PPNet: Privacy Protected CDN-ISP Collaboration for QoS-aware Multi-CDN Adaptive Video Streaming

KUTALMIŞ AKPINAR and KIEN A. HUA, Department of Computer Science, University of Central Florida

Software-defined networking introduces opportunities to optimize the Internet Service Provider's network and to improve client experience for the Video-on-Demand applications. Recent studies on SDN frameworks show that traffic engineering methods allow a fair share of bandwidth between adaptive video streaming clients. Additionally, ISPs can make better estimations of bandwidth and contribute to the bitrate selection for the clients. This study focuses on another aspect of network assistance in video delivery: CDN server selection. In a typical framework where the ISP contributes to the CDN selection, the video provider and the network provider interfaces are merged together. Clients connect to the ISP to get the best CDN server candidate for a given video. This exposes client requests to the ISP. However, video providers have been investing large resources for encrypted video provisioning to preserve their client's information from third parties, especially network providers. The typical approach is not practical due to privacy concerns. In this study, we present a framework called PPNet to allow CDN-ISP collaboration while preventing the ISP's access to the video request and availability information. Our framework introduces an isolation between the video provider's and the ISP's web interfaces. Clients connect to both of the interfaces and deliver information on a need-to-know basis. As a second contribution, PPNet introduces a practical optimization method for CDN selection. Real-time data collection capabilities of a typical OpenFlow network is used as the input for optimization. Congestion-awareness has been the priority. To adapt for changing network conditions, capability of utilizing multiple servers simultaneously for a single video is introduced. Instead of directing each video client into a CDN node, the proposed system performs request routing per video segment. Finally, we present a system prototype of PPNet and show that our multiple-host adaptive streaming method introduces a significant improvement in quality of experience when compared to the state of the art.

CCS Concepts: • **Information systems** → **Multimedia streaming**; • **Networks** → **Network privacy and anonymity**; *Wide area networks*; *Public Internet*; Application layer protocols; Wired access networks; • **Security and privacy** → *Privacy-preserving protocols*;

Additional Key Words and Phrases: Video-on-demand, software-defined networks, Internet, content delivery networks, privacy, request routing

ACM Reference format:
Kutalmiş Akpinar and Kien A. Hua. 2020. PPNet: Privacy Protected CDN-ISP Collaboration for QoS-aware Multi-CDN Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 16, 2, Article 49 (May 2020), 23 pages.
https://doi.org/10.1145/3379983

Authors' addresses: K. Akpınar and K. A. Hua, Department of Computer Science, University of Central Florida, Orlando, Florida, 32826; emails: kutalmis@knights.ucf.edu, kienhua@cs.ucf.edu.

1551-6857/2020/05-ART49 $15.00
https://doi.org/10.1145/3379983

1 INTRODUCTION

The world's IP traffic is expected to double from 2019 to 2022. With the demand for 4K technologies, video content is expected to become 80% of it [2]. Among the video services, Video-on-demand (VoD) has the biggest share. As examples, In 2015, Netflix, the most popular VoD provider in the U.S. alone accounted for 35% of all Internet traffic during peak operation, followed by YouTube with 15% [1]. Managing increasing VoD traffic became the major problem for Internet Service Providers (ISP) and Video Providers (VP) to solve. In this study, we examine the most recent advancements on video delivery to utilize local deployments of content distribution so that long-distance traffic can be prevented.

VoD delivery has evolved significantly over the past decade, and more evolution is expected. Video files are stored in content delivery network (CDN) servers for access from the clients. The current deployment of CDN servers are located in several consolidated locations such as data-centers [41]. ISPs deployed web caching technologies to reduce long-distance traffic. However, due to the increase in privacy concerns, video providers have adopted encryption standards (e.g., HTTPS) on video downloads. Encrypted web content deemed caching and deep packet inspection technologies unhelpful. With an increasing VoD traffic, there is a demand for local CDN deployments over the ISP's network to avoid costly infrastructure upgrades.

Although several edge hardware has been developed by the content providers, network providers aim to charge for deployment of those nodes into their network. However, content providers propose that those nodes are to reduce the load in ISP's network and for a better service to ISP's clients. The mutual benefit of deployment needs to be strengthened for CDN and ISP providers to agree on a widescale deployment. The solution to this problem is CDN-ISP collaboration systems, which allow them to exchange information to improve service quality and reduce service cost.

Another important migration in video technology is adaptive bitrate streaming. These methods allow clients to connect to web servers and adjust their video quality during the download process according to the parameters such as TCP bandwidth, screen resolution, and screen size. However, adaptive streaming may cause race conditions among clients when they share the same bandwidth in the ISP's network [5].

Privacy of the clients became an important concern for video providers. Un-encrypted HTTP traffic can reveal the client's information about the video requests to the network provider. There is a large amount of ISPs around the world and assuming a general trustworthiness is not possible. For example, a list of incidents reported in *citizenlab.ca* shows examples of ISPs around the world performing government spyware injections onto downloads from neighborhood countries, re-directing content into affiliate advertisements to make money, or mining cryptocurrencies through client web browsers. There is a risk that network providers will share the client's video usage information with the affiliated institutions and governments. Client's video usage information leaves them vulnerable in several ways. Unwanted targeted advertising is one of the concerns. More importantly, revealing political preferences can put people under additional surveillance and treatment from governments. To protect their clients, video providers invest resources for encryption. Use of HTTPS in VoD delivery has been initiated by Google on Youtube streams and other major video providers either migrated or promised to migrate into this encryption strategy. The migration is not easy. It requires large computation resources to perform encryption. Even before HTTPS, video providers used to employ encryption. Encrypted content used to be stored in video servers and decryption was performed by the video player in the client side. However, HTTPS is different. It requires encryption per client [34]. VoD providers need to go through major infrastructure changes to perform encryption in data rates [40]. Privacy is an investment for video providers.

Software Defined Networking (SDN) is expected to support ISPs' network in the near future. SDN defines a separation of the control plane from the network hardware. It makes the control plane programmable. This allows flexibility of deployment for several applications on a network without any or significant hardware modification. Moreover, networking decisions can be performed in real time to adapt for the changes in network condition or the requests from the applications. An example of SDN deployment over Wide Area Networks (WAN) can be given as Google's B4 [24]. Measurements over B4 show significant improvement in the efficiency of traffic engineering methods compared to the other state-of-the-art solutions. There also exists some platforms for research purposes such as CloudLab [36] and GENI [12].

SDN provides opportunities for CDN-ISP collaboration solutions proposed above. Recent studies show that VoD applications benefit from SDN deployments on ISP's network. Traffic engineering methods for adaptive streaming [9, 11, 26] allow a fair share between clients and reduce quality switches that originate from the competition. ISPs can make better estimations of Quality-of-Service (QoS) parameters and contribute to the bitrate selection during adaptive video requests from the clients [13, 14]. In this study, we aim to benefit from network assistance for another problem of video delivery, server selection.

Compared to CDN or video providers, ISPs have broader knowledge of network capabilities, bottlenecks, traffic from other applications, and video streams. This is called QoS-awareness. Given a list of CDN servers holding the content requested by a client, the ISP can make more accurate decisions on selecting the server that will provide the maximum TCP bandwidth. Thus, it will increase the quality level of the stream. Previously proposed CDN-ISP collaboration frameworks [13, 14, 18] reveal video requests from the clients to the SDN application servers to get information on server selection and bandwidth availability. Those SDN servers are managed by ISPs. CDNs share the database of video availability with ISPs so that ISP can route client requests to the selected CDN node or can give recommendations to the client [13]. A problem with those methods is that the video client requests are being revealed to the ISP, which is a third party. Given the privacy merits that video providers take as standard today, this is not acceptable.

A possible solution to the client privacy problem is to give control of those SDN application servers to the CDN provider, so that client video requests or the video availability information in CDN servers do not reach to the ISP. This would mean to reveal network information to the CDN provider through the northbound interface of the control plane. This will cause another privacy issue. This network information would reveal the business secrets of ISPs to other companies. ISPs would avoid such risks. Video provider companies can make use of this information to blame ISPs for undesired client experiences. In fact, the lack of video quality has caused VoD providers and ISPs to frequently blame each other in the past [37].

In this study, we propose PPNet, a framework to preserve privacy among three parties: (1) CDNs or video providers, which will be interchangeably used in rest of the article; (2) clients; and (3) ISPs. Our solution is to keep CDN servers that provide video availability information separate from the ISP servers that provide network availability. The focus of this study is mainly on CDN server selection, which involves all parties within the decision making process. Contributions of the study are listed as follows:

- **PPNet design:** A privacy-protected CDN-ISP collaboration framework design using SDN capabilities is presented. Unlike the previous proposals on network assisted request-routing using SDN, clients do not reveal the video content information to the ISP. It also does not require the ISP to reveal details about its network to CDN provider. In the proposed framework, client interface first connects to the video provider's web servers. From the client connection, the video provider detects the ISP provider that can collaborate in the decision.

Among the list of CDN nodes that can serve the requested content, the ones inside the ISP's network are chosen. The video provider gives client a short list of CDN addresses together with the ISP server interface to connect. The client makes a second connection to this ISP server. The short list of CDN addresses is sent to the ISP server to continuously query for CDN with the best network availability. Our design is compatible with the encrypted video delivery techniques. It can also accommodate other network assisted features such as bitrate adaptation and traffic engineering. To the best of our knowledge, this is the first design to achieve privacy-protection, while also providing server selection.

- **QoE optimization:** A problem formulation and solution for network-assisted CDN server selection is presented. Although several frameworks [13, 14, 18, 42] included server-selection in their conceptual designs, this is the first study to formulate and solve the problem. Our solution is based on network traffic information obtained from a practical SDN environment based on OpenFlow. Congestion avoidance in the network is considered the basis for selection. Our experiments show that this method outperforms the server selection methods that are based on the topology information or geo-location.
- **Multi-CDN video download:** PPNet enables multi-server download for a single client stream. A client video session can be long (e.g., 2 hours). Due to the changing network conditions, a CDN-selection decision may not hold valid through the session. In this study, we utilize the fact that adaptive video can be downloaded as separate segments in time. After an initial handshake for delivering possible CDN addresses to the ISP provider, the client can continuously receive updates from the server to determine CDN for the next segment. Without any disruption in the video session, the client can dynamically utilize several servers. Our experiments with random background traffic workloads show that this improves QoE of the client by a significant margin.
- **Prototyping.** A prototype of platform has been implemented using OpenFlow standard. Several CDN hosts added into the network supporting HTTPS-based (secure) web hosting. The reference implementation of MPEG-DASH, DASH.js has been modified to support multi-server downloads. To the best of our knowledge, our implementation is the first one to support CDN server selection among SDN platforms.

The remainder of this article is organized as follows. The ISP's networking environment that PPNet operates on is described in Section 2. PPNet framework design for privacy-protected network assistance is described in Section 3. Formulation and solution for QoS-aware CDN server selection problem is provided in Section 4. System prototype implementation is detailed in Section 5. Experimental results for multi-CDN video delivery are presented in Section 6. A summary of related studies in literature is provided in Section 7. Finally, a summary of the study is provided in Section 8.

2 OPERATING ENVIRONMENT

In this section, the operating environment for PPNet is explained. This includes status quo of CDNs, local CDN deployment opportunities on the ISP's network, the ISP's infrastructure and SDN deployment options in the ISP's network.

There are several CDN solutions that VoD providers use today. Some of them use their own servers for a CDN-like capability. For example, YouTube is known for using the same company's server infrastructure for hosting videos. As of today, an experiment we perform over IP connections reveals that download of a very popular YouTube video segment still leads to a large datacenter in Mountain View, California, while the client connection is from Florida, 4,500 km away from the server. However, some VoD providers choose to employ third-party CDN providers to

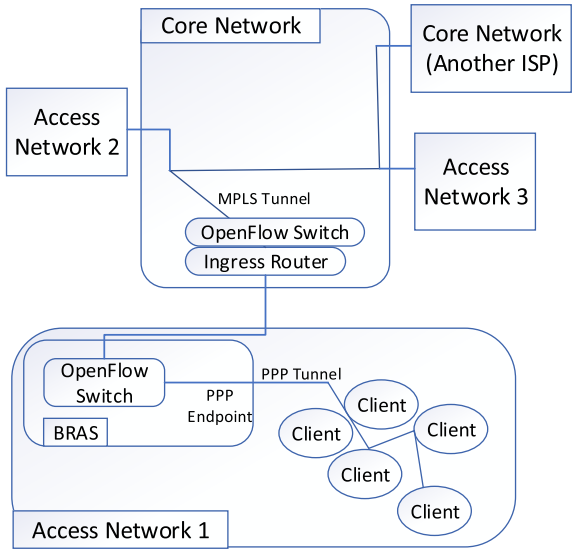


Fig. 1. A typical ISP's network is layered as a core network and many access networks.

host their content. For example, the probing of Netflix traffic [4] shows that they rely on well-known CDN providers such as Akamai and Limelight. Today's CDN servers are also located in a few large data-centers [41]. Internet Exchange Points (IXP), which are limited in number, are popular hosting locations for those.

Due to the aforementioned reasons of increasing video traffic and encrypted video content, consolidation of VoD hosts into data-centers causes challenges for ISPs and content providers. Edge CDN deployments are initiated to address the issue. In edge solutions, computation and storage nodes from the video provider are located in ISP's network, while the video provider has the complete control over the node except its network. Google Global Cache and Netflix Open Connect[1] are examples of those. In this study, our main focus will be this type of edge deployments of CDN.

A typical ISP is layered as a Core Network and several Access Networks as in Figure 1. An access network receives traffic from clients as Point to Point Protocol (PPP), which allows validation of subscriptions from the customers. It connects into the core network through Broadband Remote Access Servers (BRAS). BRAS terminates the PPP protocol and sends it to the Ingress Router located in the core network. The Ingress Router calculates the path for the package and wraps it according to Multi-Protocol-Label-Switching Protocol (MPLS), which accelerates long-distance traffic. The core network operates in MPLS and is also connected to other ISPs [42].

Within the ISP's network, edge CDN nodes are expected to be located at critical points of the core network. One candidate location is alongside the gateway to other ISPs. This allows the ISP to minimize traffic going outside of its own network. Another candidate is alongside BRASes of the most populated access networks. This node will prevent a large amount of traffic from going into the core network. Decisions can be made according to the distribution of access networks. We assume CDN nodes will be in proximity of the populated regions. This is likely to be the case for both options.

[1] https://openconnect.netflix.com/en/.

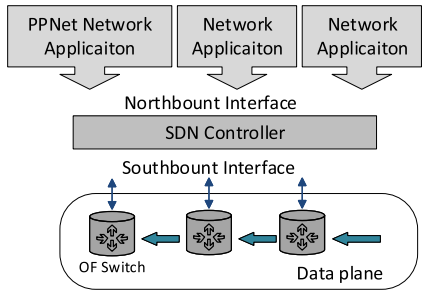


Fig. 2. Multiple networking applications can be installed on an SDN network without changing hardware. PPNet is designed to be one of them.

Among SDN technologies, OpenFlow has been the predominant open-source standard. An OpenFlow controller is a software used to constitute the control plane of the network. It interacts with multiple switches and other applications to perform decisions on network rules. An OpenFlow switch is a network element to perform a table of rules that are given by the controller.

For SDN support, OpenFlow capable hardware can be deployed over the core network and/or access network [42]. Starting from standard 1.1, OpenFlow supports management of MPLS traffic [20]. This allows deployments into the core network without replacing well-established protocols. In this study, we assume that the controller can stream network flow statistics from the switches in core network. Alternatively, the flow information in core network can be predicted from the access network switches.

An advantage of SDN infrastructure is, deployment of a networking technology does not require replacement of hardware. Just like applications installed in an operating system, networking methods can be deployed into the network by updating the controller software. These are also called networking applications (Figure 2). PPNet is designed to be a networking application. Controller software communicates with each application using the *northbound interface*. Networking decisions that do not contradict with other applications are imposed from the controller to switches using the *southbound interface*.

3 PRIVACY PROTECTED CDN-ISP COLLABORATION

CDN-ISP collaboration frameworks allow application-aware networking decisions such as routing and bandwidth throttling. They also allow use of QoS parameters on video streaming decisions such as server and bitrate selection. In this study, we focus on server selection. The aim of server selection is to pick an available CDN node inside the network that will provide the best QoE for the user. It is also called request routing. The first thing to consider for server selection is, which servers have the content that is being requested. The second thing to consider is which server will provide the best QoE. In the case of VoD applications, QoE increases with the prolonged TCP bandwidth availability.

This section provides a system design that implements QoS-aware CDN selection while preserving privacy information of the client. In addition to the CDN selection, other application-aware networking decisions can also be implemented on top of the proposed system to utilize bitrate information streamed from the client. The details of QoS-aware CDN selection algorithm is further provided in Section 4.

Although their focus of implementation was on other aspects of CDN-ISP collaboration, several frameworks include server selection in their conceptual design. Some solutions consider local CDN deployments as a cache service provided by the ISP. Those studies assume that CDN will share all

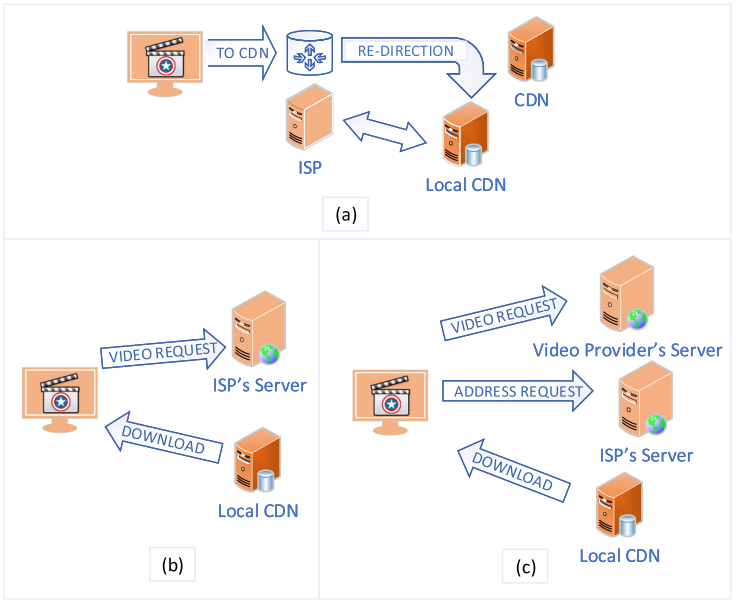


Fig. 3. Comparison of CDN-ISP collaboration frameworks. In (a) [14, 18], database of content availability is stored by the ISP provider so that ISP can re-route traffic into the desired nodes even without the client's knowledge. In (b) [13], a web server by the ISP responds to the client request so that it connects to the server address destined by the ISP. (c) PPNet: Client connects into both interfaces at the same time. While CDN interface provides a server listing for each segment and quality level for the requested video, this information is delivered to the ISP's interface without giving information about the video itself.

requests that are being made to the CDN node or the request information will be retrieved from un-encrypted traffic [14, 18]. This is shown in Figure 3(a). A database of content availability is also stored by the ISP provider so that the ISP can re-route traffic into the desired nodes even without client's knowledge. The client connects through the same address, but the server selection will be handled by the ISP. However, encrypted HTTPS connections that video providers require today cannot be re-directed due to the server certificates. In HTTPS, each server has its own certificate. The client connects a certain address and the machine associated with it through certification authorities.

Other solutions [13, 42] allow a web server by the ISP to respond to client requests so that client connects to the server address destined by the ISP (Figure 3(b)). Video availability information is again, stored in ISP's servers, and client file requests go through this server. Although this does not violate the HTTPS connections, it violates the principle behind it. Hiding client's information from the third parties such as the network provider is the fundamental reason for video providers to invest in encryption. Thus, those solutions are unlikely to be accepted by video providers.

This study focuses on the aspect of client privacy in server selection, which has not been addressed in previous studies. A possible solution to the client privacy problem is to give control of those SDN application servers to the CDN provider, so that client video requests or the video availability information in CDN servers do not reach to the ISP. This requires network information to be revealed to the CDN provider through the northbound interface of control plane. This will cause another privacy issue. This network information reveals business secrets of ISPs to other companies. ISPs want to avoid such risks. Video provider companies can make use of this information to blame ISPs for undesired client experiences [37].

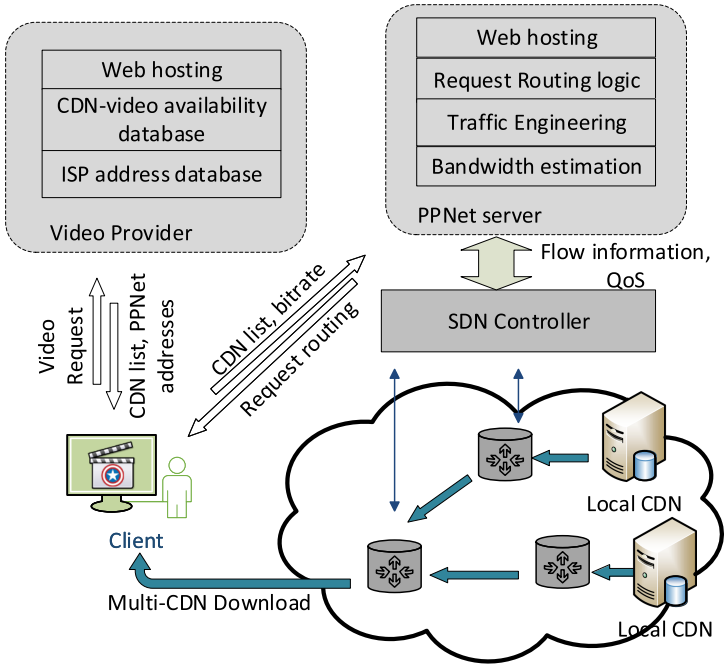


Fig. 4. Elements of video delivery in PPNet framework.

In this study, we propose an approach to preserve privacy among three parties of users: CDNs or video providers, clients, and ISP interfaces. Our solution is to keep the CDN servers that provide video availability information separate from the ISP servers that provide network availability. The client connects into both interfaces at the same time (Figure 3(c)), while the CDN interface provides a server listing for each segment and quality level for the requested video. This information is delivered to the ISP interface without giving information about the video itself.

The active components of video delivery in PPNet framework are listed as follows: (1) SDN infrastructure with controller and switches located in the ISP's network, (2) local CDN nodes located in the ISP's network, (3) PPNet server, and (4) the video provider's server. The framework is shown in Figure 4.

The video provider hosts a web server to respond to client requests. The server holds a mapping from possible client IP address ranges to the ISP providers. This list can be dynamically obtained and updated from the ISPs that have an agreement with the provider. The server also holds the database of videos and quality levels available in each CDN node. Each CDN node in the database has a mapping for the ISP provider that hosts the CDN. The video availability in each CDN is managed by the cache replacement algorithms that the CDN employs. The CDN nodes are managed by video providers, while their network is managed by the ISP.

The PPNet server is managed by the ISP. To optimize its network and provide a better service, the PPNet server interacts with the client and SDN controller. For client connections, the server hosts web services that are accessible from the client interface. Through this connection, the client sends a list of CDN candidates that contains the video. As a response, the client expects decisions on CDN addresses to connect. At the same time, the PPNet server streams QoS metrics from the SDN controller to perform decisions on CDN selection. Those metrics include bandwidth usage, jitter and package loss. Bandwidth availability is estimated via those parameters and the best available CDN

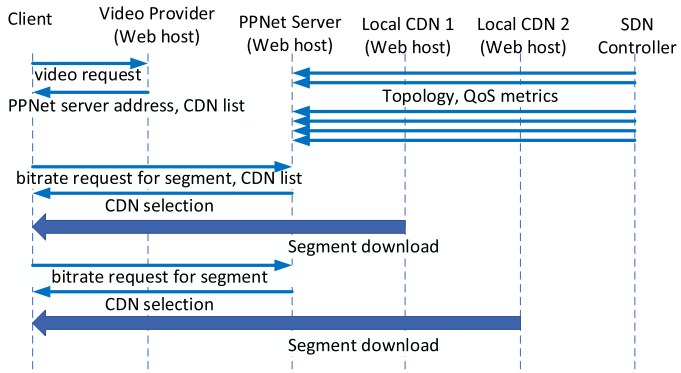


Fig. 5. The communications protocol to support the privacy-protected video streaming.

is selected. Additionally, the bitrate requests from the client are used for the traffic engineering tasks [10, 26] such as setting QoS parameters in the network for fairness between clients.

The communications to support the privacy-protected video streaming is shown in Figure 5. The application communications protocol starts with client's connection through the video provider's web interface (e.g., HTTPS-enabled website to watch a video). The server analyses the client connection and if there exists an ISP that hosts the service for the given client, the video provider suggests a PPNet server address to the client. A set of local CDN connections containing the video is also reported to the client.

As for the second step, the client connects to the PPNet server for CDN selection. This is performed for each video segment. The PPNet server evaluates the client's address and locates it inside its own network. Additionally, the PPNet server locates the list of CDN servers reported by the client. QoS availability from client to each CDN server is evaluated momentarily and selection is reported to the client. After receiving a reply from the PPNet server, the client downloads the segment from the Local CDN server indicated by the PPNet. Optionally, the client also delivers the intended segment bitrate levels to be requested. This information can be used by PPNet on aforementioned traffic engineering tasks to optimize network for providing the requested quality.

The network topology updates and QoS information is periodically streamed from the SDN controller to the PPNet server so that the server can make instantaneous decisions. This is important for the proposed system, since PPNet performs routing decisions for each video segment, rather than for the entire video. The SDN controller can retrieve bandwidth usage and package loss rate statistics directly from the switches, since they are part of the OpenFlow standard [20]. There are other SDN methods proposed for obtaining additional metrics such as jitter [8, 38]. In our implementation, we will be using the available metrics in OpenFlow.

For the traffic engineering tasks [10, 26], the PPNet server can keep a list of active streams within the network using client connections. It decides QoS features to be applied. This information is delivered to the SDN controller to be converted into rules and to be sent into the switches.

For privacy of the client, implementation of the video provider's web interface is important. For a typical web interface, scripts to connect the PPNet server is downloaded from the video provider. This gives control of the connection type to the video provider. They can restrict the PPNet server connection to only a certain set of communication messages. The client does not download any content from the PPNet server and does not send any information that the PPNet server does not need to know. The PPNet server interface can be implemented using simple web service standards (e.g., WebSockets). Although security risks are lower compared to web interfaces,

a similar challenge exists in video streaming applications such as TV or mobile apps. PPNet server communication will be limited to CDN addresses and bitrate information.

Another important design detail on the implementation is to avoid blocking download requests and playbacks because of CDN selection requests. This is important to preserve QoE of the user. This can be achieved by initiating the playback from a default CDN server. Whenever a CDN request is made, the following segment download action does not wait for the response. It proceeds from the latest received CDN suggestion or from the default CDN if none is available. The CDN suggestion effects the segment download that comes after the suggestion's arrival.

4 QOS-AWARE MULTI-CDN REQUEST ROUTING

Multi-CDN request routing gives clients the ability to adapt to changing network conditions and utilizes multiple CDN servers. While the traditional assumption for adaptive video streaming suggests that switching between servers may affect network performance in a negative way, this is not always true. Unlike the caching solutions that unexpectedly change the path and effects the adaptive bitrate performance [29], intelligent techniques that change the network path to avoid congestion against changing network conditions can enhance the user experience.

In this section, we formulate the QoS-aware request routing problem for adaptive video streaming and represent the solution with the capability to utilize multiple CDN nodes at the same time.

4.1 Problem Definition

The request routing problem is briefly defined as such: Given the network topology, real-time flow information from each link, the client in the network, a list of possible CDN nodes in the network, and bitrate request from the client, find the CDN node to provide the best QoE for the client.

The network topology is defined as a graph $G = (V, E, B, P)$. Vertices set V denotes the set of switches and endpoints. The edges $E \subseteq V \times V$ denote a directed set of links between. Each individual link e holds a bandwidth capacity $b_e \in B$. In the ISP's network, the topology being simulated is generically the core network. P denotes the paths. For each pair of vertices (e_i, e_j), the path is a pre-designated set of connected edges that packets travel through. The path is not determined by the PPNet logic. Thus, it is received as the routing rules delivered from SDN controller to the PPNet server.

The QoE metrics collected from switches through the controller are port statistics of each vertices. Each port in a switch denotes an edge, so the corresponding statistics are attributed to the links in the network. Statistics are received for each periodic time instant. Each reading at time t and edge e is a triple (c_{te}, d_{te}, l_{te}), where c_{te} is the total package count, d_{te} is the total data size received (or transmitted if outgoing edge), and l_{te} is the total number of packets lost until time t for edge e.

The QoE of a client can be modeled as the quality of the video. This infers to the bitrate selection. For adaptive video streaming, the selection is performed dynamically according to the download speed of each previous segment [39]. In network side, this refers to the TCP bandwidth between the vertices that represent the client and CDN. This can be formulated with the Mathis equation [30]:

$$Tbw = \frac{Tw}{Rtt} \times \frac{C}{\sqrt{L}}, \tag{1}$$

where Tw is TCP window size, Rtt is round-trip time, C is the link capacity, and L is the packet loss rate. In this formulation, TCP window size can be perceived as a constant and is not controlled by the network provider. Round-trip time and the packet loss rate would relate to the path. In an ideal network where fiber optic hardware is used in infrastructure with a much larger link capacity than the demand, packet loss rate and round-trip time are good enough to provide any

quality to the client. However, the existence of congestion creates jitter and increases the packet loss rate substantially. In our formulation, we will focus on estimating and minimizing network bottlenecks to avoid this situation.

Bottleneck bandwidth (Bbw) for each CDN–client pair (associated to vertices a and b) at a time t is estimated as follows:

$$Bbw(a, b, t) = \min_{e \in p_{a,b}} (b_e - u_{et}), \tag{2}$$

where $p_{a,b} \in P$ is the path between a and b, u_{et} is the bandwidth usage for the link e at a time t, and $b_e \in B$ is the capacity of the link e. In this equation, while the rest of parameters are given in topology information, u_{et} is computed as follows:

$$u_{et} = \frac{d_{et} - d_{e(t-\Delta t)}}{\Delta t}. \tag{3}$$

Given a set of CDN candidates connected to vertices V_{CDN}, and client connected to vertice a, CDN selection i maximizes the bottleneck bandwidth:

$$i = \arg\max_{b \in V_{CDN}} \left(\min_{e \in p_{a,b}} \left(b_e - \frac{d_{et} - d_{e(t-\Delta t)}}{\Delta t} \right) \right). \tag{4}$$

We emphasize that flow statistics also include packet loss rate l_{te}. This can also be used as a metric for bottlenecks; however, in a fiber-optics environment, this value is expected to show significant values after a serious congestion occurs. It will be zero most of the time.

The proposed formulation above can be implemented on Openflow switches in a passive and dynamic way. The method is passive, since it does not introduce any additional traffic into the network. It is dynamic, because it performs estimation in real time, according to traffic statistics collected from the network. Another option is to employ active bandwidth estimation methods that are proposed for OpenFlow networks [8, 38].

4.2 CDN Selection Algorithm

CDN selection algorithm aims to solve the maximization problem in Equation (4) for a given CDN set. A solution to this problem is to loop through each CDN and their path to the client as the equation suggests. We will follow this solution, because both the number of CDN deployments and the number of vertices involved (e.g., number of OpenFlow switch deployments) are expected to be small in terms of numbers and therefore computation is efficient. The reason for estimated low count is the cost of SDN switch and server deployments. The algorithm pseudo code is presented in Algorithm 1.

An important design detail is to pre-compute the bandwidth usage statistics, u_{et}, for each time period. This can be performed in SDN controller or PPNet server. If we consider the sampling rate of flow statistics as δt, then for an accurate estimation, $\Delta t \geq \delta t$. It is also preferred that Δt is integer multiple of δt. For each link, algorithm holds a short vector of measurements with length $\Delta t / \delta t$. This table is shifted and updated with the most recent data in each measurement. Additionally, bandwidth usage for the edge (Equation (3)) is computed from each vector.

Δt should be selected relative to the expected video segment size. If Δt is too small, then over-sampling of network statistics causes too many fluctuations. We emphasize that a typical video segment in adaptive video streaming application is between 2 and 10 s. So selecting Δt smaller than this will not improve system utilization. However, selecting it too large (e.g., 10–20 times the estimated segment size) is also not preferred, because it will result in a delay in flow estimation.

The CDN selection algorithm also scales in terms of number of clients. The switch-to-switch cost metrics are stored in a table and can be used for all clients and maybe even by other networking applications. It is updated as statistics are received by the controller. Each client require a service

ALGORITHM 1: CDN selection algorithm

Input:
P: path for each vertice pair.
V_{CDN}: CDN set received from the client.
U: Current bandwidth usage for each link.
Output:
max_bw_cdn

```
max_bw = 0
for all b in V_CDN do
    min_bw = INF
    for all e in P(b, a) do
        min_bw = min(min_bw, U_e)
    end for
    if max_bw < min_bw then
        max_bw = min_bw, max_bw_cdn = b
    end if
end for
```

connection to query their own metrics. Since table size is limited by the number of switches, query will be fast and efficient.

5 SYSTEM PROTOTYPE

To demonstrate the concept of privacy protection and multi-CDN video delivery, a system prototype is prepared using machine and network virtualization environments. Using the prototype, QoE metrics are measured and compared against state of the art to prove effectiveness of the study.

An overview of the setup is shown in Figure 7. The setup aims to emulate the environment described in Section 3, Figure 4. The setup is built on a machine with four-core Intel I7 CPU. VMware Workstation 14 is used as the main virtualization environment for setting up a network with several virtual machines (VM) and inter-connections between. Mininet[2] is one of the VMs. Mininet is another virtualization environment, and it allows us to emulate OpenFlow switches in the network.

5.1 Topology Generation

To determine effectiveness of the framework in the ISP's environment, it is important to chose a realistic topology. For this purpose, a publicly available dataset[3] of ISP topologies from the Topology Zoo project [27] is used. However, many of the topologies listed in the database are either continent-wide or composed of a few nodes only. We have focused on topologies in a regional scale, since this is where local CDN deployments are anticipated. We also aimed at topologies where metropolitan area locations are clear so that a realistic traffic intensity can also be implemented in the later steps. We selected the "Sago" dataset, which covers the Florida and Atlanta region in three branches. It contains 18 nodes and 17 edges. The topology is shown in Figure 6.

A Mininet script using python API is used for reading and creating the topology from the graphml files provided in the dataset. This is called the *Topology Generator*. Topology is considered as the core network of the ISP. An OpenFlow switch is attached at each node (18 nodes) within

[2] http://mininet.org/.
[3] www.topology-zoo.org/.

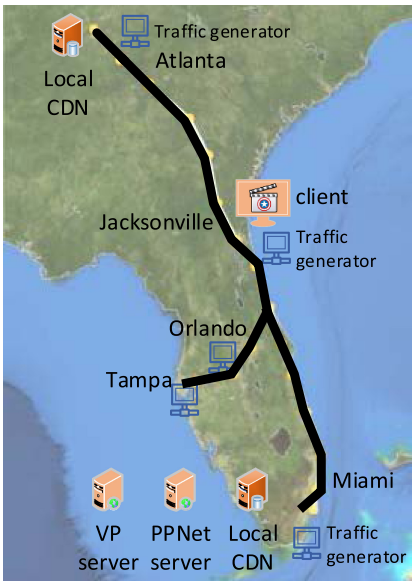


Fig. 6. *Sago* topology is used to present a local ISP located in the Florida and Atlanta region. Traffic generators are placed in five metropolitan areas with a population count of over 1 million. CDN servers are placed in the largest two cities.

the core network. Each of the switches are connected to the controller being served from the host machine. An IP address and Ethernet port accessible exclusively from the Mininet host is used for this connection so that controller interfaces are separate from the data plane being controlled.

The topology generator also handles connections to the data plane. These connections consist of video clients, the CDN server, the PPNet server, the video provider's server and other hosts for traffic generation. Video clients and CDN servers are implemented as seperate VMs within VMware. These are attached to the specific interfaces of Mininet VM so that topology generator can attach them into the OpenFlow network. A Mininet virtual host is also attached into each switch using Mininet's features (Figure 7). Those virtual hosts are used for traffic generation and connectivity testing purposes. The PPNet and video provider web servers are being served from the host machine. However, unlike the controller, the web servers are attached to the data plane (the switch in Miami). This is to preserve its feature as a client to the network. Those two are also served from web addresses using different ports so that they act as separate servers.

5.2 The Controller

The controller is implemented using POX,[4] a python-based OpenFlow controller. The controller is a program with a set of event handlers. When a connection from a switch occurs, the controller saves the connection ID into the database. This ID is an identification for the switch. As an action, the controller sends a message to the switches to prevent default behavior of forwarding to all ports. Otherwise, switching loops can occur. Our prototype implements our own forwarding algorithm using the shortest-path.

We used *discovery module* of POX to detect links between the switches. Each *link discovered* event triggers a topology update in the controller. In our case the number of links to be discovered

[4]https://github.com/noxrepo/pox.

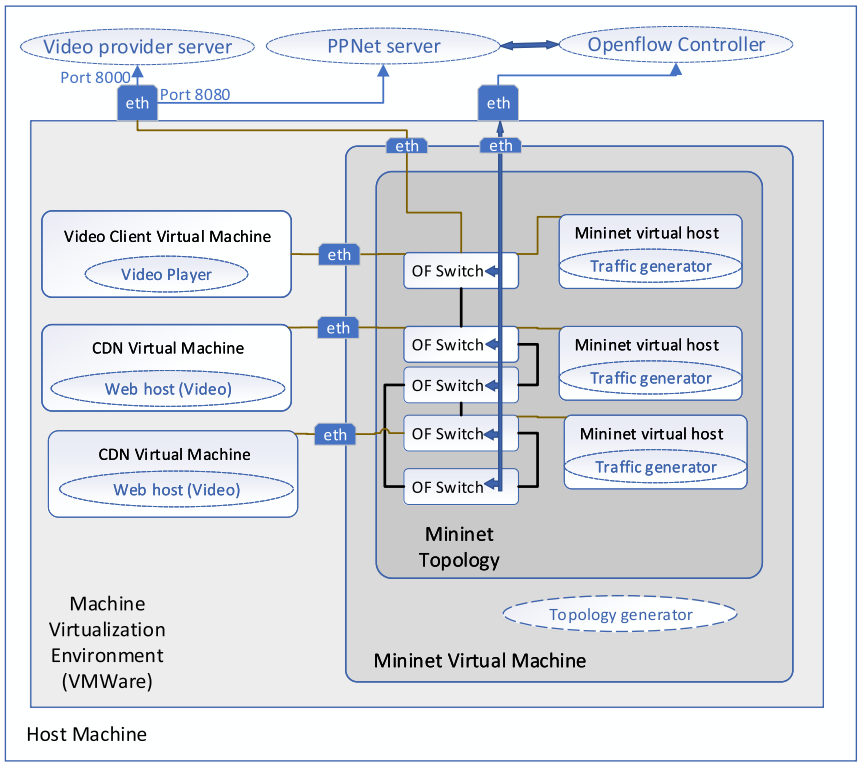


Fig. 7. System Prototype implementation using VMware and Mininet.

are known, so we implement forwarding rules only after all links are discovered. In a generic case, a time frame is introduced for discovery and forwarding rules are updated if a connection occurs after this period.

After the link discovery, addresses of the network elements are still unknown. Those are perceived when a *packet-in* event occurs. For each event, the controller records the IP address and the port associated with it so the topology is complete. For each IP connection, the controller also calculates the shortest path and sets forwarding rules on all switches. If there is an ARP request, then it also sends a response from the switch so that switches perform the duty of a router.

Another task of the controller is to retrieve traffic information from the network. This is performed by periodically sending a *port statistics* request to each switch. An interval of 2 s is used for the implementation. Feedback from each request triggers an event to record the latest status in the network. This information is forwarded into the PPNet server in a later step.

5.3 Web Interfaces: Video Provider, Client, PPNet, and CDN

The video provider server hosts an HTTPS website using *node.js*. The website simply includes an HTML page to play a video. The video player is built on top of reference MPEG-DASH client implementation, *DASH.js*.[5] This is a primary open-source standard for adaptive video streaming. *Socket.io*, a WebSocket protocol implementation for browsers is used for additional message passing.

[5]https://github.com/Dash-Industry-Forum/dash.js/wiki.

The video provider server holds a database of CDN addresses per ISPs. Client performs queries over those addresses using the WebSocket interface of the video provider. After receiving the PPNet address from the video provider, it creates the second Websocket connection. This time, it connects to the PPNet server. *DASH.js* includes an adaptive bitrate selection and video buffering mechanism for client to download each segment from the same address as the video provider. We modify this reference implementation to download the video segments from the CDN address provided by the PPNet. After bitrate selection, the PPNet interface is queried for the CDN selection. URL of each segment is replaced with the corresponding URL in the selected CDN node. This is performed by modifying the domain name, while keeping the path extension in URL.

Just like the video provider's server, the PPNet server is implemented using *node.js* and *socket.io*. The connection is hosted from a separate port address to emulate servers with different addresses. Video provider accepts two types of connections from the socket.io interface: client and the controller. The controller connects into the server using a python implementation of *socket.io* client.[6] It streams the real-time statistical information on network ports in the network. It also delivers IP address connections to each switch so that PPNet can identify the switch associated with each CDN and client. After identification, the matching problem between the CDNs and the client becomes a matching problem between the OpenFlow switches.

The CDN servers host the video content in an MPEG-DASH format. Samples of videos are downloaded from DASHDataset2014.[7] *Node.js* is also used for CDN servers; however, it hosts only a static directory of video segments.

For HTTPS, self-signed certificates are generated using OpenSSL. A public key is inserted into the associated web browsers. Client connections are performed from a Linux browser based on Firefox.[8] Additional permissions are added into the web servers using Cross-Origin Resource Sharing (CORS) mechanism. This mechanism prevents websites from connecting to content in another domain. Adding permission to connect CDN addresses is required for video providers. In our case, the PPNet server addresses also need to be given permission. This does not create a vulnerability on security and privacy.

6 EXPERIMENTAL STUDY

In this section, experiments under different workloads are detailed to show the performance of multi-CDN video delivery. Our experiments are divided into three sets. The first setup validates the functionality of the multi-CDN video delivery method using a controlled background traffic in the network. The second set of experiments shows the performance improvements under various workloads of the network using randomized traffic elements and repeated experiments. The third setup shows that additional traffic introduced by privacy protected communication is insignificant.

In all experiments, the CDN servers are located in the two most crowded metropolitan cities of the Atlanta-Florida region that the topology represents. These are Miami and Atlanta. These are also the most popular Internet Exchange Point locations[9] in the region. The client is located in Jacksonville, another populated metropolitan area in the region. It is also a critical location to perform the experiment, since it can connect to both of the CDN servers using different paths. The distances from the Jacksonville client to the CDN nodes are not equal. The Atlanta server is closer both by geo-location and the number of hubs in the network. So it provides an appropriate comparison opportunity with the traditional geo-location or shortest-path techniques that are not QoS-aware.

[6] https://pypi.org/project/socketIO-client/.
[7] http://www-itec.uni-klu.ac.at/dash/?page_id=6.
[8] https://www.mozilla.org/en-US/firefox/.
[9] https://en.wikipedia.org/wiki/List_of_Internet_exchange_points.

The comparison is performed against two different server selection approaches. The first one is the method of single selection per request. This approach performs QoS-aware server selection decision at the beginning of the stream, but it does not change it through the stream. Although no QoS-aware SDN-based server selection algorithm implementation is found by us in literature, this alternative method is parallel to the description in the previously proposed architectures [13, 18, 32].

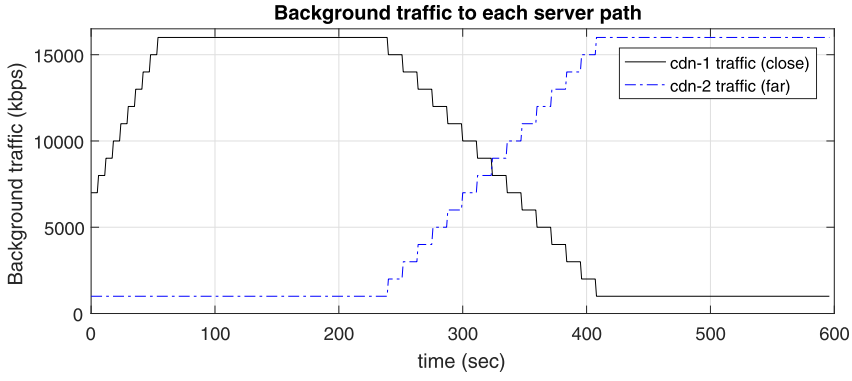
Another method of comparison is based on the network distance or geo-location. In the network distance approach, a static network delay information between the access points is received from the network provider. Server with the shortest path is chosen [28]. In the geo-location approach, which is the most traditional method [28], the server with the closest Earth coordinates is chosen. Coincidentally, in our topology, those approaches are the same. When the network delay is modeled proportional to the number of hubs, they both connect the Jacksonville client to the Atlanta server.

The background traffic is introduced using the additional nodes attached to the network. As described in the previous section, each switch is attached with one of these. Iperf tools running from these nodes are used for generating the traffic on the links between the client and the server. UDP streams are used for generating consistent amount of network traffic over a period. Under normal circumstances, the ISP's network serves an amount of data that is not possible to emulate in a cost-efficient environment. We limit the network bandwidth in each connection and scale the background traffic accordingly to emulate the bandwidth that can be reserved for a video stream, rather than the entire capacity.

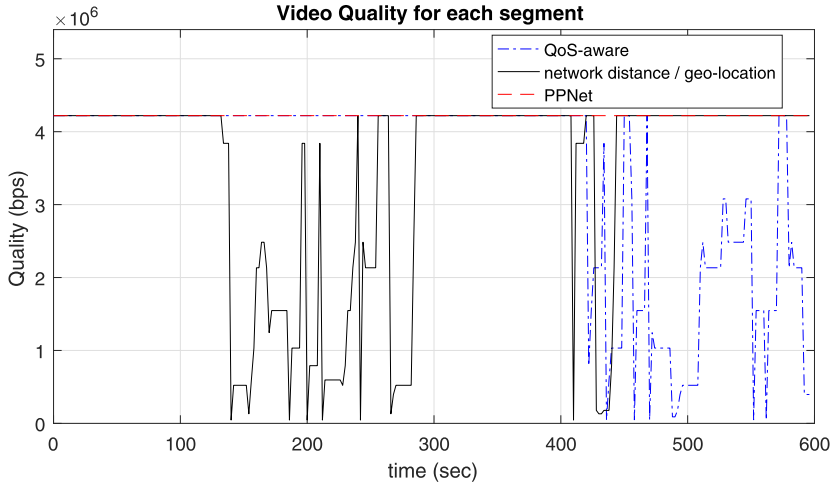
The first set of experiments demonstrate the functionality of multi-CDN video request routing in a controlled environment. In this experiment, two different paths going from client to the CDN servers are congested one by one. As a result, we expect the client to connect to the CDN with the least-populated path. The maximum congestion introduced is 16 Mbps, whereas total bandwidth is 20 Mbps.

The background traffic used in the experiment is shown in Figure 8(a). The setup is initiated with an increasing traffic on the links between the client and the nearest CDN server. During the second half of the 10-minute video session, this traffic is reduced and another traffic is introduced between the server in the far away location and the client. During the video session, we expect our framework to utilize the far CDN server in the first half and the nearby CDN server in the second half. Figure 8(b) shows video qualities of each segment transferred for different methods. The geo-location method utilizes the nearby server. In this method, we can observe the frequent quality switches in the first half when the network gets congested on that side. However, single-selection method performs QoS-aware routing at the beginning of the video. This allows it to detect congested traffic on nearby server, and it connects to the far one. However, it cannot adapt to the introduced change in the network, and its performance is degraded in the second half of the experiment. PPNet utilizes both servers in different times, and, thus, it is effected less from the oscillation and congestion over the network. The client can stream the highest quality video with the least amount of performance degrades.

The second set of experiments are performed to prove the performance of multi-CDN QoS-aware video delivery in an un-controlled, randomized background traffic. They show performance improvements in a more realistic scenario. In this experiment, many alternative streams in the network are generated of the same length as the test video, which is 10 minutes. The alternative streams may arrive every minute through a 4-hour session with a Poisson distribution. The same random seed is used for each different method. Each stream occupies 1000-kbps bandwidth using UDP streams. They can select one of the paths between CDN and the client to create a bandwidth consumption. Our first CDN client that uses the PPNet framework arrives after 10 minutes to avoid



(a)



(b)

Fig. 8. A 10-minute demo experiment via changing network conditions. Using Sago topology, two CDN servers are placed in far (Miami) and close (Atlanta) locations of the network. (a) Shows background traffic generated over two paths of the network. (b) Shows client video quality when different CDN-selection methods are chosen. The QoS-aware method selects far server due to less traffic. The network distance method (results same as geo-location method in this topology) selects closest server in network. PPNet provides the best quality by dynamically changing servers during streaming.

the transient state of the generated background traffic. For each network load and experimentation method (PPNet, geo-location, and QoS-aware), the client streams the video 20 times within the 4-hour window. If the 600-s video stream cannot be completed within 700 s, then the stream is forcefully terminated. This has been observed only in most congested network traffic configurations. Quality of the streams, the number of bitrate switches and the amount of video that has been streamed are recorded for each experiment.

Results of the experiments are shown in Figure 9. In each graph, we can observe the intensity of the total background traffic within the network. The unit of measurement is the average number of simultaneous streams on the background traffic in the network at any given time, so higher numbers in the horizontal axis indicate more network bottlenecks. The first graph, Figure 9(a),

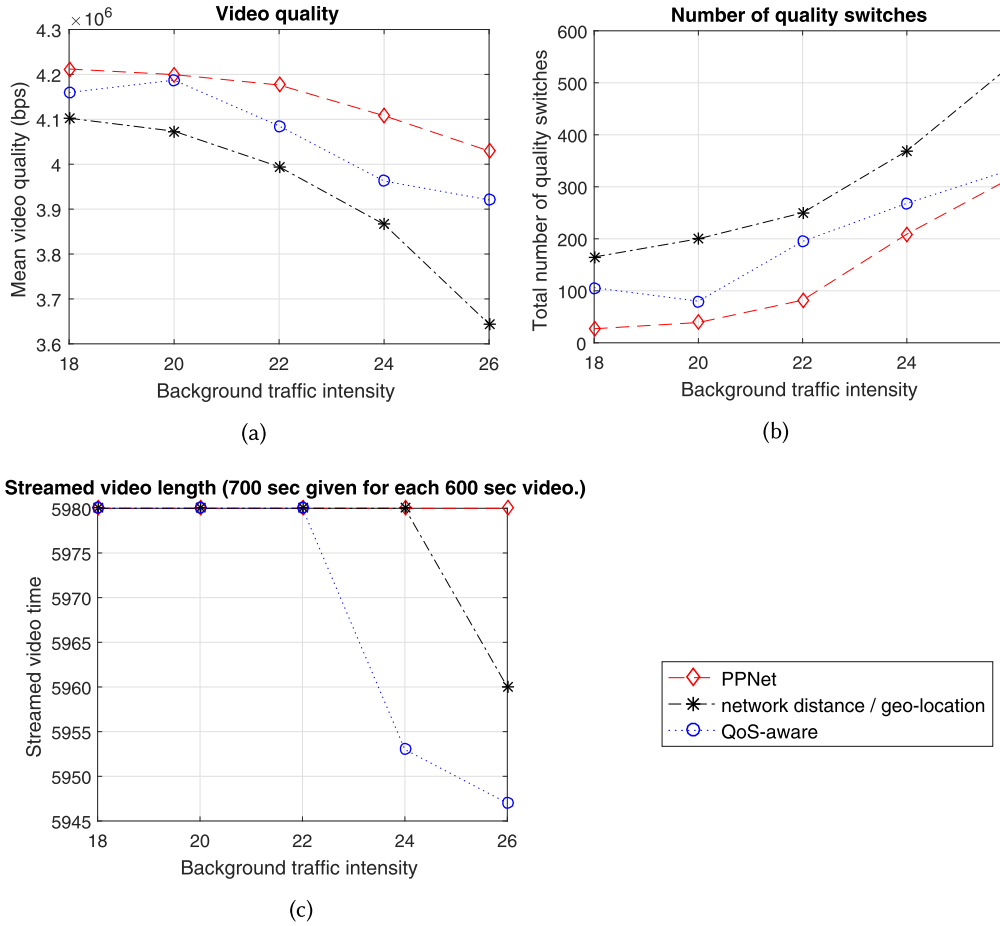


Fig. 9. Repeated experiments using randomized background traffic. A 10-minute video is repeated 20 times with different traffic distributions for each measurement point. (a) Shows the average observed video quality on client's side. Large indicates better quality. (b) Shows total number of quality switches through the experiment session. Fewer quality switches indicate better quality. (c) Shows streamed video length when 700 s is given for each video to complete.

shows the observed average quality levels. Since the same bitrate adaptation method is used in all of the experiments, this can be considered as the most important QoE parameter that can be observed. The maximum is 4.2 Mbps, which is the highest quality setting that servers offer. For all background traffic intensities, PPNet outperforms the alternative approaches in a clear margin. The second graph, Figure 9(b), shows the number of bitrate switches for the client. For this parameter, less is considered better, since it provides a more stable experience for the user. We can observe that our method is lowest under all settings. The third graph, Figure 9(c), shows the number of completed segment downloads within a given time. This value makes a difference when the traffic congestion is high and streams cannot complete due to the freezing of the video. The graph shows that PPNet streams have a higher chance of completing and less number of freezes.

From the graphs, we can also observe that a broad range of background traffic is studied. The average quality levels in Figure 9(a) show that PPNet streams at 4.2 Mbps in lowest background

Table 1. Comparison of Delay and Total Traffic with and without the Added Communication Messages

	Simple Video Server	PPNet and Video server	Difference
Incoming traffic	310,282.36 KB	310,267.20 KB	15,16 KB
Mean download request time	266,33 s	266,34 s	~10 ms

traffic. Since this is the maximum quality level served, it is understood that network congestion is unimportant for those streams. However, from Figure 9(c), we can observe that some streams cannot finish downloading the entire video. This happens even if 700 s is given for each 600-s video stream. This is an extreme case of congestion.

In the third set of experiments, it is shown that additional exchange of communication messages introduced by the privacy-protection communications protocol (Figure 5) does not have an observable effect. In this experiment, we compare two configurations: One is a simple video server located in Miami, and the other one is a video server supported by PPNet. PPNet is configured to select only the server in Miami. The only difference between those two configurations will be the additional exchange of messages to request the CDN selection from PPNet. As for the adaptive video segment size, 2 s is used in this experiment. This is fairly small to observe additional traffic per segment.

The first metric we have observed is the total amount of traffic for the client connections. For each video session, traffic has been observed via web browser's monitoring interface. The amount of traffic for a single session is listed in the first row of Table 1. As can be seen from the table, the message passing for each segment between PPNet and the client introduces about 15 KB of traffic. This amount is negligable compared to the video traffic, which is 310 MB.

Another observation is the possible delay in the video experience due to the additional communication messages. In this step, we would like to re-emphasize that the additional communication with PPNet is implemented in a way that it never blocks a video experience action such as download request or playback. When a CDN selection request is sent to the server, the client may start the download from the default or the latest recommended CDN without waiting for the response. When the response arrives, it can be applied to the next segment download. Thus, we are not expecting any significant delay.

Ten simulations are performed for each configuration. Download request times from the client script have been observed with respect to the page loading times. The average arrival times are given in the second row of Table 1. We have observed that the standard video interface connects 10 ms faster in average. This value is very low to effect the QoE. We also would like to indicate that this average value is below the precision of our measurement environment, which is 20 ms for JavaScript executed by Firefox. To further show that arrival times are similar in both cases, we have marked all arrival times from 20 simulations in Figure 10. In this figure, we further shrink the observation window to three segment downloads (between seconds 250 and 256). It is shown that arrival times of download requests are similar in both cases. We can also observe that the deviation of arrival times is much larger than 10 ms, which makes the value insignificant.

7 RELATED WORK

In recent years, several researchers have proposed SDN as a promising platform to address the congestion and QoE problems in adaptive bitrate streaming.

Some of those studies perform traffic engineering to control the QoS parameters and thus, to increase the streaming performance. In Reference [25], network optimization methods using SDN is studied for Youtube streaming as an alternative to the Deep Packet Inspection methods. In References [31, 32], multi-protocol label switching (MPLS) is incorporated to modify the routing paths

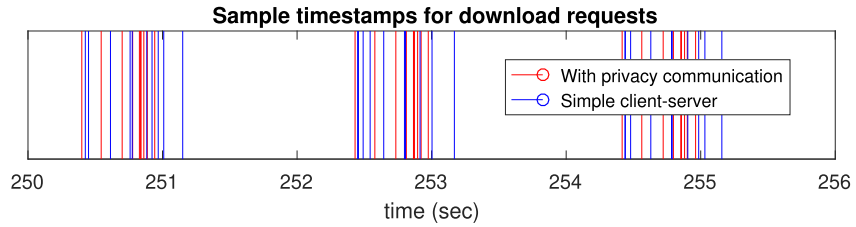


Fig. 10. Download request timestamps from 20 simulations (10 for each configuration) are shown for three of the segment downloads between seconds 250 and 256. Download timestamps from each configuration are in the neighbourhood of each other. This means communication messages with PPNet does not incur a delay in QoE.

and traffic engineering is also applied using the SDN platform. In Reference [17], benefits of traffic engineering is studied in OpenFlow networks for different video coding configurations. In Reference [9], optimization of queue allocation for the adaptive bitrate requests is studied.

Another group of studies [10, 19, 22, 26] perform traffic engineering to address fairness between the clients. In Reference [19], a home network with heterogeneous clients such as tablet, smartphone, PC, and HDTV is simulated. Outside bandwidth limitations are imposed to the home network as the video service is provided from the outside of the network. In Reference [26], study addresses the privacy of clients in case of network fairness. A similar home network is simulated. The SDN router located in the home network communicates with the clients to provide bitrate suggestions. In Reference [10], fairness in ISP networks has been studied rather than home networks.

There are studies that perform network caching or CDN-ISP collaboration for adaptive streaming using SDN. Although their improvements and experimental study is on the other aspects of collaboration, those studies include the server selection in their conceptual design. One example is OpenCache [18]. OpenCache, which is also named "Cache as a Service," allows ISP to re-direct client requests into its own cache nodes. Those nodes store a replica of the URL content upon request. Since the re-direction is seamless to the client, it does not require any change in client software. A similar study [14] presents an API based on the idea of Network Function Virtualization (NFV) to perform the same operation. In Reference [16], the idea same as OpenCache is extended with a better cache-replacement handling. We note that those sdn-based caching methods rely on URL caching, which will not work over the encrypted traffic. The network provider cannot detect traffic information such as URL requests in the encrypted case. In Reference [13], the client performs video requests over an ISP-based web server to assist bitrate selection from local CDN nodes. This method also requires video information to be exposed to the ISP and violates privacy principles of current VoD providers. To the best of our knowledge, this study is the first to address client's privacy in CDN-ISP collaborations for network-assisted resource routing.

Examples of CDN-ISP collaboration using SDN can also be seen outside of VoD applications. In Reference [15], SDN-ISP collaboration has been studied for optimizing the inter-connections between CDN nodes. In Reference [42], the DNS-based address routing method is replaced with the SDN-based routing for faster re-routing for the CDN nodes. In Reference [35], an intelligent center is incorporated into the CDN routing decision as a hierarchy to combine several SDN controllers. Other video applications studied with SDN are adaptive bitrate for Multicast in VoIP [6], constant bitrate (CBR) video streaming [21], 3D teleimmersion [7], value-added video services (e.g., web advertisements) [23], and high-throughput interactive applications such as video games [33].

Multi-CDN video delivery is a concept that has not been studied with network assistance. However, there are few studies that perform multi-CDN connections by relying on client-side decisions.

In two different studies, References [3, 4] commercial video providers have been examined for their CDN usage. In Reference [4], a multi-CDN connection scenario has been simulated in comparison with Netflix's CDN selection policy. Some studies [43] are based on CDN-P2P hybrids. To the best of our knowledge, this study is the first study to address network-assisted multi-CDN video delivery.

8 CONCLUSIONS AND FUTURE WORK

VoD has evolved significantly over the past few years, and more evolution is expected. Adaptive video streaming methods allow clients to make dynamic decisions during download time. The video providers have adopted encrypted web hosting to protect their client's privacy. SDN is emerging as the future architecture of ISP's network, and several studies on traffic engineering and bitrate adaptation show that it can improve client experience on adaptive streaming. SDN also makes CDN-ISP collaboration more practical.

In this study, we study PPNet, a CDN-ISP collaboration framework using SDN. Our main focus is CDN server selection. Server selection using network assistance is challenging in terms of system design due to the privacy concerns of the client. We propose an abstraction between video and network provider servers to provide privacy. Client initially connects to the video provider to query for the ISP server address specific to the connection and for the local CDN server availability for a certain video. The second connection is made to the ISP server to continuously query for the best available CDN server in the network. PPNet preserves clients' video request information from the network provider. It also preserves network provider's infrastructure details from the video provider. As our prototype proves, PPNet is fully compatible with web-based interfaces and encrypted transport methods.

In this study, we also experiment QoS-aware multi-CDN adaptive video delivery using PPNet framework. We formulate and solve CDN server selection problem using input from a practical SDN network using OpenFlow. Congestion avoidance has been the focus of optimization, since it can cause the largest delays in the network. On top of that, our framework has the ability to utilize multiple servers for a single stream by frequently changing the streaming server as network conditions change. Our experiments show significant improvement on QoE of the client compared to the methods that utilize a single server per stream. Experiments are performed through MPEG-DASH video streams that are played via a web browser interface. Different alternative methods for comparison utilize geo-location, network topology, and QoS parameters from SDN.

This study also leads to possible future studies on SDN technologies for adaptive bitrate video. CDN selection can be further improved by merging with routing and traffic engineering techniques. For example, optimizing route selection and CDN selection in the equation will improve load balancing. Similarly, additional information exchange between CDN and ISP on server capacity may help optimizing the server load with the traffic load. Privacy protection can be further studied by specifying the limitations of what information can be shared without a risk for providers and clients and in what extent this information can be utilized to improve service.

REFERENCES

[1] 2015. *Global Internet Phenomena, Latin America & North America—May 2015.* Technical Report. Sandvine.

[2] 2018. *Cisco Visual Networking Index: Forecast and Trends, 2017–2022, White Paper.* Technical Report. Cisco.

[3] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Volker Hilt, and Zhi-Li Zhang. 2012. A tale of three CDNs: An active measurement study of Hulu and its CDNs. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'12).* IEEE, 7–12.

[4] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. 2012. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM'12).* IEEE, 1620–1628.

[5] Saamer Akhshabi, Lakshmi Anantakrishnan, Ali C. Begen, and Constantine Dovrolis. 2012. What happens when HTTP adaptive streaming players compete for bandwidth? In *Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video*. ACM, 9–14.

[6] Christelle Al Hasrouty, Vincent Autefage, Cristian Olariu, Damien Magoni, and John Murphy. 2016. SDN-driven multicast streams with adaptive bitrates for VoIP conferences. In *Proceedings of the IEEE International Conference on Communications*.

[7] Ahsan Arefin, Raoul Rivas, Rehana Tabassum, and Klara Nahrstedt. 2013. OpenSession: SDN-based cross-layer multi-stream management protocol for 3D teleimmersion. In *Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP'13)*. IEEE, 1–10.

[8] Alon Atary and Anat Bremler-Barr. 2016. Efficient round-trip time monitoring in OpenFlow networks. In *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM'16)*. IEEE, 1–9.

[9] K. Tolga Bagci, Kemal E. Sahin, and A. Murat Tekalp. 2016. Queue-allocation optimization for adaptive video streaming over software defined networks with multiple service-levels. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'16)*. IEEE, 1519–1523.

[10] Kadir Tolga Bagci, Kemal Emrecan Sahin, and A. Murat Tekalp. 2017. Compete or collaborate: Architectures for collaborative DASH video over future networks. *IEEE Trans. Multimedia* 19, 10 (2017), 2152–2165.

[11] Abdelhak Bentaleb, Ali C. Begen, and Roger Zimmermann. 2016. SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking. In *Proceedings of the 24th ACM International Conference on Multimedia*. ACM, 1296–1305.

[12] Mark Berman, Jeffrey S. Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. 2014. GENI: A federated testbed for innovative network experiments. *Comput. Netw.* 61 (2014), 5–23.

[13] Divyashri Bhat, Amr Rizk, Michael Zink, and Ralf Steinmetz. 2018. SABR: Network-assisted content distribution for QoE-driven ABR video streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 2s (2018), 32.

[14] Matthew Broadbent, Daniel King, Sean Baildon, Nektarios Georgalas, and Nicholas Race. 2015. OpenCache: A software-defined content caching platform. In *Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft'15)*. IEEE, 1–5.

[15] Dukhyun Chang, Junho Suh, Hyogi Jung, Ted Taekyoung Kwon, and Yanghee Choi. 2012. How to realize CDN interconnection (CDNI) over OpenFlow? In *Proceedings of the 7th International Conference on Future Internet Technologies*. ACM, 29–30.

[16] Wei-Kuo Chiang and Tsung-Ying Li. 2016. An extended SDN-based in-network caching service for video on demand. In *Proceedings of the 2016 International Computer Symposium (ICS'16)*. IEEE, 159–164.

[17] Hilmi E. Egilmez, Seyhan Civanlar, and A. Murat Tekalp. 2013. An optimization framework for QoS-enabled adaptive video streaming over OpenFlow networks. *IEEE Trans. Multimedia* 15, 3 (2013), 710–715.

[18] Panagiotis Georgopoulos, Matthew Broadbent, Arsham Farshad, Bernhard Plattner, and Nicholas Race. 2015. Using software defined networking to enhance the delivery of video-on-demand. *Comput. Commun.* 69, C (2015), 79–87.

[19] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. 2013. Towards network-wide QoE fairness using openflow-assisted adaptive video streaming. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*. ACM, 15–20.

[20] Paul Goransson, Chuck Black, and Timothy Culver. 2016. *Software Defined Networks: A Comprehensive Approach*. Morgan Kaufmann.

[21] Thinh Pham Hong, An Nguyen Duc, Thoa Nguyen, Truong Thu Huong, and Nam Pham Ngoc. 2017. Adaptation method for streaming of CBR video over HTTP based on software defined networking. In *Proceedings of the International Conference on Advanced Technologies for Communications (ATC'17)*. IEEE, 16–20.

[22] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. 2012. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, 225–238.

[23] Narjes T. Jahromi, Sami Yangui, Adel Larabi, Daniel Smith, Mohammad A. Salahuddin, Roch H. Glitho, Richard Brunneri, and Halima Elbiaze. 2017. NFV and SDN-based cost-efficient and agile value-added video services provisioning in content delivery networks. In *Proceedings of the 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC'17)*. IEEE, 671–677.

[24] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. 2013. B4: Experience with a globally-deployed software defined WAN. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 3–14.

[25] Michael Jarschel, Florian Wamser, Thomas Hohn, Thomas Zinner, and Phuoc Tran-Gia. 2013. SDN-based application-aware networking on the example of youtube video streaming. In *Proceedings of the 2nd European Workshop on Software Defined Networks (EWSDN'13)*. IEEE, 87–92.

[26] Jan Willem Kleinrouweler, Sergio Cabrero, and Pablo Cesar. 2017. An SDN architecture for privacy-friendly network-assisted DASH. *ACM Trans. Multimedia Comput. Commun. Appl.* 13, 3s (2017), 44.

[27] Simon Knight, Hung X. Nguyen, Nick Falkner, Rhys Bowden, and Matthew Roughan. 2011. The internet topology zoo. *IEEE J. Select. Areas Commun.* 29, 9 (2011), 1765–1775.

[28] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. 2009. Moving beyond end-to-end path information to optimize CDN performance. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. ACM, 190–201.

[29] Danny H. Lee, Constantine Dovrolis, and Ali C. Begen. 2014. Caching in http adaptive streaming: Friend or foe? In *Proceedings of the Network and Operating System Support on Digital Audio and Video Workshop*. ACM, 31.

[30] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. 1997. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Comput. Commun. Rev.* 27, 3 (1997), 67–82.

[31] Hyunwoo Nam. 2016. *Measuring and Improving the Quality of Experience of Adaptive Rate Video*. Columbia University.

[32] Hyunwoo Nam, Kyung-Hwa Kim, Jong Yul Kim, and Henning Schulzrinne. 2014. Towards QoE-aware video streaming using SDN. In *Proceedings of the Global Communications Conference (GLOBECOM'14)*. IEEE, 1317–1322.

[33] Aous Thabit Naman, Yu Wang, Hassan Habibi Gharakheili, Vijay Sivaraman, and David Taubman. 2018. Responsive high throughput congestion control for interactive applications over SDN-enabled networks. *Comput. Netw.* 134 (2018), 152–166.

[34] David Naylor, Alessandro Finamore, Ilias Leontiadis, Yan Grunenberger, Marco Mellia, Maurizio Munafò, Konstantina Papagiannaki, and Peter Steenkiste. 2014. The cost of the S in HTTPS. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*. ACM, 133–140.

[35] Feng Qin, Zhifeng Zhao, and Honggang Zhang. 2016. Optimizing routing and server selection in intelligent SDN-based CDN. In *Proceedings of the 8th International Conference on Wireless Communications & Signal Processing (WCSP'16)*. IEEE, 1–5.

[36] Robert Ricci, Eric Eide, and CloudLab Team. 2014. Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. *; login::* 39, 6 (2014), 36–38.

[37] Zachary M. Seward. 2014. Netflix is making sure customers know whom to blame for slow, grainy video. Retrieved from https://qz.com/216609/netflixs-video-error-message-is-a-clever-attack-on-cable-companies/.

[38] Megumi Shibuya, Atsuo Tachibana, and Teruyuki Hasegawa. 2014. Efficient performance diagnosis in openflow networks based on active measurements. In *Proc. 2014 ICN*. 268–273.

[39] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM'16)*. IEEE, 1–9.

[40] Randall Stewart, John-Mark Gurney, and Scott Long. 2015. Optimizing TLS for high-bandwidth applications in FreeBSD. In *Proceedings of the Asia BSD Conference*. Citeseer.

[41] Sipat Triukose, Zhihua Wen, and Michael Rabinovich. 2011. Measuring a commercial content delivery network. In *Proceedings of the 20th International Conference on World Wide Web*. ACM, 467–476.

[42] Matthias Wichtlhuber, Robert Reinecke, and David Hausheer. 2015. An SDN-based CDN/ISP collaboration architecture for managing high-volume flows. *IEEE Trans. Netw. Serv. Manage.* 12, 1 (2015), 48–60.

[43] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. 2009. Design and deployment of a hybrid CDN-P2P system for live video streaming: Experiences with LiveSky. In *Proceedings of the 17th ACM International Conference on Multimedia*. ACM, 25–34.

Received March 2019; revised September 2019; accepted January 2020