An Intelligent Satellite Multicast and Caching Overlay for CDNs to Improve Performance in Video Applications

Chris Brinton[1], Ehsan Aryafar[1], Steve Corda[2], Stan Russo[2], Ramiro Reinoso[2], Mung Chiang[1]
[1]Princeton University, Department of Electrical Engineering, Princeton, NJ 08540
[2]SES, Princeton, NJ 08540

***Abstract*—Over the past decade, video has become the dominant form of traffic consumed over content delivery networks (CDNs). This trend, coupled with the ever-increasing subscriber base, has caused an explosion of data demands in a wide variety of scenarios. Such trends have resulted in heightened levels of congestion within today's terrestrial networks and are expected to become more acute in the coming years.**

To combat network congestion, we propose a satellite-based overlay for existing terrestrial CDNs. Satellite networking has distinct advantages over terrestrial networks in being able to distribute delay-tolerant high bandwidth content across a wide geographic area simultaneously, with few limitations to the distance between requestor and source, nor the number of locations being served. Additionally, our solution calls for cache storage at local proxy servers one-hop from the end users, which in most instances will improve the response time of current network architectures. The proposed cache algorithm leverages the homogeneous coverage area provided by satellite to allow each proxy server to compare its local network view to the global picture, learn the popularity distributions quickly, and make its own caching decisions.

Through simulations of two CDN case studies - Cellular and Video on Demand - we find that multicasting can provide significant reductions in required network bandwidth as compared to terrestrial-based unicast, for situations dominated by video traffic. Further, by leveraging advantages offered by our caching algorithm, we show that the multicast solution scales well, both with increasing cache storage and coverage area. Our solution appears robust as relevant traffic parameters, such as heavy-tail characteristics and global file popularity, are varied.

The work presented in this paper is the result of an ongoing collaboration between Princeton University and SES. We believe that our solution incorporates the technologies best suited for the networking challenges being faced today and is forward looking in its ability to scale with demand, content type and size, which enables new market opportunities for the satellite industry.

I. Introduction

In recent years, networks have experienced an unprecedented growth in data demands. Indeed, global IP traffic has increased eight-fold over the past five years, and is expected to exceed 100 exabytes per month by 2016 [1]. As opposed to standard web queries, a remarkable driver of this trend has been multimedia data: It is estimated that video alone comprises over half of all consumed Internet traffic today.

Not only is its demand on the rise, but the number of services offered to access video is growing. For instance, with the advent of the smartphones, there have been exorbitant increases in the amount of data consumed on mobile devices. In 2012, mobile video data exceeded half of the total for the first time [2], and by 2016, this fraction is anticipated to reach two-thirds, at which time mobile requests will encapsulate 10% of total IP traffic [1]. At the same time, cable TV providers have expanded their services to include consumer managed IP traffic, like IPTV and Video on Demand (VoD). VoD traffic, in particular, currently generates roughly 6 exabytes per month, and this number is expected to triple by 2016 [1].

These increases in demand are expected to cause heightened congestion on terrestrial networks in coming years [3]. For instance, cellular networks, despite employing state of the art technologies like 4G-LTE, are encountering difficulties with the onslaught of requests due to capacity issues in their legacy backhaul links [4], [5]. A significant portion of backhaul networks in the US still employ copper T1/E1 technology to connect cell sites to the core network [6], and these links will have exceeding difficulty supporting demand aggregation among towers.

Network operators typically turn to fiber technology to manage capacity issues. But fiber installation is quite expensive even in developed countries, and can cost up to millions per mile [4]. Another common practice has been for these operators to license content delivery networks (CDNs), since CDNs can load-balance demand by splitting capacity among geographically distributed surrogate (proxy) servers [7]. In such schemes, a request from a subscriber is forwarded to one of these proxys through a redirect mechanism in an attempt to keep the query from traversing far into the network. The success of CDNs depends on their ability to store popular content close to the users, both to reduce network congestion and enhance QoS through reduced latency [8].

With terrestrial CDNs, the inherent unicast architecture of the Internet has coupled well with small file sizes requested on demand from individual users. But the proliferation of video brings about a new class of highly asymmetric traffic in which each request is more significant. Supporting multiple point-to-point sessions for this content will become challenging as the number of users and the size of the files increase, due to increasing bandwidth strains. This is especially true in situations where networks are bottlenecked by legacy copper or lower-end fiber.

Needed is a network architecture that is scalable with the increasing demands for video traffic. Currently, we are investigating a practical solution to this issue. The purpose of this paper is to describe our proposal, and evaluate its efficacy under realistic traffic conditions.

One of the benefits of video traffic is that its request trends tend to follow a "heavy-tailed" distribution, in which a large fraction of requests occur for a relatively small fraction of the content. Such trends have been noticed in networks ranging from cellular [9], to user generated content [10], to IPTV [11] and VoD [12]. This implies that some videos are popular and requested with high frequency by different end users. For this reason, multicasting is a desirable feature to augment current CDNs, so that many of these requests for the same content can be served simultaneously.

IP/application layer multicasting is inherently difficult to accomplish over terrestrial fiber-based CDNs, for both technological and cost reasons. On the other hand, satellite is the ideal technology for multicasting, given the efficiency it champions with point-to-multipoint communication. This makes it an ideal multicast source for delay-tolerant, bandwidth intensive CDN traffic [13]. Video is no exception to this, because of the significant amount of pre-coded videos and movies that may be upwards of a few GB in size, especially for HD files [9], [14].

Part of our proposal is to use satellite for multicasting, to leverage its scalability with increasing demand. We will investigate such scalability properties in Section IV. But not all videos are pre-coded, and not all content is multimedia. As a result, we advocate a satellite-based *overlay* for existing terrestrial CDNs, as shown in Figure 1. This architecture combines the benefits of each technology. While fiber can handle individual requests instantaneously, satellite is better equipped to serve multiple, delay-insensitive requests simultaneously.

Since satellite networks are generally the more cost effective medium when there are multiple requests for the same content, it is important to have reasonable estimates on the number of identical requests that can be expected in a given time frame. Also note that the more frequent the requests for the same video, the less the delay each user will experience, beacuse the multicast group for that file will fill up sooner. We advocate a technique that compliments multicasting – the placement of cache storage in terrestrial networks – to minimize the number of re-transmissions for content that is frequently requested. Further, if the storage is kept close to the end users, it will enhance QoS through reduced latency.

Caching is practical for video distribution because requests exhibit a heavy-tail relationship, as stated. In order to operate the cache efficiently, it is necessary to store the files that are most likely to be requested. Determining the request probabilities for a set of files is in general a difficult problem, because the answer changes temporally as well as geographically [15]. In response, we have developed a caching algorithm that estimates the relative popularity of videos, by leveraging the satellite's global view of the network. The homogeneous coverage area of the satellite allows it to keep up-to-date popularity counters that can respond quickly to shifting trends, due to the sheer number of requests it observes in a short amount of time. Such uninterrupted, homogeneous coverage simply is not possible with current terrestrial technology [13]. Each cache can use this global view to adjust its own local estimates of popularity, allowing it to make efficient update decisions upon the arrival of new content. We will describe our algorithm in Section II, and will evaluate its efficacy in Section IV.

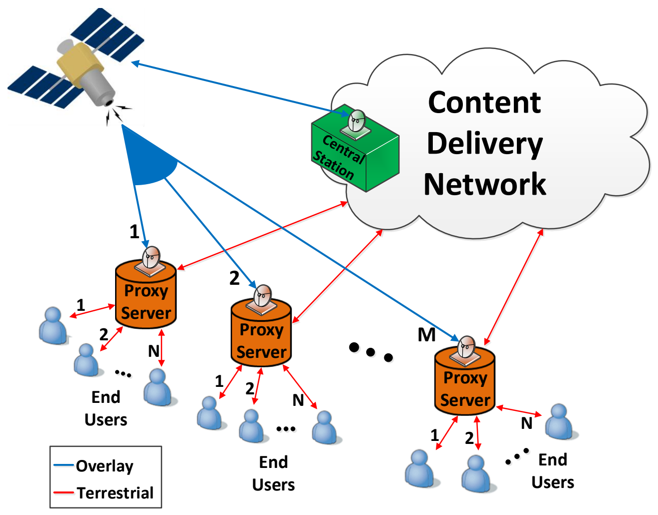

Fig. 1. High level CDN topology with satellite overlay. Each of the M proxys, with cache size S, serve requests for N end users. Directions of the arrows indicate uni/bi-directional communication.

The rest of this paper is organized as follows. In Section II, we will present the CDN overlay architecture and caching algorithms. In Section III, we will describe file popularity modeling that we use in our simulations, and in Section IV, we will evaluate our architecture and compare it to terrestrial-based unicast. In Section V some related work will be discussed, and in Section VI we will conclude our work.

II. Architectural Description

In this section, we will explore the network architecture of our proposed satellite overlay. On the satellite side we will discuss queueing and multicasting, and on the server side we will decribe the cache update algorithm.

A. Topology overview

Figure 1 depicts, from a high level, our satellite overlay to CDN networks. On the terrestrial end, each of the M proxy caches have storage capacities of size S, and store popular content one hop away from the N end users that they serve. We assume that the subscriber regions are geographically disjoint, and that the average number of users in each region (N) is constant. The proxy caches have a bidirectional connection to the rest of the CDN, which is hierarchial in nature and contains all content origin servers.

There are two reasons why the injection point is chosen to be one-hop from the users. First, it enhances QoS through reduced latency. Second, it carries the congestion alleviation through all portions of the CDN; in particular, the mobile backhaul bottleneck connecting the proxys to the rest of the CDN in the case of cellular.

The Central Station serves as the coordinator between the satellite overlay and the rest of the CDN [16]. It receives requests for content that were missed at the proxy caches and forwards them to the satellite. The satellite, in turn, fetches this content from the CDN and multicasts the requests to the proxy caches. We assume that the coverage area of the satellite is large enough to include all proxy caches served by the CDN.

B. Satellite and multicast

Here, we will give an overview of the logic we envision governing the satellite overlay network. There are two main portions: queuing and multicast.

1) Queueing: The queue keeps track of the files that are pending multicast transmission. When the network routes a request to the satellite overlay, it will first check if the file is currently queued. If not, the overlay will fetch the content from the CDN, and add it to the queue. Since this is the first outstanding request for the file since the last multicast transmission for it, the overlay will initiate a stream to be sent out in a prescribed amount of time from now, say `TIMEOUT`. Identical requests arriving within the `TIMEOUT` window will be served with this transmission.

The queue gives the satellite the ability to aggregate identical requests and multicast them simultaneously, rather than sending individual unicast streams. Of course, while larger values of `TIMEOUT` improve bandwidth efficiency, they also imply a longer average delay in delivering the content. This highlights the fact that the overlay is best suited for bandwidth intensive, delay-insensitive content, like non-real time video. Even so, it is worth mentioning that `TIMEOUT` can be kept relatively small while still witnessing significant improvement in the peak bandwidth.

2) Multicasting: Once `TIMEOUT` is reached for a given file in the queue, the overlay network will add it to the multicast stream. At this point, it will be simultaneously broadcasted to all proxys in the coverage area. This will clear all requests currently in the queue for this particular file. The number of files that can be multicasting simultaneously is limited by the satellite bandwidth, as well as the stream rate of each file currently transmitting.

It is important to ensure that the satellite stream is not overloaded, and hence, we treat the peak bandwidth provision as a design parameter to be determined in our evaluation. But once in operation, the overlay network may encounter a situation where multicast demand is greater than supply. In this case, some requests would need to be dropped, or re-routed terrestrially, as they expire in the queue. As a result, the overlay network would need to prioritize elements of the queue, to determine which will be selected upon the opening of a slot in the multicast stream. One possibility is to keep track of the number of outstanding requests for each queued file, and to prioritize by the ratio of this number to the file size. Denote the number of outstanding requests for file i as n_i, the size as f_i, the stream rate as b_i, and the satellite bandwidth as B. When a file is finished transmitting, the next file I to be multicasted can be selected as follows:

$$I = \underset{i \in \mathtt{Q}}{\operatorname{argmax}} \left(n_i/f_i : b_i \leq B - \sum_{k \in \mathtt{MC}} b_k \right), \tag{1}$$

where `Q` is the set of files in the queue, and `MC` is the set of files currently multicasting. Equation (1) is favoring queued files that are smaller (*i.e.*, they will spend less time consuming bandwidth) and have more outstanding requests, constrained by the fact that the bit rate of the file must be smaller than the available multicast rate.

Beyond multicasting, the overlay keeps track of the aggregate requests across the entire coverage area of the CDN. We call this count the global popularity, because it gives a network-wide estimate of the relative popularity of the files. The satellite frequently updates the proxy servers with these values so they have an up-to-date, global view of the network, which they can use when updating their caches. The satellite should also keep track of the request trajectory over some reasonable window, to factor in temporal shifts in popularity.

C. Proxy caches and update algorithm

Now, we will turn to the logic for managing requests at the proxys. Recall these proxy servers are at the edge of the CDN, as shown in Figure 1.

When a user requests a file, it may or may not be in the local cache. If it is, this counts as a cache `hit`, and the proxy serves the request immediately. If not, this is a cache `miss`, and the request is transferred upstream to the CDN.

Each proxy keeps track of the number of requests it has seen for each file, including those that have been requested but are not in the cache. We call this the local popularity, because it gives an estimate of the relative popularity of the files specific to the proxy. As with global popularity, temporal shifts should be taken into consideration when updating this counter over long time periods.

When a file begins multicast transmission from the satellite, each proxy needs to make two decisions: whether to stream the file, and whether to cache it.

1) Stream: If the proxy has an outstanding request from one or more clients for this piece of content, then it should be streamed to the necessary recipients. The proxy can keep track of outstanding requests by logging the information in its local cache.

2) Cache: If the remaining space in the cache is large enough to fit the file, then the proxy will download and store it. It will keep track of its local and global popularity counters accordingly. On the other hand, if there is not enough room, then a cache update decision must be made. We will describe this process now.

In our caching scheme, each proxy weights the local and global popularity of each file, to compute a hybrid popularity metric. Which one should be weighted more depends on how accurate the local estimate is expected to be, given the small number of requests each individual proxy witnesses relative to the overall. The file popularities will vary to some extent from region to region, but it is likely that the local estimate will lack the sample size necessary for accurate caching information, and hence we take the global estimate as a baseline.

For file i at cache m, denote the global popularity by G_i, and the local popularity as $L_{m,i}$. Also, let $w \in [0,1]$ be the global weighting factor. We define the hybrid popularity metric $P_{m,i}$ as follows:

$$P_{m,i} = (1-w)L_{m,i} + w\left[\frac{G_i - L_{m,i}}{M-1}\right]. \tag{2}$$

The term on the left is the scaled local popularity. The term on the right is a measure of the average requests occuring at proxys other than the current one, scaled accordingly by w. If w is set to 0, we are relying strictly on a local view. On the other hand, if w is set to 1, we are relying strictly on a global view. As we will see in Section IV, intermediate values of w yield the best performance. They exploit the large sample size obtained from the global view to learn the popularities more accurately, while remaining robust to geographical shifts in popularity.

When making the cache decision, we want to retain the files with the highest perceived popularity and lowest size. Our metric of interest is then the ratio of $P_{m,i}$ to the file size f_i. So, each proxy will find the number of files in the cache with lowest $P_{m,i}/f_i$ necessary to make room for the incoming file. If each of their priorities is lower than that of the incoming file, then they are evicted and the new content is cached. Otherwise, the incoming file will not be cached, and no change is made. Defining S to be the total cache size, and $\mathtt{C_m}$ to be the set of files in cache m, Algorithm 1 summarizes the update logic as to whether incoming file i will be cached or not.

Algorithm 1 Cache update algorithm

```
K ← $\emptyset$
cache ← 1

while $\left(S - \sum_{j \in \mathtt{C_m} \setminus \mathtt{K}} f_j < f_i\right)$ do

  $k = \arg\min_{j \in \mathtt{C_m} \setminus \mathtt{K}} (P_{m,j}/f_j)$

  if $P_{m,k} \leq P_{m,i}$ then
    K ← $\{\mathtt{K}, k\}$
  else
    cache ← 0
    break
  end if
end while

if cache $= 1$ then
  $\mathtt{C_m} \leftarrow (\mathtt{C_m} \setminus \mathtt{K}) \cup i$
end if
```

In this algorithm, K is the set of files that will be evicted from the cache if they have lower priority than file i.

III. Client requests

It is imperative to model user behavior with a high level of sophistication, to guarantee realism in our simulations. To do this, we must consider common file popularity distributions, and temporal/spatial variation thereof. In this section, we will describe our models thereof.

A. File popularities

It is a well-cited result that file popularities tend to follow a heavy-tail distribution. This means that the majority of requests occur for a relatively small fraction of the content. Observations of this characteristic has been noted in data logs for various CDNs, *e.g.* see [9], [10], [11], [12], [14].

Such a trend is readily quantified by Zipf's law. Specifically, if we ordered the files from most to least popular at a given point in time, then the relationship governing the frequency at which the file of rank i will appear is given as follows:

$$f(i) \propto \left(\frac{1}{i}\right)^{\alpha}. \tag{3}$$

That is, the probability of a request occuring for file i is inversely proportional to its rank, with a shaping parameter α. A larger α implies that more requests occur for a smaller fraction of the content, making the network more amenable to a caching solution. Values of α have been cited between 0.5 and 0.8 [9]. If we had 10,000 files, this would mean that 50% of requests occur for the top 25% and 6% of the content, respectively. In Section IV, we will investigate how the performance of our solution can vary with α.

B. Temporal and spatial modeling

File popularities change geographically as well as temporally [15]. However, it is difficult to pinpoint exactly how they change for a specific CDN without access to weekly and monthly data logs across a large number of regions. We will describe intuitive methods we use to model these variations.

1) Spatial dynamics: To model geographical shifts, we divide files into two different types: local and global.

- *Global*: Global files will not change rank geographically. In other words, they will retain the same relative popularity in each region. We expect the majority of files to fall into this category. An example would be a recently viralized YouTube video, which is likely to gain high popularity in all regions.
- *Local*: Local files will change popularity rank from region to region. An example of this would be video replays of a sports game at a stadium, which are likely to be requested frequently by those around the area, but not necessarily by those removed from the region.

These differences emphasize the importance of factoring global and local popularity into Equation (2) for use in the cache update algorithm. To simulate the different classes, we assume that a certain fraction of files (g) are global, while the others are local. For the local files, we permute their ranks in the Zipf distribution from region to region, while the global files retain the same rank everywhere.

2) Temporal dynamics: The relative popularity of the files will vary over time as well. The frequency of such shifts will depend on the type of content considered. This emphasizes the importance of our caching solution to quickly adapt to dynamic shifts over time, while remaining robust to random fluctuations. To factor this in, we consider small periods of time (on the order of a few hours), and measure the performance at the end of the time period.

Naturally, their will also be significant time-of-day fluctuations in request patterns. For instance, cellular networks tend to peak mid-day [17], while VoD is most popular at night (local time) [14]. As we are concerned with peak network conditions, we take the analysis period to be during peak hours.

IV. Case studies

We will now delve into two specific CDNs and quantify improvements that satellite-based multicasting can provide. Our case studies are Video on Demand (VoD) and Cellular. As we will see, they have different network parameters but are both amenable to our solution.

We should highlight that our driving purpose here is to compare the performance of the overlay to the terrestrial, rather than simulate the hybrid network as a whole. As a result, we analyze each separately.

A. Video on Demand

Virtually all cable providers (Comcast, Time Warner, etc.) offer video on demand services to the home. In these systems, headends are the facilities that receive the content for processing and local distribution [14]. A headend is typically equipped with several large antennas for reception of cable/satellite TV networks already, making them ideal locations for cache injection. A large city with cable could have a few headends.

We ran simulations for a large VoD system. The parameters and their values we used are shown in Table I. The default column is the value we assume unless otherwise stated, while the minimum and maximum give the range varied in some of the graphs. The file size is the exception, which was varied randomly between 2 GB (roughly 45 min HD at 6 Mbps) and 6 GB (roughly 135 min) in all simulations. With 10,000 videos, this gives a total library size of roughly 40 TB. The cache size at each of the 100 headends, by default, is 25% of this, trading off improved hit rate with cost of additional storage. The default Zipf parameter α of 0.8 limits the best possible hit rate with the default cache size to be slightly greater than 70% (that is, 70% is achieved if the cache can discover and store the most popular 25% of the videos). From intuition, we set the global popularity to 90%, meaning that 10% of the files will have varying popularities.

Our simulations will quantify: (1) the benefit of multicast (over unicast), (2) the advantage of our caching algorithm, and (3) the importance of accurately estimating the heavy-tailed popularity.

1) The benefit of multicast: One of the main benefits of the satellite overlay is the ability to send content via multicast, and service multiple requests at once. To this end, we have simulated the performance of the VoD network under satellite-based multicast, and compared it to terrestrial-based unicast. In the former, caches have a global view due to the presence of the satellite overlay, and have much more frequent opportunities to download content. In the latter, each cache operates independently: they make storage decisions on their own, and only receive what is requested by their subscribers.

TABLE I
Values assumed by the VoD parameters in simulations.

Parameter	Min	Default	Max
Cache headends, M	-	100	-
Subscribers per headend, N	-	5000	-
Average daily requests/subscriber	-	1	-
Cache storage per headend, S (TB)	1	10	40
Video size, f_i (GB)	2	4	6
Bit rate, b_i (Mbps)	-	6	-
Number of videos, L	-	10,000	-
Global popularity, g (%)	0	90	100
Global weight, w (%)	0	50	100
Zipf parameter, α	0.5	0.8	1.2

We quantify performance both in terms of the peak bandwidth required, as well as the average cache hit rate. By intuition, we expect these to be inversely related, as a higher hit rate implies that less requests reach the CDN in the first place, therefore lowering the requisite bandwidth.

The results are shown in Figure 2. For all values of cache storage, the multicast topology has higher performance than the unicast. But the important point is how well the multicast scenario scales with increasing cache storage: the peak bandwidth is decaying rapidly at all points. The performance of the unicast, on the other hand, saturates at around 10 TB. When each headend operates independently, they are at a disadvantage, because they do not have a large sample size to accurately model popularity and make instantaneous caching decisions. This results in a large number of requests that have never been seen before, and hence are not yet cached.

We can expand on this point by considering the cache size of 40 TB, which is roughly the same as the library. In this case, no cache evictions will be made, and every file requested will be stored. The sheer number of requests obtained on a global scale allows each cache to populate rapidly under multicast. As a result, it achieves an enormous decrease in peak bandwidth over unicast, from 78.5 to 5.87 Gbps (an improvement of over 90%), and an improvement in hit rate from 61.7% to 98.9%. At 10 TB, the improvement over unicast is still reasonably substantial, with a bandwidth decrease from 78.8 to 59.6 Gbps (roughly 25%) and a hit rate increase of over 7%.

It is worth mentioning the analysis period used for each topology. The multicast was only run for a simulated time of 6 hours, while the unicast was given 24 hours. The results shown in Figure 2 were taken to be the average over the last three hours in each case (the unicast was further averaged over 20 independent trials). This would put the multicast scenario at a disadvantage, since it had only a fourth of the time than that of unicast to stabilize its cache. Despite this, the multicast achieves almost ideal performance, while the unicast is not

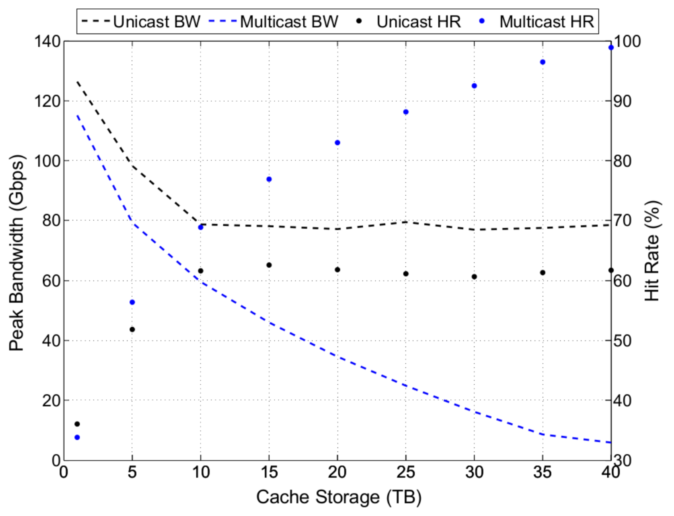

Fig. 2. Comparison between satellite-based multicast and terrestrial-based unicast for varying cache storage at each headend. We use peak network bandwidth and hit rate as our metrics. This highlights the ability of multicast to greatly leverage increased cache storage, while the performance of unicast saturates quickly.

scalable past 10 TB.

2) Caching algorithm performance: Here, we will investigate the performance of our cache algorithm described in Section II. Of particular interest is how the performance varies with different levels of global popularity, and what the resultant weight w should be to obtain the best results. For benchmarking purposes, we will also compare our results to the commonly cited, least-recently used (LRU) caching algorithm.

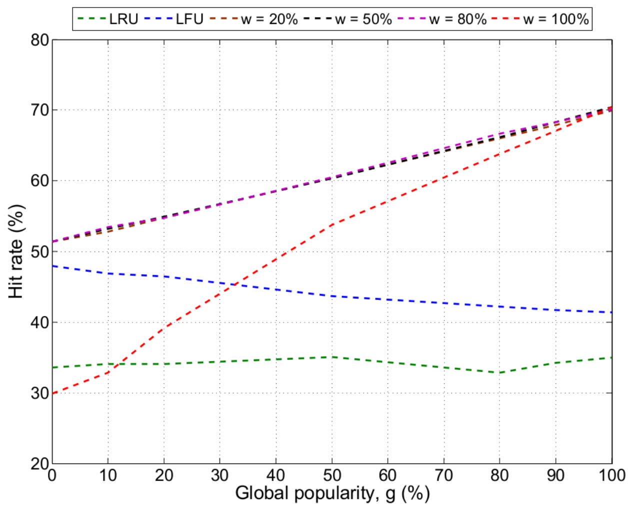

Fig. 3. Performance comparison between different caching schemes, for varying global popularity. The conventional LRU is clearly suboptimal, and used as a benchmark. The best results are obtained using our caching algorithm with a global weight somewhere between zero and one, irrespective of the global popularity. Note that our caching scheme degenerates to LFU when $w = 0\%$.

In Figure 3, we show the hit rate as the global popularity varies under different caching schemes. For LRU, we assume each cache operates independently, and the resultant hit rate is constant at roughly 35%. The other traces correspond to our caching algorithm for different weights. With a weight of zero, each cache relies on a strictly local view, and the algorithm degenerates to a form of least-frequently used (LFU). The difference from conventional LFU is that we prioritize by the ratio of the request count to the file size, rather than just the former. In this case, the performance is independent of g (save fluctuations), achieving a hit rate that is consistently higher than LRU by roughly 10%.

The reason that the (modified) LFU is superior to LRU is two-fold. First, LFU is more robust to random spikes in request counts, because it keeps track of the absolute frequency of each file, whereas with LRU the most recent request is always the highest priority. Second, LFU does not automatically evict upon a cache miss, since the new file is required to have a higher frequency than at least one file in the cache; on the other hand, LRU will always evict, even for a file that has only been requested once.

As w is increased in Figure 3, we start incorporating the global view into the mix. The hit rates for weights of 20%, 50%, and 80% are monotonically increasing with global popularity, exhibiting nice, linear trends. As the content becomes more global, the hit rate increases from a minimum of roughly 50% to a maximum of 70%. It is important to note that for each of these weights, the performance is visibly identical. As long as we factor in at least some global popularity, we enjoy the linear trend, because the requests we see globally help the popularities converge rapidly. And of course, the larger g is, the better the performance becomes.

Additionally, we want to keep w below 100%, because at this extreme the performance begins to decay. This is especially true for low values of g, because then we are relying on a global view when the popularities are predominantly local. Regardless of the global popularity, the best results will always be obtained for $w \in (0, 1)$, which gives some weight to both local and global popularity counters. While absolute performance is not independent of g, the optimal choice of w is.

3) Heavy-tail effects: The popularity characteristics of the VoD system are also important to consider. We will now investigate the effect of the Zipf parameter on performance. Intuitively, we expect performance to increase as α becomes larger, because that means more requests occur for smaller fractions of the content. This makes caching decisions much more transparent.

In Figure 4, we plot the peak bandwidth and hit rate as α varies, for different amounts of cache storage. Clearly, the bandwidth decreases at α gets larger, in all cases. For 5 TB caches, it drops from 133 Gbps at $\alpha = 0.5$ to 21.2 at $\alpha = 1.2$, which is a reduction of 84%. Also, when α is small, we obtain significant performance gains by adding more cache storage: for $\alpha \approx 0.5$, increasing by a factor of four decreases bandwidth by a factor of two. As α increases, we see diminishing returns in increasing storage, because there is less content worth caching.

We can also use Figure 4 finding necessary cache storage to limit the peak bandwidth. For instance, to obtain a peak of 60 Gbps when $\alpha = 0.5$, we need a cache size of 20 TB.

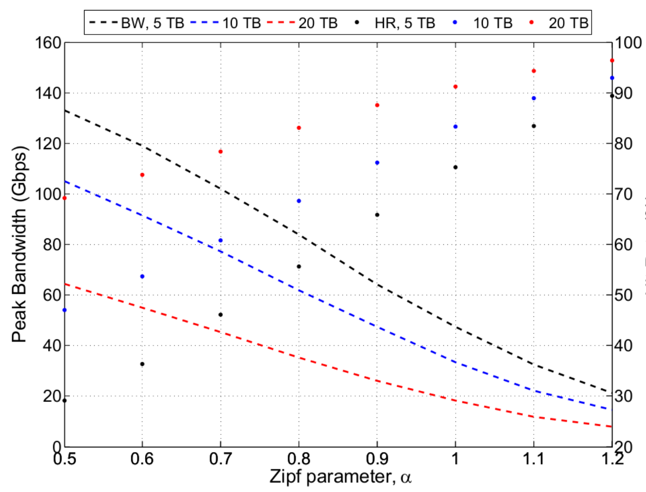

Fig. 4. Illustration of the effect that the Zipf shaping parameter has on performance. The larger α is, the better. Additionally, the required storage to guarantee a given bound on peak bandwidth decreases as α increases. This sheds light on the importance of accurately estimating α for a CDN.

But when $\alpha = 0.8$, we can tolerate 10 TB caches, and when it is 0.9, 5 TB is feasible. Clearly, proper estimation of α is an important design step, so that one can make sure enough storage is provisioned to guarantee a given bound on the peak satellite bandwidth.

B. Cellular

We envision cellular as another application for the proposed overlay. In these networks, the last hop before the end users are cell towers. There are a few hundred thousand such towers in the US. Behind them lies the mobile backhaul, and since this is typically the bottleneck location, the towers are ideal cache injection points. Some carriers claim that they do currently cache content, but to our knowledge, their methods are not available in the public domain.

In this section, we will investigate the efficacy of including different amounts of cache storage at cell towers, complimented with the satellite overlay network. In Table II, we list the cellular parameters we use during simulations. Generally, there are many more towers than headends in the case of VoD, and each tower serves a much smaller number of users (neglecting mobility). By default, we assume 10,000 towers with 250 active users each, though these numbers could be higher or lower depending on the coverage area and carrier. The file sizes are significantly smaller than before, ranging from 24 MB (roughly 10 minutes at 600 kbps) to 72 MB (30 minutes). At the same time, we assume an order of magnitude larger library size of 100,000; though there are many billions of files, the content on the Internet has an extremely heavy tail, so most of these files will rarely be requested. With a library size of 4.8 TB, the cache at each tower is varied from a very small fraction to 25% of the total. Finally, for simulation convenience, we assume all files are globally popular; from our analysis in Figure 3, we do not expect this to affect the hit rate dramatically as opposed to $g = 0.9$.

TABLE II
CELLULAR PARAMETER VALUES ASSUMED IN SIMULATIONS

Parameter	Min	Default	Max
Cell towers, M	100	10,000	64,000
Users per tower, N	-	250	-
Average daily requests/subscriber	-	2	-
Cache storage per headend, S (GB)	10	500	1200
File size, F_i (MB)	24	48	72
Bit rate, B_i (Mbps)	-	0.6	-
Number of files, L	-	100,000	-
Global popularity, g (%)	-	100	-
Global weight, w (%)	-	50	-
Zipf parameter, α	-	0.8	-

We consider simulations along two dimensions: (1) a comparison between multicast and unicast, and (2) scalability with increasing coverage area.

1) Muticast with cellular: First, we will compare satellite-based multicast with fiber-based unicast for cellular, as was done previously for VoD. The number of simulated hours for each scenario is kept the same as before, to get a lower bound on the performance for multicast.

The results are shown in Figure 5. For all values of cache storage tested, the multicast performs significantly better than the unicast. The minimum observed improvement, at 10 GB, is a reduction in bandwidth from 43 Gbps to 16.5 Gbps (62%), while the maximum, at 1.2 TB, is from 42 to 6.54 Gbps (84%).

Notice that for all values tested, the unicast does not scale with increasing storage. This is contrary to the case of VoD in Figure 2, where it did to some extent up to 10 TB. The reason for this is that each tower sees many less requests in a given amount of time than does a headend, and hence the cache is based on a smaller sample size. For cellular, irrespective of the cache storage, the smallest possible bandwidth via unicast is 42 Gbps with a hit rate of under 10%. The cache size may as well remain small.

Though the multicast scales with increased storage, there is diminishing returns. But this is expected, for two reasons. First, the bandwidth is already reasonably low, reaching 10 Gbps at 600 GB. Second, the hit rate is already more or less maximized. We can see this second point by comparing the multicast hit rate to the ideal plot in Figure 5. The ideal trace is obtained by taking points from the cumulative distribution function of the Zipf distribution, which gives the maximum possible hit rate, attainable only when each cache is filled with the most popular files. The multicast hit rate visibly achieves the ideal case for all values of cache storage, only tapering off slightly at 1.0 and 1.2 TB.

2) Increasing coverage area: We will now investigate the effect of the coverage area on the peak bandwidth required. Such a question is particularly important in the case of cellular, where the coverage can vary significantly.

In Figure 6, the peak bandwidth is plotted as the number of towers is varied. For multicast, three different cache sizes are shown, and a unicast trace is included as a comparison. We show the horizontal axis on a log scale, to better depict

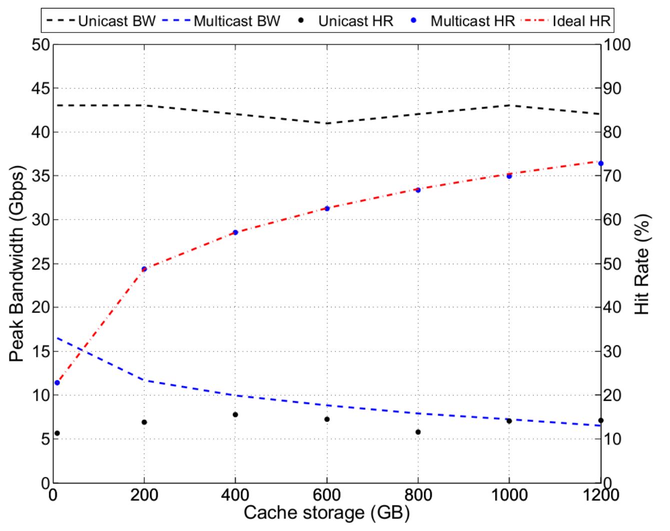

Fig. 5. Comparison between multicast and unicast solutions for the cellular topology. We see that for all values of storage, the multicast significantly outperforms the unicast, and achieves the maximum possible hit rate for a wide range of cache sizes.

the range of towers considered. The exponential characteristic of each trace indicates that the peak bandwidth varies *linearly* with the number of towers. Notice that for multicast, a larger cache size reduces the rate of increase in peak bandwidth. For instance, at 500 GB, if we start with 32,000 towers, doubling the coverage area increases the peak bandwidth from 29 Gbps to 55 Gbps (26 Gbps), whereas at 1.5 TB, it only increases from 17 Gbps to 33 Gbps (16 Gbps).

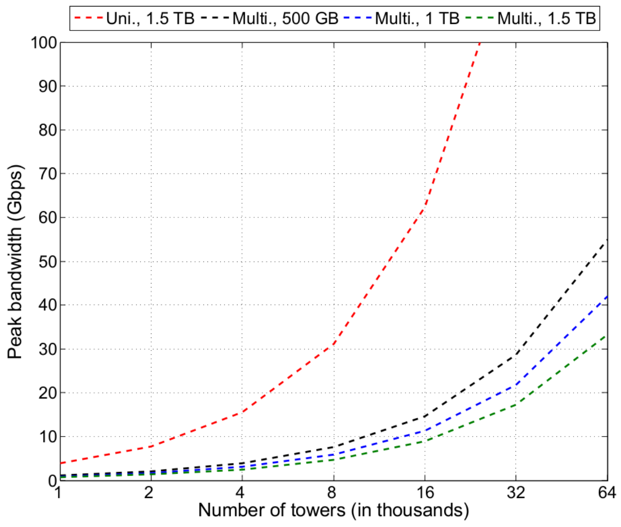

Fig. 6. Plot of the peak bandwidth as the coverage area of the CDN is increased, for different amounts of cache storage. Note the horizontal axis is on a log scale, implying a linear relationship between bandwidth and coverage area. The rate of increase becomes smaller as the cache size increases.

The linear nature of the data in Figure 6 was verified through linear regression, which returned a correlation coefficient $R > 0.999$ for each trace. We quantify the incremental multicast bandwidth per tower in Table III, giving the extra bandwidth required when 1000 towers are added to the coverage area. Increasing the cache size from 100 GB to 1.5 TB results in a 56% reduction in the incremental bandwidth. This highlights the scalability of multicast with increasing cache size, whereas unicast cannot leverage the extra storage.

TABLE III
INCREASE IN MULTICAST BANDWIDTH REQUIRED FOR EVERY 1000 ADDITIONAL TOWERS, FOR DIFFERENT CACHE SIZES.

Cache size (GB)	ΔGbps/1000 towers
100	1.18
500	0.856
1000	0.653
1500	0.516

C. Summary

We presented a wide range of results in this section. We can draw four main conclusions.

1) *Satellite–based multicast scales well with cache storage, while terrestrial–based unicast does not*. The maximum improvement from latter to former was observed to be over 80% for both the cellular and VoD topologies.
2) *With satellite, the incremental bandwidth for increasing coverage area drops considerably with cache storage*. In the cellular topology, a 45% reduction was observed for an order of magnitude increase in cache size.
3) *Incorporating a combination of local and global popularity into the cache update yields the best performance*. For the VoD topology, the observed hit rate improved linearly from 50% to 70% with increasing global fraction g.
4) *Knowledge of the Zipf shaping parameter α is an important design consideration when provisioning for peak bandwidth and cache storage*. For the VoD topology, the required storage for a 60 Gbps peak was observed to halve as α was increased from 0.5 to 0.8.

V. Related Work

Satellite-based CDNs have been investigated in the past. Specifically, there was a push for them in literature in the early 2000s. Rodriguez and Biersack studied a web cache-satellite distribution system, and quantified probabilistic upper-bounds on performance in terms of hit rate [19]. Their analysis is based on an unbounded cache size, where each accrues content as it is broadcasted via satellite. While the notion of an unbounded cache is perhaps feasible for storing webpages on the order of tens of KB each, it is certainly not with the advent of multimedia data. Around the same time, Armon and Levy similarly studied a Cache Satellite Distribution System (CSDS) for web content, where they relaxed the assumption of an unbounded cache size [16]. They quantified the notion of "cache contamination" in terms of items stored that are not likely to cause cache hits. But their analysis is based on an LRU caching scheme, and as we showed in Figure 3, LRU schemes are not suitable for the CDN parameters we investigated.

To our knowledge, we are the first to revive the notion of satellite for contemporary CDNs. The closest work we find to this was done by Galluccio et al., who studied satellite-caching for the perceived future of content-centric networks [15]. They present a strong analysis, and model user and content profiles to draw geographical correlations that are used for cache update. But to our knowledge, information-centric networks have not been deployed yet. Further, we question the applicability of a radially-decaying geographic popularity to different types of CDNs, as well as the ability of their modified LRU scheme to adapt to temporally changing profiles.

Beyond satellite-based solutions, there has been a plethora of research on caching strategies for CDNs. The first discussion on caching for a 3G cellular network was performed by Ehrman et al., in the context of HTTP traffic, where much consideration is given to webpage expiration [9]. Moreover, infinite cache sizes are assumed. Again, we do not see this as a feasible assumption for multimedia CDNs, and a largely increasing fraction of cellular data is video. In the context of VoD, Carbunar et al. recently advocated a "network-aware" caching strategy, whereby proxy servers would make caching decisions based on expected storage penalties, and fetch from one another upon cache misses for load-balancing [14]. Despite the advantages offered by proxy peering relationships, we believe such a network could benefit from a satellite overlay, in terms of offloading high-bandwidth content and retaining a global network view.

Finally, a recent work by Borst et al. considered distributed caching by analyzing the tree-like distribution structure of CDNs, and showed the network bandwidth reductions that could be obtained when defining caching clusers in various levels of the network [20]. In the future, we would like to formulate cache injection as a placement problem to be solved via optimization, similar to what they have done, and combine it with our satellite approach.

VI. Conclusions and Future Work

In this paper, we have presented an intelligent satellite-based multicast and caching overlay for existing CDNs. We have evaluated the potential benefits of multicasting via satellite through simulations on two case studies. Our results show that multicasting has the potential to provide significant bandwidth reductions from terrestrial-based unicast solutions. Additionally, it is scalable in terms of both cache storage and coverage area of the CDN.

Beyond the scope of this paper, we have a plethora of future work planned. First, there is an effort to obtain actual network traffic data to support additional simulations. Such realistic network data would greatly strengthen our claims, and add a realistic, temporal aspect to our analysis.

Second, we will investigate optimization methods along two caching dimensions: injection placement, and update algorithm. Both of these can assist in maximizing overall performance of our hybrid network. Competitive analysis of different schemes would also be possible with realistic network data.

Third, we will develop intelligent routing protocols for directing desired traffic to the overlay, as well as for multicasting. This would likely require estimating the average inter-arrival times for different files, and defining a threshold below which a file would be directed to multicast, and above which it would be sent via unicast. Additionally, two different tiers of service could be defined: "on-demand", which would be serviced terrestrially over fiber, and "delay-tolerant", which would be served via satellite. This could lead to an effective, two-tiered pricing system for service operators.

In the coming months, we plan to investigate the above three topics. Overall, we believe that our solution identifies a new market opportunity for satellite, as it shows promise of enabling scalability for CDNs in the future.

References

[1] Cisco Visual Networking Index: Forecast and Methodology, 2011 - 2016. Cisco, 2012.

[2] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012 - 2017. Cisco, 2013.

[3] Jon Brodkin. Bandwidth Explosion: As Internet use soars, can bottlenecks be averted? http://arstechnica.com/business/2012/05/bandwidth-explosion-as-internet-use-soars-can-bottlenecks-be-averted/. 2012.

[4] Daniel Venmani and Djamal Zeghlache. Demystifying Link Congestion in 4G-LTE Backhaul using OpenFlow. *IEEE NMTS*, 2012.

[5] Rob Bamforth. Mobile data congestion: Four ways to tackle the coming capacity crunch. http://www.techrepublic.com/blog/european-technology/mobile-data-congestion-four-ways-to-tackle-the-coming-capacity-crunch/204. 2012.

[6] Gregg Levin. The Economics of Gigabit 4G Mobile Backhaul. Bridge-Wave Communications, 2008.

[7] Jian Ni and Danny H. K. Tsang. Large-Scale Cooperative Caching and Application-Level Multicast in Multimedia Content Delivery Networks. *IEEE Communications Magazine*, 2005.

[8] Aditya Kishore. Operator CDNs: Making OTT Video Pay. *Heavy Reading*, 2011.

[9] Jeffrey Erman, et al. To Cache or Not to Cache: The 3G Case. *IEEE Internet Computing*, 2011.

[10] Meeyoung Cha, et al. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. *ACM IMC*, 2007.

[11] Meeyoung Cha, Pablo Rodriguez, Jon Crowcroft, Sue Moon, and Xavier Amatriain. Watching Television Over an IP Network. *ACM IMC*, 2008.

[12] Frederic Thouin and Marc Coates. Video-on-Demand Networks: Design Approaches and Future Challenges. *IEEE Network*, 2007.

[13] Gorry Fairhurst, Luca Caviglione, Bernhard Collini-Nocker. FIRST: Future Internet - A Role for Satellte Technology. *IEEE IWSSC*, 2008.

[14] Bogdan Carbunar, et al. Predictive Caching for Video on Demand CDNs. *IEEE Globecom*, 2011.

[15] Laura Galluccio, Giacomo Morabito, Sergio Palazzo. Caching in information-centric satellite networks. *IEEE ICC*, 2012.

[16] Aner Armon and Hanoch Levy. Cache Satellite Distribution Systems: Modeling and Analysis. *IEEE INFOCOM*, 2003.

[17] Utpal Paul, et al. Understanding Traffic Dynamics in Cellular Data Networks. *IEEE INFOCOM*, 2011.

[18] Cell Phone Tower Statistics. http://www.statisticbrain.com/cell-phone-tower-statistics/.

[19] Pablo Rodriguez and Ernst W. Biersack. Bringing the Web to the Network Edge: Large Caches and Satellite Distribution. *Mobile Networks and Applications*, 2002.

[20] Sem Borst, Varun Gupta, Anwar Walid. Distributed Caching Algorithms for Content Distribution Networks. *IEEE INFOCOM*, 2010.