Workgroup: Network Working Group
Internet-Draft: draft-trossen-rtgwg-rosa-02
Published: 3 February 2023
Intended Status: Standards Track
Expires: 7 August 2023
Authors: D. Trossen
Huawei Technologies
LM. Contreras
Telefonica
J. Finkhaeuser
Interpeer gUG
P. Mendes
Airbus

Routing on Service Addresses

Abstract

This document proposes a novel communication approach which reasons about WHAT is being communicated (and invoked) instead of WHO is communicating. Such approach is meant to transition away from locator-based addressing (and thus routing and forwarding) to an addressing scheme where the address semantics relate to services being invoked (e.g., for computational processes, and their generated information requests and responses).

The document introduces Routing on Service Addresses (ROSA), as a realization of what is referred to as 'service-based routing' (SBR), to replace the usual DNS+IP sequence, i.e., the off-path discovery of a service name to an IP locator mapping, through an on-path discovery with in-band data transfer to a suitable service instance location for a selected set of services, not all Internet-based services.

SBR is designed to be constrained by service-specific parameters that go beyond load and latency, as in today's best effort or traffic engineering based routing, leading to an approach to steer traffic in a service-specific constraint-based manner.

Particularly, this document outlines sample ROSA use case scenarios, requirements for its design, and the ROSA system design itself.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents

at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 7 August 2023.

Table of Contents

1. Introduction

The centralization of Internet services has been well observed, not just in IETF discussions [Huston2021] [I-D.nottingham-avoiding-internet-centralization], but also in other efforts that aim to quantify the centralization, using methods such as the Herfindahl-Hirschman Index [HHI] or the Gini coefficient [Gini]. Dashboards of the Internet Society [ISOC2022] confirm the dominant role of CDNs in service delivery beyond just streaming services, both in centralization as well as resulting market inequality, which has been compounded through the global CV19 pandemic [CV19].

The impact on routing can be seen in, e.g., [TIES2021], which goes as far as centralizing service requests into a single IP address behind which data centre (DC) internal mechanisms take over.

Thus, ROSA is being motivated by the requirements stemming from use cases where the distribution of compute, storage, and networking resources associated with a service brings not just benefits but also its distributed, runtime utilization may yield in better performance, such as improved service completion latency, utilization, and others.

At the same time, it is important to recognize that we do not aim for replacing existing service routing capabilities, most notably the DNS as the main form of resolving a service name into routing

locator; we see those capabilities working perfectly well for many Internet services. Instead, we argue in the following that some more challenging service scenarios, such as multi-access edge computing and cloud-to-thing, as well as more dynamic networking scenarios such as LEO constellations, may require service routing capabilities embedded in the networking stack, without relying on application layer translations or resolution services.

1.1. Backdrop

Providing the backdrop to this draft, [EI2021] addresses the challenge of overcoming the architectural stagnation of the Internet while supporting an increasing divergence of services, by means of an extensible Internet architecture able to support in-network services that go beyond best-effort packet delivery. Within this extended Internet, novel network services are executed in Service Nodes (SNs) interconnected by networks running IP. Deployment of SNs depends upon the use-case, but in general may be placed within Last Mile Providers (LMPs) or within Cloud Providers (CPs). The proposed ROSA framework follows a similar architectural view.

Evolving the IPv6 network layer has been part of its design from the very start. Key enabler here are Extension headers (EHs), which are part of the IPv6 specifications [RFC8200], with some observed problems, e.g., firewall traversal, in real-world deployments [SHIM2014]. Recent solutions, such as Segment routing (SR) [RFC8402], specifically SRv6 [RFC8986] build on this capability by establishing a shim layer overlay (of SR-enabled routers), utilizing an extension header to carry needed information for realizing the source routing capabilities. ROSA follows a similar approach, by using EHs to build a shim overlay above the IP layer.

1.2. Design Goals

The key problem in service routing is that of determining the routing locator for a network endpoint that realizes a specific service. For this, explicit resolution steps are usually implemented at the application level, through mechanisms such as DNS, GSLB, and Alto. The result of the explicit resolution is then used to establish a suitable communication at the application level, including transport sessions, to invoke a particular service, possibly followed by subsequent data transfer to the endpoint chosen in the resolution step (called 'affinity' in the following).

This additional resolution does cost time in the form of the resolution latency, including the latency to access the resolver, but also requires for the resolver to have available the relevant (and up-to-date) mappings for any incoming resolution request. Furthermore, updating the mappings is a difficult problem, either

requiring to push such updates to the resolver or pull suitable updates from elsewhere. As outlined in [OnOff2022], both aspects lead to problems when wanting (or needing) to support shorter interactions with service instances, while interactions may be served by different service instances.

Moreover, services that rely on application specific resolvers (e.g. DNS servers) may fail when facing intermittent connectivity to those resolvers, as can happen in moving networks (e.g. vehicle networks).

This puts three goals in the foreground that are important for use cases for ROSA, namely (i) need for 'dynamicity' and (ii) 'efficiency' as well as (iii) the 'service specificity' of the steering decision, i.e., the selection of the suitable service instance to send traffic to. The first is about the support for fast changing relations with service endpoints, while the second aims to reduce any potential latency in doing so. The third caters to the situation that the selection of one of the possibly many choices for a service endpoint is often defined through service-specific policies, including runtime decisions that may change from one transaction to another.

1.3. Summary of Contribution

The main contribution of Routing on Service Addresses (ROSA) is to replace the usual DNS+IP sequence, i.e., the off-path discovery of service name to IP locator mapping, through an on-path discovery with in-band data transfer to a suitable service instance location for a selected set of services, not all Internet-based services.

The basic functionality of ROSA can be described as follows:

1 A client sends an initial IP packet, 'directed' to service address S, to a special shim (ROSA) overlay.

2 The shim overlay routes the packet based on the service address to one of the possibly many service instances for S over an existing IP network. For this, mappings between S and the known service instance locators are used by the ROSA overlay, replacing the role of DNS records, while the selection of the 'suitable' service instance locator may use service-specific policies (and parameters).

3 The chosen service instance delivers its network locator SI in the response to the initial packet back to the client.

4 The client will now continue to use SI in native IPv6 packets to direct any subsequent packets to the chosen service instance. This is to support possible ephemeral state created at service instance as a consequence of previous exchanges.

Steps 1 through 4 are repeated for every new service transaction, allowing those transactions now to be served at any of the available service instances albeit keeping one transaction at one chosen service instance! Steps 1 through 4 may also be repeated in case of mobility. For stateless services, only steps 1, 2, and 3 are executed.

In order to react to system, e.g., network but more importantly service changes, ROSA achieves dynamicity, as mentioned in the previous section, by including a routing-based approach able to map service addresses to routing locators, where mappings of service addresses to routing locators are pushed to the (shim overlay) elements, enabling to perform the translation from a service addressed packet to an IP-addressed packet on the data path. When using, e.g., eBPF-based techniques in SW-based routers, such approach can achieve 100s of thousands of resolution steps per ingress node, as discussed in Section 9.

When it comes to efficiency, our design is positioned at L3.5, using an extension header based approach. With this, the initial packet, realizing an in-band resolution step, can include upper layer, i.e., transport and/or application-level, in-band data within the normal payload of the IP packet, further reducing the latency for completing a transaction, even opening the possibility for single packet transactions being completed in a single round trip.

Additionally, similar to application-level solutions, the positioning as a (L3.5) shim overlay faciliates the exposure of service-specific selection policies from the service to a ROSA provider through explicit commercial relations, separate from those defining the routing policies in the underlay network.

Unlike name-based routing solutions at the underlay, routing scalability is achieved by limiting the resolution to those services explicitly announced to the service routing (i.e., ROSA) overlay. Thus, ROSA does not aim to replace ALL service routing through the above proposed steps, but focus on those services explicitly announcing their desire for a ROSA-based resolution to an appropriate ROSA provider. The assumed explicit (often commercial) relationship between the service provider and the ROSA provider is what allows for controlling the scalability requirements of the elements realizing the ROSA overlay.

1.4. Overview of Draft

In the remainder of this document, we first introduce in Section 2 a terminology that provides the common language used throughout the remainder of the document. We then introduce use cases in Section 3 that drive the need for a routing on service address solution. Our

analysis in Section 4 then leads us to outline in Section 5 the requirements for ROSA and the expected benefits of ROSA in Section 6.

The main part of the document focusses on introducing the ROSA design in Section 7, elaborating on the main idea presented above in more detail, followed by possible extensions to the design in Section 8.

2. Terminology

The following terminology is used throughout the remainder of this document:

Service: A monolithic functionality that is provided according to the specification for said service. A composite service can be built by orchestrating a combination of monolithic services.

Service Instance: A running environment (e.g., a node, a virtual instance) that provides the expected service. One service can involve several instances running within the same ROSA network at different network locations, thus providing service equivalence between those instances.

Service Address: An identifier for a specific service.

Service Instance Address: A locator for a specific service instance.

Service Request: A request for a specific service, addressed to a specific service address, which is directed to at least one of possibly many service instances.

Affinity Request: A request to a specific service, following an initial service request, requiring steering to the same service instance chosen for the initial service request.

Service Transaction: A sequence of higher-layer requests for a specific service, consisting of at least one service request, addressed to the service address, and zero or more affinity requests.

Service Affinity: Preservation of a relationship between a client and one service instance, with the initial service request creating said affinity and following affinity requests utilizing said affinity.

ROSA Provider: Realizing the ROSA-based traffic steering capabilities over at least one infrastructure provider by

deploying and operating the ROSA components within its defining ROSA domain.

ROSA Domain: Domain of reachability for services supported by a single ROSA provider.

ROSA Endpoint: A node accessing or providing one or more services through one or more ROSA providers.

ROSA Client: A ROSA endpoint accessing one or more services through one or more ROSA providers, thus issuing services requests directed to one of possible many service instances that have previously announced the service address provided by the ROSA client in the service request.

Service Address Router (SAR): A node supporting the operations for steering service requests to one of possibly many service instances, following the procedures outlined in Section 7.5.

Service Address Gateway (SAG): A node supporting the operations for steering service requests to service addresses not announced to SARs of the same ROSA domain to suitable endpoints in the Internet or within other ROSA domains.

3. Deployment and Use Case Scenarios

In the following, we outline examples for use cases that exhibit the degrees of distribution in which relationship management (through explicit mapping and/or gatewaying) may become complex and a possible hindrance for service performance. The following sections only serve as illustrating examples with other work, such as the BBF Metro Compute Networking (MCN) [MCN], among others, having developed similar but also additional use cases.

3.1. CDN Interconnect and Distribution

Video streaming has been revealed nowadays as the main contributing service to the traffic observed in operators' networks. Multiple stakeholders, including operators and third party content providers, have been deploying Content Distribution Networks (CDNs), formed by a number of cache nodes spread across the network with the purpose of serving certain regions or coverage areas with the proper quality levels. In such a deployment, protection schemas are defined in order to ensure the delivery continuity even in the case of outages or starvation in cache nodes.

In addition to that, novel schemes of CDN interconnection [RFC6770] [SVA] are being defined allowing a given CDN to leverage the installed base of another CDN to complement its overall footprint.

As result, several caches are deployed in different Points of Presence in the network. Then for a given content requested by an end user, several of those caches could be candidate nodes for delivery. Currently, the choice of the cache node to serve the customer relies solely on the content provider logic, considering only a limited set of conditions to apply.

For instance, the usage of cache-control [RFC7234] allows data origins to indicate caching rules downstream. Since the original intent was quite limited, to operate between the data source and the data consumer (browser), Targeted Cache Control (TCC) [RFC9213] defines a convention for HTTP response header fields that allow cache directives to be targeted at specific caches or classes of caches.

The performance can be improved by the consideration of further conditions in the decision on what cache node to be selected. Thus, the decision can depend of course on the requested content and the operational conditions of the cache itself, but also on the network status or any other valuable, often service-specific, semantic for reaching those nodes, such data validity, end to end delays, or even video analytics. The latter is relevant since as the number of video files grows, so does the need to easily and accurately search and retrieve specific content found within them.

Furthermore, those decision points may be dynamic and could even change during the lifetime of the overall service, thus requiring to revisit decisions and therefore assignments to the most appropriate CDN node. An example encompasses the usage of satellites to enhance the content distribution efficiency in cooperation with the terrestrial network. Combining satellites with CDNs may leverage LEO (low earth orbit) satellite mobility characteristics to cache and deliver content among different static caches in the terrestrial CDN, but may also include mobile satellites serving as couriers.

3.2. Distributed user planes for mobile and fixed access

5G networks natively facilitate the decoupling of control and user plane. The 5G User Plane Function (UPF) connects the actual data coming over the Radio Area Network (RAN) to the Internet. Being able to quickly and accurately route packets to the correct destination on the internet is key to improving efficiency and user satisfaction. For this, the UPF terminates the tunnel set carrying end user traffic permitting to route the end user traffic in the network towards its destination, e.g., providing reachability to edge computing facilities.

Currently, UPF is planned to appear in two places, namely in the Core Network and at the Edge inside a Multi-Access Edge Controller

(MEC). However, in a future 6G network, it is envisioned that several UPFs can be deployed in a distributed manner, not only for covering different access areas, but UPFs can also be distributed with the attempt of providing access to different services, linked with the idea of network slicing as means for tailored service differentiation. For instance, some UPFs could be deployed very close to the access for services requiring either low latency or very high bandwidth, while others could be deployed in a more centralized manner for requiring less service flows. Furthermore, multiple instances can be deployed for scaling purposes depending on the demand in a specific moment.

Similarly, to what happens in mobile access, fixed access solutions are proposing schemas of separation of control and user plane for BNG elements [I-D.wadhwa-rtgwg-bng-cups] [BBF]. From the deployment point of view, different instances can be deployed based on the coverage, the temporary demand, etc, as before.

As a complement to both mobile and fixed access scenarios, edge computing capabilities are expected to complement the deployments for hosting service and applications of different purposes, for both services internal to the operator or hosting of services from third parties.

In this situation, either for both selection of the specific user plane termination instance, or from that point on, selection of the service endpoint after the user plane function, it makes sense the introduction of mechanisms enabling selection choices based on service-specific semantics.

3.3. Multi-homed and multi-domain services

Corporate services usually define requirements in terms of availability and resiliency. This is why multi-homing is common in order to diversify the access to services external to the premises of the corporation, or for providing interconnectivity of corporate sites (and access to internal services such as databases, etc).

A similar scenario in which external services need to be reached from within a specific location, is the Connecter Aircraft. Exploiting solutions that allow for the exploitation of multi-connected aircrafts (e.g., several satellite connections, plus air-to-ground connectivity) are important to improve passenger experience, while helping make the crew more productive with networking solutions that enable seamless, high-speed broadband. Managing a multi-connected Aircraft would benefit from mechanisms that would enable the selection of the best connection points based on service-specific semantics, besides the traffic related parameters considered by solutions such as SD-WAN, which aims to

automate traffic steering in an application-driven manner, based on the equivalent of a VPN service between well defined points.

The diversity of providers implies to consider service situations in a multi-domain environment, because of the interaction with multiple administrative domains.

From the service perspective, it seems necessary to ensure a common understanding of the service expectations and objectives independently of the domain traversed or the domain providing such a service. Common semantics can facilitate the assurance of the service delivery and a quick adaptation to changing conditions in the internal of a domain, or even across different domains.

3.4. Micro-service Based Mobile Applications

Mobile applications, installed on mobile devices such as smartphones and deployed through 'marketplace' platforms, usually install a monolithic implementation of the device-specific functionality, where this functionality may explicitly utilize remote service capabilities, e.g., provided through cloud-based services.

Application functionality may also be realized as micro-services themselves. When such micro-services are jointly deployed (i.e., installed) at the mobile device, its overall functionality resembles that of existing applications.

Micro-services architectures are usually best suited to larger, more complex applications built for scalability and agile iteration. In this scenario, a monolithic architecture can be a problem for applications as the codebase becomes unwieldy and difficult to manage.

However, micro-services may also be invoked on network devices other than the mobile device itself, utilizing service routing capabilities to forward the micro-service request (and its response) to the remote entity, effectively implementing an 'off-loading' capability. Efforts such as the BBF MCN work capture this aspect as 'edge-to-edge collaboration', where in our case here the edge does include the end user devices themselves.

A distributed system like microservices inevitably introduces additional complexity as multiple moving parts need to be synchronized in a way that allows them to work as a unified software system. If services are split across servers you will have to provision that multi-faceted infrastructure. This is where a service-centric network solution able to coordinate the chain of such micro-services could plan an important role.

The work in [I-D.sarathchandra-coin-appcentres] proposes such approach, positioning compute capabilities as forming a distributed (app-centric) data centre. The simple example in [I-D.sarathchandra-coin-appcentres] outlines the distribution of video reception, processing, and displaying capabilities as individual micro-services. With this, remote (edge computing) capabilities may be used for complex processing beyond those of the mobile device. This includes, for instance, to utilize hardware, such as displays, other than the device's built-in one.

Interaction may be one driver for dynamicity in those scenarios. For instance, the aforementioned display indirection may take place at high frequency, triggered by sensory input (e.g., gaze control) to decide which instance is best to direct the video stream to. This may be beneficial for new, e.g., gaming experiences that utilize immersive device capabilities. Other examples may include the offloading of processing capabilities (in case of 'better', i.e., more capable, processing being available elsewhere).

As briefly discussed in [I-D.sarathchandra-coin-appcentres], such micro-service design may well be integrated into today's application development frameworks, where a device-internal service registry would allow for utilizing device-local service instances first before directing the service invocation to the network to route the service request.

In conclusiom, this concept of application-centric microservices, deployed within other edge devices, including end user devices themselves, extends the concept of 'edge computing', also captured in use cases of the BBF MCN initiative [MCN], by foreseeing more focus on the device applications, aiming at higher dynamicity in relations being realized.

3.5. Constrained Video Delivery

Chunk-based video delivery is often constrained to, e.g., latency or playout requirements, while the content itself may be distributed as well as replicated across several network locations. Thus, it is required to steer client requests for specific content under specific constraints to one of the possibly many network locations at which the respective content may reside.

The work in [I-D.jennings-moq-quicr-arch] proposes a publish-subscribe metaphor that connects clients to suitable relays for delivering the desired content under the specific constraint. Within our context of service routing, the relays realize the service instances for a video delivery service, where the selection of the 'right' instance is being constrained by the requirements for the video's delivery to the client.

Instead, we suggest to complement QUICr by largely realizing the explicit publish/subscribe architecture in [I-D.jennings-moq-quicr-arch] through the traffic steering capabilities within a routing on service addresses infrastructure, specifically replace the explicit lookup for a suitable relay in [I-D.jennings-moq-quicr-arch] through a service routing operation with the aim to not just reduce any lookup latencies involved in the relay selection but also to enable a high dynamicity in the selection constraints.

3.6. AR/VR through Replicated Storage

One aspect of dynamicity in selecting content storages in the previous use case has been investigated in [OnOff2022] through the example of an AR/VR service that underlies a tight delay budget but would like to benefit from any replication of content chunks across more than one network location.

Here, a system of N clients is suggested to be retrieving content chunks from k service instances, where each chunk request is directed to any of the possible k instances; given the stateless nature of this service, any of the k instances is able to serve the chunk without knowledge of any previous one.

As shown in [OnOff2022], a retrieval that utilizes any of the k replicas significantly reduces the variance of the retrieval latency experienced by any of the N clients. Such reduced variance positively impacts the user experience through less buffering applied at the client side but also better adhering to the overall latency budget (often in the range of 100ms in AR/VR scenarios with pre-emptive chunk retrieval). Although pre-emptive retrieval is also possible in systems with explicit lookup operations, the involved latencies for such resolution may make it difficult to adhere to the latency budget for an E2E operation (see [I-D.liu-can-ps-usecases] for example latency budgets).

3.7. Cloud-to-Thing Serverless Computing

The computing continuum is a crucial enabler of 5G and 6G networks as it supports the requirements of new services, such as latency and bandwidth critical ones, using the available infrastructure. With the advent of new networks deployed beyond the edge, such as vehicular and satellite networks, researchers have begun investigating solutions to support the cloud-to-thing continuum, in which services distribute logic across the network, and storage is decentralized between cloud, the edge (most liked MEC) and the adhoc network of moving devices, such as aircraft and satellites.

In this scenario, a serverless-based service architecture may be beneficial for the deployment and management of interdependent distributed computing functions, whose behavior can be redefined in real-time. Serverless architecture is closely related to micro-services. The latter is a way to design an application and the former a way to run all or part of an application. That is the key to their compatibility. It is possible to code a micro-service and run it as a serverless function.

The consideration of serverless architectures is important for the Cloud-to-Thing continuum, since resources beyond the edge, in the adhoc part of the continuum, may be constraint and intermittently available. Hence it makes sense to leverage a serverless architecture that consists of a set of functions rather than services. The difference is that a service is permanently available whereas a function has a lifecycle as it is triggered, called, executed, runs and is then removed as soon as it is no longer needed. Serverless functions only run when they are needed, potentially saving significant resources.

In this scenario, the combination of a service oriented data plan with a model capable of delegating and adapting serverless functions in a cloud-to-thing continuum is important. The former need to be aware of the presence of different functions in order to be able to execute services based on the correct selection and invocation of different functions, within their lifetime. Most importantly, this awareness of the functions is likely to be highly dynamic in the nature of its distribution across network-connected nodes.

3.8. Metaverse

Large-scale interactive and networked real-time rendered tree dimension XR spaces, such as the Metaverse, follow the assumption that applications will be hosted on platforms, similarly to current web and social media applications. However, the Metaverse is supposed to be more than the participation in isolated three dimension XR spaces. The Metaverse is supposed to allow the internetworking among a large number of XR spaces, although some problems have been observed such as lock-in effects, centralization, and cost overheads.

In spite of the general understanding about potential internetworking limitations, current technical discussions are ignoring the networking challenges altogether. From a networking perspective, it is expected that the Metaverse will challenge traditional client-server inspired web models, centralized security trust anchors and server-style distributed computing, due to the need to take into account interoperability among a large number of XR spaces, low latency and the envisioned Metaverse pervasiveness.

In this context, an open and decentralized Metaverse, able to allow the internetworking of a large number of XR spaces, may be supported by intertwining distributed computing and networking. Hence it is expected that Metaverse applications may gain from a network able to support the execution of services while taking advantage of storage, networking, and computing resources located as close as possible from users, with a dynamic assignment of client requests to those resources.

3.9. Popularity-based Services

The BBF MCN use case report [MCN] outlines 'popularity' as a criteria to move from current explicit indirection-based approaches (such as DNS, GSLB, or Alto) to active service routing approaches.

Here, popularity, e.g., measured in service usage over a period of time, is being used as a trigger to announce a popular service to an active service routing platform, while less popular service continue to be served via existing (e.g., DNS-based) methods. Equally, services may be unannounced, thus retracted, from the service routing overlay to better control the overall cost for the provisioning of the service routing overlay.

With this, one could foresee the provisioning of a service routing overlay, such as ROSA, as an optimization for a CDN platform provider, either through commercially interfacing to a separate ROSA provider or providing the ROSA domain itself.

4. Analysis of Use Cases

We now discuss observations and suitability of existing technologies for realizing the use cases in the previous section, leading to the requirements outlined in Section 5. We then list the expected benefits for utilizing the ROSA design, presented in more detail in Section 7.

4.1. Observations from Use Cases

Several observations can be drawn from the use case examples in the previous section in what concerns their technical needs:

1 The namespace for services and applications is separate from that of routable identifiers used to reach the implementing endpoints, i.e., the service instances. Resolution and gateway services are often required to map between those namespace, adding management and thus complexity overhead, an observation also made in [Namespaces2022].

2 Service instances for a specific service may exist in more than one network location, e.g., for replication purposes to

serve localized demand, while reducing latency, as well as to increase service resilience.

3 While the deployment of service instances may follow a longer term planning cycle, e.g., based on demand/supply patterns of content usage, it may also have an ephemeral nature, e.g., scaling in and out dynamically to cope with temporary load situations as well as with the temporary nature of serverless functions.

4 Knowing which are the best locations to deploy a service instance is crucial and may depend on service-specific demands, realizing a specific service level agreement (with an underlying decision policy) that is tailored to the service and agreed upon between the service platform provider and the communication service provider.

5 Decisions for selecting the 'right' or 'best' service instance may be highly dynamic under the given service-specific decision policy and thus may change frequently with demand patterns driven by the use case. For instance, in our examples of Section 3.4 or Section 3.8, human interaction may drive the requirement for selecting a suitable service instance down to few tens of milliseconds only, thus creating a need for high frequency updates on the to-be-chosen service instance. As a consequence, traffic following a specific network path from a client to one service instance, may need to follow another network path or even utilize an entirely different service instance as a result of re-applying the decision policy.

6 Minimizing the latency from the initiating client request to the actual service response arriving back at the client is crucial in many of our scenarios. Any improvement on utilizing the best service instance as quickly as possible, thus taking into account any 'better' alternative to the currently used one, is crucial for reducing latency.

7 A specific service may require the execution of more than one service instance, in an intertwining way, which in turn requires the coordination of the right service instances, each of which can have more than one replica in the network.

We can conclude from our observations above that (i) distribution (of service instances), (ii) dynamicity in the availability of and chosing the the 'best' service instance, and (iii) efficiency in utilizing the best possible service instance are crucial for our use cases.

4.2. Suitability of Existing Internet Technologies

There exist a number of L4 through L7 based solutions that could be leveraged to fulfil the technical needs of the aforementioned use cases, with [I-D.liu-can-gap-reqs] providing an initial overview into the gaps that those solutions experience in the light of the observations above.

A key takeaway from this analysis is that the explicit indirection for service discovery, realized for instance through DNS, GSLB, or other solutions, poses a challenge to the dynamicity also observed in our use cases here due to the additional latency incurred but also due to the relatively static mapping of service name onto network locator that is maintained in most of those solutions. The work in [OnOff2022] investigates the impact of such off-path vs possible on-path decision making onto service performance and user experience.

The identifier/locator split provided by LISP [I-D.ietf-lisp-introduction] provides a similar separation of (an endpoint) identifier from its routable locator. In a way, a service address could be seen as an anycast EID in LISP. However, the reliance in LISP on a federated mapping service also positions LISP as an off-path solution with explicit resolution latency being incurred; this is due to the desired scale of LISP in which a routing-based solution (in contrast to the pull-based mapping service) is not tractable in terms of scalability, while for most services, a pull-based mapping service suffices. ROSA is based on the recognition that an explicit pull model (and its associated latency) may not be suitable for certain use cases (see Section 3), while still allowing for the traditional, e.g., DNS-based resolution methods being used for the wide range of services for which dynamicity and efficiency impact are not an issue.

Furthermore, the inherent anycast nature of service routing, when applied to replicated service instances, requires the use of anycast IP addresses, in turn often relying on centralized anycast routing architectures for delivering the service to the 'best' instance under the given anycast address. Lastly, communication over LISP does not see a difference between initial (service) requests and following (affinity) requests but instead realizes all communication through the EID abstraction.

Service instances or network service functions can be used by network operators to provide a better quality of service and manage their networks more efficiently. In this context, there is a growing interest in Service Function Chaining (SFC) [RFC7665], an ordered set of service functions that are applied to end-to-end traffic. It is expected for mobile network operators or Internet service

providers to deploy SFCs in a geographically centralized manner, such as a data center where service function chains can be easily managed and configured. However, in a Cloud-to-thing scenario, the configuration and management of the service function chain is significantly more complex, because careful orchestration strategies are required to discover proper service instances and connect them across multiple networks operated with different resource management and network policies.

The deployment of SFC in a cloud-to-thing scenario is even more complex if service instances are developed following a serverless architecture, in which case orchestration strategies to discover proper service instances need to consider the network function semantics, namely its intermittent availability in the network.

In the next section, we outline requirements for a solution that would realize those use cases and address some of the gaps outlined in [I-D.liu-can-gap-reqs], with Section 7 presenting our initial design on how to address those requirements through a shim layer atop IPv6.

5. Requirements

The following requirements for a routing on service addresses (ROSA) solution (referred to as 'solution' for short) have been identified from our use cases in the previous section

One commonality of all use cases is the communication with a 'service', realized at one or more network locations as equivalent 'service instances'. Associating the service to an 'owner' is key to avoid services being announced by fake entities, thus misdirecting the client's traffic, while obfuscating the purpose of communication (e.g., leaked through the specific name of a service) but also any possible policy to select one over another service instance may want to be kept private; this is likely the case across all of our use cases. Hence, any solution

REQ1: MUST provide means to associate service instances with a single service address.

- **(a)** MUST provide secure association of service address to service owner.
- **(b)** SHOULD provide means to obfuscate the purpose of communication to intermediary network elements.
- **(c)** MAY provide means to obfuscate the constraint parameters used for selecting specific service instances.

Across all our use cases, the knowledge of where service instances (realizing specific services) reside within the network, i.e., possibly at different network locations, is crucial for the communication to happen, at least for the ROSA domain with which the service has an association with. Such knowledge may be created by a service management platform, e.g., as part of the overall service deployment, and thus may not be initiated by the deployed service instance itself, such as in the example of Section 3.4. Furthermore, service deployment may be delegated to service or CDN platforms, e.g., in the CDN, AR/VR and video distribution examples of Section 3.1, Section 3.6 or Section 3.5, respectively, albeit with linkages needed to the service routing capabilities of ROSA. Crucially, however, is that a solution ought to use proactive pushing of suitable reachability information to service instances into the ROSA system, i.e., pursuing a routing-based approach, allowing for faster availability of information to make suitable decisions on which service instance to choose among those available. Hence, any solution

REQ2: MUST provide means to announce route(s) to specific instances realizing a specific service address, thus enabling service equivalence for this set of service instances.

- **(a)** MUST provide scalable means to route announcements.
- **(b)** MUST announce routes within a ROSA domain.
- **(c)** SHOULD provide means to delegate route announcement.
- **(d)** SHOULD provide means to announce routes at other than the network attachment point realizing the announced service address.
- **(e)** MUST allow for removing service instances that are intermittently availab,e, i.e., revoking their service announcement after a defined timeframe.

A client application may not just invoke services within a single ROSA domain. While associating with different ROSA domain may be possible, clients may simply invoke services through their existing ROSA domain, e.g., for utilizing helper services in examples like Section 3.4, expecting the service transaction to be realized regardless. The same goes for invoking services that may reside in the public Internet, without requiring an explicit awareness of the client to which ROSA domain (or the public Internet) to direct the invocation. Thus, any solution

REQ3: MUST provide means to interconnect ROSA islands.

- **(a)** MUST allow for announcing services across ROSA domains.

(b) MUST allow for announcing services outside ROSA domains.

Use cases like Section 3.4 but also video delivery ones such as Section 3.5 and Section 3.6 or the selection of an appropriate UPF (user plane functions) within a cellular sub-system in Section 3.2, may want to constrain the selection of 'suitable' service instances through service-specific constraints, such as the computing load (on the deployed service instances or their host platforms), service-level latency, but also, e.g., HW or SW, capabilities. This may also be the case for multi-homed deployments (see Section 3.3), where constraints on the multi-connectivity of the service instance may constrain the suitability for specific clients. Thus any solution

REQ4: Solution MUST provide constraint-based routing capability.

(a) MUST provide means to announce routing constraints associated with specific service instances and their realizing networking, computing and storaged resources.

(b) SHOULD allow for providing constraints in the service (address) announcement.

The work in [OnOff2022] has shown the potential gains in making runtime decisions for every incoming service transaction, where transaction lenths may be as small as single (application-level) requests. For use cases such as Section 3.5 or Section 3.6, this may lead to significant smoothening of the request completition latency, i.e., reducing the latency variance, thus enabling a better, smoother experience at the client. However, the specific mechanism may vary and, more importantly, may be highly service-specific, with solutions such as [CArDS2022] providing a simple weighted round robin, while other methods may rely on regular (service) metric reporting. Thus any solution

REQ5: MUST provide an instance selection at ROSA domain ingress nodes only.

(a) MUST allow for signalling selection mechanism and necessary input parameters for selection to the ROSA domain ingress nodes.

Explicit resolution steps, such as those in DNS, GSLB, or Alto, suffer from the need for an explicit control plane exchange. This causes additional latency before the data transfer to the chosen service instance may start. In-band data, i.e., the inclusion of application-level data in the control messages, is not supported due to the layering of such solutions at the application level itself. It is desirable, however, to already allow for the exchange of application data, including that needed for establishing secure

connections, in the process that determines the most suitable service instance to further reduce any latency for completing a given application-level service transaction. Thus any solution

REQ6: MUST provide an in-band data transfer capability in the process of determining the suitable service instance for any following data transfer within the same service transaction.

While video delivery use cases like Section 3.5 or Section 3.6 may exhibit short lived transactions of just one (service-level) request, due to the replicated nature of the video content in each service instance, service transactions may last many requests after the initial one has been sent. Ephemeral state may be created during this transaction, which would require that a change of the (initial) service instance during a transaction would share such ephemeral state with any new service instance being used. While service platforms, like K8S, provide such ability through 'shared data layer' capabilities, those are often limited to single site deployments. Any support across sites would incur additional costs or even possibly latencies for such state sharing, thus often leading to completing an ongoing service transaction with the service instance that has been originally been used (note that a service instance in ROSA may use internal methods for serving incoming requests across which state sharing would be applied - from a ROSA perspective, however, only one service instance is being used). We call the capability to retain an initial selection of a service instance for the length of a service transaction 'affinity'. Thus, any solution

REQ7: MUST adhere to the affinity towards the service instance chosen in the initial service request of the service transaction, thus directing all subsequent service transaction requests to the same instance.

All of our use cases are likely being deployed over existing network infrastructure, which makes a consideration to use its existing solutions in any realization of ROSA very important. Specifically, any solution

REQ8: Solution SHOULD use IPv6 for the routing and forwarding of service and affinity requests.

(a) Solution MAY use IPv4 for the routing and forwarding of service and affinity requests.

Most of our use cases, specifically Section 3.4 but also our video delivery examples, may be realized in inherently mobile settings with clients moving about for their experience. While mobile IP solutions exist, the service initialization in ROSA needs to be

equally supported in order to allow for invoking ROSA services on the move. Thus, any solution

REQ9: SHOULD support in-request mobility for a ROSA client.

Mobility of clients, but also varying loads in scenarios of no client mobility, may also lead to situations where moving on ongoing service transaction to another service instance may be beneficial, termed 'transaction mobility'. In other words, service instances may be replaced mid-transaction, in order to ensure the service level agreement. This may happen if, for instance, the local node where the service instance was initially installed is running out of resources, or its accessibility is reduced (which be periodically). Thus, any solution

REQ10: SHOULD support transaction mobility, i.e., changing service instances during an ongoing service transaction.

With most service transactions likely being encrypted for privacy and security reasons, supporting the appropriate transport layer methods is crucial in all our scenarios in Section 3, which is achieved by ROSA being positioned as a L3.5 solution, as presented in Section 7. While work in [OnOff2022] has shown that small service transactions in scenarios like Section 3.5 or Section 3.6 may be beneficial for significantly reducing the service-level latency, the challenge lies in initiating suitable transport layer security associations with frequently changing service instances. Pre-shared certificates may address this to allow for 0-RTT handshakes being realized but come with well-known forward secrecy problems. Thus, any solution

REQ11: SHOULD support TLS 0-RTT handshakes without the need for pre-shared certificates.

We envision the ROSA layer in ROSA endpoints to be transparently integrated in the operation of transport protocols, and thus applications, by provuding suitable interfaces to accessing the ROSA services of a specific ROSA domain. Thus, any solution

REQ12: SHOULD be transparent to applications in order to ensure a smooth deployment.

6. Expected Benefits

We expect the following benefits to be realized through the ROSA design, discussed in the next section. We here refer to

investigations in several research works by reference for more detail on the findings:

* Dynamicity: Decisions to select one out of possibly many service instance can be highly dynamic, done per service transactions, including for single packet ones. This is enabled by the move from an explicit off-path resolution step to an in-band, on-path mapping of a service address to its realizing service instance. Such dynamicity aims at improving transaction completion latency and variance, balancing load across service instances, as well as possibly deal with temporary network conditions. The work in [OnOff2022] evaluates the impact of performing traffic steering decisions at the level of ROSA rather than at application level.

* Service-specifity: The constraints for selecting a suitable service instance should not be limited to network metrics like delay or bandwidth. Instead, services should be able to define service-specific constraints, allowing for either multi-optimality routing or realising request-level and possibly compute-aware request scheduling for selecting one of possibly several service endpoints. The mechanism in [CArDS2022] outlines an example for such steering decisions, taking into account service-specific compute information. However, to avoid embedding full path information into the service routing itself, the consideration of service-specific constraints should be limited to the selection of service instances, while the forwarding of transaction data (in the form of subsequent affinity requests) solely follows the routing policies defined by the underlay network.

* Reduce dependency on DNS: Current service routing utilises a DNS-based approach, thereby requiring explicit off-path operations before being able to utilise a specific service. We aim at reducing this dependency on the DNS. The work in [OnOff2022] outlines the possible impact of reducing the use of the DNS, while also evaluating the capabilities enabled in flexible (small affinity) traffic steering under the constraint of a given latency budget.

* Efficiently support higher degree of service distribution: Typical application or also L4-level solutions, such as GSLB, QUIC-based indirection, and others, lead effectively to egress hopping when performed in a multi-site deployment scenario in that the client request will be routed first to an egress as defined either through the DNS resolution or the indirection through a central server, from which the request is now resolved or redirected to the most appropriate DC site. In deployments with a high degree of distribution across many (e.g., smaller edge computing) sites, this leads to inefficiencies through path

stretch and additional signalling that will increase the request completion time. Instead, direct or on-path solutions such as ROSA are expected to lead to a more direct traffic towards the site where the service will eventually be executed, while also allowing for application data to be already carried as part of the service instance selection process, thus keeping the request completion time close to its optimum in respect to the best site being used for execution of the request.

* Bring application namespace closer to communication relations: Reid et al [Namespaces2022] outline insights into the aspects and pain points experienced when deploying existing intra-DC service platforms in multi-site settings, i.e., networked over the Internet. The main takeaway in is the lacking protocol support for routing requests of microservices that would allow for mapping application onto network address spaces without the need for explicitly managed mapping and gateway services. While this results in management overhead and thus costs, efficiency of such additional mapping and gateway services is also seen as a hinderance in scenarios with highly dynamic relationships between distributed microservices, an observation aligned with the findings in [OnOff2022]. The use cases presented in Section 3, among others, exhibit the degrees of distribution in which relationship management (through explicit mapping and/or gatewaying) may become complex and a possible hinderance for service deployment and suitable performance.

7. ROSA Design

This section outlines the design of a shim layer relying upon IPv6 to provide routing on service addresses (ROSA). It first outlines the system overview, before outlining the interfaces to the IP layer (Section 7.2 and applications in ROSA endpoints (Section 7.3), followed with the various operational methods of ROSA in terms of forwarding operations (Section 7.4), traffic steering methods (Section 7.5), and interconnection (Section 7.6).

7.1. System Overview

Figure 1 illustrates a ROSA domain, interconnected to other ROSA-supporting domains via the public Internet through the Service Address Gateway (SAG), where a ROSA domain may span one or more IPv6 underlay domain. Section 7.6 provides more detail on how to achieve interconnection between ROSA domains.

ROSA is positioned as a shim overlay atop IPv6, using Extension headers that carry the suitable information for routing and forwarding the ROSA service requests, unlike [I-D.eip-arch] which

proposes to include extension processing directly into the transport network.

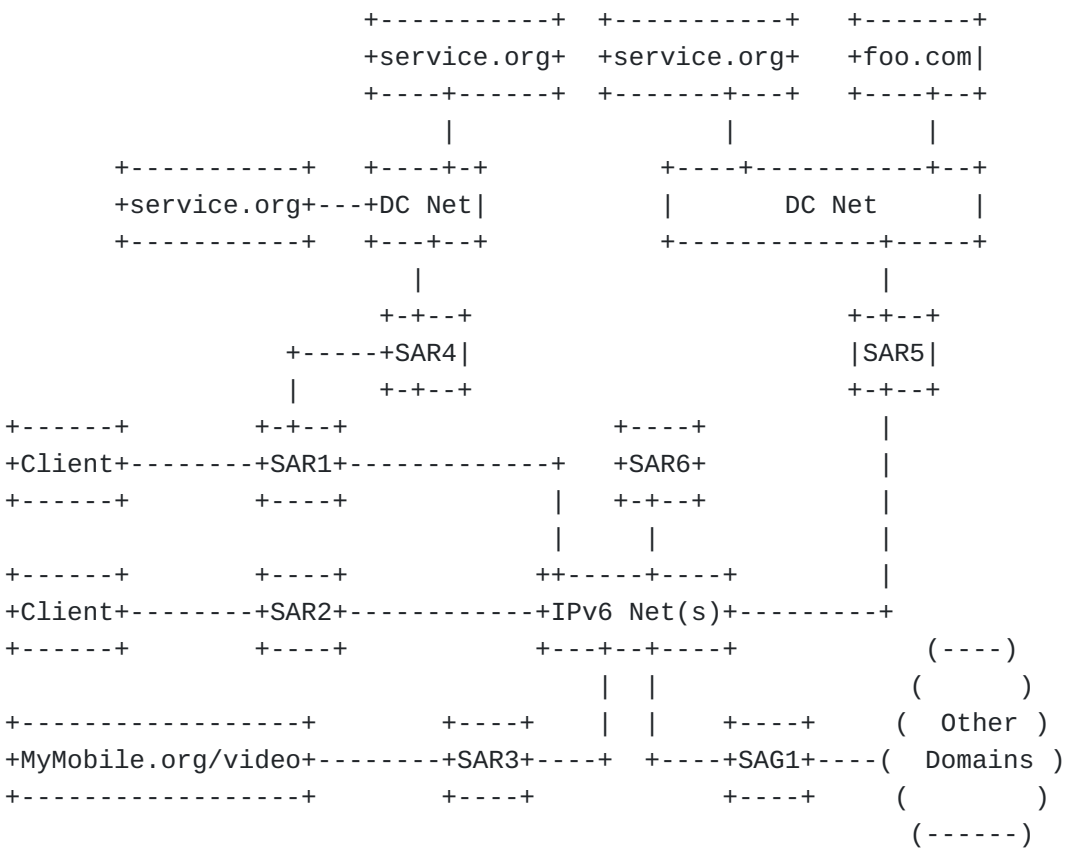

Figure 1: ROSA System Overview

ROSA endpoints start with discovering their ingress Service Address Router (SAR), e.g., through DHCP extensions or through utilizing the Session Management Function (SMF) in 5G networks [TS23501]. An endpoint may discover several ingress SARs for different categories of services, each SAR being part of, e.g., a category-specific ROSA overlay, which in turn may be governed by different routing policies and differ in deployment (size and capacity). The category discovery mechanism may be subject to specific deployments of ROSA and thus is likely outside the scope of this document at this point.

Services are realized by service instances, possibly at different network locations. Those instances expose their availability to serve requests through announcing the service address of their service to their ingress SAR, which in turn distributes suitable

ROSA routing state across the SARs in its domain. The lacking tie of service addresses to the network topology, and thus the lacking possibility to aggregate relationships of service addresses to routing locators, poses a scalability challenge (specifically to address Req 2.a in Section 7.4) However, the routing tables in ROSA are bounded by the number of services explicitly announcing their service to ingress SARs, while utilizing explicit interconnection (see Section 7.6) to other ROSA domains or the Internet for any service requested in the domain that has not previously been announced.

To invoke a service, a ROSA client sends an initial request, addressed to a service, to its ingress SAR, which in turn steers the request (possibly via other SARs) to one of possibly many service instances. See Section 7.4 for the required SAR-local forwarding operations and end-to-end message exchange and Section 7.3 for the needed changes to ROSA clients. Conversely, non-ROSA services may continue to be invoked using existing means for service routing, such as DNS, GSLB, Alto and others.

We refer to initial requests as 'service requests'. If an overall service transaction creates ephemeral state, the client may send additional requests to the service instance chosen in the (preceding) service request; we refer to those as 'affinity requests'. With this, routing service requests (over the ROSA network) can be positioned as on-path service discovery (winth in-band data), contrasted against explicit, often off-path solutions such as the DNS.

In order to support transactions across different service instances, e.g., within a single DC, a sessionID may be used, as suggested in [SOI2020]. Unlike [SOI2020], discovery does not include mapping abstract service classes onto specific service addresses, avoiding semantic knowledge to exist in the ROSA shim layer for doing so.

With the above, we can outline the following design principles that guide the development for the solutions described next:

* Service addresses have unique meaning only in the overlay network.

* Service instance IP addresses have meaning only in the underlay networks, over which the ROSA domain operates.

* SARs map service addresses to the IP addresses for the next hop to send the service request to, finally directed to the service instance IP address.

* Within the underlay network, service instance IP addresses have both locator and identifier semantics.

* A service address within a ROSA domain carries both identifier and locator semantics to other nodes within that domain but also other ROSA domains (through the interconnection methods shown in Section 7.6).

* Affinity requests directly utilize the underlay networks, based on the relationships build during the service request handling phase.

We can recognize similarities of these principles with those outlined for the Locator Identifier Separation Protocol (LISP) in [I-D.ietf-lisp-introduction] albeit extended with using direct IP communication for longer service transactions.

7.2. Message Types

Apart from affinity requests, which utilize standard IPv6 packet exchange between the client and the service instance selected through the initial service request, ROSA introduces three new message types, shown in Figure 2.

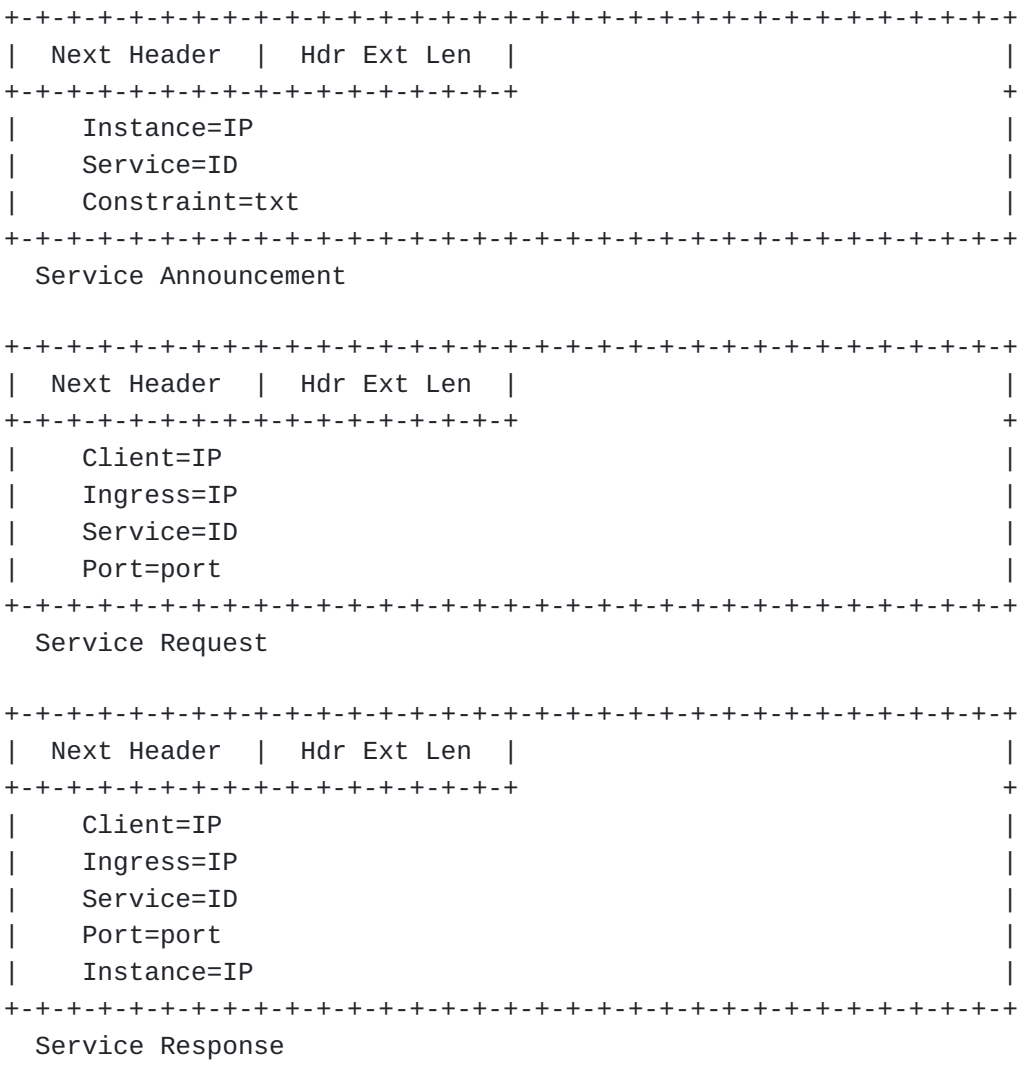

Figure 2: ROSA message types

Given the overlay nature of ROSA, clients, SARs, and service instances are destinations in the IPv6 underlay of the network domains that the overlay spans across. This is unlike approaches such as [I-D.huang-service-aware-network-framework], which place the service address into the destination address of the respective IPv6 header field, although [I-D.ma-intarea-encapsulation-of-san-header] also foresees the encapsulation into the IPv6 EH, as suggested here.

Istead, we propose to use the destination option EH [RFC8200], where Figure 2 shows the options carried, proposed here as using a TLV format for the extension header with Concise Binary Object Representation (CBOR) [RFC8949] being studied as an alternative. The EH entries shown are populated at the client and service instance, while read at traversing SARs.

A service address is encoded through a hierarchical naming scheme, e.g., using [RFC8609]. Here, service addresses consist of components, mapping existing naming hierarchies in the Internet onto those over which to forward packets, illustrated in the forwarding information base (FIB) of Figure 3 as illustrative URLs. With components treated as binary objects, the hierarchical structure allows for prefix-based grouping of addresses, reducing routing table size, while the explicit structure allows for efficient hash-based lookup during forwarding operations, unlike IP addresses which require either log(n) radix tree search software or expensive TCAM hardware solutions.

Note that other encoding approaches could be used, such as hashing the service name at the ROSA endpoint or assigning a service address through a mapping system, such as the DNS, but this would require either additional methods, e.g., for hash conflict management or name-address mapping management, which lead to more complexity.

With the service announcement message, a service instance signals towards its ingress SAR its ability to serve requests for a specific service address. Section 7.5 outlines the use of this message in routing or scheduling-based traffic steering methods.

The service request message is originally sent by a client to its ingress SAR, which in turn uses the service address provided in the extension header to forward the request, while the selected service instance provides its own IP locator as an extension header entry in the service response. In addition to the service address, suitable port information is being provided (through upper layer protocols), allowing to associate future affinity requests with their initial service requests.

The next section describes the SAR-local forwarding operations and the end-to-end message exchange that uses the extension header information for traversing the ROSA network, while Section 7.6 outlines the handling of service addresses that have not been previously announced within the client-local ROSA domain.

7.3. Changes to Clients to Support ROSA

Within endpoints, the ROSA functionality is realized as a shim layer atop IPv6 and below transport protocols. For this, endpoints need the following adjustments to support ROSA:

* Adapting network layer interface: Introducing service addresses requires changes to the current network interface for discovering the ingress SAR and issuing service requests as well as maintaining affinity to a particular service instance, i.e. mapping a service instance IP address to the initial service

address. As one possible choice, a new address type (e.g., ADDR_SA) could be introduced at the socket interface, provided during socket creation and assigning the service address to the returned handle, while utilizing socket options to assign constraints to receiving sockets, utilized in the announcement of the service address. Alternatively, supporting service addresses could be integrated with efforts such as [POSTSOCK2017] to redefine the transport interface towards applications. Any OS-level client changes, as required by introducing new sockets, could be avoided by relying on, e.g., UDP-based, encapsulation of client traffic to the ingress SAR albeit with the drawback of needing to maintain client-state at the ingress SAR.

* Transport protocol integration: We see our design aligned with existing transport protocols, like TCP or QUIC, albeit with changes required to utilize the aforementioned new address type. For the application (protocol), the opening and closing of a transport connection would then signal the affinity to a specific instance, where the semantic of the 'connection' changes from an IP locator to a service address associated to that specific service instance. With this, a new service transaction is started, akin to a fresh DNS resolution with IP-level exchange.

* Changes to application protocols: The most notable change for application protocols, like HTTP, would be in bypassing the DNS for resolving service names, using instead the aforementioned different (service) socket type. These adaptions are, however, entirely internal to the protocol implementation. Given the ROSA deployment alongside existing IP protocols, those changes to clients can happen gradually or driven through (e.g., edge SW) platforms.

7.4. SAR Forwarding Engine

The SAR operations are typical for an EH-based IPv6 forwarding node: an incoming service request or response is delivered to the SAR forwarding engine, parsing the EH for relevant information for the forwarding decision, followed by a lookup on previously announced service addresses, and ending with the forwarding action.

Figure 3 shows a schematic overview of the forwarding engine with the forwarding information base (FIB) and the next hop information base (NHIB) as main data structures. The NHIB is managed through a routing protocol, see Section 7.5, with entries leading to announced services.

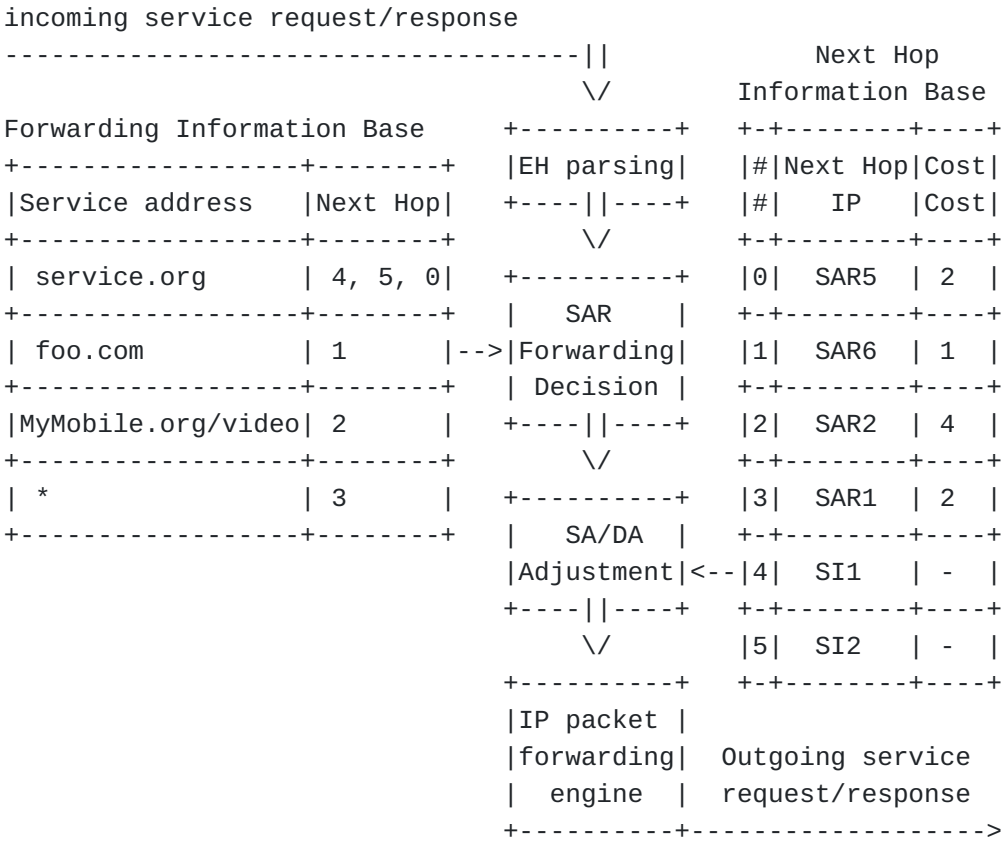

Figure 3: SAR forwarding engine model

The FIB is dynamically populated by service announcements via the intyer-SAR routing protocol, with the FIB including only one (ROSA next hop) entry into the NHIB when using routing-based methods (rows 0 to 3 in Figure 3), described in Section 7.5.2. Scheduling-based solutions (see Section 7.5.1), however, may yield several dynamically created entries into the NHIB (items 0, 4 and 5 in Figure 3, where SI1 and SI2 represent the IPv6 address announced by the respective service instances) as well as additional information needed for the scheduling decision; those dynamic NHIB entries directly identify service instances locations (or their egress as in item 0) and only exist at ingress SARs towards ROSA clients.

As stated in Section 1.2, we expect the number of forwarding entries to be limited by the explicit relations service providers may have with their ROSA provider. In other words, we do not expect the FIB to include ALL possible service names but those explicitly announcing their service (and being authorized by the ROSA provider doing so). In our use cases of Section 3, those services may be very limited in numbers, particularly if we foresee dedicated ROSA providers to aim at realizing those use cases.

For a service request, a hash-based service address lookup (using the Service EH entry) is performed, leading to next hop (NH) information for the IPv6 destination address to forward to (the final destination address at the last hop SAR will be the instance serving the service request).

Forwarding the response utilizes the Client and Ingress EH fields, where the latter is used by the service instance's ingress SAR to forward the response to the client ingress SAR, while the former is used to eventually deliver the response to the client by the client's ingress SAR, ensuring proper firewall traversal of the response back to the client. We have shown in prototype realizations of ROSA that the operations in Figure 3 can be performed using eBPF [eBPF] extensions to Linux SW routers, while [SarNet2021] showed the possibility a realizing a similar design using P4-based platforms.

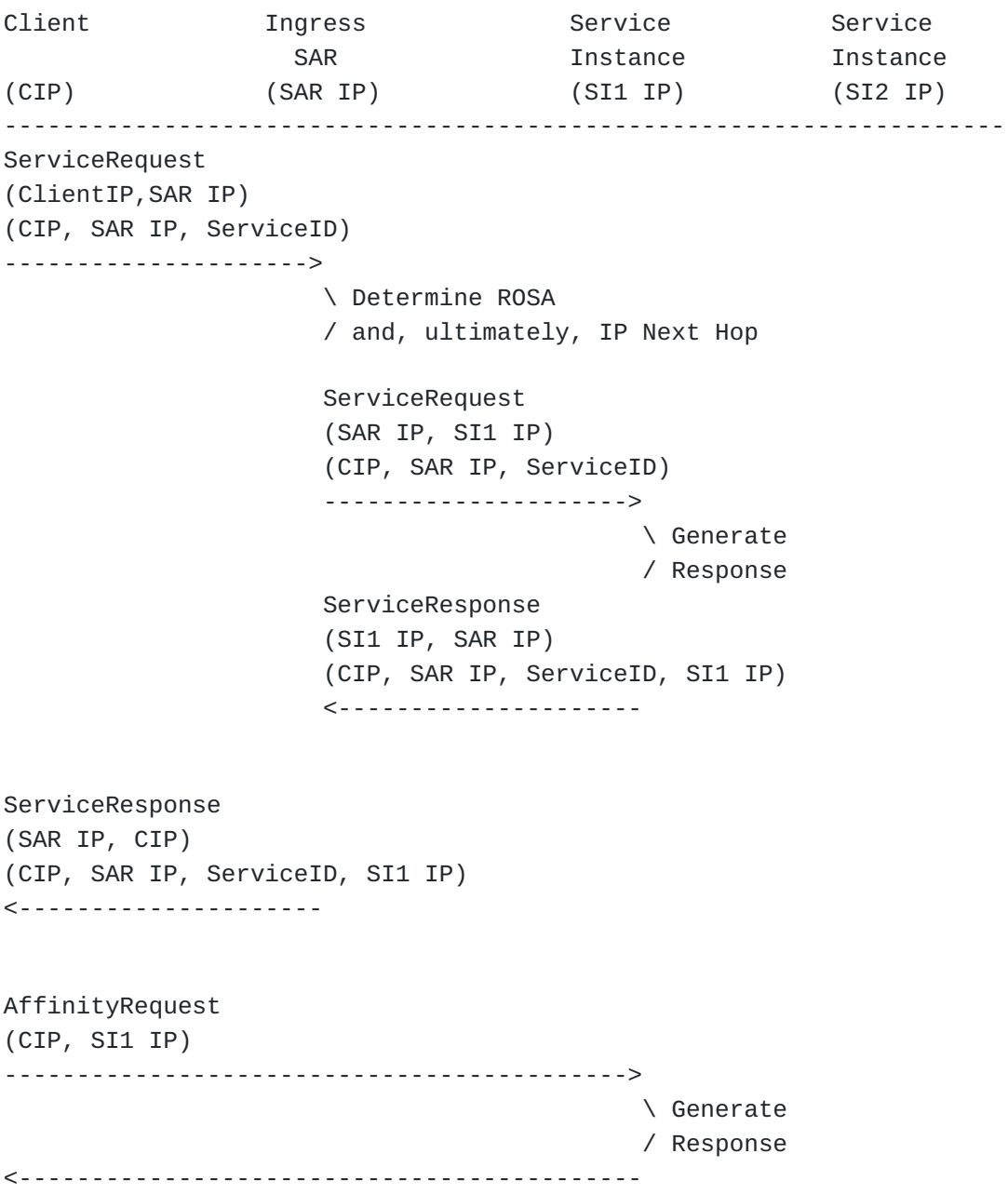

Figure 4: ROSA message exchanges

Figure 4 shows the resulting end-to-end message exchange, using the aforementioned SAR-local forwarding decisions. We here show the IP source and destination addresses in the first brackets and the extension header information in the second bracket.

We can recognize two key aspects. First, the SA/DA re-writing happens at the SARs, using the EH-provided information on service address, initial ingress SAR and client IP locators, as described above. Second, the selection of the service instance is signalled

back to the client through the additional Instance EH field, which is used for sending subsequent (affinity) requests via the IPv6 network. As noted in the figure, when using transport layer security, the service request and response will relate to the security handshake, thereby being rather small in size, while the likely larger HTTP transaction is sent in affinity requests. As discussed in Section 12, 0-RTT handshakes may result in transactions being performed in service request/response exchanges only.

7.5. Traffic Steering

Traffic steering in ROSA is applied to service requests for selecting the service instance that may serve the request, while affinity requests use existing IPv6 routing and any policies constraining traffic steering in this part of the overall system. At receiving service endpoints, service provisioning platforms may use additional methods to schedule incoming service requests to suitable resources with the ingress point to the service provisioning platform being the service endpoint for ROSA.

In the following, we outline two approaches for traffic steering. The first uses ingress-based scheduling decisions to steer traffic to one of the possible service instances for a given service address. The second follows a routing-based model, determining a single destination for a given service address using a routing protocol, similar to what is suggested in [I-D.huang-service-aware-network-framework].

We envision that some services may be steered through scheduling methods, while others use routing approaches. The indication which one to apply may be derived from the number of next hop entries for a service address. In Figure 3, service.org uses a scheduling method (with instances connected to SAR5 being exposed as a single instance to ROSA, using DC-internal methods for scheduling incoming requests), while the other services are routed via SARs.

We furthermore envision an interface to exist between the ROSA provider and the underlying network provider, exchanging routing policy relation information. The richness of this interface depends on the specific business relation between both providers, i.e., the ROSA and the network provider. In integrated settings, where ROSA and network provider may belong to the same commercial entity, this interface may provide rich routing policy relation information, such as network latency and bandwidth information, which in turn may be used in the ROSA traffic steering decisions. In other, more disintegrated deployments, the information may entirely be limited to SLA-level information with no specific runtime information exchanged between both providers. The exact nature of this interface remains for further study.

Important here is that traffic steering is limited to a single ROSA domain, i.e., traffic steering is not provided across instances of the same service in different ROSA domains; traffic will always be steered to (ROSA) domain-local instances only.

7.5.1. Ingress Request Scheduling

Traffic steering through explicit request scheduling follows an approach similar to application- or transport-level solutions, such as GSLB [GSLB], DNS over HTTPS [RFC8484], HTTP indirection [RFC7231] or QUIC-LB [I-D.ietf-quic-load-balancers]: Traffic is routed to an indirection point which directs the traffic towards one of several possible destinations.

In ROSA, this indirection point is the client's ingress SAR. However, unlike application or transport methods, scheduling is realized in-band when forwarding service requests in the ingress SAR, i.e., the original request is forwarded directly (not returned with indirection information upon which the client will act), while adhering to the affinity of a transaction by routing subsequent requests in a transaction using the instance's IP address. Scheduling commences to a possibly different instance with the start of a new transaction.

For this, the ingress SAR's NHIB needs to hold information to ALL announced service instances for a service address. Furthermore, any required information, e.g., capabilities or metric information, that is used for the scheduling decision is signalled via the service announcement, with (frequent) updates to existing announcements possible. Announcements for services following a scheduling- rather than a routing-based steering approach carry suitably encoded information in the Constraint field of the announcement's EH, leading to announcements forwarded to client-facing ingress SARs without NHIB entries stored in intermediary SARs.

In addition, a scheduling decision needs to be realized in the SAR forwarding decision step of Figure 3. This may require additional information to be maintained, such as instance-specific state, further increasing the additional NHIB data to be maintained. Examples for scheduling decisions are:

* Random selection of one of the service instances for a given service address, not requiring any additional state information per service address. Announcing the service instance is required once.

* Round robin, i.e., cycling through service instance choices with every incoming service request, requiring to keep an internal

counter for the current position in the NHIB for the service address. Announcing the service instance is only required once.

* Capability-based round robin: Cycle through service instances in weighted round robin fashion with the weight (as additional information in each NHIB entry) representing a capability, e.g., number of (normalized) compute resources committed to a service instance. Announcing the service instance requires an update when capabilities change (e.g., during re-orchestration). Weights could be expressed as numerals, limiting the needed semantic exposure of service provider knowledge and thereby supporting the possible separation of service and communication network provider. The solution in [CArDS2022] realises a compute-aware selection through such decision.

* Metric-based selection: Select service instance with lowest or highest reported metric, such as load, requiring to keep additional metric information per service instance entry in the NHIB. Frequent signalling of the metric is required to keep this information updated.

Although each method yields specific performance benefits, e.g., reduced latency or smooth load distribution, [OnOff2022] outlines simulation-based insights into benefits for realising the compute-aware solution of [CArDS2022] in ROSA.

7.5.2. Routing Across Multiple SARs

In order to send a service request to the `best' service instance (among all announced ones) using a routing-based approach, we build NHIB routing entries by disseminating a service instance's announcement for a given service address S, arriving at its ingress SAR. This distribution may be realized via a routing protocol or a central routing controller or a hybrid solution.

If no particular constraint is given in the announcement's EH Constraint field, shortest path will be realized as a default policy for selecting the `best' instance, routing any client's request to S the nearest service instance available.

Alternatively, selecting a service instance may use service-specific policies (encoded in the Constraint field of the EH, with the specific encoding details being left for future work). Here, multiple constraints may be used, with [Multi2020] providing a framework to determine optimal paths for such cases, while also conventional traffic engineering methods may be used.

Through utilizing the work in [Multi2020], a number of multi-criteria examples can be modelled through a dominant path model, relying on a partial order only, as long as isotonicity is observed.

Typical examples here are widest-shortest path or shortest-widest routing (see [Multi2020]), which allow for performance metrics such as capacity, load, rate of requests, and others. However, metrics such as failure rate or request completion time cannot directly be captured and need formulation as a max metric. Furthermore, metrics may not be isotonic, with Section 3.4 of [Multi2020] supporting those cases through computing a set of dominant attributes according to the largest reduction. [Multi2020] furthermore shows that non-restarting or restarting vectoring protocols may be used to compute dominant paths and to distribute the routing state throughout the network.

However, the framework in [Multi2020] is limited to unicast vectoring protocols, while the routing problem in ROSA requires selecting the 'best' path to the 'best' instance, i.e., as an anycast routing problem. To capture this, [Multi2020] could be extended through introducing a (anycast) virtual node, placed at the end of a logical path that extends from each service instance to the virtual node. Selecting the best path (over the announced attributes of each service instance) to the virtual node will now select the best service instance (over which to reach the virtual node in the logically extended topology).

Alternatively, ROSA routing may rely on methods for anycast routing, but formulated for service instead of anycast addresses. For instance, AnyOpt [AnyOpt2021] uses a measurement-based approach to predict the best (in terms of latency) anycast (i.e. service) instance for a particular client. Alternatively, approaches using regular expressions may be extended towards spanning a set of destinations rather than a single one. Realizations in a routing controller would likely improve on convergence time compared to a distributed vector protocol; an aspect for further work to explore.

7.6. Interconnection

There are two cases for interconnection: access to (i) non-ROSA services in the public Internet and (ii) ROSA services not domain-locally announced but existing in other domains.

For both cases, we utilize a reserved wildcard service address '*' that points to a default route for any service address that is not being advertised in the local domain. This default route is the service address gateway (see Figure 1), ultimately receiving the service request to the locally unknown service.

Upon arriving at the SAG, it searches its local routing table for any information. If none is found, it consults the DNS to retrieve an IP address where the service is hosted; those mappings could be

cached for improving future requests or being pre-populated for popular services.

For case (i), the resolution returns a server's IP address to which the SAG sends the service request with its own IP address as source address. The service response is routed back via the SAG, which in turn uses the Ingress EH information to return the response to the client via its ingress SAR.

For case (ii), the IP address would be that of the SAG of the ROSA domain in which the service is hosted. For this, a domain-local service instance would have exposed its service, e.g., Mobile.com/video Figure 1, by registering its domain-local SAG IP address with the mapping service. To suitably forward the request, the SAG adds its own IP address as the value to an additional SAG label into the extension header. At the destination SAG, the service address information, extracted from the extension header, is used to forward the service request based on ROSA mechanisms. For the service response, the destination SAG uses the SAG entry in the EH to return the response to the originating ROSA domain's SAG, which in turn uses the Ingress information of the EH to return the response via the ingress to the client.

Given the EH deployment issues pointed out in [SHIM2014], a UDP-based encapsulation may overcome the observed issues, not relying on the EH being properly observed during the traversal over the public Internet. Furthermore, while Figure 1 shows the SAG as an independent component, we foresee deployments in existing PoPs. This would allow combining provisioning through frontloaded PoP-based services and ROSA services. Any service not explicitly announced in the ROSA system would lead to being routed to the PoP-based SAG, which may use any locally deployed services before forwarding the request to the public Internet.

8. Extensions to Base ROSA Capabilities

ROSA, as defined in Section 7, can be extended to address various capabilities useful for specific or across a number of use cases. The following provides a list of those possible extended capabilities. At this stage, we would expect those capabilities to be defined in more detail in separate drafts, complementing the ROSA 'base' specification, as defined in this current draft.

8.1. Supporting Different Namespace Encodings

Although most of our examples assume the use of URL-based service addresses, encoded using [RFC8609], supporting other, e.g., corporate service, namespaces may be desired. [RFC8609] generally supports this and could thus be used.

As briefly alluded to in Section 7.2, other encodings to that following [RFC8609] may be used, focussing on different ways to represent a service address differently, including linking it to the service name used at the application level.

One such encoding may be that of a unique service address per service name, with the linkage between both provided through the DNS. Here, the client sends an initial DNS query with the URL of the purported service or application. Instead of requesting a resolution to a locator, however, is the request for mapping between the URL and the service address of ROSA, where the service address has been assigned as part of the domain name registration (which may be done after initial registration of the domain name for backward compatibility). Service addresses here may be simply encoded as numerals, where uniqueness is achieved through linking to the domain name registration and thus DNS mapping. Encoding in the respective EH header field (see Section 7.2) would be shorter and thus more efficient, still achieving the desired uniqueness in the SAR forwarding process to avoid ambiguity in forwarding decisions. The drawback is the need for the additional DNS mapping step (albeit only required once per application, where the service address could be stored persistently for later use), while also the additional DNS mapping will need standardization (likely in the form of a new DNS record).

Another possible encoding, without the aforementioned explicit DNS mapping step, could be to explicitly hash the service name into a service address at the ROSA endpoint, operating on those hash values for service announcement and requests. Due to the large service namespace, hash conflicts may, however, occur, which needs resolving at the SAR (which may operate on a service address for a service request for a different, but same hashed, service address of an announcement service). Further study is needed into the probability for such hash conflicts and possible mitigation methods for such conflicts.

If the use of different encoding methods beyond [RFC8609] was to be considered, appropriate modifications to the EH fields need to be done to signal the used encoding method for the service address.

8.2. Supporting Multi-Homing of Service Instances

Multi-homed service instances may benefit from path-aware routing decisions after mapping service addresses to service instance addresses. To that end, service instances would need to advertise multiple instance IPs as part of their service announcement.

The optimal path may differ while a client communicates with a service instance; this is in particular likely for mobile clients.

This provides some complication for affinity requests; in such a case, the service instance IP is no longer sufficient to identify a service instance, merely to locate a particular path.

Multi-homing issues in connection with aircrafts also extend to Unmanned Aerial Systems (UAS). Rather than focusing on passenger experience, multi-homing over commercial off-the-shelf (COTS) communications modules such as 5G or IEEE 802.11 provides command, control and communications (C3) capabilities to Unmanned Aerial Vehicles (UAV; drones). Despite the difference in focus, the technical challenges in maintaining connection quality are largely equivalent.

Multi-homing thus either requires an undesirable further resolution step from a service instance identifier to a (optimal path) locator. Alternatively, ROSA message types may be extended to include a distinct service instance identifier and service instance locator identifiers, i.e., IP addresses, which provides sufficient information for SARs to map to specific and changing locators, while retaining the affinity to the service instance by identifier.

8.3. Supporting 0-RTT TLS

TBD Dirk

8.4. Supporting Transaction Mobility

When it comes to the transaction mobility in which the serving service instance needs to be shifted to another selected alternative instance, the ROSA service address could provide a good starting as an location-independent ID. Other than TCP for which the client and server have to maintain strict machine state, UDP-based protocol could be extended with the service address to be treated as the connection ID rather than the traditional 4-tuple including the host destination address when the server does not have to maintain session state. The chief gain here is the service connection could remain intact when the serving service instance has been switched over at ROSA level (routing plane).

As part of the ability to switch over from one service instance, ROSA may explicitly support that mobility in that the choice of the (new) service instance is explicitly made within the service-specific traffic steering method. For this, ROSA may introduce a separate message alongside the service request message (see see Section 7.2), which not only allows for the ingress SAR to perform the same routing policy as if it was sent through a new service request message, but may also include application-specific context data to facilitate the needed application state transfer from the original service instance to the new one. Here, the in-band

capability of a ROSA request is being used to carry this context data as part of the new ROSA message.

8.5. Supporting Service Function Chaining

Service Function Chaining (SFC) [RFC7665] allows for chaining the execution of services at L2 or L3 level, targeting scenarios such as carrier-grade NAT and others. The work in [RFC8677] extends the chaining onto the name level, using service names to identify the individual services of the chain, even allowing combinations of name and L2/L3-based chains.

Although [RFC8677] is tied into a realization of the SFF (service function forwarder) using a path-based forwarding approach, the concept of name-based SFCs can equally be realized utilizing ROSA.

8.6. Supporting Privacy-Compliant Communication

The exposure of service-related information in the ROSA EHs may be seen as a privacy issue, particularly when utilizing the service name as the basis for the service address formulation. Although Section 12 outlines the possible use of service categories (instead of finer-grained service names) as input into the service address formulation, it is also desirable to protect the privacy of fine-grained service address information, should the specific ROSA deployment make use of them.

Beyond using encryption methods to protect the ROSA EH information, such privacy methods could also include the obfuscation of client and transit information as well, thus moving into the space of routing privacy, as outlined for instance in [I-D.ip-address-privacy-considerations]

9. Prototype-based Insights

To come before IETF116 with description of planned demo to demonstrate some of the benefits outlined in Section 6.

10. Open Issues

There are a number of open issues with the following list providing a non-exhaustive list of examples:

1. A ROSA control plane is required for handling aspects of ingress SAR discovery and signalling of service instance announcements to the ROSA network, either to ingress SARs only (for services utilizing traffic scheduling mode) or across all domain-internal SARs. Here, the signalling for achieving interconnections, based on the methods outlined in Section 7.6, is also required to be specified.

2 Possible segmentation of ROSA service request and responses need handling.

3 Prototypical but also possibly simulation-based insights into benefits are desirable to motivate the adoption of ROSA.

11. Conclusions

This draft outlined a methods for service-specific traffic steering through an IPv6 EH-based shim overlay, allowing for routing on service addresses as either ingress-based instance selection or through multi-SAR routing methods.

As next steps, we plan on extending various aspects of the ROSA operations, specifically to address the open issues listed in the previous section, e.g., control plane aspects such as SAR discovery and routing protocol, support for service request segmentation, and others. We expect that aspects for a ROSA control plane, e.g., to signal suitable traffic steering parameters to the ingress SARs or to establish multi-SAR routing state, are captured in separate works.

We furthermore have plans on bringing an eBPF-based prototypical realization of the forwarding behaviour in Section 7.4 to future IETF events, e.g., for a hackathon participation to showcase ROSA-based applications in a test setup.

12. Security Considerations

Aligned with security considerations in existing service provisioning systems, we address aspects related to authenticity, i.e., preventing fake service announcements, confidentiality, both in securing relationship as well as payload information, and operational integrity.

- Announcement security: A key exchange between service and network provider may be used to secure the service announcement for ensuring an authorized announcement of services. Self-certifying identifiers could be used for this purpose

- Relationship security: Using service addresses at the routing layer poses not just a privacy but possibly also a net neutrality problem, allowing for non-ROSA elements to discriminate against specific service addresses. Similar to [I-D.per-app-networking-considerations], service addresses could reflect service categories, not services themselves. Service endpoints to those category-level services can use information in the secured payload (e.g., the URL in an HTTP-based service invocation) to direct the traffic accordingly. The downside of

such model is a possible convergence towards a PoP-like model of service provisioning, since exposing an entire service category naturally requires provisioning many possible services under that category, likely favouring large-scale providers over smaller ones; an imbalance that ROSA intends to change, not favour. Work on identity privacy in ILNP [ILNP2021] has shown that ephemeral identifiers may increase the private nature of the communication relation; a direction that needs further exploration in the context of our work. Also, the service address in the extension header could be encrypted, based on a key exchange during the SAR discovery. However, the impact of such mechanism would need further study.

*Transport-level security: Given the often sensitive nature of service requests, payload security is key. We adopt techniques used in TLSV1.3 [RFC8446], providing a 1-RTT handshake for communication between formerly untrusted parties. While the initial 'Client Hello' is sent as a service request, the subsequent communication uses the topological address of the responding server in an affinity request. Using pre-shared keys may allow for communication between trusted client and service instances, e.g., where the client is provided by the service authority and preconfigured with a pre-shared key. This results in a 0-RTT handshake with the 'Client Hello' including the initial service data, encrypted with the pre-shared key. This comes with known forward-secrecy issues and should be avoided in networks with untrusted intermediary nodes. Alternatively, the service's public key could encrypt the initial security handshake, akin to the solutions proposed for Encrypted Client Hello (ECH), using the DNS for obtaining the public key.

*Bandwidth DoS: We assume network provider level mechanisms to restrict traffic injected both by the service provider and client, including for the number of service advertisements in order to control the routing traffic.

*Denying routing service: A SAR could maliciously deny forwarding of client requests, which is no different from denying IP packet forwarding. In both cases, we assume an existing commercial relationship that avoids such situation.

13. Privacy Considerations

The exposure of service-related information in the ROSA EHs may be seen as a privacy issue, particularly when utilizing the service name as the basis for the service address formulation. As discussed in Section 8.6, extensions to the base ROSA capabilities may address this issue to ensure the privacy of the clients' communication relations in ROSA deployments, where needed.

14. IANA Considerations

This draft does not request any IANA action.

15. Change Log

1 Restructured introduction to improve readability and argumentation for this draft

2 Addressing IETF115 comments in various parts of the draft, e.g., introduction, analysis (relation to other technologies), traffic steering (relation to anycast) etc

3 Added six new use cases (mobile applications - Section 3.4, chunk retrieval - Section 3.5, AR/VR - Section 3.6, Cloud-to-Thing - Section 3.7, Metaverse - Section 3.8, and popularity-based services - Section 3.9)

4 Added separate analysis section, as derived from use cases (Section 4)

5 Revised and linked requirements to use cases through additional text (Section 5)

6 Discussed possible benefits from applying ROSA in identified use cases (Section 6)

7 Revised ROSA messages figure (Figure 2)

8 Added section on possible extended capabilities to 'base' ROSA (Section 8), including multi-homing support, namespace support.

9 Added and maintaining open issues (Section 10)

10 Added missing sections, like conclusions (Section 11) and privacy considerations (Section 13)

11 Added Jens Finkhaeuser, Daniel Huang, and Paulo Mendes as co-authors.

16. Acknowledgements

Many thanks go to Mohamed Boucadair, Tommy Pauly, Joel Halpern, Daniel Huang, and Russ White for their comments to the text to clarify several aspects of the motiviation for and technical details of ROSA.

17. Informative References

[AnyOpt2021]
Zhang, Z., April, T., Chandrasekaran, B., Choffnes, D., Maggs, B. M., Shen, H., Sitaraman, R. K., Yang, X., Zhang, X., and T. Sen, "AnyOpt: predicting and optimizing IP Anycast performance", Paper ACM SIGCOMM, 2021.

[BBF] ""Control and User Plane Separation for a disaggregated BNG"", Technical Report-459 Broadband Forum (BBF), 2020.

[CArDS2022] Khandaker, K., Trossen, D., Khalili, R., Despotovic, Z., Hecker, A., and G. Carle, "CArDS:Dealing a New Hand in Reducing Service Request Completion Times", Paper IFIP Networking, 2022.

[CV19]
Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J., Vallina-Rodriguez, N., Hohlfeld, O., and G. Smaragdakis, "A Year in Lockdown: How the Waves of COVID-19 Impact Internet Traffic", Paper Communications of ACM 64, 7 (2021), 101-108, 2021.

[eBPF] "What is eBPF?", Technical Report eBPF Foundation, 2022, <https://ebpf.io/what-is-ebpf/>.

[EI2021]
Cidon, I., Culler, D., Estrin, D., Katz-Bassett, E., Krishnamurthy, A., McCauley, M., McKeown, N., Panda, A., Ratnasamy, S., Rexford, J., Schapira, M., Shenker, S., Stoica, I., Tennenhouse, D., Vahdat, A., Zegura, E., Balakrishnan, H., and S. Banerjee, "Revitalizing the public internet by making it extensible", Paper ACM Computer Communication Review, Vol. 51. 18-24. Issue 2, 2021.

[Gini] "Gini Coefficient", Technical Report Wikipedia, 2022, <https://en.wikipedia.org/wiki/Gini_coefficient>.

[GSLB] "What is GSLB?", Technical Report Efficient IP, 2022, <https://www.efficientip.com/what-is-gslb/>.

[HHI] "Herfindahl-Hirschman index", Technical Report Wikipedia, 2022, <https://en.wikipedia.org/wiki/Herfindahl-Hirschman_index>.

[Huston2021] Huston, G., "Internet Centrality and its Impact on Routing", Technical Report IETF side meeting on 'service routing and addressing', 2021, <https://github.com/danielkinguk/sarah/blob/main/conferences/ietf-112/materials/Huston-2021-11-10-centrality.pdf>.

[I-D.eip-arch]
Salsano, S., ElBakoury, H., and D. Lopez, "Extensible In-band Processing (EIP) Architecture and Framework", Work in Progress, Internet-Draft, draft-eip-arch-01, 16 December 2022, <https://www.ietf.org/archive/id/draft-eip-arch-01.txt>.

[I-D.huang-service-aware-network-framework] Huang, D., Tan, B., and D. Yang, "Service Aware Network Framework", Work in Progress, Internet-Draft, draft-huang-service-aware-network-framework-01, 22 November 2022, <https://www.ietf.org/archive/id/draft-huang-service-aware-network-framework-01.txt>.

[I-D.ietf-lisp-introduction] Cabellos-Aparicio, A. and D. Saucez, "An Architectural Introduction to the Locator/ID Separation Protocol (LISP)", Work in Progress, Internet-Draft, draft-ietf-lisp-introduction-15, 20 September 2021, <https://www.ietf.org/archive/id/draft-ietf-lisp-introduction-15.txt>.

[I-D.ietf-quic-load-balancers] Duke, M., Banks, N., and C. Huitema, "QUIC-LB: Generating Routable QUIC Connection IDs", Work in Progress, Internet-Draft, draft-ietf-quic-load-balancers-15, 24 October 2022, <https://www.ietf.org/archive/id/draft-ietf-quic-load-balancers-15.txt>.

[I-D.ip-address-privacy-considerations] Finkel, M., Lassey, B., Iannone, L., and B. Chen, "IP Address Privacy Considerations", Work in Progress, Internet-Draft, draft-ip-address-privacy-considerations-03, 10 January 2022, <https://www.ietf.org/archive/id/draft-ip-address-privacy-considerations-03.txt>.

[I-D.jennings-moq-quicr-arch] Jennings, C. and S. Nandakumar, "QuicR - Media Delivery Protocol over QUIC", Work in Progress, Internet-Draft, draft-jennings-moq-quicr-arch-01, 11 July 2022, <https://www.ietf.org/archive/id/draft-jennings-moq-quicr-arch-01.txt>.

[I-D.liu-can-gap-reqs] Liu, P., Jiang, T., Eardley, P., Trossen, D., Li, C., and D. Huang, "Computing-Aware Networking (CAN) Gap Analysis and Requirements", Work in Progress, Internet-Draft, draft-liu-can-gap-reqs-00, 23 October 2022, <https://www.ietf.org/archive/id/draft-liu-can-gap-reqs-00.txt>.

[I-D.liu-can-ps-usecases]
Liu, P., Eardley, P., Trossen, D., Boucadair, M., Contreras, L. M., Li, C., and Y. Li, "Computing-Aware

Networking (CAN) Problem Statement and Use Cases", Work in Progress, Internet-Draft, draft-liu-can-ps-usecases-00, 23 October 2022, <https://www.ietf.org/archive/id/draft-liu-can-ps-usecases-00.txt>.

[I-D.ma-intarea-encapsulation-of-san-header] Ma, L., Zhao, D., Zhou, F., and D. Yang, "Encapsulation of SAN Header", Work in Progress, Internet-Draft, draft-ma-intarea-encapsulation-of-san-header-00, 23 October 2022, <https://www.ietf.org/archive/id/draft-ma-intarea-encapsulation-of-san-header-00.txt>.

[I-D.nottingham-avoiding-internet-centralization] Nottingham, M., "Internet Consolidation: What can Standards Efforts Do?", Work in Progress, Internet-Draft, draft-nottingham-avoiding-internet-centralization-07, 17 January 2023, <https://www.ietf.org/archive/id/draft-nottingham-avoiding-internet-centralization-07.txt>.

[I-D.per-app-networking-considerations] Colitti, L. and T. Pauly, "Per-Application Networking Considerations", Work in Progress, Internet-Draft, draft-per-app-networking-considerations-00, 15 November 2020, <https://www.ietf.org/archive/id/draft-per-app-networking-considerations-00.txt>.

[I-D.sarathchandra-coin-appcentres] Trossen, D., Sarathchandra, C., and M. Boniface, "In-Network Computing for App-Centric Micro-Services", Work in Progress, Internet-Draft, draft-sarathchandra-coin-appcentres-04, 26 January 2021, <https://www.ietf.org/archive/id/draft-sarathchandra-coin-appcentres-04.txt>.

[I-D.wadhwa-rtgwg-bng-cups] Wadhwa, S., Shinde, R., Newton, J., Hoffman, R., Muley, P., and S. Pani, "Architecture for Control and User Plane Separation on BNG", Work in Progress, Internet-Draft, draft-wadhwa-rtgwg-bng-cups-03, 11 March 2019, <https://www.ietf.org/archive/id/draft-wadhwa-rtgwg-bng-cups-03.txt>.

[ILNP2021] Yanagida, R., Bhatti, S., and G. Haywood, "End-to-end privacy for identity and location with IP", Paper 2nd Workshop on New Internetworking Protocols, Architecture and Algorithms, 29th IEEE International Conference on Network Protocols, 2021.

[ISOC2022] "Internet Centralization", Technical Report ISOC Dashboard, 2022, <https://pulse.internetsociety.org/centralization>.

[MCN] ""Metro Compute Networking: Use Cases and High Level Requirements"", Technical Report-466 Broadband Forum (BBF), 2021.

[Multi2020] Ferreira, M. A. and J. L. Sobrinho, "Routing on Multi Optimality Criteria", Paper ACM SIGCOMM, 2020.

[Namespaces2022] Reid, A., Eardley, P., and D. Kutscher, "Namespaces, Security, and Network Addresses", Paper ACM SIGCOMM workshop on Future of Internet Routing and Addressing (FIRA), 2022.

[OnOff2022] Khandaker, K., Trossen, D., Yang, J., Despotovic, Z., and G. Carle, "On-path vs Off-path Traffic Steering, That Is The Question", Paper ACM SIGCOMM workshop on Future of Internet Routing and Addressing (FIRA), 2022.

[POSTSOCK2017] Kuehlewind, M., Trammell, B., and C. Perkins, "Post sockets: Towards an evolvable network transport interface", Paper IFIP Networking Conference (IFIP Networking) and Workshops, 2017.

[RFC6770] Bertrand, G., Ed., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, DOI 10.17487/RFC6770, November 2012, <https://www.rfc-editor.org/info/rfc6770>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/

RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <https://www.rfc-editor.org/info/rfc8609>.

[RFC8677] Trossen, D., Purkayastha, D., and A. Rahman, "Name-Based Service Function Forwarder (nSFF) Component within a Service Function Chaining (SFC) Framework", RFC 8677, DOI 10.17487/RFC8677, November 2019, <https://www.rfc-editor.org/info/rfc8677>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

[RFC9213] Ludin, S., Nottingham, M., and Y. Wu, "Targeted HTTP Cache Control", RFC 9213, DOI 10.17487/RFC9213, June 2022, <https://www.rfc-editor.org/info/rfc9213>.

[SarNet2021] Glebke, R., Trossen, D., Kunze, I., Lou, Z., Rueth, J., Stoffers, M., and K. Wehrle, "Service-based Forwarding via Programmable Dataplanes", Paper 1st Intl Workshop on

Semantic Addressing and Routing for Future Networks, 2021.

[SHIM2014] Naderi, H. and B. Carpenter, "Putting SHIM6 into practice", Paper 2014 Australasian Telecommunication Networks and Applications Conference (ATNAC), 2014.

[SOI2020] Jiang, S., Li, G., and B. Carpenter, "A New Approach to a Service Oriented Internet Protocol", Paper IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020.

[SVA] ""Optimizing Video Delivery With The Open Caching Network"", Technical Report Streaming Video Alliance, 2018.

[TIES2021] Giotsas, V., Kerola, S., Majkowski, M., Odinstov, P., Sitnicki, J., Chung, T., Levin, D., Mislove, A., Wood, C. A., Sullivan, N., Fayed, M., and L. Bauer, "The Ties that un-Bind: Decoupling IP from web services and sockets for robust addressing agility at CDN-scale", Paper ACM SIGCOMM, 2021.

[TS23501] "System architecture for the 5G System (5GS); Stage 2 (Release 16)", Technical Report 3GPP TS 23.501 V16.11.0 (2021-12), 2021, <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144>.

Authors' Addresses

Dirk Trossen
Huawei Technologies
80992 Munich
Germany

Email: dirk.trossen@huawei.com
URI: https://www.dirk-trossen.de

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 1st floor
28050 Madrid
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://lmcontreras.com/

Jens Finkhaeuser
Interpeer gUG
86926 Greifenberg
Germany

Email: ietf@interpeer.io
URI: https://interpeer.io/

Paulo Mendes
Airbus
82024 Taufkirchen
Germany

Email: paulo.mendes@airbus.com
URI: http://www.airbus.com