Fiftieth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
October 1 - 5, 2012

An Optimized Distributed Video-on-Demand Streaming System: Theory and Design

Kangwook Lee[†], Hao Zhang[†], Ziyu Shao[§], Minghua Chen[§], Abhay Parekh[†] and Kannan Ramchandran[†]
[†]Department of EECS
University of California at Berkeley, Berkeley, CA
{kw1jjang, zhanghao, parekh, kannanr}
@eecs.berkeley.edu
[§]Department of Information Engineering
The Chinese University of Hong Kong, Hong Kong
{zyshao,minghua}
@ie.cuhk.edu.hk

***Abstract*— We present a general framework for a distributed VoD content distribution problem by formulating an optimization problem yielding a highly distributed implementation that is highly scalable and resilient to changes in demand. Our solution takes into account several individual node resource constraints including disk space, network link bandwidth, and node-I/O degree bound. First, we present a natural formulation that is NP-hard. Next, we design a simple fractional storage architecture based on codes to "fluidify" the content, thereby yielding a convex content placement problem. Third, we use a recently developed Markov approximation technique to solve the NP-hard problem of topology selection under node degree bound, and propose a simple distributed solution. We prove analytically that our algorithm achieves close-to-optimal performance. We establish via simulations that the system is robust to changes in user demand or in network condition and churn.**

I. Introduction

The shape of the internet is changing. On the one hand, large internet exchanges and datacenters have made it possible to centralize a lot of compute and storage resources. It is clear that established players such as Netflix, Google and Amazon are more likely to exploit the reliability and economies of scale that come from such architectures, and that is of course what they are doing. On the other hand, the edge of the internet is growing exponentially. Phones, tablets, laptops, ebook readers are growing in power and sophistication to the point that that they resemble the desktop computers of a few years ago. But fully 50% of the traffic of the internet does not originate from data centers and hierarchical CDNs [1]. This traffic consists of applications such as file sharing and P2P, and relies more heavily on edge-devices which take on the role of potentially unreliable servers. As we play out the evolution of content and the internet it seems clear that both kinds of content distribution will co-exist. Videos of police action in an oppressive state are easier to detect and throttle in a centralized architecture than a distributed one, and there will always be situations in which small groups of individuals will want to share content without the "prying eyes" of a media giant.

In this paper we present a highly distributed edge-based scheme where the content is streamed video and has quality of service constraints. The demand for these videos is not specified to the system. Each edge device has limited storage but can store content from any movie (even ones that the owner of that device is not watching). The devices are assumed to have limited connectivity and the network is allowed to be unreliable. As demand and network connectivity fluctuates so does what is stored at each node. In other words our goal is to test the limits of how unreliable and distributed we can make the infrastructure and still meet the stringent quality of service constraints of streamed video. Central to our architecture is the existence of a single reliable server or Seedbox [2], that can fill in the gaps of service when our distributed algorithms cannot meet QoS constraints and the objective of our distributed algorithms is to ensure that for any set of adverse network conditions, the load on the Seedbox is minimized.

Our approach is the following. First, we formulate the problem as a static convex optimization problem and then show that a highly distributed algorithm converges to the optimal allocation. We then examine, via simulations, the dynamic problem in which network conditions change, and the demand for the underlying content fluctuates significantly. We find that our scheme is highly resilient to changes in network and in demand in addition to being near optimal in a static setting.

More concretely, in formulating the optimization problem we jointly solve the following problems:

1) Content Placement: What content should be stored at each device/node given the storage constraints, network capacity and current demand?
2) Overlay Topology: Given that each node can only support a bounded number of end devices, how should end devices be matched to nodes?
3) Minimal Server Load: When there is no available node that can serve an end device to watch a specific piece of content from, the Seedbox "fills in the gap" by streaming directly to it. We wish to minimize such occurrences.

We illustrate these problems further in the example depicted in Figure 1. The system has two 1 GB videos, A and B, which must be delivered at a streaming rate of 1 Mbps. There are 4 users: two request video A and two request video B. The three cache nodes are constrained by bandwidth, maximum degree (a bound on the number of simultaneously supported streaming connections) and storage. Figure 1(b)

978-1-4673-4539-2/12/$31.00 ©2012 IEEE

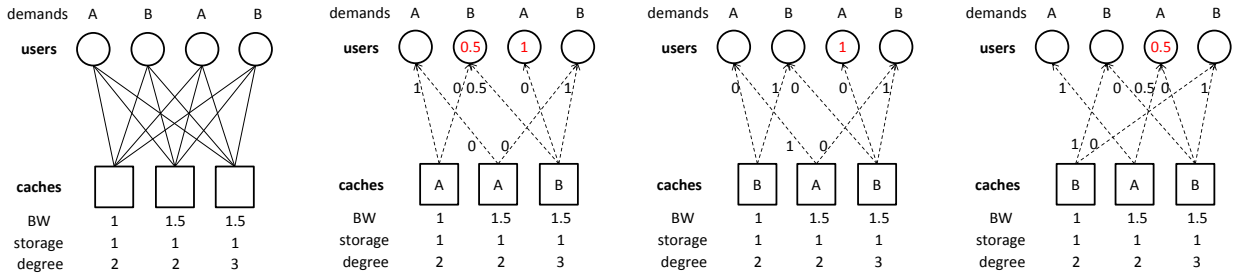


Fig. 1: A simple example of VoD caching problem. The system has two videos of size 1 GB and rate 1 Mbps and there are 2 users requesting each video. The system employs 3 cache nodes with constraints on storage, bandwidth and out-degree as shown in (a). The problem is to decide, for each cache, which videos to store, which users to connect to, and how much bandwidth to allocate for each user. These questions are coupled. The connections between the cache nodes and users in (b) form a "bad" topology, and the content placement is non-optimal. The content placement in (c) is "good". In (d), the topology is "good", and with the same content placement strategy in (c), only one user is in deficit of half of a video. In general, finding the "best" storage, bandwidth and topology combination is a combinatorially-hard problem.

shows that under a certain "bad" topology and a "bad" content allocation scheme, demand cannot be satisfied. In Figure 1(c), a "good" content placement strategy is chosen. In Figure 1(d), a "good" topology is also chosen. As we can see, the three problems enumerated earlier are closely related. Further, taken in isolation each problem is hard: there are an exponential number of possible topologies (from which we must select in a distributed manner) and as we will see (although it may be clear to some readers even at this point), the content selection problem for a fixed topology, is also NP-hard.

The contributions of this paper are four-fold. To the best of our knowledge, we are the first to jointly optimize topology selection, content placement, and link rate allocation in the design of VoD systems. Second, we design a fractional coding architecture that allows for "fluidification" of the content placement problem, thereby converting a combinatorial VoD content placement problem into a convex problem that admits a simple distributed algorithm. Third, we overcome the difficulty of the combinatorial topology graph selection problem by applying a recently-developed and novel Markov approximation technique, and design a simple and distributed "soft-worst-neighbor-choking" algorithm. Fourth, we give a theoretical proof that our solution is close-to-optimal and that the optimality gap diminishes when the number of users becomes large. We validate that our algorithm outperforms existing results via extensive real world trace simulations. While our theoretical analysis is based on a "static" setting where the demand and supply resources are fixed, we also show in our experiments that our algorithm works well even for "dynamic" cases. Our strong findings suggest that a functional implementation of our solution would perform well in practice, and this is the focus of our future work.

II. Related Work

The optimization of VoD systems has received wide attention [3]–[9]. Almeida et al. [3] studied the delivery cost minimization problem under a fixed topology by optimizing over content replication and routing. Boufkhad et al. [4] investigated the problem of maximizing the number of videos that can be served by a collection of peers. Zhou et al. [5] focused on minimizing the load imbalance of video servers while maximizing the system throughput. Tan and Massoulie [8] studied the problem of optimal content placement in P2P networks. Their goal is to maximize the utilization of peer uplink bandwidth resources. Optimal content placement strategies are identified in a particular scenario of limited content catalog under the framework of loss networks. Their work assumes that the peers' storage capacity grows unboundedly with system size. In contrast, our work does not make any assumption on the storage capacities and also takes into overlay topology. Applegate et al. [9] formulated the problem of content placement into a mixed integer program (MIP) that takes into account constraints such as disk space and link bandwidth. However, they assume the knowledge of content popularity under a fixed topology and a video is either stored in full or not stored. In our work, we use a class of network codes that enables fractional storage and do not assume any prior knowledge of demand is given. We also optimize over the topology graph selections.

With regard to network resource utilization, Borst et al. [10] solved a link bandwidth utilization problem assuming a tree structure with limited depth. An LP is formulated, and under the assumption of symmetric link bandwidth, demand, and cache size, a simple local greedy algorithm is designed to find a close-to-optimal solution. Valancius et al. [11] propose an LP-based heuristic to calculate the number of video copies placed at customer home gateways. The network topology in our work is not constrained to be a tree, and the video request patterns can be arbitrary in different network areas. Zhou and Xu [12] aimed to minimize the load imbalance among servers subject to disk space egress link capacity from servers. In contrast, we consider the link capacity constraints that may exist anywhere in the network.

Topology building is also an important design dimension and has been studied in various works [13]–[15]. While

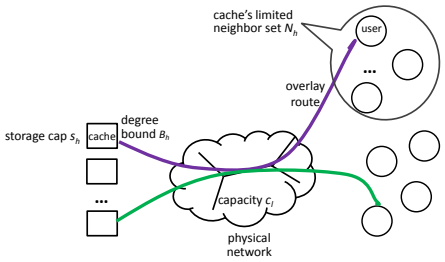


Fig. 2: Caches and users connected by a physical network. The link capacity constraints can exist arbitrarily anywhere in the network.

most works focus on enforcing locality-awareness and/or improving ISP-friendliness, they make the simple assumption that the graph is fully connected, *i.e.,* no node-degree-bound is taken into consideration. Zhang et al. solves the problem of optimal P2P streaming under node degree constraints [16]. However their topology selection algorithm depends on global statistics which are easily accessible under a live-streaming scenario. In VoD, directly applying their technique requires global statistics of all users' utility functions, which can create an enormous overhead. Our distributed algorithm requires knowledge of only local information of neighboring overlay link rates.

To the best of our knowledge, we are unaware of any other works to jointly optimize topology graph selection, content placement and link rate allocation. Our solution is fully distributed and adapts well to system dynamics as we will show in the simulations.

III. Problem Formulation

In this section, we define the VoD optimization problem. Our formulation assumes a static setting: the video catalog, the set of users and subscriptions, and the set of caches are fixed. In section IV and section V, we find a fully distributed algorithm and prove it achieves near-optimal performance. In section VI, we apply the proposed algorithm to dynamic setting and show its performance through simulations.

As illustrated in Figure 2, a set of caches(H) and a set of users(U) are connected by a fixed physical network which consists of links with capacity. Caches and users are connected by a overlay graph configuration, g, which is expressed by the set of overlay links R and the corresponding routing matrix A. We denote the set of overlay links and the routing matrix by R^g and A^g under a graph configuration g. Each overlay link $r \in R$, consists of a set of underlay links, $L_r \subset L$ and we say $l \in r$ if $l \in L_r$. An overlay link $r = (h, u)$ enables cache node h to send data to node u in the overlay graph by setting up TCP/UDP connections. The routing matrix $A := (A_{lr}, (l, r) \in L \times R)$ is defined as usual where $A_{lr} = 1$ if $l \in r$ and 0 otherwise, We denote the connected neighbor of node v by N_v^g. Node v cannot connect to more than B_v users, which leads the node-I/O constraints. Let $G = \{\forall g || N_v^g| \leq B_v \ \ \forall v \in H \cup U\}$ be the set of possible overlay graphs. Let link $l \in L$ have a capacity c_l, and let x_r be the rate on the overlay link r. This leads to

TABLE I: Key Notations

Parameters	**Definition**
H, U, M	set of caches / users / movies
U_m	set of users watching video m
γ_m, β_m	video m's streaming rate and size
s_h	storage capacity of cache h
G, B_v	set of feasible overlay graphs / I/O constraints
L, c_l	set of underlay links / link capacity
Auxiliary Variables	**Definition**
θ_l, q_r	shadow price of link l / route r
q_r	$q_r = \sum_{l \in r} \theta_l$ is the shadow price of route r
$\lambda_r, \Sigma_{h,m}$	demand index of route r / movie m at cache h
ω_h	storage price of cache h
Decision Variables	**Definition**
x_r	route rate of r
W_{hm}	storage of video m on cache h
g, A^g, R^g	overlay graph / routing matrix / overlay links
N_v^g	connected neighborhood of v under g
p_g	probability of each topology graph g

natural routing constraints as follows: $Ax \leq c$, where x is the column vector of the overlay rates x_r and c is the column vector of link capacity constraints c_l.

The video catalog consists of videos in the set M. Each video m has size β_m and is streamed at a constant rate of γ_m. Since we assume a fixed demand in this model, we denote the set of users watching m as U_m. To model the storage constraints, let s_h be the storage capacity of cache node h, and denote by $s := (s_h, h \in H)$ the column vector of the storage capacities. Let $W := (W_{hm}, h \in H, m \in M)$ be the storage matrix where $W_{hm} \in \{0, 1\}$ indicates if video m is stored on cache node h ($W_{hm} = 1$) or not ($W_{hm} = 0$). Denote by $\beta := (\beta_m, m \in M)$ the vector of the sizes (in MB) of all videos. The storage constraints can then be expressed by $W\beta \leq s$. Availability constraints are also modeled : caches can only serve stored movies. From cache h to user u, the streaming rate can be only 0 if cache does not store the movie m which is being viewed by the user. If the cache stores the movie, the streaming rate can be anything no greater than γ_m. It can be equivalently expressed as $x_{r:=(h,u)} \leq W_{hm}\gamma_m$.

Let $z_u = \sum_{r=(h,u):h \in N_u^g} x_r$ be the total received rate of user u, and $V^u(z)$ be a concave function that represents the utility of user u when the received rate is z. Table I lists all relevant notations.

Now, we have the following optimization problem. The objective is to find a graph g, content placement W, and rate allocation x, which jointly maximize the sum of user utilities under constraints.

$$\max_{\substack{g \in G \\ x_r \geq 0, W_{hm} \in \{0,1\}}} \quad \sum_{u \in U} V^u(z_u)$$
$$\text{s.t.} \quad x_{r:=(h,u)} \leq W_{hm}\gamma_m,$$
$$A^g x \leq c, \ W\beta \leq s.$$

The above optimization is highly difficult to solve due to the exponentially large size of the feasible graph set G and integer constraints of storage matrix W. Our approach to solve the problem is following. First, we decompose the problem into two parts: graph selection part and resource allocation which consists of content placement and rate allocation. In section IV, we first solve the optimal rate

allocation algorithm under a fixed graph. In the following section V, we propose an algorithm to find the near-optimal overlay graph.

IV. Resource Allocation

The sub-optimization problem under a fixed g is still difficult to solve because of the integer constraints of storage matrix W. In order to overcome this challenge, we allow caches to store videos in parts and to serve *fractional streams*.

A. Fractional Storage and Streams with Coding

We first define fractional streams.

Definition 1: Given a data stream S_0 with rate x_0, $S_0(x_0)$, a fractional stream S of rate x, $S(x)$, is defined as a data stream which satisfies the following conditions.

1. Additivity : $S(x)$ can be generated by $S(x_1)$ and $S(x_2)$ if $x_1 + x_2 \geq x$.
2. Recovery : $S_0(x_0)$ can be generated by any $S(x)$ if $x \geq x_0$.

In our techincal report [17], we propose a novel way of storing videos and generating fractional streams. In a nutshell, a fractional stream can be realized through the use of MDS codes or rateless fountain codes. The minimum storage to serve a fractional stream is found as following.

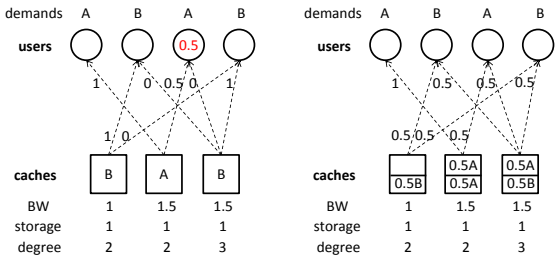
Proposition 2: Given a data stream $S_0(x_0)$, a cache can generate a fractional stream $S(x)$ if and only if the cache stores no less than $\frac{x}{x_0}$ fraction of the source file of the stream.

By using the concept of fractional storage and streams and the proposition 2, the sub-optimization problem can be reformulated. Caches can store movies in parts and upload fractional streams to users. Users receive multiple fractional streams from caches and add them. If they received enough fractional streams, they can recover the original stream. If not, they will request a fractional stream with the deficit rate at user and recover the original stream. The integer constraints are replaced with box constraints and the availability constraints remain as same by the proposition. The reformulated optimization problem is following.

$$\begin{aligned} \max_{x\geq 0, 0\leq W_{hm}\leq 1} \quad & \sum_{u\in U} V^u(z_u) \\ \text{s.t.} \quad & x_{r:=(h,u)} \leq W_{hm}\gamma_m, \\ & A^g x \leq c,\ W\beta \leq s. \end{aligned} \tag{1}$$

Note that our approach completely differs from the well-known class of relaxations to hard mixed-integer problems in the optimization literature, in which the relaxation is done to convert the problem to a computationally tractable one whose solution is then quantized back to enforce feasibility with the discrete-valued constraints of the original problem. In other words, our approach increases performance because using fractional streams expands the design space. Figure 3 shows an example where resource allocation with fractional streams outperforms one without them.

There is an added complication if we are interested in distributed operation and for the ability to have cache nodes



(a) (b)

Fig. 3: A simple example demonstrating the usefulness of fractional storage. The system has the same setup as that in Figure 1. In (a), the videos are stored in full. Even with the optimal topology, the oracle server still needs to fill in the gap of half of a video. In (b), with fractional storage and streams, caches can fully serve users even with less storage.

populated in a distributed way with content (e.g. from a distributed server system or from other cache nodes): this requires the notion of network coding which can attain the abstraction of distributed MDS codes. We use a specific class of such codes, dubbed DRESS codes [18], in our system.

B. Distributed Solution

We present the following theorem.

Theorem 3: The problem (1) is convex and can be solved by the following three-step primal dual algorithm that converges to the optimal solution

1) Step 1: Update the upload rate.

$$\dot{x}_r = [\delta_r(V_{x_r}(z_u) - \lambda_r - q_r)]^+_{x_r} \tag{2}$$

where $V^u_{x_r}(z_u)$ is the derivative of function $V^u(z_u)$ with regard to x_r. $q_r = \sum_{l:l\in r}\theta_l$ is the aggregate *route* price, where θ_l is the single link price on link l which is updated by:

$$\dot{\theta}_l = [\eta_l(\sum_{r:l\in r} x_r - c_l)]^+_{\theta_l}. \tag{3}$$

where $\delta_r, \eta_l > 0$ are adaptation parameters. λ_r is explained in the following.

2) Step 2: Update the demand index. The parameter λ_r is updated via:

$$\dot{\lambda}_r = [\kappa_r(x_r - W_{hm}\gamma_m)]^+_{\lambda_r} \tag{4}$$

where step size κ_r is a positive constant, and m is the video user u watches. This parameter captures the relative demand, *i.e.,* absolute demand minus supply, of the video delivered on route r, hence its name demand index.

3) Step 3: Update cache storage

$$\begin{cases} \dot{W}_{hm} = [\iota_{hm}(\Lambda_{hm} - \beta_m\omega_h)]^{[0,1]}_{W_{hm}} \\ \dot{\omega}_h = [\nu_h(\sum_{m\in M} W_{hm}\beta_m - s_h)]^+_{\omega_h} \end{cases} \tag{5}$$

where the Lagrangian variable ω_h is interpreted as the storage price for the storage constraint, and $\Lambda_{hm} = \gamma_m \cdot (\sum_{r=(h,u):u\in U_m, h\in N^g_u}\lambda_r)$ is the aggregate demand index and β_m the size of video m.

The proof is relegated to our technical report [17]. The above algorithms can be implemented in a fully distributed way. The update equations are intuitive. The upload rate increases linearly with the marginal utility function and decreases with the demand index and link shadow price. The demand index increases if the desired link rate exceeds what the stored video can offer, and vice versa. Cache node h increases storage of video m if there is more demand than supply, *i.e.,* when the popularity index is larger than the storage price; and vice versa.

As a special case, if our goal is to minimize the server load, the following utility function $V^u(z) = \min(\gamma_m, z)$ can be used, where m is s.t. $u \in U_m$. Maximizing such an objective function is equivalent to minimizing the server load. Thus, with the Theorem 3, we now have a fully distirbuted algorithm to find the optimal resource allocation which minimizes the server load under a fixed topology.

V. Topology Building

Given the bounded degree of each cache node, the topology selection problem is NP-hard. We apply a Markov approximation technique to establish that it is indeed a close-to-optimal solution of the general problem.

A. Problem Formulation

Let Φ_g be the optimal objective value to (1) under a fixed graph g, which we can solve using the algorithms shown in Section IV. Our goal is to maximize the overall system utility by choosing the best graph in the feasible set of graphs.

$$\max_{g \in G} \quad \Phi_g \tag{6}$$

Directly solving for (6) is challenging. Even for the small example in Figure 1(a), the total number of topology graphs is $6 \times 6 \times 4 = 144$. In fact, we can show that in general, it is NP-complete and APX-hard (no effective centralized polynomial-time approximate solution) [17]. To convert the problem into a solvable one with distributed solutions, we design a Markov approximation technique introduced in [19].

The topology building problem can be re-written in the following way:

$$\max_{\substack{p_g \geq 0 \\ \sum p_g = 1}} \quad \sum_{g \in G} p_g \Phi_g. \tag{7}$$

where p_g is the probability associated with topology graph $g \in G$. Its optimal solution is $\max_{g \in G} \Phi_g$, and is obtained by setting the probability corresponding to (one of) the "best" topology graphs to be 1 and the rest probabilities to be 0.

To mitigate the problem of exponentially large number of graphs, consider using

$$\frac{1}{\mu} \log(\sum_{g \in G} \exp(\mu \Phi_g)) \tag{8}$$

to approximate $\max_{g \in G} \Phi_g$. The reason is that while solving for $\max_{g \in G}$ is difficult, solving for (8) can be made much easier as we will explain later on. We present the following theorem adapted from [19].

Theorem 4: (8) is the optimal value to the following convex optimization problem:

$$\max_{\substack{p_g \geq 0 \\ \sum p_g = 1}} \quad \sum_{g \in G} p_g \Phi_g - \frac{1}{\mu} \sum_{g \in G} p_g \log p_g. \tag{9}$$

where μ is a positive constant. The optimal solution is given by:

$$p_g^* = \frac{\exp(\mu \Phi_g)}{\sum_{g' \in G} \exp(\mu \Phi_{g'})}, \ \forall g \in G. \tag{10}$$

Moreover, the gap between its optimal value (8) and the optimal value of the original problem in (7) is given by:

$$0 \leq \frac{1}{\mu} \log(\sum_{g \in G} \exp(\mu \Phi_g)) - \max_{g \in G} \Phi_g \leq \frac{1}{\mu} \log |G|, \tag{11}$$

where $|G|$ is the size of G.

Note that $\sum_{g \in G} p_g \log p_g$ is also the entropy $H(p)$ of the distribution $p := (p_g, g \in G)$, and we call this is an "entropy-approximated" formulation. The only difference between the objective function in (9) and that in (7) is the addition of the weighted entropy term. As $\mu \to +\infty$, $p_{g^*}^* \to 1$ where $g^* = \arg\max_{g \in G} \Phi_g$ and $p_g^* \to 0$ otherwise. Therefore, the optimal value in the entropy-approximated formulation approaches that of the original problem as μ becomes large.

Intuitively, one can see that $\frac{1}{\mu} \log(\sum_{g \in G} \exp(\mu \Phi_g))$ is a good approximation of $\max_{g \in G} \Phi_g$ when μ is large. This is because the term $\exp(\mu \Phi_{g^*})$ will dominate the sum of exponentials $\sum_{g \in G} \exp(\mu \Phi_g)$. When this happens, we have:

$$\frac{1}{\mu} \log(\sum_{g \in G} \exp(\mu \Phi_g)) \to \frac{1}{\mu} \log(\exp(\mu \Phi_{g^*})) = \Phi_{g^*} \tag{12}$$

Recall that our goal is to solve for problem (7), and we know that one approximation is given by (9), but why is this approximation useful? Given the vast number of possibilities, how would we design a distributed algorithm to achieve the optimal topology graph? The key idea is to construct a Markov chain (MC) on the topology graphs, and design its transition rates that can be implemented in a distributed way. Equation (10) is of a product form, and can be the stationary distribution of some time-reversible MC with state space the set of all the topology graphs G. When the MC is in its stationary status, the topology graphs $g \in G$ are time-shared according to the distribution p_g^* in (10). As μ becomes large, the system spends most of the time in the optimal topology graph and the gap between the approximated solution and the optimal solution approaches zero. It was shown in [16], [19] that it is possible to design such Markov chain to guide distributed algorithm designs in various domains, including wireless scheduling, channel assignment and etc. However, in our problem, directly applying these known design options in [16], [19] does not result in Markov chains that are implementable in a distributed fashion.

B. Markov Chain Design

To design the Markov chain for our topology selection, we focus on the design of transition rates $q_{g,g'}$ between states g

and g'. With slight abuse of notation, we use g and the set of the corresponding overlay routes R interchangeably. We begin by constructing a Markov chain that solves (9) exactly. This Markov chain does not lead to a fully distributed solution, but serves as a seed to our extended discussions in later parts of the paper.

To simplify the design, we allow non-zero transition rates from topology graph g to graph g' if and only if they satisfy the following set of conditions, which we name as the "direct-transition condition". Specifically for any topology graphs (or set of overlay routes) g and g', $q_{g,g'}$ is non-zero if and only if for the union of their routes $\tilde{g} = g \cup g'$:

- $|\tilde{g} \setminus g| = 1$ and $|\tilde{g} \setminus g'| = 1$;
- the only overlay route in $\tilde{g} \setminus g$ and that in $\tilde{g} \setminus g'$ should originate from the *same* node denoted by $v(g, g')$.

In other words, we allow transitions to happen only when *a single node* adds a not-in-use neighbor and then drops one active neighbor. $\tilde{g} = g \cup g'$ is a *transient* state where a node has *just* added a new neighbor before choking one of the old neighbors [1]. We have the following proposition.

Proposition 5: The following transition rates ensures that the MC over the topology graph will reach the stationary distribution in (10).

$$q_{g,g'} = \tau^{-1} \cdot \frac{\exp(\mu(\Phi_{g'} - \Phi_{\tilde{g}}))}{1 + \sum\limits_{g'' \in \mathcal{A}_{v(g,g'),\tilde{g}}} \exp(\mu(\Phi_{g''} - \Phi_{\tilde{g}}))}, \quad (13)$$

if g and g' satisfy the direct-transition condition, and $q_{g,g'} = 0$ otherwise, where $\tilde{g} = g \cup g'$ is the transient state, $\tau > 0$ is a constant, $\mathcal{A}_{v,\tilde{g}}$ is the set of topology graphs derived by node v dropping one of its active neighbors under graph $\tilde{g}$, *i.e.,*

$$\mathcal{A}_{v,\tilde{g}} = \left\{ \hat{g} \in G \mid \hat{g} = \tilde{g} \setminus \{(v,u)\}, \forall u \in N_v^{\tilde{g}} \right\}, \quad (14)$$

where (v, u) is a directed link from node v to node u, and $v(g, g')$ is the node defined in direct-transition condition.

We call this MC an *exact* MC. The above transition rates achieve the optimal value of (9). The proof and an exemplar implementation are given in our technical report [17]. The algorithm however, requires every node v to know global statistics $\Phi_{g'} - \Phi_{\tilde{g}}$ for all $g' \in \mathcal{A}_{v,\tilde{g}}$, which makes distributed implementation difficult.

In the following, we modify the exact MC to a *perturbed* MC which yields a distributed "soft-worst-neighbor-choking" algorithm.

C. Soft Worst Neighbor Choking

Let g denote the current topology graph. Each cache node $v \in H$ (and similarly for user node $u \in U$) implements the soft-worst-neighbor-choking algorithm as follows:

- **Initialization:** It randomly selects and builds connections with B_v (which is its maximum degree) number of neighbors from its neighbor list Q_v. Denote by N_v^g the connected neighbors.
- **Step 1:** It counts down to zero from a timer $T = \tau/(|Q_v| - |N_v^g|)$, where $\tau > 0$ is a constant[2].
- **Step 2:** When the count-down expires, it randomly chooses a new inactive neighbor w from $Q_v \setminus N_v^g$, and requests to connect to it. If the node degree bound of neither node w nor v is violated, the connection is established; otherwise, no new connection is made. The system transits to a temporary topology graph $\tilde{g}$.
- **Step 3**: Node v measures the overlay link rate $\bar{x}_{(v,u)}$ from every neighbor $u \in N_v^{\tilde{g}}$ (including the newly added neighbor w if there is any), and then chokes an in-use neighbor u with probability

$$\frac{\exp(-\frac{1}{2}\mu \bar{x}_{\tilde{g} \setminus g'})}{1 + \sum\limits_{g'' \in \mathcal{A}_{v,\tilde{g}}} \exp(-\frac{1}{2}\mu \bar{x}_{\tilde{g} \setminus g''})}, \quad g' = \tilde{g} \setminus \{(v,u)\}. \quad (15)$$

Afterwards, node v repeats **Step 1**.

The above algorithm is fully distributed – each node uses the overlay link rates from his neighbors as the only metric to perform the topology selection. Note that the worst link is dropped with high probability, hence the term "soft-worst" choking. This algorithm results in the following perturbed MC.

Proposition 6: The soft-worst-neighbor-choking algorithm induces the following transition rates for the perturbed Markov chain: for any two topology graphs $g, g' \in G$ satisfying direct-transition condition:

$$q_{g,g'} = \tau^{-1} \cdot \frac{\exp(-\frac{1}{2}\mu \bar{x}_{\tilde{g} \setminus g'})}{1 + \sum\limits_{g'' \in \mathcal{A}_{v(g,g'),\tilde{g}}} \exp(-\frac{1}{2}\mu \bar{x}_{\tilde{g} \setminus g''})}, \quad (16)$$

where $\bar{x}_{\tilde{g} \setminus g'}$ is the rate of the only overlay link in $\tilde{g} \setminus g'$ under topology graph $\tilde{g}$; and $q_{g,g'} = 0$ otherwise.

Comparing (16) with (13), we can see that the global quantity $\Phi_{g'} - \Phi_{\tilde{g}}$, which is the overall utility difference between 'after' and 'before' the node drops the overlay link $\tilde{g} \setminus g'$, is replaced by $-\frac{1}{2}\bar{x}_{\tilde{g} \setminus g'}$ which is a locally measurable quantity. Fortunately, we can show that under some reasonable assumptions, the perturbed MC can still achieve close-to-optimal system performance, as we show as follows.

D. Performance Guarantee

Theorem 7: Denote by $\Phi^O = \max_{g \in G} \Phi_g$ the optimal system utility, $\Phi^E = \sum_{g \in G} p_g^* \cdot \Phi_g$ the expected system utility of the exact Markov chain, and $\Phi^P = \sum_{g \in G} p_g^{'} \cdot \Phi_g$ the expected system utility of perturbed Markov chain, where p_g^* and $p_g^{'}$ are the stationary distributions of the exact and perturbed MC respectively. Let $\bar{\Phi}^O = \frac{1}{|U|}\Phi^O$, $\bar{\Phi}^E = \frac{1}{|U|}\Phi^E$ and $\bar{\Phi}^P = \frac{1}{|U|}\Phi^P$ be the corresponding system utilities averaged among the users. When the users' utility function $V(z_u) = \min(\gamma_m, \sum x_r), u \in U_m$, the optimality gaps with

[1]The node can temporarily violate the node degree bound, but the process happens instantaneously and the transient state is almost non-existent. We use it only to simplify our theoretical discussions.

[2]τ is a tuning parameter which affects the count-down time and the algorithm's mixing time. Details will be given in the subsequent section. One example is to set τ such that the count down time is one minute, which BitTorrent uses [20].

and without perturbation errors are shown as follows:

$$0 \leq \bar{\Phi}^O - \bar{\Phi}^E \leq \frac{1}{\mu} B_{\max} \log N_{\max} \quad (17)$$

$$0 \leq \bar{\Phi}^O - \bar{\Phi}^P \leq \frac{1}{\mu} B_{\max} \log N_{\max} + \frac{1}{2|U|} c_{\max} \quad (18)$$

where $B_{\max}$ is the maximum degree bound over all users, $N_{\max}$ is the max neighbor size over all users, $c_{\max}$ is the maximum underlay link capacity, and $|U|$ is the total number of users.

The proof is given in our technical report [17]. We make the following observations:

- The upper bound on the optimality gap per user of the perturbed Markov chain is $\frac{c_{\max}}{2|U|}$ away from that of the exact Markov chain, which we call "the price of local perturbation".
- When we formulated the topology selection algorithm, we made the assumption that the underlying content placement and link rate allocation algorithms have fully converged. When this is not the case however, we obtain inaccurate values of the link rates x_{uv} and therefore $\Phi_g, g \in G$. We can treat this inaccuracy as a one-dimension perturbation error to exact system utilities $\Phi_g, g \in G$. Following the same method to the proof of Theorem 7, we can still obtain bounds on utility gap similar to those in (17)-(18).
- While larger μ reduces the optimality gap, it may also increase the mixing time of the Markov chain and consequently the convergence rate of the worst-neighbor-choking algorithm. We omit the relationship between μ and the mixing time, and refer interested readers to our technical report [17].

VI. Simulation Results

In this section, we evaluate the overall system performance with realistic setting. By allowing users to join and leave the system and to change their subscription, we also empirically show that our system works well with dynamic setting. We show that our algorithm implicitly learns the video demand, and yields high bandwidth utilization.

A. Setup

We assumed there exist a media server which hosts 2000 videos. Each video is 20 minutes long and has a streaming rate of 2 Mbps. Users join the system with varying arrival rate and the maximum number of users is 40000 and leave the system after they finished watching the video. When a user joins, it randomly chooses a video from the catalog following a heavy-tailed distribution. There are 50 caches with uniform storage capacity, with a total storage capacity of 2.5 times the entire video catalog. The bandwidth of cache nodes follow a bi-modal distribution with means 1.2Gbps and 2Gbps. The aggregate bandwidth among all caches is just enough to cover all users. The node degree bound on each cache is 3200, *i.e.,* each user can be potentially served by 4 caches. We compare our results with LRU, LFU, top videos (local), top videos (global) and MIP. The methods and parameters used in the comparing schemes are:

- MIP: on each cache, the fractional storage allocation algorithm [9] is used, and then the entire video with the largest converged fraction of storage is stored, followed by the video with the second largest converged value, and so on and so forth, until the storage capacity is reached.
- LRU(LFU): every cache replaces least recently(frequently) used videos by the most popular videos.
- Top videos (local, global): cache nodes store the most locally(globally) popular videos of the period.

B. Performance

Figure 4(a) shows the percentage non-cache traffic, i.e., the fraction of the total realtime demand that is not covered by the caches, for each method and the total demand versus time. We observe that for the same amount of system resources, our scheme allows the maximum support of the caches, thanks to our theoretical guarantee of optimal performance. Specifically during peak hours where the arrival rate of users is the highest, the corresponding non-cache traffic for our scheme was 24.1% while it is 44.0%, 50.2%, 50.0%, 50.9% and 56.4% for MIP, LRU, LFU, top videos (local) and top videos (global). It is worth noting that even our scheme does not achieve 0% of non-cache traffic although the total aggregate bandwidth is enough to cover the demand in a static setup. This happens because there will always be new users joining the system who need to be fed-up directly by the server. From the results, we can see that storing the globally popular videos, as is done in most practical cases, performs the worst. This is because the video demand distribution is so heavy-tailed that caching of the vast number of unpopular videos becomes unavoidable. MIP performs the closest in performance to our scheme but is still significantly inferior. This corroborates our previous observations in the toy examples that fractional storage provides much flexibility to the caching service.

In order to better understand the performance of our algorithm, we also find the utility efficiency for video m on each cache node h as $\frac{\sum_{u \in U_{m,h}} x_{hm}}{W_{hm} \gamma_m |U_{m,h}|}$ which is the ratio between the total upload rate to users watching video m and the total available rate given the storage of video m. It quantifies the utility of any stored faction of a video on each cache. The efficiency factor for each video is averaged across all the caches which store that video. We observe that with our scheme, most of the efficiency factors are greater than 80% uniformly across the videos. On the other hand, the efficiency factors are around 50% with other schemes.

C. Scalability, Robustness, and Learning

In this subsection, we show the proposed system is scalable, robust and able to learn the demand via simulation results. In the first experiment, we increase the video catalog size and investigate how the system responds. In Figure 4(b), we plot the average server traffic for different catalog sizes. It can be seen that although the catalog sizes vary from 2000 to 10000, the non-cache traffic only increases slightly. The

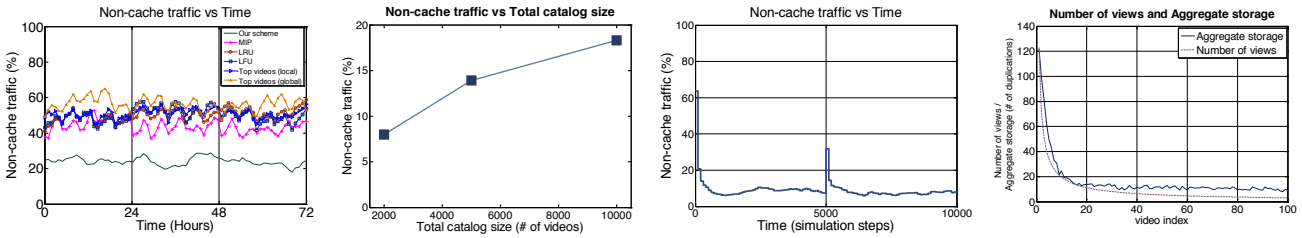


Fig. 4: (a) Performance comparison of our scheme with LRU, LFU, top videos (local), top videos (global) and MIP. (b) Average non-cache traffic varying catalog size. (c) New popular videos are added and1 unpopular movies are removed in the middle of the simulation. (d) Total storage of videos in the cache nodes and the appropriately scaled actual demand distributions.

robustness of the performance to the changes in the catalog size is especially useful when a sudden flash crowd of new videos appear and the system does not have ample time to call for more resources.

In the second experiment, we test the robustness of the system. In Figure 4(c), the server load is shown when new popular movies are suddenly added in the middle of the simulation. More specifically, we inject 10% of new most popular movies and remove 10% of least popular movies. Other 90% old movies become slightly less popular than before. The server load experiences a negligible jump when new movies are added, but quickly goes back to normal. In practice, we expect that the changes in demand to be much slower and that the system can easily adapt to such changes. Due to space limit, we omit other results which show that the system is robust to network collapses or cache collapses.

In Figure 4(d), we break up the total storage of entire caches by the videos and compare it with the demand distribution. We can see that despite running in a fully distributed manner, the system is able to implicitly learn the demand distribution, with the aggregate storage distribution similar to that of the demand. This is a useful property of the algorithm, because separately estimating the demand in practice can induce large overhead. Our scheme is thus demand-agnostic, *i.e.,* it does not require any prior knowledge of the demand, automatically learning the demand distribution in a distributed way.

VII. Conclusion

In this paper, we formulate a general framework for a distributed VoD streaming system considering storage, bandwidth, and node degree bound. We propose a fully-distributed solution and prove that it achieves close-to-optimal performance. We present detailed simulation results and show its performance, scalability and robustness. We show that our proposed scheme maximizes the video traffic supported by cache and minimizes the server load. We are actively building a prototype system and deploying it on Amazon servers to further validate its usefulness and efficiency.

References

[1] C. Labovitz, "The Other 50% of Internet Traffic," in *Proc. of North American Network Operators Group Meeting, NANOG 54*, 2012.

[2] Wikipedia, "Seedbox — Wikipedia, the free encyclopedia," 2012. [Online; accessed 7-October-2012].

[3] J. Almeida, D. Eager, M. Vernon, and S. Wright, "Minimizing delivery cost in scalable streaming content distribution systems," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 356–365, 2004.

[4] Y. Boufkhad, F. Mathieu, F. de Montgolfier, D. Perino, and L. Viennot, "Achievable catalog size in peer-to-peer video-on-demand systems," in *Proc. of IPTPS*, 2008.

[5] X. Zhou and C. Xu, "Efficient algorithms of video replication and placement on a cluster of streaming servers," *Journal of Network and Computer Applications*, vol. 30, no. 2, pp. 515–540, 2007.

[6] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, "On the optimization of storage capacity allocation for content distribution," *Computer Networks*, vol. 47, no. 3, pp. 409–428, 2005.

[7] J. Wu and B. Li, "Keep cache replacement simple in peer-assisted vod systems," in *Proc. of IEEE INFOCOM*, 2009.

[8] B. Tan and L. Massoulié, "Brief announcement: adaptive content placement for peer-to-peer video-on-demand systems," in *Proc. of ACM PODC*, 2010.

[9] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. Ramakrishnan, "Optimal content placement for a large-scale vod system," in *Proceedings of the 6th International COnference*, p. 4, ACM, 2010.

[10] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[11] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the internet with nano data centers," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 37–48, ACM, 2009.

[12] X. Zhou and C. Xu, "Optimal video replication and placement on a cluster of video-on-demand servers," in *Parallel Processing, 2002. Proceedings. International Conference on*, pp. 547–555, IEEE, 2002.

[13] Y. Liu, X. Liu, L. Xiao, L. Ni, and X. Zhang, "Location-aware topology matching in p2p systems," in *proc. of IEEE INFOCOM*, 2004.

[14] N. Laoutaris, D. Carra, and P. Michiardi, "Uplink allocation beyond choke/unchoke," in *Proc. of ACM CoNEXT*, 2008.

[15] V. Aggarwal, O. Akonjang, and A. Feldmann, "Improving user and isp experience through isp-aided p2p locality," in *proc.of IEEE INFOCOM*, 2008.

[16] S. Zhang, Z. Shao, and M. Chen, "Optimal distributed p2p streaming under node degree bounds," in *Proc. of IEEE ICNP*, 2010.

[17] H. Zhang, M. Chen, A. Parekh, and K. Ramchandran, "An Adaptive Multi-channel P2P Video-on-Demand System using Plug-and-Play Helpers," in *UC Berkeley Technical Report*, 2010.

[18] S. Pawar, S. Rouayheb, H. Zhang, K. Lee, and K. Ramchandran, "Codes for a distributed caching based video-on-demand system," in *Asilomar Conference on Signals, Systems, and Computers*, 2011.

[19] M. Chen, S. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," in *Proc. of IEEE INFOCOM*, 2010.

[20] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. of IPTPS*, 2003.