ANA Project

Autonomic Network Architecture

Sixth Framework Program

Priority FP6-2004-IST-4

Situated and Autonomic Communications (SAC)

Project Number: FP6-IST-27489

Deliverable D2.16

Final design, evaluation and prototype implementation of content-based routing mechanism

Version 1.0

ANA Project

Autonomic Network Architecture

Project Number	FP6-IST-27489
Project Name	ANA - Autonomic Network Architecture
Document Number	FP6-IST-27489/WP0/D2.8
Document Title	Final design, evaluation and prototype implementation of content-based routing mechanism
Workpackage	WP2
Editors	Serge Fdida (UPMC), Ioannis Stavrakakis (NKUA)
Authors	Yosra Barouni, Marco Bicudo, Promethee Spathis (UPMC) Konstantinos Oikonomou (NKUA) Jin Xiao (UW), Qi Zhang (UW)
Reviewers	Martin May
Dissemination level	Public
Contractual delivery date	31st January 2009
Delivery Date	January 2009
Version	1.0

Abstract:

This document presents the integration of service discovery and content centric routing to allow intra- and inter- compartment routing for the ANA Architecture.

Keywords:

Autonomic, Content Routing, Filtering, Service Discovery, Routing

TABLE OF CONTENT

1 INTRODUCTORY

WP2 focuses on the basic elements and mechanisms needed to establish communications between two or more entities. This includes naming, addressing and routing schemes (both intra- and inter-compartment) for point-to-point, group and overlay-like communications. In the context of autonomic communications it is also important for individual network nodes to be able to autonomically bootstrap and configure themselves (self-associate) in a new network context, or for the overall network to autonomously organize and dynamically form it. WP 2 also deals with the issues of routing on various aspects in intra and inter-compartment communication, and investigates the close relation between service discovery, content-based routing and routing in general.

1.1 Scope of the Deliverable

In this deliverable, we first present a content centric routing protocol for large scale intra-compartment communication without addresses to route messages between networks nodes. We investigate design of an ANA-specific content centric routing, which contributes to the autonomicity of the global ANA framework by providing it with properties like self-management and self-optimization. The proposed protocol takes benefits from the inherent properties of ANA such as modularity to be flexible.

Second, we present a routing scheme inspired from a service discovery scheme proposed in deliverable 2.13, scheme that appears to be beneficial for both operations and for the overall system.

Third, we present the design and implementation of an inter-compartment service discovery (ICSD) system that can facilitate service discovery across compartments with heterogeneous intra-compartment service discovery protocols. Furthermore, we propose a scheme for self-configuration and self-optimization in ICSD.

1.2 Structure of the Document

In the first part of the deliverable, we start by giving a general description of the content routing problem and a brief state of the art of the existing content routing solutions. In section 2.2, we describe the basic design and the different processes that are executed by the CCR protocol. In sections 2.3 and 2.4, we introduce the main communications primitives used by our protocol and analyze the benefits of using CCR in the ANA framework. In section 2.5, we present the implementation of this protocol using the ANA API.

In the second part of the deliverable, we present a routing scheme based on service discovery, where advanced mechanisms are more detailed in deliverable 2.13. After describing the basic design and the routing bricks implemented to turn out this routing effective, we present the implemented brick and its important features. We conclude the second part by a discussion about the benefits of the integration of routing and service discovery into the work-package 2.

In the third part of the deliverable, we present an inter-compartment service discovery system. In section 4.1 we present the architecture and implementation of the ICSD system. In section 4.2 we presents a scheme for self-configuration and self-optimization of its distribute overlay component.

As a conclusion to the overall document, we analyze the integration of the proposed routing schemes and their benefits on routing using the ANA framework.

2 CONTENT CENTRIC ROUTING FOR ANA

In the context of the task 2.6, content centric routing is planned to be designed for both intra- and inter-compartment communication. In this deliverable, we focus on the intra-compartment aspects [5]. Our intra-compartment content routing scheme is called CCR (Content Centric Routing) and provides the ability to access data objects in a location-independent manner. CCR is designed to support large-scale autonomic networks and at the same time to minimize the routing cost inside each compartment. CCR exploits the benefits of the ANA framework to design a content-centric routing scheme that provides the ability to access data without searching for host addresses.

For this aim, CCR doesn't make use of classical search techniques like blind flooding. Instead CCR is based on metrics designed to filter content requests hop-by-hop. This kind of filtering routing makes CCR address-agnostic and compliant with the design principles of ANA since there is no addressing scheme defined. .

In our design, content routing is based on mediation entities that are called MR (Mediation Routers) and are responsible for the efficient discovery of the piece(s) of content matching a particular query, optimize the delivery that (those) piece(s) of content to the querier by selecting the most efficient location. Content providers publish their content to specific MRs called TMRs (Terminal Mediation Routers, see figure 1). The TMRs are located at the border of the compartments and are connected with the rest of the MRs within what we define as the Media Network. A TMR is responsible for storing data items and for the dissemination in the Media Network of the publications submitted by the original publishers.

A user looking for some piece of data queries its TMR by sending a subscription request that contains keywords describing his interest in some piece of data.. The TMRs are then responsible for the request processing across the mediation network until the data item matching the request is discovered. The subscription request contains a filtering vector used by an optimization brick at the MRs to determine which of their neighbors can forward the client request to the content location without having a prior knowledge of these locations. Once content is found, the results are sent back, ranked and merged at each hop of the reverse path until they reach the original client. CCR is also adapted to the context of autonomic and large-scale networks.

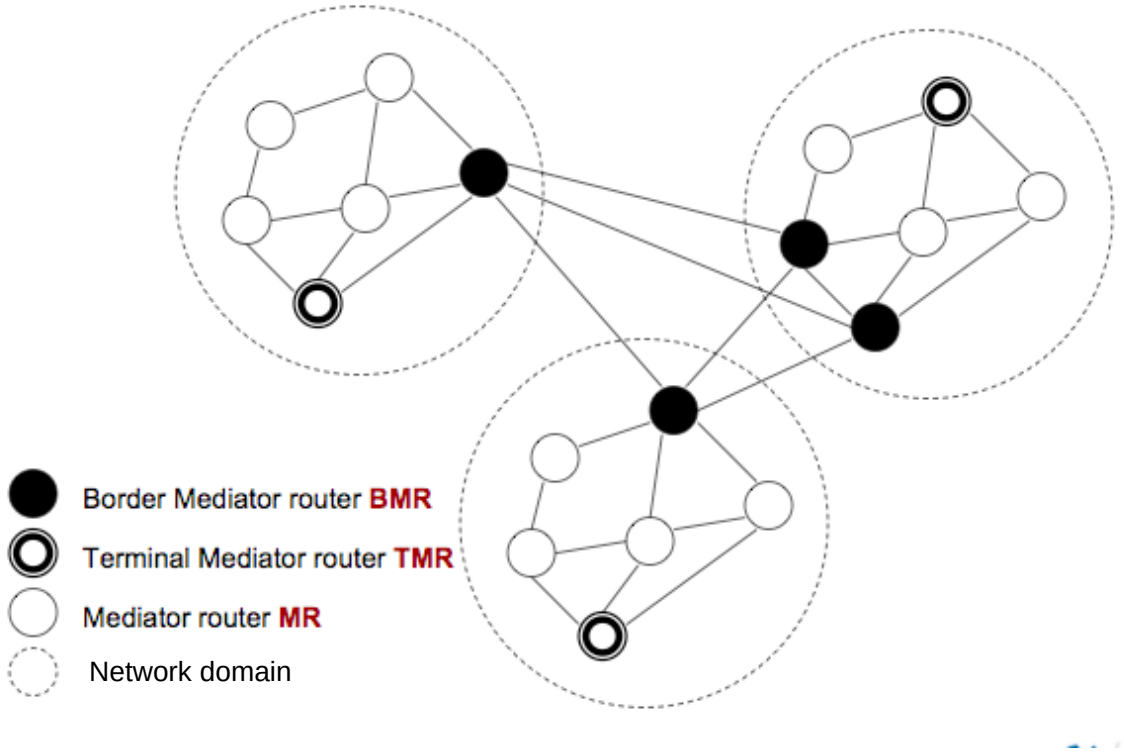


Figure 1: Organization of the mediation infrastructure inside ANA compartments

2.1 Background and state of the art

In the next section, we introduce the concept of content routing and present the main architectural approaches proposed in the context of content-networking.

2.1.1 Background

Content-networking stands for the ability of accessing data objects in a location-independent manner. This new approach changes the way communication takes place among network entities since current routing and content discovery and distribution work according to a design paradigm that have been exclusively host-centric. To relay messages from a source to the destination, both nodes are identified using unique names and addresses. Routers know how to forward messages hop by hop to their destination host, routing them to their destination, identified by its address. When a source intends to send a message to a destination, it sends a request to a server (a DNS server in the Internet) to provide the address for the given name. This process is called name resolution. Once the name is resolved, the source sends the message to the destination address.

However, usage patterns in terms of applications involve retrieving data objects or accessing services where emphasis must be given to the content rather to the location. Routing in a content network should aim essentially in providing location-independent access to an object without relying on the maintenance of network connections. Messages are routed on the basis of their contents rather than their host location. Binding contents to hosts is thus no longer necessary to provide access to them. Content access does not depend on the existence of an

end-to-end connection to the data source and attempts to retrieve any necessary content shouldn’t fail in the face of limited, partial, or intermittent connectivity.

Content networks focus on obtaining necessary content available at several places in the network. To do so, the networking infrastructure stores mappings between host addresses and pieces of content, and offers the main protocols that allow users to retrieve the corresponding piece of content given any description of interest. In addition to content names, content's attributes can be used to describe the content type and to search it. The design of a content-centric network should allow the efficient discovery of the piece(s) of content matching a particular query, and returning that piece of content to the queries from the most efficient location. Content may be replicated or moved to any node of the content network with the objective of improve content availability, increase efficiency by minimizing access time or provide resiliency to network failure and attack. In addition, security can be based on content rather than on the host storing the content.

In the last few years, there has been a growing interest in content-centric architectures that has led to numerous architectural approaches proposed in various contexts such as peer-to-peer networks, content distribution networks, publish-subscribe systems, and content-based sensor networks. Even though some of these approaches still provide access to content in a location-dependant manner, very few have succeeded to route contents independently of their host locations. The filtering scheme used to make content routing is a key issue to provide an efficient and pure address-agnostic content routing. In the next section, we propose a description and a classification of the major contributions that were proposed in the content based routing domain and especially those that are centered on content attributes.

2.1.2 State of the art

Although there has been a growing interest in content-centric architectures that has led to numerous architectural approaches ([14], [15], [16], [17], [18], [19]), our work is the first one to our knowledge that focuses on the design of a content-centric routing architecture that exhibits a set of autonomic principles.

Many criteria can be used to categorize existing content routing solutions; request routing model, system architecture, etc.

Considering request routing model criterion, existing solutions may be classified into two classes [13]: the *filter-based* approaches ([18], [19]) and the *multicast-based* approaches ([14], [15], [16],[17]).

In filter-based approaches, routing decisions are made via successive content-based filtering at all nodes from source to destination: every server along the way matches the content with remote requests from other servers, and then forwards it only toward directions that lead to matching client requests. This approach can achieve high network efficiency, but at the cost of expensive information management and high processing load at servers.

In the multicast-based approach, a limited number of multicast groups are computed before content transmission begins. For each content, the routing decision is made only once at the content provider, mapping the content into the single appropriate group. The content is then multicasted to that group, assuming there is IP multicast or application-level multicast support. Since only a limited number of multicast groups can be built, servers with different

interests may be clustered into same group, and as a consequence, contents may be sent to uninterested servers as well. The network efficiency of this approach is often highly sensitive to the data types and the dissemination scheme of content publication as well as client request [13].

Content based routing systems can be also classified according to their system architecture model and may be one of two [2]: client-server model, and peer-to-peer model.

In client-server model, specific components serve as brokers. They receive events (content providers' publication and clients' request), possibly store them, and forward them. Brokers communicate with other brokers to achieve properties such as scalability. Clients may issue requests for content as well as register content that they want to make available to the network.

In peer-to-peer model, all nodes offer the same functionalities. That is, each node can act as a content provider, client, broker or any combination thereof.

The classifications described above motivates certain decisions presented in the next section so that our proposed routing framework achieves ANA design goals like the routing performance, scalability and self-properties.

2.2 Design

2.2.1 Design Choices

The first important decision is the request routing model. As said in section 2.1.2, we have the choice between using a filter-based approach or a multicast-based approach. This decision depends closely on the design issues that we analyzed in the deliverable 2.8 [7] about the requirements of a content routing protocol in autonomic networks. Scalability and mobility are important issues that our protocol must face. A multicast-based content routing is not well adapted to very large-scale networks neither to mobile networks because of the cost of maintenance of the multicast group state that will be created for each client request in this kind of solution [21]. Compared to that, filter-based approaches seem to be more adapted to dynamic networks. Even if the information management cost in filter based routing is high, we provide simple mechanisms to reduce it, like the aggregation of both client requests/subscriptions and content metadata stored in the routing tables.

Another critical decision is the choice of system architecture model. As mentioned in the previous section, the literature points out basically two solutions: peer-to-peer based and broker based. If we decide to design our system based on the peer-to-peer model, every node belonging to the autonomic network has to store and maintain a sort of filtering table. This table stores information about the contents available in the network as well as the client's requests processed by the nodes. As we aim to support large-scale and heterogeneous (in term of software and hardware) networks, this choice is not well suited because of the large size of such. On the other hand, using broker-based model, an algorithm of broker election has to be deployed inside the autonomic network. But it offers certainly more scalability perspectives. We choose then to affect the brokering functionalities to specific nodes. The election or selection of these nodes may be done or at least inspired thanks to the leader election algorithm in deliverable 2.13.

As we decided above to use a filter-based approach, an important point to discuss is the hop-by-hop filtering metrics of the content requests. If we see the literature of the different content routing systems, such as peer-to-peer networks, content distribution networks, or publish-subscribe networks, different metrics are used to forward a request to a host. In CDNs [22] and some peer-to-peer systems like chord [23], the distance is an important criterion used to select the best server location for a content request. In others peer-to-peer systems like [24], the reputation of the peer is an important criterion to improve the semantics of the content routing service. This is also valuable to offer more security to the routing process, especially between anonymous peers. In information retrieval systems like [25], the content popularity is the most important criterion to send back the matching content to the host. We chose to combine these metrics, namely the distance, the content popularity and the satifaction ratio in a filtering vector. This vector is used by an optimization brick to select the most suited neighbor that can forward the client request to the content location without having a prior knowledge of these locations.

In the next section, we describe the main networking entities of our content routing proposition.

2.2.2 Protocol Description

We propose an intra-compartment routing scheme centered on content that we call CCR. Our protocol benefits from the properties of the ANA framework[5] to process content requests between cooperative nodes in a flexible and IP-agnostic way. This independence is important as it allows to route contents between nodes using the clients interests expressed inside their requests. The CCR protocol is implemented in a functional block that aims to find out which IDP is associated to a specific content requested by clients.

The CCR FB is based on a hybrid and a hop-by-hop based routing protocol that is centered on the metadata describing content. It implements a content centered routing protocol that filters and distributes content according to the interest of users. Users need just to express their interest by declaring some expressive keywords in their request. After that, content management and routing infrastructures, composed of specific ANA nodes that offers the CCR service, have the responsibility to match the client requests to specific filtering tables stored in each of these nodes. The matching process leads to the selection of the next neighbor(s) that provides the optimal content corresponding to the user request according to some metrics we'll present later. Then, requests are forwarded to the returned neighbors. Each ANA node offering this service has to bind an IDP to its CCR FB. FBs use a publish primitive already defined in the ANA API in order to publish about their availability to offer the associate content to their peers and register the corresponding IDP inside the IDT.

Basically, each node willing to use the CCR has to disseminate a request containing a set of keywords describing the interest of a client in registered contents. ANA nodes that offer the CCR service maintain a content dispatching table (CDT) that allows reaching the node that hosts the optimal content matching the client request without having to search for it using addresses.

As we said before, the filtering process is done "hop by hop" and no final destination address is returned by the system. This is a feature that makes our system autonomic in the sense if a request has to go through different networking infrastructures to reach a matching content to the client request, nodes do not have to translate a final destination address that can belong to

an unknown name-space. They have just to specify the next IDP that is supposed to provide the best content for the request and forward it.

After that, an optimization process is executed in order to select a limited set of the neighbors CCR IDPs to which the client request is forwarded instead of flooding the received request to all the known CCR IDPs. In order to filter these neighbors as shown in the figure 2, ANA nodes have to maintain in a vector the value of two filtering metrics for each IDP: *D* the distance expressed in number of jumps which separates the processing FB of the potential contents matching the request, and *P,* which measure the popularity of the contents associated by and IDP and expressed as the number of access to the same IDP for similar requests. Determining the most relevant IDP(s) for a content request consists in selecting the IDP maximize the *P* metric while minimizing the *D* metric. For this aim, we created a ratio called "filtering gain" FG that allow us to measure the division of P and D. The CCR system chooses the highest FGs in order to have a compromise between the maximal P and the minimal D.

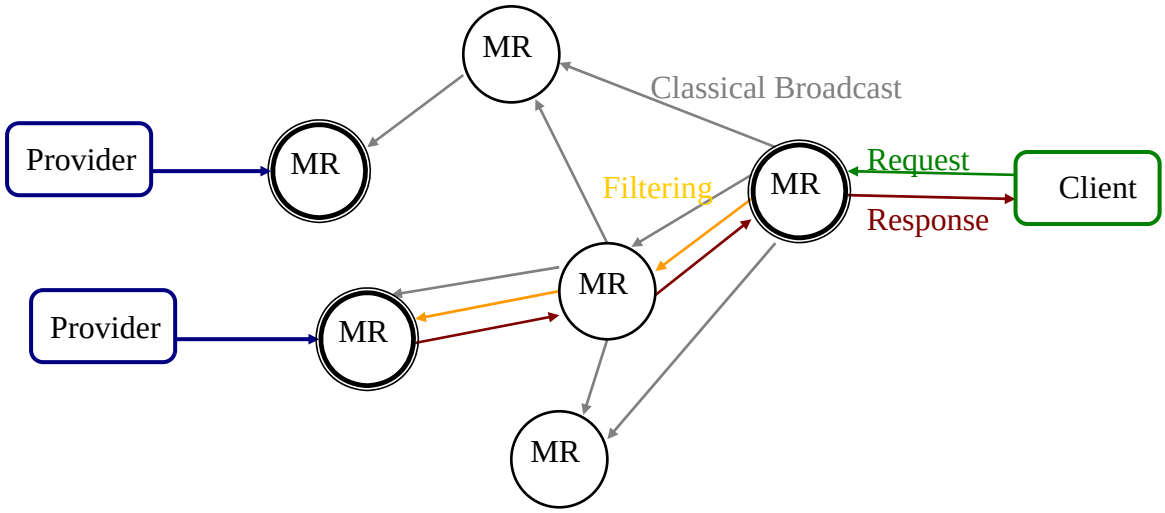


Figure 3: Principle of the CCR filtering

Our CCR protocol is aimed to work independently from the networking infrastructure in which it is deployed for two reasons. The first one is the address-independence of our filtering process. The second reason is due to the inherent properties of the ANA framework; first, contents are matched to IDPs instead of nodes addresses. Second, the inherent modularity of ANA makes it a major strength of our CCR FB as it becomes easier to make code reuse, extensibility and customization.

2.2.3 Dissemination

The dissemination process is a key factor of the performance of any routing protocol. In our case, we call content dissemination the process of distributing the content descriptions originally registered inside the TMRs to the remaining mediation network. A good dissemination scheme can allows us to achieve two goals; a performance goal, which is the minimization of the overall routing overhead, and a semantic goal, which is the maximization of the accessibility of registered contents in the mediation network. The accessibility is the

property making a pertinent and well-advertised content be reached wherever a client asks for it.

In the literature, routing protocols designers have the choice between flooding routing information, which is the most efficient dissemination scheme, and limited dissemination that can be based on random algorithms, point-to-point. In our case, we combine the flooding and the filtering dissemination like following; when a mediation router receives a client request, he searches for the best neighbours that can satisfy it. If not found, the request is broadcasted to all his direct neighbours.

2.2.4 Filtering

The content filtering consists in filtering clients requests based on the content descriptions (keywords/metadata). The content routing process used in CCR can be assimilated to the construction of an optimized filtering and mediation tree per client request.

An important property to be satisfied by the CCR protocol is how to create really a tree and avoid the routing loops inside the mediation network. This issue is resolved through an implementation trick that is described later. In order to avoid repetitive queries and optimize the processing time and overhead, we use a merging mechanism that consists in merging the inclusive requests as shown in the figure below:

Figure 4: Aggregation of clients' requests inside MRs

2.2.5 Content Delivery

Content delivery is the final step of the content routing protocol. It consists on sending a response with the content matching the client request from the optimal TMR satisfying it. We define as a simple process consisting on sending back the response on the reverse path of the processed request. Let's represent the network as a graph; the content delivery process is executed all along a tree rooted in the client that issued the request.

2.3 Communications interfaces

CCR needs specific messages to function properly. These messages have to be as flexible as possible in order to be adapted to the aim of supporting different routing platforms inside the ANA project. As shown in the figure the main communications interfaces used inside CCR are four: `request`, `publish`, `search`, and `response`.

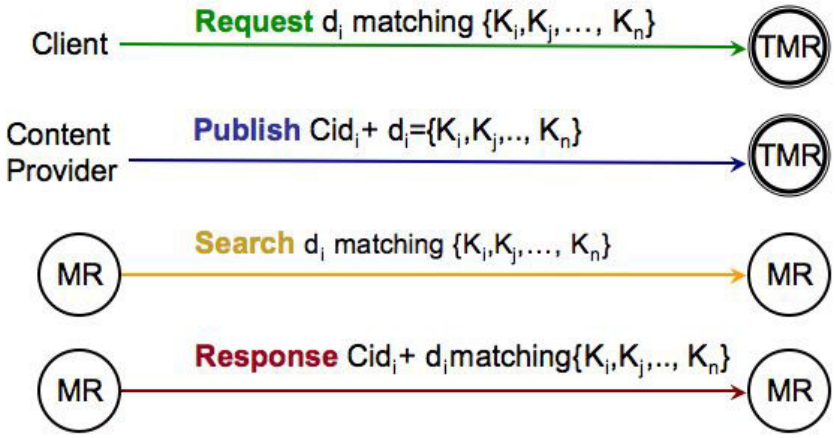


Figure 2: CCR communication primitives

The client request is a message exchanged exclusively between a client and his own terminal mediation router. The terminal mediation router (TMR) has the responsibility of processing and propagating the client requests to the rest of the mediation network, which is composed by all types of mediation routers.

A publisher who wants to push contents and their descriptions inside a TMR sends a register message. Inside this message, he has to specify the description of the content using simple keywords.

The TMR that receives a content request generates a search message that is forwarded to the neighbor IDPs of the compartment until a response is found or all hosts process the request. Finally, if a corresponding TMR is found, the response message is created and sent back to its source.

2.4 CCR contributions inside ANA

In the context of ANA, CCR provides robustness and adaptability, by continuing to send traffic in the face of a wide range of disruptions and misconfigurations. CCR's internal routing protocol are designed to be robust to a wide range of variance in network operations by adapting not only in the face of the machine failures and network partitions that today's Internet was designed to cope with, but also in the presence of the active network attacks, ubiquitous mobility, and the composition of wildly heterogeneous types of subnets that are becoming the new norm.

CCR self-adapting properties follows simple internal algorithms to determine the paths that data should take, adapting to node mobility and loss, and shielding the network from accidental or malicious misconfiguration. If the network does not require (or allow) administrators to input large amounts of control information determining where data should flow, instead determining that information dynamically on its own, then there is no opportunity for accidental or malicious manipulation of such control information to interfere with network operation.

2.5 Implementation of the CCR protocol in ANA

This section describes the content centric routing in the ANA Playground. Following, we show which implementation tools have been used and all the details and the development of the brick that make the functional block content centric routing work.

2.5.1 Overview

The content centric routing protocol is decomposed in a way that enables the client process to send CCR commands, leave it do its work, and receive the response wanted asynchronously when the request is completely processed. This is possible because the bricks are designed with the MINMEX and Playground in a modular and flexible way. The protocol is a functional block, which are composed itself by bricks. The communication between the bricks as well as the nodes is encoded as XRP messages.

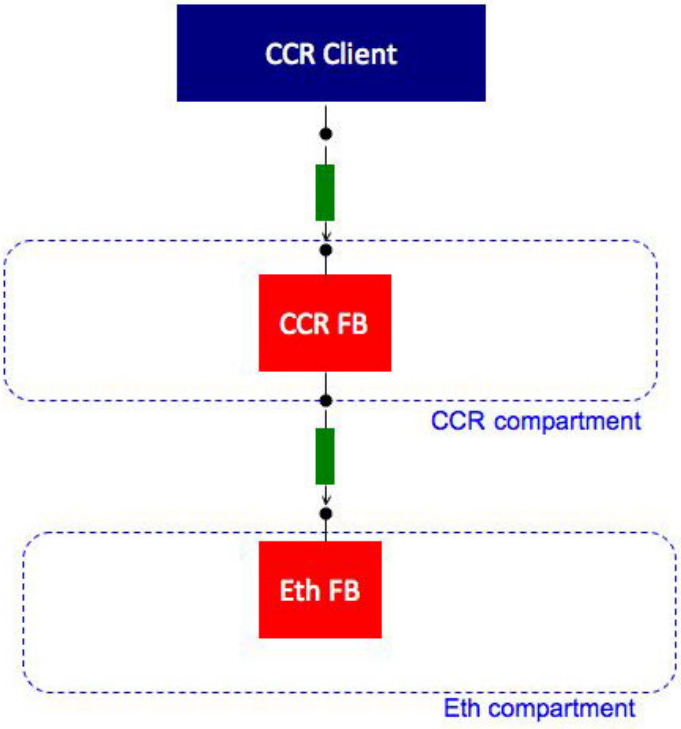


Figure 5: Protocol stack of CCR brick over a generic network compartment

2.5.2 Modular decomposition of the CCR FB

In this paragraph, we present the description of the CCR protocol as it is designed using the ANA framework[5].

As shown in the figure 5, the CCR FB can be decomposed of three main bricks: a request defragmenter brick, an optimal vector solver brick and a request processor brick.

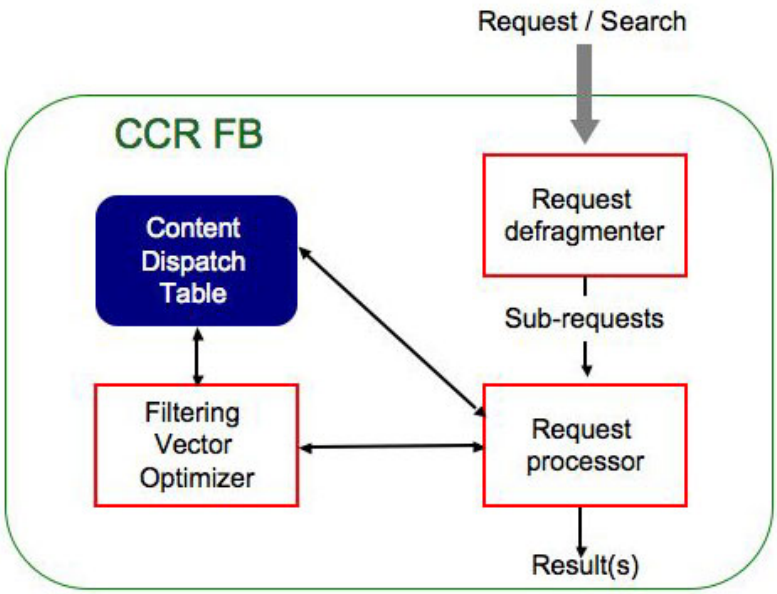


Figure 6: Decomposition of the CCR Functional Bloc

As shown in figure 4, the defragmenter brick decomposes the received request into atomic sub-requests that contain one exact keyword. These sub-requests will be treated separately inside the mediation routers in order to make the routing process faster and simpler. The generated sub-requests are treated by the filtering brick, which chooses the CDT entries that matches the optimal output interfaces satisfying each atomic sub-request. Then, the pre-selected entries are transmitted to the optimizer brick in order to compute the list of the optimal entries according to a filtering vector and thus select the adequate neighbor IDP that matches each sub-request. As a result, a final list of the most pertinent and optimized IDPs is returned back to the processor FB that has the responsibility of aggregating the different results of each sub-request and eliminating duplicated IDPs. Once the results are merged, the system is able to forward the entire request received originally from the client to the optimal neighbor IDP that satisfy it.

The benefits of the decomposition of CCR functionalities into separate functional blocks are obvious. It provides a flexibility of improving, testing and deploying each FB. It also allows testing the impact of each block on the other ones easily. But there are also some disadvantages of the existing decomposition: it increases the overhead of each sub-process called inside the routing process. Even if there can be some ways to decrease the overhead generated by this decomposition, this is not the aim of the ANA project. That’s why we do not provide a fixed solution to this problem.

The figure 6 describes a CCR communication scenario in an IP network. The scenario is started by the reception of a request by the CCR service. This service plays the role of interface with CCR clients and ANA communication system.

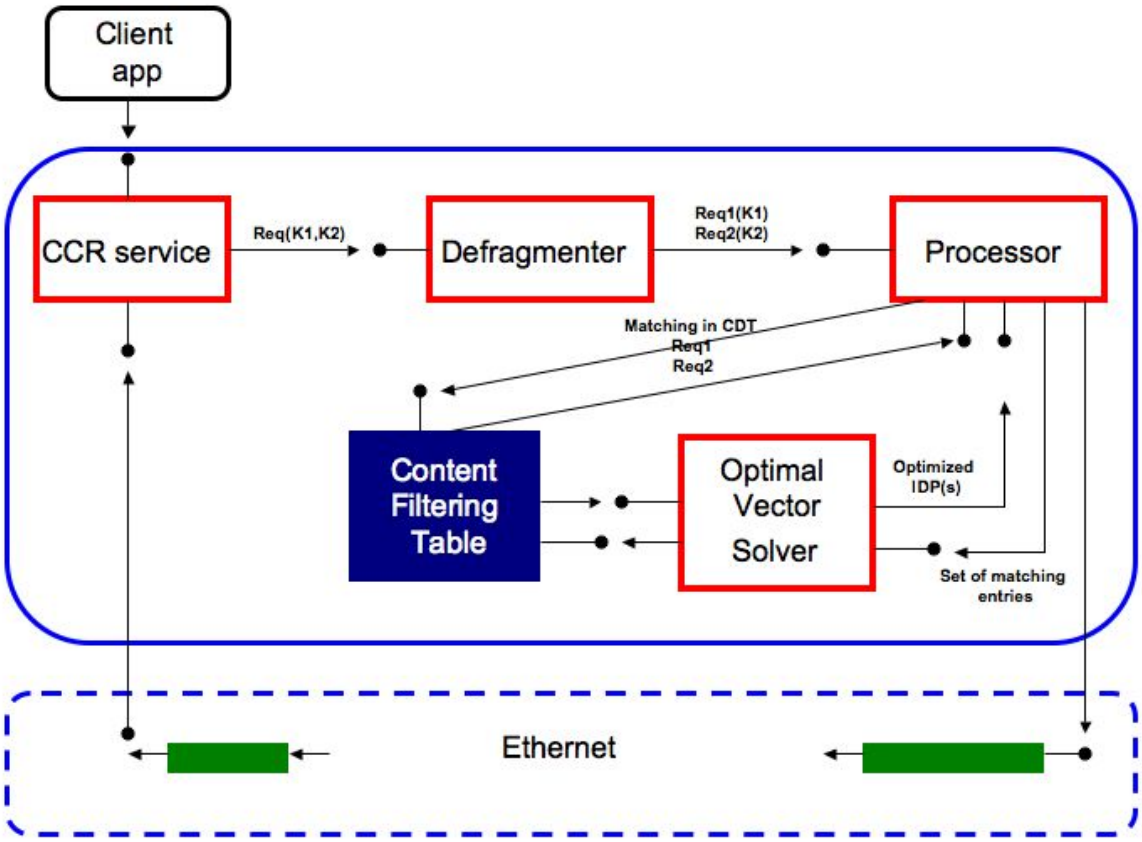


Figure 7: Interaction between the different Bricks of the CCR FB

Each FB is registered with an IDP in the IDT (Information Dispatch Table). An FB that wants to communicate with another FB must search the associated IDP in the IDT. When a client request is received by the initial FB, it is processed as explained previously and, as a result, the processor FB returns the corresponding neighbor IDPs to the client. Then, an ETHERNET FB, which is already implemented in ANA framework, resolves the returned IDPs to ETHERNET addresses and forwards the messages to the corresponding neighbor. It depicts clearly the IP-independence of the ANA framework, but it doesn't mean that it couldn't be used on IP-based sub-networks of the ANA network. Moreover, it allows a more transparent and feasible communication among any sort of sub-networks that do not use IP addressing.

2.5.3 Implementation approach

During the implementation process some particular requirements of the CCR functionality came out to be a problem in the mapping of CCR commands and ANA API functions. Briefly, they were:

- the publish command need to receive three arguments (keys, context and content) to the CCR brick and the ana_publish function has only two;
- the request/get CCR command should obtain a content, but the ana_resolve receives one idp to send content and not to receive it;

- Although to implement an hop-by-hop protocol the address of the next hop is needded, the CCR bricks throughout the network cannot store the neighbours address, either they are IP, Ethernet or any other network or under layer protocol.

Those impediments were well discussed with the team responsible for the ana-core, and adaptations in the protocol behavior and in the utilization were made. Two main changes of paradigms were about the use of the ANA API:

- The CCR publish command would use the ana_resolve() ANA API function and the CCR request command would use the ana_publish() ANA API function. The reason for this is uniquely because of the two first problems. The ana_publish() function command permits the application to receive content, which is what CCR request command needs. Also, the ana_resolve() ANA API function permits a third content to be passed after it receives an IDP.

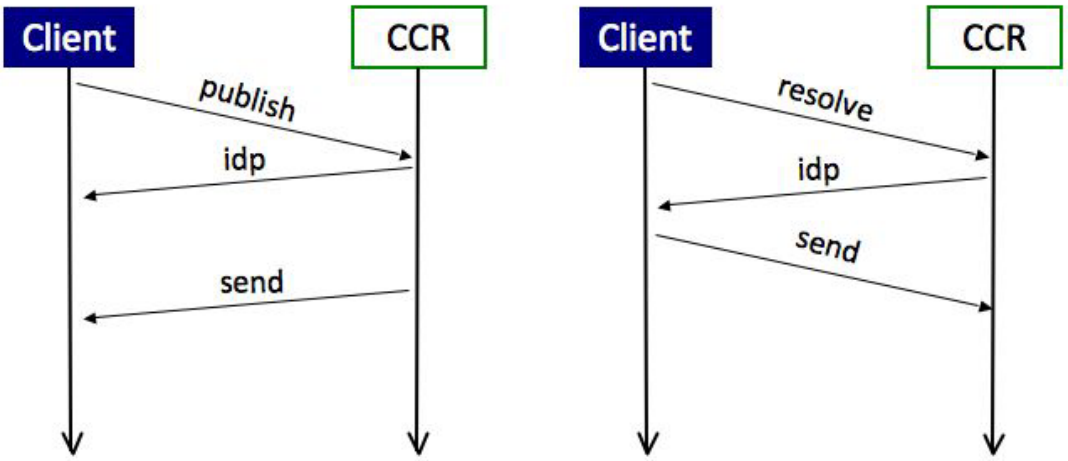


Figure 8: Scheme of request/get and publish a content, respectively

- The hop-by-hop would not use unicast at each hop, but a specific muticast IDP that reaches all the CCR bricks in the link attached to the outgoing link of the next hop. Althought this creates more network traffic and overhead, it's a solution to constraint of not violating layers' isolation.

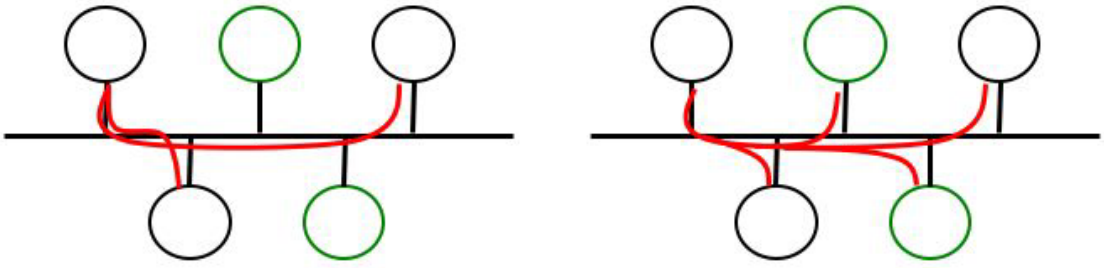


Figure 9: Unicast and multicast forwarding

Following, we describe the implementation of the CCR brick after the changes presented above in this section that affect the logic of the protocol and the implementation.

2.5.4 CCR API

After obtaining the IDP of the local CCR, the client sends commands to the CCR brick using ANA API methods: publish, unpublish, resolve, release, lookup and send. The CCR brick treats commands using a different logic then most of the network bricks. The reason for this behaviour is the number of parameters accepted by the publish function. Since the client cannot send to the CCR brick the content and the content description using the publish function, it was arranged to use the resolve function instead. This change of logic, which basically interchanges the names of the functions publish and resolve, turned out to be very practical in this situation. Using the resolve function the client can pass as arguments the content's description. Then, the client receives an IDP to send the content to the CCR brick. The content is stored by the brick and, at the end of transmission of the content, the temporary IDP is released by the client and CCR brick.

It's worth also mentioning that, just after that CCR brick receives the command and all its data, internal actions associated with the behaviour of the CCR brick could take. These actions are mostly related to network communication and processing, and they have no need of client intervention. Each of those actions is better described in the following paragraphs.

For the publish action the client uses the resolve function to obtain an IDP and, then, sends the content to the IDP just acquired. As mentioned, this awkward use is due to the lack of arguments of the publish function. In order to avoid this problem, we use the resolve function and send the content through the IDP returned. The code below depicts the earlier description.

```
anaLabel_t anaL2_resolve (anaLabel_t compIDP,
     struct context_s *ctx, struct service_s *srv,
     char chanType, struct service_s *description,
     struct timespec *timeout);

int  anaL0_send (const anaLabel_t label, void *msg, int len)
```

The retrieve of content uses the publish ANA function. The idea behind it is to publish one interest in a specific content. So, if available, the network responds the request with the response. The code following presents the flow.

```
anaLabel_t anaL2_publish (anaLabel_t compIDP,
     struct context_s *ctx, struct service_s *srv,
     AL2Callback_t callback, int separateThread,
```

```
    struct timespec *timeout);
```

The unpublish action is the only function that uses the same logic of the ANA API. To unpublish some content one should use the unpublish function, passing the content description as arguments. Then, the brick should do its own work to change the CCR Tables (CDT) throughout the network.

```
anaLabel_t anaL2_unpublish (anaLabel_t compIDP,
    anaLabel_t label, struct context_s *ctxt,
    struct service_s *srv,
    struct timespec *timeout);
```

2.5.5 Interfaces

This section describes how the communication between the CCR brick and other applications, bricks and functional blocks works. Both the messages that traffic inside the node, as those exchanged between applications and bricks, and that traffic to the network are always encoded as XRP messages. The XRP protocol standardizes the format of the messages containers. Without this clear definition of interfaces, one cannot achieve the flexibility and protocol-agnostic features aspired. The following paragraphs describe the encoding details of XRP messages as CCR messages, as well as XRP Commands and XRP Classes used and for what purpose they should be used.

As described in the CCR API Section, the application client uses the standard ANA API. In this section, we show how those functions are translated into XRP messages and how CCR brick treats and reacts to theses XRP messages.

2.5.5.1 Intra-node

The CCR functional block is built of only one brick and, doing so, there is no need for internal messages among CCR bricks. As a consequence, basically the intra-node messages are messages between the CCR brick and the client, the application itself. These messages are described below.

- The CCR PUBLISH command: translated as anaL2_resolve() function in ANA API.
    - **XRP_CMD_RESOLVE**: labels a publish command.
    - **XRP_CLASS_DST_SRV**: contains the keys.
    - **XRP_CLASS_DST_CTX**: contains the regexp (in the moment only ‘.’).
- The CCR REQUEST/GET command: translated as anaL2_publish() function in ANAAPI.

    - **XRP_CMD_PUBLISH**: labels a request/get command.
    - **XRP_CLASS_DST_SRV**: contains the keys.
    - **XRP_CLASS_DST_CTX**: contains the regexp (in the moment only '.').

- The CCR UNPUBLISH command: translated as anaL2_unpublish() function in ANA API.
    - **XRP_CMD_UNPUBLISH**: labels a publish command.
    - **XRP_CLASS_DST_SRV**: contains the keys.
    - **XRP_CLASS_DST_CTX**: contains the regexp (in the moment only '.').

2.5.5.2 Inter-node

After receiving the commands from the client application, the CCR brick start a thread to communicate with other nodes and start the CCR request, publishing or unpublish.

- The CCR REQUEST/GET command: this message is send through the network and, in a hop-by-hop manner and based in the CDT, is routed till the node that has the content.
    - **XRP_CMD_RESOLVE**: labels a request/get command.
    - **XRP_CLASS_CCR_REQID**: contains the request id of the message.
    - **XRP_CLASS_CCR_KEYS**: contains the keys searched.

- The CCR PUBLISH command: not implemented.

- The CCR UNPUBLISH command: not implemented.

- The CCR RESPONSE command: when the messages hits a node that matches the content description, totally or partially, this node prepares the message response, with the content wanted.
    - **XRP_CMD_DATA**: labels a response command.
    - **XRP_CLASS_CCR_REQID**: contains the request id of the message.
    - **XRP_CLASS_CCR_KEYS**: contains the keys searched.
    - **XRP_CLASS_CCR_CONT**: contains the content associated with the request.

2.5.6 Summary and Remarks

Bringing content centric routing to ANA framework appears to be beneficial for both elements. Content centric routing provides to ANA robustness and adaptability, by continuing to process requests even in disrupted environment. The self-optimized algorithm for CCR filtering allows to adapt routing paths to nodes mobility and appearances of new and more optimized contents. ANA framework provides modularity and self-deployment to CCR internal algorithms thanks to the brick paradigm that compose each functional block implemented using ANA framework. Each brick may be deployed in different nodes depending on the physical capability of the node. In addition, a large-scale and dynamic network environment was considered to design of the CCR protocol presented in section 2.2.

In implementation section (2.4), we described how some particular requirements of the CCR functionality came out to be a problem in the mapping of CCR commands and ANA API functions. Two major changes in ANA communication primitives were made in order to implement correctly CCR communications primitives. Implementation is ongoing and has reached an advanced stage despite the problems described above.

In the next section, we present another routing design that may be integrated with CCR protocol to a more global routing service adapted to the ANA framework.

3 ROUTING ARCHITECTURE DESIGN BASED ON SERVICE DISCOVERY

This section presents a design attempt to bring together service discovery and routing assuming large-scale, dynamic and unstructured network environments. The idea of bringing together service discovery and routing has been presented in previous deliverables (e.g., D2.1 and D2.8) and conceptual work has been carried out in the past. Here, the ANA Node platform is considered and basic design decisions are presented. Eventually, the design that is particular for service discovery (presented in the parallel deliverable D2.13) is shown that it can be reused for implementing routing schemes in the aforementioned network environment. Furthermore, a simple set of design instructions is given with respect to the efficient co-operation of both service discovery and routing.

3.1 Background

As it has been mentioned in past deliverables (e.g., D2.1 and D2.8), routing may be seen as a service offered to network nodes. Under this context, when a node requests a path towards another network node, this is similar to requesting the location of a node hosting a service. Of course, in the special case of routing, it is not the location of a node that is returned (as it is normally the case under service discovery), but rather the path towards a particular node. There is a fundamental question (often neglected): why a node wants to communicate with another node in first place (thus, initiating a request for route retrieval)? The answer is that most likely the latter node offers some service to the network nodes that the former node wants to use it. Eventually, routing and service discovery are closely related and this may be an advantage to both operations if carefully treated.

Summarizing,

1. service discovery and routing share many common elements. Therefore,

   service discovery approaches (algorithmic, design and implementation) could be reused for routing and vice versa

2. service discovery and routing have similar objectives. Therefore,

   when service discovery is aware of a node's location, it may be beneficial to share this information with routing and vice versa

3.2 A Design Scheme for Routing Based on Service Discovery

The basic design approach regarding service discovery is presented in the (parallel) deliverable D2.13 that provides for some important design decisions and some useful implementation details. As mentioned in D2.13, the adopted design approach is one that can be reused with relatively easiness for other distributed protocols (assuming the large-scale, dynamic and unstructured environment). Therefore, the basic design of D2.13 is adopted here for the case of routing. The basic details are brief mentioned and the reader is invited to read D2.13 for further details.

3.2.1 Basic Design Details

The routing functional block, as proposed here, assumes the existence of an "underlying" connectivity brick responsible for handling communication with nodes that are neighbors (i.e., nodes for which a packet transmission is possible without the need for some relay node in between). For simplicity, it is assumed that Ethernet is the underlying technology that provides for connectivity and therefore, an Ethernet brick is responsible to handle the communication with the physical medium. Note that the design is such that it may be reused even if more "elaborate" bricks like IP are considered.

In order to handle the communication with neighbor nodes, a special interface brick is required that publishes a unique name within the participating compartment. Since such a brick participates in two compartments (inside the ANA node and inside the network compartment) it should be made sure that the published name is unique in both compartments. This is not a trivial task (it would have been if, for example, the network size was small) and in order to deal with it, a "leader" is elected within each network compartment responsible for assigning unique names. Leader election is a task that takes place between the bricks responsible for handling the interfaces, thus leaving much freedom for the implementation of the particular distributed protocol.

Special messages are also defined among different bricks exchanging messages within the same compartment (network and ANA node compartment). Further information specifically for the routing case is presented next. More details about the specific mechanisms (e.g., leader election) can be found in deliverable D2.13.

3.2.2 Routing bricks inside an ANA Node

The basic brick of the routing functional block is the "scalable routing" or "sr" brick in which the implementation of routing takes place ("scalable" refers to the need for a scalable approach in the considered large-scale environment). The basic approach is to leave the sr brick as free as possible from "housekeeping" (i.e., information specific to the ANA nodes and network compartments) in order to allow for an efficient implementation of the routing protocol. Note that it is not the role of the design but rather that of the conceptual work to propose/specify the particular algorithm(s) that will be implemented inside the sr brick.

An important element of the routing function block is the "scalable routing interface" or "srif" brick, whose role is identical to the sdif brick presented in deliverable D2.13. This particular

brick, apart from assigning the unique names after a leader election process, is responsible for handling the communication with other ANA nodes within the same network compartment. Thus, it mediates between the sr brick (which has to be as simple as possible) and any other srif brick existing at some of the other ANA nodes of the same network compartment through the underlying network brick (e.g., Ethernet).

This particular case is depicted in Figure 1 and Figure 3 for an ANA node of a single and of two network interfaces, respectively. The srif bricks play an intermediate role, in order to allow for the sr brick to remain as simple as possible (for implementing the routing scheme).

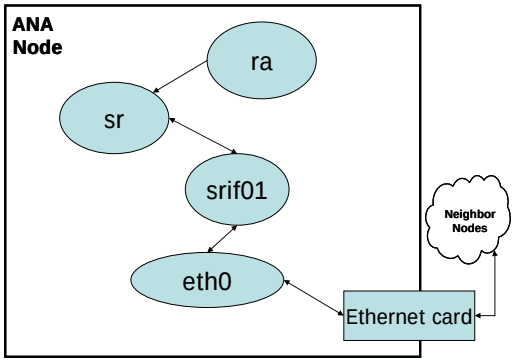


Figure 1: Routing functional block for a node with a single network interface.

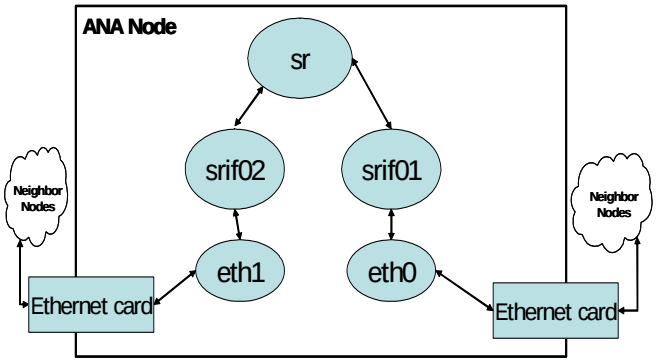


Figure 2: Routing functional block for a node with two network interfaces.

The “routing advertisement” (ra) brick shown in Figure 1 triggers the advertisement of the location of some node in the network. Note that “advertisement” is a process that is similar to constructing the routing tables in pro-active routing protocols. The basic idea in pro-active routing protocols is to create routes that will be possibly needed afterwards. The opposite idea (i.e., re-active routing protocols) lays on the philosophy that a route will be searched for as soon as a request for the particular route arrives. Clearly, if a re-active protocol is required, the routing advertisement process is not needed. However, for any possible case, a “routing search” (rs) brick is required to initiate a searching for a route. This particular rs brick is located at the same place of the ra brick in Figure 1 (triggers the sr brick of the local ANA node).

3.2.3 Implementation Specific

So far, the focus was on the design and therefore, the implemented routing algorithm inside the sr brick is a simple flooding protocol that searches the entire network in order to locate a

certain node and return the path to the particular node that initiated the request. More elaborate routing protocols can be implemented. Next we present some important structures of this simple implementation itself that give some insight of the particulars of the adopted design. Note that other important structures have already been described in D2.13 under the context of service discovery.

```
struct package{
      int num;
      char *routeR;
      anaLabel_t source;
      anaLabel_t routet;
      char *srifName;
      struct package * next;
      struct package * prev;
      float TimeToLive;
};
struct package *packageHead=NULL;
```

Structure `struct package` keeps information about routing packets that have already gone through the particular ANA node. A list of these packets is maintained (until a time expire event for each entry) and each new packet is checked with the packets already in this list. If the packet has been received before, then it is not forwarded further. This simple rule is helpful in flooding approaches to stop packets from moving indefinitely along cycles in the network. Needless to point out that other routing protocols may not require this rule.

```
struct routingBack{
      char *routeType;
      anaLabel_t destination;
      struct routingBack * next;
};
struct routingBack *routingBackHead=NULL;
```

Structure `struct routingBack` keeps information about the list of nodes that have been passed until the destination is eventually found. This is kept in all ANA Nodes and it is updated every time a routing packet is forwarded. For example, assume that a node is reached by a certain packet searching for a route. This packet will be sent back to the original node through all intermediated network nodes that have already been passed during the searching phase. Thus, these intermediate nodes will have the chance to look in the particular packet and retrieve knowledge about the route towards the particular destination.

3.3 Co-operation

As already mentioned, efficient co-operation between service discovery and routing is expected to be beneficial for the overall system. From a design point of view, which is the main goal, a common ground should be found for both service discovery and routing. A corresponding (not complete) list of design points is the following:

- There should be a common list of destinations, services hosted by those destinations and routes towards these destinations within each ANA node
- The SR brick and the SD brick (the latter explained in deliverable D2.13) should be able to
    - read from this list when a new request (either requesting a service or a route) arrives
    - write to this list when some new information (either about a service or a route) arrives
- The implemented service discovery scheme should be able to maintain route-specific information that is retrieved during the advertising/searching phase
- The implemented routing scheme should be able to retrieve information that is specific to service discovery (e.g., services hosted by a particular node)

Based on the fact that the sdif brick and srif brick have the same functionality, the co-operation scheme is eventually reduced to the one presented in Figure 5.

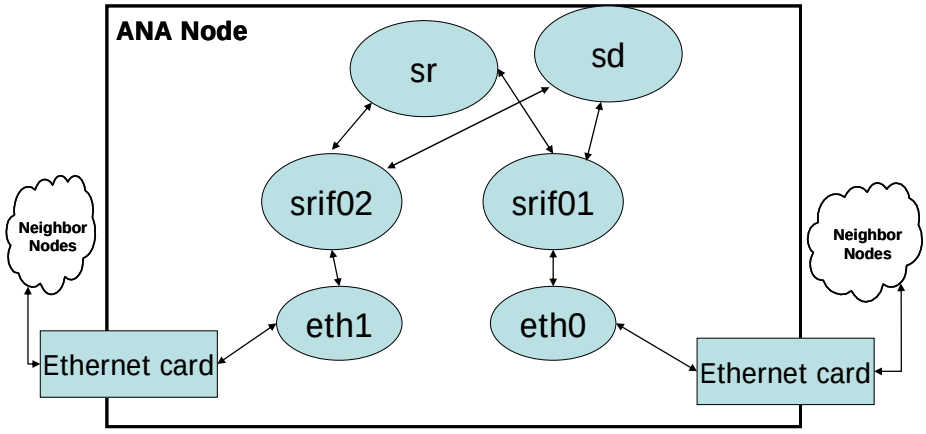


Figure 3: Example of a basic co-operation scheme between service discovery and routing.

3.4 Remarks and Conclusions

Bringing together routing and service discovery appears to be beneficial for both operations. Here a large-scale, dynamic and unstructured network environment was considered and the basic design, presented in more details in D2.13, that was proposed and implemented for service discovery was considered. Its reuse for routing purposes is straightforward and shares many similarities with the design for service discovery. A simple routing based on flooding has been implemented to show the efficiency of this design. Note that the design process does not have to obligatory propose the routing algorithm(s) that will eventually be implemented. Finally, a co-operation scheme between service discovery and routing is proposed, reusing (once more) the particular design.

4 INTER-COMPARTMENT SERVICE DISCOVERY AND SELF-CONFIGURATION

4.1 Inter-Compartment Service Discovery in ANA

The need to discover information is fundamental to any large scale and distributed environment. In the ANA context, multitude of information is required for an ANA compartment to organize itself and operate according to its objectives in autonomous ways. Such information could range from low level network configuration specifics, to inter-compartment communication protocols, to service functionalities offered by other compartments. Thus there is an apparent need for a well-designed, robust, efficient and autonomous information management system that not only could satisfy essential service discovery requirements but also advanced knowledge management functionalities. We tentatively refer to it as ANA Inter-Compartment Service Discovery (ICSD) system. The design requirements of ICSD are largely dependent on the operating context and assumptions made on the environment. For example, the SLP service discovery protocol assumes all services belong to some standardized types with structured service descriptions because the type of services they support are relatively few and well defined; DHT-based service discovery systems assume the information it stores are relatively stable and valid over long period of time; Bluetooth uses broadcasting protocol for its service discovery due to its wireless local area context, etc. Hence we have grounded our investigation of the ICSD design requirement based on the ANA requirement document and what knowledge management functionalities ANA requires. Of particular importance, we work under the following two contexts. 1) The “compartment” concept of ANA emphasizes on that no single compartment would satisfy all requirements from future networks and thus multiple Internets or compartments could co-exist, each with their own set of policies, protocols, and functionalities. Therefore, ICSD must help facilitating the service discovery and knowledge exchange process across heterogeneous compartments. 2) The range of information that could be useful and therefore considered “knowledge” is vast, such as device configuration, network access interface and protocol information, service descriptions, etc. Hence, there needs to be an efficient structuring of the information representation and storage.

In this part of deliverable, we present an inter-compartment service discovery system based on our investigation and prototyping of an open service discovery architecture. The system offers an scalable efficient and fault tolerant solution to facilitate heterogeneous cross compartment service discovery in ANA. Moreover, we present an self-configuration and self-optimization scheme under which the ICSD system can obtain optimal system performance and minimize service discovery and query latency.

4.2 Inter-Compartment Service Discovery (ICSD)

In our previous work [26] we have studied in detail various current approaches to service discovery and have defined a set of design requirements for a large-scale cross-domain service discovery system. From these requirements we have derived evaluation criteria,

against which we have compared existing service discovery systems. The comparison criteria include fault-tolerance, performance, scalability, interoperability with other discovery systems, platform independence, standardization and availability of a mature implementation. The problem of service discovery in heterogeneous service domains are quite akin to the issues we face in ANA inter-compartment service discovery. Our study has revealed that no existing discovery system satisfies all of the requirements, but many of them contain desirable characteristics. As the result, we have designed a new inter-compartment service discovery system that can provide a scalable and efficient model for cross compartment service discovery by allowing service providers and consumers to transparently discover services advertised outside their own compartment boundaries using their compartment-specific service discovery mechanisms. ICSD achieves this goal by establishing an inter-compartment distributed information storage and querying model and a unified information representation. ICSD incorporates some of the most useful elements of existing discovery systems and service infrastructures in the design of our proposed architecture. The extensibility and scalability of our system is ensured by its stateless, modular, loosely coupled components and the use of well-accepted, web-based technologies.

4.2.1 The ICSD general architecture

In this report, we use the term domain and compartment interchangeably since for the purpose of service discovery, the concept of a service domain today is similar to the concept of ANA compartment. We build on the existing domain-specific discovery systems by providing the following facilities:

- As an alternative to mirroring shared services or broadcasting queries in all the involved domains and networks (which do not scale with an increasing number of compartments and services), a structured peer-to-peer overlay is created as an inter-compartment and inter-network space where shared services are advertised and queries are solved.
- Programmable service brokers are deployed in compartments to act as an interface between the intra-compartment and inter-compartment discovery systems.
- As an alternative to converting service advertisements to all involved service description schemes, a Uni¯ed Service Description scheme is used for the advertisement of services in the inter-compartment space. In the following section, we will describe the high-level architecture of ICSD. Later, we will introduce USD, the Unified Service Description schema used to advertise and query services in our framework.

In Figure 4 we present a high-level overview of ICSD. The architecture assumes that the following local service discovery components are in place:

User Agent: acts as an interface between the end user and the discovery system. It is dependent of the compartment specific discovery technology.

Service Agent: interfaces between the service provider and the discovery system. It handles service advertisement, advertisement renewals and/or de-registrations.

Local Service Registry: an optional component whose presence depends on the intra-compartment service discovery mechanism. It is responsible for storing service records and responding to service discovery requests. It is worth noting that there can be zero or more

service registries in a compartment depending on the service discovery mechanism used in the compartment.

- If the compartment uses a fully decentralized local service discovery mechanism (such as UPnP), then there may be no Service Registry in the system.
- If a directory-based local service discovery mechanism is used in the do main, then one or more than one service registry can be used in the system.

Policy Server: is an optional component responsible for providing discovery policies that control the publication of service advertisements and service queries outside the compartment boundaries. These policies are compartment-specific and are implemented by compartment administrators. As stated before, we introduce in each compartment one or more than one service brokers and build a peer-to-peer indexing network in the inter-compartment space. The canonical setup would involve one broker per compartment and a one-to-one association between brokers and indexing peer nodes, however, the architecture supports multiple brokers per compartment and many-to-many association between brokers and peers.

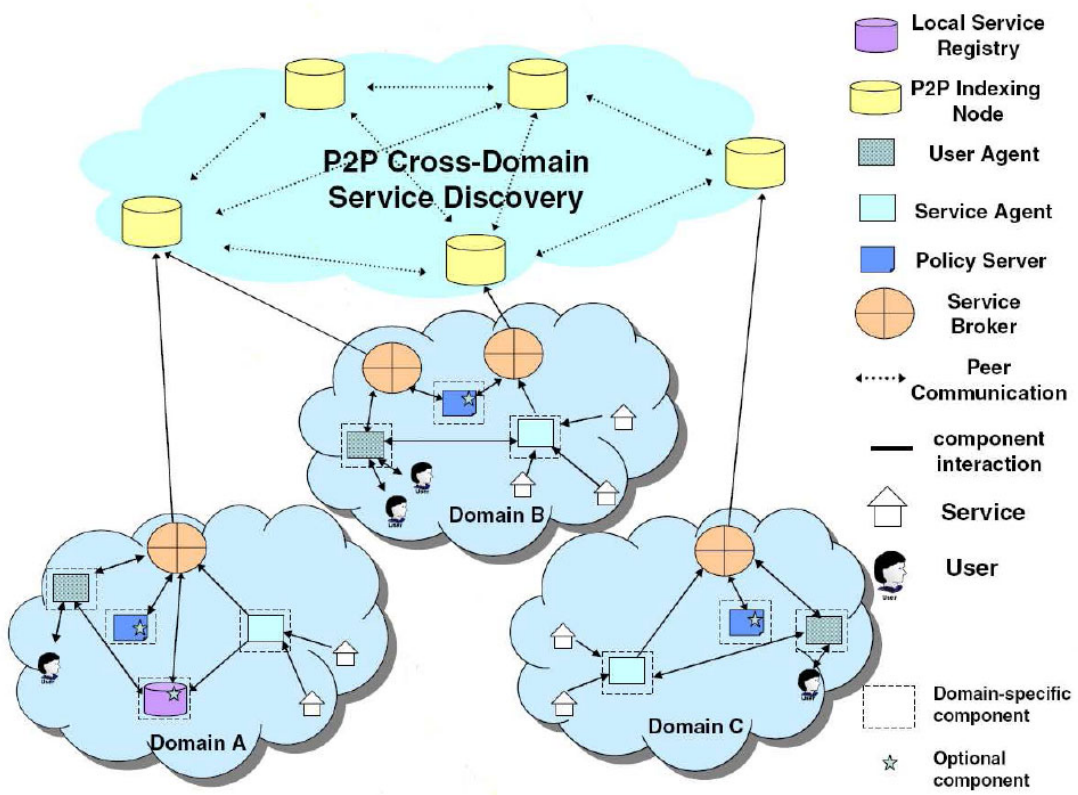


Figure 4: ICSD: High Level Architecture

The following components are integral parts of our system:

Service Broker: is responsible for handling and processing inter-compartment advertisements and queries. It acts as an interface between the local discovery technology and the inter-compartment discovery system. The broker can be divided into two layers; a technology-dependent lower layer and a technology-independent upper layer. The technology-dependent layer (referred to as an *Adapter*) abstracts the local service discovery system. It intercepts and

processes requests coming from User Agents and Service. Agents and converts advertisements and queries to a well-defined service description and query language. It also processes requests coming from the inter-compartment system and executes them in the local discovery system. The technology-independent layer handles broker-to-peer and broker-to-broker communication. It provides the necessary interfaces for the broker to be accessed by the entities involved in the inter-compartment discovery process.

Note: Since there is a fairly clear separation between the technology-dependent and the technology-independent layers of the broker, we will occasionally refer to them as two separate entities: the *Adapter* and the *Broker*, respectively.

Peer-to-peer Indexing Node: is responsible for distributing the service information in the peer-to-peer discovery network and allowing the discovery of services across compartments. The peer-to-peer network uses a Distributed Hash Table (DHT)-based architecture to store service information and solve queries. Our system supports the two main functions of a service discovery system: advertisement and querying.

Advertisement: The adapter composes an inter-compartment advertisement based on a service description and additional meta-information, like the expiry time of the service, and sends it to the broker. The broker will complement the advertisement with additional meta-information, typically its URL and the URLs of those brokers responsible for processing queries that relate to the advertised service. The advertisement is then mapped to a unified advertisement message and sent to the peer-to-peer overlay, which distributes the service information among selected nodes.

Querying: Querying in ICSD is a two-step process. First, upon interception of the query for a service, the adapter in the compartment looks into the compartments policy in order to decide to publish or not the query in the inter-compartment space. The query is expressed in the compartments specific discovery technology. Similarly to advertisements, the query will need to be mapped to a unified query message and then sent to the broker. The broker forwards the query to a peer indexing node so that it is resolved in the peer-to-peer overlay. It will receive back the set of broker URLs that have been advertised along with the services matching the query. Later, this same broker contacts one or more brokers from the received list in order to retrieve the whole information about the offered service, especially its access point. This second step involves the contacted broker to execute queries in the compartment's local discovery system.

Splitting up the querying process in this way allows the compartments to control which parts of service information are advertised to the world at large, and which parts are made available only to selected compartments. It also allows the compartments to control the amount of data forwarded to the peer-to-peer overlay by sending a single inter-compartment advertisement for a set of similar services (aggregation). The resulting architecture acts as a unifying glue that connects diverse local service discovery systems such as SLP, Jini or Salutation. The inter-compartment service discovery system does not depend on the local service discovery mechanism (i.e., both centralized and decentralized service discovery approaches are supported). If a directory-based local approach is used, then the service registry will be contacted for local service discovery. Otherwise, multi-cast or broadcast messages will be sent for local discovery.

4.2.2 Service naming, description and querying

Each service discovery system has its own way of describing a service. Supporting inter-operability among a variety of service discovery systems needs some means of vocabulary translation. Vocabulary translation can be carried out directly from one technology to the other: a Jini advertisement (or service description) can be converted to an equivalent SLP advertisement and advertised in the SLP system. However such a scheme would require $O(N^2)$ mappings, where N is the number of supported service discovery technologies. Clearly, the preferred method would be to design an intermediate, unified scheme for inter-compartment advertisements. In this case, an advertisement from a particular compartment can be translated to the agreed inter-compartment format and can then be advertised in another technology after another step of conversion. Because of its expressive power and acceptance in the Internet community, XML seems to be most appropriate as a basis for such a format, while the selection of appropriate elements for the XML description needs more detailed analysis of the existing service discovery technologies. To achieve this kind of interoperability, we propose the Unified Service description (USD), a meta-service description schema that can interoperate with most of the major service discovery systems.

As shown in Figure 5, the USD scheme consists of two major nested parts. The main envelope contains the meta-information for the advertisement, such as the type, location and expiry time of the service. The *description* component specifies the properties and capabilities of the service itself. For better understanding of the discovery process, let us focus on two key fields of the Unified Service Description: *serviceID* and *description*.

serviceID: In ICSD, a service identifier consists typically of two parts: a globally unique compartment identifier, and a compartment-unique service identifier. Together, these two parts form a globally unique identifier for each service, obviating the need for a central naming authority that assigns names to each participating service. The format of the service identifier is left up to the compartment administrator, giving each compartment discretion in naming its services. The compartment identifier is more difficult to define, since it must be globally unique. In the case where compartment implies an administrative domain, we recommend the use of DNS names. In other cases, uniqueness can be accomplished by using a Universal Unique Identifier (UUID). If a non-human-readable compartment identifier is used, such as the UUID, we recommend the use of the “Domain Name” field to include a human-readable *Domain Name* with the USD.

description: ICSD supports any description schema (or service template) provided it is written in XML. Typically, service templates are generated by service providers and may be commonly used by a number of service providers as the standard template for inter-compartment advertisements.

In addition to the meta-service description scheme, we also design two XML message formats: *Unified Request* and *Unified Response* used, respectively, to wrap inter-compartment advertisement or discovery requests, and discovery responses.

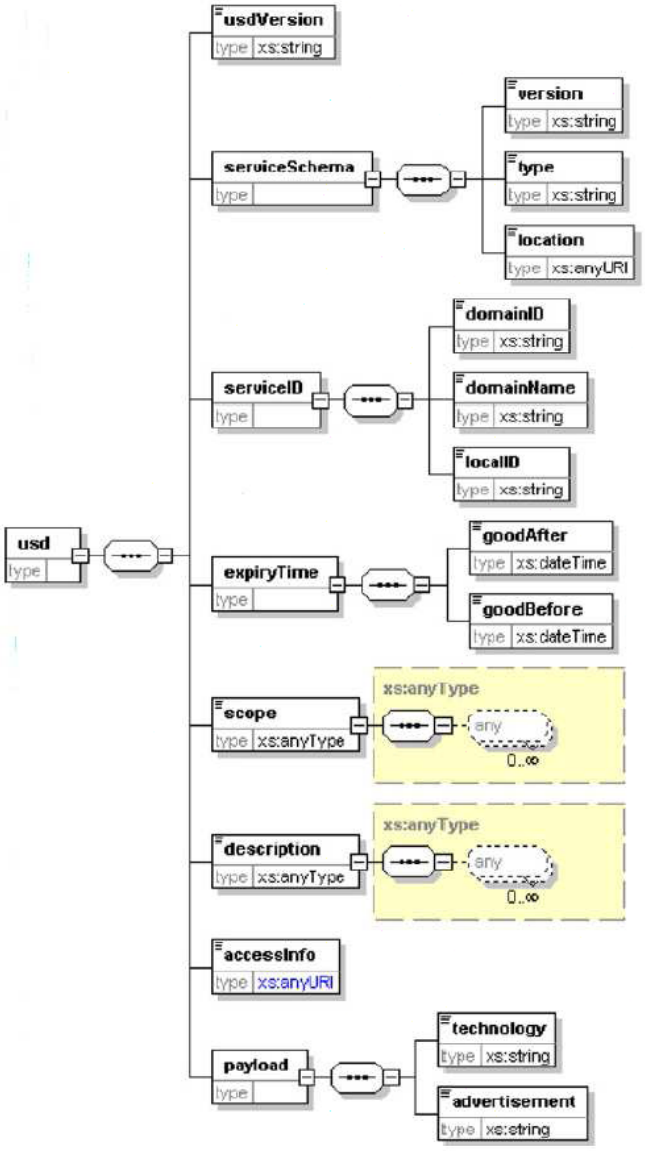


Figure 5: Unified Service Description

4.2.3 ICSD module design

ICSD is comprised of several components. Each of these components consists, in turn, of a number of smaller modules. The high-level objective of the design of ICSD modules is maximizing the system modularity through a clean separation of responsibilities between components and a clean identification of functionalities for each component in order to reducing maintenance e®ort and code duplication. In the following section, we will examine the module level design of ICSD (see Figure 6), by explaining the functionalities and interfaces of the individual modules.

4.2.3.1 The adaptor

The adapter is composed of the following four modules: the Registration Advertisement Handler, the Discovery Request Handler, the Directory Handler and the Converter.

Registration Advertisement Handler. The Registration Advertisement Handler is responsible for processing advertisement requests coming from local Service Agents. Upon interception of an advertisement, the Registration Advertisement Handler contacts the policy server (if present) so that related compartment policies are applied. If the advertisement is allowed to be propagated in the inter-compartment discovery system then the Advertisement Registration Handler first converts it to the USD format and then submits it to the Brokers Advertisement Propagator.

Discovery Request Handler. This module is responsible for intercepting and handling service discovery requests. It steers queries either to the local or global discovery systems according to the compartment policies. If a service discovery request is allowed to be propagated to the inter-compartment discovery system, it is first converted to the USD-based query format, and then submitted to the Brokers Global Discovery Handler. Similarly, upon reception of a response to a query, the response is first converted to the local description format and then forwarded to the User Agent that generated the query.

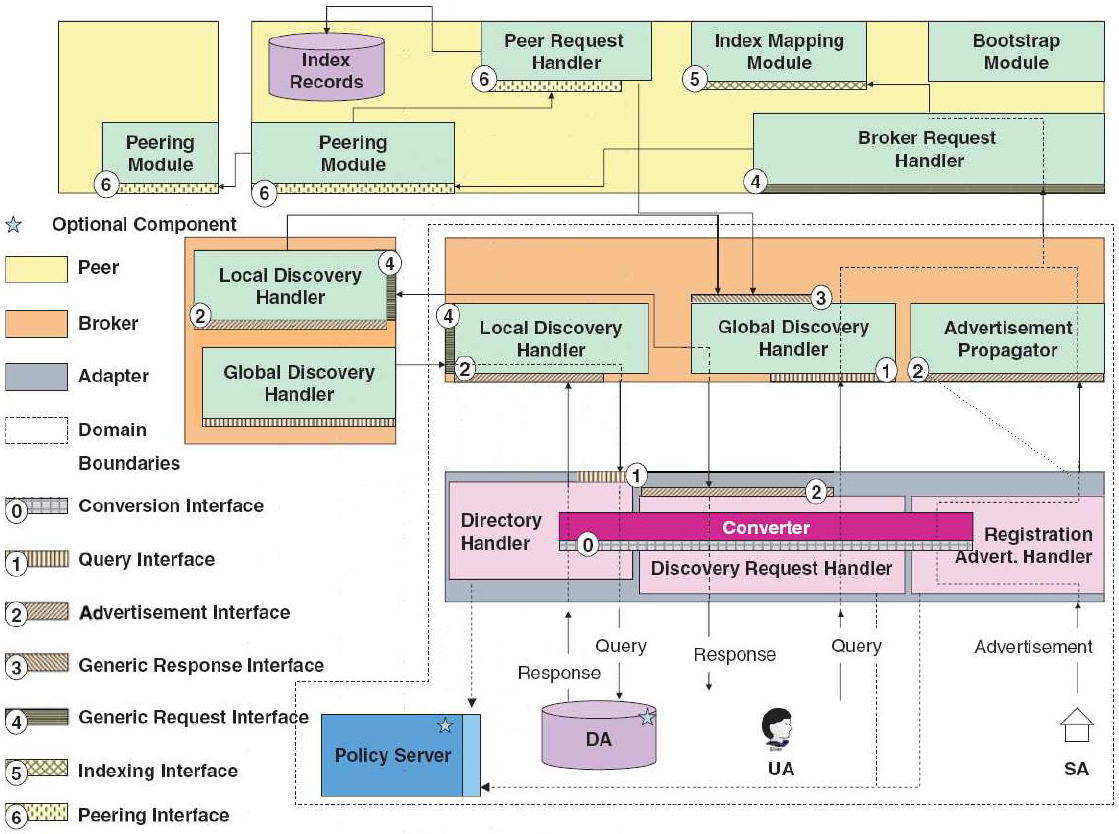


Figure 6: ICSD Modules

Directory Handler. The Directory Handler is responsible for handling Brokers service discovery requests in the second step of the querying process. It takes care of converting queries into the local query format and generating discovery requests in the local discovery system. The Directory Handler implements a user agent interface; generated queries are either sent to the service repository or multicasted/brodcasted over the network depending on the discovery mechanism deployed in the compartment.

Converter. The Converter module is used to convert queries and service descriptions into a USD-based format, or from the USD-format to the local format. A mapping between local service templates and USD templates as well as a mapping between the local query format and the Unified Query format are both required in the conversion process.

4.2.3.2 The broker

The Broker is responsible for handling cross compartment advertisement and service discovery requests. It provides well-defined interfaces that are invoked to post requests and responses. The broker is composed of the following three modules: Advertisement Propagator, Global Discovery Handler and Local Discovery Handler.

Advertisement Propagator. The Advertisement Propagator is responsible for propagating advertisements to a peer-to-peer indexing node.

Global Discovery Handler. This module is responsible for processing inter-compartment service discovery requests. Inter-compartment service discovery is achieved in two steps. In the First step, the Global Discovery Handler is responsible of the propagation of service discovery requests to the peer-to-peer network. In the next step, the Global Discovery Handler sends the same request to one or more than one broker listed in the response of the peer-to-peer network. A selection mechanism over the received list may be implemented. Responses from the contacted brokers will contain the USD of services that match the discovery request. The Global Discovery Handler can also be contacted directly, by services and clients, without having to go through a pre-existing local discovery system.It is worth noting that the number of contacted brokers influences both the inter-compartment query overhead and the completeness of the query result.

Local Discovery Handler. The Local Discovery Handler is responsible for processing the requests sent by remote Global Discovery Handlers. Queries embedded in the received requests are sent to the Directory Handler in order to be solved into the USD of matching services. USDs are sent back to the Global Discovery Handlers that generated the requests.

4.2.3.3 The peer

The network of ICSD peer nodes forms a distributed hash table. The nodes are responsible for storing index records corresponding to previous inter-compartment advertisements, and for solving cross compartment queries. A peer node is composed of a Broker Request Handler, an Index Mapping Module, a Peering Module, a Peer Request Handler and a Bootstrap Module.

Broker Request Handler. The Broker Request Handler is the access point to the peer-to-peer network. It intercepts and processes inter-compartment advertisement and query requests sent

by Brokers and then direct them in the peer-to-peer network. Given a service advertisement or query, the Broker Request Handler first internally extracts from the capability description section the data that will serve to index the request. The data is then passed on to the Index Mapping module, to convert it into one or more hash keys. Each hash key, along with the original request, is passed to the Peering Module in order to be routed to the appropriate peer.

Index Mapping Module. The Index Mapping Module uses a hash function to convert the data received from the Broker Request Handler to a key. The hash is returned back to the Broker Request Handler.

Peering Module. Each peer in the peer-to-peer overlay is responsible for a set of keys. Upon receiving a key request pair, the Peering Module routes the request to the appropriate peer. If the given key is one of this peers own keys, the key-request pair is passed to the local Peer Request Handler.

Peer Request Handler. A peer-to-peer node maintains a database which associates each hash key with one or more than one index record. Upon receiving a request-key pair, the Peer Request Handler either executes the contained query or stores the contained advertisement in the index record database within the set of index records associated with the given key. Before being executed, a query is first converted to the format supported by the database. The advertisements stored in the peer-to-peer network are soft-state: each stored index record contains the information about its lifetime. While the advertisement renewal process is handled by the Discovery Request Handler at the compartment level, cleaning the peer database from expired index records must be handled by the Peer Request Handler if not supported by the used database management system.

Bootstrap Module. The Bootstrap Module is responsible for joining the peer-to-peer network. The join process is specific to the DHT architecture used in the peer-to-peer network.

It is worth noting that, the inter-component communication is asynchronous and the modules are stateless, i.e., they do not keep track of current requests. This enhances the systems tolerance to short failures or disconnections. For example, if multiple brokers are known by an adapter, they can be used interchangeably if one of them fails. Moreover, since information about the source of the query is embedded in each request, a broker can be restarted or replaced between sending the query and receiving the responses. If a broker goes down while waiting for query responses to arrive, those responses can be sent to an alternate broker (if known), or to the originating broker, if it becomes accessible in the meanwhile. All ICSD components, and most of the modules are designed to be loosely bound. Contingent on access-control policies, an adapter can communicate with any broker in its compartment (allowing several adapters per broker, or vice versa), while a broker can communicate with any peer indexing node. Such extensibility and degree of fault-tolerance is mandatory, since ICSD is designed to be a core supporting function for an Internet-scale network of resources and services.

4.2.4 Implementation technology

The main goal of ICSD is to be as open and universal as possible. We used this motivation as a guide in our choice of tools and technologies that would implement our system. We focused on well-known, tested, freely available and open-source components such as JXTA, Chord, JBoss, INS/ Twine, Jetty and web-based technologies such as SOAP. Because of the need for platform independence, we used Java as the programming language, HTTP as the transport protocol for broker-to-broker and broker-to-peer communications and XML as the format for all communications. In addition to greatly simplifying the implementation process (in comparison to re-inventing the communication/data format wheel), using primarily web-based technologies allowed us to create an extensible and modular system. If needed, many of the components, such as the index record database, or the peer-to-peer routing mechanism can be relatively easily replaced with other technologies. Moreover, because of the loose coupling between components, the individual parts of the system can be flexibly deployed on separate systems or even all on the same machine, allowing ICSD to scale gracefully in face of increasing load.

Broker: The broker functions as a stand alone Enterprise Java server. The broker modules are implemented as stateless-session Enterprise Java Beans (EJBs) running in the JBoss application server, and deployed as Web Services. This way, the broker components may be invoked either using RMI/IIOP or SOAP/HTTP. Currently, Local and Global Discovery Handlers are accessed by peers and other brokers, cross compartment boundaries, through SOAP/HTTP which offers many attractive advantages like the support of security mechanisms and the ability to work through firewalls. One of the most interesting advantages of this implementation is that it does not require a local service discovery system to be deployed in a compartment. Because the Global Discovery Handler is implemented as a Web Service, a web client can be easily created and then used to discover services in other compartments.

Peer Nodes:
The *Broker Request Handler* is implemented as a SOAP service and runs on top of the lightweight Jetty HTTP server. We use INS/Twine libraries to extract from advertisements and queries the data that will be used to generate indexing keys. The service capabilities part of advertisements is split into strands, using the INS/Twine model, and all strands are hashed into keys by the Index Mapping Module. Advertisements are routed to and then stored/replicated in each peer corresponding to one of the resulting hash key. On the other hand, queries are forwarded to the peer that is associated to the key resulting from the longest strand.

Index Mapping Module: The Index Mapping Module uses an MD5 hash to turn strands into hash keys later used for routing.

Peering Module: The Peering Module is implemented as a JXTA peer to take advantage of the bootstrapping, authentication, and secure communication facilities of the JXTA environment. Peering Modules are organized in a Chord ring, where Chord is used to route requests between peers. We chose Chord as a base for peer-to-peer network because of its fault-tolerance and self-stabilizing properties as well as its effective method of evenly distributing information and query process load. This design is necessary in dealing with the large volume of advertisement messages and query operations expected in an Internet-scale

network. Below, we provide a short analysis of the query costs in the ICSD implementation. Unlike broadcast-based approaches, the Chord DHT guarantees an upper bound of $O(logN)$ hops between peers for each routed request, where N is the number of peers in the overlay. After being advertised locally, each advertisement generates k strands, and hence k keys to be distributed among the peer-to-peer nodes, resulting in $O(klogN)$ messages. The advertisement overhead therefore combines the cost local advertisement with the peer-to-peer advertisement overhead. Additionally, since the peer-to-peer layer uses a soft-state approach, each advertisement must be periodically re-advertised to remain valid, which makes the total overhead dependent on the frequency of advertisement renewal. Because a query generates a single strand, the peer-to-peer query overhead is lower at $O(logN)$ messages.

Peer Request Handler : The Peer Request Handler stores and retrieves XML-based index records to and from a database. For this purpose, we used the eXist open-source native-XML database. Like JXTA, and our SOAP interfaces, eXist relies on the lightweight Jetty HTTP server, and includes many useful features such as a web interface and XPath/XQuery processing. While the canonical INS/Twine search would return all service descriptions corresponding to the given key, the Peer Request Handler goes a step further by converting the original query (possibly containing complex predicates currently, XSet range predicates are supported) into an XQuery. The XQuery is then performed against the index records corresponding to the given key, returning only the relevant set of matching documents.

4.2.5 Key features of ICSD for ANA

The design of ICSD meets the main requirement for inter-compartment service discovery in ANA. A number of key features of ICSD are:

- Scalability: the filtering and aggregation features of the service broker allow ICSD to scale with an increasing number of advertised services. In addition, the use of a structured peer-to-peer overlay, as opposed to unstructured systems, reasonably bounds advertisement and query routing at the inter-compartment layer, and hence minimizes the communication overhead.
- Openess: service brokers make ICSD a pluggable solution that does not require any change in the local discovery systems and that is extensible to new participating compartments.
- Fault-tolerance: service registration in the peer-to-peer overlay is soft-state; advertisements are evicted if the corresponding services are not re-advertised. This way ICSD guarantees the consistence of query responses and is tolerant to service failures. In addition, given that brokers an peers are stateless and loosely coupled, ICSD is able to recover their failures.

The implemented ICSD prototype maximizes these features. In fact, in addition to choosing open and well-accepted technologies, we have designed standard interfaces and asynchronous messaging protocols. The resulting system has been tested an validated through some case scenarios.

4.3 Self-Configuration and Self-Optimization in ICSD

An essential part of the ICSD design is the DHT based P2P overlay. The ability to configure an efficient and fault tolerant P2P network is critical to the operation of ICSD, and thus the natural points of investigation for autonomic networking. In this section, we describe a server placement problem for automatically selecting Peer nodes in ICSD such that the advertisement and query latency are bounded. We terms our general problem formulation as the server placement problem.

4.3.1 The server placement problem

In this section we introduce the server placement problem (SPP) we formulated for selecting DHT-based Peers (termed Servers from here on) in ICSD. Given a network domain-level topology graph, our objective is to place a set of servers in the graph, such that the total latency for advertisement and query is minimized. To formulate the problem mathematically, we make the following assumptions:

1. Data is replicated such that the path length of individual lookup request is constant. It is well-known that, the popularity of web objects, including service records, follows a Zipf-like distribution, i.e. a few objects are highly popular while most of the remaining objects are unpopular. Hence, a direct implementation of a DHT (such as pastry) for DNS record may result in hot-spots in the overlay. This problem can be addressed using proper caching and replication techniques, as described in [27, 28]. Furthermore, [27] presented a caching strategy to achieve *O(1)* lookup path length. In the formulation we present below, we assume the lookup path length is *O(1)*. However, it is straightforward to modify our solution for replication strategies that require $O(\log n)$ length.
2. The routing (forwarding) load of individual servers is roughly the same. This is a desired condition since we wish to achieve load balancing among the servers. For any proximity based DHTs, e.g. pastry and tapestry, this can be achieved by selecting neighbors based on routing load rather than physical proximity [28]. A direct consequence of this assumption is that, probability that an individual will be reached is the same for all the servers.

Mathematically, we can represent the Internet topology as a graph $G = (V,E)$. For each ICSD broker i, we denote s_i in S as the server of i where S is the set of deployed servers. Each client i has a search frequency q_i. Based on the previous 2 assumptions, suppose the lookup path length is l hops, and each node have equal probability of being reached. The total query latencies for all the search operations over unit time can be expressed as:

$$Cost_{query} = \sum_{i \in V} q_i (d(i, s_i) + \frac{l}{|S|} \sum_{s \in S} d(s_i, s))$$

In addition, each server s has a capacity constraint *caps* that upper-bounds the number of clients it is capable of supporting. We also define a placement cost f_s deploying the server at node s. Hence the server placement problem can be stated as:

Server Placement Problem (SPP): Given a graph $G = (V,E)$, a set of potential server locations F, assume there is an opening cost f_s and a capacity constraint *caps* for each s in F, we wish to find a set of super-peers S subset of F and assign one server to each node i in V to minimize the total cost

$$Cost = \sum_{i \in V} q_i d(i, s_i) + \frac{l}{|S|} \sum_{i \in V} \sum_{s \in S} u_i d(s_i, s) + \sum_{s \in S} f_s$$

SPP is NP-hard, since by setting $c = 0$, the problem becomes the capacitated facility location problem (CFLP), which is NP-hard. Furthermore, when the facility capacity constraint must be strictly obeyed, the resulting hard-capacitated CFLP problem is difficult to solve using linear-programming techniques due to an unbounded gap between the fractional solution and the integer solution. The current best algorithm achieves constant approximation ratio using iterative search techniques [29]. Our problem di®ers from CFLP due to the inclusion of the communication cost between the servers in the objective function. In fact, our problem resembles the capacitated single allocation hub location problem [30], which is also known to be NP-hard. Existing solutions either rely on branch-and-cut techniques or using meta-heuristics such as simulated annealing [31]. However, in the next few section, we show that SPP is a special case that can be reduced to CFLP while losing a constant approximation factor.

4.3.2 An approximation algorithm for SPP

In this section, we present our approximation algorithm for SPP. There are two key challenges for solving SPP. First, the term $d(s_i, s)$ is non-linear when written in an Integer Program (IP) form. Second, the value of $1/|S|$ is dependent on the solution S. Hence traditional algorithms for CFLP cannot be directly applied to solve SPP. To address the first challenge, we reduce SPP by a constant approximation factor to a simpler problem called *Star-Facility Location Problem (SFLP)*. For the second challenge, we perform a iterative search on value $|S|$. Using both techniques, we can derive a constant factor approximation algorithm for SPP.

We first formally state the SFLP problem as follows: Given a graph $G = (V,E)$, a set of facility candidates F, each with unique capacities (i.e. $caps_s$ for all s in F). Let u_i and c be defined as above, let L denote the weighted 1-median of the graph. We wish to find locations of facilities S, and assign each node i to one of the facilities s_i to minimize following cost function while satisfying the capacity requirement of each s in F:

$$cost_{SFLP} = \sum_{i \in V} u_i^* d(i, s_i) + \frac{c^*}{|S|} \sum_{s \in S} d(s, L) + \sum_{s \in S} f_s$$

Lemma 3.1 *Every r-approximation algorithm of SFLP is also a* $\max(r+3cr+c;\ 2r+1)$*-approximation algorithm of SPP.*

Proof omitted.

However, SFLP is still difficult to solve due to the $1=|S|$ component in the objective function. Our strategy is to perform an iterative search on the value of $|S|$. Suppose we pick a value k for $|S|$, then the SFLP for this value of k becomes a k-capacitated facility location problem (kCFLP), of which the objective is to open exactly k facilities, each having opening cost (c/k) $d(s,L)+f_s$, to minimize the total connection and facility opening cost

$$cost_{kCFLP} = \sum_{i \in V} u_i d(i, s_i) + \frac{c}{k} \sum_{s \in S} d(s, L) + f_s$$

Suppose we have an *r*-approximation algorithm for *k*CFLP, we can repeatedly solve *k*CFLP for every *k* between 1 and |*F*|. The solution with the smallest cost would be an *r*-approximation of the optimal solution of SFLP. Although, *k*CFLP is still a complicated problem to solve, the following lemma shows there is a simple way to approximate the optimal solution of SFLP at a fixed value of *k*. Let OPT_k^{SFLP} and SOL_k^{SFLP} denote the optimal cost and cost of the solution produced by the algorithm, respectively. We can show that

Lemma 3.2 *For any k, 1 ≤ k ≤ |F|, we can construct a feasible solution for SFLP such that the cost is no more than* (1 + *r*) *times* OPT_k^{SFLP}.

Lemma 3.2 details a technique for solving SFLP: We can exhaustively construct feasible solutions of SFLP for every *k* between 1 and |*F*|. The solution that produces the smallest cost among all the solutions is a (1+*r*) approximation of this SFLP instance. If we set *r* = 5.83 according to the current best algorithm for CFLP [29], we have a constant approximation algorithm for SPP.

Theorem 3.3 *There is a max*(5.83 + 18.49*c;* 12.66) *approximation algorithm for SPP.*

Furthermore, with $f_k^s = (c/k)\, d(s,L) + f_s$ and *N*(*k*) denoting the *k* facilities having the least values of f_k^s , the following result is a direct consequence of Lemma 3.2:

Corollary 3.4 *For any k, 1 ≤ k ≤ |F|, we have:*

$$SOL_k^{CFLP} + \sum_{s \in N(k)} f_k^s \leq (1 + r) \cdot OPT_k^{SFLP}$$

However, the above algorithm requires solving CFLP repeatedly for every *k* between 1 and |*F*|. As an problem optimization technique, in the next section we show that it is possible to obtain high quality solutions that only requires solving CFLP *O*(log *n*) times. This technique will significantly reduce the running time of the algorithm.

4.3.2.1 Searching for optimal *k*

In this section we present a technique to find a suitable *k* value in *O*(log *n*) iterations. Recall that $SOL_k^{CFLP} + \sum_{s \in N(k)} f_k^s$ gives an (1 + *r*)-approximation of the solution for SSP at *k*. It is easy to see that there is a trade-off between SOL^{CFLP} and the cost of opening *k* cheapest facilities. This trade-off is conceptually depicted in Figure 7: as *k* increases towards |*F*|, the cost of SOL^{CFLP} drops due *c*=|*S*| factor in the objective function. At the same time, however, more facilities in *N*(*k*) must be opened in the solution. Hence our objective is to find a trade-off point where the cost is minimized. However, in practice, the real trade-off curve may not be smooth like the one shown in Figure 7, due to the approximation algorithm used. In fact, there may be multiple local minima at different values of *k*. Hence it is difficult to find the optimal value of *k* without testing every *k* by brute force. However, it is possible to approximate the optimal value of *k* by computing the solution at *O*(log *n*) values, using an iterative search algorithm.

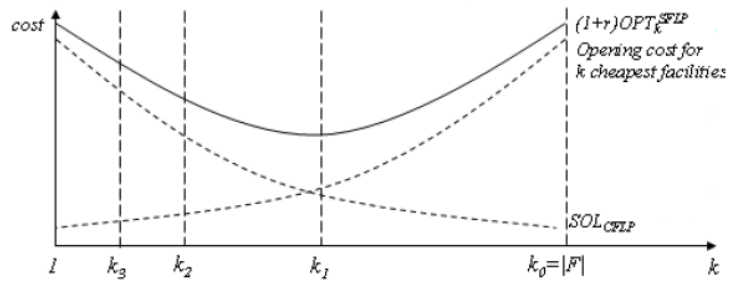


Figure 7: Trade-off Curve

Our search algorithm can be described as follows: We start with $k_0 = |F|$, and compute a feasible solution SOL_k^{SFLP} using Lemma 2 as our initial solution. Then, in the next round, we set $k_1 = k_0/\alpha$, where α is an adjustable constant, and compute a feasible solution SOL_{k1}^{SFLP} for the k_1. This process repeats for k_2, k_3 ..., each iteration k is divided by α. This search process ends when SOL_k^{CFLP} becomes greater than SOL_{k0}^{SFLP} for some previous value of k_0, or some other constraints, such capacities, prohibits k to be further reduced. Finally we select the k that produces the smallest value of SOL_k^{SFLP}. This solution is the output of our algorithm for SSP. Since we only compute $\log_\alpha |F|$ solutions in our search algorithm, it is necessary to show that these $\log_\alpha |F|$ solutions are able to bound the cost of optimal solution by a constant factor.

Lemma 3.5 *For any i and m between k_i and k_{i-1}, we have* $SOL_{k_i}^{SFLP} \leq \alpha(1+r)OPT_m^{kCFLP}$

Lemma 3.5 implies that the best solution out of the $\log_\alpha |F|$ solutions is an $\alpha(1+r)$ approximation of the optimal solution. In addition, note that α is an adjustable constant, meaning that we can obtain better approximation by decreasing α, or improve the running time of the algorithm by increasing α. Nevertheless, the number of steps performed is always logarithmic with respect to the number of candidates.

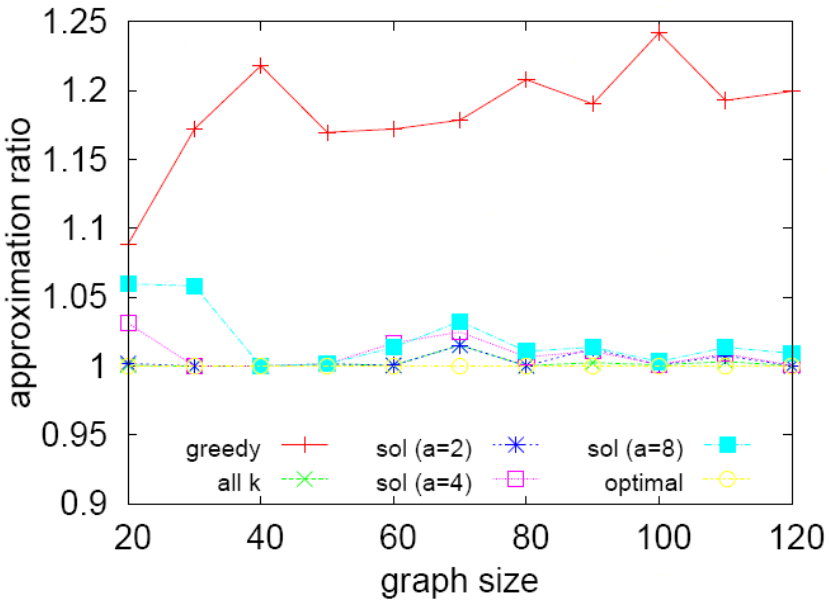
4.3.3 Simulation results

We have evaluated the performance of our proposed SPP algorithms. We construct topology graphs using the GT-ITM generator [32], which can generate transit-stub topologies that simulates latencies between hosts on the Internet. We specify the average communication latency at intra-transit, stub-transit and intra-stub domain links to be 20ms, 5ms and 2ms respectively [33].

In our experiment, we constructed transit-stub models of sizes between 20 and 140. For each model, 20% of the nodes are randomly selected as candidate locations for placing name servers. For each graph, we compute the optimal solution, the solution that obtained by computing CFLP for every possible k values, and solutions with α between 2 and 6. We also compare our solutions with the solution produced by a greedy CFLP algorithm that does not consider communication among servers. We use $c = 0.5$ and $f_s = 10$ for all graph instances. Figure 8 illustrates the approximation ratios of all the algorithms. Our centralized algorithm consistently outperforms the greedy CFLP algorithm. Furthermore, it can be observed that the

solutions with α between 2 and 6 are comparable to the solution obtained by exhaustively computing all k values.

Figure 8: Approximation Ratio



4.3.4 Summary

The performance of the DHT-based Peer overlay is critical to the performance of the ICSD system. In this section, we studied the problem of automatically configure such DHT-based server net in large scale networks that can minimizes the average communication latency. Specifically, we presented a mathematical formulation of the server placement problem, taking into consideration the communication between Peers. We also devised an approximation algorithm for this problem, and showed the effectiveness of the algorithm through simulation.

5 SUMMARY AND CONCLUSIONS

The main focus of Task 2.6, as presented in this deliverable, was in the design and implementation of efficient routing systems. It considers the basic properties enumerated in the deliverable 2.8 talking about the main issues that have to be satisfied by any autonomic routing scheme to be adapted to the ANA framework.

In particular, the CCR design that was described here assumes a large-scale network environment and tries to solve problems like routing contents between users that use different addressing schemes to locate content hosts. Concerning about the routing-based service discovery, it assumes large-scale, dynamic and unstructured environment, which is typically the environment in which we aim to adapt the ANA framework.

The role of this deliverable document is not only to propose separate protocols for different routing mechanisms, but also to integrate them in a more global routing service. Different frameworks were presented and it was described the main entities and conceptual mechanisms needed to provide in one side, content routing mechanisms, and in another side, service discovery based routing using the ANA framework and new networking paradigms presented in the blueprint [5].

In content centric routing, the content requested by nodes is more important than the content host location of the path towards it. This is a view shared by SD routing as the service asked by a node is more important than is location of the path to its host. CCR approach is based on the publication of every content to the network, just as the done in sd routing for services. We can state then that service discovery based routing and content centric routing share similar objectives and, therefore, the mechanisms used in each of them may be shared with the other. For example, the mediation infrastructure used in the CCR protocol may be exploited in order to disseminate and filter the sd routing requests. Another example of integration is the use of the information returned by the sd routing to the client which cloud be used to gather filtering metrics and update them. Instead of having a proactive collection mechanism of those metrics as we stated in the section 2, we may take advantage of the sd routing information and the requests disseminated in the network to also disseminate the filtering information.

Concerning implementation, we presented detailed descriptions of the implementation of the proposed protocols in section 2.5 and 3.2.3. These implementations are the proof that the ANA API could be the basis of a more various implementations and protocol deployments in ANA network. However, in the case of content routing for example, we have to say that the implementation task was not straightforward since we had to adapt the basic communication primitives that are imposed in this API to our specific protocol needs. This point lead us to schedule a future analysis of the evolutions that may be required by the API to make the implementation of such protocols easier without increasing the complexity nor affecting the others advantages that are provided by this API as described in [8].

On the issue of inter-compartment service discovery (also applicable to any form of information exchange among compartment with heterogeneous discovery/routing protocols), we have described an inter-compartment service discovery system that is capable of providing an open extensible and unified medium for information exchange. The ICSD system

integrates with ANA compartment seamlessly through a broker-based system and efficiently manages information at the global level through a dynamic DHT-based overlay. We further augment the ICSD system with self-configuration and self-optimization properties that are essential for the operation of ICSD in ANA.

6 REFERENCES

[1] T. Plagemann and V. Goebel and A. Mauthe and L. Mathy and T. Turletti and G. Urvoy-Keller, From content distribution networks to content networks - issues and challenges, Computer Communications, 551-562. 2006

[2] Y. Liu and B. Plale, Survey of publish subscribe event systems, Technical Report TR574, Indiana University. 2003.

[3] M. Pathan and R. Buyya, A Taxonomy and Survey of Content Delivery Networks, Technical Report GRIDS-TR-2007-19, The University of Melbourne, Australia. February 2007.

[4] C. Jelger and C. Tschudin and S. Schmid and G. Leduc, Basic Abstractions for an Autonomic Network Architecture, World of Wireless, Mobile and Multimedia Network, WoWMoM. 2007

[5] ANA Blueprint - First Version Updated, Workpackage 1 Deliverable 1.4/5/6, ANA Project FP6-IST-27489, Decembre 2007

[6] D2.1: ANA Requirements, Workpackage 1 Deliverable 1.2, ANA Project FP6-IST-27489, Decembre 2007

[7] D2.8: Service Discovery and Routing Schemes for intra- and inter-Compartment Service Provisioning, workpackage 2 Deliverable 2.8, ANA Project FP6-IST-27489, Decembre 2007.

[8] D1.8b: ANA Core Documentation, workpackage 1 Deliverable 1.8b, ANA Project FP6-IST-27489, February 2007.

[9] A. Keller and T. Hossmann and M. May and G. Bouabene and C. Jelger and C. Tschudin A System Architecture for Evolving Protocol Stacks, Proceedings of ICCCN 2008, IEEE. 2008.

[10] T. Heussman and A. Keller and S. Dudler and M. May, Implementing the Future Internet : A Case Study of Service Discovery using Pub/Sub in the ANA Framework, International Conference on Future Internet Technologies (CFI08). 2008

[11] F. Legendre and M Dias de Amorim and S. Fdida, Some Requirements for Autonomic Routing in Self-Organizing Networks, Proc. WAC2004, Germany. October 2004

[12] M.A. Sheldon and A. Duda and R. Weiss and J. W. O'Toole Jr. and D. K. Gifford, Content routing for distributed information servers Proc. Fourth International Conference on Extending Database Technology, Cambridge, England. March 1994

[13] F. Cao and J. P. Singh, Efficient Event Routing in Content-based Publish-Subscribe Service Networks, Infocom 2004.

[14] R. Shah, Z. Ramzan, R. Jain, R. Dendukuri, F. Anjum, "Efficient Dissemination of Personalized Information Using Content-Based Multicast," IEEE Transactions on Mobile Computing, vol. 3, no. 4, pp. 394-408, October, 2004.

[15] A. Riabov, Z. Liu, J. L. Wolf, P. S. Yu, L. Zhang, "New Algorithms for Content-Based Publication-Subscription Systems," Distributed Computing Systems, International Conference on, vol. 0, no. 0, pp. 678, 23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03), 2003.

[16] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[17] M. Castro, M. B. Jones, A-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang and A. Wolman, "An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer overlays", Infocom 2003, San Francisco, CA, April, 2003.

[18] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, “Design and evaluation of a wide-area event notification service”, ACM Trans. Comput. Syst. 19, 3, 332-383. August 2001.

[19] Z. Jerzak, and C. Fetzer, “Bloom filter based routing for content-based publish/subscribe”, In Proceedings of the Second international Conference on Distributed Event-Based Systems, DEBS '08.

[20] H. T. Kung and C. H. Wu, "Content Networks: Taxonomy and New Approaches," 2002, In The Internet as a Large-Scale Complex System, Kihong Park and Walter Willinger (Editors), published by Oxford University Press as part of Santa Fe Institute series, 2002.

[21] A. Popescu, D. Constantinescu, D. Erman and D. Ilie, "A Survey of Reliable Multicast Communication," Next Generation Internet Networks, 3rd EuroNGI Conference, vol., no., pp.111-118, 21-23 May 2007.

[22] M. Pathan and R. Buyya, "A Taxonomy and Survey of Content Delivery Networks", Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia. 12 february, 2007.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for Internet applications. In Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications SIGCOMM, 2001.

[24] D. Donato, M. Paniccia, M., Selis, C. Castillo, G. Cortese, and S. Leonardi. 2007. New metrics for reputation management in P2P networks. In Proceedings of the 3rd international Workshop on Adversarial information Retrieval on the Web, AIRWeb, 2007.

[25] C. Fengyun and J. P. Singh,. "Efficient event routing in content-based publish-subscribe service networks," INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, vol.2, no., pp. 929-940 vol.2, March 2004.

[26] R. Ahmed, N. Limam, J. Xiao, Y. Iraqi, and R. Boutaba, “Resource and service discovery in large-scale multi-domain networks”, IEEE Communications Surveys & Tutorials, 2007.

[27] V. Ramasubramanian and E. G. Sirer, “Beegive: O(1) lookup performance for power-law query distributions in peer-to-peer overlays”, Proceedings of First

USENIX conference on Networked Systems Design and Implementation (NSDI), 2004.

[28] S. Bianchi, S. Serbu, P. Felber, and P. Kropf, “Adaptive load balancing for DHT lookups”, Proceedings of International Conference on Computer Communications and Networks (ICCCN), 2006.

[29] J. Zhang, B. Chen, and Y. Ye, “Multi-exchange local search algorithm for capacitated facility location problem”, Math. of Oper. Research, 2004.

[30] B. Kara and S. Alumur, “Network hub location problems: The state of the art”, European Journal of Operational Research, 190, 2008.

[31] A. Ernst and M. Krishnamoorthy, “Solution algorithms for the capacitated single allocation hub location problem”, Annals of Operations Research, 1999.

[32] Gtitm homepage. *http://www.cc.gatech.edu/projects/gtitm/*

[33] S. Ratnasamy, M. Handley, R. Karp, and S. Scott, “Topologically-aware overlay construction and server selection”, IEEE INFOCOM, 2002.