Community-Lab: vision and architecture of a community networking testbed for the Future Internet

***Abstract*—Community networks are an emerging and successful model for the Future Internet across Europe and far beyond. The CONFINE project complements existing FIRE (Future Internet Research and Experimentation) infrastructures by establishing Community-Lab, a new facility built on the federation of existing community IP networks constituted by more than 20,000 nodes and 20,000 Km of links. These unique infrastructures pose an interesting set of challenges different from existing testbeds. In this paper we present the benefits of having such testbed for the research community as well as the improvement and evolution of community networks themselves. Additionally, we sumarize the challenges encountered and describe our long-term vision for the community network testbed and its current state of implementation.**

I. Introduction

In the recent years, the perception of the key role of Internet access for the digital society and the size, growth and scope of the Internet and has put under stress its architecture and protocols. The growing Internet penetration at homes, schools, workplaces, and any public or private organisation, the introduction of very demanding and QoS-sensitive applications like distribution of large files, interactive multimedia, real-time audio and video, results in an explosive growth in the demand for capacity.

However, the characteristics of the investment needs, cost of ownership, and expected return of investments outside the traditional telecommunication markets in rural areas or in urban areas with low income individuals or special needs not covered by standard products makes those areas very unattractive for traditional telecom providers. It has lead to the appropiation of these strategic technologies by the communities themselves, by means of self-provision, self-ownership and self-operation. These underserved areas have taken their connectivity needs into their own hands, hence, they developed into "community networks".

Community networking (also known as bottom-up networking) is an emerging model for the Future Internet across Europe and beyond where communities of citizens build, operate and own open IP-based networks, a key infrastructure for individual and collective digital participation.

These "community networks" (CN) are part of the Internet but differ in several key aspects due to their nature: they are open and transparent as there are no barriers of entry for participation; governance and knowledge is open, with public documentation on every technical and non-technical aspect; they are self-owned as community members own the network. Self-managed with decentralized and uncoupled structure that can be open for the participation of community members; self-growing in topology and capacity, based not only on planning but also on immediate demand, as community members can learn and decide to extend the nework to reach a new member or provide more capacity- Thus, these networks are not just a path to reach the outside Internet but a local network with internal servers, services and internal traffic connected to the Internet.

Community-Lab is a testbed being deployed by the Confine European Integrated Project[1]. This testbed supports experimentally-driven research on Community-owned Open Local IP Networks. It integrates and extends three existing community networks: Guifi.net (Catalonia, Spain), FunkFeuer (Wien, Austria) and AWMN (Athens, Greece); each is in the range of 500 – 20,000 nodes, a greater number of links and even more end-users. These networks are extremely dynamic and diverse. They combine successfully different wireless and wired (optical) link technologies, static, dynamic and adhoc routing schemes, and management schemes. They run multiple self-provisioned, experimental and commercial services and applications. This testbed provides researchers with access to these emerging community networks, supporting any stakeholder interested in developing and testing experimental systems and technologies for these open and interoperable network infrastructures.

The ultimate motivation behind the deployment of such infrastructure is to support the growth of community networks in terms of their scalability and sustainability by providing the means to conduct experimentally driven research. We expect that this research will directly impact the quality of community networks by improving current short-commings and providing the basis for a sustainable model of community networking.

Currently, there are several challenges and lines of research to enhance current community networks. Among others, CNs must consider carefully the impact on the privacy of their users given their opennes and degree of decentralization; content distribution should be efficient given the constrained resources by improving network robustness – e.g. routing and cross-layer protocol optimizations; a critical issue is the robustness of the nodes participating on the network (e.g. automatic reconfiguration, deployment and failure handling, etc.) because they may sit on remote locations not easily accessible on

[1] http://www.confine-project.eu

a day-to-day basis by community owners; and finally, CNs must handle a whole range of heterogeneous physical networks ranging from wireless links to broadband fiber.

However, these challenges are being addressed currently by means of simulations or on very small controlled lab environments because there exists no real testbed to experiment on, which makes the results hard to translate to real scenarios. The Community-Lab testbed being deployed by CONFINE tries to provide such infrastructure for researchers to evaluate their proposal on an environment with the characteristics (for the best and for the worst) of real community networks.

II. Related Work and Background

There are several initiative in Europe without considering the three CNs participating in this initial phase of the testbed –guifi.net (Catalonia, Spain), FunkFeuer (Wien, Austria) and AWMN (Athens, Greece). The Berlin RoofNet project is an experimental mesh network in Berlin consisting of nearly 50 indoor nodes mainly to provide students with Internet access as well as VoIP [1]. In the USA there are similar initiatives as well. Cities like New York, Seattle, Washington or Austin cooperate with public and non-profit organizations to develop and improve free wireless hotspots within their cities to provide citizens with Internet access. Also, the MIT Roofnet [2] project is a mesh network consisting of about 50 nodes and covers an area of about four square kilometers allowing for small scale and controlled experimentation thanks to the Click Modular router integrated within their firmware.

However, all these wireless networks deployments are expected to provide a service (Internet connectivity mainly) to users and do not cover the necessity of researchers to perform experimentation on top of them.

The first initiative of a large-scale testbed to conduct experimental research was introduced by PlanetLab [3]. They currently manage a planetary-scale experimental testbed composed of hundreds of nodes to deploy and evaluate distributed applications and services in a real environment. The key characteristic of this testbed is the concept of resource sharing (given their scarcity) among different researchers by means of virtualization. The introduced a basic abstraction called *slice* which enables the isolation of different experiments. A *slice* is a ortogonal fraction of the testbed, or in other words, a set of nodes on which applications are assigned a fraction by means of a kind of virtual machine called *sliver*. This concept of *slice* has evolved over time to also account for network isolation and other type of resources.

If we look at the wireless research area, there are also several initiatives to create a testbed to conduct experimental research, but none of them address the problem of high scale deployments nor sharing of resources – two of the main principles behind Community-Lab. For instance, the IBBT w-iLab [4] in Flanders (Belgium), consists of mainly 200 indoor locations spread across three office floors and is mainly composed of sensor networks. In Greece, the NITOS [5] testbed enable researchers to completely control the software on the testbed nodes, which denies the possibility of several simultaneous users. Besides, it consists of 40 wifi nodes which only enables small scale experiments. Finally, the ORBIT testbed [6] is a large indoor two-dimensional grid of arround 400 802.11 radio nodes which can be dynamically interconnected into specified topologies, which makes it the largest indoor wireless lab.

Discussion. There are three major characteristics which makes the Community-Lab testbed of the CONFINE project unique compared to the deployments presented in this section: i) the larger scale of the testbed with an expected size of thousands of nodes by the end of the project compared to the few tens of nodes of current deployments; ii) the integration within already existing production community networks which provides a more realistic scenario for experimentation oposed to controlled lab environments; iii) the support for long term studies of new networks protocols and parallel experimental services which share the same wireless infrastructure – both from the testbed and from the participating wireless community networks – by means of node and network virtualization oposed to other testbeds which provides dedicated nodes to testbed users. Of course, the development of current deployments presented in this section can and should provide a solid foundation for building this unique infrastructure.

III. Challenges and Requirements

Deploying, managing, and using experimentation facilities in an existing CN, where ownership and administration of network infrastructure is divided over many entities, imposes a set of new challenges and requirements to the design of the CONFINE architecture. A selection of most important requirements is briefly discussed by the following Subsections.

A. Lightweight, Low Cost, Free, and Open Source

In order to achieve a widespread deployment of CN nodes supporting the CONFINE testbed, existing installations must be extended with new components which comes with additional equipment acquisition and maintenance cost (eg increased energy consumption) for the node owners. Therefore, lightweight and low cost solutions are important to keep expenses as low as possible.

Reusing of general accepted approaches and free and Open Source software is mandatory to ease reviewing and trust, lower the burden for contribution, and to aim for sustainability and acceptance in the free networks communities.

B. System Stability

To facilitate the preparation and execution of experiments, researchers should be provided with an environment they are familiar with and which allows them to reuse already existing implementations and prototypes. Such environment is given by a Linux OS in combination with root permissions which has evolved to a de-facto standard environment for networking experiments.

On the other hand, from a security point of view, it must be ensured that experiments enabled with root permissions can not affect the general CONFINE node system. This can

be achieved by isolating experiments on dedicated hardware or with virtualization techniques. But, despite many mature virtualization solutions exist for UNIX systems, few of them have been successfully ported for embedded hardware which are commonly used for nodes in CNs.

C. Network Stability

Since Confine is aiming to enable community integrated network experiments, some (controlled) level of interaction between the experiments and the main CN must be possible. On the other hand it must be ensured that experiments do not interfere seriously with each other nor with the production network. The potential harm of experiments can be classified into two types:

- Resource Consumption: It must be ensured that each experiment gets a limited but fair share of the available link capacities. Traffic Shaping and Distributed Rate Limiting Techniques can be used here.
- Conflicting Protocol Behavior: Experimentation with key network services (medium access control, routing, DNS) and other protocol resources (address, port, channel assignments) must be isolated or rigorously controlled. Firewall or Software Defined Networking techniques (like VLAN tagging) or even the complete physical isolation of link resources are considered depending on the lowest accessed network layer of the experiments.

D. Privacy of CN users

While researchers should be conceptually enabled to collect user and network data (for statistical evaluation or as real-time feedback for protocols), privacy regulations of the corresponding CN licences and peering agreements must be respected. Therefore, the CONFINE system must provide convinving mechanisms for traffic filtering and anonymization of networking data before it is provided to researchers.

E. Management Robustness

One of the interesting characteristics for research in real-life CNs is the variety of link setups and stabilities that arise du to the decentral management of CN infrastructure and the objective to provide connectivity even to remote locations at low cost. Often, such locations are abandoned by commercial ISP because of deployment challenges and poor revenue perspectives.

Therefore, the system (in charge of managing the distributed experiments in CONFINE enabled nodes) must be prepared to deal with connectivity starvation and the spontanious appearance and outage of links and nodes.

F. IP-Address Provisioning

The objective to allow active experiments on top of existing CN infrastructure as well as interaction with users demands the provisioning of CN public IP addresses for slivers.

However, due to the general scarcity of IPv4 address space, the allocation of unique addresses for experiments is difficult. Also, many CN are using their own address allocation sceme based on the limited private address ranges for internal routing and end users.

This non-coordinated allocation of IPv4 addresses is raising another challenge for the long-term objective to enable federation of CONFINE testbeds deployed in different CNs.

Relying on IPv6 seems the natural solution to this challenge but must consider the fact that only few existing CNs yet support native IPv6 routing.

G. Compatibility

Apart from the general requirement for the CONFINE architecture to be compatible with the different networking models employed by CNs, the CONFINE node system must be compatible to the original system used on nodes in the CN.

Because nodes in CNs are setup individually and may be based on proprietary and closed software systems, no modifications to the original community node firmware should be required and only few assumptions should be made on supported features or the concrete implementation of a node.

H. Role-specific APIs and Usability

The operation, deployment and usage of a testbed integrated in existing CNs involves different roles. In particular, the roles of testbed managees, node owners, and researchers have been identified.

In order to simplify the operations related to each role and prevent misconfigurations due to overlapping or missing responsibilities, a role-specific APIs and access control should be provided.

Finally, the tools and APIs provided for managing and deploying a CONFINE testbed should be as simple and intuitive as possible (for each involved role and despite the intrinsic complexity).

IV. CONFINE ARCHITECTURE

A CONFINE testbed consists of at least one *server* or *controller* and a set of *nodes* which can be spread over several community networks (CNs). A server is a normal computer which is publicly reachable inside of a CN. A node can be an integral part of a CN: in contrast with a boundary host, a CONFINE node can relay other CN traffic not related with CONFINE.

The design described in this section intends to facilitate the addition of CONFINE nodes in any location of a CN by connecting additional devices to existing CN ones with minimum or no CONFINE-specific changes to their configuration. At the same time, it provides researchers with *familiar Linux environments* with *root access* and *rich connectivity options* (see IV-A4) including:

- Simple NAT access (like that of most home computers).
- Public CN addresses (to serve incoming connections).
- Full L2 access to VLAN-tagged networks (to implement routing experiments).
- CN traffic sniffing (with anonymization).
- Raw L1 access (under certain conditions).

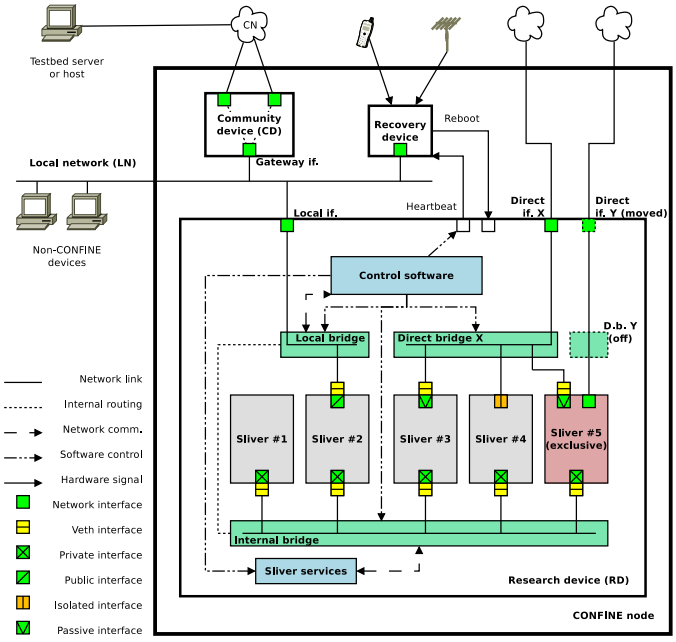


Fig. 1. The architecture of a CONFINE node.

A. Long-term vision

As shown in figure 1, a **CONFINE node** (node) consists of two or three devices: the **community device** (CD), the **research device** (RD) and an optional **recovery device**. This separation tries to preserve the stability of the CD (and thus of the CN) and is most compatible with any frmware it may run. All devices in a node are linked by a wired **local network** (LN), possibly shared with other non-CONFINE devices like CN clients using the CD as a gateway.

1) The community device: The CD is a *completely normal community network device* with at least two different network interfaces: a wired or wireless one connecting to the CN (a **community interface** with a **community address**) and a wired one to the LN (the **gateway interface** with its **gateway address**). It can act as a simple gateway for hosts connected to the LN, some of them with fixed addresses not assigned to other devices via DHCP or similar. This allows the use of a uniform address scheme for CONFINE elements in the node.

The node is accessed by testbed servers, administrators and researchers via the community interfaces of the CD, although node administration may occasionally proceed directly through the LN.

2) The research device: The RD is a relatively powerful device (compared to the CD) running a *customised OpenWrt firmware* provided by CONFINE which allows running several *slivers* at the same time as Linux containers [7]. Slivers have limited access to the device's resources and to one another, thus ensuring slice and network *isolation*. This is guaranteed by the **control software** run in the RD through tools like `ebtables`, `arptables`, `iptables`, `tc`, Open vSwitch...

The RD implements an **internal bridge** with an **internal address** that is the same in all testbed nodes and which belongs to a private network which does not clash with CN or LN addresses. The RD offers some basic **sliver services** (see IV-A5) on the internal address, including NAT gateway access to the CN.

The RD also implements a **local bridge** which connects to the LN through a wired interface (the **local interface**). The bridge is used for simple network layer access to the CN through the CD's gateway address, and the **local address** of the RD in it is fixed and used for testbed management and remote administration. For easy RD setup and local administration, the local interface may also sport a **recovery address** that is easily predictable or the same in all testbed nodes and that belongs to a private network which does not clash with CN or LN addresses (nor those of the internal bridge). A **debug address** which is also private and easily predictable can be used to access different RDs in the same local network for debugging purposes.

The RD may have additional **direct interfaces**, each one connected to its own **direct bridge**. These interfaces may be connected to the CN at the link layer and used for experiments below the network level (see IV-A4).

All the aforementioned bridges are managed by the control software in order to ensure network isolation between slices (i.e. between slivers running in the RD) as mentioned above, and to keep CN stability and privacy.

3) The recovery device: The node may include some simple recovery device whose purpose is to *force reboot* of the RD in case of malfunction using some direct hardware mechanism (like a GPIO port connected to the power supply of the RD), thus avoiding the need for physical presence for rebooting devices in places with difficult access.

The recovery device may get remote instructions from the CN (via the LN) or via different sensors, preferably based on wide-range technologies suffering from low interference and differing from those used by the CN (like ham radio signals, GSM calls or SMS). It may also receive a *heartbeat signal* kept by control software via some direct link like a serial line; when the recovery device misses a number of heartbeats, it reboots the RD.

A more advanced version of this device may help the RD boot some recovery environment (e.g. via PXE) or collaborate in some techniques for safe device upgrade and recovery (see [8]) to allow restoring its firmware to a known state.

4) Node and sliver connectivity: The research and recovery devices have fixed addresses in the LN and they need no routing protocol software to reach the CN: they simply use the CD's gateway address as a default gateway. Static configuration or DHCP suffice as long as the addresses they get are fixed.

In contrast, the connectivity of a sliver is determined by the network interfaces it sports, which are requested by the researcher at sliver definition time and depend on the interfaces provided by the RD and their features. Default routing configuration is explicitly controlled by the researcher to avoid traffic unexpectedly flowing through unwanted interfaces.

- Every sliver has a **private interface and address** whose

host side *veth* interface is placed in the internal bridge. The address is automatically assigned from the RD's private network, thus allowing access to the RD's internal address and sliver services (see IV-A5). The researcher may choose to use the latter address as the default gateway, in which case traffic is routed by the RD through the local bridge to the CD's gateway address after performing NAT. This allows *client access to the CN* but not connections from the CN to the sliver (similar to a home computer behind a NAT gateway on the Internet) *nor traffic between slivers*. Thus the sliver is ensured that there will be no incoming connections on that interface, obviating the need for firewalls or access control.
Sliver container 1 in figure 1 only has a private interface in the internal bridge.

- If the RD has been allocated some public addresses, the researcher may request a **public interface and address** whose host side *veth* interface will be placed in the local bridge, thus allowing access to the LN. The address is automatically assigned from the RD's public ones, and the researcher may choose to use the CD's gateway address as the default gateway. This allows *connections from the CN to the sliver* (similar to a computer directly connected to the Internet through a normal gateway).
  Sliver container 2 in figure 1 has a public interface in the local bridge.
- If the RD has a direct interface, the researcher may request a **passive interface** (with no address) whose host side *veth* interface is placed in the associated direct bridge, thus allowing direct access to the CN. Permission is granted only for *traffic capture* on the passive interface, which is anonymized by control software (e.g. an OpenFlow controller [9] on an Open vSwitch-based bridge [10]). This allows *CN traffic analysis* while respecting privacy.
  Sliver containers 3 and 5 in figure 1 have passive interfaces in direct bridge X.
- If the RD has a direct interface, the researcher may request an **isolated interface** (with no address), i.e. a VLAN interface on the associated direct bridge using one of the VLAN tags allocated to the slice at creation time. *Any kind of traffic can be transmitted and captured* on such an interface at the cost of being isolated from the CN at the link layer. This allows *routing experiments* to operate safely on close groups of CONFINE nodes.
  Sliver container 4 in figure 1 has an isolated interface on direct bridge X.
- If the RD has a direct interface, the researcher may request *raw access* to the interface's network device. The **raw interface** is *moved* into the sliver container and the associated direct bridge is disabled while the sliver is running. Since the sliver has *full physical control* on the network device, network isolation can not be guaranteed, so only that sliver is allowed to run in the RD. Moreover, this access can disrupt CN operation and privacy, so it should only be allowed *under very particular circumstances* (e.g. out of reach of the CN).
  Sliver container 5 in figure 1 *owns* the RD's direct interface Y.

Besides the included private interface, a sliver may be granted several interfaces using the local bridge or any direct bridge. Conversely, the local bridge or a direct bridge may provide several slivers with shared access to the same interface.

The setup of a sliver's networking is accomplished by populating its image with appropriate configuration files. Static configuration allows testbed servers to know the addresses of a sliver *a priori* (so they can be propagated to other slivers in the same slice) and it can be helpful in the configuration of routing, filtering and traffic control rules in the RD.

5) Sliver services: The RD offers on its internal address some basic services which become available to slivers through their private interfaces. This setup intends to relieve researchers from configuring those services themselves while providing some features tailored for CONFINE slivers (some of which are not feasible at the sliver level), all in a trivial and reliable manner by accessing an address which is always the same on an interface which is always available and closed to external access. Factoring services out of the slivers also saves resources and minimizes the changes of breakage by accidental misconfiguration on the sliver. Some examples of these (optional) services are:

- DNS: The RD acts as a name server, possibly with customized testbed- or slice-wide domains.
- SMTP: The RD acts as a mail gateway, possibly rewriting source mail addresses for easy sliver identification.
- A NAT gateway for slivers needing basic client-only connectivity to the network.
- A DHCP service which can be used by a sliver as a last resort to regain minimum (NAT) connectivity in case of lost network configuration.
- DLEP [11] can offer slivers information about link quality and speed information from network devices (useful for routing experiments using cross-layer information).
- Other CONFINE-specific services for retrieving testbed or slice information, or running privileged operations in the RD (like PlanetLab's vSys [12]).

6) Out-of-band access to slivers: Regardless of whether a slice has network access at a given moment, a researcher should be able to access any of its slivers through the RD's local address in a way similar to PlanetLab, allowing for remote command execution and file transfer for collecting experiment results.

7) The community container: This architecture also allows a low-cost version where the CD runs inside the RD as a **community container** (CC) thus saving on hardware at the price of stability. In this case each community interface is considered a direct interface in the RD, and the CC has a *veth* interface placed in its associated bridge. The gateway interface of the CC is a *veth* interface placed in the local bridge, and the RD's local interface can be kept for wired node

administration and for other non-CONFINE devices. The CC has few restrictions on the local and direct interfaces while slivers can still access them via passive or isolated interfaces (but not as raw interfaces). The CC may manage several direct interfaces in this way.

B. Current state

At the time of writing this document (End of Mai 2012) we are just finishing our first release of the CONFINE Node System (CNS) for RDs [13]. The main objective of this first version is to provide a proof-of concept for selected technologies and be able to gather experience from the execution of first experiments on real deployments.

CNS is build on top of the upcoming "Attitude Adjustment" release from OpenWrt [14]. The decision to base our work on the current developer branch (trunk) of OpenWrt was a trade off between the requirement to provide up-to-date implementations for quite a number of relatively new Linux networking and virtualization technologies and the lack of maturity that comes with development branches.

The current implementation provides role-specific APIs and user interfaces for node owners, testbed management, and researchers.

By default, the task of integrating a new CONFINE RD into an existing CN is under the control of the node owner. After the initial flashing of the RD with our firmware images (implementing the confine node system for a pre-selection of supported X86 hardware like Alix or Comell), a node system must be customized for its specific deployment. Therefore, a small set of testbed, and node-specific attributes (NODE-ID, public keys, ..) must be exchanged between the node owner and the testbed operators via a central web portal and used to complete the UCI-based configuration template of the RD. The node-management functions confine_node_enable/disable can be used at any time to cleanly activate or deactivate the participation of the RD in the CONFINE testbed.

The abstract concept of a sliver is realized via virtualization. To provide researchers with a common experimentation environment we decided to use Linux Container (LXC) as a very light-weight virtualizatinon solution. This allows us to grant researchers root access to an OpenWrt or Debian guest system (his slivers), but running completly controlled by the CONFINE RD host system. LXC User-space tools have been ported for OpenWrt as a new package and are used by the CNS to control LXC container internally. This "sliver" environment given by the container OS can be further extended by the researcher with additional tools, data, and init scripts to automatically execute the processes that define his experiments. Interactive access to slivers is provided at any time via ssh and the CONFINE management (tinc IPv6 overlay) network.

The role of the testbed operators is to manage the deployment and life cycle of experiments (via the concepts of slices and slivers). The following set of functions were implemented as bash scripts for remote invocation (via ssh and the CONFING management network) from a central server of the testbed.

- confine_sliver_allocate is used to allocate the resources (like local namespaces, IP addresses, storage) on a RD that are necessary to execute an experiment. The function requires a sliver description in UCI syntax specifying the required interfaces, addresses and OS type. On success, it returns precise information about the allocated sliver attributes. The attributes of all slivers of a slice must be collected by the server for later summarization and provisioning as slice attributes during the deployment of each sliver. Thereby, each sliver has access to sliver attributes (eg IP addresses) of all other slivers of a slice before it is started.
- confine_sliver_deploy is used to setup the sliver environment in a RD. Therefore a linux container and root file system is created and customized according to RD, slice, and sliver specific attributes. The function requires the slice attributes in UCI syntax as input.
- confine_sliver_start, confine_sliver_stop, confine_sliver_remove can be used to start, stop, and remove the LXC container related to a given sliver (in the RD where the command is invoked). The last function does also release all allocated resources reserved for that sliver.

1) The IPv6 overlay: Creating testbeds that span several community networks (CNs) is drastically hampered by their lack of widespread IPv6 support. They normally use private IPv4 addressing that (in spite of initiatives like Free Networks [15]) can not be trusted to be compatible across CNs. Also, the diversity of CN devices using IPv4 mean that research devices may sit behind NAT boxes or firewalls further limiting their reachability. IPv4 address scarcity also keeps the testbed from using a clean and uniform address scheme for their elements.

IPv6 migration solutions like 6to4 [16], 6in4 [17], or Teredo [18] either have problems with protocol 41 handling in NAT boxes or firewalls, or use host IPv4-based addresses which can clash between CNs. Together with most VPN solutions they get complicated to setup in a mesh-like cloud, or use centralized architectures which alter the resulting topology and can turn the VPN server into a bottleneck.

A CONFINE testbed works around these problems by creating an **IPv6 overlay** based on the *tinc* [19] VPN software. *Tinc* allows setting up a mesh network where data is exchanged directly between endpoints with the VPN software taking care of routing traffic and propagating endpoint information, thus needing minimal configuration.

As shown in figure 2, a *tinc* daemon in a testbed node only connects to the testbed server or a CN testbed gateway. Gateways connect between themselves through another network like the Internet. Together they form a single routed IPv6 network where individual hosts (servers, gateways and clients) and subnetworks (node devices and slivers) become available to one another.

Since the overlay makes all elements in the testbed easily reachable through predictable IPv6 addresses with automatic

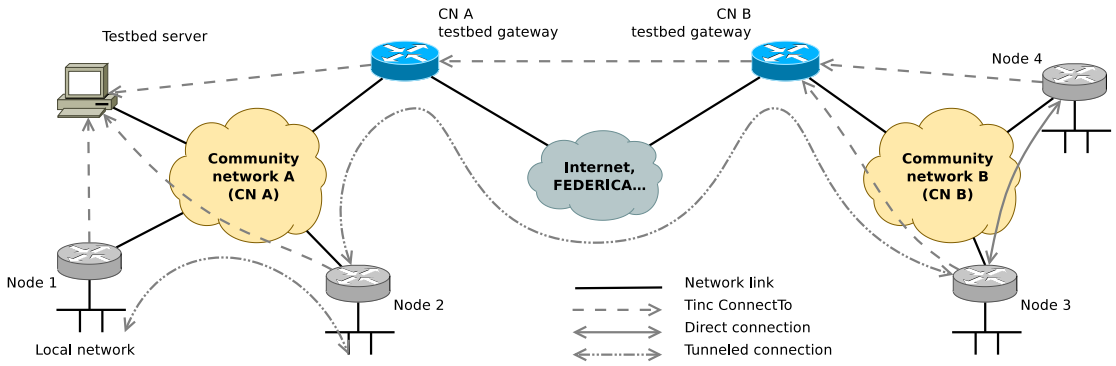


Fig. 2. The IPv6 overlay spanning two community networks.

encryption and authentication between endpoints, the IPv6 overlay is used as a **management network** while CNs gain widespread IPv6 support.

C. Next steps

3.3 next steps - 2n year implementation (between CONFINED and envisioned)

D. Community-Lab integration in existing community networks

What makes the Community-Lab testbed different from existing ones is the integration with actual community networks (CNs). The nodes belonging to the testbed are deployed among the community network to be able to perform experiments in a real enviroment with real users. However these networks are indeed production networks, so the experiments must have some limitations or provide some degree of protection to community network owners. For instance, nodes placed in the core of the network cannot, as a general rule, execute link layer experiments because it can easily disrupt the corrent operation of the production environment.

Community networks have multiple owners, a property called *distributed property*, which means that each user is the owner of his own device. In a city, where people live in buildings, the users normally share a single device placed on the roof – a good practice though not always possible.

A typical community network deployment looks like figure 3a. The green lines represent WiFi links. Most of these networks are using this technology because it is cheap, easy to use and can be easily obtained. However it might be any other kind of technology such as WiMax, Ethernet or Optical Fiber depending on the owner needs.

The integration of the Community-Lab devices inside existing CN requires the collaboration of at least one part of the community. To make it possible, the CONFINE project provides additional resources in exchange to cover some of the costs like new links or new nodes. We envision three different levels of integration.

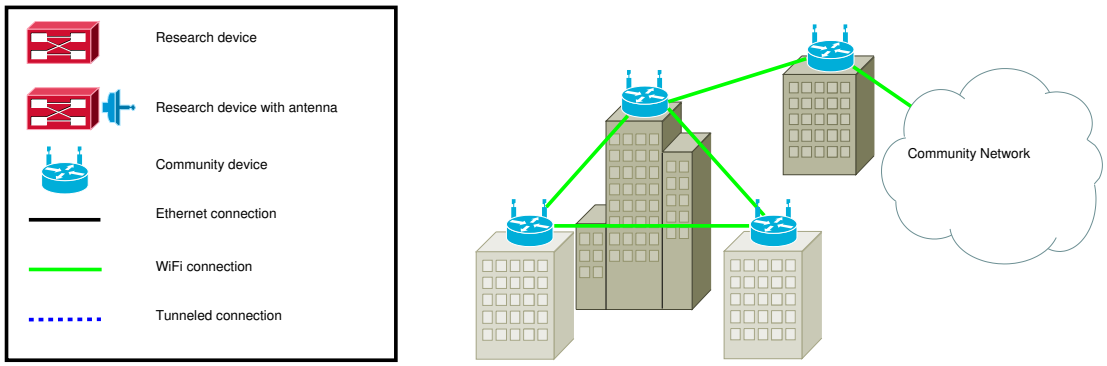
1) Single research devices connected to an existing node: As shown in figure 3b (left), a research device (RD) without antennas is adopted by a community user not necessary related with the project. Its device will be connected to the community node (which is owned by the person who adopt it) through a ethernet connection. The research device will obtain a valid community network IP – depending on the availability of a DHCP server on the community device (CD) – and it will be reachable from the rest of the network and from the Community-Lab server. The experiments between research devices can be performed, restricted to application layer experiments because of the lack of direct access to the physical layer. If the owner allows it, a WiFi antenna might be added to the research device to permit researchers only obtain valuable link layer information, which could be usefull to perform long term studies on the real usage of such links.

In this case the CONFINE project only needs to install a single device (the RD) because it uses a existing CD owned by the user.

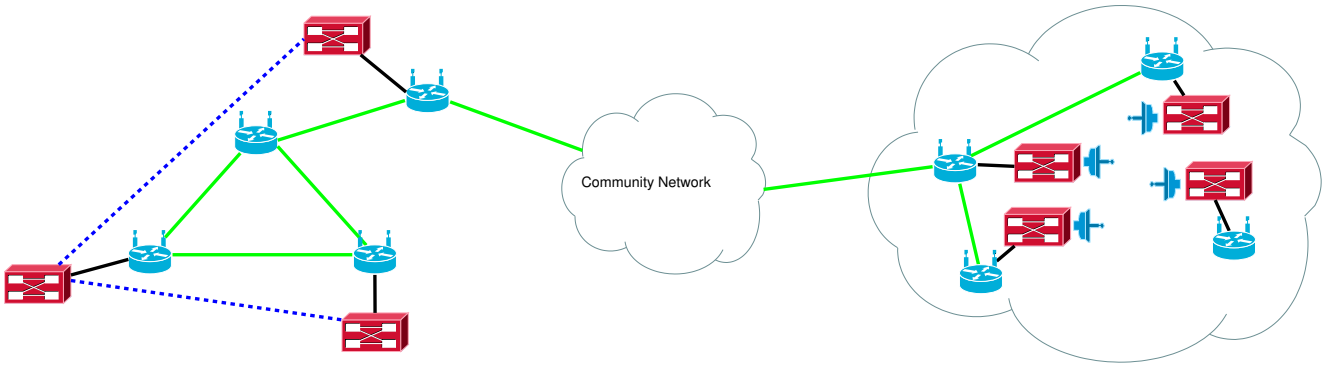
2) Nodes to extend the current network: One of the explicit objectives of the CONFINE project is to support the growth and usage of this *distributed property* network model. In some cases, where the community network needs one or several extra links to improve performance or availability, the project will contribute to provide this extra resources. These resources will be placed in a community member location, and so the adoption of a research device (following the integration model before described) will be a requirement. In this case the CONFINE project will add a new CD and the common RD.

3) Multiple nodes (cloud) deployed in a specific scenario: In some specific scenarios controlled by a CONFINE partner – i.e. University Campus – an entire cloud may be deployed. As shown in figure 3b (right), it implies the installation of several nodes. The CD must be a node compatible with the existing CN to be able to connect to. The RD may contain one or several WiFi antennas and radio devices depending on characteristics of the experiments to perform. Having several radio devices will allow researcher to execute any kind of link layer experiments given that the access to the physical layer will be exclusive.

If the set of links (between the nodes) are interesting for the community, the users may make use of them. Moreover, they must be aware that such a cloud is a experimental network and its availability and stability cannot be guaranteed. However,



(a) Community network deployment



(b) Research devices connected to an existing node (left) and a complete confine cloud with community and research devices (right)

Fig. 3. Community-Lab integration with existing community networks.

this bilateral open access between community network users and researchers can provide highly valuable information to evaluate new protocols and services.

In this case the CONFINE project will add the new CD, the common RD and any other hardware needed (like wires, metallic mast, electricity box, etc.).

V. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] R. Sombrutzki, A. Zubow, M. Kurth, and J.-P. Redlich, "Self-organization in community mesh networks the berlin roofnet," in *Operator-Assisted (Wireless Mesh) Community Networks, 2006 1st Workshop on*, sept. 2006, pp. 1 –11.

[2] D. Aguayo, J. Bicket, S. Biswas, R. Morris, B. Chambers, and D. De Couto, "Mit roofnet," in *Proceedings of the International Conference on Mobile Computing and Networking*, 2003.

[3] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/956993.956995

[4] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester, "The w-ilab testbed," in *Proceedings of TRIDENTCOM*, 2010.

[5] A. Anadiotis, A. Apostolaras, D. Syrivelis, T. Korakis, L. Tassiulas, L. Rodriguez, and M. Ott, "A new slicing scheme for efficient use of wireless testbeds," in *WINTECH'09: Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*, 2009, pp. 83–84.

[6] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 3. IEEE, 2005, pp. 1664–1669.

[7] "Linux containers." [Online]. Available: http://lxc.sourceforge.net

[8] "Safe upgrade of embedded systems." [Online]. Available: https://archive.fosdem.org/2012/schedule/event/safe_upgrade_of_embedded_systems.html

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[10] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," *Proc. HotNets (October 2009)*, 2009.

[11] S. Ratliff, D. Satterwhite, S. Jury, G. Harrison, and C. Cisco, "Dynamic link exchange protocol (dlep)," 2012.

[12] S. Bhatia, G. Di Stasi, T. Haddow, A. Bavier, S. Muir, and L. Peterson, "Vsys: A programmable sudo," in *Proceedings of the 2011 USENIX conference on USENIX annual technical conference*. USENIX Association, 2011, pp. 22–22.

[13] "A hack: 1st milestone of the confine project," May 2012.

[14] "The openwrt embedded development framework," 2012. [Online]. Available: http://www.openwrt.org

[15] "Free networks association." [Online]. Available: http://freenetworks.org/

[16] B. Carpenter and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," RFC 3056 (Proposed Standard), Internet Engineering Task Force, Feb. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3056.txt

[17] E. Nordmark and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers," RFC 4213 (Proposed Standard), Internet Engineering Task Force, Oct. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4213.txt

[18] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," RFC 4380 (Proposed Standard), Internet

Engineering Task Force, Feb. 2006, updated by RFCs 5991, 6081. [Online]. Available: http://www.ietf.org/rfc/rfc4380.txt
[19] "Tinc vpn."