Cache peering in multi-tenant 5G networks

Konstantinos V. Katsaros, Vasilis Glykantzis, George Petropoulos

Intracom SA Telecom Solutions,

Peania, 19002, Greece

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 the form of cache peering relationships between co-located VNOs. Upon a cache miss, co-located caches benefit from content cached at their peers, taking advantage of the shared nature of the infrastructure in reducing latencies and traffic overheads. Our approach allows VNOs to autonomously manage their peering links without the involvement of the infrastructure operator.**

Index Terms—**5G, NFV, SDN, multi-tenancy, 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, IDS, 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, 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. In turn, this capability supports the emergence of Virtual Network Operators (VNOs), enabling multi-tenancy scenarios, where multiple VNOs are realized on top of the same, shared physical infrastructure.

Typically, communication between tenants is expected to take place by traversing the corresponding mobile network gateways, since VNOs will be realized as entirely distinct networks. However, the shared nature of the underlying infrastructure presents the opportunity for shortcuts in this communication. As network functions of different VNOs may reside on the same IT infrastructure i.e., micro-data center (μ-DC), inter-VNO traffic can be exchanged locally, thus promising significant benefits in terms of end-to-end latency and overall traffic overheads [1]. Building on this observation, we focus on

the particular case of caching, proposing the establishment of cache peering relationships between *co-located* VNOs. In the envisioned environment, VNOs instantiate transparent virtual caches (vCaches) to reduce traffic overheads and improve performance for their users. Co-location takes the form of vCaches instantiated within the same μ-DC, either on the same or different compute hosts (i.e., physical servers). Cache peering then takes advantage of the proximity of vCaches i.e., upon a local cache miss, peering vCaches mutually benefit from the content cached at co-located VNOs. As illustrated in Figure1(a), in the absence of cache peering, a cache miss results in the traversal of the virtualized infrastructure (VI) 4 times, before the content can be delivered to the end user i.e., a content request exits the VI to reach the content origin, bringing the content to the vCache before exiting the VI again to reach the end user. In contrast, cache peering allows content to be locally fetched from a co-located vCache, promising substantial traffic and latency savings. What is more, the inherently low latency/high bandwidth μDC environments are expected to facilitate communication between peering vCaches, including both the exchange of content availability information and the delivery of content.

The envisioned setup contributes to the emergence of new business models, for the NFV-enabled cooperation between VNOs, in an analogy to inter-domain peering agreements between autonomous systems in the Internet [2]. However, the considered environment presents increased complexity. Cache peering involves the implicit sharing of compute and storage resources, in addition to network resources. Moreover, VNOs are expected to pursue their autonomy in managing their (business) peering relationships, consequently calling for technical solutions that do not require the intervention/involvement of the (third party) infrastructure operator.
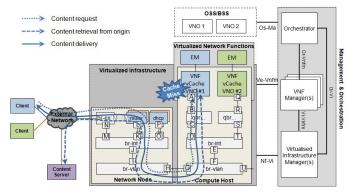
In this paper, we report on our ongoing work towards the realization of cache peering relationships in the aforedescribed context. We present technical details for the configuration of the proposed solution for a typical OpenStack-supported μ-DC environment, identifying key challenges related to virtualization overheads, network slice traffic isolation, as well as the tight control of resource consumption.
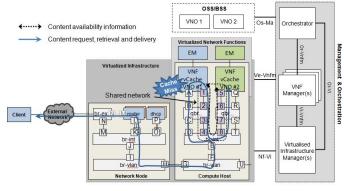
II. BACKGROUND

A. NFV and multi-tenancy

The NFV concept has been under intense investigation by the research community during the last years, largely building on a series of architecture specifications produced by the ETSI standardization body [3], as well as a series of open source

---

[1]This operation raises challenges related to the GTP tunnels used to deliver traffic to the mobile network gateway. Potential workarounds have been proposed in literature [1]. This issue is consider out of this paper's focus area, which is centered around the particular issue of cache peering.

(a) Cache miss: no cache peering

(b) Cache miss: peering with co-located cache

Figure 1: Caching in NFV: cache misses and the effect of cache peering

tools e.g., Open Source Mano (OSM)[2]. Figure 1 provides an illustration of this architecture. The management and orchestration (MANO) plane provides infrastructure operators with the capability to manage their virtualized infrastructure (VI), allocating resources for the realization of virtual network functions (VNFs), in the form of virtual machines (VMs), and interconnecting them with the rest of the (virtual) network infrastructure, into end-to-end network services. In the context of 5G multi-tenancy support, these services span the entire network and IT infrastructure of the infrastructure operator, comprising the network slices allocated to each instantiated VNO.

Taking a closer look at the VI, and in order to understand the technical implications of MANO operations in the targeted environment, we adopt a typical[3] OpenStack configuration, also shown in Figure 1. The IT and network resources of the VI are organised in Compute and Network hosts/nodes. Compute nodes (or hosts) host VNFs in the form of VMs. Network nodes are responsible for handling ingress/egress traffic, isolating and optimising traffic flow across tenants.

This is accomplished by a series of Layer 2/3 forwarding nodes (e.g., Open vSwitch instances (OVS)[4]). qbr switches are introduced to handle security issues, while br-ex is responsible for handling traffic from/to the external network. The br-int switch has a central role in switching packets on a node level. The br-vlan switch is responsible for handling traffic labeled with 802.1Q VLAN tags. A separate VLAN ID can be used for each VNO to isolate traffic at Layer 2.

B. Caching

Content caching has been intensively investigated in the past, with a particular focus on Web (HTTP) caching (e.g., [4]). Caches are a typical part of networks and/or content delivery networks (CDNs) in the form of middleboxes/dedicated servers. Traffic is typically intercepted by caches via means of user request redirection, either in the form of DNS redirections, in the case of CDNs, or via the end user proxy cache configuration. Both cases are associated to penalties related to the DNS signaling latencies and complexity and/or the cumbersome configuration of user devices. In an alternative approach, transparent caching is building on the network configuration, to forward traffic to a cache instance.

Taking advantage of distributed cache resources, cooperative caching schemes have been developed, enabling the exchange of cached content by remote caches. This exchange is realized with the establishment of relationships between the caches including *cache peering* relationships (alternatively, peering caches are known as *siblings*). When a content request results in a cache miss i.e., the content is not locally available, a cache can direct the request to a sibling cache. The receiving sibling cache responds either with the content or with a failure message. Sibling caches indicate the availability of content to each other, either proactively e.g., exchanging compact representations of their cached contents[5] or reactively (i.e., upon a request) by directly querying each other [5].

III. CACHE PEERING FOR 5G TENANTS

A. Baseline tenant configuration

Enabling cache peering in an NFV-enabled multi-tenant environment, first goes through the end-to-end network service orchestration, including the realization of the caching service at each tenant separately. This first involves the configuration of the network so as to ensure the isolation of tenant traffic. To this end, we consider the establishment of a VLAN segment per VNO. Edge network devices (on the access network) are configured by the MANO plane to tag the incoming traffic with a VLAN ID allocated to the VNO. All involved forwarding elements in the infrastructure e.g., backhaul links, are also correspondingly configured to associate VNO traffic (and allocated resources e.g., bandwidth) with the specific VLAN ID. The MANO plane subsequently orchestrates the instantiation of the vCaches, allocating the requested IT resources e.g., CPU, RAM, storage, etc. In this stage, a separate tenant is

created within the VI i.e., OpenStack in our case. This results in the secure isolation of allocated resources across tenants of the VI, including network isolation i.e., a separate virtual network is created for each tenant within the VI in the form of a VLAN. At this point, the Orchestrator component is responsible for coordinating the overall network configuration, so that the entire forwarding fabric (within and across μDC borders), appears as a single VLAN domain.

B. Traffic interception

The next stage involves the configuration of the network for the interception of traffic by the vCaches. As the cache VMs are configured to participate in the tenant's VLAN domain, all traffic can reach them using well known techniques (see Section II). However, the overheads associated with a cache miss in the context of NFV caching, as illustrated in Section I, call 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 intercepted by caches, and appropriately steering traffic through flow rules, subject to content availability and load conditions [1], [6], [7]. However, realizing these solutions in an NFV-enabled, multi-tenancy environment requires careful consideration of the shared nature of the infrastructure. Intelligent traffic interception mechanisms, currently build on content availability lookups upon each content request. As such, they are prone to significant resource consumption of switching fabric, such as `br-int` at the network node, raising 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, a close look at Figure 1 illustrates the drawback of this approach: traffic steering decisions are taken once traffic flows have already traversed the virtualized infrastructure towards the VNF; flows that will eventually bypass the vCache, pay the corresponding traversal delay penalties.

Considering this tradeoff, our on-going work focuses on the establishment of traffic interception flow rules on the shared `br-int`, 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/cache index, 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.

C. Peering configuration

Based on the aforementioned configurations, 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 VNO's vCaches cannot communicate with each other, even if they are instantiated within the same Compute Host. The establishment of a cache peering relationship, then calls for the careful configuration of the network environment, adhering to the following set 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 unauthorised 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 (Req.4).

Towards these ends, the proposed solution includes a mixture of network 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 vCache instances i.e., the VNF VMs of the involved 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 is able to grant access to the peering VNO by its tenant ID. The latter VNO is able then to proceed with attaching its vCache(s) to the shared network. When the vCaches are collocated on the same Compute Host, the resulting configuration allows the peering traffic to reach the corresponding VMs by only traversing the `br-int` switch (see Figure 1(b)). When vCaches are located at different Compute Hosts of the same μDC, traffic reaches the host of the peering vCache through a direct link typically available between `br-vlan` switches of co-located Compute Hosts (not shown due to length limitations).

The described configuration so far satisfies Req. 1 and 4. To further satisfy Req. 2 and 3, application level configurations come into play. Namely, vCaches build on the Squid cache

---

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

implementation[7], a mature and widely adopted solution. The configuration of Squid first includes the establishment of the peering link, through the `cache_peer` directive[8], which allows the specification of the peering vCache IP/hostname and listening ports. Satisfying Requirement 2, goes through the `iptables` configuration of a vCache, dropping all not legitimate input traffic e.g., traffic destined to a non-peering port or originated by an IP address other than the vCache IP address[9]. Satisfying Requirement 3, goes through the appropriate `login` configuration options of the `cache_peer` Squid directive. The configuration of the `icp_access` access control list, enables access control of cache peering control traffic.

It is noted that the proposed configuration does not rely on the direct involvement of the infrastructure operator i.e., both the establishment of the shared network and the application layer configuration are carried out by the tenants themselves. This is considered as an important aspect of the solution, simplifying and facilitating the management of peering links by the VNOs, without directly exposing business relationships to third parties.

As the establishment of a vCache peering link realizes a business agreement between the VNOs, monitoring and controlling the amount of *exchanged* resources becomes particularly important, as it allows VNOs to ensure the symmetry of the peering link in terms of resource consumption. This means that VNOs require firm control over the compute, storage and network resources devoted to the peering agreement (Req.5). In the context of caching, this translates to the compute load for the lookup of content for peering requests, the I/O storage load for the retrieval of cached content and the load for the transmission of the content to the peering vCache. Our preliminary approach on this issue, builds on the Delay Pools feature of the Squid cache implementation, which provides the means to limit the bandwidth of certain requests based on any list of criteria[10]. In particular, we define a class 2 delay pool for the shared network between the vCaches.

IV. RELATED WORK

Caching has been identified as one of the key applications areas for the NFV paradigm, right from the beginning of the NFV concept [8]. Since then, several commercial NFV-enabled solutions have appeared, including caching as a key building block of a broader-CDN oriented solution e.g., [9], [10]. However, limited information has been revealed about the service configuration, as these solutions are proprietary, while, to the best of our knowledge, no commercial solution focusing on multi-tenancy. 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 [11]. However, again, to the best of our knowledge, no research efforts are being devoted to the particular case 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) [12]; 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 an important step in progressing beyond the mere NFV-based instantiation of virtualized caches, focusing on a better understanding of the particular technical (and business) opportunities brought in the field by virtualization and network programmability, namely, for multi-tenancy.

V. CONCLUSIONS AND FUTURE WORK

Building on NFV and SDN technical advances, and the emerging capability to support multi-tenancy in 5G networks, in this paper we proposed the realization of cache peering relationships between VNOs. The purpose is to take advantage of the potential co-location of virtual caches within μDCs, thus promising reduced latency and traffic overheads. We presented our on-going work, identifying key challenges and potential solutions in the specific context of NFV. The proposed technical path towards the realization of the envisioned setup enables the autonomous management of peering links from VNOs.

REFERENCES

[1] M. Rodrigues, G. Dn, and M. Gallo, "Enabling transparent caching in lte mobile backhaul networks with sdn," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2016, pp. 724–729.

[2] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 75–86. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851194

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

[4] J. Wang, "A survey of web caching schemes for the internet," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 36–46, 1999.

[5] 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

[6] M. Kimmerlin, J. Costa-Requena, and J. Manner, "Caching using software-defined networking in lte networks," in *2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, Dec 2014, pp. 1–6.

[7] P. Georgopoulos, M. Broadbent, B. Plattner, and N. Race, "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] "Network Functions Virtualisation Introductory White Paper," 2012. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[9] 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/],

[10] Juniper/Akamai, "The Elastic CDN Solution," 2014. [Online]. Available: https://www.juniper.net/assets/de/de/local/pdf/solutionbriefs/3510532-en.pdf

[11] 5G PPP, "5G PPP Phase 1 Projects," 2016. [Online]. Available: https://5g-ppp.eu/5g-ppp-phase-1-projects/

---

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

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

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

[10]http://wiki.squid-cache.org/Features/DelayPools

[12] G. Bertrand, E. Stephan, T. Burbridge, P. Eardley, K. Ma, and G. Watson, "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