European Commission
Information Society and Media

European Research
7th Framework Programme

COOPERATION

CONNECTED COMMUNITIES
CONFINE

Project title: Community Networks Testbed for the Future Internet.

Tools for experimental research (Year 2)

Deliverable number: D.4.8

Project Acronym: CONFINE
Project Full Title: Community Networks Testbed for the Future Internet.
Type of contract: Large-scale integrating project (IP)
contract No: 288535
Project URL: http://confine-project.eu

Editor:	Bart Braem, iMinds
Deliverable nature:	Report (R)
Dissemination level:	Public (PU)
Contractual Delivery Date:	20/09/2013
Actual Delivery Date	TODO
Suggested Readers:	Project partners, future open call partners
Number of pages:	29
Keywords:	WP4, testbed design, requirement analysis, experimental research, community networks, testbed
Authors:	Bart Braem, iMinds Michael Voorhaen, iMinds Christoph Barz, Fraunhofer FKIE Fatih Abut, Fraunhofer FKIE Henning Rogge, Fraunhofer FKIE Felix Freitag, UPC Roger Baig Viñas, Guifi
Peer review:	Ester Lopez, UPC Blaine Tatum, OPLAN

Abstract

This deliverable gives an overview of the different tools used and made available for experimental research by the CONFINE project consortium, consisting of three parts: open data, benchmarking and best practices.

The chapter on open data introduces the general concept of open data and applies this to the CONFINE project. Next, tools to generate and handle open data are introduced, followed by a description of open data sets made public during year 2 of the project. Chapter two introduces and describes a benchmarking framework. While this is still work in progress, already the proposed approach is given. Finally, the last chapter documents best practices for the community-lab testbed. A best practice to handle network simulation in an emulated testbed is introduced, followed by good practices from the open call partners.

Contents

List of Figures

1. Open Data

This chapter will introduce open data and a strategy followed by the CONFINE project. Next, three methods to generate open data in the CONFINE project will be presented: a tool to map existing community networks, open data sets generated from systems within existing community networks, and finally a tool to generate open data from community-lab experiments based on the DLEP integration.

1.1. Introduction

Open data is described on Wikipedia as "*the idea that certain data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control*". In the scope of CONFINE, we consider open data to be data from the project which is freely and easily available to everyone interested[1]. To this end, the project wants to make it straightforward to obtain, interpret and analyze the data resulting from project efforts as well as project partners.

As a motivation for generating open data, the project would like to refer to open data in its more frequently used government context. Governments and organizations around the world are starting to publish data on policy, resulting in troves of information freely available to developers and users around the world. Famous examples include Data.Gov[2], the European Union Open Data Portal[3] and the World DataBank[4]. Further, research initiatives, such as the measurement lab[5] have begun to release their research data using open data standards. A good starting point for more information on open data around the world is the website of the Open Knowledge Foundation[6], which also publishes lists of data sets and courses on handling open data.

In the scope of CONFINE, open data should be considered as data generated, both from experiments and from the community networks. This closely corresponds to the position of CONFINE in the Future Internet Research and Experimentation Initiative (FIRE): the project performs experiments, and, as such, studies the feasibility of Community Networks as a future internet. The open data from the project will include information from both the experiments themselves and all partnering community network.

1.2. CONFINE Open Data Strategy

The project has outlined a strategy to generate, handle and offer open data, which we will introduce in this section.

1.2.1. Storage

The storage of the open data is also inside or very nearby the source of the data. In the case of community network data, this means the networks themselves host the data[1]. This gives greater sustainability, and simplifies communicating with interested external parties.

[1]This is not yet the case for AWMN, but this is something which we plan to actively work on in the coming months.

To simplify obtaining the data, the project has set up `http://opendata.confine-project.eu/` using the Comprehensive Knowledge Archive Network (CKAN)[7] software. This central catalog points to open data available from the different CONFINE partners. With CKAN, the datasets can be easily tagged and commented on. The project also wants to make sure all data can be easily downloaded in a single step. E.g., we want to avoid requiring separate downloads of weekly data sets. To this end, some of the data set generators are modified to generate larger sets. This is still a work in progress, and will probably continue to be an effort during the project.

To simplify interpreting data, the project wants to add a description and explanation to each dataset available in the data catalog. This will require documenting the different fields of each data set, which we hope will result in more usage of the data as it avoids costly and tedious deciphering of the data formats. Finally, to simplify analysis of the data, the project will point to existing usage of the data sets. By handing examples to the potential users of the open data sets, the barrier to usage can be lowered.

1.2.2. Licensing

A final step in offering data involves taking care of the licensing. While there are numerous solutions, including using no licensing at all, the project has decided to use the Open Data Commons Open Database License (ODbL)[8]. This gives a large freedom to the users of the data, as long as *attribution*, *share-alike* and *keep open* are respected. For more information on this license, we would like to refer to the excellent Open Data Commons website[9].

1.3. A Community Network Mapper

This section gives a rough description of the current work on a community networks information retrieval system called *community network mapper*. The first results of this implementation have been submitted as a paper, which at the time of writing this deliverable was still under review. As a consequence, this section will not go into details. Future deliverables will go into further detail.

1.3.1. Goal

The development of a community network mapper is motivated by a number of goals. The first and foremost is to lower the burden of documenting a network in a node database. Especially when the community network is altered, e.g. moving a node or upgrading its hardware, manually updating the node database is often neglected.

A mapping system that runs at fixed intervals, e.g. daily, can also provide feedback on and document the dynamics of a community network. As a simple example, the growth figures from Guifi.net[10] show how the community network increased during its lifetime. With more data available, e.g. the geographical data of new nodes or the resulting changes in the routing protocols, researchers will be able to study the dynamics of a network. And this could also help community networks to understand how other networks grow and possibly learn from it. Again, this forms interesting feedback to the deployment of community networks as a future internet.

The data generated by a mapper can also be fed into systems that can monitor the community network and provide feedback to the community network. Section 1.3.7 provides results obtained by analyzing the results of our community network mapper.

1.3.2. Approach

In what follows, the implementation of the community network mapper is outlined. It is based on SNMP and the RouterOS API for data retrieval and plain text storage. The mapper consists of four phases: *discovery*, *data extraction*, *grouping* and *result generation*.

1.3.3. Discovery

During the first phase, *discovery*, the mapper tries to discover all devices by their IP address. To achieve this, a starting IP address has to be configured. Starting from this device, a breadth first search algorithm is run. It is based on the neighbors of a device, as exposed by its neighbor discovery protocol. This can be the set of OLSR neighbors[11], IP addresses discovered by the Cisco Discovery Protocol (CDP)[12] or IP addresses discovered by the MikroTik Neighbor Discovery Protocol (MNDP)[13]. After a number of iterations this results in a stable set of discovered IP addresses.

It is important to take care of devices with multiple IP addresses during this phase, because a device can be discovered over multiple addresses. In our implementation, upon discovery we add all IP addresses of a device to the list of discovered network IP addresses. Grouping together all IP addresses from all devices in a node will be performed during the Grouping phase.

1.3.4. Extraction

At the end of the first phase, the discovery phase, all IP addresses in the network are available.

In the second phase, *extraction*, data is extracted from the discovered devices. The data to extract depends on the interests of the community network and/or the researchers operating the mapper. However, in the next phase of our implementation, the interface table and the ARP table are required. More data can be fetched with additional scripts, using SNMP and/or the RouterOS API to query the devices.

1.3.5. Grouping

To group nodes, we want to identify devices in a single location. We assume that devices connected via a wire belong to a single node, and that devices connected over a wireless connection belong to a different node. This assumption holds in the studied community network Wireless Antwerpen. However, this assumption has to be improved for community networks like Guifi.net who roll their own fiber. One could argue that the interface type interface type (Ethernet, IEEE 802.11, virtual interfaces, ...) will simply identify wired connections. However, this information is not available directly nor consistently over SNMP. The authors did not want to use proprietary APIs, to maintain general applicability.

The main complexity for grouping can be observed in figure 1.1, depicting two nodes. From a device point of view, two MikroTik routers are connected over two Ubiquiti antennas. As mentioned before, the Ubiquiti devices do not expose any IP information over SNMP, and have to be discovered via the MikroTik routers. The MikroTik routers discover each other and the Ubiquiti devices via MNDP, as depicted by the dotted line. However, they do not discover any hop count information because the Ubiquiti routers form a layer 2 link. In this case grouping can be considered determining a reduction from six (discovered) neighbor edges to two wired and one wireless connection.

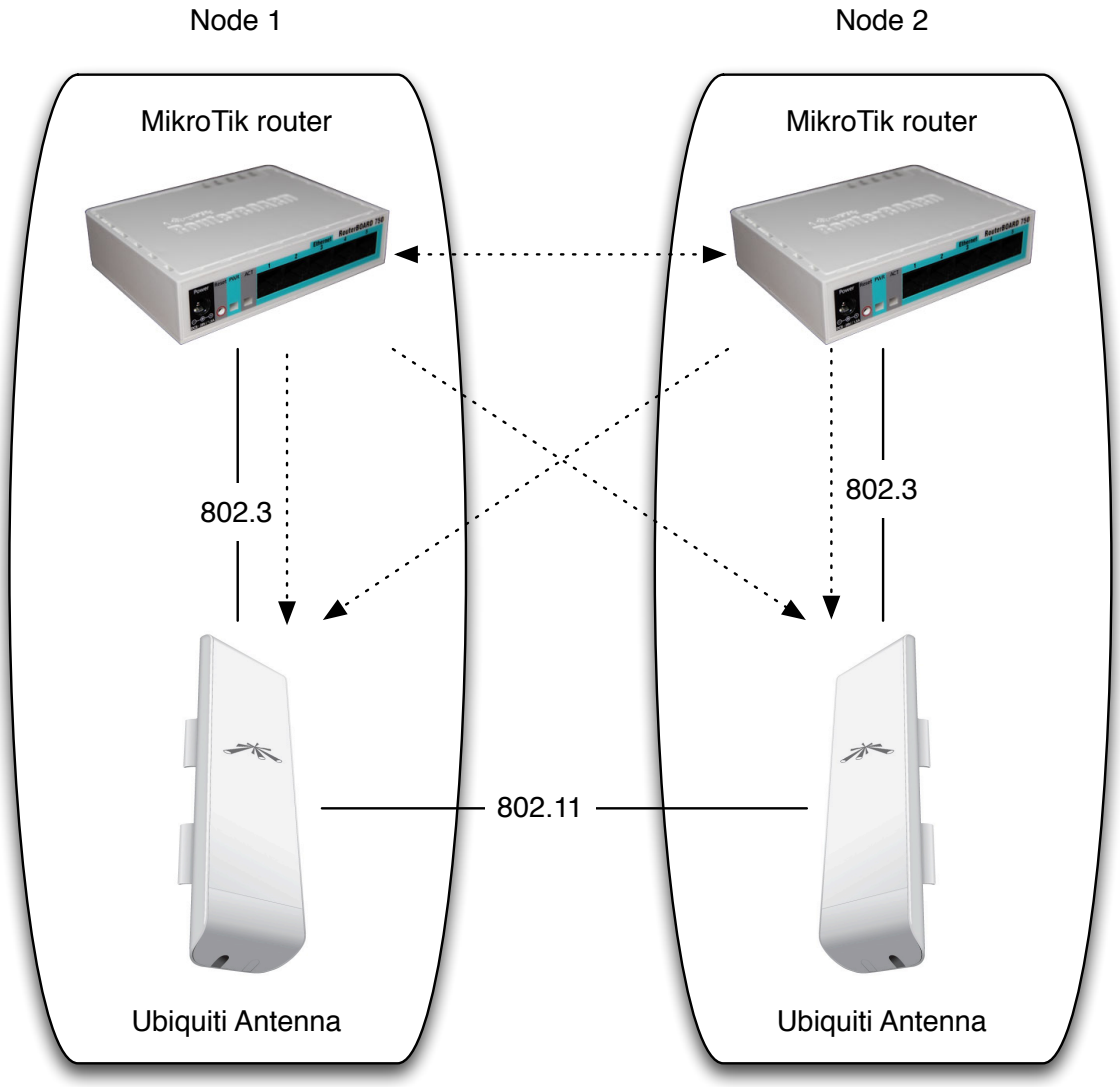

Figure 1.1: Grouping scenario, dotted lines show neighbor discovery.

Finally, we identified wireless connections between grouped nodes. The main challenge in this case comes from the fact that wireless connections are usually handled by devices that act like ISO layer two bridges rather than ISO layer three routers. As such, the devices are not visible at IP level in the data of the access point they are connected to. Identifying these connections can be solved by SNMP, because both the BSSID a node is connected to and the BSSID of the access point are exposed. Combined, we can easily match access points clients with their access point, as outlined in the second part of listing 1.1.

Listing 1.1: Grouping Discovered Results

```
Function Group (discovered devices)
  For each discovered device i (* Group wired links *)
    wired_links(i) := neighbors(i) intersect (specific_routes(i) union arptable(
        i))
  End
  SSIDs = the SSIDs of all discovered devices
  For each discovered device i (* Group wireless links *)
    If (SSID of i is known and SSID of i is in SSIDs)
```

```
      wireless_links(i) := SSID of i
    End
  End
End
```

1.3.6. Result Generation

After discovery, data extraction and grouping, the final phase of the community network mapper is *result generation*. To match the grouping information and the topology data, as generated by the community network mapper, we have decided to output our result in the form of a Graph Exchange XML Format (GEXF) graph[14], annotated with the extracted node and link properties. This XML format allows for easy manipulation in different systems, while being very extensible.

The pseudo code in listing 1.2 describes the discovery and extraction phases of our community network mapper, and refers to the grouping phase shown before.

Listing 1.2: Community Network Mapper: discovery, extraction and grouping phases

```
Function Extract (d_data)
(* Extract data of device d over SNMP or some other API *)
Function Symlink (d_data, e_data)
(* Create symbolic link from data file(s) of device d to data file(s) of device
   e *)
discovered := {startingdevice} (* discovered devices (IP addresses grouped by
   device) *)
previous := {} (* previously discovered devices *)
While (discovered != previous)
  For each i in discovered and not in previous
    Extract(i_data)
    Extract(i_interfaces)
    Extract(i_neighbors)
    For each j in i_interfaces
      Symlink(i_data, j_data)
      Symlink(i_interfaces, j_interfaces)
      Symlink(i_neighbors, j_neighbors)
    End
  End
  previous += discovered
  For each i in discovered
    discovered += i_interfaces + i_neighbours
  End
End
Group(discovered)
```

1.3.7. Results

Currently, we have a working Community Network Mapper which runs three times per day on the Wireless Antwerpen community network. Figure 1.2 gives the current resulting network graph, after node grouping. A number of unconnected nodes can be observed, because the grouping is not optimal at this moment. On average, one entire run takes about three hours to discover about 3000 IP addresses. This can vary depending on the performance of SNMP on the devices and the load on the host system while grouping.

Figure 1.2: Our current graph visualized.

A first lesson learned comes with this result: developing a community network mapper proved to be a difficult task. Tweaking the system to be faster or to include new grouping approaches takes time. Generating and processing all data multiple times can be very demanding for the hardware.

We also noticed a strong need for documentation in community networks, to collect information on which hardware and which software is deployed where in the large network. We hope this can help leverage the cost of documenting. In our tests with Wireless Antwerpen, the differences between our mapped network data and the contents of the node database became clear very quickly. The difference usually originates from the strong growth of this community network. Documenting networks of this scale is hard and as such we believe this work will help community networks.

This is exactly where a community network mapper should become the complementary live documentation of the community network. We would like to take this concept even further in the future, and want to advocate for node databases to contain both static and dynamic information. In these *hybrid* node databases, at fixed intervals the dynamic information is fed back into and compared to the static documentation.

Another interesting result from our Community Network Mapper is the feedback we can provide to the community network. By documenting the live network situation, we can more quickly give feedback on a number of aspects and also use this data. As a nice example, we have been able to generate a list of duplicate IP addresses present in the community network. The community network has been able to verify and correct these problems, which also strengthens our faith in this stage of our development.

A final lesson learned considers the data access, or to be more exact the API. From our work, we believe it will become more important to ask device vendors to enable data access, e.g. via SNMP. As community networks become larger, using a standardized API like SNMP should become the standard way of handling device data. Proprietary interfaces and incomplete SNMP data complicate network maintenance. However, vendors keep resorting to proprietary APIs which have less tool support and are even more subject to change.

1.4. Open Data Anonymization

An important aspect of open data in the context of networking is anonymization. When the data exposed includes information, not only about the network topology and its dynamics, but also about the traffic flowing between the nodes in the network, anonymization of the data transfered and the end points addresses is essential.

In the following subsections, an anonymization approach used in FunkFeuer is outlined.

1.4.1. pycryptopan

CryptoPAn[15] is a well developed and described anonymization algorithm for IPv4 addresses. After anonymization ip addresses retain their network characteristics (e.g. IP addresses that were in one network before, will be in the same network after anonymization). To better integrate this anonymization protocol in existing collection tools, a python implementation of the CryptoPAn algoritm was written by FunkFeuer. It was published both on github[16] and the python package index[17].

This code will further be used to anonymize the FunkFeuer data described in subsection 1.5.1.

1.5. Community Network Open Data

Section 1.3 describes data being generated actively by research tools. However, most community networks already collect a considerable amount of data about their own network. All community network partners in the CONFINE project, following the open data initiative, have agreed to openly offer part of this data within the context of the project. In the following subsections, the data generated by each community network and its characteristics is detailed.

1.5.1. Funkfeuer

The FunkFeuer network started to collect statistics and data on their network at `http://stats.funkfeuer.at`. The site contains multiple data sets all openly licensed.

1.5.1.1. Topology

FunkFeuer is the only community network partner running solely on OLSR as a routing protocol, this allows export the network topology and further statistics as seen by specific nodes in the network. The statistics site also shows a force-directed graph layout of the current topology as an example on how to use the data.

1.5.1.2. Traceroute statistics

Routing in the network is done via OLSR, since the topology will not always reflect current routing decisions, regular traceroutes are run across the network to describe routing information. Comparing this information with the toplology data mentioned above can give insights in how to improve mesh-routing in real-world networks.

1.5.1.3. Ping statistics

Using ping statistics, the FunkFeuer network traces the availability of various nodes. These statistics can help to gauge expected quality of service for end-users in such a network.

1.5.2. Guifi

1.5.2.1. Guifi.net Database

At network description level, all the guifi.net network information is stored in a database. This data is made public using the Community Network Mark Up Language (CNML). For the nodes, the CNML is obtained inserting "/cnml" in the URL after "/guifi", and "/node" at the end (i.e. `http://www.guifi.net/ca/node/17600` → `http://www.guifi.net/ca/guifi/cnml/17600/node`). The zones (groups of nodes or other zones) are also described using CNML. In this case the options are "/zones", for the description of its subzones, "/nodes", for the nodes of the zone, and "/details", for the complete description including devices and their NICs. This information is used by the guifi.net website to perform multiple statistics (`http:`

//www.guifi.net/ca/guifi/menu/stats/nodes) and dynamic representations (http://www.guifi.net/ca/guifi/menu/stats/growthmap).

The network operation information is mainly collected by the graph servers via SNMP by periodically querying their associated devices (each network device is associated to one graph server). Additionally the graph servers also perform availability (ping) tests. All this data is made available publicly (daily, monthly, yearly) using RRD. For instance, the guifi.net website itself uses this information to populate the nodes, zones, etc. pages (http://www.guifi.net/ca/elserrat).

The openness of the network information (specially the presented by the CNML) is essential in a community network because it is the information that allows the understanding and the expansion of the network. Although the guifi.net database has not been yet licensed (it is expected to be licensed soon under the ODbL) de default license of the guifi.net website is CC by NC-SA, and thus, the database can also be considered already implicitly open.

1.5.2.2. Guifi Proxy Logs

In guifi.net, the access to the web is a service available to all participants through most of the internet gateways (mostly ADSL). Proxy technology has become the standard way to make the web service available, for a number of reasons. Due to technical reasons: in guifi.net there are no default routes to the internet and there is uplink scarcity. Proxy technology is also used because of certain restrictions of some of the providers of this service, i.e. public administrations and public services such as libraries and schools. They often require identification with user name or authentication with a password.

For the identification/authentication the guifi.net community has developed a proxy federation system which allows access to all federated proxies (over 300) with the same user name and password pair. The proxy service (Squid) as well as the federation tool are part of the guifi.net GNU/Linux distribution (guinux), the distribution used to set up almost all the guifi.net servers. (The servers, as well as the rest of the infrastructure, are set up and maintained by community members.)

As part of the efforts of guifi.net to make resources available to the research community, during the second year of CONFINE, the logs of several servers have been made available under certain conditions. The servers have been chosen to be as much as representative as possible in terms of population (villages, towns, cities) and usage (heavy, medium, light).

The restrictions, aimed at keeping the privacy of the users are the following:

- Access to the data sets:
  - just for research purposes.
  - accessible with username/password (HTTPS).
- Anonymization
  - the anonymization is done adding strings and hashing. The correspondence between the source string and the resulting hash is kept.
  - user names are always anonymised
  - either connection source information or connection destination information (i.e. IPs and ports) is always anonymised. This anonymisation is done according to the parity of the date of the log.

The main characteristics of the data currently available are:

- Total servers: 8
- (guifi.net) proxies IDs: 10473, 18202, 33596, 5126, 57064, 5735, 7652, 8258

- log format: Squid default
- Number of log lines per day: several millions
- Number of GB of data served by the proxies per day: several GB

The logs of the proxies are centrally stored in a server in the UPC Lab, only available to selected researchers.

1.5.3. AWMN

Aligned with the strategy outlined under "Cross-layer analysis of community networks" in deliverable D4.2, the Athens Wireless Metropolitan Network has made BGP data available as open data, at http://opendata-awmn.confine-project.eu/. This data is currently hosted at a virtual machine server at iMinds, because of uplink constraints. The data sets are comprised of RIB and update messages, logged every 15 minutes on a Quagga BGP daemon running in the AWMN network. Software from iMinds fetches this data and makes it publicly available.

Two goals motivated the publishing of this data. First, to allow external researchers to investigate different BGP data than the regular internet BGP dumps available at e.g. RIPE RIS[18]. Second, because both iMinds and AWMN are interested in performing research of this BGP data, and want to publish the data sets used in the research. This will allow others researchers to verify the performed studies and potentially extend them.

1.6. DLEP-Based Open Data Collection

During the CONFINE project, open data sets will be collected using the Data Link Exchange Protocol (DLEP) and provided to external researchers. This section firstly gives a brief introduction to DLEP and then discusses the proposed DLEP-based open data set collection architecture that uses the cOntrol and Management Framework (OMF) as the experiment controlling software.

1.6.1. Introduction to DLEP

DLEP is a proposed IETF standard for protocol for radio-router communication. The intention is to be able to transport information such as layer 1/2 information from a radio to a router, e.g. over a standard Ethernet link. Depending on the final feature-set the router might even be able to configure certain settings of the radio.

The CONFINE prototype implementation of DLEP based is modular and consists of the following components: dlep-app, nl80211listener plug-in, service plug-in, client plug-in and layer2viewer plug-in (Figure 1.3). At the moment, it is not compliant to the current draft version of DLEP but will be modified accordingly as soon as the protocol definition gets to a mature state.

The *dlep-app* is the daemon itself and can be seen as the core application. It comes with relatively few functionalities on it's own. Most functionalities are implemented by plug-ins. DLEP plug-ins can be categorized in two groups: data provider plug-ins and data consumer plug-ins. Data provider plug-ins gather the data from any kind of source and put them into the database while data consumer plug-ins take the data from the database and process/forwards them.

The *nl80211listener* is a data provider plug-in and uses a Linux netlink socket to query the layer 2 informations from a WIFI card in a regular time interval and store them into a local database. In this local database filled by nl80211listener plug-in, both layer 2 network data (information concerning the

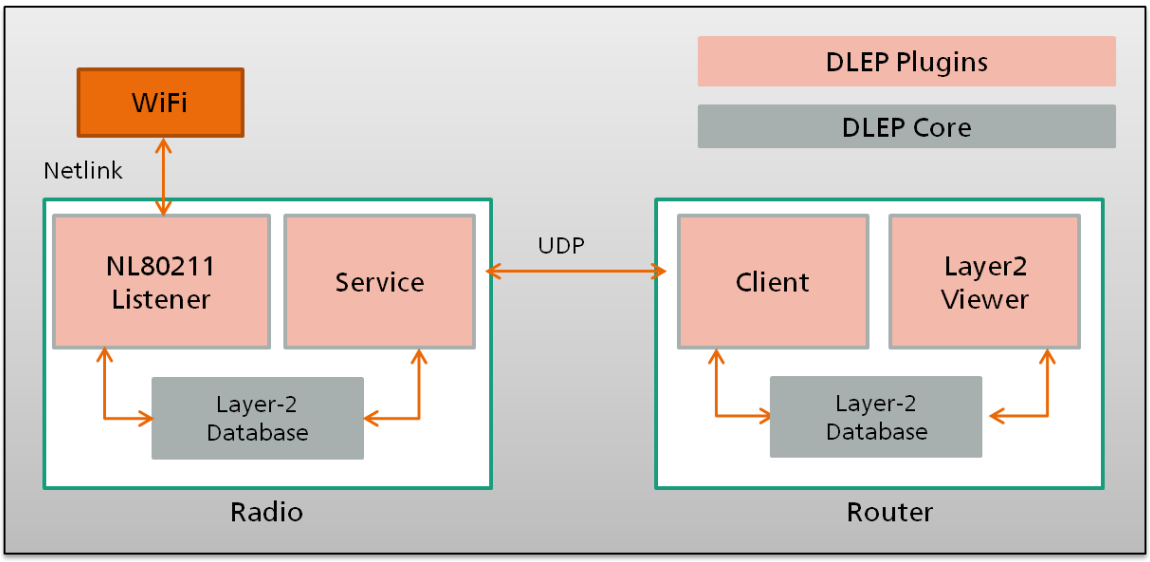

Figure 1.3: Architecture overview of DLEP for a radio router communication

whole layer 2 network) and neighbour data (information concerning the link to a specific neighbour) are stored. 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. 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. 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 *service* is a data consumer plug-in and read these informations from the local database to send them to interested entities. Normally this function is performed using a link-local multicast. The functionalities of the *nl80211listener* and the *service* plug-ins implement the functionality of a DLEP-capable radio.

The *client* is an another data provider plug-in running on the router. It is capable of receiving the data transmitted from the radios. This data may be used by external (e.g. routing) applications, using the related API accordingly. Another possibility to access the data is to connect a plaintext TCP

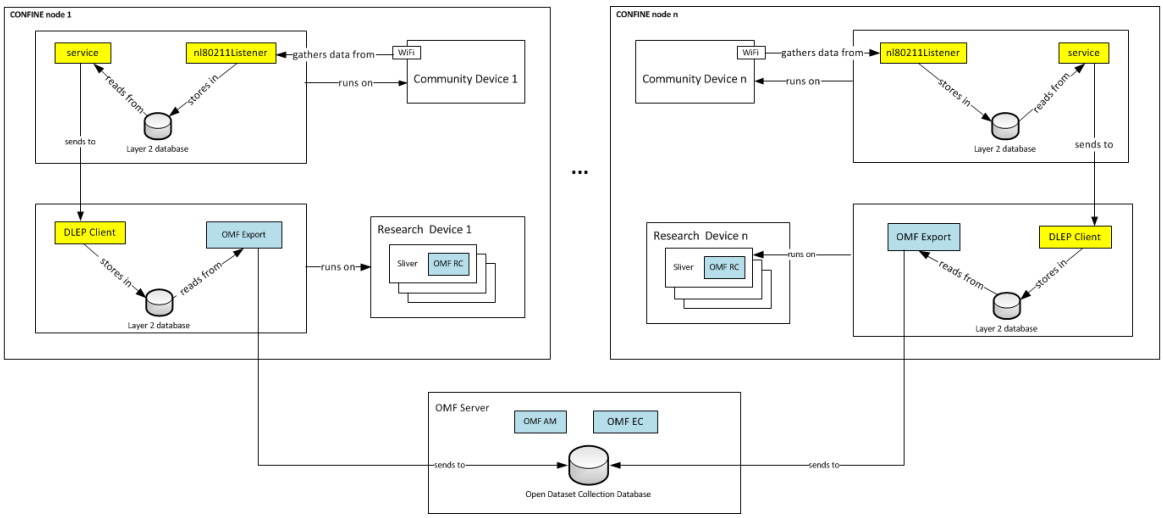

Figure 1.4: Open dataset collection architecture based on DLEP and OMF

connection to the dlep-app (i.e. a *telnet interface*). In this case, the *layer2viewer* is used as a data consumer plug-in to provide the information on the status port.

1.6.2. Architecture for Open Data Set Collection

DLEP is combined with OMF to provide a dataset collection architecture for physical and link layer data extracted from users real network traffic. In this setup, DLEP is used as the data collection software while OMF is required to start, control and stop the DLEP experiments and to retrieve the measurement data. At large, the architecture for open data set collection shown in Figure 1.4 corresponds to the DLEP architecture described in the previous section. However, in this case, the radio and router nodes from Figure 1.3 corresponds to the CONFINE community and research devices respectively. Analogously, two DLEP core applications (dlepapp) are started at the community and research devices. Based on the type of the device, different plug-ins are loaded into these applications. On the community device, the 80211listener and service plug-ins are loaded. The former collects layer 2 information from a WIFI card and writes them into a local database. The latter reads the informations from there and sends them to interested entities over a UDP connection per multicast.

On the research device, one separate sliver is created which is only used by DLEP for dataset collection purposes. On this sliver, the client plug-in is loaded. This plug-in is responsible to receive the UDP stream sent by the service plug-in on the community device and store the informations in its local database. In other words, the service plug-in on the community device and the client-plug-in on the sliver of the research device are used to replicate the layer 2 databases between the two nodes.

The collected layer 2 informations on the sliver are further forwarded to the central open dataset collection database in which measurements from all existing research devices of the FKIE testbed are stored. To this end, we have provided a further plug-in, called OMF Export, that transports the replicated database informations from the slivers on the research devices to this central open dataset collection database that is created on the OMF server.

A DLEP-based data collection session can be considered as a single experiment which is defined as a customizable time interval in which layer 2 informations from a WIFI interface on a community device are collected. To enable a systematic data collection, OMF is used as the experiment controlling software. More specifically, OMF triggers every customizable time interval (e.g. every day)

a DLEP experiment. In doing so, tables containing the collected data sets are created in the open dataset collection database in every pre-defined time interval which then can be provided to external researchers. The automatic start of the DLEP-experiments via OMF can be triggered, e.g., using the Linux cron jobs mechanism.

It should be noted that the collection of datasets from the community network can cause privacy issues as the captured data is extracted from real users network traffic. MAC addresses are typically bound to hardware devices which could allow gathering movement profiles of network users if multiple data collectors are present in the network. Thus additional anonymization services should be applied on the collected data.

Using OMF as the experiment controller, DLEP can also be used to validate the accuracy of active measurement tools to be evaluated. More specifically, assuming that DLEP is able to obtain the same metric as the active tool under test it can provide reference values with which estimates of the active tool are compared together to give an indication about their consistency and accuracy level. As described in the previous section, DLEP provides a variety of different physical and link layer informations. To produce these informations, the DLEP application requires network data which is collected and delivered by the 802.11listener plug-in. The key rationale here is that the artificial probing traffic of the active tool under test can be simultaneously used as the passive traffic required by 80211listener plug-in to obtain the same metric using the DLEP application. For this purpose, the active tool and DLEP are started in parallel by the OMF experiment description. This way, comparative evaluations between active tools and DLEP can be performed.

To give an example for a comparative evaluation with DLEP, validation of a packet loss measurement tool is described in the following. Consider an end-to-end path consisting of a number of links between a sender and receiver host. The active tool under test is started along the path to measure the number of packet losses within the probing stream sent. During the measurement, the active probing traffic is in parallel used for the acquisition of DLEP informations (including the packet loss metric) by the 80211listener. By running the DLEP application on each node of the path, packet loss on each link segment can be obtained separately. Concatenation of packet loss measurements of different link segments enables to estimate this metric at the end-to-end scope. Thus, upon the end of the active measurement session, the packet loss is measured in two different ways using one and the same probing stream. Firstly, one estimate is given by the packet loss measurement tool actively, and an another estimate is given by DLEP passively. Then, a comparative evaluation of the two metric estimates can be undertaken to assess the accuracy of the tested active tool.

Similarly, the accuracy of many other active tools can be validated using other provided DLEP informations including capacity, available bandwidth, signal strength or retransmission counter measurements as reference values.

2. Development of benchmarking framework

When developing a testbed, the result has to be validated and benchmarked to allow scientific validation and result confidence. This chapter will propose a framework to benchmark the CONFINE community-lab experimental infrastructure, based on existing testbed benchmark frameworks.

2.1. Introduction

A considerable amount of research on benchmarking of wireless networks has been done within the CREW[19] project. The proposed benchmarking framework is based on this research.

As stated in [20]; "*In the scope of the performance analysis of computer systems, we define benchmarking as the act of measuring and evaluating computational performance, networking protocols, devices and networks, under reference conditions, relative to a reference evaluation.*". In this definition, benchmarking consists of two main parts or activities to complete a benchmark.

- *Measuring* is the act of performing an experiment under reference conditions and collecting relevant data during the experiment.
- *Evaluating* is the act of combining the gathered data into quantitative metrics and interpreting or comparing those metrics to a reference evaluation.

Benchmarking is preformed using benchmarks. A benchmark is a set of detailed descriptions describing how the benchmark should be set up, which data should be gathered, how this data is converted to metrics, how the metrics should be interpreted, and so on. It should provide all the necessary information to preform an experiment under the exact same conditions as stated and should result in the exact same results. This set of descriptions can be divided in the following items.

- A *scenario* should include the set-up and all the necessary parameters of the experiment.
- The *criteria* describe the high-level focus of the experiment.
- A *metric* provides a quantitative measure to evaluate a particular aspect of the experiment. Those metrics are determined using a well defined methodology.
- A *benchmarking score* is a combination of the different metrics of the benchmark and is used to evaluate the experiment.

The success of a benchmark is highly dependant on the definition of the used terms in its items. This means that a scenario, the criteria, the metrics and the benchmarking scores should be described clear and unambiguously.

2.2. Goal

The goal of benchmarking is to compare the performance of different experiments or different parameter settings of a single experiment. To achieve this, the *comparability* and *interoperability* requirement of the benchmarking framework should be satisfied.

"*Comparability should be a fundamental property of any benchmark; comparability means that two independently executed benchmarks can be meaningfully compared to each other.*"[20]. This means

that a benchmark should be independent of the testbed used and independent of the moment it was executed. Thus for a given benchmark, running it on the same testbed at different moments, should give the same results. This requirement is called the *repeatability* of a benchmark. Note that not only the unit to be tested should be monitored but also the environment which it operates in to achieve repeatability.

Also for a given benchmark, running it on a testbed with similar capabilities, should give us the same results. This is called the *interoperability* requirement. Important for the interoperability requirement is the *configurability* of a testbed environment. The testbed should support at least all of the configuration settings of a benchmark to successfully execute a benchmark.

To summarise, the benchmarking framework should fulfil the following requirements.

- The comparability requirement and thus also the repeatability requirement.
- The interoperability requirement and thus also the configurability requirement.

2.3. Proposed approach

The proposed benchmarking framework is based on the work from the CREW[19] project.

2.3.1. Framework

The user or actor that wants to run the benchmark, provides to the system a generic benchmark scenario. With this scenario, the experiment configuration, the metrics used and the benchmark score calculation are defined as indicated with the arrows in Fig. 2.1.

The experiment config is interpreted by the testbed controller and translated in specific instructions of the testbed to set-up the benchmarking parameters, such as the test environment, the monitoring parameters and the data conversions. The testbed controller is the interface between the testbed specific configuration and the generic configuration.

During the benchmark test run, the monitoring block monitors the parameters provided by the testbed controller which in turn are generically described in the benchmark scenario. This could be for instance SNR monitoring, number of transmitted packets, system load, etc. Besides generating data by the monitoring block, it also verifies that any criteria or requirements of the test are fulfilled.

The data gathered by the monitoring block is testbed specific and is translated by the data conversion block to the generic metrics as described in the benchmark scenario. The testbed controller sets up the data conversion block by using the settings in the experiment config. For example, the benchmark requires the packet drop ratio between two hosts. The testbed controller translates this into monitoring the number of packets sent from the first to the second host and the number of packets received from the first host on the second host. The data conversion will gather both counts and calculate the ratio of dropped packets as the metric.

The metrics are combined by the benchmarking score block into the benchmarking score. This benchmarking score is then provided to the user as the result of the benchmark.

2.3.2. Definitions

As mentioned before, the success of a benchmark is highly dependent of the definitions used. For example, what is being monitored by the monitoring block, effectively can correctly be translated

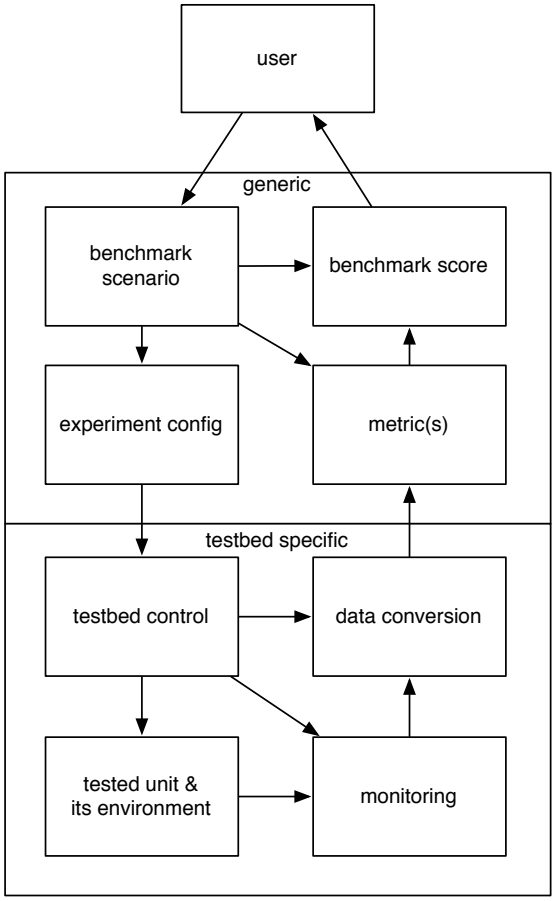

Figure 2.1: High level elements in the framework and their relationships.

into the metrics defined in the benchmarking scenario. Therefore the definitions for the following items should be clear and agreed upon.

- The methods of monitoring. It should answer how the raw data should be gathered.
- The conversion from raw monitored data to the metrics. This definition answers how the raw data should be converted in the metric.
- The benchmarking score which answers how the metrics are combined in the benchmarking score.

In the end, there is a pool of metrics, each with their set of standardised methodologies and operations for creating the raw data for the metric. This process of creating the pool is an iterative process. When adding a new metric, the following checks should be verified.

A metric should be

- unambiguous,
- generic,
- independent of a particular testbed and
- complete. Is there data not monitored for the metric that influences the metric?

A standardised methodology for a metric should be

- unambiguous,
- generic and

- independent of a particular testbed.

A benchmarking score can then be defined using a subset of the pool of metrics and operations between them. Also for a benchmarking score the following checks should be verified.

A benchmarking score should be

- unambiguously.
- generic.
- complete. Are there metrics not in the benchmarking score that influence the interpretation of the benchmarking score?

2.3.3. Interfaces

Another important aspect of the benchmarking framework is the transition between the generic and the testbed specific section. The experiment configuration should be provided to the testbed controller in a standardised data format such as OMF. This allows every testbed controller implementation to easily adopt to the benchmarking framework.

Also the interface between the data conversion and the metric block needs a standardised way of exchanging data such as the JSON format.

2.3.4. Resources

To allow rapid adaptation of testbeds to the benchmarking framework, documentation, implementation and other resources of the benchmarking framework should be available. For example, scripts to generate data for a given metric, implementation of a monitoring system, etc.

Also a set of testing scenarios should be provided. Such a test scenario is the unit test for a given benchmarking score. It allows testbeds to verify their implementation of a benchmarking scenario and the resulting benchmarking score. Creating such a set of testing scenarios should be incorporated in the process of creating and defining a benchmarking score.

2.3.5. Application to Community-Lab

The proposed benchmarking framework still has to be formally applied to the CONFINE testbed Community-Lab. This is an important task during year three. However, a number of components are already in place.

The experiment config is handled by the CONFINE controller in a CONFINE-specific format, but an OMF integration is also being developed. More information on this OMF integration can be found in deliverable D3.2. The testbed control is performed by the CONFINE controller, which is described in detail in deliverables D2.1 and D2.2. Monitoring is performed by the CONFINE monitor, which is outlined in the section “A Monitoring system for Community-Lab” in deliverable D4.2.

The data exchange format and testing scenarios are the most important missing resources to implement a benchmarking framework, apart from defining the benchmark methodologies. This will be the main focus of the benchmarking efforts during year three.

2.4. Conclusion

A great benefit of the proposed benchmarking framework is the generality it provides to the user. The user can create a scenario independent of any testbed used. The result of the benchmark is a simple score which should be easily interpreted by the user. Yet to provide such ease-of-use, the framework should be unambiguous en well defined. The metrics should be complete, the methodologies and interfaces should be standardised. Building the set of metrics and methodologies is an iterative process where testing scenarios help future testbed implementors to verify their testbed controller implementation. The CONFINE project already has a number of components in place to create the proposed benchmarking framework. The efforts in year three will focus on the missing components.

3. Documentation of best practices

The best practices documentation of CONFINE aims at gathering the experiences made by the different experimenters that use the Community-Lab facility.

This chapter describes initial results from the open call experiments, which were initiated during year 2. In addition, this chapter describes tools to assist in setting up realistic experiments using the CONFINE testbed. Hints which help to make experimentation easier are provided.

3.1. Experimentally driven research

The main feed for this documentation are the projects that in experimentally driven research use the Community-Lab facility. As a result of CONFINE's first open call in 2012, five projects were selected that are currently using Community-Lab. Within the FIRE objective in FP7 Call 8 for experiments on FIRE facilities, a FIRE research project was selected that also uses the Community-Lab facility.

The tool to share the experiences of the testbed users among each other and with external users in terms of a best practice document was found to be the CONFINE Wiki with a specific page on best practices (https://wiki.confine-project.eu/bestpractice:start) and linking to a collection of experimenter's experiences (https://wiki.confine-project.eu/bestpractice:experiences-experiment).

We note that the content of these Wiki pages are edited in an ongoing way, following the evolution of the experiments and the gaining of additional insights in the testbed usage. Editing is open to the CONFINE consortium members so that each experimenter is able to add and share his/her experiences.

The content is expected to be useful for future users of Community-Lab to see experiments from a practical perspective, and to see the steps that other experiments followed for using Community-Lab.

For reference and documentation purposes, appendix A gives a copy of the Wiki page at the time of writing this deliverable.

Finally, the tutorials (http://wiki.confine-project.eu/tutorials:start) and the manual (http://wiki.confine-project.eu/usage:start) also form a perfect example of best practices. Both are available to everyone on the CONFINE wiki.

3.2. Link Error Model for VCT

The CONFINE testbed infrastructure cannot only be used for real world testbeds, it can also be used as a completely virtualized testbed for local experiments. Each CONFINE node is implemented as a virtual machine on a server system, which are connected by a Linux bridge.

Unfortunately using the Linux bridging code limits the types of connectivity that can be used in the emulated mesh network. It is easy to prevent certain pairs of nodes to communicate with each other by using the EBTables subsystem, but this constrains the virtual testbed to a set of perfect links without any packet loss. While Linux includes a subsystem called NetEm (Network Emulator) to emulate

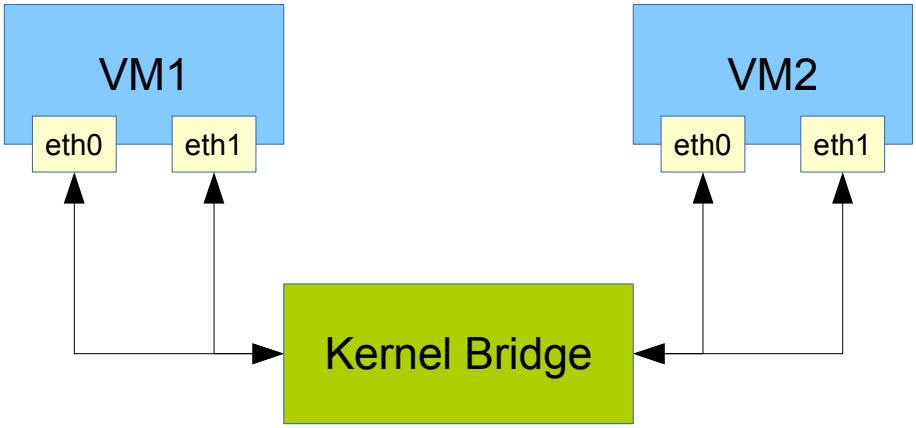

Figure 3.1: Connection of virtual machines with central bridge for link error model.

packet loss, delay and packet reordering, it is also very hard to setup and use because NetEm is part of the traffic shaping subsystem of Linux.

To make it easier to setup a virtualized testbed with lossy links, the we developed a set of scripts using only EBTables and IPTables subsystem to generate configurable packet loss between each pair of virtualized CONFINE nodes. EBTables is used to mark forwarded packets based on the incoming and outgoing interface. This marker is then used by the IPTables statistics module to drop a certain proportion of the packets.

Additionally, we used measurements of IEEE 802.11 packet loss to write a function that can be used to estimate the unicast or broadcast packet loss based on the distance between the two communication partners. This functions allow to setup a topology based on coordinates on a 2D grid and quickly calculate a rough estimate of the packet loss between each pair of nodes.

While this link error model cannot emulate the effects of collisions between packets or the delay caused by transmission speed or retransmissions, it allows to easily setup a more realistic virtual testbed.

To connect multiple virtual machines with the link layer model, all of the virtual machine interfaces must be connected to the same bridge (see Fig. 3.1). The bash shell script (see wiki for script: [21]) then allows the user to easily setup (static) connections between pairs of nodes, each of them with a configurable unicast and broadcast packet loss. The names of the virtual machine interfaces just have to be edited into the shell script.

The following example script is setting up a simple line connection between a group of four virtual machines, each of them with a unicast success rate of 1.0 and a broadcast success rate of 0.7. There is also connectivity between two-hop neighbours, but with a unicast success rate of 0.7 and a broadcast success rate of 0.5.

Listing 3.1: Simple topology setup for VCT

```
#!/bin/bash

. ./route_basic.sh
```

```
route_init

set_route_eth1  1 2 100 70
set_route_eth1  2 3 100 70
set_route_eth1  3 4 100 70

set_route_eth1  1 3 70 50
set_route_eth1  2 4 70 50
```

Based on an earlier work of Fraunhofer FKIE the script also includes a function to calculate a rough guess of Wi-Fi transmission success rate based on a distance in meters. This function can be used to calculate the success rate from a set of node positions.

Listing 3.2: Calculating packetloss from distance

```
#!/bin/bash

. ./route_basic.sh
route_init

uc_loss=$(calc_multicast_success_rate 200)
mc_loss=$(calc_unicast_success_rate 200)

set_route_eth1  1 2 ${uc_loss} ${mc_loss}
set_route_eth1  2 3 ${uc_loss} ${mc_loss]
```

Unicast packet loss is estimated as a failure of five packets, each sent with broadcast loss rate.

A paper on this research will be presented during the CNBuB ’13 workshop in October 2013.

A. Appendix: Experiment Experience

Note: this appendix reflects the state of the Wiki page https://wiki.confine-project.eu/bestpractice:experiences-experiment at the moment of writing this deliverable, and serves as offline documentation for this online resource.

```
This page is an ongoing work and describes experiences and ways
that were followed to run experiments of different projects using
the Community-Lab facility.
```

A.1. Open Source P2P Streaming for Community Networks

A.1.1. Introduction

The goal of the experiment is demonstrating that Community Networks can support advanced multimedia services such as real-time video and TV distribution.

The study of efficient video streaming techniques will be done through the Open Source P2P streaming software PeerStreamer.

The project will produce a stable version of the application tailored for Community Networks. The experiments will give useful indications on design and dimensioning of the networks as well as on the best practices to implement streaming services with various delay constraints on top of Community Networks.

A.1.2. Experiences and Steps

Installation of VCT and experiments on a dedicated machine to run experiments before deployment on the community-lab;

Preparation for experiments with real slivers in Community-Lab.

Experiments on functional tests using the IPv6 version of PeerStreamer either among the Community-Lab's slivers or with UniTN peers connected through tinc.

A new PeerStremer source has been setup in UniTN to increase the video streaming offer in the Community-Lab when possible. The source will be moved within the Community-Lab upon the availability of resources.

Investigation of IPv4 network connectivity limits in the Community-Lab related to both single and multiple NAT levels. Issues with nested NATs have been identified. P2P communications are sensitive to NAT punching limitations.

Assessing communications among slivers on different islands and between slivers and UniTN external peers (not connected through tinc).

A.2. Anonymous communication with unobservability

A.2.1. Introduction

In this project we propose to study the practical applicability of an online advertising network called AdLeaks through the CONFINE testbed. AdLeaks leverages the ubiquity of unsolicited online advertising to provide complete sender unobservability when submitting disclosures. AdLeaks ads compute a random function in a browser and submit the outcome to the AdLeaks infrastructure. Such a whistleblower's browser replaces the output with encrypted information so that the transmission is indistinguishable from that of a regular browser. Its back-end design assures that AdLeaks must process only a fraction of the resulting traffic in order to receive disclosures with high probability. We have implemented the AdLeaks system design and we have evaluated it through mathematical analysis and micro-benchmarks.

A.2.2. Experiences and steps

Data access: together with Guifi partners, real data from Guifi proxies was obtained. For this purpose, Guifi set up a proxy that that generated logs for a longer time period. Data anonimization: the log data was anonymized. The collected anonymized logs were cleaned and some post-processing was done. Data models: the data helped us to complete our statistical models and thus being able to recreate realistic traffic conditions for our experiments.

Independently of the data gathering, a browser extension for the blogging platform has been developed.

Work on the traffic gathering extension will start soon.

We ported our benchmarking tools and server code to OpenWRT in preparation of a deployment within the CONFINE testbed.

A.3. Exploitation of information Centric network principles in wireLess cOmmunity NEtworks

A.3.1. Introduction

Our experiments seek to exploit the instruments of an Information Centric Network (ICN), namely in-network caching and routing-by-name, to shorten the multi-hop path through a dynamic replication of information and services, on community devices. Following an evolutionary approach, ICN functionality is deployed over IP, without compromising the operating regime of IP-based community services. We prototype a community web hosting service, named WSaaS, that uses storage and computation resources of community user's hosts, to dynamically replicate Web pages of community users. We use Community-Lab facility to carry out comparative experimentations, ICN vs. IP, showing performance improvements obtained both for basic point-to-point data transfer and within the community web hosting use-case.

A.3.2. Experiences and steps

Familiarisation and usage of the Virtual CONFINE Testbed

The VCT tool helped to assess the possibility of deploying ICN tool.

Creation of slivers in the Community-Lab testbed.

A.4. Wi-Fi network Infrastructure eXtension (WiFIX)

A.4.1. Introduction

INESC TEC has defined a solution for WMN, named Wi-Fi network Infrastructure eXtension (WiFIX), that considers (1) unicast, multicast, and broadcast routing, (2) channel assignment, and (3) multi-hop medium access control aspects, in order to support existing and new applications on top. WiFIX overcomes the disadvantages of existing WMN solutions, namely by: i) reducing routing signalling overhead; ii) considering a new approach for multicast/broadcast traffic diffusion that takes advantage of Wi-Fi built- in unicast data rate control and delivery guarantee; iii) defining a topology-aware channel assignment algorithm that increases WMN performance and scalability; iv) considering a multi-hop scheduling mechanism overlaid on the 802.11 MAC, which enables efficient and fair WMN multi-hop medium access; Experiments in Community-Lab aim to complement our evaluations of WiFIX with real-world, large-scale WMN experiments.

A.4.2. Experiences and steps

With the CONFINE consortium, the option of the WiBed testbed was found in order to enable in an easier way part of our requirements for our experiments.

Familiarization with WiBed testbed through its documentation.

Familiarization with the WiBed testbed tools.

Compilation and configuration of the WiBed related software, creation of the image file to be installed in the mesh nodes, preparatory steps at local nodes at INESC.

A.5. Confidentiality in the open CONFINE world

A.5.1. Introduction

Our experiments look at data protection and confidentiality issues by evaluating and analyzing to which extent existing privacy-preserving routing techniques applied on the Internet can be transferred and tailored to the needs of community-based networks. To this end we want to test which of the available privacy-preserving routing techniques can be efficiently deployed in the community-based networks. We put our focus on lightweight methods developed by ourselves that are specially designed for environments with limited resources, lack of a central point of trust, and that are able to deal with a high churn rate.

A.5.2. Experiences and steps

Research work on WiFi access point fingerprinting to protect against spoofing

Familiarization with virtual CONFINE testbed (VCT).

Sliver images that are fully pre-configured are an interesting option that should be better supported, while the option of installing manually at runtime via SSH should rather be a backup solution.

A.6. Clouds in Community Networks

A.6.1. Introduction

This project aims to provide community services organised as community clouds. Experimentally-driven research, using the Community-Lab testbed will be used for the evaluation of the community cloud.

A.6.2. Experiences and steps

Remote researchers have access to the Guifi community network.

Confine controller creates slices of VMs. These slices can contain VMs on geographically distant cloud resources.

VM templates could be customized for our purposes and can be deployed through the Confine-controller to our VMs.

Additional hardware for cloud resources could be added to the Guifi community network through the Community-Lab infrastructure.

Due to the open source policy of the CONFINE project which facilitates easy access to all source code, the potential for federation of Confine-controller with Cloud management platforms can be explored.

B. Conclusions

This deliverable has described the tools for experimental research, as developed and studied by the CONFINE project in task T4.3 and milestone MS13.

The first chapter, on open data, primarily serves as a source of information for external researchers. It describes the open data aspects of Community-Lab. The concepts of open data are introduced, followed by a strategy to generate open data in the CONFINE project. Then, three methods to generate open data are presented: a mapping tool, publishing existing open data from community networks and a DLEP experiment integration method.

The second chapter, on benchmarking, will primarily serve as a reference point for internal researchers, as it provides a means to validate their results. It outlines the CONFINE testbed benchmarking framework, based on a framework from an existing EU FP7 project. It describes a incremental approach, which will be implemented during year 3 of the project.

The third and final chapter, on best practices, is input to both internal researchers who use the testbed and external researchers who want to implement a similar experimental facility. This chapter describes best practices from working with community-lab, both from ongoing open call experiments and from the development of a simulation tool which can help predict testbed behaviour.

Overall, the research reported in this deliverable presents good progress and shows a number of interesting opportunities to further improve the testbed and its community network infrastructure.

Bibliography

[1] Bart Braem, Chris Blondia, Christoph Barz, Henning Rogge, Felix Freitag, Leandro Navarro, Joseph Bonicioli, Stavros Papathanasiou, Pau Escrich, Roger Baig Vinas, Aaron L. Kaplan, Axel Neumann, Ivan Vilata i Balaguer, Blaine Tatum, and Malcolm Matson, "A case for research with and on community networks," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 68–73, July 2013. 1.1

[2] "Data.gov," http://www.data.gov/. 1.1

[3] "European union open data portal," http://open-data.europa.eu/. 1.1

[4] "World databank," http://data.worldbank.org/. 1.1

[5] "Measurement lab," http://measurement-lab.org/. 1.1

[6] "Open knowledge foundation," http://okfn.org/. 1.1

[7] "CKAN," http://ckan.org/. 1.2.1

[8] "Open data commons open database license," http://opendatacommons.org/licenses/odbl/. 1.2.2

[9] "Open data commons," http://opendatacommons.org/licenses/odbl/. 1.2.2

[10] "Guifi.net Growth Map," http://guifi.net/en/guifi/menu/stats/growthmap. 1.3.1

[11] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, Laurent Viennot, et al., "Optimized link state routing protocol (olsr)," 2003. 1.3.3

[12] "Cisco Discovery Protocol," http://www.cisco.com/en/US/docs/ios-xml/ios/cdp/configuration/15-mt/nm-cdp-discover.html. 1.3.3

[13] "MikroTik Neighbor Discovery Protocol," http://wiki.mikrotik.com/wiki/Manual:IP/Neighbor_discovery. 1.3.3

[14] "Graph Exchange XML Format," http://www.gexf.net/format/. 1.3.6

[15] Jinliang Fan, Jun Xu, Mostafa H. Ammar, and Sue B. Moon, "Prefix-preserving {IP} address anonymization: measurement-based security evaluation and a new cryptography-based scheme," *Computer Networks*, vol. 46, no. 2, pp. 253 – 272, 2004. 1.4.1

[16] "Pycryptopan github repository," https://github.com/FFM/pycryptopan. 1.4.1

[17] "The python package index - pycryptopan," http://pypi.python.org/pypi/pycryptopan. 1.4.1

[18] "RIPE RIS raw data," http://www.ripe.net/data-tools/stats/ris/ris-raw-data. 1.5.3

[19] "CREW project," http://www.crew-project.eu. 2.1, 2.3

[20] Stefan Bouckaert, Jono Vanhie-Van Gerwen, Ingrid Moerman, Stephen C Phillips, Jerker Wilander, Shafqat Ur Rehman, Walid Dabbous, and Thierry Turletti, "Benchmarking computers and computer networks.," . 2.1, 2.2

[21] "Documentation of best practices," https://wiki.confine-project.eu/bestpractice:start. 3.2