Distributing Content Simplifies ISP Traffic Engineering

Abhigyan Sharma[†] Arun Venkataramani[†] Ramesh K Sitaraman[†⋆]

[†] University of Massachusetts Amherst [⋆] Akamai Technologies

UMASS Computer Science Technical Report: UM-CS-2012-002

ABSTRACT

Several major Internet service providers today also offer content distribution services. The emergence of such "network-CDNs" (NCDNs) is driven both by market forces as well as the cost of carrying ever-increasing volumes of traffic across their backbones. An NCDN has the flexibility to determine both where content is placed and how traffic is routed within the network. However NCDNs today continue to treat traffic engineering independently from content placement and request redirection decisions. In this paper, we investigate the interplay between content distribution strategies and traffic engineering and ask whether or how an NCDN should address these concerns in a joint manner. Our experimental analysis, based on traces from a large content distribution network and real ISP topologies, shows that realistic (i.e., history-based) joint optimization strategies offer little benefit (and often significantly underperform) compared to simple and "unplanned" strategies for routing and placement such as InverseCap and LRU. We also find that the simpler strategies suffice to achieve network cost close to those of a joint-optimal strategy with future knowledge.

1. INTRODUCTION

Content delivery networks (CDNs) today provide a core service that enterprises use to deliver web content, downloads, streaming media, and IP-based applications to a global audience of their end-users. The traditional and somewhat simplified, tripartite view of content delivery involves three sets of entities as shown in Figure 1. The *content providers* (e.g., media companies, news channels, e-commerce providers, software distributors, enterprise portals, etc.) produce the content and wish to provide a high-quality experience to end-users accessing their content over the Internet. The *networks* (e.g., telcos such as AT&T, MSOs such as Comcast, and traditional ISPs) own the underlying network infrastructure and are responsible for provisioning capacity and managing traffic flowing through their networks. Finally, the *CDNs* (e.g., Akamai, Limelight) are responsible for optimizing content delivery to end-users on behalf of the content providers, residing as a global, distributed overlay service on top of the networks.

Recent powerful trends are reshaping the simplified tripartite view of content delivery. A primary driver is the torrid growth of video [17, 6] and downloads traffic on the Internet. For example, a single, popular TV show with 50 million viewers, each viewer watching an HD-quality stream of 10 Mbps, generates 500 Tbps of network traffic! The increasing migration of traditional media content to the Internet and

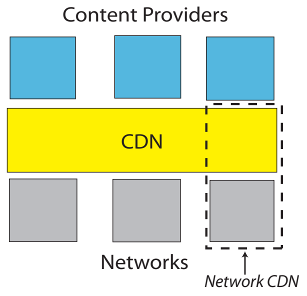


Figure 1: A tripartite view of content delivery.

the consequent challenges of scaling the network backbone to accommodate that traffic has necessitated the evolution of *network CDNs* (or NCDNs)[1] that vertically integrate CDN functionality such as content caching and redirection with traditional network operations (refer Figure 1). A second economic driver of NCDNs is the desire of networks to further monetize the "bits" that flow on their infrastructure by contracting directly with content providers. Finally, NCDNs also enable better performance for their own end-user subscribers and open up avenues for new, differentiated services (e.g., Verizon's recent offering that delivers HBO's content to FIOS subscribers [22].

The evolution of NCDNs significantly changes traditional engineering concerns as NCDNs must make decisions about content placement and request redirection in addition to the underlying network routing. As NCDNs own both the content distribution and network infrastructure, they are in a powerful position to place content in a manner that "shapes" the traffic demand to their advantage, potentially enabling traffic engineering to achieve a significantly lower cost. However, NCDNs today treat content distribution and traffic engineering concerns separately, perhaps because it is easier to continue doing things as they were being done. This disparity raises several research questions such as: (1) How do content demand patterns and placement strategies impact traffic engineering objectives? (2) How should an NCDN jointly determine placement and routing decisions so as optimize network cost? (3) How do demand-aware strategies (i.e., using knowledge of recently observed demand patterns or hints about anticipated future demands) for placement and routing compare with demand-oblivious strategies?

[1]NCDNs are sometimes referred to as Telco CDNs, or Carrier CDNs. Further, they are referred to as a Licensed CDN when a pure-play CDN such as Edgecast[5] licenses the CDN software to a network to create an NCDN.

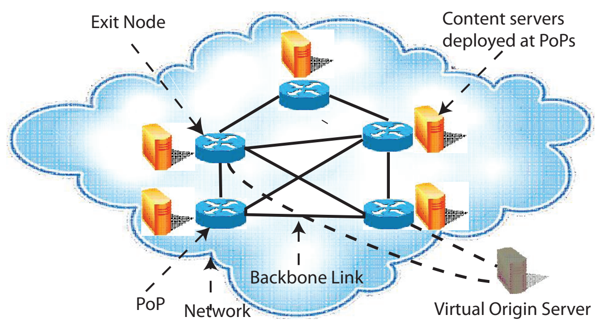


Figure 2: NCDN Architecture

Our primary contribution is to empirically analyze the above questions for realistic content demand workloads and ISP topologies. To this end, we collect content request traces from Akamai, the world's largest CDN today. We focus specifically on on-demand video and large-file downloads traffic as they are two categories that dominate overall CDN traffic and are significantly influenced by content placement strategies. Our combined traces consist of a total of 28.2 million requests from 7.79 million unique users who downloaded a total of 1455 Terabytes of content across the US over multiple days. Our trace-driven experiments using these logs and realistic ISP topologies reveal the following somewhat surprising conclusions:

- Simple demand-oblivous schemes for placement and routing (such as Least Recently Used and InverseCap) significantly outperform (by 2.2× to 17×) a joint-optimal placement and routing strategy with knowledge of the previous day's demand[2].
- Traffic demand can be "shaped" by effective content placement so that traffic engineering, i.e., optimizing routes with knowledge of recent traffic matrices, yields little improvement in cost compared to demand-oblivious routing (InverseCap) in conjunction with any reasonable placement.
- A demand-oblivious placement and routing is at most 4% sub-optimal compared to a joint-optimal placement and routing with perfect knowledge of the next day's demand at higher storage ratios (≈ 4) with simple optimizations such as content chunking and link-utilization-aware redirection.

In the rest of this paper, we first provide an overview of the NCDN architecture highlighting why it changes traditional traffic engineering concerns (§2). Next, we formulate algorithms that jointly or individually optimize content placement and routing as optimization problems (§3). We then describe how we collected real CDN traces (§4) and then evaluate our algorithms using these traces and real ISP topologies (§5). Finally, we present related work (§6) and conclusions (§7).

2. BACKGROUND AND MOTIVATION

A typical NCDN architecture, as shown in Figure 2, resembles the architecture of a global CDN but with some important differences. First, the content servers are deployed

[2]We use the term "optimal" because placement and routing is calculated by solving an optimization problem.

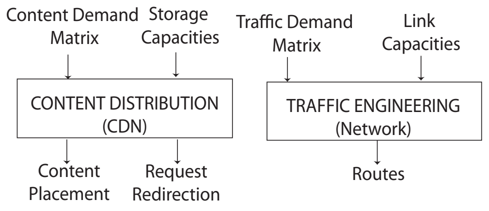


Figure 3: Traditional formulation with content distribution and traffic engineering optimized separately.

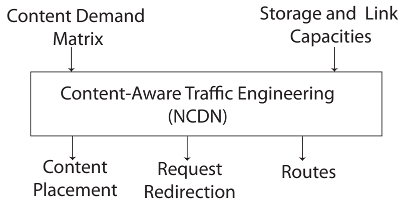

Content Demand Matrix
Storage and Link Capacities
Content-Aware Traffic Engineering (NCDN)
Content Placement
Request Redirection
Routes

Figure 4: Our new formulation of content-aware traffic engineering for NCDNs as a joint optimization.

at points-of-presence (PoPs) within the network rather than globally across the Internet as the NCDN is primarily interested in optimizing content delivery for its own customers and end-users. Second, and more importantly, a NCDN owns and manages the content servers as well as the underlying network. Content providers that purchase content delivery service from the NCDN publish all of their content to *origin servers* that they maintain external to the NCDN itself.

Each PoP is associated with a distinct set of downstream end-users who request content such as web, video, downloads etc. An end-user's request is first routed to the content servers deployed at the PoP to which the end-user is connected. If a content server at that PoP has the requested content in their cache, it serves that to the end-user. Otherwise, if the requested content is cached at other PoPs in the network, the content is downloaded from a nearby PoP and served to the end-user. If the content is not present in any content server in the network, it is downloaded directly from the content provider's origin servers.

2.1 Why Do NCDNs Change the Game?

Managing content distribution as well as the underlying network infrastructure makes the costs and objectives of interest to an NCDN different from that of a traditional CDN or a traditional ISP. The traditional model of content distribution and traffic engineering as performed by a traditional CDN and a traditional ISP is shown in Figure 3. However, as we elaborate below, we propose a new model appropriate for NCDNs that jointly optimizes content distribution and traffic engineering as shown in Figure 4.

2.1.1 Content Distribution

A traditional CDN has two key decision components—*content placement* and *request redirection*—that seek to optimize the response time perceived by end-users and balance the load across its content servers (see Figure 3). Content placement decides which objects should be cached at which nodes. Note that any particular object may be replicated at multiple nodes in the network or not stored in the network at all and be served from the origin server instead. Request redirection determines which server storing a replica of the requested object is best positioned to serve the request.

Content placement schemes can be classified as *demand-aware* or *demand-oblivious*. A demand-aware scheme takes as input a *content matrix*, i.e., information about the volume of demand for each content at each content server location, and optimizes the placement accordingly. The content matrix is typically learned by monitoring a recent history of system-wide requests potentially in conjunction with any available hints from content providers about anticipated demand for some objects. A demand-aware placement scheme uses a recent content matrix to decide on a placement periodically (say, once a day) but does not alter its placement in between. In contrast, a demand-oblivious placement scheme can continually alter its placement potentially even after every single request. A simple example of a demand-oblivious placement scheme is least-recently-used (LRU), where each node serves the requested object and adds it to its cache evicting previously stored objects if necessary in LRU order in order to make room for the requested object.

2.1.2 Traffic Engineering

A key component of ISP network operations is traffic engineering, which seeks to route the traffic demands through the backbone network so as to balance the load and mitigate hotspots. A common view of traffic engineering is as a routing problem that takes as input a *traffic matrix*, i.e., the aggregate flow demand between every pair of PoPs observed over a recent history, and computes routes so as to minimize a network-wide cost objective (see Figure 3). The cost seeks to capture the severity of load imbalance in the network and common objective functions include the maximum link utilization (MLU) or a convex function (so as to penalize higher utilization more) of the link utilization aggregated across all links in the network [10]. Networks commonly achieve the computed routing either by using shortest-path routing (e.g., the widely deployed OSPF protocol [10]) or by explicitly establishing virtual circuits (e.g., using MPLS [9]). The former requires additionally engineering a set of link weights such that using shortest paths achieves the desired routing, while the latter is more flexible and can in principle achieve any desired routing including those that split traffic between a given pair of PoPs across many paths in arbitrary ratios.

Routing can also be classified as *demand-aware* or *demand-oblivious* similar in spirit to content placement. Traffic engineering schemes as explained above (as well as online traffic engineering schemes [16] that are rarely deployed today) are implicitly demand-aware as they optimize routing for recently observed demand. In contrast, demand-oblivious routing schemes rely upon statically configured routes, e.g., Inverse Cap that is a shortest-path routing scheme that simply uses the inverse of capacity as link weights and is a common default scheme in commercial routers. More sophisticated demand-oblivious routing schemes [3, 4] seek to compute a static flow-splitting strategy that works reasonably well for all possible traffic demands (though it may be sub-optimal for most or all of them).

2.1.3 Content-aware Traffic Engineering

An NCDN can perform content-aware traffic engineering by leveraging content distribution to achieve traffic engineering goals such as network cost minimization. Unlike traditional ISPs, an NCDN can place content and redirect requests in a manner that "shapes" the traffic demands to its advantage and thereby achieve significantly lower network cost.

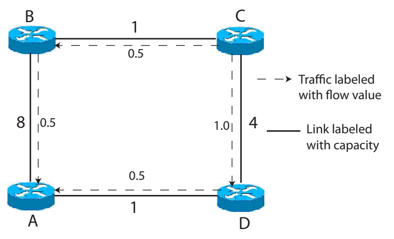


Figure 5: A Simple NCDN Example

A central thesis of this paper is that intelligent content placement and request redirection achieve significant cost reduction for NCDNs and thereby marginalize the role of traditional traffic engineering. To appreciate this point, consider the simple, illustrative example in Figure 5. Node C has an object in its cache that is requested by end-users at nodes A and D. Suppose that one unit of traffic needs to be routed from C to A and 0.5 units from C to D to satisfy the demand for that object. The routing that achieves the minimum MLU of 0.5 to serve the demanded object is shown in the figure. Note that the routing that achieves the MLU of 0.5 is not possible with a simple, demand-oblivious protocol like InvCap as that would route all the traffic demand from C to A via B, resulting in an MLU of 1. Thus, a (demand-aware) traffic engineering scheme is necessary to achieve an MLU of 0.5.

On the other hand, content-aware traffic engineering can shape the traffic demand matrix by using a judicious placement and redirection strategy. Suppose that there is some space left in the content server's cache at node B to accommodate an additional copy of the demanded object. By creating an additional copy of the object at B, the traffic demand of A can be satisfied from B and the demand of D from C achieving the an MLU of 0.125. In this case, judicious content placement decreased the MLU by a factor of 4. Even more interestingly, this best MLU can be achieved using a simple routing scheme like InvCap. Although this is clearly a "toy" example, it illustrates the sophisticated interaction between content placement and traffic engineering and potential opportunities for NCDNs to both reduce cost and simplify traffic engineering by doing it in a content-aware manner.

3. CONTENT-AWARE TRAFFIC ENGINEERING STRATEGIES

In this section, we formalize the NCDN model and the problem of determining the optimal content placement and routing strategy as a mixed integer program (MIP). We also present two variants of this formulation: (1) a MIP that determines the optimal placement strategy for a given routing scheme; and (2) a linear program that optimizes routing for a given placement strategy. All of these formulations take as input a content matrix, i.e., the demand for each piece of content at each network point-of-presence (PoP), and seek to compute a placement and/or a routing strategy that minimizes the maximum link utilization (MLU) while respecting

Input variables and descriptions	
V	Set of nodes where each node represents a PoP
E	Set of edges where each link represents a communication link
o	Virtual origin node that hosts all the content in K
X	Set of exit nodes in V
D_i	Disk capacity at node $i \in V$ (in bytes)
C_e	Capacity of link $e \in E$ (in bits/sec)
K	the set of all content accessed by end-users
S_k	Size of content $k \in K$.
T_{ik}	Demand (in bits/sec) at node $i \in V$ for content $k \in K$
Decision variables and descriptions	
α	MLU of the network
z_k	Binary variable indicating whether one or more copies of content k is placed in the network
x_{jk}	Binary variable indicating whether content k is placed at node $j \in V \cup \{o\}$
f_{ij}	Total traffic from node j to node i
f_{ije}	Traffic from node j to node i crossing link e.
t_{ijk}	Traffic demand at node $i \in V$ for content $k \in K$ served from node $j \in V \cup \{o\}$

Table 1: List of input and decision variables for the NCDN problem formulation.

link capacity and storage constraints.

3.1 NCDN Model

Table 3 lists all the input parameters and variables used in this model. An NCDN consists of a set of nodes V where each node represents a PoP in the network. The nodes are connected by a set of directed edges E that represent the communication links provisioned between PoPs in the network backbone. The set of content requested by end-users is represented by the set K. For instance, content $k \in K$ could represent either an on-demand video that can be viewed or a file that can be downloaded by the end-user. The primary resource constraints are the link capacities $C_e, e \in E$, and the amount of storage $D_i, i \in V$, available at the nodes. We implicitly assume that the content servers at the PoPs in the NCDN have adequate compute resources to serve locally stored content.

A *content matrix* specifies the demand for each content at each node. An entry in this matrix, $T_{ik}, i \in V, k \in K$, denotes the demand (in bits/second) for content k at node i. The content matrix is assumed to be measured by the NCDN a priori over a coarse-grained interval, e.g., the previous day, by monitoring the request rate for content at the PoPs and combining them with the known content sizes $S_k, k \in K$. The infrastructure required for this measurement is comparable to what ISPs have in place for monitoring traffic matrices today.

All content is published by content providers of the NCDN service and initially stored at a set of origin servers owned and maintained by the content providers. We assume that origin servers are hosted external to the NCDN. These origin servers are typically mirrored across different data centers and multihomed across different networks. For simplicity, we model a single virtual origin node o external to the NCDN that can be reached from any node $i \in V$ by routing to the closest exit node $x \in X$, where $X \subset V$ is the set of all exit nodes in the NCDN (See Figure 2). Since we are not concerned with traffic engineering links outside the NCDN, we model the edges (x, o), for all $x \in X$, as having infinite capacity. The virtual origin node o always maintains a copy of all the requested content. Nodes in the network may additionally store local copies of some of the content. A request for a content is served from the virtual origin node only if no copy of the content is stored at any node in V of the NCDN. In this case, the request is assumed to be routed to the virtual origin via the exit node closest to the node where the request was made (in keeping with the commonly practiced *early-exit* or *hot potato* routing policy).

ISP networks carry transit traffic in addition to NCDN traffic, which can be represented as a transit traffic matrix (TTM). Each entry in the TTM contains the volume of transit traffic between two PoPs in the network.

3.2 Optimal Strategy as a MIP

The content-aware traffic engineering problem for NCDNs (NCDN problem, for short) seeks to compute a placement and routing strategy that minimizes the MLU and satisfies the demands specified by the content matrix while respecting link capacity constraints and storage constraints at the nodes. This optimization goal can be formulated as a mixed integer program (MIP). Unlike the traditional traffic engineering problem that can be formulated as a multi-commodity flow problem and solved using a linear program, the NCDN problem needs to make binary placement decisions, i.e., whether or not to place a content at a PoP, and then route the demand accordingly. These placement decision variables (denoted $x_{jk}, j \in V, k \in K$) as well as other decision variables required for the MIP are listed in Table 3. The MIP to minimize the MLU α is as follows:

$$\min \alpha \tag{1}$$

$$\text{s. t.} \sum_{j \in V} t_{ijk} + t_{iok} = T_{ik}, \quad \forall k \in K, i \in V \tag{2}$$

$$\sum_{k \in K} t_{ijk} = f_{ij}, \ \forall j \in V - X, i \in V \tag{3}$$

$$\sum_{k \in K} t_{ijk} + \sum_{k \in K} \delta_{ij} t_{iok} = f_{ij}, \ \forall j \in X, i \in V \tag{4}$$

where δ_{ij} is 1 if j is the closest exit node to i and 0 otherwise. Note that δ_{ij} is not a variable but a constant that is determined by the topology of the network, and hence constraint (4) is linear.

$$\sum_{p \in P(l)} f_{ijp} - \sum_{q \in Q(l)} f_{ijq} = \begin{cases} f_{ij} & \text{if } l = i, \\ -f_{ij} & \text{if } l = j, \\ 0 & \text{otherwise,} \end{cases} \quad \forall i, j, l \in V \tag{5}$$

where $P(l)$ and $Q(l)$ respectively denote the set of outgoing and incoming links at node l.

$$\sum_{i \in V, j \in V} f_{ije} \leq \alpha \times C_e, \quad \forall e \in E \tag{6}$$

$$\sum_{k \in K} x_{ik} S_k \leq D_i, \quad \forall i \in V \tag{7}$$

$$x_{ok} = 1, \quad \forall k \in K \tag{8}$$

$$\sum_{i \in V} x_{ik} \geq z_k, \quad \forall k \in K \tag{9}$$

$$x_{ik} \leq z_k, \quad \forall k \in K, i \in V \tag{10}$$

$$t_{ijk} \leq x_{jk} T_{ik}, \quad \forall k \in K, i \in V, j \in V \cup \{o\} \tag{11}$$

$$t_{iok} \leq T_{ik}(1 - z_k), \quad \forall k \in K \tag{12}$$

$$x_{jk},\ z_k \quad \in \quad \{0,1\}, \quad \forall j \in V, k \in K$$
$$f_{ije},\ t_{ijk},\ t_{iok} \quad \geq \quad 0, \quad \forall i, j \in V, e \in E, k \in K$$

The constraints have the following rationale. Constraint (2) specifies that the total traffic demand at each node for each content must be satisfied. Constraints (3) and (4) specify that the total traffic from source j to sink i is the sum over all content k of the traffic from j to i for k. Constraint (5) specifies that the volume of a flow coming in must equal that going out at each node other than the source or the sink. Constraint (6) specifies that the total flow on a link is at most α times capacity. Constraint (7) specifies that the total size of all content stored at a node must be less than its disk capacity. Constraint (8) specifies that all content is placed at the virtual origin node o. Constraints (9) and (10) specify that at least one copy of content k is placed within the network if $z_k = 1$, otherwise $z_k = 0$ and no copies of k are placed at any node. Constraint (11) specifies that the flow from a source to a sink for some content should be zero if the content is not placed at the source (i.e., when $x_{jk} = 0$), and the flow should be at most the demand if the content is placed at the source (i.e., when $x_{jk} = 1$). Constraint (12) specifies that if some content is placed within the network, the traffic from the origin for that content must be zero. Updating the content placement itself generate traffic and impacts the MLU in the network. A formal description of the corresponding constraints is deferred to the Appendix.

Finally, a simple extension to this MIP presented in the Appendix jointly optimizes routing given a TTM as well a CM. We have presented a CM-only formulation here as our findings (in §5) show that a joint optimization of the CM and TTM is not useful for NCDNs.

3.2.1 *Computational hardness*

Opt-NCDN is the decision version of the NCDN-problem. The proofs for these theorems are included in Appendix A.

Theorem 1 *Opt-NCDN is NP-Complete even in the special case where all objects have unit size, all demands have binary values, and link and storage capacities have binary values.*

Theorem 2 *Opt-NCDN is inapproximable within a factor β for any $\beta > 1$ unless P = NP.*

3.3 Partial Optimization Strategies

As one of our goals is to analyze the relative importance of optimizing placement and routing, we introduce two heuristic variants of the optimization formulation above. The first variant optimizes content placement for any given (possibly suboptimal) strategy for routing between nodes of the network (e.g., Inverse Cap [11]). The second variant optimizes routing for any given placement strategy.

Optimal Content Placement with Fixed Routing: This problem can be solved with a straightforward modifications to the MIP introduced above. Assume that the given (fixed) routing strategy is specified in terms of the variables $r_{ije}, 0 \leq r_{ije} \leq 1$, for $i, j \in V$ and $e \in E$, that represents the fraction of flow f_{ij} that flows on the link $e \in E$. Assuming that the specified routing is valid, constraint (5) of the MIP that enforces the conservation of flow is no longer needed and can be removed. Further, all variables f_{ije} can be replaced in the MIP with by $r_{ije} f_{ij}$, resulting in a reformulation of the LHS of constraint (6) of the MIP.

Optimal Routing with Fixed Content Placement: If the placement is fixed, then a linear program (LP) suffices to compute the optimal routing instead of the MIP above. This is because the integer variables x_{jk} are no longer necessary. Since the placement is known, we know the values of the variables, x_{jk} and by consequence z_k. Substituting these values and removing constraints (7)–(10) from the MIP results in the corresponding LP.

Note that for a demand-oblivious placement strategy such as, LRU, we model route optimization as a multi-commodity flow problem (also an LP) identical to the traditional traffic engineering problem [10]. We assume that the NCDN measures the traffic matrix over the immediately preceding monitoring interval and computes routes that optimize the MLU for that matrix. The matrix incorporates the effect of the demand-oblivous placement strategy and the implicit assumption is that the content demand and demand-oblivous placement strategy result in a traffic matrix that does not change dramatically from one monitoring interval to the next—an assumption that also underlies traffic engineering as practiced by ISPs today.

3.4 Approximation Techniques

As solving the MIP for very large problem scenarios is computationally infeasible, we use two approximation techniques to tackle such scenarios.

The first is a two-step local search technique. In the first step, we "relax" the MIP by allowing the integral variables x_{jk} and z_k to take fractional values between 0 and 1. This converts a MIP into an LP that is more easily solvable. Note also that the optimal solution of the relaxed LP is a lower bound on the optimal solution of the MIP. However, the LP solution may contain fractional placement of some of the content with the corresponding x_{jk} variables set to fractional values between 0 and 1. However, in our experiments only about 20% of the variables in the optimal LP solution were set to fractional values between 0 or 1, and the rest took integral values of 0 or 1. In the second step, we search for a valid solution for the MIP in the local vicinity of the LP solution by substituting the values for variables that were set to 0 or 1 in the LP solution, and re-solving the MIP for the remaining variables. Since the number of variables in the second MIP is much smaller, it can be solved more efficiently than the original MIP.

The second approximation technique is to reduce the number of unique content in the optimization problem in order to make it tractable. To this end, we use two strategies. First, we discard the tail of unpopular content prior to optimization. While the discarded portion accounts for only 1% of all requests, this simple technique reduces the number of content by 50% or more in our traces. Second, we sample content from the trace and, in our experiments, select trace entries corresponding only to the sampled content. These approximations reduce the number of content from tens of thousands to less than 5000. An MIP of this size can be solved using local search within an hour by a standard LP solver [13] for the ISP topologies in our experiments.

4. DATA COLLECTION

To conduct a realistic simulation of end-users accessing content on an NCDN, we collected extensive traces for different traffic types from one of the world's largest CDNs as described in §4.1. Then, we used the topologies for two ISP networks as described in §4.2.

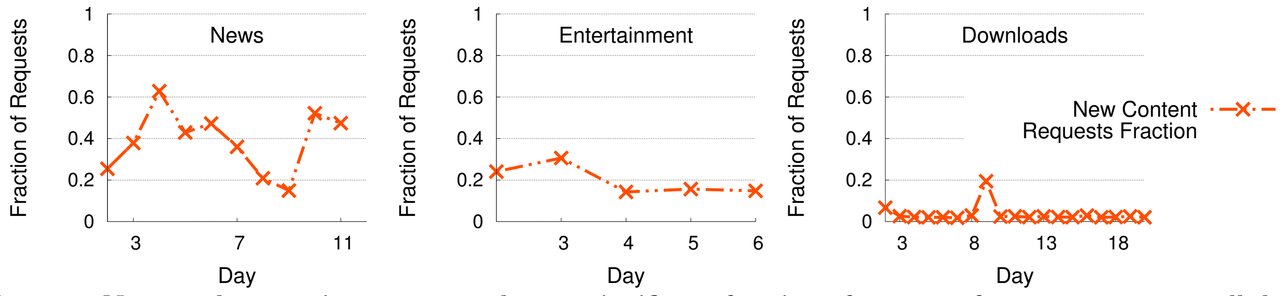


Figure 6: News and entertainment traces have a significant fraction of requests for new content on all days. Downloads trace has a small fraction of requests for new content on all days except one day.

4.1 Akamai CDN Traces

We collected traces for the two major sources of CDN traffic that we describe in turn.

Video Traces. Videos are the primary source of traffic on a CDN and is growing at a rapid rate. In addition, many NCDNs are likely to be disproportionately focused on video due to the attractive business model of Telcos delivering on-demand video to their subscribers [22]. Our video trace consists of actual end-users accessing on-demand videos on the Akamai network over multiple days. To make the traces as representative as possible, we chose content providers with a whole range of business models, including major television networks, news outlets, and movie portals. The videos in our traces include a range of video types from short-duration video (less than 10 mins) such as new clips to longer duration (30 min to 120 min) entertainment videos representing TV shows and movies. In all, our traces represent a nontrivial fraction of the overall traffic on Akamai's media network and accounted for a total of 27 million playbacks of over 85000 videos, 738 TBytes of traffic, served to 6.59 million unique end-users around the US[2].

The data collection was done at two different time periods. First we collected data from a news outlet for an 11-day period in Sept 2011. The videos in this data set consists mostly of news video clips, but also include a small fraction of news TV shows. We refer to this dataset as the *news trace*. In the second round, we collected data from three content providers for a 6 day period in January 2012. This dataset includes a variety of videos including TV shows, clips of TV shows, movies and movie trailers. We call the second dataset as the *entertainment trace*.

The trace collection mechanism utilized a plugin embedded in the media player that is capable of reporting (anonymized) video playback information. Our traces include a single log entry for each distinct video playback and provides time of access, user id, the location of the user (unique id, city, state, country, latitude, and longitude), the url identifier of the content, the content provider, the total length of the video (in time and bytes), the number of bytes actually downloaded, the playback duration, and the average bitrate over the playback session.

Downloads Traces. The second largest traffic contributor in a CDN is downloads of large files over HTTP. With the software business migrating to the web, numerous enterprises deliver and update their software on the web, e.g.,, Microsoft's Windows update, Apple's iTunes, and Symantec's security updates. Further, music, books, and movies are downloaded through CDNs as well. The large file downloads typically use a client-side software called the download manager [18]. We collected extensive anonymized access data reported from the download manager using Akamai's NetSession interface [14] for a large fraction of content providers across the entire network for a period of a month (December 2010). Our traces represent a nontrivial fraction of the overall US-based traffic on Akamai's downloads network and accounted for a total of 1.2 million downloads, 717 TBytes of traffic, served to 0.62 million unique end-users around the US. Our traces provide a single long entry for each download and provide time of access, user id, location of the user (city, state, country, latitude, and longitude), the url identifier of the content, content provider, bytes downloaded, and file size.

Figure 6 shows the fraction of requests for new content published each day for news, entertainment, and downloads traces. The x-axis shows the day of trace and the y-axis shows the fraction of requests for content published on that day. The news trace has the highest fraction of requests (up to 63%) due to new content because news clips are generated each day and the latest news clips are the most popular videos on the website. The entertainment trace also has up to 31% of requests each day due to new content such as new episodes of TV shows, and the previews of upcoming TV shows. The downloads trace has only 2-3% of all requests due to new content on a typical day. However, on the 9th day of the trace major software updates were released, which were downloaded on the same day by a large number of users. Therefore, nearly 20% of requests on that day were for newly published content. The fraction of requests for new content impacts the performance of demand-aware placement strategies as we show in §5.

4.2 Network Topologies

We experimented with network topology maps from two US ISPs. First is the Abilene ISP topology [21] and second is is a large tier-1 US ISP topology (referred to as US-ISP).

5. EXPERIMENTAL EVALUATION

We conduct trace-driven experiments to compare different content-aware traffic engineering strategies for NCDNs using the CDN traces and network topologies described above. Our high-level goal is to identify a simple strategy that performs well for a variety of workloads. In addition, we seek to asses the relative value of optimizing content placement versus network routing; the value of being demand-aware versus being demand-oblivious; and the value of having future knowledge of demand. A summary of our findings is as

[2]Since we only had US-based network topologies, we restricted ourselves to US-based traffic.

follows.

- A simple combination of demand-oblivious placement (LRU) and demand-oblivious routing (InvCap) significantly outperforms (by 2.2× to 17×) a joint-optimal placement and routing with knowledge of the previous day's demand.
- Traffic engineering, i.e., routing optimized with knowledge of recent traffic matrices, yields little improvement in network cost compared to demand-oblivious routing (InvCap) in conjunction with any reasonable placement strategy.
- A demand-oblivious placement and routing is at most 4% sub-optimal compared to a joint-optimal placement and routing with perfect knowledge of the next day's demand at higher storage ratios (≈ 4) with simple optimizations such as content chunking and link-utilization-aware redirection.
- Simple hybrid strategies (partly demand-aware and partly demand-oblivious) do not improve upon a completely demand-oblivious strategy.

5.1 Simulation Methodology

In order to realistically simulate end-users accessing content in an NCDN, we combine the CDN traces with ISP topologies in §4 as follows. We map each content request entry in the Akamai trace to the geographically closest PoP in the ISP topology in the experiment (irrespective of the real ISP that originated the request). Each PoP has a content server as in Figure 2, and the request is served locally, redirected to the nearest (by hop-count) PoP with a copy, or to the origin as needed.

MLU computation. We compute the traffic that flow through each link periodically. To server a requested piece of content from a PoP s to t, we update the traffic induced along all edges on the path(s) from s to t as determined by the routing protocol using the bytes-downloaded information in the trace. To compute the MLU, we partition simulation time into 5-minute intervals and compute the average utilization of each link in each 5-minute interval. We discard the values of the first day of the trace in order to warm up the caches, as we are interested in steady-state behavior. We then compute our primary metric, which is the 99-percentile MLU, as the 99^{th} percentile of the link utilization over all links and all 5-minute time periods. We use 99-percentile instead of the maximum as the former is good proxy for the latter but with less experiemental noise. Finally, for ease of visualization, we scale the 99-percentile MLU values in all graphs so that the maximum 99-percentile MLU across all schemes in each graph is equal to 1. We call this scaled MLU the *normalized MLU*. Note that only the relative ratios of the MLUs for the different schemes matter and scaling up the MLU uniformly across all schemes is equivalent to uniformly scaling down the network resources or uniformly scaling up the traffic in the CDN traces.

Storage. We assume that storage is provisioned uniformly across PoPs except in §5.3.4 where we analyze heterogenous storage distributions. We repeat each simulation with different levels of provisioned storage. Since the appropriate amount of storage depends on the size of the working set of the content being served, we use as a metric of storage the *storage ratio*, or the ratio of total storage at all PoPs in the network to the average storage footprint of all content accessed in a day for the trace. The total storage across all nodes for a storage ratio of 1 is 228 GB, 250 GB, and 895 GB for news, entertainment and downloads respectively.

Content chunking. We analyze the impact of content chunking in §5.3.2 and §5.3.5. In these experiments, we split videos into chunks of 5 minute duration. The size of a chunk of a video depends on the bitrate of the video, e.g., if bitrate is 2 Mbps, chunk size is 75 MB. For the downloads trace, we similarly split content into chunks of size 50 MB. In our experiments, a demand-aware placement scheme uses the demand of each chunk, instead of the demand for the original content to calculate content placement. Demand oblivious placement (caching) treats each chunk as a distinct content to be either cached or evicted.

5.2 Schemes Evaluated

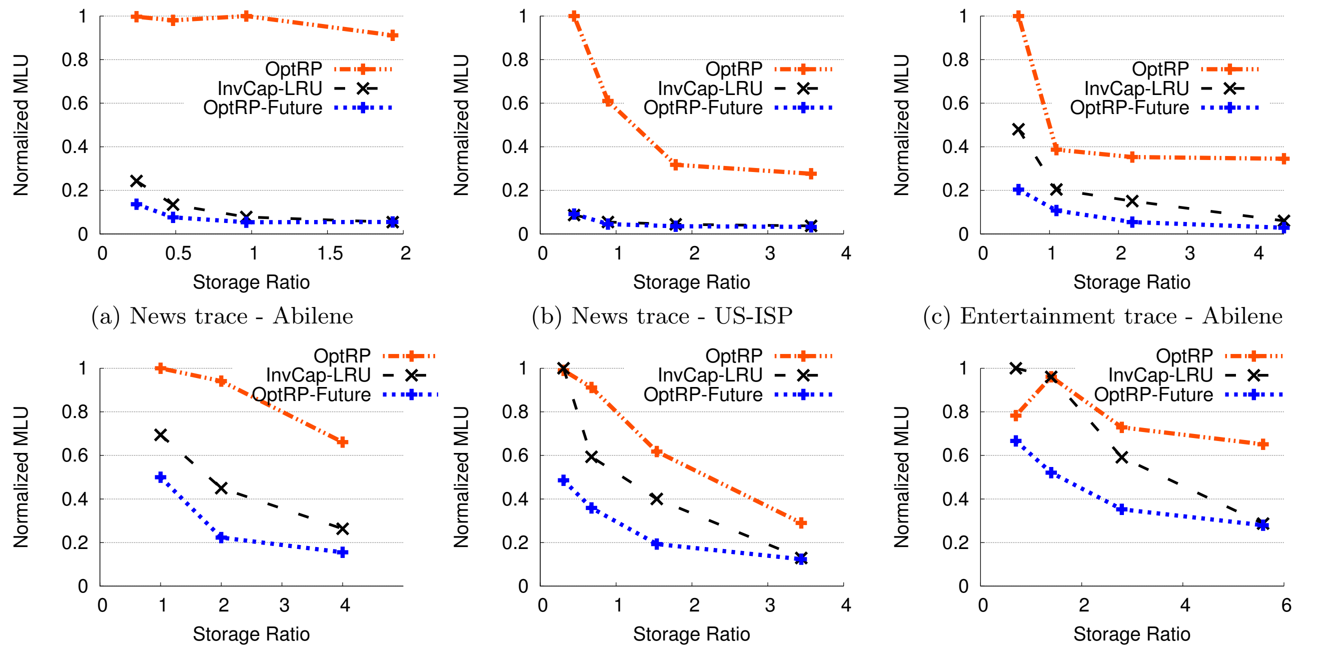
Each evaluated scheme has a content placement component and a routing component. We schematically label each evaluated scheme using the notation Routing-Scheme+Placement-Scheme as follows.

InvCap-LRU uses Inverse Cap (with ECMP) as the routing strategy and LRU as the cache replacement strategy. Inverse Cap is a static, shortest-path routing scheme where link weights are set to the inverse of link capacity. This scheme requires no information of either content demand or the traffic matrix. If content is available at multiple PoPs in the network, we fetch the content from the PoP which is closest based on hop count distance, breaking ties randomly among PoPs with equal hop count distance. **OptR-LRU** calculates an optimal routing every three hours based on the traffic matrix measured over the past three hours using the multi-commodity flow optimization discussed in §3.3; it uses the LRU cache replacement strategy.

We add a straightforward optimization to LRU where if a user terminates the request before 10% of the video (file) is viewed (downloaded), the content is not cached (and the rest of the file is not fetched); otherwise the entire file is downloaded and cached. This optimization is used since we observe in our traces that a user watching a video very often stops watching it after watching the initial period. A similar phenomenon is observed for large file downloads, but less frequently than video.

OptRP computes a joint optimization of placement and routing based on the previous day's content matrix using the MIP formulation of §3.2 once a day. **OptRP-Future** has oracular knowledge of the content matrix for the next day and uses it to calculate a joint optimization of placement and routing. OptRP and OptRP-Future are identical in all respects except that the former uses the content matrix of the past day while the latter has perfect future knowledge. These two schemes help us understand the value of future knowledge. In practice, it may be possible for an NCDN to obtain partial future knowledge placing it somewhere between the two extremes. For instance, an NCDN is likely to be informed beforehand of a major software release the next day (e.g., new version of the Windows) but may not be able to anticipate a viral video that suddenly gets "hot".

To determine the value of optimizing routing alone, we study two more schemes: **InvCap-OptP** and **InvCap-OptP-Future**. These can be viewed as variants of OptRP and OptRP-Future respectively where Inverse Cap routing is used in the network but content placement is optimized, rather



(a) News trace - Abilene (b) News trace - US-ISP (c) Entertainment trace - Abilene

(d) Entertainment trace - US-ISP (e) Downloads trace - Abilene (f) Downloads trace - US-ISP

Figure 7: OptRP performs much worse than InvCap-LRU. OptRP-Future performs better than InvCap-LRU at small storage ratios but the difference decreases at higher storage ratios.

than jointly optimizing both. The MIP used by these schemes is described in §3.3.

For all schemes that generate a new placement each day, we implement the placement during the low-traffic period from 4 AM to 7 AM EST. This ensures that the traffic generated due to changing the content placement occurs when the links are underutilized. For these schemes, the routing is updated each day at 7 AM EST once the placement update is finished.

5.3 Experiments

5.3.1 Analysis of Video & Downloads Traffic

Figure 7 shows the results for the news, entertainment and downloads traces on Abilene and US-ISP. Our first observation is that a realistic demand-aware placement and routing scheme, OptRP, performs significantly worse than a completely demand-oblivious scheme, InvCap-LRU. OptRP has 2.2× to 17× higher MLU than InvCap-LRU even at the maximum storage ratio in each graph. OptRP has a high MLU because it optimizes routing and placement based on previous day's content demand while a significant fraction of requests are for new content not accessed the previous day. As Figure 6 shows, the fraction of requests for new content is up to 63%, 31%, and 20% for the news, entertainment, and downloads traces respectively. Due to new content, the incoming traffic from origin servers is significant, so the utilization of links near the exit nodes connecting to the origin servers is extremely high.

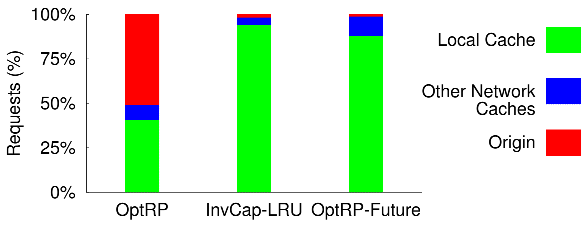
The fraction of requests served from the origin is much higher for OptRP compared to InvCap-LRU as well as OptRP-Future on the news and entertainment traces. Figure 8 shows that OptRP serves 50% and 21% of requests from the origin for news and entertainment respectively. In comparison, InvCap-LRU and OptRP-Future serve less than 2% of requests from the origin. Therefore, OptRP has a much higher MLU than both InvCap-LRU and OptRP-Future on the two traces.

The downloads trace differs from other traces in that, except for one day, the traffic is quite predictable based on the previous day's history. This is reflected in the performance of OptRP, which performs nearly the same as OptRP-Future on all days except the ninth day of the trace (see Figure 9). The surge in MLU for OptRP on the ninth day is because nearly 20% of requests on this day is for new content consisting of highly popular software update releases (see Figure 6). The surge in MLU on this one day is mainly responsible for the poor performance of OptRP on the downloads trace.

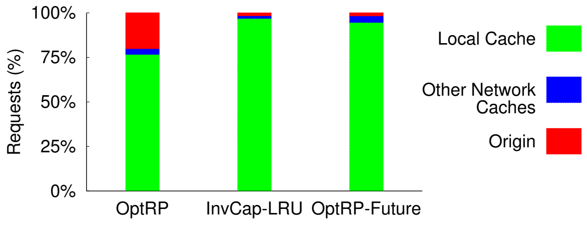
The relative difference between InvCap-LRU and OptRP grows larger as storage ratio increases. This is because InvCap-LRU utilizes the additional cache at each location to increase its hit rate and thereby reduce the traffic on congested links in the network. On the other hand, OptRP continues to serve the requests for new content not accessed the previous day from the origin and performs poorly even at high storage ratios.

Impact of future knowledge.

Next, we observe that InvCap-LRU does underperform compared to OptRP-Future that has knowledge of future content demand. However, InvCap-LRU improves with respect to OptRP-Future as the storage ratio increases. The greatest difference between the two schemes is for the experiment with the entertainment trace on US-ISP. In this case, at a storage ratio of 1, InvCap-LRU has twice the MLU of the OptRP-Future scheme; the difference reduces to 1.6× at a



(a) News trace - Abilene



(b) Entertainment trace - Abilene

Figure 8: OptRP serves 50% and 21 % from origin in news trace and entertainment trace respectively. InvCap-LRU and OptRP-Future serve less than 2% requests from origin.

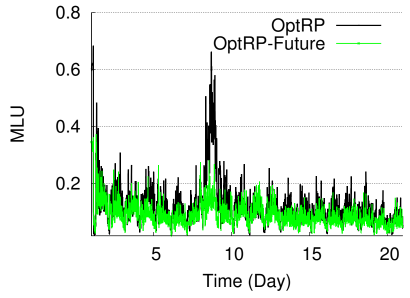
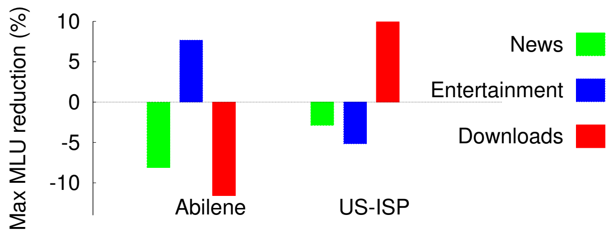


Figure 9: On downloads trace, OptRP incurs a very high MLU on one day. US-ISP.

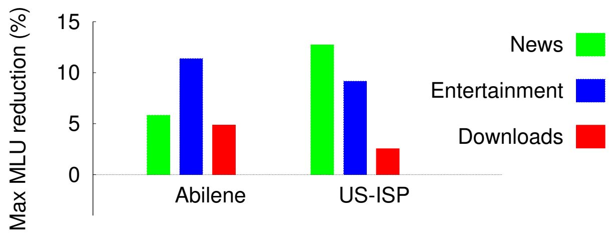
storage ratio of 4. This shows that when storage is scarce, demand-aware placement with future knowledge can significantly help by using knowledge of the global demand to maximize the utility of the storage. However, if storage is plentiful, the relative advantage of OptRP-Future is smaller. An important implication of these results is that an NCDN should attempt to do demand-aware placement only if the future demand can be accurately known or estimated, otherwise a simpler demand-oblivious scheme such as LRU suffices.

Impact of optimizing routing.

How are the above conclusions impacted if InvCap-LRU were to optimize routing or OptRP-Future were to use InvCap routing? To answer this question, we analyze the maximum reduction in MLU by using OptR-LRU over InvCap-LRU across all storage ratios in Figure 10. We similarly compare OptRP-Future and InvCap-OptP-Future. We find that OptR-LRU improves the MLU over InvCap-LRU by at most 10% across all traces suggesting that optimizing routing is of little value for a demand-oblivious placement scheme. OptRP-Future reduces network cost by at most 13% compared to InvCap-OptP-Future. As we consider OptRP-Future



(a) MLU reduction with OptR-LRU compared to InvCap-LRU



(b) MLU reduction with OptRP-Future compared to InvCap-OptP-Future

Figure 10: Optimizing routing gives little improvement to MLU of either InvCap-LRU or InvCap-OptP-Future

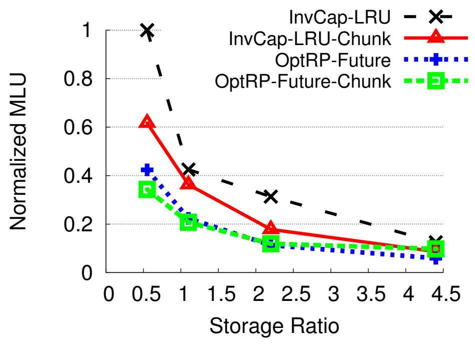
to be the "ideal" scheme with full future knowledge, these results show that the best MLU can be achieved by optimizing content placement alone; optimizing routing adds little additional value.

Non-monotonic "optimal" behavior.

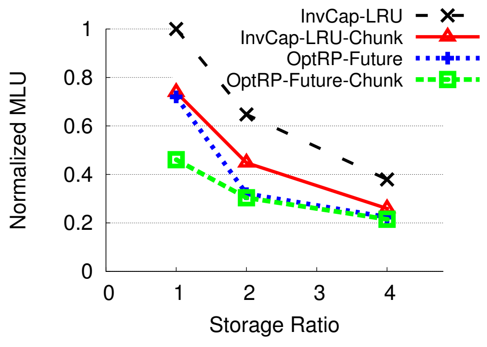
Somewhat counterintuitively, the MLU sometimes increases with a higher storage ratio for the OptRP scheme. There are at least three reasons that explain this. First, the optimization formulation optimizes for the content matrix assuming that the demand is uniformly spread across the entire day, however the requests may actually arrive in a bursty manner. So it may be sub-optimal compared to a scheme that is explicitly optimized for a known sequence of requests. Second, the optimization formulation optimizes the MLU for the "smoothed" matrix, but the set of objects placed by the optimal strategy with more storage may not necessarily be a superset of the objects placed by the strategy with lesser storage at any given PoP. Third, and most importantly, the actual content matrix for the next day may differ significantly from that of the previous day. All of these reasons make the so-called "optimal" OptRP strategy suboptimal and in combination are responsible for the nonmonotonicity observed in the experiments.

5.3.2 Content Chunking

Content chunking is widely used to improve content delivery efficiency. For example, HTTP [19] and Apple's HLS protocol [1] support content chunking and BitTorrent distributes content in small chunks [7]. In our context, both demand-aware and demand-oblivious placement benefit from chunking. A demand-oblivious placement improves with chunking because a cache can store a partially downloaded content if a user aborts the download before completion. Chunking helps demand-aware placement for multiple reasons. First, chunks of a content differ in popularity. There-

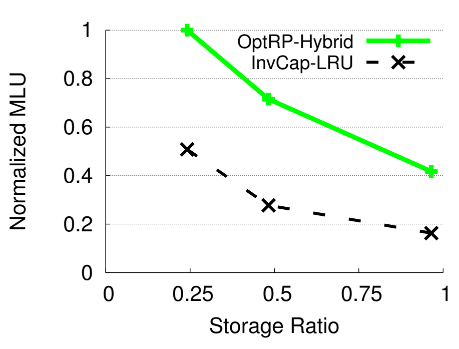


(a) Entertainment trace - Abilene

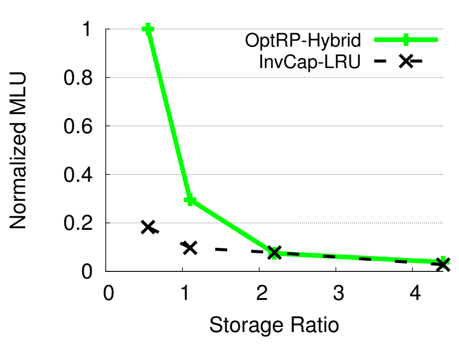


(b) Entertainment trace - US-ISP

Figure 11: Content chunking improves performance of InvCap-LRU relative to OptRP-Future on both topologies.



(a) News trace - Abilene



(b) Entertainment trace - Abilene

Figure 12: Hybrid placement strategies either perform as well as InvCap-LRU or worse for both traces.

fore, placing more popular chunks at more locations is better. Second, chunking enables storing more content at each node, e.g., a large file may not fit at any PoP due to storage constraints, but its chunks can be stored across a set of PoPs. Third, splitting a content across multiple locations spreads the traffic for that content over more links, which reduces network cost.

Next, we describe our findings on the impact of chunking (refer §5.1 for the methodology) on NCDN strategies. We find that although chunking improves performance of both InvCap-LRU and OptRP-Future as expected, it significantly improves the performance of InvCap-LRU relative to OptRP-Future. Figure 11 shows the results of our experiments on the entertainment trace. Due to chunking, the maximum difference between the MLU of InvCap-LRU and OptRP-Future reduces from 2.5× to 1.4×. At the maximum storage ratio in each case, InvCap-LRU is at most 20% worse compared to OptRP-Future with chunking. Our experiments (omitted for brevity) on the downloads trace have qualitatively similar conclusions. Chunking makes a small difference on our experiments with the news trace as more than 95% content is of duration less than our chunk size. Overall, chunking strengthens our conclusion that a demand-oblivious placement and routing achieve close to the best possible network cost for an NCDN.

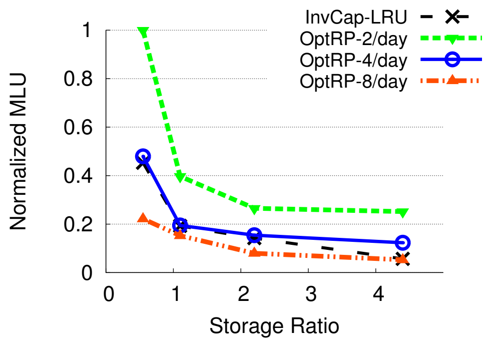
Even with content chunking, the demand-aware placement and routing strategy, OptRP, has up to 7× higher network cost compared to InvCap-LRU in the experiment with entertainment trace (not shown in Figure 11). This is because chunking does not help OptRP's primary problem of not being able to adapt effectively to new content, so it continues to incur a high cost.

5.3.3 *Alternative Demand-aware Schemes*

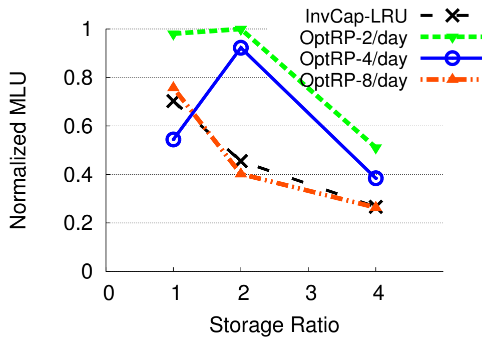
The experiments so far suggest that a demand-aware scheme that engineers placement and routing once a day based on the previous day's demand performs poorly compared to a demand-oblivious scheme, InvCap-LRU. In this section, we evaluate the performance of two alternative demand-aware schemes. The first is a hybrid scheme that combines demand-aware and demand-oblivious schemes, and the second is a demand-aware scheme that optimizes placement and routing multiple times a day. We discuss each of them in turn.

Hybrid placement: We evaluate a hybrid placement scheme that splits the storage at each node into two parts - one for a demand-aware placement based on previous day's content demand and the other for placing the content in a demand-oblivious LRU manner. This hybrid strategy is similar to that used in [2]. We present the results for experiments with news trace and entertainment trace on the Abilene topology in Figure 12. For this experiment, we used 20% of storage at each node as a LRU cache and the rest of the storage to place content exactly as in OptRP. We find that in both these cases InvCap-LRU performs either as well or better than the OptRP scheme. We also experimented with assigning a greater fraction of storage to demand-oblivious placement (omitted for brevity), but the above conclusions remain unchanged in those experiments.

We also tried a second hybrid placement scheme that uses



(a) Entertainment trace - Abilene



(b) Entertainment trace - US-ISP

Figure 13: Demand-aware placement and routing can match the performance of InvCap-LRU if computed eight times per day.

LRU caching only for content with zero demand on the previous day, e.g., new content published on the present day; planned placement is used for content with a non-zero demand on the previous day. As before, we used 20% of storage as an LRU cache and remainder for demand-aware placement. The MLU for this hybrid scheme was higher than that of the InvCap-LRU scheme as well.

This experiment shows that simple strategies that combine demand-oblivious placement and demand-aware placement do not perform better than a demand-oblivious content placement scheme. Of course, a carefully designed hybrid placement scheme by definition should perform at least as well as the demand-oblivious and demand-aware schemes, both of which are extreme cases of a hybrid strategy. However, we were unable to design simple hybrid strategies that consistently outperformed fully demand-oblivious placement and routing.

OptRP multiple times per day: Next, we analyze the performance of demand-aware schemes that engineer placement and routing multiple times each day at equal intervals - twice per day, four times per day, and eight times per day. In all cases, we use the content demand in the past 24 hours to engineer placement and routing. In Figure 13, we compare the performance OptRP for different frequencies against InvCap-LRU scheme. As OptRP engineers more frequently, its performance improves. This is because it updates placement more quickly based on the demand for new content and a change in demand for older content. However, OptRP needs to engineer eight times per day to match the performance of a demand-oblivious placement. In all other cases, InvCap-LRU performs better. The experiments with the entertainment trace shown here represents the best case for OptRP. Typically, OptRP incurs a higher MLU InvCap-LRU even when engineering is done eight times per day, e.g., on the news trace, we find OptRP incurs up to 4.5× higher MLU compared to InvCap-LRU even on engineering eight times per day.

Considering the effort involved in executing a demand-aware placement—measuring content demand at all PoPs, solving a computationally intensive optimization problem, moving content to new locations—and the frequency at which it needs to be done—eight times a day—even to match the cost achieved by a demand-oblivious strategy, our position is that the pain does not justify the gain. In practice, NCDNs are better served by opting for a much simpler demand-oblivious strategy and provisioning more storage if optimizing the cost further is deemed necessary.

5.3.4 *Impact of Experimental Parameters*

We perform an extensive set of experiments to understand the effect of some of the parameters that could have biased our findings: (1) storage distribution across nodes, (2) frequency of routing updates, (3) number of cache deployments, and (4) number of exit nodes that connect to the origin. However, we find that our overall conclusions are robust to variations in the values of these parameters.

Demand-aware routing (OptR-LRU) parameters: We ask if OptR-LRU, shown in the previous section to have nearly the same network cost as InvCap-LRU, performs significantly better if we either (1) change the interval at which routing is updated, or (2) optimize routing based on a traffic matrix other than the one measured over the past three hours. We perform two sets of experiments to answer these questions. The first experiment evaluates MLU for several routing update intervals: 15 min, 30 min, 3 hr (default), 6 hr, and 24 hr. In each case, we compute routing based on TM measured since the last routing update. We find that the network cost of OptR-LRU remains almost unchanged irrespective of the routing update interval (graphs omitted for brevity).

The second experiment optimizes routing based on the TM measured on the previous day instead of TM measured since the last routing update, e.g., if routing is being updated at 9am for the period of 9am-12pm, then we use TM measured from 9am-12pm on the previous day. We set the routing update interval to 3 hours and then to 6 hours. We find that the network cost in in both these cases is nearly identical to OptR-LRU's network cost with our default parameters (graphs omitted for brevity). This experiment suggests that OptR-LRU is unlikely to improve even if we optimize routing based on other previously measured TMs. In summary, these experiments reinforce the finding that InvCap-LRU is as effective as OptR-LRU.

Heterogenous storage: In this experiment, the storage at each PoP is set proportional to the fraction of requests at the PoP in each trace. We draw out two main observations from our results (graphs omitted for brevity). First, OptRP continues to perform poorly compared to InvCap-LRU. This is expected as heterogenous storage makes little difference to the "miss" traffic generated because of newly published content. Second, network cost of InvCap-LRU as well as OptRP-Future increases with heterogenous storage compared to homogenous storage but the increase is more for InvCap-

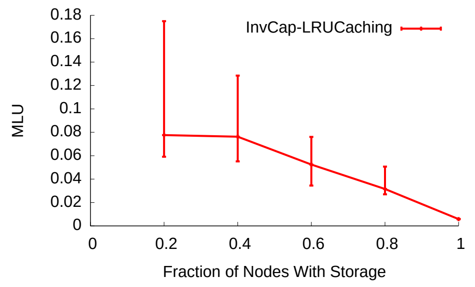


Figure 14: If caches are deployed on all PoPs, MLU is significantly lower compared to scenarios when caches are deployed only at a fraction of PoPs; the total storage across PoPs is same in all scenarios. Median MLU value along with max and min values across five repetitions are shown. (Entertainment trace, US-ISP)

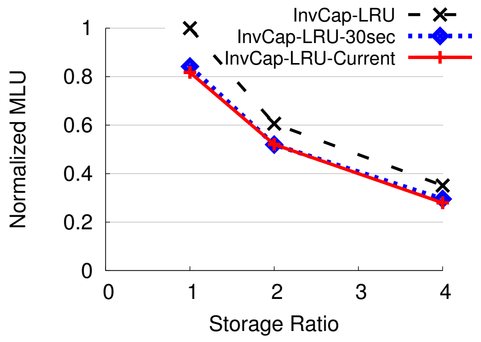


Figure 15: Link utilization aware request redirection helps InvCap-LRU reduce network cost up to 21%. Link utilizations updated every 30 seconds performs as well as using current link utilizations. (Entertainment trace, US-ISP)

LRU compared to OptRP-Future. We observe that InvCap-LRU's network cost increases up to 30% or more in four out of six experiments. But OptRP-Future's network cost increases up to 20% or less (except for the entertainment trace on US-ISP). The increase in InvCap-LRU's network cost is likely because the PoPs that have the smallest fraction of all requests are assigned extremely small values of storage, which increases the cache miss rate at these PoPs.

We evaluated InvCap-LRU's performance with two more heterogenous storage distributions: (1) storage proportional to capacity of outgoing links at PoP (2) storage proportional to the number of outgoing links at PoP. Compared to homogenous storage, the network cost of InvCap-LRU is up to 180% more when storage is set proportional to capacity and is up to 60% more when storage is set proportional to number of degree of the PoP. It is unclear to us if a simple heuristic to select a heterogenous storage distribution exists that significantly improves InvCap-LRU's network cost over a homogenous storage distribution. Overall, as both InvCap-LRU and OptRP-Future perform better with homogenous storage than the heterogenous storage distributions we considered, our earlier results with homogeneous storage should be considered as more representative.

Number of caches: To understand how many caches should an NCDN deploy, we evaluate the performance of InvCap-LRU when caches are deployed at a fraction of PoPs selected randomly. In each experiment, total storage is fixed but the fraction of PoPs with caches is varied from 0.2 to 1.0; storage is provisioned homogeneously across caches. As Figure 14 shows, an increase in the number of caches reduces the MLU, and deploying caches at all the PoPs results in the least MLU. Even a cache deployment at 80% of PoPs is sub-optimal, and results in up to 5× higher MLU in comparison to a cache deployment at all PoPs. This experiment shows that NCDNs should deploy caches at all PoPs, as we have assumed in earlier experiments.

Number of exit nodes to origin: The experiments so far assumed that the origin server can be reached through exactly three exit locations in the network. Next, we perform our experiments with one exit location and five exit locations. Our main findings from the prior sections, namely–InvCap-LRU performs significantly better than OptRP; OptRP-Future performs better than InvCap-LRU; and optimizing routing adds little value;–remain unchanged as we vary the number of exit nodes. Thus, our findings are not sensitive to the number of exit nodes connected to the origin.

5.3.5 *Redirection Schemes for LRU*

The demand-oblivous placement scheme, LRU, specifies a cache replacement strategy, but there are several request redirection schemes that can be used in combination with LRU. In this section, we first propose a novel redirection scheme that performs request redirection considering link utilizations in order to reduce network cost. Next, we analyze whether requests should be redirected to all PoPs or a smaller subset of them in order to minimize network cost. We use content chunking for all experiments in this section.

Link utilization aware request redirection: This scheme is designed to reduce link utilization in an NCDN. The idea is to periodically measure the utilization of all links and then redirect requests over the paths that have less utilized links. For example, if a request can be redirected to either of two PoPs, but the links along the path from one of the PoPs are more utilized than the links along the path from the other PoP, then the PoP whose path has *less* utilized links is chosen. Compared to our current redirection strategy, i.e., choose the closest PoP based on hop count distance, we expect that request redirection considering link utilizations will achieve lower network cost.

Our implementation works as follows. When a request results in a cache miss at the local PoP (the PoP first contacted by a user), it fetches content from other PoPs that may have the content available. If multiple PoPs have the content, the local PoP fetches content from the PoP for which the utilization of the most utilized link along the path from that PoP to the local PoP is the least. We break ties based on hop count distance and then randomly.

We experiment with two versions of the above algorithm, which differ in the frequency at which link utilization levels are measured across all PoPs. In the first version, InvCap-LRU-Current, all PoPs in the network know the link utilizations of all links at every moment. In the second version, InvCap-LRU-30sec, link utilization is measured at all links every 30 seconds, and is updated across all PoPs. Our baseline for comparison is InvCap-LRU scheme. The goal of our

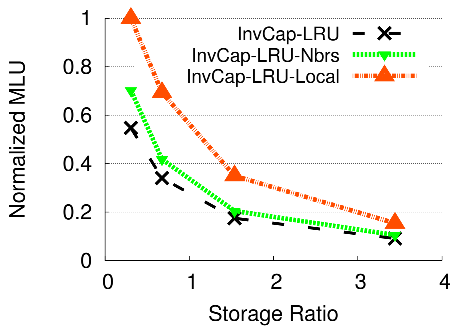


Figure 16: InvCap-LRU-Nbrs redirects requests only to neighbors while InvCap-LRU-Local redirects to no other PoPs and sends requests directly to the origin. InvCap-LRU-Nbrs has a network cost within 27% of InvCap-LRU while InvCap-LRU-Local incurs up to a 100% higher network cost. (Downloads trace, Abilene)

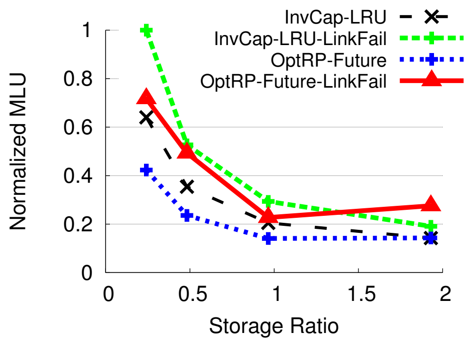
experiments is to identify if link-utilization-based redirection improves performance at all compared to the baseline; further optimizing the frequency of monitoring link utilization levels is beyond the scope of this paper.

Our results show that request redirection using link utilizations does reduce network cost compared to InvCap-LRU. Figure 15 presents the results for the experiment with the entertainment trace on US-ISP (with other graphs omitted for brevity). InvCap-LRU-Current reduces network cost up to 19% compared to InvCap-LRU. In the experiments with other traces (graphs omitted), we find that InvCap-LRU-Current reduces network cost by 10% to 21% compared to InvCap-LRU. In addition, InvCap-LRU-30sec matches the network cost of InvCap-LRU-Current on all traces except for downloads trace on Abilene topology, where its network cost is higher by up to 9%. This result implies that an implementation in which link utilization levels are updated at PoPs every 30 seconds can achieve most of the benefits of link utilization aware request redirection.

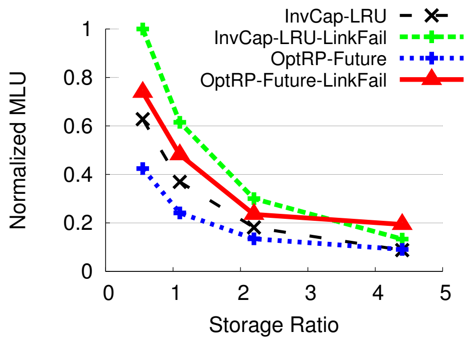
Link utilization aware redirection and content chunking, in combination, nearly blur the difference between InvCap-LRU and OptRP-Future. After these optimizations are used, InvCap-LRU is at most 4% sub-optimal compared to OptRP-Future at the highest storage ratio in each experiment. In some cases, InvCap-LRU even achieves a lower network cost than OptRP-Future, e.g, on the Entertainment trace on the Abilene topology, InvCap-LRU performs 22% better than OptRP-Future.

Request redirection to neighbors: In our InvCap-LRU implementation, a PoP redirects request to all PoPs upon a cache miss, which maximizes the hit rate of network caches and reduces load on origin servers. To examine whether request redirection to all PoPs is necessary and to quantify its benefit, we compare InvCap-LRU against two strategies that redirect requests to a much smaller set of PoPs. First, is InvCap-LRU-Nbrs, where a PoP redirects requests only to its one-hop neighbors on a cache miss before contacting the origin. Second, is InvCap-LRU-Local, where a PoP sends requests to the origin on a cache miss at a PoP.

In Figure 16, we see that InvCap-LRU-Nbrs has between



(a) News trace - Abilene



(b) Entertainment trace - Abilene

Figure 17: Due to link failures, InvCap-LRU and OptRP-Future see up to 110% and 99% increase in MLU respectively. In case of link failures, InvCap-LRU can even achieve a lower network cost than OptRP-Future at high storage ratios.

6% to 27% higher network cost than InvCap-LRU depending on trace and topology. In comparison, InvCap-LRU-Local has up to 25% to 100% higher network cost than InvCap-LRU. These results show that request redirection to other network caches helps reduce network cost, but most of this reduction can be had by redirecting requests only to neighboring PoPs.

InvCap-LRU-Local serves between 3× to 7× more requests from origin than InvCap-LRU, which increases the utilization of links near the exit nodes connecting to origin nodes. InvCap-LRU-Nbrs also serves more requests from origin than InvCap-LRU, but the increase is between 2× to 3×. Therefore, network cost for InvCap-LRU-Nbrs is smaller compared to InvCap-LRU-Local.

5.3.6 *Is Traffic Engineering Necessary?*

Our results suggest that optimizing routing yields little improvement to network cost for the NCDN portion of the traffic, but this finding does not imply that traffic engineering by ISPs is unnecessary. An important reason for ISPs to engineer traffic is that they route transit traffic in addition to NCDN traffic. Since an ISP does not control either content placement or request redirection for transit traffic, traditional traffic engineering methods, e.g., OSPF traffic engineering, can reduce the network cost due to transit traffic. The benefit of traffic engineering, or lack thereof, depends on the fraction of transit traffic vs. NCDN traffic in an ISP.

Minimizing network cost in case of link failures is also an objective of traffic engineering. Next, we study whether the demand-oblivious placement and routing scheme, InvCap-LRU, that is highly effective in the failure-free scenario, also provides resilience to link failures. To this end, we compare the performance of InvCap-LRU and OptRP-Future in case of link failures. The set of failure scenarios we consider includes all single-link failures. We calculate the network cost for a failure scenario by simulating the entire trace with the failed link. Across all such failure scenarios for each scheme, we select the one with the highest network cost and attribute that as the network cost for the scheme for purposes of our comparative evaluation.

For this experiment, we add a constraint to OptRP-Future that ensures that the set of paths between any pair of nodes has at least two paths with no links in common. As a result, the routing computed is resilient to any single link failure. In terms of variables in Table 3, our constraint is

$$f_{ije} \leq \beta \times f_{ij}, \quad \forall e \in E, i, j \in V$$

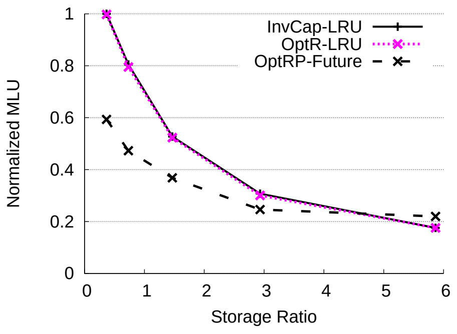
where we set $\beta = 0.75$. This constraint means that a link can carry at most 75% of traffic between any pair of nodes. OptRP-Future calculates placement and routing assuming a failure-free scenario. In case of a link failure, we update routing by excluding all paths that cross the failed link. For each pair of nodes, traffic splitting ratios for the working paths are multiplied by a normalizing constant such that traffic splitting ratios add up to one. InvCap-LRU updates shortest path routes in case of a link failure while keeping existing link weights the same.

Figure 17 shows the network cost of InvCap-LRU and OptRP-Future in the failure-free scenario and their network costs during failures. InvCap-LRU and OptRP-Future see up to a 110% and 99% increase in network cost respectively during failures. During failures, we find that InvCap-LRU has a significantly higher network cost than OptRP-Future at small storage ratios, but the difference reduces at higher storage ratios with InvCap-LRU even achieving a lower network cost than OptRP-Future at the maximum storage ratio in each graph. Comparing the failure-free and failure scenarios, the relative sub-optimality of InvCap-LRU with respect to OptRP-Future remains the same at small storage ratios but reduces at higher storage ratios. The nonmonotonicity of OptRP-Future is because it optimizes routing and placement assuming a failure-free scenario and hence performs sub-optimally when link failures occur.

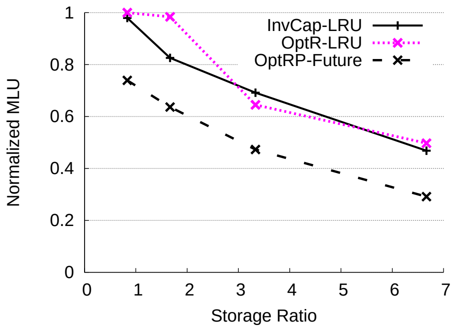
We believe that our analysis of the network cost during failures could be further improved, e.g., we consider only the worst case network cost in case of link failures, and our modification of OptRP-Future does not guarantee that the modified routing will be optimal in case of link failures. Our primary focus in this paper is on minimizing hotspots, so a more detailed evaluation of fault-tolerant traffic engineering in NCDNs is deferred to future work.

5.3.7 *Experiments with synthetic workloads*

As our experiments are done with a fixed set of workloads and ISP topologies, this raises the question whether our conclusions are generalizable in other cases. To address this issue to some extent, we experiment with synthetic content workloads, which assume a Zipfian content popularity distribution. This workload consisted of 1000 uniform sized videos of 10 minute duration each. The distribution of aggregate popularity of videos follows a Zipf distribution with parameter $\alpha = 1.0$. Experiments with $\alpha = 0.5$, and $\alpha = 0.25$ yielded qualitatively similar results. The requests for an object are uniformly distributed at all PoPs at all times.



(a) Synthetic trace - Abilene



(b) Synthetic trace - US-ISP

Figure 18: Experiments with a synthetic trace show that OptR-LRU yields almost no improvement in MLU over InvCap-LRU. The OptRP-Future scheme outperforms InvCap-LRU even at higher storage ratios on the US-ISP topology.

Figure 18 shows two main findings. First, OptR-LRU yields almost no improvement in MLU over InvCap-LRU, which is consistent with our earlier findings. Second, the OptRP-Future scheme outperforms InvCap-LRU even at higher storage ratios on the US-ISP topology. This suggests that the simple InvCap-LRU scheme may not give the close to optimal costs in all scenarios. An analysis of relative performance of schemes for general topologies and workloads is an interesting open question, and would be considered in our future work.

6. RELATED WORK

Traffic engineering and content distribution have both seen an enormous body of work over more than a decade. To our knowledge, our work is the first to pose the NCDN problem, wherein a single entity seeks to address both concerns, and empirically evaluate different content-aware traffic engineering strategies. Nevertheless, a significant body of recent research has studied the interaction of content distribution and traffic engineering in various forms, and we explain below how our work relates to and builds upon this prior work.

Joint Optimization.

Recent work has explored the joint optimization of traffic engineering and "content distribution", where the latter term refers to the *server selection* problem. P4P proposed by Xie et al. [24] seeks to improve application performance for peer-to-peer (P2P) traffic while also reducing cost for the ISP. P4P assumes a cooperative model where the ISP supplies hints called *p-distances* to P2P applications that when used by them improves their performance and also reduces interdomain transit costs and the MLU for the ISP. In [15] and [8], the authors study the interaction between traffic engineering and content distribution using a game-theoretic model and show that, without a joint optimization, the equilibrium of this interaction may not result in a socially optimal solution. In [15], it is shown that a joint optimization can achieve benefits of up to 20% for ISPs and up to 30% for CDNs as compared to the case when there is no cooperation between them. CaTE [12], like P4P, shows that both ISPs and content providers can benefit if content providers perform request redirection based on network information provided by ISPs.

Compared to all of these works that equate content distribution to server selection (or request redirection in our parlance), the NCDN optimization formulation additionally considers content placement itself as a degree of freedom. As our results show, optimizing placement is powerful and can single-handedly reduce the MLU significantly even in conjunction with simplistic request redirection and routing strategies.

Placement Optimization.

Applegate et al. [2] study the content placement problem for a VoD system that seeks to minimize the aggregate network bandwidth consumed assuming a *fixed routing* in the network. They compare different video placement algorithms and find that an optimized, demand-aware placement strategy with a small local cache (similar in spirit to our "hybrid" strategy in §5.3.3) outperforms purely demand-oblivious LRU-like strategies. Our work differs from theirs with respect to both the problem addressed and the qualitative findings as follows. We model an NCDN that in addition to placement also controls routing, and assessing the interaction and relative importance of routing and placement strategies is one of our contributions.

Furthermore, unlike [2], we find that a simple, demand-oblivious, LRU-like strategy significantly outperforms an optimized, demand-aware (static or hybrid) placement strategy. There are two explanations for this seeming disparity. First, we consider a comprehensive trace of CDN requests with a variety of on-demand video as well as download traffic that exhibits significant daily churn; in comparison, their workload appears to be reasonably predictable even over weekly timescales. Second, the optimized, demand-aware placement strategy they consider also has some benefit of future knowledge and is therefore somewhat comparable to the optimized scheme with future knowledge that we analyze (OptRP-Future in §5). In general, obtaining knowledge about future demand may not be practical for all types of content, e.g., news videos, for a large NCDN. Furthermore, our analysis of different storage ratios suggests that LRU performs worse only at small storage ratios, and the difference between LRU and optimized content placement reduces significantly on increasing the storage ratio.

Traffic Engineering.

Traffic engineering schemes have seen a long line of work and a variety of schemes such as OSPF link-weight optimization [10], MPLS flow splitting [9], optimizing routing using multiple traffic matrices [23, 25], online engineering strategies [16, 9], and provably near-optimal oblivious routing [3, 4] have been studied. All of these schemes assume that the demand traffic is a given to which routing must adapt. However, we find that an NCDN is in a powerful position to change the demand traffic matrix, so much so that even a naive scheme like Inverse Cap, i.e., no engineering at all, suffices in conjunction with a judicious placement strategy and optimizing routing further adds little value. In this respect, our findings are comparable in spirit to Sharma et al. [20]. However, they focus on the impact of location diversity, assuming a fixed number of randomly placed replicas of each content, and find that even a small number of random replicas suffice to blur differences between different traffic engineering schemes with respect to a capacity metric (incomparable to MLU), but find that engineering schemes still outperform static Inverse Cap routing.

7. CONCLUSIONS

We posed and studied the content-aware traffic engineering problem where content distribution and traffic engineering decisions are optimized jointly as opposed to being optimized separately as it is done today. This paradigm is of key importance to NCDNs who own and manage the infrastructure for content distribution as well as the underlying network. Our trace-driven experiments using extensive access logs from the world's largest CDN and realistic ISP topologies resulted in the following key conclusions. First, simple demand-oblivious schemes for routing and placement, such as Inverse Cap and LRU, outperformed sophisticated placement and routing schemes that utilized the prior day's demand for its optimization. Second, traffic demand can be "shaped" by effective content placement to the extent that in many cases no engineering of traffic is needed. Finally, we studied the value of the future knowledge of demand for placement and routing decisions. While future knowledge helps, what is surprising is that a small amount of additional storage allows demand-oblivious schemes to perform as well as demand-aware ones that know the future.

8. REFERENCES

[1] Apple. HTTP Live Streaming. `http://bit.ly/MgoUED`.

[2] D Applegate, A Archer, V Gopalakrishnan, S Lee, and K K Ramakrishnan. Optimal content placement for a large-scale VoD system. In *Co-NEXT*, 2010.

[3] D Applegate and E Cohen. Making routing robust to changing traffic demands: algorithms and evaluation. *IEEE/ACM Trans. Netw.*, 14:1193–1206, December 2006.

[4] Y Azar, E Cohen, A Fiat, H Kaplan, and H Racke. Optimal oblivious routing in polynomial time. In *STOC*, 2003.

[5] Edge Cast. `http://www.edgecast.com/solutions/licensed-cdn/`.

[6] Cisco. Visual Networking Index, 2011. `http://bit.ly/KXDUaX`.

[7] B Cohen. BitTorrent Protocol. `http://bit.ly/KDhQV1`.

[8] D DiPalantino and R Johari. Traffic Engineering vs. Content Distribution: A Game Theoretic Perspective. In *INFOCOM*, 2009.

[9] A Elwalid, C Jin, S Low, and I Widjaja. MATE: MPLS adaptive traffic engineering. In *INFOCOM*, 2001.

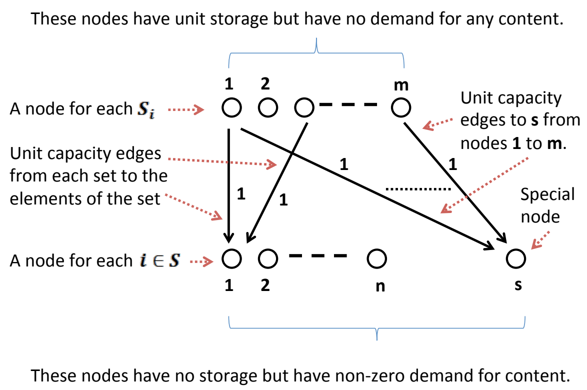

These nodes have unit storage but have no demand for any content.
A node for each S_i
1
2
m
Unit capacity edges to s from nodes 1 to m.
Unit capacity edges from each set to the elements of the set
1
1
1
1
Special node
A node for each $i \in S$
1
2
n
s
These nodes have no storage but have non-zero demand for content.

Figure 19: Reduction from SetCover to Opt-NCDN

[10] B Fortz and M Thorup. Internet traffic engineering by optimizing OSPF weights. In *INFOCOM*, 2000.

[11] B Fortz and M Thorup. Optimizing ospf/is-is weights in a changing world. *JSAC*, May 2002.

[12] B Frank, I Poese, G Smaragdakis, S Uhlig, and A Feldmann. Content-aware Traffic Engineering. *ArXiv e-prints*, 2012.

[13] IBM. ILOG CPLEX. `http://ibm.co/KRuqhB`.

[14] Akamai NetSession Interface. `http://www.akamai.com/client`.

[15] W Jiang, R Zhang-Shen, J Rexford, and M Chiang. Cooperative content distribution and traffic engineering in an ISP network. In *SIGMETRICS*, 2009.

[16] S Kandula, D Katabi, B Davie, and A Charny. Walking the tightrope: responsive yet stable traffic engineering. In *SIGCOMM*, 2005.

[17] Nielsen. Online Video Usage Up 45%. `http://bit.ly/MiXiPU`.

[18] E Nygren, R K Sitaraman, and J Sun. The Akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, August 2010.

[19] RFC. 2616. `http://www.ietf.org/rfc/rfc2616.txt`.

[20] A Sharma, A Mishra, V Kumar, and A Venkataramani. Beyond MLU: An application-centric comparison of traffic engineering schemes. In *INFOCOM*, 2011.

[21] Abilene Topology. `http://bit.ly/Lf8k7a`.

[22] Verizon. HBO for FIOS Customers. `http://bit.ly/JQ2dn8`.

[23] H Wang, H Xie, L Qiu, Y R Yang, Y Zhang, and A Greenberg. COPE: Traffic Engineering in Dynamic Networks. In *SIGCOMM*, 2006.

[24] H Xie, Y R Yang, A Krishnamurthy, Y G Liu, and A Silberschatz. P4P: Provider Portal for Applications. In *SIGCOMM*, 2008.

[25] C Zhang, Y Liu, W Gong, J Kurose, R Moll, and D Towsley. On optimal routing with multiple traffic matrices. In *INFOCOM*, 2005.

APPENDIX

A. COMPLEXITY OF NCDN PROBLEM

Opt-NCDN is the decision version of the NCDN problem described in §3. Opt-NCDN asks if the MLU of the network can be α while satisfying the constraints of the problem.

THEOREM 1 *Opt-NCDN is NP-Complete even in the special case where all objects have unit size, all demands have unit value, and link and storage capacities have binary values.*

Proof: We show a reduction from the well known SetCover problem. We first define the SetCover problem that we will reduce to Opt-NCDN.

SetCover: Let $S = \{1, 2, ..., n\}$ be a set of n elements. Let $X = \{S_1, ..., S_m\}$ where $S_i \subseteq S, 1 \leq i \leq m$. Let k be an integer. SetCover asks if there exists $Y = \{Y_1, ..., Y_k\}$, where $Y_k \in X$ and $Y_1 \cup ... \cup Y_k = S$. Set Y is called a set cover of size k.

The reduction from SetCover to Opt-NCDN is described using the network in Figure 19. Set $V_1 = \{1, ..., m\}$ refers to nodes in the top row. Each node $i \in V_1$ maps to the set $S_i \subset S$. Set $V_2 = \{1, ..., n\}$ refers to nodes in the bottom row excluding node s. Each node $i \in V_2$ maps to element $i \in S$. Node s is called a special node.

Directed links (i, j) exist from all nodes $i \in V_1$ to all nodes $j \in V_2$. The capacity of (i, j) is 1 unit if $i \in S_j$, otherwise capacity is zero. Node s has incoming links (i, s) from all nodes $i \in V_1$ such that the capacity of all incoming links is 1 unit. All nodes in the top row V_1 have unit storage whereas nodes in the bottom row $V_2 \cup \{s\}$ have zero storage.

The set of objects is $\{o, 1, 2, ..., (m - k)\}$ and all objects have unit size. Object o is a special object that has unit demand at nodes in set $V_2 = \{1, ..., n\}$ and zero demand at all other nodes. Objects 1, 2, .. $(m - k)$ have unit demand at special node s and zero demand at all other nodes.

CLAIM: There is a set cover of size k if and only if the above network can achieve MLU ≤ 1.

If there is a set cover of size k, then the network can achieve MLU of 1. Store the special object o at the k set cover locations in the top row and satisfy demand for o at nodes $V_2 = \{1, ..., n\}$ in the bottom row from these locations with MLU $= 1$. The remaining $(m-k)$ nodes in the top can be used for objects $\{1, 2, ..., (m - k)\}$ to satisfy the demand at special node s with MLU of 1.

If there is no set cover of size k, then the network must have a MLU > 1. Objects must be placed in some $(m - k)$ nodes in the node $V_1 = \{1, ..., m\}$ in the top row to satisfy the demand for special node s. Thus, at most k nodes are available for placing special object o. Since there is no set cover of size k, some bottom node $i \in V_2$ must satisfy its demand for special object o using an edge whose capacity is zero resulting in MLU $= \infty$ on that edge.

It is easy to show that Opt-NCDN $\in$ NP. Hence, Opt-NCDN is NP-Complete.

THEOREM 2 *Opt-NCDN is inapproximable within a factor β for any $\beta > 1$ unless $P = NP$.*

The proof of THEOREM 1 shows that if there is a set cover of size k, MLU $= 1$ and MLU $= \infty$ otherwise. Thus, if we find a solution for which MLU is finite, it implies that MLU $= 1$, which immediately gives a solution to the corresponding SetCover instance.

Lets assume a β-approximation ($\beta > 1$) exists for Opt-NCDN. Then, we can solve SetCover in polynomial time by mapping SetCover instance to Opt-NCDN instance, and checking if MLU $\leq \beta$ (which implies MLU $= 1$). As SetCover $\in$ NP-Complete, therefore, no $\beta-$approximation for Opt-NCDN exists unless P $=$ NP.

B. JOINT OPTIMIZATION OF TRANSIT TRAFFIC MATRIX AND CONTENT MATRIX

We present here a modification to the MIP in §3.2 to jointly optimize routing for an ISP transit traffic matrix (TTM) and a content matrix. Let D be a TTM and D_{ij} denote the traffic from PoP i to PoP j. We modify only the constraints (3) and (4) in the earlier MIP as follows:

$$\sum_{k \in K} t_{ijk} + D_{ij} = f_{ij}, \ \forall j \in V - X, i \in V \tag{13}$$

$$\sum_{k \in K} t_{ijk} + \sum_{k \in K} \delta_{ij} t_{iok} + D_{ij} = f_{ij}, \ \forall j \in X, i \in V \tag{14}$$

C. LIMITING CONTENT PLACEMENT UPDATE TRAFFIC

In this section, we describe our extension to the MIP presented in §3.2 which allows us to limit the MLU due to traffic from updating the content placement. To this end, we add constraints to the MIP to ensure that the MLU due to the placement update traffic is less than a constant β. In our experiment, we dynamically update the value of β to be 2/3 of the MLU within the past 24 hours of the experiment.

We follow the same notation as in §3.2. The binary variable x_{ik} denotes if the content $k \in K$ is stored at node $i \in V$, S_k denotes the size of the content. To describe the constraint, we define the following parameters: T denotes the duration over which traffic due to placement update will be spread out. X_{jk} denotes whether content $k \in K$ is stored at node $j \in V$ currently. The current routing in the network is r_{ije}, the fraction of traffic from node j to node i crossing link e. In addition we define a function $\gamma(i, j, k)$. $\gamma(i, j, k) = 1$ if $X_{jk} = 1$ and node j is the closest node in terms of hop count from node i which has stored a copy of content k, otherwise. Both r_{ije} and $\gamma(i, j, k)$ depend on current routing and placement in the network and hence are known constants. We assume that the transfer of content of size S_k happens at a constant bit rate of S_k/T. In terms of these variables we can define the total traffic u_e on any link $e \in E$ during the placement update period.

$$u_e = \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} \gamma(i, j, k) r_{ije} x_{jk} S_k / T \quad \forall e \in E \tag{15}$$

Finally, in order to limit the maximum utilization of any link $e \in E$, we add the following constraint,

$$u_e / C_e < \beta \quad \forall e \in E \tag{16}$$