Future Service Oriented Networks

www.fusion-project.eu

Deliverable D4.1

Initial specification of service-centric routing protocols, forwarding & service-localisation algorithms.

Public report, Version 1.0, December 20 2013

Authors

UCL David Griffin, Miguel Rio, Raul Landa, Richard Clegg

ALUB Frederik Vandeputte

TPSA Dariusz Bursztynowski

SPINOR Folker Schamel

IMINDS Piet Smet, Pieter Simoens, Bart Dhoedt

Abstract This deliverable describes the initial functional requirements, interfaces and protocols as identified by the FUSION consortium for realizing a network layer that provides name-based selection of service instances. In addition, we report algorithms for network-driven selection of service instances and geographically-sensitive routing protocols providing delivery to the closest members of a group.

Keywords FUSION, service-aware networking, requirements, algorithms

Project funded by the European Union under the
Information and Communication Technologies FP7 Cooperation Programme
Grant Agreement number 318205

Revision history

Date	Editor	Status	Version	Changes
17/09/2013	Pieter Simoens	TOC	0.1	Initial ToC
15/10/2013	Pieter Simoens	Initial Version	0.2	Partial input
02/12/2013	Pieter Simoens	Draft	0.3	All sections updated
18/12/2013	Pieter Simoens	Stable	1.0	Ready for review
20/12/2013	Pieter Simoens	Final	1.1	Review comments included

EXECUTIVE SUMMARY

This document is a public deliverable of the "Future Service-Oriented Networks" (FUSION) FP7 project. It describes the functional requirements and interfaces of the service routing layer in the FUSION architecture. The key challenge for this layer is the routing and load-balancing of service requests to the best instance, given the existence of many different potential execution points and dynamically instantiated service instances.

This document has three main contributions: 1) the description of the functional requirements for the service routing layer, 2) the specification of the architecture blocks and their interfaces, and 3) the presentation of service-oriented algorithms for service selection and routing updates.

The realization of efficient name-based selection of services instances as the main primitive of the service routing layer breaks down into different functional aspects. These include appropriate naming and addressing schemes, efficient propagation of service availability between service routers, and selection strategies.

This deliverable provides an initial specification of architecture and interfaces for the service routing plane. A scan of existing approaches revealed many design considerations that are discussed in this document, such as centralized or distributed selection, the interworking between service instances and the network in the selection strategy, or the QoS guarantees. Also more subtle issues arising from distributing service instances across the network are being addressed, including security and evolvability. The protocols and interfaces between the components of the FUSION service routing layer are described at meta-level rather than in concrete implementation details. In year 2, these protocols and interfaces will be further refined and implemented.

The last part of this deliverable describes algorithms for instance selection and for delivery of routing table updates. The network-driven service selection algorithm maps user demand to a fixed set of service instances, taking into account both network latency and the size of the request queue for each service instance. The result of this mapping is then translated into routing tables for each service router for runtime load-balancing.

The proposed "N-casting" algorithm introduces geographic awareness in overlay topologies. The algorithm provides delivery to the n-closest members of a group and can be used for efficient propagation of service availability updates across zones.

The work presented in this deliverable serves as the basis for the remainder of the FUSION project. In the next year, the service routing layer architecture and protocols will be further extended. Selection algorithms will be further refined, in particular a more distributed approach will be studied. An update of the current deliverable is scheduled at the end of year 2.

TABLE OF CONTENTS

1. SCOPE OF THIS DELIVERABLE

This deliverable focuses on the service routing layer of the FUSION architecture. The principal functionality of this layer is the name-based selection between service instances running in multiple execution zones. Realizing this functionality requires the cooperation of many functional blocks such as naming, addressing, routing, monitoring and selection algorithms.

In the first year of the project, the FUSION consortium decided to adopt a breadth-first approach when tackling the multidimensional problem of service-oriented networking. This allowed to identify valuable results and experiences from past projects, and to set research priorities for the remainder of the project.

The architecture, interfaces and algorithms presented in this deliverable will be gradually expanded throughout the project's lifetime. For example, the algorithm presented for service selection is centralized: each service instance reports the size of its request queue to a central entity. As more advanced monitoring algorithms and service advertisement protocols have been defined, we will aim for more decentralized approaches.

2. FUNCTIONAL REQUIREMENTS FOR THE SERVICE ROUTING LAYER

Multiple instances of a service component (SCI) will be running in geographically dispersed execution zones. The service routing layer interconnects these execution zones and routes service requests to the most appropriate instance, taking into account monitoring information, service parameters and orchestration layer policies.

This selection primitive induces many functional requirements. The goal of this section is to set out all functional blocks of the service routing layer.

2.1 Naming and addressing

Multiple instances of the same service will be simultaneously running in one or more execution zones. The service routing layer must provide the functionality to name and address services and service instance components. Whereas a naming scheme is used for identification of the communicating entity (i.e. *which* service?), the addressing scheme refers to the actual network location of this entity (i.e. *where* is an instance running). Note that naming schemes should not necessarily be human readable.

The naming and addressing scheme used in the routing layer should decouple identification from location. All instances of the same service should be identified by a common service name; but each instance should also be individually addressable. In this respect, FUSION is related to the research initiatives on name-based networking (CCNx, NetInf, PSIRP…), but differentiates from this work by focusing on name-based selection of services instead of content. Especially for services, decoupling naming from addressing allows for late-binding: the address of the selected instance is not necessarily known in advance by the service component that wants to set-up a session.

2.1.1 Naming and identification

In FUSION, the following entities need to be identifiable:

- **service**: the service identifier (ID) refers to the functional entity that was registered by the service provider through the Orchestration Layer (see D3.1). Each instantiation of the service is identifiable by the same service ID. Today, domain names are used to identify webservices. This namespace is globally governed by IANA.
- **service instance:** likely, multiple instances of the same service will be deployed (in one or more execution zones). Stateful services may need to identify a particular service instance. Using the locator of that instance for this identification violates the requirement of separating naming from

addressing. For example, the instance may have been migrated to another host. After migration, the instance should keep the same instance identifier, but will have been assigned another IP address. Today, service component instances can be identified by their tuple (`hostIP`, `hostPort`), which actually reflects the location of that instance.

- **sessions**: a FUSION service session is the association between a client and sevice instance within which a set of one or more communication channels exists between a client and the service instance serving the request. Each service instance can typically handle multiple service requests (FUSION service sessions) in parallel. Endpoints must thus be able to relate messages to the correct FUSION session; e.g. when multiple messages are exchanged to negotiate session parameters. Note that a FUSION session should not be confused with an application session, which may comprise one or more independent FUSION service sessions.

  FUSION service session identifiers may also be used in service routers where, in combination with transport-related attributes, they may serve to configure forwarding rules (e.g. through OpenFlow protocol) or flow-based QoS for related communication channels, and at the endpoints for demultiplexing to the correct service instance. Today, sessions are only identified at the endpoints and transparent to the network (apart from middleboxes like NAT). If multiple threads are listening on the same port, demultiplexing is performed on the tuple (clientIP, clientPort, serverIP, serverPort).

Important design considerations for the FUSION namespace include [BARI2012]:

- ensuring globally unique names: namespace authority
- name integrity
- name structure: flat or hierarchical

No two services can share the same name and that name must refer to the same service across all orchestration and routing domains. For service instances, this constraint can be relaxed: instances may only be identifiable inside a single routing domain. Ensuring globally unique names is typically performed in two ways: either by relying on a namespace authority (typically hierarchical) or by relying on hashing.

One possibility is to leverage today's DNS system and naming hierarchies. In this case, services could be published under the domain name of the service developer. DNS is already used by CDN providers like Akamai for identification of content. Alternatively, the name allocation could be completely decentralized, allowing each service provider to construct its own name. To ensure uniqueness, one relies on the statistical properties of hashing functions.

Name integrity implies providing guarantees that the identified service instances are genuine. Some initiatives in Information-centric networking achieve this goal by making the name self-certifying: the name is a hash of the owner's public key. For example, NetInf names have two parts. The first part is the hash of the owner's public key, and the second part is a fixed ID. A digital signature stored in meta-data ensures content integrity. For FUSION, self-certifying names seems to be infeasible since there is no static content to be signed. Other measures will be needed to avoid spoofing of service names.

For the naming structure, one can discriminate between flat and hierarchical names. Hierarchical names have been proposed by Van Jacobson [VST09]. Hierarchies are based on organizational structures, folder structures, etc. However, this naming scheme introduces fixed interdependencies that can be avoided by flat namespaces. For example, with flat name spaces organizational changes do not necessarily reflect in name changes.

NetInf [SAIL2013] names are flat (i.e., have a flat part) when considering name comparison, e.g. when checking whether some content is available in the cache. However, NetInf names are not flat

(contain a routable part that allows for prefix aggregation) when considering routing of messages. Routing schemes can take into account the content of the authority part of the URI when considering how to route requests or do name resolution. For FUSION, we may adopt and extend the NetInf naming scheme, which is currently on standardization track [RFC6920].

The addressing scheme is used to route traffic to the physical attachment point of the host of the targeted entity. Given the huge number of service instances that will be running in the network, the addressing scheme is preferably based on IPv6. By using IPv6, the FUSION architecture can be deployed on the existing Internet infrastructure.

Table 1 lists the different entities present at the service routing layer.

Entity	Identifiable?	Addressable?
Service	yes	no
Service instance	yes	yes, based on IPv6
Session	yes	no

Table 1: Identifiable and routable entities for the service routing layer

2.1.2 Addressing

The network layer must provide **functionality for mapping names to addresses**. One service name must be mapped on multiple service instance names and/or service instance addresses. Since the address of a service instance is tightly coupled to its host, the address might change during the lifetime of this component.

Service instances running in Execution Zones might be migrated or shut down, and new service instances might be instantiated by the Zone Manager (e.g. to optimize resource usage) or the Orchestration Layer (see D3.1). Mobile clients may attach to another subnet (e.g. when changing from LTE to Wi-Fi). In this case, the service instance running on the client will be assigned a new IPv6 address.

2.1.3 Requirements

The requirements for naming and addressing in FUSION are listed below:

Req no.	Description	Level[1]
NA-1	Each service is identifiable by a globally unique name.	M
NA-2	Each service instance is identifiable by a globally unique name.	S
NA-3	The naming scheme for services and service instances is decoupled from the underlying hosting and network infrastructure.	M
NA-4	A global governing procedure for generation of service names and service instance names is defined by FUSION.	M

[1] The abbreviations indicate the level of importance and follow the MoSCoW method. http://en.wikipedia.org/wiki/MoSCoW_Method

NA-5	IPv6 is used as addressing scheme for individual service component instances.	M
NA-6	The network layer offers name-to-address translation functionality.	M
NA-7	Service sessions are identifiable by the communicating service instances.	M
NA-8	Service sessions are identifiable by intermediate network routers.	C

2.2 Forwarding

The forwarding of data by the FUSION service routers and endpoints underpins all FUSION service router layer functionality. Both FUSION signalling information and application-specific data must be forwarded.

We envision the FUSION forwarding plane as running on top of today's IPv6 stratum, but stretching beyond the host network attachment points to the individual service instances. As such, the forwarding plane also handles the exchange of signalling and application-specific information between instances running on the same host.

Signalling information includes at least management of service sessions, monitoring information andd routing table updates.

Inter-process communication between service instances can use various mechanisms, from single-host communication through shared memory or pipes, to socket-based communication between different hosts. FUSION should not confine the range of available IPC mechanisms. Because it is unfeasible to provide all possible IPC channels, service components may set-up communication channels (both intra- en inter-host) that run outside of the FUSION forwarding plane. Basic information to be used for setting such channels can be exchanged by means of FUSION signalling.

Req no.	Description	Level
DP-1	The forwarding plane transports signalling information between FUSION network elements.	M
DP-2	The forwarding plane forwards application layer data exchanged between service component instances.	C
DP-3	Applications can establish communication channels outside of the forwarding plane.	M

2.3 Routing

Multiple zones may advertise the same service component, and service instances may come and go. Service routers must be provided with the necessary information to construct their forwarding tables. The advertisement of service information may go beyond the mere announcement of service instantiation; it may also include monitoring information (detailed in section 2.5). The exchange of routing information may use various mechanisms, from simple flooding to all neighbours to more sophisticated routing algorithms as those that will be developed by FUSION. The N-casting algorithm presented in section 6 provides delivery to the n-closest members of a group, and is one possible strategy to optimize exchange of routing information.

Execution zones must announce the *availability* of services by injecting this information via the zone gateway. These announcements only indicate that the zone will serve requests for that particular service, but does necessarily discloses any information on the actual number of running instances. For example, execution zones may decide to instantiate on request (late binding). A corollary of this requirement is that service availability is announced at zone granularity.

As described in Deliverable 2.1, the service routing layer is composed of multiple autonomous routing domains each running a proprietary routing protocol. A routing domain includes all FUSION service routers that are under the authority of the same FUSION network provider. Domains peer via edge service routers in a default-free zone (DFZ), as in today's Internet. FUSION will specify the routing protocol for this DFZ and may study additional intra-domain protocols.

Service announcements will be forwarded between all service routers of the same domain. However, inter-domain service announcement may be governed by business policies from both the service provider and the network provider. Service providers may impose geographical constraints on its customer base, and network providers may use the service announcements for traffic engineering to and from peering domains.

Req no.	Description	Level
ROU-1	Execution zones announce the availability of services.	M
ROU-2	Service routers exchange service announcements.	M
ROU-3	The forwarding of service announcements can be restricted by business policies.	S
ROU-4	Service routing domains peer in a default-free zone with a FUSION specific routing protocol.	M

2.4 Service instance selection

The basic primitive of the service routing layer is to deliver service requests to an instance of this service by selecting between the many zones that advertise the service. Various parameters can be taken into account: application-specific metrics, monitoring information from the network and the hosting infrastructure and business policies.

Essentially, the service instance selection is the process of translating a service name into the IP-locator of the selected instance[2]. This name resolution should not necessarily be a single step (e.g. querying a name resolution service like DNS), but may occur gradually by name-based forwarding of service routers. Each service router further refines the selection process by choosing the next branch of the path towards the finally selected execution zone.

Ultimately, one service router will translate the service name into the public IPv6 address of the selected service instances. This service router may be the zone gateway, but may also happen earlier in the request path. However, name resolution can also be used in to identify intermediate routers. For example, one routing domain might use name resolution to determine the gateway of the next routing domain.

[2] This IP address does not necessarily reflect the physical host of the selected instance. For example, this locator may be the IPv6 address of the execution zone gateway.

Service requests can be handled in two ways by the service routing plane. Clients may ask the service routing plane to take an active role in the decision process (*network-driven*) or may only need support in the discovery of candidate instances (*client-driven*). Between these two extreme variants of the selection process, intermediate ways will be studied by FUSION. For example, network management policies may restrict the forwarding of requests to a specific subset of the zones, or services may ask the FUSION network layer to only discover instances that meet certain QoS requirements.

2.4.1 Client-driven

In the client-driven variant, one service instance requests the service routing plane to construct a list of candidate instances of a specific service. Service routers may forward requests to multiple neighbours. Service instances receiving this request may choose to ignore the request, or to reply with a service offer. The requesting instance autonomously selects one candidate among all received offers. Then the establishment of the actual session for data exchange may be run completely transparently to the FUSION routing plane.

The advantage of this *client-driven* selection scenario is that application-specific metrics can be taken into account by the requesting client service. The disadvantage is that the selected candidate may be suboptimal with respect to network parameters. Services can be highly dynamic, and the time needed for the client to collect a list of candidates, query each candidate with application specific parameters and the actual establishment may be too high. Also in this scenario, QoS management may be limited, since the actual data transfer session may be running outside of the FUSION framework.

2.4.2 Network-driven

In the network-driven scenario, clients request the service routing plane to carry out the instance selection. In the simplest case, routers will forward service requests on only one neighbour, although more complex policies can also be considered. Note that the service router plane will only perform the selection process at the granularity of the execution zone. It is left to the execution zone (zone manager in particular) to load balance incoming requests to the hosted service instances.

The advantage of this *network-driven* selection scenario is that network monitoring information or QoS policies can be taken into account more comprehensively compared to the client-driven case. Moreover, as the request is being forwarded, the necessary QoS reservations can be made. A major limitation of this scenario is that only rather generic service selection parameters can be used for reasons of scalability and generality in the service routers. This may limit the quality of the selection process in terms of the resultant service performance.

2.4.3 Router operations for selection

For intelligent selection, service routers may perform more advanced actions than merely consulting the forwarding table and transferring the request to one outgoing interface.

These actions include:

- forward the request to one or more neighbours
- query a name resolution service to resolve the requested name into the locator (IPv6) of the next FUSION network element involved in the selection process
- contact the service orchestration layer, e.g. to boot a new instance, before forwarding the request

The implementation of these actions requires appropriate interfaces towards the monitoring infrastructure and the FUSION Orchestration Layer.

2.4.4 Requirements for Instance Selection

The table below lists all requirements related to the service instance selection process.

Req no.	Description	Level
SEL-1	Service routers deliver service requests to one or more zone gateways advertising the requested service.	M
SEL-2	Different selection strategies can be used in different parts of the service routing plane, even within a single routing domain.	S
SEL-3	Service routers can fork requests to multiple neighbours and leave the final selection to the initiating service component.	S
SEL-4	The service routing layer performs selection at least at the level of the zone gateway.	M
SEL-5	Service routers can take more advanced actions than forwarding, e.g. querying a name resolution service or contacting the service management layer.	S
SEL-6	Service routers have some level of decision autonomy w.r.t. to the orchestration layer	S

2.5 Monitoring

The FUSION service routing layer requires monitoring infrastructure to provide an up-to-date view on the network and server load. Relevant network-layer metrics include bandwidth and latency, but also parameters more directly related to services should be monitored, such as service demand patterns or server load. Monitoring interfaces are required between the service routers and the Orchestration Layer, and between the service routers and the underlying network.

Service routers, including zone gateways, can take part in the monitoring framework by reporting statistics to the Orchestration Layer on service demand patterns. Statistics may be reported by routers grouped by neighbour, source address of the requesting instance, etc.

The service availability can be monitored via the abstraction of session slots, which is introduced and discussed in detail in D3.1. Session slots are a service-invariant and easy-to-use metric indicating the availability of resources for that particular service. Zone gateways will advertise their available session slots.

Since many services will be hosted by execution zones and the demand per service can be highly variable, the optimal selection strategy for routing service instances should be adaptive. Therefore, service routers may directly consume and react to monitoring updates.

Another important consideration is the degree of network awareness in FUSION. In the most general case, when there is no monitoring information provided by the underlying network, the service routers will have to actively gather network monitoring information. In this case, service routers could request to be notified of instance that was finally selected for a request. In other cases, e.g. when a FUSION service routing domain coincides with a trusted IP domain (e.g. of an ISP), more detailed network information could be available.

Req no.	Description	Level
MON-1	Service routing plane elements exchange monitoring information.	M
MON-2	Service routing plane can interface with the monitoring infrastructure of the underlying IPv6 plane.	M
MON-3	FUSION routers are notified of the selected instance of requests they have forwarded.	C

2.6 Data plane QoS

Service components can establish long lived flows between them. The goal of flow management is to provide QoS guarantees by engineering the data traffic routing.

An important design consideration is the degree of network awareness that is taken into account during the selection process. This was already discussed in the previous section for monitoring information, but is equally important for the selection process. Service routers could take into account the IPv6 QoS mechanisms for their service request routing.

Existing data plane models offer no or only limited QoS support. In general, it is unrealistic to assume that the FUSION Orchestration Layer or the FUSION service routers would be able to configure or reserve resources in the underlying IP stratum on end-to-end basis. Emerging Network-as-a-Service (NaaS) paradigms typically refer to the aggregation of long-lived flows, whereas FUSION is more focusing on individual sessions. Local resource reservation schemes could resemble, e.g., those known from IMS (PCRF and Rx interface).

At this stage, the FUSION consortium decides to assign lower priority to providing QoS mechanisms in the service routing plane.

2.7 Security and integrity

The service routing layer can only take a supporting role in providing security and integrity in the FUSION service routing plane. As described in D2.1, most security aspects should be handled and provided by the FUSION Orchestration Layer.

The most important challenge to be addressed is to ensure that a service request is routed to an instance (or zone) that truthfully exposes the service. Measures must be taken to avoid that malicious services spoof the namespace.

Req no.	Description	Level
SEC-1	The selection mechanism is robust w.r.t. service instance spoofing.	M
SEC-2	Forwarding of encrypted binary data between Service Component Instances is supported.	M

2.8 Evolvability and backwards compatibility

Evolvability means that the FUSION architecture can be gradually deployed over today's Internet, rather than starting from a clean-slate design. Starting from an overlay network on top of the IPv6 stratum, more and more functionality could gradually be included natively in the network layer. This

also means that the FUSION routing protocols and selection strategies should cope with the situation in which not all parts of the network understand the FUSION syntax.

Second, FUSION should be backwards compatible with legacy services. This might require the development of adapters between today's socket API and the FUSION network API.

Req no.	Description	Level
EB-1	The FUSION network layer can operate as an overlay of today's IPv4/IPv6 networks.	M
EB-2	The FUSION network layer is backwards compatible with legacy services.	S

3. NETWORK LAYER ARCHITECTURE AND INTERFACES

3.1 Design considerations

Before we decided which solutions FUSION would adopt we explored several scenarios and analysed their advantages and disadvantages and how much potential they would have to meet FUSION's requirements described in section 2.

3.1.1 Native IP anycast

This scenario uses IP addresses (IPv4 or IPv6) as names. A service gets allocated an IP address which gets announced. The routing protocol announces the several replicas (e.g. execution zones where the service is accessible) from different places in the network and builds the distribution trees and install IP entries in normal routing tables. Different routers will point to different servers. Session slot and evaluating function can be sent as an extension to BGP.

When a service is going to be deployed, the service orchestrator requests an IP address. This can be done in real-time by an organization like IANA. The orchestrator will attempt to send an estimation of which ASes will have copies of the replicas.

The address allocator will try to choose an address that minimizes fragmentation in the address space and returns it to the service orchestrator which then deploys the services and allocates the IP address given.

The IP address gets announced through BGP to the Internet. Extensions to BGP may have to be defined to help with the selection at each point in the network.

If service geography changes significantly, the orchestrator may request a new address with the new parameter and migrate its users to the new address maintaining both of them for some time.

IP addresses may be obtained from DNS without any changes to today's operation. Potentially TTLS need to be managed properly for appropriate address migration.

Advantages:

- It works with today's routers
- No need for service routers
- No changes to end systems stacks
- Incremental deployment

Disadvantages:

- How to allocate the IP addresses in an efficient and coordinated manner (We would need to estimate what is the absolute best aggregation that we would be able to achieve, in order to estimate the minimum fragmentation induced in routing tables. If this minimum is excessive, the problem needs to be solved first)
- How to build the distribution trees minimizing fragmentation (one possibility is to use compact routing and smart allocation of names)
- How to do "load-balancing" in real-time. Selecting based on parameters from the service and/or client
- Limited possibility for parameter based forwarding decisions
- Very hard to do multi-metric, especially session slots. At first glance one would need one "forest (set of anycast trees)" per metric

3.1.2 Serval

Serval was presented in [NDKG12] and proposes a new Service Access Layer (SAL) that sits above an unmodified network layer, and enables applications to communicate directly on service names. The SAL provides a clean *service-level* control/data plane split, enabling policy, control, and in-stack name-based routing that connects clients to services via diverse discovery techniques. By tying *active sockets* to the control plane, applications trigger updates to service routing state upon invoking socket calls, ensuring up-to-date service resolution. With Serval, end-points can seamlessly change network addresses, migrate flows across interfaces, or establish additional flows for efficient and uninterrupted service access

Advantages:

- Helps with mobility

Disadvantages:

- Operating system stacks need to change
- Applications need to change

It would be difficult to meet all set of FUSION requirements

3.1.3 Extensions to DNS

This scenario uses extensions to DNS to implement service resolution. Requests are issued as normal DNS query which returns IP address(es) of machine(s) running the instance(s). This is an incremental change to existing architecture with Hierarchical naming. One would need a novel routing protocol to be run between DNS servers. The DNS interface from end hosts could remain the same.

Advantages:

- No change to the IP
- No change to end system stacks
- Minimal change to apps
- Offers the possibility of return of several options and let the end system decide

Disadvantages:

- Extra delay for resolving the query
- Very difficult to implement service invocation
- It is a packet based protocol which is more difficult to extend

- To implement full list of requirements one would need to change the client/application interface with the DNS
- Tied to a very specific name space. Difficult to extend to full URIs or different name spaces

3.1.4 Use of an established application layer protocol

This family of solutions consists of using a well-established application level protocol and extend it to implement FUSION functionality. The two most representative candidates were HTTP and SIP. Service request resolutions or invocations would consist of GET or INVITE commands and then extensions would have to be defined for these protocols to implement our list of requirements. The implementation of service router functionality would be delegated to proxies.

Advantages:

- Extensive know how in the community on how to deploy these services
- There is partial match of some of the functionality needed for the FUSION requirements (for example HTTP REDIRECT could be used to inform clients of another server)

Disadvantages:

- Interaction between clients and servers is not a complete match to FUSION requirements
- Some security requirements match madly with these protocols
- Routing protocol requirements would need to be fitted as extensions to these protocols.

3.1.5 A new overlay protocol

This family of solutions involves an application layer protocol present at the end nodes and service nodes. The client and servers run an application layer library used to route the request. Server routers are application layer entities running in the network. This option can be used to contact the service instances directly or the zone managers that can return an identifier.

Advantages:

- No change to IP
- Possibility of complex parameter based forwarding decisions
- Incremental deployment
- All set of requirements can be met

Disadvantages:

- Needs the deployment of a new protocol, a new set of service routers
- Applications need to be modified

3.1.6 IP tunnelling

Here the “end2end” packet gets encapsulated in an IP packet which gets passed to the service router. Service routers decide where to forward the packets and encapsulate in the new tunnel. Conceptually this is similar to an overlay solution but functionality is done at network layer.

Advantages:

- Little change to routers

Disadvantages

- Work needs to be done in routers

- End system stacks or applications need to be modified

By putting functionality in the network layer it is difficult to implement more elaborate functions

3.1.7 Design considerations: conclusions

After careful consideration on the pros and cons of all the alternatives above we concluded that the FUSION solution will diverge to consider two approaches:

- A new overlay text based protocol which will try to achieve all the requirements of FUSION. This will be the main service routing solution of this project. We concluded that on balance it would be worth the extra costs of developing a new protocol in order to achieve the desired functionality. None of the existing alternatives allows the implementations of all FUSION requirements. To do this we will get inspiration from the community experience in designing and deploying several of our candidates, especially DNS, SIP and HTTP. We expect service routers to be run primarily from network providers in the same "spirit" as DNS.
- An exploratory native layer solution that proposes changes to the control plane of IPv6 anycast and can implement a subset of the requirements with a low number of modifications to today's Internet fabric. This will also be used as a comparison of cost/benefit with the main solution above.

3.2 Existing approaches to Service-Aware Networking

In this section we compare briefly several state-of-the-art architectures with respect to the way of enabling networks to be service aware. We will evaluate each architecture against the same criteria, summarized in the following comparison table:

Integration with IP	Is the architecture conceived as an overlay of IP or as an overlay?
Network layer interface offered to Service Component Instances	What is the functionality provided by the network? How do services talk to the network? Is the traditional socket API still valid?
Structure of naming scheme for Service Instances and Service Instance Components	Is the naming scheme flat or hierarchically structured?
Routing of session data	Is application data routed through the overlay, or directly over IP?
Service advertisement	How are services registered and advertised through the network?
Instance selection	What is the actual selection mechanism to bind requests to instances?
Addressing scheme for service instances	How are service instances being addressed? IPv6 or another scheme?
Support for late-binding	Should a requesting instance know the locator of the remote service it wants to communicate with?

Routing table population	What is algorithm to construct routing tables?
Forwarding policy	Can routers take intelligent decisions when forwarding requests?

3.2.1 IRMOS/ISONI (Intelligent Service Oriented Network Infrastructure)

The general goal of IRMOS [SAIR08] is to enhance SLAs in a grid/cloud computing platform with strict quality guarantees in the transport network. To this end, all elements of IRMOS platform (computational nodes, network links and storage boxes) should be able to provide guarantees to individual activities while the physical resources are shared across multiple services.

IRMOS provides means for automatic deployment of services on best fitting resources distributed in a network. The deployment and instantiation of developers' service is based on an abstract description of all the execution environment requirements of the service (in the form of Virtual Service Network, VSN), including the description of the interconnections between service components and their individual QoS demands. Within the IRMOS platform, ISONI (Intelligent Service Oriented Network Infrastructure) implements the overall architecture including resource control plane, path manager, and execution environment.

The overall coordination of service deployment process in ISONI is done by the Deployment Manager who in functionally centralised way matches computing and network resources by negotiating needed resources with Resource Manager and Path Manager, respectively.

Computing and storage resources are managed by ISONI's resource manager in a way similar to that in clouds/grids (virtualisation, workflows, resource reservation, etc.). This is the responsibility of the execution environment that provides global resource management and allocation policy for scheduling services enhanced with real-time attributes. In particular, the composition of applications into service components in a workflow, and their timing requirements are taken into account.

The unique feature of ISONI consists in integrating into one platform the network resource management and allocation functions and cloud services. This integration is based on the concept of virtual service networks with QoS guarantees created for each service instance, and Path Manager is the key functional block that implements related functions. In particular, Path Manager is responsible for resource discovery, selection and required configurations and supervision of all network allocations required during the VSN life-cycle. Path Manager can be thought of as a bandwidth broker whose operation could comply the NaaS paradigm. It adopts a network-resource model derived from the ITU-T G.805 series. A 2-level hierarchical architecture of Path Manager in IRMOS strictly corresponds to the hierarchical organisation of IRMOS platform into IRMOS "nodes" that contain (gather sets of) IRMOS "physical hosts" and IRMOS "domains" that are composed of physically interconnected IRMOS nodes.

Integration with IP	IP overlay (capable of using also other transport technologies). IRMOS provides a framework for a QoS enabled cloud/grid architecture where QoS provisioning is based on the use of proprietary ISONI eXchange Box nodes to build virtual networks for each instance of a composite service.
Authentication and security	Relies on out-of-the-box solutions as IPSec for virtual networks, SSH for the client and WS-Security for PaaS services. In particular, ISONI Gateway maintains IPSec tunnels for inter-domain communication within a given VSN, and uses WS

	interface for external clients; it is implemented using Globus Toolkit 4 including its Grid Security Infrastructure Framework.
Service advertisement	Information service is used by infrastructure providers to advertise the ISONI capabilities with the purpose to select (suggest) the appropriate candidates for deploying the applications (based on the capabilities, price, …).
Naming scheme for service (instances)	For external clients naming is subject to the rules defined by Web Services.
Addressing scheme for service instances	Within a VSN, each service component is assigned virtual address (from a private VSN address space); this service component can have multiple instances and each instance is assigned a pool element address (from private VSN address space); moreover, each service component is assigned a physical address that is bound to the physical host on which the corresponding VM runs.
Service discovery and resolution	– Discovery service provided as by ISONI (IaaS) is responsible for finding registered candidate ISONI providers that meet the low level QoS constraints defined in the technical SLA. It applies the rules based on the advertised capabilities of the ISONI providers and the QoS types that are supported. – External clients interface IRMOS through IRMOS Gateway so they use Web Service-based mechanisms.
Service binding	Accomplished by the SLA Management Component at the IRMOS (PaaS) level.
Routing table population	ISONI eXchange Box (IXB) layer is responsible for routing on the transport layer. On the other hand, service routing capability not defined explicitly; its role is embedded in the SLA Management Component.
Routing policy	IRMOS/ISONI Provider dependent.

Table 2: IRMOS/ISONI and FUSION

3.2.2 IMS/SIP-based architectures

IMS [CMA07] is a service control overlay based on SIP designed by 3GPP and ETSI for controlling session-based services in 3GPP and converged NGN networks. Apart from session-oriented services it also supports other functions as, e.g., simple subscribe/notify service, and instant messaging and presence based on SIP.

Out-of-band service routing plane based on SIP can access the SPD FE (Service Policy Decision) that derives and translates service requirements and negotiates network-level QoS. Essentially, physical transport of user data is separate from request routing, and transport control capabilities are located in network domain visited by the client.

In IMS, the S-CSCF (Serving Call-Session Control Function) function in home domain able to reroute SIP initial requests towards application servers in order to trigger services; this re-routing is done based on Initial Filtering Criteria (IFC) downloaded by S-CSCF from HSS on user's registration. An IFC

is a logical expression matching a message to bind triggers with application servers and from FUSION perspective can be seen as a simple service graph (or even a service chain).

Integration with IP	IP overlay. IMS provides a framework for control and delivery of session-based services over diverse IP-based networks with QoS support by the underlying networks, enhanced with simple composition of component services that can be plugged into the service through rerouting the signalling messages.
Authentication and security	In the application server context, each client can be authenticated using different methods like, e.g., Digest authentication or trusted host authentication.
Service advertisement	Could be based on IMS presence service model based on SIP presence.
Naming scheme for service (instances)	Public Service Identities, typically in the form of SIP URI, in order to identify services hosted on application servers.
Addressing scheme for service instances	Name to IP mapping is performed by querying the Service Registry, often DNS-like in practice.
Service discovery and resolution	Could be based on IMS presence service model based on SIP presence.
Service binding	A service request in the form of SIP message is routed through a chain of IMS-defined SIP proxies. In particular, S-CSCF may contact databases and use application servers to reroute initial requests to appropriate services.
Routing table population	A the level of IMS proxies (CSCFs), routing is partially defined by a fixed chain of CSCFs involved in routing SIP messages. Next-hop_name-to-address resolution for initial requests is made using DNS. Similar mechanism applies to application servers; application servers can additionally change routing on-the-fly by modifying the Route header in SIP request messages (underlying mechanisms, APIs, and policies etc. are application-server specific).
Routing policy	Load balancing among application servers at S-CSCF level is not fully addressed in IMS. Few propositions rely on the use of DNS, but such capability seems in general to be vendor-specific.

Table 3: IMS and FUSION

Suitability for FUSION: it seems that NGSON is an approach to service-oriented networks that generalises IMS concepts.

3.2.3 NGSON: Next Generation Service Oriented Network

NGSON [NGSON11] was proposed by the IEEE, and is designed as an overlay framework for control and delivery of composite services over heterogeneous IP-based networks. In NGSON, service configurations can be customized and adapted to the dynamic context of users, devices, services and networks.

NGSON specifies a functional architecture that provides advanced service and transport-related functions to support context-aware, dynamically adaptive, and self-organizing networks. To this end it defines functional entities and abstract protocol mechanisms, but does not provide their efficient implementation. These functions are realised by strategically located nodes with service-specific forwarding and control capabilities. According to NGSON, the general organisation of "the network" follows the split into the signaling/control plane (to achieve service awareness) and transport plane. NGSON is declared to target current transport networks (IP, and also other technologies), and also cloud networks (in the sense that the services supported by NGSON may run in clouds, and also the functions of NGSON platform can be run as cloud applications, e.g., for auto-scaling purposes).

Integration with IP	IP overlay. NGSON provides a framework for control and delivery of composite or component services over diverse IP-based networks (e.g. legacy IP, P2P, IP Multimedia Subsystem), possibly with QoS support by the underlying networks.
Authentication and security	Carried out by the Identity Management (IDM) functional entity. The authentication and authorization is categorized into two levels, user level and service level. User level identity management deals with determining the identity of the user and the appropriate agreements to use a service. Service level identity management deals with determining the identity of the service and the appropriate agreements to use other services in NGSON. NGSON uses the global ID for an end user to sign in to multiple services, without needing to create service specific IDs and passwords (Single Sign-On).
Service advertisement	Provided by the Service Register functional entity. It assists Service Routers to discover where and how to route a service request. It maintains current location information of the services.
Naming scheme for service (instances)	Global and local IDs. Global IDs uniquely represent a user, independent of service providers or services. Local IDs are identities that uniquely represent a user in a specific service or service provider. Mapping is performed by the IDM.
Addressing scheme for service instances	Name to IP mapping is performed by quering the Service Registry function.
Service discovery and resolution	Service Discovery and Negotiation is typically used when abstract service is specified in the request. The service requestor may provide a number of criteria, such as service interface, availability, QoS, SLA, version, network area, regional area. The Service Discovery and Negotiation function obtains a list of similar or relative services (instances) and returns it to the requester. In a second phase, Service Routing will contact the Service Register functional entity for name-to-address mapping.
Service binding	A service request is sent to a service router. Service router may query the Service Registry and/or Context Information Management function in order to get additional information about the requested service that is needed to route towards the

	service. Alternatively, the request can be routed to Service Discovery and Negotiation function element that selects the best service. Setting up the policies for the use of either of these binding mechanisms is out of scope of NGSON specification.
Service routing	Responsible for routing of service requests received from an end-user or a service to an appropriate service instance along the overlay network. A service request contains the functional requirements of the target service and input data to be delivered to the target service for interactions. The target service can be specified with either functional description (i.e., service type) of the expected service or a concrete service instance. If only the abstract service type is given in the service request, the service discovery and negotiation function selects the best candidate service instance instead. Thus, the service routing function binds and invokes the target service instance to forward the service request. A response to the request from the target service is forwarded back to the service requestor by the service routing function. Plays a key role in conveying service requests and responses among functional elements (e.g. Service Discovery and Negotiation function, Service Composition function), and contacting other supporting functions during this process (e.g. Service Policy Decision).
Routing table population	Not addressed.
Routing policy	Addressed partially. For example, the SDN functional entity may filter several candidate service instances by different criteria, such as availability or cost, which may vary with the context. The context is obtained by processing and aggregating the raw context information collected from different context sources, such as underlying user profile services, underlying networks, and terminal devices. Moreover, Service Routing is allowed to send requests to one or multiple destinations to select the best instance of the requested service.

Table 4: NGSON and FUSION

Suitability for FUSION: NGSON identifies several individual architectural components and functionalities that are relevant for FUSION. As of today, only the functional architecture of NGSON has been standardized, but no interface specifications are available. Thus, an overall suggestion is that NGSON can serve as a blueprint for FUSION in several aspects as explained in the following. Service communication model: NGSON tries to integrate session-based (conversational) services (i.e., those that transfer their service data using dedicated sessions/connections rather that NGSON-level signalling) and transactional (non-session) services (i.e., those that transfer service data by piggybacking them in NGSON signalling). FUSION does not assume explicitly that services can exchange their data using service routing signalling. Thus, from the point of view of the role of signalling, FUSION could build on NGSON as it potentially requires a subset of NGSON capabilities. Resource management: FUSION explicitly targets management of cloud-based resources by addressing service placement problems; NGSON only maps service requests to the best running service instance (through Service Routing function that uses Service Discovery & Negotiation function and Service Policy Decision function). Thus, FUSION could NGSON by considering also service placement (this function, attributed to the orchestrator in FUSION, could correspond to a new block

in NGSON). Regarding network resources, NGSON adopts a classical bandwidth-broker approach to network resource and QoS control (through Service Policy Decision fed by the service routing). Obviously, this approach can in theory be adopted by FUSION, but its tight integration with the management of cloud-based resources (computation, storage) may rise strong scalability concerns.

3.2.4 CCN (Content-Centric Network)

CCN architecture (a.k.a. NDN – Named Data Networking) has been proposed under the heading of Information-Centric Networks as a clean-slate solution for the future Internet [JSTP09], [NDN]. CCN defines a forwarding plane of a transport network that operates based on the route-by-name principle assuming hierarchical names of content objects. According to CCN, content objects (data chunks of named content) are delivered in response to requests specifying the name of the requested object. Data chunks follow the reverse path of corresponding requests and can be cached in the intermediate nodes for future reuse. Thus, CCN as such defines a thin transport plane while all required control/management functionalities are missing from the original proposition (in particular routing and higher level functions such as, e.g., publish-subscribe). The work on these (and other) topics in the context of CCN has been undertaken in several works, e.g. [HAAW13], [CAJF11].

Integration with IP	Clean slate, although CCN “faces” can run on top of different layers including IP.
Authentication and security	Name-data binding authenticated by the signature of data object owner (the use of private/public is assumed).
Service advertisement	Could be built using routing capabilities of CCN (however, routing is still an open issue in CCN, although several propositions have been proposed).
Naming scheme for service (instances)	Hierarchical names with textual syntax and conventions as to naming consecutive chunks of a given object (including streams).
Addressing scheme for service instances	To be developed, preferably based on CCN naming scheme..
Service discovery and resolution	Discovery not addressed explicitly. Resolution at low level done by forwarding Interest packets (requests) according to the forwarding rules.
Service binding	Not addressed explicitly. Preferably should be built into Interest forwarding.
Routing table population	Done by routing protocols; application of IGP/EGP protocols has been proposed, but also other schemes are considered by different researchers including interfacing with external controllers.
Routing policy	Dependent on the setting of routing protocols and other mechanisms including local rules implied by the “strategy layer” (although the latter does not seem to have been studied extensively). The style could be similar to that known from IP networks, but additional concerns known from e.g. CDN interworking, may be superposed.

Table 5: CCN and FUSION

Suitability for FUSION: Has some potential, but more insight is needed into different scenarios of using CCN in cloud environments. We notice that CCN, and the ICN principle in general, assumes the knowledge of names of content at the level of network infrastructure. As content retrieval can easily be generalised to accessing services "by name" we expect ICN to be a potential candidate solution for FUSION. Early attempts to such generalisations have already been proposed an thus can be a starting point for a future study.

3.2.5 NetInf

NetInf [SAIL13] has been developed by FP7 SAIL project as a proposition for future Internet ICN. NetInf layer defines NetInf protocol for named data object publishing, searching and retrieval along with name-based routing and name resolution services. Corvengence layer allows Netif to run on top of any transport, e.g., HTTP/TCP, IP, UDP, Ethernet. NetInf defines a new URI scheme ("ni" URI) for naming NetInf data objects to accommodate name hierarchies.

Integration with IP	IP overlay in the sense that can run on legacy IP networks, and that data transfer can be over TCP/IP stack. Clean slate in the sense that NetInf protocol can run without IP support. In the latter case NetInf data transfer resembles CCN, although reverse path is maintained by using label stack header (similar to Via header in SIP or HTTP).
Authentication and security	Authenticating name-data integrity by the inclusion of hashes in "ni" URI.
Service execution	Outside the scope of NetInf.
Service advertisement	PUBLISH method is used for publishing data objects including objects themselves and their metadata. Details depend on the policies of the Name Resolution Service.
Naming scheme for service (instances)	"ni" URI with hierarchical authority part (for routing/aggregation purposes) and named object identifier (used for name matching)
Addressing scheme for service instances	Would have to be based on "ni" URI.
Service discovery and resolution	Using SEARCH method of NetInf. Name resolution achieved either by means of request forwarding (like HTTP proxy) or by Name Resolution Service. NRS is based on Multilevel DHT, and can respond with either routing hints to be used for request forwarding purposes or locators used to access data objects directly through underlying transport network.
Service binding	Adopting GET method.
Routing table population	Using IGP/EGP protocols suggested as a possible option.
Routing policy	Domain-depedent for request (GET) forwarding.

Table 6: NetInf and FUSION

Suitability for FUSION: Has some potential, but more insight is needed into different scenarios of using NetInf in cloud environments. According to the ICN paradigm, NetInf assumes the knowledge of names of content at the level of network infrastructure which is key to recognising "services" at

network level. Interestingly, in the SAIL project cloud computing has been extensively studied and the potential role for NetInf mentioned therein (apart from running NetInf functions ono clouds) is the access to cloud services that use NetInf naming. Handling metadata directly by the NetInf layer seems to be an enabler to directly support such a scenario by NetInf.

3.2.6 PSIRP/PURSUIT

PSIRP/PURSUIT architecture [PSIRP10] introduces an ICN-like architecture that is based on a publish-subscribe forwarding paradigm where DHT-based Pub/Sub rendezvous function (subsystem) allows for matching publications with subscriptions, topology management and formation function serves the routing purposes to provide delivery information to the forwarding function (as labels representing source-routed paths in the form of Bloom filters), and the forwarding function provides the relaying capabilities (including QoS mechanisms) for packets using the source-routed labels provided by topology management function.

Integration with IP	Clean slate. Pub/Sub model assumed for data registration and search capabilities at the Rendezvous layer, and source routing based on Bloom filters is used in the data plane. Source routes are delivered by a separate Topology management and formation layer.
Authentication and security	The Pub/Sub component can provide advanced capabilities while serving publications of and subscriptions to context/services. The forwarding plane (based on Bloom filters to specify source-routed packets) provides some level of protection against unwanted traffic.
Service advertisement	Could be built adopting the capabilities of the Rendezvous function. PURSUIT claims to support mobility, so Rendezvous potentially offers a good platform to support dynamic service instantiation.
Naming scheme for service (instances)	PURSUIT labels (flat names) organised into scopes at the Rendezvous. The applicability of scoping to support service-oriented functions needs further study.
Addressing scheme for service instances	Probably could be based on PURSUIT labels (flat names) of the Rendezvous function.
Service discovery and resolution	PURSUIT Rendezvous function should preferably be used for this purpose. The use of scoping may assist in creating various policies in accessing the services.
Service binding	Rendezvous and topology management and formation should be able to support this function (binding to forwarding labels in the form of Bloom filters).
Routing table population	For transport layer, Topology management and formation function is responsible for this function. Routing (i.e., searching) at the Rendezvous layer is based on the DHT principles.
Routing policy	For the transport layer. SLA and QoS requirements may be taken into account by Topology management and formation layer.

Table 7: PSIRP/PURSUIT and FUSION

Suitability for FUSION: PURSUIT/PSIRP assumes the knowledge of names of content at the level of network infrastructure. Its Pub/Sub function handles metadata to provide rich service registration and discovery capabilities, and in combination with the topology function it can enable efficient service binding. The architecture has potential to be adopted by FUSION, but similarly to NetInf, a deeper analysis of different scenarios of its use in cloud environments is needed.

3.2.7 SERVAL

Serval [NDGK12] is a new modification to the internet architecture where service access is central. Serval allows users to find a service instance based on several metrics such as load and latency, and maintain a connection with that service instance. It is designed by Princeton University and source code is available since 2012. Two demos are available: the first one is the website of Serval, where each request is load-balanced between two Serval enabled servers. The second demo is a mobile user listening to a music stream while demonstrating how Serval is able to migrate an active connection when a more appropriate connectivity is found (e.g. use Wi-Fi over cellular data when available to reduce cost).

Serval introduces a service oriented layer (named SAL – Service Access Layer) located between the transport and the network layers. Its main role is to enable applications to communicate using service names. More specifically, Serval allows request packets to be forwarded by specialised Serval nodes to the targeted instance of service; both responses to confirm the connection and subsequent traffic are sent only through the IP layer thus bypassing the SAL layer (Serval nodes). So, the main role of request packet is only to trigger connection setup between the requestor and appropriate service instance. For this purpose, and in addition to normal IP addresses and application-level data, a request packet contains new elements defined by Serval (serviceID) that are used by the SAL layer.

In Serval, a separate service controller layer is responsible for appropriate configuration of service forwarding tables at the SAL layer. Thus, its functionalities are related to those of the service orchestrator in FUSION. However, the architecture for interworking between FUSION and Serval is an open topic.

One feature of Serval particularly relevant to FUSION is that setting appropriate rules for handling requests at the SAL level enables notifying the service controller of service registering/deregistering which in turn can update service resolution database (and service forwarding tables where appropriate). A useful capability is that forwarding of service request can be queued and delayed be a service router in order to notify the service controller and wait until it responds, thus allowing, e.g., to install service forwarding (resolution) rules in real time (on-demand).

Serval may by definition be used to allow communication using service names controlles by a service-aware control plane (missing from Serval itself) that supports such functions relevant for FUSION as service routing and resolution.

Integration with IP	IP overlay. Serval adds a Service Access Layer that sits between the transport and network layers. The SAL maps service names to IP addresses before connections are made. Once a connection is established, communication is based on standard IP.
Authentication and security	Serval uses random nonces to protect users from off-path attacks. Further security is not handled by Serval.
Service advertisement	An application calls bind() on a Serval socket API and registers a local forwarding entry. Local service controller can propagate

	this local announcement to upstream service controllers.
Naming scheme for service (instances)	Services are identified by service name.
Addressing scheme for service instances	A connection with a service instance is identified by a service name combined with a flow ID.
Service discovery and resolution	SAL present on each Serval enabled host/router translates service names to network addresses.
Service binding	When a user calls connect() or send() on application level, the client's SAL assigns a local flowID and adds a forwarding rule. A SYN packet is created, along with the local flowID and the service name. Each intermediate SAL will apply a forwarding rule until a service host is reached. The service host creates its own flow ID and replies with a SYN-ACK along with its own address and flow ID. All following communication uses the end-point addresses and bypasses the SAL.
Routing table population	Not mentioned.
Routing policy	Each SAL can take a set of pre-defined actions associated with a forwarding entry. This action is set by the local service controller, SAL only follows the active forwarding entries using longest prefix matching.

Table 8: SERVAL and FUSION

Suitability for FUSION: Serval has similar goals as FUSION (e.g. focusing on service access through name based requests). One of the important questions in FUSION is how we will establish a long-lived connection with a service instance. Serval's approach is a possible solution to this problem. Also having a service layer below the transport layer could increase performance compared to an overlay approach.

3.2.8 eXpressive Internet Architecture (XIA)

XIA [HADL12] is the result of collaboration between Boston University, Carnegie Mellon University and the University of Wisconsin/Madison. XIA recognizes the shift from IP-based Internet to new architectures focusing on different principals, such as content or users. XIA describes an architecture designed to support different principals and allows new principals to be added over time. It also allows incremental deployment due to a fallback mechanism for routers which don't support XIA.

Integration with IP	Both. XIA addressing uses clean slate service ID addressing schemes, but it allows for intermediate routers to fall back to IP based addressing if XIA is not supported.
Authentication and security	XIA identifiers are intrinsically secure: e.g. cryptographically derived from the 2 associated communicating entities in a principal type specific fashion [HADL12]. IDs are generated by hashing the public key of the destination entity.
Service advertisement	Services use a XIA socket API: a service calls bind() to bind itself to the public key of that service. This bind inserts the service ID

	in the host's forwarding table. Available services are propagated, following a given scope. Propagating this information is future research.
Naming scheme for service (instances)	XIA uses a restricted directed acyclic graph (DAG) representation of XIDs to specify XIP addresses. XIDs (principal identifiers) are retrieved by hashing the public key associated with that principal. These XIDs are unique for each principal instance.
Addressing scheme for service instances	There is no specific instance addressing. All identifiers are based on the public key of the service, and key pairs can be shared amongst trusted instances.
Service discovery and resolution	When a client wants to connect to the service, it first contacts the name resolution service to obtain the service address.
Service binding	The client initiates a connection by sending a packet destined to the service ID using the socket API. The source address specifies the client AS, Host ID and client service ID. This packet is routed to the destination AS and then to an instance of the service. After the initial exchange, both processes will establish a session, which includes, for example, establishing a symmetric key derived from their public/private key pairs.
Routing table population	Not mentioned.
Routing policy	XIA addresses contain several identifiers, such as AS, Host ID and service ID. A router starts at the service ID and falls back to the host address or AS address respectively if it does not support XIA. The next hop address is the furthest known node along the path.

Table 9: XIA and FUSION

Suitability for FUSION: XIA allows users to create new principals and specify how requests and data for that principal should be handled. In FUSION a similar approach can be used to support different policies for different types of services. The addressing scheme used in XIA might be useful for an incremental FUSION deployment on top of the IP infrastructure.

3.2.9 Summary: Models for Name Resolution and Network Control

In this section we summarise the discussion given in previous subsections by presenting a high-level overview of the way selected state-of-the-art architectures enable networks to be service aware.

Network resolution and network control models corresponding to the architectures selected in previous are presented in Figure 1. The figure has been drawn to expose the separation of transport functionalities from the control ones. Also, subtleties related to the directionality of data streams are shown. To this end, the data flows between service components (or users) are always annotated as red/blue bars on appropriate level. For example, in the case of IMS/NGSON two bars are used at the service session level to show in a symbolic way the allowance for multiple uni- and bidirectional streams that are coordinated within a single signalling "dialog" as one compound data session; in the case of Serval, the two bars correspond to two unidirectional flows that can be set up as a result of a single request (like two flows in a TCP connection). In contrast to the above, in the CCN case data flows (the chunks transferred in response to requests for consecutive pieces of a larger object) are always unidirectional because of the simple request-response network communication model in

CCN. In the case of IRMOS, the flow is marked as unidirectional, although ISONI may have capabilities to control also bi-directional connections (it uses ITUT G.805 network resource model for the transport network).

Another important aspect exposed in the figure is the way service-level control information is conveyed and seen by the network to support service resolution (and also network resource control). I the case of IRMOS, ISONI plays the role of a (logically) centralised controller responsible for reservation and interconnecting all resources (computation, network) based on service descriptions (in the form of virtual service networks in the IRMOS terminology). In IMS/NGSON, it is the role of out-of-band signalling to provide basic information on the service requested during the resolution process. Thus, there is a possibility for services to annotate their requirements themselves – using signalling; from the FUSION perspective this would mean that some information about the service that in IRMOS can only be described statically in the form of a service graph, now can be created and announced dynamically using signalling. Notice that this applies to Serval only partially, because currently Serval does not allow for rich negotiations like in SIP using SDP, and also due to its positioning in the OSI stack which is different than that of SIP. While SIP (and also Serval in the request/SYN phase) is out-of-band- and out-of-path signalling, CCN uses in-path signalling because requests and data packets follow the same route (in the opposite directions). This in-path signalling capability combined with service-name awareness (due to name-based routing) could be used by the forwarding nodes to monitor and/or control, e.g., quality of service, but respective policies would have to be set up and executed by some service control function which is not defined in CCN explicitly.

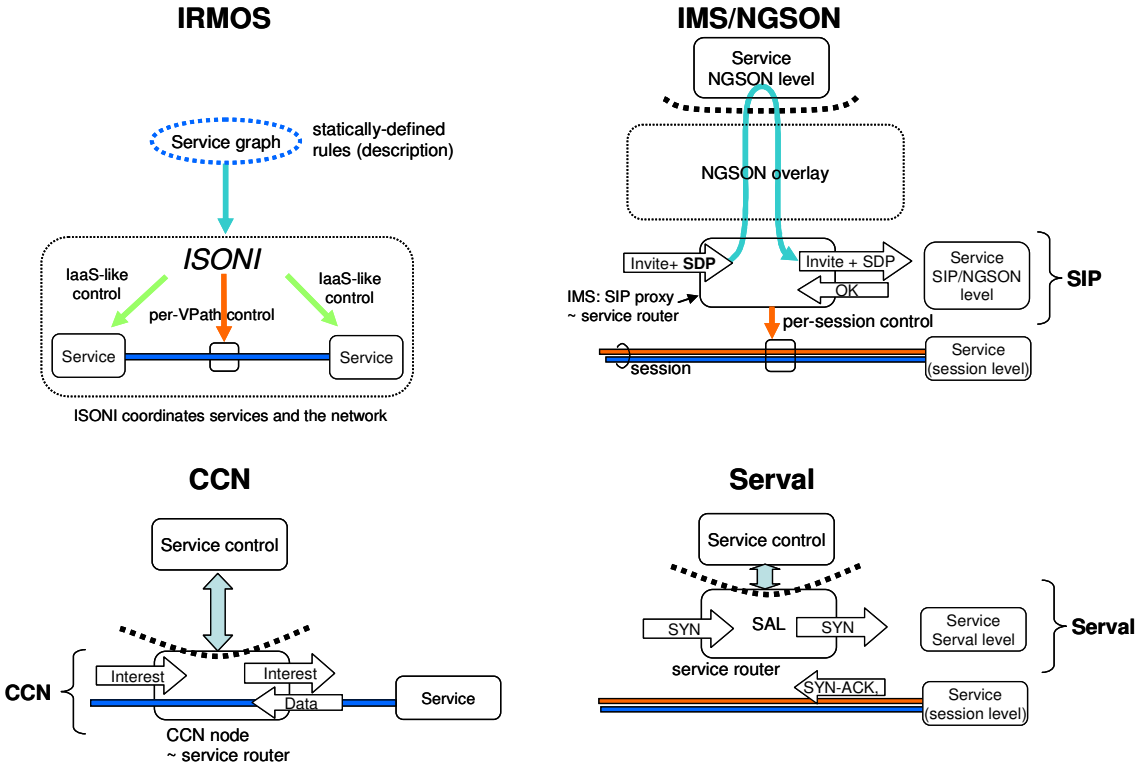


Figure 1: Models for name resolution and network control in selected architectures.

Analysing the models presented one can conclude that appropriate combination of those corresponding to IRMOS and IMS/NGSON (SIP/HTTP) seems to be a suitable starting point for subsequent FUSION developments. In particular, IRMOS provides overall coordination of computing and network resources based on explicit description of complex services (service graph/manifest in the form of VSN). At the same time, deriving information about services from signalling (mainly

during the resolution phase, easily achievable with SIP/HTTP) seems to be best suited for the case of highly distributed resources that FUSION makes case for.

3.3 Service Routing Node Design and Operations

The Service Routing layer enables the discovery and selection of an instance of a particular service. The selection process may be based on application-specific and/or network metrics.

The network layer transports service requests, monitoring information and routing table updates. To ensure the evolvability of the proposed solution, the FUSION network layer is conceived as an overlay of today's IP stratum. A high-level view showing the main functionality of the FUSION network is shown in Figure 2.

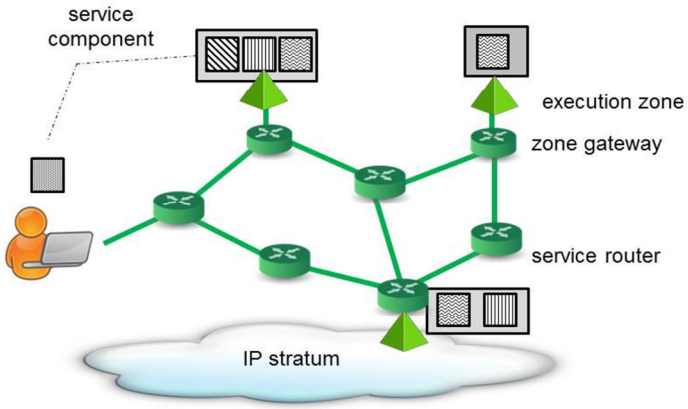


Figure 2: Service components running on clients and in execution zones are interconnected through the FUSION network as overlay of today's IP stratum.

The main component of the Service Routing Layer is the Service Router, responsible for the instance selection of name-based service requests by service components. The zone gateway is the entry point for zones, but basically performs the same operations as the other service routers.

The goal of this section is to consider the design of the service routers, and to demonstrate their mutual interworking, as well as their interfaces with other components to realize the functionality of a service-oriented network layer.

3.3.1 Service router

The functional blocks of the FUSION Service Router are illustrated in Figure 3.

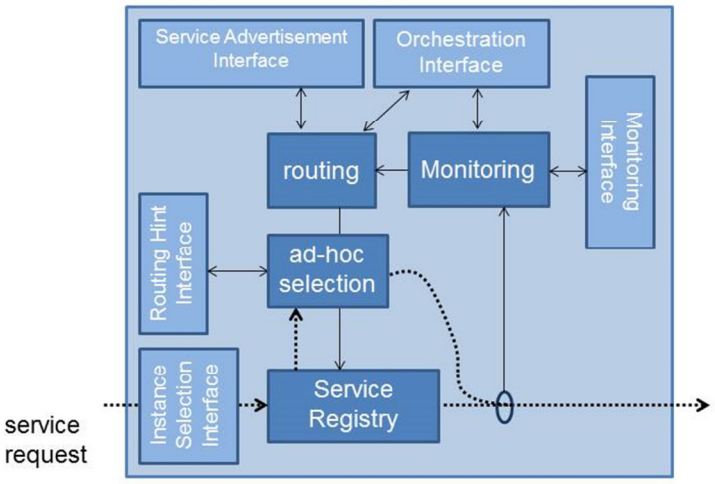


Figure 3: functional blocks and interfaces of the service router

Incoming service requests are processed by looking up the requested service name in the Service Registry. This table maps service names to specific actions. If an entry is found in the Service Registry, the request will be directly forwarded to one or multiple outgoing interfaces.

If the service name is not found in the Service Registry, the request is escalated outside of the fast-forwarding path of the router to the ad-hoc selection component. This component may query external components, such as the name resolution service before determining the outgoing interface. More details on retrieving routing hints are provided in section 3.3.4.

The Service Registry is being updated by the routing component. Routing paths are calculated based on information retrieved from other routers via the service advertisement interface, or from the Orchestration Layer. Moreover, monitoring information can be taken into account for dynamic updating of the routing entries in the Service Registry. The routing component thus plays an important role in the overall optimisation of the service instance selection process in the FUSION architecture.

3.3.2 Zone gateway

The zone gateway abstracts the proprietary management policies of the Execution Zone from the FUSION service routing layer. Its main functionalities are instance selection and service advertisement. The minimal set of components and interfaces of the zone gateway is shown in Figure 4. It contains a routing component that is responsible for advertising services hosted by the zone. Such advertisements are issued based on the information provided by the proprietary Zone Manager. The functionality of the Zone Manager is further described in D3.1

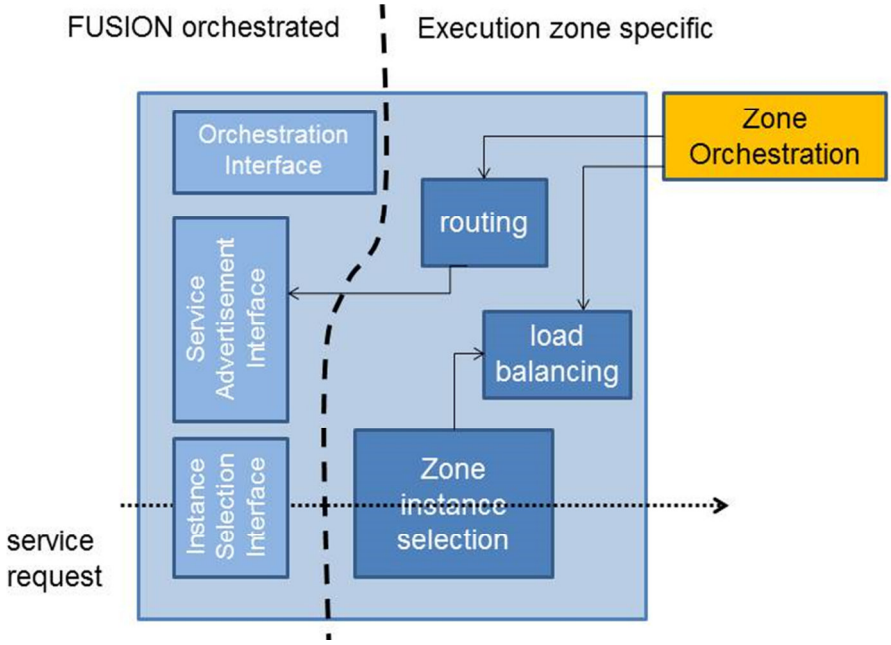


Figure 4: functional blocks and interfaces of the zone gateway

Whereas the 'core' service routers select the optimal *zone* for serving a particular service request, the zone gateway will map incoming requests to actual *instances*. Several scenarios can be identified:

- At least one instance of the requested service is already running. The zone gateway performs internal load-balancing without exposing the actual number of instances running in the Execution Zone to the FUSION service routing layer. The owner of the zone could apply any proprietary load-balancing policy.
- No instance of the requested service is running. The gateway temporarily stalls the requests and asks the Zone Manager to instantiate a new instance, before forwarding the request.
- For stateful services with data forwarding handled by the FUSION service routing overlay (instead of IP), the service identifier in the request will indicate the instance to forward the request to.

The zone gateway will also advertise the available services. Without exposing too much internal details, the zone gateway will provide information about:

- The services offered in the zone. Note that the announcement of an available service does not imply that an instance of that service is actually running. It merely indicates the availability of resources to host an instance.
- The number of session slots. Session slots are introduced in D3.1 as an abstraction of per-service resource availability.

3.3.3 Service instance selection

The process of service instance selection involves the translation of a location-independent service name into the locator (or sometimes into the identifier) of an actual instance of the requested service. The selection process is gradually carried out by name-based forwarding of service requests between FUSION network routers, as illustrated in Figure 5.

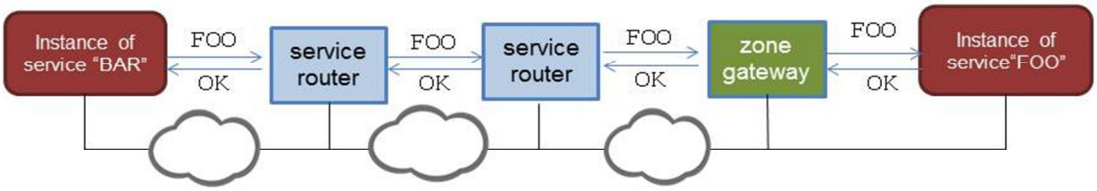


Figure 5: Forwarding of service requests by FUSION routers.

As explained in section 0, the actual selection between the many candidate instances of a particular service may be carried out by the requesting service, or by the FUSION network. The network-driven approach was already illustrated in Figure 5: the network autonomously decides to which instance a service request is routed.

In the *client-driven* approach, the FUSION network is only queried about available instances. Routers forward the discovery request to one or more outgoing interfaces, and/or respond themselves with a list of known instances found in their service registry. Zone gateways advertising the requested service may also respond to the request. This approach is illustrated in Figure 6.

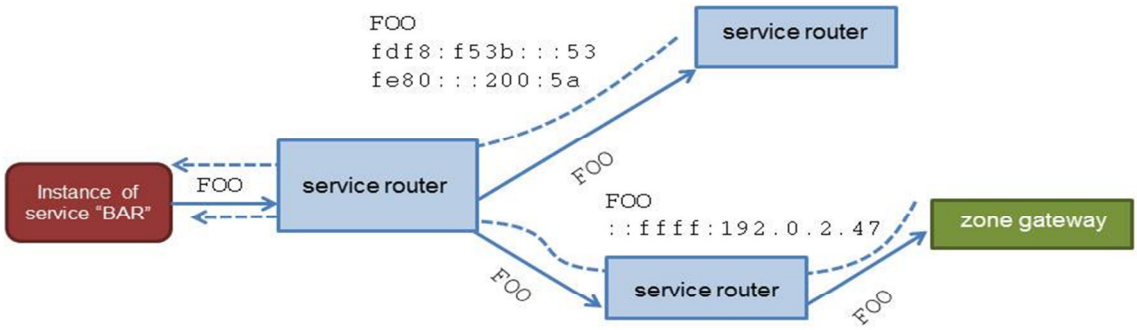


Figure 6: Client-driven selection

The requesting service will construct a list from all responses received during a client-specific waiting period. Although in Figure 6 only IPv6 addresses are returned, several types of identifiers and locators can be returned:

- service instance identifiers (FUSION namespace)
- service instance locators (IPv6)
- zone identifiers: zone gateways may respond with their locator, deferring the load balancing until the actual request arrives

The requesting service will eventually connect to one instance from the list. Session management is discussed in section 3.3.6.

3.3.4 Service request forwarding operations

The service registry of each service router will contain per service name the IPv6 address of the next node involved in the selection process: the next FUSION router, or the address of a zone gateway. How this forwarding table is populated will be discussed in section 3.3.5.

Name-based routing is only one mechanism to aggregate and cache the information of service availability across execution zones. If no entry is found, the selection decision is escalated outside the fast-forward path to perform ad-hoc resolution. Requests will be temporarily stalled, and the ad-hoc selection component may query external components for *routing hints,* as shown in Figure 7.

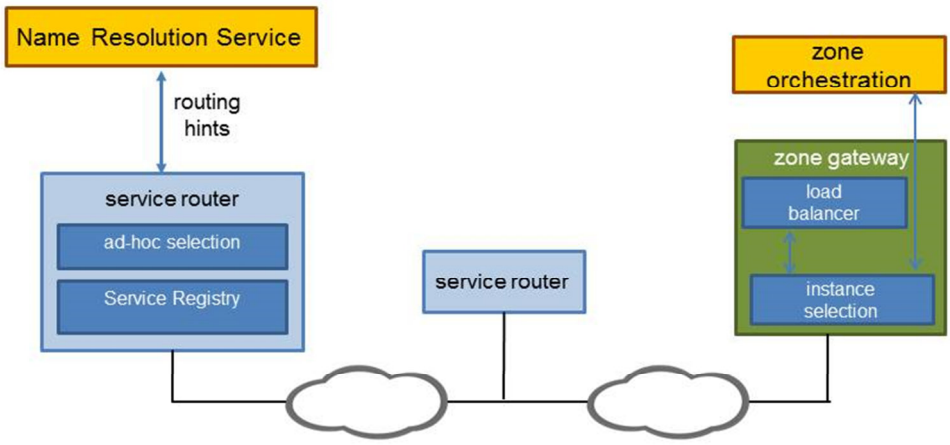


Figure 7: FUSION routers and zone gateways may query external components for routing hints.

External components delivering routing hints may be a name resolution service, or the FUSION Orchestration Layer. Name resolution service may provide routing hints like the locators of next service routers to forward the request, or even the actual locator of the instance to be selected. Zone gateways receiving a service request may escalate the instance selection logic to the load balancer of the zone, or to the zone orchestration layer. If needed, the orchestration can first instantiate a request and subsequently send the instance address as routing hint back to the zone gateway.

The service routing plane forwards and resolves service requests to a particular zone that announces the requested service. The intra-zone selection mechanism is proprietary and left to the Zone Owner. The concept of *late-binding* is hence a two-stage process: the network first selects the most appropriate zone, and the zones will internally load-balance the request to one of the instances.

3.3.5 Service Announcement and Routing

To populate the service registry, service routers need to be provided with information about service availability per zone. For this reason, each zone will advertise services to the routing domain it is attached to. This information may be aggregated and forwarded between all routers of the same domain.

If a user requests a service that is not being advertised by one of the zones of its routing domain, the request must be routed to other routing domains. Routing domains will peer through their border routers that run a standardized inter-domain routing protocol, akin to BGP.

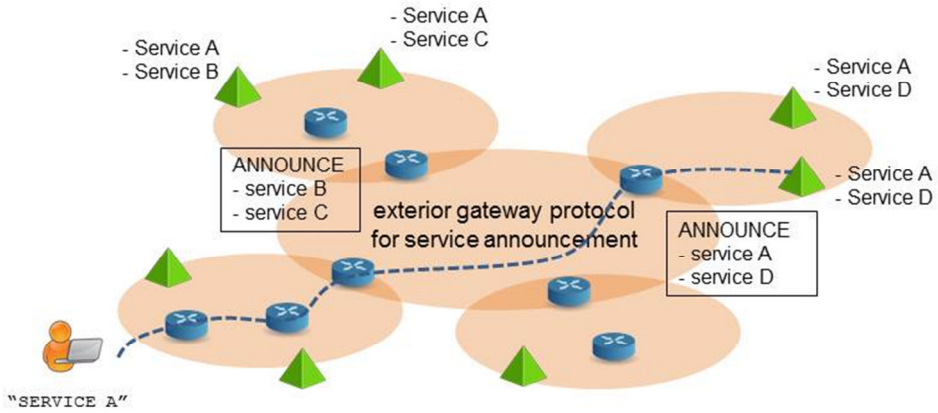


Figure 8: Service availability is exchanged between domains, enabling cross-domain forwarding of service requests.

Figure 8 shows a sample scenario, where a few routing domains are globally interconnected via a (currently undefined) FUSION exterior gateway protocol. Each domain announces its services to other domains. Business rules may prohibit the announcement of hosted services to other domains, for geographical reasons or to shape incoming traffic requests. As an example: Netflix may not be reachable from other countries.

3.3.6 Session set-up

After service instance discovery and selection, the service components will establish an application session. This application session may be routed through the Service Routers or natively in the IP layer. In any case, service components must be able to add service-specific information to the discovery request that is sent on the network.

3.3.7 Monitoring

Service routing decisions may in general depend on two types of monitoring information, namely the information about the transport network and the information about service instances.

In fact, the types and the amount of information about the network gathered by the monitoring component can depend on the algorithms used in a particular implementation of the FUSION architecture. Typical parameters will probably include bandwidth and latency. But the best practices that suggest the options particularly suitable for FUSION may only be created over time. At the current stage of the project it is thus sufficient to assume that monitoring of the transport network for the needs of service routing will be accomplished in accordance with known frameworks and using available network monitoring tools.

Apart from network parameters like bandwidth and latency, the most important metric related to services is the availability of and the load on the instances. However, this information might not be available in all parts of the network (e.g. because of administrative boundaries).

The return path of the answer by queried service instances could differ from the original request path. When forwarding a service request, service routers could add their own address to the request header, indicating that the answer should be routed back via these routers. This information could provide valuable additional input to service routers about the decision policies in other parts of the network.

Based on the above one can see that monitoring service level parameters for the needs of service routing can in theory be accomplished in three basic ways: by piggybacking this information in signalling messages (responses to service requests), or piggybacking it in service routing protocol announcements, or using a dedicated protocol for service monitoring at the service routing layer. We believe that additional studies and practical experience are needed to decide on the suitability of

these forms for FUSION. In the current stage of the project it is thus sufficient to assume the existence of an abstract monitoring interface that allows the exchange of required service monitoring information for the purposes of service routing.

4. INTERFACE AND PROTOCOL SPECIFICATION

Based on the above discussion, the following interfaces can be identified:

- service query and invocation interface: to discover and possibly select service instances
- service announcement interface: to announce service availability
- routing hint interface: to ask non-service routers for routing hints
- monitoring interface

The specifications below should be seen as a first version. In the second year of the FUSION project, we will study and refine these interfaces.

4.1 Service query and invocation interface

This interface is used by service instances (running in zones or on client devices) to query the Service Routers for suitable instances of the requested service. A similar interface is needed between routing domains for multi-domain service invocation.

Figure 9: query and service invocation interface

The protocol used for service query and invocation must support the following operations:

- **discovery** of instances by name
- **selection** of instance by name
- **parameterization** of requests, adding additional constraints for discovery or selection

For this interface, we propose a protocol that is akin to HTTP, with several mandatory header fields.

QUERY message: allows a service to query for a list of service instances available in the network, or to ask the network to select one instance.

```
QUERY URI
query-id: [XXX]
notify-id: [XXX]
filter: [XXX]
invoke: [bool]
[ ext ]
```

- `URI`: the identifier of the requested service, following the naming scheme specified in section 4.2.
- `query-id`: randomly generated to match `query-resp` messages
- `notify-id`: ID of the entity that wants to be notified of the response
- `filter`: JSON-encoded string of additional arguments narrowing down the service selection, readable for all service routers
- `invoke`: boolean; true if the network is allowed to select only one instance
- `[ext]`: optional information

Note: Service routers may rewrite QUERY messages by adding a `notify-id`: header field to indicate that they want to be notified of the responses.

Responses are returned via **QUERY-RESP** messages.

```
QUERY-RESP URI
orig-query-id: [XXX]
notify-id: [XXX]
instances: N
N x instance-id
```

- `orig-query-id`: matching the query-id from the initial query message
- `instances`: the number of discovered service instances listed in this message
- a list of instance-id's

Note: if additional `notify-id`: header was added to the QUERY message, these headers must be copied by the responding entity into the QUERY-RESP message. Service routers receiving the response must remove their `notify-id` entry from the message and forward to the next router in the Via: header. This may be the client itself.

4.2 Naming

At this stage, we propose to adopt the flat naming scheme that was initially developed by NetInf and is now in standards track as RFC 6920 [RFC6920].

A generic URI example is provided below:

```
ni://example.com/sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk
```

The three main components of the URI are:

- Authority (e.g. `example.com`) is an optional field. According to RFC 6920, it may assist applications in accessing the object. In the scope of FUSION, the Authority field could be used to indicate the developer of the service (e.g. spinor.com)
- Digest algorithm: the name of the digest-algorithm, as specified in the IANA registry.
- Digest Value: encoded using the base64url encoding

Although the authority field is optional, it could be useful for FUSION in the following scenarios:

- aggregation per service developer of routing hints or zone service availability advertisements
- the authority field could help to identify the authority for name resolution of a particular service provider. The service provider could host its own resolution service and hence maintain full control over the resolution process.

Optionally, the URI could be extended with query parameters as a "tag=value" list of optional query parameters, separated by a '&' character. The drawback of this approach is (limited) extra processing in the service routers.

FUSION will further study which is the best format to provide additional parameters to a service query: as URL parameters, or as an additional header field.

4.3 Service announcement

Zone gateways will inject service announcement messages into the service routing plane. These messages will contain information about the available services for which the zone is ready to serve requests. Additional attributes could be added per service. One notable example is the session slots indicating the relative load or resource usage of a particular service.

Header (type, length, CRC…)
Zone ID
Withdrawn Service #1
…
Withdrawn Service #N
Service #1 + attributes
…
Service #M + attributes

These messages will be flooded between all service routers of the same routing domain and will allow routers to update their Service Registry. In the above, we assume that each router has the routing information per Zone ID. This routing information could be constructed using (and extending) existing protocols such as OSPF.

As explained in Figure 8, service announcements will have to be forwarded intra- and inter-domain. The study of the inter-domain service announcement has been scheduled in Year 2 of the project.

4.4 Routing Hint Interface

When no match is found in the Service Registry, Service Routers and Zone Gateways contact a Routing Hint Provider (RHP) for mappings from service identifiers to routing hints. Depending on the RHP contacted, the provided routing hints could either map to a locator or to an identifier of the next service router.

Routing Hint Providers that translate service identifiers to locators could leverage the DNS, either by defining a new DNS record, or by using Service (SRV) records to link to a dedicated resolver [Ahlgren2013, SAIL13]. Also the load balancer in an Execution Zone could be seen as a Routing Hint Provider, selecting one instance per request.

Routing Hints can also be used to compress forwarding tables [NARA12]. Topological (locator) routing hints are used as labels in routing, allowing to compress the routing tables topologically.

4.5 Monitoring Interface

In fact, the information (at least its semantics) about service instances and related resources conveyed within service routing layer through the monitoring interface can be similar to (or derived from) relevant information available through Amazon Cloud Watch API or OpenStack Ceilometer API. It can be assumed that this information can be exchanged using JSON format or XML, and the final decision which one to use will be taken in the next steps of the project along with other decisions concerning the development of the FUSION demonstrator.

More details on the monitoring interface are provided in D3.1.

5. NETWORK-DRIVEN SERVICE SELECTION

In this section, we assume that the FUSION orchestration logic has already deployed service instances in a subset of execution zones. Service load is abstracted as the queue size, and could be seen as a first approximation of the session slot concept introduced in D3.1.

The proposed algorithms are a first approach to optimal selection between zones and routing of service requests according to this selection strategy. We assume a large number of execution zones distributed in a single routing domain. The network-driven service selection algorithm maps user demand to a fixed set of service instances, taking into account both network latency and server queue size. The selection is then mapped to a multipath name-based routing protocol to perform load balancing at-runtime on service routers. Simulation results show that this approach results in lower average system response time perceived by the users in comparison to a greedy algorithm and fair share approach.

5.1 QoS-aware service selection for low-latency applications

We propose QuLa, a Service-Centric Networking (*SCN*) architecture which combines Information-Centric Networking and Cloud Computing for large networks. Our goal is to provide more natural service access leveraging name-based routing. QuLa extends the ICN principles on named-data to a service-centric paradigm, combined with network-driven service management algorithms.

Users request a service only by name and in-network load balancing techniques route requests towards the instance which gives the best QoS. Our research focuses on minimizing the total system response time, which is the sum of forwarding (network) latency, queuing delay at the service instance and the actual service processing time.

We aggregate user demand at network edge nodes and seek an optimal distribution of the request load across the deployed instances. Service routers forward and load balance user demand over multiple service instances. Embedding the optimal load resolution to the forwarding tables of the service routers would require source-based routing. However, source-based routing excludes the possibility of routing table aggregation, making this approach practically infeasible. Therefore, we study the degradation in system response time when service routers only perform a statistical load-balancing, not taking the source of the request into account.

5.2 Related work

QuLa is related to several areas of research as it addresses service selection, routing and both ICN and SCN. In this section we briefly discuss work relevant to QuLa for the different aspects of our architecture.

Information-Centric Networking. ICN received a great amount of attention from various research groups. Some of the most significant contributions are made by CCN/NDN [NDN10], PSIRP/PURSUIT [FNTP12], DONA [KECK07] and NetInf [DKOF13]. Despite all focusing on content, they differ in design choices and resulting solutions. CCN, for example, uses a tightly coupled routing scheme, where name resolution and packet forwarding is performed by the same components. PSIRP/PURSUIT on the other hand, leverages a publish-subscribe mechanism called a rendezvous point. DONA opted for a flat naming scheme while CCN relies on hierarchical naming to improve scalability. For more information on these ICN architectures we refer to a case study in [ADIK12]. QuLa does not focus on naming as this can be adopted from previously established work in above mentioned ICN architectures.

Service-Centric Networking. ICN architectures do not consider service requirements and constraints, rendering them less suited for service usage. This ICN limitation led to an increasing amount of research aiming to surpass ICN for more efficient service usage, commonly named SCN. T Braun *et al.* addressed common problems in ICN for service support [BHHR11], which are in line with our design considerations presented in the next section. One of the main research challenges in SCN is setting up connections between end-hosts for service sessions without prior knowledge about the host addresses.

Serval [NDGK12] proposes a Service Access Layer to translate service names to instance addresses, while simultaneously setting up the TCP connection between both end-hosts. Serval focuses on

services running on mobile devices and aims to support late-binding and service migration. Despite making great contributions towards multi-homed services and seamless relocation of both users and services, Serval does not discuss service lifecycle management nor does it provide a routing protocol.

SCAFFOLD [FAGK10] emphasizes on a flow-based anycast mechanism, allowing multiple instances to be addressed by the same service name. However, the suggested approach relies on underlying virtualization and changes to the existing network stack. SCAFFOLD does not focus on routing, thus not providing any routing protocol or service selection algorithm.

Routing and service selection. Akamai [NSS10], one of the largest CDN providers, uses a centralized selection algorithm to map user demand to servers located in the network edge. Although Akamai leverages different metrics and a different optimization problem, it proves that a centralized mapping scheme is feasible. Akamai first performs high-level cluster mapping followed by low-level server selection. QuLa deviates from Akamai's approach by utilizing one selection level in several smaller domains.

DONAR [WJFR10] focuses on selection for Cloud services and presents a decentralized mapping scheme considering both load and latency as metrics. This work describes an interesting API for users to define cost functions used by the mapping nodes. QuLa presents a similar problem statement as DONAR but solves this for time-sensitive services by minimizing the response time. Also, DONAR solves mapping between clients and server replicas whereas QuLa maps between clients and service replicas, considering the possibility of several services running on one server.

SoCCeR [SSRV11] is a decentralized routing protocol for services built on top of CCNx. Routing in SoCCeR is based on Ant Colony Optimization, a decentralized and probabilistic optimization heuristic which configures the Forwarding Information Base of CCNx nodes. However, this decentralized approach requires periodic *Interest Ants* to be sent out on every link in the network. We argue that metrics affecting service time change often and require frequent monitoring, making a decentralized monitoring scheme sensitive to high network load.

In comparison, QuLa extends ICN with service-centric algorithms and does not require changes of the existing IP stack. The network-driven algorithms, presented in the following sections, assume a hybrid approach which combines centralized service selection with a decentralized routing protocol. Using this approach we aim to minimize monitoring overhead while maintaining a scalable approach.

5.3 Load distribution

We envision network management components containing a reactive loop running frequently, responding to changes in user demand or service load by reconfiguring the routing tables to a fixed set of instances. In the next section we describe the assumptions made by our first approach algorithm. Next, we present the problem statement and objective to be minimized. We describe the QuLa algorithm implemented as a meta-heuristic and compare with several alternative approaches.

5.3.1 Assumptions

The service selection algorithm maps user demand to available service instances. However, considering every client individually is not feasible when load-balancing must be done in real-time. Therefore, we aggregate the load by a group of users located in a nearby geographical into an aggregated load from client node *i*. Execution zones are represented as server nodes *j*, with a queue for incoming requests.

Next, we assume a fixed service placement: the number of running instances per execution zone is assumed to be constant. We further assume a fixed service execution time; the execution time depends on many factors, such as hardware, influence of other processes on the system and the service itself. The problem statement presented in section 5.3.2 allows use of any model to represent service execution time. However, our first approach uses the M/D/1 queuing theory model and

assumes a fixed execution time for all service requests. Last, we assume a fixed user demand. Dynamic demand requires input from feedback to learn from previous decisions, which is not considered in this first approach algorithm.

In future work we will expand our selection algorithm to support dynamic demand patterns using feedback.

5.3.2 Problem statement

Consider a network graph containing edges *E*, nodes *N* and services S. The lambda values $\lambda(i,s)$ are the requests per second from client node *i* for service *s*. Nodes with belong to the client nodes $N_c \subset N$. Nodes hosting at least one instance of a service $s \in S$ belong to the server nodes $N_E \subset N$. The distribution matrix *R(i,j,s)* ∈ [0,1] denotes the fraction of user demand from client node *i* for service *s*, to be processed in server node *j*. A service takes $T_{j,s}$ seconds to process a request for service *s* on server node *j*, not considering queue delay.

A generic objective to map client demand to service instances, using demand fractions, is the following:

$$\sum_{i \in N_c} \sum_{j \in N_E} \sum_{s \in S} cost(i,j,s) * (R(i,j,s) * \lambda(i,s))$$

The objective in the current QuLa implementation is to optimize the average system response time given a fixed service placement and fixed user demand. There are two major factors that affect the response time of a service: the time spent on the network and the time spent on the server. Servers processing a larger demand have more impact on the average response time.

Taking into account the above, we find the following objective:

$$\min \frac{\sum_{i \in N_c} \sum_{j \in N_E} \sum_{s \in S} (T_{Latency} + T_{processing}) * (R(i,j,s) * \lambda(i,s))}{\sum_{i \in N_c} \sum_{s \in S} \lambda(i,s)}$$

The first part of the objective function represents the response time of a single request, considering the network latency and the estimated queuing delay and processing time. The second part represents the partial user demand sent to the service, which is taken into account when calculating the contribution to the average system response time. Finally, the numerator is normalized by dividing with the total user demand.

$T_{Latency}$ is the Round Trip Time (RTT) between the client node *i* and server node *j*. The objective supports use of any path between client and server nodes. $T_{processing}$ denotes the time spent on the server, including queue delays and execution time.

We illustrate this with an example assuming that our system is an M/D/1 queuing system. We find $T_{processing} = \frac{(2-\rho)}{2*(1-\rho)} * T_{j,s}$, where ρ denotes the total incoming request rate on node *j* divided by the outgoing rate. Considering the influence of $R(\mathrm{i,j,s})$ we find with $1/T_{j,s}$ being the outgoing rate. The objective implies that nodes which do not send any requests for a service (e.g. $\lambda(i,s)$ is zero) will not contribute to the system response time. Also, the response time between node *i* and node *j* only contributes to the system response time if $R(\mathrm{i,j,s}) > 0$.

To guarantee that the demand of each client node is completely satisfied, the objective is limited by the following constraint:

$$\forall i \in N_c, \forall s \in S: \ \lambda(i,s) > 0 \rightarrow \sum_{j \in N_E} R(i,j,s) = 1$$

Constraint 1: all client demand must be satisfied

In the following sections we describe several alternative algorithms to find a load distribution matrix R.

5.3.3 Algorithm outline

Name resolution via DNS typically does not consider the server capacity or load induced by other clients in the network. Several clients may receive the same IP address for a selected name, and hence may overload the server.

A better approach is a greedy algorithm which prioritizes client-server pairs with the least latency. In each iteration, the algorithm picks the client-server pair with the least network latency and assigns as much requests as possible from that client to the corresponding server, without exceeding the server capacity. We iterate through client-server pairs until all user demand is satisfied.

This approach is very intuitive, assigning most demand to the client-server pairs which induce the lowest response time. However, by prioritizing a certain client-server pair, we indirectly penalize other clients which do not get to use this server's full capacity anymore. This implies that a non-greedy decision for client-server pairs with low latency can have positive effects on the response time of several nearby clients.

The greedy algorithm always induces high workload on few servers while more distant servers remain idle most of the time. An alternative, algorithm 2, is to give each server an equal share of user demand, trading reduced workload for increased network latency.

The greedy algorithm assigned large user demand to few servers while algorithm 3 attempts to assign little user demand to all available servers. It is likely that the optimal solution, which minimizes the objective, is located somewhere in between. However, the distribution matrix R in the objective function takes floating point numbers as value, creating an infinitely large solution space.

We leverage a Simulated Annealing meta-heuristic to find a plausible solution in near real-time. Simulated Annealing does not guarantee to find an optimal solution in a finite amount of time. Instead, it searches through a large solution space in a short timeframe, inspecting multiple local minima on the solution curve. For a more detailed explanation of Simulated Annealing we refer to online sources (Simulated Annealing - Wikipedia).

5.3.4 Evaluation

Using Brite [BRITE13] we generate four types of network topologies: (1) small topologies with sparse connectivity, (2) small topologies with dense connectivity, (3) large topologies with sparse connectivity and (4) large topologies with dense connectivity. For each topology type we generate several sample networks, assign client-server roles, place service replicas, and generate user demand patterns. Using the average objective of these simulations for a set of fixed load values, we obtain reliable data required to make a confident assessment of the quality of each algorithm.

5.4 Statistical load-balancing in service routers

When demand is high, user requests must be load-balanced over multiple service instances. This is reflected in the selection algorithm by using a load distribution matrix *R*, assigning partial demand to available instances. At runtime, service routers load-balance user demand utilizing the fractions assigned in the load distribution matrix. In this section we describe the configuration process of a statistical load-balancing in the service routers to approximate the service selection.

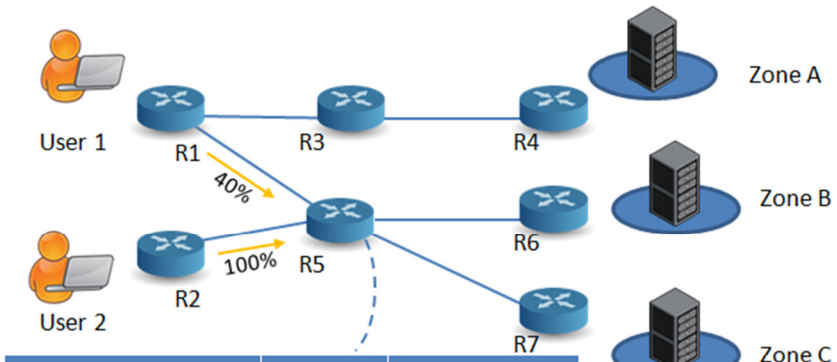
We propose for routers to know only the address of the next hop towards the service instance. Paths between end-hosts can be stipulated using custom policies, and each service router is involved in the selection process by forwarding to the next hop according to its forwarding table. Routers like R3-R7 in Figure 10 need to take into account the source of the service request to determine the outgoing interface (*source-routing)*. As this inflates the routing tables, we seek an approximation by a statistical load-balancing based on the optimal load distribution. An important research question is how much this approximation degrades the overall system response time.

We investigate two variants to tackle this problem.

Variant 1: source routing. We configure service routers with separate forwarding entries for each source address to reflect the service selection. Consider $\lambda_{i,s}$ the total demand from user *i* for service s, $R(i,j,s)$ the percentage of that demand to be processed on server *j*, $P_{k,in}(i,s)$ the incoming percentage of $\lambda_{i,s}$ on service router *k,* and $P_{kl,out}(i,s)$ the percentage of $P_{k,in}(i,s)$ forwarded to router *l* on router *k*. Initially, all P_{in} and P_{out} are set to zero.

The forwarding tables are configured as follows: (1) we stipulate a path for each client-server pair (*i,j*) and a given service *s*. (2) For each router *k* on that path, $R(i,j,s)$ is added to both $P_{k,in}(i,s)$ and $P_{kl,out}(i,s)$, where *l* is the next service router on path. (3) After all pairs (i,j) are traversed for service *s*, we express $P_{kn,out}(i,s)$ as fraction of $P_{k,in}(i,s)$, so all values are in range [0,1].

Figure 11 shows a sample configuration: 40% of user 1's demand is sent to R5, which forwards 50% to R6 and 50% to R7. Thus, 20% of user 1's demand reaches zone B and 20% reaches zone C, as per selection.



Service Name	Source	Next Hop
Serv-facedetection	U1	R6 (50%), R7 (50%)
Serv-facedetection	U2	R6 (20%), R7 (80%)

Figure 10: a first step towards true hop-based routing: source routing.

Variant 2: weighted average. Source routing is not scalable for many users and services due to the large forwarding tables this produces. We present a statistical method to realize true hop-based routing, using only service names as input.

Eliminating sources addresses by averaging the outgoing percentages $\sum_{i \in N_c} P_{kl,out}(i,s)$ does not suffice to reflect the service selection. We observe that outgoing demand on a service router *k* is most influenced by both larger λ_i and $P_{k,in}(i,s)$. However, using the method described in source routing, we find the total incoming demand for service *s* on router *k*, $D_{k,in}(s) = \sum_{i \in N_c} \lambda_i * P_{k,in}(i,s)$, and the outgoing demand to router *l*, $D_{kl,out}(s) = \sum_{i \in N_c} \lambda_i * P_{k,in}(i,s) * P_{kl,out}(i,s)$. Both incoming and outgoing demand already considers λ_i and $P_{k,in}(i,s)$. Thus, we eliminate source address *i* using a weighted average:

$$P_{kl,out}(s) = \frac{D_{kl,out}(s)}{D_{k,in}(s)} = \frac{\sum_{i \in N_c} \lambda_i * P_{k,in}(i,s) * P_{kl,out}(i,s)}{\sum_{i \in N_c} \lambda_i * P_{k,in}(i,s)}$$

where $P_{kl,out}(s)$ denotes the new outgoing percentages using only service name as input. Only $P_{kl,out}(s)$ is configured in the forwarding tables, enabling service routers to load-balance requests as dictated per service selection without considering source address.

5.5 Experimental results

We implemented a prototype by extending the open source CCNx code provided by PARC [CCNX13]. Contributions made include new packet types, periodic monitoring functionality, forwarding tables using probability forwarding and other features necessary for service usage. Our prototype was evaluated through emulations on the Virtual Wall [VWALL13] testbed. The prototype's evaluations confirmed the necessity of both the network latency and server queue sizes to be considered in the objective to minimize the average system response time.

In order to evaluate large network topologies we created a proof-of-concept simulation environment using CloudSim [CLOUDS13] a framework for modelling and simulation of Cloud Computing infrastructures and services. We extended CloudSim's data centre objects to add an internal service router and host exactly one computing machine per data centre. To simulate M/D/1 queue systems (1) requests are scheduled using CloudSim's 'Space Shared' Scheduler, (2) components are configured so that each request has a fixed execution time, and (3) client applications implement a Poisson process. Consecutive samples of a Poisson process follow an exponential distribution but produce an average demand converging to the input parameter.

We use Brite to generate four types of network topologies and retrieve several sample topologies from each type. Using the generated topologies and fixed service placement, service selection is performed by algorithm 3 and returns the load distribution matrix R. The CloudSim broker configures service routers residing in each data centre based on the load distribution matrix. In the final configuration step, the broker deploys the service process on each server node, ready to process requests.

To validate that each routing protocol correctly reflects the service selection load distribution matrix, we implemented three brokers and three data centre types; (1) the direct routing approach with requests going from client to server using IP routing, (2) the source routing approach (variant 1 in section 5.4), and (3) the weighted average hop-based routing protocol (variant 2 in section 5.4).

5.6 Future work and Research Challenges

While we are able to access services with minimal response time using name-based request, the first approach selection algorithm is still very constrained. Assuming a fixed execution time is not ideal and despite allowing different server models to be used, we still aim to develop our own model to deal with multiple services running on one server. This problem is more problematic than simply dividing the execution time by the amount of instances running on one system, as each instance could have a different base execution time or have input data affecting its execution time. Once we've establish such a model, the QuLa selection can support any combination of instances, running on various servers in the network.

The next step in our research is to create a dynamic selection algorithm. Currently, all demand is assumed fixed, and the service selection algorithm searches for the best possible distribution given a fixed set of instances. When user demand changes, the previous selection is subject to change, requiring reconfiguration of the network.

A naïve approach runs the static selection algorithm each time user demand changes. This approach is prone to oscillation (over-reacting to changes) and in a worst case scenario the system keeps returning to a previous state, only to repeat this pattern. To avoid circulating previous configurations and to become less sensitive to oscillation, feedback is necessary. We plan on developing a self-learning algorithm and implement control loop feedback. This allows use of prediction algorithms and trend detection, giving the network components a solid foundation to build decisions on.

Once the previous challenges are complete, we repeat this process for the service placement algorithm. We start with a static algorithm and a simple set of constraints. We then create a dynamic and self-learning algorithm to cope with dynamic demand and topology changes. We envision both service selection and service placement modules to cooperate, as it is likely that the need for relocation is established when the selection algorithm cannot find a feasible solution.

6. ROUTING ALGORITHMS

In this section we propose a first approach to discovering the closest running instances of a service. We assume a large number of execution zones distributed around the globe. The FUSION orchestration logic has previously deployed service instances in a subset of these execution zones – according to the application developer's deployment strategy, the predicted demand from users and the services' performance targets. Each service is addressed by *ServiceID* and when a user wishes to invoke a service she injects a request into the FUSION service routing plane to discover the N closest execution zones running the requested *ServiceID*. We assume in this initial version of the FUSION routing protocol that the service routers focus on returning the N best execution zones from a network performance perspective and that instance selection on any service-specific metrics is achieved in a second phase interaction between the requesting user and the N returned execution zones/service instances. The sequence of operations from the users' perspective is therefore:

1. A user issues a request to her local FUSION service router for N closest execution zones running the required ServiceID.
2. The FUSION routing plane operates as described below to return the locators of the N-closest execution zones.
3. The user (or the application running in the user's end device) may contact one or more of the N execution zones to refine the selection according to the load on the servers or on other service-specific metrics that can be queried from the execution zones/service instances. Alternatively the user/application may directly select one of the N execution zones without querying for additional information.
4. The user (or application) invokes the service at the selected execution zone.

We develop a hierarchical clustering technique and accompanying routing protocols that allow the creation and maintenance of geographically-sensitive service routing plane. For static services with a small numbers of instances running in a few execution zones, this can be done easily with a centralised system. However, in order for the system to appropriately scale for large numbers of execution zones and service instances, it becomes necessary to implement it as a distributed system, partitioning the resource information indexes and distributing the processing of resource discovery requests between the service routers. Traditionally, this is achieved using *Distributed Hash Tables* (DHTs, [MAYM02] [ARTI07] [ZOEL06]). However, due to their long search times, conventional DHTs are unsuitable for demanding multimedia applications [FALK07]. To address this, the presented system uses routing and forwarding algorithms similar to those of a DHT, but with additional optimisations aimed at increasing accuracy and speed for reaching geographically close execution zones.

In this system, users/applications use the system to directly send messages to a set of N *geographically close* execution zones running a specific *ServiceID*. Conceptually, the operation of the system is similar to that of *anycast*, which routes messages to the closest member of a given group, but generalising it to efficiently route messages to the N best options (in this case, the geographically closest ones).

6.1 n-casting Overview

We begin by defining the following entities:

- execution zone
- service group – the execution zones running instances of a specific ServiceID
- cluster
- service router
- leaf cluster

In n-casting execution zones join a service group by informing their local service router. Service routers are organized in a hierarchical structure (see Figure 11) which they use to disseminate service group membership globally (black lines in Figure 11). When a user wants to route a message to the N closest execution zones running a ServiceID it sends a message to its local service router which is part of the FUSION service routing plane. The message is routed through service routers (yellow arrows in Figure 11) and replicated if needed. The last service routers deliver the message(s) to the N closest group members.

In order to build the routing tables, the design of the n-casting system is centred around three main concepts:

1. a hierarchical structure of clusters of execution zones built based on their proximity;

2. a protocol for creating connections to exchange information about service group members available in these clusters;

3. an adaptive number representation system to aggregate service group membership and estimate the number of distinct members per ServiceID.

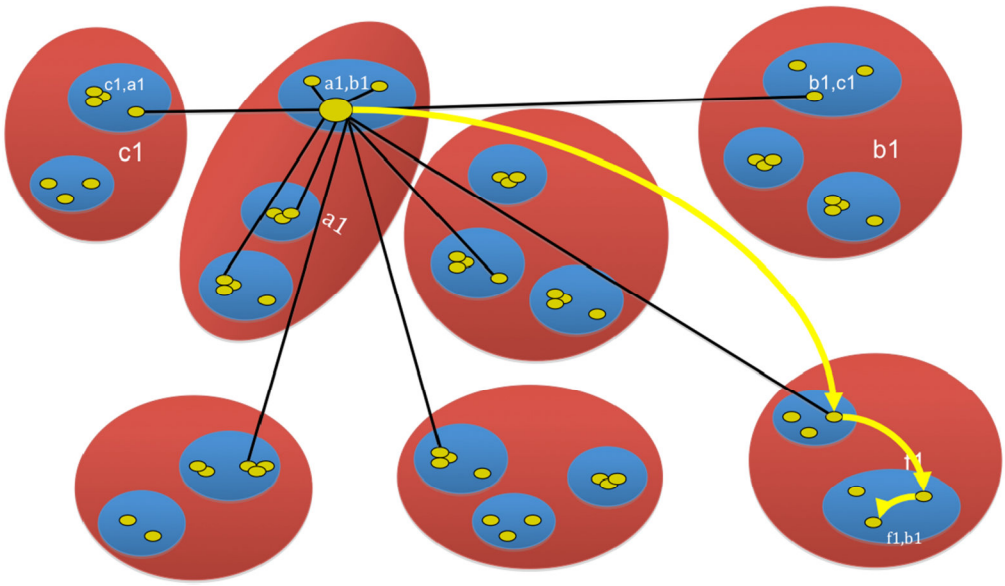


Figure 11: Message Forwarding in N-Casting

While the typical approach in structured overlays (e.g. DHTs) is to partition the key space of the resource identifiers, in n-casting we partition the surface of the earth following Internet node density. Execution zones are organised in clusters according to their geographic proximity. To limit the management cost for each one of these clusters, they are designed to contain a limited number of execution zones. This allows local servers, which shall be denoted as *n-casting service routers*, to maintain information about service group memberships within the cluster without the need for aggregation. These low level clusters are merged into larger clusters at several levels of hierarchy, allowing for exchanging information for areas of increasingly larger size. In is important to note that every service router has not only full information of the smallest cluster that they belong to, but also progressively more summarised information of both all the larger clusters that contain it and other

areas further away from it. This ensures that information about service group members is less aggregated in the most desirable areas where execution zones lie in closer geographic distance. Each service router maintains a view of service group membership for all groups in different areas, with greater level of detail for closer as opposed to more distant areas. Service routers then form an overlay network and exchange aggregated information about the service group membership of execution zones in their cluster. This information is captured in a *floating point* data structure that allows the estimation of how many distinct execution zones are members of the service group identified by that key.

6.2 Evaluation

Due to its impact on building high-performance overlays, the *accuracy* of message delivery is one of the most important performance parameters of the n-casting system. Informally, this is a measure of how far away from the optimum is the geographic locality performance of the n-casting system.

In the general test case, *queries* are issued by any node in the system requiring delivery to the N geographically closest representatives of a particular membership group. In order to test the system in widely different conditions, we consider groups with different *popularity*, i.e. with different total numbers of group members. Popularity has a direct impact on the accuracy of the n-casting messages results. To see why, consider a group with a large population of members (a highly popular group). Whenever any member of this group sends an n-casted message, appropriate recipients can be usually found in the same leaf cluster where the sending node is, or in close low-tier clusters. Conversely, for groups with small numbers of members (highly unpopular groups), members will be only be available several hops away, increasing thus the potential for errors introduced by the hierarchical clustering structure.

The number of update messages exchanged under different group popularity settings has been evaluated using simulations. The simulator models the view from a particular node, the update messages it sends to its remote connections to announce the membership fluctuations in its local cluster, and the update messages it receives as a result of membership fluctuations in the remote clusters it is connected to, hereby referred to as outgoing and incoming messages respectively.

6.3 Related Work

Anycasting, defined by RFC 1546 [PART93], proposed a method for service discovery. Work has been presented on making IP layer anycast viable through managing the routing of anycast groups. The classification of groups as being *local* and their frequency of use allows Katabi [KATA00] to cache popular routes and not waste resources on rarely used routes. The premise is that from a certain location certain services simply are not going to see the same level of usage.

To create an overlay that is geographically aware, IP geolocation has been shown to provide accurate, predictable results [FREE06] [AGAR09]. This has been confirmed not only by third-party datasets such as Peerwise [LUME07] and iPlane [MADH06], but also by our own extensive active measurements [LAND13a] [LAND13b].

Overlays such as OASIS [FREE06] require nodes to maintain multiple databases to obtain knowledge of overlay membership, popular groups and proximity data. Our n-casting system requires nodes to maintain a single data structure, with the overlay hierarchical clustering providing intuitive proximity and popularity information to nodes.

Hierarchical overlays have been proposed before however the reason for creating a hierarchy was to group nodes that had similar resource levels such as bandwidth, computational power and uptime in overlays. Evaluation of such overlays was conducted by Lu [LUWU07], Zoels [ZOEL06] and Artigas [ARTI07] showing that grouping nodes based on certain characteristics can improve overlay performance. The system proposed creates a hierarchy that does not group nodes by resources or

uptime, but rather, by their physical location; hence, higher tiers of the hierarchy provide an abstracted view of the geography of those tiers below.

Locality-aware hierarchical overlays have been shown to provide positive results. Experiments conducted by Xu [XUTAN03] showed as the number of RTT measurements increase, the latency stretch decreases when expanding ring search and hybrid landmarking algorithm plus RTT was used. The effectiveness of landmarking is highlighted by Xu [XUMIN03], where HIERAS, a hierarchical DHT exhibits lower routing latency with an increase in landmarking nodes.

N-casting can be also positioned as an operation that fills the spectrum of network communication space between anycast and multicast [CART03]. Previous efforts to introduce systems that had an element of n-casting was investigated by Leung [LEUN06] and Nguyen [NGUY04]. Leung's paracasting system aimed to deliver content concurrently that had been broken up into chunks, similar to that of Bittorrent, from multiple sources for content that is replicated on multiple servers. Neither Leung or Nguyen's proposals consider the overlay maintenance cost, sending to multiple members in the same anycast group and in the case of Leung, considers replicas stored on high-bandwidth connections.

7. CONCLUSION

This deliverable presented the results of the breadth-first approach that has been taken by the FUSION consortium for the Service Routing layer. This layer provides name-based, in-network selection between Execution Zones for service requests. This main selection primitive was crystallized into a set of functional requirements for naming, forwarding, routing, selection and monitoring. An extensive scan on service-aware networking approaches identified several design considerations.

The work plan in the second year is focused on making the architectural choices more concrete in terms of implementation and evaluation. Specifically, the following topics will be studied:

- routing scalability: given the huge number of services, how can routing tables be compressed? Lines of thought include an intelligent use of routing hints and compact routing schemes.
- selection agility: service availability is prone to rapid changes over time and space. A major research topic is how the in-network selection can quickly adapt to these changing conditions.
- interplay with the orchestration layer: an important consideration is the split of functionality between the service routing layer and the orchestration layer.

8. REFERENCES

[ADIK12] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, "A Survey of Information-Centric Networking", *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 26-36, 2012.

[AGAR09] S. Agarwal and J. Lorch. Matchmaking for online games and other latency-sensitive P2P systems. *ACM SIGCOMM Computer Communication Review*, 39(4):315–326, 2009.

[AHLGREN13] Bengt Alghren, NetInf global connectivity – routing and forwarding http://www.ietf.org/proceedings/interim/2013/02/14/icnrg/slides/slides-interim-2013-icnrg-1-1.pdf

[ARTI07] M. Artigas, P. Lopez, and A. Skarmeta. A comparative study of hierarchical dht systems. In *Proc. of IEEE LCN*, pages 325–333. IEEE, 2007.

[BANE06] A. Banerjee and J. Ghosh. Scalable clustering algorithms with balancing constraints. *Data Mining and Knowledge Discovery*, 13(3):365–395, 2006.

[BARI12] M. Bari, S. Chowdhury, R. Ahmed et al., "A Survey of Naming and Routing in Information-Centric Networks", IEEE Communications Magazine, December 2012.

[BHHR11] T. Braun, V. Hilt, M. Hofmann, I. Rimac et al., "Service-Centric Networking", *Communications Workshops (ICC), 2011 International Conference on*, 2011.

[BRITE13] "BRITE: Boston university Representative Internet Topology gEnerator," [Online]. Available: http://www.cs.bu.edu/brite/.

[CAJF11] J. Cheny, M. Arumaithuraiy, L. Jiao ; X. Fu, K.K. Ramakrishnan, COPSS: An Efficient Content Oriented Publish/Subscribe System, 2011 Seventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2011) , New York, 2011.

[CART03] C. Carter, S. Yi, P. Ratanchandani, and R. Kravets. Manycast: exploring the space between anycast and multicast in ad hoc networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 273–285. ACM, 2003.

[CART12] A. Brodersen, S. Scellato, and M. Wattenhofer. YouTube around the world: geographic popularity of videos. In *Proceedings of WWW*, WWW '12, pages 241–250, 2012.

[CHA08] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain. Watching Television Over an IP Network. In *Proceedings of ACM IMC*, pages 71–84, 2008.

[CIA] The World Factbook. Country comparison: Internet users. https://www.cia.gov/library/publications/the-world-factbook/rankorder/2153rank.html.

[CLOUDS13] The CLOUDS Lab: Flagship Projects – Gridbus and Cloudbus," The University of Melbourne [Online]. Available: http://www.cloudbus.org/cloudsim/

[CMA07] Camarillo, Gonzalo, and Miguel-Angel Garcia-Martin. The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds, Wiley, 2007

[CNNX13] Project CCNX, PARC [Online] http://www.ccnx.org

[CUEV10] R. Cuevas, N. Laoutaris, X. Yang, G. Siganos, and P. Rodriguez. Deep diving into BitTorrent locality. *SIGMETRICS Perform. Eval. Rev.*, 38(1):349–350, June 2010.

[DKOF13] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, "Network of Information (NetInf) – An Information-Centric Networking Architecture", *Computer Communications*, vol. 36, no. 7, pp. 721-735, 2013

[FAGK10] M. J. Freedmann, M. Arye, P. Gopalan, S. Y. Ko et al, "Service-centric networking with Scaffold", Princeton Universities, 2010

[FALK07] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson. Profiling a million user DHT. In *Proc. of ACM SIGCOMM IMC*, USA, 2007.

[FNTP12] N. Fotiou, P. Nikander, D. Trossen, G. Polyzos, "Developing Information Networking Further: From PSIRP to PURSUIT", *Broadband Communications, Networks, and Systems*, vol. 66

[FREE06] M. Freedman. OASIS: Anycast for Any Service. In *Proc. of Symp. on NSDI, USENIX, 2006*, 2006.

[GEONAMES] GeoNames. http://www.geonames.org, 2013.

[GUTT84] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. of the ACM SIGMOD*, 1984.

[HAAW13] M. Hoque, S. Obaid Amin, A. Alyyan, L. Wang, B. Zhang, L. Zhang, NLSR: Named-data Link State Routing Protocol, The 3rd ACM SIGCOMM Workshop on Information-Centric Networking, August 2013.

[HADL12] Dongsu Han et al. XIA: Efficient Support for Evolvable Internetworking. The 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI'12) (San Jose, CA) April 25-27, 2012.

[JSTP09] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard (PARC) Networking Named Content, CoNEXT 2009, Rome, December, 2009. http://www.ccnx.org/.

[KATA00] D. Katabi and J. Wroclawski. A framework for scalable global IP-anycast (GIA). *ACM SIGCOMM Computer Communication Review*, 30(4):3–15, 2000.

[KAUN09] S. Kaune, K. Pussep, C. Leng, A. Kovacevic, G. Tyson, and R. Steinmetz. Modelling the Internet Delay Space Based on Geographical Locations. In *Proc. of the Euromicro PDP*, pages 301–310, feb. 2009.

[KECK07] T. Koponen, A. Ermolinskiy, M. Chawla, K. H. Kim et al. "A Data-Oriented (and Beyond) Network", in *SIGCOMM*, Kyoto, Japan, 2007.

[LABO10] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Proceedings of ACM SIGCOMM*, pages 75–86. ACM, 2010.

[LAND13a] R. Landa, J. T. Araujo, R. G. Clegg, E. Mykoniati, D. Griffin, and M. Rio. The Large-Scale Geography of Internet Round Trip Times . In *Proc. of IFIP Networking*, 2013.

[LAND13b] R. Landa, R. G. Clegg, J. T. Araujo, E. Mykoniati, D. Griffin, and M. Rio. Measuring the Relationships between Internet Geography and RTT . In *Proc. of IEEE ICCCN*, 2013.

[LEON08] D. Leonard and D. Loguinov. Turbo King: Framework for Large-Scale Internet Delay Measurements. In *Proc. of IEEE INFOCOM*, pages 31–35, 2008.

[LEUN06] K. Leung and V. Li. A paracasting model for concurrent access to replicated Internet content. *Multimedia, IEEE Transactions on*, 8(1):90–100, 2006.

[LIBE05] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33):11623–11628, August 2005.

[LUME07] C. Lumezanu, D. Levin, and N. Spring. PeerWise discovery and negotiation of faster paths. In *Proc. of ACM HotNets*, 2007.

[LUWU07] M. Lu, J. Wu, K. Peng, P. Huang, J. Yao, and H. Chen. Design and evaluation of a P2P IPTV system for heterogeneous networks. *Multimedia, IEEE Transactions on*, 9(8):1568–1579, 2007.

[MADH06] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *Proc of USENIX NSDI*, pages 367–380. USENIX Association, 2006.

[MAXMIND] MaxMind GeoLite City. http://www.maxmind.com/app/geolitecity, 2013.

[MAYM02] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Peer-to-Peer Systems*, pages 53–65, 2002.

[NARA12] A. Narayanan, D. Oran, NDN and IP Routing: Can It Scale?

http://tools.ietf.org/group/irtf/trac/raw-attachment/wiki/icnrg/IRTF%20-%20CCN%20And%20IP%20Routing%20-%202.pdf

[NDGK12] Serval: An End-Host Stack for Service-Centric Networking. By Erik NordstrÃ¶m, David Shue, Prem Gopalan, Rob Kiefer, Matvey Arye, Steven Ko, Jennifer Rexford, and Michael J. Freedman. In Proc. 9th Symposium on Networked Systems Design and Implementation (NSDI â ™12), San Jose, CA, April 2012.

[NDN] Named Data Networking, project Web page, available at http://named-data.net/project/

[NDN10] L. Zhang, D. Estrin, J. Burke, V. Jacobson et al. “Named Data Networking (NDN) project”, PARC, 2010

[NEUSTAR] Neustar IP Geolocation. http://www.neustar.biz/solutions/ip-geolocation, 2013.

[NGUY04] T. Nguyen and A. Zakhor. Multiple sender distributed video streaming. *Multimedia, IEEE Transactions on*, 6(2):315–326, 2004.

[NGSON11] NGSON (Next Generation Service Oriented Network) – IEEE 1903, “Standard for the Functional Architecture of Next Generation Service Overlay Networks”, 2011.

[NSS12] E. Nygren, R. K. Sitaraman, J. Sun, “The Akaimai network: a platform for high-performance internet applications”, *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, 2010.

[PADH98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. *SIGCOMM Comput. Commun. Rev.*, 28(4):303–314, 1998.

[PART93] C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. *RFC*, 1993.

[PSIRP10] Deliverable D2.5 – Final Updated Architecture http://www.fp7-pursuit.eu

[RFC6920] Naming Things With Hashes, http://datatracker.ietf.org/doc/rfc6920/

[SAIL13] D.B.3 – Final NetInf Architecture http://www.sail-project.eu

[SAIL13-3.2] D-3.2 NetInf Content Delivery and Operations http://www.sail-project.eu

[SAIR08] “State of the Art on IRMOS technologies”, Interactive Realtime Multimedia Applications on Service Oriented Infrastructures ICT FP7-214777, Deliverable D2.3.1, 2008.

[SCEL11] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft. Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. In *Proceedings of WWW*, pages 457–466, 2011.

[SSRV11] S. Shanbbag, N. Schwan, I. Rimac, M. Varvello, “SoCCeR: Services over content-centric routing”, in *ACM Workshop on Information-Centric*, Toronto, 2011.

[VNI16] Cisco Visual Networking Index: Forecast and Methodology, 2011-2016.

[VST09] Van Jacobson, D. K. Smetters, J. Thornton et al. “Networking Named Content”, in Proc. of the 5th International Conference on Emerging Networking Experiments and Technologies (CONEXT), 2009.

[VWALL13] "iLab.t Virtual Wall | Internet Based Communication Networks and Services," IBCN, [Online]. Available: http://www.ibcn.intec.ugent.be/content/ilabt-virtual-wall.

[WITT10] M. P. Wittie, V. Pejovic, L. Deek, K. C. Almeroth, and B. Y. Zhao. Exploiting locality of interest in online social networks. In *Proceedings of ACM CoNEXT*, pages 25:1–25:12. ACM, 2010.

[WJFR10] P. Wendell, J. W. Jiang, M. J. Freedman, J. Rexford, “Donar: decentralized server selection for cloud services”, *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, 2010

[XUMIN03] Z. Xu, R. Min, and Y. Hu. HIERAS: A DHT Based Hierarchical P2P Routing Algorithm. In *Proceedings of the International Conference on Parallel Processing*, pages 187–196, 2003.

[XUTAN03] Z. Xu, C. Tang, and Z. Zhang. Building topology-aware overlays using global soft-state. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 500–508, 2003.

[ZHAN06] B. Zhang, T. S. E. Ng, A. Nandi, R. Riedi, P. Druschel, and G. Wang. Measurement based analysis, modeling, and synthesis of the internet delay space. In *Proceedings of ACM IMC*, pages 85–98, 2006.

[ZOEL06] S. Zoels, Z. Despotovic, and W. Kellerer. Cost-based analysis of hierarchical dht design. In *Proc. of IEEE P2P*, pages 233–239. IEEE, 2006.