IP45: Architecture, Design and Implementation

Tomáš Podermański
Brno University of Technology
Center of Computer and Information Services
Antonínská 1, 601 90 Brno, Czech republic
Email: tpoder@cis.vutbr.cz

Miroslav Švéda
Brno University of Technology
Faculty of Information Technology
Božetěchova 2, 612 66 Brno, Czech republic
Email: sveda@fit.vutbr.cz

***Abstract*—This dissertation outline discusses an experimental network architecture and protocol design. The aim of the work is to prove that a network architecture which follows some key principles can bring interesting build-in features. As a proof of concept, a model of architecture was designed into protocol specification which extends existing protocols. One of the interesting feature is that such protocol can regain End-to-End addressing in the NAT based architecture without laborious transition to a new protocol.**

Keywords*—*Internet Protocol; End-to-End Communication

I. Introduction

Soon, it will be more than twenty years since the IPv6 protocol was proposed and standardised as a long term solution for the next generation Internet. According to original plan [1] the whole process of a transition to IPv6 should have been finished before the depletion of the IPv4 address pool. With the benefit of hindsight, we know that the transition process was more difficult than anybody could have anticipated and currently, even the most optimistic predictions expect that the transition will not take less than a decade. In the meantime, the common solution at least to mitigate the problem with diminishing address space is *addres translation* (NAT).

From the other perpective, insufficient address space is not the only reason why NATs are used. One of the key advantages provided by NAT is the *address independency* and *reusing single IP address* by many devices. In practice, it does not matter how many NATs are connected in the chain and the IP network topology can be arranged into hierarchical and recursive structure. Thanks to that we can easily implement features, e.g. small site multihoming without running a BGP router [2], or to change the upstream ISP without readdressing all devices. Another frequent reason for using NAT is the consequence of *reusing a single IP address*. Firstly, it hides the internal structure of the network and secondly, it forbids a connection with a device in the NATed network from the outside. Both of them are considered as security features. According to our observation the popularity of NAT is not only the result of shortage of IPv4 address space, but it also provides many benefits and reveals more about the nature of network addressing.

The key motivation for this work is to prove that *address space independency* and *hierarchical addressing schema* together with other principles can bring desired features into network design which can make the protocol design much simpler and decrease the cost of operating networks.

II. Identified Challenges

There are some criteria that can be considered while discussing network architectures. In our work we focused on the following main first-principles and features:

Sattzer's End-to-End argument represents one of the basic principles when a new network architecture is designed [3]. According to our observation, following the principle determines whether a network architecture will be succesfull or not.

Thanks to **End-to-End addressing** every device in the network is able to send data to any other node connected to the network directly. In current networks this principle is broken by NAT and the transition to IPv6 should have solved the problem.

Address space hierarchisation allows to keep a number of routing entries, especially at the top in the hierarchy of the global network, at the necessary minimum. The good example of such architecture is IPv6 in which every sub-network (in IP world known as Autonomous System) can be determined by a single routing entry. In practice, the benefit of hierarchised address space in IPv6 is impaired by *Provider Independent* addresses which are used for multihoming or to avoid demanding readdressing within the site.

Address space independency allows the network to re-connect a part of subtree to another node without any need for readdressing within a child subtree. Address space independency can solve some crucial problems of hierarchical address space mentioned in the previous paragraph. Protocol IPv6 is a nice example which shows that the absence of *address space independency* implies some solutions where one problem can lead to another. In IPv6 every host (or interface) needs to know a *network prefix* to which is connected. It can be set either manually or advertised by a router. For the distribution of prefixes across routers, IPv6 must have an additional mechanism called *Prefix Delegation* [4]. If network prefix is changed, new prefix information has to be distributed to every IPv6 host. As the readdressing of hosts is in many cases very complicated, IPv6 offers some solutions to mitigate the problem. Firstly, every IPv6 interface can configure an additional, IPv6 address[1]. But the existence of multiple addresses brings another complication. To choose a proper IPv6 address as the source address in the outgoing packets, every IPv6 host must keep the *Priority Table*. Another, relatively controversial, solution is known as *IPv6-to-IPv6 Network Prefix Translation*

[1]It could be Link-Local, ULA or another global address.

[5]. It allows to map the global IPv6 prefix assigned from the ISP to the *Unique Local IPv6 Unicast Address* used inside of the site. But it brings many disadvantages and limitations to IPv6 architecture caused by NAT. In IPv4 the absence of address space independency is not as noticeable as in IPv6. The problem is mitigated by NATs which add address space independency into IPv4 address schema. On the contrary, architectures such as RINA [6] or IPNL [7] use address space independency as a key feature of protocol design.

Session independency together with address space independency is an important prerequisite to multihoming and mobility. Thanks to session independency the address of end nodes can be changed during running session. Today, neither IPv4 nor IPv6 support session independency and the feature must be solved on top of those protocols.

Scalability is a feature which allows to expand the nework without creating some kind of bottleneck. In current network architectures there are many potential bottlonecks which could restrict future network growth. For example, current global routing table grows every day and all routers must keep all routes connected to the global routing system. Another example with poor scalability is NAT, where a NAT device has to keep the state for every running session in memory.

Implementation and operational complexity are not directly related to the architectural principles, however, they represent a very important factor considering implementation and operational cost. It is obvious that a more complex design makes implementation, testing debugging more difficult and more expensive.

Adoption difficulty is a factor similar to the previous one. The protocol design must take into account existing protocols, technologies and application interfaces. The transition process of IPv6 points out how important it is.

It may seem that every principle or feature previously mentioned has been already solved in some way, but adding them together into one architecture is a real challenge. For example, an architecture which supports address space hierarchy (e.g. LNAT [8] or IPNL [7]), usually contradicts the design requirements of multihoming and mobility which again disagree with scalability and Saltzers's end-to-end principle. Similar situation is with NATs. NAT perfectly implements address space hierarchisation and independency and can be adopted very well, but it does not support End-to-End addressing and has poor scalability. Concerning mobility, both IPv4 and IPv6 designs do not support session independency and the mobility solution is built on top of protocols such as MIPv6, MPTCP, PMIPv6 or SHIM6 [9]). As a result, none of those solutions is practically used and mobility in IPv4 and IPv6 is therefore rather a theoretical concept than a feature used on daily basis.

The goal of the work is to achieve very good knowledge of existing solutions from both theoretical background and practical applicability. The key to find a solution is to understand what makes some network architectures sucessfull and why some of them are doomed to failure even though there were huge expecations from them (e.g. ISO 7498, ATM).

The work also proposes an experimental network architecture and specifies the protocol design called IP45. As a proof of concept the protocol is implemented to mostly used platforms and is used to perform experiments in real networks. Based on experimental results the original design is continuously adjusted and improved.

III. The Actual Model of IP45 Networks

After several experimental concepts we have come to the following model which seems to satisfy the entry requirements.

The IP45 network model is represented by the collection of *Administrative Domains (AD)* which are organised into a tree structure. Every AD represents a node of the tree and the depth is defined as a *Level*. At the top of the hierarchy there is the *root AD* called *Administrative Domain Level 0 (AD 0)*. An *AD* can be connected to the parent or child *AD* through *Border Gateway (BGW)*. *BGW* configures an *Upstream Address (UA)* and a *Downstream Prefix (DP)*. The *UA* is an address of the address space which belongs to the parent *AD*. The *DP* defines relevant address space for devices connected within *AD*. When a packet is supposed to be delivered to or through the parent *AD*, *BGW* performs the operation, as shown in equation 1. While a packet with destination (*D*) and the source (*S*) address passes *BGW*, the *BGW* extends (*E:*) a relevant part of the source address (*S*) with configured *UA* (α). In the opposite direction, there is a symmetric operation. When a packet is delivered to or through child *AD* the destination address ($D\beta$) is reduced (*R:*) by *DP* (β) to get the destination address (*D*) within the child *AD*.

$$\frac{S}{D} \xrightarrow{E:\alpha} \frac{S}{D\alpha} \quad , \quad \frac{S\beta}{D} \xrightarrow{R:\beta} \frac{S}{D} \tag{1}$$

When we used these two simple operations within the tree structure of *ADs*, we come across two basic equations. The first one (2) describes a situation when a packet from the host *S* placed within AD α_n goes to the root AD (AD 0), and the second one (3) refers to a situation when the packet is delivered to its destination - the host D placed in the AD β_n. To make it simple, we can say that the source address of a packet extends when it passes through *BGWs* to the root AD, and the destination shortens when it goes to the designated AD. Figure 1 illustrates such a situation.

When the host *D* gets a packet, the source and destination addresses are exchanged (4) and packets with a response can be sent back to the host *S* the same way.

$$\frac{S\alpha_n\alpha_{n-1}\ldots\alpha_0}{D} \rightarrow \frac{D}{S\alpha_n\alpha_{n-1}\ldots\alpha_0} \tag{4}$$

The architecture, however, does not strictly relay on the incremental sequence of the level of *ADs*. Any *AD* can be omitted and the extension or reduction operation is avoided. Formally said, the *DP* (α) and *UA* (β) has an empty value. Practically, it means that *AD 4* can be directly connected to *AD 0*. There is also a possibility to make some shortcuts in the tree structure. Such a situation is illustrated in figure 1 with a dotted line. Two *ADs*, which are not a part of the same subtree, are connected via direct link. In that case the *BGW* performs several extension and reduction operations at once and delivers data directly to the neighbouring *AD*.

$$\frac{S}{D\beta_n\beta_{n-1}\dots\beta_0} \xrightarrow{E:\alpha_n} \frac{S\alpha_n}{D\beta_n\beta_{n-1}\dots\beta_0} \xrightarrow{E:\alpha_{n-1}} \frac{S\alpha_n\alpha_{n-1}}{D\beta_n\beta_{n-1}\dots\beta_0} \cdots \xrightarrow{E:\alpha_0} \frac{S\alpha_n\alpha_{n-1}\dots\alpha_0}{D\beta_n\beta_{n-1}\dots\beta_0} \tag{2}$$

$$\frac{S\alpha_n\alpha_{n-1}\dots\alpha_0}{D\beta_n\beta_{n-1}\dots\beta_0} \xrightarrow{R:\beta_0} \cdots \frac{S\alpha_n\alpha_{n-1}\dots\alpha_0}{D\beta_n\beta_{n-1}} \xrightarrow{R:\beta_{n-1}} \frac{S\alpha_n\alpha_{n-1}\dots\alpha_0}{D\beta_n} \xrightarrow{R:\beta_n} \frac{S\alpha_n\alpha_{n-1}\dots\alpha_0}{D} \tag{3}$$

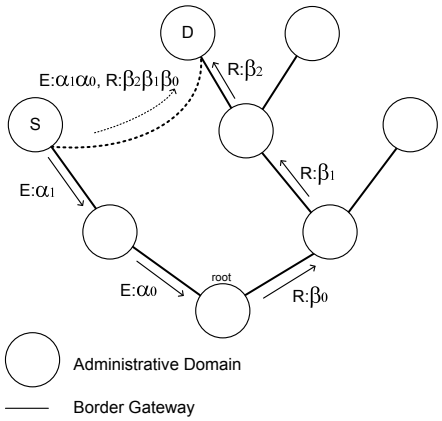

Fig. 1: The Model of IP45 Network

The communication between applications in the IP45 network expects the use of the traditional client server model. The application can use connection-oriented or connectionless abstract layers. When a client needs to connect to the server, it uses the server's address (to identify the interface of the host) and the port (to identify the application). In the traditional TCP/IP approach, every connection *C* is defined as a tuple *C(CA, CP, SA, SP)* where *CA* is a client IP address, *CP* stands for TCP port and *SA* and *SP* represents server address and server port. The connectionless communication is organized in a similar way. In the IP45 model of the network, the connection is defined as a triplet *C(SID, CP, SP)*, where *CP* and *SP* have the same meaning as before and *SID* refers to the *Session Identifier* of the connection. The *SID* is generated as a unique number when a client sends the request to establish connection or when the first packet is sent in connectionless protocols.

IV. Current Status and Future Work

Actually, we made an intesive investigation around past and potential future network architectures and we used the knowledge to build our own one. After many experiments and failures it seems that we reached a solution that matches the best initial requirements. Based on the model of the architecture we created the protocol specification [10] that covers all necessary parts: packets specification, actions performed on the hosts and borders of ADs. The protocol specification puts the emphasis on the compatibility with existing protocols and applications.

To demonstrate the feasibility of the architecture and design, we implemented all necessary components for commonly used platforms (Windows, Linux, OS X, OpenWrt) and we started using the protocol on daily basis together with ordinary applications (mail, web, ssh, rdp). Our implementation also proved that the end host support can be easily implemented even on "closed" platforms without any need for modifications of existing applications. All source codes and binaries are publicly available on the project web site [11] so anybody can test it immediately. The web site also contains video recording demonstrating how extended addressing and mobility features can work with existing applications like ssh and vnc.

In the near future, there are several areas we would like to focus on. Firstly, extend the spread of implementation to as many platforms as possible. In order to obtain broader experience in a heterogeneous environment, we would like to extend the support to mobile platforms and make packages available on standard application delivery platforms such as the *App Store*, *Google Play* and *Windows Phone Store*. We believe that wider experience will help us to uncover problematic parts of the protocols that have not been revealed and enable us to adjust it. We would also like to pursue a profound research of possibilities provided by session independency. This area is full of potential to unearth interesting and inspirational capabilities in area of multihoming and mobility which have not been explored enough.

We believe that our work and obtained results will contribute to the whole community and future research related to the network architectures and protocols.

References

[1] J. Curran, "An Internet Transition Plan." RFC 5211 (Informational), July 2008.

[2] I. Pepelnjak, D. Markovič, and D. Spasojevič, "Small Site Multihoming." Online http://stack.nil.com/ipcorner/SmallSiteMultiHoming. Accessed Jun 20, 2014.

[3] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, pp. 277–288, Nov. 1984.

[4] O. Troan and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6." RFC 3633 (Proposed Standard), Dec. 2003. Updated by RFC 6603.

[5] M. Wasserman and F. Baker, "IPv6-to-IPv6 Network Prefix Translation." RFC 6296 (Experimental), June 2011.

[6] J. Day, *Patterns in Network Architecture - A Return to Fundamentals.* Prentice Hall, 2008. ISBN 978-0-13-225242-3.

[7] P. Francis and R. Gummadi, "IPNL: A NAT-extended internet architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 69–80, Aug. 2001.

[8] V. Jacobson, "LNAT — Large scale IP via Network Address Translation." Working Draft. Online ftp://ftp.ee.lbl.gov/.van/nat.pdf, 1992. Accessed May 02, 2014.

[9] B. M. Sousa, K. Pentikousis, and M. Curado, "Multihoming Management for Future Networks," *Mob. Netw. Appl.*, vol. 16, pp. 505–517, Aug. 2011.

[10] "IP45 - Protocol Specification." Online http://ip45.org/documentation/spec/. Accessed Jun 20, 2014.

[11] "ip45.org - Project Webpage." Online http://www.ip45.org/. Accessed May 02, 2014.