Exploring The Benefits of In-Band Service Routing

1st Emilia Ndilokelwa Weyulu*
Max-Planck-Institut für Informatik
Saarbrücken, Germany
eweyulu@mpi-inf.mpg.de

2nd Dirk Trossen
Huawei Munich Research Center
München, Germany
dirk.trossen@huawei.com

***Abstract*—Current Internet applications use a decade old model to explicitly resolve domain names onto routing identifiers through the Domain Name System (DNS). Service routing postulates an in-band resolution approach, removing the need for explicit discovery. Complemented with routing-based updates of the required service name mappings to routing locators, such an approach promises reduced latency. In this paper, we explore the potential benefits of such an approach. We introduce a taxonomy of communication patterns and key performance indicators (KPIs) for selected use cases, and provide exemplifying insights into the latency promises by combining modelling with prototypical insights for a specific service routing realization named Routing on Service Addresses (ROSA). Our results show, for instance, the potential of lowered latencies, with page retrieval times in web browsing reduced up to 60%.**

***Index Terms*—Service Routing, DNS, Taxonomy, KPIs**

I. Introduction

Traditional IP-based service delivery incurs high latencies from explicitly resolving domain names, introduced by the DNS+IP model, onto IP addresses by means of an explicit DNS query via the (global) DNS system. Furthermore, updates to DNS mapping records have been reported to be extremely slow in practice, often taking hours or even longer to propagate across the DNS infrastructure. This is a challenge for application deployment models that make use of virtualization or micro-service architectures as they require low latencies, and therefore fast resolution of domain names, as such endpoints may be instantiated in seconds or less to provide services.

Significant efforts have been invested in reducing latency on the internet, such as moving hardware and services closer to users by means virtualization in solutions such as edge computing or content deliver networks (CDNs). Moreover, browser and HTTP optimizations (e.g., pipelining, connection multiplexing) are employed to further lower web page load times. On the transport layer, higher initial TCP congestion windows accelerate data transfers [1], while new transport protocols e.g., QUIC [2] improve download times. At the routing layer, proposals such as content-centric networking [3] propose placing content in ubiquitous network locations, then including content information in requests to avoid the latency incurred by explicit DNS resolutions.

*Work done as a student researcher at Huawei

ISBN 978-3-903176-63-8 ©2024 IFIP

Despite these efforts, latency remains a performance bottleneck for many applications. This latency is often accumulated, for instance, in web browsing scenarios where downloading several distinct objects from possibly different servers is typical, leading to inflated overall latencies, as each object may require separate resolution steps. High latencies not only impact simple applications (e.g., web browsing) but also complex cloud-based services, leading to poor user experience and low reliability. This in turn leads to high user churn and economically impacts CDN providers, with estimates of revenue loss from Google and Microsoft in the range of $75M [4].

In this paper, we investigate the benefits of adopting an in-band resolution approach to reduce latency. We build upon the idea of service-based routing (SBR) [5], where the requested *service* is the focal point for the routing mechanism rather than its *location*. We make use of the Routing on Service Addresses (ROSA) [5] proposal, which utilizes an overlay network with IPv6 extension header-based in-band processing of initial resolution requests. Through this, a resolution request of an object or service is piggybacked on the first IP packet to find its way to one-of-many possible destinations that can serve such a request. The responding destination is then chosen as the suitable *service instance* location, with subsequent data requests routed to that destination. This promises substantially faster resolution times, and thus higher throughput rates than current DNS systems can provide. To provide timely updates between services and the mapping (i.e., routing) tables, routing protocols are employed, thus utilizing typical convergence times allowing e.g., load balancing.

Key to evaluating the benefits of such approach is a systematic view on its benefits along well formulated KPIs. We summarize our contributions as follows:

⋆ We develop a *taxonomy* of communication patterns used in typical services that would benefit from in-band resolutions and show how these patterns can be employed with KPIs to evaluate benefits of service routing.

⋆ We quantify the expected benefits, both analytically and experimentally, using example use cases.

⋆ Our results show reduced page retrieval times, up to 60% in web browsing, when in-band resolution is applied to typical CDN-based deployments. We present insights for file retrieval scenarios that show significant reductions to the latency variance at high load, leading to better user experience even under considerable service request load.

II. Primer into In-Band Service Routing

The DNS infrastructure is key to communicating with any networked application service. Here, mapping service names to IP addresses happens through explicit resolutions, incurring latency for both the request and the response before data communication can commence. When services are served via CDNs, there may exist several destinations that can serve the requested application or service, resulting in an *anycast semantic* when requests are send to the service endpoints. To enable load balancing and determine the best CDN-located destination to serve the request, solutions such as Global Server Load Balancing (GSLB) [6] were introduced, albeit limited to a single CDN provider. However, as observed in [7], updates to the mapping tables that occur across the DNS infrastructure, e.g., caused by configurations, detected better choices for a nearby service or one less loaded, take time to propagate, thus limiting the dynamicity with which the choice of service endpoints may change.

In-band service routing proposes to piggyback the service resolution request (from requesting endpoint to routing locator) onto the first IP packet towards the 'best' destination, where 'best' may follow service-specific constraints, e.g., closest destination, minimum response time, etc. The IP address of the responding destination is used by the requesting endpoint, i.e., client, for any subsequent packets, leading to direct communication between the two endpoints. Re-initiating the service resolution request may lead to another destination being chosen, thus supporting the deployment of service endpoints in several network locations, e.g., for redundancy or load balancing. This procedure mimics the workings of a DNS-based system in that a resolution step is done after a specific service has been requested, but with the benefit of reduced completion time replacing the explicit resolution step through a piggybacked in-band resolution.

Routing on Service Addresses (ROSA) [5] is one approach to realize in-band resolutions, utilizing an overlay of Service Address Routers (SARs) that process resolution requests addressed to it in order to suitably forward the IP packet towards its final destination. For the piggybacking, IPv6 extension headers (EH) [8] are used to store the client IP, requested service ID, and the SAR IP [5]. The SAR then returns the IP address of the chosen endpoint to the client via the overlay, after which the client can directly communicate with the chosen endpoint. Note that the SARs only replace the 'local DNS server' functionality, all other communications such as ARP, TCP carry on normally.

III. Taxonomizing Service Routing

Network applications employ different communication patterns and protocols, and thus require different evaluation criteria of how effective such communication patterns and protocols are. The differences stem from the variety of offered services, the generated data workloads, and the communication requirements, e.g., resource constrained IoT devices versus multi-CPU, high-powered computing servers. However, there are no standards of application communication patterns, and system engineers are required to choose the appropriate communication protocol, while leveraging advances in routing and transport protocol designs or even hardware. This is particularly exacerbated when facing new communication protocols such as those introduced in §II, and which KPIs to quantify the benefits of doing so remains open.

Contrasting against an extensive body of prior work [9, 10, 11, 12], we develop a taxonomy of communication patterns and KPIs tied to the subset of use cases that we believe would benefit from an in-band resolution approach. Due to this approach, existing communication protocols require slight tweaking and the development of a taxonomy is thus necessary. The taxonomy outlined here not only helps to deepen the understanding of existing communication requirements and patterns, but also serves as a starting point for future application designs that utilize service routing in contrast to the traditional DNS+IP model.

A. Communication patterns

Let us first taxonomize the communication patterns typically used in the development of applications that need to operate over the internet, e.g., applications located in different networks. Since communication is initiated by an endpoint in a remote network, the communication pattern to be used should take into account certain aspects, e.g., expected type of workload, utilized hardware, etc, to ensure best service performance as the data goes over the network.

Similar to prior work, we define the *Request-Response* service invocation as the key dimension as it allows to implement patterns such as publish/subscribe or streaming, where responses may either be a single message or a *stream response*. The selection of the responding endpoint (referred to as *service instance*) occurs by either *unicast* (one of many), *bestcast* (selecting one of possibly many based on some expressed policy, such as random selection, linear order routing or utilizing an instance at the ingress of the service routing overlay (see §II)), or *multicast* (selecting many of many).

Applying *Request-Response* to a typical web server use case requires differentiation between client- and server-initiated communication, because the response type changes based on the initiating endpoint. For example, the response to server-initiated communication may include dynamically created content, e.g., as seen on social media websites, which differs from a client requesting several, already existing web objects. However, both instantiations make use of unicast and bestcast invocation semantics. Methods for selecting one of possibly many service instances may range from random selection over some (partial) linear order policy, e.g., representing a delay- or distance-centric selection, to an ingress-based selection. For the latter, compute-aware mechanisms, such as proposed in [13], may be used albeit assuming a static assignment for compute units to the deployed service instances, similar to methods in GSLB [6]. Equally, instance selection methods may be applied for web-based chunk retrieval, although the stream responses here return larger chunks of data. Prior work has shown the benefits of adopting such compute-aware mech-

anisms in AR/XR scenarios for smoother play-out through improved latency variance for OTT chunk responses [13, 7].

The need to remove explicit DNS resolutions and adopt in-band service routing is amplified when considering scenarios that involve Internet of Things (IoT) devices. IoT devices usually lack computing, memory, and energy resources [14], thus having the IoT aggregator also act as a SAR in such scenarios helps alleviate the complexity in device design.

Communication semantics in IoT scenarios may vary from single destination requests to more complex *multicast* communication across multiple network devices. For instance, a device in an industrial network may update several controllers at once. Applying linear order routing policies here may yield interesting benefits if such policies are communicated along with the data provided, e.g., in cases where physical devices sense their environment and exchange data with other devices. For instance, expressing a time-specific data selection policy such as *"Send me the latest data for x.."* as an anycast routing policy where the anycast group members consists of those sensors possibly providing value x, allows for using the convergence of the in-band service routing protocol to establish a relation with the sensor fulfilling the policy, avoiding aggregator or polling approaches at the application level. Other examples for such policies may be *max/min of all values x* or *latest update against a checkpoint c*, expressed as a multi-optimality problem but extended to anycast selection, and implemented using a distance vector protocol.

Finally, we consider the Distributed Ledger Technologies (DLTs) use case, a fundamental building block to realize distributed consensus [15] in applications like cryptocurrencies to distributed file storage systems. Here, information about transactions is stored as immutable ledgers in all participating network nodes. Key is a *diffusion* relationship between service instances realizing the DLT services when diffusing to and querying from ledgers, which can be expressed as a *1-to-random M out of N* multicast relation. Today's systems realizing DLTs, such as Ethereum, realize this multicast semantic through iterative unicast relations [16] that are implemented entirely at the participating DLT peers.

B. KPIs

To complement our communication patterns, we outline the KPIs that may be used to evaluate the benefits of in-band service resolution. We focus on two key categories, namely *latency* and *efficiency*.

1) Latency: We describe several metrics that can be used to quantify the benefits of in-band service routing in terms of time incurred by a transaction. *Request completion time (RCT)*, i.e., the time it takes for the transaction to be completed, is one such metric where replacing the explicit resolutions with an in-band approach is the key reduction factor. However, we expect this reduction to diminish with longer transactions or for scenarios that may utilize a client-local DNS cache entry. Nevertheless, the latter depends on the frequency with which the DNS cache is flushed as doing so regularly will force renewed resolutions, and thus longer resolution times. In use cases such as web page retrieval, we express this latency as a *composite completion latency* because it captures the sum of the individual object RCTs, where each object's resolution may require steering to a service address (e.g., a URL) when the objects are not co-located. Additionally, the *request completion time variance* may be an important metric to judge the stability of in-band resolutions, compared to, e.g., DNS-based resolutions.

Other metrics include *time to readiness* as the latency after which a service instance is ready to service a request. This allows us to evaluate how fast a service instance may serve requests after becoming available, compared against the time to propagate DNS changes. To evaluate the benefits of linear order routing policies described in §III-A, we describe *linear order request completion time* as the time taken to serve a request based on some linear order across several possible service instances, e.g., *'give me the maximum value across those available at N service instances'*. Additionally, *instance selection latency* describes the time to switch affinities between service instances (in a stable system). We use this to evaluate the impact of a renewed service instance selection, in contrast to a cold start resolution in DNS. Likewise, we use *instance selection latency variance* to measure the variability experienced in switching affinities, which may be important for, e.g., transaction mobility when a client transfers from one service instance to another mid-transaction. Such variance may also serve as a latency threshold to determine if a 'better' service instance than the one currently used is available to switch to.

2) Efficiency: Here, we describe metrics that capture aspects of performance and cost required for in-band service routing. *Linear order request efficiency* compares the use of linear order routing approaches against application-layer, client, or aggregator polling, and is measured in terms of the number of messages exchanged or costs for ephemeral storage needed for aggregation. *Service announcements overhead* captures the signalling overhead for announcing services in the overlay, compared to the propagation of DNS updates in today's systems. We use *service routing stretch* to measure the path stretch of the initial service request through the overlay, exposing any penalties for piggybacking the resolution request may have on the initial data packet. Finally, *resolution rate* measures the number of requests per second that can be resolved in an in-band service routing overlay, compared to the aggregate DNS resolution rate of the same set of networks across which the resolution overlay spans.

IV. Evaluation

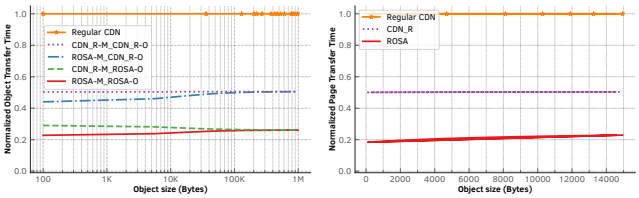
In this section, we use the communication patterns and KPIs described in §III to quantify the benefits of in-band service resolution for two use cases, web browsing and AR/XR chunk retrieval. We focus only on latency KPIs for brevity, however, this is sufficient for showing the aforementioned benefits.

A. Webpage retrieval

To better understand the benefits of in-band resolution for web browsing, we use the basic *Request-Response* model to

TABLE I: *Simulation Parameters, RTT based on [17]*

Parameter	*Value*
C_DNS_{RTT}	10 ms
CDN_{RTT}	10 ms
Bandwidth (L)	100 Mbps
MTU	1500 Bytes
Main Page size	10 Kilobytes
Number of objects	72
Object size range	[100 Bytes - 1 Megabytes]



(a) Page Transfer Time (b) Time to first byte

Fig. 1: OTT in the different domains

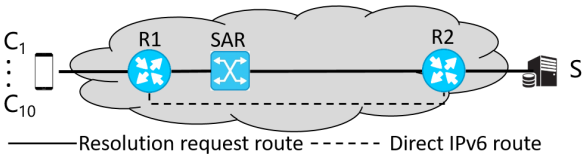


Fig. 2: Experiment topology

first give an overview of how web browsing generally occurs. A user enters a URL into the search bar of a web browser, initiating a lookup of the main page followed by the fetching of any referenced objects. The main page and referenced objects each require a DNS lookup if they are not co-located with any previously resolved objects. The time it takes to fetch an object is influenced by object size, location, etc.

To substantiate in-band resolution benefits, we focus on the *Request completion time (RCT)* described in §III-B considering a scenario where all web objects are served via CDNs. We analytically model the time to load the entire page as the *page transfer time (PTT)*, instead of the more common *page loading time* (PLT) in order to focus our evaluation on the *transfer* and not on the rendering of objects (which is largely independent from the transfer of objects over the network).

In our model, we assume that objects may either be located (i) in ROSA-adapted servers, thus being exposed via the in-band resolution system (see §II) and (ii) on CDN servers. We further assume that a ROSA SAR is located at the ingress of the CDN and the client only uses ROSA to request the webpage objects, either leading to case (i) or (ii) above. We denote the object location probabilities for (i) and (ii) above as P_R and P_C with $P_R + P_C = 1$. For simplicity, we do not model web pages that have zero embedded objects. We then compare this access to webpage objects against a baseline, termed *Regular_CDN*, of accessing all objects in the CDN after a DNS resolution.

We denote the link speed as L and the RTT for receiving packets from the CDN as CDN_{RTT}. Here, L is constrained by the user's access link and we assume the web traffic having exclusive access to the link, thus not considering any cross traffic, congestion, etc. Using insights from the QUIC 1-RTT handshake [18], we model the retrieval of a single object as *object transfer time* (OTT), made up of five constituent packets, namely one each for QUIC/TLS handshake, object request and object response as well as two acknowledgements (ACK), to be transferred from the CDN via the access link with constrained speed L, upper bounding the latency by assuming packets are MTU (maximum transfer unit) bytes. Lastly, we assume object sizes distributed with $O(k)$, following one of several possible distributions (e.g., long-tailed Pareto or LogNormal), thus leading to $\frac{O(k)}{MTU}$ packets to be sent to the client. Furthermore, we assume that for any DNS resolution in case (ii) a latency of C_DNS_{RTT}. This leads us to the following OTT equation, where the first row represents the transmission over the user access link, the second row being the latency to deliver the packets over the client-CDN link, and the last one considering the possible DNS latency for case (ii):

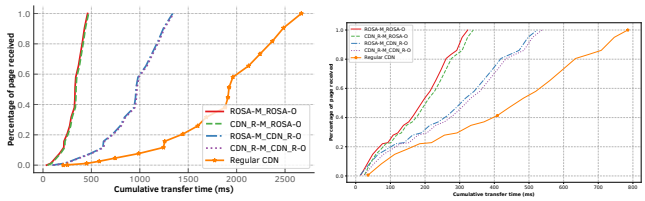
$$OTT = \frac{5 \times MTU + O(k)}{L} + (5 + \frac{O(k)}{MTU}) \times \frac{CDN_{RTT}}{2} + P_C \times C\_DNS_{RTT} \quad (1)$$

From equation 1, we derive the sequential PTT (i.e., retrieving one object at a time) in equation 2 and the concurrent PTT (i.e., retrieving all objects in parallel after receiving the main page) in equation 3. Here, we differentiate between retrieving the main page M and then the main page's referenced objects $O_i, 1...O(k)$. For this, we consider that objects not co-located with other retrieved objects will require their own resolution requests and model this aspect through a distribution of the *object uniqueness*, defined as follows: For a web page j, the object uniqueness u_j is a uniformly distributed random variable in the range $[1-5]$, where larger values for u_j represent a large number of objects being unique, thus increasing the impact of the discovery component in the PTT.

$$S\_PTT = OTT(M) + \left(\sum_{i=1}^{O(k)} OTT(O_i) \right) u_j \quad (2)$$

$$C\_PTT = OTT(M) + \left(\max_{1 \le i \le O(k)} (OTT(O_i)) \right) u_j \quad (3)$$

We use the simulation parameters described in Tab. I to evaluate the performance of our PTT model. We set the number of referenced objects by a main page to 72 based on the reported median object requests on desktops as of 01 June 2023 [19]. As described earlier, we use the *Regular_CDN* configuration as our baseline with all objects located in a standard CDN, while *CDN_R* refers to a CDN with a SAR located at its ingress using in-band resolution instead. *ROSA_M* and *ROSA_O* represent a scenario where the main page and



(a) Simulation (b) Experiment

Fig. 3: What content to serve first

all objects are located on ROSA servers (within the CDN), respectively, thus discovered using in-band resolution.

Using the PTT model described in equation 3, we simulate a web download with the results shown in Fig. 1a. Assuming all objects are requested in parallel, the resulting PTT depends on how long it takes to download the largest object. Note that the object size shown on the x-axis is the maximum size of the referenced objects, however, the resulting PTT calculation takes into account the size of the Main Page given in Tab. I. As expected, we observe that serving objects via ROSA servers offers the most reduction in transfer time, nearly 70% compared to the *Regular_CDN* for small objects. For the *CDN_R* scenario, we observe that a minimal implementation of in-band resolutions through the ingress-located SAR offers significant benefits for CDNs, about 50% reduction in transfer time as shown by the *CDN_R-M_CDN_R-O*, even if none of the objects are actually located on ROSA servers. Such deployment enables a *web accelerator* scenario since the resolution requests for the CDN all have reduced discovery time through the in-band resolution, piggybacked on already sent data. Placing either main page only (*ROSA_M* configuration) or even all objects (*ROSA_O* configuration) on ROSA-enabled servers further improves on the PTT, as expected, down to about 65% of the *Regular_CDN* values.

Prior work indicates that displaying useful parts of a web page as fast as possible is key to user retention. Google recommends that a web page's *Largest Contentful Paint* (LCP), which measures the time to rendering the largest object [20], to be at most 2.5 seconds. This shows the user that the page is useful by showing content that is relevant for them to decide on their browsing experience. Web page designers usually follow the 14KB rule [21], which advocates web server responses to fit within the TCP initial window. With the above in mind, we calculate PTT improvement for small object sizes (up to 14KB) for the aforementioned 3 domains of content locations, and show the results in Fig. 1b. We observe about 75% reduction in transfer time for objects on ROSA servers compared to the Regular CDN. *This demonstrates that using in-band resolutions can present significant improvements on the 14KB rule sought by web developers.*

We showcase the ability provided by in-band resolutions to improve LCP in Fig. 3. We use the parameters described in Tab. I but change the distribution of object sizes to [100 bytes - 150 Kilobytes]. This allows us to zoom in and show how long it would take to serve a certain percentage of a web page based on object location. Additionally, we limit the number of objects to 20 to match the experiment, where we only serve 20 distinct objects to limit the load on the server. As before, all objects are requested in parallel. We show the cumulative time taken to serve a certain percentage of the total web page, e.g., based on the largest object served, and observe that more than 20% of the page can be delivered faster than the recommended 2.5 seconds using ROSA. Even the minimal *CDN_R* deployment provides a 1 second reduction in serving about 40% of the page. We run a similar experiment to request the objects using the sequential model described in equation 2 and confirm that, regardless of the retrieval model, the degree of improvement is similar in both instances. We omit the results here for brevity.

To support our analytical results, we set up an experiment in which a prototype forwards the in-band resolution requests described in §II via eBPF [22], and use user space socket libraries at both the client and server. We realize the topology shown in Fig. 2 in a containerized virtual IO environment on an 8-core i7 laptop. *Server S* hosts the distinct objects and we obtain the various placements (in ROSA, CDN_R, or Regular CDN) by simulating delayed responses to the client requests according to the parameters in Tab. I.

Fig. 3b shows results for time to serve content, and we observe a similar trend in the curve as in Fig. 3a. We note a maximum performance improvement of about 2.6 between serving objects from ROSA and the Regular CDN, although there is a decrease in transfer time, compared to our simulation, due to the higher back-end network capacity.

The results in Fig. 3 may motivate content delivery strategies where web content providers devise page caching strategies considering which relevant objects should be served from the domain with the lowest loading time. For instance, caching content based on how much loading time will be needed to load 50, 70, or 90% of the web page. An e-commerce website that yields revenue from serving as many Ads as possible may devise a strategy that serves Ads before the rest of the web page, i.e., by ensuring that the Ads are served as part of the first 20% of the objects loaded.

While provider content placement strategies often seek to minimize transportation cost and data storage, we posit that it is equally important to consider web page content's *'relevance-to-user'*. Consider search engine results where the user can immediately make a decision of whether to follow a certain link or change the search parameters based on a subset of information that is loaded first. Instead of waiting for all results to load before deciding whether to click next or not, the web page designer might want to serve the crucial information first so that the user can decide whether the results about to load are relevant to their search or not. This would save time and network resources. *We posit that the improvements seen here through in-band resolution approaches improve on those capabilities within the acceptable limits for latency.*

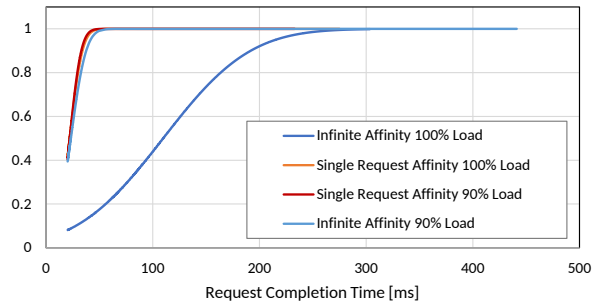


Fig. 4: Latency variance for single request vs infinite affinity

B. HTTP chunk (video) retrieval

Let us now consider the HTTP-based retrieval of data chunks, e.g., for videos, which follows the same *Request-Response* pattern as web browsing, where the request is repeatedly issued towards the same server albeit with changing file components in the URI. Although the resolution latency does play a role, the key focus here lies on the ability to quickly change the server endpoint with which a client may communicate, again we focus on *Request completion time (RCT)* described in §III-B. We assume the chunks are replicated in more than one network location and are retrieved following a typical AR/VR inter-arrival time for chunk requests, similar to the work in [7]. To highlight the benefits of the service routing approach, we define two retrieval modes: The *single request affinity* utilizes the stateless nature in that a content chunk can be retrieved from any of the replica instances with the transaction consisting of a single, e.g., HTTP request, thus changing the replica with every transaction. In contrast, *infinite affinity* emulates Layer 7 techniques where longer-term decisions on replica selection are performed upfront (through the explicit resolution), with the transaction lasting for the duration of the service, thus not changing the choice of replica from which data chunks are retrieved.

We use the experimental setup used in the last section, including the topology in Fig. 2. Server *S* hosts five replica of the same service and we signal transaction affinity by 'resetting' the service socket, thus reinitiating the in-band (piggybacked) resolution, possibly assigning the request to another replica as a result. We implement the mechanism in [13] for replica assignment, and assign equal weights for compute capabilities, resulting in round robin scheduling. Each replica serves a request following a 20ms exponential distribution, while client requests are generated through a uniform distribution within an interval from 100 to 500ms. Here, we follow the rationale in [7] to accommodate user interaction that may lead to short retrieval intervals for new content in AR/VR scenarios. We run evaluations for one minute at 90% and 100% system load, and show both average service completion time and its variance in Fig. 4 for both affinity modes.

We observe that the difference at 90% load (the overlapping lines on the left) is minimal, however, the average completion time for request level affinity (at 22ms) is slightly improved compared to infinite affinity (at 23ms average latency). At 100% load, we observe the deterioration of the infinite affinity performance, with the request-level affinity average completion time identical to the 90% load performance curve. We attribute this superior performance to being able to use any of the five service instances, unlike the infinite affinity mode which is linked to a single instance at the start of the service.

Load on the resolution system is an important insight. While the number of generated client requests in our setup is still small (up to 250 requests per second in single affinity mode), scaling such scenario with the DNS could quickly overwhelm it. For instance, decreasing the request handling time to 5ms would increase the number of requests at 100% load to 1000 requests per second, which would additionally increase if scaled across a number of deployed use cases. Thus shortening the affinity, even down to a single request, becomes problematic or entirely impossible for DNS-based systems. The eBPF-aided lookup of service names in SARs in our setup allows for lookup request handling rates of several hundreds of thousands, thus being able handle shorter affinities. Further gains may be obtained if the current SW-based approach is improved through, e.g., network card support for eBPF or native kernel integration of the forwarding operation.

Latency budget is another concern, particularly in AR/VR scenarios. As shown in [23], even entry-level AR/VR services require much lower motion-to-photon latency (around 20ms) than we applied here, hence, the resolution latency is a problem for short affinity requests using explicit discovery. The range provided in [7] is between 15 to 45ms for CDN DNS systems, thus, the 20ms delay budget may already be exceeded, making short affinities impossible. We argue that the insights from our AR/VR use case also generally transfer to other use cases that employ the same communication pattern of frequent file retrieval from possibly replicated endpoints. *Since this pattern is used in e.g., over-the-top video, SW downloads, etc, as shown in §III, our insights could be transferred to more than 82% of Internet traffic [24].*

V. Related Work

Prior work on reducing latency for efficiency and to improve user QoE is extensive. Methods such as edge computing focus on placing services closer to users while virtualization and serverless computing remove the need for hosting services on explicit server hardware. Caching (temporarily storing frequently accessed closer to the user) and prefetching (resolving domain names and putting content into the cache before users click on the link [25]) also help in reducing latency. CDN providers typically employ strategic content placement policies where large objects are pushed to caches to reduce pressure on the origin servers [26], based on observed delay [27] or learning algorithms [28]. While this reduces latency, these methods rely on the traditional DNS+IP model to resolve service requests, thus still incurring the DNS resolution latency. The benefits of replacing this reliance on the DNS with an in-band resolution was introduced in [5] and initially investigated

in [7], showing improved service request completion times; an insight we verified experimentally in §IV.

Service routing builds on earlier work on Information-Centric Networking (ICN) [3] in which routing is based on information identifiers. Both works decouple data (services in our case) from specific *network locations* of the hosts realizing the service (or storing the data). ICN deployment has, however, been minimal due to lack of "a clear incremental deployment strategy" [29]. Service routing circumvents this by relying on existing IP protocol capabilities, specifically extension headers.

VI. Conclusion

In this paper, we investigated a method for reducing communication latency by piggybacking the service resolution request in the initial data packet, instead of incurring the latency of traditional DNS resolutions. We developed a taxonomy of communication patterns and KPIs to provide exemplifying insights into application performance. Across selected use cases, we identify significant improvements, such as the reduction of page retrieval times in web browsing of up to 60% and a significant reduction in latency variance for smoother playout experience in AR/XR scenarios. Future work will look at testing the proposed PLT model with object size distributions from longitudinal web studies such as in [10].

References

[1] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing TCP's Initial Window ," RFC 6928, Apr. 2013.

[2] J. Iyengar and M. Thompson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT*, 2009.

[4] E. Schurman and J. Brutlag, "Performance related changes and their user impact," in *Web Performance and Operations Conference*, 2009.

[5] D. Trossen, L. M. Contreras, J. Finkhäuser, and P. Mendes, "Routing on Service Addresses," Feb. 2023, internet-Draft.

[6] "What is gslb?" https://www.efficientip.com/what-is-gslb/, 2022.

[7] K. Khandaker, D. Trossen, J. Yang, Z. Despotovic, and G. Carle, "On-path vs off-path traffic steering, that is the question," in *ACM SIGCOMM FIRA Workshop*, 2022.

[8] B. Hinden and D. S. E. Deering, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, Dec. 1998.

[9] "Mqtt publish/subscribe architecture (pub/sub) – mqtt essentials: Part 2," https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/, 2023.

[10] A. Asrese, E. A. Walelgne, V. Bajpai, A. Lutu, O. Alay, and J. Ott, "Measuring web quality of experience in cellular networks," in *PAM*, 2019.

[11] T. Wilkie, "The red method: key metrics for microservices architecture," https://www.weave.works/blog/the-red-method-key-metrics-for-microservices-architecture/, 2017.

[12] "Measuring video quality and performance: Best practices," https://www.akamai.com/resources/white-paper/measuring-video-quality-and-performance-best-practices, 2020.

[13] K. S. Khandaker, D. Trossen, R. Khalili, Z. Despotovic, A. Hecker, and G. Carle, "Cards: Dealing a new hand in reducing service request completion times," in *IFIP Networking*, 2022.

[14] S. J. Saidi, S. Matic, O. Gasser, G. Smaragdakis, and A. Feldmann, "Deep dive into the iot backend ecosystem," in *ACM IMC*, 2022.

[15] "Distributed ledger technology and blockchain: Perspectives for eu-lisa and large-scale it systems: Research and technology monitoring report," https://www.eulisa.europa.eu/Publications/Reports/DLTs and blockchain report.20Dec2019.pdf, 2019.

[16] D. Guzman, D. Trossen, M. McBride, and X. Fan, "Insights on impact of distributed ledgers on provider networks," in *Blockchain - ICBC*, 2022.

[17] R. Atlas, "Comparative dns root rtt," https://atlas.ripe.net/results/maps/comparative-dns-root-rtt/, 2024.

[18] M. Thomson and S. Turner, "Using TLS to Secure QUIC," RFC 9001, May 2021.

[19] H. Archive, "Report: Page weight," https://httparchive.org/reports/page-weight#bytesTotal.

[20] G. C. Team, "Largest Contentful Paint (LCP)," https://web.dev/lcp/, 2023.

[21] S. H. Eskildsen, "Increase http performance by fitting in the initial tcp slow start window," https://sirupsen.com/napkin/problem-15, 2021.

[22] "What is ebpf?" https://ebpf.io/what-is-ebpf/, 2022.

[23] P. Liu, P. Eardley, D. Trossen, M. Boucadair, L. M. Contreras, C. Li, and Y. Li, "Computing-Aware Networking (CAN) Problem Statement and Use Cases," 2022, internet-Draft.

[24] Cisco, "Cisco annual internet report (2018-2023)," 2020.

[25] "Dns prefetching," https://www.chromium.org/developers/design-documents/dns-prefetching/, 2023.

[26] "Content characteristics & object delivery," https://techdocs.akamai.com/object-delivery/docs/content-characteristics-and-od, 2023.

[27] G. Pallis, A. Vakali, K. Stamos, A. Sidiropoulos, D. Katsaros, and Y. Manolopoulos, "A latency-based object placement approach in content distribution networks," in *Third Latin American Web Congress*, 2005.

[28] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "Making content caching policies 'smart' using the deepcache framework," *ACM SIGCOMM*, 2019.

[29] G. Carofiglio, L. Muscariello, J. Augé, M. Papalini, M. Sardara, and A. Compagno, "Enabling icn in the internet protocol: Analysis and evaluation of the hybrid-icn architecture," in *ACM ICN*, 2019.