On-path vs Off-path Traffic Steering, That Is The Question

Karima Khandaker, Dirk Trossen, Jinze Yang, Zoran Despotovic
Huawei Research
Munich, Germany
{karima.khandaker,dirk.trossen,yangjinze,zoran.despotovic}@huawei.com

Georg Carle
Technical University of Munich
Munich, Germany
carle@net.in.tum.de

ABSTRACT

Service-level traffic steering in the Internet has been using an indirection-based model for decades now, using the DNS to resolve a name to a locator, often complemented with load balancing techniques. Contrasting this off-path realization, service information as part of the data packet itself may determine the one of possibly many communication endpoints on-path while traversing the network. This paper compares both design choices regardless of the specific decision mechanism used. For this, we assume a compute-aware traffic steering mechanism for both approaches and determine latency penalties through off-path resolution steps as well as distributing scheduling decisions to on-path network ingress points. Lastly, we investigate latency variances and resilience in an AR/VR scenario.

1 INTRODUCTION

Service provisioning in recent years has been largely driven by two trends. Firstly, *virtualization* has enabled flexible service provisioning in more than one network location. Technologies have progressed from virtual machines to containers, enabling sub-second service availability. Secondly, the *cloud-native* paradigm postulates agile development and integration of code, decomposing services into smaller micro-services, to be deployed and scaled independently, yet chained towards a larger service objective. Such deployment flexibility allows to bring services 'closer' to consumers, terminating up to 72% of traffic in customer access networks [17].

These trends continue to be realized with a decades-old model that uses DNS for mapping service domains onto one of a set of IP addresses, often based on load or geo-information. Those IP addresses as well as port assignments identify network interfaces and sockets for service access. Such assignments typically remain static, contrasting against the evolution of *availability* of resources and *location independence* that the aforementioned trends have been driving.

The Internet community has developed solutions to cope with the limitations of the DNS+IP model, such as Global Server Load Balancing (GSLB) [12], DNS over HTTPS (DoH) [21] as well as HTTP-level [19] or, more recently, transport-level [32] indirection. At the routing level, efforts like [28, 36, 40] but also those related to information-centric networking [39], instead propose to include suitable service information into traversing service requests for on-path traffic steering, without the need for an explicit resolution step.

This paper does not advocate any specific solution but instead investigates the impact of either design choice on end-to-end service performance. To cater to emerging requirements of highly distributed computing resources, e.g., in 6G, we utilize compute-aware traffic steering decisions [28, 31] instead of mere random ones, removing any impact from needed signalling of decision metrics from our considerations. Instead, we focus on the service transaction performance in term of *penalties* incurred through indirection or distributed decision making. We further investigate the *dynamicity* that can be supported in the form of service transaction lengths, utilizing distributed service endpoints with frequently changing traffic steering decisions, possibly down to single request choices. We showcase the impact of this dynamicity on parameters like end-to-end service latency as well as maximum number of users being supported in a given service deployment setup, showing that on-path systems can significantly outperform off-path ones. With this, our main contributions are as follows:

- We formulate the problem of traffic steering in a, e.g., virtualized, service environments (Section 2).
- We describe high-level details for off-path and on-path systems-at-tests (Section 4) that adequately represent existing and proposed solutions.
- We then outline a traffic steering mechanism (Section 5), representative for those used in existing systems.
- Finally, we present (Section 6) our performance comparison, outlining a reduced latency variance and higher resilience for on-path systems in an AR/VR scenario.

2 PROBLEM FORMULATION

We now formalize the problem that the systems at test in this paper intend to address. Let us define a service S, which is realized by possible *service instances* $S_1, ..., S_n$ in dedicated network locations $L_1, ..., L_n$. Furthermore, we assume one or more *clients*, defined as $C_1, ..., C_m$, wanting to access service S as part of their interaction across the wider network.

The problem at hand can be described as finding a 'suitable' $S_k(C_i)$ out of the n possible ones for every client C_i wanting to send a request to S, where 'suitable' defines the fulfilment of a service-specific optimality criteria across *attributes* representing service-specific performance metrics.

Examples for suitability criteria may be shortest path, lowest latency, highest throughput or more complex multi-optimality criteria (see [25] to achieve multi-optimality routing that aligns with our model presented here).

3 RELATED WORK

Beyond our algorithm in Section 5, there exist many methods for DC-internal traffic steering, such as [27, 30, 35, 37]. As a solution for Internet traffic steering, the solution in [31] was designed for service function chaining scenarios but can be applied for ingress-based scheduling, very much like the work in [40], although the former merely applies site-local load balancing, while the latter requires regular metric signalling to ingress points, incurring significant routing costs. As we discuss in Section 5, we select the traffic scheduling solution for our investigation with a focus on minimizing control signalling, while still being compute-aware. Regardless, the mentioned works here may be applied as possible decision logics to be implemented either on- or off-path.

New architecture proposals, such as those on content- or information-centric networking, decouple content information from locators. Examples presented in Named Data Networking [39], NetInf [6, 9] or PURSUIT [7, 8], allow for on-path decision making in some cases. Building on ICN concepts, work in [15, 16, 18, 20, 29, 33, 38] has shifted focus on service information being used for routing albeit embedded into NDN interest-data packet delivery semantic. However, the decision for steering traffic among a set of possible destinations has not been the focus of those works.

We can observe from our study of related work that comparing a given algorithm as to its efficacy in an on- vs off-path system has not been studied. *We see insights into this comparison as crucial for postulating the idea of on-path traffic steering as a new direction for routing in the future Internet.*

4 SYSTEMS AT TEST

We distinguish two categories of systems that address the problem in Section 2, termed *off-path* and *on-path* in short and illustrated in Figure 1.

Two key aspects need to be realized by those systems. First, data transfer at the IP level to the serving service instance (out of the set of possibly many choices) needs to be

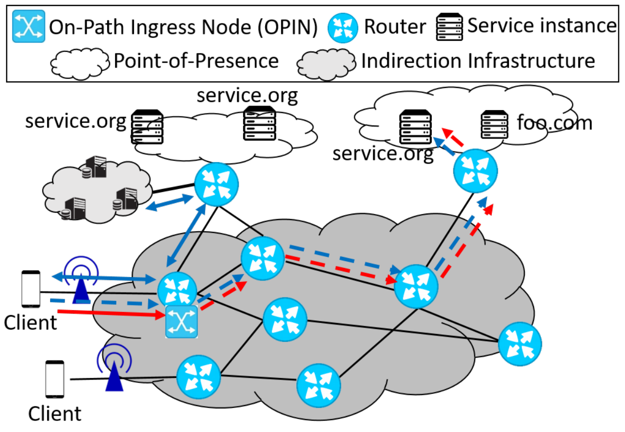


Figure 1: Off-path (blue arrows) vs ingress-based on-path resolution (red arrows)

preceded with an explicit step of *identifier resolution*, where such identifier may be encoded as, e.g., a uniform resource locator (URL) or a binary encoded structured name [34].

Second, the continuity of a *service transaction* must be ensured, where a transaction consists of one or more service requests. The continuity here relates to any subsequent service request needing to be directed to the same service instance that has been initially resolved in the aforementioned resolution step; this is due to ephemeral state that may be created during the transaction and would otherwise require explicit handling, e.g., through a shared database or similar, if the service instance was to change during the transaction.

Let us outline in the following how both our systems-at-test realize those key aspects.

Off-path resolution is characterised through resolving service names through an explicit indirection with operations off the path of the actual data transfer, as illustrated in Figure 1 with bold blue arrows. For this, an initial indirection is sent from the client to a separate *indirection architecture*, returning the IP locator for the chosen service instance, followed by the data transfer via the intermediary IP routers throughout the transaction. Utilizing different service instances across different transactions is realized by repeating the (off-path) resolution step for every new transaction. Here, client-side caches must be suitably flushed to avoid using stale resolution state which may point to the wrong 'best' instance at the time of the renewed resolution.

There are a number of examples for indirection architectures. The Domain Name Service (DNS) provides the most used indirection for resolving domain names into IP locators. Extensions to DNS allow for transporting the DNS request over HTTP [21], as often used by cloud providers, such as Google or CloudFlare, to unify service provisioning with the resolution service leading to the services. In this case, the indirection architecture would reside within the *point-of-presence* (PoP) shown in Figure 1 albeit separately managed from the service infrastructure which may eventually be used as a result of the indirection.

PoP providers also often provide sophisticated load balancing solutions, such as Global Server Load Balancing (GSLB [12]), to select one out of possibly many network locators based on server and network load information. In that case, the indirection step shown in Figure 1 comprises of many more steps inside the indirection architecture.

Indirection architectures may also overlap with the service infrastructure, such as those provided in PoP, when using methods like QUIC load balancing [32], where the initial indirection occurs to a service instance, which then redirects to another service instance, either in another or the same PoP site. The same occurs when applying HTTP-level indirection [19] instead albeit at higher, here HTTP, layer.

Common to all approaches is the complexity through additional signalling as well as mapping operations, where the latter often involve application-level database lookups at various entities in the indirection architecture; a complexity that is significantly different from that for low level packet forwarding operations, with Section 6.1 providing a model that captures the latency for this complexity.

Other approaches, such as those defined in the IETF Alto [22] efforts, provide alternatives to DNS-based indirection architectures but we limit our evaluation in this paper to those based on DNS due to their widespread use.

Ingress-based on-path resolution differs crucially, as the name suggests, in performing the resolution from a service identifier to a network locator *on-path* from the client to the chosen service instance. That means that a mapping, akin to the name-locator mapping in off-path systems, occurs albeit not requiring the explicit indirection architecture. This is shown in Figure 1 through the bold red arrow from the client to the *on-path ingress node* (OPIN), which is mapped onto a dashed red line through some sort of mapping process. Subsequent requests within the same transaction may then use direct IPv6 routing, i.e., following the blue dashed line route. An IP router and OPIN may be collocated but OPINs may also be dedicated (shim overlay) nodes. Given that mapping occurs at the OPIN, different service instances can be used for subsequent transactions, if suitable knowledge for such renewed mapping exists at the OPIN.

Recently, several proposals have been made for on-path architectures. The approach in [40] encodes a service identifier as an IP anycast address, while the approaches in [28, 36] use a separately encoded service address, e.g., provided in an IPv6 extension header. All approaches rely on mapping the initial service identifier to one of several possible IP addresses, where the mapping is being announced to the OPINs in a separate announcement step, akin to a route announcement in IPv6. This mapping operation is similar to a next hop table operation in existing IP routers (where the possibility to hash service names may improve on the table lookup performance, not requiring longest prefix match operations) and can therefore likely support high mapping rates.

5 ALGORITHM USED

As outlined in the introduction, our choice of traffic steering algorithm needs to fulfil two requirements. Firstly, it has to be *compute aware* to cater to emerging requirements in edge computing scenarios [26, 40], where the 'best' instance may often not be the best one to steer traffic to. A range of solutions, such as those in [12, 28, 30, 31, 35, 37], provide this capability, e.g., using compute load and network latency for the decision making. Secondly, we want to focus our evaluation on the traffic steering decision only, thereby leaving any insights on potential costs for distributing suitable metric information to the decision points for future work.

For those reasons, we selected CArDS (compute-aware distributed scheduling) [28], realized either in a central, off-path solution or across on-path ingress points (the OPIN in Figure 1). Let us provide some details on this algorithm, referring to [28] for more details, including a comprehensive evaluation of various design aspects of the solution.

CArDS abstracts the compute capabilities, assigned to each service instance for processing incoming service requests, as *compute units* [28]. Those may represent, e.g., the number of CPUs or threads assigned to the service instance at the local compute resource; this fulfils our second requirement above.

The compute units and IP network locators of all service instances are distributed, e.g., using a central management interface or a routing protocol, to all OPINs (for on-path architectures) or to the off-path infrastructure. It is assumed here that compute unit assignments are done during the deployment of the overall service and thus only change infrequently, addressing our first requirement above.

With the compute units available per network locator at which the compute resources are reachable, the traffic steering decision is that of a *weighted round robin*, where the compute units are used as weights. With this, traffic is sent to service instances in proportion to their computational capabilities, captured in those compute units.

As argued in [28], for on-path systems, the scheduling decision can be implemented in data plane technologies such as P4 [11], with [36] outlining a realization over available programmable switches. With this, we can assume on-path decision latencies close to line speed (for ingress points, which usually run at lower forwarding speeds), resulting in little to no penalty for such decision in on-path architectures.

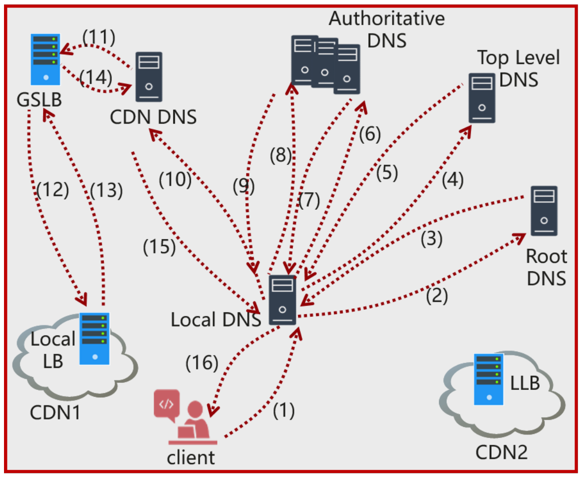


Figure 2: Signaling complexity in GSLB

6 EVALUATION

The following evaluation first outlines insights to the initial request latency that is incurred for the first request of a (possibly longer) transaction, followed by evaluating the impact of distributing traffic steering decision compared to an idealised central solution, before discussing how well smaller affinities may be supported by on- and off-path systems. We close with an AR/VR use case evaluation.

6.1 Initial request latency

To shed light on the latency for the initial service request that needs steering to one out of possibly many service instances, we first investigate the typical complexity of a modern indirection architecture. Figure 2 shows a typical DNS-based load balancing architecture for CDNs, as can be realized through GSLB [12] or DNS over HTTP [21].

We can see that there may be different *levels* of indirections, which all depend on the given service identifier. The authors in [24] capture these indirections as probabilities for referrals, leading overall probabilities for average response times for DNS resolution requests. This work, however, did not account for CDN deployments at the scale of today's Internet, where many services are 'frontloaded' so that the probability for additional referrals become rather low.

To align Figure 2 to simpler indirections, as found in HTTP [19] or QUIC indirections [32], we focus in our investigation on cases where CDNs are able to resolve and, thus, load balance services as part of their infrastructure; this involves steps (1) and (10)-(16) in Figure 2 only.

We can observe that signalling messages are realized at the application layer, e.g., as DNS over HTTP, while operations include application-level, e.g., database, operations. Studies [14] show that first hop resolver resolution in a CDN may provide latencies of between 10 to 100ms, compared to lower 10ms for operator-provided DNS [4]. Although the latter yields lower latencies, it is the coupling with a load balancing decision that is of interest to us.

Hence, we will assume an initial request latency for off-path systems of around 45ms, used in our following evaluations.

Note that this represents the best case for off-path systems, where significantly longer latencies, possibly in the 100s of milliseconds [13], may occur if services are not CDN-hosted.

Furthermore, this latency may accumulate when considering typical Internet scenarios since a higher-level *user experience*, e.g., loading a web page in a browser, often consists of many parallel and consecutive service transactions, each incurring DNS interactions. In 2019, a web page loaded on a desktop included on average 70 resources (75 for a page on mobile access) [3], each requiring DNS resolution albeit some resolved through the client-local cache. Furthermore, the average *time to first byte* (TTFB) was 1.28s for desktop and 2.59s for mobile access [3], while it took, on average, about 4.7s to load a page on a desktop versus 11.4s on a mobile [3]. However, webpages had an average size of 1.9 MB (for desktop access) or 1.78 MB (for mobile access) [3]. Although this discrepancy between web page size and time to load is likely not just to blame on DNS operations, they do play a key role, as also shown in [13], with further studies needed to quantify the exact impact of frequent DNS operations.

For on-path systems, we can observe that there is no additional signalling, due to the on-path nature, while the complexity may often be significantly lower, when relying on solutions that are based on existing IP forwarding solutions, e.g., [28, 36, 40], therefore leading us to assume near lines peed latencies for on-path systems.

6.2 Impact of On-Path Distribution

The CArDS algorithm was evaluated in [28], comparing an idealized centralized scheduling with an increasingly distributed one for a traffic model of uniformly distributed arrivals between 2.5 and 7.5ms of single service requests (i.e., no affinity) with overall 20 service instances distributed equally across 5 sites, with an added network latency of 7.4ms, while the number of clients ranges from 315 to 1710 clients, representing 20% and 110% load, respectively.

The comparison showed that a distributed realisation shows no significant difference in mean request completion time. Only when approaching full system capacity, the distributed system performs up to 11% worse than the centralized one [28]. These performance insights also sustain, if the number of ingress points scales significantly to several hundreds (up to 3105 were simulated in [28]) while maintaining a reasonable load.

Let us put these results into the context of, e.g., 5G mobile networks. Here, the *User Plane Function* (UPF) [2] is responsible for handling user data traffic, enabling decentralization of packet processing and traffic aggregation closer to the network edge. The UPF is a perfect place to integrate on-path scheduling in a 5G network, i.e., realizing the role of the OPIN in Figure 1 within 5G. The report in [1] gives an overview of possible and typical deployments. In the most decentralized deployment as an *Access Edge* ([1] Figure 4.5 on page 43), UPFs are deployed in access edge clouds, close to the radio access nodes (base stations), acting as ingress nodes for traffic from at least several base stations. For a country wide 5G network deployment, one can count with a range of UPF instances from few hundreds to several thousands.

Therefore, we can conclude that on-path service scheduling aligns well with scalability needs of 5G system rollouts.

Note that the centralized solution in [28] is an idealized on-path solution. For any off-path solution, the resolution latency of about 45ms (see Section 6.1) must be added to the initial request for both centralized and distributed solutions that is observed in [28]. Hence, a centralized off-path solution performs significantly, i.e., about 7 times, worse than a distributed one, while only load situations close to 100% will close that gap to a distributed on-path solution.

With a system load of 110% representing 1710 clients in this evaluation, the observed load on the indirection infrastructure (exhibiting more than 200000 requests per second with the chosen interarrival model in [28]) would pose the real challenge, as we will discuss in our next sub-section.

6.3 Supporting Small Affinities

To shed light on supporting small affinities, let us take an AR/VR example. Here, chunk retrieval techniques are used by N clients to obtain replicated content from a set of distributed service instances. Typical chunk lengths, according to [41], are 500ms albeit relying on 'early termination' capabilities of HTTP/2 or QUIC [23], where the client's physical movements may lead to terminating an ongoing chunk retrieval, followed by the relevant next chunk. This may lead, in highly dynamic VR/AR scenarios, to retrieval intervals of few tens milliseconds to prevent motion sickness. For instance, if we assumed an average retrieval interval of 100ms for every client, $N*10$ retrieval requests per second would be sent, all of which would need a traffic steering decision.

DNS server configurations usually limit clients to few tens of requests per minute [5], while providers such as Google apply ISP-level rate limits, by default 1500 queries/min per public IPv4 address [10]. With this in mind, despite the stateless nature of chunk retrieval, transaction lengths for off-path systems would require to be rather long, in the order of minutes or even for the entire session. Particularly problematic here is the dependence on the overall number of users, not just for the scenario itself, but for all users accessing an ISP's network (in cases where DNS rate limitations are applied across all ISP users), making it difficult for the application itself to gauge the right transaction length to be used.

On-path systems, on the other hand, are merely limited by the capability to map from service identifier onto network locator in the OPIN. As previously mentioned, work in [28, 36, 40] has shown this to be feasible at near line speed.

6.4 Impacting Use Case Performance

We now use our AR/VR use case to show how on-path traffic scheduling impacts use case performance, measured in *latency variance* and *resilience* to service availability; both critical aspects in providing superior user experience.

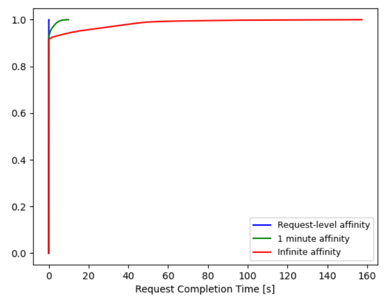
For this, we use the Python-based simulation environment described in [28] with the setup described in Section 6.2, i.e., clients are uniformly distributed across 5 ingress nodes, retrieving their content from service instances across 5 sites with equal compute capacity. Clients join the system within uniformly distributed short periods of time at the beginning of the total simulation time of 10 minutes.

Based on our insights in Section 6.3, the client request interarrival times t_i are set to 100ms (to account for possible early terminations due to client interaction) with a random variation of 5ms. We assume each content server is able to sustain 100MBit/s throughput, serving about 100KByte per AR/VR chunk. With this, we set the a processing rate of 125 requests per second for each service instance in our setup.

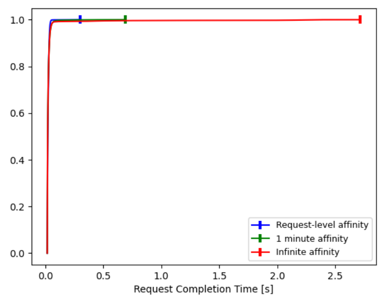
For an acceptable user experience, we assume 10ms for client-local packet handling (buffering, decoding and displaying), adding a further 15ms margin for possible packet latency variations; leading to an *upper latency bound* of 75ms after which we would expect service degradation.

We first evaluate the impact of choosing different affinity lengths on the *request completion time* (RCT). For this, we apply affinity at request-level together with two longer affinity modes. In the first, a client will be assigned to a service instance for the entire duration of the session (*infinite affinity*), while we also realize a *1 minute affinity*, where each client performs the resolution steps for a new transaction every minute, thus adding a 45ms indirection latency to the first request of the affinity, as per our insights from Section 6.1.

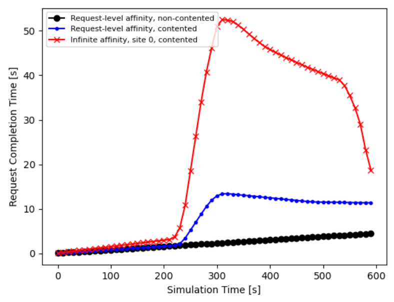
Figure 3 (a) shows the CDF for the RCT in our setting for 375 clients (60% load). While the mean RCT values are very close for all affinity modes, the CDF tails significantly differ, i.e., experiencing significant differences in *latency variance*. This is also reflected in a higher number of requests exceeding the upper latency bound of 75ms with 0.014%, 8.2% and 8.7% of packets for single request, 1 minute and infinite affinity, respectively.



(a) CDF for $t_i = 100ms$



(b) CDF for $t_i \in [100ms; 500ms]$



(c) Average RCT in server overload case

Figure 3: Results for AR/VR use case

This observation prevails when altering the clients' interarrival time to being uniformly distributed between 100ms and 500ms for each request with the number of clients scaled accordingly from 185 to 1875, shown in Figure 3 (b) for 1125 clients (60% load). Although, as expected, the mean RCT increases due to the larger variance in interarrival times, the differences in the respective CDF tails prevail. We explain this behaviour with the increased *diversity of choice* for N clients among *k=20* service instances, compared to assigning $\frac{N}{k}$ clients to a single instance, thereby reducing contention by smoothening instances' input queues.

Improving RCT variance is not the only benefit of selecting service instances at request-level; they may also improve on *resilience* against localized contention, e.g., caused by internal HW failure or overload. To simulate this, we reduce the processing rate for service instances on site 0 to half of that of all other sites, starting at 4 minutes in the simulation and lasting for 2 minutes. The client load is maintained at 70%, i.e., 435 clients, to ensure high system contention.

Figure 3 (c) shows the average RCT along 10 minutes of simulated time. The black line serves as the baseline average RCT, i.e., not having any contention at the sites but observing the already existing contention through the high system load. The red line shows the average RCT for those clients associated with the affected site 0 in infinite affinity mode. Since clients are not assigned to any instance for longer than one request in single request affinity, we show as the blue line the average RCT for all clients in single request affinity.

As expected, we can see from the red line that the clients at the affected site are severely impacted by the contention with an up to 21 times increase in RCT, while request-level affinity lowers this impact significantly to 4.6 times.

However, the above results cannot be equally applied across on- and off-path systems, since the latter incurs the aforementioned 45ms latency for every request being sent (as well as creates significant load on the indirection architecture). Reducing the affinity length, possibly down to single requests, significantly increases the load on the indirection architecture, as already discussed in Section 6.2. For our use case, this load increases to up to 6250 requests/s when moving towards full system capacity. As discussed before, usual rate limits in DNS system would not support such rates, making thus the use of small affinities as a means to increase system utilization impossible.

We can conclude from our use case evaluation that applying request-level affinities significantly improves RCT variance and resilience to local contention, while applying them in on-path systems avoids the significant additional costs for indirection.

7 CONCLUSION

We compared traffic steering performance in on- vs off-path systems, where the steering is compute-aware and requires little control signalling. Our analysis demonstrates that off-path systems suffer from significant signalling and complexity overhead associated with explicit resolution steps, while on-path systems allow for significantly smaller transaction lengths, served in distributed service instances. In emerging use cases, such as AR/VR, on-path systems significantly reduce latency variance as well as increase resilience to local contention in the server infrastructure, thereby positively improving on overall end-user experience, while avoiding the significant costs that off-path systems would require for scaling to the number of requests entering the system.

We plan on enhancing our insights by studying the impact of distributing control signalling for on-path systems compared to the centralized realization of off-path systems.

REFERENCES

[1] 2021. *5G-Vinni Project Deliverable D1.4. (2021). Design of infrastructure architecture and subsystems v2.* Technical Report. 5G-VINNI. https://doi.org/10.5281/zenodo.4066381

[2] 2021. *System architecture for the 5G System (5GS); Stage 2 (Release 16), TS 23.501 V16.11.0 (2021-12).* Technical Report. 3GPP. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[3] 2022. *Average Page Load Times for 2020 – Are you faster?* Technical Report. Machmetrics. https://machmetrics.com/speed-blog/average-page-load-times-for-2020/

[4] 2022. *DNS-over-HTTPS performance.* Technical Report. SamKnows. https://www.samknows.com/blog/dns-over-https-performance

[5] 2022. *DNS Rate Limits.* Technical Report. deDec DNS API. https://desec.readthedocs.io/en/latest/rate-limits.html

[6] 2022. *EU Project 4WARD website.* Technical Report. European Commision. https://cordis.europa.eu/project/id/216041

[7] 2022. *EU Project PSIRP website.* Technical Report. PSIRP project. https://www.psirp.org/

[8] 2022. *EU Project PURSUIT website.* Technical Report. European Commision. https://cordis.europa.eu/project/id/257217

[9] 2022. *EU Project SAIL website.* Technical Report. SAIL project. https://sail-project.org

[10] 2022. *Google Public DNS for ISPs.* Technical Report. Google. https://developers.google.com/speed/public-dns/docs/isp

[11] 2022. *P4 Open Source Programming Language.* Technical Report. Open Network Foundation. https://p4.org/

[12] 2022. *What is GSLB?* Technical Report. Efficient IP. https://www.efficientip.com/what-is-gslb/

[13] V. Bajpai A. Lutu O. Alay J. Ott A. S. Asrese, E. A. Walelgne. 2019. Measuring Web Quality of Experience in Cellular Networks. In *Proc. of IEEE Conference on Passive and Active Measurements (PAM).*

[14] V. Bajpai P. Sarolahti J. Ott A. S. Asrese, S. J. Eravuchira. 2019. Measuring Web Latency and Rendering Performance: Method, Tools & Longitudinal Dataset. *IEEE Transactions on Network and Service Management* 16, 2 (2019), 535–549.

[15] M. Sifalakis C. Tschudin. 2014. Named functions and cached computations. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th.* 851–857.

[16] J. Chen, S. Li, H. Yu, Y. Zhang, D. Raychaudhuri, R. Ravindran, H. Gao, L. Dong, G. Wang, and H. Liu. 2016. Exploiting ICN for realizing service-oriented communication in IoT. *IEEE Communications Magazine* 54, 12 (2016), 24–30.

[17] Cisco. 2020. *Cisco Annual Internet Report (2018-2023) White Paper.* Technical Report.

[18] M. J. Reed; J. Riihijärvi; M. Georgiades; N. Fotiou; G. Xylomenos D. Trossen. 2015. IP over ICN - The better IP?. In *European Conference on Networks and Communications (EuCNC).*

[19] R. Fielding and J. Reschke. 2014. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.* RFC 7231. IETF. http://tools.ietf.org/rfc/rfc7231.txt

[20] G. Rutz G. White. 2016. *Content Delivery with Content-Centric Networking.* Technical Report. CableLabs. http://www.cablelabs.com/wp-content/uploads/2016/02/Content-Delivery-with-Content-Centric-Networking-Feb-2016.pdf

[21] P. Hoffman and P. McManus. 2018. *DNS Queries over HTTPS (DoH).* RFC 8484. IETF. http://tools.ietf.org/rfc/rfc8484.txt

[22] IETF. 2022. *Application-Layer Traffic Optimization Working Grpu: Charter.* Technical Report. IETF Alto WG. https://datatracker.ietf.org/group/alto/about/

[23] J. Iyengar and M. Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport.* RFC 9000. IETF. http://tools.ietf.org/rfc/rfc9000.txt

[24] H. Balakrishnan R. Morris J. Jung, E. Sit. 2002. DNS performance and the effectiveness of caching. *IEEE/ACM Transactions on networking* 10, 5 (2002), 589–603.

[25] M. A. Ferreira J. L. Sobrinho. 2020. Routing on Multi Optimality Criteria. In *ACM SIGCOMM.*

[26] S. Robitzsch M. Boniface S. Philips K. Haensge, D. Trossen. 2019. Cloud-Native 5G Service Delivery Platform. In *2nd Intl Workshop on Mobility Support in Slice-based Network Control for Heterogeneous Environments.*

[27] J. T. Humphries A. Belay D. Mazieres C. Kozyrakis K. Kaffes, T. Chong. 2019. Shinjuku: Preemptive scheduling for μsecond-scale tail latency. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI).*

[28] R. Khalili Z. Despotovic A. Hecker J. Carle K. Khandaker, D. Trossen. 2022. CArDS: Dealing a New Hand in Reducing Service Request Completion Times. In *IFIP Networking (to appear).*

[29] Michał Król, Spyridon Mastorakis, David Oran, and Dirk Kutscher. 2019. Compute first networking: Distributed computing meets icn. In *Proceedings of the 6th ACM Conference on Information-Centric Networking.* 67–77.

[30] S. Dharmapurikar R. Vaidyanathan K. Chu A. Fingerhut V. T. Lam F. Matus R. Pan N. Yadav G. Varghese M. Alizadeh, T. Edsall. 2014. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters. In *Conference of the ACM Special Interest Group on Data Communication (SIGCOMM).*

[31] Khalili L. Wang P. Eugster M. Bloecher, R. 2020. Letting off STEAM: Distributed Runtime Traffic Scheduling for Service Function Chaining. In *IEEE Infocom.*

[32] N. Banks M. Duke. 2021. *QUIC-LB: Generating Routable QUIC Connection IDs.* Technical Report. IETF. https://datatracker.ietf.org/doc/draft-ietf-quic-load-balancers/

[33] D. Oran D. Kutscher Y. Psaras M. Krol, K. Habak. 2018. RICE: Remote Method Invocation in ICN. In *ACM Conference on Information-Centric Networking.*

[34] M. Mosko, I. Solis, and C. Wood. 2019. *Content-Centric Networking (CCNx) Messages in TLV Format.* RFC 8609. IETF. http://tools.ietf.org/rfc/rfc8609.txt

[35] Y. C. Hu G. Lu J. Padhye L. Yuan M. Zhang R. Ghandi, H. H. Liu. 2014. Duet: Cloud scale load balancing with hardware and software. In *Conference of the ACM Special Interest Group on Data Communication (SIGCOMM).*

[36] I. Kunze Z. Lou J. Rüth M. Stoffers K. Wehrle R. Glebke, D. Trossen. 2021. Service-based Forwarding via Programmable Dataplanes. In *First Workshop on Semantic Addressing and Routing for Future Networks at IEEE Conference on Intl. Conference on High Performance Switching and Routing.*

[37] P. B. Godfrey Y. Ganjali A. Firoozshahian S. Ghorbani, Z. Yang. 2017. DRILL: Micro Load Balancing for Low-Latency Data Center Networks. In *Conference of the ACM Special Interest Group on Data Communication (SIGCOMM).*

[38] M. Hofmann I. Rimac M. Steiner M. Varvello T. Braun, V. Hilt. 2011. Service-centric networking. In *2011 IEEE Intl. Conference on Communications.* 1–6.

[39] J. D. Thornton M. F. Plass N. H. Briggs R. L. Braynard V. Jacobson, D. K. Smetters. 2009. Networking Named Content. In *ACM Conext.*

[40] S. Gu G. Zhuang F. Li Y. Li, Z. Han. 2021. Dyncast: Use Dynamic Anycast to Facilitate Service Semantics Embedded in IP address. In *Workshop on Semantic Addressing and Routing for Future Networks at IEEE Intl. Conference on High Performance Switching and Routing.*

[41] Shou-Cheng Yen, Ching-Ling Fan, and Cheng-Hsin Hsu. 2019. Streaming 360° Videos to Head-Mounted Virtual Reality Using DASH over QUIC Transport Protocol. In *Proceedings of the 24th ACM Workshop on Packet Video (PV '19).* Association for Computing Machinery, New York, NY, USA, 7–12. https://doi.org/10.1145/3304114.3325616