Smart Routing: Fine-Grained Stall Management of Video Streams in Mobile Core Networks

Jun He and Wei Song*

Faculty of Computer Science
University of New Brunswick
Fredericton, NB, Canada
Emails: jhe2@unb.ca, wsong@unb.ca

Abstract

Video traffic has dominated the global mobile data traffic and creates the fundamental need for continuous enhancement and fast evolution in mobile networks so as to accommodate its unprecedented growth. Despite the surging interests in Radio Access Networks (RANs), the latest technologies on dense and heterogeneous wireless networks are shifting the bottleneck of mobile networks to the core networks. However, managing the stalls of video streams in mobile core networks remains challenging. In an evolving mobile system, the core network needs to i) determine the data rate for each video streaming request, ii) distribute the video request among multiple sources, and iii) route the so-generated peer-to-peer flows. In this paper, we exploit user context and propose an optimized routing scheme (termed as Smart Routing) for stall management in mobile core networks, which adaptively schedules data rates with respect to user context and strategically routes so-scheduled video demands. The proposed smart routing scheme simultaneously addresses the above three aspects by formulating them in a joint optimization problem and solving the formulated problem with a fast algorithm with provable approximation guarantee. Computer simulations validate the efficiency of the proposed scheme.

Index Terms

Mobile core networks, video service, user context, stall management, optimization model, smart routing.

*Corresponding author. This research was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC).
I. INTRODUCTION

A. Mobile Video Traffic and System Evolution

The rapid proliferation of smart phones and tablets is profoundly changing the way we behave and consume content, shifting more and more data traffic from fixed networks to mobile networks. Global mobile data traffic is expected to explode by nearly 11-fold between 2013 and 2018, predominantly fueled by video traffic [1]. Up from 53% in 2013, mobile video will represent 69% of global mobile data traffic by 2018. Such substantial growing demands in both traffic volume and speed are out-pacing the ability of current mobile networks [2], e.g., the Long Term Evolution (LTE) systems of the Third Generation Partnership Project (3GPP), which is designed as “interconnecting middle-boxes” between public networks and mobile customers. On the other hand, recent developments in dense and heterogeneous wireless networks are shifting the bottleneck of mobile systems from Radio Access Networks (RANs) to the backhauls, i.e., mobile core networks [3,4].

To accommodate the surging demands, components and protocols of LTE core networks are enhanced or re-factored by integrating transparent video caching and incorporating more efficient request-scheduling algorithms [5]–[8]. Meanwhile, mobile systems are undergoing fast evolution from the current fourth generation (4G) technologies, i.e., LTE/LTE-Advanced, to the fifth generation (5G) [4] with virtualized networking functions [9]–[11]. Various video sources, such as in-system caching [7], Mobile Content Distribution Networks (MCDNs) [12], cloud service points [10] and improved routers [13], will be deployed within mobile core networks so as to move data close to customers. Transport and routing technologies, together with computing and storage resources, will be embedded in mobile core networks to build up a converged infrastructure which orchestrates the delivery of data traffic. To promote manageability and flexibility, the control plane is separated from the data plane in LTE systems and evolving mobile systems. As such, control information is delivered in the control plane while data traffic is transferred in the data plane. This enables centralized networking management in mobile core networks, e.g., through a coordinating component implemented at the Packet Data Network (PDN) gateway of an LTE system or
the central controller deployed in a Software Defined Networking (SDN)-enabled mobile core [14].

Considering video services, it is critical and challenging to enhance the Quality of Experience (QoE) of mobile customers. However, the evolving systems still rely on conventional best-effort routing protocols [15,16] running in lower layers, which fail to make use of the new functionalities to improve QoE and will inevitably let the network bottleneck incline to the core networks. Therefore, mobile core systems are in an urge need of advancing the routing approach with the new network developments and supporting more efficient data delivery and improved QoE.

As we know, QoE is a comprehensive assessment of user-perceived service quality and it depends on a variety of key performance indicators such as latency, data rate, video resolution (spatial and temporal), packet loss and stalling. In particular, stalling is very detrimental to the QoE of video streams [17]. Hence, in this paper, we specifically focus on stall management and aim to provide fine-grained traffic control for video streams in an evolving mobile core network as depicted in Fig. 1. The mobile core consists of a central controller and a set of nodes, including (virtualized) internal content sources\(^1\), intermediate

\(^1\)These internal sources are virtualized sources using the technologies in [9,10] and can fulfill all the video requests from the controller’s view. For example, physical external sources, e.g., CDNs, can be virtualized as internal sources located at the interconnecting gateways.
forwarding servers and edge servers. Edge servers sitting at the edge of the mobile core provide network service to RANs. The solid lines therein represent the data plane, while the dashed lines represent the control plane. In the mobile core network, the central controller periodically gathers network and traffic information, and accordingly shapes network flows and optimizes the network performance.

This architecture is an instant abstraction of future mobile networks with SDN-enabled mobile cores [14]. By mapping the controller to the PDN gateway, edge servers to serving gateways, and intermediate servers to routers, we note that this architecture can also be considered as an enhanced LTE system with built-in content storage. Furthermore, in some cases, internal sources and serving functionalities can be incorporated in evolved routers. This maps the mobile core in Fig. 1 to an information-centric network [13]. Therefore, this model under study is quite general.

B. Request Routing and Stall Management

Video streaming requests are initiated by mobile devices and enter the core network through edge servers. Existing works on video streaming support in RANs often assume that the required data are always “pre-fetched” at edge servers [18]. Despite the numerous factors that affect user-perceived service quality, managing the stalls that occur during video sessions is essential to achieve better user experience. As the backhaul is becoming the network bottleneck, efficient request routing in the core network is of vital importance for stall management. Request routing in the core network is to determine how to deliver requested video streams to corresponding edge servers. As each video request can potentially be fulfilled by multiple sources through multiple paths inside the network, request routing hereby makes decisions on both source redirection, i.e., how to redirect requests to respective sources, and flow routing, i.e., how to route so-generated peer-to-peer flows in the network.

Current mainstream mobile devices equipped with advanced hardware generally have plenty of memory for buffering video streams. For a particular mobile device, we specifically consider the user context with respect to the video data buffered at user devices as well as the fashion in which the corresponding user consumes such data. With buffer, temporary transmission failure does not necessarily cause stalls (also
termed as jitters in [19]). As shown in Fig. 1, transmission curves represent how edge servers deliver data, while playout curves indicate how the data are actually consumed.

Detailed information of these curves is shown in Fig. 2, where x-axis is the time and y-axis is the aggregate amount of data received or consumed. The playback curve is a characteristic of corresponding video and is independent of underlying transmission and user context. Stalls happen only when the buffered data are not able to support normal playback. Depending on the data transmission and the stall-recovery scheme, a stall lasts for certain duration, referred to as *stall delay*. The term *playout lead* is adopted from [17], and represents the duration of time that the video can be played using only the data already buffered in the mobile device.

The playout lead plays a critical role in determining whether video playback stalls, and thus is a key factor to improve the QoE of mobile customers. For an interval considered, the playout lead of a video stream at the end of the duration depends not only on the amount of buffered data at the beginning of the interval, but also on the data rate adopted during the interval. Therefore, the data rates of video requests should be adapted according to respective user context, which is the third essential aspect of smart routing, *i.e.*, *data-rate selection*, in addition to source redirection and flow routing.
C. Scope and Contributions

In this paper, we focus on stall management of video streams in evolving mobile core networks with virtualized sources. Video requests are collected from edge servers, while adaptive streams are also delivered to edge servers as responses. Here, we consider (YouTube-like) on-demand video services, for which video data are delivered and buffered at mobile devices before playback. Also, it is assumed that stalls occur only when the buffered video data cannot support the playback. Data transmission from edge servers to RANs and within RANs is not considered in this paper, but it is also an active research area in recent years, e.g., heterogeneous wireless access [20] and SDN RANs [9]. We consider video streams that are delivered in a progressive downloading mode, i.e., a mobile device could buffer up to the entire video clip in its local memory. We also assume that the data consuming information at the buffer of remote devices can be efficiently tracked or estimated by core networks and is readily accessible to the controller [17] in the system under study. For example, in an LTE system, the amount of data delivered to remote devices is gathered by the Policy and Charging Rules Function (PCRF) component.

In this work, we not only aim to optimize request routing upon the instant request pattern and networking conditions, but also attempt to adjust instant demands of video requests upon user context. To address this, we formulate a joint optimization problem of request routing and data-rate selection. We then conduct extensive analysis on the formulated problem and propose fully polynomial-time approximation schemes to solve the problem for the scenarios with continuous data rates and discrete data rates, respectively. Routing decisions are displayed in a path-flow form, which are readily converted to a routing protocol and dispensed to routing nodes using the techniques from [10,21].

In summary, our contributions are three-fold. Firstly, to the best of our knowledge, we are the first to exploit user context for fine-grained stall management of video streams in mobile core networks. While most of recent research interests focus on RANs, this paper studies the essential routing problem in core networks, which will be the new bottleneck of mobile systems. As one of the key features of future mobile networks [4], a route of adapting network behavior to user context is sketched out in this work.
The problem and the system model we consider here are essential to a robust and high-performance core system in various mobile networks evolving to 5G.

Secondly, we formulate an optimization framework which jointly considers data rate adaptation, source redirection and flow routing. We aim to manage stalls of video streams by maximizing minimum playout lead over the whole network while maintaining the maximum link utilization under a given threshold. By tracking the average playout lead over concurrent requests, our framework reveals what performance can be achieved by strategic routing, and provides important insights on how the network should be re-factored and whether hardware upgrade is required.

Thirdly, we analyze the hardness of the formulated problem and propose fast algorithms to solve it. The problem is proven to be NP-hard when data rates are discrete, while the resulting linear programming (LP) problem with continuous data rates is shown to be over-sized for LP solvers. Algorithms are developed for both cases and are extensively studied via theoretical analysis and computer simulations. We also show that the proposed algorithms achieve provably approximation ratios with tolerable computing cost and negligible traffic overhead.

The rest of this paper is organized as follows. Section II presents related works. In Section III, we describe the system model, propose the optimization framework and analyze the hardness of the formulated problems. Section IV gives our solutions to the formulated problems. Simulation studies are presented in Section V. Finally, Section VI concludes the paper.

II. RELATED WORKS

Extensive recent research has been conducted towards the wireless communications domain in mobile networks [3,22,23]. As the techniques on dense and heterogeneous networks are maturing, the efforts on RANs tend to consider more about mobility management and energy efficiency, while the network bottleneck is now being shifted to the core networks [3]. On the contrary, conventional routing protocols work in a best-effort manner and fail to accommodate the growing mobile traffic or provide the necessary
management functionalities for the evolving mobile systems. This paper fills such a research gap and proposes a more efficient and manageable routing solution for mobile core networks.

Transparent caching has attracted broad recent interests from both academia and industry, as an efficient approach to improve the performance of rich multimedia services, e.g., video streams, over mobile networks. Most works in this area focus on content placement, e.g., [8,24,25]. On the other hand, mobile networks are undergoing fast evolution. SDN-enabled mobile networks [26] are becoming the mainstream design of 5G, where the separation of control plane and data plane will be enhanced and network virtualization together with source virtualization [9] will be implemented in mobile core networks. In an SDN network, network and traffic dynamics are monitored and controlled by a central controller. A prototype of an SDN-like mobile core was developed in [14]. To enable fine-grained traffic control, future networks will attempt to adapt their behaviors depending on user context [4]. This paper presents the first step towards exploiting user context in traffic control and stall management for video streams in mobile core networks.

Provided the fact that source selection or CDN redirection remains intractable in generic networks [27,28], request routing, which involves both source redirection and flow routing, is therefore a rather difficult problem. The request routing problem was first formulated and studied in our previous work [21], where we proposed a fast approximation algorithm to solve it. A complete routing protocol with source virtualization is then developed in [10] for networks evolving to 5G, where a central controller makes routing decisions such that internal video sources are then virtualized as one “super” sourcing server to the remote devices. This paper complements the above studies by considering adjustable video demands from interval to interval, while referring to [10,21] for post-procedure of protocol implementation after the formulated problem is solved.

As the dominant traffic in mobile networks, the characteristics of video traffic have been widely studied in the literature [29,30]. The playback characteristic of a video is an essential factor that can be exploited to manage stalls and improve QoE. In [17], Dutta et al. defined the playout lead and attempted to prevent
stalls by maximizing the minimum lead over multiple streams. Likewise, Liang et al. aimed to reduce stall delay by adapting variable bit-rate of video streams to dynamic wireless channels and user buffer [19]. These studies are conducted towards a single-hop wireless network while assuming all requested data have been pre-fetched and stored at base stations. On the contrary, our work exploits user context to optimize data traffic in the core network. Then, advanced techniques such as [17,19] can be employed in the RAN after the core network has conveyed requested data through edge servers to base stations using the schemes developed in this paper. Therefore, our study completes the application of existing works to mobile networks by validating the pre-fetching assumption.

III. SYSTEM MODEL AND FORMULATIONS

A. System Model and Basic Notations

We consider an evolving mobile core network as depicted in Fig. 1. Video requests are initiated by mobile devices and reach the core network through corresponding edge servers. Every video request can be fulfilled by one or more sources in the core network. We assume that the controller has complete static information of the core network, and is able to periodically collect dynamic load information and video requests from the control plane. As envisioned in many works [10,13,14,16,21], the evolving mobile core networks tend to maintain a central controller to gather the network-wide real-time information from a separate control plane and optimize network performance through “decisions” or “policies” to network components via the control plane. The trend of centralization and separation of control plane and data plane will be reinforced in future SDN-enabled mobile networks. Compared to the cost of supporting charging and policy functions, the overhead of gathering video request information is negligible [10]. Therefore, it is feasible, practical and affordable for the controller to obtain such dynamic information.

Taking advantage of the collected information, the controller can run request optimization from interval to interval. Different from [17] where the interval should be small enough to catch up with the dynamics of the wireless channel condition, here in the core networks, we consider a much greater interval duration
in a time scale comparable to the dynamics of user’s engagement, typically a few seconds [31]. Within the period of each interval, the request patterns are assumed fixed, and the related information, such as video clips, buffered data amount at user devices and playing time, is assumed available. As pointed out in [17,19], the eNodeBs or access points at the last hop to user devices can efficiently collect or estimate the information such as buffered data amount and playing time. Since these access points are directly connected to the edge servers in the mobile core networks, it is reasonable to assume that the controller can obtain such information with a sufficient accuracy.

Conventional routing schemes target optimizing certain network metric by selecting paths for fixed peer-to-peer flow demands. In our model, the controller also needs to solve the source-redirection problem, i.e., how to select sourcing nodes to fulfill each request. To reduce stalling and thus enhance QoE, we further consider the adaptation of video demands depending on respective playing time and data amount at remote buffer. Thus, our solution integrates complementary aspects such as load balancing over multiple paths and multiple servers as well as fairness among sessions, which can thereby mitigate the performance fluctuation caused by inaccurate information if any. Even though the inaccurate information leads to sub-optimal control decisions, the adverse impact will last for a short period, since more accurate updates in the subsequent time intervals are expected to correct the decisions timely.

Table I summarizes the important notations used in this paper. Here, the mobile core network is formulated as a directed graph $G = (N, V, E)$, where $N$ is the set of video clips (or interchangeable chunks) with $n = |N|$, $V$ is the set of interconnected networking nodes with $v = |V|$, and $E$ is the set of links with $m = |E|$. These networking nodes include intermediate servers, edge servers as well as data-source servers. Link $e \in E$ has a capacity $c(e)$, of which up to $\lambda_0$ fraction can be used for video delivery. For every pair of nodes $i, j \in V$, let $\mathcal{P}_{ij}$ be the set of paths from $i$ to $j$, and $\mathcal{P} = \bigcup_{(i,j)} \mathcal{P}_{ij}$ be the union of all path sets. Moreover, let $\mathcal{P}_e$ be the set of all paths in $\mathcal{P}$ that use edge $e \in E$.

Regarding video requests, we define $S_k$ as the set of sourcing nodes capable of serving requests for video clip $k \in N$. The set of requests is denoted by $R$ with $r = |R|$. For a request $(i, k) \in R$ at edge
<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>Set of in-network links</td>
</tr>
<tr>
<td>$L(i, k)$</td>
<td>Playout lead of request $(i, k)$ at the end of next interval</td>
</tr>
<tr>
<td>$N$</td>
<td>Set of video clips cached in the system</td>
</tr>
<tr>
<td>$\mathcal{P}$</td>
<td>Set of all in-network paths</td>
</tr>
<tr>
<td>$\mathcal{P}_e$</td>
<td>Set of paths using edge $e$</td>
</tr>
<tr>
<td>$\mathcal{P}_{ij}$</td>
<td>Set of paths from node $i$ to node $j$</td>
</tr>
<tr>
<td>$\mathcal{P}^k_i$</td>
<td>Set of paths available to request $(i, k)$</td>
</tr>
<tr>
<td>$R$</td>
<td>Set of all video requests</td>
</tr>
<tr>
<td>$R_i$</td>
<td>Set of requests at node $i$</td>
</tr>
<tr>
<td>$S_k$</td>
<td>Set of nodes that store video $k$</td>
</tr>
<tr>
<td>$V$</td>
<td>Set of networking nodes</td>
</tr>
<tr>
<td>$b^k_i$</td>
<td>The data buffered at user device w.r.t. request $(i, k)$</td>
</tr>
<tr>
<td>$c(e)$</td>
<td>The capacity of edge $e$</td>
</tr>
<tr>
<td>$d^k_i$</td>
<td>The adjustable data rate of request $(i, k)$</td>
</tr>
<tr>
<td>$l^k_i$</td>
<td>The minimum scheduling data rate for request $(i, k)$</td>
</tr>
<tr>
<td>$t^k_i$</td>
<td>The current playing time of request $(i, k)$</td>
</tr>
<tr>
<td>$x(P)$</td>
<td>Amount of flow routed on $P \in \mathcal{P}^k_i$ w.r.t. request $(i, k)$</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>The routing scheduling interval</td>
</tr>
<tr>
<td>$\lambda_0$</td>
<td>The maximum fraction of link capacity for video service</td>
</tr>
</tbody>
</table>

node $i$ for video $k$, we further use $\mathcal{P}^k_i$ to denote the set of all available paths\(^2\) to serve request $(i, k)$, i.e., $\mathcal{P}^k_i = \bigcup_{j \in S_k} \mathcal{P}_{ji}$.

At the remote side, a central controller is assumed to gather or estimate the real-time user context. Here, the user context includes the following specific factors: 1) requested video, i.e., request $(i, k)$ is

\(^2\)Available paths are logical paths with respect to specific requests. For instance, if two or more requests share one common physical path, it is considered repeatedly for each request, i.e., no sharing paths.
associated with a corresponding playback curve; 2) the current playing time $t^k_i$ of the video; and 3) the amount of video data $b^k_i$ that is already buffered in the user device. For the next interval of duration $\Delta t$, the scheduled data rate of request $(i, k)$ consists of a fixed lowest rate $l^k_i$ and an adjustable rate of $d^k_i$. To optimize request routing for each time interval of $\Delta t$, we need to select the adjustable data rate $d^k_i$ for request $(i, k)$, assign multiple requests to sources and route so-generated peer-to-peer video streams.

**B. Objectives and Optimization Framework**

The ultimate concern for providing high-quality video streams is how to provide a smooth playback experience without or with the minimum stall (or re-buffering) delay. To strike this, we adopt the concept of playout lead from [17]. The playout lead of a video request represents the duration of extra time the remote user can play the video using only its buffered data. Accordingly, the playout lead of request $(i, k)$ at the end of current scheduling interval of $\Delta t$, denoted by $L(i, k)$, is defined as

$$L(i, k) = q_k(b^k_i + d^k_i \Delta t + l^k_i \Delta t) - t^k_i - \Delta t$$

where $q_k(\cdot)$ is the playout function of video $k$, which maps the aggregate amount of data to playout time (see Fig. 2). Different from [17], the playout lead definition here also includes the minimum data rate $l^k_i$. In other words, we need to maintain certain lowest data rate for all requests even though some of them can have greater playout lead. This is reasonable in our scenario with dynamic requests. Take the following instance as an example. A new session has just been attached, while an old session (termed as “last-second” request) has much greater lead and is about to terminate, e.g., 95% of the clip or chunk has been delivered. Since the new session has much less lead, the controller tends to assign more resources to it. Without the reservation of the lowest data rate, it is very likely that the old session stays in a “starvation” status without any resource for a long time before it is detached. Therefore, the guarantee of a minimum data rate also provides certain fairness among requests.

At the beginning of an interval, we route requests aiming to maximize the minimum playout lead by the end of the interval, while keeping the maximum link utilization under a given threshold. The problem
is then formulated as follows:

\[
\max_{(i,k) \in R} \min_{(i,k) \in R} L(i, k) \\
\text{s. t.} \quad \sum_{P \in \mathcal{P}} x(P) \leq \lambda_0 c(e), \quad \forall e \in E \quad (3) \\
\sum_{P \in \mathcal{P}} x(P) \geq d^k_i + l^k_i, \quad \forall (i,k) \in R \quad (4) \\
x(P) \geq 0, \quad \forall P \in \mathcal{P} \quad (5)
\]

where the objective \( L(i, k) \) is defined in (1), the first constraint indicates the link capacity limitation, scaled by factor \( \lambda_0 \), and the second constraint requires the demand for request \((i, k)\) be fulfilled. We note that the variables herein include path flow values \( \{x(P)\} \) as well as the adjustable rates \( \{d^k_i\} \). Here, \( \{d^k_i\} \) can be either continuous or discrete values, depending on the rate adaptation methods adopted.

Before carrying further discussion on problem (2), we first review the traffic engineering (TE) problem studied in our previous work [21], in which we attempt to minimize the maximum link utilization, \( i.e., \) the ratio of traffic load to link capacity. The TE problem is expressed as

\[
\min \lambda \\
\text{s. t.} \quad \sum_{P \in \mathcal{P}} x(P) \leq \lambda c(e), \quad \forall e \in E \quad (7)
\]

together with (4) and (5). Here, the first constraint in (7) indicates the link capacity limitation, scaled by the utilization factor \( \lambda \). We note that the optimal value of the TE problem is proportional to the attached traffic load, \( i.e., d^k_i + l^k_i \). The TE problem is a reduced case of problem (2), where data rates are fixed. In other words, if there exists an oracle for selecting the data rate set \( \{d^k_i\} \) for the next interval, solving the TE problem is sufficient to determine whether the selected rate set is routable under the maximum link-utilization limitation, \( i.e., \lambda_0 \).

C. Problem Analysis and Hardness Results

In [21], we reveal that the TE problem is an LP problem with an exponential number of variables \( x(P) \). Even though this number can be converted into polynomial size by re-formulating the problem
using edge-flow variables [32], the large number of video requests makes it infeasible to be solved by modern LP solvers, e.g., CPLEX [33].

Likewise, problem (2) has a similar size as the TE problem. If \( \{d^k_i\} \) are continuous variables, i.e., assuming the rate adaptation components at sourcing nodes can produce video streams of arbitrary bit rates, problem (2) is then an LP problem. However, the rudimentary size of variables prevents it from being efficiently solved by LP techniques. On the other hand, a feasible solution to problem (2) must compute routing decisions within certain time cost that is acceptable for \( \Delta t \). Therefore, a fast algorithm with approximation guarantee is more desirable.

The other case of problem (2) is a Mixed Integer Programming (MIP) problem, where each adjustable data rate \( d^k_i \) can only be chosen from a pre-defined rate set associated with video \( k \). This is a typical scenario in prevailing systems, e.g., videos are encoded and streamed in layers. Unfortunately, with discrete values of \( \{d^k_i\} \), there is no efficient algorithm for problem (2) unless P=NP. We state the hardness result in the following theorem.

**Theorem 1.** If the adjustable rates \( \{d^k_i\} \) are discrete, the optimization problem (2) is NP-hard.

**Proof.** We show that problem (2) at least has the same complexity as the following problem.

**Multiple Choice Knapsack Problem (MCKP):** There are a group of sets of items, \( I_1, I_2, \ldots, I_n \), and a knapsack with capacity \( W \). Each item \( i \in I_r, r = 1, \ldots, n \), has profit \( v_i \) and size \( w_i \). Fill the knapsack within its capacity so that exactly one item per set is selected and the aggregate profit of items in the knapsack is maximized.

First, we consider a slightly modified variant of problem (2), where the objective is to maximize aggregate lead over all requests (termed as MAL for ease of interpretation). Similar to [34], we further simplify the scenario by ignoring the routing problem. We assume that there is an oracle which can schedule any rate combinations under the condition that the aggregate data rate is within a fixed value \( W \). Then, we map the alternative rate set of each request to a set of items, the data rate to size \( w_i \) and the resulting lead to \( v_i \). Thus, problem MAL is reduced to MCKP. Since problem MCKP is a classical
NP-hard problem, we conclude that the problem of maximizing aggregate lead is also NP-hard.

Next, we reveal the relation of problem (2) and problem MAL by considering their decision form. The decision form of problem (2) is represented as: for an additional input $K > 0$, does there exist a feasible scheduling with minimum lead greater than $K$? The answer is positive only if the answer to the following question is “yes”: for the same $K$, does there exist a feasible flow assignment such that the aggregate lead is not less than $rK$? Therefore, problem (2) at least has the same hardness as problem MAL.

Accordingly, we conclude that problem (2) is at least NP-hard. This completes the proof.

We comment that even though the multiple choice knapsack problem allows dynamic programming solutions, the oracle assumed remains challenging for problem (2). In fact, routing network flows also has high complexity [35]. Hence, developing fast algorithms with approximation guarantee is more desirable than pursuing the optimal solution.

IV. SMART ROUTING: THE APPROXIMATE SCHEME

In this section, we propose a fast approximate solution to problem (2), where both time complexity and approximation ratio are guaranteed. We first briefly review the solution to the TE problem, and then present the proposed algorithm for problem (2) with the max-min lead. After that, we analyze the approximation ratio as well as the time complexity of the proposed algorithm.

A. The Approximate Algorithm for the TE Problem

The TE problem formulated in (6) has been extensively studied in our previous work [21]. For completeness, we here review the critical points of the algorithm from [21] and highlight the main conclusions therein. First, we formulated an equivalent formulation of problem (6) as follows:

$$\max \quad \pi \quad \text{ (8)}$$

$$\text{s. t.} \quad \sum_{P \in P} y(P) \leq c(e), \ \forall e \in E \quad \text{ (9)}$$

$$\sum_{P \in P_i^k} y(P) \geq \pi d^k_i, \ \forall (i, k) \in R \quad \text{ (10)}$$

$$y(P) \geq 0, \ \forall P \in P. \quad \text{ (11)}$$
In the preceding formulation, we scaled the data rate $d^k_i$ by factor $\pi$, and attempted to maximize such a factor. The equivalence of the TE problem and problem (8) can be easily shown by letting $x(P) = y(P)/\pi$ and $\lambda = 1/\pi$.

We then showed the similarity of problem (8) with the maximum concurrent flow problem in the category of multi-commodity flow (MCF) problems [36]. While the conventional MCF problem identifies a commodity by its source-destination pair, such source-destination pairs need to be determined in problem (8). Problem (8) can be viewed as a variant of the maximum concurrent flow problem with multiple sources and one destination for each commodity. Such a variant was first studied in [21], where we leveraged the results in [36] and proposed $(1 + \omega)$-approximation algorithm for it for any $\omega > 0$. In the following, we introduce the main results regarding this $(1 + \omega)$-approximation algorithm, while detailed proofs and discussions can be found in [21].

**Theorem 2.** There exists a fully polynomial-time approximation scheme (FPTAS) for problem (8), which achieves an approximation ratio of $(1 + \omega)$, for any $\omega > 0$.

**Theorem 3.** Let $r_{\text{max}}$ denote the maximum number of requests initiated from one node, i.e., $r_{\text{max}} = \max_{i \in V} |R_i|$. The algorithm in [21] computes a $(1+\omega)$-approximation solution to problem (8) in $\widetilde{O}((\omega^{-2} + \log r)m \cdot T_s)$ time, where $\tilde{O}(f) = O(f \cdot \log^{O(1)} m)$ and $T_s = O(v \log v + m + r_{\text{max}})$, for any $\omega > 0$.

Accordingly, the TE problem (6), and its equivalent problem (8), can be $(1 + \omega)$-approximately solved with computational complexity proportional to $\omega^{-2}$, for any $\omega > 0$.

**B. The Binary-Search Scheme**

Recalling the relation of the TE problem (6) and problem (2) revealed in Section III-B, we can solve problem (2) in the following via a binary-search scheme. By introducing a bound value $\alpha$, we can rewrite the max-min objective of problem (2) in the equivalent form below:

$$\max \quad \alpha$$

s. t. $L(i, k) \geq \alpha, \ \forall (i, k) \in R$
Corresponding to each lower bound $\alpha$ selected for the playout lead in (13), there is a set of adjustable data rates $\{d_k\}$ computed through the definition of the playout lead in (1). On the other hand, for a given data-rate set, the min-max link utilization can be estimated by solving a corresponding TE problem. The data-rate set is feasible if the estimated link utilization does not exceed the pre-set upper bound $\lambda_0$. Accordingly, the feasibility of a lower bound $\alpha$ is therefore verified. The objective of problem (12) is then to find the feasible upper bound for $\alpha$.

Apparently, we can obtain such an upper bound using binary search. The main procedure is listed in Alg. 1. Starting with a feasible initial lower bound $\alpha_-$ and an infeasible upper bound $\alpha_+$, Alg. 1 iteratively searches for the greatest feasible $\alpha$ until an accuracy threshold $\epsilon$ is reached. Note that $\epsilon$ and $\omega$ are in the same order.

We comment that Alg. 1 outputs a path-flow solution, in which adjustable data rates are implied. Together with the flow-conversion algorithm in [21], we obtain a set of hop-by-hop routing decisions, which jointly account for the three subproblems discussed in Section I, including data-rate selection, source server selection and flow routing.

We also note that $\alpha_+, \alpha_-$ and $\alpha$ in Alg. 1 are not necessarily positive values. For instance, when the system is under extremely heavy load, the proposed algorithm computes a path-flow assignment that minimizes the stall delay of the remote user with the worst buffer condition.

In the following, we will analyze the approximation results of Alg. 1, starting with continuous data rates and then the discrete scenarios.

C. Approximation Results with Continuous Data Rates

We denote the optimal results of the TE problem (6) and problem (12) by $\lambda_{OPT}$ and $\alpha_{OPT}$, respectively. We further denote the corresponding computational results generated by the algorithm in [21] and Alg. 1
Algorithm 1: Binary-Search Algorithm for Problem (2).

**Input:** Network graph $G = (N, V, E)$, capacities $\{c(e)\}$, video distribution $\{S_k\}$, request set $R$, minimum request demands $\{l^k_j\}$, link utilization upper bound $\lambda_0$, search bounds $\alpha_-$ and $\alpha_+$, accuracy threshold $\epsilon$, and approximation ratio $\omega$.

**Output:** Max-min lead result $\alpha$, link flow assignment $x$.

1: **while** $\alpha_+ - \alpha_- > \epsilon$ **do**
2: $\alpha \leftarrow (\alpha_+ + \alpha_-)/2$;
3: Compute adjustable data-rate set $\{d^k_i\}$ according to (1) such that (13) is satisfied;
4: For the given input flow demand, compute a $(1 + \omega)$-approximate result $\pi^*$ for problem (8) with output $y$ [21];
5: $\lambda \leftarrow 1/\pi^*$;
6: **if** $\lambda > \lambda_0$ **then**
7: Print “infeasible”;
8: $\alpha_+ \leftarrow \alpha$;
9: $z \leftarrow y$, $\pi^{fes} \leftarrow \pi^*$; // Record the latest feasible solution
10: **else**
11: Print “feasible”;
12: $\alpha_- \leftarrow \alpha$;
13: **end if**
14: **end while**
15: $x \leftarrow z/\pi^{fes}$;
16: Output $\alpha_-$ and $x$;

by $\lambda_c$ and $\alpha_c$, respectively. An instant result from Theorem 2 gives

$$\lambda_{OPT} \leq \lambda_c \leq (1 + \omega)\lambda_{OPT}. \quad (14)$$

According to the procedure of Alg. 1, we have the following result:

**Lemma 1.** If problem (12) is optimally solved to have the objective value $\alpha_{OPT}$, then in the solution, the corresponding subproblem of traffic engineering also reaches its optimal value $\lambda_{OPT}$. Furthermore, we have $\lambda_{OPT} = \lambda_0$.

**Proof.** For the first part of the theorem, the optimal solution to problem (12) produces a data-rate set
\( \{d^k_i\}_{OPT} \) and the optimal objective value \( \alpha_{OPT} \). For a given data-rate set \( \{d^k_i\}_{OPT} \), we have the TE subproblem (6). Meanwhile, the path flow in the optimal solution also results in a maximum link utilization value, denoted by \( \lambda' \), where \( \lambda' \leq \lambda_0 \). Assuming that the TE subproblem is not optimally solved by the solution, we have \( \lambda_{OPT} < \lambda' \).

Recalling the topology of the core network, we note that the network is a strongly connected graph, i.e., any two in-system nodes are reachable to each other through one-hop or multi-hop paths [37]. In fact, most mobile core networks, such as the evolved packet core (EPC) of LTE systems, are organized in certain hierarchical topologies, where each network entity can reach any other component through direct interfaces (e.g., X11) or higher-level nodes. Suppose that the minimum playout lead \( \alpha_{OPT} \) is associated with certain request\(^3 \) \((i_x, k_x)\). Therefore, if we re-schedule the path flow for \( \{d^k_i\}_{OPT} \) such that \( \lambda_{OPT} \) is reached, we can always add the amount of flow to feed request \((i_x, k_x)\) while keeping the maximum link utilization under \( \lambda' \), hence \( \lambda_0 \). This violates the initial assumption that \( \alpha_{OPT} \) is optimal. The analysis here also implies that in the optimal solution, the constraint of (3) must be tight, i.e., \( \lambda_{OPT} = \lambda_0 \).

This completes the proof.

In order to show the relation between \( \alpha_c \) and \( \alpha_{OPT} \), we next develop an upper bound for the optimal value \( \alpha_{OPT} \). We first assume that there exists an oracle for the TE problem in Alg. 1, which computes the optimal path-flow assignment such that \( \lambda_{OPT} \) is reached. Using the oracle for feasibility checking, Alg. 1 produces an objective value \( \alpha'_{c} \). According to Lemma 1, the theoretical optimal objective value is then \( \alpha_{OPT} \). Apparently, we have the following relation:

\[
\alpha'_{c} \leq \alpha_{OPT} \leq \alpha'_{c} + \epsilon \tag{15}
\]

\[
\alpha_{c} \leq \alpha'_{c}. \tag{16}
\]

For the sake of presentation, we rewrite the definition of playout lead in (1) as

\[
L(i, k) = \Delta t d^k_i / r^k_i + w^k_i + c \tag{17}
\]

\(^3\)The scenarios that multiple requests have the lead value \( \alpha_{OPT} \) can be analyzed in exactly the same way.
where $w_i^k$ is the constant part with respect to a specific request $(i, k)$, and $c$ is an independent constant value. Besides, $r_i^k$ represents the data rate of the playback curve, which is associated with the buffered data $b_i^k$ and is assumed to be a constant value for a small duration of interval [17]. The only variable to be determined in Alg. 1 is the adjustable data rate $d_i^k$.

On the other hand, we learn from [21] that the amount of flow demands is proportional to the maximum link utilization produced. That is, if we scale up all the transmission demands by multiplying a factor $\kappa$, then the maximum link utilization is increased by the same factor. Let $\{d_i^k\}$ be the adjustable data-rate set computed by Alg. 1. Together with (14), if we could compute an optimal solution to the TE subproblem, the resulted delivering data rate for requests would be at most $(1+\omega)$ times of that generated by Alg. 1, i.e., for request $(i, k)$, the fulfilled data rate is at most $(1+\omega)(l_i^k + d_i^k)$ in the optimal solution. Correspondingly, $d_i^k$ is increased by $(1+\omega)$ at most.

Therefore, the lead of any request $(i, k)$ in the optimal solution, denoted by $L'(i, k)$, satisfies

$$L'(i, k) \leq \Delta td_i^k(1+\omega)/r_i^k + w_i^k + c$$

$$= L(i, k) + \Delta t\omega d_i^k/r_i^k.$$  (18)

If $(i', k')$ is the request with the minimum playout lead, it also satisfies (18). Hence, we have

$$\alpha_c' \leq \alpha_c + \omega d_{i'}^k \Delta t/r_{i'}^k.$$  (19)

Then, according to (15), we have

$$\alpha_{OPT} \leq \alpha_c' + \epsilon$$

$$\leq \alpha_c + \omega d_{i'}^k \Delta t/r_{i'}^k + \epsilon.$$  (20)

Defining $\delta = \omega d_{i'}^k \Delta t/r_{i'}^k + \epsilon$, we then arrive at the conclusion that

$$\alpha_c \leq \alpha_{OPT} \leq \alpha_c + \delta.$$  (21)

Although $\delta$ is associated with the minimum adjustable data rate $d_{i'}^k$, it can be arbitrarily minimized by proper selection of $\omega$ and $\epsilon$. In real systems, the minimum data rate $d_{i'}^k$ produced by Alg. 1 is not far greater than its playback rate $r_{i'}^k$, e.g., $0 \leq d_{i'}^k/r_{i'}^k \leq 3$. Hence, $\delta$ is in the same order as $\omega$ and $\epsilon$. 
Finally, we conclude the following result:

**Theorem 4.** For problem (2) with continuous adjustable data rates, there exists a polynomial-time approximation algorithm which computes a \(\delta\)-suboptimal solution, for any \(\delta > 0\).

### D. Approximation Results with Discrete Data Rates

In the discrete data-rate scenarios, adjustable data rate \(d^k_i\) for each request \((i, k)\) is selected from a pre-defined data-rate set instead of any arbitrary positive value. Under this assumption, we denote the output objective of Alg. 1 by \(\beta_c\), and the optimal objective for problem (12) by \(\beta_{OPT}\), which is proven NP-hard to be found.

We note that Theorem 1 no longer stands here. However, if there exists an oracle to solve the corresponding TE subproblem optimally, using this oracle for feasibility checking in Alg. 1, we still have the following inequality:

\[
\beta'_c \leq \beta_{OPT} \leq \beta'_c + \epsilon
\]

where \(\beta'_c\) is the objective value output by the oracle-embedded algorithm.

Now with the output of Alg. 1, we assume that extra improvement to playout lead with fractional adding data rate is allowed. With the same technique as that in Section IV-C, we can always find some \(\Delta \beta\) controlled by \(\omega\), such that

\[
\beta_c \leq \beta'_c \leq \beta_c + \Delta \beta. \quad (22)
\]

Accordingly, by introducing \(\delta = \Delta \beta + \epsilon\), we have

\[
\beta_c \leq \beta_{OPT} \leq \beta_c + \delta. \quad (23)
\]

Therefore, the approximate conclusion of Theorem 4 is still tenable when adjustable data rates are discrete.

### E. Running Time and Protocol Overhead

The algorithm given in Alg. 1 includes \(\log B\) times of solving the TE subproblem, where \(B\) is the largest number used to specify binary search with respect to the initial bound setting \((\alpha_-, \alpha_+)\), and the accuracy
factor $\epsilon$. Together with Theorem 3, we can find a solution to problem (2) in $\tilde{O}(\omega^{-2} + \log r)m \cdot T_s \cdot \log B)$ time. If $\omega$, $\epsilon$, and $\delta$ are in the same order, we conclude that Alg. 1 computes a $\delta$-suboptimal solution to problem (2) in time complexity that is proportional to $\delta^{-2}$.

With the output value of $\{x(P)\}$ and data-rate set $\{d_k\}$, Alg. 1 is readily converted to a hop-by-hop, destination-driven routing protocol, using the techniques developed in [10,21]. Such a conversion takes a time linear to the number of requests as well as the graph size of the network. Detailed procedure can be found in [10,21]. After the conversion, flow-splitting decisions are disseminated via the control plane. The traffic overhead is negligible compared to regular control commands, e.g., charging or policies, in the control plane.

V. PERFORMANCE EVALUATION

A. Setup

We employ the simulator developed in [21] and implement the rate adaptation algorithms with around 1000 lines of C++ code. We evaluate our algorithms in a core network with 50 nodes including 30 edge nodes and 20 intermediate nodes. We consider that 200,000 video clips are randomly distributed among the intermediate nodes so that each clip has an average number of 5 replicas. To further capture the diversity of videos and exploit their playback curves for request routing, we randomly map each of the 200,000 simulated videos to one video in the dataset from [38]. The videos in the dataset are formatted in MPEG-4 Variable Bit Rate and play out at a fixed frame rate of 30 frames per second. The data rate of each video can be adjusted from 128 kbps to 1.5 Mbps continuously or discretely with step size of 128 kbps. To simulate YouTube-like short videos, we only consider a clip of each mapped video for a random duration. The mean play length of all simulated videos is ensured at 252 seconds [39]. The threshold for maximum link utilization $\lambda_0$ is 0.9. The default values of $\epsilon$ and $\omega$ are 0.1 and 0.05, respectively.

The basic topology in use is a hybrid star-ring network, where 49 nodes are interconnected in a ring and each node on the ring has a link to the node at the star center (see Fig. 3). To further evaluate the
efficiency of dynamic routing, we also simulate a random network generated by Inet-3.0 [40] with a mean degree of 4 for each node. All links are assumed to be bi-directional with capacity of 1 Gbps for each direction. In the simulations, we control a load factor named traffic density, which denotes the average number of requests collected from edge servers at the beginning of each interval.

B. Dynamic Request Routing vs. Static-Configuration Routing

Current video systems, e.g., Netflix [27], use static configurations for source redirection, which potentially configures each client to one source with the minimum measured delay. A client-to-source configuration is changed only when the client experiences an over-threshold delay. Flow-level optimization in current systems depends on conventional lower-layer schemes, such as the Open Shortest Path First (OSPF) protocol. For comparison purposes, we implement the static-configuration routing scheme in our experiments, where each request is fulfilled by the nearest source, and the so-generated IP traffic is optimized by the OSPF algorithm.

We align the dynamic request routing in this paper with the static-configuration routing, and show the results of maximum link utilization in Fig. 4(a) and Fig. 4(b) for the random topology and the star-ring topology, respectively. To demonstrate the performance gain, a dividing line is shown in Fig. 4(b) (labelled as “over-cong line”), which indicates that the link is over congested with the maximum utilization of 1. We
can see that the dynamic request routing significantly outperforms the scheme with static configurations. The performance gain is attributed not only to the dynamic optimization of flow routing, but also to the concurrent fulfilling of requests by multiple sources, \textit{i.e.}, each stream can be served by multiple servers, and the strategic routing of each flow over multiple paths, thus achieving better load balancing.

\textbf{C. Context-Aware Rate Adaption vs. Speculative Schemes}

For comparison fairness, we assume that the dynamic request routing is adopted by all remaining experiments in this section, when evaluating the efficiency of context-aware rate adaptation and comparing it to speculative schemes.

One baseline scheme for comparison is to associate a fixed data rate with the corresponding encoding rate, termed as the \textit{Fixed-Rate} scheme. This scenario exists in non-buffer streaming service, \textit{e.g.}, live streaming or dynamic adaptive streaming over HTTP (DASH) \cite{41}. Here, we use 1.20 times of the encoding rate as the transmission data rate. Another speculative scheme is the \textit{Burst & Fixed} strategy, (possibly) adopted by YouTube \cite{39}. Servers commence a download by an initial burst of 40 seconds of data at the maximum available bandwidth the network can support, and apply a throttling algorithm imposing a data rate of 1.25 times of the video encoding rate. These rates are proportionally scaled down...
if needed to assure the maximum link utilization threshold $\lambda_0$.

We run the star-ring network for 200 intervals. Data rates are selected at the beginning of each interval by each scheme. The length of each interval is 6 seconds. Video requests arrive at edge servers with a Poisson arrival rate of 80 per interval. Here we focus on the occurrence of stalls while assuming that stalls are recovered only by accumulating a fixed amount of playout data in the buffer [17] before video playback resumes.

Fig. 5 shows the number of stalls that happen during the experiments and the average number of active video sessions per edge server, respectively. From the figure, we can see that our schemes produce much fewer stalls than the alternative ones, while the scheme with continuous data rates slightly outperforms the scheme with discrete data rates. The Burst & Fixed curve experiences large fluctuations due to the burst behaviors as seen in Fig. 5(a), while it attempts to finish sessions as quickly as possible, resulting in a lower average number of active sessions in Fig. 5(b). On the contrary, the Fixed-Rate scheme has no intention to quickly detach sessions. Sessions gradually accumulate as seen in Fig. 5(b), producing a high probability of stalls in Fig. 5(a).

We then track the delivery process of one session, where the client requests a 240-second video clip
of the movie “Matrix III” with a mean bit rate of 0.52 Mbps, a mean frame size of 17.1 KB, and a file size of 124 MB [38]. The results are shown in Fig. 6. From these figures, we can see that our schemes start with a higher rate and decreases the rate while the playout lead increases. The final part of the video is delivered with the minimum data rate, i.e., \( R_i \) = 128 kbps, in both cases with continuous and discrete rates. The Burst & Fixed scheme aggressively obtains a greater playout lead for the first 80 seconds, while the data rate decreases quickly thereafter due to the aggressive behaviors from new sessions in the network. No stalls occur with our schemes and the Burst & Fixed scheme. In contrast, as seen in Fig. 6(b), the Fixed-Rate scheme even results in negative playback lead well below the “Zero-Lead Line”. This Zero-Lead Line indicates that there is zero playout lead at the user buffer. As a consequence, the Fixed-Rate scheme causes many stalls and finishes the delivery at around 300 seconds after the commence of the session. Thus, it produces an aggregate stall delay of \( 300 - 240 = 60 \) seconds.

From these experiments, we conclude that Alg. 1 can intelligently adapt data rates with respect to network conditions as well as user context, thus efficiently reducing the occurrence of stalls. On the other hand, the non-cooperative behaviors of the Burst & Fixed scheme potentially produce large fluctuations in the network, while the conservative Fixed-Rate scheme neglects both network conditions and user context,
which results in the worst performance in our simulations.

VI. CONCLUSION

In this paper, we exploited the information of user context to optimize video delivery in mobile core networks which are evolving from 4G to 5G. We formulated a joint optimization problem for data-rate selection, source redirection and flow routing. We managed the stalls of video streams by maximizing the minimum playout lead using the static information of video playback curve as well as the dynamic information of the estimated amount of buffered video data and playing time. A fast algorithm with approximation guarantee was proposed for the formulated problem. Theoretical analysis and computer simulations showed the efficiency of the algorithm and significant performance gain over alternative solutions. We conclude that this work made an important contribution towards smart routing, where videos are delivered upon network conditions as well as instant playing demands.

REFERENCES


