ANA Project

Autonomic Network Architecture

Sixth Framework Programme
Priority FP6-2004-IST-4
Situated and Autonomic Communications (SAC)

Project Number: FP6-IST-27489

Deliverable D.4.4

Specification of the IP to ANA migration layer

– Possible designs –

ANA Project

Autonomic Network Architecture

Project Number	FP6-IST-27489
Project Name	ANA - Autonomic Network Architecture
Document Number	FP6-IST-27489/WP4/D.4.4
Document Title	Specification of the IP to ANA migration layer
Workpackage	WP4
Editor	Christophe Jelger (UBasel)
Authors	Christophe Jelger (UBasel)
Reviewers	Rudolf Roth (Fokus)
Dissemination level	Public
Contractual delivery date	31^{st} December 2007
Delivery date	15^{th} February 2008
Version	Version 1.0

Abstract:

This document presents potential design options for the migration layer that will allow legacy IP-based applications to be used with ANA. We describe and discuss the various alternatives currently being considered and also consider implementation issues. The objective is to finalize the design of the migration layer and implement a first prototype before July 2008.

Keywords:

IP to ANA migration layer.

Foreword

In the description of work of the project for the period M13-M30, the title of this deliverable was indicated as being "Specification of the IP to ANA migration layer". However, the current document presents some possible designs for the IP to ANA migration layer, which we now actually prefer to call "IP to ANA *adaptation* layer". That is, this document does not yet provide the specifications of this adaptation layer.

This change is mainly due to delays in the decision process for choosing the best alternative for the IP to ANA adaptation layer. However, at the time of writing this deliverable, it seems (but has not yet been officialy decided) that the prefered solution for developing the IP to ANA adaptation layer is to use a virtual DNS domain as described in section 2.1. Our current plan is to release the specifications and the implementation of this solution before July 2008.

Table of content

1 Introduction

As initially stated in the initial description of work of the project, the design of ANA follows a "clean-slate" approach. That is from the start, we did not want to restrict the design of ANA by imposing any backwards compatibility constraints. However, that does not mean we do not want to support the very rich palette of existing applications that were designed for the current "IP by default" networking world. This hence requires that ANA supports a *migration layer* that permits to use *legacy* applications "at the edges of ANA". Here the term "edges" refers to software and has no topological meaning in terms of network deployment: that is, we refer to the frontier between ANA software and legacy "IP-based" software. The adjective *legacy* refers to standard applications and networking software that have been developed to operate "by default" over the Internet (e.g. web browsers, email clients and servers, video-conferencing applications, but also network protocols such as ARP, DNS, IP, etc).

It is important to note that we currently restrict the scope of the IP to ANA migration layer to the following scenario illustrated by figure 1. In this scenario, two legacy applications running on separate ANA nodes communicate via the ANA "network" via dedicated ANA software emulating the operation of standard IP protocols. Our main motivation is to be able to re-use reference end-user applications (e.g. a web browser on one node and a web server of the other node), session protocols (e.g. HTTP), and data formats (e.g. HTML), while data is actually being transported over ANA. This allows developers to concentrate on autonomic communications and mechanisms inside ANA since existing applications can be re-used to demonstrate and validate the operation of ANA. Note that more advanced scenarios involving communications across the Internet and ANA via dedicated gateway nodes will be considered at a later stage of the project.

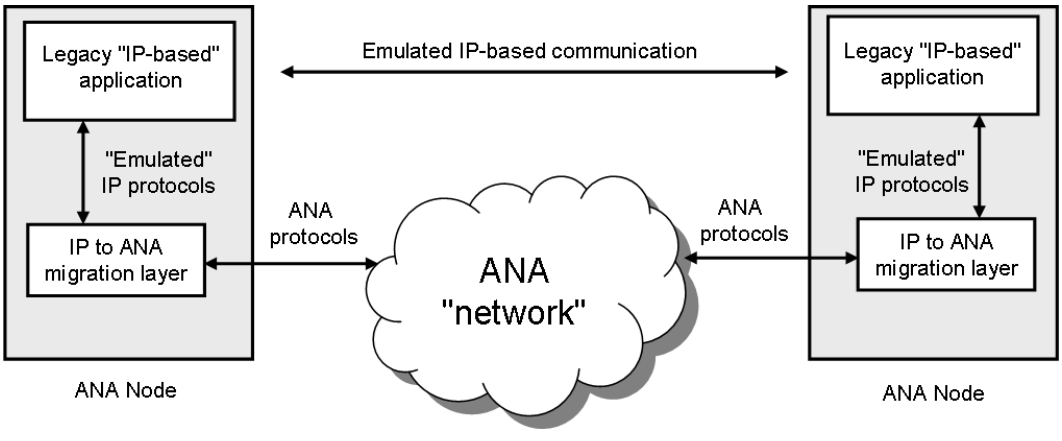

Figure 1: Emulation of "Internet protocols".

Without yet going into details, such a migration layer involves two main activities. First, the packets sent by certain legacy applications must be captured and forwarded towards ANA but, in the mean time, we want to let other applications operate without interfering. This is needed to let legacy Internet network services operate normally while only the network traffic of certain applications is redirected towards ANA. The second and more critical activity of the migration layer is to emulate the current operation of the suite of Internet protocols: that is for example, existing applications typically generate DNS and ARP requests while initiating a communication with a remote peer. For reasons described in the following sections, the ANA migration layer will have to exhibit the network operation of at least the ARP protocol (and in certain cases of also the DNS protocol) to make the applications "believe" that they actually operate in a normal Internet network environment.

Clearly, the first activity of selectively capturing/redirecting packets to ANA is straightforward and there exist multiple alternatives for performing this: we describe three of them in section 2 and discuss their advantages and drawbacks and implementation complexity. The second activity of emulating "Internet protocols" is not very complex but can be really time consuming to fully emulate the operation of protocols which have multiple networking options and behaviors (e.g. DNS). Nevertheless, UBasel has some experience in this and has already implemented DNS and ARP emulation for the LUNAR protocol [1]. This is actually further discussed in section 3.

2 Capturing packets ...

As just briefly described, the first requirement for re-using IP-based applications with ANA is to be able to selectively redirect or capture the packets sent by certain applications while the traffic of other applications is not affected and transits via the Internet. In the following subsections, we present three alternatives for performing such a selective packet redirection and briefly discuss the advantages and drawbacks of each approach.

2.1 ... via a virtual DNS domain

Before describing this solution, it is important to recall the typical networking steps that occur when setting up some communication over the Internet. First, a communication typically starts with some DNS request to resolve a name (e.g. `www.example.com`) into an IP address. To achieve this, each host is configured with the IP address of a DNS server to which it can send DNS lookup requests. However before sending the DNS request, the host first needs to resolve the link-layer MAC (Medium Access Control) address of either the DNS server (if located on the same IP subnet) or the next hop gateway to the DNS server. This resolution is handled by the standard ARP (Address Resolution Protocol) and maps an IP

address into a MAC address. Once the MAC address is obtained, the DNS lookup request can be sent to the DNS server which eventually returns the IP address of the destination host. At that point an additional ARP resolution is required to map this IP address into either the MAC address of the destination host or the next hop gateway that must be used to reach the destination.

Building from our previous experience [1] and based on the typical communication setup we just described, we propose to perform the redirection of selected applications by introducing a virtual DNS domain, e.g. `.ana`. That is, we use domain names to differentiate whether data should be sent via the legacy Internet or via ANA. For example, data sent to `www.example.com` would be sent via the Internet while data sent to `www.example.com.ana` or `somehost.ana` would be sent via ANA. To achieve this, all DNS lookup requests are sent to a fake DNS server which, depending on the name being looked up, either simply forwards the request to some real DNS server or performs the DNS name resolution via ANA and whatever protocol handles such resolution inside ANA. This procedure is illustrated by figure 2.

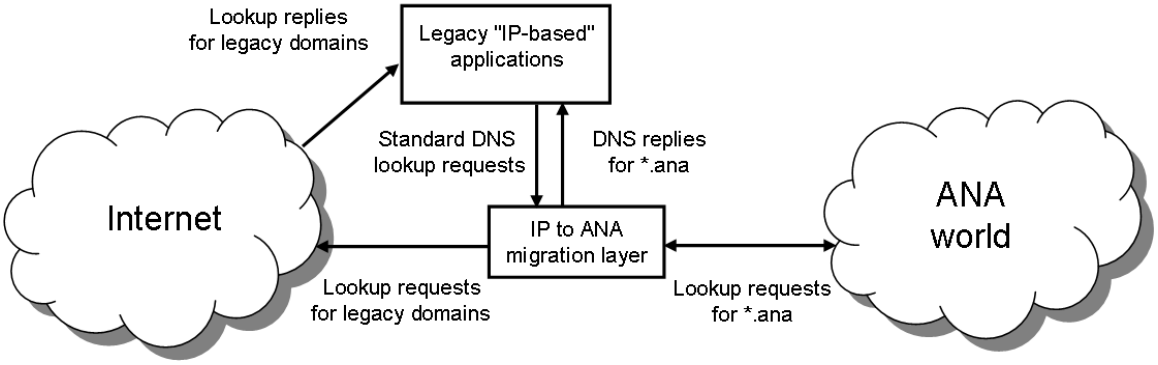

Figure 2: Redirecting traffic via a virtual DNS domain.

The main advantage of this solution is that it is very easy for a user to specify whether some application traffic should be redirected towards ANA. Also if the redirection of DNS requests is performed after the ARP resolution (e.g. via netfilter with a matching rule for port 53), it leaves the standard DNS configuration of hosts untouched and is hence a less disruptive approach. Note that the fake DNS replies for `*.ana` names should return IP addresses from a private IP address range as for example `10.42.0.0/16`.

To then redirect data sent to `10.42.0.0/16` towards ANA, the IP to ANA migration layer can create a virtual interface (e.g. `IPtoANA`) on the OS and configure a static routing table entry such that all packets sent to `10.42.0.0/16` are sent via the virtual interface which forwards packets towards ANA. This forwarding phase is illustrated by figure 3. Another (maybe more flexible) alternative is to use Linux's netfilter to redirect all packets sent to `10.42.0.0/16` to the IP to ANA migration layer.

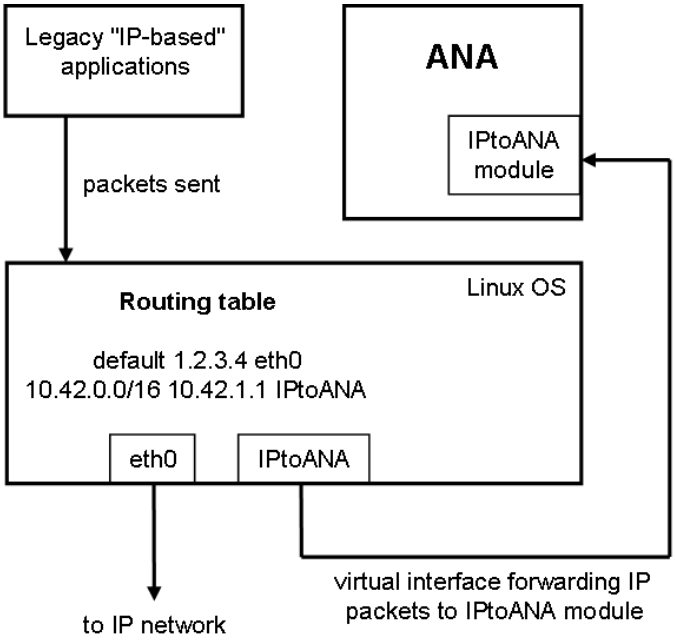

Figure 3: Creating a virtual interface.

The main advantage of this solution is that is requires very little static configuration and provides a very simple and human-friedly technique for redirecting network traffic towards ANA. The main drawback is that there must exist inside ANA some dynamic protocol for resolving DNS names into ("emulated") unique IP addresses (with respect to the ANA "network") so this solution relies on an additional development. Another option would be to implement "private address maps" [1] where each node maintains its own local mappings between DNS names and IP addresses without requiring any global coordination for using unique addresses. However this technique implies that special NAT software (i.e. application level gateways or ALGs) is used to properly "NATify" packets at destination nodes. While such mature software currently exists, it requires very complex interactions with Linux's netfilter and we could not yet build a proof-of-concept prototype for validating this advanced functionality. At the time of writing this deliverable, it is however not clear whether this alternative should be further investigated or not.

2.2 ... via a new netfilter target

The second alternative for redirecting network traffic towards ANA uses the existing Linux's netfilter framework. It is actually a straightforward technique that here operates after the initial DNS and ARP resolutions (as described earlier) have taken place. This means that we rely on the traditional DNS system to resolve a name (e.g. `somehost.ana.unibas.ch` into a valid global IP address before data can be redirected towards ANA.

For specifying which data is to be forwarded towards ANA, one can use the very rich and powerful options of the netfilter framework to specify *matching rules*.

In brief, each matching rule consists in a boolean combination of rules that check some part of each packet flowing through Linux's network stack. For example, one can specify that all packets sent to some particular IP address and destination port number should be forwarded towards ANA. Note that netfilter permits to specify very advanced matching rules based for example on the state of TCP connections or some specific value of any header of the packets. This alternative for forwarding packets towards ANA is illustrated by figure 4. Note that on the implementation side this requires that a small kernel module (here labeled `IPtoANA.ko`) must be developed as a standard extension of the netfilter framework. It is worth mentioning here that UBasel also has experience in developing extentions to netfilter.

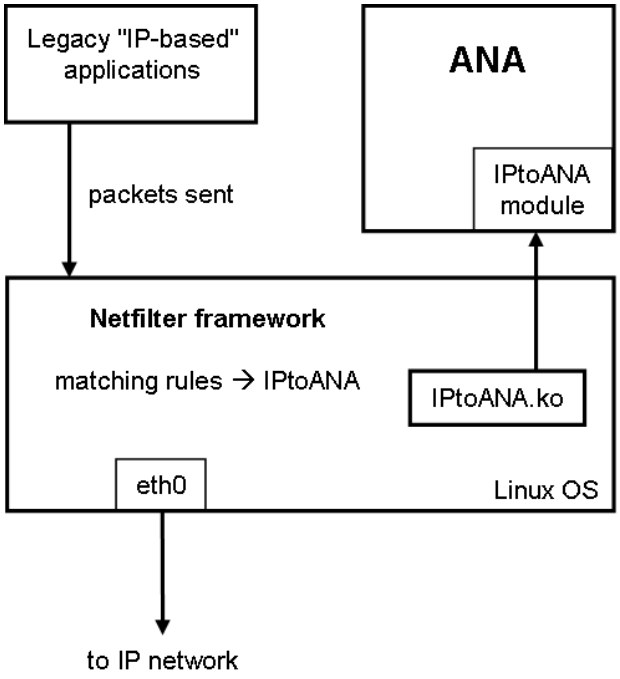

Figure 4: Creating a netfilter target.

The main advantage of this solution is that it does not require that name resolution is performed inside the ANA framework as we re-use the legacy DNS infrastructure. Another advantage is that the netfilter framework is very mature and offers a very rich and powerful palette of matching rules so one can be very selective when specifying which packets are to be forwarded towards ANA. The main disadvantage of this solution is that the matching rules and DNS configuration would be relatively static and would require that the partners of the ANA consortium agree to all use a common private addressing scheme for ANA nodes. In addition, the IT offices/centers of the ANA partners might be relunctant or might simply refuse to include DNS records for research purposes into their DNS configuration.

2.3 ... via an ANA socket library

The third alternative for redirecting the network traffic of certain applications towards ANA requires that these applications are re-compiled (actually re-linked) using an ANA *wrapper library* for the socket API. That is, all function calls of the socket API will be ("transparently") replaced with functions that will actually use the appropriate ANA API primitives to setup communications via ANA. This technique is illustrated by figure 5. It is worth mentioning here that the GNU linker `ld` for Linux has a standard and built-in mechanism (the `-wrap` option) for replacing standard system calls with customized functions.

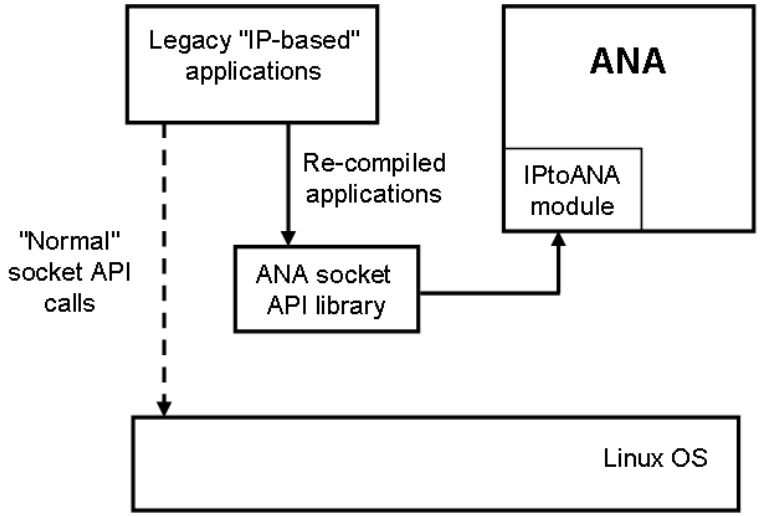

Figure 5: Creating an "ANA socket library".

The main advantage of this solution is that it requires no additional mechanisms for capturing and redirecting packets towards ANA. That is, a user only needs to install an "ANA-ready" application in order to be able to re-use legacy applications over ANA. This solution however requires that target applications are entirely re-compiled and this is not always straightforward with open-source software that often requires the right versions of multiple libraries.

Note that it does not necessarily mean that all data sent by re-compiled applications is redirected towards ANA: the functions of the ANA-socket wrapper library might indeed only redirect certain traffic towards ANA and let the rest of the requests be handled by the real socket API. However, the cost of developing a "good enough" wrapper library for the socket API might be very high if one wants to support most of the options and features that are currently supported by the standard API. Hence this alternative seems to be the most complex in terms of implementation although it might be the solution best fitted for long-term migration of applications towards ANA.

3 Emulating Internet protocols

As described in the previous subsections, the first activity of the IP to ANA migration layer is to redirect certain network traffic towards ANA. In addition to capturing packets, the solution desribed in subsection 2.1 requires the development of a system that can emulate the operation of a DNS server. Moreover, the two solutions described in subsections 2.1 and 2.2 require that the IP to ANA migration layer also emulates the operation of the ARP protocol. Actually UBasel has already developed emulation layers for both the ARP and DNS protocols for the LUNARng protocol [1] and it seems feasible to adapt and extend this existing code for the IP to ANA migration layer.

However we would still need to develop the data structures and mechanisms that will be used to map IP packets into appropriate "ANA packets" and vice-versa. In particular, it is essential to study whether a *statefull* or *stateless* approach is prefered. A stateless solution would imply that each ANA packet carries sufficient information for being correctly demultiplexed to the right legacy application at the destination node (e.g. the encapsulation of IP packets into ANA packets might be enough), while a statefull approach would imply that destination nodes store the information required to perform the appropriate demultiplexing operation (e.g. similar to what is being done in today's NAT routers).

Actually a more challenging task would be to write an emulation layer for TCP. Indeed, TCP connections are setup via a 3-way handshake (exchange of packets) and also require a relatively complex exchange of information for controlling the rate at which packets are being transmitted. Actually this is exactly what NAT (Network Address Translation) routers perform for all TCP connections passing through them. However based on our previous experience, it is quite complex to re-use the features of the netfilter framework to achieve this emulation in the context of the IP to ANA migration layer.

In fact, it is possible to avoid doing any emulation of the TCP protocol. That is, the payload of the IP packet + the TCP header can be encapsulated and sent to the destination (via ANA) and we would let the legacy TCP/IP stacks in the two end-nodes handle the TCP handshake and the exchange of control information. This solution has the advantage of not breaking the fundamental end-to-end principle of TCP and greatly reduces the complexity of designing the migration layer.

4 Future work

At the time of writing this deliverable, it seems that we will use the technique described in subsection 2.1 for redirecting IP packets towards ANA as it is very flexible and permits users to easily specify whether a communication should be handled by ANA or not. This however requires that ANA supports on one hand, a protocol for resolving DNS names and, on the other hand, a specific routing

protocol for forwarding data to the appropriate destination. This requires further collaboration with activitities going on in Work Package 2 on intra- and inter-domain routing and name resolution.

As an early proof of concept, a first prototype implementation of the IP to ANA migration layer will be developed during the first half of 2008 in order to be able to quickly re-use legacy IP-based applications over ANA. While a number of technical challenges still need to be resolved, this milestone seems realistic as we can re-use a lot of the existing expertise gained by UBasel with the development of LUNARng [1].

References

[1] C. Jelger and C. Tschudin. Dynamic Names and Private Address Maps: Complete Self-Configuration for MANETs. In *2nd ACM Conference on Future Networking Technologies (CoNEXT'06)*, December 2006. Lisboa, Portugal.