Cache peering in multi-tenant 5G networks

Konstantinos V. Katsaros, Vasilis Glykantzis, George Petropoulos
Intracom SA Telecom Solutions,
Email:{konkat, vasgl, geopet}@intracom-telecom.com

***Abstract*—Building on the adoption of the Network Functions Virtualization (NFV) and Software Defined Networking (SDN) paradigms, 5G networks promise distinctive features including the capability to support *multi-tenancy*. Virtual Network Operators (VNOs) are expected to co-exist over the shared infrastructure, realizing their network functionality on top of virtualized resources. In this context, we observe the emerging opportunity for establishing synergies between co-located tenants of the infrastructure. In this work, motivated by the rapidly increasing mobile content delivery traffic, we focus on synergies in the form of cache peering relationships between co-located VNOs. Caches co-located within the same (micro-)data centers, benefit from content opportunistically cached at their peers, taking advantage of the shared nature of the infrastructure to reduce latencies and traffic overheads. In this paper, we take a close look at the management and orchestration requirements of the envisioned services, illustrating our on-going approach.**

***Index Terms*—5G, NFV, SDN, multi-tenancy, management, orchestration, caching**

I. Introduction

5G networks are expected to adopt the NFV paradigm, opening the way to a series of benefits. By realizing key network functions (*e.g.*, caches, firewalls, DPI, etc.) on top of virtualized compute, storage and network resources, NFV promises a series of benefits such as the reduction of associated CAPEX/OPEX due to the shared nature of the equipment, as well as the increased flexibility in network management and programmability. In the context of 5G networks, these still shaping capabilities, facilitate a series of key features and advances, including the assembly and management of isolated sets of virtual resources (often termed as *network slices*) tailored for the operational needs of third parties *i.e.*, the tenants of the shared infrastructure. This capability further supports the emergence of Virtual Network Operators (VNOs), enabling multi-tenancy scenarios, where multiple VNOs share the same physical infrastructure.

At the same time, the rapidly increasing volumes of content delivery traffic in mobile networks are expected to put the virtual infrastructure of VNOs under severe stress [1]. Caching solutions are considered as an important countermeasure; already, mobile operators cache content traversing their network, transparently to content providers[1], as a form of network acceleration function reducing traffic and latency overheads [3]. However, introducing this solution in multi-tenancy environments, bears the risk of excessive redundancy *i.e.*, the same content is cached by the various VNOs, reducing the effective capacity of the overall physical infrastructure.

[1]Standardization activities have already engaged in the establishment of mechanisms overcoming the emerging considerations related to traffic encryption (HTTPS) and intellectual property rights (IPR) [2].

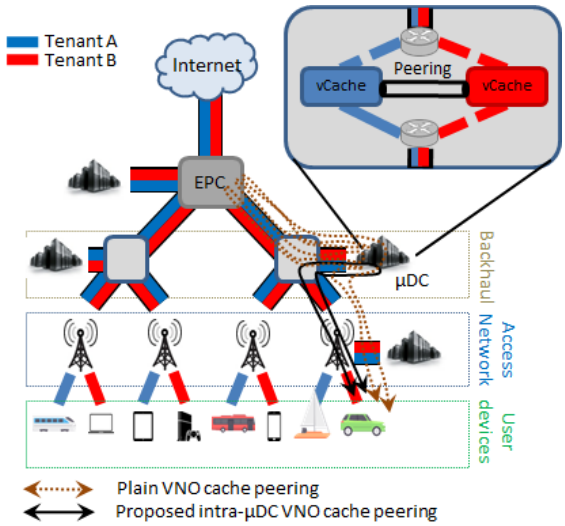

Figure 1: Cache peering in multi-tenancy scenarios. Intra-μDC communication between co-located cache peers optimizes traffic, bypassing the EPC.

In this context, we consider the emergence of cache peering relationships between tenants of the same infrastructure (VNOs). The envisioned VNO synergy, builds on the shared nature of the underlying resources *i.e.*, caches within the same (micro-) data center (μDC) exchange content already cached within the μDC, so as to avoid redundant communication with the content origin server. However, traffic isolation between VNOs results in inefficient communication paths between peering caches as all inter-VNO traffic typically traverses the corresponding mobile network gateways, within the evolved packet core (EPC) (see Figure 1). Content cached in a co-located peer needs to traverse the entire network before reaching the same μDC. This in turn calls for the orchestration of an optimized, secure, intra-μDC communication path (see Figure 1).

In this paper, we take a first step in identifying a series of management and orchestration (MANO) requirements for the realization of the envisioned cache peering VNO synergies, such as the aforementioned optimized intra-μDC, inter-VNO communication, and further illustrate our on-going work in addressing the identified challenges. We start from challenges related to the NFV nature of our context *e.g.*, cache auto-scaling (Section II), and proceed with particular requirements for the establishment of optimized cache peering links (Section III), including security aspects. We discuss related work in Section IV and conclude in Section V.

II. vCache Management and Orchestration

A. Baseline setup

As shown in Figure 2, the envisioned caching services build on the ETSI NFV architecture framework [4][2]. Caches are realized as virtualised network functions (VNFs) within compute hosts (*i.e.*, servers) of the virtualised infrastructure (VI). For our work we employ the Squid cache implementation[3], a mature and widely adopted solution, though our setup does not depend on Squid-specific features. Following a typical μDC configuration employing OpenStack[4], our baseline setup also involves a Network Node responsible for steering traffic within the μDC. Apart from vCaches, a load balancer (LB) VNF is further considered, to balance the load across the available vCache instances (see subsection II-D)[5].

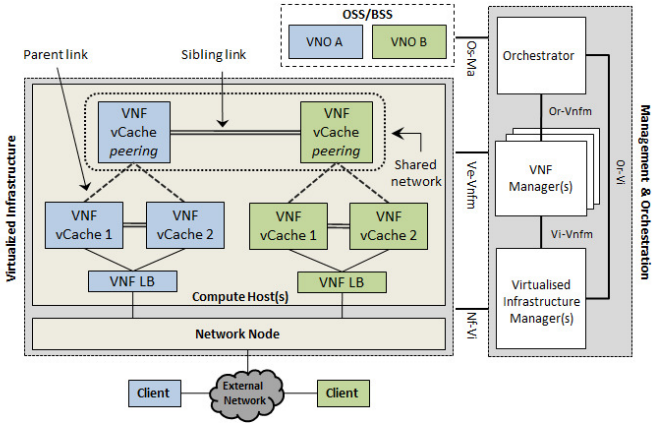

Figure 2: Baseline vCache (peering) setup.

B. Cache placement

The NFV MANO framework is responsible for the overall placement and dimensioning of the vCaching service. The virtual character of the underlying compute, storage and network resources provides the flexibility for the dynamic realization/placement of the vCache instances across the 5G network infrastructure. The emerging challenges relate then to the selection of the network locations for the vCaches, as well as the amount of allocated resources. This is a typical Facility Location Problem (FLP), that has been extensively investigated in the past (see [5] and references therein). In the considered NFV-enabled environment, the Orchestrator component becomes responsible for solving this optimization problem. To this end, apart from network topology information, extensive traffic profiling and μDC monitoring data is required across the (virtual) network so as to identify the demand and availability of resources at each area of the network, thus deriving the input parameters for the formulated FLP.

[2]We omit reference to Element Managers for clarity reasons.
[3]http://www.squid-cache.org/
[4]https://www.openstack.org/
[5]The scalability of LB itself is consider out of scope of this paper

C. Traffic handling

Given an (initial) placement decision, the baseline vCache setup further requires the configuration of the network so that traffic traverses the vCaches. Typical mechanisms include the redirection of user requests, through means of client proxy cache configuration, and/or the transparent delivery of all user traffic to the caches via means of network configuration. However, these methods are dimmed to fail or prove inefficient in the presence of GTP tunnels used to deliver traffic to the mobile network gateway/EPC *i.e.*, users' traffic always reaches the mobile network gateway where it gets decapsulated.

A workaround has been proposed by Rodrigues *et al.*, based on a splicing network function [6], which allows decapsulation (and re-encapsulation) to take place inside the network; in our case at the μDC. Still however, by subsequently applying client proxy configuration or other traditional mechanisms, all traffic reaches the cache and consequently cache misses result in delay penalties. Such penalties raise particular concerns in the considered NFV environment, as traffic has to traverse the entire virtualized network infrastructure of the μDC, before reaching a vCache. This calls for a more agile approach, where vCaches are reached mainly/only by flows that are likely to result in a cache hit. The emergence of SDN has introduced the missing agility and flexibility in dynamically identifying the traffic to be steered towards caches, subject to content availability and load conditions [6], [7]. Nevertheless, such mechanisms build on content availability lookups upon each content request (similar in nature to DNS redirections in CDN environments [8]). As such, they are prone to significant consumption of the shared switching fabric resources, *i.e.*, at the Network Node (Figure 2). In the context of multi-tenancy, this raises concerns regarding the overall forwarding performance, even for VNOs with no vCaches in operation. Alternatively, realizing these solutions on a VNF level, *i.e.*, allocating VM resources for the content availability lookups, would confine the impact within VNO resources. However, in this case traffic steering decisions are taken once flows have already traversed the virtualized infrastructure towards the VNF; flows that eventually bypass the vCache, still pay the corresponding traversal delay penalties.

Considering this tradeoff, our on-going work focuses on the establishment of traffic interception flow rules on the Network Node, however avoiding the aforementioned lookup operations and the associated overheads. Instead, the definition of interception rules relies on the pro-active processing of vCache access logs, with the purpose of identifying target IP addresses prone to cache misses *e.g.*, popular web sites serving personalized content. Our future work plans include a detailed assessment of this design, focusing on the accuracy of the traffic interception rules, their memory footprint and impact on lookup and latency savings.

D. Auto-scaling

The advent of virtualization and NFV/SDN capabilities brings resource elasticity features to vCaching, realized

through auto-scaling events *i.e.*, additional resources are dynamically (de)allocated subject to monitored load *e.g.*, CPU utilization for cache index lookups. As scale-up events *i.e.*, increasing allocated resources on-the-fly, are currently not readily supported by virtulization technology, auto-scaling translates to scale-out events *i.e.*, new vCache instances are created and the LB is updated to appropriately distribute the load. Careful consideration is required *wrt.*, the state of the new vCaches instances *i.e.*, cache index and content. We build on network proximity of cache instances to bring content from existing ones instead of the content server. This is accomplished by configuring peer/sibling links between the newly scaled-out and the existing vCache instances. Sibling links allow the vCaches to exchange availability information, content requests and the content itself. Keeping local copies of the content at each cache sibling results in potentially multiple cached copies of the most popular content items. However, maintaining disjoint contents across sibling caches, so as to reduce redundancy, hinders load balancing *i.e.*, all requests for a certain item reach the same vCache instance, while it complicates scale-down events *i.e.*, purging content of underutilized vCaches.

III. vCache Peering Management and Orchestration

A. Peering traffic management

Based on the setup described in Section II, each VNO is in the position to provide caching support within its network slice. The established network configuration does not allow traffic to cross VNO borders *e.g.*, an HTTP request of a user in VNO A can never reach any VNO B vCache, and both VNOs' vCaches can only potentially communicate with each other via the EPC (see Figure 1), even if they are instantiated within the same Compute Host. Establishing a cache peering relationship, then calls for the careful configuration of the network environment, adhering to a range of requirements.

The main objective is to allow peering vCaches to communicate so as to exchange content availability information, cache requests and content (Req.1). The provided solution however should not allow any other form of traffic to traverse the inter-VNO communication link (Req.2). The reason is that a peering agreement requires a well defined interface, over which the aforementioned control and data plane peering traffic is solely exchanged, avoiding misuse of the established communication link/network *i.e.*, VNOs should not be allowed to directly offload user traffic to peering vCaches, so as to reduce local resource consumption. In the same vein, the authentication/authorization of the involved vCaches is required to mitigate any hijacking of the peering communication link/network from malicious third parties *i.e.*, malicious tenant exploiting VNO network configuration vulnerabilities to make unauthorized use of the peering vCache resources (Req.3). At the same time, the communication between the peering vCaches requires a low latency and high bandwidth communication link/network so as to avoid delay penalties in the discovery and delivery of the requested content from peering vCaches, thus preserving the key benefits of and motivation for peering. As such, communication is required not to involve the EPC but rather remain within the borders of the VI (Req.4).

Towards these ends, the proposed solution includes a mixture of μDC network, VM and application level solutions. On the network side, our approach foresees the creation of a shared network between the involved vCaches. This network is shared exclusively by the involved vCache instances of the peering tenants. The Role-Based Access Control (RBAC)[6] feature introduced in Liberty version of OpenStack, enables tenants to grant access to network resources for specific other tenants. Building on this feature, a VNO first creates a network instance, further also configuring its own vCache(s) by adding to them an interface to the new network. Subsequently, the VNO grants access privileges for the shared network to its peer VNO. The latter VNO is allowed then to attach its vCache(s) to the shared network. The physical infrastructure operator is required to mediate in this process by enabling the exchange of VNF location and configuration information between peering VNOs. The resulting configuration satisfies Req. 1 and 4. To further satisfy Req. 2 and 3, VM and application level configurations come into play. The configuration of Squid first includes the establishment of the peering/sibling link, through the `cache_peer` directive[7], which allows the specification of the peering vCache IP/hostname and listening ports[8]. Satisfying Req. 2, goes through the `iptables` configuration of a vCache, dropping all not legitimate input traffic *e.g.*, traffic destined to a non-peering port[9]. Satisfying Req. 3, goes through the appropriate configuration of access control mechanisms supported by Squid (*e.g.*, `login` configuration options of the `cache_peer` directive) as well as other cache implementations *e.g.*, Varnish[10].

B. Resource accounting

As the establishment of a vCache peering link realizes a business agreement between the VNOs, monitoring and controlling the amount of resources devoted to a peering agreement becomes particularly important, as it allows VNOs to ensure the symmetry of the peering link in terms of resource consumption. This means that increased load on the peering link should not affect the performance of local vCaching (Req.5). Towards this end, our preliminary work investigates the introduction of a separate vCache instance (marked as *peering* in Figure 2). In the simplest setup, a peering/sibling link is established between the *peering* vCaches of the involved VNOs. *Peering* vCaches are configured as parents of all local VNO vCache instances. When a content request first reaches a local vCache through the LB, the peering links

[6]http://docs.openstack.org/liberty/networking-guide/adv-config-network-rbac.html

[7]http://www.squid-cache.org/Doc/config/cache_peer/

[8]Cache peering is based on standardized protocols, such as ICP [9] *i.e.*, application-level peering is not Squid-specific.

[9]A malicious peering VNO can still though offload all its HTTP traffic.

[10]https://www.varnish-cache.org/docs/4.1/users-guide/vcl-syntax.html

between the local vCache instances are used to discover the content locally. If the content is not available, the request reaches the parent *peering* vCache, which in turn may query all peering caches at other VNOs. If the content is not found there either, it is brought from the content origin, cached locally (to serve future peering and local requests) and subsequently forwarded to the local vCaches. *Peering* vCaches only serve a request if the content is locally available, thus no query from a peering VNO ever reaches a non-*peering* local vCache. Excessive load from peering VNOs can be monitored, identified and throttled[11] on a *peering* vCache level. However, overloading of the *peering* vCache may still affect local vCaching, as all content is served to local vCaches through the *peering* vCache. We examine an alternative setup (not shown due to space limitations), where content is asynchronously pre-fetched to *peering* vCaches, from the local vCaches, achieving better performance isolation at run-time *i.e.*, pre-fetching takes place asynchronously allowing for the appropriate scheduling to minimize its impact on local vCaching. In this case, the cached content in the peering vCaches is not always up to date, while complexity is introduced for the prefetching mechanism. Towards an efficient solution, our current work focuses on the modeling and quantification of the illustrated tradeoffs.

IV. Related Work

Caching has been identified as a key function, right from the beginning of the NFV concept [10]. Since then, several commercial NFV-enabled solutions have appeared, including caching as a key building block of a broader-CDN oriented solution e.g., [11]. Additionally, the currently on-going 5G PPP Phase 1 EU H2020 research projects put substantial effort in integrating and enhancing the NFV paradigm within the 5G landscape [12]. However, to the best of our knowledge, no commercial solution or research effort focuses on the particular challenges of cache peering in multi-tenancy environments. It is also worth noting that, in the recent past, a series of efforts have been devoted in enabling the extension of (caching) service footprint through peering, in the context of Content Delivery Network interconnection (CDNi) [13]; however, these efforts aimed at the design of interfaces between (application level) CDNs, rather than building on the emerging NFV capabilities and the overall integration of IT resources within the 5G network infrastructure.

In all, we consider the proposed cache peering approach as a step beyond the mere NFV-based instantiation of virtualized caches, focusing on the opportunities brought in the field by virtualization and network programmability, namely, for multi-tenancy. This brings corresponding opportunities for new business models for the NFV-enabled cooperation between VNOs, in an analogy to inter-domain traffic peering agreements [14]. However, cache peering further involves the implicit sharing of compute and storage resources, calling for the identification of the related challenges and appropriate approaches. Our work here aims at taking a first step in this direction.

[11]The *Delay Pools* feature of Squid, provides the means to this end. [Online]: http://wiki.squid-cache.org/Features/DelayPools

V. Conclusions

In light of the emerging NFV and SDN technical advances, and the corresponding capability to support multi-tenancy in 5G networks, in this paper we propose the realization of cache peering relationships between VNOs. The proposed peerings realize a synergy between VNOs, that builds on the shared nature of the underlying infrastructure to lower perceived latencies and traffic overheads. We envision similar synergies in the context of the Mobile Edge Computing (MEC), building as well on the virtualization capabilities at the edge of the network to collaboratively enhance service to end users. Our current work includes a detailed performance evaluation of the proposed approach, assessing the impact of virtualization, GTP de-tunneling, auto-scaling and traffic handling mechanisms.

Acknowledgment

This work has been supported by the CHARISMA project, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 671704.

References

[1] CISCO, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20152020," 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.pdf

[2] M. Thomson *et al.*, "An architecture for secure content delegation using http," Working Draft, IETF Secretariat, Internet-Draft draft-thomson-http-scd-00, March 2016. [Online]. Available: http://www.ietf.org/internet-drafts/draft-thomson-http-scd-00.txt

[3] S. Woo *et al.*, "Comparison of caching strategies in modern cellular backhaul networks," in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '13. New York, NY, USA: ACM, 2013, pp. 319–332. [Online]. Available: http://doi.acm.org/10.1145/2462456.2464442

[4] ETSI, "GS NFV 002 V1.1.1, Network Functions Virtualisation (NFV); Architectural Framework," 2013.

[5] Y. Wang *et al.*, "Optimal cache allocation for content-centric networking," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, Oct 2013, pp. 1–10.

[6] M. Rodrigues *et al.*, "Enabling transparent caching in lte mobile backhaul networks with sdn," in *Proc. IEEE INFOCOM WKSHPS*, April 2016, pp. 724–729.

[7] P. Georgopoulos *et al.*, "Cache as a service: Leveraging sdn to efficiently and transparently support video-on-demand on the last mile," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Aug 2014, pp. 1–9.

[8] J. Pan *et al.*, "An overview of dns-based server selections in content distribution networks," *Computer Networks*, vol. 43, no. 6, pp. 695 – 711, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128603002937

[9] D. Wessels and K. Claffy, "Internet Cache Protocol (ICP), version 2," RFC 2186 (Informational), Internet Engineering Task Force, Sep. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2186.txt

[10] "Network Functions Virtualisation Introductory White Paper," 2012. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[11] Qwilt, "NFV-based Caching and Acceleration Solution to Power Video on Mobile Networks," 2014. [Online]. Available: http://qwilt.com/qwilt-launches-nfv-caching-acceleration-solution/],

[12] 5G PPP, "5G PPP Phase 1 Projects," 2016. [Online]. Available: https://5g-ppp.eu/5g-ppp-phase-1-projects/

[13] G. Bertrand *et al.*, "Use Cases for Content Delivery Network Interconnection," RFC 6770 (Informational), Internet Engineering Task Force, Nov. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6770.txt

[14] C. Labovitz *et al.*, "Internet inter-domain traffic," in *Proc. of ACM SIGCOMM*, 2010, pp. 75–86.