Optimal Content Delivery with Network Coding

Derek Leong, Tracey Ho
Department of Electrical Engineering
California Institute of Technology
Pasadena, California 91125
Email: {*derekleong, tho*}*@caltech.edu*

Rebecca Cathey
Advanced Information Technologies
BAE Systems
Arlington, Virginia 22203
Email: *rebecca.cathey@baesystems.com*

***Abstract*—We present a unified linear program formulation for optimal content delivery in content delivery networks (CDNs), taking into account various costs and constraints associated with content *dissemination* from the origin server to storage nodes, data *storage*, and the eventual *fetching* of content from storage nodes by end users. Our formulation can be used to achieve a variety of performance goals and system behavior, including the bounding of fetch delay, load balancing, and robustness against node and arc failures. Simulation results suggest that our formulation performs significantly better than the traditional minimum k-median formulation for the delivery of multiple content, even under modest circumstances (small network, few objects, low storage budget, low dissemination costs).**

***Index Terms*—content delivery network (CDN), network coding, subgraph selection, placement problem**

I. Introduction

Content delivery networks (CDNs) are designed to improve end user experience, which is commonly measured by the availability of content and the cost incurred in accessing it. Content replication is the de facto strategy, with much research on related subproblems such as mirror or cache placement [1], [2], [3], and the end user selection of mirror sites [4], [5], [6]. In this work we investigate the application of network coding in CDNs. By allowing content to be algebraically coded and split, network coding can improve the cost-performance tradeoffs. It also admits a unified linear program formulation that can optimize performance for various costs and constraints associated with content *dissemination* from the origin server to storage nodes, data *storage*, and the eventual *fetching* of content from storage nodes by end users. Our work extends the formulation presented in [7] to address additional issues, including bounding of fetch delay and robustness against node/arc failures. We also investigate the multiple content delivery problem, and compare the performance of our formulation and the traditional minimum k-median formulation through simulation experiments.

II. Background and Related Work

CDNs have been around since the late 1990s [8]; today they play an integral role in many businesses on the Internet. Industry leader Akamai reports that it serves 75 of the top 100 U.S. online retail companies, as well as leading media companies globally [9]. Content replication has become the basis of most work on CDNs. We distinguish between two forms of content replication — caching and mirroring. *Caching* involves the passive replication and storage of content flowing through a node. Proxy caches and reverse proxy caches are often deployed by end users (e.g. residential ISPs) and origin servers (e.g. content providers), respectively, to reduce delay and bandwidth usage. We do not address caching in this paper, but note that it can be applied on top of our scheme to improve performance. *Mirroring* involves the use of mirror sites, each of which is a storage node that replicates the whole content at the origin server. Such nodes are deployed across different geographical locations to distribute the fetch load imposed by end users. The origin server proactively disseminates or "pushes" content to these nodes.

The placement problems in caching [3], [10], [11], [12], and mirroring [1], [2], [13] are largely similar; they differ mainly in the explicit consideration of content cacheability and cache misses. Our work most closely resembles these problems, whose variations have been studied in many fields [14], [15]. Past efforts were essentially based on complex combinatorial optimization problems (e.g. minimum k-median problem [1]) which naturally motivated the development of heuristics to approximate the optimal solution. The application of network coding to CDNs alleviates the complexity issues in traditional routing and content replication. For instance, a major problem simplification arises from allowing storage nodes to store partial network-coded content; they are not constrained to store the whole content or object in its entirety as is the case in [1], [2], [12], [13], *etc.* Moreover, our formulation handles many important aspects of content delivery, including load balancing, and robustness against node and arc failures. All in all, network coding enables us to provide a simple yet expressive practical formulation for optimal content delivery.

III. Linear Program Formulation

Consider a static network $G(N, A)$ comprising nodes N and lossless point-to-point arcs A. The formulation extends easily to accommodate lossy hyperarcs as in [16]. For multicasting from a source $s \in N$ to a set of receivers $T \subseteq N$, the subgraph selection problem for traditional network-coded multicasting can be formulated as a linear program which is easily solved in polynomial time. In the case of a CDN, we recast the

This material is based upon work under a subcontract #069153 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-08-C-2013. Approved for Public Release, Distribution Unlimited.

problem with two distinct stages in time — a dissemination stage, and a fetch stage. Transmissions initiated or "pushed" by the source (i.e. origin server, assumed to be collocated with a node in the CDN) occur during the *dissemination* stage, while transmissions requested or "pulled" by the receivers (i.e. end users, assumed to be collocated with nodes in the CDN) occur during the *fetch* stage. Fig. 1 illustrates this time-expanded representation of the network. Storage in node memory is modeled as a flow *over time* from the dissemination stage to the fetch stage. We allow network coding during only the dissemination stage; fetch flows for individual receivers are *unicast* flows, and may occur independently and asynchronously. As in [7], we can formulate this problem as a linear program, with the objective of minimizing the expected total cost of content delivery, which in general can include dissemination, storage, and fetch costs.

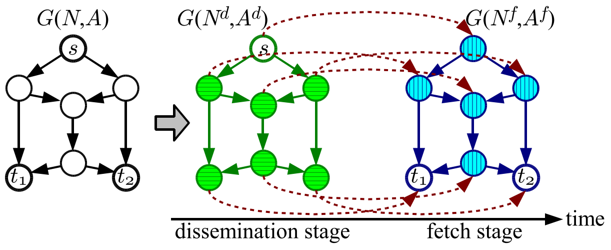


Fig. 1. Time-expanded representation of the network $G(N, A)$, over which source s wishes to multicast to the set of receivers $T = \{t_1, t_2\}$. The dashed arcs correspond to storage flows in the memory of the nodes.

A. Notation

s	source node (i.e. origin server)
T	set of receivers (i.e. end users)
R	multicast rate
A^d	set of arcs for the dissemination stage
A^s	set of arcs for the storage flows in node memory
A^f	set of arcs for the fetch stage
N^d	set of nodes for the dissemination stage
N^f	set of nodes for the fetch stage
$x_{a,t}$	flow on arc $a \in A^d \cup A^s \cup A^f$ for receiver $t \in T$
x_a	shared flow on arc $a \in A^d \cup A^s$, as a result of the flows for all receivers $t \in T$
c_a	cost per unit flow on arc $a \in A^d \cup A^s$
$c_{a,t}$	cost per unit flow on arc $a \in A^f$ for receiver $t \in T$
z_a	capacity of arc $a \in A^d \cup A^s \cup A^f$
α_t	expected number of requests for the content by receiver $t \in T$

In a general setting, we may assume that all nodes in the CDN are potential receivers, i.e. $T = N^f$. The costs per unit flow should reflect the monetary costs of bandwidth usage ($c_a, c_{a,t}$, $a \in A^d \cup A^f$) and data storage (c_a, $a \in A^s$), taking into account the relevant protocol overhead. We therefore expect the cost per unit flow on an arc to be higher during the fetch stage than the dissemination stage in practice, i.e. $c_{a,t} > c_a$ for the corresponding arcs in A^f and A^d respectively. The expected number of requests α_t can be estimated for each receiver t by considering usage patterns for the particular content.

B. Basic Linear Program Formulation

$$\text{minimize} \sum_{a\in A^d} x_a\, c_a + \sum_{a\in A^s} x_a\, c_a + \sum_{\substack{a\in A^f\\ t\in T}} \alpha_t\, x_{a,t}\, c_{a,t} \qquad (1)$$

subject to

$$x_a \geq x_{a,t}\ , \qquad \forall\ t\in T,\ a\in A^d\cup A^s \qquad (2)$$

$$z_a \geq x_{a,t} \geq 0\ , \qquad \forall\ t\in T,\ a\in A^d\cup A^s\cup A^f \qquad (3)$$

$$\sum_{\{a|\text{tail}(a)=i\}} x_{a,t} - \sum_{\{a|\text{head}(a)=i\}} x_{a,t} = \begin{cases} R & \text{if } i=s,\\ -R & \text{if } i=t,\\ 0 & \text{otherwise,}\end{cases}$$

$$\forall\ t\in T,\ i\in N^d\cup N^f \qquad (4)$$

The three summation terms in the objective function (1) represent the expected total dissemination, storage, and fetch costs, respectively. Although our formulation assumes independent and asynchronous unicast flows during the fetch stage, we note that in practice, simultaneous fetch flows on an arc can be shared using network coding to reduce costs. Constraint (2) ensures that the resultant shared flow on each arc is able to support the required flow for each receiver on that arc, (3) ensures that the flow on each arc is realizable within its capacity, and (4) reflects the conservation of flow through each node.

C. Modified Formulations

1) Potential Storage Nodes: To restrict the set of potential storage nodes to some subset $S \subseteq N$, we can set $z_a = 0$ if the node corresponding to $a \in A^s$ is not in S. The problem becomes analogous to that of constrained mirror placement in [2].

2) Storage Budget Constraint: In addition to specifying the storage capacity for individual nodes (i.e. z_a, $a \in A^s$), we can also impose an aggregate storage budget over all nodes, i.e. $\sum_{a\in A^s} x_a \leq Z_s$, where Z_s is the storage budget. This is analogous to the constraint on the number of mirrors in [1], [2], and the storage capacity budget in [13].

3) k-hop Fetch Constraint: For *quality of service*, we can bound the delay incurred when accessing content by restricting fetch flows to within the k-hop neighborhood of each receiver. This can be done by adding the equality constraint $x_{a,t} = 0$ for each arc $a \in A^f$ outside the k-hop neighborhood of receiver t. Other fetch constraints can also be applied, e.g. maximum round-trip time.

4) Fetch Load Constraint: To bound the expected load on nodes and arcs during the fetch stage, we can introduce the constraint $\sum_{t\in T} \alpha_t\, x_{a,t} \leq \beta_a\, z_a$, $a \in A^s \cup A^f$, where β_a is the load factor for arc a. This performs *load balancing* by restricting the expected storage flow served by each node ($a \in A^s$), and the expected fetch flow on each arc ($a \in A^f$).

5) Dissemination Stage Receivers: To have a node receive the content at the end of the dissemination stage without having to fetch, we can add the corresponding node in N^d to the set of receivers T. This is equivalent to forcing the node to be a storage node for the whole content.

6) Delivery of Multiple Content: To accommodate the delivery of multiple content from possibly different source nodes, we introduce separate flows for individual content. The problem becomes analogous to that of object placement in [13], and can be described by the following linear program:

$$\text{minimize} \sum_{a\in A^d} x_a\, c_a + \sum_{a\in A^s} x_a\, c_a + \sum_{\substack{a\in A^f\\ t\in T\\ w\in W}} \alpha_t^{(w)}\, x_{a,t}^{(w)}\, c_{a,t} \tag{5}$$

subject to

$$x_a^{(w)} \geq x_{a,t}^{(w)}\ , \quad \forall\ w\in W,\ t\in T,\ a\in A^d\cup A^s \tag{6}$$

$$z_a \geq x_a \geq \sum_{w\in W} x_a^{(w)}\ , \quad \forall\ a\in A^d\cup A^s \tag{7}$$

$$z_a \geq x_{a,t}^{(w)}\ , \quad \forall\ w\in W,\ t\in T,\ a\in A^f \tag{8}$$

$$x_{a,t}^{(w)} \geq 0\ , \quad \forall\ w\in W,\ t\in T,\ a\in A^d\cup A^s\cup A^f \tag{9}$$

$$\sum_{\{a|\text{tail}(a)=i\}} x_{a,t}^{(w)} - \sum_{\{a|\text{head}(a)=i\}} x_{a,t}^{(w)} = \begin{cases} R_w & \text{if } i=s_w,\\ -R_w & \text{if } i=t,\\ 0 & \text{otherwise,}\end{cases}$$
$$\forall\ w\in W,\ t\in T,\ i\in N^d\cup N^f \tag{10}$$

We denote the set of content or objects by W, the source node for content w by s_w, the multicast rate for content w by R_w, and the expected number of requests by receiver $t\in T$ for content w by $\alpha_t^{(w)}$, where $w\in W$. Flows for individual content are distinguished by their (w) superscript. Note that we allow network coding during only the dissemination stage and among flows for the same content.

D. Storage for Robustness Against Node or Arc Failures

Distributed storage can be used to improve robustness of data availability in unreliable networks. Intuitively, in a network where nodes or arcs fail probabilistically, the probability of a receiver being able to fetch data successfully increases with the amount of redundant storage and the proximity of storage nodes.

Suppose we wish to ensure that each receiver can still successfully access content in the event that some nodes or arcs fail during the fetch stage. An exact but prohibitively complex way to approach this is to consider the exhaustive set of mutually exclusive failure events (including the zero-failure event), each associated with a probability of occurrence, and a set of nodes or arcs that fail. We then replace each receiver $t\in T$ with a set of *virtual receivers*, one for each failure event that affects it, allowing fetch flows only on the unaffected nodes and arcs, i.e. fetch flows on the failed nodes and arcs are forced to be zero. The objective function would be modified to include the fetch cost incurred by each virtual receiver, weighted by the probability of the corresponding failure event, so that it continues to express the expected total cost. The k-hop fetch constraint (see Section III-C3) reduces the number of virtual nodes required, for example, to protect against the failure of up to any m arcs in the network, from $\sum_{i=0}^{m}\binom{|A|}{i}$ virtual receivers per receiver if there were no hop constraint on fetching, to $\sum_{i=0}^{\min\{m,d^-(t)\}}\binom{d^-(t)}{i}$ virtual receivers for each receiver $t\in T$ if a 1-hop fetch constraint was applied, where $d^-(t)$ is the indegree of node t. However, the number of virtual receivers still grows exponentially with the number of hops k.

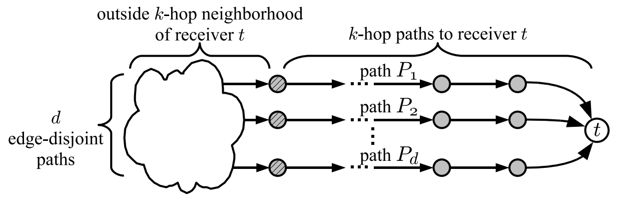

outside k-hop neighborhood of receiver t
k-hop paths to receiver t
d edge-disjoint paths
path P_1
path P_2
path P_d
t

Fig. 2. Example demonstrating the use of virtual receivers to achieve a minimum probability of successful content delivery. Virtual receiver v_0 corresponds to the zero-failure event, while virtual receiver v_i, $i=1,2,\ldots,d$, corresponds to the event where only the arcs on paths $P_j, j\neq i$ are allowed to carry fetch flow for v_i.

To allow flexibility in the number of hops k in the fetch constraint while remaining tractable, we can consider the following approach whose fetch success probability can be lower bounded as a function of k, the node (arc) failure probability, and the number of disjoint paths in the k-hop neighborhood of the receivers. Fig. 2 gives an illustration. Suppose each node (arc) in the network fails independently with probability p. If a receiver t has d node (arc)-disjoint paths in its k-hop neighborhood, then by using the specified $(d+1)$ virtual receivers, receiver t would achieve a success probability $\geq \mathbb{P}(\text{at most one path fails}) = (1-p)^{k(d-1)}(d-(d-1)(1-p)^k)$. For simplicity, the objective function includes the fetch costs only for the zero-failure event.

E. Problem Size Reduction

In the interest of tractability, we can reduce the size of the linear program by picking only a subset of nodes in the CDN as receivers. One heuristic would be to choose a set of receivers so that each node in the CDN is at most k hops from some receiver. Such a set could be greedily constructed, e.g. by adding nodes in descending order of degree, while skipping over those that are already in the k-hop neighborhood of a previously added node. A more general approach is to organize the nodes in the CDN into an appropriate hierarchy so that we need only consider a small number of nodes at the *top* level and the arcs between them; other nodes at *lower* levels will be assumed to be collocated with a top level node. Such a hierarchical organization of nodes may follow naturally from the geographical or administrative topology of the CDN, e.g. national, regional, and institutional levels proposed in [17].

IV. Performance Evaluation

To demonstrate the benefits of our formulation, we evaluate its performance for delivering multiple content. This is traditionally approached as an object placement problem, which can be formulated as a multi-commodity generalization of the NP-hard minimum k-median problem [13], [1]. We describe this formulation with the following mixed integer linear program (*cf.* Section III-C6):

$$\text{minimize} \sum_{\substack{a\in A^f\\ t\in T\\ w\in W}} \alpha_t^{(w)}\, x_{a,t}^{(w)}\, c_{a,t} \tag{11}$$

subject to

$$x_a^{(w)} \geq x_{a,t}^{(w)}\,, \quad \forall\ w\in W,\ t\in T,\ a\in A^d\cup A^s \tag{12}$$

$$z_a \geq x_a \geq \sum_{w\in W} x_a^{(w)}\,, \quad \forall\ a\in A^d\cup A^s \tag{13}$$

$$z_a \geq x_{a,t}^{(w)}\,, \quad \forall\ w\in W,\ t\in T,\ a\in A^f \tag{14}$$

$$x_{a,t}^{(w)} \geq 0\,, \quad \forall\ w\in W,\ t\in T,\ a\in A^d\cup A^s\cup A^f \tag{15}$$

$$\sum_{\{a|\text{tail}(a)=i\}} x_{a,t}^{(w)} - \sum_{\{a|\text{head}(a)=i\}} x_{a,t}^{(w)} = \begin{cases} R_w & \text{if } i=s_w,\\ -R_w & \text{if } i=t,\\ 0 & \text{otherwise,}\end{cases}$$

$$\forall\ w\in W,\ t\in T,\ i\in N^d\cup N^f \tag{16}$$

$$\sum_{a\in A^s} x_a \leq k \tag{17}$$

$$x_a^{(w)} \in \{0, R_w\}\,,\ \forall\ w\in W,\ a\in A^s \tag{18}$$

We note that the objective function (11) comprises only the expected fetch cost. Also, the storage budget k is to be allocated over the set of content or objects W (17), and objects must be stored in their entirety (18). For brevity, we refer to this multi-commodity version of the **minimum k-median formulation** as **KMF**. For comparison, we use the linear program in Section III-C6, with zero storage costs and the storage budget constraint $\sum_{a\in A^s} x_a \leq k$. We refer to this **network coding formulation** as **NCF**.

We also make the following simplifying assumptions: unit dissemination costs $c_a = C_d$, $a\in A^d$, unit fetch costs $c_{a,t} = C_f$, $a\in A^f$, $t\in T$, and expected number of requests $\alpha_t^{(w)} = \alpha^{(w)}$, $t\in T$, $w\in W$, where $C_d \geq 0$, $C_f > 0$, and $\alpha^{(w)} \geq 0$, $w\in W$ are constants. Following the analysis in [13], we assume a Zipf-like request distribution over the set of objects W, which gives $\alpha^{(w_i)} \triangleq \frac{1}{i^m}\left(\sum_{j=1}^{|W|}\frac{1}{j^m}\right)^{-1}\alpha$, where $w_i\in W$ is the i^{th} most popular object, $\alpha > 0$ is the expected total number of requests by a receiver over all objects, and m is the skewness parameter of the request distribution. It follows that the objective functions in KMF and NCF can be replaced respectively by

$$\sum_{\substack{a\in A^f\\ t\in T\\ w_i\in W}} \frac{1}{i^m} x_{a,t}^{(w_i)}\,, \qquad \text{and} \tag{19}$$

$$\theta \sum_{a\in A^d} x_a + \sum_{\substack{a\in A^f\\ t\in T\\ w_i\in W}} \frac{1}{i^m}\left(\sum_{j=1}^{|W|}\frac{1}{j^m}\right)^{-1} x_{a,t}^{(w_i)}\,, \tag{20}$$

where $\theta \triangleq \frac{C_d}{\alpha C_f}$, without changing the optimal solutions. Using the expected total dissemination and fetch cost

$$C_d \sum_{a\in A^d} x_a + \alpha C_f \sum_{\substack{a\in A^f\\ t\in T\\ w_i\in W}} \frac{1}{i^m}\left(\sum_{j=1}^{|W|}\frac{1}{j^m}\right)^{-1} x_{a,t}^{(w_i)}$$

as the performance measure, we compute the relative performance of NCF with respect to KMF:

$$\frac{\left(C_d \sum_{a\in A^d} x_a + \alpha C_f \sum_{a\in A^f, t\in T, w_i\in W} \frac{1}{i^m}\left(\sum_{j=1}^{|W|}\frac{1}{j^m}\right)^{-1} x_{a,t}^{(w_i)}\right)_{\text{NCF}}}{\left(C_d \sum_{a\in A^d} x_a + \alpha C_f \sum_{a\in A^f, t\in T, w_i\in W} \frac{1}{i^m}\left(\sum_{j=1}^{|W|}\frac{1}{j^m}\right)^{-1} x_{a,t}^{(w_i)}\right)_{\text{KMF}}}$$
$$= \frac{\left(\theta \sum_{a\in A^d} x_a + \sum_{a\in A^f, t\in T, w_i\in W} \frac{1}{i^m}\left(\sum_{j=1}^{|W|}\frac{1}{j^m}\right)^{-1} x_{a,t}^{(w_i)}\right)_{\text{NCF}}}{\left(\theta \sum_{a\in A^d} x_a + \sum_{a\in A^f, t\in T, w_i\in W} \frac{1}{i^m}\left(\sum_{j=1}^{|W|}\frac{1}{j^m}\right)^{-1} x_{a,t}^{(w_i)}\right)_{\text{KMF}}} \triangleq \Phi_\theta$$

To obtain the dissemination costs in KMF, we solve a separate traditional network coded multicasting problem with the selected storage nodes as receivers, with the objective of minimizing the total dissemination cost.

For a given network, set of objects W, and choice of k, m, and R_w, $w\in W$, the optimal solutions and the relative performance Φ_θ can be parameterized by θ; therefore we vary θ as the independent variable in our comparisons. We apply both formulations on randomly generated networks with symmetrical point-to-point arcs, all nodes as receivers, multicast rates $R_w = 1$, $w\in W$, and arc capacities $z_a = 1$, $a\in A^d\cup A^s\cup A^f$. A network is generated by starting with the empty graph and adding symmetrical point-to-point arcs selected uniformly at random until the graph becomes connected. The source node s_w for each object $w\in W$ is selected uniformly at random from the set of all nodes.

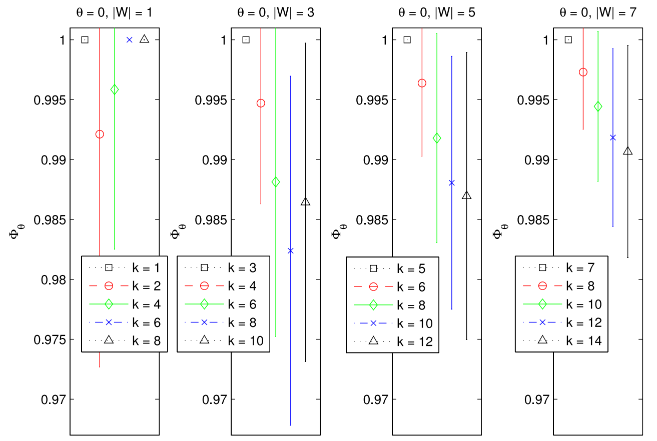
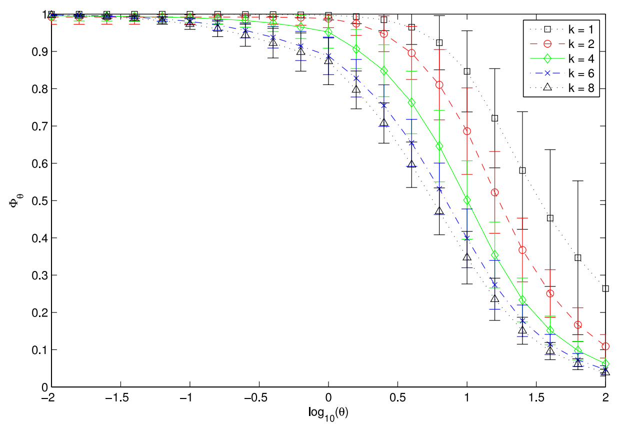
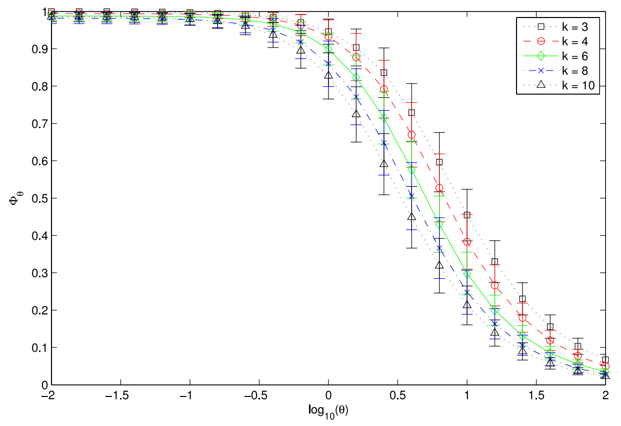


Fig. 3. Relative performance of NCF with respect to KMF (Φ_θ) for $\theta \triangleq \frac{C_d}{\alpha C_f} = 0$, for 1, 3, 5, 7 objects ($|W|$), and storage budgets $k = |W| + \{0, 1, 3, 5, 7\}$. Each data point represents the mean value and standard deviation of Φ_θ over 50 random networks.

Figs. 3, 4, 5 summarize our results for the placement of 1, 3, 5, 7 objects with request distribution skewness parameter $m = 0.9$ (as in [13]), for different storage budgets k and values of $\theta \triangleq \frac{C_d}{\alpha C_f}$, over 50 randomly generated 15-node



(a) Number of objects $|W| = 1$



(b) Number of objects $|W| = 3$

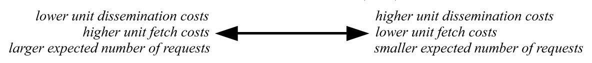
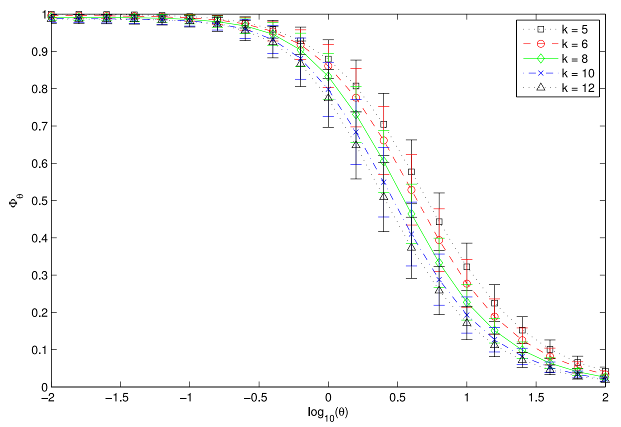
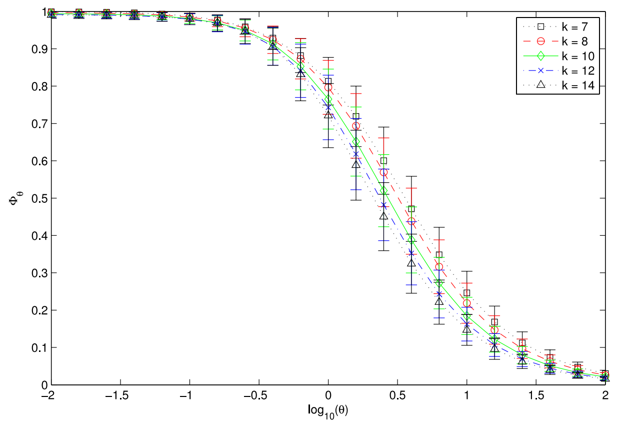

lower unit dissemination costs
higher unit fetch costs
larger expected number of requests
higher unit dissemination costs
lower unit fetch costs
smaller expected number of requests

Fig. 4. Relative performance of NCF with respect to KMF (Φ_θ) against $\theta \triangleq \frac{C_d}{\alpha\, C_f}$, for (a) 1 and (b) 3 objects ($|W|$), and storage budgets $k = |W| + \{0, 1, 3, 5, 7\}$. Each data point represents the mean value and standard deviation of Φ_θ over 50 random networks.



(a) Number of objects $|W| = 5$



(b) Number of objects $|W| = 7$

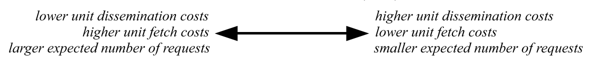

lower unit dissemination costs
higher unit fetch costs
larger expected number of requests
higher unit dissemination costs
lower unit fetch costs
smaller expected number of requests

Fig. 5. Relative performance of NCF with respect to KMF (Φ_θ) against $\theta \triangleq \frac{C_d}{\alpha\, C_f}$, for (a) 5 and (b) 7 objects ($|W|$), and storage budgets $k = |W| + \{0, 1, 3, 5, 7\}$. Each data point represents the mean value and standard deviation of Φ_θ over 50 random networks.

networks. Fig. 3 shows that when $\theta = 0$ (i.e. dissemination is free), NCF performs only very slightly better than KMF. This is to be expected since both formulations are using the same objective function (19), (20), and differ only in that NCF allows objects to be algebraically coded and split across different nodes. This advantage would be more pronounced when node storage capacities and arc capacities are severely limited. Figs. 4, 5 suggest that NCF's lead over KMF improves as (i) θ increases (i.e. dissemination becomes relatively more expensive), (ii) the number of objects $|W|$ increases, and (iii) the storage budget k increases. In particular, we note that even at $\theta = 1$ (which could correspond to $C_d = 0.5$, $C_f = 1$, $\alpha = 0.5$, for example), we can already observe a cost reduction of almost 20% for $|W| = k = 7$.

V. Discussion

A. Implementation Considerations

Network coding in our formulation can be implemented using random linear codes [18], [19]. The resulting optimal solution assumes a fluid model; how closely we approach this in practice depends on how we translate fractional flows to integral numbers of packets. If the generation size (i.e. number of source packets being coded together) is large, then simple rounding up may suffice. Other integral network coding strategies are also possible [20].

We have assumed that the origin server and end users are collocated with CDN nodes in our formulation. In practice, a domain name system (DNS) could be used to direct the origin server and end users to the nearest available CDN node, as is used in Akamai [21].

Packet routing through the CDN could be accomplished using source routing. The origin server solves the linear program, and disseminates the content along fixed routes; downstream nodes will perform network coding as determined by the origin server. A robust routing protocol should be able to work around node or arc failures that occur midstream. The origin server also notifies all potential end users of the storage nodes from which to fetch content, and the amount of content to fetch from each of them. As this may be impractical for very large networks, we could, alternatively, use a DNS to store this information for each end user; the origin server would update the relevant entries and leave the end users to resolve their fetch queries. To fetch content, an end user would contact the relevant storage nodes, which would then send the necessary amount of content via the shortest paths.

Information about the network state should be collected as often as needed to accurately compute the parameters of the linear program (e.g. arc costs and capacities, expected number of requests by end users). In a dedicated CDN, we expect such information to be available through the network monitoring system; otherwise, each node could just broadcast a status report periodically. In very large networks where flooding is prohibitively costly, we could collate the information at predetermined nodes, e.g. the root nodes at each level of the hierarchy.

Nodes in the CDN could also cache content flowing through them during the fetch stage, so as to reduce bandwidth consumption and delay in fetching. In particular, we expect better caching performance with network coding because a coded packet returned by a cache is more likely to be innovative than an uncoded packet.

B. Augmenting with a Peer-to-Peer (P2P) Network

A hybrid CDN–P2P network achieves better scalability than a pure CDN since end users help contribute bandwidth, storage, and computation resources to deliver content [22], [23]. End users can access content through one or more nearby nodes in the CDN, or through other peers. In such a hybrid network, the CDN provides a reliable backbone for content delivery, and prevents severe service degradation in the face of high churn rates among peers.

CDN nodes are also natural candidates for trackers which help coordinate interactions between end users, as used in BitTorrent [24]. Furthermore, we can modify our formulation to support various content storage and lookup mechanisms in P2P systems [25]. For instance, for fast lookups of storage nodes in the CDN by end users, we may adopt a distributed hash table (DHT) approach [26], [27] whereby we specify a set of dissemination stage receivers (see Section III-C5) according to the hash of the content.

VI. Conclusion and Future Work

We presented a unified linear program formulation for optimal content delivery in CDNs. Our simulation results suggest that the formulation performs significantly better than the traditional minimum k-median formulation for the delivery of multiple content, even under modest circumstances (small network, few objects, low storage budget, low dissemination costs). We look forward to addressing the more challenging problem of content delivery in dynamic environments (e.g. mobile ad hoc networks) which demand greater robustness against topology changes, and node and arc failures.

References

[1] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, “On the placement of Web server replicas,” in *Proc. INFOCOM*, Apr. 2001.

[2] E. Cronin, S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, “Constrained mirror placement on the Internet,” *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1369–1382, Sep. 2002.

[3] P. Krishnan, D. Raz, and Y. Shavitt, “The cache location problem,” *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 568–582, Oct. 2000.

[4] R. L. Carter and M. E. Crovella, “Server selection using dynamic path characterization in wide-area networks,” in *Proc. INFOCOM*, Apr. 1997.

[5] S. Seshan, M. Stemm, and R. H. Katz, “SPAND: Shared passive network performance discovery,” in *Proc. USENIX Symp. Internet Technol. and Syst.*, Dec. 1997.

[6] J. Pan, Y. T. Hou, and B. Li, “An overview of DNS-based server selections in content distribution networks,” *Comput. Netw.*, vol. 43, pp. 695–711, 2003.

[7] A. Jiang, “Network coding for joint storage and transmission with minimum cost,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2006.

[8] F. Douglis and M. F. Kaashoek, “Scalable Internet services,” *IEEE Internet Comput.*, vol. 5, no. 4, pp. 36–37, Jul./Aug. 2001.

[9] Akamai Annual Report 2007, Akamai. [Online]. Available: http://www.akamai.com/dl/investors/Akamai_07_Annual_Report.pdf

[10] S. Guha, A. Meyerson, and K. Munagala, “Hierarchical placement and network design problems,” in *Proc. Symp. Found. Comput. Sci.*, Nov. 2000.

[11] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, “Placement algorithms for hierarchical cooperative caching,” in *Proc. ACM–SIAM Symp. Discrete Algorithms*, 1999.

[12] T. Kelly and D. Reeves, “Optimal Web cache sizing: scalable methods for exact solutions,” *Comput. Commun.*, vol. 24, pp. 163–173, 2001.

[13] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, “On the optimization of storage capacity allocation for content distribution,” *Comput. Netw.*, vol. 47, pp. 409–428, 2005.

[14] D. Dowdy and D. Foster, “Comparative models of the file assignment problem,” *ACM Comput. Surveys (CSUR)*, vol. 14, no. 2, pp. 287–313, Jun. 1982.

[15] B. Gavish and O. R. L. Sheng, “Dynamic file migration in distributed computer systems,” *Commun. ACM*, vol. 33, no. 2, pp. 177–189, Feb. 1990.

[16] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, “Minimum-cost multicast over coded packet networks,” *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2608–2623, Jun. 2006.

[17] J. Ni and D. H. K. Tsang, “Large-scale cooperative caching and application-level multicast in multimedia content delivery networks,” *IEEE Commun. Mag.*, vol. 43, no. 5, pp. 98–105, May 2005.

[18] T. Ho and D. Lun, *Network Coding: An Introduction*. Cambridge University Press, 2008.

[19] P. A. Chou, Y. Wu, and K. Jain, “Practical network coding,” in *Proc. Annu. Allerton Conf. Commun., Control, and Comput.*, Oct. 2003.

[20] T. Cui and T. Ho, “Minimum cost integral network coding,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2007.

[21] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, “Globally distributed content delivery,” *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, Sep./Oct. 2002.

[22] D. Xu, S. S. Kulkarni, C. Rosenberg, and H.-K. Chai, “Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution,” *Multimedia Syst.*, vol. 11, no. 4, pp. 383–399, 2006.

[23] J. Wu, Z. Lu, B. Liu, and S. Zhang, “PeerCDN: A novel P2P network assisted streaming content delivery network scheme,” in *Proc. IEEE Int. Conf. Comput. and Inf. Technol.*, Jul. 2008.

[24] B. Cohen, “Incentives build robustness in BitTorrent,” in *Proc. Workshop Econ. Peer-to-Peer Syst.*, May 2003.

[25] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell, “A survey of peer-to-peer storage techniques for distributed file systems,” in *Proc. Int. Conf. Inf. Technol.: Coding and Comput. (ITCC)*, Apr. 2005.

[26] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems,” in *Proc. IFIP/ACM Int. Conf. Distrib. Syst. Platforms*, Nov. 2001.

[27] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for Internet applications,” in *Proc. ACM SIGCOMM*, Aug. 2001.