Experimental research on testbeds for community networks (year 1)

Deliverable/Milestone: D4.1

Date: Nov 12 2012

Version: 1.0
Abstract

This document describes the experimental research on the testbed for community networks, until M12.

First the research roadmap and recommendations are outlined, describing the development of the research roadmap for CONFINE based on a requirements analysis based on prototype experiments.

Then research on obstacles and limitations of Community Networks is described, focused on current routing protocols and followed by a study of two existing community network routing protocols: OLSR and BMX.

This is followed by an overview of the design, implementation and standardisation of the radio-to-router communication protocol DLEP, and an overview of the current activities with respect to self-management.

Finally, results of a prototype experiment execution are given, followed by an overview of on-going research with respect to best practices for experimentation and initial open data set efforts.
Executive Summary

This deliverable details the current status of work package 4 of the CONFINE project. It gives an overview of our work to identify a research roadmap, as well as the encountered obstacles and limitations of community networks. It details the efforts on cross-layer interactions, self-management and best practices for experimentation and open data sets.

The deliverable shows the positive progress of this work package, with all progress on track and the open data set research even slightly ahead of planning.
# Table of Contents

1. RESEARCH ROADMAP AND RECOMMENDATIONS ........................................................................................................... 1
   1.1. IDENTIFICATION OF THE RESEARCH ROADMAP ................................................................................................. 1
   1.2. RECOMMENDATIONS FOR THE TESTBED DESIGN AND RESEARCH ROADMAP ............................................. 6
   1.3. CONSIDERED HARDWARE AND SOFTWARE ......................................................................................................... 7

2. RESEARCH ON OBSTACLES AND LIMITATIONS OF COMMUNITY NETWORKS .................................................... 8
   2.1. NETWORKING .......................................................................................................................................................... 8
   2.2. OLSRV2 DEVELOPMENT .................................................................................................................................... 11
   2.3. BMX6 EVALUATION IN COMMUNITY NETWORKS ................................................................................................. 13

3. CROSS-LAYER INTERACTIONS AND OPTIMIZATIONS ............................................................................................... 17
   3.1. INTRODUCTION ....................................................................................................................................................... 17
   3.2. RADIO TO ROUTER COMMUNICATION PROTOCOLS ............................................................................................. 17

4. SELF-MANAGEMENT .................................................................................................................................................. 22

5. BEST PRACTICES FOR EXPERIMENTATION ........................................................................................................... 23
   5.1. A PROTOTYPE BITTORRENT EXPERIMENT ......................................................................................................... 23
   5.2. OVERHEAD ANALYSIS OF EMBEDDED WIRELESS TESTBEDS WITH OMF ................................................. 23
   5.3. DEVELOPMENT OF A TEST ENVIRONMENT FOR NETWORK MEASUREMENT TOOLS BASED ON OMF .................................................................................................................................. 27

6. OPEN DATA SETS ......................................................................................................................................................... 41
   6.1. DATA SET TYPES .................................................................................................................................................... 41
   6.2. RESEARCH ON THE TOPOLOGY OF COMMUNITY NETWORKS ......................................................................... 42

7. CONCLUSIONS ............................................................................................................................................................ 52

8. REFERENCES ................................................................................................................................................................ 53

9. APPENDIX ................................................................................................................................................................... 56
   9.1. PROTOTYPE EXPERIMENTS .................................................................................................................................... 56

---

# Figures

FIGURE 2.1: STRUCTURE OF FRAMEWORK CODE DECOMPOSED INTO FIVE LIBRARIES ........................................ 11
FIGURE 2.2: BMX6 AND OLSR OVERHEAD ON NETWORK START UP ............................................................... 14
FIGURE 2.3: MEASUREMENT OF CONVERGENCE TIME ......................................................................................... 14
FIGURE 2.4: EXPERIMENT CYCLE ............................................................................................................................... 15
FIGURE 2.5: OVERHEAD RESULTS ............................................................................................................................ 16
FIGURE 2.6: CONVERGENCE TIME RESULTS ........................................................................................................... 16
FIGURE 3.1: IMPROVED NODE DESIGN USING RADIO TO ROUTER COMMUNICATION PROTOCOLS 17
FIGURE 5.1: DETAILED OVERVIEW OF OBSERVED THROUGHPUT .................................................................................. 26
FIGURE 5.2: PATH-PERSISTENT CROSS TRAFFIC SCENARIO ......................................................... 38
FIGURE 5.3: HOP-PERSISTENT CROSS TRAFFIC SCENARIO .................................................. 39
FIGURE 6.1: CNML TREE ......................................................................................................... 44
FIGURE 6.2: ZONES CONSIDERED IN THE EVALUATION ....................................................... 44
FIGURE 6.3: CNML ELEMENTS' ATTRIBUTES ........................................................................ 44
FIGURE 6.4: SUMMARY OF ZONE GRAPHS. FOR NODE DEGREE IT IS GIVEN MIN/MEMAN/MAX. FOR THE RANK AND HOPS IT IS GIVEN EXPONENT/SAMPLE CORRELATION COEFFICIENT (P) .................................................. 46
FIGURE 6.5: BASE GRAPH OF CATALUNYA ZONE. AXIS ARE IN KM ........................................ 47
FIGURE 6.6: CORE GRAPH OF CATALUNYA ZONE. AXIS ARE IN KM ........................................ 47
FIGURE 6.7: RANK LOG10-LOG10 PLOT OF BASE GRAPH OF CATALUNYA ZONE ....................... 47
FIGURE 6.8: RANK LOG10-LOG10 PLOT OF CORE GRAPH OF CATALUNYA ZONE ..................... 47
FIGURE 6.9: HOPS COUNT LOG10-LOG10 PLOT OF BASE GRAPH OF CATALUNYA ZONE .......... 48
FIGURE 6.10: HOPS COUNT LOG10-LOG10 PLOT OF CORE GRAPH OF CATALUNYA ZONE ............ 48
FIGURE 6.11: CATALUNYA ZONE CORE GRAPH: HIDDEN TERMINALS CECDF ......................... 48
FIGURE 6.12: CATALUNYA ZONE CORE GRAPH: AVERAGE NUMBER OF HIDDEN-TERMINALS VS CORE-DEGREE .................................................................................................................. 48
FIGURE 6.14: ECDF OF THE HIDDEN TERMINALS' MEAN OF THE LEAF ZONES ......................... 50
FIGURE 6.15: LINK LENGTH DISTRIBUTION OF CATALUNYA ZONE ........................................ 51
FIGURE 6.16: LINK LENGTH DISTRIBUTION FITTING (M1, M2 AND M3 ARE IN KM) ...................... 51

Tables

TABLE 1: SURVEY OF TYPICAL NETWORK MEASUREMENT AND MANAGEMENT TOOLS ............ 29
TABLE 2: OVERVIEW OF NETWORK MANAGEMENT TOOLS THAT HAVE SO FAR BEEN INTEGRATED WITH THE OMF FRAMEWORK ........................................................................................................ 30
1. RESEARCH ROADMAP AND RECOMMENDATIONS

The main objectives of the CONFINE work package 4 are:

- to perform research on specific obstacles for setting up or extending a large-scale testbed or large-scale systems based on the testbed model,
- to provide specific guidance and support for the experiments selected in open calls,
- to offer researchers a number of open data sets which they can use for experimentation,
- to develop a framework which allows benchmarking components studied in the experiments, and
- to provide documentation which easily shows the type of experiments and how they can be made on the experimental facility.

In order to address these objectives, a systematic approach has been chosen in order to give recommendations on the testbed design from the perspective of the research that will be performed in WP4. In addition, a work plan has been set up that makes extensive use of potential synergies between the research topics addressing the obstacles regarding the community node design and the design of the research node and thus supporting WP2. In this context, this section also includes an overview of considered hardware and software. More extensive information is available in the deliverables from WP2, here we include the research which motivated the choices.

The results of WP4 will contribute to WP2 with improved algorithms to be integrated in the system software and services in the enhancements of the testbed system. The work on radio-to-router communication protocols has already been integrated into the first year version of the research node. In addition, the WP4 results will also influence the long-term sustainability of community networks, as it will decrease maintenance requirements, and for that it will be an input for WP5.

1.1. IDENTIFICATION OF THE RESEARCH ROADMAP

Starting a testbed is a difficult task, which requires focus. A testbed cannot support any experiment a researcher can imagine, it requires a focus testbed design.

As such, CONFINE decided to identify a number of prototype experiments. They served multiple goals. First, to focus the development of the testbed, to avoid a too general testbed which would not be a good fit for the CONFINE case. Another important goal was facilitating parallel development of multiple aspects of the testbed. While some partners developed the software running on the testbed nodes, others developed the software architecture and others designed potential experiments to run on the testbed, which they served as a source of inspiration.

To abstract away from the prototype experiments, and to avoid a too specific testbed, a classification was then introduced. The prototype experiments were classified according to multiple criteria, and not just one. This was a good reminder to everyone that experiments can be very diverse, but more importantly it also helped to identify experiments the project wants to support and experiments which we do not want to support or even cannot support. E.g., one important classification criterion was the impact on the community network. The larger the possible impact, the more strict the CONFINE software must be to isolate experimental impact from the real network.
As a first step, a classification schema has been defined (see section 1.1.1) to categorise potential experiments to be carried out in the testbed (see sections 1.1.2).

1.1.1. EXPERIMENT CLASSIFICATION SCHEMA

In cooperation with work package 2, an important aspect regarding the categorization of experiments refers to the maximum impact of an experiment on the operation of community networks. This is an important measure, as the community networks are production networks with actual users. Impacts of the experiments may have a direct influence on the Quality of Experience provided to the members of the network.

**Impact on community networks and their users:**

Four categories regarding the potential impact on community networks and users were defined:

1) **Passive, historic**: Experiments interested in only historic data extracted from the network (topology over time, aggregated traffic, aggregated load, sample traffic traces, routing logs, routing tables, etc.) are categorized as “passive, historic”. These passive/off-line experiments will require an extra effort to collect data either during the experiment e.g. on a periodic basis or after the experiment is completed. Traces, logs, data might be anonymized before being made public.

   Real world example: On Incentives in Global Wireless Communities [BIC2009] (testbed_mapping like but based on public CONFINE traces).

   Testbed requirement: Means to extract info (logs) are provided e.g. by the OMF integration. However, these mechanisms are not necessarily be open to any experimenter. Only a trusted researcher that follows specified pre-processing rules before data is made public should be granted access to these mechanisms.

2) **Active, good “netizens”**: An experiment that cannot be distinguished from the normal usage of community networks is classified as “active, good netizens”. It uses existing interfaces into the community networks, but does not interfere with them. Examples are network performance experiments where a set of nodes makes periodic measurements to specific hosts with minimal traffic, similar to http://projectbismark.net/.

   Testbed requirements: Means to deploy software packages/component, either initiated by the node owner or by the experimenter (node leaser) have to be provided.

3) **Active, potentially disturbing**: An experiment that is beyond normal usage of the community network and that has the potential to disturb production networks is classified as “active, potentially disturbing”. Potential reasons might be that the experiment affects the lower levels of communication (e.g. channel assignments, address assignments) or generates traffic beyond normal limits (e.g. a stress test affecting parts of the network). Protection mechanisms may be required (e.g. traffic or CPU resource limits, consider mechanisms of Planetlab nodes).

   Real world examples: http://coralcdn.org/

   Testbed requirements: Similar to 2) with more control regarding the experiments (react to disturbance) ...

4) **Active, conflicting**: An experiment that needs access to third-party traffic (e.g. an experiment requiring access to data payload/deep-packet-inspection, affecting personal information), or affects key services (e.g. experimentation with DNS, routing tables, address assignments) is classified as “active, conflicting”. Isolation (e.g. network virtualization/isolation such as a VPN), filtering (e.g. an anonymizing intermediary) and containment mechanisms (e.g. any mechanism to protect/isolate the net from the experiment) may be required.
Real world examples: A stress test of a new/modified ad-hoc routing protocols such as OLSRD.

Testbed requirements: Similar to 3) + experiment isolation.

In addition to the impact, we defined the following categories that have an impact on the design of the testbed:

**Lowest impacted layer of the protocol stack:**

*Following the ISO/OSI model, which is the lowest layer impacted by this experiment?*

To define impact, consider the link to the node: will it see special L3 packets? New L2 MAC protocols? A different physical layer? A special L4 protocol? A new L7 application?

**Lowest accessed layer of the protocol stack:**

*Following the ISO/OSI model, which is the lowest layer access by this experiment? I.e., which is the lowest layer where information is read from, rather than modified.*

To define this, consider which information will be read and not changed. E.g. in cross-layer experimentation, information will frequently be used but only seldom will it be changed.

**CPU power and memory footprint:**

*Which is the scale of required CPU power? Will this be an experiment requiring cluster class CPU power? Or is an embedded device sufficient?*

Suggested replies: cluster, desktop, (multimedia) server, embedded device

**Bandwidth profile:**

*How much bandwidth will be required over the run of the experiment? Will this experiment be bandwidth intensive or rather use almost no bandwidth?*

Suggested replies: bandwidth intensive (try to specify peak rates or dependency), low bandwidth

The five categories will be the basis for classification of the experiments described in section 1.1.2

1.1.2. SELECTION OF CANDIDATES FOR PROTOTYPE EXPERIMENTS

The following table gives a summary of the potential experiments that were taken into account as candidates for WP4 and classified according to their requirements.

In appendix 9.1 all experiments are described in detail.
The experiments described in the previous section have been categorized according to the criteria defined in section 1.1.1 as shown in this table.

<table>
<thead>
<tr>
<th>Experiment/category</th>
<th>Impact</th>
<th>Lowest impacted layer</th>
<th>Lowest accessed layer</th>
<th>CPU power and memory footprint</th>
<th>Bandwidth profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlay / Peer-to-peer Networking</td>
<td>2 - Active, good “netizens”</td>
<td>7 – Application</td>
<td>3 – Network</td>
<td>Server/Desktop</td>
<td>Potentially bandwidth intensive (depends on user)</td>
</tr>
<tr>
<td>Application Usability Evaluation</td>
<td>2 - Active, good “netizens”</td>
<td>7 – Application</td>
<td>7 – Application</td>
<td>Embedded/Server/Desktop</td>
<td>Potentially bandwidth intensive (depends on application)</td>
</tr>
<tr>
<td>Application Optimization</td>
<td>2 - Active, good “netizens”</td>
<td>7 – Application</td>
<td>3 – Network</td>
<td>Server/Desktop</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td>Delay Tolerant Networking</td>
<td>2 - Active, good “netizens”</td>
<td>7 – Application</td>
<td>3 – Network</td>
<td>Server/Desktop</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td>Content Distribution</td>
<td>3 - Active, potentially disturbing</td>
<td>7 – Application</td>
<td>3 – Network</td>
<td>Server/Desktop</td>
<td>Potentially bandwidth intensive (depends on application)</td>
</tr>
<tr>
<td>Transport-Layer Optimization</td>
<td>3 - Active, potentially disturbing</td>
<td>4 – Transport</td>
<td>3 – Network</td>
<td>Embedded/Server/Desktop</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td>Hybrid routing in mesh networks</td>
<td>4 - Active, conflicting</td>
<td>3 – Network (custom protocol)</td>
<td>3 – Network (routing tables)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td>Multi-Topology Routing</td>
<td>4 - Active, conflicting</td>
<td>3 – Network (custom protocol)</td>
<td>3 – Network (routing tables)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td>Routing Security</td>
<td>4 - Active, conflicting</td>
<td>3 – Network (custom protocol)</td>
<td>3 – Network (routing tables)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td>Link layer feedback to routing protocols</td>
<td>4 - Active, conflicting</td>
<td>3 – Network (custom protocol)</td>
<td>2 – Link (MAC information required)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td>Category</td>
<td>Level</td>
<td>Description</td>
<td>Metrics</td>
<td>Device</td>
<td>Bandwidth</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Mesh Routing Scalability</td>
<td>4</td>
<td>Active, conflicting</td>
<td>3 – Network (custom protocol)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 – Link (link quality information)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross Layer Routing Metrics</td>
<td>4</td>
<td>Active, conflicting</td>
<td>3 – Network (custom protocol)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 – Link (link quality information)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routing Protocol Auto Configuration</td>
<td>4</td>
<td>Active, conflicting</td>
<td>3 – Network (custom configuration)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 – Link</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Separation of router and radio device</td>
<td>4</td>
<td>Active, conflicting</td>
<td>3 – Network</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 – Link (link quality information)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Router Power Consumption</td>
<td>1</td>
<td>Passive (or higher)</td>
<td>1 – Physical (measurement HW)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 – Network</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy enforced spectrum sharing</td>
<td>3</td>
<td>Active, potentially disturbing</td>
<td>1 – Physical (channel switch)</td>
<td>Embedded device</td>
<td>Low bandwidth</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 – Link (link occupation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active Network Measurements</td>
<td>3</td>
<td>Active, potentially disturbing</td>
<td>3 – Network</td>
<td>Server/Desktop</td>
<td>Potentially bandwidth intensive (depends on measurements)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 – Network</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passive Network Measurements</td>
<td>1</td>
<td>Passive</td>
<td>3 – Network</td>
<td>Server/Desktop</td>
<td>Potentially bandwidth intensive (depends on measurements)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 – Network</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1.2. RECOMMENDATIONS FOR THE TESTBED DESIGN AND RESEARCH ROADMAP

Experiments with a high impact on the testbed either need to be carried out in a separated test network or with special means of isolation. These requirements are addressed in deliverable D2.1.

Another important observation is that many routing experiments, while needing cross-layer access to link layer information (lowest accessed layer), actually only operate on layer 3 and higher. Today, access to link layer status information in Linux-based systems is often provided by Netlink sockets. This mechanism requires raw access to the WiFi interfaces. However, radio-to-router communication protocols (see experiment: “Separation of router and radio device”) can decouple the Netlink access with access to the interface device from the experiments accessing these information. Thus, with an appropriate radio-to-router communication protocol, the experiments do not need raw interface access (see D2.1) which has privacy issues. Isolated interface access in addition to a radio-to-router communication service at the research device is sufficient. In addition, it allows for a research node configuration that does not require all research WiFi interfaces to be plugged into the research node itself. Instead, small and cheap radio devices connected to the research node via LAN can be used in conjunction with a powerful research device. Thus it allows for a cheaper, more flexible but at the same time more powerful node architecture. For later revisions, it is also planned to include remote configuration features into the protocol, allowing for a policy based control of the research device by the control software and/or the researcher. Besides the advantages for the research node architecture, a radio-to-router communication protocol also allows for a revolutionary way of node design for community networking. Additional information on the concepts and the improvements for community networks are presented in section 3.2 and in the paper [BR2012].

A third observation is that in general, application experiments require a more powerful research device than lower layer experiments. The opportunities provided by a radio-to-router communication protocol (e.g. the physical separation of the router from the radio hardware) allow for more flexible CONFINE nodes (also see section 3.2)

Thus the development of a radio-to-router communication protocol is a high priority and was one of the first developments addressed. Our implementation shares the same code basis with our current NHDP [RFC-6130] and future OLSRv2 [OLSRv2-17] implementations for embedded devices (see section 2.2). Thus, we were able to exploit synergy effects between both developments and thus laying the basis for all enhancements to PacketBB [RFC-5444] based protocols.

Finally, note the very broad scope of experiments, especially from an OSI point of view. CONFINE will have to support a large number of experiments, preferably in a user-friendly way. Therefore, the partners have opted to develop OMF integration in the testbed. OMF is a widely known and widely used testbed framework, which allows for portability of experiments over testbeds. In some case studies of federated testbeds, OMF was considered to play an important role to enable experiments. CONFINE integrates OMF, to prepare the work on best practices of experimentation. Moreover, CONFINE will also develop a test-environment for network measurement tools based on OMF. The latter will become a tool to generate open data sets, as required by task T4.3.1, and a foundation for the development of a benchmarking framework, as described by task T4.3.3. This also allows for the integration of the radio-to-router communication protocol DLEP with OMF, which is currently work under progress, to provide information for the open data set generation.
1.3. CONSIDERED HARDWARE AND SOFTWARE

Deciding on which hardware to deploy in CONFINE was not straightforward. We faced serious challenges, challenges posed by the requirements of researcher on the one hand the limitations imposed by the deployment inside the community networks on the other hand. The hardware choice is discussed in deliverable D2.1.

With respect to software, the CONFINE testbed runs on OpenWRT, a Linux distribution for embedded devices. The motivation of this choice and alternative options are discussed in deliverable D2.1.
2. RESEARCH ON OBSTACLES AND LIMITATIONS OF COMMUNITY NETWORKS

2.1. NETWORKING

Although the community networks involved in CONFINE of course are fully functional, they have already experienced quite some obstacles and limitations in their network. This proved to be interesting information for the project, and is an on-going area of research.

2.1.1. ROUTING PROTOCOLS

At the moment, the three community networks each have different routing protocols in place. In this section, we will describe the routing protocols in the networks, together with advantages and disadvantages experienced by the networks.

2.1.1.1. AWMN

AWMN currently uses BGP. There have been a lot of iterations until the current state and choice of protocol for routing within the community. As the network was growing the community’s routing protocol iterated from static to, RIP, OSPF, BGP and finally set in the current schema which is BGP with certain local confederations or unifies Autonomous Systems (AS) that utilize other internal intra domain routing protocols (IGP) that are more dynamic and flexible such as the OLSR, BATMAN, BMX and other popular meshing protocols.

The main routing protocol is BGP and all backbone nodes (nodes with more than two point to point links) are assigned an AS number based on the NodeID number they are given by the community Node Database WiND (wind.cube.gr). The choice of this protocol was on the basis that

- it is proven over the Internet,
- members coming from the ISPs world were familiar with it;
- it is extensible,
- supports thousands of nodes,
- its highly configurable,
- It is robust,
- well supported by many operating systems and
- well documented.

It is what makes the whole Internet work and it was the first choice to counteract the instabilities and the issues faced with the previous OSPF deployment.

Currently the BGP routes on the routing table of Athens have reached close to a thousand together with the summarized subnets from other areas in Greece.

AWMN considered OSPF. It is stable as it is but in the original RFC version not really suited for wireless networks because it’s agnostic to the quality of the medium, BGP has been challenged by new dynamic protocols developed specifically for the task of meshing in the wireless world. OLSR has been used in order to facilitate link quality and best route selection in a small scale. Mainly because of its scalability problems back in the first releases and mainly due to the reason that
AWMN has already grown a backbone large enough to make migration difficult, small areas with bottleneck have been chosen to test this protocol and utilize it as an IGP protocol.

AWMN has also worked on OLSR integration. To integrate those small areas where OLSR is deployed back to the main backbone and exchange routes, developers of the community had to program integration code between OLSR daemons and Quagga.

Quagga\(^1\) is a routing software suite, providing implementations of OSPFv2, OSPFv3, RIP v1 and v2, RIPng and BGP-4 for Unix platforms, particularly FreeBSD, Linux, Solaris and NetBSD. Quagga is a fork of GNU Zebra\(^2\) which was developed by Kunihiro Ishiguro. Natively this package is used in most communities to deploy BGP routing within Linux routers.

The Quagga Plugin was built for OLSRd and it works as an extension plugin in order to provide interoperability between OLSR and BGP.

It allows OLSRd to redistribute from various Quagga-protocols as well as to export OLSR-routes to Quagga so that they can be redistributed by the Quagga-routing-daemons.

In a typical AWMN installation an area of backbone nodes will form a BGP Confederation and unite several AS numbers under a single Confederated AS number. This Confederation will be visible as one AS number to the rest of the network.

The benefits of this is that it best utilizes local closely wirelessly connected clusters of nodes since the traffic of the confederation always stays within this area best selecting routes not only on hop count but also by link quality.

The concern has always been that since those areas become only one hop count since they engulf many ASes to one Confederated AS, thus cutting the length of many AS Paths that included in the past those nodes, the confederation can become a congestion point since the rest of the network using BGP tries to route traffic through that virtually shorter path. Therefore careful design has to be made in order to both select the correct nodes to be included in such a confederation and raise availability through alternative paths which means that possibly new redundant links have to be made within the Confederation AS.

2.1.1.2. GUIFI.NET

Guifi.net was born in a rural area of Catalonia in 2004 and expanded among all the Catalunya, Valencia and other areas (see http://guifi.net/maps). As of October 2012 it has more than 18.600 working nodes and over 40000km working links.

Over the time it has become an umbrella for many already existing Community Networks. These communities have adopted Guifi.net methodologies and joined efforts. This is one of the reasons explaining why Guifi.net has become the largest Community Network although it was started later than many others.

Other reasons to understand why Guifi.net has become a de facto standard are the highly scalable tools developed by the community to design, deploy and manage the infrastructure. Thanks to these tools IP allocations, device configurations, services registration, etc. are completely automatic, hence, human errors are reduced to the minimum as well as the knowledge needed to join the network.

Regarding IPv4 ranges, Guifi.net still follows the old Freenetworks.org allocations (a currently unmaintained project born a decade ago meant to coordinate RFC1918 IP allocations among Community Networks world wide). The reuse of some IP ranges coming from some of the smaller communities that merged with Guifi.net has helped a lot in this sense. At the moment Guifi.net IP

\(^1\)http://www.quagga.net
\(^2\)http://www.zebra.org/
allocations (both RFC1918 IPs and public v4 and v6 IPs, and ASs as well) are registered in a new interoperability effort (see http://interop.wlan-si.net/wiki/IPAddressing/List).

From the routing point of view, Guifi.net follows the Internet approach, i.e. a BGP network interconnecting more than 2000 Autonomous Systems (AS), most of them running OSPF internally.

From the topology point of view the infrastructure mode (i.e. a backbone made out of dedicated links and access points connecting end user CPEs) is the predominant. Even though efforts to offer alternatives base on AdHoc mode plus dynamic routing protocols (like BMX6 or OLSR) also exist (for instance QMP.cat).

2.1.1.3. FUNKFEUER

Funkfeuer – as opposed to most community wireless networks – has the luxury of owning public IpV4 and IpV6 space since its beginning. Therefore, the structure of IP addressing is such that each new node will get the next free public IP address. In this sense, the tinc tunnel that most community networks need in order to connect to the CONFINE images is strictly speaking not really necessary from Funkfeuer’s perspective. However, in order to stay compatible with the other networks and to ease federation, the tinc tunnel mechanism will be used as well.

Concerning the routing protocol in use: Funkfeuer's mesh network exclusively uses the OLSR\(^3\) mesh routing protocol. Members of the Funkfeuer network are also active developers in the olsr.org open source project.

Between OLSR and our uplink to the Internet, there is an OSPF routing zone which connects the routers which announce internet connectivity to the OLSR mesh. From the OSPF zone, there are links to our BGP gateway. At the time of this writing, there is one BGP gateway with multiple links to different upstream internet providers. Funkfeuer is multi-homed and peers with a couple of local Internet service providers.

2.1.2. NETWORK OVERLAYS

The information above shows that networking in community networks and connecting community networks with each other is a serious challenge. Therefore, the general approach followed by CONFINE testbeds to overcome connectivity limitations inside and between community networks is to set up a mesh VPN. This runs the tinc software, forming an overlay management network using its own IPv6 addressing.

Gateway hosts help extend the testbed management network over different community networks by connecting across external infrastructure like the Internet or the FEDERICA research backbone. The management network and more information on tinc are available in deliverable 2.1 Initial system software and services.

2.1.3. CONCLUSIONS

As described in the previous sections, all community networks will benefit from enhancements to ad-hoc routing protocols like OLSR and BMX6. Improved scalability and better reactions to changing link properties in wireless environments are a key factor for the further growth and spreading of community networks.

\(^3\)www.olsr.org
2.2. OLSRV2 DEVELOPMENT

OLSR is currently the de-facto open source routing protocol used by most community mesh networks. It is still used by most community mesh networks because in any real world mesh network with actual users, stability is paramount and beats all other nice properties that other mesh routing protocols might offer. If there is no stable network, people will not join it or use it. OLSR is based on the IETF RFC 3626 [RFC-3626] from 2003. Since OLSR has been around for many years, the community wireless networks (mostly the Freifunk networks as well as Funkfeuer) have been using their network to “test” and “debug” OLSR over the years. This resulted in high stability. In this context and under these circumstances, and in order to be of practical use for larger wireless networks, the protocol has been modified several times. OLSR.org⁴, which provides the code basis for many deployments especially in the field of community networks and also for many commercial products, is a prominent example. Unfortunately, the actual original specifications, RFC 3626 has not been changed to reflect the enhancements used in mesh network deployments.

However, in the last years the IETF MANET working group has been working on a new revision of OLSR called OLSRv2. OLSRv2 has been split into four different documents to allow for better re-usability.

The first three documents have already been finalized. They describe the packet format [RFC-5444], the encoding of time intervals [RFC-5497], and neighbourhood discovery [RFC-6130].

The OLSRv2 draft 17 [OLSRv2-17] has already passed its last call and will (most likely) become an RFC in 2012. This document specifies the core components of the routing protocol, including the handling of routing metrics.

To the best of our knowledge, no complete free/open source implementation of OLSRv2 exists. The implementation of OLSRv2 by the CONFINE project especially considers embedded systems and thus will allow for community mesh networks and researchers to use a modernized, flexible, and extensible routing protocol. Its built-in flexibility will also make it easier to do basic research on link-state routing.

To make it easier to reuse parts of the OLSRv2 code for other projects, the code has been split into two parts. A basic framework includes all parts of the code that are not OLSRv2 specific. The second part is the OLSRv2 application based on the framework.

2.2.1. BASIC FRAMEWORK

The framework code (see Figure 2.1) is a set of five libraries written in C, which can be used to write network applications for embedded and non-embedded systems. A lot of the code was originally written for the OLSR.org routing daemon, but has been cleaned up and extended for the OLSRv2 project.

Figure 2.1: Structure of framework code decomposed into five libraries.

⁴www.olsr.org
The five libraries are called Common, Config, RFC5444, Core, and Tools. The more complex libraries (Core and Tools) are built on the simpler ones (Common, Config, and RFC5444) to provide higher level abstractions. This allows a developer to use both high level abstractions and low level access where needed.

**Common-library:** Provides a set of helper functions that have been originally developed for the OLSR.org code. Among other things it contains code for linked lists, AVL trees, network address objects and variable sized buffers.

**Config-library:** Implements a lightweight configuration system with pluggable configuration formats and sources, automatic testing of configuration against a pre-defined schema, and conversion of configuration data into binary C structures. It can handle multiple sets of configuration at the same time and can track differences between multiple sets of configuration.

**RFC5444-library:** Provides all the necessary functions to parse and create RFC5444 compatible packets. It allows multiple listeners for overlapping parts of incoming packets and also automatically generates packets based on a list of Content Providers. This makes it easier to extend existing RFC5444 messages with new content through plug-ins, without changing the main code. It is also one of the few RFC5444 implementations (the only one we are aware of) that handles automatic address compression.

**Core-library:** Contains a large set of modules for building standard network centric applications and daemons. It contains a flexible logging system, memory block handling, a plug-in loader, and a timer and network scheduler that can handle multiple timers and sockets simultaneously. A thin abstraction layer can handle configuration, dual-stack issues and access control for TCP and UDP sockets.

The Core library also contains an abstraction of the necessary operation system dependent functions, which allows the OLSR.org routing agent to run on Linux, Android, Windows and BSD/Mac.

**Tools-library:** Contains high level utility functions. It provides code to configure the logging system with the Config library and additional code to implement a full Commit & Rollback capable system based on the Config library. It also contains a full multiplexer/demultiplexer module/routine that connects the RFC5444 library with the Core libraries socket handling.

In addition to this the Tools library contains an embedded Telnet and HTTP server, which can be used by registering any number of callbacks for Telnet commands or HTTP sites.

The framework can be used for any kind of network application. It will not be considered stable until the OLSRv2 implementation is also considered stable, but it will not depend on any code from the OLSRv2 implementation.

2.2.2. LIGHTWEIGHT IMPLEMENTATION OF THE NEIGHBORHOOD DISCOVERY PROTOCOL

The Neighbourhood Discovery Protocol has been standardized by RFC 6130 in April 2011. It is a revised version of the Hello protocol of OLSRv1 which can be used independently from OLSRv2. During the development process of OLSRv2 several parties asked that the Neighbour Discovery part should be put into its own RFC so it could be reused by different protocols.
One of the goals of CONFINE is to implement a standard compliant OLSRv2 routing daemon to allow easier experimentation and a long-term platform for the Community Mesh Network routing. Our implementation of NHDP is directly based on the framework described in section 2.2.1. It consists of a series of databases for incoming and local data, message generation and message parsing code. The first complete prototype implementation was tested at the end of September 2012.

The next steps for our NHDP implementation will be the creation of an optional plug-in that checks the content of the incoming NHDP messages for illegal data and drops the messages if necessary. These checks can be implemented without changing the code for the main protocol functionality.

The upcoming implementation of OLSRv2 will use NHDP as described in the corresponding IETF document to build a full Mesh Routing protocol. It will still keep the split between OLSRv2 and NHDP to allow the NHDP code to run independently by just removing the OLSRv2 part.

2.2.3. TESTS

To increase the stability of the framework, several automated tests have been added to the framework repository. These contain mostly tests of the lower level libraries (Common, Config and RFC5444) to ensure that no bugs are introduced by re-factorings of the libraries. The automated tests check several kinds of basic operations and corner cases of the framework as well as test basic use cases for the configuration and RFC5444 library. Finally, they check the RFC5444 libraries’ compliance to the OLSRv2 interoperability tests of 2010.

We plan to add more automated tests during the further work on the OLSRv2 implementation.

Our NHDP implementation has also already been tested against the U.S. Naval Research Laboratory (NRL) implementation [NRL-NHDP]. After a couple of bug fixes on both sides, both implementations seem to be able to interoperate with each other in a number of basic tests.

We are planning a larger interoperability test with the NRL’s implementation and a closed source implementation (JOLSRv2) provided by Thomas Clausen, one of the authors of the OLSRv2 RFCs.

2.3. BMX6 EVALUATION IN COMMUNITY NETWORKS

Community Wireless Networks have an organic growth that now-a-days reaches the tens of thousands of nodes and keeps increasing. Since these nodes have no centralised or readily manual control, community networks can benefit from MANET routing protocols, that self-adapt to network changes, determining the best path for end-to-end delivery of messages across the network.

Within CONFINE, we have performed an evaluation of a couple of such routing protocols, to provide some insight about their scalability. Specifically, we have studied and compared the performance of OLSR and BMX6, taking advantage of the presence of developers from both protocols in the CONFINE consortium.

2.3.1. EXPERIMENT DESCRIPTION

The proposed experiment consists on running both protocols on several networks of different sizes and analyse the effect of the network size on their performance. We consider two different metrics to measure a protocol’s performance:

http://www.herberg.name/projects/79-olsrv2-implementation/71
• **Protocol overhead**: The overhead of a routing protocol is the quantity of control traffic required to be sent through the network in order to work properly. As shown in Figure 2.2, both protocols generate more overhead during start-up, so we only have considered the overhead generated after stabilization, giving both protocols a margin of a couple of minutes to stabilize.

• **Convergence time**: The convergence time measures the time it takes to the network to become aware of a change in it. Therefore, it depends not only on the network characteristics and routing protocol, but also on which the change in question and the definition of awareness. We propose as change, to add a new node to the network and connect it to a randomly selected node. Then, we measure the convergence time, as the time it takes to this node to be capable of communicating with the node that is farthest away in the network (Figure 2.3).

![Figure 2.2: BMX6 and OLSR overhead on network start up](image1)

![Figure 2.3: Measurement of convergence time](image2)
2.3.2. EMULATION ENVIRONMENT AND FIRST EXPERIMENTS

As a first step, we have produced several results of the experiment described under an emulated environment. The emulated network topologies replicate a piece of Guifi.net network, as retrieved from their website in CNML format. The network description of Guifi.net has been processed to generate networks of different sizes by means of randomly selecting the desired amount of nodes and all the links existing among them. Following this approach we have generated 20 different topologies over 7 different zones of Guifi.net, with sizes between 10 and 90 nodes.

Then each topology has been emulated using Mesh Linux Containers (MLC)\(^6\), which is a collection of scripts that use Linux Containers and ip tools to virtualise the network nodes and the network topology. The whole experiment cycle is shown in Figure 2.4.

![Figure 2.4: Experiment cycle](image)

Figures 2.5 and 2.6 show the performance results of both protocols obtained from the emulation. As we can see in Figure 2.5, BMX6 overhead grows slower as the network size increases, obtaining better results than OLSR for networks with more than 30 nodes. On the other hand, we discovered that the convergence time does not depend as much on the network size in terms of number of nodes, as it does in the network diameter. Figure 2.6 shows how the convergence time increases as the number of hops between the new node and the farthest away node in the network, as we can see, the effect of the OLSR’s fish-eye extension [FISH EYE] is shown as a step when there are more than 8 nodes. More details about the results and the experimentation process itself can be found on paper [BMX6].

In conclusion, our first results show that BMX6 outperforms OLSR in the selected metrics as networks grow, which proves that BMX6 is marginally affected by the addition of new nodes and the size of the network in both metrics studied.

\(^6\)[https://www.github.com/axn/mlc]
2.3.3. EXPERIMENT IN THE CONFINE TESTBED

These results obtained through emulation are the first ones for this experiment, but we expect to be able to repeat an equivalent experiment on the CONFINE testbed and validate them.

As you can realise from descriptions on section 1.1.3, this experiment is categorised as active, conflicting and requires L3 access. For these reasons its execution is restricted to CONFINE clouds, where nodes can be granted an isolated interface and community networks’ QoS is not affected by the experiment.

Being restricted to a CONFINE cloud accentuates the importance of validating the results obtained through emulation, since the size of a CONFINE cloud is small compared to the size an emulated network can achieve.
3. CROSS-LAYER INTERACTIONS AND OPTIMIZATIONS

3.1. INTRODUCTION

Cross-layer optimizations are a network optimization where information from different ISO/OSI layers is used to optimize the network protocols which usually only operate on a single layer.

When considering the challenges listed in the previous sections, it is clear that cross-layer solutions can help improve the community networks. In the first year, the project has focused on the radio-to-router communication protocols, to support the testbed itself. We expect open data sets to give us more insights in the networks to further introduce cross-layer interactions in the networks.

3.2. RADIO TO ROUTER COMMUNICATION PROTOCOLS

Node design in current community networks poses a challenge because the creators face a problem common for larger nodes. While there is cheap hardware available with one or two WiFi radios, getting hardware with more WiFi interfaces can be difficult and expensive. There is also an issue with connecting lots of antennas to a central device, because long antenna cables can easily degrade the signal quality and decrease the data rate of WiFi.

The common solution of these problems is to install multiple cheap devices at a single location and connect them via Ethernet. Each of the devices will run its own routing daemon and a full firmware image.

While this solves the problems mentioned in the first paragraph, it introduces a couple of new issues. Instead of managing one device, the administrator of the node has to manage multiple devices, each with its own configuration. In addition to this, the complexity of the routing topology increases with the number of used devices, which can become a challenge for large scale community networks.

![Figure 3.1: Improved node design using radio to router communication protocols](image-url)
A better solution for this problem is building larger nodes by combining a number of cheap devices with a central routing device as depicted in Figure 3.1. Each of the lightweight 'radio' devices bridges the incoming traffic to the attached Ethernet, which allows the central routing device to handle the combined WiFi devices similar to the local ones.

This strategy helps to simplify maintenance, because the 'radio' devices can be equipped with a much more primitive software that mostly relies on auto-configuration.

The unresolved question about this type of “distributed Node” is the control channel between the central router and the radios. A detailed discussion was published at the 1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB 2012) [BR2012].

### 3.2.1. DESIGN OF A USABLE PROTOCOL FOR COMMUNITY MESH NETWORKS

There are two RFCs describing the transport of link metrics over PPPoE (Point to Point Protocol over Ethernet), RFC 4938 [RFC4938] and its successor RFC 5578 [RFC5578]. Both define an extension to the PPPoE to transport metric data from a modem or radio to the router and provide a credit window scheme based flow control.

But PPPoE is not the right choice for wireless mesh networks. First, the PPPoE protocol adds a wrapper to each data packet sent over the mesh, which increases the overhead for the user traffic. In addition to this PPPoE is only designed for point to point connections (as the protocol name declares), which means that it would be necessary to create one PPPoE connection between each pair of mesh nodes within communication range. As a side effect, this would also make the usage of multicast impractical.

The IETF MANET (Mobile Ad-hoc Networking) working group is currently discussing a new protocol draft called DLEP (Dynamic Link Exchange Protocol), which should be able to solve the control channel issue between radio and router.

It defines a communication protocol between radio and router which includes automatic discovery of radios, transport of metrics and statistics to the router, and configuration of the radio by the router.

The concept of DLEP already includes the existence of multiple unicast and multicast destinations on a radio. Communication happens exclusively between the router and the local radios, which introduces no overhead for the traffic sent between two radios.

The current draft [DLEP-03] introduces a TLV (type-length-value) based message format, which allows adding data to the messages on demand. This will allow coming DLEP implementations to supply additional metric types without breaking compatibility with the existing ones.

DLEP will allow community mesh networks to build large scale nodes with simple radio devices connected to a central powerful router with Ethernet. It will also help to extend the capabilities of the CONFINE research nodes, by allowing the researchers to access link-layer metrics through a network protocol without giving the researcher direct access to the hardware.

### 3.2.2. IMPLEMENTATION BASED ON THE OLSRV2 BASIC FRAMEWORK

Our prototype implementation of DLEP is not standard compliant, because the current draft is still undergoing changes (see section 3.2.3).

Both the former and the current DLEP draft [DLEP-02] [DLEP-03] leave too many aspects of the protocol to 'implementation details'. This will likely result in DLEP-compliant implementations which cannot interoperate with each other.
In addition to this both drafts define only a small set of metric TLVs, some of them with questionable value encoding.

To provide an example, the RelativeLinkQuality (RLQ) metric TLV defines a range of 0-100 for the quality of the link, with both perfect links and links that cannot be measured (or support no quality measurement) be returned as 100. Using a TLV like this as a routing metric will make it impossible for the router to shift the traffic away from links that cannot be measured to the ones that can.

To simplify the change to a standard compliant protocol, the implementation splits up the necessary code of an application into “link-layer data providers” and “link-layer data consumers”. Providers and consumers are connected by a lightweight Layer-2 database which can store and retrieve link-layer data about networks or neighbours on a network.

“Data providers” gather the data from any kind of source and put them into the database while “data consumers” take the data from the database and process them.

The Layer-2 database stores both network data (layer-2 information about a whole network) and neighbour data (layer-2 information about the link to a specific neighbour).

Each layer-2 network data set (one for every radio attached) is identified by a MAC address of the radio. For each of the networks, the database stores whether it is active or not. The data entries are automatically set to inactive if the validity time or the information has expired without an update. In addition, the interface index is stored (zero for remote networks via DLEP).

The optional data fields for layer-2 networks are currently

- **SSID**: Identification of the network
- **Last Seen**: Number of milliseconds since the network was active the last time
- **Frequency**: Frequency of the network in Hz
- **Supported Rates**: Array of supported data rates in bit/s

The layer-2 link data is identified both by the radio MAC and the MAC of the link partner. The database stores if the neighbour is active, similarly to the layer-2 network data. It also stores the interface index if the neighbour belongs to a local interface.

The optional data fields for layer-2 neighbours are currently

- **Signal**: Last measured signal strength of the neighbour in dBm
- **Last seen**: Number of milliseconds since the neighbour was active the last time
- **Tx/Rx bit rate**: Current transmission/reception unicast bit rate in bit/s
- **Tx/Rx packets**: Number of packets sent/received with this neighbour
- **Tx/Rx bytes**: Number of bytes sent/received with this neighbour
- **Tx retries**: Number of link layer retransmissions with this neighbour
- **Tx failed**: Number of failed transmissions with this neighbour

The prototype implementation consists of four plug-ins for the Framework.

The NL80211-Listener is a data provider which uses the Linux Netlink interface to the WiFi stack to query the operation system in a regular interval about the available layer-2 data and put it into the layer-2 database.
The DLEP-Radio plug-in is a data consumer that reads the local database and transmits its values over a UDP connection to a DLEP-Router plug-in. The DLEP-Router plug-in is the corresponding data provider which parses the incoming UDP packets and puts the content into the Layer-2 database.

The final plug-in (a data consumer) is the Layer-2 Viewer, which allows for a user to query the Layer-2 database over the Telnet interface of the Framework. This plug-in supports a number of Telnet commands to display either network or neighbour data, both with normal text output, JSON format or a user supplied template for the output.

This interface makes it easy for researchers to query the data without implementing their own C code:

```
echo /layer2 neigh json | nc 127.0.0.1 2007
{
    "neighbor" : "00:15:6d:84:0a:03",
    "radio" : "00:15:6d:84:08:bd",
    "ifindex" : 0,
    "interface" : "",
    "active" : true,
    "shortactive" : "",
    "lastseen" : 2.566,
    "signal" : 7,
    "rxbitrate" : 56623104,
    "rxbytes" : 13293418,
    "txbitrate" : 56623104,
    "txbytes" : 1068,
    "txpackets" : 10,
    "txretries" : 0,
    "txfailed" : 0
}
```

We plan to add a plug-in to the OLSRv2 implementation that will use the content of the Layer-2 database to calculate a link metric for the routing protocol as soon as the OLSRv2 implementation is done.

The separation of data consumers and data providers will make it easier to replace the DLEP prototype with new code without changing the software parts that collect information from the operation system or process the collected information.

It also makes it trivial to put the plug-ins together in different configurations, for example using the NL80211-Listener directly together with a consumer like this Layer-2 Viewer.

### 3.2.3. STANDARDIZATION AT THE IETF

Since March 2012 we have been participating in the standardization process of DLEP at the IETF, both on the mailing list of the MANET working group and at the IETF meetings.
Unfortunately the standardization is progressing slowly.

DLEP draft 02 was available in March 2012, but used the standardized RFC5444 packet format in an unacceptable way. In addition to this the draft 02 was underspecified, leaving a lot of details up to the implementations.

We wrote a private draft [DLEP-STATELESS] to present how a DLEP like protocol could be done with using RFC5444 in a standardized way. This draft replaced some of the more complicated protocol mechanisms of DLEP with simpler ones derived from existing MANET protocols. After a long private discussion with a few MANET working group members, we decided against publishing the draft before the IETF in Vancouver to keep the door open for a cooperative effort for DLEP.

The private draft has been shared at this time with a few members of the working group, including the authors of the original DLEP draft itself.

DLEP draft 03 has been released at the end of August 2012, shortly after the IETF in Vancouver. Despite travelling to the IETF in Vancouver and providing a lot of input to the authors of the draft, DLEP-03 didn’t become better than DLEP-02.

While this revision replace the former packet format with a completely new (and cleaner) creation, the protocol itself got even more degrees of freedom and “implementation details”. At the moment, the standard will allow multiple valid implementations which might not be able to interoperate with each other.

In November 2012 we published our private draft on request of the original DLEP authors as an official IETF document.

We will continue to participate in the standardization and work on transforming DLEP into a useful and well-defined standard for radio-to-router communication.
4. SELF-MANAGEMENT

Self-management is a research topic in CONFINE, where the goal is to make the networks and the testbed more autonomous. The network infrastructure should be able to recover from issues, without or with limited manual intervention.

At the moment, research on this specific topic has been limited in CONFINE, although the work on OLSR and BMX is strongly related (see sections 2.2 and 2.3). Routing protocols are designed to cope autonomously with network failures.

We expect more research on self-management while the testbed is used by the open call projects. A number of topics are foreseen, including address allocation. Especially with the introduction of IPv6 addressing in the networks, self-management will be assessed. We do expect the impact of this task to depend on the different networks, as introducing self-management and its results will depend on the current networking situation.
5. BEST PRACTICES FOR EXPERIMENTATION

The project has taken initial steps towards a list of best practices for experimentation. A first step was the execution of a prototype Bittorrent experiment. This provided more insight in the current state of the testbed and its documentation. Then, for the OMF integration, an overhead analysis of OMF in embedded devices has been performed, together with the development of a test environment for network measurements, based on OMF. We expect these steps to help us compiling best practices for experimentation in the coming years.

5.1. A PROTOTYPE BITTORRENT EXPERIMENT

The "Bittorrent experiment" is based on D'Elia et.al work [DEL2011], where the authors propose a cross-layer solution for P2P Bittorrent traffic optimization in WMNs based on the ALTO (Application Layer Traffic Optimization) service recently defined by the IETF. Our interest on replicate this experiment rather than evaluate the proposed optimization, was to evaluate the experience of deploy an entire experiment on CONFINE. We selected this experiment among others because it is based on Peer-to-peer application, which nowadays is a well-known example of distributed and large-scale networking application with a considerable amount of traffic over the Internet. Additionally, it is deployed and evaluated under CWN scenario. Also, the experiment itself is enough simple to be used as a "tutorial experiment" for the rest of the community, but also sufficiently complex to show how all the pieces developed can work together.

The expected results of this experimentation are not focused on measure the download time or verify the protocol improvement, as did the authors of the original work. Instead, we achieved some goals more useful for the project.

First, we created and published both tutorials – one in video format, another on our wiki-page – showing how to use the tools provided by the project to execute this experiment on CONFINE testbed. Second, we have detected and resolved a numerous amount of software bugs and hardware improvements.

The whole work, and a life demonstration, has been shown on the Int. P2P conference [DCN2012].

5.2. OVERHEAD ANALYSIS OF EMBEDDED WIRELESS TESTBEDS WITH OMF

Frameworks like the cOntrol and Management Framework (OMF) are helpful software tools to organize, control and instrument testbeds. However, community networks have a very heterogeneous infrastructure in which less powerful, embedded devices will play an important role due to their low cost. This section describes the analysis performed in CONFINE of the overhead generated by the OMF framework when deployed on embedded devices.

Two hardware platforms were used in the analysis. Two identical devices of each hardware platform were used to study the performance impact.

One platform used a Dell Desktop system which will be referred to as the Silver nodes. The other platform is a low performance embedded system of PCEngines referred to as the Alix nodes. When executing the experiments, the antennas (type Reverse polarity SMA) of the wireless cards in the
nodes were at a small distance (~50cm). The nodes were also exposed to interference from other wireless networks near the nodes. The Transmit Power of the Silver and the Alix nodes is 27 dBm.

An overview of the hardware in each platform is given in this table:

<table>
<thead>
<tr>
<th></th>
<th>Silver Nodes</th>
<th>Alix Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Pentium 4 3.0GHz</td>
<td>AMD Athlon Geode 1GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>1024 MB</td>
<td>256 MB</td>
</tr>
<tr>
<td>Disk</td>
<td>Seagate 250 GB 7200 RPM SATA</td>
<td>Transcend 133x 4 GB Compact Flash</td>
</tr>
<tr>
<td>Wireless</td>
<td>Atheros AR5413/AR5414</td>
<td>[AR5006X(S) 802.11abg] (rev 01)</td>
</tr>
</tbody>
</table>

Table 1: Overview of used hardware

The experiments were performed on two different software systems. The software on the Silver nodes was based on the Ubuntu 12.04 beta distribution, the Alix nodes were running the OpenWrt Trunk (the version used in CONFINE).

An overview of the used software and version numbers can be found below in table 2:

<table>
<thead>
<tr>
<th></th>
<th>Silver Nodes</th>
<th>Alix Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
<td>Ubuntu 12.04 beta</td>
<td>OpenWrt Trunk</td>
</tr>
<tr>
<td>Linux Kernel</td>
<td>3.2.0-20 generic-pae #33-</td>
<td>3.2.12</td>
</tr>
<tr>
<td></td>
<td>Ubuntu SMP</td>
<td></td>
</tr>
<tr>
<td>Optimizations</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ruby</td>
<td>1.8.7p352</td>
<td>1.9.2p0</td>
</tr>
<tr>
<td>Iperf</td>
<td>2.0.5 pthreads</td>
<td>2.0.5 single threaded</td>
</tr>
<tr>
<td>OMF</td>
<td>5.4</td>
<td></td>
</tr>
<tr>
<td>OML Iperf</td>
<td>2.0.5 pthreads</td>
<td></td>
</tr>
<tr>
<td>OTG</td>
<td>2.5</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Overview of used software

An attempt was made to make the difference between the two systems as small as possible. The used software — discussed below — has the exact same version on each platform. To be able to use a new Kernel the beta version of Ubuntu 12.04 was installed because it was the only release that contained a recent Kernel version. The OpenWrt software on the Alix nodes was compiled with optimizations for the specific type of hardware.
In this analysis, the OMF Measurement Library provided by OMF is used. OML allows experimenters to define multiple measurement points inside their application. These measurement points can be aggregated e.g. to sum the data. That data is aggregated in a so-called sampling window. The size of this window is given by the samples parameter which will return measurements for every X measurements or by the interval parameter which will return measurements for every X seconds.

OML uses a client/server architecture. While the experiment is running, the client sends its measurement data received from the application(s) to the OML Collection Server. The client will reside on the RC and the Collection Server on another server. The experimenter can access the results on the Collection Server.

Three different traffic generation tools were used: Iperf, Iperf for OML and OTG/OTR.

Iperf is an open-source application developed by NLANR/DAST to study the maximum throughput performance of network systems. Iperf can send both TCP and UDP packets from client to server. The data stream can easily be modified by changing parameters such as bandwidth or datagram length. Network characteristics such as bandwidth, delay jitter and datagram loss are reported.

At the end of each run the Iperf client reports how much data was sent during which time frame, and how much of it was received by the Iperf server. The results are based on the rate that was indicated by the receiving machine.

Iperf for OML is a version of the Iperf application which was modified by adding OML measurement points to the source code. This way, the data reported by Iperf is sent to the OML server. This application is started by an OMF experiment.

Iperf for OML was run in two different configurations for each machine. Both configurations use UDP.

In the Iperf OML All configuration all packets statistics were assembled and sent to the OML server. The transmission and reception timestamps of each packet are recorded.

In the Iperf OML Aggregated configuration all packet data was observed on the resource controllers, but the results were aggregated in aggregated samples over 30 seconds. For each sample the sum and average of all packet sizes is calculated and passed to the OML server.

OTG/OTR (ORBIT Traffic Generator / Receiver) achieves the same goal as Iperf, i.e. studying the throughput of a network link. OTG supports OML out-of-the-box, but has fewer features than Iperf. E.g., it supports only UDP traffic while Iperf also supports TCP traffic. This application is started by an OMF experiment.

OTG was configured, like Iperf for OML, with two different configurations similar to the ones of Iperf for OML.

The first configuration also sends all packet information like packet size and transmission/reception time to the OML server.

The second configuration aggregated packet information over an interval of 30 seconds and sent the transmission/reception time of the first and last packet and the sum of all packet sizes.

All programs were configured to use the UDP protocol. Using different packet sizes will have an impact of maximum achievable throughput [10]. All experiment configurations have a packet size of 1470, nearly the maximum size before fragmentation occurs. This is the default in some applications, like e.g. Iperf.
When the first result (Iperf OML All) is compared with the results of the “plain” Iperf results the impact of the OML reporting can be observed. The second experiment (Iperf OML Aggregated) has a lower reporting rate, to study the impact of OML measurement aggregation.

Every type of experiment was repeated 100 times and the duration of each experiment was 300 seconds. After all experiments were executed the results of the different runs of each experiment were averaged. In Figure 5.1 the average throughput of the different systems is shown, the error bars in the graph indicate the standard deviation. No result is shown for the OML Iperf Alix All experiment, as the reported performance was very low. In what follows, the experiments will be discussed separately. In the tables, the average, minimum, maximum and standard deviation $\sigma$ of the throughput will be considered.

The detailed results of this analysis are available in the work "Overhead Analysis of Embedded Wireless Testbeds", published in the proceedings of the CNBuB workshop [SAM2012]. A short summary follows. In all studied configurations the reported throughput results were lower than the one measured in the normal Iperf experiment. No significant difference can be seen between the OTG experiments and the OML Iperf experiments, except for the All experiment on the Alix nodes. This can be expected since the two applications in these configurations should be performing the same task.

It is clear that OML introduced some overhead on the throughput of the systems on both powerful machines and on embedded devices. In only one experiment significant differences were found between the Silver nodes and the Alix nodes. Using the OML framework on the other hand will have an impact of about 6%, whether run on embedded system or on powerful machines. In one particular case the impact was so dramatic that the throughput was less than 25% of the throughput in the other experiments. This turned out to be a logging problem.

In general, we believe the overhead of embedded testbeds is limited, when researchers take care while logging experiment data.
5.3. DEVELOPMENT OF A TEST ENVIRONMENT FOR NETWORK MEASUREMENT TOOLS BASED ON OMF

The evaluation of network measurement tools, to be applicable to for real-world networks, requires the conditions of the test network to be as close as possible to the networks in which they are later to be deployed. Thus, having an existing real-world network which can be utilized for tests (while also allowing reconfiguration of certain characteristics/features) would be of significant interest to researchers. A broad survey reveals a special interest in the real-world conditions of two categories of network characteristics: those related to traffic control and those related to topology control. Provisioning the researcher with corresponding management tools with which they can artificially cause such conditions and topology changes would help them in realizing realistic test bed setups and scenarios in test environments.

Traffic control-related conditions:

- Network latencies and packet anomalies
  - Queuing delays
  - Packet losses
  - Packet duplications
  - Packet corruptions
  - Packet re-orderings
- Traffic Shapers [KAN2011]
- Different Queuing Disciplines
  - Classless (FIFO, red, stochastic fair queuing etc.)
  - Classful (class based queuing, hierarchical token bucket etc.)

Topology control-related conditions:

- Multipath diversity / Route alternation [DOW1999]
- Symmetric / asymmetric links [JIA1999] [CHE2005]
- Single / multiple bottleneck [LAI2001] [KAT2004]
- Infrastructure / Ad-hoc wireless modes [SIN2011][JOH2004]
- One share medium (wireless links) / separate collision domains (wired links)

To allow the evaluation of network experiments in terms of such challenges, practical issues and difficulties, researchers need corresponding management tools with which they can artificially cause such conditions and topology changes in test environments.

In addition to such traffic and topology control tools, network experiments also require corresponding testbed setups and further tools to assist the execution of the experiment scenario (i. e. cross traffic generators, passive packet analyzers etc.). Our survey in the field of network management related literature shows that there are some testbed setups and assisting tools which are typically used. Enabling the integration and use of such testbed setups and third-party tools with OMF (cOntrol and Management Framework) would significantly reduce the work to be done by the researchers in order to perform such typical network measurement experiments on OMF-enabled research networks.
In this section, we describe the rationale and development of a test environment for network measurement tools based on OMF. We aim mainly to support the OMF researchers in defining and conducting their experiments by

- providing application and measurement point definitions for well-known / often used third-party network management and measurement tools running under OpenWrt
- providing configurable experiment descriptions that can automatically set up typical evaluation testbeds and scenarios derived from literature using the CONFINE research nodes

Finally, all the application and measurement point definitions of the selected tools as well as provided template scripts will be made available in the form of an OMF image that can be easily loaded to an arbitrary research node within the OMF-enabled test environment. The OMF image will also contain the already installed versions of all OMLized tools.

### 5.3.1. SURVEY OF TYPICAL NETWORK MANAGEMENT AND MEASUREMENT TOOLS FOR OMF INTEGRATION

As mentioned above, we performed a survey of the literature related to network management research in order to identify the best-known or most often used tools [ABU2012][LAB2005]. Our initial review resulted in the list of tools shown in Table 1 that are categorized into three different groups: Traffic control tools, topology control tools and network measurement tools.

For each tool from Table 1, the following templates will be provided:

1. The application definition file for the tool to be integrated with the OMF framework
2. The modified source code of the tool containing the measurement point definitions
3. And finally a sample experiment description file (to demonstrate how to run an experiment).
### Traffic Control Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tc</td>
<td>A tool that can cause artificial queuing delays, packet losses, duplications, corruptions and re-orderings as well as shape traffic</td>
</tr>
<tr>
<td>iptables</td>
<td>Administration tool for IPv4 packet filtering and NAT</td>
</tr>
</tbody>
</table>

### Topology Control Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ethtool</td>
<td>Display or change Ethernet card settings (e.g. interface speed, communication channel etc.)</td>
</tr>
<tr>
<td>mii-tool</td>
<td>View and manipulate media-independent interface status</td>
</tr>
<tr>
<td>ifconfig</td>
<td>A tool to configure a wired network interface</td>
</tr>
<tr>
<td>iw, iwconfig</td>
<td>A tool to configure a wireless network interface</td>
</tr>
<tr>
<td>route</td>
<td>Show and manipulate the IP routing table</td>
</tr>
</tbody>
</table>

### Tools for assisting the network measurement experiment

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-ITG</td>
<td>Distributed Internet traffic generator</td>
</tr>
<tr>
<td>tcpdump</td>
<td>A well-known packet analyser</td>
</tr>
<tr>
<td>httperf</td>
<td>A tool for measuring web server performance</td>
</tr>
<tr>
<td>owamp</td>
<td>One way delay measurement tool</td>
</tr>
<tr>
<td>iperf</td>
<td>Iperf is a tool to measure achievable throughput along a path. In case of UDP, iperf also reports delay jitter and datagram loss.</td>
</tr>
<tr>
<td>Sting</td>
<td>A tool to accurately measure the packet loss rate on both the forward and reverse paths between a pair of hosts</td>
</tr>
<tr>
<td>Badabing</td>
<td>A tool to measure packet loss frequency and duration</td>
</tr>
<tr>
<td>capprobe, pathrate</td>
<td>End-to-end capacity estimation tools based on Packet Pair / Train technique</td>
</tr>
<tr>
<td>pathload, pathchirp, yaz</td>
<td>End-to-end available bandwidth estimation tools based on Probe Rate Model</td>
</tr>
<tr>
<td>spruce</td>
<td>End-to-end available bandwidth estimation tools based on Probe Gap Model</td>
</tr>
<tr>
<td>ShaperProbe</td>
<td>A tool to detect traffic shaping nodes along a path</td>
</tr>
</tbody>
</table>

**Table 1: Survey of typical network measurement and management tools**

Table 2 lists the network measurement tools along with their measurement point definitions and manipulated source code files that so far have been integrated with the OMF framework.
<table>
<thead>
<tr>
<th>Tool</th>
<th>Measurement Point</th>
<th>Measurement Point Metrics</th>
<th>Modified source files</th>
</tr>
</thead>
<tbody>
<tr>
<td>httpperf</td>
<td>connection</td>
<td>• connrate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• conntime</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• connlength</td>
<td></td>
</tr>
<tr>
<td></td>
<td>request</td>
<td>• reqrate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• reqsize</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reply</td>
<td>• reprate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• reptime</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• repsize</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• repstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>load</td>
<td>• cputime</td>
<td>• httpperf.c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• netio</td>
<td>• stat/basic.c</td>
</tr>
<tr>
<td></td>
<td>errors</td>
<td>• errtotal</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• clienttimmo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• sockettimmo</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• connrefused</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• connreset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fdinfo</td>
<td>• fdunavail</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• addrunavail</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ftabfull</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• other</td>
<td></td>
</tr>
<tr>
<td>pathload</td>
<td>pathload_in</td>
<td>• avbw_min</td>
<td>• pathload_rcv.c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• avbw_max</td>
<td>• pathload_rcv_func.c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• latency</td>
<td></td>
</tr>
<tr>
<td>pathchirp</td>
<td>pathchirp_in</td>
<td>• avbw</td>
<td>• pathchirp_run.c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• timestamp</td>
<td></td>
</tr>
<tr>
<td>ifconfig</td>
<td>ifconfigstat</td>
<td>• hw_addr</td>
<td>no source code manipulation (use of wrapper definition around the ifconfig command)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ip_addr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ipv6_addr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• bcast</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• mask</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Overview of network management tools that have so far been integrated with the OMF framework

5.3.2. INTEGRATING THIRD-PARTY TOOLS WITH OMF

The task of integrating third-party applications with OMF consist of three steps to be followed:

- Step 1: Creating the measurement point definitions inside the source code of the application
5.3.2.1. CREATING THE MEASUREMENT POINT DEFINITIONS

This section presents how a third-party application can be integrated into the OMF measurement library. For this purpose, we use the pathload tool to describe the major stages to be taken. Pathload is a tool which estimates end-to-end available bandwidth and converges to a range of available bandwidth that is defined by a lower and upper bound.

On a practical point of view, a user can run this tool by starting a measurement server and client. Then, the final estimated available bandwidth range (defined by the lower and upper bound) will be displayed on the receiver side. To record this range as the final estimation result, it should be sent to the so-called OML server instead of the standard out. To achieve this goal, the accomplishment of a total of 6 steps is needed:

1. Include the oml2/omlc.h header file in the source file and declare a measurement point object
2. Call omlc_init(), passing in the command line arguments from main().
3. Add measurement point(s) using omlc_add_mp().
4. Call omlc_start() to start the measurement collection process.
5. Call omlc_inject() whenever a measurement sample should be recorded.
6. Call omlc_close() when the program has finished all measurement collection activities.

Each of these steps is described in more detail below.

Step 1: Definition of Pathload’s Measurement Point

As mentioned above, Pathload provides the current estimated available bandwidth in form of a range which is defined by a lower and an upper bound:

- \( \text{avbw\_min} \) (lower bound)
- \( \text{avbw\_max} \) (upper bound)

To store this information on the OML server, we define the measurement point (MP) \text{pathload\_in} which is declared in the file pathload.rcv_func.c as follows:

```c
#include <oml2/omlc.h>
OmlMP* pathload_in;
```

After that, the MP needs to be associated with a MP definition:

7 OML is the abbreviation for OMF M\underline{e}as\underline{u}rement Library. The OML server provides the service which stores the customized measurements in a database.
static OmlMPDef pathload_mp_def[] = {
    {"avbw_min", OML_DOUBLE_VALUE},
    {"avbw_max", OML_DOUBLE_VALUE},
    {NULL, (OmlValueT)0}
};

This MP definition named 'pathload_mp_def' declares the two variables avbw_min and avbw_max which are of type double. The last element of the array is a sentinel that marks the end of the MP definition. It must always be included as shown. Note that up to that point in time, both the MP and MP definition are only declared but not created or associated yet.

Step 2: Starting OML: omlc_init()

As a next step, we need to start the OML collection thread inside the pathload tool. This has been done inside the pathload_rcv.c file as follows:

omlc_init("pathload", &argc, (const char**) argv, NULL);

The first argument is the name of the application. The second and third arguments should come from the argc and argv arguments of the C main() function. The last argument is for custom logging and can be set to NULL for most applications (OML uses its own internal logging functions in that case).

Step 3: Creating and Associating MP with its MP Definition: omlc_add_mp()

After the OML is initialized, we now need to create the MP 'pathload_in' based on its corresponding MP definition. To register an MP with OML, the omlc_add_mp() function is used:

OmlMP* mp = omlc_add_mp("pathload_in", pathload_mp_def);
if (mp == NULL) {
    fprintf (stderr, "Error: could not register Measurement Point "pathload_in\n"));
    exit (1);
}

If omlc_add_mp() succeeds, then it returns a pointer to the MP object created, which is used in subsequent calls to omlc_inject(). Multiple MP's can be defined in an application by calling omlc_add_mp() multiple times. Each time it is called, it will return a pointer to a different, unique MP object.

Step 4: Starting OML measurement collection: omlc_start()

Once the MP is defined and created, the measurement can be started via omlc_start() function:
result = omlc_start();
if (result == -1) {
    fprintf (stderr, "Error starting up OML measurement streams\n");
    exit (1);
}

If there was some problem with the configuration, omlc_start() will return -1, indicating a failure. If omlc_start() fails, no measurement can be performed, because the necessary data destinations will not have been set up correctly.

**Step 5: Recording measurements: omlc_inject()**
If the omlc_start() function is successful, the omlc_inject() function can be called to record the pathload tool’s measurements. The function accepts an MP object as its first argument - this is the MP into which the measurement will be injected. It should be obtained from a call to omlc_add_mp() function (see stage 4). The second argument is an array of OmlValueU objects that contains the pathload metrics to be recorded (i.e. avbw_min and avbw_max). An OmlValueU is an untyped representation of a value. Each element of the array should have the same type as the corresponding element of the OmlMPDef used to create the MP.

As the final step, in addition to the display on the standard out, we inject pathload metrics using the omlc_inject() function to the OML server.

```c
OmlValueU v[2];
omlc_set_double(avbw_min, value referencing to pathload's lower bound estimate);
omlc_set_double(avbw_max, value referencing to pathload's upper bound estimate);

omlc_inject(oml_mp, v);
```

**Step 6: Closing OML: omlc_close()**
Once all measurement activities are finished, the final omlc_close() function can be called to shut down the measurement system. This should be done before exiting the pathload tool. It ensures that all measurement streams are flushed and all measurements are recorded (or at least sent to the file or server).

### 5.3.2.2. DEVELOPING THE APPLICATION DEFINITIONS
To be able to use a third party application in an OMF experiment, declaration of a corresponding OMF application definition is needed.

An application definition can be viewed as an interface that makes the OMF Experiment Controller aware of the application. More specifically, it defines which parameters it accepts, which measurement points it provides, or where to find the application to be executed.
The following section discusses how to write an OMF application definition for the pathload tool and how to use it within an experiment description. Since pathload is a both-end tool that is deployed on both the sender and receiver hosts, we call its sender component as pathload_snd and the receiver component as pathload_rcv.

Listing 5.1: Application definition for the receiver component of the pathload tool

```
- defApplication('pathload-rcv-app', 'pathload_rcv') {app

app.path = '/usr/bin/pathload_rcv'
app.version(1, 3, 2)
app.shortDescription = "End-to-end available bandwidth estimation tool"
app.description = "Pathload is based on the rationale that the one-way delays of a periodic packet stream show increasing trend when the stream rate is larger than the avail-bw. The measurement algorithm is iterative and it requires the cooperation of both the sender and the receiver."

app.defProperty('ip_of_sender', 'ip of sender', '-s', {type => :string, :dynamic => false})
app.defProperty('res', 'user specified bw resolution', '-r', {type => :integer, :dynamic => false})
app.defProperty('log', 'write log in user specified file [default is pathload.log]', '-l', {type => :string, :dynamic => false})
app.defProperty('verb', 'verb mode', '-v', {type => :integer, :dynamic => false})
app.defProperty('quick', 'quick mode', '-q', {type => :integer, :dynamic => false})

- app.defMeasurement('pathload_in') { mp
  mp.defMetric('avbw_min', Float)
  mp.defMetric('avbw_max', Float)

}
```

Listing 5.1. is an example of a valid application definition for the pathload application at the receiver side.

**Line 1** declares the URI and name for this OMF application definition. The URI shows where the application definition resides. This application definition is saved in a file 'pathload-rcv-app-rb', which is located in the same directory as the experiment description that is using this application definition. The human-readable name of the application definition is 'pathload_rcv'.

**Line 3** declares the absolute path where the application resides on the target resource. Here, the application pathload is located in /usr/bin/pathload_rcv on the receiver node.

**Lines 4-9** show more information about the pathload tool including its version and descriptions.

**Lines 12-17** declare the properties of the pathload tool. These are the command-line parameters of the tool, that can be to set within the experiment description.
Lines 21-24 declare an OML Measurement Point (MP) and the associated metrics that the application is providing. These are the MP and metrics that were previously added to the application source code described in the previous section. This step is optional if the application doesn’t provide measurements.

Since pathload is a both-end tool requiring the cooperation of both the sender and receiver components, an additional application definition should be written for the sender component of this tool. Listing 5.2 shows a valid application definition similar to Listing 5.1. However, since the actual measurements are only performed by the receiver component, the application definition file for this sender component doesn’t provide any declarations of measurement points. Its main purpose is to generate carefully scheduled packets and send them to its corresponding receiver component.

| 1 | def Application('pathload-end-app', 'pathload_end') { app |
| 2 |   app.path = "/usr/bin/pathload_end" |
| 3 |   app.version(1, 3, 2) |
| 4 |   app.shortDescription = "End-to-end available bandwidth estimation tool" |
| 5 |   app.description = "Pathload is based on the rationale that the one-way delays of a periodic packet stream show increasing trend when the stream rate is larger than the avail-bw. The measurement algorithm is iterative and it requires the cooperation of both the sender and the receiver."
| 6 |   app.defineProperty('imode', 'iterative mode', '-i', {type => :boolean, :dynamic => false}) |
| 7 |   app.defineProperty('omode', 'quiet mode', '-q', {type => :boolean, :dynamic => false}) |
| 8 | }

Listing 5.2: Application definition for the sender component of the pathload tool

5.3.2.3. USING THE APPLICATION IN OMF EXPERIMENTS
To run an experiment with OMF, it should be firstly described into an experiment description file. It contains detailed descriptions of the resources involved in an experiment and the sets of actions to perform in order to realize that experiment. An experiment description is written using the OMF Experiment Description Language (OEDL).

This section describes how to use the pathload tool in an OMF experiment, via its application definition described in the previous section. Listing 5.3 shows a simple pathload experiment description. The scenario consists of a communication between a sender (10.0.0.201/24) and a receiver (10.0.2.202) that are connected via two intermediate routers. Additionally, a separate host running both the XMPP server and the OMF Experiment Controller is connected to the experiment network that can configure and manage each of the existing resources involved in this experiment. The sender and the modified receiver components of the pathload tool were installed on the sender and receiver hosts respectively. The experiment description in listing 5.3 is supplied as an input to the OMF Experiment Controller that finally coordinates and performs the entire experiment.
D4.1 Experimental research on testbeds for community networks (year 1)

Listing 5.3: A simple pathload experiment description

```ruby
- defGroup('Sender', "omf.nicta.node1") do |node|
  - node.addApplication("pathload-snd-app") do |app|
    app.setProperty('imode', true)
  end
end

- defGroup('Receiver', "omf.nicta.node2") do |node|
  - node.addApplication("pathload-rcv-app") do |app|
    app.setProperty('addr', '10.0.0.201')
    app.measure('pathload_out', :samples => 1)
  end
end

- defEvent(:MY_OWN_EVENT) do |event|
  app_status = allGroups.state("apps/app/status/@value")
  event.fire if allEqual(app_status, "DONE_OK") or oneEqual(app_status, "DONE_ERROR")
end

- onEvent(:ALL_UP_AND_INSTALLED) do |node|
  info "This is an easy pathload experiment"
  group('Sender').startApplications
  group('Receiver').startApplications
  info "All my Applications are started now..."
end

- onEvent(:MY_OWN_EVENT) do |event|
  allGroups.stopApplications
  info "All my Applications are stopped now."
  Experiment.done
end
```

Listing 5.3: A simple pathload experiment description

**Line 1:** A new group of resources, called Sender is defined. This group includes a unique node, which is identified by a unique id omf.nicta.node1.

**Lines 2-4:** This group is associated with the existing application called pathload_snd. This tool will be run on each node of this group (here, only on omf.nicta.node1) in iterative mode.
We define a similar group, called Receiver, which will only include node omf.nicta.node2 running the existing application pathload_rcv that performs the actual measurements. Finally, we request the data from the udp_in Measurement Points to be captured.

Usually, the duration of an OMF experiment will be explicitly customized in the experiment description file. However, since the pathload tool is based on an iterative algorithm that converges to a final estimate depending on the current network conditions, its exact duration cannot be anticipated in advance. Thus, for the parallel termination of the pathload and the experiment description executions, we use event-driven mechanisms that are provided by the OMF framework. More specifically, the Experiment Controller holds and updates an XML tree of states about the running experiments and the involved resources that can be periodically observed and caught via event-driven mechanisms. Examples of some experiment states are:

- start of an application
- termination of an application with success
- termination of an application with failure

Fortunately, OMF allows querying such states during the experiment runtime. Thus, using an event-driven mechanism, we periodically check whether the pathload execution has finished and trigger a corresponding event which then terminates the entire experiment.

The new event named :MY_OWN_EVENT is defined. This event is fired if and only if all applications involved in the experiment description (in this case, only the pathload tool) have successfully finished their executions or at least one of applications has reported failure.

Once the defined event named :MY_OWN_EVENT is triggered, all applications are stopped and the experiment is terminated.

5.3.3. SURVEY OF TYPICAL TESTBED SET-UPS AND SCENARIOS FOR EVALUATING NETWORK MEASUREMENT TOOLS

In addition to the provided management and measurement tool templates, we also provide sample experiment descriptions that can realize representative testbed set-ups and scenarios from related literature [GOL2010][MAN2003][HU2003]. Below we present such two different testbed scenarios.

In the following section, we distinguish two types of traffic: probe traffic and artificial cross traffic. The former will be generated by the researcher’s tool/application to be evaluated. The latter will be generated by cross traffic tools provided by our templates.

Testbed 1: Path-persistent cross traffic scenario

The first case considers a path-persistent cross traffic scenario, meaning that the cross traffic generator components will only be deployed on both end hosts and their generated cross traffic will be transmitted over the same path to be probed.
The experiment description to be provided for this testbed will include

- the node definitions for S, R, Cs, Cr, R1 and R2 as well as their network configurations including routing entries as illustrated in Figure 5.2
- the application definition files of the cross traffic generators on nodes Cs and Cr as well as their parametrization
- regulation of parallel experiment executions (i.e. starting the cross traffic generator and the probing tool concurrently) and their synchronous termination.

In the provided experiment description file, the researcher should only configure

- the names of the research nodes to be used for this experiment
- the settings of the cross traffic configuration (rate, packet size, distribution etc.)
- and finally the settings of their own measurement tool/application they want to evaluate.

**Testbed 2: Hop-persistent cross traffic scenario**

The second test scenario considers hop-persistent cross traffic behaviour. As in the first case, the probing traffic traverses the entire path between sender and receiver. However, as compared to first scenario, for each hop, a separate cross traffic generator with a different burstiness degree and behaviour can be defined.
S, R
Sender and receiver

C1s, C1r
First pair of cross traffic sender and receiver

C2s, C2r
Second pair of cross traffic sender and receiver

C3s, C3r
Third pair of cross traffic sender and receiver

R1, R2, R3, R4
Router 1, 2, 3 and 4

**Figure 5.3: Hop-persistent cross traffic scenario**

The experiment description to be provided for this testbed will include
- the node definitions for S, R, C1s, C1r, C2s, C2r, C3s, C3r, R1, R2, R3 and R4 as well as their network configurations including routing entries as illustrated in Figure 5.3
- the application definition files of the cross traffic generators on nodes C1s, C1r, C2s, C2r, C3s and C3r as well as its parametrization
- regulation of parallel experiment executions (for starting the cross traffic generators and the probing tool at the same time) and synchronous termination.

In the provided experiment description file, the researcher should only configure
- the names of the research nodes to be used for this experiment
- the settings of the cross traffic configuration (rate, packet size etc.)
- and finally the settings of their own measurement tool/application they want to evaluate.

As a final remark, to be able to execute such provided template scripts, the researchers should ensure that
D4.1 Experimental research on testbeds for community networks (year 1)

- the OMF-enabled research network to be used includes at least the same number of research nodes that are involved in the experiment scenario as defined in the template script
- and in case of wireless nodes, that they are not out of range.

The aim objective of CONFINE project is to provide an experimentation infrastructure to allow researchers around the world to deploy and run their research inside a federation of Wireless Community Networks. As a part of the work did during this first year, CONFINE has developed a set of software and hardware solutions to build this experimentation testbed. In absence of real researchers who can use the testbed and give us feedback, we have created a base experiment - named "Bittorrent experiment" - to test and evaluate the CONFINE’s current development.
6. OPEN DATA SETS

6.1. DATA SET TYPES

Open data present an interesting opportunity to external researchers to learn more about community networks, e.g. to simulate realistic networks. During our meeting in Athens, five different types of potentially interesting open data were considered, and a number of questions were identified. These questions are still open for discussion, as work on this task is only expected to start in month M13.

After obtaining a data, the results also have to be “polished”: normalised, and when possible correlated. This highly depends on the collected data, and might involve the development of additional data processing and correlation tools. While researchers using the data sets will have to perform similar tasks, especially correlation might require action from our side. This is also still open for discussion.

In what follows we give an overview of the five identified types of data sets.

6.1.1. FLOW DATA

Having access to all packets in a network creates a large source of information with respect to the network usage. While this is an excellent source of information, immediately a number of remarks have to be made as this is very privacy sensitive data:

• to protect privacy, only the headers of packets can be exposed, we only publish flow data never payloads
• anonymization should be applied to the headers
• this is something which the community members have to agree on, we cannot just start sniffing
• While some tools exist to anonymise netflow data, providing these data sets will require a lot of preparation.

Open questions:
• what is permissible for each community?
• how do we notify and get agreement from participating community members?
• which tools do we use to anonymise data?
• how do we handle logging data in a live network, to avoid logging the logs?

6.1.2. NETWORK TOPOLOGY DATA

The topology of community network is very interesting for network researchers. It also offers a good source of information about real-world protocol deployments. This information is less privacy sensitive, although care should be taken. Again, tools to gather this information continuously have to be looked for. As this highly depends on the specific community network, this task will involve cooperation from each network which wants to give this information.

Open questions:
D4.1 Experimental research on testbeds for community networks (year 1)

- does each community network want to give this information?
- which information is available?
- which tools will we use?

6.1.3. NODE INFORMATION

Each community network maintains a database of nodes (WIND DB, a Drupal node DB, ...), which is currently being standardised in the commonNodeDB. This information is static, but easily accessible and a good candidate for correlation.

Open questions:
- are there possible privacy issues with this data?
- is it straightforward to convert from one DB format to that of another DB?
- can this data be released?

6.1.4. LINK LAYER INFORMATION

The integration of DLEP in our software will allow us to easily collect L2/L1 information. This can be very helpful to analyse routing protocols and estimate link quality. This information is strongly related to the topology data, and should preferably be collected simultaneously.

Open questions:
- which tool do we use to continuously record DLEP information?
- can any privacy implications be expected from this data?
- do the community networks mind if this information is released?

6.1.5. RADIO PLANNING DATA

Some community networks, e.g. FunkFeuer, use radio planning software like RadioMobile to predict attenuation between different radio nodes. Together with the DLEP information, this could be valuable information to estimate link quality and assess both radio planning software quality and hardware quality.

Note: this depends on the quality of the propagation model and the input parameter sets (e.g. ground reflection coefficient, ...) used to configure the model, which might require additional tuning and information.

Open questions:
- this is data created when planning, is this easily available afterwards?
- is referring researchers to the radio planning tools sufficient?

6.2. RESEARCH ON THE TOPOLOGY OF COMMUNITY NETWORKS

Although work on the open data still has to start, one network (Guifi.net) already offers easily available open data. Therefore, we already started using this data to start initial open data research.
Community Wireless Networks (CWN) have generated a great expectations in recent years, as a promise of low-cost and participatory connectivity solutions for citizens particularly useful in under-developed countries or isolated areas that are builded, managed, organized and owned by the same people who is using them: community users. A distinguishing characteristic of these initiatives, caused by their openness nature, is that the network topology has been growing organically, without a strictly planned deployment or any consideration other than connecting nodes from new participants linking to an existing one. This deployment flexibility has been helped community wireless networks to grown, but may not handle other important factors that greatly affect the quality of the connectivity, network stability or failures.

One of the main objectives of CONFINE project is to promote the growth of such community networks of two different ways: through their federation with other CWN's and helping them with additional nodes, network interfaces, links and computing resources. In order to fulfil the second objective of the most useful possible way, it is important to understand the characteristics, benefits, problems and needs of community networks topological structure and main services. To do so, the work in [CER2012a, CER2012b, DCN2012] have been done inside the CONFINE project to analyse the topology properties of guifi.net. In this section are presented some selected results developed in this work.

Guifi.net [OZB2010] is a neutral, govern independent Wi-Fi community mesh network majority placed on Catalonia (Spain) with more than 18,000 operational nodes and more than 30,000 km of links, perhaps the world's largest and still with an exponential growth. As a result of its organic growth, its openness and the pragmatic attitude of its members to try out inexpensive solution that does the job, Guifi.net has a plethora of heterogeneous wireless devices and, in consequence, a large diversity of routing protocols being used, including infrastructure and MANET routing protocols.

6.2.1. GUIFI.NET CNML OPEN DATA SET

The analysis carried out in this section is based on the information provided by Guifi.net portal in Community Networks Markup Language (CNML). CNML is an XML specification designed inside Guifi.net to document the network. Figure 6.15 shows the XML-elements of the CNML tree, and Figure 6.3 their attributes.
The attributes that were used to produce the results presented in this section are marked in bold in figure 6.3. The meaning of these attributes is the following (he notation element@attribute is used):

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>cml</td>
<td>generated, server.id, server.net, version</td>
</tr>
<tr>
<td>class</td>
<td>mapping, network.description</td>
</tr>
<tr>
<td>network</td>
<td>ap, cli, devices, links, nodes, services</td>
</tr>
<tr>
<td>zone</td>
<td>access, points, box, clients, created, devices, dns, servers, graph, server, id, links, mhp, servers, parent, id, services, time, zone, title, updated, zone, nodes</td>
</tr>
<tr>
<td>node</td>
<td>access, points, antenna, allocation, clients, created, devices, graph, server, id, lat, links, lon, services, status, title, updated</td>
</tr>
<tr>
<td>device</td>
<td>created, firmware, graph, server, id, name, sinit, index, status, title, type, trans, radio, antenna, channel, antenna, gain, channel, device, id, id, node, protocol, sinit, index, sinit, name, ssid</td>
</tr>
<tr>
<td>service</td>
<td>created, id, status, title, type, updated</td>
</tr>
<tr>
<td>interface</td>
<td>id, ip, mac, mask, type</td>
</tr>
<tr>
<td>link</td>
<td>id, link, status, link, type, linked, device, id, linked, interface, id, linked, node, id</td>
</tr>
</tbody>
</table>

Figure 6.3: CNML elements’ attributes

- **node@id**: node unique identifier.
- **node@lat,lon**: node’s geographical position (latitude and longitude).
- **node@status**: Nodes can be in one of the status: Building, Planned, Reserved, Testing, Working.
Inside CNML the information is arranged using the geographical zones in which the network is organized. Guifi.net zones are initiated by users installing Guifi.net software and interconnecting their nodes. CNML zones are organized hierarchically. The root zone is named Guifi.net World. Each zone can be divided in other zones that cover smaller geographical areas. We shall refer to each division as levels, being the root zone level 1. In this section we have considered the zones up to level 5. From them, we have formed the topology graphs, and we have chosen the 15 leafs zones having the graphs with the largest number of nodes. Figure 6.15 shows the zone tree that resulted following this approach. In the analysis we have also considered Catalunya level 4 zone. This allows studying whether the aggregated zones keep the same characteristics as the leafs. We have considered Catalunya because most of its leafs form a connected graph. At lower levels there are many disconnected clusters, which are connected through the Internet, but not using links reported in the CNML file. The zones analysed in this section are shown in Figure 6.15 numbered in decreasing order of their number of nodes (e.g. Osona is the level 5 zone having the largest number of nodes).

In order to build the graphs only nodes marked in Working status and having one or more links with the link@linked_node_id attribute pointing to another node in the zone are considered. Additionally, we have discarded disconnected nodes, thus, all graphs considered are connected. All links are bidirectional, thus, we use undirected graphs.

The results presented in this section have been obtained using the CNML file downloaded from Guifi.net\(^8\) on 19th April 2012.

This file can be downloaded here:
https://wiki.confine-project.eu/_media/guifinetcnml:guifinet-cnml-detail-world-2012-4-19.xml.gz

6.2.2. TOPOLOGICAL PROPERTIES

In this section are analysed some topological properties of the selected group of Guifi.net zones shown in Figure 6.15. For each zone we have considered the base-graph and the core-graph, which are defined as follows:

- Base-graph: Original graph.
- Core-graph: Graph obtained upon removing the terminals in the base-graph (i.e. nodes in the base-graph with degree 1).

We shall refer by hidden-terminals to the terminals of the base-graph. We study the core-graph and hidden-terminals because, as will become apparent in the following analysis, we will construct a Guifi.net topology by: (1) build the core-graph, and (2) attach to core-graphs nodes the hidden-terminals.

As in the well known three Faloutsos paper [FFF1999], we investigate the power law properties of the graphs. To do so we have considered the following parameters:

- Rank: Ordinal number of the nodes in decreasing order of degree.
- Hops: Total number of pairs of nodes having less than or equal number of hops, including self-pairs (i.e. every node is considered to be 1 hop of itself), and counting all other pairs twice (i.e. the hops between two different nodes is counted twice). The number of hops of a pair of nodes is given by the shortest path first between them.

\(^8\)http://guifi.net/en/guifi/cnml/3671/detail
Figure 6.4(a) summarizes the number of nodes, links, and degree of base-graph and core-graph. For the degree it is given the minimum, mean and maximum values. Figure 6.4(b) gives the values of the main parameters obtained for these graphs, namely: rank and hops exponents of the power law fitting of base and core graphs, and the parameters of the gamma distribution fitting of the number of hidden terminals. For each fitting figure 6.4(b) gives the sample correlation coefficient (\(\rho\)).

The last rows of Figure 6.4(b) give the sample mean (\(\mu\)), its 95% confidence interval (\(c_i\)), the standard deviation (\(\sigma\)) and the coefficient of variation (\(c_r = \sigma/\mu\)) of the values corresponding to the leaf zones (all but 1: Catalunya) in the respective columns.

We have first obtained the following graphs for Catalunya zone:

- Figures 6.5 and 6.6 shows the base and core graphs.
- Figure 6.9 shows the rank log10-log10 plots of base and core graphs.
- Figure 6.11 shows the hops count log10-log10 plots of base and core graphs.
- Figure 6.13 shows: (a) the hidden terminals CECDF semi-log10 plots; (b) the average number of hidden- terminals vs core-degree.
Figure 6.5: Base graph of Catalunya zone. Axis are in km.

Figure 6.6: Core graph of Catalunya zone. Axis are in km.

Figure 6.7: Rank log10-log10 plot of base graph of Catalunya zone

Figure 6.8: Rank log10-log10 plot of core graph of Catalunya zone
In Figure 6.7 the nodes are placed in their geographical position, and the axis give the distances in km with respect of the origin of coordinates. The boxes shown in the figure correspond to the Catalunya leaf zones listed in Figure 6.2. Figure 6.7 shows that zones 5 and 13 are empty. This is because in the CNML file there were not reported links between these zones and the others. Therefore, they were not included in the graph, in order to keep it connected.

For the leaf zones listed in Figure 6.7 there were obtained figures in line with those obtained for Catalunya zone. For the sake of space, there were not included in the deliverable (see [CER2012b] for more details).

As shown in Figure 6.4(a), the number of nodes of the zones analysed varies more than an order of magnitude: from 161 to 6,112. Additionally, Figure 6.4(b) shows that base-graph of all zones, except zone 14 (Barcelonés), are not well fitted by a power-law. In fact, the correlation coefficient (ρ) is in the range [0.63, 0.88] for these zones, while it is 0.94 for Barcelonés. The rank plot shown
in Figure 6.9 depicts this deviation from the power law fitting. We note that all zones, but Barcelonés, correspond to rural areas.

Regarding the core-graph, Figure 6.4(b) shows that it is indeed much better fitted by a power-law: with the exception of zone 15, which has $\rho = 0.84$, $\rho$ is now in the range [0.90, 0.96]. Nevertheless, even for zone 15 the power law fitting improves in the core with respect the base graph. Additionally, zone 15 has a few number of nodes (170), and thus, cannot be considered very representative.

It is interesting that the rank exponent of the core-graphs is $-0.64$ with a coefficient of variation of only 15%. This fact suggests that a rank exponent around $-0.64$ is a characteristic of this type of graphs. Regarding the hop exponent, Figure 6.4(b) shows that the power law approximation is very accurate ($\rho \approx 1$ in all cases). Figure 6.11 shows the goodness of this fitting. However, Figure 6.4(b) shows that it has $c_v = 23\%$, thus, having a higher variability than the rank exponent.

Regarding the hidden terminals, Figure 6.4(b) shows the goodness of the gamma distribution fitting ($\rho$ is in the range [0.93, 1]). Figure 6.13(a) gives a pictorial view of this fact. The values of the shape and rate parameter of the gamma distribution vary significantly ($c_v = 0.63$ and $c_v = 1$, respectively). The value $c_v = 1$ for the rate parameter suggests it is exponentially distributed. This is confirmed by the semi-log plot of the CECDF of the rate parameter depicted in Figure 6.14.

Figure 6.1 shows the ECDF of the hidden terminals’ mean of the leaf zones. The figure shows that the distribution is well fitted by a uniform distribution. Additionally, a Pearson’s Chi-squared test of independence of the mean and the rate parameter of the leaf zones gives a p-value equal to 0.23. Thus, the independence hypothesis of these parameters cannot be discarded. This fact suggests the following method to generate the hidden terminals distribution of Guifi.net zones:

1) Choose the mean number of hidden terminals of the zone, $E[r]$, uniformly distributed between 0 and the desired maximum.

2) Choose the rate parameter of the gamma distribution of the hidden terminals distribution of the zone, $\beta$, exponentially distributed with parameter $\lambda = 0.087$.

3) Choose the shape parameter of the gamma distribution of the hidden terminals distribution of the zone as $\alpha = E[r] \beta$.

Comparing the figures of Catalunya zone with the leaf zones, we obtain the same conclusions as for the rural area zones: (1) Upon removing the terminals, the core-graph is well fitted by a power-law. (2) The terminals of the base graph are gamma-distributed. This fact shows that there is some self-similarity in the topology of Guifi.net.
6.2.3. LINK LENGTH DISTRIBUTION

Link length distribution is of interest in a wireless network, due to its strong influence on the signal transmission in the radio channel. We have found that the link length distribution can be approximated by a mixture of 2 exponentials. Let $L$ the complementary CDF of the link length, $X$, then:

$$L(x|\lambda_1, \lambda_2, \theta) = P(X > x) = \theta e^{-\lambda_1 x} + (1 - \theta) e^{-\lambda_2 x}$$

Figure 6.16 shows the semi-log10 plot of the link length distribution of Catalunya zone and its fitting using Equation (1). Figure 6.16 summarizes the parameter fitting. In the table are given the link length means in km: $\mu_1 = 1/\lambda_1$, $\mu_2 = 1/\lambda_2$, $\mu_T = \theta \mu_1 + (1-\theta) \mu_2$. The entries have been organized such that $\mu_1 \leq \mu_2$. Last rows of Figure 6.16 give the sample mean ($\mu$), its 95% confidence interval ($c$), the standard deviation ($\sigma$) and the coefficient of variation ($cv = \sigma/\mu$) of the values corresponding to the leaf zones (all but 1: Catalunya) in the respective columns.
The sample correlation coefficient ($\rho$) in Figure 6.16 shows that the mixture of two exponentials gives a good fitting of the link length distribution. This can also be observed in Figure 6.16, which shows the link length distribution of Catalunya zone. An intuitive explanation of this result could be that links can be grouped in two sets: One set of short links characterizing connection of nodes located in closer geographical areas, for instance, villages in rural zones, or suburbs in Barcelona. Another set formed by longer distance links interconnecting nodes from different groups of short links. For instance, if we consider Catalunya zone (first row in Figure 6.16), we have that 87% of nodes belong to the first group ($\theta = 0.87$), with a mean link length of $\mu_1 = 0.8$ km. The remaining 13% belong to long distance links with mean $\mu_L = 5.1$ km.

Figure 6.16 shows that the mean link lengths, and proportion of links that belong to the group of short and long distance links varies significantly from one zone to another. This is a logical result, since the groups of short and long distance links depends on many factors: population clouds, topography of the area (e.g. the existence of high antennas where many nodes can connect), etc.

### 6.2.4. CONCLUSIONS

The numerical results shown in this section show that: None of the rural zones of Guifi.net is well fitted by a power law. On the other hand, by removing the terminal nodes we obtain a core-graph which is in all cases reasonably well fitted by a power law. Interestingly, in all cases the number of terminal nodes is very well fitted by a gamma distribution. These results validate the topology generator proposed in [CER2012a] for this type of zones. Catalunya zone has been also found to satisfy these properties. This suggests some degree of self-similarity on the topology of Guifi.net.

Finally, we have investigated the link length distribution. Numerical results show that it is well fitted by a mixture of two exponentials.

In the future, we want to extend this analysis to the other community networks in the CONFINE consortium and derive suggestions for the community networks.
7. CONCLUSIONS

This deliverable detailed the current state of work package 4 of the CONFINE project. It is shown that all activities are on track, and that research is progressing well in the defined tasks.

Research on obstacles and limitations in community networks has identified a number of obstacles, related to routing protocols and hardware choice. The routing protocols include a number of cross-layer optimizations based on the radio-to-router protocol DLEP. Improved routing protocols will form a foundation for improved self-management.

To start the best practices of experimentation, a number of prototype experiments were defined and one experiment on BitTorrent was executed, to learn more from the current state of the testbed. Open data set research has already been started, and has led to some topology studies.
## 8. REFERENCES

<table>
<thead>
<tr>
<th>Reference</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Title</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reference Code</td>
<td>Title and Authors</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
9. APPENDIX

9.1. PROTOTYPE EXPERIMENTS

9.1.1. OLSR MULTI-TOPOLOGY, SCALABILITY, HETEROGENEITY

The integration of Multi-Topology Routing (MTR) into OLSRv2. Multi-Topology Routing allows for the definition of several logical topologies within a default topology that contain certain subsets of links. For every topology, separate link costs and a separate forwarding table are maintained. MTR allows for policy based routing (e.g. to define individual forwarding topologies and metrics for different service classes in a DiffServ based QoS-environment). This mechanism is of special importance in a heterogeneous network with multiple link technology. Here, MTR can be used to e.g. to keep video streams from links with low data rates.

Requirements on testbed: The topology must be sufficiently complex with different link characteristics. In addition, traffic classification based on IP-headers must be performed to select the correct forwarding topology.

Feasibility: The experiments should be performed in a separate testbed or with means for the isolation of experiment control and data plane traffic.

Opportunities: Better Quality of Experience for users in heterogeneous, wireless networks by using different topologies and routing metrics for different traffic classes with minimal protocol overhead. In contrast to conventional traffic engineering methods, this approach is not based on manually defining paths and thus still allows for a highly dynamic network.

9.1.2. OVERLAY / PEER-TO-PEER NETWORKING

Overlay networking refers to a network with another, virtual network topology on top of the existing network. Often based on a peer-to-peer like topology, this allows for extensive network topology testing.

Requirements on testbed: the ability to install multiple layer 3 or even layer 2 solutions on a running network.

Feasibility: highly feasible, as this is something community networks already do or try to achieve.

Opportunities: allows the formation of isolated subnetworks, which can be used for experimenting within the community network by its members.

9.1.3. APPLICATION USABILITY EVALUATION

In the context of Quality of Experience research, it is interesting to consider the usability of applications running on top of the CONFINE testbed and community networks in general. Do they function as expected? What about the impact of variable delay and bandwidth?

Requirements on testbed: This kind of experiments needs access to upper layer and some extra mechanisms to retrieve information from the lower layers (networking). Additionally node hosting requires a considerable amount of disk space to store the collected data or a fast processor to analyse it in real-time.

Feasibility: The current implementation of the testbed supports application setup and execution under restricted users permission policies. Transport and networking layer monitoring is allowed.
using third party tools from inside the container. Tools will be provided next year to monitor network and computing resources from outside the containers.

Opportunities: Usability evaluation is not only necessary to perform experimentation on community networks, it is also useful to evaluate the progress of the testbeds and their robustness during the different stages of their development.

9.1.4. APPLICATION OPTIMIZATION

It is well known that applications can and/or should be optimized to fit the network in which they are working, to improve performance. Is this feasible in community networks, as the network may be highly varying? And does the CONFINE testbed abstract some of these network characteristics?

Requirements on testbed: This kind of experiments needs access to the higher layer and some extra mechanisms to retrieve information from the lower layers (networking). Additionally, it should be able to run experiments under different networking and usage conditions.

Feasibility: The current testbed is able to install and execute applications under restricted users permission policies. However, to evaluate application optimization, it will be necessary a minimum the level of networking isolation. That will be provided by the testbed during the next year.

Opportunities: There exist a lot of applications based on network computation and storage located at data centres that can be transformed and optimized to run under community networks. This will save energy, traffic peering and provides extra locality and privacy control to community users.

9.1.5. DELAY TOLERANT NETWORKING

Delay tolerant networking considers networks which are only intermittently connected. This means classical routing protocols can no longer be used, and different solutions have to be studied.

Requirements on testbed: The testbed needs to provide a heterogeneous hardware scenario with a medium set of nodes geographically sparse. Some extra traffic from communities is not strictly necessary, but will help to create real-like network situations.

Feasibility: The current deployment of the testbed includes a set of nodes attached to community network devices, spread along the participant wireless community networks. Routing protocols can be deployed inside research nodes without hindrance.

Opportunities: New protocols able to find the best paths – in terms of quality of service – under wireless conditions are of great significance to improve the quality of experience of wireless community users. The chance to test this with real user's traffic gives an important value to the experimental results.

9.1.6. CONTENT DISTRIBUTION

Content distribution studies the problem of distributing a large amount of data in a large network in an efficient way. The amount of data, the scale of the network, the topology of the network and the definition of efficiency all can have an impact on the resulting optimal protocols which are looked for.

Requirements on testbed: An experiment of such characteristics affects only on the application level, but may require information from the network layer. Given the characteristics of the experiment, expecting a high amount of data to be shared and stored, the node hosting that data requires a considerable amount of disk space.
Feasibility: The current implementation of the testbed supports such experiments. It should be deployed over indoor nodes, which have higher capacity, both in terms of CPU and disk space.

Opportunities: Developing efficient mechanisms for the distribution of contents within the CONFINE context is specially useful for solving issues such as software updates, and for sharing between the nodes new versions of the distribution. It becomes also useful for the deployment of experiments that require a heavy customization of the image or a big amount of data.

9.1.7. TRANSPORT-LAYER OPTIMIZATION

Wireless networks are notoriously less stable than wired networks, and this has a serious impact on transport layer protocols. Especially TCP shows poor performance in wireless networks, as the protocol was designed for wired networks.

Requirements on testbed: Optimization of the transport layer impacts on layer 4, but may require information of the network layer. The consumption in terms of CPU and memory is reduced, so it is feasible to run them on embedded systems.

Feasibility: The current status of the testbed supports Transport Layer Optimization experiments, both within the nodes that are distributed in the community networks as within the CONFINE clouds.

Opportunities: Optimization of the transport layer to support paths with heterogeneous links and equipment are necessary for community networks, since current approaches do not fully support the variability of wireless links, neither coexistence on the same path of links with wired and wireless characteristics.

9.1.8. HYBRID ROUTING IN MESH NETWORKS

Typically, routing protocols are reactive or proactive. This means that they either lookup routes when needed, or they proactively maintain their routing table continuously. In mesh networks, a hybrid approach might prove better to be more suitable with regards to scalability.

Requirements on testbed: a means to run custom routing protocols, in a topology which can be customised or discovered.

Feasibility: highly feasible, as this is something community networks also do and test.

Opportunities: new routing protocols can be tested, in a very realistic environment and on a larger scale.

9.1.9. MULTI-TOPOLOGY ROUTING

Normal routing protocols define a cost metric and create local routing table entries that forward the traffic along the shortest path. Multi-Topology Routing define multiple costs metrics for the same link to create multiple independent routing tables. This allows routing traffic of a certain type (QoS type or IP-version type) to be routed along different paths.

Requirements on testbed: the installation of new and possibly multiple routing protocols, isolated from the current community network routing protocols.

Feasibility: this is feasible, as the testbed allows for new routing protocols to be tested.

Opportunities: multi-topology routing could help the quality of experience of a community network to increase, as high priority traffic can be given more reliable, bandwidth constrained paths.
9.1.10. ROUTING SECURITY

Routing protocols are crucial for the proper functioning of a mesh network. As such, they are also very vulnerable to all kinds of security issues.

Requirements on testbed: the ability to tamper with routing protocols, possibly also while running new routing protocols. An important additional requirement is the isolation and protection of the routing protocols running in the community network, as experiments should not be able to bring down the existing network.

Feasibility: feasible, although additional precautions and checks have to be introduced to ensure the continuous protection of the community network.

Opportunities: even without hostile intents, routing security problems can arise in community networks and other mesh networks. Research on new counter measures is highly beneficial to protect the networks.

9.1.11. LINK LAYER FEEDBACK TO ROUTING PROTOCOLS

Routing protocols are usually designed to work with limited information from the link layer. However, this rich source of information can improve routing protocol performance drastically.

Requirements on testbed: running custom routing protocols and access to link layer information.

Feasibility: feasible, if the link layer can be abstracted or protected. Radio-to-Router protocols can be a good solution to enable this.

Opportunities: improved protocols, which react better to link changes and use information which is usually already available.

9.1.12. MESH ROUTING SCALABILITY

A mesh routing protocol is usually designed for a mesh of tens or at most hundreds of nodes. If we consider larger scales, scalability and performance start to be challenging.

Requirements on testbed: deployment of routing protocols to large-scale networks.

Feasibility: not initially feasible, but later when the number of nodes in the CONFINE testbed will grow.

Opportunities: research on mesh routing protocols is usually done in small-scale lab testbeds, which do not represent the realistic characteristics of a community network. Simulation is also a popular option, but completely misses realism. The CONFINE testbed could present both realism and large-scale, a perfect fit for this experiment.

9.1.13. CROSS LAYER ROUTING METRICS

Related to link layer feedback to routing protocols, both physical layer and even transport layer information can improve routing protocol performance.

Requirements on testbed: access to link layer information when running new routing protocols.

Feasibility: as radio-to-router protocols will be in use, this is feasible.

Opportunities: improvement of routing protocols, where the abstraction of the link layer is removed to allow for better reaction to link layer events.
9.1.14. ROUTING PROTOCOL AUTO CONFIGURATION

A routing protocol needs quite some configuring to tailor the protocol to a specific network. It should be possible to adapt existing protocols to a “plug and mesh” like solution, which use auto-configuration combined with fitting defaults for easy mesh node deployment.

Requirements on testbed: ability to install new routing protocols and to quickly generate scenarios which trigger the auto configuration.

Feasibility: depending on the parameters the routing protocol reacts to, this can be easily or harder to achieve.

Opportunities: auto-configuration of routing protocols increases network reliability, and avoids frequent and costly manual interventions by people in the community networks to solve problems.

9.1.15. SEPARATION OF ROUTER AND RADIO DEVICE

The number of wireless interfaces is limited, even for larger router hardware. Today large mesh nodes can either use expensive hardware (which is still limited in the number of wireless interfaces) or split the node into multiple cheap devices, running a routing protocol on each of them (which is a scalability issue). By combining small devices to be used as wireless interfaces in bridging mode with a central router device, this limitation could be overcome.

Requirements on testbed: presence of small devices which can be combined.

Feasibility: the first versions of these radio-to-router protocols have started to be integrated in M12.

Opportunities: a cheaper combination of devices, which are more powerful and easier to control or customise.

9.1.16. ROUTER POWER CONSUMPTION

Routers use quite some energy when deployed without any optimizations. This can hinder deployment in community networks, where users have to pay the power bill.

Requirements on testbed: ability to measure power consumption by routers.

Feasibility: less feasible, as this involves additional hardware to measure power consumption which is not currently present or budgeted.

Opportunities: better visibility of power consumption, leading to a more green community network and testbed.

9.1.17. POLICY ENFORCED SPECTRUM SHARING

Spectrum sharing is one of the fundamental characteristics of cognitive radio networks. It assures proper spectrum allocation to and among different secondary systems without excessive harmful interference to the primary systems. There are several possibilities solutions for enabling secondary users for sharing the spectrum, one of them is to engage a policy system to decide and help the system to decide on the radio frequency communication parameters.

Requirements on testbed: The experiment requires access to the physical layer of the radio. Ideally, several radio interfaces, with different frequency ranges should be available, but allowing the researcher to switch between different WiFi channels should be enough.

Feasibility: The current status of the testbed does not support this type of experiments. Future iterations, with the integration of commands in the DLEP protocol might support such
experiments. Experiments like this one that require full control of the radio and direct interaction between the deployed slivers need to be run within CONFINE clouds.

Opportunities: Efficient mechanisms for spectrum sharing, which still guarantee primary usage, will allow a much more efficient, dynamic and flexible management of the spectrum.

9.1.18. ACTIVE NETWORK MEASUREMENTS

Active measurements allows a detailed mapping of the capacity and characteristics of network paths in exchange of injecting traffic on the network, which can be disturbing to other users. These active measurements can be performed continuously or can be occasional. It may even be possible to have a mechanism to reduce the impact on other uses by running measurements in periods of low activity by using mechanisms to detect low levels of activity.

Requirements on testbed: Depending on the tools used for network measurements, it may require normal application-level access to transport (e.g. UDP) or privileged access required to lower level traffic such as some access to raw sockets or deal with ICMP traffic.

Feasibility: The current testbed can allow different tools that do not require network access below the transport layer: application-based network measurements. Lower level access may require adjustments on the node distribution and may have disturbing effect on other traffic in the network.

Opportunities: Active network measurements are required to have a detailed mapping of the effective characteristics of community networks. These metrics can be the basis for many experiments as reference values.

9.1.19. PASSIVE NETWORK MEASUREMENTS

Passive measurements is a less intrusive way of characterising a network. They allow to obtain an approximate map and measurement of the capacity and characteristics of network paths without a visible impact on the traffic in the network, contrary to active measurements. These measurements take advantage of another preexisting traffic to measure times based on logs or any other access to traffic provided by a separate application.

Requirements on testbed: The only requirement is the possibility of instrumenting a pre-existing application by taking measurements inside its code or accessing log files. It depends on a mechanism to inject code to take samples or have access to traffic logs.

Feasibility: The current testbed can allow some of these passive network measurements. However the required mechanisms to obtain such information should be enabled, taking care of avoiding any effect on the performance or stability of the base system.

Opportunities: Passive network measurements are helpful to obtain some metrics on network traffic and network performance without introducing any extra load or disturbance. These metrics can help researchers to have reference values, but there are metrics that cannot be obtained with enough quality due to the shared access to the network and the lack of control and the lack of detailed information about the network traffic.
The CONFINE project
November 2012
This document is licensed under the following license:
CC Attribution-Share Alike 3.0 Unported
<http://creativecommons.org/licenses/by-sa/3.0/>
CONFINE-201211-D4.1-1.0