20-05-2014

Open Call Deliverable OCC-DS2.1 Design of a Multi-site OpenFlow-assisted Video-on-Demand Distribution Architecture (CEOVDS)

Open Call Deliverable OCC-DS2.1

Grant Agreement No.:	605243
Activity:	NA1
Task Item:	10
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	Lancaster University
Document Code:	GN3PLUS14-1212-46
Authors:	Panagiotis Georgopoulos, Matthew Broadbent, Nicholas Race

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 605243 (GN3plus).

Abstract

This deliverable describes the design of a multi-site OpenFlow-assisted Video-on-Demand (VoD) distribution service that aims to address the challenges that networks face when large video files are streamed to end-users repeatedly. OpenCache, our devised VoD distribution service, uses OpenFlow to provide transparent, flexible and highly configurable video content caching, developed for multi-site operation. In this deliverable, we describe the requirements for high-quality video streaming and efficient in-network caching that our approach aims to satisfy. Furthermore, we discuss the powerful interfaces that we designed for interaction with OpenCache, and for the flexible intercommunication of its components. Finally, we emphasise the key benefits that a Software Defined Networking solution can yield in this problem domain, and highlight the key benefits that OpenCache aims to provide.

Table of Contents

Document Revision History

Version	Date	Description of change	Person
0.5	11-04-14	Full draft issued (ready for internal review)	Panagiotis Georgopoulos, Matthew Broadbent, Nicholas Race
	20-05-14	Internal Review	Afrodite Sevasti
1.0	09-06-14	Final version	Panagiotis Georgopoulos
1.1	04-03-15	Final version (added document codes)	Panagiotis Georgopoulos
		Review	
		Approved	

Executive Summary

Over the last 15 years, the Internet has had to support a significant growth in the distribution of video content. Globally, Internet video traffic was 57% of all consumer Internet traffic in 2012 and is predicted to be 69% in 2017 [Cisco-2012a, Cisco-2013]. Storage and caching servers have traditionally been used to store web content (e.g. HTML pages, images, web documents etc.) in an effort to improve the efficiency of content delivery in a network. Not surprisingly though, there is now a particular interest in being able to cache video inside a network itself.

With a Video-on-Demand (VoD) service, individuals can retrieve previously recorded content at a time after it was initially broadcast or made available. However, when huge content is delivered through independent unicast flows, naively ignoring that much of it is identical to transmissions hours or days earlier, network provision is greatly challenged. Mechanisms are therefore sought to reduce the impact of repeated delivery of identical content over the network, and to reduce the cost of delivery to the network operator. We call Software Defined Networking (SDN), and OpenFlow in particular, to the rescue.

To address the aforementioned VoD distribution challenge, we have been developing OpenCache; an OpenFlow-assisted VoD in-network caching service. Building on previous work, this deliverable reveals the requirements for high-quality video streaming and for efficient multi-site in-network caching. We then describe the core design and implementation principles of our approach that focus on satisfying the revealed requirements. OpenCache's functionality is based on powerful interfaces that we designed for providing intelligent and transparent caching with the use of OpenFlow in a multi-site environment (the GÉANT OpenFlow facility). The core entities of OpenCache, namely the OpenCache Controller and the OpenCache Nodes, their operation and intercommunication are also described in detail in this deliverable. Finally, we emphasize the benefits that an SDN based solution can bring to this problem domain, namely, the high transparency, programmability and flexibility of our approach. OpenCache focuses on monitoring and optimising important VoD delivery metrics, such as buffering time, throughput, and video quality. We aim to use OpenCache's monitoring functionality to optimise its performance, and eventually target to improve the overall network utilisation whilst simultaneously improving the Quality of Experience perceived by the end-user during VoD streaming.

1 Introduction & Motivation

Recent years have seen a huge growth in the popularity of video streaming, for both live and on-demand services in wired and mobile end-devices (Figure 1, Figure 2). In 2012, Internet video traffic accounted globally for 57% of all consumer Internet traffic [Cisco-2012a], and is predicted to increase even more, to 69% by 2017 [Cisco-2013]. There is a similar prediction for mobile devices (Figure 2), where approximately 69% of the overall mobile traffic is predicted to be video by 2018 [Cisco-2014]. 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 [Cisco-2012b]. 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 [Cisco-2012b]. By 2016, HD Internet video will comprise 79% of VoD [Cisco-2012b]. There is no doubt that high quality online video streaming has become an essential part of consumers' every-day life.

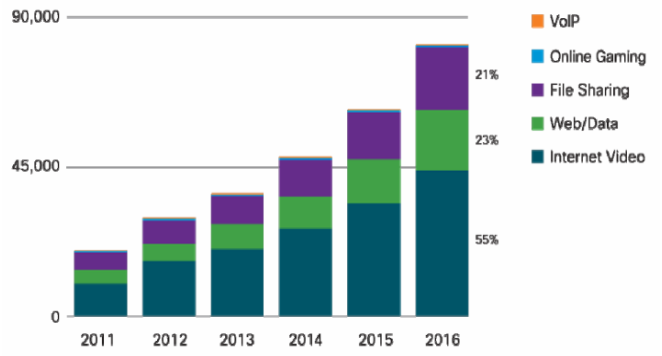

Figure 1 : Global consumer Internet traffic in petabytes per month [Cisco-2012a]

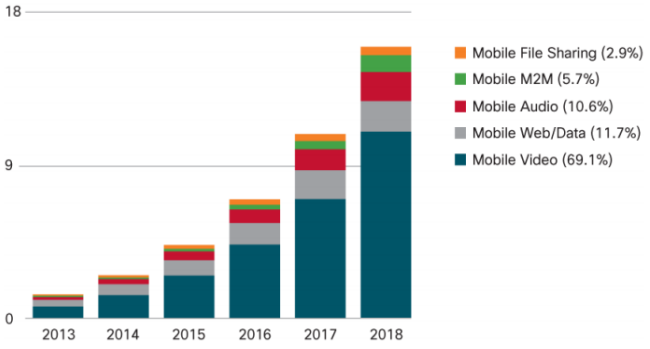

Figure 2 : Mobile consumer Internet traffic in exabytes per month [Cisco-2014]

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.

The distribution of live content involves the streaming of data to all target users simultaneously, so it is able to exploit multicasting mechanisms that provide such simultaneous delivery. In contrast, **VoD requests** must be handled individually, leading to an ***independent media flow*** in the distribution network ***for each user request***. Using such a unicast content delivery paradigm, an ***enormous amount of identical media objects*** (in the

order of gigabytes for each HD film) is likely to be delivered on the same network segments ***repeatedly***. This is the problem statement behind our GN3plus work, as 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; we call Software Defined Networking (SDN) to the rescue.

The main focus of the CEOVDS Open Call project is to improve the VoD distribution efficiency. To this effect, during the course of CEOVDS we aim to design and evaluate a multi-site OpenFlow-assisted Video-on-Demand distribution service based on transparent in-network caching, onwards called OpenCache. Lancaster University participated in the OFELIA EU FP7 project [OFELIA], where we designed a prototype single-site architecture, which offers in-network intelligent and transparent VoD caching with the use of OpenFlow [McKeown]. In the name of OpenCache, our work in GN3plus is to extend the OFELIA developed VoD architecture, to facilitate efficient caching and distribution of VoD flows that have to traverse multiple geographically distributed sites. The GÉANT OpenFlow Facility (GOFF) [GOFF-Manual] provides the ideal realistic setting for development and experimentation of such an OpenFlow based service, as it provides a distributed multi-site facility over the Internet.

OpenCache aims to provide a transparent, flexible and highly configurable in-network caching service for VoD streaming. OpenCache uses Software Defined Networking to provide ***cache as a service*** for media content in an efficient and transparent fashion. By leveraging SDN, and OpenFlow [McKeown] in particular, we provide a control plane that orchestrates the media caching and distribution functionalities, and transparently pushes the content as close to the end-user as possible without requiring any changes to the delivery methods or the end-hosts. This way, OpenCache provides high quality VoD streaming.

Our approach, building an SDN based in-network caching service, aims to achieve three important contributions, which we plan to demonstrate during the CEOVDS project. First, it aims to improve network utilisation and minimise 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 aims to minimise 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 [Krishnan, Dobrian, Liu].

To this effect, this report extends previous work and focuses on discussing the requirements and extended design of an OpenFlow-assisted Video-on-Demand Distribution service (i.e. OpenCache) to run multi-site (over the GOFF). Section 2 describes related work that tries to address the VoD content distribution problem. Section 3 details the requirements for high-quality video streaming end-to-end that a potential solution should aim to satisfy. In addition, Section 3 discusses the requirements that our solution should satisfy for multi-site VoD in-network caching. Section 4, describes the design of OpenCache, including its core components and the developed APIs for their intercommunication and efficient functionality. Finally, Section 5 highlights the key benefits that an SDN based solution can yield in this problem domain. Finally, we conclude this deliverable with the benefits that OpenCache aims to provide for multi-site in-network caching of VoD content.

2 Can SDN Provide a Solution? A Dive into Related Work

The popularity of video streaming over the Internet, coupled with the users' requirement for high quality streaming and high Quality of Experience, has managed to capture the attention of the research community. Over the years, multiple solutions have been developed to provide high-quality end-to-end video streaming, each with their own advantages and disadvantages. In this section, we discuss related work and how we think SDN can contribute to this problem space.

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 [Diot], 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 [Diot, Aggarwal], holding a portion of the clients' bandwidth in reserve [Eager], or requiring from the clients the receipt of two or more video streams simultaneously, each at the playback rate [Hua-a, Hua-b, Eager]. 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 [Pouwelse]. 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 [Zhao].

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. [Cha] demonstrates the 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 [Chesire], 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 [Squid]) 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. Typically, they leave 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 [Nygren, Zhao]. 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 middle or 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. 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 their administration and maintenance cost, Akamai introduced a hybrid CDN-P2P based solution, that complements their service by pushing content closer to end-users [Zhao]. 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.

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 [McKeown], 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 a quick, easily deployable and flexible manner. With the centralised network perspective that OpenFlow provides through its controller, an experimenter has an overarching view of the current network status and has the ability to programmatically introduce new 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, vendor-agnostic and transparent manner. The transparency that we achieve through the use of SDN is particularly important as no end-client applications or media delivery services have to be updated.

3 Requirements

This section describes the requirements for high-quality end-to-end video streaming that a VoD distribution service should take into account (Section 3.1). Subsequently, Section 3.2 describes the functional and non-functional requirements that our own VoD distribution service has for multi-site operation.

3.1 Requirements for High-quality Video Streaming End-to-End

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 [Nygren, Sen, Merwe]. 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 [Nygren, Sripanidkulchai], this is a stark problem. This fragmentation means that content that is centrally hosted must travel over multiple networks to reach end-users. As this is the case, 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 lead to lower QoE [Krishnan, Dobrian, Liu, Sen, Merwe]. 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 [Wang, Sripanidkulchai, Sen, Merwe]. 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 [Nygren, Sen]. 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.

OpenCache aims to address the aforementioned challenges and ensure that the media content resides as close to the user as possible. Such an approach potentially ensures lower latency and higher throughput end-to-end, eventually leading to higher video quality and lower start up and buffering times [Liu, Krishnan, Dobrian].

3.2 Multi-site OpenCache Requirements

[OFELIA-D11.1] described the functional requirements of an OpenFlow-assisted VoD distribution service based on in-network caching. Although these requirements were focused on a single-site operation, they are applicable to a multi-site service as they mostly describe the essential functionality of a VoD in-network caching service. Therefore, we repeat these requirements here for completeness, and enhance them for more optimised functionality and multi-site operation.

For efficient VoD distribution, OpenCache should satisfy the following functional requirements :

- Should identify cachable content without any significant impact on the user's request.
- Should cache content transparently to the user with no or minimal performance impact (i.e. delay).
- Should deliver content transparently to the user with no or minimal performance impact (i.e. delay).
- Should retain the underlying content delivery mechanism to avoid fundamental changes to the service. Neither server nor client side applications should require modifications.
- Should be content agnostic and media independent.
- Should be easily integrated in a production network.
    - The case where not all network equipment is OpenFlow-enabled should be considered.
- Should be able to use multiple cache instances.
    - Multiple caches should be considered in both single and multi-site deployments.
- Should be able to add or remove cache instances without service interruption, irrespective of the cache's placement (e.g. located on the same site with the OpenFlow controller or a different site).
- Should provide monitoring functionality on its caching behaviour for both single and multi-site operation.

For efficient VoD distribution, OpenCache should satisfy the following non-functional requirements :

- Should optimize network utilization based on network and caching monitoring metrics. For example, it should not unreasonably cache content that is infrequently requested and thus increase network utilisation unnecessarily.
- Should adjust its run-time functionality and improve the users' QoE by maintaining a high level view of the network based on network and caching run-time metrics (e.g. user latency, buffering times, cache-hits/cache-misses etc.). This functionality is particularly important for operation across multiple-sites as, for example, removal of content from a particular cache or delivering content from a distant cache could deteriorate user's QoE even when compared to a scenario without a cache.
- Should identify requests for the same content stored on different sites for optimised storage and delivery.
- Should support load balancing between carefully and strategically located in-network caches for single or multi-site deployment.

- Should support pre-caching of popular content to enhance QoE and optimise VoD content distribution.

4 Multi-site OpenFlow-assisted Video-on-Demand Distribution Service

This section describes the design of our multi-site OpenFlow-assisted VoD distribution service; OpenCache. In particular, Section 4.1 builds on [OFELIA-D11.2] and discusses the extended design of OpenCache, its entities and their placement in a network. Subsequently, Section 4.2 details the main entities of OpenCache, namely the OpenCache Controller (OCC) and the OpenCache Node (OCN) and describes their extended functionality for multi-site operation. Section 4.2 describes the interfaces that we have designed for the intercommunication between the OpenCache entities, in order to achieve efficient and transparent in-network caching of video assets with the use of OpenFlow.

4.1 Design

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

OpenCache offers a powerful interface that provides ***cache as a service***. This interface is not intrinsically linked to a particular type of content (i.e. it is content-agnostic), 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, 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 OpenCache's designated interfaces. OpenCache exposes these interfaces through powerful and flexible JSON-RPC based APIs (discussed in Section 4.2), which allow VoD content to be cached closer to the end-user. This placement of cached content also increases the user's QoE when streaming it. 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 [CDNI-RFC], 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.

The initial design of OpenCache (described in [OFELIA-D11.2]) although focused on single-site operation, was generic enough to provide the architectural entities for multi-site operation. We carefully revised the functionality of all the entities involved, renamed them to match their new functionality and extended their interfaces (new additions will be described in Section 4.2). From a design's point of view, OpenCache's architecture includes the following components (Figure 3):

1. **OpenFlow Switch** : This could be any hardware or software based OpenFlow switch (e.g. Open vSwitch [OpenvSwitch] that is used in the GOFF) that has users connected to it. It must also be able to communicate with the VoD server(s), the OpenCache Caching Nodes (OCNs) and the OpenFlow controller, but not necessarily via a direct connection.

2. **Video-on-Demand Server :** This is a common VoD server that acts as the primary source for the video assets and could be located anywhere on the Internet reachable by its IP address. In scenarios running on the GOFF, the VoD server could be located in any of the PoPs (or even in a number of them), or even been placed on the Internet. If the cacheable content we are concerned with is not video, then the VoD server should be replaced by another type of server providing the appropriate source content.

3. **OpenFlow Controller :** This could be any kind of OpenFlow controller (e.g. Floodlight [Floodlight], NOX [NOX], POX [POX] etc.) and should be reachable by the OpenFlow switches of the network. The OpenFlow switches communicate with the OpenFlow controller using the OpenFlow protocol in order to add and maintain flows based on the applications' requirements.

   In our VoD caching scenario, we are using the Floodlight OpenFlow controller [Floodlight] for its performance and robustness. The controller behaves as a standard L2 Learning Switch: instructing the switch to forward packets based upon a learned MAC-to-port pairing. In addition to this behaviour, we have implemented a supplementary functionality into the controller to provide our OpenCache Controller (OCC) with alerts. These are generated when two types of events occur. Firstly, when content of interest is requested by the client, and secondly, when content of interest is delivered to the client. In this sense, the OpenFlow Controller monitors both incoming and outgoing traffic on the switch, and communicates (via JSON [JSON]) with our OCC when an alert is generated. The OpenFlow Controller also accepts rules pushed back from the OCC when, firstly, a copy of content "in-transit" is to be made to our local cache, or, secondly, a locally cached copy of content is to be delivered to the client.

4. **OpenCache Controller (OCC)** : The OpenCache Controller (OCC) is the orchestrator of the VoD caching and distribution functionalities and will be described in detail in Section 4.2.1.

5. **Database** : This database is used for maintaining a list of all the video assets that have been requested for caching, the ones that have already been cached and the specific cache(s) (i.e. OCNs) that these files are stored in. In addition, this database holds information about the cache(s) that operate in the network (i.e. are "online"), their location and their available resources. It is worth noting that the database is agnostic to the type of content that is being cached, thus making our architecture applicable to other cacheable content types.

   OpenCache uses mongoDB [mongoDB] as database (replacing the Redis key-value store in earlier versions of OpenCache [OFELIA-D11.2]), due to its open-source, document based and NoSQL nature. MongoDB offers great benefits compared to database systems, such as fast in-place updates, scalability, flexible aggregation and data processing, full index support and replication and high availability. The OCC regularly updates the information that mongoDB stores using the pymongo python library [mongoDB-PY].

6. **OpenCache Node(s) (OCN(s))** : The OpenCache Node(s) (OCN(s)) are essentially the cache entities of our VoD in-network caching service and will be described in detail in Section 4.2.2.

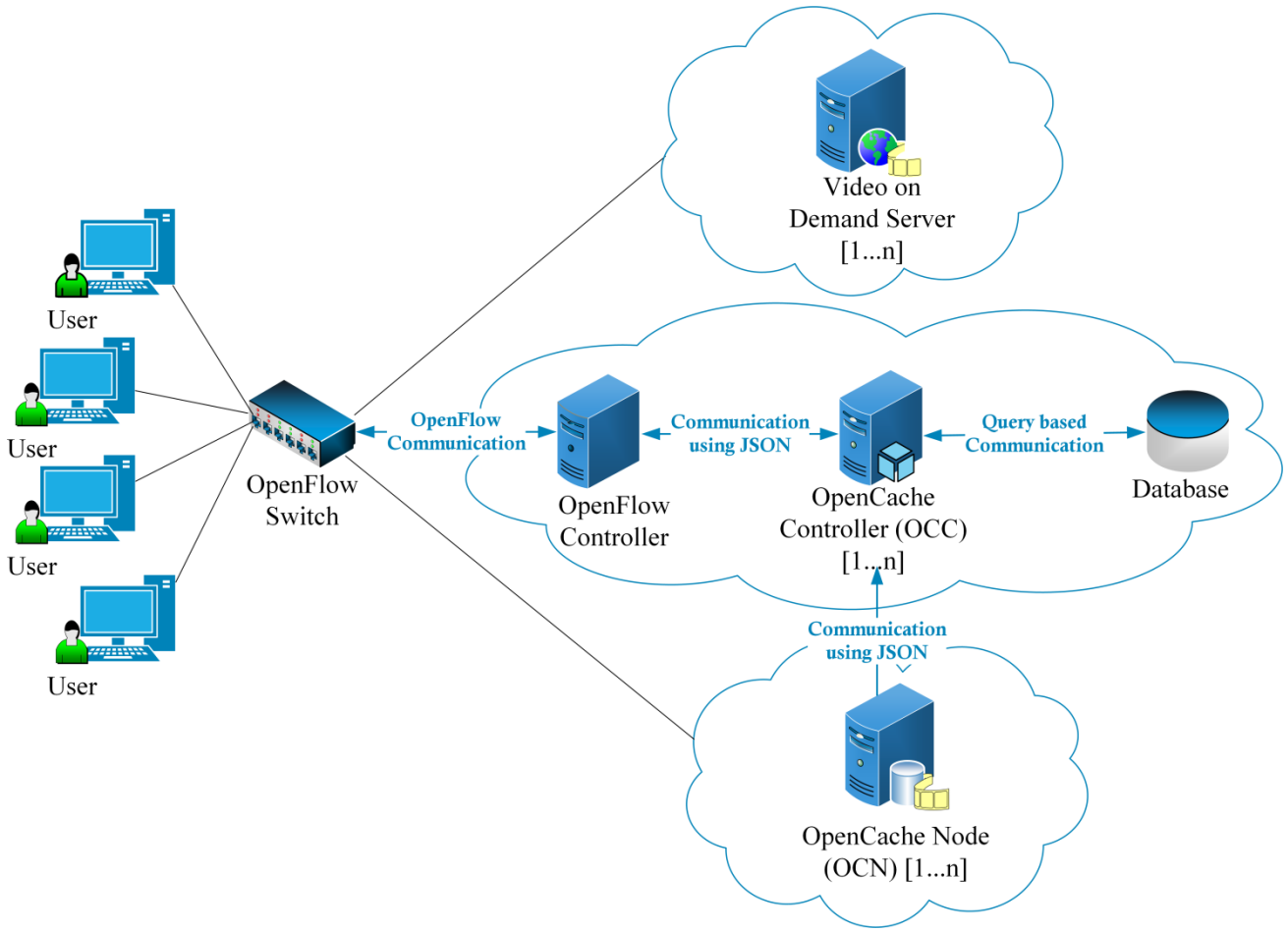

Figure 3 : OpenCache Architecture

With respect to the placement of the OpenCache's entities, the OCC, along with its partner elements (the OpenFlow controller and the database) 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. In the GOFF this is realised by deploying an OCN instance on the same VM as the end-user clients on a PoP. This deployment would offer the lowest latency and fastest response time, and thus ensure higher QoE for the end-users. However, generally speaking, the use of multiple OCNs at different points in a 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.

4.2 Core Entities and APIs

This section describes the two core entities of OpenCache, namely the OpenCache Controller and the OpenCache Node, and the APIs they expose to interact with network administrators and for their own intercommunication.

4.2.1 OpenCache Controller

The OpenCache Controller (OCC) is the main orchestrator of the in-network caching functionality that OpenCache provides, and implements the five fundamental functionalities that will be described in the following five sections.

4.2.1.1 *Handle Requests for Content of Interest*

The OCC receives requests for content of interest that should be cached in the network's OCNs (Figure 4), by exposing a JSON-RPC based API (Table 1). Network administrators or content providers invoke the methods provided by this interface, 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` method, Table 1) the OCC ensures that this interest is stored in the database and that all the OCNs are initialised and aware of the content that was requested for caching (Figure 4). 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 via the FlowPusher API [FlowPusher]. 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` method, Table 1), then the OCC updates the database appropriately and ensures that all the matching content and flows are removed from the networks' OCNs and OpenFlow switches, respectively.

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` and `stop` 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.

Apart from the `start` and `stop` functionalities, OCC's interface offers five additional new methods, compared to earlier versions of OpenCache [OFELIA-D11.2], presented in Table 1 (i.e. `pause`, `fetch`, `seed`, `refresh` and `stat`; the last two will be described in the next section). Through this OCC interface the network administrator has the ability to pause caching at given cache instances (i.e. OCN), temporarily stopping them from handling requests for content, but without removing the content from their storage. Pausing certain OCNs is beneficial for maintenance and administrator purposes where OCN resources have to be, for example, upgraded but without the need to remove all the already cached content.

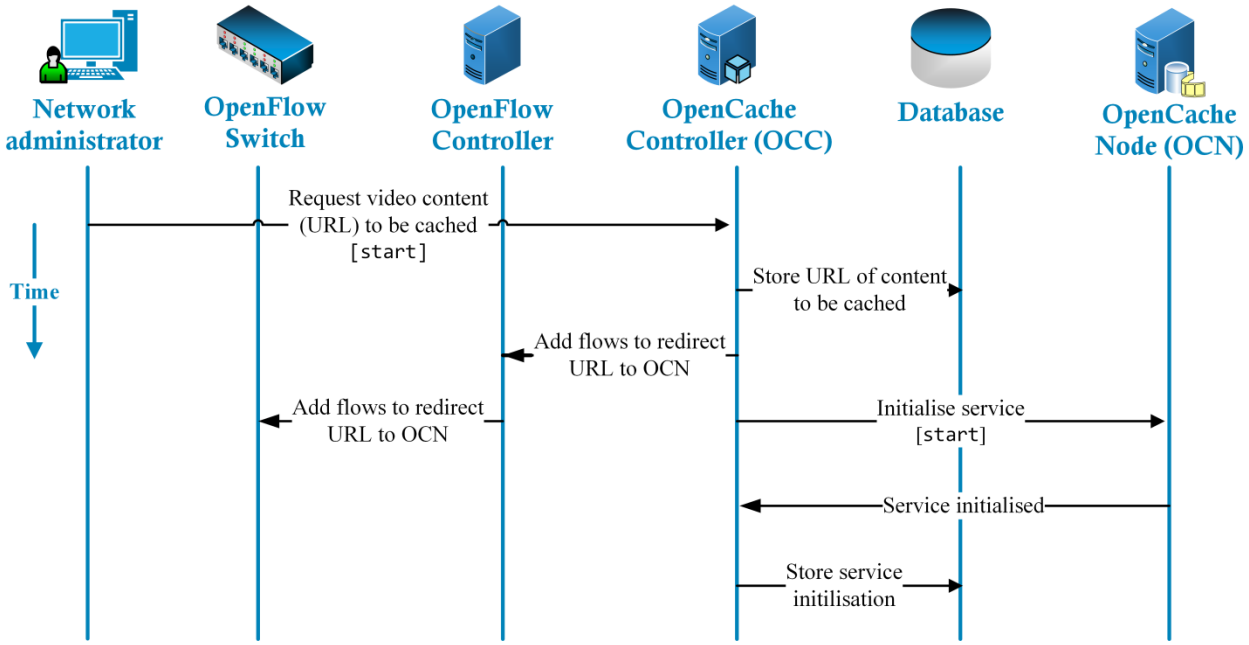

Figure 4 : Declaring Content of Interest as Cacheable

With the `fetch` method (Table 1), the network administrator can declare to the OCC that an OCN (or a set of them) should fetch certain content from a remote location, even though this content might not have been explicitly requested from a user. This method allows OpenCache to essentially implement pre-caching on the OCNs of popular content. This is a very important functionality as certain content could be pre-cached during periods where the network utilisation is low (e.g. overnight) and thus further improve the network utilisation on peak hours.

The purpose of the `seed` method (Table 1) is to map multiple web addresses with a specific video file in order to reduce storage utilisation. Therefore, the network administrator invokes the `seed` method in order to add to the OCNs (via the OCC) a number of fully resolved expressions (web addresses) that serve the same video file. All the expressions are equivalent to each other, in the sense of, instructing the OCNs to serve the same video content to the user irrespective of the web address the user requested that content from.

Table 1 presents the parameters that can be used when invoking the methods of the OCC's interface. Any parameter in brackets is optional. If a parameter is not included, then a wildcard all (*) is used by default. The `<node-id>` parameter refers to OCN instances and can either be a single node-id of an OCN, a vertical-bar-separated (|) list of node-ids, or a wildcard (*) for all existing OCNs. An `<expr>` can either be a single OpenCache expression, a vertical-bar-separated (|) list of OpenCache expressions, or a wildcard (*) to cache all HTTP traffic.

METHOD	PARAMETERS	RESULT
start	{ ("expr" : <expr>), ("node" : <node-id>) }	<boolean>
stop	{ ("expr" : <expr>), ("node" : <node-id>) }	<boolean>
pause	{ ("expr" : <expr>), ("node" : <node-id>) }	<boolean>
fetch	{ ("expr" : <expr>), ("node" : <node-id>) }	<boolean>
seed	{ ("expr" : <expr>), ("node" : <node-id>) }	<boolean>
refresh	{ ("expr" : <expr>), ("node" : <node-id>) }	<boolean>
stat	{ ("expr" : <expr>), ("node" : <node-id>) }	[<cache_hit>, <cache_miss>...]

Table 1 : JSON-RPC based Interface to Interact with the OpenCache Controller

4.2.1.2 *Report Aggregate Cache Statistics*

Through a newly developed interface, specifically designed during CEOVDS for OpenCache's multi-site operation, the OCC provides the ability to report aggregate cache statistics using the `stat` method (Table 1). Using this interface, each OCN periodically reports statistics back to the OCC that stores these centrally to reduce latency. To manually request fresh statistics from a given OCN, the `refresh` method can be used before the `stat` method.

The ability to monitor and report aggregate cache statistics is a particularly important function for cache optimisation and multi-site operation. For example, based on the statistics that the OCC provides, load balancing or pre-caching of popular content could be implemented. The `stat` method provides results for a variety of caching metrics developed for CEOVDS (presented in Table 2). Indicatively, the OCC monitors and could report the state of each OCN (`start`, `stop` or `pause` state), the OCNs' hard drive load, the number of cache hits or cache misses in event numbers or bytes, or the videos objects that are cached in each OCN.

FIELD	NOTE
start	Number of cache instances in “start” state
stop	Number of cache instances in “stop” state
pause	Number of cache instances in “paused” state
cache_miss	Number of cache miss (content not found in cache) events (one per request)
cache_miss_size	Number of bytes served whilst handling cache miss (content not found in cache) events
cache_hit	Number of cache hit (content already found in cache) events (one per request)
cache_hit_size	Number of bytes served whilst handling cache hit (content already found in cache) events
cache_object	Number of objects currently stored by the cache
cache_object_size	Number of bytes for the cached objects on disk (actual)
total_node_id_count	Number of unique node IDs present in results
total_expr_count	Number of unique expressions present in results
total_response_count	Number of unique responses seen
node_id_seen	List of those node IDs present in results
expr_seen	List of those expressions present in results
node_expr_pairs_seen	Pairs of nodes and expressions present in results, including current status and average load

Table 2 : Cache statistic metrics offered by OCC via a JSON-RPC based interface

4.2.1.3 *Implement the Caching Logic*

Since the OCC is the orchestrator of the in-network caching functionality, it is the central entity that is responsible to implement the caching logic. The caching logic essentially dictates what content should be cached and where (i.e. to which OCN) at each point in time. The focus of CEOVDS is not to define and implement the caching policies that could be used for a particular type of content, since these are network specific and also widely explored in the research community. The main goal and advantage of OpenCache, is to be able to provide (through its OCC) centrally controlled caching and allow the network administrator to program and deploy his caching logic at will. The OCC has not only the ability to monitor and report caching

statistics that can be used for the run-time change of the caching behaviour in the network, but also has the means to enforce this change to the caching instances and the network's OpenFlow switches.

The implementation of a caching logic could be based on the specific parameters that a network administrator 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.

4.2.1.4 *Manage the Caching Resources*

The OCC is responsible to manage the available OCNs' resources in the network. This includes two important functionalities. The first one concerns the ability to understand the status of the OCNs (if they are online or offline) and the second one concerns the propagation of the caching decisions to the OCNs. These are implemented using the methods of another JSON-RPC based interface that the OCNs expose (Table 3), developed during CEOVDS.

With respect to the first functionality, an important part of the caching resource management is to be able to handle the addition and removal of caches in a network dynamically. For this reason, the OCC exposes three methods (namely `hello`, `keep-alive` and `goodbye`) via a JSON-RPC based API that allows the communication of the OCC with a number of OCNs (Table 3). 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` message 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 3).

With respect to the second functionality, the OCC has the ability to control the caching occurring in a network by propagating the caching decisions to the OCNs. This functionality is accomplished with five additional new methods (Table 3) that we developed during CEOVDS for multi-site and multiple-caches operation. Essentially, the extended OCNs' interface provides new functionality that essentially matches the methods that a network administrator is able to invoke on the OCC itself. In particular, the OCC has the ability to invoke the `start`, `stop`, `pause`, `fetch` and `seed` methods to the particular OCNs that should act accordingly (described in Section 4.2.1.1). These methods essentially allow the OCC to propagate the caching decisions to specific OCNs, following the caching logic that the network administrator has determined. Finally, the OCNs provide a `stat` method that pushes information on all the caching metrics (presented in Table 2) of the particular OCN to the OCC. As discussed, the OCC invokes this method to receive caching statistics from all OCNs (proceeded by `refresh` if the latest statistics are needed) and then aggregates those to help the network administrator take informed decisions.

METHOD	PARAMETERS	RESULT
hello	{ "host" : <host>, "port" : <port> }	<node-id>
keep-alive	{ "node-id" : <node-id> }	<boolean>
goodbye	{ "node-id" : <node-id> }	<boolean>
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>) }	[<cache_hit>, <cache_miss>...]

Table 3 : JSON-RPC based interface to interact with the OpenCache Nodes

4.2.1.5 *Maintain the In-network OpenFlow Flow Entries*

The OCC is responsible to manage and maintain the OpenFlow flow entries in the network dynamically via the Static Flow Pusher API that the FlowVisor controller provides [FlowPusher]. According to the caching logic, the OCC dynamically determines the appropriate flows that should be in the OpenFlow switches, so that each user's request gets redirected to an OCN in his vicinity (Figure 4). With the management of flows, namely adding and removing OpenFlow rules on the switches, the OCC manages to propagate 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.

4.2.2 OpenCache Node (OCN)

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 using the `hello` method (Table 3) and makes its resources available to it. Subsequently, as a reply to its `hello` message the OCN obtains a `node-id` from the OCC, initialises its operation and awaits users' requests.

When a user makes a video request, and if the content has been declared as cacheable (using the process described in Section 4.2.1.1), 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 (Figure 4). Following these rules, an OpenFlow switch rewrites the packet header of the user's request and redirects it appropriately to the closest OCN. 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 5), 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 that rewrite the packet header 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 OCC in order to serve subsequent requests. Furthermore, the OCN informs the OCC of this transaction for resource provision and management purposes (Figure 5).

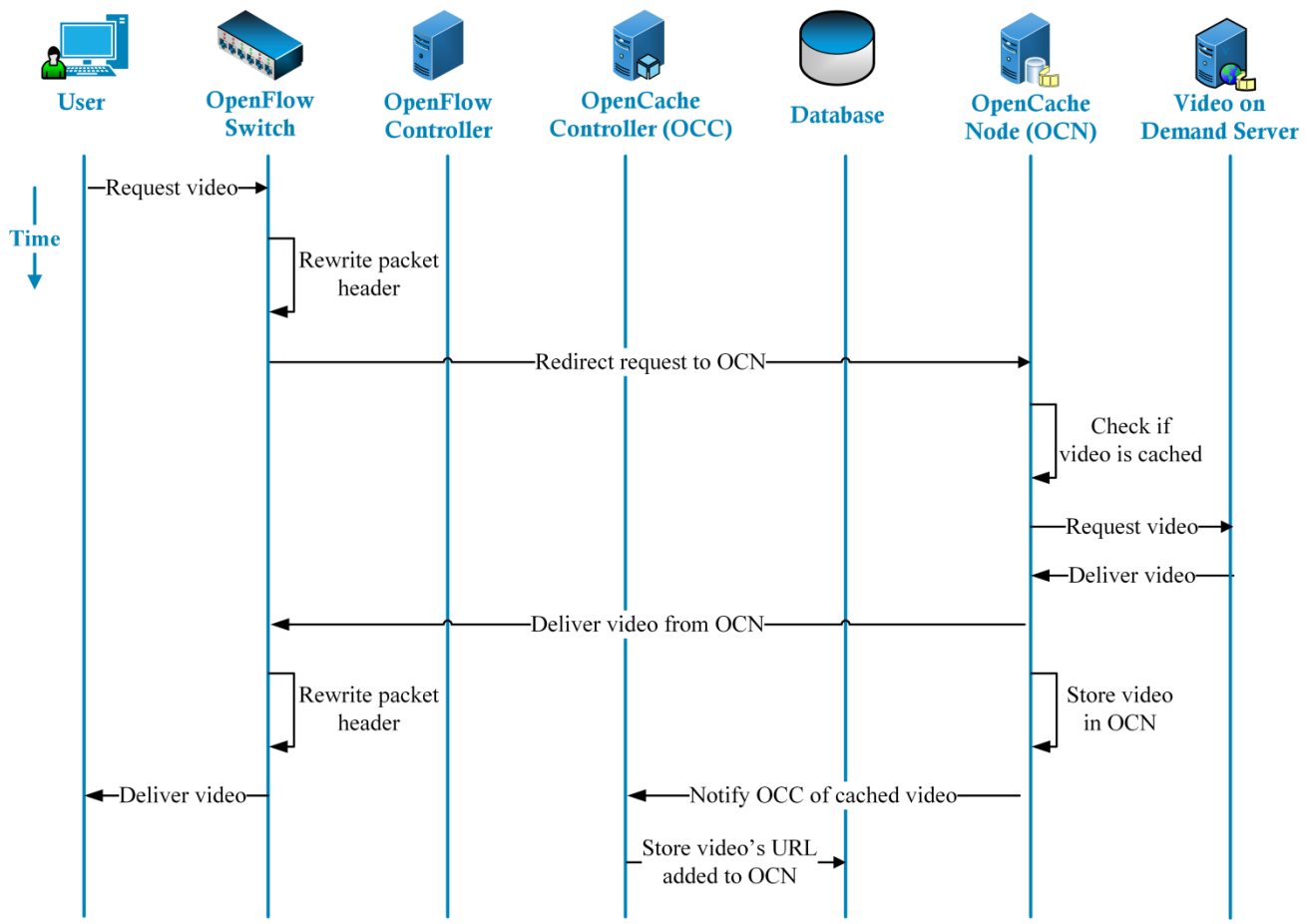

Figure 5 : A Cache-miss Scenario

If the content for a particular request is already stored in the OCN (a cache-hit scenario depicted in Figure 6), the OCN delivers that video directly to the user in a transparent fashion. As in a cache-miss scenario, the rules on the OpenFlow switch of the network rewrite the packets' headers so that the content always appears to originate from the VoD server the client originally requested it from. It has to be noted though that 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 (e.g. in multiple GOFF's PoPs) to facilitate users' requests and content caching as efficiently as possible.

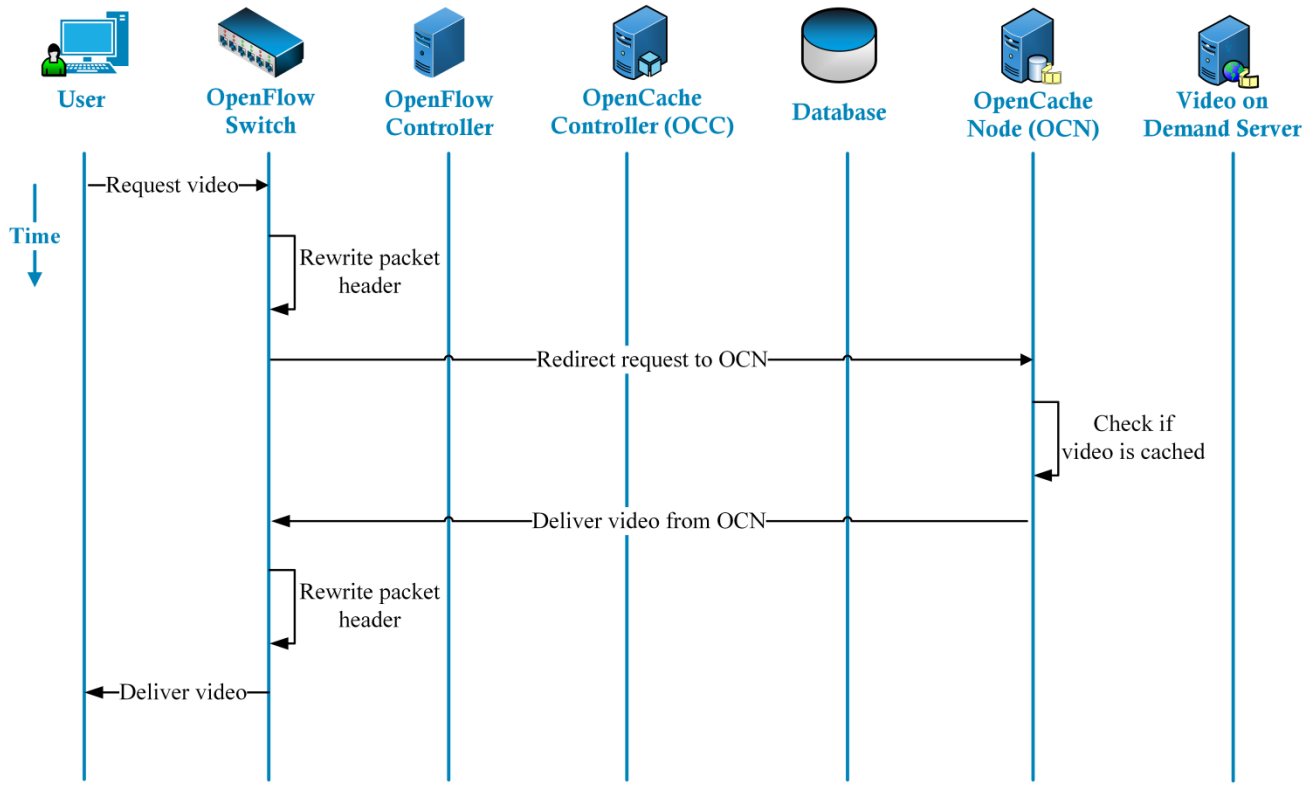

Figure 6 : A Cache-hit Scenario

5 Key Software Defined Networking Benefits

By using Software Defined Networking, 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. proxy servers, middleboxes) or clients.

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 (or even Open vSwitch [OpenvSwitch]) 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, we can do this close to the user 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 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.

Conclusion

In this deliverable, we described OpenCache; an OpenFlow-assisted VoD in-network caching service being designed and implemented for multi-site operation. OpenCache aims to bring an efficient, transparent and highly configurable caching service for VoD content and address the underlying challenge that the network faces when the same video files are streamed to end-users repeatedly using independent unicast flows.

During CEOVDS we extended the requirements set for single-site operation described in [OFELIA D-11.2] to match multi-site operation over the GOFF. In addition, we extended the previous design, functionality and implementation of all the OpenCache's entities. Most importantly, we enriched all OpenCache's interfaces in respect to monitoring caching and networking metrics (interfaces described in Tables 1, 2 and 3). In addition, CEOVDS augmented OpenCache's functionality to handle the addition and removal of multiple caches irrespective of their placement in GOFF, and also added functionality to pre-cache and load-balance multiple caches' contents.

OpenCache aims to provide the following key benefits :

1) Provide cache as a service by offering an interface to declare cacheable content of interest in an open, highly configurable and flexible manner.

2) Support 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 technology requires.

5) Provide caching very close to the user that allows the external link usage to get reduced and the overall network utilisation to be improved as end-user requests will now be served locally. Furthermore, since the content is served locally, the video client will observe higher throughput, lower latency, higher video quality and smaller start up and buffering times, eventually leading to higher QoE for the end-user.

Deliverable 2.2, entitled "Multi-site VoD Distribution Service Formal Demonstrator", is the follow up companion deliverable that aims to provide a demonstrator of the prototype service that was described in this deliverable. The two deliverables together achieve the main goals of CEOVDS; to design, implement and demonstrate a multi-site OpenFlow-assisted VoD in-network caching service.

References

[Aggarwal] C. Aggarwal, J. Wolf, and P. Yu. On Optimal Piggyback Merging Policies for Video-on-demand Systems. In ACM SIGMETRICS 1996, pages 200-209, 1996.

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

[Cha] 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, pages 1-14, 2007.

[Chesire] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy. Measurement and Analysis of a Streaming-media Workload. In 3rd USENIX USITS 2001, pages 1-1, 2001.

[Cisco-2012a] The Zettabyte Era – Trends and Analysis, Technical report, CISCO, 2012.

[Cisco-2012b] Visual Networking Index: Forecast and Methodology, 2011-2016. Technical report, CISCO, May 2012.

[Cisco-2013] Visual Networking Index: Forecast and Methodology, 2012-2017. Technical report, CISCO, May 2013.

[Cisco-2014] Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018. Technical report, CISCO, February 2014.

[Dobrian] 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, pages 362-373, 2011.

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

[Eager] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand. In SPIE Multimedia Computing and Networking 2000, pages 206{215, 2000.

[Floodlight] “Floodlight Is An Open SDN Controller” [Online]. Available: http://floodlight.openflowhub.org/

[FlowPusher] “Floodlight Controller: Static Flow Pusher API” [Online]. Available: http://www.openflowhub.org/display/floodlightcontroller/Static+Flow+Pusher+API+%28New%29

[GOFF-Manual] GÉANT OpenFlow Facility – GOFF, User Manual (v0.4), December 2013. Available: https://openflow.geant.net/static/media/geant/other/Geant%20OF%20Testbed%20Users%20Manual.pdf

[Golubchik] L. Golubchik, J. Lui, and R. Muntz. Reducing I/O Demand in Video-on-demand Storage Servers. ACM SIGMETRICS Performance Evaluation Review, 23(1):25-36, May 1995.

[Hua-a] K. A. Hua, Y. Cai, and S. Sheu. Patching: A Multicast Technique for True Video-on-demand Services. In 6th ACM MULTIMEDIA 1998, pages 191-200, 1998.

[Hua-b] K. A. Hua and S. Sheu. Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-demand Systems. In ACM SIGCOMM 1997, pages 89-100, 1997.

[JSON] “JSON in Java” [Online]. Available: http://json.org/java/

[OFELIA] “OFELIA EU Project Webpage” [Online]. Available : http://www.fp7-ofelia.eu/

[OFELIA-D11.1] Panagiotis Georgopoulos, Matthew Broadbent, Nicholas Race, Mu Mu, and Steven Simpson. “OpenFlow Storage Extensions Specification”. OFELIA Deliverable 11.1, March 2013.

[OFELIA-D11.2] Panagiotis Georgopoulos, Matthew Broadbent, Nicholas Race. “OpenFlow Video on Demand Cache Demonstrator”. OFELIA Deliverable 11.2, October 2013.

[OpenvSwitch] "Open vSwitch : An Open Virtual Switch" [Online]. Available: http://openvswitch.org/

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

[Liu] 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, pages 359-370.

[McKeown] 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, 38(2):69-74, Mar. 2008.

[Merwe] 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.

[MongoDB] "MongoDB: Open-source Document Database" [Online]. Available: http://www.mongodb.org

[MongoDB-PY] "PyMongo: Python Driver for MongoDb" [Online]. Available: https://pypi.python.org/pypi/pymongo/

[NOX] "NOX SDN Controller" [Online]. Available: http://www.noxrepo.org/

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

[Pouwelse] 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, volume 5, pages 4599-4605 vol.5, 2004.

[POX] "POX Open SDN Controller" [Online]. Available: http://www.noxrepo.org/pox/about-pox/

[Sen] S. Sen, J. Rexford, and D. Towsley. Proxy Prefix Caching for Multimedia Streams. In 19th IEEE INFOCOM '99, volume 3, pages 1310-1319 vol.3, 1999.

[Sripanidkulchai] K. Sripanidkulchai, B. Maggs, and H. Zhang. An Analysis of Live Streaming Workloads on the Internet. In 4th ACM SIGCOMM IMC 2004, IMC '04, pages 41-54, 2004

[Squid] "Squid Proxy Server" [Online]. Available : http://www.squid-cache.org/

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

[Zhao] 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, pages 31-42, 2013

Glossary

CDN	Content Delivery Network
GOCF	GÉANT OpenFlow Control Framework
GOFF	GÉANT OpenFlow Facility
GUI	Graphical User Interface
HD	High Definition
OCN	OpenCache Node
OCC	OpenCache Controller
P2P	Peer-to-Peer
PoP	Point-of-Presence
QoE	Quality of Experience
SD	Standard Definition
SDN	Software Defined Networking
VM	Virtual Machine
VoD	Video-on-Demand