# A case study in designing an autonomic wireless mesh network

Lee M.S. Bash, Christophe Jelger, Christian Tschudin
Computer Networks Research Group
University of Basel, Switzerland
Contact author: Christophe.Jelger@unibas.ch

*Abstract*— **Out-of-the-box meshing is the lead scenario for wireless mesh networks: End users should be able to join a mesh without any configuration hassle, they should be able to extend the mesh by simply adding more mesh routers without further configuration, and they should be able to instantly use network community services to network themselves in a spontaneous way.**

**We have integrated MANET style routing with a dynamic naming scheme and complete self-configuration, resulting in the `BaselMesh` system that runs on Linksys wireless routers. In this paper we report on the properties of the `BaselMesh` system, the offered functionality, and the obstacles that had to be overcome to make it an "autonomic" mesh.**

## I. Introduction

Wireless mesh networking [1][2] is a nurturing and commercially appealing application of multi-hop radio protocols and technologies. In contrast to mobile ad hoc networks (MANETs) which have no fixed infrastructure, wireless mesh networks typically consist of a quasi-static topology of wireless backbone devices via which mobile *client* (end-user) nodes can connect to the services (e.g., Internet access) provided by the mesh [3].

An appealing design option (when compared to MANETs) is that the client nodes do not necessarily need to run any mesh-specific protocols (e.g., routing) and may only use standard LAN-oriented paradigms such as default-route forwarding and DHCP. This scenario implies that only the wireless backbone devices need to run specialized protocols while client nodes remain unmodified, removing any burden on end users to install, configure, and run any dedicated software.

A key requirement and enabler for mesh networking is to minimize human configuration and maintenance as much as possible. Using terminology from IBM's autonomic computing initiative [4], a wireless mesh network should *self-configure* and *self-organize*. Following this trend, research has focused on protocols designed for auto-configuring part of the mesh backbone and for dynamically improving the performance of multi-hop radio forwarding [5]. However, existing research efforts usually consider very specific (and narrow) problems and do not aim at providing a fully integrated solution.

As an effort to partially fill this lack, this paper presents the design of the `BaselMesh` network, a self-configuring wireless mesh network that mimics an "Internet-like" communication service based on dynamic and distributed name resolution and asynchronous "email-style" messenging. We report on the properties of the system, the offered functionality, and provide some implementation details.

## II. The BaselMesh project

The `BaselMesh` project was initially developed by the Master Students of the "Autonomic Computer Systems" course at the University of Basel in Winter 2006/2007. The goal of the project was to release a fully self-configuring system that could be used "out-of-the-box" by non-specialized users to setup a spontaneous wireless mesh backbone network. A key challenge was to use and tightly integrate as much existing software as possible and to minimize the use of specialized software[1]. `BaselMesh` was developed on the popular WRT54G Linksys devices with the `OpenWRT`[2] firmware. The `BaselMesh` software is currently shipped as a custom firmware image that can easily be used to re-flash a router in less than a minute. With the only exception of configuring the device with a unique name, this is all that is needed to configure a `BaselMesh` router.

## III. Network architecture

### A. Topology

The `BaselMesh` network is composed of two types of nodes: mesh routers which form the 'backbone' of

[1]We here refer to protocols specifically developed for MANETs.
[2]See http://www.openwrt.org

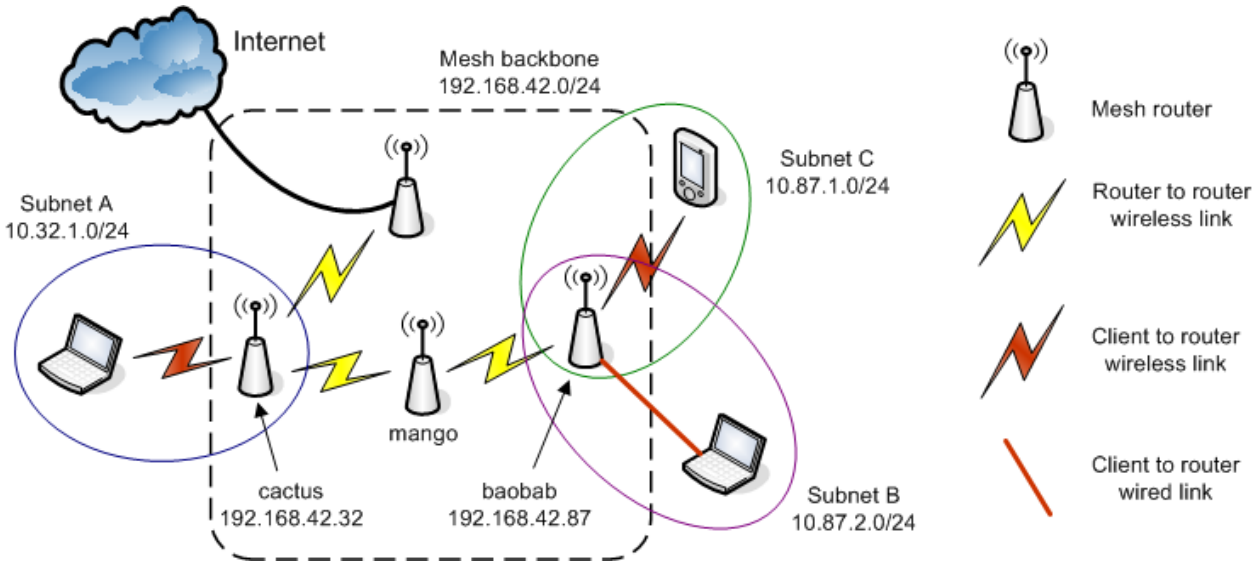


Fig. 1. Mesh architecture

the network and support the operation of the mesh, and mesh clients which are standard computer devices solely running legacy networking protocols (e.g., DHCP and DNS) and technologies (e.g., 802.3 and 802.11). The current "ownership" model is very simple: either all the mesh routers belong to some entity (organisation or person) or they all belong to different entities. The first model relates to usage cases where e.g. a University or city wants to have full control over the entire mesh, while the second model relates to usage cases where e.g. individuals together build a mesh network but do not sufficiently trust eachother to delegate management to one (or a group of) person(s).

A client typically connects to the mesh via one of the mesh routers which, as a traditional infrastructure-based 802.11 access point router, appears to the client as a DHCP and DNS server or proxy, default gateway, NAT box, and firewall. The link connecting a mesh client and a mesh router can be wired or wireless. Note that in contrast to infrastructure-based networking which is based on the 802.11 `managed` mode of operation, the wireless transceiver of a mesh client must be configured in 802.11 `ad hoc` mode if the mesh router it is *attached to* has only one wireless NIC (as for the Linksys WRT54G). In that case, the same transceiver is indeed used by a mesh router to communicate both with other peer routers and with client nodes. Figure 1 illustrates a possible architecture with 4 mesh routers and 3 client nodes.

### B. Subnet model

When fully (self-)configured, the `Baselmesh` network is organized into multiple IP subnets as shown on Fig. 1. Routing among mesh routers is performed by the LUNAR [6] protocol which operates below IP in an underlay at layer 2.5 and provides a subnet illusion: for the routers, the multi-hop mesh appears like a single IP subnet where all routers can be reached via one IP hop. This networking model resembles the wired equivalent of Ethernet bridging where multiple layer 2 segments form a single IP subnet. As detailed in [7], each instance of the LUNAR protocol can configure itself with a unique[3] IP address in the pre-configured `192.168.42.0/24` subnet.

On Fig. 1, the LUNAR instances of the mesh routers `cactus` and `baobab` respectively self-configure host ids `32` and `87` in the `192.168.42.0/24` LUNAR subnet. For each mesh router, the LUNAR-configured host id is then used to automatically setup two additional subnets for client nodes attaching to the router via either its wired or wireless interface. For example on Fig. 1, the router `baobab` with id `87` automatically configures the subnet `10.87.1.0/24` for its wireless clients and the subnet `10.87.2.0/24` for its wired clients. For a mesh client, routing is achieved via a default-route towards the mesh router it is attached to (e.g., towards `10.87.2.1` for the client in subnet B). For mesh routers and thanks to the subnet illusion created by LUNAR, routing to client subnets `10.x.0.0/16` is

[3]with respect to other LUNAR nodes.

simply configured via the corresponding mesh router with the address `192.168.42.x`. Multi-hop paths inside the backbone mesh are dynamically established in the underlay by LUNAR as detailed in [6] and [7].

### C. Name resolution

With today's predominance of the Internet, DNS names and their derivatives (e.g., email addresses, URLs, SIP addresses) are the primary means via which users initiate communications over the Internet. One cannot seriously design a network that would not support DNS name resolution because end users are relunctant (or technically unable) to use IP addresses. Compared to the administratively managed Internet, the challenge in a constantly changing mesh environment is to dynamically configure and update DNS name resolution in a decentralized manner.

In `BaselMesh`, name resolution is partially achieved by introducing a virtual namespace (`*.net.lunar`) where the instances of the LUNAR protocol mimic the operation of DNS name resolution (see [7] for details). In particular, this allows LUNAR instances to resolve and setup paths with names (e.g. `baobab.net.lunar`) instead of IP addresses. However, this feature only allows name resolution among LUNAR instances inside the mesh backbone. In order to provide name resolution of client nodes, the `BaselMesh` system extends the LUNAR-based scheme by dynamically configuring subdomains of `*.net.lunar`.

For example on Fig. 1, each mesh router is automatically configured such that DNS queries of the form `*.baobab.net.lunar` are forwarded to the mesh router `baobab` with address `192.168.42.87`. In practice, each LUNAR instance periodically advertises its presence (and name) to other LUNAR nodes: this information is used to maintain a "yellow-pages" database of all active LUNAR nodes (again see [7] for details). This database is then used by each mesh router to dynamically setup appropriate entries in the local configuration file of the DNS server running on each router. Note that the entire procedure is automatic and requires no human intervention.

### D. Client autoconfiguration

As briefly introduced in section III-A, each mesh router appears to all the clients attached to it as a DHCP and DNS server or proxy, default gateway, NAT box, and firewall. In particular, this means that an end user (as in a LAN environment) can fully rely on DHCP to autoconfigure the IP address and netmask, default route, DNS server address and default domain name of his/her computer device. The only exception is that when using a wireless connection, the user also has to configure the wireless card to operate in 802.11 ad hoc mode and to use the SSID "BaselMesh".

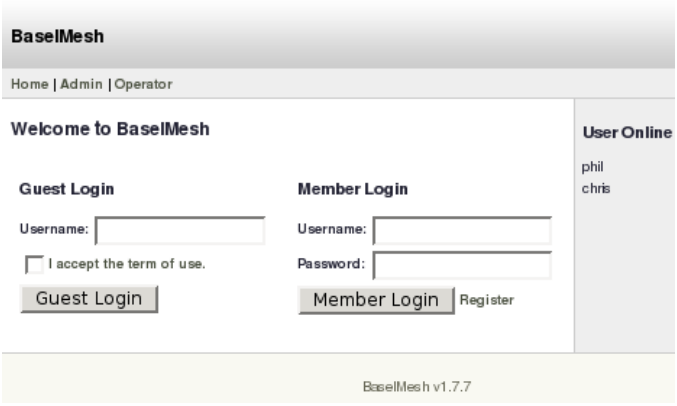


Fig. 2. BaselMesh registration and login.

## IV. Services

In addition to providing self-configuring properties, `BaselMesh` also offers an intra-mesh communication service in order to allow users to share information and form *communities*, two features considered essential in mesh networking [3][2]. To our knowledge, very little research has focused on protocols and applications to support intra-mesh communication services and `BaselMesh` is partially filling this lack.

While connecting for the first time to the `BaselMesh` network, each user is asked to register a username and password via a web-based interface to which any HTTP or HTTPS request is redirected by default. Once registered and logged in, the HTTP redirection is removed and users can potentially browse the web if the mesh is attached to the Internet. The registration and login interface is shown in Fig. 2.

The login procedure is used to support a web-based messenging system (illustrated by Fig. 3) that allows registered users to send and retrieve messages in an asynchronous manner (as for standard emails). Sent messages waiting to be retrieved are stored in a DHT system: upon login, each user is informed whether he/she has received messages waiting to be read. In the near future, we hope to extend this initial feature with file sharing and instant messenger functionalities. Typically, end users can learn the names of other registered users via the standard web-based interface that is used for login and logoff operations and to send/retrieve messages. Also note that


Fig. 3. Reading a message.

a restricted guest access is also supported although in that case the messenging service is not useable.

As an extra feature, the web-based interface also includes a management console that can be used by an authorised (super-)user to blacklist some usernames and MAC addresses. It also gives access to some status information for each mesh router. This feature can be very useful if all the mesh routers are owned by a single entity (as could be the case for example in areas not covered by ADSL or cable operators).

## V. Implementation details

As described earlier, access to the mesh is granted after some basic authentication via a username and a password. Before successful authentication, all HTTP or HTTPS requests are redirected to the login webpage and packets sent to all other ports are dropped (except SSH and DHCP packets sent by a client node to the mesh router it is attached to). To perform this basic filtering we use the MAC addresses of mesh clients and Linux's `iptables` functionality.

The `BaselMesh` system also implements a distributed hash table (DHT) storage system based on the well known CHORD framework [8]. The DHT system is used to store various types of information inside the mesh in a distributed manner like e.g., intra-mesh messages, usernames and passwords, and session information (e.g. MAC addresses). Passwords are encrypted with the MD5 cipher before being transmitted and stored in the DHT.

In addition to the DNS emulation scheme implemented inside LUNAR, name resolution is based on the standard and very flexible `dnsmasq` DNS software. In particular, we use `dnsmasq` to specify which mesh router is responsible for a given subdomain (e.g., `baobab.net.lunar`) and to locally store in the corresponding mesh router the `<name,IP>` tuples of client nodes (e.g., `<phil.baobab.net.lunar,10.87.1.148>`).

Finally, `BaselMesh` routers can also detect and autoconfigure Internet gateways. Note that custom-built mechanisms ensure that a gateway is operational before adding a default route towards this gateway. This for example prevents nodes from configuring a default route towards a faulty gateway. Gateway nodes are also responsible for performing network address translation between the mesh network and the "outside" world and also serve as proxies for the Internet global DNS.

As mentioned in section II, the `BaselMesh` software is a customised distribution of the `OpenWRT` firmware. In addition to the extra LUNAR kernel module and the DHT implementation both written in C, the software contains html and cgi code for the web interfaces and the associated functions (e.g., login, send/recv messages, etc), and about two dozen elaborated shell and perl scripts which set and dynamically update various configuration parameters and files. These scripts are either called periodically by cron jobs or triggered by end user actions and network events.

To test and validate the operation of our mesh software, we have successfully deployed up to about a dozen Linksys mesh routers with a maximum path length of 3 hops. The experience gained during this very practical project will be used to define a new development phase that will focus more on *self-otimization* and *self-healing* properties in order to increase the robustness of the mesh in face of network failures and attacks.

## VI. Limitations

Despite its advanced self-configuration properties, the current version of the `BaselMesh` system still has a number of limitations, the main one being that each mesh router still needs to be manually configured with a unique name. While we have solved this problem with a more recent LUNAR prototype (see [9] for details), our new scheme based on NAT and keyword-derived names has not yet been integrated into `BaselMesh`. The main bridle here is that our new scheme is based on a "home-made" NAT module which is not fully compatible with applications that embed IP addresses and port numbers

in their payload. This problem will be solved in the next few months by interfacing the new version of LUNAR with the standard NAT module of the Linux kernel.

Another limitation of `BaselMesh` is that client roaming accross different mesh routers is not yet supported. This is mainly due to the fact that layer 2 roaming is not supported in 802.11 ad hoc mode so nodes movements have to be handled by higher layers. It is not clear yet how one can handle this problem without introducing a high control overhead as e.g., with periodic polling or heartbeat protocols. This problem clearly needs further work and research in order to find a suitable solution.

## VII. Conclusions and future work

In this paper we have presented the `BaselMesh` project, an "autonomic" mesh networking implementation that allows non specialized users to easily setup and deploy a wireless mesh network with very little overhead (i.e., assign unique name and change root password of mesh router). The autonomic features of `BaselMesh` include the dynamic autoconfiguration of IP addresses, IP subnets, routing entries, and name resolution servers. Note that the self-configuration of `BaselMesh` is achieved in a fully distributed way and dynamically supports the addition or removal of mesh routers and clients. The system also provides basic authentication mechanisms and a simple asynchronous messenging system.

As future work, we would like to integrate a new verson of LUNAR (detailed in [9]) which removes the burden of having to manually configure unique names for mesh routers. We would also like to extend the system to gracefully support the roaming of mesh clients in a secured manner. Finally, we would like to implement and integrate a web-based and real-time messenger application.

This first version of the `BaselMesh` software is expected to be publicly released in Fall 2007 when the documentation is complete and the final software validation is performed. It will be available via the `http://cn.cs.unibas.ch` website.

## VIII. Acknowledgements

`Lee M.S. Bash`, the first author of this paper, is not a real person. The `BaselMesh` project was developed by the Master Students (M.S.) of the "Autonomic Computer Systems" course at the University of Basel. Most of the credits goes to them: it was however impossible to mention all their names but fortunately someone once invented anagrams.

## References

[1] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," in *Proceedings of ACM Mobicom 2004*, September 2004, Philadelphia, PA, USA.

[2] V. Bahl (organizer), "Wireless Community Mesh Networks - Hype or the Next Big Frontier?" in *Panel Discussion at ACM Mobicom 2004*, September 2004, Philadelphia, PA, USA.

[3] R. Bruno, M. Conti, and E. Gregori, "Mesh Networks: Commodity Multihop Ad Hoc Networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 123–131, March 2005.

[4] J. Kephart and D. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, January 2003.

[5] I. Akyildiz and X. Wang, "A Survey on Wireless Mesh Networks," *IEEE Communications Magazine*, vol. 43, no. 9, pp. S23–S30, September 2005.

[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. Jelger and C. Tschudin, "Model Based Protocol Fusion for MANET-Internet Integration," in *Proceedings of WONS'06*, January 2006, Les Ménuires, France.

[8] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications," in *Proceedings of ACM SIGCOMM 2001*, August 2001, pp. 149–160, San Diego, CA, USA.

[9] C. Jelger and C. Tschudin, "Dynamic Names and Private Address Maps: Complete Self-Configuration for MANETs," in *Proceedings of CoNEXT 2006*, December 2006, Lisboa, Portugal.

Fig. 4. Linksys router ready to be flashed.