ANA Project

Autonomic Network Architecture

Sixth Framework Programme

Priority FP6-2004-IST-4

Situated and Autonomic Communications (SAC)

Project Number: FP6-IST-27489

Deliverable D2.6

Design and Evaluation of Self-Association and Self-Organization Mechanisms for Network Compartments

Version 1.0

ANA Project

Autonomic Network Architecture

Project Number	FP6-IST-27489
Project Name	ANA - Autonomic Network Architecture
Document Number	FP6-IST-27489/WP2/D2.6
Document Title	Design and Evaluation of Self-Association and Self-Organization Mechanisms for Network Compartments
Workpackage	WP2
Editor	Marcus Schöller (NEC)
Authors	Ghazi Bouabene (UBasel), Martin Karsten (UWater), Konstantinos Oikonomou (NKUA), Marcus Schöller (NEC)
Reviewers	Christophe Jelger (UBasel)
Dissemination level	Public
Contractual delivery date	31st December 2007
Delivery Date	13th February 2008
Version	1.0

Abstract:

This document presents the draft deliverable on mechanisms for self-association and self-organization of the ANA architecture. It encompasses bootstrapping mechanisms which enable self-association and on top of this new clustering algorithms, an addressing scheme and an intra-compartment routing scheme.

Keywords:

Bootstrapping, Self-Association, Clustering, Addressing, Intra-Compartment Routing, Self-Organization

Change Note:

Executive Summary

Mechanisms for self-association and self-organisation are fundamental to build an autonomous system. This deliverables presents the results achieved in task 2.4 (Self-Association and Self-Organisation) of the ANA project.

Starting a new network node in any environment requires the ability to sense this environment and associate itself accordingly. Therefore, the bootstrapping of the node has to encompass the setup of basic communication means. The attachment of a brick on top of the network device driver is the first step required and builds the basis for the ANA architecture. Higher layer compartments will associate themselves to the node compartment instantiated on any ANA node. Communication between compartments is realised by XRP (eXtended Representation Protocol) which is specified in more detail in deliverable D1.8. Encoded in XRP are the four main functions: publication of a compartment, revocation of a compartment, lookup of a compartment, and resolve of a compartment. This basic system provides the technical foundations to build self-organisation mechanisms within ANA and is central to the ANA architecture.

On top of this basic system three self-organisation mechanisms are designed: clustering, addressing, and intra-compartment routing. Two new, strictly localized algorithms for distributed, directed, budget-based clustering of large-scale networks are proposed: the Directed Budget-Based and the Directed Budget-Based with Random Delays algorithm. The algorithm utilizes local information about unclustered regions and directs the cluster tokens accordingly. An enhancement of this first algorithm was realized by introducing random delays when sending the cluster tokens into new areas. Secondly, we started to investigate a new addressing scheme where addresses are no longer permanent identifiers of a node but rather temporal leases of variable length combined with rendezvous schemes. If each node can have multiple addresses representing different possible paths towards its current location, address aggregation can be maintained at all times. Furthermore renumbering is much easier and a flexible addressing scheme can accommodate special connectivity scenarios more efficiently. Last, an intra-routing scheme for clustered network topologies is presented. Instead of building a shortest path graph only the routing scheme can utilize other metrics, too. A first prototype measures the stability of wireless links and uses this as a second metric to compute the forwarding graph. To keep the routing overhead small the fisheye principle is used. In a distance based routing protocol local information is held more up to date than information about far away nodes. Introducing a new metric opens new alternatives for the information distribution scheme.

Table of Contents

1 Introduction 1
1.1 Structure of this Document 1
2 Algorithms and Mechanisms 2
2.1 Bootstrapping and Node Monitoring 2
2.2 Clustering 9
2.2.1 The Directed Budget-based clustering algorithm 10
2.2.2 The Directed Budget-Based Algorithm with Random Delays 11
2.2.3 Some Concluding Remarks 12
2.3 Address Allocation 12
2.3.1 Related Work 14
2.3.2 Proposal 15
2.3.3 Discussion 16
2.3.4 Open Issues 17
2.4 Intra-Compartment Routing 18
2.4.1 Routing within a G*node 19
2.4.2 Open Issues 21
3 Conclusion 22
4 References 23

1 INTRODUCTION

An increasing complexity of setting up, maintaining, and managing networks burdens a huge overhead on network operators. One goal of autonomic networking is to develop mechanisms to make the network perform these operations autonomously. Booting a node autonomously into a network compartment requires an awareness of the system's context and mechanisms to self-associate it with the network. Next, the node has to configure itself according to the sensed environment. It chooses one or more addresses it will be known by and make this information available to others. This addressing scheme has to be flexible enough that it can adjust itself to simple, multi-homing, or mobile scenarios. Based on the addressing information intra-compartment routing and forwarding has to be established. We have investigated hierarchical routing schemes with self-adapting routing metrics on each level. The routing scheme is supported by the clustering algorithm that allows a fast formation of clusters with low associated costs.

This deliverable presents algorithms and evaluation results for these mechanisms developed for the ANA architecture. Furthermore, we describe how system internals can be presented to the user and how the user can interact with the system by expressing i.e. operational policies.

1.1 Structure of this Document

In section 2.1 the bootstrapping of an ANA node is presented including node monitoring. Clustering of nodes is described in section 2.2, followed by an address allocation scheme in section 2.3. An intra-compartment routing and forwarding scheme is presented in section 2.4. Last, in section 2.4.2 the yellow pages system of the ANA nodes and gateway mechanisms are introduced. A summary concludes this deliverable and provides directions for future work.

2 ALGORITHMS AND MECHANISMS

The first draft of this deliverable presents ideas and first prototypes for self-association and self-organization. First, the bootstrapping of an ANA node is described, followed by clustering mechanism to group nodes. The third section introduces a novel approach to addressing in autonomous networks. Last, an intra-compartment routing mechanism based on the previously introduced clustering and address is introduced.

2.1 Bootstrapping and Node Monitoring

Bootstrapping in ANA includes two domains: the initialization of an ANA node in a new physical environment and also the startup of an autonomous service in a previously bootstrapped node. Bootstrapping of an autonomous functionality (node or service) can be decomposed in two steps: self-association and self-organization. In order of execution, the self-organization part usually comes after the self-association part. In this section we will detail on the self-association part of the bootstrap procedure and treat different self-organization aspects in further sections of this deliverable.

Self-association can also be decomposed in two steps: First the current context of the environment it should associate itself to has to be discovered. Second association protocols (transactions) for the environment (or some interesting parts of it) have to be executed.

In this section, we will describe our proposals for the two mentioned steps of self-association. A main constraint of the proposed solution is its adaptability. Indeed, today's networking environments are very heterogeneous, i.e. composed of many different networks. A main goal of the ANA project is to increase this heterogeneity by allowing for "network innovation". Thereby, the adaptability requirement for self-association mechanisms becomes very prominent. By adaptability, we mean that the same self-association mechanisms applied by the autonomous node or service should be able to work inadvertently of the environments composition.

2.1.1 Self association of an ANA node

A functional block which is able to interact directly with this new physical environment (medium) has to be inserted into the Playground of an ANA node to enable the bootstrapping and building of complex functionalities. The idea is that this special functional block will build an initial (level-0) compartment that will help other services and compartments to bootstrap.

Level-0 Compartment:

The Level-0 compartment is different from other compartments in its ability to bootstrap itself by using directly the physical medium and does not rely on any other compartment to provide it with communication facilities.

This special compartment will then be able to do the necessary bridging between the ANA world, functioning with compartments, IDPs, Information Channels and all sorts of abstractions, and the "real" physical environment of the ANA node. Thereby, it will provide the basic ANA functionality that will facilitate the bootstrap of other functional blocks providing "higher-level" services.

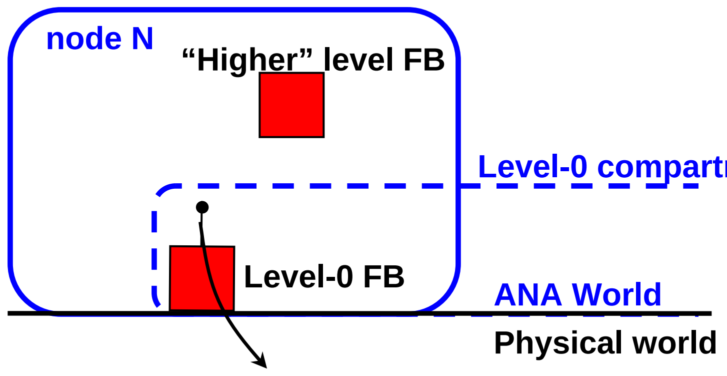

Figure 1: Level-0 functional block

Context discovery for the level-0 compartment:

In an ANA node the Functional block providing the Level-0 compartment is assumed to be sitting on top of the physical medium. How it was able to access the hardware interfaces is purely a low level operating system issue and is outside of the scope of this document. This assumption covers the context discovery part of the bootstrap of the level-0 functional block.

Self-association transactions for the level-0 compartment:

Except to the node compartment (by publishing its presence), the level-0 functional block does not have to self-associate itself to any other compartment. Indeed, since it is supposed to offer the initial self-association facilities to other compartments, it should be "magically" already self-associated. Since this level-0 FB has access to the physical medium, it can discover its neighbour peers (i.e. self organize) by sending messages directly "on the wire" and therefore jump directly to the self-organization part of the bootstrap.

Naturally, since today's networks have different physical media (hardware), chances that the same level-0 functional block is able to work on top of different underlying physical environments are thin. So in order to increase the adaptability of an ANA node to different physical environments, we need to provide it with multiple level-0 functional blocks bridging the gap between the ANA world and several different underlying systems.

2.1.2 Self association of autonomous services:

We assume that autonomous services associate themselves to an ANA node that had previously bootstrapped correctly. This means that a Level-0 compartment provider and maybe some higher level compartments are already instantiated.

Considering that a service will need to self-associate itself, first to the node compartment and then probably to multiple different compartments, we would like to avoid having multiple context discovery and self-association mechanisms that are specific to each compartment. Therefore, we introduce a common communication scheme between functional blocks, as well as a generic API that will help in having the same context discovery and self-association mechanisms which work for all compartments.

Common presentation layer:

In order for a service to be able to interact with all the compartments in a similar manner and self-associate to them, there is a need for a common communication ground between the service and the compartment providing functional blocks. For this reason, in ANA we introduce a common presentation and binding layer shown in Figure 2.

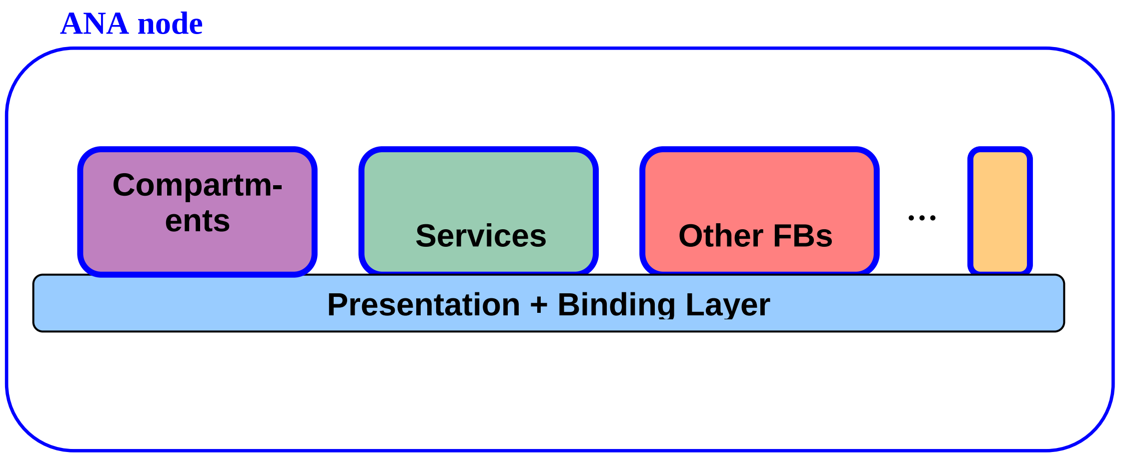

Figure 2: The ANA common presentation and binding layer

This description layer is based on a common formatting of messages that follows the XRP (eXtended Representation Protocol) specification (see deliverable 1.8 documentation). We will not speak here of technical details of XRP, but rather show how it is useful to self-association. Messages exchanged between functional blocks are of the following form (abstract form):

Message Type	**Nb Arguments**
Arg1 Description	Arg1 Value
Arg2 Description	Arg2 Value
…	

The most important fields to notice in the above table are the message type and the arguments descriptions. These provide a very basic meta-data (information) about the message and the data it contains. By using these information fields, a functional block receiving some data that it can not handle can still have a semantic understanding of its nature. For example, an IPv4 compartment provider functional block receiving an IPv6 address as argument in an XRP message will not know how to make use of it but is still able to understand that it is an address.

The idea is then to use this description layer in order to establish a very minimal self-association and context discovery protocol between the services and the compartment providing functional blocks.

Compartment generic API:

Due to the simple meta-data provided by the presentation layer and with addition of some minimal conventions, the interaction between functional blocks is enabled. This eases the self-association process significantly. Our proposal is to extract the most common functionalities needed by all autonomous nodes and services to bootstrap. These functionalities are:

- Publish: that allows a service to make itself visible to other compartment users
- Un-publish: withdraw a service after it was published
- Resolve: to obtain from a compartment an information channel to communicate with a certain compartment user. This functionality is vital for the self-organization part of the bootstrap since there is a necessity to communicate with peer services.
- Lookup: to obtain further information from a compartment about a certain compartment user.

These functionalities are needed by any type of service interested in associating itself to a compartment. We mandate that every compartment entity provides (or at least knows how to handle) these four functionalities as a basic generic API. Therefore a minimal convention at the description layer is needed. This convention consists in assigning specific XRP message types and argument descriptions for the four functionalities above and their arguments. All compartment providers and users have to agree to this minimal convention. All functional blocks which are implemented so far and which use the ANA library implicitly adhere to this convention. The following table shows an example of a publish command sent through the description layer:

Type = publish	2
NAME	"chat application"
IDP	0xAABBCCDDEE

The flexibility of the description layer allows the integration of new interactions between services and compartments, which might be needed for bootstrapping.

Context Discovery:

As a first step of its self-association process, an autonomous service needs first to discover what compartments are present on the local ANA node. To do so, the service can query the node compartment using either the resolve or lookup primitives. Resolve primitive can be used when the service knows exactly the description of the compartment it is looking for and wants to immediately obtain and information channel to it. Lookup, on the other hand, can be used when a service is looking for a relaxed description of a group of compartments and wants to know what different options it has. For example a chat application wants to know which compartments are present on its node, before it decides to associate itself to one or many of them. Therefore, we suggest that all compartment provider functional blocks include keywords in their published description.

For the currently implemented compartments, this common description is the keyword "compartment" published in the node compartment's Key Value Repository. This way, our example chat application has only to query the node compartment with this agreed sub-description to obtain information about all the networking capabilities of its running ANA node. Afterwards, when the autonomous service wants to discover the context inside a specific compartment, e.g. the Ethernet one, it can simply use the same primitives: resolve and lookup in a similar way as with the node compartment.

Self-association transactions:

By using the generic API, the same association mechanisms made by an autonomous service can be applicable to all compartments. These self-association mechanisms are publishing themselves and are obtaining communication channels to peers in order to start the self-organization process. Since all compartment provider functional blocks agree on the message type and argument descriptions of a publish command, they are all able to decode the message and treat it in a compartment specific way. We will see in a later paragraph a scenario showing how the meta-data contained in the publish message, can help an autonomous service attach to many different compartments. After publishing itself an autonomous service can use the resolve and lookup primitives as described before to obtain communication channels to peer services having published their presence in the compartment,

2.1.3 Bootstrap scenario:

To help better understand how these notions of level-0 compartment description layer and generic API are useful to self-association, we illustrate the bootstrapping procedure of the Ethernet and IP compartments already implemented in the current ANA core.

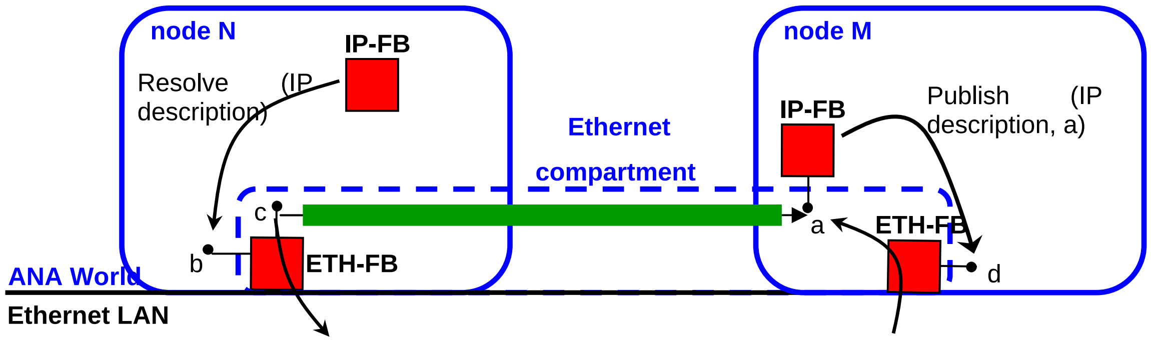

Figure 3: Bootstrap scenario of the IP compartment

In this scenario we suppose that the node M is already bootstrapped. We will describe the chronological steps needed for node N and its Ethernet and IP functional blocks to bootstrap.

Level-0 compartment bootstrap:

1) First the Ethernet functional block in node N obtains access to the Ethernet interface. This step is independent of ANA business.
2) It then associates itself to the node compartment inside node N by publishing its presence. To do so, it sends a publish request using the generic API to the node compartment. The request contains the IDP label “b” and the keywords

description (“Ethernet”, “Compartment”). At this point, the IDP “b” becomes visible in the node compartment context.

Autonomous IP compartment bootstrap:

3) The IP functional block first discovers the networking capabilities of its node. To do this, it sends a lookup request, using the generic API to the node compartment. The query looks for any functional block that published itself with the keyword “compartment”. Since the node has only one compartment available (the Ethernet one), it returns the IDP label “b” to the IP functional block.
4) Now the IP functional block will try to bootstrap itself in the Ethernet compartment. It will first try to discover its neighbour peers and then publish its presence.
    a. Using the generic API, the IP functional block sends a resolve request on IDP “b”. The resolve specifies that it wants to reach all peers matching an IP specific description in the entire Ethernet compartment. After the Ethernet compartment specific resolution has taken place, the IP functional block receives the IDP “c” attached to an information channel leading to the peer IP functional block on node M.
    b. Now, the IP functional block on node N wants to become visible in the Ethernet compartment. To do so, it sends a publish request on IDP “b”. It indicates to the Ethernet compartment a description of the service it wants to publish. It is important to note here that the Ethernet functional block on node N does not have to understand this description. In fact all it cares about is the fact that *it is a service description*. Now, all other Ethernet users that want to reach this IP functional block have to know this description and resolve it as in 4.a). For IP peers, this common description knowledge is an agreement among the IP compartment providers.
5) Now that the IP functional block has an information channel leading to its neighbour peers, it can start all kinds of self-organization procedures, like address autoconfiguration, etc, that we will not talk about in this section.

Note that the previous steps would have been exactly the same if node N had e.g. a Bluetooth functional block as a level-0 compartment due to the generic API and the common description layer.

6) To finalize the bootstrap, the IP functional block finishes its self-association to the node compartment. It publishes its presence as being a functional block providing an IP compartment. The published description could be for example the two keywords (“IP, “compartment”).

2.1.4 **User control over bootstrap procedures**:

For the moment, not much focus has been put on the user control part since all the autonomous bootstrap procedures were still to be defined. At present, a status interface has been developed to inform the user about what is going on in the minmex (published IDPs, present functional blocks).

In the future the user should be able to influence the autonomous behaviour through compartment policies. As an example, if a user wants to forbid a file sharing application from accessing the internet, he could specify in the node compartment, a policy forbidding all functional blocks matching the file sharing application's description from accessing the IP functional block. This way the node compartment will ignore all resolve requests targeted to the IP functional block, coming from file sharing applications. Furthermore, there can also be a policy in the IP compartment (local IP functional block), to drop all communication coming from a functional block matching the file-sharing description. It is noteworthy to mention that not all services will be fully autonomous, and can not decide on their own to which compartments to self-associate to. Indeed, in ANA there is an ongoing work on functional composition, where the goal is to design an "autonomous brain" that would associate the functional blocks in the right order to build the desired information flows. The human user will then be able to tune the bootstrapping procedure also on that side.

2.2 Clustering

One of first self-organization mechanism after bootstrapping a node into a new environment is associating the node with a cluster. A clustered network architecture is typically advantageous to be based on clusters of fixed size (reduced routing protocol overhead, better accommodating specific service requirements, etc). At the same time, it is very important that the self-organization process keeps the associated overhead under control and does not over consumes critical resources such as, e.g., the batteries of the sensor nodes [Krishnan06]. It is thus crucial that self-organization algorithms not only yield clusters of sizes close to the fixed targeted value, but also complete the network self-organization process within a short time, or by executing a low number of steps or requiring low message exchanges.

Another constraint to consider when designing algorithms for self-association and self-organization purposes is due to the fact that each individual node is typically bound to communicating with its immediate neighbors only. Long-haul communications (especially in wireless environments) are not desired due to severe energy constraints or not possible due to non-connectivity (especially for not wireless environments). Consequently, it is important to design algorithms and protocols which are *localized* or *even strictly localized*, as explained in [Estrin99]. A strictly localized protocol is a localized protocol in which all information processed by a node is either: (a) local in nature or (b) global in nature, but obtainable by querying only the node's neighbors or itself [Chan04]. The local interactions are primarily enabled through the exchange of periodic local HELLO messages, which account for building and maintaining the local neighborhood and may also carry any information needed to achieve the desired global objectives.

Two algorithms for message efficient distributed clustering are proposed in [Krishnan06]. Note that they have been proposed mainly for wireless sensor networks but the main principles apply for any self-organization and self-association scenario. The goal is to decompose an autonomous network into clusters of bounded size while keeping the overall message complexity low. These algorithms are supplemented by a randomized technique for specifying the clustering initiation process. The focus of the aforementioned algorithms is to design distributed, energy efficient algorithms for network organization.

Recently there has been a focus on designing localized algorithms for various network functions, such as flooding, broadcasting and spanner construction [Orecchia04], [Tseng03]. These algorithms utilize the existing periodic HELLO message exchanges to maintain and update the local neighborhood of each node (through HELLO exchanges with first-hop neighbors) and to build the desired structure over it.

In the work that has so far been undertaken under ANA, [Tzevelekas06], a new strictly localized clustering protocol is proposed for wireless sensor networks which aims at decomposing large scale networks into non-overlapping clusters of fixed size, an important aspect in autonomic networks like the ANA environment. The primary focus of this work is to provide a fast network decomposition algorithm, which constructs clusters with sizes (in number of nodes) as close as possible to a desired upper bound. The proposed algorithm is considered to be executed in rounds that coincide with the periodic exchanges of HELLO messages and takes advantage of their presence in order to convey (with minimal additional overhead through a flag) some elementary and limited clustering process status information.

The proposed work falls into the category of strictly localized protocols, since all the required information for the protocol to run at each node is obtainable from the immediate, local neighbors of that node. There is no communication between nodes that are more than one hop away at any stage of execution of the proposed clustering protocol.

2.2.1 *The Directed Budget-based clustering algorithm*

The two aforementioned algorithms distribute tokens (i.e., a given number of permits to join a cluster) blindly in the sense that the state of the neighbor (i.e., whether already clustered or not) is not taken into consideration. Consequently, tokens are wasted (or returned) when distributed to nodes which are already clustered by another initiator. This blindness is the reason for their poor clustering performance in terms of cluster sizes or time to network decomposition completion (Persistent). To reduce the inefficiencies due to the blindness of the token distribution process, it is proposed in [Tzevelekas06], that nodes update their neighbors regarding their clustering status as described next. This low overhead status exchange will enhance the effectiveness of the token distribution process, as nodes will utilize this status information in order to direct tokens towards areas of (yet) unclustered neighboring nodes and reduce token waste.

In the proposed Directed Budget-Based (DBB) algorithm, the unclustered neighborhood is identified for each node after each HELLO message exchange and tokens are distributed (equally) over this unclustered neighborhood. Compared with an algorithm distributing tokens over the physical topology (e.g. the Rapid algorithm in [Krishnan06]), the proposed DBB algorithm directs the token distribution process away from clustered regions by operating on the unclustered (as opposed to the physical) topology. The clustered and unclustered topologies typically consist of multiple island-regions scattered throughout the network. Each island-region of clustered nodes may be viewed as defining a boundary that bounces (or reflects) away any incoming (and bound to be wasted or returned) tokens and directs the clustering (token distribution) process towards unclustered regions.

2.2.2 *The Directed Budget-Based Algorithm with Random Delays*

The proposed DBB algorithm saves tokens and is expected to enhance the clustering process by reducing or eliminating inter-cluster token distribution contentions. Tokens are also likely to be wasted due to intra-cluster token distribution contentions arising when two nodes have common (firsthop) neighbors. High node densities result in many nodes having common (first-hop) neighbors in their local neighborhood. Thus, when two such nodes take part in the growing of a cluster under a given initiator (i.e. they both have part of the budget to distribute further), it is likely that they will pick the same common neighbor (or neighbors) to forward part of their tokens to. The intra-cluster token distribution process contentions for Rapid, Persistent and DBB are due to the fact that these algorithms proceed with the budget distribution at each node immediately upon receiving a budget from one of their neighbors.

However, the overall clustering performance of the DBB algorithm would be enhanced if, for example, each distributing node was aware that some of its neighbors were already clustered due to receiving tokens from the local neighborhood and diverted its tokens towards another unclustered node, even if that required that it waited for a later round of HELLO message exchanges to complete its budget distribution. The DBB algorithm with Random Delays can be seen as a modified version of the DBB algorithm aiming to reduce primarily intra- but also inter- cluster token distribution contentions. The basic idea is to delay the token forwarding to a neighbor by a random number of rounds (or HELLO exchanges) to reduce the probability of token distribution contention (notice that this Random Delay is zero for the DBB algorithm). This way, the neighboring, distributing nodes will likely be aware of earlier budget distribution effects on the local neighborhood (which nodes are clustered/not clustered) and thus divert their tokens towards unclustered local neighbors only. The cost associated with the introduction of Random Delays in the DBB algorithm is an additional delay in the overall time until the network decomposition is completed. This additional delay results in “de-synchronizing” the execution of the potentially synchronized token distribution process at neighboring nodes and reduces the likelihood for collisions.

2.2.3 *Some Concluding Remarks*

Two new, strictly localized algorithms for distributed, directed, budget-based clustering of large-scale networks are proposed: the Directed Budget-Based (DBB) and the Directed Budget-Based with Random Delays (DBB-RD) algorithm. The basic, innovative idea is to utilize clustering status information that can be readily available to reduce or eliminate token distribution contentions (both intra- and inter- cluster) that severely limit the effectiveness of earlier budget-based approaches. The algorithms take advantage of the periodic exchanges of standard HELLO messages between neighbor nodes (apparent in real-world networks) to update their physical, as well as unclustered topology of their local neighborhood. They use the updated topology to direct distributing tokens away from already clustered nodes (both intra- and inter- clustered), thus significantly improving the overall clustering performance. Furthermore, the algorithms involve only moderate overhead (a simple 0/1 flag at the end of each HELLO message) to the network.

The proposed algorithms could be useful for organizing ANA nodes in clusters of a targeted size by considering carefully overhead, decomposition time and divergence from the target size, when the ANA nodes exhibit characteristics such as those described in [Tzevelekas06]. In addition to serving self-organization and self-association needs in ANA, such clustering schemes could be utilized also in the context of service discovery, as it may become part of some of the service discovery phases (discussed in detail later), e.g., the service advertising phase that could build a set of ANA nodes that are aware of the existence of some service. It is not mandatory to be used for self-organization or self-association in ANA. Rather they are proposals to be considered as possible future directions when considering the fact that neighborhood discovery may be seen as an alternative form of service discovery. Here, the role of discovery is undertaken by the exchanged HELLO messages and the result is the organization of the network in cluster in an efficient and scalable manner rather than the location of a particular node.

2.3 Address Allocation

Address allocation and routing are intrinsically related. Hierarchical addressing and routing is essential for large-scale networks to keep the size of routing and forwarding tables manageable by aggregating multiple forwarding entries that can be represented by a single entry. The set of aggregated addresses is expressed as their common fixed-size binary prefix. This introduces certain partitioning requirements: a) address blocks must be of size power of 2, and b) if any address within a contiguous block requires a separate forwarding entry, the block has to be split up into multiple smaller parts. An alternative approach to circumvent this problem is to perform a longest-prefix comparison to determine the matching forwarding entry. Longest prefix matching, which is used by the current Internet routing and forwarding mechanisms, trades off computational simplicity for reduced forwarding table size.

Address aggregation forms a tree overlaying the overall connectivity graph. If nodes are permitted to keep their addresses when changing their location in the connectivity graph, then the overlaying tree is typically not a proper subset of the connectivity graph. This can happen, for example, if a customer node or network chooses to change network providers. In the Internet, the logical addressing tree is rooted at a fictitious node with address 0.0.0.0. Address allocation is handled recursively from the top to bottom of the tree. Because addresses are globally unique and have a fixed length, there are obvious fragmentation challenges. Historically, addresses have been allocated and configured manually and statically and have taken the role of quasi-stable node identifiers for servers in the Internet. Therefore, dynamic address allocation has been limited to the very edge of the network using DHCP. Overall, this approach significantly limits the room for autonomous (i.e. local) decisions about the size and structure of subnetworks.

While this aspect can be somewhat mitigated through larger address spaces, such as suggested for IPv6, this “solution” does not address the essential fundamental design constraints associated with inflexible address allocation. Also, a larger address space poses its own new performance problem. For the current 32-bit IPv4 addresses, fast lookup technology for high-speed routers can barely keep up with current link speeds. Performing longest-prefix match lookups for a larger address space will aggravate this challenge considerably.

When designing a new addressing and address allocation scheme, additional important design goals need to be taken into account to ensure that a proposal is truly future-proof. First, communication paradigms such as multi-homing, mobility and multicast break address aggregation, since in these cases a single stable address cannot have rendezvous semantics and topological relevance at the same time. This is the classical ID/locator split problem. Another important goal is path diversity. Currently, network providers typically run single-path routing protocols and end users have no control over the end-to-end traffic path. To improve communication quality, robustness, and economic competition, it has been argued, e.g. in [Yang07], that more control should be given to end users.

Our proposed solution is based on a hierarchy of labels of which only the top level needs to be globally coordinated. A variable-size list of fixed-size local labels is used as strictly topology-oriented address. This can be regarded as a simple yet comprehensive solution for the ID/locator problem. The foremost goal is to be feasible for a future clean-slate approach to internetworking, regardless of other network protocol functionality. One of the key requirements is the potential for forwarding performance at line speeds, i.e. implementation in low-level system components. We design a simple addressing scheme that delegates as much control as possible to local agreement and requires only minimal global administrative coordination. It is combined with an agile rendezvous service operating between DNS and the addressing layer. Both schemes take into account the economic and/or political realities of an Internet formed by a diverse set of networks.

The addressing scheme can unify many different forwarding paradigms, such as destination routing, source routing, virtual circuit, or combinations of these. In cooperation with the rendezvous service, the system can support different communication paradigms, such as multicast, mobility, etc. The introduction of a separate rendezvous service enables the decoupling and reorganization of functionality that is currently performed in the IP forwarding plane (e.g. policy-based routing or traffic engineering) or the DNS naming plane (e.g. load-balancing). This leaves the addressing and forwarding plane lean, flexible and essentially strategy-free, which should result in enhanced clarity and hopefully performance.

2.3.1 *Related Work*

There are many ongoing standardization activities [HIP] and existing technology standards, such as GRE, MPLS, Mobile IP, etc., that address subsets of this problem domain. Further, the IRTF routing working group [RRG] has produced a number of interesting proposals to alleviate global routing concerns related to address aggregation.

Many of the recent proposals for DHT-based lookup services can also be considered as related work. One particular example is the Internet Indirection Infrastructure (I3) [Stoica04]. It contains many of the mechanisms proposed in our scheme and covers some of the same basic ground in the overall problem domain, such as supporting multicast and mobility. However, it also differs in a number of key properties. First, it is a hybrid rendezvous and forwarding mechanism, whereas we propose two separate mechanisms. Further, it is intrinsically tied to the notion of flat DHT-like data structures. For example, it requires large randomly chosen labels to minimize identifier collisions in the absence of centralized administration. Ultimately, it is thus inherently limited to being a scenario-specific overlay network. In contrast, our work is tuned for low-level widespread usage under realistic assumptions. These assumptions include the support for control and delegation in a hierarchical rendezvous structure.

Similar proposals for addressing and routing have been given in the NIMROD [Castineyra96] and NIRA [Yang07] architectures. Both of them provide mechanisms for client networks to discover the internetwork topology, thus enabling them to compute routes for packet delivery. In particular, NIMROD uses hierarchical addresses to represent networks and network elements. It provides a map-distribution mechanism to inform individual network nodes about the topology of particular regions in the hierarchy. However, the proposal does not explain how NIMROD can efficiently fit into the current policy-based routing paradigm. In addition, NIMROD effectively mandates source routing (potentially combined with virtual circuit forwarding) and does not support destination-based routing. NIRA, on the other hand, is quite similar to our approach in terms of basic design aspects, and it is designed for the current customer-provider based routing policy. In NIRA, each network is assigned a number of addresses. Each address denotes a path from a top-level provider to the network. A domain level route can be determined by concatenating a source and a destination address. However, as a source routing scheme, NIRA requires the source network to specify the entire inter-domain routing paths, which leaves transit networks with no control over their traffic. Also, NIRA is somewhat incomplete in that route discovery is limited to paths along a strict customer/provider hierarchy with only limited support for peering routes.

In contrast, our approach attempts to achieve more than previous work by designing an autonomous address allocation scheme that increases flexibility and allows shared control over routing choices. At the same time, we attempt to design a more general proposal that also includes solutions for the general ID/locator split problem.

2.3.2 *Proposal*

The actual Internet topology is a meshed graph with some hierarchy (often characterized as a power-law hierarchical graph). In contrast, hierarchical IP addressing establishes a strict tree relationship between networks and nodes, originating at a virtual root node with IP address 0.0.0.0. All paths in the network that do not follow the single addressing tree cause a distortion of the address aggregation scheme. There are several ways in which a different addressing and rendezvous scheme can alleviate the situation:

- If each node can have multiple addresses representing different possible paths towards its current location, address aggregation can be maintained at all times.
- By allowing variable-length addresses, the overall network does not have to settle for an either too small (IPv4) or too big (IPv6) address space. Instead, it is often possible for core nodes to forward packets based on only a small address prefix, while any number of nodes in an arbitrary topology can still be supported. Address space fragmentation can be contained to those areas in the global network that effectively cause the fragmentation.
- Eliminating the inherent assumption that addresses are permanent identifiers makes renumbering easier.
- Supporting relative addresses as well as absolute addresses increases flexibility to accommodate special connectivity scenarios efficiently.

The addressing scheme described here satisfies all the above goals. In addition, it unifies a variety of addressing approaches for different forwarding paradigms and thereby helps eliminating or at least clarifying cumbersome middle-box functionality.

An address is a stack of fixed-size integer labels. Top-level transit networks are assigned a unique label each by a centralized authority, such as IANA. We consider the set of these networks as the nucleus of the global Internet and denote it as *core*. The overall Internet is formed by other networks directly or indirectly interconnecting with this core. All other addressing is localized and potentially uses dynamic leases rather than permanent address allocation - ultimately a local decision. Within a network, each host and subnetwork is assigned a local label. Host and subnetwork labels are drawn from the same local label space in each network. Typically, a customer network needs to be an assigned a local label by a provider network that it connects to, but not vice versa. When networks interconnect, they can also assign each other local labels.

A stack of labels leading from a core network to another network or end system forms an absolute address for this entity and also directly describes the path from the top-level network to the network or end system. Multiple paths result in multiple addresses being available for an entity. On the other hand, the addressing scheme also supports using a label stack as a relative address beginning at any particular network, if each network along the path has assigned a local label to the next network.

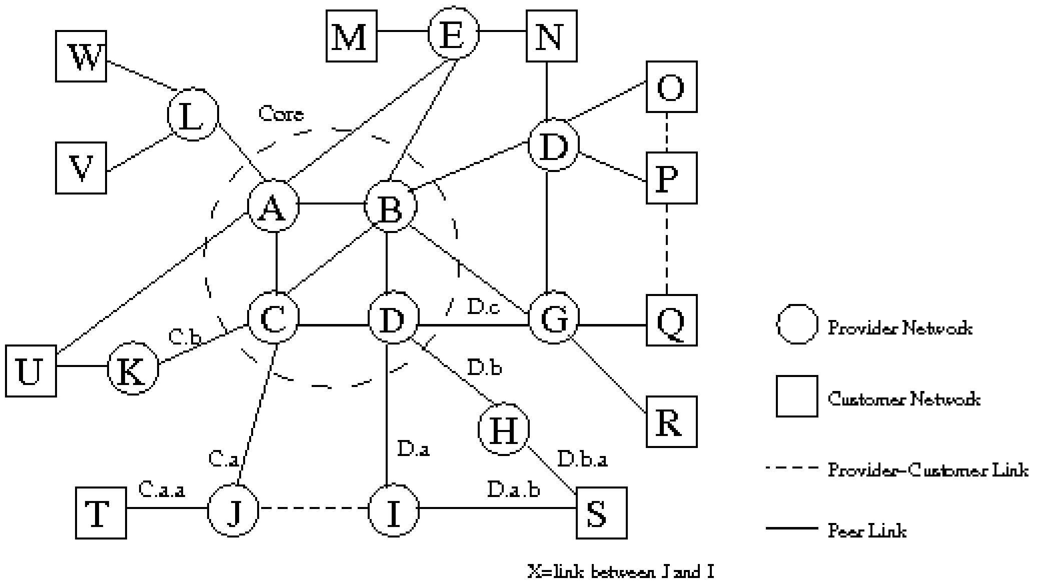

Figure 4: Addressing scheme example

To illustrate the addressing scheme, consider Figure 4. Networks A, B, C, and D form the core. Other ISP and customers networks are connected hierarchically to the core and have peering links with each other. The core networks are shown with capital letters. Other networks are also shown with a globally unique capital letter for easier referencing. Local address labels are denoted by lower-case letters. For example, Network S receives two absolute addresses D.b.a and D.a.b representing the two paths leading from the core to network. The addressing scheme and corresponding routing mechanisms operate at the inter-domain level exclusively. By choosing appropriate source and destination addresses, end systems can thus control the path inter-domain through the network.

2.3.3 *Discussion*

Some considerations are still preliminary, as we further develop this proposal. However, several promising observations can be made about the addressing system. The simplest and probably most common case is given by communication paths that lead from an end system to the core following customer-provider links and then back "out" to another end system following provider-customer links. The term "valley-free" route has been coined for this scenario [Yang07]. In this case, a multi-homed sender can choose the outbound part of the end-to-end path by selecting from its outgoing links. A multi-homed receiver can choose the inbound part of the path for future incoming traffic by using the appropriate source address in outgoing messages. Each end system is requested to use the latest received source address as destination address for future responses. Thereby, a node can control which path future incoming messages will take.

Assuming that individual address labels are rather small (our current working assumption is 16 bits), it is feasible to eliminate the separate addressing schemes to identify transport protocols (protocol number) and application instances (port number). Instead, all addressing needs can be integrated into a single network-layer addressing scheme. This scheme can easily be augmented by giving forwarders the ability to rewrite source addresses and keep forwarding state indexed by the rewritten label stack. This essentially reproduces NAT functionality, but now it is integrated into the architecture as a first-class citizen. Even further, this functionality can be used to efficiently support mobility, similar to existing proposals for Mobile NAT [Buddhikot05]. As mentioned before, address labels are only leased to the networks and nodes. Since nodes can have multiple addresses at the same time and a correspondent node can be informed about the impeding address change by using the new address as source address, a make-before-break hand-over mechanism can be supported without requiring any further specialized control protocol. In fact, an end system does not even have to be aware of its absolute address, if it only acts a transport client, i.e. if it is always the initiator of a transport association. A message can be sent without a source address at all and the absolute source address is gradually built by intermediate systems. All these schemes assume symmetric paths at the respective routing granularity at which they operate.

The collection of mechanisms proposed here permit the construction of internetworking communication that seamlessly supports arbitrary combinations of datagram routing and virtual path service. Traditional virtual circuit protocols suffer from path setup latency and overhead associated with per-flow state maintenance and cleanup complexity. Since our proposal does not require a separate signaling protocol, it is possible to piggyback control onto data messages and avoid a separate path setup latency, similar to XTP [Strayer92]. The establishment of non-default paths and/or addresses does not require per-flow state, but per-flow state can be uheysed where appropriate. Short request/reply interactions will not incur any path setup overhead at all.

2.3.4 *Open Issues*

The simple and most common case assumption is that a core networks form a very dense mesh and only valley-free paths are being used. In reality, the second assumption will not hold often enough, so that alternative scenarios have to be studied. The goal is to provide mechanisms for flexible path discovery different policies along paths that include peering links. Various techniques are possible to discover such paths and mediate between the potentially diverging interests of various network operators and customer networks. We will study the trade-offs in this problem domain and in particular, will consider the business framework and economic incentives of participants.

Although the addressing scheme operates at the inter-domain level, resilience and recovery need to be studied. Some preliminary results are published as part of the NIRA study [Yang07]. We also note that resilience is studied in WP3.3 of the ANA project and hope to benefit from those findings.

As this proposal is still in its early stages, a number of smaller issues can be expected to arise once we arrive at a more specific specification and prototype implementation of this scheme.

2.4 Intra-Compartment Routing

Intra-compartment routing has to be applied to small networks like a WLAN cloud or a sensor network but also to very large networks like an IP compartment. For the later clustering as shown in chapter 2.2 can be used to build manageable topologies. The work on intra-compartment routing is inspired by the Netsukuku routing and clustering work proposed in [NETS]. Netsukuku uses clusters to build a distributed system up to the scale of the current Internet. But in contrast to the Internet they do not use a fixed address space but can assign arbitrary address sizes to each level in the cluster hierarchy. An example Netsukuku hierarchy is shown in Figure 5.

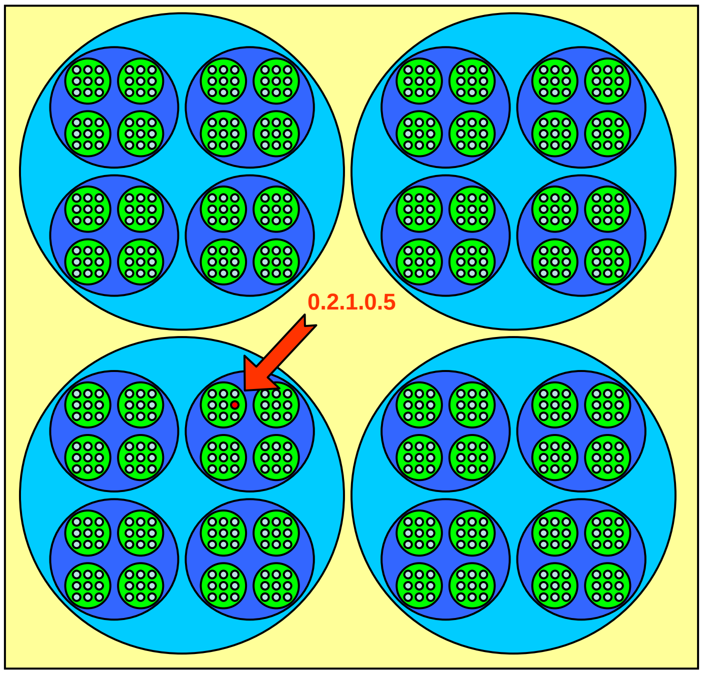

Figure 5: Hierarchical clustering of Netsukuku

The figure shows a Netsukuku hierarchy which is structured in 5 levels as it would like if all possible addresses are taken by a node. The hierarchy is constructed by aggregating a set of nodes – nine in the example – and grouping them into a so called Gnode. These Gnodes themselves act as normal nodes and can be aggregated to a GGnode and so on. The address of a node is composed by concatenating the addresses of all G*nodes in the hierarchy the node lies within. An example node address is: (0.)2.1.0.5. On the top level of the hierarchy only one G*node exists which we give the address 0. Therefore, this 0 is usually omitted. On the forth level 4 GGGnodes exist. The node is part of GGGnode 2. On the third level the node is grouped into GGnode 1 and on the first level in Gnode 0. Within this Gnode the node has the address 5.

A routing protocol distributes topology information to all nodes within a G*node. Every node has to store the routing information for all nodes in all G*nodes it inherits through the hierarchy. Assuming an 8 bit address space for each level of the hierarchy in the above example leads to 256*5 = 1280 entries in the forwarding table for 8^5 = 32767 nodes which are addressable in the network.

To forward a packet through the network a node first has to identify the level in the hierarch where the higher level G*node addresses of the node itself matches the destination address. On this level the routing protocol provides a path from the node to the G*node with the same address as the destination on this level. The forwarding process on this level returns the next hop G*node and a border node – which of course is a G*node of the next lower level in the hierarchy – which acts as a gateway to this G*node. If the border node is not in the same Gnode as the sender the forwarding process recursively calculates the border nodes at each level of the hierarchy until the address of the next hop within the Gnode is returned.

2.4.1 *Routing within a G*node*

The authors of Netsukuku propose their own routing protocol called QSPN2 [QSPN] but Netsukuku itself is independent of the active routing protocol. They use QSPN2 as the routing protocol an each level in the hierarchy. In our simulation we used a modified version of FSR (fisheye state routing) [Iwata99].

Organising a network in a hierarchy as Netsukuku does makes the network easily scale to a large size but has some requirements to the routing protocol. First, the routing protocol depends on the characteristics of the G*node and the level in the Netsukuku hierarchy. Second, the routing protocol of an autonomous system must be adaptable to a variety of metrics.

G*nodes are expected to vary significantly depending on the characteristics of the physical nodes and the level in the hierarchy. A G*node consisting of wireless nodes only has different requirements towards the routing algorithm compared to mixed wired/wireless G*node or an all fixed nodes G*node. The independence of the routing protocol allows two ways to explore: First, the use of different routing protocols in different G*nodes. That would require that during the G*node creation the nodes would have to negotiate which routing protocol to use and that mobile nodes must be capable of executing a range of different routing protocols. The later could be solved by on-demand downloading of the routing protocol program provided that trust issues and heterogeneity of end systems can be solved. Second, the same routing protocol is used in all G*nodes and can identify the G*node's characteristics on its own and adapt itself accordingly. This requires the negotiation and publication of the used metrics within the G*nodes.

Furthermore, the routing protocol must be able to adapt to different metrics. The metrics used to construct the forwarding graph depend on the application requirements. Currently most routing protocols use shortest path algorithms to compute the shortest paths to all possible destinations using hop count as the metric. But other application might regard path stability higher than end to end delay. Within Task 3.3 "Resilience Framework" a set of quality of service metrics for resilience are defined. Other quality of service metrics encompasses performance and security. Again the used metrics within different G*nodes may vary according to the different nodes in the G*nodes and their missions. The routing protocol must be open enough to use the required metric or a combination of metrics to create the forwarding graph.

We used the fisheye principle introduced in FSR as the foundation of our experiments. Routing updates are sent to neighbouring nodes periodically containing any new information the node has obtained since the last update. By defining update scopes and update intervals some information about the system's topology is distributed more often to neighbouring nodes than other information. Using the fisheye analogy FSR distributes topology information about close by nodes three times as often as topology information about nodes farer away.

As a first metric we used link stability in a wireless network environment. First results show the space which can be explored. Since routing updates are sent periodically to all neighbours – broadcast with TTL 1 – monitoring how high the loss rate of routing message is easy. Routing updates are sent periodically every five seconds. Not receiving an update message is recorded as a packet loss and the link stability is decreased. Three consecutive missed update messages indicate that the node is no longer a direct neighbour. We currently investigate which other metrics can be used to calculate the path stability to other nodes and how they can be measured.

To accomplish the exchange of the measured parameters the message format of FSR had to be extended in two ways: First, the locally monitored downlink stability of all neighbouring nodes is included in each update message. This information is included in a variable length list between the message header and the topology information. The length of this list depends on the number of neighbours the host has discovered. For now the complete list is published in each routing update message which introduces an overhead to every message. In order to reduce this overhead we think about restricting the list to entries which parameters have changed by a certain threshold. Further investigations have to show how to balance the savings I overhead and the inaccuracy in the metric computation. Second, the uplink stability to all neighbours is added to each topology entry. To calculate the forwarding graph the node can use this stability information as an additional input. Besides building shortest paths to all known destination – the distance in hop count to other nodes is inherently included in the topology information – a node can calculate the most stable path to all destinations – which might be undesirably long – or the shortest path to all destination with a minimum stability (if such a path exists).

Secondly, we investigated the scope definition. Introducing a second metric into the routing algorithm for the shortest path calculation enables more choices in the definition of the routing update scope. Instead of sending changed topology information about nodes with a routing distance below a defined limit more often than the information for nodes farer away we used the path stability as the metric. Information about nodes to which a path exists which exceeds a defined stability limit is sent more often to neighbouring nodes than the information about nodes below this limit.

2.4.2 *Open Issues*

The next steps include the design of a uniform interface to enable the configuration of the routing protocol. The node must be able to set which metrics to use for the shortest path calculation and which relation to use to compare two (sets of) values. Furthermore, scope definitions and update intervals must be provided by the node.

An other open issue is to explore ways to gather needed information depending on the used metric. To improve the link stability metric we used in our first experiments we envision using statistics gathered while forwarding regular traffic. Depending on the technology of the nodes and challenges experienced in the network link layer statistic can be used for this. Furthermore, we envision using statistics gathered on the transport level. The quality of an end-to-end path can improve the metric computation for the path to the destination.

Finally, a prototype for the ANA framework will be implemented. It will be a functional block that runs in the user space part of the MINMEX only as today’s routing protocols are running in user space. The MINMEX provides an easy and universal interface to update the forwarding table of the appropriate in-path functional block which can run in user or – more likely – in kernel space.

3 CONCLUSION

In this deliverable we present fundamental building blocks for autonomic networking systems. The bootstrapping procedure provides the basis for the ANA architecture. The node compartment is created where following compartments can publish themselves, un-publish themselves, lookup other compartments and resolve other compartments. These functionalities enable self-association and self-organization mechanisms to be build. Examples of such self-organization mechanisms have been presented next. Our research on directed budget based clustering has been published in [Tzevelekas06]. It significantly improves the clustering process by utilizing local information which neighbouring nodes are not yet part of the cluster. Secondly, a design for a new addressing scheme has been introduced. It is based on variable length addresses which are temporarily assigned to the node in contrast to today's fixed length permanent addresses. Last, an intra-compartment routing scheme has been presented. It is designed to calculate the forwarding table based on multiple metrics. A first prototype using the link stability as an additional metric to the distance metric showed to be promising.

In the upcoming months research in task 2.4 will concentrate on three objectives: First, design and development of generic self-association and self-organization algorithms to allow the FBs of a compartment to "bootstrap" communications. Secondly, continue ongoing work on the proposed "simple addressing scheme" that aims to provide scalable and efficient support for multi-homing and mobility. And thirdly, explore the design space for the routing scheme in all its dimensions. For all these activities prototypes will be implemented which will operate on the ANA MINMEX system. Therefore, the next deliverable will be software package and an accompanying document.

4 REFERENCES

[Buddhikot05] M. Buddhikot, A. Hari, K. Singh, and S. Miller: "MobileNAT: A New Technique for Mobility Across Heterogeneous Address Spaces"; Mobile Networks and Applications, 10(3): 289-302 2005.

[Castineyra96] I. Castineyra, N. Chiappa, M. Steenstrup: "The Nimrod Routing Architecture"; RFC1992, August 1996

[Chan04] H. Chan and A. Perrig: "Ace: An emergent algorithm for highly uniform cluster formation"; in Proc. of IEEE European Workshop on Wireless Sensor Networks (EWSN'04). Springer Verlag, January 2004, pp. 154-171.

[Estrin99] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar: "Next century challenges: Scalable coordination in sensor networks"; in Proc. of ACM MobiComm'99, August 1999, pp. 263-270.

[HIP] IETF Working Group: "Host Identity Protocol (hip)"; http://www.ietf.org/html.charters/hip-charter.html

[Iwata99] A. Iwata, C. Chiang, G. Pei, Mario G. Chen: "Scalable Routing Strategies for Ad hoc Wireless Networks"; IEEE Journal on Selected Areas in Communications, Volume 17, No, 8, pp.1369-1379, IEEE, August 1999.

[NETS] The Netsukuku group; "Netskuku topology"; http://netsukuku.freaknet.org/; Last visited 01/08.

[Krishnan06] R. Krishnan and D. Starobinski: "Efficient clustering algorithms for selforganizing wireless sensor networks"; Ad Hoc Networks, vol. 4, no. 1, pp. 36&-59, January 2006.

[QSPN] The Netsukuku group: "Quantum Shortest Path Netsukuku"; http://netsukuku.freaknet.org/; Last visited 01/08.

[RRG] IRTY Routing Research Group (rrg); http://www.irtf.org/charter?gtype=rg&group=rrg

[Orecchia04] L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti: "Localized techniques for broadcasting in wireless sensor networks"; in Proc. of ACM DIALM-POMC'04, Philadelphia, USA, October 2004.

[Stoica04] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana: "Internet Indirection Infrastructure"; IEEE/ACM Transactions on Networking, 12(2):205-218, April 2004.

[Strayer92] W. T. Strayer, B. Dempsey, and A. Weaver: “XTP: The Xpress Transfer Protocol”; Addison Wesley, Reading, MA, 1992.

[Tseng03] Y. C. Tseng, S. Y. Ni, and E. Y. Shih: “Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network”; IEEE Trans. Comput., vol. 52, no. 5, May 2003.

[Tzevelekas06] L. Tzevelekas and I. Stavrakakis: “Directed Budget-Based Clustering for Wireless Sensor Networks”, 2nd International Workshop on Localized Communication and Topology Protocols for Ad-hoc Networks (LOCAN2006) held in conjunction with IEEE MASS 2006, October 9-12, 2006, Vancouver, Canada.

[Yang07] X. Yang, David Clark, and A. Berger: “NIRA: A New Inter-Domain Routing Architecture”; IEEE/ACM Transactions on Networking, 15(4):775-788, Dec. 2007.