Underlay Fusion of DNS, ARP/ND, and Path Resolution in MANETs

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

***Abstract*— Name resolution is a key pillar of the fixed Internet: Without the existing distributed hierarchical infrastructure of name servers, neither eMail (MX records) nor web browsing (all links are expressed with logical names) would be possible anymore. A DNS substitute has to be provided for MANETs which permits, both in isolated network mode as well as in the mesh network case, to use logical names at the application level.**

In this paper we present a backward compatible distributed and decentralized name resolution scheme for MANETs: The particularity of our scheme is that name resolution, address resolution (IPv4) or neighbor discovery (IPv6) as well as routing path establishment are merged in a single operation. Using the underlay approach of LUNAR we integrate DNS logic at layer 2.5 such that classic Internet applications as well as operating systems settings and libraries do not have to be modified.

I. Introduction and motivation

While there has been quite a large amount of research related to unicast routing in mobile ad hoc networks (MANETs), less effort has been made in order to provide name resolution as a standard feature of MANET routing protocols. However, it is impossible to imagine the adoption of wireless ad hoc networking without name resolution: users simply do not deal with IP addresses, and this is even more true with 128-bit long IPv6 addresses. If we want to witness a wider adoption of MANET related technologies and protocols, name resolution MUST become a standard core element of MANET networking.

In the Internet, name resolution is performed via the Domain Name System (DNS) which relies on a hierarchy of servers distributed around the world. While this architecture has proven to work very well in the fixed Internet, it is not suitable to MANETs as has also been pointed out by [1]: in a MANET, one cannot rely on the presence of a DNS server..

This work was carried out during the tenure of an ERCIM fellowship (see http://www.ercim.org for details).

A more *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 [1]-[2] 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. As detailed in the next section, there are serious implementation issues that need to be resolved. In this paper, we extend the functionalities of an underlay routing protocol with an optimized distibuted name resolution scheme which merges the resolutions of names and link-layer addresses, and the establishment of multi-hop paths in a single network operation.

II. Name resolution in MANETs

A. General constraints

The implementation of a DNS-compatible name resolution system in a MANET is challenging in many ways. One issue is that dynamic name resolution in traditional IP networks at all time assumes that there exists a reachable DNS server: the whole operation of name resolution collapses if no server is available. Even worse, all 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. This means that in a MANET where no DNS server is present, one must trick the operating system so that it falsely believes that a name server is present.

Moreover, a name resolution scheme for MANETs should not prevent a node from resolving names in the classical way when it is connected in a wired 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.

B. *Existing approaches*

One *natural* way of performing name resolution in a MANET is to use a distributed and decentralized approach [1]-[2] similar to what is done by on-demand MANET routing protocols such as AODV [3] and DSR [4]. With such a scheme, a node that wishes to resolve a name broadcasts a specific name request message in the entire MANET. If a node of the MANET sees its name in the request, it replies to the originator of the message with a name reply message that indicates its current IP address.

While this on-demand request-reply procedure seems straightforward, it is quite difficult to implement in real systems because they are strongly constrained by the classical operation of name resolution. By default, a node configured with a DNS server address sends its unicast DNS request messages via the network interface towards the server. In an autonomous MANET, this procedure becomes irrelevant and it should be replaced with a specifically designed mechanism.

One solution is to add hooks and dedicated applications in the user process space. In [1], each node runs a MANET-specific name server that handles name requests within the ad hoc network. Traditional DNS requests sent by a MANET node are captured and translated into MANET-specific name request messages. Note that special filtering rules must also stop the classical DNS request from being transmitted on the ad hoc network. One difficulty with this scheme is to detect when the filtering rules should be removed, i.e. for example when the MANET becomes connected to the Internet via a wired network where a DNS server is available. Another solution proposed in [2] requires modifications in both the IP stack and the name resolver library. Unfortunately, these *patching* requirements can limit the adoption of such a scheme as most users are often relunctant or unqualified to apply kernel patches to their operating systems.

III. Underlay fusion : merging protocols

To support decentralized name resolution within a MANET, we have extended the original operation of a reactive routing-only protocol. In particular, we combine IPv4 and IPv6 path setup, link-layer address resolution (ARP for IPv4, and neighbor discovery (ND) [5] for IPv6), and name resolution in a single request/reply operation. In most cases, a name request procedure is indeed triggered by an application that wishes to communicate with some host. One can therefore perform *path setup with the name* instead of the IP address. Moreover, this single operation serves to gather all necessary information that permits the initiating node to answer ARP and ND request without having to put a second query on the MANET. Part of the DNS operation is now implemented at layer 2.5 instead of the application layer.

A. *Underlay approach*

To explore the feasibility of this approach, we use the LUNAR [6] routing protocol, which positions itself between the IP and Ethernet layers, creating a subnet illusion in a MANET. This underlay leads to immediate implementation benefits as LUNAR has full control on all traffic coming in and out of a node. In particular, the historical barriers between the somehow isolated protocols involved at the network layer (i.e. name resolution, link-layer address resolution, routing, address configuration) all disappear: this next generation LUNAR (LUNARng) becomes a single entry point where all networking information flows. In contrast to existing approaches, we do not need to redirect and block traditional DNS request messages as they naturally flow through the LUNAR module.

In practise, LUNAR is implemented as a Linux kernel module that can be dynamically loaded on a host. 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. For example, LUNAR implements a virtual DHCP server which assigns the IP address to the virtual interface when the user wants to automatically configure this interface via DHCP. This mechanism is illustrated by Fig. 1a. In step 1 of Fig. 1a, 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

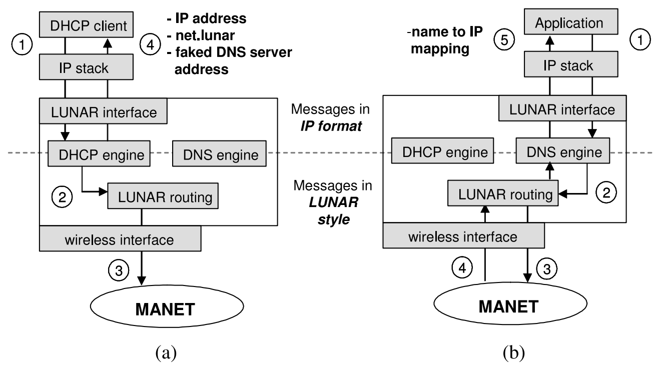

Fig. 1. LUNAR underlay operation

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.

B. Merging protocols

In a similar way, LUNAR intercepts link-layer address resolution messages (i.e. ARP for IPv4, and ND for IPv6) and DNS requests. As we use network pointers [7] to forward data at the LUNAR layer, path setup and link-layer address resolution can easily be done by the name resolution request procedure. This is made possible since the faked DHCP server provides a faked DNS server address to the operating system so that it believes that a DNS server is available in the network via the LUNAR interface. The interception of a DNS request is illustrated by Fig. 1b. In step 1 of Fig. 1b, 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 immediatly 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 set and the link-layer address of the next hop node is already known: the appropriate ARP/ND entry is added by the LUNAR module in the ARP/ND table. Networking protocols that are usually carried out independently were successfully merged in a single operation.

C. net.lunar

In order to distinguish between hostname lookups within the MANET, and FQDN (Fully Qualified Domain Name) requests, we introduce the use of a virtual namespace called **net.lunar**. The net.lunar domain name is configured at startup by the LUNAR module as being the default domain of a MANET node. When entering a MANET, it is indeed unlikely that a node can carry on using the name of the domain it originally belongs to, since it makes little sense as a MANET might not even be connected to the Internet. Also the IP address associated to a node in the global DNS system might not be valid any more when the node enters a MANET and potentially uses a different IP address.

Hence, a hostname request (i.e. not fully qualified) is transformed by the operating system into a net.lunar request which is regognized by the LUNAR module as being a MANET name resolution. It thus 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*). In other words, a user can easily express its desire to trigger a name resolution inside the MANET. In contrast to hostname requests, FQDN 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, the classical DNS request can be forwarded to the gateway which will potentially contact a traditional DNS server. This extra procedure is however out of the scope of this paper.

D. Name cache

When possible and to avoid bandwidth waste, LUNAR also uses a name cache in order to reply to 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* $\rightarrow$ *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. We keep a cache table in the LUNAR DNS engine in order to avoid the overhead of sending unnecessary PTR requests onto the MANET. Finally, to cope with the volatility of ad hoc networks, the name cache is frequently drained in order to handle unpredictable address or name changes.

E. Implementation details

In current operating systems, name resolution is a user-space process which is mainly controlled via a text file in Unix-like systems (usually /etc/resolv.conf), or the registry in Windows. In the fixed Internet, this information specifies the domain name of the host computer and a list of addresses of its prefered name servers. While the name resolution configuration can be edited manually (i.e. by the root user) or automatically via DHCP, the contained information is rather static as it is dependent of the particular networking context of the host computer. However, in an infrastructure-less MANET there is no such context, and one must find a way in order to automatically adapt the name resolver to the specificities of MANET networking. As already explained in Section III-A, we use a faked DHCP server in order to configure the IP address of a faked DNS server reachable via the LUNAR virtual interface. In practise, the dhcp client used by the operating system will edit the appropriate file (or registry entry), and applications will consequently send their DNS requests via the LUNAR virtual interface. At this point, it becomes possible to capture these requests and replace them with specific LUNAR-style name requests.

IV. Open issues

In this section we discuss three issues: name collissions, network partition merging, and user discovery.

One problem to still handle in LUNARng 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 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: LUNARng keeps 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*, we plan to use at the LUNAR level a node/user directory system (e.g. via the /proc filesystem in Linux) which shows the content of the cache i.e., a peer's LUNAR name as well as its original name.

The second problem relates to the case of merging network clouds: This can lead to a MANET where some hosts have identical IP addresses and/or identical LUNAR names. We (already) solve this problem by introducing stealth "host identifier tags": 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, not the name or IP address. That is: as long as our name cache contains an entry for a peer, we will address this peer using its HIT. Any other 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 [8] and which is now also considered in [9], permits to maintain TCP connections although new hosts appeared in the MANET with the same IP address.

The third issue is user or host discovery, which is an important element of spontaneous networking. Again we will use the binding cache for providing this information, where we see three ways of populating its entries: A first one is the LUNAR mechanism when a path is explicitely established between two peers; The second mechanism would rely on passively populating the cache by listening to RREQ messages which carry all the initiator's coordinates. The third would be the use of explicit discovery request by the user and would use LUNAR's broadcast support to explicitely solicit presence messages from any LUNAR neighbor. However, routes will still be built on-demand, preferably by specifying the LUNAR names provided by the directory system.

V. Conclusion

LUNARng has been successfully implemented on the Linux operating system as an easy-to-use kernel module[1]. LUNARng (i.e. LUNAR with name resolution) is a first step towards a functional re-composition of IP-related protocols outside layering constraints. By merging all the information flowing to and from the IP layer, LUNARng optimizes network operations that are traditionaly carried out by independent protocols.

In this paper, we have shown and demonstrated that the underlay scheme permits to merge the network operations of name resolution, link-layer address resolution, and network path setup in a unique and efficient procedure. In particular this is done without any modifications of the existing operating systems, applications, and name resolver library. Because current MANET-best-practises break many of the traditional paradigms assumed by IP networking, the use of an underlay approach is very appealing as it provides full control on the semantics and behavior of IP-related network protocols.

[1]See http://core.it.uu.se/AdHoc/ImplementationPortal

References

[1] 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.

[2] 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.

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

[4] 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.

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

[6] 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*, Feb. 2004, St. Petersburg, Russia.

[7] 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.

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

[9] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Internet Draft - Host Identity Protocol, draft-ietf-hip-base-02.txt," February 2005.