A Distributed Caching Architecture for Over-the-Top Content Distribution

Rui Dias*†, Adriano Fiorese†‡, Lucas Guardalben†, Susana Sargento*†
*University of Aveiro, Portugal
†Instituto de Telecomunicações - Aveiro, Portugal
‡Santa Catarina State University (UDESC) - Brazil
Email: ruidias@av.it.pt, adriano.fiorese@udesc.br, guardalben@av.it.pt, susana@ua.pt

***Abstract*—Over-the-Top (OTT) services encompass video and audio delivery over the Internet without the strict and direct control of Telecom operators, being attractive, low-cost and profitable. Due to the increasing use of the Internet and the increasing bandwidth provided by Telecom operators, excellent conditions were leveraged for the raise and growth of OTT multimedia streaming services, shown by the huge success of YouTube and Netflix.**

OTT services grow at a fast pace driven by a low barrier of entry, mostly because of little to no investment being required in infrastructures traditionally necessary to reach the masses. This fast-paced growth presents an opportunity for all the involved partners, but comes with several challenges, especially in terms of OTT content distribution scalability with high perceived QoE levels experienced by the consumers.

This paper presents a novel Distributed Smart Management Cache (DSMC) architecture based on distributed caching model taking advantage of all horizontally available edge caches, thus avoiding repetition of content among caches of the same members group, which horizontally solves scalability challenges, enhancing clients' perceived Quality of Experience (QoE) in comparison to traditional decentralized architectures. Exhaustive tests were performed with several clients using a real QoE probe in different scenarios (fixed at home and mobile at street), with particular conditions imposed by consumer habits, and by network impairments (jitter, latency and poor bandwidth, etc.) from the involved technologies (Fiber, Wi-Fi and 4G LTE). The obtained results confirm that the proposed DSMC architecture plays an important role on providing QoE requirements and bandwidth consumption in the OTT delivery, reducing edge cache loads and consumed bandwidth by upstream content server/origin.

***Index Terms*—Distributed cache, OTT, Proxy cache, Distributed mechanism, Networks and Multimedia**

I. Introduction

In recent years, due to the increasing bandwidth and reduction of access times provided by telecommunication operators, consumer habits regarding content consumption have been changing [1], which might be demonstrated by the clear increase in the trend of non-linear television (TV) video watching versus broadcast TV services [2]. Moreover, with the huge growth of mobile market, it is possible to watch content anywhere, anytime, and at any device. However, Over-the-Top (OTT) content is not directly controlled by Telecom operators, and Quality of Service (QoS) is not guaranteed, therefore compromising final users' Quality of Experience (QoE). To maintain high QoE standards, it is necessary to invest in the multimedia delivery infrastructure, optimizing delays and avoiding video degradation that results on final users' experience frustrations. Typically, video streams use different types of streaming protocols, being different in just a few details in the implementation [3].

Independently of the video streaming protocols, optimizations in the content delivery pipeline between clients (players) and servers are mandatory, especially in pull-based progressive download case, where the server does not need to keep statefull data for each client. There are several solutions that have been recurrently explored to offer customers the best content caching without compromising QoE.

The most common approaches are based on proxy caching [4] [5] , Peer-to-peer (P2P) [6], [7] and Hybrid Delivery [8], [9]. Regarding the proxy cache approaches, some of them propose layers of caches building up a caching hierarchy system [10], [11]. Algorithms comprising an offline cache aware of future requests (greedy) to estimate the maximum efficiency expected from any online algorithm, and an optimal offline cache (for limited scales) are proposed in [12]. Caching systems addressing information centric overloaded networks in vehicular and edge computing environments are also proposed in [13], [14].

The problem of controlling the placement of content in the available caching points also received recent attention in the literature [11], [15]. A multi-attribute caching strategy for Content Centric Networks (CCN) devising efficient caching mechanisms that allow maximum availability of content while consuming minimum possible resources is proposed in [16]. Also, in the realm of CCN, a contribution on the cache allocation problem, dealing with how to distribute the cache capacity across routers in a constrained storage network, is present in [17]. Moreover, efficient cache eviction policies were proposed in order to manage popularity of content being requested on the several caches on the system [18]–[20].

Although these works address several issues regarding content placement, eviction policies, architectural layering and other problems comprising the existence of content caching systems, they are proposed to improve performance in a core-based manner lacking traction on the edge-side and on particular OTT content characteristics. Thus, a more edge-based cache approach, based on distributed global caching mechanisms, is expected to deal with the OTT content distribution challenges over mobile scenarios.

This work focuses on providing a proxy cache solution in a distributed manner, increasing the efficiency of content

ISBN 978-3-903176-02-7 © 2018 IFIP

allocation through a novel Distributed Smart Management Cache (DSMC) architecture based on distributed edge caching model, taking advantage of all horizontally available edge caches, thus avoiding repetition of content among caches of the same members group, which horizontally solves scalability problems, enhancing clients' perceived Quality of Experience (QoE) in comparison to traditional decentralized architectures. Additionally, the proposed DSMC architecture differs from traditional CDNs in twofold, i) the content is also stored and fetched horizontally in the edge caches, close to the user's consumption, ii) pushing applications logic to the edge caches, off loading origins/aggregators load requirements towards a distributed intelligent network.

Exhaustive tests were performed with several clients using a real QoE probe [21] behaving as a real player in different scenarios (fixed at home and mobile at street), with particular conditions imposed by consumer habits, and by network impairments (jitter, and latency, poor bandwidth, etc.) from the involved technologies (Fiber, WiFi and 4G LTE). The obtained results confirm that the proposed DSMC architecture plays an important role on providing QoE requirements and bandwidth consumption in the OTT delivery, reducing load and consumed bandwidth by upstream content server/origin.

The remainder of this paper is organized as follows. Section II describes the proposed architecture, its modules and its functionalities. Section III presents the proposed architecture's evaluation performed on several scenarios and analyses the achieved results. Finally, Section IV depicts conclusions and future work.

II. Distributed Smart Management Cache

This paper proposes a Distributed Smart Management Cache (DSMC) architecture for OTT adaptive video, taking advantage of all horizontally available edge caches, thus avoiding repetition of content among caches of the same members group.

Fig. 1 shows an overview of the distributed content delivery pipeline starting from the consumer point of view with all modules involved on the communication. At a first glance, the content consumption pipeline starts from consumers which request chunks of a particular video to an edge cache. Then, the edge cache has to know if the content is in the global cache structure formed by its edge caches' group, where an edge-cache group is related with a particular aggregator. If the group has the content, then the edge cache delivers it immediately to the client/consumer. Otherwise, it will request chunks of content to the aggregator. If the aggregators do not have stored that content, then the request is forwarded to the origin server. On the return, those chunks of content are stored in the requesting aggregator and forwarded to the requested edge cache that delivers it to the consumer, and also decides where to keep it stored in the global cache provided by the whole edge cache group members. To accomplish the distributed edge global cache, a strategy for adding, removing and updating edge-nodes is proposed, which minimizes the content remapping among the edge nodes group, i.e., minimizes the need of moving data among nodes to keep a consistent cache state.

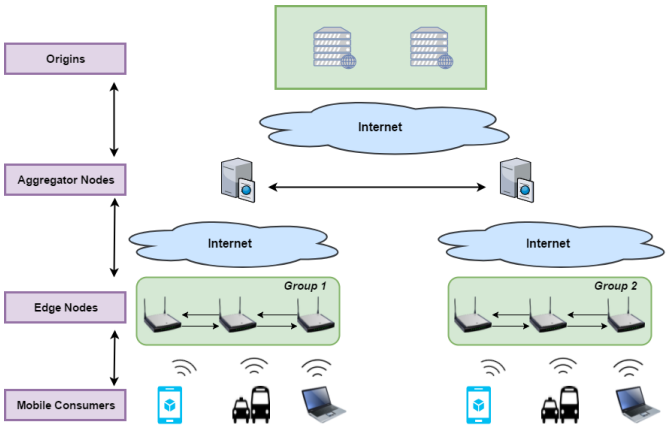


Fig. 1: Distributed Content Delivery Pipeline Overview.

In traditional Content Distribution Networks (CDNs) each cache node typically has a built-in local cache that does not horizontally share content with other cache nodes. In our proposed architecture, a n-tier approach is used, where the first cache tier and the one closer to the consumers which comprises edge cache nodes responsible to distribute content to consumers or groups of consumers. This approach pushes applications and services logic to edge caches, off loading origins/aggregators load requirements towards a distributed intelligent network.

The second cache tier is provided by Aggregator nodes, which are intermediary between edge caches and origin servers, which have the main functionality of reducing excessive backend traffic to the origin servers. Therefore, we have defined a group as a bunch of edge-cache nodes that work in a cooperative way, followed by the DSMC architecture to dynamically share the content load between edge group members. The group formation relies on the consistent hashing method, which are responsible to control and manage the content being cached at edge-cache members.

The key components of the proposed edge-cache nodes in the DSMC architecture will be individually described on the following subsection.

A. Proposed DSMC Architecture

The Nginx module handles the HTTP requests from the downstream or from consumers, and with the help of the DSMC, it checks if the required content is in the global cache. If the content is not in the global cache, the Nginx module sends a request to the upstream server that retrieves it on behalf of the edge cache through the Nginx reverse proxy feature, which provides an additional level of abstraction and control to ensure the smooth flow of network traffic between the edge cache and the origin servers. When the upstream server returns the content, it is sent to the client and a copy of it is stored in the global cache with the help of DSMC, working as a transparent cache that might intercept, modify and forward the requests to their destinations.

ISBN 978-3-903176-02-7 © 2018 IFIP

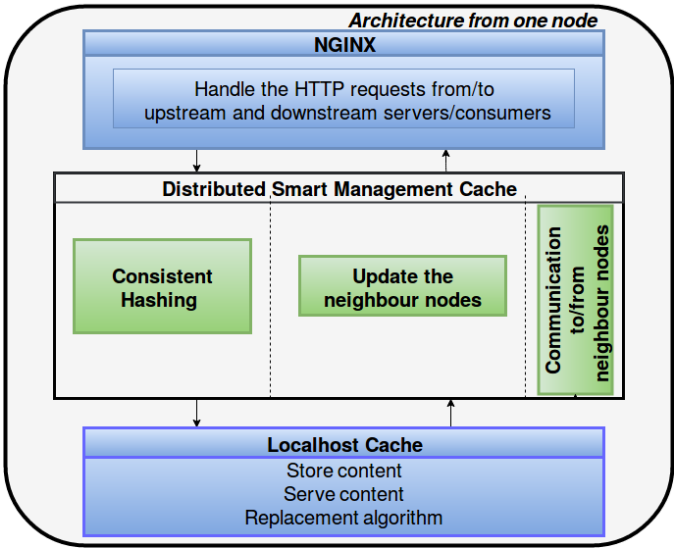


Fig. 2: DSMC architecture from edge-caches view

The DSMC module provides three main functions, by means of its three sub-modules:

- Consistent Hashing: This module decides on how to distribute the received content chunks by edge cache neighbours in a based ring topology and by local cache, through the use of a based distributed consistent hashing algorithm. As soon as the content needs to be stored/fetched, the consistent hashing algorithm will automatically choose an edge cache where it can store and/or fetch the data. When the decision relies in only one edge cache, the decision is easy; otherwise, as soon as multiple edge caches make part of the ring, which leads to our case, the consistent hashing shall find a way to store the content across multiple edge caches. We have proposed a simple version of consistent hashing [22] distributed algorithm, which allows addition and removal of edge cache nodes from the ring, and remapping of only one part of the content. The key fragments of the video are placed on each edge cache accordingly to a weighted distribution, which prioritizes edge caches with more resources available (storage and processing capacity). This approach allows to balance the content distribution among edge caches, minimizing access overhead and bandwidth consumption on few popular nodes. Therefore, the DSMC has to update the neighbours, as well as to receive the messages from Nginx and, through the consistent hashing, send the messages to the neighbour caches. Each video fragment has an associated value/content key, whose hashing is used for video content fragments management and control across the DSMC architecture. Virtual nodes pointing out to physical edge cache servers are adopted in order to allow a better load balance between the physical nodes. The virtual nodes' structure is managed and controlled by the consistent hashing module.
  Fig. 3 shows the ring topology control connecting all the edge cache servers. Thus, the content can be mapped in any part of the ring, that is, in any cache server belonging to the ring. So, the video chunk content distribution is based on the content chunk ID, extracted from the URL, hashing. This results in an integer, which should correspond to the content's cache server position in the ring. Therefore, it is necessary to move clockwise along the ring to find the edge cache server where this content will be stored. If the exact position does not exist, the content is stored in the cache server in the closest (in a clockwise manner) position.

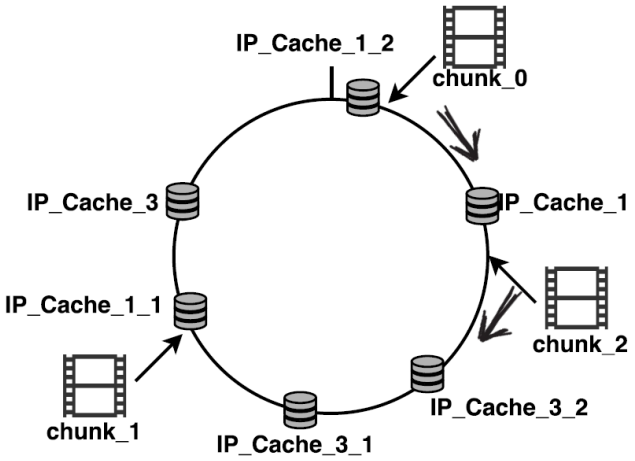


Fig. 3: Consistent hashing ring topology with virtual nodes

  With regards to addition and removal of edge caches in the ring, only part of the content needs to be remapped. Thus, when a cache server is removed from the ring, the previously cached content in that server is missed in a new user request. Thus, content needs to be fetched (upstream servers) and stored again in the next cache server position (in clockwise way) in the ring.
- Communication to/from Neighbours: This module communicates with the other edge cache nodes as well as with Nginx proxy, through TCP messages, to send the content which is basically composed by fragments of the video chunks. The messages used are predefined by the Memcached protocol[1], comprising messages of type GET, to search the content, and SET, to cache the content at the edge cache servers. In addition, it is also necessary to exchange more information between the edge caches status, the removal and addition of a node, as well as a set of messages to ensure that the edge caches in the consistent hash ring are synchronized. These messages are implemented using UDP protocol, and they are managed by the Updated edge cache Neighbours module.
- Update Neighbours: This module has to find and be aware on how many neighbours exist. This is accomplished by sending broadcast messages using UDP, informing

[1]https://memcached.org/

ISBN 978-3-903176-02-7 © 2018 IFIP

about the cache storage size and broadcaster IP address. Through these messages exchange, it is possible to have an updated list of neighbour edge caches to be used by the consistent hashing method to decide where to distribute content. Therefore, DSMC also checks and maintain, when searching for the content, a track list of simultaneous identical content requests recently made that were a miss (content not found in the edge cache storage), avoiding to forward to upstream servers all that requests. A mechanism to control the requests through a lock and release based approach is proposed in order to allow the content to be received and stored, and then delivered to the consumers. The edge cache responsible for this lock is the one that is in charge of the given chunk/content. The edge cache that is in charge of fetching content from the origin is the first to verify that the chunk is not in cache, while the others have to wait for this edge cache to SET (store) the content.

The Localhost cache module performs the locally in-memory content storage of the edge cache following rules of a content eviction policy. Although there are several eviction policies, this work uses the Least Recently Used (LRU) eviction policy from Cachelot[2] solution. LRU deletes the objects (content) that are not used for the longest period of time, i.e., not necessarily the largest object and even not the first object stored in the edge cache. Cachelot is adopted to be in charge of the locally stored objects management. Cachelot uses an in-memory key-value data model that contains a single value for each key, and that value is the content being accessed through the search of the respective key. As soon as Cachelot receives a message SET from the Communication to/from Neighbours module, it will store the content locally, and then Cachelot responds if the request is stored successfully. In case of GET messages, Cachelot checks if the content is in local cache, which returns the required content (a hit) in a positive case; otherwise a miss is returned and then the video content is locally stored in memory.

III. Evaluation Setup and Experimental Results

To evaluate the DSMC proposed architecture, we have performed assessments on distributed and decentralized content delivery models. Comprising the distributed model, whose test setup is depicted in Fig. 4, each edge cache communicates horizontally to their neighbours edge caches, thus providing a global cache. On the other hand, differently from the distributed model, the decentralized model do not communicate and share content horizontally between other edge-caches, thus representing the traditional CDN approach.

Regarding the implementation of both architectural models, the origin server is a Windows Server running IIS Smooth Streaming [3]. It keeps and serves all the content of all the videos. The aggregator is a server machine that has a local cache and the ability to do reverse proxy and transparent cache.

[2]http://cachelot.io/

[3]https://www.iis.net/downloads/microsoft/smooth-streaming

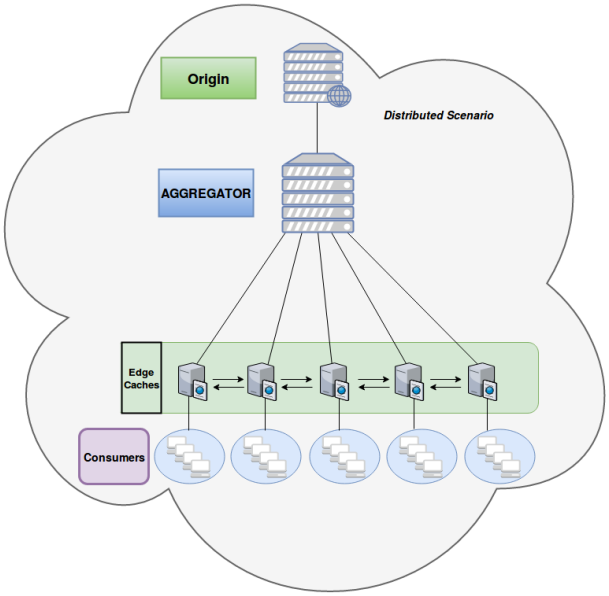


Fig. 4: Test Setup of the Distributed Architectural Model

The consumers are instances of a Microsoft Smooth Streaming probe [21] that calculates the video QoE.

Two content distribution scenarios are taken into account considering both architectural models to assess the proposed architecture contributions. The first one considers a non mobile consumer watching content with different video popularities in a stable network that does not suffer from high latency or low bandwidth and supports all traffic, which are typical conditions of a consumer at the home scenario. The second one considers a mobile content consumer as illustrated in Fig. 5. In this scenario there are several clients that are spread over 3 different wireless access regions, where each region presents different latencies. Each edge cache node plays the role of access point for the content consumers. In this scenario, QoE is measured for each costumer group composed of costumers in each region, comprising the QoE variation regarding the several network limitations imposed on the three groups (three wireless access regions). Moreover, it is also measured the edge caches performance as well as the impact of bandwidth consumption considering perfect and non perfect conditions of the network.

Both architectural models are assessed considering different levels of content popularity required by the consumers, given by the Zipf's law [23]. Particularly, two different popularity curves comprising s = 1 and s = 3 are used to predict the consumer's requested content chunks. In the street mobile scenario, it is performed exhaustive tests comprising the distributed and decentralized models, considering that consumers are mobile, which means that the link between edge cache nodes and the consumer suffers from limitations, such as some latency and higher jitter due to physical obstacles and the distance of clients from the Base Station (BTS), which are

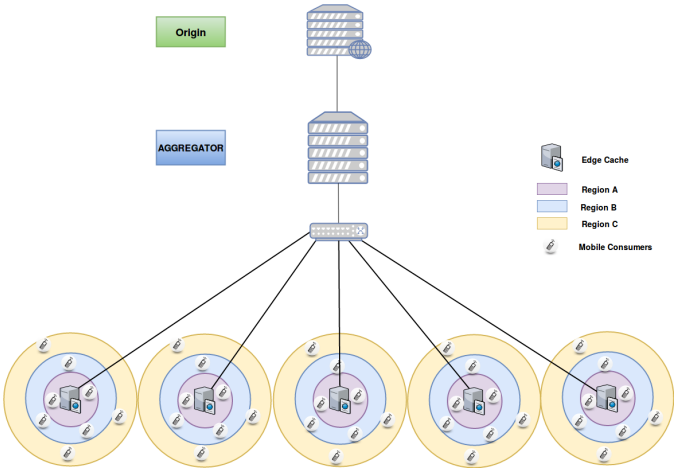


Fig. 5: Street Mobile Scenario

characteristics of mobile networks. Therefore, in this mobile scenario consumers have more limitations compared to the previous scenario, characterizing consumers walking on the street.

To perform the evaluation of the proposed architecture according to the different architectural model flavours (decentralized and distributed), on different content consumer scenarios, virtual machines (VMs), each one with 2GB of RAM, two 2500MHz cores and Ubuntu 14, are used in a testbed representing edge caches and one aggregator. Moreover, each edge cache has 512 MB of memory LRU cache provided by Cachelot, with a memory page predefined on 2MB. The aggregator's VM has 1GB of LRU memory cache provided by the NGINX. The uplink and downlink of the several VMs are of 1Gbps, since they are in the same host. VMware is used to install and manage the several virtual machines.

The experiments were performed using 10 different videos consumed according to a certain popularity defined for each scenario. Also, each video size is of approximately 11 minutes. In addition, each experiment is executed 5 times, in order to ensure a confidence interval. All the experiments used 5 edge caches, one aggregator and one origin. Some metrics are used to evaluate the results: cache performance, hit and miss ratio, as well as consumed bandwidth.

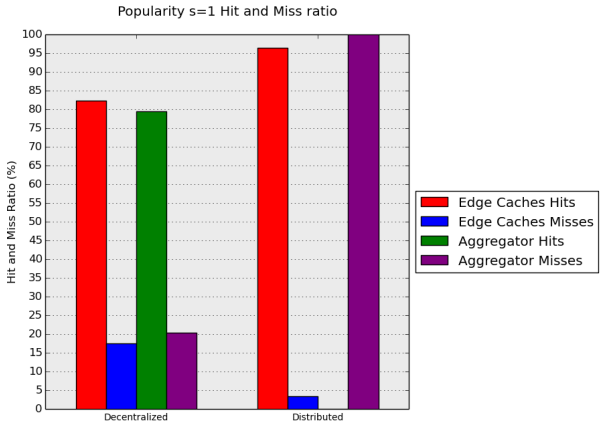
To test the user at home scenario with different content popularity, the consumers watch different videos based on a certain probability, given by the function Zipf's law, using s=1 and s=3 as its parameters. For each popularity curve of the Zipf's law (given by s parameter), exhaustive tests are performed with the same number of clients, which is 200. In this case, cache performance (hits and misses) and edge cache nodes consumed bandwidth are measured in order to draw conclusions about customer behaviour and network impact.

To test the mobile scenario, 150 clients are used. Each costumer group (i.e., group of costumers in the same wireless access region/range) is composed of 50 clients, that are spread over the several edge cache nodes. Each edge cache node has 10 clients in region A, 10 clients in region B and 10 clients in region C. In the testbed a wireless region is defined as a group of clients that have latency and jitter restrictions.

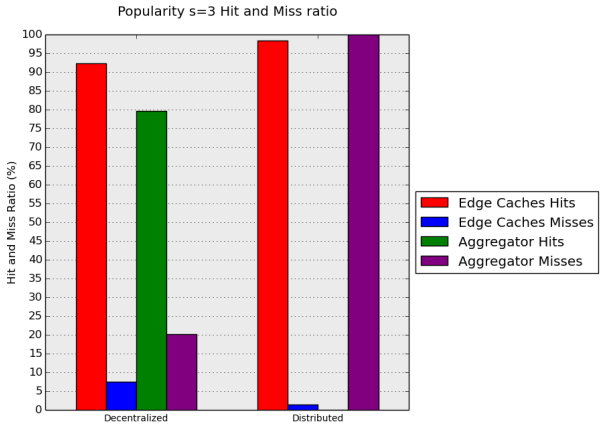
In this specific case, the clients of region A have an average latency and jitter of 80ms and 40ms, respectively. The latency distribution is defined by a normal distribution. In region B clients face average latency and jitter of 350ms and 200ms, respectively, also defined by a normal distribution. Finally, in region C clients present latency with an average latency and jitter of 800ms and 250ms, following a normal distribution.

Latency is introduced by the command $netem/tc$ from the operating system Linux. All figures are plotted with a confidence interval of 95%, except the figures of QoE that have a confidence interval of 65%, and the ones at 100% for stable network conditions.

Fig. 6a and Fig. 6b depict results of cache hits and misses comprising experiments on the consumer at home scenario for different content popularities, according to Zipf's law. Fig. 7a and Fig. 7b depict results of bandwidth consumption in the same scenario.



(a) Hit and Miss ratio, s=1



(b) Hit and Miss ratio, s=3

Fig. 6: User at Home, Hits and Misses - different video popularities

According Fig. 6a and Fig. 6b, with the popularity defined by the parameter s = 1 of the Zipf's law function, it is observed that the distributed architectural model presents a higher hit ratio compared to the decentralized one. However,

edge caches present a higher consumed bandwidth, as can be seen in Fig. 7a, since in this case, in addition to consumption, edge caches also provide content to the other edge caches.

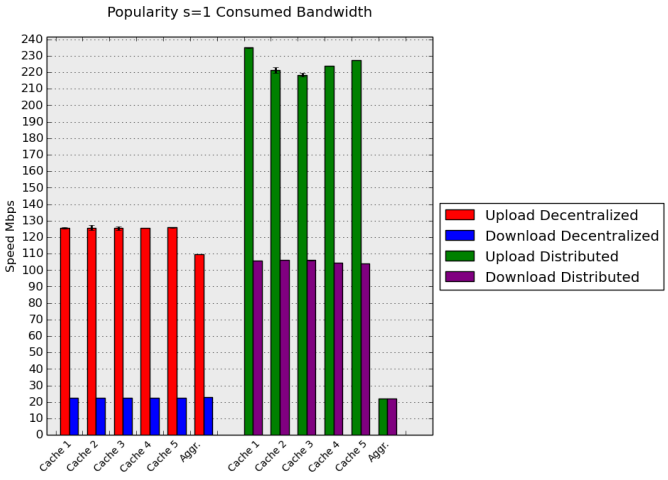
Also, it is noted that the aggregator present 100% of miss ratio in the distributed model, independently of video popularity, as observed in Fig. 6a and Fig. 6b. This happens because, in this model, the content is not required twice to the aggregator, i.e., in this case, when content is requested to the aggregator, a miss will be generated because content is not there, and consequently, it will be fetched in the origin, stored in the aggregator and forwarded to the edge cache, the other edge caches will not fetch this content in the aggregator because it is in the global cache of edge caches. When a new request of the same content is made, content will be found in one of the edge cache nodes composing the group global cache, thus never reaching the aggregator until LRU eviction policy freshes it from edge caches. In the decentralized model, when there is not a global cache, the aggregator can score hits because the content request does not find the content in isolated edge cache, therefore it needs to be forwarded to the aggregator, resulting in the aggregator to handle multiple requests for the same content.

When popularity is defined by the parameter s = 3 (Fig. 6b), there is a higher frequency in the user views of the same video, that is, there is more popular content than with s=1 (Fig. 6a). Thus, when comparing cache performance results to those with parameter s = 1 (Fig. 6a), it is possible to note that the distributed architectural model presents higher hit ratio compared to the decentralized one. On the other hand, similarly to the results with popularity defined by s = 1, the distributed model consumes more bandwidth (Fig. 7b) by the same reasons.

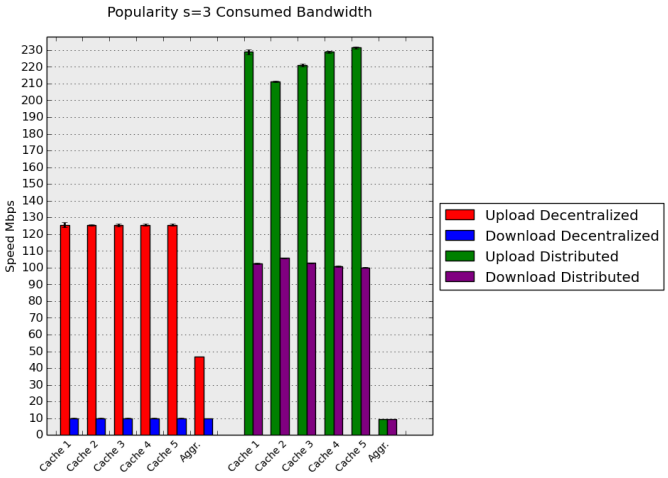
Analysing Fig 7a and Fig. 7b, it is possible to note that, depending on the habits of the clients, i.e., depending on the popularity of the videos, the bandwidth consumption as well as the load on the aggregator are changed. Overall, it is also observable that, independently of the video popularity, the distributed model presents higher use of the bandwidth in the edge caches side, when compared with the decentralized model. However, the aggregator side presents lower load and also a reduction of the consumed bandwidth compared to the decentralized model, allowing the aggregator to be able to respond to a higher number of clients/requests, thus improving QoE. Moreover, it is also possible to observe that the more popular the video, the lower load and bandwidth consumption on the aggregator side. Thus, it is possible to state that the distributed model presents a better performance, in terms of reducing computational and network load in the upstream server, i.e., the aggregator. However, in the edge caches side the distributed model presents a higher utilization of the network when compared to the decentralized model.

Fig. 8a and Fig. 8b show QoE results comprising the mobile scenario. Analysing them, it is possible to verify that, with network conditions deterioration, in other words, with increasing latency and jitter, the QoE is lower.

It is observed that in the region A (the closest to the edge cache), the latency had an insignificant impact on the



(a) Consumed Bandwidth, s=1



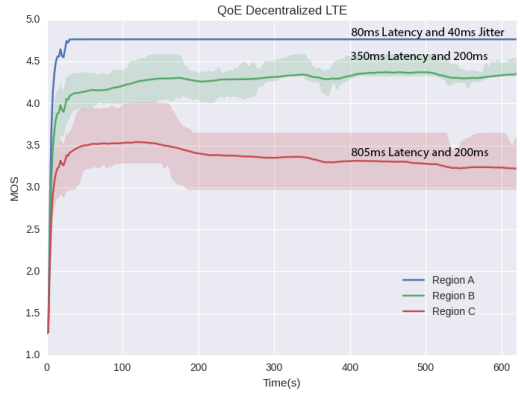
(b) Consumed Bandwidth, s=3

Fig. 7: User at Home, Bandwidth - different video popularities

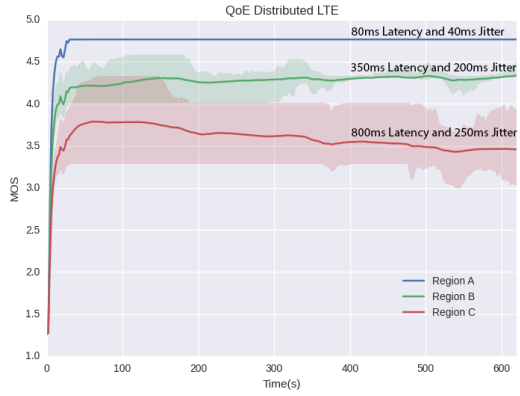
QoE, which means a Mean Opinion Score (MOS) always larger than 4. In region B there is a slight degradation of QoE, though MOS continues to be above 4. Moreover, it is observed that, due to jitter there is a higher discrepancy of QoE among the different customers. Regarding region C, there is a significant QoE degradation compared to the other regions. In this case, MOS average is below 4. However, and most important, it is possible to note that the distributed model performs better in comparison with the decentralized one, allowing improvements of 0.3 points in average on the QoE, especially in region C that has the worst network conditions.

In addition to QoE, cache hit and miss ratios are also evaluated in both models, distributed and decentralized, in this mobile scenario. The impact of the hit and miss ratios is evaluated with and without network limitations, i.e., with additional latencies in the 3 regions already mentioned, and also without them. Fig. 9 presents the results of the cache hit and miss in the mobile scenario.

According to Fig. 9, it is possible to observe that in both models, decentralized and distributed, with the same



(a) Decentralized model



(b) Distributed model

Fig. 8: Mobile Scenario - QoE Analysis

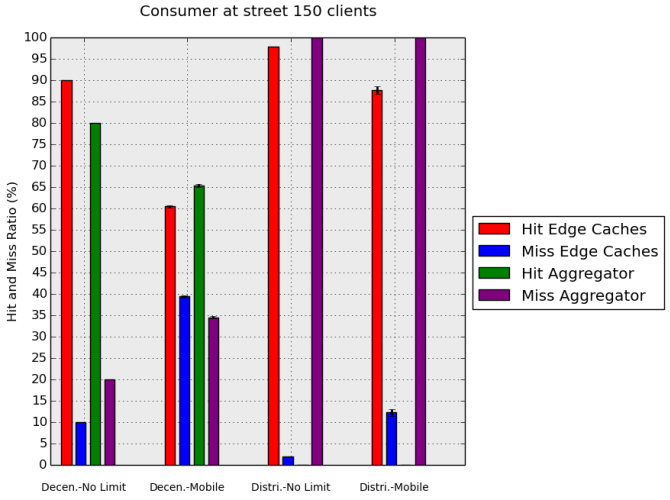
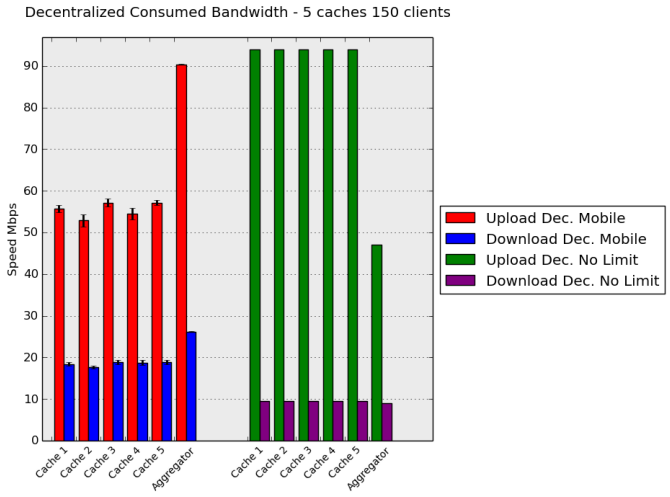
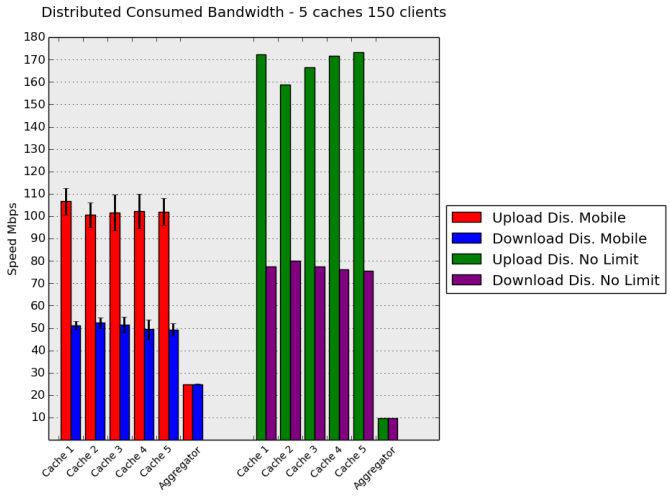


Fig. 9: Mobile scenario - hit and miss ratio

percentage of clients watching the same videos and with network latencies, the performance in the edge caches side decreases. In other words, experiments using higher latencies present lower hit ratio and higher miss ratio, when compared with results from experiments without network limitations. Anyway, in both cases, the distributed model always presents better results in terms of edge caches performance, compared to the decentralized one.



(a) Decentralized model



(b) Distributed model

Fig. 10: Mobile Scenario - consumed bandwidth

Fig. 10a and Fig. 10b show results of consumed bandwidth for both decentralized and distributed models in the mobile scenario. According to them, it is possible to observe that, independently of limitations comprising latency and jitter on the network, consumed bandwidth for upload and download in the edge cache nodes is reduced in the decentralized model when compared to the distributed one. This can be explained in part because in the decentralized model there is a consumption of a lower video resolution, implying a smaller bit rate per client. Also, it is possible to observe that the consumed bandwidth by the edge caches when there are latency and jitter limitations is

ISBN 978-3-903176-02-7 © 2018 IFIP

lower than when there are no limitations. However, the aggregator presents higher bandwidth consumption when there are such limitations. This happens because there are less content requests homogeneity, i.e., more video resolution qualities will be requested at the same time since consumers will often adapt their video resolution quality requests to the network conditions, which means more consumer requests have to be forwarded to the aggregator since that type of content is not in the edge caches. Overall, analysing results of the two models, distributed and decentralized, it is possible to state that in the distributed model the consumed bandwidth in the edge caches side is higher than in the decentralized model. However, the distributed model presents lower consumed bandwidth in the aggregator side than in the decentralized model.

IV. Conclusions and Future work

This work proposes DSMC, a distributed caching architecture for OTT content distribution. This architecture relies on a content distribution model where caches are split in two intermediary tiers between consumers and content origin server. The proposed architecture is composed of several modules that deal with the content distribution among the edge caches, which are closer to the consumers. These modules are responsible to communicate among the edge caches to control the global cache formed by edge caches, to distribute the content at this global cache, as well as to serve consumers.

The proposed architecture evaluation was performed taking into account its implementation on two architectural models. 1) decentralized, where edge caches are independent from one and another relying on traditional CDN models; 2) distributed, where the edge caches interconnect themselves forming a group global cache. For both models, experiments were performed comprising two scenarios: i) wired stable network suitable to consumer at home scenario; ii) a mobile scenario, comprising some conditions of mobile networks suitable to mobile consumer at street. Results regarding edge caches performance and bandwidth consumption indicate that the proposed DSMC architecture is well suited to the distributed model in both scenarios, since QoE has been improved in places far from the base stations, in the mobile scenario in the street; and consumed bandwidth has been saved in upstream servers in both scenarios.

Future work includes the test of the distributed model in a large set of real scenarios, including high velocity ones, such as in a train. In addition, it is a plan to enhance and test the proposed architecture with several types of streaming technologies, beyond Microsoft Smooth Streaming, such as Apple HTTP Live Streaming, Adobe HTTP Dynamic Streaming and MPEG-Dash. Integration with pre-fetching and QoE optimization approaches is also intended.

Acknowledgment

This work is supported by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project UltraTV with Nr. 017738 (POCI-01-0247-FEDER-017738)] and the Eureka Celtic Plus project MONALIS.

References

[1] G. Petersen, "The Rise of OTT Video and Average Profit Per User ," Calix, Tech. Rep., 08 2016.

[2] "Linear vs non-linear viewing: A qualitative investigation exploring viewers'behaviour and attitudes towards using different TV platforms and services providers," KantarMedia, Ofcom, Tech. Rep., 05 2016.

[3] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over http," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1842–1866, thirdquarter 2017.

[4] J. X. Jiangchuan Liu, "Proxy caching for media streaming over the Internet," *IEEE Communications Magazine*, vol. 42, no. 8, 08 2004.

[5] L. B. Claudiu Cobarzan, "Further Developments of a Dynamic Distributed Video Proxy-Cache System," in *Parallel, Distributed and Network-Based Processing, 2007. PDP '07. 15th EUROMICRO International Conference on*, 03 2007, pp. 349–357.

[6] J. Li, "On peer-to-peer (P2P) content delivery," *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 45–63, 11 2007.

[7] D. Ghosh, P. Rajan, and M. Pandey, "P2P-VoD Streaming:," in *Advanced Computing, Networking and Informatics- Volume 2*, ser. Smart Innovation, Systems and Technologies. Springer, Cham, 2014, pp. 169–180.

[8] B. A. S. M. Y. Seyyedi, "Hybrid CDN-P2P architectures for live video streaming: Comparative study of connected and unconnected meshes," in *Computer Networks and Distributed Systems (CNDS), 2011 International Symposium on*.

[9] G. K. Mikael Goldmann, "Measurements on the spotify peer-assisted music-on-demand streaming system," in *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, 10 2011, pp. 206–211.

[10] T. X. Tran, A. Hajisami, and D. Pompili, "Cooperative Hierarchical Caching in 5g Cloud Radio Access Networks," *IEEE Network*, vol. 31, no. 4, pp. 35–41, Jul. 2017.

[11] G. Domingues, E. de Souza e Silva, R. Leão, D. Menasché, and D. Towsley, "Search and Placement in Tiered Cache Networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 2, pp. 9–11, Sep. 2016.

[12] K. Mokhtarian and H. A. Jacobsen, "Flexible Caching Algorithms for Video Content Distribution Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, Apr. 2017.

[13] W. Zhao, Y. Qin, D. Gao, C. H. Foh, and H. C. Chao, "An Efficient Cache Strategy in Information Centric Networking Vehicle-to-Vehicle Scenario," *IEEE Access*, vol. 5, pp. 12 657–12 667, 2017.

[14] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in Mobile-Edge Computing networks," in *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Feb. 2017, pp. 165–172.

[15] D. Tuncer, V. Sourlas, M. Charalambides, M. Claeys, J. Famaey, G. Pavlou, and F. D. Turck, "Scalable Cache Management for ISP-Operated Content Delivery Services," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2063–2076, Aug. 2016.

[16] S. Naz, R. N. B. Rais, and A. Qayyum, "Multi-Attribute Caching: Towards efficient cache management in Content-Centric Networks," in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan. 2016, pp. 630–633.

[17] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and Evaluation of the Optimal Cache Allocation for Content-Centric Networking," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 95–107, Jan. 2016.

[18] J. Nogueira, D. Gonzalez, L. Guardalben, and S. Sargento, "Over-The-Top Catch-up TV content-aware caching," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, Jun. 2016, pp. 1012–1017.

[19] M. Bilal and S. G. Kang, "A Cache Management Scheme for Efficient Content Eviction and Replication in Cache Networks," *IEEE Access*, vol. 5, pp. 1692–1701, 2017.

[20] A. Araldo, D. Rossi, and F. Martignon, "Cost-Aware Caching: Caching More (Costly Items) for Less (ISPs Operational Expenditures)," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1316–1330, May 2016.

[21] A. Salvador, J. Nogueira, and S. Sargento, *QoE Assessment of HTTP Adaptive Video Streaming*. Cham: Springer International Publishing, 2015, pp. 235–242. [Online]. Available: https://doi.org/10.1007/978-3-319-18802-7_32

[22] T. L. R. P. M. L. D. L. David Karger, Eric Lehman, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web," in *STOC 97 Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 654–663.

[23] B. A. H. Lada A. Adamic, "Zipf's law and the Internet," *Glottometrics*, vol. 3, pp. 143–150, 2002.