda(L/b-\max[T_1-T_i''])} - e^{-\lambda(L/b-T_i')})f_i(1-c\Omega)\} \\
&+b\{(1-e^{-\lambda_1(L/b-T_1)}) + (e^{-\lambda_1(L/b-T_1)} - e^{-\lambda(L/b-T_1)})c\lambda_1 T_1 \\
&\quad + (e^{-\lambda(L/b-T_1)} - e^{-\lambda L/b})f_1 \\
&\quad + e^{-\lambda(L/b-T_1)}(1-f_1)c(\lambda T_1 - (1-e^{-\lambda T_1}))f_1\}
\end{aligned} \quad (6)$$

where

$$\Omega = \lambda_1 \max[T_1, T_i''] + (\frac{\lambda \max[0, T_1 - T_i'']}{1-e^{-\lambda \max[0, T_1 - T_i'']}} - 1)f_1. \quad (7)$$

A derivation of this expression is outlined in the appendix. In addition to delimiting the best achievable performance of at-arrival policies, this policy also provides another baseline against which more practical policies (that may or may not allow server selection decisions to be changed after arrival) can be compared.

C. Dynamic Policy

Similar to the optimal lower bound policy, our candidate dynamic policy uses a threshold on the number of outstanding requests from local clients in determining if a server's fountain should be activated. In contrast to the optimal policy, however, an activated fountain stays active until there are no outstanding requests from local clients, so as to reduce the frequency with which clients must switch servers. Through exploration and evaluation we have found a particular candidate dynamic policy particularly promising. This policy is described next.

In our candidate dynamic policy, a new request from a group i client is served locally at server i if and only if one of the following conditions holds (note that no more than one of these conditions can hold simultaneously): (i) server i's fountain is currently active, (ii) no fountains are currently active, or (iii) there is some other server j whose fountain is currently active, and serves at least $2(1-c)/c$ group i clients. If the last condition holds, in addition to serving the new request locally, those group i clients receiving data from server j switch to server i (whose fountain is now activated).

Otherwise, the server j that is serving other group i clients is chosen, if any; if there are none, the server j whose fountain became active the most recently is chosen.

Finally, we use the following additional rule. Whenever a group j client completes its download from server j's active fountain, if the only remaining clients receiving data from server j are from groups other than group j, all of these clients switch to server i^*, and server j's fountain is deactivated. Here, i^* is the index of the client group with the maximum number of clients receiving data from server j, with ties broken according to the longest remaining required download time.

Note that this policy does not require knowledge of request rates (or assumptions of Poisson request processes). Also, at any point in time, all group i clients currently downloading the file are downloading from the same server, simplifying the collection of the state information needed for server selection decisions. Fountain switching can be greatly reduced in comparison to the lower bound policy since an active fountain at a server i is deactivated only when there are no remaining active group i clients. Finally, note that when fountain switching does occur, the new server is always a server with an inactive fountain, which is then activated. This would allow easy use of uncoded fountains, without requiring careful coordination of all servers (as would be necessary with uncoded fountains in the case of the lower bound or optimal at-arrival policies, so as to ensure clients did not receive duplicate data blocks).

D. Performance Comparisons

Figures 3(a) and (b) show the average server cost per request and the average remote access cost per request, respectively, as functions of the request rate from each client group, for the improved policies and the lower bound policy and using our default parameter settings. Figure 3(c) shows the percentage increase in the total file delivery cost with each of the improved policies, in comparison to with the lower bound policy. In these and subsequent figures, results for the candidate dynamic policy, for which we do not have exact analytic cost expressions, were obtained by averaging the results from simulations of ten randomly generated request sequences, each with 1,000,000 request arrivals. (Such simulations were also used to validate the analytic cost expressions for the other policies.)

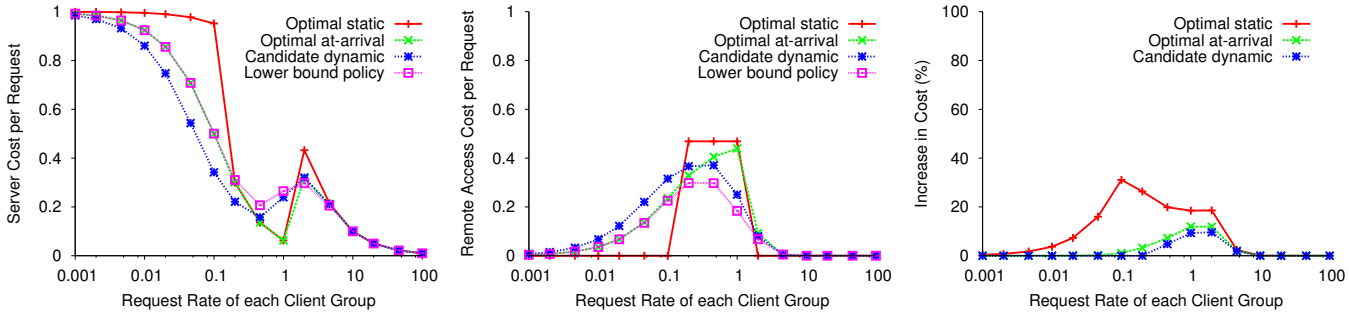
As seen previously in Figure 1(a) for the lower bound policy, the portion of the server cost curve for each of the improved policies where the server cost is increasing as the request rate increases, corresponds to a region where the remote access cost is decreasing (at a higher rate than the server cost is increasing), as the policy increasingly favors local service. Comparing the optimal at-arrival and the candidate dynamic policy, the latter policy is seen to achieve a marginally lower total file delivery cost at intermediate request rates, even though it is a much simpler policy using less state information.

Figures 4(a), (b) and (c) show the impact of alternative parameter settings. As in Figure 3, the candidate dynamic policy achieves a somewhat lower total file delivery cost than the optimal at-arrival policy, and this cost is within 10% of the lower bound in all cases. (While we do not have exact analytic expressions for the candidate dynamic policy, we note that these results suggest that the total cost can be estimated fairly well with the at-arrival and lower bound expressions.)

Finally, Figures 5(a) and (b) show the average number of times a client switches servers during its download time, and the average size of the group of clients that move to a new server when a fountain switch occurs, respectively, for a number of example scenarios. One of these scenarios uses our default parameter settings, while in each of the others one of the three primary parameters (c, N, and α) is increased or decreased while the other parameters remain fixed. Note that the average number of times a client switches servers is typically significantly below one. When a switch does occur, in most of the example scenarios typically only one or two clients are involved, but in two scenarios (that with 64 servers, and that with $c = 0.3$), larger groups of clients switch.

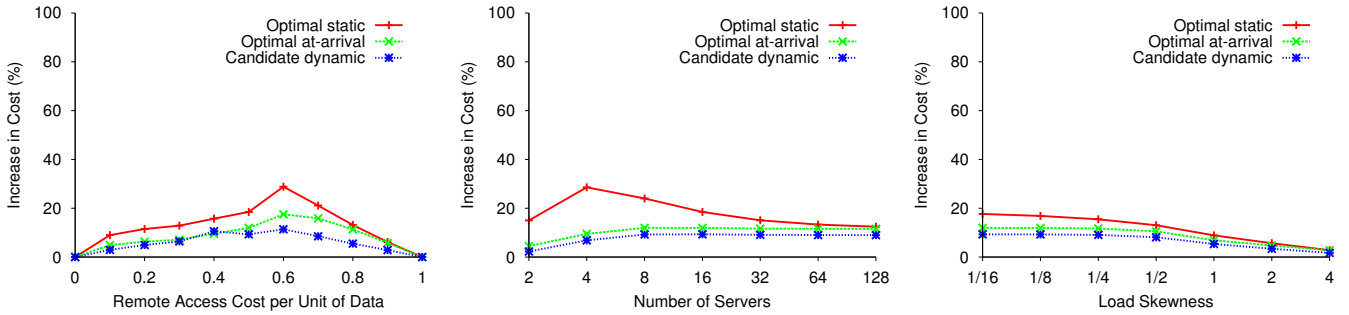
VI. Streaming Video Delivery

In this section we apply the cost model and policies developed in the previous sections for content download, to streaming video delivery. We use the Reliable Periodic Broadcast (RPB) protocols [12], in which the video file to be delivered is divided into segments of increasing lengths, each of which is delivered using a digital fountain. As with content download, the cost of delivering data long distances over the Internet suggests that a minimal cost solution may entail replication of each segment at multiple geographically distributed servers. This leads to a server selection problem analogous to that considered earlier, but now for each of multiple video file segments.



(a) Server cost (b) Remote access cost (c) Relative difference

Fig. 3: Performance of improved policies ($c = 0.5, N = 16, \alpha = 0$).



(a) Remote access cost per data unit (c) (b) Number of servers (N) (c) Load skewness (α)

Fig. 4: Impact of system parameters on improved policies (default parameters: $c = 0.5, N = 16, \lambda = N, \alpha = 0$).

A. Protocol Description

The RPB protocols add efficient error recovery (through use of digital fountain delivery of each segment) to the Optimized Periodic Broadcast (OPB) Protocols [12]. The segment lengths used in the OPB protocols are optimized so as to achieve the minimum possible startup delay for a given total server bandwidth usage, assuming that: (1) clients receive each segment entirely before beginning playback of the segment, and (2) each segment is continuously transmitted (e.g., on its own multicast group), regardless of whether there are any clients receiving it.

With OPB, the video file is divided into K segments (each of which will be continuously transmitted at rate r), where K is chosen depending on the desired total server bandwidth usage. A new client immediately begins reception of the first s segments. Once a segment has been completely received, reception of the next segment that the client has not yet started downloading, if any, is initiated. Playback can commence once the first segment has been received. Each subsequent segment is chosen to be as long as possible, subject to the condition that the segment must be received prior to when its data is needed for playback. Measuring r in units of the playback bit rate, and denoting the length (playback duration) of segment i by l_i and the total length of the video by L, this yields [12]:

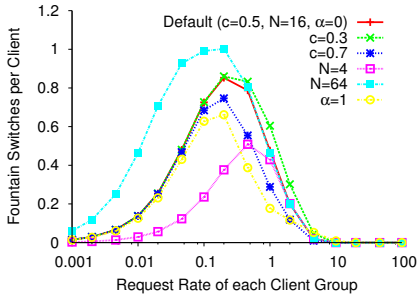
$$l_k = \begin{cases} l_1 + r\sum_{j=1}^{k-1} l_j, & 1 < k \leq s \\ r\sum_{j=k-s}^{k-1} l_j, & k > s, \end{cases} \tag{8}$$

where l_1 is selected such that $\sum_{j=1}^{K} l_j = L$.

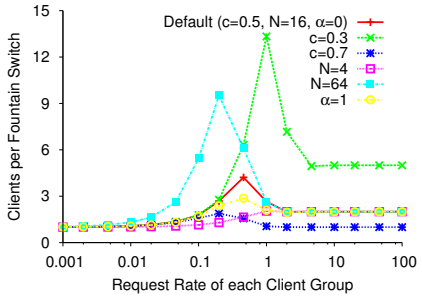
B. Optimality Discussion

Interestingly, we have found that OPB does not necessarily achieve the minimum possible server bandwidth usage for a given startup delay, when segment transmission is not continuous; i.e., when transmission of a segment is stopped if there are no clients receiving it. For example, consider $r = 1$, $s = 2$, $K = 4$, and $L = 1$. Inserting these values into equation (8), we note that the OPB segment lengths in this case are $\frac{1}{11}, \frac{2}{11}, \frac{3}{11}$, and $\frac{5}{11}$, and the startup delay is $\frac{1}{11}$. Consider now an alternative scheme in which the video file is divided into six segments, the first 5 of length $\frac{1}{11}$ and the last of length $\frac{6}{11}$, and in which clients begin reception of the first and last segments initially, begin playback after reception of the first segment (yielding a startup delay of $\frac{1}{11}$), and begin reception of segment i ($2 \leq i \leq 5$) once reception of segment $i - 1$ is complete. Although the server bandwidth usage with OPB is lower than with this alternative scheme at high request rates, the alternative scheme can have slightly lower server bandwidth usage at intermediate request rates (at most, by about 0.22%, at $\lambda = 0.96$), in the case where transmission of a segment is stopped if there are no clients receiving it.

Although not guaranteed to be optimal when segment transmission is not continuous, we have not been able to find examples in which OPB is significantly sub-optimal. Thus, we believe that OPB as extended by using digital fountain delivery for each segment (i.e., as in the RPB protocols), is a good candidate for cost-efficient streaming video delivery.



(a) Fountain switches per client



(b) Clients per fountain switch

Fig. 5: Fountain switching with candidate dynamic policy (default parameters: $c = 0.5, N = 16, \lambda = N, \alpha = 0$).

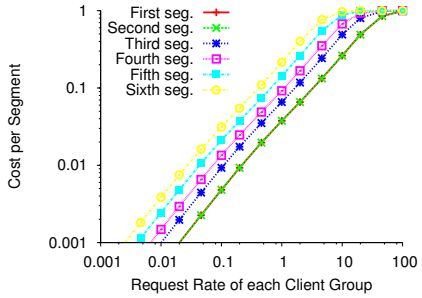


Fig. 6: Delivery cost for each segment with RPB ($K = 6, s = 2, r = 1$) using the candidate dynamic server selection policy for each segment ($c = 0.5, N = 16, \alpha = 0$).

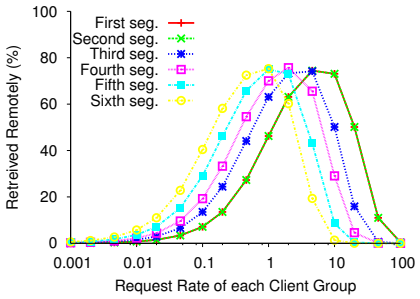


Fig. 7: Percentage received from a remote server with RPB ($K = 6, s = 2, r = 1$) using the candidate dynamic server selection policy for each segment ($c = 0.5, N = 16, \alpha = 0$).

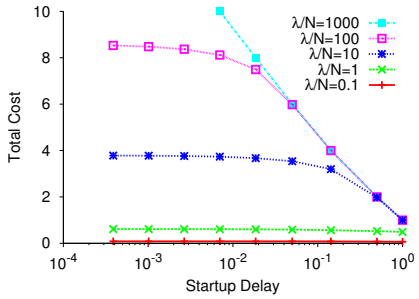


Fig. 8: Tradeoff between startup delay and delivery cost for RPB ($s = 2, r = 1$) using the candidate dynamic server selection policy for each segment ($c = 0.5, N = 16, \alpha = 0$).

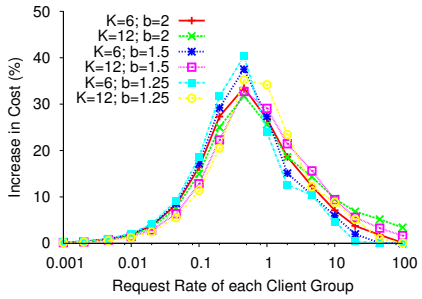


Fig. 9: Cost increases when using RPB ($s = 2, b = rs$) with the static optimal server selection policy rather than the candidate dynamic policy ($c = 0.5, N = 16, \alpha = 0$).

C. Performance Evaluation

We now consider the delivery cost with a system using RPB (with fountains being active only if there are clients to serve), replication of each segment at multiple geographically distributed servers, and the candidate dynamic server selection policy for each segment. For simplicity, in these experiments we assume that packet loss is negligible (i.e., the parameters a_k as defined in [12] are set to 1). Figures 6 and 7 show the total delivery cost of each segment and the average percentage of each segment that is received remotely, respectively, as a function of the request arrival rate, for an example set of RPB parameters. (The startup delay with this set of parameters, using as before $L = 1$, is $\frac{1}{32}$.[2]) Note that for most of the considered parameter space, the total delivery cost is dominated by the delivery costs of the last (longest) segments.

An interesting observation is that the identity of the segments with the highest average percentage received remotely is highly dependent on the request rate. For example, the last (longest) segments have the *largest* average percentage received remotely at low request rates, and the *smallest* at high request rates, while the first (shortest) segments have the *smallest* average percentage received remotely at low request rates, and the *largest* at high request rates. This is consistent with the results shown for the candidate dynamic policy in Figure 3(b), keeping in mind that the key parameter with respect to the efficiency of fountain delivery is the average number of requests arriving during the download time. Note that the download time is longer for later segments, implying that, for a fixed request rate, the average number of requests arriving during the segment download time is greater for later segments. This observation has obvious implications for systems in which not all segments are replicated at all servers, owing to storage limitations for example.

Figure 8 illustrates the tradeoff between startup delay and total file delivery cost. Here, the startup delay is varied by varying the number of segments K, resulting in varying total file delivery cost. Note that dramatic reductions in startup delay can be achieved with relatively small increases in delivery cost.

Figure 9 illustrates that the server selection policy used for each segment can have a substantial impact on the delivery cost. Each curve shows the percentage increase in cost incurred

[2]The segment lengths in this case are $\frac{1}{32}, \frac{2}{32}, \frac{3}{32}, \frac{5}{32}, \frac{8}{32}$, and $\frac{13}{32}$.

when using the static optimal policy, rather than the candidate dynamic policy, for a system using the specified RPB parameters. Note that the cost increases exceed 30% in some cases, suggesting that, as in the download context, there may be significant benefit to using the candidate dynamic policy rather than a static server selection policy.

VII. Related Work

A. Cyclic and Digital Fountain Delivery

Considerable prior work has considered cyclic delivery, and variants using erasure coding, as a method of achieving scalable download of large files from a single server [6], [8]–[10], [22]. With an "ideal" digital fountain [6], the server transmits an unbounded stream of encoded data blocks, all of which are distinct, and such that any collection of these blocks equal in number to the number of blocks in the uncoded file is sufficient, with minimal effort, for recovery of the original data. Different erasure-coding techniques yield different compromises with respect to these ideal properties; i.e., with respect to cost, repetition of blocks, and how many blocks must be received for file recovery [23]–[26].

Use of multiple cyclic delivery channels, or digital fountains, has been proposed for the case in which clients have differing rates at which they are able to receive data (e.g., [27], [28]). Each client listens to a subset of these. By careful selection of the order in which data blocks are transmitted on each channel [22], [29], or use of erasure codes with long stretch factors [26], receptions of the same data block on different channels can be reduced or eliminated.

Parallel download has been proposed for scenarios with multiple distributed servers, so as to eliminate the need for server selection and to minimize download times [30], [31]. However, as each connection is associated with some overhead, the advantages of parallel download may decrease as the portion of clients using parallel download increases [32].

In this paper, we assume a single rate at which all clients are to receive data. Under this assumption, cost is minimized by choosing each fountain's transmission rate as this single rate, rather than choosing a lower rate and then using parallel download. Each active client listens to only a single fountain at each point in time, although in some of the server selection policies that we consider a client may switch servers, possibly multiple times, during its time in system.

B. Server Selection

Prior work on the server selection problem has assumed individual rather than aggregated service [3]–[5], or has considered aggregated service but only in the specific context of streaming video delivery and service aggregation techniques appropriate to this domain [21], [33]–[35]. Within this latter category, Almeida *et al.* [33] consider the problem of server placement, as well as server selection and media stream routing, with the objective of minimizing a weighted sum of network and server bandwidth usage. They show that use of service aggregation can result in optimal placement, selection, and routing solutions that are very different from those for systems without service aggregation. Carlsson and Eager [21] use a system model similar to that employed here, to compare classes of server selection policies for systems in which multiple servers implement a batched video-on-demand service. They conclude that server selection policies using dynamic system state information can potentially yield large improvements in performance, while deferred rather than at-arrival server selection has the potential to yield further substantial performance improvements for some regions of the parameter space. To our knowledge, no prior work has considered the server selection problem in systems using digital fountain delivery at each server.

VIII. Conclusions

With content delivery (download and streaming) contributing to the majority of today's Internet traffic, cost-efficient content delivery is an important step towards better overall resource usage (and/or a "greener" Internet). We consider the problem of minimizing the total content delivery cost in systems with distributed servers, each using digital fountain delivery. Using a simple system model, that enables analytic comparison of server selection policies, we formulate and analyze policies that are optimal within their respective class, including a tight lower bound policy. Based on the characteristics of the lower bound policy, a readily-implementable policy is proposed that is found to achieve within 10% of the lower bound. Finally, we show how our results for content download can also be applied to streaming video delivery.

IX. Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the Informatics Circle of Research Excellence (iCORE) in the Province of Alberta.

Appendix

This appendix outlines an analysis of the optimal at-arrival server selection policy for the case in which the client groups may have differing request rates. The optimal at-arrival policy and the threshold parameters T_i, T_i', and T_i'' are as defined in Section V-B.

As in the case in which all client groups have the same request rate, the analysis proceeds by considering the state of the system at an arbitrary time t. Consider first a server and client group i other than server/group 1. If there has been at least one request arrival from client group i in the time interval $[t - L/b, t - T_i]$, server i's fountain will be active at time t. If there have been no requests from client group i but at least one request from some other client group j in the time interval $[t-L/b, t-T_i]$ (and thus server j's fountain is active at time t), all requests that arrive from client group i in the time interval $[t-T_i, t]$ will be receiving service from a remote server at time t. If there have been no requests from any client group in the time interval $[t - L/b, t - T_i]$ but at least one request in the interval $[t-T_i, t]$, and the first such request was from a client group other than group i, all requests that arrive from client

group i in the time interval $[t-T_i, t]$ will be receiving service from a remote server at time t (note that the expected number of such requests must be conditioned on there being at least one request arrival, with the first such being from other than group i). If there have been no requests from any client group in the time interval $[t - L/b, t - T_i]$ but at least one request in the interval $[t - T_i, t - T_i']$, or no requests from any client group in the time interval $[t - L/b, t - T_i'']$ but at least one request in the interval $[t - T_i'', t]$, and the first such request was from client group i, then server i's fountain will be active at time t. The above corresponds to the first four terms within the first set of curly brackets of equation (6).

The remaining case that has non-zero expected cost is where there have been no requests from any client group in the time interval $[t-L/b, t-T_i']$ but at least one request in the interval $[t-T_i', t-T_i'']$, and the first such request was from client group i. In this case, all requests that arrive from client group i in the time intervals $[t-T_i', t-T_i'']$ and $[t-T_i'', t]$ will be receiving service from server 1 at time t. Referring to equation (6), terms six and seven in the first set of curly brackets correspond to the remote access cost associated with these requests, while terms eight and nine correspond to the server cost at server 1, as incurred for these requests (while compensating for the fact that the analysis for server 1 does not take these requests into consideration), at the times during the intervals $[t - T_i', t - \max[T_1, T_i'']]$ and $[t - \max[T_1, T_i''], t - T_i'']$, respectively.

The analysis for server and client group 1 follows a similar approach. In the resulting analytic expression for the total file delivery cost, as shown in equation (6), the terms for server and client group 1 (within the second set of curly brackets), neglect the fact that requests from other than client group 1 can cause server 1 to be scheduled; this is compensated for with the last two terms for each server/client group i (within the first set of curly brackets).

REFERENCES

[1] Sandvine, "2009 global broadband phenomena," Tech. Rep., Oct. 2009.

[2] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of Web server replicas," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[3] R. L. Carter and M. E. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proc. IEEE INFOCOM*, Kobe, Japan, Apr. 1997.

[4] E. W. Zegura, M. H. Ammar, Z. Fei, and S. Bhattacharjee, "Application-layer anycasting: a server selection architecture and use in a replicated Web service," *IEEE/ACM ToN*, vol. 8, no. 4, Aug. 2000.

[5] K. L. Johnson, J. F. Carr, M. S. Day, and F. Kaashoek, "The measured performance of content distribution networks," *Computer Communications*, vol. 24, no. 2, Feb. 2001.

[6] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc. SIGCOMM*, Vancouver, Canada, Sept. 1998.

[7] C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *Proc ICMCS*, Hiroshima, Japan, June 1996.

[8] L. Rizzo and L. Vicisano, "A reliable multicast data distribution protocol based on software fec techniques," in *Proc. HPCS*, Chalkidiki, Greece, June 1997.

[9] S. Rost, J. Byers, and A. Bestavros, "The cyclone server architecture: Streamlining delivery of popular content," in *Proc. WCW*, Boston, MA, June 2001.

[10] K. V. Almeroth, M. H. Ammar, and Z. Fei, "Scalable delivery of web pages using cyclic best-effort (udp) multicast," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1998.

[11] L. Vicisano, L. Rizzo, and J. Crowcroft, "Tcp-like congestion control for layered video multicast data transfer," in *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 1998.

[12] A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel, "Scalable on-demand media streaming with packet loss recovery," in *Proc. ACM SIGCOMM*, San Diego, CA, Aug. 2001.

[13] B. Cohen, "Incentives build robustness in bittorrent," in *Proc. Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[14] J. Erman, A. Gerber, M. T. Hajiaghayi, D. Pei, and O. Spatscheck, "Network aware forward caching," in *Proc. WWW*, Barcelona, Spain, Apr. 2009.

[15] S. Nedevschi, S. Ratnasamy, and J. Padhye, "Hot data centers vs. cool peers," in *Proc. HotPower*, San Diego, CA, Dec. 2008.

[16] L. A. Barroso and U. Hölze, "The case for energy- proportional computing," *IEEE Computer*, vol. 40, no. 12, Dec 2007.

[17] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, "Delivering energy proportionality with non energy-proportional systems – optimizing the ensemble," in *Proc. HotPower*, San Diego, CA, Dec. 2008.

[18] K. Le, R. Bianchini, M. Martonosi, and T. D. Nguyen, "Cost- and energy-aware load distribution across data centers," in *Proc. HotPower*, Big Sky, MT, Oct. 2009.

[19] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Magg, "Cutting the electric bill for Internet-scale systems," in *Proc. ACM SIGCOMM*, Barcelona, Spain, Aug. 2009.

[20] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon, "Analysis of educational media server workloads," in *Proc. NOSSDAV*, Port Jefferson, NY, June 2001.

[21] N. Carlsson and D. L. Eager, "Server selection in large-scale video-on-demand systems," *ACM TOMCCAP*, vol. 6, no. 1, Feb. 2010.

[22] Y. Birk and D. Crupnicoff, "A multicast transmission schedule for scalable multirate distribution of bulk data using non-scalable erasure-correcting codes," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar/Apr. 2003.

[23] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The use of forward error correction (fec) in reliable multicast," RFC 3453, IETF, Dec. 2002.

[24] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proc. ACM STOC*, El Paso, TX, May 1997.

[25] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM CCR*, vol. 27, no. 2, Apr. 1997.

[26] A. Shokrollahi, "Raptor codes," Digital Fountain Inc., Tech. Rep. DF2003-06-001, 2003.

[27] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. ACM SIGCOMM*, Stanford, CA, Aug. 1996.

[28] M. Luby, V. K. Goyal, S. Skaria, and G. B. Horn, "Wave and equation based rate control using multicast round trip time," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[29] S. Bhattacharyya, J. F. Kurose, D. Towsley, and R. Nagarajan, "Efficient rate-controlled bulk data transfer using multiple multicast groups," in *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 1998.

[30] J. W. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads," in *Proc. IEEE INFOCOM*, New York, NY, Mar. 1999.

[31] P. Rodriguez and E. W. Biersack, "Dynamic parallel-access to replicated content in the Internet," *IEEE/ACM ToN*, vol. 10, no. 4, Aug. 2002.

[32] C. Gkantsidis, M. Ammar, and E. Zegura, "On the effect of large-scale deployment of parallel downloading," in *Proc. WIAPP*, San Jose, CA, June 2003.

[33] J. M. Almeida, D. L. Eager, M. K. Vernon, and S. J. Wright, "Minimizing delivery cost in scalable streaming content distribution systems," *IEEE TMM*, vol. 6, no. 2, Apr. 2004.

[34] Z. Fei, M. H. Ammar, and E. W. Zegura, "Multicast server selection: Problems, complexity and solutions," *IEEE JSAC*, vol. 20, no. 7, Sept. 2002.

[35] M. Guo, M. H. Ammar, and E. W. Zegura, "Selecting among replicated batching video-on-demand servers," in *Proc. NOSSDAV*, Miami Beach, FL, May 2002.