Caching Using Software-Defined Networking in LTE Networks

Mael Kimmerlin, Jose Costa-Requena, Jukka Manner, Damien Saucez, Yago Sanchez

Caching Using Software-Defined Networking in LTE Networks

Mael Kimmerlin
Aalto University
School of Electrical Engineering
Otakaari 5
02150 Espoo
mael.kimmerlin@aalto.fi

Jose Costa-Requena
Aalto University
School of Electrical Engineering
Otakaari 5
02150 Espoo
jose.costa@aalto.fi

Jukka Manner
Aalto University
School of Electrical Engineering
Otakaari 5
02150 Espoo
jukka.manner@aalto.fi

Damien Saucez
Inria
Sophia Antipolis
France
damien.saucez@inria.fr

Yago Sanchez
Fraunhofer HHI
Einsteinufer 37 10587 Berlin
Germany
yago.sanchez@hhi.fraunhofer.de

ABSTRACT

The data consumption is increasing rapidly in mobile networks. The cost of network infrastructure is increasing, which will lead to an "end of profit" within next few years. Thus, mobile operators require technology that allows increasing the network capacity within low network costs. Therefore, using caching is the most evident solution to be used in their backhaul networks. However, the current architecture of LTE network does not provide sufficient flexibility to place the caches in the most optimal locations. The current work on Software-Defined Networking in Evolved Packet Core virtualization has enabled us to integrate dynamically the caching functionality in a LTE network and improve the caching system performance. In this paper, we present the solution we designed for this aim based on Software Defined Networking technology. Moreover, we develop a testbed for the proof-of-concept and we present performance analysis of this solution.

Categories and Subject Descriptors

H.3.4 [**Information Systems**]: Systems and Software—*distributed systems - information networks*; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Caching, Software-Defined Networks, LTE networks

Keywords

caching, Software-Defined Network, LTE networks, content relocation

1. INTRODUCTION

The explosion of data traffic in mobile networks due to the massive adoption of smart-phones and tablets creates new challenges for network operators. While customers always ask for more bandwidth. On the other hand, they are not willing to pay more for their mobile data plan. Unfortunately, to increase the bandwidth, operators must invest (e.g., re-provision their backhaul, deploy new technology) but cannot echo the investment in the data plan price, progressively leading to an "end of profit" for networks operators in 2015 according to a recent study [9]. In this new context, it is essential to optimise network capacity usage in order to reduce operational expenditure (OPEX). The best solution to improve the capacity usage of the network is to place caches in strategic locations within the network. Such caches are commonly used in Internet Services Providers (ISP) for wired networks but are largely ineffective in mobile LTE networks. The reason is that LTE networks rely on tunnels to allow seamless mobility (i.e., without communication interruption) [**?**]. The usage of tunnels between the base stations where users connect their devices to the ISP network and the gateways that connect the ISP network to the Internet makes it difficult to place caches in the most appropriate locations in the LTE network. The presence of tunnels indeed implies that caches are located at the ends of tunnel which in practice means either in the base station or in the ISP core network. Such centralisation of caches is not adequate because of the distributed and random nature of content consumptions. The general research trend today for wired network is to propose in-network caching where each network component disposes of storage and opportunistically caches contents passing through it [**?**]. In-network caching leverages traffic locality so that contents are likely to be delivered from caches located close to the consumers. Applying such model to LTE network would result in an overall backhaul link usage reduction.

To overcome the limitation of mobile networks, and more particularly the operational constraints imposed by LTE, we propose to use a *Software Defined Networking* (SDN)

approach to remove the need of establishing tunnels. With SDN, packets are forwarded according to rules installed on the switches by a centralised *controller* [15]. The solution we propose enables in-network caching in the core and backhaul of the LTE provider network. As packets are not encapsulated, routers have direct access to the IP packets as sent by end-hosts and can thus apply caching. Moreover, the fact that forwarding rules are installed by a controller instead of a distributed routing algorithm permits to scale as signaling messages are not flooded in the network but sent directly to the appropriate switches.

The contribution of this paper is three-fold:

- we propose an architecture to enable in-network caching in LTE network by replacing the tunnels that are established to support mobility by a programmable dataplane controlled with an SDN controller;
- we propose an optimisation mechanism formalised with an Integer Linear Program to determine the best caching decision, taking into account the various constraints of the LTE network;
- we propose a closed loop mechanism to dynamically and autonomously adjust caching decision according to the dynamics of traffic inherent to mobile networks.

The paper is organised as follows. Sec. 2 analyses the related work. Sec. 3 precisely defines our SDN architecture that removes the need of tunnelling packets in the backhaul and the technical choices that have to be respected to keep the system compliant with the LTE protocol suite. Sec. 4 formalises with an Integer Linear Program the optimal caching decision scheme for SDN-based LTE networks. Sec. 4.1 proposes a closed-loop algorithm to adjust caching decision to traffic dynamics induced by mobility. Our architecture and caching decision optimization is evaluated with extensive simulations in Sec. **??** that uses the Gauss-Markov model [**?**] to simulate mobility and Sec. 5 concludes our work.

2. RELATED WORK

Previous works, such as [6], have investigated which location for the caches could be the most beneficial for the operators. It was observed that the further from the edge, the more cache hits happen and the easier to manage it is but also the more resources are consumed. It was found out that caching at the base station makes the number of hits decrease due to the small amount of users per base station. However, distributed caching systems were not considered, which would permit to increase the number of cache hits.

Such distributed solutions are used in Information-Centric Networks (ICN). Previous works[7, 12] have focused on ICN and Software Defined Networks (SDN), aiming at merging both technologies for higher caching performances. As such, a transparent content management layer[7] has been designed to integrate ICN based on SDN in mobile networks. This layer analyzes the clients' requests through a proxy and the SDN controller selects the locations where the contents are stored. Then the controller takes the establishment of the rules in charge, to forward the requests and responses, respectively.

This proposal requires a lot of computational resources to implement the SDN logic that manages the transactions to access the cached content. When using this approach, the SDN functionality might become a bottleneck since the mobile network control plane is already supporting a high load on signaling for connection setup i.e. user attachment and handover procedure. Therefore, we propose a caching system with a similar approach based on a content management layer but with a optimized usage of SDN functionality that overcomes the complexity issue, reducing the required computational resources.

3. LTE IN-NETWORK CACHING ARCHITECTURE

The purpose of our LTE in-network caching architecture is to direct the content retrieval requests to the most appropriate cache instead of going directly to the server providing the requested content. Three classes of solutions exist to achieve this goal. First, the network can ask the content requester to change the destination of its request to direct it to the cache instead of the server. This solution is largely adopted by Content Delivery Networks to balance the load between their servers. Drawbacks are that it is specific to the application layer protocol used to request a content and that it requires additional communication between the mobile device that requests a content and the entity that suggests the redirection. However, our objective is to reduce as much as possible the traffic at the edge and in the backhaul. Another option is to ask the base station – the ENode B – to tunnel the traffic directly to the adequate cache for the requested content. Alas, tunnelling prevents in-network caching as intermediate nodes between the ENode B and the cache only see opaque packet passing through them and cannot determine to which content they are related. The possibility we are exploring is to ensure directly at the forwarding level that all packets of the flow corresponding to a particular request are forwarded to the appropriate cache. In this situation, packets are moved in the network on a flow basis instead of a destination basis. The common approach used in SDN to achieve this goal is to dynamically setup a forwarding path for each flow. This approach is the ideal as it gives the highest possible flexibility. In this work, we adopt this last solution.

We identify three components to implement a Software Defined Networking caching system for LTE infrastructure: (*i*) the *Caching* component that is composed of the storage itself and a logic unit deciding where and when content must be cached; (*ii*) the *Forwarding* component that ensures that packets are forwarded to the most appropriate cache; and (*iii*) the *Mapping* component that binds forwarding and caching together.

The role of the mapping component is essential. Indeed, for efficiency reasons the forwarding plane cannot operate at the application layer level. The reason is that forwarding must be accomplished at link rate which is only possible when lookups are performed on fixed-length fields found in well defined locations in packets. We thus have to limit the granularity of the forwarding plane to the lowest layers of the network stack. However, the network layer of a packet only provides information about the server where the content can be retrieved, but nothing about the content itself

and several contents can be provided from the same server. The mapping component thus have to analyse the application layer information contained in the packet to determine the content that is requested and then translate this information into information that can be processed at the networking layer. To do that, the mapping component uses information from both caching and forwarding components. The caching component permits to know which cache must be used to retrieve the content while the forwarding component tells how to reach the cache. With this information, the mapping component annotates packets so that they can be forwarded to their appropriate cache without requiring routers to be aware of the content.

3.1 System implementation

To implement our LTE in-network caching architecture we rely on OpenFlow [15]. OpenFlow implements the SDN concept with a communication protocol that permits to a centralized controller to install specific forwarding rules to the different switches composing the network. A rule associates a predicate to an action. If a packet traversing a switch holds true for the predicate of a rule, the switch executes the action associated to the matching rule, such as switching the packet to a particular port or modifying the packet. While it is possible to setup a different forwarding path to every flow with OpenFlow, it would come at the expense of a high signaling overhead and our objective is to minimize the traffic at the edge and backhaul of LTE networks. We therefore relax this by forwarding packets based on annotations: packets are annotated based on the cache the request should be directed to. The advantages is that forwarding entries are independent of the flow dynamics and can be pre-installed hence minimizing the signaling overhead. The annotation is performed by the mapping component that is functionally equivalent to a deep-packet inspector. The packet is inspected by the mapping element when it enters the network to determine the content it is related and is tagged with the annotation corresponding to the cache it has to be transmitted to. After this first inspection step, the packet is natively forwarded in the network.

Different packet annotation solutions are supported by OpenFlow to annotate packets, such as VLAN tag or MPLS labels. However, they all require every node in the network to support the mechanism. To reduce the costs, we hence decide to rely on native IP forwarding. For that we map a specific object with an IPv6 address. This IPv6 address becomes the temporary identifier of the object. Since we are using the IP fields in a different way, the internal cache network is isolated from the usual traffic. We select an initial IPv6 prefix to cover all objects, which is divided into sub-prefix, one per cache. Those sub-prefixes are assigned to each cache. As a result, the IP address prefix is used for forwarding decisions in the network while the host part of the address is used by the various caching component to identify the content related to the packet without having to inspect the packet. The IPv6 address used as identifier is then unique per cache and per content type. The mapping component is the entry point of the caching infrastructure via which every packet from mobile clients passes. The mapping component is located directly at the ENode-B and inspects each packet to determine the content it is related to. It then queries the caching component to determine the cache to which the packet must be forwarded to and the forwarding component to know the IPv6 prefix attached to the cache. The packet is then forwarded in the network using the IPv6 address.

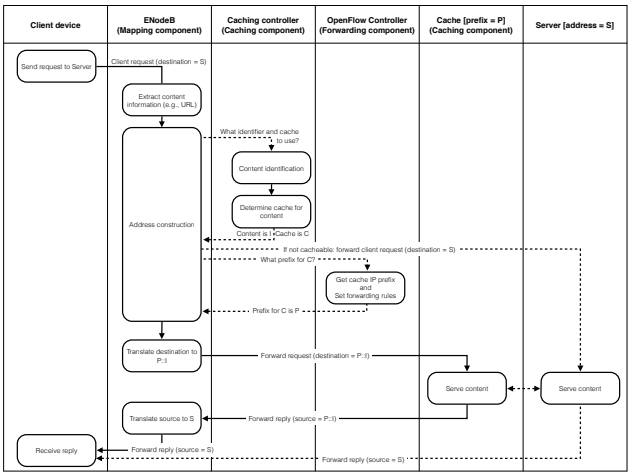

Figure 1: LTE caching architecture workflow

Fig. 1 summarizes the operations.

4. CACHING ALLOCATION

In Sec. 2 a SDN-based architecture to enable in-network caching in LTE networks is presented. In this section, we provide an allocation scheme to optimally decide where to cache contents in the network. We follow an offline approach where the caching logic calculates an optimal location for each object based on their expected demands for the near future.

To that aim, we use an Integer Linear Program formulation.

Let $G = (E, V)$ be a finite directed graph where every edge (i.e., line) $e \in E$ has capacity $c_e \geq 0$ and each vertex (i.e., network node) $v \in V$ has storage memory $m_v \geq 0$. Let O be the set of objects of size s_o. The objective is to find a $|O| \times |V|$ binary allocation matrix A where $A_{o,v} \in \{0, 1\}$ indicates whether or not object $o \in O$ must be stored on the network node $v \in V$.

The storage limitation is accounted with the following constraint:

$$\forall v \in V : \sum_{o \in O} A_{o,v}\ s_o \leq m_v. \quad (1)$$

Line capacity is accounted with

$$\forall l \in E : \sum_{i \in V} \sum_{o \in O} \sum_{e \in V} A_{o,e}\ B_{o,i,e,l} + \left(1 - \sum_{e \in V} A_{o,e}\right)\ B_{o,i,d_o,l} \leq c_l \quad (2)$$

with $B_{o,i,e,l} = (up_{o,i}\ P_{i,e,l} + down_{o,i}\ P_{e,i,l})$ where $P_{s,d,l} \in 0,1$ indicates if edge $l \in E$ is on the shortest path between $s \in V$ and $d \in V$ and $down_{o,n} \geq 0$ (resp. $up_{o,n} \geq 0$) is the expected download (resp. upload) transfer rate observed for object $o \in O$ issued from node $n \in V$. d_o is the location from where object $o \in O$ must be retrieved if not stored in the network.

The first term of Eq. 2 accounts for the bandwidth consumption on the lines on the path to the storage position in the network, while the second term takes into account the traffic for object that have to be retrieved from their original server. It is worth noticing that we neglect the bandwidth consumed to copy objects to their local storage node.

To ensure that an object is stored in only one location in the network, we add the following constraint:

$$\forall o \in O : \sum_{n \in V} A_{o,n} \leq 1. \quad (3)$$

If the objective of using caches in the network is to minimize the network load, then the objective is to find an allocation matrix that minimizes the following function while respecting the constraints given above:

$$\min \sum_{o \in O} \left[\sum_{e \in V} A_{o,e} \left(p_{o,a_o,e} + \sum_{i \in V} \lambda_{o,i}\ p_{o,i,e}\right)\right] + \left[\sum_{i \in V} \left(\prod_{e \in V} (1 - A_{o,e})\right) \lambda_{o,i}\ p_{o,i,d_o}\right] \quad (4)$$

$$\min \sum_{o \in O} \left[\sum_{e \in V} A_{o,e} \left(p_{o,a_o,e} + \sum_{i \in V} \lambda_{o,i}\ p_{o,i,e}\right)\right] + \left[\sum_{i \in V} \left(1 - \sum_{e \in V} A_{o,e}\right) \lambda_{o,i}\ p_{o,i,d_o}\right] \quad (5)$$

where $\lambda o, i$ is the demand for object $o \in O$ issued from node $i \in V$ and $p_{s,d}$ is the cost associated to the path between nodes $s \in V$ and $d \in V^+$ with V^+ being the union of V and the set of serververs. a_o is the location where object $o \in O$ was retrieved from before the relocation.

4.1 Dynamic caching allocation adjustment

The offline allocation model we propose requires to be recomputed periodically to take into account the dynamics of traffic caused by user mobility and the variation of object demand with time. We propose to automatically adjust the time period to the dynamics of traffic with a feedback control algorithm inspired from the additive-increase/multiplicative-decrease (AIMD) congestion avoidance algorithm of TCP [16]. The period to wait until the next relocation is determined when the new allocation is computed. This period is determined from the relative difference between the value of the objective function as computed with the optimal allocation matrix for the current period and the value of the objective function using the previous allocation matrix. If the difference of values is minor (i.e., below a threshold fixed by configuration), the relocation was not necessary and the period is linearly increased (i.e., additive increase with a factor $alpha > 0$) to reduce the relocation frequency. On the contrary, if the values significantly differ, it means that the period was too long between two relocation and the time period is abruptly reduced (i.e., multiplicative increase with a multiplicative factor $0 \leq \beta < 1$) to increase the relocation frequency. The operator can fix the lower and upper bound.

Data: period: relocation period of previous run
Data: A: allocation matrix of previous run
Data: A': allocation matrix for current run
Data: obj(·): objective function for current run
Data: τ: relative difference threshold
Data: min_period: minimum time between two relocations
Data: max_period: maximum time between two relocations
Result: Period of time between two relocations
if *|obj(A) - obj(A')|/obj(A')* $< \tau$ **then**
  period ← period + α
else
  period ← period * β
end
return *min(max_period, max(min_period, period))*

Algorithm 1: Rellocation period computation

5. CONCLUSIONS

Mobile operators are currently interested into deploying cache solutions to bring the contents closer to the users to improve their quality of experience but also to reduce OPEX. To help in the adoption of caches in LTE networks, we propose a caching system architecture leveraging the Software Defined Networking principle. The role of SDN is primarily to enable the possibility of removing the GTP tunnelling, thus removing all limitations to place the cache in the part of the network. We also propose an optimal caching allocation scheme. As traffic and devices are dynamics within LTE networks, we propose a caching relocation system based on simple, hence implementable, feed-back loop algorithm.

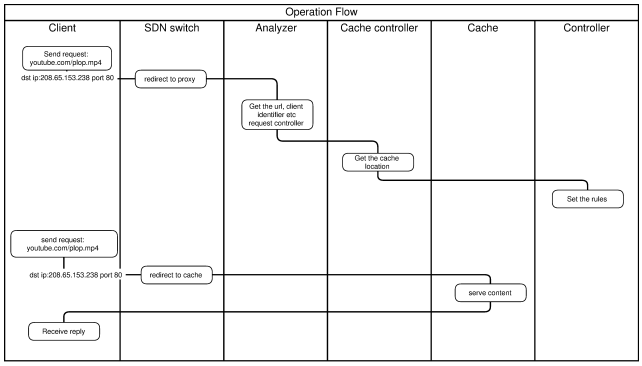

Figure 2: The Operation Flow in the Caching System.

5.1 The Infrastructure of the System

The design of our system includes SDN nodes, which consist of the SDN controller and SDN switch and three additional elements. Those elements are web proxy performing traffic analysis, caches and caching controllers. The system design can be improved by integrating the traffic analyzers with the caches. The caching controller includes a module for interfacing with the SDN controller which will set the rules on the SDN switches. Figure 2 presents the proposed system design and the above mentioned elements.

All the SDN switches are configured with a default rule (e.g. all requests with destination port=80) that will redirect to the proxy the first request of a client to a website that was not already fetched. The proxy analyzes the request, gets the name of the destination host to which the original web request was sent and sends it to the cache controller. The cache controller resolves the host name received from the proxy. The same host name might be associated to multiple IP addresses because the destination web server has associated several IP addresses for load balancing. The cache controller associates the IP addresses received to a cache location. Next, the cache controller interacts with the SDN controller to set new rules in the switches that will redirect to the caches all the HTTP traffic with a destination address that matches one of the IP addresses resolved from the host name. Thus, the system maintains a correspondence between destination IP addresses and the cache locations where the objects are stored. The following request will be directly redirected to the cache storing the content for the destination website.

The advantage of this design is that it permits to easily relocate the object from one cache to another. Apart from transferring the data from one cache to the other, changing the rules on the SDN switches to change the redirection is a simple operation.

5.2 System Implementation

In order to prove the efficiency of the proposed design, Figure 3 shows the testbed we implemented including all the required elements. We performed different test cases using HLS (Http Live Stream) video that was downloaded by the mobile clients. The test cases consist of several users distributed between different base stations and consuming a

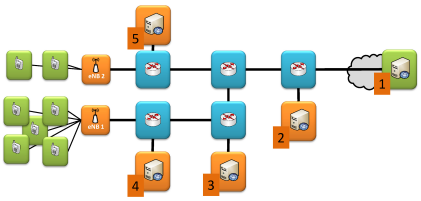

Figure 3: Testbed.

streaming service at 2Mb/s bitrate. In the test cases, we consider two base stations with seven users (5 on the first and 2 on the second). The SDN switches are OpenVSwitch[13] bridges and the caches are instances of TrafficServer[2] with a dedicated module. The analyzer is included in the TrafficServer module in order to make the process more efficient, by reducing the number of intermediaries in case of locally cached content. We run different test cases which consist of adding artificial congestion (i.e. we use netem and qdisc commands to increase the delay in the selected openvswitch links) in some segments of the network to the effect of having different numbers of users in different base stations. The first test case is fetching the objects of the website directly from the public Internet outside the mobile network and this case is used as reference to compare the load generated by the other test cases when content is cached in different locations as shown in Figure 3.

6. RESULTS ANALYSIS

6.1 Performances of the System

In order to quantify the impact of our caching system on the network, we consider the sum of the loads of all the links as network load metric. Thus, a request of 1MB going through 7 links will generate a network load with value of 7MB. This network load metric is used to quantify the reduction of the load in the total network when introducing the caches. This metric also includes the resource consumption required to move contents. We quantify this network load metric as

$$\sum_{i=0}^{N}\sum_{j=0}^{n} x_i * y_{i,j} + \sum_{i=0}^{N}\sum_{j=0}^{n} x_i * z_{i,j}$$

with N being the number of requests issued from all the base stations, n the number of links, x_i being the size of the response, $y_{i,j}$ being 0 if the response does not go through the link, 1 otherwise, $z_{i,j}$ being 0 if the response has been moved between caches through the link, 1 otherwise.

Figure 4 shows the results of the network load metric. We can see the percentage of load reduction achieved for different test cases where we change the location of the caching content as depicted in Figure 3. The results show that obviously locating the content in the cache, which is closest to a higher number of users provides the best load reduction.

Next, we consider the impact on the variation of the network load reduction in relation with the distribution of the users. In this case the number of users remains constant (7 users), but their location changes from one base station to another. Figure 5 presents the network load reduction achieved when changing each cache location based on the mobility of the users. The results show that the same cache can achieve

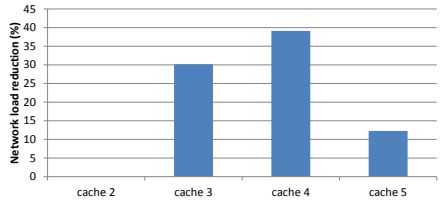

Figure 4: Load Reduction depending on caching location.

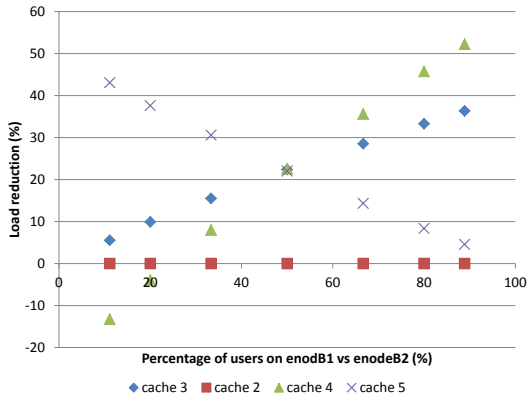

Figure 5: Load Reduction depending on repartition of the users.

both the worst and the best load reduction, since a badly selected location that is not up to date with the users' location and requests, increases the load and degrade the performances of the whole system by increasing the load. This demonstrates the need for short periods in relocation operations, since outdated locations can damage the performance.

6.2 Proposed System Optimizations

6.2.1 Main Process

After extensive testing, we noticed that the proposed design has some flaws. The main drawback is the usage of the IP address to redirect the request to the cache. This IP address is the one obtained when the cache controller resolves the host name of the target web site. Using the IP address does not allow storing different contents from the same website in separate locations depending on their demand. The usage of the IP address allows redirecting the whole website to a cache but not specifically a single object. Next, we propose an improved version of the system design, where we added a feature to map a specific object with an IPv6 address. This IPv6 address becomes the temporary identifier of the object. Since we are using the IP fields in a different way, the internal cache network is isolated from the usual traffic. We select an initial pool of IPv6 addresses, which is divided into several smaller pools, one per cache. Those smaller pools of IPv6 address assigned to each cache can also be subdivided to separate the content types. For example, the n first bytes identify the cache, the y next bytes indicate the type of the object and the remaining bytes identify the object itself. The IPv6 address used as identifier is then unique per cache and per content type. Thus, if we duplicated the content, the same object would be identified differently on different caches. With these IPv6 subnets, the SDN rules

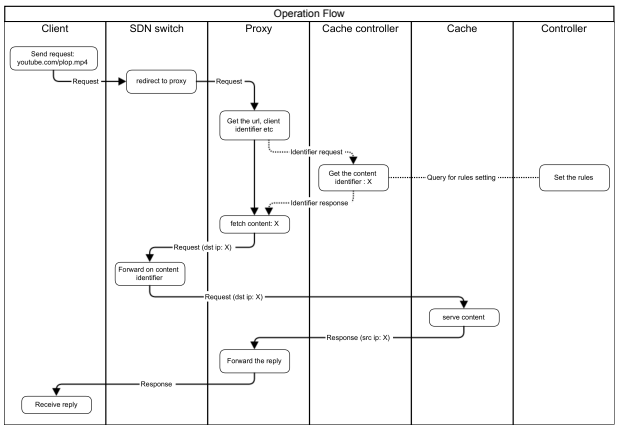

Figure 6: The Operation Flow in the Optimized Caching System.

on the switches can be aggregated per cache (and content type) which allows applying QoS policies to different content objects.

Thus, all the requests from the mobile clients go through the proxy to the analyzer that performs a mapping between the URL and an IPv6 address. Proxies are entry points that, based on the results of the analysis of the requests, will query the cache controller to get the content identifiers. Then, the proxies will forward the requests using those identifiers over the SDN network that will route it to the right cache. In order to reduce the number of operations needed, the IPv6 address assigned as identifiers are based on the chosen location of the address pool. This will allow the requests can be routed by static rules using masks to match only the bytes identifying the cache are set on all the SDN switches to forward each pool to the right cache. When retrieving a new object, the analyzer would be given its identifier and would thus be able to fetch it without the need for new rules.

It is worth noticing that the time spanned between two relocations must be selected carefully and we let this study for future work.

We propose three methods for relocating an object. In the first one, the relocation is done by updating the analyzers from the cache controllers so that the following requests will be forwarded with a new content identifier to the new cache location directly. This method is referred to as a Non-SDN relocation method in the remaining. The second method is using the SDN capabilities of the network. It consists of inserting a new rule on the switches with a higher priority matching the exact content identifier to route it to its new location; This rule will match instead of the rules routing the identifier pool to the previous cache, at least until the analyzers entries for the relocated object gets outdated. The rule will have a timer so that the identifier can be used again later. When its entry will get outdated, the analyzers will fetch the new identifier from the cache controller. Both methods have advantages and drawbacks, so we propose a third method that is an hybrid version, selecting the most efficient method in terms of operations for every relocation.

7. IMPACT OF SYSTEM OPTIMIZATION

7.1 Simulation Setup

In order to determine the impact of those improvements, we have set up a simulation. We simulate a network of 16 base stations on which we randomly distribute users (up to 10 per base station), with 31 switches, all of them SDN nodes with caching capabilities. The network is pyramidal and we attribute increasing costs to the links as getting closer to the edge. Figure 7 presents the simulated network.

Let the time be discrete, the requests be unrelated to each other and N be the number of objects on the internet, ranked by popularity order. The first object is the most popular and the last is the least one. Let all the users send a request at every time slot. Let a be the identifier of the base station and x_a be the number of users on the base station a. Let t be the interval between relocations.

We assume the requests of the users follow a Zipf-like distribution: the probability that the user fetches the object i (position in the popularity list) is

$$P_{N,\alpha}(i) = \frac{1}{\sum_{j=1}^{N} \frac{i^\alpha}{j^\alpha}}$$

The value of α might be different for each network. We will arbitrarily set this variable to 0.9[5] . Besides, we limit the number of existing contents that can be fetched by the users to 1500 items such as web pages or images, and we consider that the average size of the responses is 1MB, considering the repartition of size of the requests on the web[14].

7.2 Limitations of the simulation

The main limitation of our model is about the number of objects used which for simplicity is restricted to a total of 1500 items. Our simulation is based on the usage of requests based on a Zipf-like law. Laws of this kind are directly dependent of the number of contents. The arbitrary choice of α does not reflect the reality either since this value varies from a single network to another. Finally, we do not focus on the physical constraints such as processing and storage capacity in the nodes. In case of storage problems, previous work has focused on replacement models to ensure the highest cache hits[3].

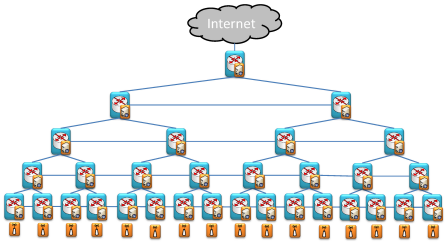

Figure 7: Simulated Network.

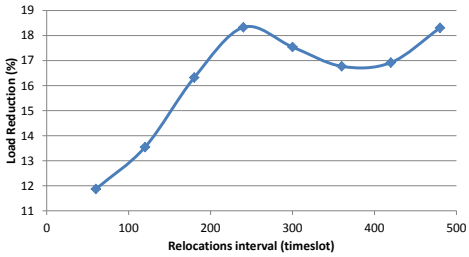

Figure 8: Network Load Reduction.

7.3 Simulation Results

Using the network load metric discussed earlier, Figure 8 presents the load reduction quantifier depending on the relocations interval.

The load reduction presents three distinct phases. The first phase is increasing the efficiency due mainly to the reduction of the number of relocation which leads to reduced bandwidth consumption for the relocations. The number of requests also increases and is thus more representative of the whole behavior of the clients for the most requested contents, which leads to a better placement. In the second phase, the efficiency is decreasing due to the lack of reactivity of the system. This in practice means that when the period between relocations is high the users request content that is not properly allocated. This phase is a transition between the state where we prefer a reactive system that detects small changes in the user's requests and is adapting to it compared with a system relying more on the trends as used in the third phase. In the third phase the network load reduction increases again and we consider the reason is that users are fetching the same content which has been reallocated properly and does not need to be changed.

We focused then on the different relocation methods permitted by our system. Considering that updating a proxy or a switch is a single operation, we compare the number of operations needed for each method. Figure 9 displays the number of relocation operations per period of relocation and shows that the hybrid method is the most efficient. We figured out that the relocation method not using SDN performs slightly better with small relocation periods but far worst with larger relocation periods than the method using

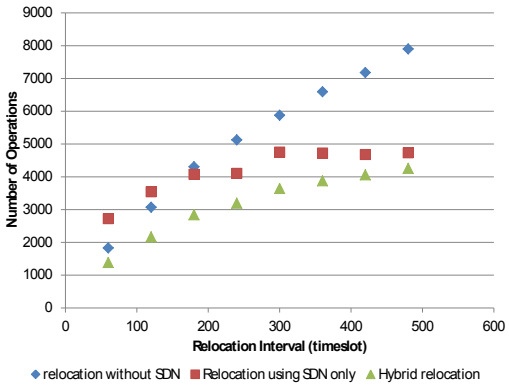

Figure 9: Comparison of the relocation Method.

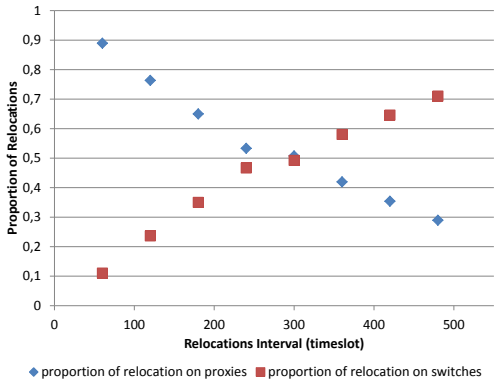

Figure 10: Proportion of each method in the relocation processes.

only SDN. The high relocation period induces many requests from most base stations on most of the contents, making it more expensive to reconfigure than to configure the switches in general. Those results are biased by the small size of the network used. In order to correct it for real networks, a coefficient adapted to the network situation (depending on the SDN controller load for example) can be set in order to promote a method over the other. The third method then provides a way to take advantage of both methods, even if the load on the controller is already significant.

Considering that the cost of an operation on a switch is similar to the cost of an operation on the analyzer (both are reconfigured over the network by a controller), Figure 10 presents the proportion of relocations processed with the SDN method and the non-SDN method. As expected of the previous results, the use of the non-SDN method is much higher with short relocation periods and much lower with high relocation periods than the use of the SDN method.

Thus, Figure 11 presents the average number of operations for each method, and separated for the third one, the operations on the analyzers (proxies) and on the switches. The hybrid average number of operations per relocation is lower than the minimum average number of operations per relocation of the two methods separately, since for any cases when it would be higher for the non-SDN method, the relocation

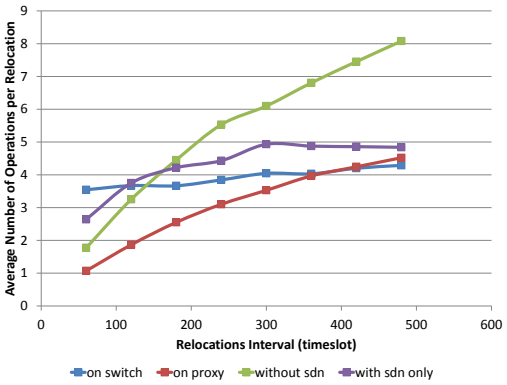

Figure 11: Average Number of Operation per relocation.

is performed using SDN and vice versa.

8. CONCLUSIONS

Mobile operators are currently interested into deploying cache solutions to bring the contents closer to the users. Therefore we propose a new system to provide in-network caching in the backhaul networks, using the capabilities of Software-Defined Networking. We first present a solution and analyze its performances and then based on the obtained results we design an optimized solution which is validated with simulations. The usage of SDN allows us to dynamically relocate the contents from one cache to another located in any part of the LTE network. Our method uses either a feature of the proxies or the SDN capabilities to process to the relocation, depending on their efficiency. This solution performs better than any of those methods separately. Our design permits to place the content at optimal location, minimizing the bandwidth consumption and thus the costs. The relocation method proposed permits to keep the content at their optimal location with the lowest costs. This permits to improve the quality of experience (QoE) and save power since the content is downloaded faster. Moreover, the system is not dependent on a single relocation decision algorithm. The system can use any algorithm implemented for that purpose. This solution is aimed at offering new caching possibilities for mobile operators in order to reduce the resource consumptions the most possible. The obvious solution would be to place the cache closer to the user but the results show that based on the number of users a more centralized location of the cache has an impact on the overall network load. Another important remark is the relevance of SDN to deploy the proposed solution. The role of SDN is primarily to enable the possibility of removing the GTP tunneling, thus removing all limitations to place the cache in the part of the network. Besides this feature, SDN should not have any major role in caching but the results show that using SDN can still reduce the number of operations required to reallocate the cache. A final result worth considering is the intervals between cache reallocation which seems to be dependent on the content. This is left for further work since it requires further analysis in order to identify the optimal interval between cache reallocation.

9. ACKNOWLEDGMENTS

This work was done in the SIGMONA research project funded by the Finnish Funding Agency for Technology Innovation and industry. This work benefited from the advice of Jesus Llorente Santos and Vicent Ferrer Guasch.

10. REFERENCES

[1] AltoBridge. Data-at-the-edge, 2013.

[2] Apache Foundation. Trafficserver, 2013.

[3] A. Balamash and M. Krunz. An overview of web caching replacement algorithms. *Communications Surveys Tutorials, IEEE*, 6(2):44–56, Second 2004.

[4] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E.-D. Schmidt. A virtual sdn-enabled lte epc architecture: A case study for s-/p-gateways functions. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, pages 1–7, Nov 2013.

[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134 vol.1, March 1999.

[6] X. Cai, S. Zhang, and Y. Zhang. Economic analysis of cache location in mobile network. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 1243–1248, April 2013.

[7] A. Chanda and C. Westphal. Contentflow: Mapping content to flows in software defined networks. *arXiv preprint arXiv:1302.1493*, 2013.

[8] J. Costa-Requena. Sdn integration in lte mobile backhaul networks. In *Information Networking (ICOIN), 2014 International Conference on*, pages 264–269, Feb 2014.

[9] M. Hibberd. Tellabs "death clock" predicts end of profit for mnos, February 2011.

[10] S. H. Kim. Intelligent caching solution to address quality-of-experience and data deluge in mobile multimedia delivery networks, 2013.

[11] L. Li, Z. Mao, and J. Rexford. Toward software-defined cellular networks. In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pages 7–12, Oct 2012.

[12] X. N. Nguyen, D. Saucez, and T. Turletti. Efficient caching in content-centric networks using openflow. In *INFOCOM, 2013 Proceedings IEEE*, pages 1–2, April 2013.

[13] OpenvSwitch Community. Openvswitch, 2014.

[14] S. Souders. Http interesting stats, 2014.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. 2008. OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, 2 (March 2008), 69-74.

[16] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," Internet RFC 5681, Sept. 2009.