Towards Joint Resource Allocation and Routing to Optimize Video Distribution over Future Internet

Yichao Jin and Yonggang Wen
School of Computer Engineering
Nanyang Technological University
{yjin3, ygwen}@ntu.edu.sg

Cedric Westphal
Huawei Innovation Center &
University of California, Santa Cruz
cedric.westphal@huawei.com, cedric@soe.ucsc.edu

***Abstract*—Given the exploding growth of video traffic, efficient video distribution is essential to the future Internet. Therefore, how to optimize its networking cost is a critical research problem. In this paper, we introduce Network Function Virtualization (NFV) in conjunction with Software-Defined Networking (SDN) to minimize the cost via joint orchestration of caching, transcoding and routing functions. Specifically, we propose a two-step iterative approach. First, in NFV-based resource allocation phase, we maximize total cache hits by optimally allocating storage and computing resources for a giving routing policy. Second, in SDN-based routing phase, we minimize the networking cost by optimally configuring the routing matrix for a given resource placement. Finally, we analytically prove their iterative repeat converges to the joint optimum. Through extensive simulations, we verify its convergence, and performance gains compared with the optimal solution of either phase alone. By examining numerical results, we obtain some operational guidelines. From the resource allocation aspect, we should allocate more resources to the node with heavier request rate. From the routing aspect, for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path alone if there is only one.**

I. Introduction

The exploding growth of video traffic over the Internet [1], [2], poses significant challenges to the existing network architecture. In particular, first, the large-scale distribution of video content requires tremendous bandwidth resources, which are difficult to sustain efficiently based on the current network infrastructure [3]. Furthermore, video streaming, nowadays, is usually consumed by a set of heterogeneous end-devices, with different resolutions, formats, and bitrates. This requires careful scheduling of the storage and computing resources to transcode video contents into multiple versions and cache them at some intermediate nodes within the network [4]. However, such fine-grained in-network resource management scheme is not well supported by the existing Internet framework.

Network Function Virtualization (NFV) [5], in conjunction with Software-Defined Networking (SDN), introduces a new dimension to efficiently orchestrate networking and computing services, including video distribution. Specifically, NFV implements network functions (e.g., caching and transcoding) in software, that can be run on virtual machines and migrated among various locations in the network. This enables the elastic management of networking and computing resources. Combined with SDN, NFV provides an opportunity to address those challenges of efficient video distribution. Nevertheless, this solution space has not been well investigated thus far.

In this work, we aim to improve the cost efficiency of video distribution over media cloud [6], [7], by jointly considering the NFV-enabled storage and computing resource allocation, and the SDN-enabled routing. In particular, we consider the orchestration of routing, caching and transcoding functions. At each node, the caching function brings the contents to a closer place to users, saving the bandwidth cost. Its benefit for distributing adaptive video can be further increased by keeping only the highest bitrate, and transcoding other versions on the fly upon request. And at each link, the routing function schedules the traffic load, balancing the traffic network-wide. Therefore, the media cloud operator needs to decide which items to cache or transcode, and how to coordinate these functions with appropriate routing policy in the network.

Our approach iteratively proceeds in two phases. First, in the resource allocation phase, we study how to optimally allocate a fixed amount of virtualized in-network storage and computing resources over the whole network, with an objective to maximize overall cache hits. However, this phase may fail to take congestion into consideration, resulting in tremendous networking cost. As a result, we need an additional network routing phase, to find the optimal routing matrix that balances the traffic flow over each link, for a given resource placement strategy. Its objective is to minimize the total networking cost, based on the traffic engineering model. Finally, we show that, the iterative repeat of these two phases converges to the joint optimal resource allocation and routing.

The contributions of this paper are multi-fold, including:

- We systematically build a set of models to describe the adaptive video distribution system over NFV and SDN enabled media cloud. Based on them, we formulate an optimization problem, to jointly optimize the virtualized resource allocation and the routing control.
- We propose a two-step iterative approach, to find the joint optimal configuration on the virtualized resource allocation and its corresponding routing policy. In addition, we also present the analytical proof on the convergence and optimality of our proposed approach.
- Through extensive simulations, we show that our approach quickly converges to its optimum. It achieves sig-

ISBN 978-3-901882-68-5

nificant cost savings, compared with the optimal solution over either resource allocation or routing alone.
- We obtain operational guidelines by examining the joint optimal solution. From the resource allocation aspect, more resource should be allocated to the node with heavier request rate. From the routing aspect, for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path alone if there is only one.

Although our approach is specific to video distribution, the two-step model would be also applicable to the orchestration of other virtualized network functions with bandwidth, storage and computing costs. Those obtained insights potentially ease the adoption of NFV and SDN for the future Internet.

The rest of this paper is organized as follows. Section II reviews those related works. Section III presents the system overview, system models, and the problem formulation. Section IV shows theoretical analysis for the problem, and proposes algorithms to derive the optimal solution. Section V numerically evaluates the performance based on extensive simulations. Finally, Section VI summarizes this work.

II. Related Works

We had witnessed the leverage of SDN to optimize the network routing in a diversity of networking applications over the past few years. In particular, McKeown *et al.* [8] initially proposed OpenFlow and SDN to decouple network control and data forwarding for research innovation in campus networks. Lately, Egilmez *et al.* [9] introduced OpenFlow controller to dynamically re-route scalable video streaming with the objective to improve the PSNR (Peak Signal-to-Noise Ratio). Arefin *et al.* [10] achieved a SDN-based cross-layer multi-stream multi-site protocol for 3D teleimmersion to improve the application interactivity and resource utilization. Li *et al.* [11] explored SDN to schedule the exclusive data flow in the data center network in an energy-aware manner.

At the same time, a number of works focused on improving the performance or efficiency of video distribution, by carefully scheduling network functions (e.g., caching and transcoding). Although none of them were specific to the NFV framework, their solutions were closely related to the NFV-based resource allocation. Specifically, Dai *et al.* [12] developed a hierarchical caching framework coupling with dynamic request routing to improve the caching and bandwidth utilization. Li *et al.* [13] investigated the optimal in-network caching space provision for each individual router, to optimize the overall networking cost. Jin *et al.* [14] designed a partial in-network transcoding scheme to reduce the overall operational cost of delivering adaptive video streaming over information centric network. Wang *et al.* [15] jointly studied video transcoding and delivery for adaptive video streaming, aiming at reducing both computing and storage costs.

This work clearly differs from above researches in following aspects. First, we investigate the cost-efficient video distribution in SDN and NFV enabled media cloud, where the application scenario is obviously different from the one of those previous works. Second, we focus on the joint optimization problem of NFV-based virtualized resource allocation and SDN-based routing, whereas most existing works addressed only one of them. Finally, we propose a two-step iterative approach to find the optimal solution, and develop an analytical framework to prove its convergence and optimality. To the best of our knowledge, this work is the first attempt to systematically introduce SDN and NFV to orchestrate adaptive video streaming services in cloud computing environment.

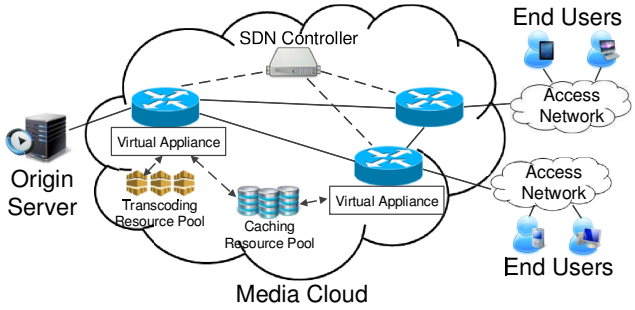


Fig. 1: Architecture of NFV and SDN enabled media cloud

III. System Overview & Problem Formulation

In this section, we first give an overview of the adaptive video distribution system over media cloud, to provide necessary background. Then, we introduce our system models and formulate an optimization problem.

A. System Architecture

Figure 1 presents a systematic end-to-end view of delivering adaptive video streaming over a NFV and SDN enabled media cloud. The system consists of three parts,

- On the server side, the origin server publishes all video segments with all candidate formats, which can be delivered to end-users in an on-demand manner.
- On the user side, they connect via access networks to the media cloud from different regions. Due to the diversity of end-devices, the requested video streaming typically uses various bitrates and formats.
- In the middle, some switches act as backbone service entry points, serving the aggregated user requests from the access network. The cloud service provider can rent virtualized caching and transcoding resources from a resource pool, and dynamically attach them into any set of those switch nodes as virtual appliances based on the NFV concept. In addition, with the SDN technology, the cloud service provider has a full view of the underlying traffic load, and complete control over the routing matrix in a centralized manner.

One critical design objective of such system is to minimize the networking cost incurred by delivering adaptive video streaming. Indeed, such cost highly depends on the joint orchestration of caching and transcoding functions. Specifically, on the one hand, in-network caching keeps popular contents

TABLE I: Notation table

Symbol	Definition
$G(V, E)$	Network topology with node set V and edge set E.
d_e	Bandwidth capacity at edge e.
c_{tot}	Total amount of caching resource in terms of the size.
o_{tot}	Total amount of transcoding resource in terms of the rate.
R_{lv}	Fraction of traffic on node v's l^{th} path.
I^e_{lv}	Indicates if node v's l^{th} path using link e.
m	Total number of segments published by origin server.
r	Total number of frequently accessed bitrate versions.
s_{ij}	j^{th} bitrate version of chunk i.
b_j	Segment size of the j^{th} bitrate version.
b_m	Mean size of all bitrate versions.
λ_v	Aggregated user request arrival rate at node v.
$P(s_{ij})$	Prob. of requesting s_{ij} .
p_j	Prob. of requesting the j^{th} bitrate version.
α	Zipf exponent.
x_v	# different segments can be served by node v.
y_v	# different segments are served by exact cache at node v.
c_v	Required cache space at node v.
o_v	Required transcoding rate at node v.
$f(u_e)$	Convex cost function with respect to link utilization u_e.
p^v_{hit}	Total cache hit ratio at node v.
p^v_{tr}	Transcoding hit ratio at node v.
p^v_{miss}	Cache miss ratio at node v.

at the edge of network, saving the networking cost. And the transcoding function transcodes the cached contents into different formats with different bitrates, further improving the cache hit ratio. On the other hand, given limited in-network storage and computing resources, the cost saving could be quite small, if they are not appropriately scheduled. As a result, there is an opportunity to optimize the operational cost of delivering adaptive video streaming, by carefully allocating these two resources, and operating the network routing accordingly.

B. System Models

This subsection illustrates our system models to mathematically capture key features of the system. For clarity in the discussion, we summarize important notations in table I. Note, we only present the meaning of some small letters in the table, and we will use their boldface form to denote vectors or matrix (e.g., $\mathbf{x}$ with x_i as its i-th component). We also refer letter t as the iteration number (e.g., $\mathbf{R}(t)$) in our iterative approach.

1) Network Model: We model the media cloud network as an undirected graph $G = (V, E)$, where V denotes the set of switches, and E is the set of network links among those switches. Note, in this work, we consider there is only one origin server, which can be accessed via a switch. But this network model can be easily extended to multiple origin servers by connecting them each with a direct link with infinite capacity to a "single super-server". Moreover, we assume this core network is operated by a single network administrator.

This topology $G(V, E)$ is built upon a set of resources and constraints. Specifically, we assume the cloud service provider rents a total amount of caching (i.e., c_{tot} in terms of the cache space) and transcoding (i.e., o_{tot} in terms of virtualized CPU cycles, which are in turn in proportion to the bitrate of transcoded contents) resources over the whole network. Those resources can be dynamically allocated to any set of nodes, running as virtual appliances. Besides, each link $e \in E$ is with limited bandwidth capacity d_e. Moreover, the cloud service provider can choose one or more node-server paths to transmit data between an intermediate node and the origin server, and decide the traffic load at each path subject to the link capacity, by installing the routing matrix $\mathbf{R}$ at the SDN controller. For each node-server path, we use $I^e_{lv} \in \{0, 1\}$ to indicate if node v's l-th path traverses link e. This information can be calculated offline for a known network topology.

2) Content Model: To provide adaptive video streaming services, we assume that all content objects are chunked into m different segments with fixed length, and each segment has r different bitrate versions in different formats, that are frequently accessed. We denote $\mathbf{s} = \{s_{11}, ..., s_{ij}, ..., s_{mr}\}$ as the set of all segments with all bitrate versions, where an element s_{ij} refers to the j-th bitrate version of chunk i, that $i = 1, ..., m$ and $j = 1, ..., r$. We also denote $\mathbf{b} = \{b_1, ..., b_r\}$ as the set of ordered segment bitrate, where b_1 and b_r refer to the lowest and the highest bitrate, respectively. As we focus on the adaptive streaming service, r is always larger than 1.

We assume the aggregated user requests at each entry switch towards those m different segments follow Zipf distribution [16], [17]. In addition, we adopt the independent reference model [18], [19], that the request arrival pattern at each entry point is independent, following Poisson process with the arrival rate as λ_v at node v. In this way, we have the probability of requesting s_{ij} at each node as,

$$P(s_{ij}) = \frac{1/i^\alpha}{p_j \sum_{k=1}^{m}(1/k^\alpha)} = \frac{1/i^\alpha}{p_j H_{m,\alpha}}, i = 1, ..., m, \quad (1)$$

where p_j is the probability of requesting the j-th bitrate of a segment, and $H_{m,\alpha} = \sum_{k=1}^{m}(1/k^\alpha)$ is the m-th generalized harmonic number, and α is the exponent parameter of the Zipf distribution, that $\alpha > 0$. Typically, α is between 0.5 and 1.5 in practice [13]. A larger α indicates more requests for the popular content and fewer request of the unpopular content.

3) Request Processing Model: We adopt the partial in-network transcoding model from [14], that each node caches all bitrate versions for a few top popular segments, and only the highest version for some segments that are less popular, constrained by the allocated caching space. As a result, there are three possible conditions to serve requests at each node.

- An *exact cache hit* occurs, when the node keeps the exact copy of the requested content in its local cache space. In this case, the node can directly serve the user.
- A *transcoding hit* occurs, when the node has the highest bitrate version that can be used to transcode into the requested version in real-time. In this case, the node is still able to serve the user locally, involving the usage of the computing resource.
- A *local cache miss* occurs, when the node holds neither the exact content nor a transcodable version. In this case, we assume all cache misses will be directly forwarded to the origin server.

Note that, because of the tremendous amount of accessible contents over Internet, those caching resources can never keep

all of them within the network [3], [13]. As a result, it is necessary to optimize the occurrence of these above conditions by intelligently allocating virtualized resources.

We model these conditions by focusing on the number of segments that can be cached or transcoded at each node. Specifically, we denote x_v as the number of segments, that can be directly served by node v. We further assume node v caches all the bitrate versions of the top y_v popular segments (where $y_v < x_v$), and only the highest bitrate version of the $x_v - y_v$ segments, with its popularity rank between y_v+1 and x_v. Thus, we have he required caching space at node v as,

$$c_v = y_v \sum_{i=1}^{r} b_i + (x_v - y_v) b_r, \tag{2}$$

and the required transcoding rate at node v is,

$$o_v = \lambda_v p_{tr}^v \sum_{j=1}^{r-1} p_j b_j, \tag{3}$$

where $\sum_{j=1}^{r-1} p_j b_j$ denotes the mean size of those transcoded versions of one segment, (note, we assume the transcoding resource consumption is in proportion to the output bitrate [20]), and p_{tr}^v refers to the transcoding ratio at node v as,

$$p_{tr}^v = \sum_{i=y_v+1}^{x_v} \sum_{j=1}^{r-1} P(s_{ij}) = \frac{(1-p_r)(H_{x_v,\alpha} - H_{y_v,\alpha})}{H_{m,\alpha}}. \tag{4}$$

Similarly, we have the overall cache hit ratio, including both exact cache hit and transcoding hit, at node v as,

$$p_{hit}^v = \sum_{i=1}^{x_v} \sum_{j=1}^{r} P(s_{ij}) = \frac{H_{x_v,\alpha}}{H_{m,\alpha}}, \tag{5}$$

and the local cache miss ratio at node v as,

$$p_{miss}^v = 1 - p_{hit}^v. \tag{6}$$

C. Problem Formulation

We formulate an optimal scheduling problem, by jointly considering the network routing policy, and the allocation of in-network caching and transcoding functions.

1) Network Routing: We consider the network routing phase that closely reflects the practical operations in IP backbone networks [21]–[23]. In particular, we define a routing matrix R_{ev} to capture the fraction of flow going to node v that traverses each link e. This routing matrix can be configured in accordance with the known underlying network topology, the placement of in-network caching and transcoding resources, and observed traffic loads. The objective of such configuration is to minimize the networking cost.

Mathematically, we capture this traffic-engineering practice, by building the optimization problem over routing matrix $\mathbf{R}$, for the given caching and transcoding allocation,

$$\min_{\mathbf{R}} \quad \sum_{e \in E} f\Big(\sum_{v \in V} \sum_{l} R_{lv} \lambda_v b_m p_{miss}^v I_{lv}^e / d_e\Big), \tag{7}$$

$$s.t. \quad \sum_{l} R_{lv} = 1, \; R_{lv} \in [0,1], \tag{8}$$

where $b_m = \sum_{j=1}^{r} p_j b_j$, denotes the mean segment size of all candidate bitrate versions, and $f(u_e)$ is the cost function that is strictly increasing and convex, with respect to the link utilization $u_e = \sum_{v \in V} R_{ev} \lambda_v b_m p_{miss}^v / d_e$, [24], [25]. In practice, f can be an exponential function or a piecewise-linear function that has the similar shape, to penalize the link congestion. Besides, the product of multiplying $\lambda_v b_m p_{miss}^v$ can be understood as the incoming transmission rate to serve cache misses of node v, and $I_{lv}^e \in \{0,1\}$ indicates if node v's l-th path uses link e. Finally, the constraint (8) indicates the summation of traffic load over all possible paths must be one. By minimizing the total networking cost, this optimization problem implicitly achieves the load balance, by shifting flows from heavily utilized links to those less utilized ones.

2) Virtualized Resource Allocation: The resource allocation phase aims to maximize the overall cache hits for a given routing matrix, so that more requests can be served at the edges, reducing the total traffic load within the media cloud. Specifically, we have the following cache hit maximization problem over $\mathbf{x}$ and $\mathbf{y}$ as,

$$\max_{\mathbf{x},\mathbf{y}} \quad g_0 = \sum_{v \in V} \lambda_v p_{hit}^v, \tag{9}$$

$$s.t. \quad g_1 = \sum_{v \in V} c_v \le c_{tot}, \tag{10}$$

$$g_2 = \sum_{v \in V} o_v \le o_{tot}, \tag{11}$$

$$\forall e \in E \quad g_i = \sum_{v \in V} \sum_{l} R_{lv} \lambda_v b_m p_{miss}^v I_{lv}^e \le d_e, \tag{12}$$

where Eq. (10) (i.e., g_1) captures the total caching space constraint, Eq. (11) (i.e., g_2) captures the total transcoding resource constraint, and Eq. (12) (i.e., g_i, $i = 3, ..., |E|+2$) captures the bandwidth constraint at each link. By solving this optimization problem, we aim to update the allocation of in-network caching and transcoding functions with a lower cost.

3) Joint Problem: By jointly investigating the network routing and resource allocation problem, we propose a two-step approach in a feed-back loop.

First, at time $t+1$, the optimal network routing step computes new routing matrix based on the previous resource placement scheme $\mathbf{x}$ and $\mathbf{y}$ at time t as,

$$\mathbf{R}(t+1) = \arg\min_{R} \sum_{e \in E} f\Big(\sum_{v \in V} \sum_{l} R_{lv} \lambda_v b_m p_{miss}^v I_{lv}^e / d_e\Big), \tag{13}$$

under the constraint (8).

Then the resource allocation step updates the allocation of in-network caching and transcoding functions in terms of $\mathbf{x}$ and $\mathbf{y}$, based on the obtained routing matrix $\mathbf{R}(t+1)$ at the previous iteration, by following,

$$\mathbf{x}(t+1), \mathbf{y}(t+1) = \arg\max_{\mathbf{x},\mathbf{y}} \sum_{v \in V} \lambda_v p_{hit}^v, \tag{14}$$

$$s.t. \quad Constraint \text{ (10), (11), } and \text{ (12).} \tag{15}$$

The iterations of Eq. (13) and Eq. (14) repeat over time, with resource allocations adapting to the new routes, and the routing

matrix adapting to the updated traffic rates. This process ends when the decision variables $\mathbf{x}$ and $\mathbf{y}$ have converged. The convergence and the optimality of our solution will be verified in the next section. Note, this work only focuses on an one-shot optimization problem for a specific request pattern λ_v and $P(s_{ij})^v$. As the user request patterns change over time, this approach still works by periodically repeating the iteration.

IV. Analysis & Algorithms

In this section, we first analytically prove our iterative approach converges to a provably stable and optimal solution. Then we present two algorithms to solve this joint problem.

A. Convergence and Optimality

Our proof consists of three steps. First, we show that either the network routing configuration or the resource allocation phase actually solves a convex optimization problem, respectively. Second, we construct an alternative optimization form, with its Gauss-Seidel method equivalent to the original joint problem. Finally, we provide a sufficient condition to guarantee the convergence to its optimality.

1) Convexity of resource allocation and network routing: First, we check Eq. (9) to Eq. (12) with respect to $\mathbf{x}$ and $\mathbf{y}$, focusing on the existence of their first order derivative and the positivity/negativity of their second derivative. By doing so, we have the following lemma.

Lemma 1: Each one-shot optimal virtualized resource allocation problem to Eq (12)) is jointly convex in $\mathbf{y}$ and $\mathbf{z}$, where $z_v = x_v^{1-\alpha}$ is an auxiliary variable.

Proof: See Appendix A for a completed proof. ■

Besides, by checking Eq. (7) and constraint (8), we find that, the optimal network routing problem is also a convex optimization problem over $\mathbf{R}$, because the networking cost function f is strictly convex in $\mathbf{R}$ by definition, and the equality constraint is affine in terms of $\mathbf{R}$.

2) Equivalent optimization form: Second, we modify our joint problem, by combining two objectives (i.e., minimizing overall networking cost, and maximizing overall local cache hit ratio) into one function as,

$$h(\mathbf{R},\mathbf{y},\mathbf{z}) = \sum_{e\in E} f(\sum_{v\in V}\sum_{l}\frac{R_{lv}\lambda_v b_m p^v_{miss} I^e_{lv}}{d_e}) - \sum_{v\in V}\lambda_v p^v_{hit}.$$

This modified problem is equivalent to a successive optimization over $\mathbf{R}$, then $\mathbf{y}$ and $\mathbf{z}$. Its objective is to make the network routing update to correspond exactly with the optimal resource allocation scheme over $\mathbf{y}$ and $\mathbf{z}$, in Gauss-Seidel algorithm [26]. In this way, we need to prove the solution of this modified problem converges to the optimum of,

$$\arg\min_{\mathbf{R},\mathbf{y},\mathbf{z}} \quad h(\mathbf{R},\mathbf{y},\mathbf{z}), \tag{16}$$

$$s.t. \quad Eq.\ (8),\ (10),\ (11),\ and\ (12). \tag{17}$$

3) Convergence of the joint problem: Based on Lemma 1 and the modified optimization problem, we complete the convergence proof by obtaining the following theorem.

Theorem 1: The joint optimal resource allocation and network routing problem converges to its optimum.

Proof: See Appendix B for a completed proof. ■

B. Algorithms to the Joint Optimal Solution

We propose two alternative approaches to separately reach the optimal solution of the network routing and resource allocation policy at each iteration. Specifically, we adopt KKT (Karush-Kuhn-Tucker) conditions to the optimal network routing, and subgradient method to the optimal resource allocation. These two processes iteratively repeat until the obtained solutions converge. In this way, the results (i.e., $\mathbf{R}$, $\mathbf{x}$, and $\mathbf{y}$) must converge to the optimum of the joint problem, as we have already proved its convergence and optimality.

1) Algorithm to optimal network routing: We use KKT condition to solve this one-shot optimal network routing configuration problem for a given resource allocation. Specifically, the Lagrange function is given by,

$$L(\mathbf{R},\gamma) = \sum_{e\in E} f(\sum_{v\in V}\sum_{l} R_{lv}C) + \gamma(\sum_{l} R_{lv} - 1), \tag{18}$$

where $C = \lambda_v b_m p^v_{miss} I^e_{lv}/d_e$ denotes the constant part in function f, and γ is the KKT multiplier.

Thus, setting the gradient $\nabla L(\mathbf{R},\gamma) = 0$ yields the following equations for the optimal solution,

$$\frac{\partial L(\mathbf{R},\gamma)}{\partial R_{lv}} = \sum_{e\in E} C\frac{\partial f(R_{lv}C)}{\partial R_{lv}} + \gamma = 0, \tag{19}$$

$$\frac{\partial L(\mathbf{R},\gamma)}{\partial \gamma} = \sum_{l} R_{lv} - 1 = 0, \tag{20}$$

$$R_{lv}\mu_{lv} = 0,\ \forall l \in L,\ \forall v \in V, \tag{21}$$

where Eq. (21) is the complementary slackness condition with μ_{lv} as the KKT multipliers, capturing the positive constraint of R_{lv}. In this way, we have $2|L||V|+1$ equations with $2|L||V|+1$ unknown variables. As a result, we use fsolver in Matlab to solve these equations, and obtain the optimal solution.

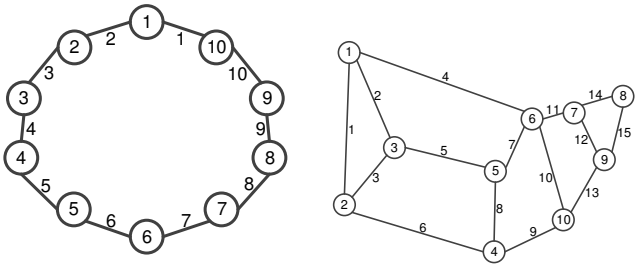
2) Algorithm to optimal resource allocation: We adopt subgradient method as presented by Algorithm 1, to find the optimal virtualized resource allocation at each node for a given routing matrix. In particular, the algorithm begins with an initial feasible solution $\mathbf{y}^{(0)}$ and $\mathbf{z}^{(0)}$. At each iteration, it takes a step β_k along with its subgradient $g^{(k)}$ of the objective or one of the constraint functions. The optimal criteria of this process is that, if the current point is feasible, it uses an objective subgradient; otherwise, the algorithm chooses a subgradient of any violated constraint. We repeat the iteration, until it gets converged. The proof on the convergence and optimality of applying subgradient method to solve constrained convex problem, can be found in [27].

The complexity of algorithm 1 is $O((|V|+|E|)/\epsilon^2)$. Specifically, at each iteration, we need to check $|E|+2$ constraints, and correspondingly update $\mathbf{y}$ and $\mathbf{z}$, which contain $|V|$ elements respectively. And the subgradient method takes $O(1/\epsilon^2)$ iterations to converge [27].

Algorithm 1 Subgradient method

Input: Routing matrix $\mathbf{R}(t+1)$
Initial resource allocation $\mathbf{y}^{(0)}$ and $\mathbf{z}^{(0)}$
Output: Optimal resource allocation $\mathbf{y}(t+1)$ and $\mathbf{z}(t+1)$

1: initiate $k = 0$
2: **do**
3: **if** $\forall i = 1, ..., |E| + 2$, $g_i(\mathbf{z}) < 0$, **then** $g^{(k)} = \nabla g_0(\mathbf{z})$
4: **else** $g^{(k)} = \nabla g_j(\mathbf{y}, \mathbf{z})$, for the j that $g_j(\mathbf{y}, \mathbf{z}) > 0$
5: update $\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} - \beta_k g^{(k)}$
6: update $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - \beta_k g^{(k)}$
7: update $k = k + 1$
8: **while** $|z_v^{(k)} - z_v{}^{(k-1)}| > \epsilon$, $\exists v \in V$ **or** $g_i(\mathbf{y}, \mathbf{z}) > 0$, $\exists i = 1, ..., |E| + 2$
9: return $\mathbf{y}^{(k)}$ and $\mathbf{x}^{(k)} = \mathbf{z}^{(k)}{}^{1/(1-\alpha)}$ as the optimal solution



(a) 10-node ring (b) Internet2 Layer3

Fig. 2: Network topologies used in the simulation

V. Performance Evaluation

In this section, we evaluate the performance of our proposal via extensive simulations, based on the settings that reflect real application scenarios. The objective is not only to verify the convergence and optimality of our proposal, but also to understand the impact of various system parameters, ultimately obtaining operational guidelines for real deployment.

A. Parameter Settings

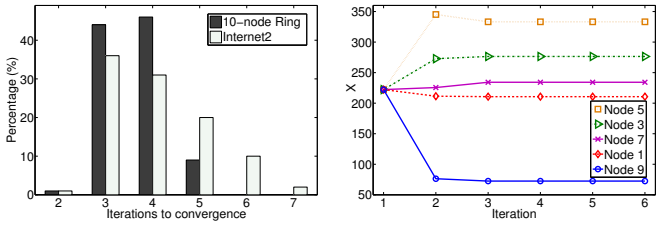
For the network topology model, we use two examples as shown in Figure 2, including a simple and symmetric 10-node ring topology, and the real Internet2's layer 3 network in the U.S [28] which was built specifically for research and testing purpose for the future Internet. To facilitate our discussions, all nodes and links are indexed by the number on them. In the ring topology, each source-destination pair has exactly two paths (i.e., clockwise and counter-clockwise). And in Internet2, we choose four paths that have minimum hops among all possible paths for every source-destination pair. As such, we readily have I_{lv}^e for both topologies. In addition, we assume the link capacity follows a truncated Gaussian distribution bounded below by zero [25], where $d_e \sim \mathcal{N}(1, 0.15)$ Gbps. Besides, we replace the networking cost function by $f(u_e) = nu_e^n$, as suggested by some traffic engineering studies [25].

For the content model, we assume there are in total $m = 20,000$ different segments, where each segment has $r = 5$ different bitrate versions for different devices, and the length of each segment is normalized into one second. We use real bitrate data from a real adaptive streaming system [29], as shown in Table II. The popularity of different segments is artificially generated based on Zipf distribution, and the popularity of different bitrate follows a truncated Gaussian distribution as $p_j \sim \mathcal{N}(0.2, 0.02)$. At each node, we assume the Zipf distribution (i.e., α) is roughly the same, and the request rate also follows a truncated Gaussian distribution [25] as $\lambda_v \sim \mathcal{N}(200, 20)$ requests per second. Finally, we assume the total caching resource is 10% of all segments, and the total transcoding resource can transcode 1% segments in real-time.

We will adopt those settings, and use $f(u_e) = 3u_e^3$, $\alpha = 0.9$ in following experiments, unless otherwise stated.

TABLE II: Bitrate versions in real adaptive streaming system

Type	240p	360p	480p	720p	1080p
Bitrate	0.2Mbps	0.5Mbps	1.2Mbps	2.0Mbps	3.0Mbps



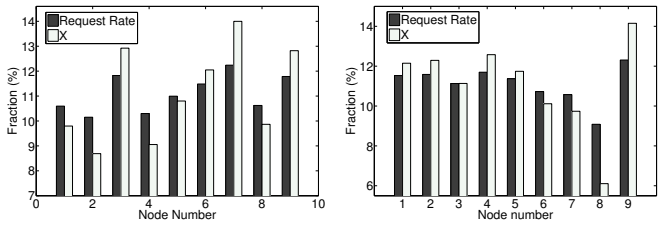
(a) Convergence rate (b) An example of resource allocation at each node over Internet2

Fig. 3: Convergence performance of our iterative approach

B. Convergence Verification

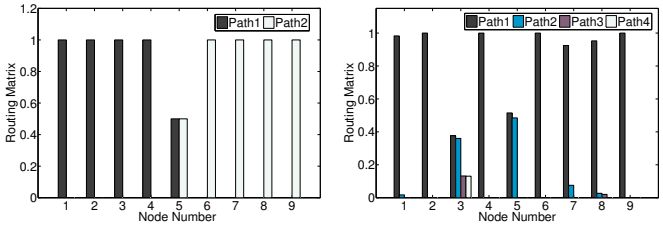
In this subsection, we evaluate the convergence of our iterative approach, by simulating 100 random configurations and obtaining the mean convergence rate. In particular, In each simulation round, we randomly pick a node as the origin server for both topologies, and start with a uniform caching and transcoding resource allocation over all nodes. The convergence of each simulation round is determined by whether the difference between the newly updated value and the previous one is smaller than $\epsilon = 0.001$.

Figure 3 verifies the convergence of our iterative approach, where we count the completion of both allocation and routing phase as one iteration. According to Figure 3(a), our proposal successfully converges to the joint optimum within 7 iterations, for all simulation rounds. For both topologies, more than 65% cases get converged within 4 iterations. As an example, Figure 3(b) shows how optimal resource allocation at each node changes over iterations in Internet2, where the origin server is set at node 10. It is clear that, after only 3 iterations, our solution quickly converges to its optimum. Besides, we notice that, the ring topology needs less iterations than Internet2 on average. This is because when applying Gauss-Seidel type iterative approach, the solution at each iteration follows conjugate directions of its decision variables. Since the ring topology has less possible paths for each node in the routing



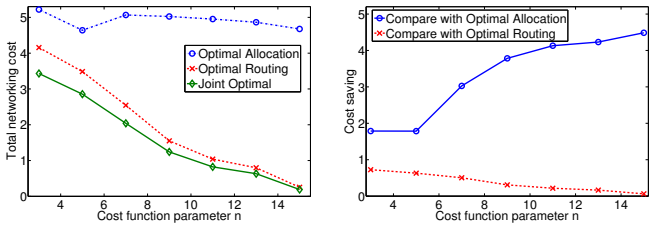
(a) Optimal allocation over ring (b) Optimal allocation over Internet2

Fig. 4: Optimal virtualized resource allocation



(a) Optimal routing over ring (b) Optimal routing over Internet2

Fig. 5: Optimal network routing policy



(a) Total cost in Internet2 (b) Cost saving in Internet2

Fig. 6: Total networking cost vs. cost function $f(u_e) = nu_e^n$

phase, its system matrix is smaller than the one of Internet2. This leads to the faster convergence for ring topology.

C. Joint Optimal Solution

This subsection demonstrates the obtained optimal resource allocation and routing strategy, through a few examples. Here, we set node 10 as the origin server in both topologies.

1) Optimal resource allocation scheme: Figure 4 illustrates the optimal virtualized resource allocation at each node for both topologies, where the y-axis denotes the fraction of the allocated resource or request rate at each node over the total amount from all nodes (i.e., $\lambda_v / \sum_{v \in V} \lambda_v$, and $x_v / \sum_{v \in V} x_v$). We report two observations and their corresponding insights from those data, as follows.

First, we find that, more resources should be allocated to the node with heavier user request rate. For example, in Figure 4(a), node 2 has the lowest request rate, and the lowest resource amount. In contrast, node 7 is with the highest request rate, and the highest resource amount. We explain this phenomenon based on the load balance nature of our optimization problem. In particular, on the one hand, given the identical popularity distribution at each node, more requests leads to more local cache misses. On the other hand, by placing more resources, we can reduce those cache misses. As a result, to balance the cache miss traffic at each node, we need to coordinate the resource allocation with the request arrival rate.

Second, the total cache hits can be unchanged, if we increase the total amount of caching resource but reduce the computing resource, and vice versa. Indeed, the in-network transcoding function was initially proposed to increase the local cache hit ratio, in addition to the limited caching space at each node [30]. In practice, this implies, the operator of media cloud can elastically choose different combinations of total caching and computing resource, to maintain the same cost efficiency level.

2) Optimal network routing policy: Figure 5 demonstrates the optimal routing matrix of one source destination pair (i.e., the node-server pair) in both topologies. This experiment focuses on the impact of underlying topology, and uses the deterministic uniform model by setting the standard deviation of the request arrival rate at each node, and the link capacity at each edge as zero.

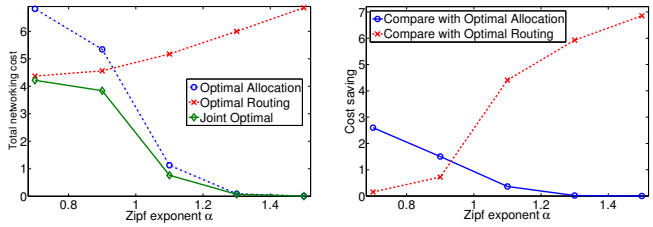
The results reveal that, for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path if there is only one. For instance, in Figure 5(a), all nodes except node 5 have only one shortest path, and they only use it to forward cache misses. Whereas node 5 splits its traffic evenly across two paths, both of which are 5 hops away from the origin server. This also applies to Figure 5(b), where node 4, 6, and 9 only use their 1-hop path, whereas node 3 splits the traffic over 4 different paths, which are all 3 hops away from the origin server. The reason for this phenomenon is intuitive that, using the shortest path to transmit data minimizes the traversed hops, and splitting the traffic across multiple shortest paths balances the traffic load at each link.

D. Performance Comparison

This subsection compares our joint optimal solution with two baselines, where only the optimal resource allocation or the optimal routing is considered. Specifically, the first baseline optimizes the virtualized resource allocation, where every node always retrieves cache misses by using only one path with the shortest hop distance to the origin server. While the second baseline optimizes the network routing, where caching and transcoding resources are always uniformly distributed over all nodes. We evaluate their performance with various system parameters, to obtain some operational guidelines. Similarly, we run 10 random configurations for each setting, and only report the average value. Note that, since solutions from either topology almost have the same shape, we only present the results from Internet2, in following discussions.

1) Total cost vs. networking cost function: Figure 6 shows the relationship between the total networking cost and the cost function. Specifically, we change the exponent parameter of cost function to simulate different levels of congestion penalty. We present both absolute cost of all three strategies, and the cost saving of our proposal compared with baselines.

The results reveal several insights. First, we find our joint optimization solution performs better than those baselines in



(a) Total cost in Internet2 (b) Cost saving in Internet2

Fig. 7: Total networking cost vs. Zipf exponent α

all simulation cases, including both Figure 6 and Figure 7. This verifies the optimality of our method. Second, the total cost in the ring topology is higher than the one in Internet2. This is because that, when balancing the same total amount of traffic over more edges, the congestion penalty can be successfully released. Third, in Figure 6(a), the cost of all the cases goes down, as n increases. We explain this observation, by checking the first derivative of function $f(n)$ as $f(n)' = u_e^n \log u_e$. Since $u_e \in [0,1]$, there must exists $f(n)' \leq 0$. Thus, we prove that, $f(n)$ is a monotonic decreasing function with respect to n. Finally, according to Figure 6(b), as n increases, the cost saving to the optimal routing reduces, while the saving to the optimal allocation increases. The reason is that, when the congestion penalty is heavy (i.e., large n), the routing phase dominates the joint problem. Thus, in this case, the joint optimal solution leans to the routing optimization phase.

2) Total cost vs. popularity distribution: Figure 7 plots the total networking cost as a function of Zipf exponent α. In particular, by changing the value α, we are able to simulate different user request patterns (i.e., larger α indicates larger fraction of requests on those very popular segments), and examine its impact on the total cost.

We obtain the following observations from this experiment. First, in Figure 7(a), as α increases (i.e., more requests on popular segments), both the joint and the routing scheme yield less cost, while the cost of single optimal resource allocation method grows. This still can be understood by examining its first derivative with respect to α as $f(\alpha)' = \sum_{v \in V} C\lambda_v (m(x_v^\alpha - x_v)\log m - x_v(m^\alpha - m)\log x_v)$, where C is a positive constant. By substituting $m = 20000$, and $x_v = 220, \forall v \in V$ from the optimal routing baseline case into $f(\alpha)'$, we find $f(\alpha)' > 0$ for $\alpha > 0,\ \alpha \neq 1$. On the other hand, in optimal allocation baseline and our joint method, x_v is set corresponding to λ_v. Therefore, for some nodes with large λ_v and x_v, there is $(m(x_v^\alpha - x_v)\log m - x_v(m^\alpha - m)\log x_v) < 0$. This leads to their summation $f(\alpha)' < 0$. Second, from Figure 7(b), we find the cost saving to the single optimal routing scheme decreases, as α increases. This implies that, when the popularity distribution has small tail, the optimal resource allocation phase dominates the joint problem.

VI. Conclusion and Future Works

In this paper, we jointly investigated the NFV-based resource allocation and SDN-based routing for cost-efficient video distribution over future Internet. Specifically, we propose a two-step iterative approach. First, in the resource allocation phase, we maximize total cache hits by optimally allocating storage and computing resources for a giving routing policy. Second, in the routing phase, we minimize the networking cost by optimally configuring the routing matrix for a given resource placement. Finally, we analytically prove their iterative repeat converges to the joint optimum. Through extensive simulations, we verified the convergence and optimality of our approach. Besides, by examining those numerical results, we got some operational guidelines. From the resource allocation aspect, more resource should be allocated to the node with heavier request rate. From the routing aspect, for each node-server pair, the node should split the traffic across multiple paths with identical shortest hops if there are many, or use the shortest path alone if there is only one.

In our future work, first, we are in the process of implementing a real test-bed, by using OpenStack [31] as the NFV orchestrator, Open vSwitch [32] as the SDN switch, and OpenDaylight [33] as the SDN controller, on top of a private cloud. Second, we will try to apply this iterative approach to orchestrate other virtualized network functions.

Appendix

A. Proof of Lemma 1

In order to ease the analysis and derive meaningful results, we assume the total segment amount m is sufficiently large ($m >> 1$), and the Zipf parameter α can not be exactly equal to 1 (but can be arbitrarily closed to 1). Thus, we approximate the local cache hit ratio by using a continuous function as,

$$p_{hit}^v \approx \frac{\int_1^{x_v} t^{-\alpha} dt}{\int_1^m t^{-\alpha} dt} = \frac{x_v^{1-\alpha} - 1}{m^{1-\alpha} - 1},\ \alpha > 0,\ \alpha \neq 1. \quad (22)$$

Similarly, we approximate the local transcoding ratio as,

$$p_{tr}^v \approx \frac{x_v^{1-\alpha} - y_v^{1-\alpha}}{m^{1-\alpha} - 1}(1 - p_r),\ \alpha > 0,\ \alpha \neq 1. \quad (23)$$

Based on those approximations, we then check the convexity of the objective function and each constraint.

First, by importing $z_v = x_v^{1-\alpha}$ as an auxiliary variable, we rewrite the objective function (9) into, $g_0(\mathbf{y}, \mathbf{z}) = \sum_{v \in V} p_{hit}^v = \sum_{v \in V} \frac{z_v - 1}{m^{1-\alpha} - 1}$, which is clearly linear to both z_v and y_v.

Second, we rewrite the constraint (10) into,

$$g_1(\mathbf{y}, \mathbf{z}) = \sum_{v \in V} (z_v^{\frac{1}{1-\alpha}} b_r + y_v (\sum_{i=1}^{r-1} b_i - b_r)) - o_{tot} \leq 0, \quad (24)$$

where we have $\frac{\partial^2 g_1}{\partial y_v^2} = \frac{\partial^2 g_1}{\partial y_v \partial z_v} = \frac{\partial^2 g_1}{\partial z_v \partial y_v} = 0$, and $\frac{\partial^2 g_1}{\partial z_v^2} = \frac{\alpha b_r z_v^{\frac{-1-2\alpha}{1-\alpha}}}{(1-\alpha)^2}$, where $\alpha > 0$, $b_r > 0$, and $z_v^{\frac{-1-2\alpha}{1-\alpha}} > 0$, leading to $\frac{\partial^2 g_1}{\partial z_v^2} > 0$. Therefore, we obtain its Hessian matrix with respect to $\mathbf{y}$ and $\mathbf{z}$ as a diagonal matrix that, entries outside the main diagonal are all zero. Thus, the k-th determinant minor of $\mathbf{H}(\mathbf{g_1})$ is,

$$|\mathbf{H}(\mathbf{g_1})_k| = \begin{cases} \prod_{i=1}^k \frac{\partial^2 g_1}{\partial {z_i}^2}, & k \leq |V| \\ 0, & |V| < k \leq 2|V| \end{cases}. \quad (25)$$

This readily shows that, the Hessian matrix is positive semidefinite, that the determinant of all principal minors are greater or equal to 0. Thus, $g_1(\mathbf{y}, \mathbf{z})$ is jointly convex in $\mathbf{y}$ and $\mathbf{z}$.

Next, we examine constraint (11) by rewriting it into,

$$g_2(\mathbf{y}, \mathbf{z}) = \sum_{v \in V} \lambda_v \frac{(z_v - y_v^{1-\alpha})(1-p_r)}{m^{1-\alpha} - 1} \sum_{j=1}^{r-1} p_j b_j \leq 0, \quad (26)$$

where we still have $\frac{\partial^2 g_2}{\partial z_v^2} = \frac{\partial^2 g_2}{\partial y_v \partial z_v} = \frac{\partial^2 g_2}{\partial z_v \partial y_v} = 0$, and,

$$\frac{\partial^2 g_2}{\partial {y_v}^2} = \frac{\lambda_v \alpha(\alpha-1)(1-p_r)}{1 - m^{1-\alpha}} {y_v}^{-\alpha-1} \sum_{j=1}^{r-1} p_j b_j. \quad (27)$$

When $0 < \alpha < 1$, there are $\alpha - 1 < 0$, and $1 - m^{1-\alpha} < 0$, resulting in $\frac{\partial^2 g_2}{\partial {y_v}^2} > 0$. When $\alpha > 1$, we have $\alpha - 1 > 0$, and $1 - m^{1-\alpha} > 0$, which still leads to $\frac{\partial^2 g_2}{\partial {y_v}^2} > 0$. Therefore, we prove $\frac{\partial^2 g_2}{\partial {y_v}^2} > 0$, for all $\alpha > 0$, $\alpha \neq 1$. Following this, we have its Hessian matrix $\mathbf{H}(\mathbf{g_2})$ still as a positive semidefinite diagonal matrix. Thus, $g_2(\mathbf{y}, \mathbf{z})$ is jointly convex in $\mathbf{y}$ and $\mathbf{z}$.

Finally, since $p_{miss}^v = 1 - p_{hit}^v$ is linear function in $\mathbf{z}$, the constraint (12) is also linear over $\mathbf{y}$ and $\mathbf{z}$.

In summary, since all these equations are jointly convex in $\mathbf{y}$ and $\mathbf{z}$, we conclude that, the virtualized resource allocation problem is a convex optimization problem.

B. Proof of Theorem 1

In this subsection, we accomplish the proof of theorem 1, by examining the sufficient conditions for convergence of the Gauss-Seidel algorithm. In particular, from [26], if h and all constraints are 1) bounded from below; 2) differentiable; 3) marginally convex in $\mathbf{R}$, $\mathbf{x}$, and $\mathbf{z}$; and 4) jointly convex in $\mathbf{R}$, $\mathbf{y}$, and $\mathbf{z}$, then it will converge to the minimizer of h.

The first three conditions are satisfied through the definitions from our system models. Specifically, condition 1 is satisfied, because $\mathbf{x} \succeq 0$, $\mathbf{y} \succeq 0$, $\mathbf{z} \succeq 0$, and $\mathbf{R} \succeq 0$ by definition. Condition 2 is satisfied, since h and all constraints are differentiable. Condition 3 is satisfied, since p_{hit}^v, f and all constraints are convex in either $\mathbf{z}$ or $\mathbf{R}$ as shown in Lemma 1.

Next, we show the last condition is also satisfied. In particular, we introduce a new variable $w_{lv} = R_{lv} \lambda_v b_m p_{miss}^v$ to indicate the traffic flow size at link e that is heading towards node v. Thus, constraint (8) also changes into,

$$\sum_l w_{lv} = \lambda_v b_m p_{miss}^v, \quad (28)$$

which is still linear jointly to $\mathbf{w}$ and $\mathbf{z}$. Following this, those original constraints are still jointly convex over $\mathbf{w}$ and $\mathbf{z}$, because constraint (10) and (11) are independent of $\mathbf{R}$ and $\mathbf{w}$, and constraint (12) is clearly linear to $\mathbf{w}$. Finally, we complete the proof on the convexity of the modified objective function,

$$h(\mathbf{w}, \mathbf{z}) = \sum_{e \in E} f(\sum_{v \in V} \sum_l w_{lv} I_{lv}^e / d_e) - \sum_{v \in V} \lambda_v p_{hit}^v, \quad (29)$$

where $f(\sum_{v \in V} \sum_l w_{lv} I_{lv}^e / d_e)$ is strictly convex in $\mathbf{w}$ by definition, and p_{hit}^v is convex in $\mathbf{z}$ as shown in Lemma 1.

REFERENCES

[1] Sandvine, "The global internet phenomena report–1h2013," 2013.

[2] Cisco, "Cisco visual networking index: Forecast and methodology, 2013-2018," White Paper, 2013.

[3] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM*, vol. 49, pp. 101–106, 2006.

[4] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A server's perspective of Internet streaming delivery to mobile devices," in *IEEE INFOCOM*, 2012, pp. 1332–1340.

[5] AT&T *et al.*, "Network functions virtualisation - introductory white paper," http://portal.etsi.org/nfv/nfv_white_paper.pdf, 2012.

[6] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011.

[7] Y. Wen, X. Zhu, J. Rodrigues, and C. Chen, "Cloud mobile media: Reflections and outlook," *IEEE Transactions on Multimedia*, 2014.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[9] H. E. Egilmez, B. Gorkemli, A. M. Tekalp, and S. Civanlar, "Scalable video streaming over openflow networks: An optimization framework for qos routing," in *IEEE ICIP*, 2011, pp. 2241–2244.

[10] A. Arefin, R. Rivas, R. Tabassum, and K. Nahrstedt, "OpenSession: SDN-based cross-layer multi-stream management protocol for 3D teleimmersion," in *IEEE ICNP*, 2013.

[11] D. Li, Y. Shang, and C. Chen, "Software defined green data center network with exclusive routing," in *IEEE INFOCOM*, 2014.

[12] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *IEEE INFOCOM*, 2012, pp. 2444–2452.

[13] Y. Li, H. Xie, Y. Wen, and Z.-L. Zhang, "Coordinating in-network caching in content-centric networks: model and analysis," in *IEEE ICDCS*, 2013, pp. 62–72.

[14] Y. Jin and Y. Wen, "Paint: Partial in-network transcoding for adaptive streaming in information centric network," in *IEEE/ACM IWQOS*, 2014.

[15] Z. Wang *et al.*, "Joint online transcoding and geo-distributed delivery for dynamic adaptive streaming," in *IEEE INFOCOM*, 2014.

[16] W.-P. Yiu, X. Jin, and S.-H. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1717–1731, 2007.

[17] Z. Li *et al.*, "On the geographic patterns of a large-scale mobile video-on-demand system," in *IEEE INFOCOM*, 2014.

[18] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *IEEE INFOCOM*, vol. 1, 1999, pp. 126–134.

[19] S. Guo, H. Xie, and G. Shi, "Collaborative forwarding and caching in content centric networks," in *IFIP Networking*, 2012, pp. 41–55.

[20] AmazonEC2, "Amazon elastic transcoder pricing," http://aws.amazon.com/elastictranscoder/pricing/, 2014.

[21] A. Chanda, C. Westphal, and D. Raychaudhuri, "Content based traffic engineering in software defined information centric networks," in *IEEE Infocom NOMEN workshop*, 2013.

[22] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *ACM SIGCOMM*, 2013, pp. 3–14.

[23] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *NSDI*, vol. 13, 2013, pp. 29–42.

[24] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002.

[25] J. Rexford, "Route optimization in IP networks," in *Handbook of Optimization in Telecommunications*, 2006, pp. 679–700.

[26] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Athena Scientific, 1997.

[27] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[28] "Internet2 layer 3 topology," http://www.internet2.edu/media/medialibrary/2013/10/01/I2-Network-Infrastructure-Layer-3.pdf.

[29] S. Lederer *et al.*, "Distributed dash dataset," in *ACM Multimedia Systems Conference*, 2013, pp. 131–135.

[30] R. Grandl, K. Su, and C. Westphal, "On the interaction of adaptive video streaming with content-centric networking," in *Packet Video*, 2013.

[31] "Openstack," https://www.openstack.org/, 2014.

[32] "Open vswitch," http://openvswitch.org/, 2014.

[33] "Opendaylight," http://www.opendaylight.org/, 2014.