# Using Software Defined Networking to Enhance the Delivery of Video-on-Demand[☆]

Panagiotis Georgopoulos[a,*], Matthew Broadbent[b], Arsham Farshad[b], Bernhard Plattner[a], Nicholas Race[b]

[a]*Communication Systems Group, TIK, ETH Zurich, Zurich, 8092, Switzerland*
[b]*School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, United Kingdom*

**Abstract**

High quality online video streaming, both live and on-demand, has become an essential part of many consumers' lives. The popularity of video streaming, however, places a burden on the underlying network infrastructure. This is because it needs to be capable of delivering significant amounts of data in a time-critical manner to users. The Video-on-Demand (VoD) distribution paradigm uses a unicast independent flow for each user request. This results in multiple duplicate flows carrying the same video assets that only serve to exacerbate the burden placed upon the network. In this paper we present OpenCache: a highly configurable, efficient and transparent in-network caching service that aims to improve the VoD distribution efficiency by caching video assets as close to the end-user as possible. OpenCache leverages Software Defined Networking technology to benefit last mile environments by improving network utilisation and increasing the Quality of Experience for the end-user. Our evaluation on a pan-European OpenFlow testbed uses adaptive bitrate video to demonstrate that with the use of OpenCache, streaming applications play back higher quality video and experience increased throughput, higher bitrate, and shorter start up and buffering times.

[☆]This is an extended version of our previous work appearing in ICCCN 2014 [1].
[*]Corresponding author
[1]Email addresses: panos@tik.ee.ethz.ch, m.broadbent@lancaster.ac.uk, a.farshad@lancaster.ac.uk, plattner@tik.ee.ethz.ch, n.race@lancaster.ac.uk

*Keywords:*
Video-on-Demand (VoD), Caching, Quality of Experience (QoE), Software Defined Networking (SDN), OpenCache

## 1. Introduction

Online video streaming has seen a huge growth in popularity during recent years. This includes the consumption of both live and on-demand content. In 2013, Internet video traffic represented 66% of all global Internet traffic, and is predicted to increase to 79% by 2018 [2]. At the same time, the popularity of Video-on-Demand (VoD) traffic also continues to increase, with consumer VoD traffic expected to double by 2018: transporting the equivalent of 6 billion DVDs per month [2]. High Definition (HD) video traffic has already surpassed that of Standard Definition (SD) [3], and with the introduction of Ultra-High Definition (UHD) content providers will continue to push expectations in the availability of higher video quality and bitrates. Undoubtedly, high quality online video streaming has become an essential part of many consumers' lives.

In a Video-on-Demand scenario, individuals are able to retrieve content for playback after the initial broadcast. The growth in VoD traffic, coupled with the trend towards content of higher resolution and quality, such as HD and UHD, presents significant challenges. In particular, the evolution in video quality requires networks that are capable of transferring significant amounts of data, in the order of tens or hundreds of Mbps for a single video stream, in a time-sensitive manner. This increase, together with the general growth in traffic, places an additional burden on the underlying network and distribution infrastructure.

Currently, VoD requests are handled individually, leading to *an independent flow* in the distribution network serving *each user's request.* Using such a unicast content delivery paradigm naively ignores that much of the content is identical to transmissions minutes, hours or days earlier. Hence, a very large amount of identical media objects, in the order of gigabytes for a typical HD film, are delivered over the same network segment repeatedly. In order to efficiently support such VoD streaming, the end-to-end capacity of the network must continuously match the increasing number of Internet video users and the growing popularity of higher resolution content. Mechanisms are therefore sought to improve the efficiency of VoD distribution.

In this paper we introduce OpenCache: a transparent, flexible and highly configurable in-network caching service for VoD streaming. OpenCache's contribution is to provide a programmable service that allows any caching strategy, e.g. [4, 5, 6, 7, 8], to be easily deployed within the network infrastructure. To achieve this goal, OpenCache uses Software Defined Networking (SDN) to provide a *cache as a service* for media content in an efficient and transparent fashion. This is achieved through powerful interfaces, designed to directly benefit last mile environments. By leveraging SDN, and OpenFlow in particular [9], we provide a control plane that orchestrates the caching and distribution functionalities, and transparently pushes the content as close to the user as possible without requiring any changes to the delivery methods or the end-hosts.

Our approach, building an SDN-based in-network caching service, has three important contributions. Firstly, it improves network utilisation and minimises the external link usage on the last mile. Secondly, OpenCache reduces the distribution load from the VoD content provider and all the transient networks along the path of the VoD server to the end-user. Thirdly, by transparently caching the content closer to the user, OpenCache minimises the distance between the VoD streaming server and the user. This provides significant improvements to the Quality of Experience (QoE) of the end-user, as the streaming application observes higher throughput, higher minimum and average streaming bitrate, and smaller start up and buffering times; key QoE differentiators [10, 11, 12].

The remainder of the paper is organised as follows. Section 2 provides the background of this work, whilst related work is presented in Section 3. Section 4 introduces the main components and functionality of OpenCache, whereas Section 5 describes the benefits achieved with using SDN. Evaluation is described in Section 6 and finally, Section 7 concludes the paper.

## 2. Background

This section introduces the motivation and problem statement that underpins our work. We then present a new networking approach, called Software Defined Networking, that is key to the solution we provide in this problem space.

### *2.1. Motivation & Problem Statement*

To achieve high quality VoD streaming, a potential solution should be capable of addressing these primary requirements:

**1) Provide high throughput end-to-end:** High quality video streaming demands quick and reliable transmission of high bitrate encoded content end-to-end. It is often the case that the intermediate networks become the bottleneck for high quality video streaming. It is not sufficient to simply ensure adequate origin server capacity, but adequate network bandwidth must be available in all the intermediate networks between the content server and the end-user [13, 14, 15]. Considering the fragmented nature of the Internet, illustrated by the fact that even the largest network worldwide accounts for only 5% of user traffic and needs over 650 networks to reach 90% of access traffic [13, 16], this is a stark problem. This fragmentation means that content that is centrally hosted must travel over multiple networks to reach end-users. Therefore, the burden falls on the intermediate networks to ensure adequate capacity is available to achieve the necessary end-to-end throughput for high quality streaming.

**2) Minimise distance between VoD server and user:** Large geographical distance between the content server and the end-user introduces the potential for higher latency and packet loss in today's best-effort Internet. High latency and packet loss are particularly important as, when present, the user will observe greater start up and buffering times and may also be subjected to frame drops and playback freezing. Ultimately, these events result in a lower Quality of Experience (QoE) [10, 11, 12, 14, 15, 17]. In order to minimise packet loss and benefit from reliable transmission, major VoD content providers (e.g. Netflix, Amazon's Instant Video, YouTube etc.) use TCP to stream VoD content

[18, 16, 14, 15, 19]. However, TCP's performance is highly affected by latency and packet loss, which is noticeably present when the VoD server and client are far away from each other. This is because TCP's throughput is inversely related to network latency or RTT [13, 14]. Therefore, from both a networking and QoE perspective, the distance between the server and the end-user can become a significant bottleneck in maintaining high quality video streaming.

A potential solution should address the aforementioned challenges and ensure that the media content resides as close to the user as possible. Such an approach would ensure lower latency and higher throughput end-to-end, eventually leading to higher video quality and higher QoE overall [12, 10, 11].

### *2.2. Software Defined Networking*

Software Defined Networking (SDN) is a new networking approach that facilitates the decoupling of the control plane in a network (i.e. the decision making entity) from the data plane (i.e. the underlying forwarding mechanism). OpenFlow [9], a prominent SDN protocol, defines the communication between the Layer 2 switches and the controller of a network in an open and vendor-agnostic manner. OpenFlow allows experimenters, application developers and network administrators to exploit the true capabilities of a network in an easily deployable and flexible manner. With the centralised network perspective that SDN provides (through its controller), an administrator has an overarching view of the current network status and has the ability to programmatically introduce new network-wide functionality without having to interact with each individual network or user device. OpenCache, our in-network caching service, uses OpenFlow to dynamically cache and distribute media content within a network in a highly efficient and transparent manner.

## 3. Related Work

Related work that seeks to improve VoD distribution efficiency, spans across multiple domains. These vary from application based solutions, such as Peer-to-Peer, to server based solutions, such as cache and proxy servers or dedicated infrastructure, such as Content Delivery Networks (CDNs). In this section we provide a brief overview of each approach.

The efficiency of a **Peer-to-Peer** (P2P) based networking solution for video streaming depends heavily upon the participation of users and their willingness to share their limited storage and network resources [20]. P2P pushes the content closer to end-users and can deliver a live video stream to multiple users simultaneously. This is possible because peers can sustain the short-term retention of live video using their own limited resources. However, P2P is much less effective for VoD distribution, as the time between requests for identical content may be in the order of hours, days or even months. This imposes additional resource requirements. In addition, peers may join and leave the service at will, making quality assurance very challenging. Furthermore, the distributed nature of P2P brings a lack of central control, particularly for authentication, authorisation, accounting and security. This prohibits administrators and content providers from making informed decisions and improving the service that they provide. In addition, it prevents these parties from using intelligent caching and distribution techniques [21].

Alternatively, traditional in-network **cache and proxy** approaches aim to provide additional network and storage support by focusing on delivering the content to users locally. For example, [6] demonstrates the significant benefits of caching YouTube content, where even a very basic caching policy (i.e. a static cache with long-term popular videos) can achieve a 51% cache-hit ratio. Similar benefits are demonstrated in [7], where a simple two hour expiration caching policy yields an aggregated request and byte hit rate of 24% using cache storage of a size less than 2% of the overall data transferred. The benefits of simple (i.e. never flushed) chunk-based transparent caching over two different timescales (one hour and one day) are further illustrated in [8]. The authors of [8] report that with a 10GB cache an average cache-hit ratio of approximately 25% is achieved with no significant difference over the two different timescales, with the cache-hit ratio increasing to approximately 45% with a 1TB cache.

Historically the most common use of caching and proxy servers has been to serve static web content. Thus, existing solutions (e.g. Squid [22]) are not usually optimised for the large storage, high bandwidth and low latency application requirements of modern video delivery. Furthermore, they are typically difficult to configure and customise, particularly in a dynamic and flexible fashion.

At the other end of the spectrum, commercial caching solutions are often perceived as black boxes in the network, running on dedicated hardware and requiring third-party support. They may run with minimal or no administrator intervention or configuration, but they typically provide little flexibility as to the content that should be cached and in what manner.

Another mechanism to improve the efficiency of VoD delivery is to use a dedicated **Content Delivery Network** (CDN). CDNs deploy a large number of caching servers worldwide, in order to push content to the edges of the Internet [13, 21]. CDNs are typically deployed in order to achieve goals similar to those noted in Section 2.1. From a content service provider's perspective, CDNs are an efficient distribution and cost effective solution. However, from an ISP's perspective, CDNs do not reduce the bandwidth utilisation on last mile connections, as multiple requests for the same video content will create an equal amount of flows serving the same amount of content to end-users. Even in scenarios where a dedicated CDN cache can be deployed within an ISP's network [23], this cache is specific to a particular content provider and has strict hardware, software and networking requirements that must be met before a deployment is warranted. This makes it unsuitable for small to medium scale ISPs or last mile deployments. In addition, despite the fact that CDNs deploy their servers worldwide, it is unrealistic to expect them to deploy in every network.

Even CDNs themselves have recognised their inability to deploy in last mile environments. In order to address this problem and reduce maintenance and administration cost, Akamai introduced a hybrid CDN-P2P based solution. This complements their service by pushing content closer to end-users [21]. However, such an approach has the same drawbacks as the P2P networking model mentioned earlier, namely, requiring user involvement to download and install software, and consuming the limited storage and network resources of end-users.

It is without doubt that a more flexible, configurable and transparent in-network caching service, located closer to the user, would complement CDNs and truly benefit last mile environments.

## 4. OpenCache

OpenCache is an OpenFlow-assisted in-network caching service that provides an efficient, transparent and highly configurable caching and distribution of VoD content in the last mile.

OpenCache offers a powerful interface that provides *cache as a service*. The key contribution of OpenCache is having the means to enforce any caching policy in a network at ease, rather than implementing a very specific replacement policy (which has been covered extensively in related work, for example, [4, 5, 6, 7, 8]). This functionality is not intrinsically linked to a particular type of content, or to a specific hardware technology or software implementation. In this scenario, the control and decision of what content should be cached is passed to the ISP in which the node is placed. With this, a network administrator has now the ability to optimise his network's utilisation and external link usage. This can be achieved by enabling in-network caching for specific content. OpenCache exposes an interface through a flexible JSON-RPC based API, which allows VoD content to be cached closer to the end-user. This placement also increases QoE when streaming VoD content. Given appropriate SLAs, OpenCache's interface could also be used by content providers (e.g. CDNs) to declare their content as cacheable on last mile environments, without having to physically deploy and administer their own caching hardware. This would reduce their capital and operational expenditure.

### *4.1. Architecture*

Figure 1 presents the architectural components of OpenCache when deployed on a production network. On an SDN-enabled, OpenFlow-based network, users are connected to Layer 2 OpenFlow switches. The behaviour of these switches

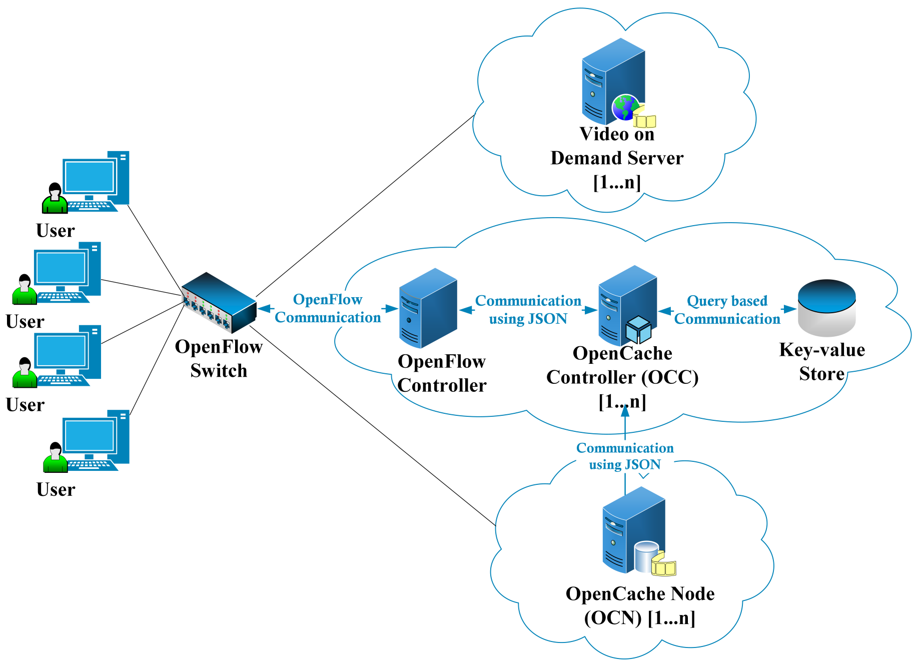


Figure 1: OpenCache Architecture

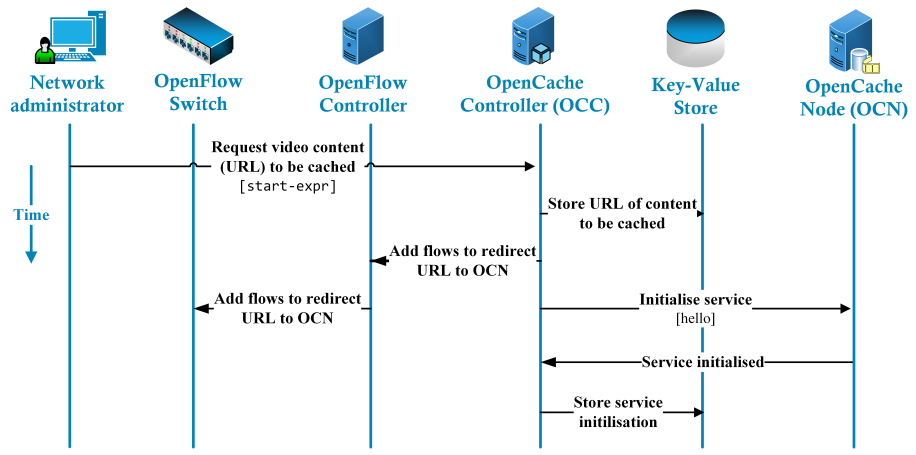
is determined by a network controller using the OpenFlow protocol [9]. The main component of OpenCache, namely the OpenCache Controller (OCC), orchestrates the VoD caching and distribution functionalities with the aid of a key-value store, acting as a database. The OCC communicates with the OpenFlow controller of the network via a JSON-RPC interface. A VoD server is the primary source for the video assets and could be located anywhere on the Internet reachable by IP. Finally, the OpenCache Nodes (OCNs) are the actual manifestation of caches, deployed at various locations within the network.

OpenCache's components, the OCC along with its partner elements (the OpenFlow controller and the key-value store), would ideally be located in the same network as the end-users. For small-scale networks all three components could even be co-located in one physical server, with OCC running as an application on top of the OpenFlow controller. A single, widely reachable OCC would be able to coordinate caching amongst a number of OCN instances (i.e. caches). It is important to note that it is not a requirement that OpenFlow be deployed throughout the network; the connecting network hardware could be entirely non-OpenFlow. Similarly, not all edge switches need to support OpenFlow, as a deployment with fewer hardware and/or software-based [24] OpenFlow switches is also entirely feasible and allows OpenCache to be fully functional. Evidently, the more OpenFlow capable switches are deployed in a network, the higher the potential performance of the service, as a result of the distribution of traffic requests across the switches. We propose that OCNs are connected directly to OpenFlow switches on which clients are also attached. This deployment would offer the lowest latency and fastest response time, and thus ensure higher QoE for the end-users. However, the use of multiple OCNs at different points in the network (e.g. attached to aggregation switches in an enterprise or University campus network) is also feasible. In fact, a hierarchical approach has the potential to provide further benefits, if a greater proportion of requests can be fulfilled without leaving the LAN.

### *4.2. The OCC's Functionality and Interfaces*

The OpenCache Controller (OCC) is the main orchestrator of the in-network caching service and provides the following five main functionalities.

**1) The OCC handles explicit requests for content** that should be cached in the network's OCNs (Figure 2), via a JSON-RPC based API (Table 1). Administrators or content service providers invoke the methods provided by this interface, using appropriate authentication credentials, to declare content that should explicitly be cached from this time onwards. If there is a request for content to be cached (using the *start* method), the OCC ensures that this is stored in the key-value store and that selected OCNs are initialised and aware of the content that was requested for caching. In addition, the OCC will interact with the network's controller and instruct it to add the matching OpenFlow redirecting rules for the cacheable content in the OpenFlow switches of the network. These flows ensure that all the users' requests for that content are redirected to their closest OCN. Finally, if there is a request to stop caching content previously added (using the *stop* method), then the OCC updates the key-value store



Network administrator
OpenFlow Switch
OpenFlow Controller
OpenCache Controller (OCC)
Key-Value Store
OpenCache Node (OCN)
Time
Request video content (URL) to be cached
[start-expr]
Store URL of content to be cached
Add flows to redirect URL to OCN
Add flows to redirect URL to OCN
Initialise service
[hello]
Service initialised
Store service initilisation

Figure 2: Declaring Content of Interest as Cacheable in OpenCache

Table 1: Interface for Declaring Content of Interest

| Method | Parameters | Result |
|---|---|---|
| start | { "expr" : <expr> } | <boolean> |
| stop | { "expr" : <expr> } | <boolean> |
| pause | { "expr" : <expr> } | <boolean> |
| fetch | { "expr" : <expr> } | <boolean> |
| seed | { "expr" : <expr> } | <boolean> |
| refresh | { "expr" : <expr> } | <boolean> |
| stat | { "expr" : <expr> } | <boolean> |

appropriately and ensures that all the matching content and flows are removed from the network's OCNs and OpenFlow switches, respectively.

**2) The OCC reports aggregated cache statistics** using the *stat* method of the interface that OpenCache exposes (Table 1). The administrator can also manually request updated statistics from the OCC (that is centrally collecting statistics from the OCNs) by invoking the *refresh* method before using the *stat* method. The ability to monitor and report aggregate cache statistics is a particularly important function for optimising network and cache resources (e.g. to implement cache load balancing). The *stat* method provides information for a variety of metrics, such as the state of each OCN (start, stop or pause), the OCNs' available storage, the number of cache hits or cache misses (in event counts or bytes), or the video objects that are cached in each OCN.

**3) The OCC orchestrates the caching strategy** that determines the content that should be cached and the caching location (i.e. a specific OCN). OpenCache's ability to control the caching logic centrally on the OCC allows the administrator to program and deploy their desired caching behaviour at will. The OCC not only has the ability to monitor and report caching statistics that can be used to make dynamic changes to caching behaviour in the network (as mentioned previously), but also has the means to enforce this change in both the network (by managing flow rules in the OpenFlow switches) and on the constituent OCNs.

To this effect, the OCC's interface offers three additional methods for managing content caching, i.e. *pause*, *fetch*, *seed* (Table 1). With the use of the *pause* method, the administrator has the ability to pause caching at given cache instances, temporarily stopping them from handling requests for content, but without removing the content from their storage. This pausing could be particularly beneficial for certain caching policies or for maintenance and administration purposes, where OCN resources have to be, for example, upgraded but without the need to remove all the previously cached content.

With the *fetch* method, an administrator can declare to the OCC that an OCN (or set of them) should fetch certain content from a remote location, even though this content might not have been explicitly requested by a user thus far. This method essentially allows OpenCache to pre-cache popular content. This is a very important functionality as certain content could be pre-cached during periods where the network utilisation is low and thus further reduce the network utilisation during peak hours.

The purpose of the *seed* method is to map multiple web addresses with a specific video file in order to reduce storage utilisation. When the *seed* method is invoked, a number of fully resolved expressions (web addresses) that serve the same video file are added to the OCNs via the OCC. All the expressions are equivalent to each other: the same content is served irrespective of the URL the user requested that content from.

The aforementioned methods allow the administrator to define optimised caching policies based on a variety of parameters that might be important in his network. For example, he could program the caching logic to minimise the streaming latency of recorded video lectures on a University's network, or to implement the pre-caching of popular content closer to end-users overnight, when the network is underutilised. It is important to note the ease and speed at which the administrator can actually implement the caching logic with the use of OpenFlow in OpenCache.

**4) The OCC manages the caching resources** of the network. An important part of resource management is to be able to dynamically handle the addition and removal of cache instances. For this reason, the OCC exposes another JSON-RPC based API that allows the OCC to communicate with a number of OCNs. This includes 3 methods; *hello*, *keep-alive* and *goodbye*. When an OCN is added to the network, it invokes the *hello* method to let the OCC know that it is now available. In turn, the OCC replies with a *node-id* that is assigned to this particular OCN. From that point on, the OCN will periodically send a *keep-alive* message to indicate that it is still in the network and functioning correctly. If the OCC does not receive a *keep-alive* call from an OCN every 15 seconds (a configurable option), then it assumes that the OCN is not reachable, either because of network congestion or because it has been taken offline. Consequently, the OCC will remove the "unreachable" OCN from the list of caching resources that it has at its disposal. Alternatively, an OCN may also leave the network gracefully with the transmission of a *goodbye* message.

The OCC is also responsible for propagating the caching strategy that the administrator has determined for the OCNs. This functionality is accomplished by allowing all the OCC interface methods (Table 1) to be also invoked on individual OCNs directly. In particular, the OCC has the ability to invoke the *start*, *stop*, *pause*, *fetch* and *seed* methods on any OCNs that it requires to act accordingly. Finally, the OCC is able to invoke the *stat* method directly on an OCI, and consequently receive counter values for all the caching metrics of that particular cache instance. Through this process, the OCC aggregates important network and caching metrics to help the administrator make informed decisions.

**5) The OCC dynamically manages the in-network flow entries** via a Flow Pusher API that an OpenFlow controller provides (e.g. Floodlight controller [25]). The OCC dynamically defines the appropriate flows that should be in the OpenFlow switches (Figure 2), so that each user's request gets redirected to an OCN in his vicinity. With the management of flows, the OCC also propagates the caching logic to the network. This ensures that the caching and distribution functionalities remain purely in the network and are fully transparent to end-users.

## *4.3. The OpenCache Node's Functionality*

The OpenCache Node (OCN) is responsible for caching the appropriate video content, and delivering it to users if they request it. When the OCN comes online in a network, it communicates with the OCC and makes its resources available to it. Subsequently, when the OCN obtains a *node-id* from the OCC, it initialises its operation and awaits users' requests.

When a user makes a video request, and if the content has been declared as cacheable, the request will be received by the closest OCN to the user. This is in contrast to it traversing the external link, which would be the case without OpenCache in place. This is possible due to the rules installed in the OpenFlow switches when explicitly declaring content to cache (Figure 2). Following these rules, an OpenFlow switch redirects the user's packets appropriately. When the OCN receives such packets, it examines if it has the requested content already cached. If the particular video is not cached in the OCN (a cache-miss scenario depicted in Figure 3), the OCN requests the video from the original VoD server. Once the first packet of this flow is received, the OCN will begin forwarding these back to the client. This process is intended to reduce any latency induced by the caching process. The delivery of this content traverses the OpenFlow switch too, and additional OpenFlow rules ensure that the packet received by the client appears to be from the expected source. With the completion of this process, the session has remained transparent to the user and there is no interruption to the service. Once the full flow has been handled in this way, the payload of the delivery is stored by the OCN in order to serve subsequent re-

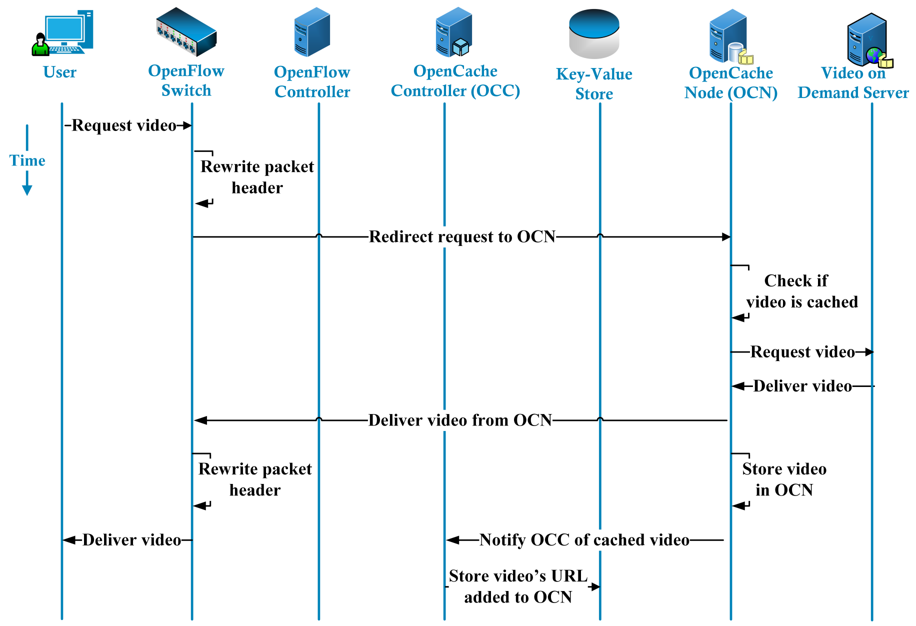


Figure 3: Cache-miss Scenario

quests. Furthermore, the OCN informs the OCC of this transaction for resource provision and management purposes.

If the content for a particular request is already stored in the OCN (a cache-hit scenario), the OCN delivers that video directly to the user in a transparent fashion. As in a cache-miss scenario, the content always appears to originate from the VoD server the client originally requested it from. However, in a cache-hit scenario, no traffic would have left the user's network into other networks, thus saving external link utilisation and significantly reducing startup and buffering delays. The role of the OCN is such that there is an inherent need to have multiple instances of it distributed in the network to facilitate users' requests and content caching as efficiently as possible.

## 5. Benefits of SDN

The use of Software Defined Networking (SDN), and OpenFlow in particular, provides OpenCache with the ability to redirect requests for content to a running cache instance. This is achieved in a truly transparent fashion: content that reaches the client appears to originate from the origin server rather than the cache. It is important to note that this is possible without the costly and time-consuming modification of existing delivery techniques or end-client devices.

The only requirement for OpenCache to function is the presence of a single OpenFlow-capable switch on the path from client to the server. We present OpenCache in a last mile environment, where traditionally there has existed no such cache process. However, the exact same process can be applied in other situations without any modifications.

Using OpenFlow affords us with unparalleled access to real-time network conditions. In conjunction with OpenCache's API, these metrics can be used to effectively satisfy any possible caching requirements. This includes the novel ability to load balance requests at a network level.

## 6. Evaluation

In order to evaluate the efficacy of OpenCache, we carried out a number of VoD streaming experiments representing different scenarios over a large-scale pan-European OpenFlow testbed; the GÉANT OpenFlow Facility (GOFF) [26]. The following sections describe the testbed setup, the chosen evaluation criteria and the experimental scenarios that we devised to evaluate OpenCache. In addition, we describe Scootplayer, an open-source tool that we implemented to measure key QoE metrics, in order to extensively evaluate OpenCache's performance. Finally, we present and analyse our experimentatal results.

### *6.1. Experimentation Testbed & Setup*

The GN3plus EU project [27] provides the GOFF, which is composed of a number of OpenFlow-capable software-based switches (i.e. Open vSwitches [24]) and virtualised computing resources. These resources are located in five different countries across Europe, as depicted in Figure 4. All sites are connected as to produce a large federated experimentation environment, orchestrated by a

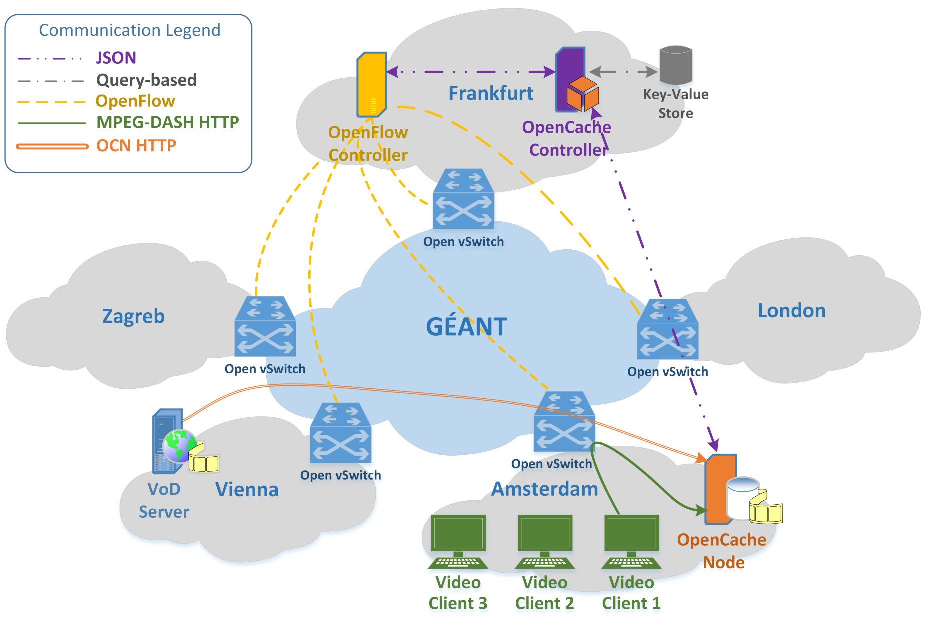


Figure 4: The GÉANT OpenFlow Facility and our Experimentation Setup

control framework. This framework works together with a web-based front-end to expose the experimentation services of the testbed to its end-users.

For our experimentation, three GOFF sites were used, namely, the Amsterdam, Vienna and Frankfurt site. Three video streaming clients and one OCN were set up in virtual machines located at the Amsterdam site, and one VoD server was deployed in Vienna. Furthermore, an OpenFlow Controller, an OCC and a key-value store were deployed in Frankfurt. Figure 4 depicts the experimentation setup we created on GOFF, with the OpenFlow controller in Frankfurt federating all the sites into one OpenFlow network.

## *6.2. Experimentation Scenarios*

We defined two VoD streaming experimentation scenarios, with each including two distinct tests. Scenario A involves single video client tests (only VoD Client 1, Figure 4), whereas Scenario B involves multiple video client tests (all three VoD Clients, Figure 4). In each scenario, Test 1 is carried out with the default link characteristics of the testbed as a baseline, whereas Test 2 is car-

Table 2: Network Link Characteristics for Each Test

| Scen. A & B | Link | Description |
|---|---|---|
| **Test 1** | Default Link Setup | No additional link latency or packet loss is added to the testbed links. Indicatively, VoD Clients experience on average 30ms RTT delay when communicating with the VoD Server. |
| **Test 2** | Emulated Link Setup | The clients' site link characteristics fall into the following three categories at each point in time: 45% have default link characteristics, 45% have additional 50ms RTT and 0.1% packet loss, and 10% have additional 150ms RTT and 0.1% packet loss. |

ried out by emulating additional link latency and packet loss on the clients' site egress link. This is because GOFF is a relatively bandwidth-rich, low latency and reliable environment, which may not necessarily represent a typical domestic or enterprise network. Therefore, to complement Test 1, in Test 2 of both scenarios we used a Linux network emulation tool, called dummynet [28], to introduce additional variable latency and packet loss and hence further evaluated OpenCache's performance in networks with different link characteristics (detailed in Table 2).

For both tests of Scenario A (single VoD client with default and emulated link setup), we carried out 20 VoD streaming runs for each of the following three cases: without a cache (which provides a baseline), a cache-miss (where content was not found on the OCN, and thus fetched from the original content server) and a cache-hit (where content was found and delivered from the OCN only). These 3 cases provide a direct comparison of the effects of the cache that OpenCache provides on content delivery, across a number of important metrics described in Section 6.4.

For both tests of Scenario B (multiple VoD clients with default and emulated link setup), we carried out 20 VoD streaming runs of the following case. At time zero, Client 1 starts streaming video content first, and after 30 seconds Client 2 starts streaming as well. Then, 60 seconds after Client 2 has started, Client 3 also starts streaming content. This scenario allows us to evaluate the performance of OpenCache and its potential QoE improvements to end-users, with multiple clients and multiple concurrent VoD TCP flows in the network.

In all experimental runs, the video client(s) requests the same reference video file (i.e. "Big Buck Bunny"[2]) from the original VoD server, with playback lasting for 9 minutes and 56 seconds. The content is streamed using an adaptive video streaming technology, namely DASH (Dynamic Adaptive Streaming over HTTP) [29]. MPEG-DASH facilitates the dynamic adjustment of the streaming bitrate by offering various bitrate encodings of the reference video, fragmented into fixed time chunks. This content is described in the Media Presentation Description (MPD) file used by DASH and requested by the video client(s). In our tests, the MPD file describes 20 different video quality representations in chunks of 15 seconds each, from bitrates of 50Kbps up to 8000Kbps.

## *6.3. Scootplayer: VoD Streaming Evaluation Tool*

For use in the aforementioned experimental scenarios, we built a unique VoD evaluation tool, called Scootplayer. Scootplayer is an open-source tool that we offer freely to the research community [30]. It was designed to help experimenters better understand the effect of network conditions and their relationship with the end-users' QoE, by accurately logging an extensive number of video streaming metrics as experienced during playback on the VoD clients. Scootplayer is compatible with MPEG-DASH [29] manifest files, and can be used to create realistic HTTP adaptive streaming traffic in a network. In addition, we chose to develop Scootplayer because it generates scientifically reproducible results.

## *6.4. Evaluation Criteria*

The main aim of our testing scenarios is to evaluate OpenCache's impact on the network link utilisation and its potential impact on the QoE of the end-users. To this effect, and influenced by related work [12, 10, 11], we define four key QoE metrics, namely, (a) startup time, (b) bitrate changes during playback, (c)

[2]http://www.bigbuckbunny.org/

Table 3: QoE Evaluation Criteria

| **Metric** | **Definition** |
|---|---|
| **Startup Time** | The time it takes a VoD client to start playback. |
| **Video Bitrate Changes** | The number of times a VoD client has to change the video streaming bitrate during playback. |
| **Weighted Average Video Bitrate** | The average video bitrate experienced by a VoD client weighted by its duration during playback. |
| **Minimum Video Bitrate** | The minimum video bitrate a VoD client experienced during playback. |

average bitrate achieved during playback (weighted based on its duration) and (d) minimum bitrate requested during playback. The description of each metric is further given in Table 3. We designed and implemented Scootplayer to report these metrics directly at the end of each playback run. These metrics were chosen for their direct impact on QoE as, for example, the higher the average bitrate achieved or the higher the minimum bitrate requested during playback, the higher the QoE of the end-user potentially is. In addition, we report the improvement that OpenCache brings to network utilisation by reporting cache hits and cache misses as an evaluation metric. These cache based measurements indicate how much OpenCache saved from traversing the network, by not having to fetch all the content from the original VoD server and hence utilising less of the external network capacity.

### *6.5. Results*

In this section, we present the results for our two experimentation scenarios.

#### *6.5.1. Scenario A: Single Video Client Experiments*

The results from our experiments with a single video client (Scenario A, as explained in Section 6.2) are shown averaged for our 20 runs of each test in Figure 5. They clearly demonstrate the benefits of serving the content from OpenCache, across all four QoE metrics we measured. We would like to also highlight the efficiency of OpenCache in cache-miss cases, where OpenCache presents only a 15% overhead in startup times when compared to the client

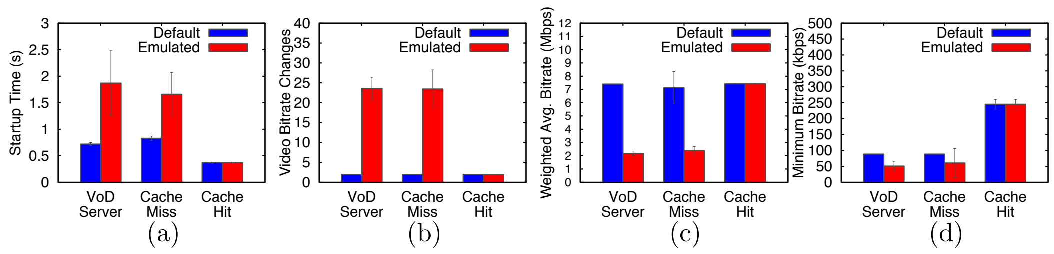


Figure 5: Scenario A: Single Client Results for Default and Emulated Link Setup

directly fetching the content from the original server. This is despite the fact that an additional leg in the communication is introduced.

Figure 5a clearly depicts the approximately two times quicker startup times that OpenCache achieves in cache-hit cases with the default link setup. This is when compared to the startup times recorded in direct VoD server communication or in the cache-miss cases. The improvement in startup times is even higher (more than 4 times faster) with the emulated link setup, where the path to the VoD server has longer link delay and higher packet loss. The degraded link characteristics of the emulated link setup are also reflected in the error bars of Figure 5a that show a higher standard deviation when compare to the default link setup.

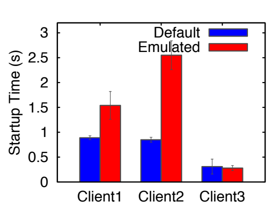
Figure 5b illustrates the improvements that OpenCache offers by lowering the number of bitrate changes the end-client has to carry out in cache-hit scenarios of the emulated link setup. A low number of bitrate changes indicates network and playback stability, and an increase in QoE when coupled with the improvements in the minimum and average bitrate achieved during playback (Figure 5c and 5d, respectively). In particular, in the emulated link setup for direct VoD communication or cache-miss cases, the end-client has to perform 23 bitrate changes on average, but with OpenCache present in cache-hit scenarios only 2 changes are necessary (Figure 5b). This is observed only in the emulated link setup as the RTT delay and packet loss are very low in the default link setup. OpenCache achieves additional QoE improvements in the minimum and average bitrate achieved during playback in cache-hit scenarios (Figure 5d and 5c, respectively). This comparison is between cache hits, and the direct or indirect VoD communication in the emulated link setup, as the throughput observed in the default link setup is constantly high. In the scenarios of the emulated link setup, we observe over three times higher weighted average bitrate and over four times higher minimum bitrate during playback, key indicators of higher QoE.

### *6.5.2. Scenario B: Multiple Video Client Experiments*

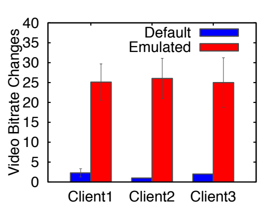
The results from our experiments with multiple video clients (Scenario B, as explained in Section 6.2) are shown averaged for 20 runs of each test in Figure 6. In both tests of Scenario B, we have three video clients starting at 0, 30 and 90 seconds, respectively. In this scenario, Client 1 always experiences cache misses as it starts first and the OCN's cache is empty, whereas, both Client 2 and Client 3 benefit from the content cached during Client 1's streaming.

During Scenario B for the default setup, we observe, on average, 57% of requests from Client 2 and Client 3 to be cache hits. In the emulated link setup, cache hits amount to 61% of requests on average. This has a direct impact on reducing the external link utilisation on the clients' site. This would typically lead to a reduction in transit traffic costs and better utilisation of existing network resources. Our analysis also shows that Client 3 exhibits more cache hits than Client 2, illustrating that with a higher number of clients we are further increasing cache hits. Consequently, this produces an increase in QoE and a reduction in external link usage.

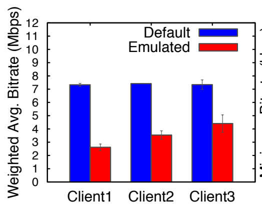
In particular, Figure 6a depicts the average startup time for all three clients and illustrates that Client 3's startup time shows approximately a threefold improvement for the default link setup and an up to sixfold improvement for the emulated link setup. We notice the greater improvement in the emulated link setup, because in this test the cache-miss cost is higher due to the increased link delay and packet loss on the path between the OCN and VoD server. For Client 2 however, the startup time is higher in the emulated link setup when compared to Client 1. This is because Client 2 estimates its available bandwidth based on downloading all the initialisation files specified in the MPD file,



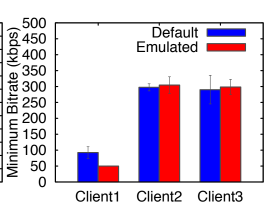
(a)



(b)



(c)



(d)

Figure 6: Scenario B: Multiple Client Results for Default and Emulated Link Setup

which are all cached in the OCN during Client 1's streaming. Based on this estimation, Client 2 then requests higher bitrate chunks than what Client 1 requested, which, as they are being requested for the first time, have to be fetched from the original VoD server by the OCN. Admittedly, these cache misses increase Client 2's startup times, but as higher bitrate chunks are requested, they also improve its minimum and weighted average bitrate achieved during playback (Figure 6d and 6c, respectively), when compared to Client 1. Apparently, Client 3 and every potential client from that point on, benefits directly as OpenCache is able to serve more higher bitrate chunks from its OCN.

Figure 6b depicts the video bitrate changes for both tests, which are at comparable levels for all clients per test. We observe that the presence of OpenCache does not influence the number of bitrate changes in multiple client scenarios, as not all requests are cache hits, and also more TCP flows are present in the network and streaming is therefore impacted. We note that the number of bitrate changes are higher in the emulated link setup compared to the default link setup as a result of the cache misses and the additional link delay and packet loss, coupled with the multitude of video streams in the network.

Finally, Figure 6c illustrates that, in the emulated link setup, with each additional client OpenCache manages to improve the average bitrate observed during playback. There are also similar improvements shown in Figure 6d regarding the minimum bitrate achieved in both default and emulated link setups. As an example, Client 3 has approximately 1.75 times higher average bitrate and 6 times higher minimum bitrate during playback when compared to the respective results for Client 1 in the emulated link setup. This illustrates that

OpenCache achieves higher QoE by increasing the overall quality of the video streaming even when multiple clients are present in the network.

## 7. Conclusion

In this paper we presented OpenCache: an efficient, transparent and highly configurable OpenFlow-assisted in-network caching service for VoD streaming. OpenCache aims to address the underlying challenge that the network faces when the same video files are streamed to end-users repeatedly using independent unicast flows.

OpenCache provides *cache as a service* by offering an interface to declare cacheable content of interest in a highly configurable and flexible manner. Furthermore, OpenCache supports centrally controlled caching that provides a platform for many additional services to be programmed on top of it with ease (e.g. load balancing, pre-caching, different caching replacement policies etc.). There is also a distinct advantage in that it is easily deployable within a production network; there are no changes required in the underlying video delivery mechanisms and all existing hardware and software can be retained. Furthering this, OpenCache is also fully transparent to the end-user; the user does not need to install any extra software, or have to sacrifice any of his local network or storage resources to stream video content with high efficiency, which other technologies require [21, 20].

In addition, and as demonstrated through single and multiple client experimentation, OpenCache provides caching very close to the user with three important benefits. Firstly, the external link usage is reduced leading to lower transit traffic costs and improved network utilisation, as end-user requests are now served locally. Secondly, OpenCache reduces the distribution load on the origin content provider and all the transient networks along the path between the origin and the end-user. Thirdly, since the content is served locally, the video client observes higher throughput and lower latency. This directly leads to higher minimum and weighted average streaming bitrate, and smaller start up and buffering times, eventually providing higher QoE to the end-user.

## 8. Acknowledgments

The work presented in this paper has been funded by the EU FP7 GN3plus project (FP7-INFRASTRUCTURES-605243) and the EU FP7 OFELIA project (FP7-ICT-258365).

## References

[1] P. Georgopoulos, M. Broadbent, B. Plattner, N. Race, Cache as a Service: Leveraging SDN to Efficiently and Transparently Support Video-on-Demand on the Last Mile, in: 23rd International Conference on Computer Communication and Networks (ICCCN) 2014, 2014, pp. 1–9. doi:10.1109/ICCCN.2014.6911775.

[2] Visual Networking Index: Forecast and Methodology, 2013-2018, Tech. rep., CISCO (June 2014).

[3] Visual Networking Index: Forecast and Methodology, 2011-2016, Tech. rep., CISCO (May 2012).

[4] A. Dan, D. Sitaram, Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads, SPIE Multimedia Computing and Networking 2667 (1996) 344–351. doi:10.1117/12.235887.

[5] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, R. Tewari, Buffering and Caching in Large-scale Video Servers, in: COMPCON '95: Technologies for the Information Superhighway, 1995, pp. 217–224. doi:10.1109/CMPCON.1995.512389.

[6] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, S. Moon, I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System, in: 7th ACM SIGCOMM IMC 2007, 2007, pp. 1–14. doi:10.1145/1298306.1298309.

[7] M. Chesire, A. Wolman, G. M. Voelker, H. M. Levy, Measurement and Analysis of a Streaming-media Workload, in: 3rd USENIX USITS 2001, 2001, pp. 1–1.

[8] C. Imbrenda, L. Muscariello, D. Rossi, Analyzing Cacheable Traffic in ISP Access Networks for Micro CDN Applications via Content-centric Networking, in: 1st International Conference on Information-centric Networking (INC) 2014, 2014, pp. 57–66. doi:10.1145/2660129.2660146.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: Enabling Innovation in Campus Networks, SIGCOMM CCR 38 (2) (2008) 69–74.

[10] S. S. Krishnan, R. K. Sitaraman, Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs, in: ACM SIGCOMM IMC 2012, 2012, pp. 211–224.

[11] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, H. Zhang, Understanding the Impact of Video Quality on User Engagement, in: ACM SIGCOMM 2011, 2011, pp. 362–373. doi:10.1145/2018436.2018478.

[12] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, H. Zhang, A Case for a Coordinated Internet Video Control Plane, in: ACM SIGCOMM 2012, 2012, pp. 359–370. doi:10.1145/2342356.2342431.

[13] E. Nygren, R. K. Sitaraman, J. Sun, The Akamai Network: A Platform for High-performance Internet Applications, SIGOPS OS Rev. 44 (3) (2010) 2–19.

[14] S. Sen, J. Rexford, D. Towsley, Proxy Prefix Caching for Multimedia Streams, in: 19th IEEE INFOCOM 1999, Vol. 3, 1999, pp. 1310–1319 vol.3.

[15] J. V. D. Merwe, S. Sen, C. Kalmanek, Streaming Video Traffic: Characterization and Network Impact, in: Int. Web Content Caching and Distribution Workshop, 2002.

[16] K. Sripanidkulchai, B. Maggs, H. Zhang, An Analysis of Live Streaming Workloads on the Internet, in: 4th ACM SIGCOMM IMC 2004, 2004, pp. 41–54.

[17] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, N. Race, Towards Network-wide QoE Fairness Using Openflow-assisted Adaptive Video Streaming, in: ACM SIGCOMM 2013 Workshop on Future Human-centric Multimedia Networking (FhMN), 2013, pp. 15–20.

[18] B. Wang, J. Kurose, P. Shenoy, D. Towsley, Multimedia Streaming via TCP: An Analytic Performance Study, ACM Trans. MCCA 4 (2) (2008) 16:1–16:22.

[19] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, W. Dabbous, Network Characteristics of Video Streaming Traffic, in: 7th ACM CoNEXT 2011, 2011, pp. 25:1–25:12.

[20] J. Pouwelse, J. Taal, R. Lagendijk, D. H. J. Epema, H. Sips, Real-time Video Delivery using Peer-to-Peer Bartering Networks and Multiple Description Coding, in: IEEE Int. Conference on Systems, Man and Cybernetics 2004, Vol. 5, 2004, pp. 4599–4605 vol.5. doi:10.1109/ICSMC.2004.1401257.

[21] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, M. Ponec, Peer-assisted Content Distribution in Akamai Netsession, in: 13th ACM SIGCOMM IMC 2013, 2013, pp. 31–42.

[22] Squid Proxy Server, `http://www.squid-cache.org/`.

[23] Netflix Open Connect Platform, `http://www.netflix.com/openconnect`.

[24] Open vSwitch: An Open Virtual Switch, `http://openvswitch.org/`.

[25] The Floodlight Controller, `http://floodlight.openflowhub.org/`.

[26] The GÉANT OpenFlow Facility (GOFF), `https://openflow.geant.net/`.

[27] The GÉANT Project, `http://www.geant.net/`.

[28] Dymmynet: Network Emulation Tool, `http://info.iet.unipi.it/~luigi/dummynet/`.

[29] ISO-IEC 23009-1:2012 Information Technology, Dynamic Adaptive Streaming over HTTP (DASH).

[30] Scootplayer: VoD QoE Evaluation Tool, `github.com/broadbent/Scootplayer`.

## Appendix

Following Elsevier policy, please find below the original paper accepted for publication at the 23rd International Conference on Computer Communications and Networks (ICCCN) 2014.

**Please remove this Appendix for final publication**.

# Cache as a Service: Leveraging SDN to Efficiently and Transparently Support Video-on-Demand on the Last Mile

Panagiotis Georgopoulos*†, Matthew Broadbent†, Bernhard Plattner* and Nicholas Race†
*Communication Systems Group, ETH Zurich, 8092 Zurich, Switzerland
†School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK
panos@tik.ee.ethz.ch, m.broadbent@lancaster.ac.uk, plattner@tik.ee.ethz.ch, n.race@lancaster.ac.uk

***Abstract*—High quality online video streaming, both live and on-demand, has become an essential part of consumers' every-day lives. The popularity of video streaming has placed a heavy burden on the network infrastructure that now has to transfer an enormous amount of data very quickly to the end-user. To further exacerbate the situation, the Video-on-Demand (VoD) distribution paradigm uses a unicast independent flow for each user request. This results in multiple duplicate flows carrying the same video assets many times end-to-end. We present OpenCache: a highly configurable, efficient and transparent in-network caching service that aims to improve the VoD distribution efficiency by caching video assets as close to the end-user as possible. OpenCache leverages Software Defined Networking to benefit last mile environments by improving network utilisation and increasing the Quality of Experience for the end-user. Our evaluation on a pan-European OpenFlow testbed uses adaptive video streaming and demonstrates that with the use of OpenCache, the external link utilisation is reduced by 100%. Furthermore the streaming application receives better quality video and observes higher throughput, lower latency and shorter start up and buffering times.**

***Keywords—Video-on-Demand (VoD), Caching, Quality of Experience (QoE), Software Defined Networking (SDN), OpenFlow, OpenCache***

## I. Introduction

Recent years have seen a huge growth in the popularity of video streaming, for both live and on-demand services. In 2012, Internet video traffic accounted globally for 57% of all consumer Internet traffic, and is predicted to increase even more, to 69% by 2017 [1]. Correspondingly, the popularity of Video-on-Demand (VoD) traffic continues to increase, with the volume of VoD traffic predicted to reach the equivalent of 6 billion DVDs per month by 2017 [2]. At the same time, High Definition (HD) video traffic has already surpassed Standard Definition (SD) traffic, making HD the de facto video quality level consumed by users [2]. There is no doubt that high quality online video streaming has become an essential part of consumers' every-day life.

With a VoD service, individuals can retrieve previously recorded content at a time after it was initially broadcast or made available. With the increasing growth of VoD and the popularity of HD content, an alarming challenge to the underlying network infrastructure is becoming apparent. The network now has to transfer an enormous amount of data to the end-user, and do so as quickly as possible. At the same time, the available content continues to improve in terms of resolution and overall video quality, and this trend will continue as we move from HD through to Ultra HD and 3D video. These changes in video quality require delivery throughput in the order of tens of Mbps for just one stream, and place additional burden on the underlying network infrastructure for supporting their distribution.

Currently, VoD requests are handled individually, leading to *an independent flow* in the distribution network serving *each user's request*. Using such a unicast content delivery paradigm naively ignores that much of the content is identical to transmissions minutes, hours or days earlier. Hence, a very large amount of identical media objects, in the order of gigabytes for each HD film, is delivered on the same network segments repeatedly. In order to efficiently support such VoD streaming, the end-to-end capacity of the network must continuously match the increasing number of Internet video users and the growing popularity of HD content. Mechanisms are thus sought to improve the VoD distribution efficiency.

In this paper we introduce OpenCache: a transparent, flexible and highly configurable in-network caching service for VoD streaming. OpenCache uses Software

Defined Networking (SDN) to provide *cache as a service* for media content in an efficient and transparent fashion. This should directly benefit last mile environments. By leveraging SDN, and OpenFlow in particular [3], we provide a control plane that orchestrates the caching and distribution functionalities, and transparently pushes the content as close to the user as possible without requiring any changes to the delivery methods or the end-hosts.

Our approach, building an SDN based in-network caching service, has three important contributions. First, it improves network utilisation and minimises the external link usage on the last mile that is often costly. Second, OpenCache reduces the distribution load from the VoD content provider and all the transient networks along the path of the VoD server to the end-user. Third, by transparently caching the content closer to the user, OpenCache minimises the distance between the VoD streaming server and the user. This provides great improvements to the Quality of Experience (QoE) of the end-user, as the streaming application observes higher throughput, less latency and smaller start up and buffering times; key QoE differentiators [4], [5], [6].

The remainder of the paper is organised as follows. Section II provides the background of this work, whilst related work is presented in Section III. Section IV introduces the main entities and functionality of OpenCache, whereas Section V describes the key benefits achieved with SDN. Evaluation is shown in Section VI and finally, Section VII concludes the paper.

## II. Background

### A. Motivation & Problem Statement

To achieve high quality VoD streaming, a potential solution should be able to address the following two primary requirements :

**1) Provide high throughput end-to-end:** High quality video streaming demands quick and reliable transmission of high bitrate encoded content end-to-end. It is often the case that the intermediate networks that have to transfer the media content quickly, become the bottleneck for high quality video streaming. It is not enough to ensure adequate origin server capacity, but adequate network bandwidth must be available in all the intermediate networks between the content server and the end-user [7], [8], [9]. With the Internet being highly fragmented, namely, the largest network worldwide accounting for only 5% of user traffic and needing over 650 networks to reach 90% of access traffic [7], [10], this is a stark problem. This fragmentation means that content that is centrally hosted must travel over multiple networks to reach end-users. Therefore, the burden falls on the intermediate networks to ensure adequate capacity is available to achieve the necessary end-to-end throughput for high quality streaming.

**2) Minimise distance between VoD server and user:** Large geographical distance between the content server and the end-user presents the potential for higher latency and packet loss in today's best-effort Internet. High latency and packet loss are particularly important as, when present, the user notices larger start up and buffering times. In addition, frame dropping and frame freezing are observed. Ultimately, these eventually lead to lower QoE [4], [5], [6], [8], [9], [11]. To minimise packet loss and benefit from reliable transmission, major VoD content providers (e.g. Netflix, Amazon's LoveFilm, YouTube etc.) use TCP to stream VoD content [12], [10], [8], [9], [13]. However, TCP's performance is highly affected by latency and packet loss (noticeably present when the VoD server and client are far away from each other). This is because TCP's throughput is inversely related to network latency or RTT [7], [8]. Therefore, from both networking and QoE perspectives, the distance between the server and the end-user can become a significant bottleneck in maintaining high quality video streaming.

A potential solution should address the aforementioned challenges and ensure that the media content resides as close to the user as possible. Such an approach would ensure lower latency and higher throughput end-to-end, eventually leading to higher video quality and lower start up and buffering times [6], [4], [5].

### B. Software Defined Networking

Software Defined Networking (SDN) is a new, very promising, networking approach that facilitates the decoupling of the control plane in a network (i.e. the decision making entity) from the data plane (i.e. the underlying forwarding system). OpenFlow [3], currently the prominent SDN protocol, defines the communication between the Layer 2 switches and the controller of the network in an open and vendor-agnostic manner. OpenFlow allows experimenters, protocol developers and network administrators to exploit the true capabilities of a network in an easily deployable and flexible manner. With the centralised network perspective that OpenFlow provides through its controller, an administrator has an overarching view of the current network status and has the ability to programmatically introduce new network-wide functionality without having to interact with each individual network or user device. OpenCache, our in-network caching service, uses OpenFlow to dynamically cache and distribute media content within a network in a highly efficient and transparent manner.

## III. Related Work

**Multicast** is a technology that can deliver the same media assets to multiple users simultaneously by reducing the delivery throughput on the origin server to one stream. Setting aside the multiple real-world deployment problems that multicast entails [14], multicast is, by design, an efficient solution for live video streaming, where, all users' requests are for the same content at the same time. However, with VoD, this is not the case, as requests can occur any time after the content becomes available. Related work that has used multicast to improve the delivery efficiency of VoD often involves unnecessary complexity, such as merging streams by speeding up or slowing down the clients' viewing rate [15], [16], holding a portion of the clients' bandwidth in reserve [17], or requiring from the clients the receipt of two or more video streams simultaneously, each at the playback rate [18], [19], [17]. Also, these solutions are not transparent and require client or server modifications.

The efficiency of a **Peer-to-Peer** (P2P) based networking solution for video streaming depends heavily upon the participation of users and their willingness to share their limited storage and network resources [20]. P2P pushes the content closer to end-users and can deliver a live video stream to multiple users simultaneously. This is possible because peers can sustain the short-term retention of live video using their own resources. However, P2P is much less effective for VoD distribution, as the time between requests for identical content may be in the order of hours, days or even months. In addition, P2P peers may join and leave the service at will, making VoD streaming quality assurance very challenging. Furthermore, the distributed nature of P2P brings a lack of central control, particularly for authentication, authorisation, accounting and security. This prohibits network administrators and content providers from making informed decisions and improving the service that they provide. In addition, it prevents these parties from using intelligent caching and distribution techniques [21].

Alternatively, traditional in-network **cache and proxy** approaches aim to provide additional network and storage support by focusing on delivering the content to users locally. For example, [22] demonstrates the significant benefits of caching YouTube content, where even a very basic caching policy (i.e. a static cache with long-term popular videos) can approximately achieve a 51% cache-hit ratio. Similar benefits are demonstrated in [23], where a simple two hour expiration caching policy yields an aggregated request and byte hit rate of 24% using cache storage of a size less than 2% of the overall data transferred.

Historically though, the most common use of caching and proxying servers is to serve static web content. Thus, existing solutions (e.g. Squid [24]) are not usually optimised for the high storage, bandwidth and the very demanding application requirements of video delivery (e.g. skipping to a certain part of a video stream). Some popular caching servers are too complex to customise and configure, and require constant attention and tuning from network administrators. To the other extreme, some commercial caching solutions provide little flexibility, customisation and configurability as to the content that should be cached and their caching policies. These solutions are essentially black boxes in the network, running on dedicated hardware and requiring third-party support. Such solutions typically leave network administrators with minimal control and resource monitoring of devices located within their own network.

Another mechanism to improve the efficiency of VoD delivery is to use a dedicated **Content Delivery Network** (CDN). CDNs deploy a large number of caching servers worldwide, in order to push content to the edges of the Internet [7], [21]. CDNs are typically deployed in order to achieve goals similar to those noted in Section II-A. From a content provider's perspective, CDNs are an efficient distribution and cost effective solution. However, from a consumer ISP's perspective, CDNs do not reduce the bandwidth utilisation on last mile connections, as multiple requests for the same video content will create an equal amount of flows serving the same amount of content to end-users. Even in scenarios where a dedicated CDN cache can be deployed within an ISP's network [25], this cache is specific to a particular service and has strict hardware, software and networking requirements (e.g. video traffic higher than a threshold), that deems it unsuitable for medium-scale ISPs or last mile deployments. In addition, despite the fact that CDNs deploy their servers worldwide, it is unrealistic to expect them to deploy in all ISP networks or last mile environments.

Even CDNs themselves have recognised their inability to truly reach out to last mile environments. For example, in order to address this problem and reduce maintenance and administration cost, Akamai introduced a hybrid CDN-P2P based solution, that complements their service by pushing content closer to end-users [21]. However, such an approach has the drawbacks of P2P networking mentioned earlier, namely, requiring user involvement to download and install software, and consuming the limited storage and uploading resources of end-users. It is without doubt that a more flexible, configurable and transparent in-network caching service, located closer to the user, would complement CDNs and truly benefit last mile environments.

## IV. OpenCache

OpenCache is an OpenFlow-assisted in-network caching service that provides efficient, transparent and highly configurable caching and distribution of VoD content in the last mile.

OpenCache offers a powerful interface that provides *cache as a service*. This is not intrinsically linked to a particular type of content, or to a specific hardware or software implementation. The control and decision of what content should be cached is passed on to the network administrator of the ISP, who now has the ability to optimise his network's utilisation and external link usage. This can be achieved by enabling in-network caching for specific content via a designated interface. OpenCache exposes this interface through a powerful and flexible JSON-RPC based API, which allows VoD content to be cached closer to the end-user. This placement also increases their QoE when streaming the content. Given appropriate SLAs, OpenCache's interface could also be used by content providers (e.g. CDNs) to declare their content as cacheable on last mile environments, without having to physically deploy and administer their own caching hardware. It is envisaged that OpenCache's interface will eventually be compatible with CDNI [26], a collaboration protocol currently under development by the research community. This functionality can be used by CDNs to define their interconnections, and hence ease interoperability and communication between them and potentially OpenCache.

Fig. 1 presents the entities of OpenCache when deployed on a production network. On an SDN-enabled, OpenFlow-based network, users are connected to Layer 2 OpenFlow switches. The functionality of these switches is dictated by a network controller using the OpenFlow protocol [3]. The main entity of OpenCache, namely the OpenCache Controller (OCC), orchestrates the VoD caching and distribution functionalities with the aid of a key-value store, that acts as a database. The OCC communicates with the OpenFlow controller of the network via a JSON-RPC interface. A VoD server is the primary source for the video assets and could be located anywhere on the Internet reachable by its IP address. Finally, the OpenCache Nodes (OCNs) are the caches of the service, inherently being deployed in various locations in the network.

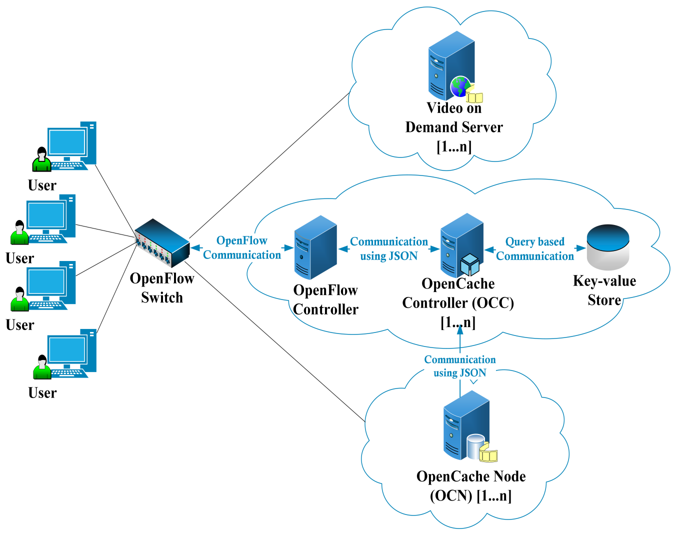


Fig. 1: OpenCache Architecture

With respect to the placement of the OpenCache's entities, the OCC, along with its partner elements (the OpenFlow controller and the key-value store) would ideally be located in the same network as the end-users. A single, widely reachable OCC would be able to coordinate caching amongst a number of OCN instances (i.e. caches). It is important to note that it is not a requirement that OpenFlow be deployed throughout the network; the connecting network hardware could also be entirely non-OpenFlow. The only specific OpenFlow switch requirement is at the last hop, closest to the user. We propose that OCNs are connected directly to OpenFlow switches on which clients are also attached. This deployment would offer the lowest latency and fastest response time, and thus ensure higher QoE for the end-users. However, the use of multiple OCNs at different points in the network (e.g. attached to aggregation switches in an enterprise or University campus network) is also entirely feasible. In fact, a hierarchical approach has the potential to provide further benefits if a greater proportion of requests can be fulfilled without leaving the LAN.

## A. *The OCC's Functionality and Interfaces*

The OpenCache Controller (OCC) is the main orchestrator of the in-network caching functionality that OpenCache provides, and implements the following four main operations :

**1) Receives requests for content of interest** that should be cached in the network's OCNs (Fig. 2), by exposing a JSON-RPC based API (Table I). Network administrators or content providers invoke the methods provided by this interface, using the appropriate authentication credentials, to declare what content should be cached from this point on in the network. If there is a request for certain content to be cached (using the *start-expr* method) the OCC ensures that this interest is stored in the key-value store and that all the OCNs are initialised and aware of the content that was requested for caching. In addition, the OCC will interact with the network's controller and instruct it to add the matching OpenFlow redirecting rules for the cacheable content in the OpenFlow switches of the network. These flows ensure that all the users' requests for that content are redirected to their closest OCN. If there is a request to stop caching content previously added (using the *stop-expr* method), then the OCC updates the key-value store appropriately and ensures that all the matching content and flows are removed from the networks' OCNs and OpenFlow switches, respectively. Finally, the OCC exposes a *list-expr-all* method over its API that lists to the requester all the content that has been requested for caching.

It is important to emphasise the granularity and flexibility that this interface provides: a regular expression-like syntax can be used to define parameters in the *start-expr* and *stop-expr* methods. The level of granularity is only constrained by the matching capabilities of OpenFlow. This allows administrators to fine tune their requests. For example, a request can be made for a specific video to be cached, or the videos of a whole domain or even a certain type of video from any domain.

**2) Implements the caching logic** which dictates what content should be cached and to which OCN at each point in time. OpenCache's ability to control the caching logic centrally on the OCC allows the network administrator to program and deploy their desired caching behaviour. This request can be based on the

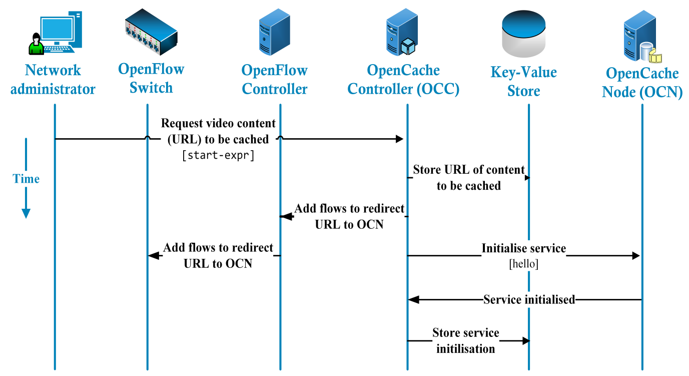

Network administrator
OpenFlow Switch
OpenFlow Controller
OpenCache Controller (OCC)
Key-Value Store
OpenCache Node (OCN)
Time
Request video content (URL) to be cached [start-expr]
Store URL of content to be cached
Add flows to redirect URL to OCN
Add flows to redirect URL to OCN
Initialise service [hello]
Service initialised
Store service initilisation

Fig. 2: Declaring Content of Interest as Cacheable

TABLE I: Interface for Declaring Content of Interest

| Method | Parameters | Result |
|---|---|---|
| start-expr | { "expr" : <expr> } | <boolean> |
| stop-expr | { "expr" : <expr> } | <boolean> |
| list-expr-all | none | [ {"expr" : <expr>, "port" : <port>}, ... ] |

TABLE II: Interface to Interact with OCNs

| Method | Parameters | Result |
|---|---|---|
| hello | { "host" : <host>, "port" : <port> } | <node-id> |
| keep-alive | { "node-id" : <node-id> } | <boolean> |
| goodbye | { "node-id" : <node-id> } | <boolean> |

specific parameters that he wants to optimise in his network. For example, an administrator could program the caching logic to minimise the streaming latency of recorded video lectures on a University's network, or to implement the pre-caching of popular content closer to end-users overnight, when the network is underutilised. It is important to note the ease and speed at which the network administrator can actually implement the caching logic with the use of OpenFlow in OpenCache.

**3) Manage the available OCNs' resources** in the network. An important part of resource management is to be able to handle the addition and removal of caches in a network dynamically. For this reason, the OCC exposes another JSON-RPC based API that allows the communication of the OCC with a number of OCNs (Table II). When an OCN is added to the network, it invokes the *hello* method to let the OCC know that it is now available on the network. In turn, the OCC replies with a *node-id* that is assigned to this particular OCN. From that point on, the OCN will periodically send a *keep-alive* message to indicate that it is still in the network and functioning correctly. If the OCC does not receive a *keep-alive* call from an OCN every 15 seconds (a configurable option), then it assumes that the OCN is not reachable, either because of network congestion or because it has been taken offline. Consequently, the OCC will remove the "unreachable" OCN from the list of caching resources that it has at its disposal. Alternatively, an OCN may also leave the network gracefully with the transmission of a *goodbye* message (Table II).

**4) Manage and maintain the OpenFlow flows in the network dynamically** via a Flow Pusher API that an OpenFlow controller provides (e.g. [27]). The OCC defines dynamically the appropriate flows that should be in the OpenFlow switches, so that each user's request gets redirected to an OCN in his vicinity (Fig. 2). With the management of flows, the OCC also propagates the caching logic to the network (e.g. expire content, perform load balancing or pre-caching). This also ensures that the caching and distribution functionalities remain purely in the network and are fully transparent to end-users.

### B. *The OpenCache Node's Functionality*

The OpenCache Node (OCN) is responsible for caching the appropriate video content, and delivering it to users if they request it. When the OCN comes online in a network, it communicates with the OCC (via the interface in Table II) and makes its resources available to it. Subsequently, when the OCN obtains a *node-id* from the OCC, it initialises its operation and awaits users' requests.

When a user makes a video request, and if the content has been declared as cacheable, the request will be received by the closest OCN to the user. This is in contrast to it traversing the external link, which would be the case without OpenCache in place. This is possible due to the pre-populated rules installed in the OpenFlow switches when declaring content of interest (Fig. 2). Following these rules, an OpenFlow switch redirects the user's packets appropriately. When the OCN receives such packets, it examines if it has the requested content already cached. If the particular video is not cached in the OCN (a cache-miss scenario depicted in Fig. 3), the OCN requests the video from the original VoD server. Once the first packet of this flow is received, the OCN will begin forwarding these back to the client. This process is intended to reduce any latency induced by the caching process. The delivery of this content traverses the OpenFlow switch too, and additional rules ensure that the packet received by the client appears to be from the expected source. With the completion of this process, the session has remained transparent to the user and there is no interruption to the service. Once the full flow has been handled in this way, the payload of the delivery is stored by the OCN in order to serve subsequent

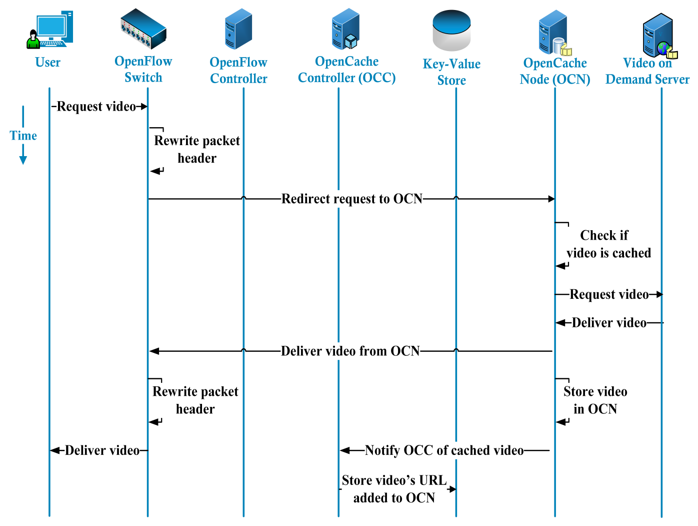


Fig. 3: Cache-miss Scenario

requests. Furthermore, the OCN informs the OCC of this transaction for resource provision and management purposes.

If the content for a particular request is already stored in the OCN (a cache-hit scenario), the OCN delivers that video directly to the user in a transparent fashion. As in a cache-miss scenario, the content always appears to originate from the VoD server the client originally requested it from. However, in a cache-hit scenario, no traffic would have left the user's network into other networks, thus saving external link utilisation and significantly reducing start up and buffering delays. The role of the OCN is such that there is an inherent need to have multiple instances of it distributed in the network to facilitate users' requests and content caching as efficiently as possible.

## V. Key SDN Benefits

By using Software Defined Networking (SDN), and OpenFlow in particular, we are afforded a number of key abilities that OpenCache exploits to their fullest extent. Most critically to the operation of OpenCache is the ability to transparently redirect requests for content to a running cache instance, as mentioned previously. More specifically, this is achieved by rewriting the packet header information, and intentionally forwarding it towards a cache or a client. The use of OpenFlow to achieve this functionality allows the rewriting procedure to happen in a distributed fashion on the switches of the network, and hence removes the need to overload a specific server with the burden of doing so. It also ensures full user transparency as the content that reaches the client appears to originate from the origin server rather than the cache. It is important to note that this is all possible without the costly and time-consuming modification of existing delivery techniques (e.g. caching or proxy servers, middleboxes) or end-client devices.

To compliment this, SDN provides both hardware and software abstraction to our in-network caching service. OpenCache can be used on commodity hardware, without the need to perform complex configuration and setup. The only requirement for OpenCache to function is the presence of a single-piece of OpenFlow-capable hardware on the path from client to the server. The ability to perform the necessary operations anywhere in the network where OpenFlow is supported grants OpenCache with even greater flexibility. For example, and as presented in the paper, we can do this close to the user in a last mile environment, where traditionally there has existed no such cache process. However, although not discussed here, the exact same process can be applied in other situations and environments without modifying OpenCache or OpenFlow itself; the same logic is applicable.

Furthermore, utilising OpenFlow also allows us to monitor the networking hardware contained within our topology, and use this feedback in OpenCache itself. This gives OpenCache a perspective of the network not typically afforded to application-layer technologies. This information can be used in conjunction with the programmability that OpenFlow permits to effectively satisfy any caching requirements. This is possible without the need to consider the peculiarities of differing hardware devices in the network. In addition, monitoring information, when used in conjunction with the redirecting action described previously, allows us to load balance requests on-the-fly and in real-time. This level of reactivity has not been previously possible, particularly through the use of a single, unified API.

## VI. Evaluation

In order to evaluate the efficacy of OpenCache, we carried out a number of VoD streaming experiments over a large-scale pan-European OpenFlow testbed provided by the OFELIA project [28]. OFELIA is composed of a number of OpenFlow-capable hardware switches and virtualised computing resources located in many different countries across Europe. Each site (or "island") is connected as to produce a large federated experimentation environment. For our experiments, a video client runs on a virtual machine located at the ETH Zurich OpenFlow island in Switzerland, where we also deployed an OpenFlow controller, an OCC and an OCN. Two VoD servers were deployed on virtual machines on two different OFELIA islands; at CREATE-NET in Italy and at i2CAT in Spain. Finally, the three islands are federated together using an OFELIA island in Belgium.

Three main metrics are used to evaluate the effectiveness of OpenCache; start up delay (key QoE metric [4], [5], [6]), external link utilisation and video playback bitrate. For all of metrics, we use the same evaluation environment; a VLC client in Switzerland accesses the same reference video content ("Big Buck Bunny"[1]) hosted at one of the VoD servers located in either Italy or Spain. The content is streamed using an adaptive video streaming technology, namely, DASH (Dynamic Adaptive Streaming over HTTP) [29]. MPEG-DASH facilitates the dynamic adjustment of the streaming bitrate by offering various bitrate encodings of the reference video, fragmented into fixed time chunks. MPEG-DASH aims to improve the overall QoE for the end-users by dynamically matching the bitrate requested to the available bandwidth.

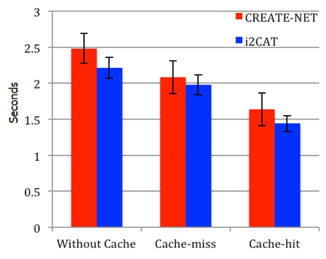
We carried out 20 video streaming requests from the video client to each of the VoD servers, in three scenarios; without a cache (as a baseline), a cache-miss (where content was not found on the OCN, and thus fetched) and a cache-hit (where the content was found and delivered from the OCN). These scenarios produced six unique sets of results (3 per island), which are shown averaged in Table III. For each experimental run, we record three metrics. First, the start up time of the VLC video player when playing back the reference video asset. Second, we recorded the number of cache-miss and cache-hit events on the OCN and the respected fetched and served bytes. This metric essentially demonstrates OpenCache's reduction in external link utilisation and origin server load, which consequently relinquishes resources for use by other users. Third, we recorded the video bitrate of each chunk requested by the client during the playback of the whole video.

The results of our experiments are shown in Table III, Fig. 4a, Fig. 4b and Fig. 5. They clearly demonstrate that OpenCache has reduced the start up delay for clients up to approximately 35%. This is reinforced by relatively low standard deviation values (taking into account that four sites across Europe are involved), which demonstrates a high level of statistical confidence. It is

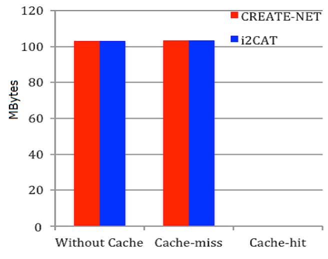
[1]http://www.bigbuckbunny.org/

TABLE III: VoD Streaming Experimental Results

| | CREATE-NET (Italy) | | | i2CAT (Spain) | | |
|---|---|---|---|---|---|---|
| | Without Cache | Cache-miss | Cache-hit | Without Cache | Cache-miss | Cache-hit |
| Average Start Up Delay (s) | 2.484 | 2.088 | 1.639 | 2.212 | 1.982 | 1.441 |
| Improvement over baseline (%) | - | 16.02 | 34.02 | - | 10.40 | 34.85 |
| Standard Deviation ($\sigma$) | 0.208 | 0.225 | 0.226 | 0.145 | 0.138 | 0.109 |
| External Link Usage (MB) | 105.7 | 105.8 | 0 | 105.7 | 105.8 | 0 |



(a) Average Start Up Delay



(b) Ext. Link Utilisation

Fig. 4: Experimental Results

particularly interesting to note that we have improved the situation even when the OCN has to fetch the content (compared to the baseline), attributed to the download technique used by the OCN.

Results are similarly optimistic for the external link usage. When the content is stored on the OCN, it is delivered directly to the client and the external link usage gets reduced 100%, essentially to zero bytes. Indicatively, streaming the full 9:56 minute reference video from the OCN, saves approximately 101 MB transfer just for one client session. Without an OCN or with a cache-miss, we observe the full content traversing the external link.

A further advantage that we observed when OpenCache is present, is that the playback client estimates that more bandwidth is available between the client and the OCN. As a result, it requests higher quality video chunks when content is delivered directly from the cache. This is illustrated in Fig. 5, which graphs the bitrate of the requested chunks over the playback of the entire video, with and without OpenCache. It is clearly illustrated that in the case where OpenCache is present, the player requests a bitrate which is over 8 times higher. This is a direct improvement in the quality of the streaming video and consequently the QoE for the end-user. It is important to note that the bitrate achieved with OpenCache present (i.e. 8000kbit/s) is the greatest bitrate available in this case. Evidently, the aforementioned QoE multiplier will be dependant on network conditions and should only be seen as an example. Nonetheless, OpenCache clearly provides the potential to improve the end-user's QoE by increasing the video quality distributed to the client.

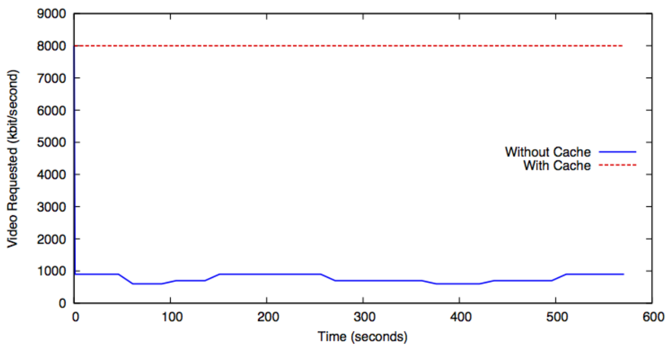


Fig. 5: Quality Requested by MPEG-DASH Client

Finally, it is important to note that the evaluation environment used within OFELIA is one which is relatively bandwidth-rich and well-connected and not necessarily representative of a typical domestic user. Nonetheless, even in such a rich environment, the OpenCache evaluation results show promising benefits to both the network and the QoE of the end-users.

## VII. Conclusion

As part of our future work, we plan to improve OpenCache by exploring the cache placement problem in conjunction with different caching policies discussed in related research work. We hope to identify the beneficial position of being in close proximity to end-users whilst maximising the use of the caching resources. In addition, we plan to extend OpenCache's functionality and evaluate it further in different environments and against other commercial or research based caching services.

In this paper we presented OpenCache: an efficient, transparent and highly configurable OpenFlow-assisted in-network caching service for VoD streaming. OpenCache aims to address the underlying challenge that the network faces when the same video files are streamed to end-users repeatedly using independent unicast flows. OpenCache provides the following key benefits :

1) Provides *cache as a service* by offering an interface to declare cacheable content of interest in an open, highly configurable and flexible manner.
2) Supports centrally controlled caching that provides the forum for many additional services to be programmed on top of it with ease (e.g. load balancing, pre-caching, different expiration policies etc.).
3) Is easily deployable in a production network; there are no changes required in the underlying delivery video mechanisms and all existing hardware and software can be retained.

4) Is fully transparent to the end-user; the user does not need to install any extra software, or have to sacrifice any of his local network or storage resources to stream video content with high efficiency, which other technologies require [21], [20].
5) As demonstrated from inter-island experiments on a pan-European testbed using adaptive video streaming, OpenCache provides caching very close to the user with three important benefits. Firstly, the external link usage gets reduced 100% and the network utilisation gets improved as end-user requests are now served locally. Secondly, OpenCache reduces the distribution load from the VoD content provider and all the transient networks along the path of the VoD server to the end-user. Thirdly, since the content is served locally, the video client observes higher throughput, lower latency, higher video quality and smaller start up and buffering times, eventually leading to higher QoE for the end-user.

## References

[1] "Visual Networking Index: Forecast and Methodology, 2012-2017," CISCO, Tech. Rep., May 2013.

[2] "Visual Networking Index: Forecast and Methodology, 2011-2016," CISCO, Tech. Rep., May 2012.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[4] S. S. Krishnan and R. K. Sitaraman, "Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs," in *ACM SIGCOMM IMC 2012*, 2012, pp. 211–224.

[5] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the Impact of Video Quality on User Engagement," in *ACM SIGCOMM 2011*, 2011, pp. 362–373.

[6] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A Case for a Coordinated Internet Video Control Plane," in *ACM SIGCOMM 2012*, 2012, pp. 359–370.

[7] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai Network: A Platform for High-performance Internet Applications," *SIGOPS OS Rev.*, vol. 44, no. 3, pp. 2–19, Aug. 2010.

[8] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," in *19th IEEE INFOCOM 1999*, vol. 3, 1999, pp. 1310–1319 vol.3.

[9] J. V. D. Merwe, S. Sen, and C. Kalmanek, "Streaming Video Traffic: Characterization and Network Impact," in *Int. Web Content Caching and Distribution Workshop*, 2002.

[10] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An Analysis of Live Streaming Workloads on the Internet," in *4th ACM SIGCOMM IMC 2004*, 2004, pp. 41–54.

[11] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards Network-wide QoE Fairness Using Openflow-assisted Adaptive Video Streaming," in *ACM SIGCOMM 2013 Workshop on Future Human-centric Multimedia Networking (FhMN)*, 2013, pp. 15–20.

[12] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study," *ACM Trans. MCCA*, vol. 4, no. 2, pp. 16:1–16:22, 2008.

[13] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network Characteristics of Video Streaming Traffic," in *7th ACM CoNEXT 2011*, 2011, pp. 25:1–25:12.

[14] C. Diot, B. Neil, L. Bryan, and K. D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network*, vol. 14, pp. 78–88, 2000.

[15] L. Golubchik, J. Lui, and R. Muntz, "Reducing I/O Demand in Video-on-demand Storage Servers," *ACM SIGMETRICS Performance Evaluation Review*, vol. 23, no. 1, pp. 25–36, May 1995.

[16] C. Aggarwal, J. Wolf, and P. Yu, "On Optimal Piggyback Merging Policies for Video-on-demand Systems," in *ACM SIGMETRICS 1996*, 1996, pp. 200–209.

[17] D. Eager, M. Vernon, and J. Zahorjan, "Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand," in *SPIE Multimedia Computing and Networking 2000*, 2000, pp. 206–215.

[18] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-demand Systems," in *ACM SIGCOMM 1997*, 1997, pp. 89–100.

[19] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-demand Services," in *6th ACM MULTIMEDIA 1998*, 1998, pp. 191–200.

[20] J. Pouwelse, J. Taal, R. Lagendijk, D. H. J. Epema, and H. Sips, "Real-time Video Delivery using Peer-to-Peer Bartering Networks and Multiple Description Coding," in *IEEE Int. Conference on Systems, Man and Cybernetics 2004*, vol. 5, 2004, pp. 4599–4605 vol.5.

[21] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec, "Peer-assisted Content Distribution in Akamai Netsession," in *13th ACM SIGCOMM IMC 2013*, 2013, pp. 31–42.

[22] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," in *7th ACM SIGCOMM IMC 2007*, 2007, pp. 1–14.

[23] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy, "Measurement and Analysis of a Streaming-media Workload," in *3rd USENIX USITS 2001*, 2001, pp. 1–1.

[24] Squid Proxy Server, http://www.squid-cache.org/.

[25] Netflix Open Connect Platform, http://www.netflix.com/openconnect.

[26] B. Niven-Jenkins, F. L. Faucheur, and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement," IETF RFC 6707, Sep 2012.

[27] The Floodlight Controller, http://floodlight.openflowhub.org/.

[28] A. Köpsel and H. Woesner, "OFELIA: Pan-European Test Facility for OpenFlow Experimentation," in *ServiceWave*, 2011, pp. 311–312.

[29] ISO-IEC 23009-1:2012 Information Technology, "Dynamic Adaptive Streaming over HTTP (DASH)."