Service Oriented Networking

David Griffin, Miguel Rio
University College London, UK

Pieter Simoens, Piet Smet
iMinds/University of Ghent, Belgium

Frederik Vandeputte, Luc Vermoesen
Alcatel-Lucent Bell NV, Belgium

Dariusz Bursztynowski
Orange, Poland

Folker Schamel
Spinor, Germany

***Abstract* — This paper introduces a new paradigm for *service oriented networking* being developed in the FUSION project[1]. Despite recent proposals in the area of information centric networking, a similar treatment of services – where networked software functions, rather than content, are dynamically deployed, replicated and invoked – has received little attention by the network research community to date. Our approach provides the mechanisms required to deploy a replicated service instance in the network and to route client requests to the closest instance in an efficient manner. We address the main issues that such a paradigm raises including load balancing, resource registration, domain monitoring and inter-domain orchestration. We also present preliminary evaluation results of current work.**

Keywords—Service oriented networking, orchestration, routing, services, placement algorithms, anycast.

I. Introduction and Motivation

The Internet was originally conceived as a data communications network to interconnect end-hosts: user terminals and servers. The focus was on delivering data between end points in the most efficient manner. All data was treated in the same way: as the payload of packets addressed for delivery to a specific end-point. In recent years, since the development of the world-wide web, the majority of traffic on the Internet originates from users retrieving content. The observation that many users were downloading the same content led to the development of content delivery/distribution networks (CDNs). CDNs cache content closer to the users to reduce inter-provider traffic, and improve users' quality of experience by reducing server congestion through load balancing requests over multiple content replicas. In a content-centric world, communications are no longer based around interconnecting end-points, but are concerned with *what* is to be retrieved rather than *where* it is located. CDNs achieve this by building overlays on top of the network layer but recent research has taken matters a stage further by routing requests for named content to caches which are dynamically maintained by the network nodes themselves, rather than having predefined locations of the content, pushed a priori based on predicted demand. Such an approach represents a basic paradigm shift for the Internet.

[1] The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) in the FUSION (Future Service Oriented Networks) project under grant agreement n° 318205.

Although information centric networking (ICN) has received enormous attention recently [1][2][3], the approach, like CDNs, is limited to non-interactive content where identical copies are distributed to multiple consumers. Cloud computing on the other hand has been developed to deliver applications and services in a scalable manner to cope with elasticity of demand for computing resources, exploiting economies of scale in multi-tenancy data centres (DCs). Just as with CDN services in the past, cloud resources are now being deployed in local ISPs and other distributed network locations, presenting a much more complex problem than can be solved with generalised resource assignment algorithms in individual DCs or cloud infrastructures with only a handful of geographical locations. While new networking paradigms for intra-data-centre communications have been developed to facilitate the distribution of data-processing intensive applications over a flexible number of computing devices within the same DC [4], these techniques and technologies are limited to specific DCs and services and have not been rolled out to the wider-area Internet. Cloud federation has received a lot of attention in recent years [6] but techniques have been aimed at improving scalability of cloud-based applications and they do not address the problem of fine grained localisation of processing nodes in the network between the federated clouds.

We envisage a situation where large numbers of service nodes – which we term *execution zones* – are distributed throughout the Internet: in access points close to the users; co-located with routers within an ISP's network; in local data-centres owned and operated by ISPs; and in traditional data-centres and service farms operated by cloud and service providers. Given this rich set of resources, a set of functions and algorithms is required to enable services to be flexibly and efficiently deployed to optimise the placement of service instances according to the performance requirements of the application, the location of its users and their demand patterns. At the network level a service-anycast capability is needed so that service instance selection can be optimised on the grounds of network metrics as well as server load.

Section II presents an overview of our service oriented networking architecture. Section III focuses on the service layer of the solution covering service orchestration and execution while section IV describes the network layer covering the routing aspects of service queries and invocations. Section V presents some initial results evaluating algorithms for service routing and load balancing based on a combination of network and server level metrics. Section VI discusses related work.

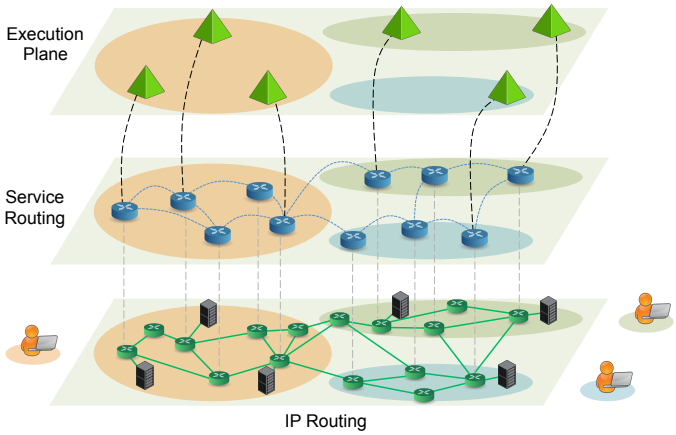

Fig. 1. FUSION Framework

II. ARCHITECTURE

The FUSION framework can be seen in Fig. 1. Functionality is divided into 3 layers. At the lower level IP routing forwards packets using traditional end-to-end protocols. At the top layer the execution plane consists of all the execution zones where the services' instances will run. In the middle the service router layer will forward request from clients to the appropriate service instances.

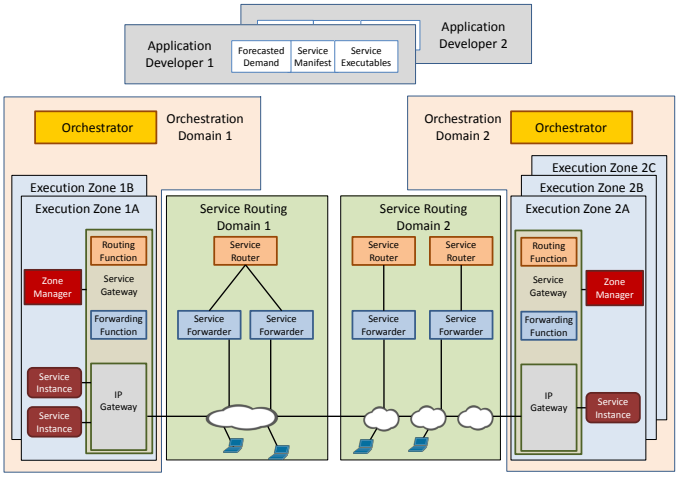

Fig. 2. FUSION architecture

The basic operation of the FUSION system is that orchestration domains – consisting of a potentially large number of geographically distributed execution zones – deploy services on behalf of application developers or service providers in one or more execution zones according to the expected demand by service users. Service routing domains, consisting of one or more service routers, are responsible for matching service requests referring to a service by service identifier (serviceID) to execution zones containing running instances of the requested service. Service routing is anycast in nature – the user simply requests a service and it is the responsibility of the service routing plane to find the "best" available instance for that request. Once a specific service instance in a specific execution zone has been selected for the user request, data plane communications take place in the data forwarding plane depicted by "IP Routing" in the lower layer of Fig. 1. Note that physical DCs are depicted in the lower IP routing layer as data-plane communications will be directly between users and service instances running in physical DCs, while the abstract representation of execution zones – a logical partition of a DC – are shown in the upper execution plane.

The main functional entities in the FUSION architecture are depicted in Fig. 2. The three main entities are the orchestrator, execution zone and service router.

The ***orchestrator*** manages its orchestration domain resources including execution zones and services which it manages on behalf of application developers (or service providers).

The ***execution zone*** is the logical representation of a collection of physical computational resources in a specific location, such as a DC. The orchestrator has an abstract view of an execution zone and the detailed internals are managed by a zone manager. The zone manager is responsible for managing service instances within its zone but under the instruction of the orchestrator. It will select the specific physical location including virtual machine (VM), machine and rack of individual service instances and interact with the local infrastructure management platform of the DC/cloud node for VM lifecycle management. The execution zone interacts with the communications infrastructure of the outside world through a service gateway.

The ***service router*** is responsible for maintaining and managing service routing information to create forwarding paths for queries/invocation requests from users and other service instances to be resolved or forwarded to execution zones containing available running instances of the specified serviceID.

III. SERVICE ORCHESTRATION AND EXECUTION

In this section, we elaborate on the orchestration and execution plane of the FUSION architecture, highlighting some of the design decisions and strategies for automatically managing demanding complex services across heterogeneous execution environments.

A. Design decisions

For many reasons including scalability, the overall orchestration and management of services and resources has been divided in two layers, namely a logically centralised domain orchestrator and a distributed set of independent execution zones, each managed by their own zone manager. Whereas a domain orchestrator has a high-level overview of all services that are registered and deployed within its domain, a zone manager is responsible for the lower-level details of mapping services onto the available resources and managing both the services and resources. The domain orchestrator has no direct control or view of the available resources and delegates this functionality to the zone managers. A domain orchestrator can also decide to dynamically deploy a zone manager in a new DC, effectively converting some of its resources into a new execution zone onto which FUSION services can be deployed.

For the design of an execution zone, we opted in the general case for an overlay approach, in which the zone and its zone manager are running on top of the existing DC management layer. The zone manager communicates with a DC abstraction layer for translating FUSION commands into DC specific commands for deployment and monitoring purposes as shown in Fig. 3. The FUSION Platform as a Service (PaaS) capabilities could also be integrated into the native DC management layer via FUSION-specific OpenStack plug-ins, for example, for more fine-grained control over services and resources.

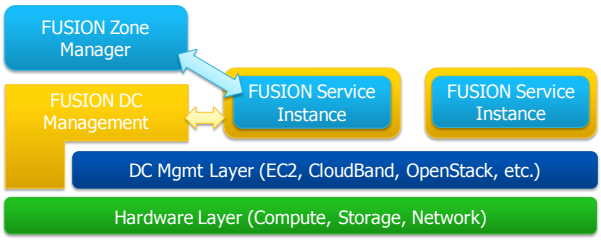

Fig. 3. Execution Zone Overlay Approach

B. *Composite services*

With the FUSION framework, we want to support composite services, which consist of a graph of connected service components. We make a clear distinction between a service graph, which is the graph at specification time, and a service instance graph, which is the graph at runtime, both of which can be static or dynamic in nature. Both require different approaches and mechanisms and allow for different optimisation opportunities. It is the responsibility of the FUSION domain orchestrator to deal with these complexities in an automated way when deploying new instances across one or more execution zones.

C. *Session slots*

Service instances typically can handle a number of requests in parallel, depending on the allocated resources and the service characteristics. For services with long-term sessions, each request will consume some resources for several minutes or hours. Service routers need to route to service instances that have available resources to process the new incoming request. To solve this issue in a scalable way we propose the concept of session slots. The core idea is that each service instance keeps track of its available session slots based on the available resources and exposes this information to the zone manager, which can inject this information via the service gateway into the service routing plane, which can use this information for routing an incoming service request to an instance that still has some slots available. Several advantages of this approach include light-weight service routing, a clear separation of resource allocation and routing, the possibility for hierarchical aggregation, service type neutrality, simplifying auto-scaling decisions and billing.

D. *Distributed service placement via evaluator services*

A fundamental problem that FUSION orchestration addresses is the scalable and optimal placement of (composite) services across a distributed set of zones, taking into account application-specific requirements and constraints. These requirements can be modelled by the affinity or anti-affinity degree of service components with respect to each other, the source and client end points and their execution environments. A simple example of this is depicted in Fig. 4 for a composite service consisting of two service components A and B that can be mapped onto one or more execution zones Z_i.

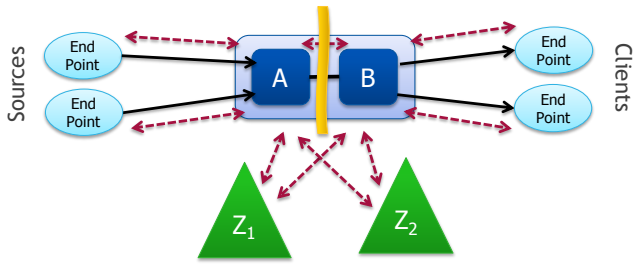

Fig. 4. Distributed placement problem

As a first-order scalable and flexible strategy, we address the problem with a five-step distributed approach using application-specific evaluator services, as depicted in Fig. 5.

1. The placement component of the domain orchestrator preselects a number of viable zones for each service component.
2. Next, each of the selected zones will be asked to return an offer with respect to deploying one or more instances of a service component in that zone.

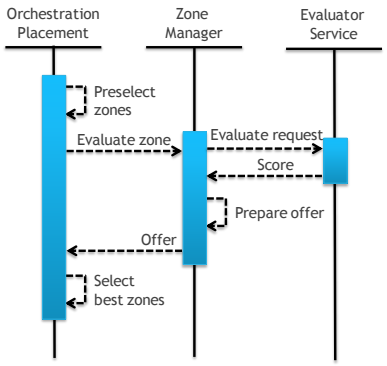

Fig. 5. Distributed placement strategy

3. The zone manager triggers application-specific evaluator services to provide a score regarding the deployment request within the zone. The score is a multi-dimensional metric that can be interpreted by the zone manager. The evaluator service is a plug-in service that allows for very fine-grained assessments regarding the specific capabilities of the zone whether the graphics processing unit (GPU) in that zone supports a specific shared model, for example.
4. The zone evaluates the score and returns an offer to the domain orchestrator.
5. The domain orchestrator then finds the optimal zones for each service component based on the received offers, policies, network information and overall application requirements.

IV. SERVICE ROUTING

To design our service routing framework we looked at several possibilities. Clean slate approaches were considered unrealistic for deployment in the short to medium term due to it being highly disruptive at a high cost to ISPs. The option of using the Domain Name System (DNS) as currently deployed has the disadvantage of not easily fitting the requirement for resolving serviceIDs without compromising the service naming scheme to fit existing DNS resource records. Even if new resource records are defined for FUSION-compatible serviceIDs and the client-DNS server protocol could remain intact, new functionality for resolving and forwarding queries is required for DNS servers to act as service routers and the benefits of retaining the existing DNS protocol would be very limited.

We concluded that an overlay routing solution would be the most appropriate since it is more easily deployable, can achieve all our requirements and by being able to run at the application layer will have much weaker requirements on memory usage. Hence the service routing layer accommodates the following three scenarios:

Firstly, a resolution option where the FUSION service routing plane resolves serviceID to a specific execution zone/service instance and returns one or more IP addresses to the client for subsequent selection and invocation. This corresponds to an enhanced anycast service where both network status and server performance characteristics are taken into account.

Secondly, an invocation option where the FUSION service routing plane forwards a client's invocation request to a selected execution zone and the service instance is invoked directly, with the service response being routed back via the service routing overlay. However, we do not foresee this option being used for either long-lived sessions or request-response sessions where the quantity of returned data is excessively large without establishing a separate data-plane communication channel through IP.

Finally, an option inspired by software defined networking (SDN) for the case when there are no running service instances and the service routing plane should invoke orchestration (or a zone manager) to deploy an instance on demand.

V. PRELIMINARY RESULTS

In traditional IP routing, the locator of the destination host is encoded in the IP packet and is used by routers to look-up the outgoing interface to forward the packet on. The FUSION service router plane, however, uses location-agnostic serviceIDs. Since this serviceID resolves to multiple instances, service routers must use statistical load-balancing to determine the outgoing interface.

Using simulated annealing, we explored the infinite solution space to construct the optimal load balancing matrix $R(i,j,s) \in [0,1]$. Given the user demand of node i for service s, $R(i,j,s)$ indicates which fraction of that demand is forwarded to the instances of service s running on node j. We assume that the number of instances is fixed, and that user demand from node i can be modelled as a Poisson process with parameter $\lambda(i,s)$ measured as requests per time unit. Our objective function minimizes the Round Trip Time (RTT) and takes into account the network latency between client and service instance, as well as the request queuing times at the instance.

As network latency is subject to change due to load caused by background traffic, FUSION requires monitoring of the network load and topology. Frequent monitoring provides service routers with more accurate measurements to load-balance requests, yet also contributes to the load caused by background traffic. Server load also varies over time and influences the request queuing times at the instance. To account for both network and server load, FUSION requires monitoring tools to gather the information which is used when allocating tasks. Also, a trade-off between frequent updates and increased network load must be made.

Embedding this mapping exactly in the service router forwarding tables requires the source identifier to be included, which would introduce unacceptable bloat of the routing tables. In a scalable solution, we can only set weights for statistical load-balancing.

We implemented two heuristics to determine the weights in each service router. In a first approach, we ranked for each client the running instances by increasing latency and greedily assigned demand to the closest instances, without exceeding the maximum server load. We compared this with an 'Equal Share' approach where the demand from a single node is equally spread over all servers.

Fig. 6 shows initial simulation results for sparse and dense network topologies. Each topology contains 20 routers, 5 clients to generate demand and 3 servers running services. The results are averaged over 50 topologies generated by the Waxman-Brite generator [5].

The simulation results show that for sparse network topologies, the greedy approach closely matches the optimal solution retrieved via Simulated Annealing. Sparse networks contain fewer paths and so traffic is concentrated onto fewer and longer paths. On the one hand, this introduces more difference in the latency from a client node to the different instance nodes. Even if the closest server is already heavily loaded, the gain in processing time by sending requests to more distant instances (with presumably less load) does not offset the additional network latency. This effect is clearly visible for the Equal Share approach.

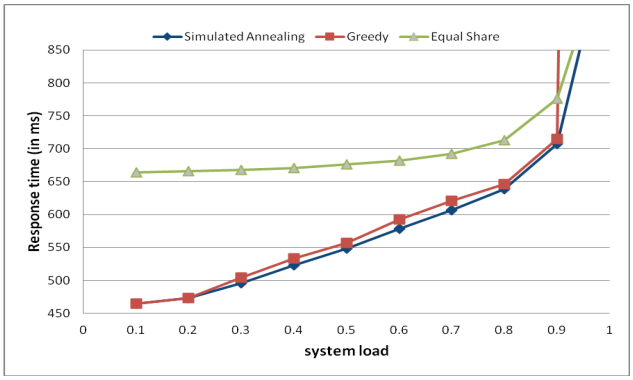

Fig. 6. Expected quality of service selection result for a sparse network

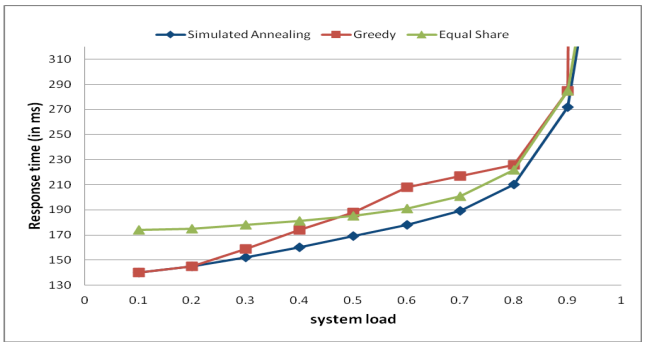

Fig. 7. Expected quality of service selection result for a dense network

In dense network topologies (Fig. 7) there are more paths available to load-balance between. The greedy approach utilises a minimal subset of servers while other servers remain idle, even when the network latency to these servers is not significantly higher. This concentration results in decreased response time compared to the better spread of load of the Equal Share algorithm.

Our initial results indicate the importance of optimal load-balancing by service routers, taking into account network topology, network metrics and server load.

VI. Related Work

There is a wide range of previous work upon which we build. We share with recent proposals on ICN (for example [1] [2]) the need for scalable named-based routing but adding crucial service centric features. We also build on currently deployed services like DNS [7] and the session initiation protocol (SIP) [8] by building a resolution and invocation protocol for services which could be seen as a significant extension to DNS functionality with SIP-like signalling features. One of the few prior studies on service oriented networking, SoCCeR [9] applies ant colony optimisation to CCNx for service routing on combined network/service metrics. In [10] the authors propose an object oriented approach to naming services in CCNx. XIA [11] uses a restricted directed acyclic graph for a common naming scheme for hosts, content and services.

Several distributed service management architectures such as IRMOS [12] or NGSON [13] have been proposed in recent years. In contrast to IRMOS, we target the wide area Internet and thus have to overcome the requirement of IRMOS for providing strict QoS guarantees as these can be reliably offered only in managed networks. We also augment the NGSON paradigm by providing a powerful service orchestration layer that is capable of allocating and load balancing service instances through dynamic cooperation with a distributed execution environment.

We also build on cloud technologies like Openstack [14] by providing additional PaaS capabilities. We will build upon light-weight isolation and virtualization strategies like Linux Containers (LXC) [15] OpenVz [16] and Docker [17] and will leverage TOSCA [18] for the description of services as well as OpenStack Heat for managing/orchestrating services within a FUSION domain.

VII. Conclusions

This paper introduces a novel service oriented networking paradigm that centres around service placement, orchestration and service routing. We believe that these new primitives will enrich the Internet architecture enabling new kinds of QoE demanding applications to be deployed efficiently across distributed execution environments. Current and future work include the detailing of the service routing plane to be able to direct requests taking into account a myriad of metrics which include network characteristics, server load and deployment costs.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard, Networking Named Content, in Proc. of ACM CoNEXT 2009, Rome, December 2009.

[2] P. Jokela, A. Zahemszky, C. Esteve, S. Arianfar and P. Nikander, LIPSIN: Line Speed Publish/Subscribe Inter-Networking, in Proc of ACM SIGCOMM'09, Barcelona, Spain, 2009.

[3] C, Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, H. Karl, Network of Information (NetInf) – An Information-Centric Networking Architecture, Elsevier Computer Communications journal, special issue on ICN, vol. 36, no. 7, pp. 721–735, April 2013.

[4] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: A Scalable Fault-tolerant layer 2 Data Center Network Fabric, in Proc. of ACM SIGCOMM'09, Barcelona, Spain, 2009.

[5] A. Medina, I. Matta, and J. Byers, On the Origin of Power Laws in Internet Topologies, ACM Computer Communications Review, April 2000.

[6] A. Celesti, F. Tusa, M. Villari, A. Puliafito, How To Enhance Cloud Architectures To Enable Cross-federation, in Proc. of 3rd IEEE International Conference on Cloud Computing (IEEE Cloud 2010), Miami, Florida, USA July 2010.

[7] P. Mockapetris, RFC 1035 - Domain Names: Implementation and specification, November 1987.

[8] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Petersoni, R. Sparks, M. Handley and E. Schooler, RFC 3261 - SIP: Session Initiation Protocol, June 2002.

[9] S. Shanbhag, N. Schwan, I. Rimac, and V. Varvello, Soccer: Services over content-centric routing, in Proc. of ACM SIGCOMM Workshop on Information-Centric Networking (ICN'11), Aug. 2011.

[10] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, and M. Varvello, Service-Centric Networking, in Proc. of IEEE ICC FutureNet IV workshop, Kyoto, Japan, June 2011.

[11] D. Han et al., XIA: Efficient Support for Evolvable Internetworking, in Proc. of 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI'12), San Jose, CA, April 2012.

[12] IRMOS Project Deliverable, D3.1.4, Updated Final version of IRMOS Overall Architecture, ICCS/NTUA and other partners, 2011.

[13] NGSON (Next Generation Service Oriented Network) – IEEE 1903, Standard for the Functional Architecture of Next Generation Service Overlay Networks, 2011.

[14] OpenStack, https://www.openstack.org/, 2014.

[15] Linux Containers, http://linuxcontainers.org/, 2013.

[16] Kolyshkin, K., Virtualization in Linux, http://download.openvz.org/doc/openvz-intro.pdf, 2006.

[17] Docker, https://www.docker.io/, 2013.

[18] OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) specification, version 1.0, http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs01/TOSCA-v1.0-cs01.pdf.