NAN Tools: An Open-Source Tool Suite for Interoperable Neutral Access Networks

Roberto Del Bianco Andrea Seraghiti Alessandro Bogliolo
Information Science and Technology Division of DiSBeF
University of Urbino
Urbino, Italy 61029
Email: alessandro.bogliolo@uniurb.it

***Abstract*—Although IP traffic has been growing exponentially for years and it is expected to keep following the same rate in the near future, current business models do not allow operators to benefit from traffic growth and they fail in motivating investments in next generation networks. In the absence of sufficient investments, operators are induced to avoid congestion by means of traffic management strategies which raise neutrality issues. In spite of the heated debate on net neutrality, all the players involved in the Internet value chain agree in saying that new models are required to overcome broadband stagnation and support innovation. Neutral access networks (NANs) have been recently proposed as means for enabling the development of the Internet without threatening network neutrality. Although the NAN model has been implemented and tested for several years in the *Urbino Wireless Campus* test-bed, its actual applicability requires the availability of a tool suite flexible enough to adapt to different situations and to enable inter-operation among networks managed by different entities. This paper presents the *NAN tools*, an open-source tool suite which enables the employment of scalable and inter-operable NANs on top of Linux. The NAN tools have been successfully ported on embedded systems based on OpenWrt and tested on low-cost MikroTik RouterBOARD.**

***Keywords*-Neutral access network; Tunneling; Policy routing; Scalability; Interoperability**

I. INTRODUCTION

It is a fact that the vertically integrated business models adopted by network operators, combined with flat-fee access rates, have become inadequate to sustain the development of network infrastructures. Although such a model has played a fundamental role in the diffusion of the Internet, it has created a short-circuit between end-users and *over-the-top* (OTT) service providers which has determined an exponential growth of IP traffic with limited benefits for network operators [1]. The misalignment between costs and revenues, together with the imbalance of capitalization in the Internet market [2], is bringing access networks to congestion, inducing operators to adopt counter-measures which threaten neutrality either by applying traffic management policies, or by signing discriminatory agreements with OTT operators. The public consultations on network neutrality launched worldwide by the regulation authorities provide evidence of the awareness of the urgency of a transformation [3]. It has been recently shown that a service-oriented network model, as opposed to an access-based model, could help overcome the stagnation without threatening network neutrality [4]. *Neutral access networks* (NANs), originally proposed as a mean for bridging digital divide in market-failure regions [5], provide a viable support to the implementation of the changes required to sustain the development of the Internet.

This paper presents the *NAN tools*, an open-source tool suite which enables the implementation of a NAN on top of a Linux-based OS, according to the model described by Seraghiti et al. in 2009 [6]. In particular, the paper is focused on scalability and interoperability issues and solutions. Section II provides an overview of the NAN model and architecture, pointing out scalability and interoperability issues, Section III outlines the solutions provided by the 1.1 release of the NAN tools, Section IV describes the embedded version of the NAN tools and reports the results of preliminary experiments conducted on a low-cost MikroTik RB133 running OpenWrt, Section V concludes the work.

II. NAN MODEL AND ARCHITECTURE

A NAN is an open access network which provides a shared access infrastructure with three main features: *i)* it allows end-users to associate with the network even if they are not registered with an operator; *ii)* it allows service providers to expose their service to unauthenticated users directly within the access infrastructure; *iii)* it allows end-users to dynamically select the gateway (i.e., the Internet service provider) to be used to connect to the Internet. Although the NAN model is access-technology agnostic, its most natural embodiment is provided by open Wi-Fi networks exposing a common SSID, because of the ease of association of any kind of mobile and portable devices.

Figure 1 shows the reference architecture of a NAN, as proposed by Seraghiti et al. [6]. The network is open to any unauthenticated user, who associates with an it access island and is assigned (either statically or dynamically) with a unique IP address. In general, IP addresses can be taken either from public or from private pools. For the sake of explanation, private IPs are used in Figure 1 to annotate network nodes.

 ISBN: 978-1-61208-205-9

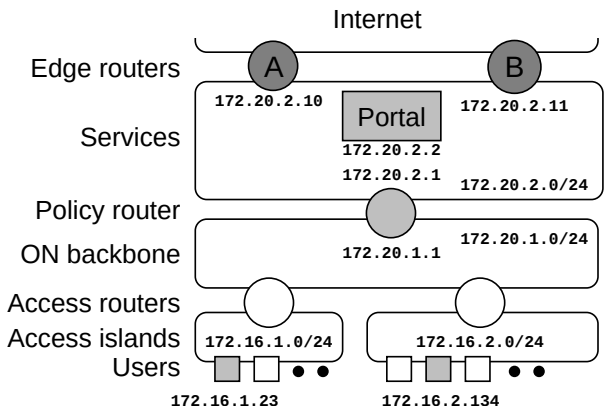

Figure 1. Reference architecture of a NAN.

Technological diversity and evolution are guaranteed by the coexistence of multiple access islands possibly managed by different operators. All of them are connected to the *operator-neutral* (ON) backbone by means of access routers. In the simplest scenario, access routers are network termination points, which can be Wi-Fi hot spots, Hiperlan/WiMax base stations, or digital subscriber line access multiplexers (DSLAMs). The ON backbone is the networked infrastructure which provides connectivity between access routers and services.

The main architectural element is the router placed between the backbone and the service network, called *policy router*. The name denotes its ability to implement advanced routing policies possibly based not only on destination IP addresses, but also on other parameters, such as the source IP address used for the so-called *source-based policy routing*. For the sake of simplicity here we assume that the ON backbone is organized as a unique sub network and that there is only one policy router, which is the default gateway of the ON backbone. Generalization and scalability will be discussed in Section II-A.

Online services available within the access network are published in a service subnet, which includes a captive portal (denoted by Portal in Figure 1) used to redirect end-users to a predefined landing page as they attempt to access an external URL which is not included in any white-list. The landing page belongs to the internal web portal which grants direct access to all internal services and allows end-users to make a choice among many different edge routers (managed by different SPs/ISPs) to gain access to the external services and to the Internet.

As the user makes his/her choice, a source-based policy rule is dynamically created on the policy router in order to forward across the selected edge router all the external packets originated from the IP address of the user.

A. Scalability and interoperability

There are three main issues which limit the scalability of the NAN architecture depicted in Figure 1: *i*) the ON backbone is implemented as a single broadcast domain, *ii*) all the services are published within the same subnetwork, and *iii*) the policy router is a bottleneck for the traffic generated by the end users. Figure 2 shows a generalized architecture which has been proposed to address such issues by allowing network segmentation, load balancing, and interoperation [6].

Network segmentation can be used both to split the backbone into several broadcasting domains and to organize services and edge routers in several subnetworks. Subnetworks containing edge routers need to be connected to the backbone by means of policy routers, while standard routers can be used to grant access to those subnetworks which publish only internal services (this is the case of service Sy in Figure 2).

Load balancing and path optimization strategies can be implemented in the ON backbone by using more than one policy router. Each policy router can be either statically defined as default gateway for specific access islands, or dynamically chosen by the routing protocol, such as *boarder gateway protocol* (BGP) and *open shortest path first* (OSPF). In Figure 2, two policy routers (namely, 1 and 2) can be used to gain access to service Sx and to edge routers A and B_L in the left-hand side NAN.

Interoperability among different NANs can be achieved by creating a tunnel between their backbones and by allowing end-users to traverse the tunnel as any other edge router. Figure 2 shows two inter-operating NANs, denoted by **L** (i.e., left) and **R** (i.e., right), the ON backbones of which are connected by means of a tunnel.

III. NAN Tools

The NAN tools exploit the support for advanced routing provided by the Linux kernel [7] in order to implement the key elements of the NAN architecture of Figure 2, namely, the policy router and the portal.

In the simple example of Figure 2 we have two NANs (**L** and **R**), three policy routers (1, 2, and 3), two portals (Portal$_L$ and Portal$_R$), 4 edge routers (A, B$_L$, B$_R$, and C), and a tunnel. Each policy router contains specific routing tables for all the edge routers which can be selected by the end-users. All the tables are statically created and listed in the `rt_tables` file with the corresponding priorities. Whenever an end-user makes his/her choice, his/her source IP is dynamically added to the corresponding table in the policy router in order to set the preferred edge router as default gateway for that user by means of *source-based policy routing* [7]. It is worth noticing that tunnels leading to other NANs are treated as edge routers at this level. For instance, policy routers 1 and 2 in NAN **L** contain three

 ISBN: 978-1-61208-205-9

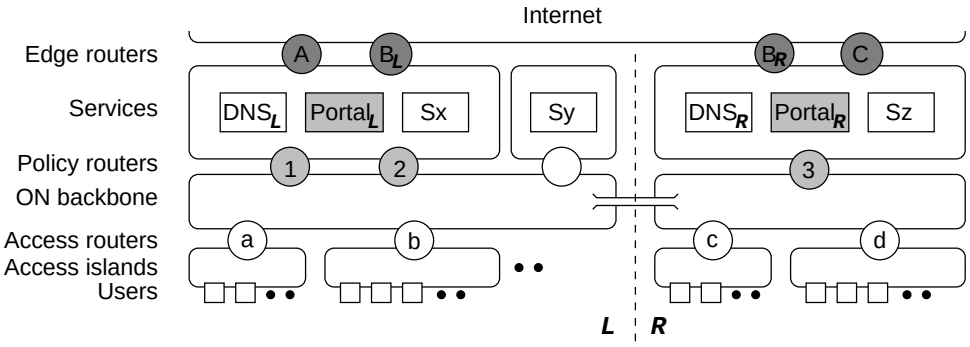

Figure 2. Generalized architecture of two inter-operating NANs.

routing tables for A, for B_L, and for the tunnel leading to NAN **R**.

The NAN tools prevent the IP address of a given user to appear in two or more tables simultaneously. Synchronization between the web-based front-end, which allows the end-user to make his/her choice, and the back-end, which implements the corresponding rule, is based on a data base: the front-end adds a pending rule in the DB upon user's request, while the back-end periodically checks the DB, applies all pending rules, and changes their status to active.

When the edge router of choice is a tunnel towards a different NAN (say, NAN **R** in our example), the end-user is redirected to the captive portal of the target NAN (i.e., $Portal_R$) where he/she can eventually make a further choice among the edge routers available in that network (i.e., B_R and C).

This simple mechanism, which has been implemented in the release 1.0 of the NAN tools and applied on the *Urbino Wireless Campus* test-bed since 2009 [8], [6], assumes that each policy router has its own DB and works independently of all other policy routers. These assumptions, however, limit the flexibility of the architecture depicted in Figure 2 and make it difficult to maintain consistency in case of multiple inter-operating NANs. The solutions adopted in release 1.1 of the NAN tools are outlined in the rest of this section.

A. Dynamic load balancing and redundancy

If more than one policy router is made available on a NAN, the trivial solution for load balancing consists on statically assigning different access islands to each of them. In this case, each access island has one of the policy routers specified as default gateway and it works exactly as if it was the only policy router in the NAN. In particular, each policy router contains only the rules of the end-users assigned with it. For instance, in our example routers 1 and 2 could be used by access islands a and b, respectively.

This situation, however, is incompatible with the optimizations possibly performed at run time by dynamic routing protocols which route packets towards the least loaded nodes. In fact, the traffic generated from access island b cannot be routed across policy router 1 if this latter doesn't contain the rules set by end-users originally assigned to policy router 2.

To provide support to dynamic load balancing, policy routers 1 and 2 should be allowed to share the same DB of policy routes. This can be done by connecting the back-end of both policy routers to the same DB, and by replacing the status flag associated with each rule with a status table which specifies the status associated with each `(rule, router)` pair. In this way, the back-end daemons of the policy routers are allowed to independently access the DB and change the status flags of their rules without interfering with each other.

The same mechanisms can be exploited to enhance reliability by allowing each user to specify a secondary router to be used in case of failure of the default gateway without losing the settings.

B. Loop avoidance

According to the basic inter-operation mechanism described so far, any user of a NAN can reach one of the edge routers available on another NAN provided that a path exists between his/her own default gateway and the policy router leading to the target edge router. Consider, for instance, a user associated with access island `a` who wants to reach edge router C. First, he/she has to set a rule on policy routers 1 and 2 in order to forward his/her traffic towards the tunnel leading to router 3. Then he/she has to set a new rule on 3 in order to be forwarded to edge router C. The captive portal of NAN **R**, however, allows end-users not only to choose between edge routers B_R and C, but also to be forwarded to NAN **L** across the tunnel (which is always treated as an additional edge router). If, by any chance, the end-user coming from **L** decides to make this last choice, a loop occurs which causes the packets originated by that end-user to keep bouncing between the policy routers of the two NANs until the *time to live* expires and the packets are dropped.

In order to avoid this situation, each policy router should prevent end-users from choosing the network they come

 ISBN: 978-1-61208-205-9

from. This can be done by making policy routers aware of the pools of source IP addresses belonging to each NAN. In the NAN tools, this mapping is stored in a specific table of the DB the records of which associate the identifier of each NAN with one or more IP ranges. It is worth noticing that *network address translation* (NAT) can be performed at the termination points of the tunnel established between two NANs, by changing the addresses of the IP packets traveling across the tunnel, in order to relax the constraints on the address ranges used in each NAN.

C. Go back mechanism

The loop avoidance solution described in the previous subsection has an annoying side effect: it doesn't provide to end-users any intuitive mechanism to go back to their original networks. To this purpose, in fact, they should explicitly go to the landing page of their original captive portal (the reachability of which could be guaranteed by top-priority routing tables local to the original policy routers) and select a different edge router to avoid their traffic to be redirected towards the tunnel.

Instead of pretending end-users to remind the addresses of the captive portals of their own networks, the landing page of the captive portal of the host NAN (**R** in our example) should contain a `GO BACK` button directly linking to their original NAN (**L**). Since the actual link associated with the `GO BACK` button depends on the NAN from which the user comes, such a link is stored in the same table introduced in previous subsection for loop avoidance reasons, which contains the IP ranges associated with each neighboring NAN.

When the user presses the `GO BACK` button, he/she is redirected to the landing page of the captive portal of his/her NAN, the rule is removed from the DB, and a new edge router can be chosen.

It is worth mentioning that multiple hops can be made by the same user across inter-operating NANs before reaching the target edge router leading to a specific service or to the Internet. In case of multiple source-based policy rules, the `GO BACK` mechanism allows the end-user to backtrack step by step.

D. Persistence

As explained so far, source-based policy rules are dynamically applied upon user's request. Each rule, however, can be specified either as volatile or as persistent by setting a flag in the DB. Persistent rules are restored upon reboot of the policy router, while volatile rules are reset. In the NAN tools this choice is made by the network administrator, since it has to be consistent with network configurations. In particular, since policy rules are based on source IP addresses, it would be inconvenient to have persistent rules in a network which dynamically assigns IP addresses to users' terminals. For the same reason, a maximum idle time can be specified which allows the policy router to remove the rules of idle users.

IV. Embedded version

The tool suite described so far makes use of three main components: the support for advanced networking provided by Linux, a data base management system (dbms), and a http server with PHP support. In general, the policy router provides only networking functionalities, while it relies on external servers for dbms and httpd functionalities. Nevertheless, in a simple scenario of a NAN with a few access islands serving a limited number of users (say, less than 100 simultaneous users) it would be more practical and less expensive (both in terms of investments and in terms of operating costs) to have the entire tool suite running on an all-in-one appliance. Examples of such a scenario include common situations like small companies, hotels, and coffee shops.

The embedded version of the NAN tools has been developed to provide a low-cost all-in-one solution to these needs, targeting *MIPS* architectures running *OpenWrt* (a Linux distribution for embedded systems providing packet management and a writable file system [9]).

OpenWrt provides all the packets required to solve the dependences of NAN tools 1.1 without any porting effort: namely, Apache web server, MySQL dbms, PHP, openvpn, and the advanced networking functionalities of the Linux kernel (i.e., *iptables* and *iproute2*). In order to minimize hardware requirements, however, the embedded distribution of the NAN tools makes use of *SQLite* [10] in place of MySQL and *uHTTPd* [11] in place of Apache.

While the PHP code written for Apache is fully compatible with uHTTPd, switching from MySQL to SQLite required a partial redesign of some of the classes due both to the serverless nature of SQLite and to the different syntax it understands. The object-oriented paradigm adopted in the development of the NAN tools enabled a suitable encapsulation of the changes required. As for the firmware, an image of OpenWrt was created containing all the packages required to run the NAN tools. Once the image file is installed on the target device, the *unified configuration interface* (UCI) can be used for configuring the policy router before installing the NAN tools. An example of system configuration is provided in Figure 3.

A. Performance

Conservative performance tests of the embedded version of the NAN tools were run on a MikroTik RouterBoard RB133, featuring a 175MHz MIPS CPU with 32MB of DRAM, 3 ethernet interfaces, 3 MiniPCI slots, 2 802.11g wireless interfaces, and 128MB of NAND flash. The computational power of the RB133 is lower than that of all state-of-the-art low-cost RouterBoards featuring at least 3 network interfaces [12].

 ISBN: 978-1-61208-205-9

```
# uci set system.@system[0].hostname=NANTools
# uci set system.@system[0].zonename=Europe/Rome
# uci set system.@system[0].timezone=CET-1CEST,M3.5.0,M10.5.0/3
# uci commit system
# echo "$(uci get system.@system[0].hostname)" > /proc/sys/kernel/hostname
# uci set network.lan.ipaddr=192.168.1.222
# uci set network.lan.netmask=255.255.255.0
# uci set network.lan.gateway=192.168.1.1
# uci set network.lan.dns=8.8.8.8
# uci commit network
```

Figure 3. Example of configuration of a policy router based on OpenWrt.

Tests where performed using *ab*, the Apache http server benchmarking tool, and *curl-loader*, an open-source tool simulating application load coming from multiple clients with their own IP addresses.

Since the typical usage pattern of a NAN entails a set-up phase, in which a new client choses its preferred edge router and the corresponding rules are set in the policy router, and a navigation phase, in which the rules are used to forward client's packets to the edge router of choice, two sets of experiments were needed to test the performance. The first set was aimed at measuring the time taken by the policy router to serve incoming requests, while the second set was aimed at measuring the overhead introduced by source-based routing during normal transactions.

The average time per request was of: 16.23ms to provide a static html page containing a single line of text, 518.35ms to provide a dynamic PHP page with the same content, 615.48ms to access a PHP page reading a record in the DB, 922.05ms to access a PHP page inserting a record in the DB, 666.75ms to insert a rule in iptables, and 761.00ms to insert a rule in iproute2. The most time consuming tasks were PHP executions (around 500ms) and DB insert operations (about 400ms). All the tests were iteratively performed for increasing levels of concurrency (up to 250) to test both robustness and performance scalability. All incoming requests were properly served, while, as expected, the delay scaled linearly with the number of concurrent requests. No significant overhead was introduced by source-based policy routing in terms of round-trip time and throughput.

To summarize the performance of the low-cost embedded platform used as policy router, we can say that it can take up to 1 second to set up a rule, while it introduces a negligible overhead on any other network operation.

V. Conclusions

The *NAN tools* are an open-source tool suite for implementing *neutral access networks* (NANs) [5]. This paper has presented the solutions provided by the NAN tools to the main scalability and interoperability issues which have to be faced in order to enable the actual applicability of the NAN model. The proposed solutions rely on source-based policy routing, as made available by the Linux support for advanced networking.

All the solutions outlined in this paper have been implemented in release 1.1 of the NAN tools, which is now under test within the *Urbino wireless campus* testbed [8], which counts more than 20,000 registered users and serves up to 500 simultaneous users.

An all-in-one embedded version of the NAN tools has been also implemented and tested on a MikroTik RouterBoard RB133 running OpenWrt, the performance of which can be regarded as a lower bound of state-of-the-art low-cost network processors. Experimental results demonstrate the practical applicability of the proposed solution.

The source code of the NAN tools is available online at http://www.wireless-campus.it:8000/nan-tools/.

Acknowledgment

The research leading to these results has received funding from the EU IST Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n 25741, project ULOOP (User-centric Wireless Local Loop), and from the Italian ICT4University Programme, project U4U (University for University).

The authors would like to thank Andrea Mazza and Tommaso Battazzi for their valuable contribution to the implementation of the NAN tools.

References

[1] A. T. Kearney, "A Viable Future Model for the Internet," *A.T. Kearney report*, 2010.

[2] ——, "Internet Value Chain Economics," *The Economics of the Internet, Vodafone Policy Paper Series*, 2010.

[3] European Commission, "Report on the public consultation on The open internet and net neutrality in Europe," *Electronic Communications Policy*, 2010.

[4] E. Pigliapoco and A. Bogliolo, "A Service-Based Model for the Internet Value Chain," in *Proceedings of Int. Conf. on Access Networks*, 2011.

[5] A. Bogliolo, "Introducing neutral access networks," in *Proceedings of Int.l Conference on Next Generation Internet Networks (NGI 2009)*, 2009.

 ISBN: 978-1-61208-205-9

[6] A. Seraghiti and A. Bogliolo, “Neutral access network implementation based on linux policy routing,” in *Proceedings of the 2009 First International Conference on Evolving Internet*. IEEE Computer Society, 2009, pp. 158–162.

[7] M. Marsh, *Policy Routing Using Linux*. SAMS, 2006.

[8] A. Bogliolo, “Urbino wireless campus: A wide-area university wireless network to bridge digital divide,” in *Proceedings of AccessNets-07*, 2007, pp. 1–6.

[9] F. Fainelli, “The OpenWrt embedded development framework,” in *Free and Open Source Software Developers’ European Meeting*, 2008.

[10] C. Newman, *SQLite (Developer’s Library)*. Sams, 2004.

[11] OpenWrt Wiki, *Web Server Configuration (uHTTPd)*, 2012. [Online]. Available: http://wiki.openwrt.org/doc/uci/uhttpd

[12] MikroTik, *RouterBoard Products*, 2012. [Online]. Available: http://routerboard.com/

 ISBN: 978-1-61208-205-9