

# Lilith: an Interconnection Architecture Based on Label Switching for Spontaneous Edge Networks

Vincent Untz      Martin Heusse      Franck Rousseau      Andrzej Duda  
*LSR-IMAG Laboratory*  
*Grenoble, France*  
{*Vincent.Untz, Martin.Heusse, Franck.Rousseau, Andrzej.Duda*}@imag.fr

## Abstract

We consider the problem of interconnecting hosts in spontaneous edge networks composed of various types of wired or wireless physical and link layer technologies. We argue that this kind of networks requires a more sophisticated approach than standard IP forwarding: communication paths should be managed on a per flow basis, multiple paths need to be maintained to cope with link failures or changing topologies, and the interconnection architecture should provide a means for acquiring the information on destination reachability. To experiment with our approach, we have designed and implemented Lilith, a prototype of an interconnection node for spontaneous edge networks. We handle network dynamics by establishing MPLS (Multi Protocol Label Switching) label switched paths (LSP) on demand with a reactive ad hoc routing protocol. We present some measurements that show good performance with respect to the standard IP forwarding and important performance gains when multiple paths are used.

## 1. Introduction

Ubiquitous computing involves a wide range of electronic equipment such as sensors and actuators, home appliances, consumer electronics, and various computing devices. Situated at places like homes, offices, or public sites, they can be connected via heterogeneous wired or wireless physical and link layer technologies. Some devices have permanent location and others, such as handheld computers, move around changing their current point of attachment. All devices may benefit from the global Internet connectivity via one or more border routers. We consider the problem of interconnecting hosts in such *spontaneous edge networks* as illustrated in Figure 1.

The performance of wireless local area networks increases with the evolution of the physical layer (802.11a and 802.11g) so that they begin to connect audio/visual home

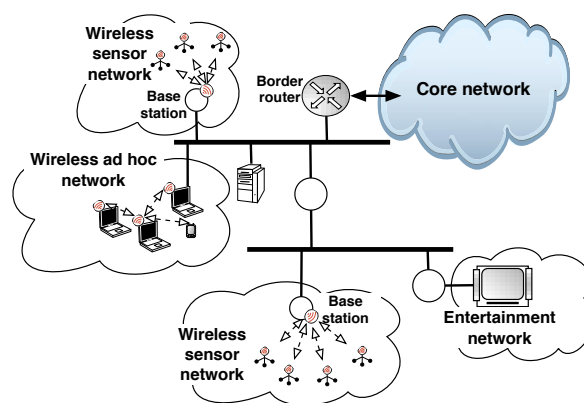


Figure 1. Spontaneous edge network

entertainment devices, but only over short distances. To cover larger areas, we can organize nodes into a multi-hop network to provide increased aggregated capacity, greater redundancy, and potentially multiple paths. Hence, we assume that all or some of hosts in a spontaneous network are organized as a multi-hop ad hoc network.

Another aspect of spontaneous networks is related to sensor networks in which small low power devices communicate over short range wireless links by using protocols optimized for low power consumption. Such sensor networks are usually organized in a hierarchical way: sensors communicate with base stations or application gateways having more power and communication bandwidth. A spontaneous edge network can suitably interconnect clusters of sensors, gateways, and appliances as well as provide access to the global Internet.

In this paper we propose an interconnection architecture suitable for spontaneous edge networks. Obviously we can use IP for interconnection, however we argue that this kind of networks requires a more sophisticated approach: communication paths should be managed on a per flow basis, multiple paths need to be maintained to cope with link fail-

ures or changing topologies, and the architecture should provide a means for acquiring the information on destination reachability. At the same time, the proposed interconnection protocol should be lightweight and efficient.

We have designed and implemented Lilith, a prototype of an interconnection node for spontaneous edge networks. It uses MPLS (*Multi Protocol Label Switching*) [7], the standard layer 2.5 for efficient forwarding of packets over various links. A flow follows a Label Switched Path (LSP) established on demand by an ad hoc routing protocol. Flows with different QoS requirements may use different LSP paths, for example we can make time-sensitive flows such as video go over high capacity links whereas Web traffic can use other links with lower capacity, or we can construct paths over links with good radio channel quality. While the best path (in the sense of some metrics) is used at a given instant, Lilith searches for other possible paths to use in case of a link failure or a change in topology. A Lilith node expects to periodically receive messages with statistics of the traffic on each LSP received by all its immediate neighbors. Such information can then be used to decide if a given link is broken, in which case it switches to another path. If it is not the case, Lilith uses the information to estimate link quality, the metrics that can be injected into the routing protocol.

In this paper we focus on the implementation and performance measurements of Lilith. A companion paper [8] provides more details on the motivation of our approach, comparisons with other proposals, and details of the design. The rest of the paper is organized as follows. We first briefly discuss the related work (Section 2) and describe the architecture of our proposal (Section 3). Then we present the implementation and measurement experiments (Section 4). Finally, we present conclusions and future work (Section 5).

## 2. Related work

Spontaneous edge networks are mainly related to ad hoc routing protocols and interconnection architectures for sensor networks.

The dynamic structure of spontaneous networks speaks in favor of considering them as ad hoc networks [5]. The MANET IETF working group has chosen to provide connectivity to a group of wireless hosts at layer 3: each host acts as a router to forward packets to other hosts, so the network behaves as a standard IP network without any advanced functionality.

Wireless sensor networks have recently received considerable attention [1]. Their characteristics are so different from the ad hoc networks studied by MANET that new protocols had to be developed. The main difference is the limited power and computational resources, so the proposed MAC and network layer protocols try to optimize these as-

pects and focus less on traditional performance objectives. The authors of the survey notice the importance of new interconnection schemes to allow easy communication between sensor networks and external networks such as the Internet [1].

Interconnecting clusters of sensors has become an important aspect of research. Relay Organization (ReOrg) is a topology control protocol which systematically shifts the network's routing burden to energy-rich nodes, exploiting heterogeneity [3]: high end powered nodes communicating via a 802.11 mesh network overlay sensor networks to increase the performance of sensor networks and their life time [3]. Similar approach has been adopted in the PEN project (Prototype Embedded Network) [9] that aims to build a wireless ubiquitous network composed of small, low cost, and low power nodes. End-to-end communication over a network for home automation such may benefit from a simple "*wormhole routing*": packets traverse a higher performance wired link between sensor radio spaces. A more elaborated switched routing service called R-Link has also been designed and built to support routing of data between arbitrary devices on demand.

The interconnection of hosts in home and office networks has been analyzed by Krishnamurthy *et al.* [4]. They have identified the main requirements of wireless home networks: autoconfiguration and meeting performance objectives by using multi-hop wireless technologies. However, no solutions have been proposed. We share their analysis and provide an interconnection architecture to satisfy the requirements.

## 3. Architecture of Lilith

We propose an interconnection architecture based on existing standard technologies (MPLS, on demand ad hoc routing) which glued together present several interesting features for spontaneous networking. Its idea is to organize a spontaneous network as a single IP subnetwork (broadcast or scoped multicast packets propagate to all hosts), to interconnect various links with MPLS, the standard layer 2.5 protocol, and to manage LSP (Label Switched Path) paths on demand with a reactive ad hoc routing protocol.

Integrating a connection based MPLS forwarding scheme with dynamic establishment of label switched paths presents several advantages for spontaneous networks. First, transient loops can be easily avoided, because LSP paths are explicitly created between end point hosts. Such a property is particularly interesting in the context of wireless networks with limited resources. Second, we can provide multiple paths for load balancing or traffic isolation for different QoS classes, including per flow, or per source or destination address. Another advantage is the possibility of acquiring the information

on the reachability of established LSPs: an interconnection node periodically sends a message with statistics of traffic received from all its neighbors.

Our approach can be characterized as *reactive*, because Lilith finds the best path at a given instant on demand. It is also *proactive*, because it maintains existing paths and tries to find alternate paths that can be immediately used as a back up after a link failure. Note that the hybrid approach allows elimination of broadcast storms if there is an existing alternate path when a link fails.

To experiment with our approach, we have designed and implemented Lilith, a prototype of an interconnection node for spontaneous networks based on MPLS.

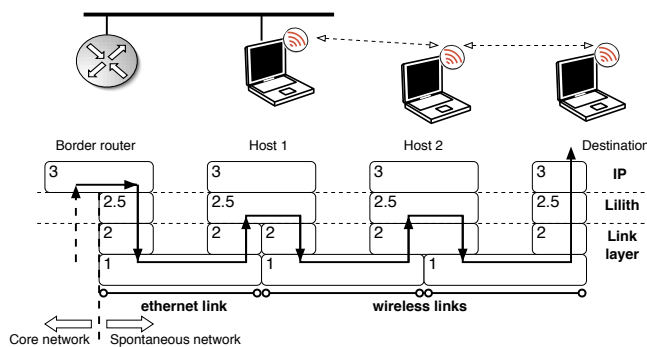


Figure 2. Protocol stack and packet path

Lilith forms with MPLS a layer 2.5 that presents to IP the abstraction of a single broadcast (scoped multicast in IPv6) link layer network. In this architecture, routes and data paths are dissociated so that different protocols can be used for finding a route and establishing a data path. A route is a list of intermediate nodes between a source and a destination. It is not used for forwarding packets, but for establishing a path between the source and the destination. Figure 2 presents the structure of the protocol stack at hosts and intermediate nodes as well as the data path taken by a packet.

The architecture of Lilith is presented in Figure 3. It relies on an ad hoc routing protocol, which in the current implementation is based on a simple reactive routing mechanism similar to AODV [6]. When describing the elements of Lilith below, we use the routing protocol as an example, however it can be replaced by any other ad hoc routing protocol, even a proactive one such as OLSR [2]. Lilith is composed of the following elements:

*Interception module.* It intercepts unicast packets for which no LSP exists and all broadcast packets. The unicast packets are passed to the Routing Protocol module that starts the search for a route, which in turn will trigger the establishment of a LSP. When it is done, the packet can be sent over

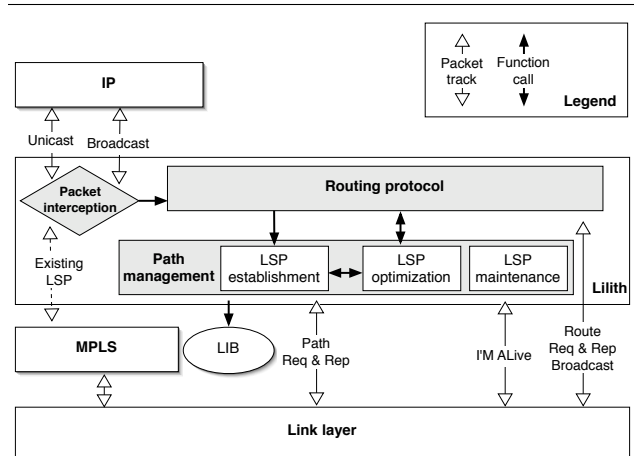


Figure 3. Architecture of Lilith

the LSP. Broadcast packets are passed to the Routing Protocol that uses the underlying route discovery mechanism (flooding) to propagate them to all hosts in the subnetwork.

*Routing Protocol module.* It implements the routing protocol, a reactive ad hoc protocol similar to AODV though less complex. Its purpose is to provide at least one path for a route, so we can establish a LSP to the destination. The protocol makes use of two messages: ROUTE REQUEST and ROUTE REPLY. When a route is needed, it generates a ROUTE REQUEST message containing a record of visited nodes. The request is flooded across the subnetwork and each intermediate node adds itself to the record. Once the request reaches the destination host, it generates a ROUTE REPLY that comes back on the reverse path. The destination host can reply to more than one request and thus initiate the construction of more than one LSP so that the requesting host may choose among several possible paths. The protocol uses BROADCAST message to propagate a broadcast or a scoped multicast to all nodes of the subnetwork. Our current implementation uses flooding for this purpose.

*LSP Establishment module.* The module uses a LSP establishment protocol based on two messages: PATH REQUEST and PATH REPLY. PATH REQUEST propagates to the destination that replies with PATH REPLY carrying the label bindings for LSP establishment. At each node on the reverse path the LSP Establishment module modifies the LIB (Label Information Base), the table used by MPLS for forwarding packets based on label switching. The LIB modification actually creates a LSP path at a given intermediate node or a host. In the current implementation the LSP establishment protocol is closely coupled with our reactive routing protocol—PATH REQUEST is included in ROUTE REQUEST message and PATH REPLY in ROUTE REPLY.

*LSP Maintenance module.* It acknowledges traffic received over established LSPs by periodically sending I'M ALIVE (IMA) message to all immediate neighbors using a link layer broadcast. The message carries a list of labels and the amount of data received on each active LSP during the last period of time. This information can then be used to decide if a given link is broken, in which case it switches to another path. If it is not the case, Lilith uses the information to estimate link quality, and metrics that can be injected into the routing protocol. If there is no traffic received for an outbound label during an interval, the corresponding entry in the LIB is deleted. Lilith detects existing long-lived TCP connections that do not send data for long periods of time and sends probe MPLS packets over the corresponding LSP to keep them active.

*LSP Optimization module.* It tries to find alternate routes for existing active destinations using the Routing Protocol module. This search is done in the background and does not interfere with the data flow over existing LSPs. The module evaluates the quality of candidate alternate routes and, if relevant, establishes paths using the LSP establishment module. A newly established LSP can either be used as a backup or become the primary LSP if it is significantly better.

## 4. Implementation

We have implemented Lilith as a user space daemon using the Linux version of MPLS for forwarding (we do not use any MPLS signaling protocols such as LDP). The interaction with the kernel needed for modifying the routing tables or controlling the MPLS layer takes place via well defined interfaces, so that the next version can be easily integrated within the kernel or ported to other platforms having sufficient support for intercepting packets.

The implementation of the packet interception module uses `netfilter` with `libipq` that enable user-space applications to get packets from the kernel and process them. If there is no route in the kernel routing table for the destination of a packet, Lilith stores the packet in a buffer and activates the routing protocol module to find a route. Lilith directly manipulates the LIB (Label Information Base) to establish or delete LSP paths when needed. The buffered packet is sent by the MPLS layer once a LSP path is established for the destination. The establishment of a LSP path creates an entry in the kernel routing table for the destination node so further packets to this destination are no more buffered in the packet interception module. The routing protocol and LSP establishment modules are implemented as application level protocols over UDP.

## 4.1. Experiments

To evaluate our prototype we have first measured communication performance in a wired network composed of four nodes connected via a 100Mb/s Ethernet hub (the channel is shared by all nodes) or switch (simultaneous transmissions on different ports are possible) as illustrated in Figure 4. A and D are hosts and B and C intermediate nodes. In the experiment, we wanted to evaluate the overhead of using Lilith with respect to standard IP forwarding. Using a wired network instead of a wireless LAN such as 802.11 enables us to eliminate some sources of performance inaccuracy—the behavior of the Ethernet hub is similar to that of a wireless shared medium LAN and we avoid taking into account wireless artifacts such as hidden receivers. Although interconnected at layer 2, we needed to make the platform behave like a multi-hop network, so we have set up filtering rules that allow only neighbor nodes to communicate with each other (dashed lines).

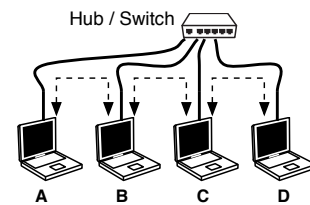


Figure 4. Setup for the first experiment

We have measured the round trip time and the throughput between hosts A and D by using `ping` and `netperf`. The measured cases are the following:

1. IP forwarding through a static route across four nodes: this case serves as a reference as it measures the expected performance when the interconnection is done at layer 3,
2. IP forwarding as in case 1, but now each packet passes through the user space daemon: this case allows us to compare the current implementation of Lilith with IP forwarding,
3. MPLS forwarding: we use Lilith to establish a LSP path and then disable Lilith to measure the raw performance of MPLS forwarding,
4. MPLS forwarding over a LSP established by Lilith, Lilith continues to work.

To understand what happens when Lilith is running (case 4), we give the sequence of events that occur when `ping` generates the ICMP Echo Request (similar events occur for the ICMP Echo Reply):

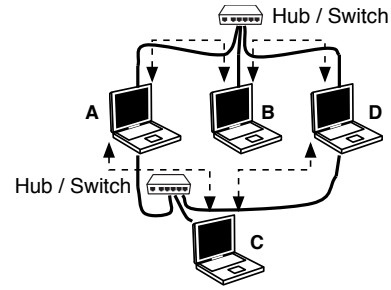
1. A intercepts the ICMP Echo Request and looks up the kernel routing table for an entry for D; there is no such entry, so the packet is buffered,
2. A floods the ROUTE REQUEST packet combined with a PATH REQUEST for D,
3. B receives the packet and floods it,
4. C receives the packet and floods it,
5. D receives the packet and starts establishing a LSP path by sending the ROUTE REPLY combined with the PATH REPLY to A,
6. the packet goes through C and B that establish the LSP by setting up entries in the LIB tables.
7. A establishes the LSP in a similar way, creates an entry in the kernel routing table for D and sends the buffered ICMP Echo Request.

Case	1	2	3	4
First ping (ms)				
ARP cache cleared, 64B packet size	0.9	1.0	0.8	6.0
Average ping (ms) 64B packet size	0.3	0.3	0.3	0.5
Average ping (ms) 1000B packet size	0.8	0.8	0.8	1.0
Throughput (Mb/s) Connection via a hub	22.6	22.5	22.5	22.3
Throughput (Mb/s) Connection via a switch	91.4	91.4	91.1	91.0

**Table 1. Performance comparison**

Table 1 presents the results of the measurements. We can observe an important overhead of the LSP establishment for the first ping (case 4), but then the difference in the round trip time is fairly small. At the beginning, Lilith has to find a route to the destination and establish a LSP path for both the ICMP Echo Request and Echo Reply. We can observe that the throughput is only slightly decreased: 1.3% when a hub is used and 0.4% for a switch. The overhead is due to the MPLS overhead, the kernel/user space copy for each packet at the source node, and I'M ALIVE packets sharing the links. The measurements confirm that the data transport over a LSP path has almost the same performance as standard IP forwarding.

The second experiment shows the advantage of using multiple routes. We consider the network illustrated in Figure 5. There are two routes between A and D: route 1 going through node B and route 2 going through node C. We run



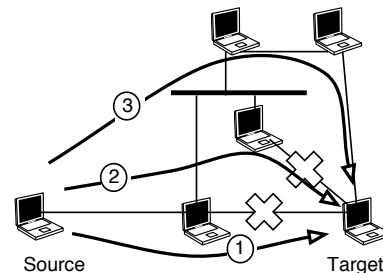
**Figure 5. Setup for the second experiment**

Route	Hub / Switch	Average RTT (ms)	RTT standard deviation (ms)
1	Switch	5.3	0.08
2	—	0.5	0.07
1	Hub	12.0	3.55
2	—	0.3	0.07

**Table 2. Performance over multiple routes**

netperf between A and D to generate greedy TCP traffic (A has always data to send) over route 1. We then measure the round trip time on both routes using ping. The measurement results are presented in Table 2.

We can observe an important impact of the TCP traffic generated by netperf on the round trip time for route 1, while there is no impact on the round trip time on route 2: the round trip time for route 1 is more than ten times longer than that of route 2. Our label switched architecture makes it possible to use a second route to communicate with a given destination, which is not possible in the standard IP forwarding. For example, we can send a high bandwidth, streaming traffic over route 1 and time sensitive traffic over route 2.



**Figure 6. Setup for the third experiment**

In the third experiment, we observe the dynamic behav-

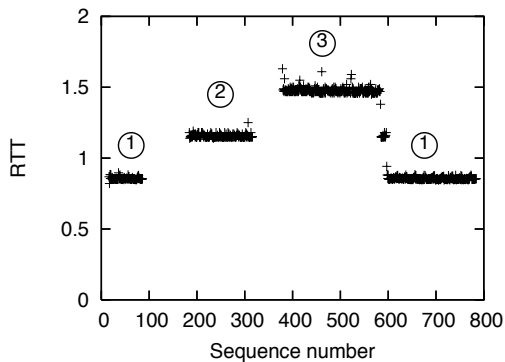


Figure 7. Measured round trip time

ior of Lilith in a setup with several possible routes (cf. Figure 6). The experiment illustrates how Lilith reacts to link failures and performs route optimization if necessary. Figure 7 presents the measured round trip time (in ms) during the experiment. At the beginning, the source node starts transmitting packets to the target node. Lilith queues the packets up while establishing a path so that the RTT of the first ones do not appear in the graph (their RTT is greater than the scale of the graph). Then we can observe a stable delay less than 1 ms over path 1. Afterwards, we cut the link on path 1 so that after three missing IMA messages<sup>1</sup> Lilith switches to path 2. Then we repeat the same manipulation with path 2. When this path also disappears, path 3 takes over the traffic until path 1 is operational again, in which case the optimization module reroutes the traffic. We can see that route optimization is not simultaneous on the route towards the target and back to the source—some intermediary RTT values can be observed between phase 3 and 4.

## 5. Conclusions and future work

When sensor networks and ubiquitous computing devices are interconnected and accessible from the global Internet, they naturally form a spontaneous edge network. Usually the communication and computing resources of such networks are limited so the interconnection architecture requires careful design. We have presented an original approach based on dynamic label switching. Our goal is to allow different communication paths on a per flow basis, provide seamless switching between operational and back-up paths, and make available information on destination reachability. By interconnecting all links at layer 2.5 so that they appear as one single IP subnet, we are also able to

<sup>1</sup> This number is configurable—it was 3 in our experiments and the frequency of sending IMA was 2 s.

get rid of administration burden—standard autoconfiguration protocols such as DHCP, router discovery in IPv4, IPv6 router and neighbor discovery, service discovery (mDNS, LLNMR, UPnP, SLP, JINI) can run as if they were on a single LAN.

Using on demand label switching is a first step towards handling QoS aspects. LSP paths can take into account some QoS parameters in route discovery, for instance preferring routes that use more wired hops or feature wireless links with better quality. The connection oriented nature of Lilith opens new possibilities for various types of QoS based routing. For instance we plan to investigate different algorithms for choosing the best routes based on QoS metrics.

Our first prototype shows very good performance compared with traditional IP forwarding. We observe only a slight throughput degradation when using Lilith, however we benefit from all the features provided by MPLS, for instance taking advantage of multiple paths for different traffic classes. In the current implementation, broadcasts may be not as efficient as they can be because we use simple flooding. We want to investigate other approaches for supporting broadcast (or scoped multicast).

## References

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: a Survey. *Computer Networks*, 38(4):393–422, 2002.
- [2] T. H. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, 2003.
- [3] W. S. Conner, J. Chhabra, M. Yarvis, and L. Krishnamurthy. Experimental evaluation of synchronization and topology control for in-building sensor network applications. In *Proc. WSN '03*, San Diego, September 2003.
- [4] L. Krishnamurthy, S. Conner, M. Yarvis, J. Chhabra, C. Ellison, C. Brabenac, and E. Tsui. Meeting the Demands of the Digital Home with High-Speed Multi-Hop Wireless Networks. *Intel Technology Journal*, 6(4), 2002.
- [5] IETF MANET Working Group, 2000. <http://www.ietf.org/html.charters/-manet-charter.html>.
- [6] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, 2003.
- [7] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, 2001.
- [8] V. Untz, M. Heusse, F. Rousseau, and A. Duda. On Demand Label Switching for Spontaneous Edge Networks. In *Proc. SIGCOMM Workshop on Future Directions in Network Architectures*, Portland, August 2004.
- [9] J. Weatherall and A. Jones. Ubiquitous Networks and their Applications. *IEEE Wireless Communications*, 9(1):18–29, February 2002.