Model Based Protocol Fusion for MANET-Internet Integration

Christophe Jelger, Christian Tschudin

Model Based Protocol Fusion for MANET-Internet Integration

Christophe Jelger
Fraunhofer Institute FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
Christophe.Jelger@fokus.fraunhofer.de

Christian Tschudin
Computer Networks Research Group
University of Basel, Bernoullistrasse 16,
CH-4056 Basel, Switzerland
Christian.Tschudin@unibas.ch

***Abstract*— With the wide adoption of wireless communication technologies, the current networking design of the Internet architecture has shown some limitations. Restricted by inherent layering constraints, valuable networking information cannot flow freely inside the network stack and potential operational optimizations are impossible to achieve. To overcome these limitations, we extend the current trend of cross-layer approaches with a framework called *underlay protocol fusion*: the basic building blocks of Internet functionality are factorized out and merged in a function pool where information sharing and operational optimizations are performed.**

To illustrate our approach, we present LUNARng (LUNAR next generation). It is a fully distributed underlay protocol designed for the Internet integration of wireless ad hoc networks (MANETs) where fundamental services such as name resolution, address autoconfiguration, and IPv4/IPv6 routing are transparently available whether the MANET is connected or not to the Internet. Internet integration refers here to the ability to *insert/remove* a MANET *into/from* the logical organization of the Internet without any loss of functionality. Moreover by using *protocol models*, the underlay nature of LUNARng allows to optimally merge (with respect to the multi-hop nature of MANETs) network operations which are traditionally carried out at different layers of the protocol stack.

***Index Terms*— Underlay MANET, Internet Integration, Protocol Fusion.**

I. Introduction

A. Internet integration

The design of the Internet is based on a thirty-years old layering approach which aimed at factoring out functionality. Networking concepts were sought which are able to stretch from local scale to global size, from slow links to highspeed trunks, from PDAs to supercomputers. In spite of its slow evolution and monolithical design, the "canonical set" of protocols collected in the Internet-Suite has done a surprisingly good job during the last decade. Today however, the limitations of this one-size-fits-all approach have become visible especially with the advent of wireless communications.

These limitations can be linked to the incapacity of the Internet to scale in a functional way at the networking layer. It has evolved through a patch style which has not added variety: additions were made in a stealth way and have not dared to radically change or extend the core of IP forwarding. We refer here to the introduction of the hidden routing hierarchy and mechanisms of AS, CIDR or MPLS as well as other less successful projects (in terms of large-scale deployment) like RSVP, IP Multicast, and Mobile IP. Due to the end-to-end principle, the place where the Internet has envisaged and endorsed functional scaling, that is the possibility to freely add arbitrary customized functionality, is the application layer. This is where remarkable breakthroughs have been achieved and variety was obtained: DNS, eMail, Web, VPN, VoIP and P2P are the highlights to mention here.

The low flexibility of the current Internet protocol suite is particularly striking when considering mobile wireless ad hoc networks (MANETs). The inherent distributed and infrastructure-less nature of MANETs has indeed highlighted how fundamental services of the Internet rely on a centralized client-server model. That is, the absence of basic services such as name resolution or address allocation does not strictly prevent networking, but it strongly restrains the adoption of wireless ad hoc networking as a plug-and-play technology for the masses. Therefore the ability of an autonomous MANET to exhibit Internet-like functionalities (while not being connected to the Internet) is one facet of Internet integration: users should not experience a loss of commodity other than the loss of global connectivity.

A complementing aspect is the seamless integration of (mesh) MANETs with the logical (e.g. global addressing) and operational (e.g. name resolution) organization of the Internet. This property is the second facet of Internet integration: the ability for a MANET to adapt its internal behavior in order to *insert* itself into a larger organization such as the Internet.

B. Underlay design for MANETs

As stated earlier, the strict layering approach of the existing legacy TCP/IP model is slowing down or even preventing innovative functionalities to appear at the network layer. It is commonly agreed that more inter-layer coordination is needed in wireless networks when considering on one side the network layer, and on the other side the physical and the link layers [1]. However, more coordination is also required among higher

This work was carried out while Christophe Jelger was being sponsored by an ERCIM fellowship during his stay in the Computer Networks Research Group at the University of Basel. He is now with Fraunhofer Fokus, also sponsored by an ERCIM fellowship.

layer protocols. For example, we already demonstrated in [2] how a single DNS name resolution request procedure can gather enough *cross-protocol* data to fulfill the tasks of link-layer resolution and path setup.

In order to achieve such optimizations, we introduce an underlay shim that performs *protocol fusion* based on *protocol models*. The goal is first to gather the previously isolated information provided by different task-specific protocols of the network stack, and second to optimize the operation of these protocols by anticipating their needs. In contrast to more classical cross-layer techniques which provide hints and triggers between layers, we use an underlay located between the IP and Ethernet layers in order to have full control over the data coming in and out of a node. Tasks that were previously carried out by remote servers at different layers are now performed at layer 2.5 (i.e. hence the name *underlay*), and the historical barriers between the somehow isolated protocols involved at the network layer are suppressed. As a result, effective optimizations can now be achieved. Moreover, to realize the two facets of Internet integration, the functionalities provided at layer 2.5 are activated on-demand when the MANET is not connected to the Internet, or they can be bypassed depending if a given service is provided by the infrastructure-based network which provides the global connectivity.

The paper is organised as follows. In the next section, we situate our approach with respect to traditional cross-layer schemes and we introduce our underlay technique. We then briefly introduce LUNARng and summarize the features it provides. We then describe the mechanisms that we use to perform the Internet integration of wireless ad hoc networks, and we also detail the concept of protocol fusion. We also present some implementation details of our approach which has been successfully validated and deployed on a real testbed. Finally, we conclude with a discussion of future open research challenges.

II. Cross-layering vs. protocol fusion

A. Traditional cross-layer design

Cross-layer design is an active field of research which so far has not looked into Internet integration issues. Instead, the focus is on *wireless* networking, mainly because many protocols and services of the *wired* Internet are inefficient in the presence of unreliable wireless links and unpredictable topological changes. Without cross-layer design, the existing layer boundaries unfortunately prevent the development of potential optimizations[1]. Since the differences between wired and wireless networks lie in the physical and link layers, i.e. the layers located below the IP layer, a large majority of cross-layer techniques concentrates on providing lower-layer feedback to the network layer [1] (e.g. to notify link-layer events such as layer 2 handovers) and, to a smaller extent, to the transport layer (e.g. to achieve TCP performance optimizations via wireless-specific fine-tuning of TCP parameters). We denote such approaches as lower-layer feedback techniques, and illustrate them by Fig. 1 where valuable information is passed from the lower-layers to the higher layers, which can subsequently optimize their operation. The main advantage of this approach is that it becomes possible to re-design a given protocol such that it specifically reacts to some lower-layer signals, i.e. the protocol becomes aware of what happens in lower-layers. However, such lower-layer awareness does not come without a cost: it requires code modifications inside the protocol stack which can restrict the possible wide-scale adoption of such changes. As a consequence, most cross-layer schemes remain research prototypes. It is also worth mentioning that cross-layer designs using an *information-bus* (i.e. a transversal layer that receives/sends specific feedback from/to all layers via specific function calls) unfortunately suffer from the same implementation and deployment restrictions.

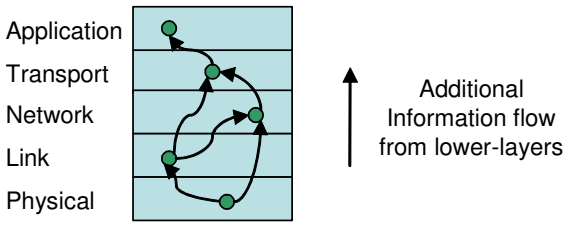

Fig. 1. Lower-layer feedback

B. Case study: the failure of name resolution

In the Internet, name resolution is performed via the Domain Name System (DNS) which relies on a hierarchy of servers distributed around the world. One issue is that dynamic name resolution in traditional IP networks assumes that there exists a reachable DNS server at all time: the whole operation of name resolution collapses if no server is available. All existing operating systems do not even try to send a name request if no DNS server is configured in the system: if a node is not configured with a DNS server address, it simply assumes that dynamic name resolution is not available. The implementation of a DNS-compatible name resolution system in a MANET is therefore challenging in many ways.

Actually, a *natural* way of performing name resolution in a MANET is to use a decentralized approach in which a node of the MANET replies to a broadcasted name request for which it is the target. Different flavors [2][3][4] of such an approach can be found in the literature. Although the operation of distributed name resolution resembles the route discovery procedure of a reactive routing protocol, it is more difficult to implement than routing since the DNS operation is *hard coded* in current operating systems and applications. That is, by default, a node configured with a DNS server address sends its unicast DNS request messages via the network interface towards the server. In a MANET, this procedure becomes irrelevant and it should be replaced with a MANET specific mechanism.

[1]One can note that the *restricted-scope* of competencies of standardization bodies (i.e. IEEE vs. IETF) also restricts the design and potential outcomes of cross-layer optimizations.

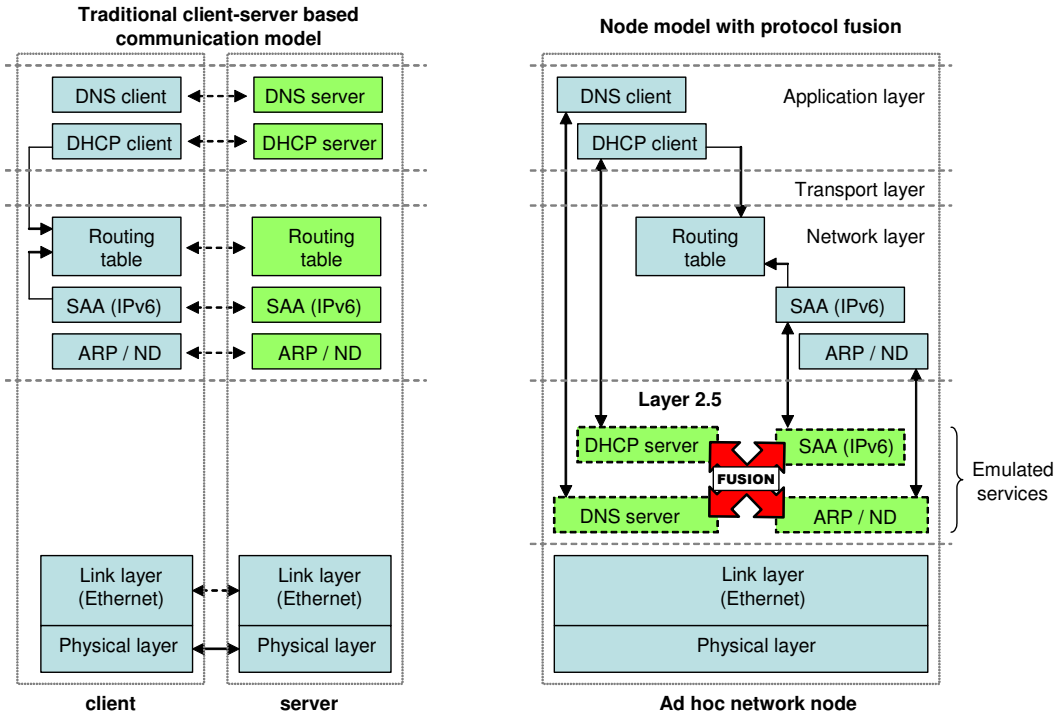

Fig. 2. Protocol emulation and fusion

Moreover, a name resolution scheme for MANETs should not prevent a node from resolving names in the classical way when the MANET is connected to an infrastructure-based network. In particular, the existing APIs and related protocols should remain unchanged since it is not conceivable to modify the huge amount of existing applications such as web browsers and email clients.

C. Internet integration (part I): service emulation with models

As introduced earlier, our approach relies on an underlay protocol located below the IP layer and above the Ethernet layer. Being located below IP, the underlay protocol is aware of all traffic coming in and out of a node. By manipulating the stream of messages passing by, it becomes possible to perform some *cross-protocol* optimizations. By cross-protocol optimizations we mean that, among other options it is possible to merge the operation of independent protocols by anticipating the needs of a given protocol: the global behavior of a TCP/IP-based protocol stack is indeed well known. For example, with a reactive routing protocol for wireless ad hoc networks (such as AODV [5] and DSR [6]), a successful route request (RREQ) procedure will always be followed by a link-layer resolution request for the next hop node towards the destination. One can therefore design the route request procedure such that it also performs the anticipated link-layer address resolution. A main advantage of using an underlay protocol is that it requires no modifications to the existing protocol stack. Furthermore, protocol fusion is invisible to legacy layers: it is thus possible to *fool* the higher layers by sending appropriate control messages to the protocols located in both the network and the application layers. It thus becomes possible to build *protocol models* which mimic the behavior of some specific network functionality in order to provide Internet-like services while the MANET is not connected to the Internet. We define this property of Internet integration as *service emulation*. One can note that the underlay can also be used to hide the multi-hop nature of a wireless ad hoc network such that the IP stack believes that the node is connected to a classical IP subnet on a single layer 2 link.

The concept of protocol emulation and fusion is illustrated by Fig. 2. The left side of the figure shows the client-server model used in traditional Internet-like networking. Fundamental services such as domain name resolution (DNS) and address autoconfiguration (DHCP for IPv4/IPv6, and SAA [7] for IPv6) fully rely on the presence of a dedicated server. Other protocols such as ARP and ND [8] are not based on the client-server model but for optimization purposes (as described later) their operation is also preempted by the underlay. In the right part of Fig. 2, we illustrate the underlay-based models located at layer 2.5. In order to overcome the absence of legacy servers, models of the basic Internet services are implemented in the underlay: these *emulated* functionalities are activated on-demand when required. Moreover, at the underlay level it is now possible to optimize the operation of protocols which were previously opaque to eachother. With this protocol fusion, network operations become optimized to the topological and operational properties of wireless ad hoc networks.

III. LUNARng

To explore and demonstrate the feasibility of underlay fusion, protocol emulation, and Internet integration, we have extended the original operation of LUNAR [9], i.e. a reactive routing protocol initially designed to back up the development of network pointers [10]. As for the features provided by the protocol fusion in the underlay, our next generation LUNAR (LUNARng) combines IPv4 and IPv6 path setup, link-layer address resolution (ARP/ND), and name resolution in a single request/reply operation optimized for distributed wireless ad hoc environments. Moreover, thanks to the use of network pointers as the basic forwarding abstraction, an IPv6 multi-hop data path can include nodes which are only IPv4 enabled (and vice-versa).

The two facets of Internet integration are also covered, since LUNARng provides Internet-like services via protocol emulation when the MANET is autonomous, and Internet adaptation via coherent global addressing, routing, and name resolution when the MANET is connected to the Internet. Moreover Internet integration is transparent to the MANET users, in the sense that they only witness the appearance or loss of global connectivity.

In practice, LUNAR is implemented as a Linux kernel module[2] that can be dynamically loaded on a host and which requires no single modification to the Linux kernel code. LUNAR positions itself between the IP and Ethernet layers (actually just above the wireless device driver) and creates a subnet illusion with respect to the IP stack. Upon startup, the LUNAR module creates a virtual network interface that is internally linked with the real wireless interface connected to the MANET. Hence, all traffic that flows via the virtual interface is seen by the LUNAR module which can specifically react to particular messages.

IV. Internet integration implemented

In this section, we describe the mechanisms developed in LUNARng in order to perform Internet integration and optimize the operation of Internet-like protocols with respect to distributed wireless networking.

A. Filling expectations and building models

As stated previously, the basic steps and behavior of a communication startup with the TCP/IP protocol stack is well known. In a wired network, the initiation of a communication usually conforms to the following steps (we assume that each step is successful):

1. The user specifies the name of the host s/he wishes to communicate with,
2. The IP stack triggers an ARP/ND (IPv4 Address Resolution Protocol or IPv6 Neighbor Discovery) request in order to resolve the link-layer address of the next hop towards the DNS server (or of the DNS server itself),
3. The system sends a DNS request to resolve the target host name,
4. The IP stack sends an ARP/ND request in order to resolve the link-layer address of the next hop towards the target (or of the target itself).

In a MANET that uses a reactive routing protocol, if we assume that a node is configured with a DNS server address, and if we assume that there exists a DNS server in the MANET, the following steps are executed:

1. The user specifies the name of the host s/he wishes to communicate with,
2. The MANET routing module triggers a route request (RREQ) procedure to find a path to the DNS server,
3. The IP stack triggers an ARP/ND request in order to resolve the link-layer address of the next hop towards the DNS server (or of the server itself),
4. The system sends a DNS request to resolve the IP address of the target host name,
5. The MANET routing module triggers a RREQ/RREP procedure to find a path to the IP address of the target host,
6. The IP stack sends an ARP/ND request in order to resolve the link-layer address of the next hop towards the target (or of the target itself).

It is clear that the specificities of MANETs already increase the total overhead because routes have to be discovered on-demand. Moreover, we assumed here that there existed a DNS server, but this assumption will usually be false in real infrastructure-less MANETs. Hence to resolve the specific issues introduced by the distributed and autonomous nature of wireless ad hoc networks, we introduce an optimized scheme which specifically considers and addresses the particular features of MANETs.

B. Revisiting route requests

The key element of our underlay approach is that in a MANET, one can trigger the route request procedure with the **name** of the destination host rather than with its IP address. At the same time, if the RREQ procedure can gather enough information, the number of required steps can now be greatly reduced:

1. The user specifies the name of the host s/he wishes to communicate with,
2. The MANET routing module triggers a route request RREQ/RREP procedure to find a path to the specified target host name. The RREP eventually contains the target IP address(es) and the link-layer address of the next hop towards the target.

Once the originating host receives this information bundle, it knows all relevant details to answer subsequent information requests (ARP) internally, and the communication can start immediately. The path discovery procedure is thus triggered by the name request message, i.e. an application layer protocol. The route is then discovered (network layer), and the link-layer address is resolved at the same time (network and link layers). Our underlay approach perfectly suits to the above optimization since we can capture the initial DNS request and translate it to an appropriate RREQ message: with a single RREQ/RREP procedure, we have performed name resolution, path setup, and link-layer address resolution.

[2]Available at http://core.it.uu.se/adhoc - The Uppsala/Basel Ad-Hoc Implementation Portal.

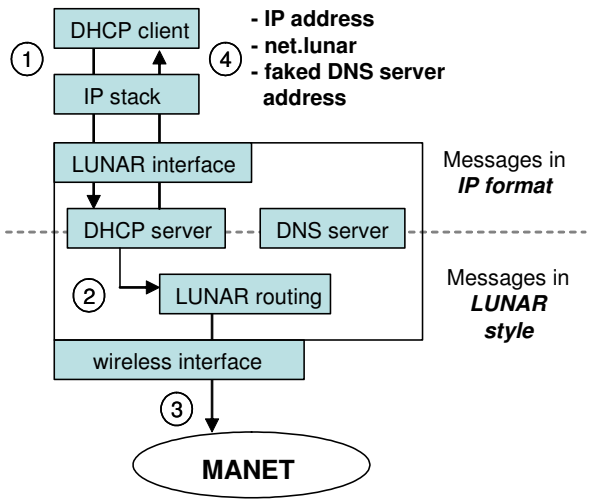

Fig. 3. LUNAR DHCP operation

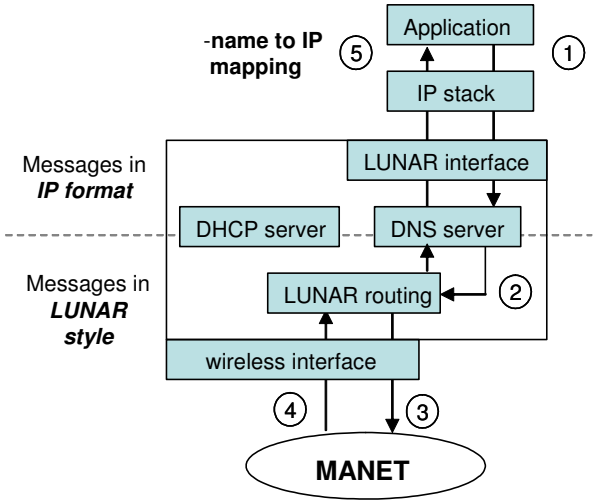

Fig. 4. LUNAR DNS operation

C. Internet integration (part II): Internet adaptation

On top of the merging of resolution requests, LUNARng supports address autoconfiguration by implementing a virtual DHCP server which assigns the IP address to the virtual interface, should the user want to automatically configure this interface via DHCP. This mechanism is illustrated by Fig. 3. In step 1 of Fig. 3, a DHCP client sends a request towards the LUNAR interface. This request is intercepted by the DHCP engine of LUNAR which randomly chooses an address within a pre-defined LUNAR subnet (e.g. 192.168.42.0/24), and which then checks the uniqueness of this address by trying to build a path towards this address (steps 2 and 3). If the path setup fails (i.e. indicating that the address is not used), a faked DHCP message is sent to the dhcp client application (step 4). This message includes the IP address to be used on this interface, the *net.lunar* domain name, and the address of a faked DNS server reachable via the LUNAR interface.

We also use a similar mechanism in order to perform IPv6 stateless address autoconfiguration (SAA [7]). LUNAR intercepts router solicitation (RS) messages sent by the IP stack, and returns back a faked router advertisement (RA) message which contains a MANET-global prefix[3].

Moreover, in order to perform Internet adaptation when the MANET is connected to the world-wide network, LUNARng also supports global address autoconfiguration based on *prefix continuity* [11]. That is when there exists a gateway to the Internet, LUNARng can coherently distribute a topologically correct and globally routeable prefix to the nodes of the MANET. The subnet illusion is maintained, and IPv6 multi-homing is also possible (if multiple gateways are present). Moreover, DNS requests for targets which are not inside the MANET are forwarded to the gateway: this is made possible since we introduce a virtual namespace as described in subsection IV-E.

[3]Since IPv6 scope-local addresses have been deprecated, we currently use unique local IPv6 unicast addresses (FC00::/7) as MANET-local addresses. See the recently standardized IETF RFC-4193 for details.

D. DNS operation

The interception of a DNS request is illustrated by Fig. 4. In step 1 of Fig. 4, an application triggers the sending of a DNS request that is intercepted by the LUNAR DNS engine. This triggers a route request procedure which uses the name of the target to identify the expected destination (steps 2 and 3). When the target discovers its name in the route request message, it sends back a route reply message which contains its IP address (steps 4 and 5). Note that this message also contains the MAC address of the next hop node towards the target destination: the node which triggered the name resolution request therefore also implicitly performs the link-layer resolution usually carried out by the ARP and ND protocols. The LUNAR module can then send back a classical DNS reply message to the application which eventually learns the IP address of the target. At that point, the network path is already established and the link-layer address of the next hop node is already known. When the IP stack subsequently issues an ARP or neighbor discovery (ND) request, the LUNAR module, which also intercepts these messages, can reply immediately without sending out any messages on the network.

E. net.lunar

The *net.lunar* domain name is configured at startup by the LUNAR module as being the default domain of a MANET node. Hence, a hostname request (i.e. not fully qualified) is transformed by the operating system into a net.lunar request which is recognized by the LUNAR module as being a MANET name resolution. In this way it becomes possible to identify a simple hostname lookup within the MANET if the user/application only specifies a hostname (e.g. the request for *cjelger* becomes a request for *cjelger.net.lunar*).

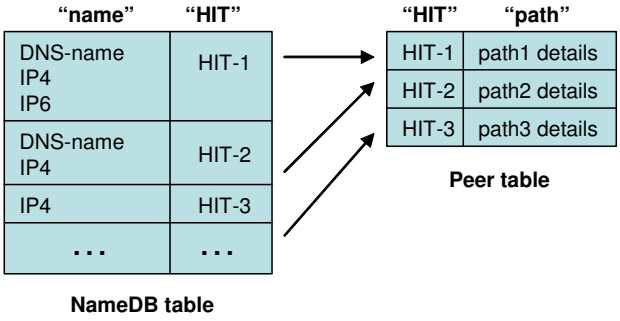

Fig. 5. NameDB and peer tables

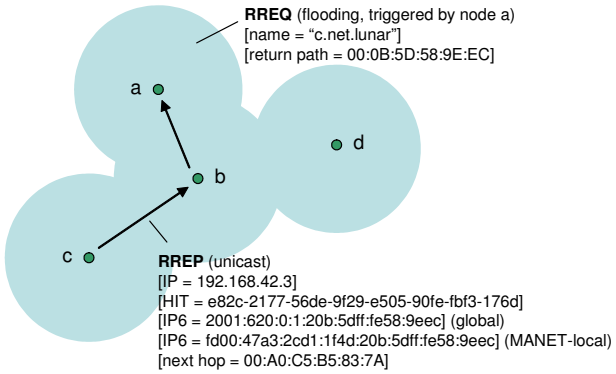

Fig. 6. RREQ/RREP procedure

In other words, a user can easily express its desire to trigger a name resolution inside the MANET. In contrast to hostname requests, FQDN (fully qualified domain name) requests (e.g. informatik.unibas.ch) are left unchanged by the operating system and the LUNAR module will recognize them as being a request for a host located outside the MANET. If the MANET is connected to the Internet via a gateway, non *net.lunar* DNS requests can be forwarded to the gateway which will potentially contact a traditional DNS server.

F. "Name" cache and path discovery

In order to avoid unnecessary path discovery procedures, each LUNAR node maintains a "name" cache, the so called NameDB (DataBase) table as shown by Fig. 5. This table contains all the known identifiers of a given correspondant (also called a *peer*): a DNS-name, one IPv4 and possibly multiple IPv6 addresses, and a host identifier tag or HIT inspired from the Host Identity Protocol (HIP [12]). As a HIT we use a random 128-bit string which becomes the entity which LUNAR uses to re-establish paths to a peer. That is, with either name resolution or plain ARP or ND resolution, we bind a peer's name and address to its HIT. All subsequent path lookup will be carried out with the HIT: as long as our name cache contains an entry for a peer, we will address this peer using its HIT. In order to populate the NameDB table, we use the LUNAR route discovery procedure to obtain the identifiers of a given target, as illustrated by Fig. 6.

Moreover, each node can gratuitously populate its NameDB table by overhearing the information contained in the RREQ messages it forwards. Also, a quiet node which does not send RREQ messages can also send unsollicited HELLO messages in order to notify the network about its existence. A user can thus learn the names of other computers connected to the MANET, since a summary of the information contained in the NameDB table is available via the Linux `/proc/net/lunar` file, illustrated below by Fig. 7. On this figure we can see the *emulated* DNS server peer, the localhost *baobab* which is IPv4 and IPv6 enabled, the peer *mango* for which a path (IPv4 and IPv6) is active (peer is resolved), the IPv6-only gateway *banana*, and the peer *cactus* for which a

```
# lunar (32 table entries, 4 peers)

FLAGS: N=name, H=hit, 4=IPv4, 6=IPv6,
       r=resolved, s=search, G=gateway

baobab.net.lunar    NH46..   -> Localhost
dns.net.lunar       N.4.r.   -> No lifetime
mango.net.lunar     NH46r.   -> Last heard 6s ago
banana.net.lunar    NH.6rG   -> Last heard 13s ago
cactus.net.lunar    N...s.   -> No lifetime
```

Fig. 7. The `proc/net/lunar` file.

RREQ procedure has been started (i.e. search state). Note that for all resolved nodes the HIT is always known.

When possible and to avoid bandwidth waste, LUNAR also uses the NameDB table in order to reply to DNS-PTR requests for the net.lunar domain. We remind that the goal of this *inverse resolution* procedure is to obtain the DNS name associated with a given IP address. One must note that with the traditional DNS operation, a host will send a PTR request to its DNS server even if it just resolved the IP address of the corresponding name. This additional overhead occurs because current operating systems do not implement a name cache and therefore a previously resolved *name → IP address* mapping cannot be re-used to perform the inverse resolution. In contrast, if a net.lunar name has recently been resolved into the corresponding IP address by a given node N, no PTR request is sent into the MANET if the node N wants to resolve this IP address into a name, as we use the NameDB table as a cache. Finally, to cope with the volatility of ad hoc networks, the name cache is periodically drained in order to handle address or name changes.

V. Open issues

An open issue is the case of two nodes picking identical hostnames in their FQDN. For example, two nodes respectively named *john.domain1.net* and *john.domain2.com* will both end up being identified inside the MANET as

john.net.lunar. To resolve this issue, we plan to add a mechanism to check for duplicate names at the same time when we check for duplicate IP addresses i.e., with the same RREQ message. Similar to picking another random IP address in case of a collission, LUNAR will start adding a suffix to the hostname and test again with the new name. For the previous example, the two nodes would thus end up being for example named *john.net.lunar* and *john33.net.lunar*.

Note that the new name is only used inside LUNAR i.e., at layer 2.5 and is mostly relevant to the *other* MANET nodes trying to contact the node with the new name: No attempts are made to change the host's original FQDN (*john.domain2.com*), which (a) would be a challenging implementation excercise and (b) could also be an unwanted source of confusion to the end user. In other words: we keep a mapping table between the new names and the LUNAR IP addresses, not the old (derived from FQDN) names and the old IP addresses. To allow users to distinguish between *john* and *john33* which with very high probability should have different HITs, a user could use the `/proc/net/lunar` file to distinguish the two nodes if the HIT is derived from a well-known public key identity (see [12]).

A second issue relates to the case of merging network clouds: This can lead to a MANET where some hosts have identical IP addresses[4] and/or identical LUNAR names. We (already) solve this problem by introducing stealth "host identifier tags" (HITs): any node joining the network with a colliding name or IP address will not be discovered by the LUNAR path establishment procedure as it has a different HIT. This strategy, which was proposed by [13] and which is inspired from [12], permits to maintain TCP connections although new hosts appeared in the MANET with the same IP address. However, we still need to implement a scheme to perform a large-scale IP(v4) renumbering.

VI. Conclusion

In this paper, we have shown and demonstrated how a MANET network can exhibit full Internet integration, thanks to the use of a dedicated underlay scheme which positions itself just below the IP layer. A key insight of this paper is that this underlay has to incorporate a model of the IP stack that it serves, should it wish to create a perfect fixed Internet illusion for it.

In addition to Internet integration, the underlay scheme also allows to optimize network operations with respect to the specific properties of distributed wireless ad hoc networking. This protocol fusion permits to merge the network operations of name resolution, link-layer address resolution, and network path setup in a single and efficient procedure. In particular this is done without any modifications of the existing operating systems, applications, and name resolver library. Since MANETs break several implicit assumptions (like client-server orientation) of IP networking, our underlay approach permits to rearrange basic functionalities in a MANET-friendly way. It is a first step towards a functional re-composition of IP-related protocols outside layering constraints.

[4]Note that address collision is merly an IPv4 issue, since the probability of IPv6 address collisions is very low, thanks to the use of Ethernet Unique Identifiers of 64 bits (EUI-64).

References

[1] S. Shakkottai, T. Rappaport, and P. Karlsson, "Cross-Layer Design for Wireless Networks," *IEEE Comm. Mag.*, vol. 41, no. 10, pp. 74–80, October 2003.

[2] C. Jelger and C. Tschudin, "Underlay Fusion of DNS, ARP/ND, and Path Resolution in MANETs," in *Proceedings of ADHOC'05*, May 2005, Stockholm, Sweden.

[3] 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.

[4] J. Jeong, J. Park, and H. Kim, "Name Service in IPv6 Mobile Ad-hoc Network connected to the Internet," in *Proceedings of IEEE PIMRC'03*, Sept. 2003, Beijing, China.

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

[6] D. Johnson, D. Maltz, and Y.-C. Hu, "Internet Draft - The Dynamic Source Routing Protocol for Mobile Ad hoc Networks (DSR), draft-ietf-manet-dsr-10.txt," July 2004.

[7] S. Thomson and T. Narten, "RFC-2462 - IPv6 Stateless Address Autoconfiguration," December 1998.

[8] T. Narten, E. Nordmark, and W. Simpson, "RFC 2461 - Neighbor Discovery for IP Version 6 (IPv6)," December 1998.

[9] 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.

[10] C. Tschudin and R. Gold, "Network Pointers," in *Proceedings of First ACM Workshop on Hot Topics in Networks (HotNets-I)*, October 2002, Princeton, NJ, USA.

[11] C. Jelger and T. Noel, "Proactive Address Autoconfiguration and Prefix Continuity in IPv6 Hybrid Ad Hoc Networks," in *Proceedings of IEEE SECON'05*, September 2005, Santa Clara, CA, USA.

[12] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Internet Draft - Host Identity Protocol, draft-ietf-hip-base-03.txt," June 2005.

[13] N. Vaidya, "Duplicate Address Detection in Mobile Ad Hoc Networks," in *Proceedings of ACM Mobihoc'02*, June 2002, Lausanne, Switzerland.