Dynamic Names and Private Address Maps: Complete Self-Configuration for MANETs

Christophe Jelger and Christian Tschudin
Computer Science Department
University of Basel, Switzerland
{Christophe.Jelger,Christian.Tschudin}@unibas.ch

ABSTRACT

In this paper we propose a self-configuring framework for mobile ad hoc networks (MANETs). Our architecture is based on the degenerated use of network address translation (NAT) where each node locally builds its "private address map", i.e. its own view of the addressing plan of the entire MANET. In contrast to previous schemes, MANET nodes do not need to coordinate in order to generate unique (IP) addresses.

A key aspect of our scheme is that routing is based on names rather than on addresses as the latter play a debatable role in ad hoc networks. While this requires nodes to choose unique names, we show how a simple yet efficient distributed scheme can solve this problem by generating dynamic names based on a set of keywords locally defined by each MANET user. We discuss the architectural impact of this approach in the bigger context of names, addresses, routability, and self-configuring networks, and report on our implementation.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*

General Terms

Design.

Keywords

MANET self-configuration, address autoconfiguration, name autoconfiguration.

1. INTRODUCTION

Mobile Ad hoc NETworks (MANETs) are formed by wireless nodes that collaborate to route packets inside a multi-hop network. A key property of MANETs is that they do not rely on any existing network infrastructure or any sort of pre-established coordination between the network's nodes. By coordination, we refer to any kind of a priori human intervention required to "run" a MANET, as for example address and name configuration.

A large number of mechanisms have been proposed in order to dynamically autoconfigure the IP addresses of the nodes of a MANET [1, 2]. The main trend in self-addressing is that each node generates a tentative IP address and ensures that no other node uses this address inside the MANET: if this is the case, the node is allowed to use the address. The process of address generation is usually based on some random and/or cryptographical techniques while the validation phase usually relies on a discovery scheme similar to the route discovery procedure used in reactive routing protocols.

This questionable "address guessing" approach which requires many heuristics regarding timeouts and probing scope can be avoided alltogether if each MANET node locally chooses a unique address for each of its active communicating peers. In this paper we describe in more detail how this *private address map* technique works and how it meshes with other non network-layer related issues like name resolution and human-friendly identifiers. The final outcome is a fully self-configuring MANET architecture.

The remaining of this paper is organised as follows. In the next section, we discuss and challenge one of the core assumption of the MANET community, namely the use of an IP-centric networking model. In Section 3 we present the original concept of private address maps and show how it solves the problem of IP address autoconfiguration in MANETs. In Section 4 we extend our proposal with a dynamic distributed naming scheme which is one of the first attempts to tackle the challenging MANET naming problem. We then discuss some of the details related to our proposal and implementation and later present some related work. We finally conclude the paper.

2. REVISITING THE MANET NETWORKING MODEL

In his seminal paper [3], J. Shoch has proposed a set of abstract definitions for the three fundamental networking concepts of names, addresses, and routes:

The *name* of a resource indicates <u>what</u> we seek,
an *address* indicates <u>where</u> it is, and
a *route* tells <u>how to get there</u>.

The main advantage of these definitions is that they are

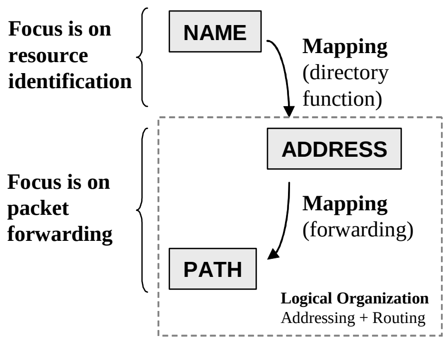
Focus is on resource identification
NAME
Mapping (directory function)
Focus is on packet forwarding
ADDRESS
Mapping (forwarding)
PATH
Logical Organization
Addressing + Routing

Figure 1: Name to address to path mapping.

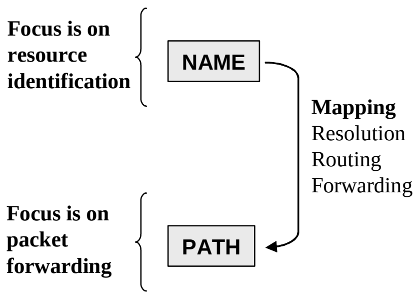
Focus is on resource identification
NAME
Mapping
Resolution
Routing
Forwarding
Focus is on packet forwarding
PATH

Figure 2: Name to path mapping.

very simple[1] if one considers the current Internet networking model. In the Internet, most names (e.g. email and SIP addresses, URLs) are based on the use of fully qualified domain names (FQDNs) which are directly mapped into IP addresses via DNS resolution. An IP address is then mapped into a path via the repeated next-hop lookup that takes place during packet forwarding. This predominant networking model is illustrated by Fig. 1.

The core assumption of this networking model is that the address space is topologically organised in the sense that addresses have a topological meaning: an address serves to locate an entity. Because the network part of an IP address indicates to which part of the network the address belongs to, IP addresses have to be topologically correct and are hence not portable: if the topological point of attachment of a device changes, then its previous address will not be valid anymore (in most cases).

Addresses are an intermediate artifact that has merely been introduced to hierarchically organise the network (for scalability reasons) and to perform packet forwarding and routing with a fixed-length bit sequence. The underlying fundamental activity, however, is the mapping between the name ("what we seek") and the route ("how to get there"). Another implementation of this activity (than Shoch's intermediate step involving addresses) is known as "routing by name" [4] and is illustrated in Fig. 2 where a name is directly mapped into a forwarding path. Note that some (small-scale) network architectures have used this concept prior to the predominance of IP (e.g., IBM's NetBEUI where NetBIOS names are directly mapped into MAC addresses).

2.1 The quest for address uniqueness

To date, the MANET community has largely embraced the fundamental assumption that routing and forwarding should be based on unique IP addresses. Following the Internet paradigm, the routing protocols specified by the IETF MANET working group ([5, 6, 7]) create routes that are labeled with globally unique identifiers[2]: the same IP address cannot be used by different nodes without jeopardising the accuracy of the routing protocol. This networking model perfectly suits the Internet because address uniqueness is resolved in an authoritative way: the allocation of global IP addresses is strictly controled by administrative entities which ensure that a given address is not assigned to two different hosts.

However, a MANET is typically expected to run without any existing infrastructure or any a priori configuration involving human coordination. As a result, a large number of mechanisms have been proposed in order to dynamically autoconfigure the IP addresses of MANET nodes in a decentralized manner [1, 2]. Address autoconfiguration is however not a trivial operation and remains an open problem: MANET nodes have to coordinate in order to generate unique IP addresses, avoid address leaks, and resolve potential addressing conflicts due to the merging of MANET clouds. We here insist on the fact that the sole objective of address autoconfiguration is to satisfy the address uniqueness requirement. The current design space of address autoconfiguration is indeed fundamentally bounded by the assumption that MANET routing and forwarding must be performed at the IP layer with unique addresses.

As mentioned in the previous subsection, the essential networking operation is yet the mapping between a name and a path and this mapping does not necessarily require the use of addresses. In a clean-slate approach to solve the address autoconfiguration problem, one can wonder whether it is possible to apply the "routing by name" networking model to MANETs. The fundamental issue relates to the role of addresses in the current IP-based MANET model, and if these addresses need to be unique. In the next subsection, we briefly analyse how the fundamental networking concepts of names, addresses, and routes are implemented in the current IP-based MANET model. Our controversial position is that (global) address uniqueness is not a fundamental requirement for the correct operation of MANETs because Shoch's original concept of an address does not apply to MANET networking.

2.2 Names, addresses, and routes

While Internet routing is based on the use of IP addresses, a human user interfaces with the network via different types of identifiers which are generally derived from domain names[3]. Typically, the name to address mapping is performed via the DNS system. It is important to note that in the Internet this resolution process is decoupled from the actual

[1]Shoch's paper is being refered to in RFC-791 in order to situate IP with respect to naming, addressing and forwarding.

[2]If the MANET is not connected to the Internet or if NAT is used at the edge(s) of the MANET, an address is "only" required to be unique inside the MANET.

[3]Domain names are indeed embedded in all sorts of identifiers such as URLs (e.g. http://example.com), SIP names and emails (e.g. bob@unibas.ch).

data forwarding. Although the data can only be sent once the name is mapped into an IP address, IP routing itself does not depend on name resolution. However in MANETs and unlike in the Internet, the lack of infrastructure does not allow the users to resolve identifiers into IP addresses via the traditional DNS system. Nevertheless, there exist two alternatives to *identify* nodes.

The first alternative is to directly use IP addresses. For example, if a user cannot specify that he wants to communicate with e.g., bob.unibas.ch, then the only way to reach Bob is to know an IP address of the device associated with Bob's identifier(s). In such a scenario and using Shoch's terminology, an IP address takes on the role of an identifier. That is, the user (mis)uses an IP address to express "what he seeks". In that scenario, IP addresses must be unique as they are used to identify the nodes of the MANET. However there is a major flaw when using IP addresses as identifiers. In a MANET where address autoconfiguration is dynamically performed, one cannot expect a device to be always configured with the same address so using an IP address as an identifier can lead to the wrong destination. Moreover, human users are reluctant to use IP addresses so using them as identifiers is not satisfactory.

The second alternative is to use a MANET-specific scheme to perform name resolution in a distributed and decentralized way. One proposal as originally described in [8] is to perform the node lookup procedure of a reactive routing protocol with a (DNS) name instead of an IP address. Typically, a route request message that contains the name to be resolved is broadcasted over the entire MANET, and eventually the target node sends back a route reply message that contains its IP address. This procedure creates the appropriate forwarding states in intermediate nodes and in the two communicating nodes. Note that this scheme still requires to have unique addresses and hence relies on the correct operation of an address autoconfiguration protocol.

However, the second scenario has a major difference when compared to traditional name resolution in the Internet: the name lookup procedure is closely coupled with routing as it creates the appropriate forwarding states in the MANET. We here remind that MANET routing is in essence host-based because nodes are mobile and because routing aggregation is hard or even impossible to achieve[4]. This results in a flat addressing space in which the role of an IP address is reduced to the sole identification of one (or multiple) forwarding path(s). With a reactive routing protocol, forwarding paths are built on-demand without any hint about the location of the target node. That is, whatever the IP address of a node is, a given path leading to this node only depends on the underlying physical topology and the address in itself carries no topological information. In other words, in a MANET an IP address looses its fundamental nature ("where it is") which is (as in the Internet) to locate an entity inside a topologically organised address space. Still using Shoch's terminology, we argue that the role of an IP address is reduced to the definition of a route, i.e. "how to get there". This observation is the key driver of our position which is to "simply remove" IP addresses from the MANET networking model, but keep IP compatibility for the applications.

[4]In a MANET there is usually no notion of IP subnetworking and even if there was such a notion, the nodes sharing a given network prefix would not necessarily be geographically located such that routing aggregation could be efficiently performed.

2.3 Summary of contributions

Building from our previous work, we believe that the "routing by name" networking model is more suitable to MANETs than the traditional Internet-like Shoch's model. We have already shown in [9] how the operations of both node lookup and path setup can be performed with DNS names instead of IP addresses. While seeking backwards compatibility, we show that it is possible to resolve a DNS name (i.e. "what we seek") into a forwarding path (i.e. "how to get there") without having to rely on IP addresses for naming, routing and forwarding. To label forwarding paths we rely on label-switching forwarding, i.e. a forwarding technique that does not require the use of globally unique identifiers (except in the ingress nodes). Label-switching forwarding is used in widely deployed protocols such as MPLS [10] and (previously) ATM, and it was also demonstrated in MANET protocols that operate below IP at layer 2.5 [11, 12].

However, our previous work still relied on nodes being configured with unique addresses (autoconfigured) and unique names (manually configured). The main reason to maintain unique addresses is that Shoch's networking model is deeply embedded in modern applications which largely operate in an *IP-by-default* mode where applications typically expect a given hostname to be resolved into a unique address. Moreover in order to identify MANET nodes, we assumed in our previous work that each host was manually configured with a unique name. Compared with our former work, the contribution of this paper is twofold. First, to satisfy the local "IP requirements" of IP-oriented applications, we propose a degenerated use of network address translation (NAT) that can free MANET nodes from having to maintain globally unique addresses. We call this scheme *private address maps*. Second, in order to avoid the requirement that names are manually configured, we use a dynamic naming scheme that creates human-friendly names based on a set of text attributes. We call this framework *dynamic shortest discriminating names*. Note that each of these two schemes is fully independent and does not rely on the other scheme. For example, private address maps can be used with any naming scheme (e.g. manual configuration with FQDNs, peer-to-peer naming, etc). Reciprocally, our protocol implementing dynamic shortest discriminating names could be used on top of any MANET routing protocol (with some assumptions).

3. PRIVATE ADDRESS MAPS

As mentioned earlier and thanks to the success of the Internet, modern network applications largely operate in an *IP-by-default* mode. The main consequence of this IP hegemony is that a scheme based on "routing by name" must be extended in order to satisfy the local IP requirements of each network node. Typically, these requirements include the resolution of a DNS name into an IP address and the creation of a suitable entry in the routing and/or ARP tables. To this end, we propose to make an extreme use of network address translation (NAT) in order to create private address maps.

3.1 The dual-NAT scenario

As described in its original specifications (RFC-1631), network address translation (NAT) was introduced during the mid-90's as a short-term solution to the foreseen address depletion problem of IPv4. Far beyond its initial ambition, NAT is now massively deployed in the Internet because it also provides a useful security feature in the sense that nodes inside a NATed domain are not globally reachable (or *visible*) from the rest of the Internet.

The idea of making an extreme use of NAT in order to solve the IP address autoconfiguration problem of MANETs comes from the following scenario illustrated by Fig. 3. A node A initiates a communication towards (say a web server) node B which DNS name resolves into the IP address 2.1.67.92. Actually the node B is located in a NAT domain and the real address of B is 10.1.34.22. The node A is also located in a NAT domain. The particularity of this "dual-NAT" scenario is that each node has a distinct view of the addressing plan as shown in Table 1. As for MANET address autoconfiguration is concerned, we will strech this dual-NAT scenario to an extreme case where each node locally creates its own private map of the MANET addressing plan.

View of	IP Address of A	IP Address of B
Node A	192.168.23.65	2.1.67.92
Node B	1.2.32.89	10.1.34.22

Table 1: Addressing views in a dual-NAT scenario.

3.2 Let me choose your address

To overcome the problem of address uniqueness, we propose a scheme where each node locally chooses the IP addresses of its correspondants and creates its own *private address map* of the network. Each active path is thus easily *labeled* with a conflict-free <source,destination> IP address pair. Figure 4 illustrates the concept of private address maps where each node with active connections creates its own local view of the MANET addressing plan (for space reasons the map of node E is not shown). In essence, our proposal can be viewed as a degenerated extension of the previously described dual-NAT scenario where the NAT functionality is moved to the edges of a communication, i.e. in each of the two end-nodes.

Note that while the path discovery scheme described below is taken from our previous work [9], the concept of private address maps is actually fully independent of the protocol used to create routes. We here describe one possible implementation of this concept but this does not restrict the applicability of the framework. To describe how private address maps are used let us consider the MANET shown in Fig. 4 and the specific example described below and illustrated by Fig. 5.

Assume that node A initiates a route discovery towards an identifier (e.g. bob.net.lunar) that happens to be an identifier of Node B (Fig. 5a). Node B receives the route request message, notices that it is the target node and generates a route reply message towards node A that builds the label-switched forwarding path (Fig. 5b). At the same time, node

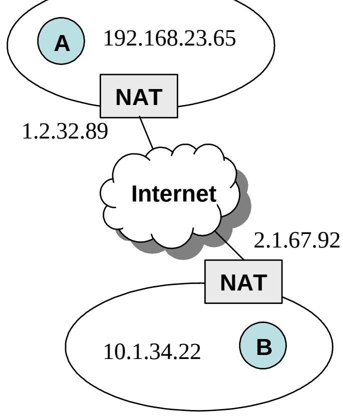

Figure 3: Dual-NAT scenario.

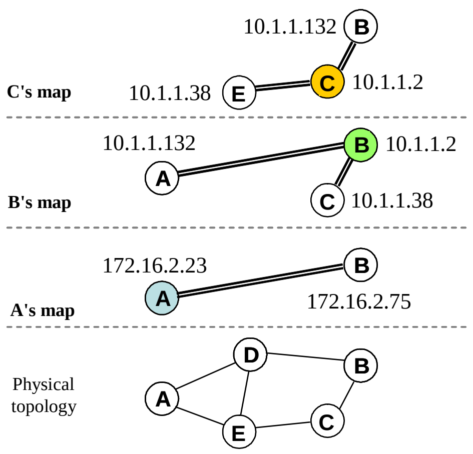

Figure 4: Private address maps.

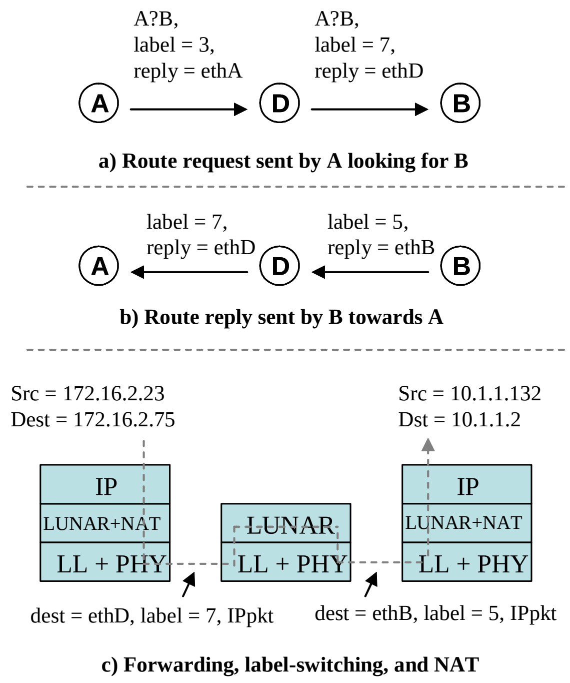

Figure 5: Private address maps: an example.

B locally assigns[5] to node A a randomly chosen IP address (from its locally configured address range) that it has not yet assigned to any other node. Upon receiving the route reply message, Node A also locally assigns to node B a randomly chosen IP address. Note that there is no sort of global coordination at the MANET level in order to generate globally unique addresses. Each node creates its own local view of the MANET addressing plan. In our example, one possible addressing view for each of the two nodes is shown in Table 2.

View of	**IP Address of A**	**IP Address of B**
Node A	172.16.2.23	172.16.2.75
Node B	10.1.1.132	10.1.1.2

Table 2: Local addressing views.

Now if node A wants to send data to node B, the IP header of the outgoing packets will contain addresses from A's private map: the source address will be 172.16.2.23 and the destination address will be 172.16.2.75. Note that before transmitting, our routing protocol will add an appropropriate label to the outgoing packets such that they will follow the already established forwarding path up to node B. When a packet reaches node B, the NAT module inside B *translates* the addresses of the IP header into suitable addresses from B's private map (Fig. 5c). That is, the source address becomes 10.1.1.132 and the destination address is changed to 10.1.1.2. Both the IP and transport layer checksums are then re-calculated and the packet is passed to the IP stack. Reciprocally when node B wants to send data to node A, it uses the IP addresses of its private address map and, at the receiver side, the NAT module inside node A will replace the IP addresses according to the local addressing map.

The key feature of this radical address autoconfiguration technique is that it is fully conflict free. Each node indeed creates its own view of the addressing plan and address collisions are avoided without any kind of global coordination among the MANET nodes: Address uniqueness is no longer an issue. If we again consider our previous example, we show in Table 3 a valid addressing plan where some IP addresses are used multiple times. Note that our proposal also supports the extreme case where all nodes are configured with the same IP address but where each node maintains a conflict-free private address map.

View of	**A's addr**	**B's addr**	**C's addr**	**E's addr**
A	172.16.2.23	172.16.2.75	-	-
B	10.1.1.132	10.1.1.2	10.1.1.38	-
C	-	10.1.1.132	10.1.1.2	10.1.1.38

Table 3: Private address maps with address re-use.

3.3 Private address maps implemented

Pragmatically, and in order to suppress the burden of any NAT configuration by the end-user, we have integrated and automated our MANET-specific NAT processing in the operation of LUNAR[6][11]. One of the core design principle of LUNAR is the creation of an *underlay*, i.e. a subnet illusion at the IP layer (see [11, 9] for details). Although not strictly required any more, we keep this design principle because it permits to have a unique routing entry at the IP layer of each node. The use of an underlay implies that each node creates its private address map from a locally configured subnet taken in the range of private IP addresses (e.g. 192.168.1.0/24). As for IPv6 is concerned, we use the fc00::/7 range of unique local IPv6 addresses [14].

It is worth mentioning that the local binding between an identifier and an IP address is maintained as long as the correspondant node is known to be alive. For this purpose, LUNAR uses a heartbeat-style protocol to populate a "yellow-pages" database that contains the names of all the reachable MANET nodes. This mechanism is essential because it helps to maintain a consistent mapping between names and IP addresses over longer time spans: it is indeed very common for applications to solely use IP addresses once names have been resolved.

4. DYNAMIC SHORTEST DISCRIMINATING NAMES (DSDN)

Our previous work on the self-configuring LUNAR protocol relied on humans to configure unique names. However, in spontaneous MANETs which are expected to self-configure, relying on manual naming is by far too restrictive. We have therefore extended and tested[6] LUNAR with a naming scheme implementing "dynamic shortest discriminating names".

Naming is a key pillar of the Internet: it is by far the dominant framework used by human users to interface with the network. As for addresses, the attribution of DNS names in the Internet is resolved in an authoritative manner: hierarchically organised entities allocate names and subdomains of the domain upon which they have full administrative authority. For a particular domain, human users use *off-line* techniques (e.g. manual configuration, scripts, database tools) to ensure that each assigned name is unique.

In a MANET however, one cannot rely on an administrative authority to assign unique names. This is especially the case in MANETs which are not connected to the Internet. In such spontaneous environment, the network nodes are expected to use distributed algorithms in order to self-configure. Note that this issue is not restricted to the "routing by name" networking model since in Shoch's model naming is also the core interface between human users and the network. Surprisingly (to the knowledge of the authors), there has been very little research on distributed algorithms for MANET name autoconfiguration. Most of the research proposals related to name resolution usually assume that nodes are already configured with a unique name (e.g. [8, 9, 15]). We also briefly note that peer-to-peer systems generally rely on a central database to maintain unique identifiers (e.g. Skype).

Actually the only practical solution to self-configuring MANET naming is by Jeong et al. [16]. The main advantage of their scheme is that names include a large numerical field that guarantees a very high probability of uniqueness. Un-

[5]In our current implementation, this *assignment* is performed by sending back to the application layer a forged DNS reply message which contains the locally chosen address. This technique is similar to the "Bump-In-The-Stack" mechanism described in [13].

[6]This particular version of LUNAR has been implemented on the Linux operating system. It is available on the Uppsala/Basel Ad-Hoc Implementation Portal (http://core.it.uu.se/adhoc).

fortunately in practice, names end up being somehow unsuitable for human users (e.g. PAUL.36-56-78-FF-FE-9A-BC-DE.EUI-64.ADHOC). We argue that MANET names should be short and *human-friendly* in the sense that they should be easy to manipulate and remember. We also believe that a user should have the flexibility to *steer* the naming algorithm such that it creates a name that suits his or her needs.

4.1 Name generation

Borrowing ideas from [17], our scheme requires that each user specifies a set of keywords that will be used dynamically to create a unique name and to resolve potential naming conflicts. However, while [17] focuses on the advertisement of services that are defined by a set of <attribute,value> pairs, we promote the use of an ordered list of unrestrained keywords. The use of attributes as proposed in [17] indeed implies that there exists some pre-established list of categories (i.e. an ontology)[7] which (we believe) limits expressiveness. We rather promote an open scheme where each user is free to choose whatever keywords better suit the user's preferences.

Let us consider an example. Say that John specifies the following list of ordered keywords

`{John,Doe,UniBasel,CSDept,Switzerland}`

while his brother Paul chooses

`{Paul,Doe,UniBasel,BioDept}`.

The goal of our proposal is to dynamically create shortest discriminating names by appending keywords until a unique name is created or a name conflict is resolved. To force "diversity", a name must always contain at least two keywords; to avoid "over-long" names, a name must always be as short as possible. In our current example, the first node will create the initial hostname `john-doe` and the second host will create the initial hostname `paul-doe`. Note that implementation-wise, we use a virtual domain space (i.e. net.lunar) in order to allow MANET nodes to identify MANET-local name resolution requests from potential Internet-wide requests [9]. Hence John's name is really `john-doe.net.lunar` but for clarity we will ommit the virtual domain suffix. Also note that alike the DNS system, names are internally stored with lower-letter characters.

4.2 Resolving name conflicts

As mentioned earlier, our LUNAR implementation uses a heartbeat-style mechanism in order to maintain a "yellow-pages" database that contains the names of all the reachable MANET nodes. Note that the "yellow-pages" are typically used by human users in order to learn the names of the nodes reachable via the MANET. To populate this database, each node periodically broadcasts its name over the entire MANET. Hence name conflicts can easily be detected. In our previous example, say that a third user joins the MANET and appears to use the following keywords

`{John,Doe,HighTechCie,SanJose,USA}`.

Upon joining the MANET, the current name of this third user is also `john-doe`. This user will soon broadcast its name over the entire MANET, and the swiss John will eventually detect the name conflict (note that for timing reasons the opposite case may occur). In practice, a name broadcast message creates a concast tree towards the sender to allow conflict messages to be sent back. When the swiss John detects the name conflict, he can send back a conflict message to the american John: this message simply contains the keywords list of the swiss John. When the american John receives the conflict message, he can compare both keywords lists and find the first discriminating keyword that can be used to extend his current name: the resulting name is `john-doe-hightechcie`. At the same time, he also sends its keywords list to the swiss John who extends his name to `john-doe-unibasel`. At that time, the name conflict is resolved. Note that when a node creates a tentative name, it always checks in LUNAR's "yellow-pages" that the new name does not already exist. Also note that the conflict resolution scheme can be run many times. Hence if a subsequent name collision occurs, the name `john-doe-hightechcie` could be further extended to e.g. `john-doe-hightechcie-sanjose`. It is also worth mentioning that multiple collisions can be handled simultaneously by our conflict resolution scheme.

Let us now consider a few design choices. First and in order to emphasize impartiality, we decided that during a conflict all conflicting nodes must change their names. This is also done to help non-conflicting users to clearly distinguish all the (previously) conflicting names. Second, we always keep the first two keywords because they are expected to be the most explicit ones. Third, we only append discriminating keywords. That is, if for some (clumsy) coincidence the name of the company located in San Jose was `UniBasel`, the conflict would have produced the names `john-doe-sanjose` and `john-doe-csdept`. Our intention is to avoid producing long names that become difficult to use, so the non-discrimitating `unibasel` keyword is removed. One should however note that this debatable design choice has no influence on the ability to resolve name conflicts because these keywords are non-discriminating. However, while this design choice may have an influence on the expressiveness of generated names (e.g. `CSDept` without `UniBasel` is less precise), users are free to use composite keywords (e.g. `CSDept@UniBasel`) if they wish to do so. Finally, in the rare event that two users exactly picked up the same list of keywords, we simply append a number and a warning keyword (e.g. "collision!") after the first two keywords.

4.3 Identifying nodes

Although (human readable) names are the prime abstraction used by users to identify a network node, networking protocols do not necessarily rely on names to identify nodes. For example, TCP uses IP addresses (and port numbers) to identify a correspondant. In our previous work [9], we use host identifier tags (HITs [18])[8] in order to uniquely identify a LUNAR node. Typically once the HIT of a node is known, LUNAR exclusively uses the HIT to build routes and to further identify the node. That is, what really identifies a node at the LUNAR level is the HIT and not the name.

Hence while our naming scheme introduces dynamic names, LUNAR always maintains an up-to-date mapping between

[7] Agreeing on a limited set of attributes can quickly become a standardization nightmare.

[8] In LUNAR, a HIT is a 128-bit pattern that has a very high probability of uniqueness. It is currently built at LUNAR startup by appending 6 bytes obtained from the hostname, the MAC address of the interface used by LUNAR, and the system date coded in network order on 4 bytes (seconds elapsed since Epoch).

names and HITs. This ensures that on-going connections are maintained if a node suddenly changes its name. Note that while names may change, the IP addresses assigned to correspondants by our private address maps framework do not change because each mapping is also linked to a specific HIT. This is very important because applications generally rely on IP addresses to identify connections once name resolution was successfully performed. For example, an on-going ssh connection would not be affected at all by a name modification. However, some applications embed names in their playload and, in particular cases, this would require specific corrective actions. This issue is discussed in subsection 5.3.

4.4 Name reduction

Another desired property of our dynamic naming scheme is that names should remain as short as possible. This means that while a name can grow to resolve a name collision, it should also shrink if a previous colliding node leaves the MANET. In our current implementation, a name reduction is triggered when an entry in a node's "yellow-pages" database times out. Concretely, the node checks the database to see whether it could remove a keyword from its current name without causing a name collision. If this is possible, the node simply removes the keyword and advertises its new name. Because name conflicts are resolved by adding keywords on the right side of the names, the prefered scheme to perform a name reduction is to remove the rightmost keyword. However, if this is not possible, a node is also allowed to remove another keyword (except the first two keywords).

5. DISCUSSION

In this section, we discuss various topics that relate to our proposed architecture. Our objective is to proactively answer some of the questions that our proposal may have raised. The first three subsections situate our proposal with respect to the current Internet, best-practices in NAT deployment, and naming compatibility with existing applications. The fourth subsection briefly discusses some MANET-specific issues related to naming and addressing. Finally, the last subsection presents a *by-product* of our proposed architecture.

5.1 Interworking with plain and global IP

While the concept of private address maps is particularly suitable to MANETs not being connected to the Internet, it can also be used in MANETs that have a connection to the Internet. In our previous work [9], we have shown how the use of a virtual domain name (i.e. net.lunar) allows MANET nodes to distinguish MANET-local node lookups from Internet-wide lookups. Communication requests to nodes located outside the MANET can be forwarded to an Internet gateway running a traditional NAT module. For Internet hosts, the whole MANET appears like yet another NATed domain. For a MANET node, an Internet host is just another correspondant for which an IP address was locally assigned in the node's private address map. Moreover and in order to distinguish between MANET-local and Internet-wide IP addresses within a private map, a MANET node should assign two different network prefixes for Internet hosts and MANET nodes. Also note that for packets travelling to the Internet, NAT is solely performed by the MANET gateway; in the reverse direction, NAT is performed by the MANET node[9].

5.2 NAT: problems and solutions

While NAT is widely deployed in today's Internet, it is known that certain applications can be affected if the two end nodes of a communication are not in the same addressing realm [19]. These problems can however be resolved via the deployment of application level gateways (ALGs), i.e. application-aware programs living in NAT boxes and fixing the problems caused by address translation. In addition, some protocols support NAT via dedicated extensions (e.g. [20]), and there also exist more generic solutions for NAT traversal (e.g. [21]). Due to its wide deployment, the problems introduced by NAT are well known and with careful design it is possible to avoid them [22]. Hence while private address maps are based on a generalized use of NAT, we believe that there exists sufficient know-how to make such a solution a viable one.

5.3 Applications and names

One potential issue introduced by dynamic names is that a small set of applications have strong naming dependencies. For example, a web page can contain an absolute HTML link that refers to a fully qualified domain name (FQDN) that is hardcoded in the source code. This of course becomes a problem if the target node changes its name dynamically. Note that this problem does not exist with relative links that do not include a FQDN.

A straightforward way of solving that kind of problems is to rely on the same mechanisms used by application level gateways (ALGs) in NAT boxes. While such programs translate IP addresses, one could plausibly extend their functionality to also translate names. However in a MANET context, complex techniques are not necessarily required. For example, the web pages of a MANET node hosting a web site with only relative links would be reachable without having to dynamically modify the links in the source code of the pages. To handle a name change, a local ALG could simply send to the browser application an HTML 'redirect tag' in order to update it with the new name.

5.4 On naming and addressing

While this paper proposes a dynamic naming scheme, we would like to remind that the naming issue is not a particularity of the "routing by name" networking model. IP-based MANET architectures are also expected to solve this problem while also having to resolve the issues related to address autoconfiguration. Hence with private address maps we do not transform the problem of configuring unique addresses into the problem of fixing unique names: our architecture solves the first issue, allowing research to be focused on the more challenging naming problem. We indeed believe that naming is a key factor for successful MANET deployments because it provides the main user-to-network interface while addressing remains a technical networking issue.

5.5 Off by default!

An extra feature of the combination of reactive routing, label-switching forwarding, and the absence of global addressing is that it enables each MANET node to explicitly

[9]If the Internet host belongs to a NATed domain then an extra NAT operation is performed at the host site.

select the set of nodes that are allowed to communicate with it. That is, and contrarily to the Internet communication model, node reachability is "off" by default [23]. This feature is attractive because it prevents nodes from receiving unwanted traffic and can hence help to partially reduce the effect of denial of service attacks.

In contrast to [23], the default-off behavior of our architecture is achieved without the use of a reachability protocol and without having to perform reachability checks at each intermediate hop during packet forwarding. In a somehow similar way to [24], the core idea is that forwarding states are being setup during the path establishment procedure of reactive routing. However while [24] proposes to construct addresses on-the-fly during communication setup, our scheme is based on a specific feature of MANET reactive routing protocols. During path setup, forwarding states are activated by the route reply message that travels from the node being searched to the originator of the route request. Therefore, a node can restrict its reachability by simply not responding to unwanted (e.g. non cryptographically signed) route request messages. Note that this is achieved with some small overhead because route request messages always have to reach the target node (i.e. intermediate nodes are not allowed to reply even if they have a valid route).

A misbehaving node could however still jam the network with route request messages, i.e. a well-known attack against MANET reactive routing protocols. While this topic is out of the scope of our main research interest, in the future we expect to integrate attack mitigation techniques developed by other researchers (e.g. [25]).

6. RELATED WORK

This section discusses some architectural proposals that, while not directly considering MANET networks, also advocate the use of names as end-node identifiers at the network layer.

To cite just a few, TRIAD [26] and IPNL [27] introduce network architectures that rely on FQDNs as host identifiers. However these two proposals rely on the DNS architecture to perform name lookups, i.e. a requirement that cannot be satisfied in spontaneous and infrastructure-less networks. While our current lookup scheme relies on flooding (as existing reactive routing protocols like [5]), our architecture is not confined to this scheme. One could for example replace this part of the architecture with a lookup mechanism based on distributed hash tables. However, recent work [28] has shown that a similar approach to flooding could scale to very large networks at the realistic condition that they are hierarchically organised (e.g. in a BGP style).

As routing is considered, both TRIAD and IPNL rely on existing routing information gathered by some proactive routing protocol. TRIAD relies on a name-based routing protocol and IPNL re-uses existing Internet routing to connect private realms. Actually, both protocols still operate with topologically organised addresses (either inside a realm or inside the existing global Internet). In contrast, our proposal merely uses IP addresses to maintain a backward compatibility with existing applications that mostly operate in an IP-by-default manner. Strictly speaking, the "routing by name" networking model does not involve addresses.

Another difference relates to forwarding. In this proposal we advocate the use of label-switching forwarding in order to avoid the overhead of maintaining global labels. This contrasts to both TRIAD and IPNL which use either addresses (IPNL) or names (TRIAD) during the forwarding process. However label-switching forwarding, while simple to implement, does not naturally support label aggregation. At the scale currently considered for MANETs this is not a serious issue but we nevertheless currently work on a label-switching scheme that will support some form of label aggregation.

7. CONCLUSION

The first technical contribution of this paper is a simple yet efficient conflict-free IP address autoconfiguration scheme for mobile ad-hoc networks. It is based on the use of private address maps where each MANET node maintains its own view of the addressing plan. NAT is used at each end-point of a communication in order to translate incoming packets into the local addressing map. In contrast to existing schemes, private address maps do not require any kind of coordination among MANET nodes.

Our second technical contribution is a dynamic naming scheme where each MANET node creates (for itself) a discriminating name based on an ordered list of keywords. Our protocol handles and resolves naming conflicts by dynamically extending conflicting names with discriminating keywords. This effort is one of the first attempts to propose a flexible and dynamic naming scheme for MANETs.

On an architectural side, we have discussed the adequacy of Shoch's communication model with respect to mobile ad hoc networks. We challenge the assumption that the Internet-like IP-based networking model should be blindly applied to MANETs. Our central argument is that the use of addresses is ineffective because MANETs are not topologically organised networks. In contrast, the spontaneous nature of these networks makes it more suitable to use a networking model where human-friendly identifiers are directly mapped into forwarding paths.

In practice, we have successfully implemented a LUNAR version that uses private address maps and dynamic shortest discriminating names. To our knowledge, this is the first fully self-configuring architecture for ad hoc networks, i.e. an essential prerequisite when it comes to deploying spontaneous networks on-demand.

8. ACKNOWLEDGEMENTS

Part of this work was carried out while Christophe Jelger was being sponsored by an ERCIM (see www.ercim.org) postdoctoral fellowship during his stay at Fraunhofer FOKUS Berlin. His current work at the University of Basel is part of the Autonomic Network Architecture (ANA) project (see www.ana-project.org), which is funded by the European Commission under the proactive initiative on Situated and Autonomic Communications.

9. REFERENCES

[1] K. Weniger and M. Zitterbart. Address Autoconfiguration in Mobile Ad hoc Networks: Current Approaches and Future Directions. *IEEE Network*, 18(4):6–11, July 2004.

[2] C. Bernardos and M. Calderon. Internet Draft - Survey of IP address autoconfiguration mechanisms for MANETs (expired), July 2005.

[3] J. Shoch. Inter-network naming, addressing, and routing. In *Proceedings of 17th IEEE Conference on Computer Communication Networks*, pages 72–79, 1978. Washington, D.C., USA.

[4] B. Hauzeur. A Model for Naming, Addressing, and Routing. *ACM Transactions on Office Information Systems*, 4(4):293–311, October 1986.

[5] C. Perkins, E. Belding-Royer, and S. Das. RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing, July 2003.

[6] T. Clausen and P. Jacquet. RFC 3626 - Optimized Link State Routing Protocol (OLSR), October 2003.

[7] R. Ogier, F. Templin, and M. Lewis. RFC 3684 - Topology Dissemination Based on Reverse-Path Forwarding (TBRPF), February 2004.

[8] P. Engelstad, D. V. Thanh, and G. Egeland. Name Resolution in On-Demand MANETs and over External IP Networks. In *Proceedings of IEEE ICC'03*, May 2003. Anchorage, Alaska.

[9] C. Jelger and C. Tschudin. Model Based Protocol Fusion for MANET-Internet Integration. In *Proceedings of WONS'05*, January 2006. Les Ménuires, France.

[10] E. Rosen, A. Viswanathan, and R. Callon. RFC 3031 - Multiprotocol Label Switching Architecture, January 2001.

[11] C. Tschudin, R. Gold, O. Rensfeld, and O. Wibling. LUNAR - A Lightweight Underlay Network Ad-Hoc Routing Protocol and Implementation. In *Proceedings of NEW2AN'04*, February 2004. St. Petersburg, Russia.

[12] V. Untz, M. Heusse, F. Rousseau, and A. Duda. On Demand Label Switching for Spontaneous Edge Networks. In *Proceedings of ACM Sigcomm Workshop in Future Directions in Network Architecture*, September 2004. Portland, OR, USA.

[13] K. Tsuchiya, H. Higuchi, and Y. Atarashi. RFC 2767 - Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS), February 2000.

[14] R. Hinden and B. Haberman. RFC 4193 - Unique Local IPv6 Unicast Addresses, October 2005.

[15] X. Hong, J. Liu, R. Smith, and Y.-Z. Lee. Distributed Naming System for Mobile Ad-Hoc Networks. In *Proceedings of International Conference on Wireless Networks (ICWN'05)*, June 2005. Las Vegas, USA.

[16] Jaehoon Jeong, Jungsoo Park, and Hyoungjun Kim. NDR: Name Directory Service in Mobile Ad-Hoc Network. In *Proceedings of the 5th International Conference on Advanced Communication Technology (ICACT 2003)*, January 2003. Phoenix Park, Korea.

[17] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, December 1999. Kiawah Island Resort, SC, USA.

[18] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Internet Draft - Host Identity Protocol, draft-ietf-hip-base-06.txt, June 2006.

[19] M. Holdrege and P. Srisuresh. RFC 3027 - Protocol Complications with the IP Network Address Translator, January 2001.

[20] J. Rosenberg and H. Schulzrinne. RFC 3581 - An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing, August 2003.

[21] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. RFC 3489 - STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), March 2003.

[22] D. Senie. RFC 3235 - Network Address Translator (NAT)-Friendly Application Design Guidelines, January 2002.

[23] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by Default! In *Proceedings of Fourth ACM Workshop on Hot Topics in Networks (HotNets-IV)*, November 2005. College Park, USA.

[24] M. Handley and A. Greenhalgh. Steps Towards a DoS-resistant Internet Architecture. In *Proceedings of ACM Sigcomm Workshop on Future Directions in Network Architecture*, September 2004. Portland, OR, USA.

[25] Z. Eu and W. Seah. Mitigating Route Request Flooding Attacks in Mobile Ad Hoc Networks. In *Proceedings of the International Conference on Information Networking (ICOIN'06)* , January 2006. Sendai, Japan.

[26] D. Cheriton and M. Gritter. TRIAD: A New Next-Generation Internet Architecture, July 2000. Stanford Computer Science Technical Report.

[27] P. Francis and R. Gummadi. IPNL: A NAT-Extended Internet Architecture. In *Proceedings of ACM Sigcomm'01*, pages 69–80, August 2001. San Diego, CA, USA.

[28] J. Pujol, S. Schmid, L. Eggert, M. Brunner, and J. Quittek. Scalability Analysis of the Turfnet Naming and Routing Architecture. In *Proceedings of First ACM Worshop on Dynamic Interconnection of Networks (DIN'05)*, September 2005. Cologne, Germany.