# OpenCache: Leveraging SDN to Demonstrate a Customisable and Configurable Cache

Matthew Broadbent*, Panagiotis Georgopoulos*†, Vasileios Kotronis†, Bernhard Plattner†, Nicholas Race*

* Lancaster University, United Kingdom
{m.broadbent, p.georgopoulos, n.race}@lancaster.ac.uk

† ETH Zürich, Switzerland
{panos, vasileios.kotronis, plattner}@tik.ee.ethz.ch

*Abstract*—**Efficient content delivery is a constantly evolving challenge on the modern Internet. Reducing the impact of duplicate deliveries of identical content is a key factor in reducing congestion and transit costs for smaller networks. This work leverages SDN concepts and mechanisms in order to transparently store and deliver content from a local cache to the client, thus lightening the load on the WAN and relieving the necessity for urgent network capacity upgrades. An open interface to the cache presents owners with new possibilities for cache control and maintenance. This demonstration showcases a prototype implementation in action on a large-scale OpenFlow testbed deployed across Europe.**

## I. Introduction

The ever increasing throughput requirements of content delivery on today's Internet places more pressure on current infrastructure capacity. As such, a method to increase the distribution efficiency is desirable. Caching is one remedy to this problem; by storing and delivering a copy of the content closer to the user, we can prevent identical traffic from traversing external links after an initial request. This process reduces the volume of traffic reaching the origin server and minimises congestion on external links.

Traditionally, web caches have been used to cache online content. However, establishing these systems often requires heavy configuration and customisation by network administrators. This work proposes, through the use of Software Defined Networking (SDN), a caching method completely transparent to both the client and the server. It seeks to be both easier to deploy and more flexible in use. SDN is consistently gaining ground within the research and business communities, offering powerful open interfaces to the data plane of networks. With this work, a strong use case is provided to demonstrate its potential for efficient content delivery.

OpenFlow is a SDN mechanism which offers a standardised interface between the control and forwarding planes of switching hardware, effectively opening up the closed network "black boxes". This is achieved through the use of a software controller, which can be modified to suit the needs of the user. In the case of this work, this flexibility is leveraged to redirect requests for content without the need for clients to update their software or hardware. Furthermore, it is no longer a necessity to upgrade the delivery method or techniques used to distribute this content. This is particularly important as many modern services now use conventional HTTP methods for distribution, such as MPEG-DASH video [1].

More recently, Content Distribution Networks (CDNs) have been used to scale up content distribution to meet the massive demand. Yet CDNs are usually not deployed in smaller networks or last-mile environments due to the cost of doing so. This work offers a prototype cache implementation which has the flexibility to be deployed on many different hardware configurations in such a last-mile environment, tailored to the budget and needs of the network in which it is used.

This demonstration is designed primarily to showcase the aforementioned efficiency gains possible with existing video delivery techniques. Video-on-demand is an important use-case for caching, primarily driven by the effect of playback characteristics on user engagement [2].

## II. Testbed and Prototype

Based on a previous design [3], a prototype of an open, customisable and configurable cache, named OpenCache, was developed. This implementation functions by storing previously requested content locally, and delivering this when a subsequent user attempts to retrieve the same content. Subsequently, the traffic associated with the delivery of this content is never present within the wider network. Instead, it is now localised between the client and caching instance.

OpenCache has a novel function: cacheable content "of interest" can be expressed and declared on the fly. This results in the ability to instantaneously redirect a request for content towards the cache, without the need to propagate redirection information through the network. Furthermore, expressions can be easily removed or modified, resulting in a highly configurable cache. Through this same open interface, the cache can also report metrics and statistics in real-time, including load and usage monitoring.

In order to evaluate the efficacy of using OpenFlow to redirect these requests, OpenCache was deployed on a large-scale testbed; the OFELIA project established a pan-European OpenFlow facility with network and computing resources located in many different countries [4]. Each of these member sites consists of a number of OpenFlow switches collocated with virtualised computing capacity. These facilities are then connected together to produce a large federated experimentation environment running across Europe. The work presented herein can equally be demonstrated on any large-scale OpenFlow facility, such as the GN3plus testbed [5].

## III. Demonstrator

Preliminary results from video playback experiments on the OFELIA testbed show a ~35% reduction in client startup delay and a 100% reduction in external (over the WAN) traffic in the case of direct cache delivery of the full content [6]. This experimental setup and process will be recreated for this demonstration, clearly illustrating the behaviour of OpenCache in combination with the flexibility and openness of SDN.

The proposed experimental topology spans multiple sites: one hosts a number of video clients who request video content located on one of two geographically remote servers. As such, traffic associated with a client request has to travel over a number of different connections and through multiple networks between the client and origin server. It is expected that these network flows will incur a significant latency delay when compared to delivering the content from within the same physical location. Such flows are also exposed to potential latency fluctuations and packet loss caused by congestion within the network between the client and the server.

This demonstrator will illustrate how, by using OpenCache to store this content within the same facility, there is a reduction in the potential for playback quality to be affected by factors outside of an operator's network. This is achieved transparently to the user: the client is unaware that the content is being delivered from the cache rather than the origin server.

The demonstration begins with no caching in place within the network, which forms the baseline scenario. All OpenFlow-enabled switches present in the topology have Layer 2 switching functionality enabled. Link utilisation and client startup times are monitored and displayed. Once video playback starts, conditions for delivery cannot be guaranteed. This will be highlighted by the difference in startup delays present when using either of the two content servers.

Subsequently, the OpenCache prototype is introduced into the topology. This will be placed within the same facility as the clients, connected to a switch on the path to the content server. In its initial state, no content has been declared as cacheable and thus no additional OpenFlow rules are present within the switch. The client requests another video, and the same conditions persist as before. Following this, an expression of interest for content located on one of the remote servers is created. This action will add the necessary OpenFlow rules needed to redirect requests to the cache. The cache itself now starts a service in preparation to receive requests matching the expression of interest.

A client then makes a request for this content. Rather than traversing the external link, the request is instead received by the cache. This is possible due to the rewriting of header information when the packet is processed in the OpenFlow switch. The cache reads this request, and verifies if the content is already present. As this is the first request for the content, it is not found, and thus considered a cache-miss. Therefore, the cache will then fetch the content from the destination originally defined in the request. Once the first packet of this flow is received, the cache 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. Once the full flow has been handled in this way, the payload of the delivery is stored by the cache prototype in order to serve subsequent requests.

The final step illustrates the performance and stability improvements possible with OpenCache: a second client requests the same content already cached within the prototype. As before, the OpenFlow switch will redirect the request to the caching node. The cache will check to see if it has the object already present within its storage. This time however, the object is present and can be delivered directly from the cache to the client, and is considered a cache-hit. As before, the delivery of this content appears to originate from the server the client originally requested it from. In this final step, no traffic will have left the facility into other networks. Furthermore, the startup delay will be significantly and noticeably reduced. This can be demonstrated visually through the use of a web based tool developed in conjunction with this work. It accurately monitors and graphs statistics from the client, server and cache. The various steps are clear to see, and the flow of events on the cache are evident.

Furthermore, this work will showcase the open experimentation and administration interface developed for OpenCache. With this, adding, removing and modifying expressions in real-time can be shown. For example, on a subsequent run of the previously described demonstration, an expression for the second content server can be added, and the expression for the previously used server can be removed. This would produce a clear display of the cache in operation, and demonstrate direct interaction with an OpenCache deployment. It is envisaged that this same process will be used in the future to implement a direct interface between OpenCache and CDNs, fostering collaboration and reducing wasted resources.

## IV. Acknowledgment

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

## References

[1] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[2] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang, "Understanding the Impact of Video Quality on User Engagement," *ACM SIGCOMM CCR*, vol. 56, no. 3, pp. 91–99, 2013.

[3] M. Broadbent and N. Race, "OpenCache: Exploring Efficient and Transparent Content Delivery Mechanisms for Video-on-Demand," in *Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop*, ser. CoNEXT Student '12, 2012, pp. 15–16.

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

[5] GN3plus OpenFlow Facility. [Online]. Available: http://openflow.geant.net/

[6] P. Georgopoulos, M. Broadbent, and N. Race. "OFELIA Deliverable 11.2: OpenFlow Video on Demand Cache Demonstrator". [Online]. Available: http://opencache.io/ofelia/d11-2.pdf