Live Video Streaming for Community Networks, Experimenting with PeerStreamer on the Ninux Community

Leonardo Maccari
DISI - University of Trento
leonardo.maccari@unitn.it

Luca Baldesi
DISI - University of Trento
luca.baldesi@unitn.it

Renato Lo Cigno
DISI - University of Trento
locigno@disi.unitn.it

Jacopo Forconi
ARCI Firenze
forconi@arci.it

Alessio Caiazza
Ninux Firenze
ac@firenze.ninux.org

ABSTRACT

P2P Live video streaming could be a distinguishing feature of Community Networks, due to the affinity to both technical and social characteristics of such networks. It can help binding together communities, it provides a good means for inclusion of people as well as to deliver information and local events. Video streaming in community networks, however, is still problematic; this work describes how to customize PeerStreamer, an open source P2P video streaming platform, to an existing Community Network in the city of Florence, Italy. The paper exposes the motivations that make PeerStreamer a perfect match with the philosophy and the technical features of a community network and describes how the community network of Florence can be a very good testbed given the mixture of technical and social skills that animate it. The proposed adaptation and implementation exploits a so-far underused feature of PeerStreamer: the possibility of separating the streaming engine from the play-out part on different hosts. This feature makes it possible to install the streaming engine, which is very efficient and has a very small memory footprint, directly on the community network routing nodes, so that the streaming topology can be adapted to the community network topology by directly accessing routing information. On the other hand, the player can run on standard PCs and use standard interfaces to access the stream.

Categories and Subject Descriptors

C.2.4 [**Computer System Organization**]: Distributed applications

Keywords

P2P video streaming, community networks, wireless mesh networks, cross-layer design

1. INTRODUCTION

DIYNetworking'15, May 18, 2015, Florence, Italy.

http://dx.doi.org/10.1145/2753488.2753491 .

Video streaming has always been the killer application of networks, though until recent years it was first a chimera and then a dream. Today, instead, on-demand streaming is a standard, albeit demanding, network application on the Internet[1], while TV-like broadcasting is available in many places on specialized IP overlays supporting multicasting[2]. Video streaming is enabled by powerful cloud- or Content Delivery Networks (CDN)- based server architectures matched to the typical Internet Service Provider (ISP) backbone plus ADSL network architecture.

Community Networks (CNs) do not match the typical ISP network architecture and they don't have powerful data center or cloud services. Moreover, they normally don't have a high capacity interconnection with the Internet, thus video streaming remains not only a killer application for CNs, but a killer tout-court, making it very difficult to deliver video streams either coming from the Internet or locally generated.

However, there is an increasing demand for streaming services in WCNs. In Guifi CN[3] for example there are fourteen VoIP servers, five video conference servers and nine radio broadcasting servers. CNs are mesh networks with symmetric links, and thanks to modern WiFi devices they normally have fairly large capacity. This characteristic, together with the bottom-up, cooperative approach that sustains them, makes them a perfect match for P2P (peer-to-peer) technologies. P2P video streaming attracted a lot of interest in the mid 2000s, but then was mostly abandoned in favor of more traditional techniques, albeit several commercial services still exists[4].

The software and protocols supporting commercial P2P TVs are definitely not a good match for bottom-up voluntary-based CNs (explaining why goes beyond the scope of this paper). Some of the authors of this paper, instead, main-

[1]YouTube, Netflix, Vimeo, are examples of on-demand video streaming, normally provided as a web service.
[2]IP-TV differs from a web-service because it is provided as an unbundled service separate from Internet access, although normally commercially associated to it. A partial list of IP-TV providers can be found at `http://www.skytide.com/products/iptv-providers-list`
[3]http://guifi.net
[4]See for instance LiveP2PStream `http://www.livep2ptvstream.com/`, SopCast `http://www.sopcast.org/`, or PPTV `http://www.pptv.com/`

tain as open source software project the heritage of NAPA-WINE [5], a very successful EU-FP7 project[5].

PeerStreamer, the name of the P2P streaming system, is one of the few available and fully configurable open source platforms. It is based on a generic mesh topology that can be adapted to the underlay topology requirements [10], it includes advanced solutions for optimal video chunk scheduling [1], and reliable chunk delivery negotiation based on push/pull, offer/select protocols [7]. PeerStreamer is shortly described in Sect. 3.1 for what is functional to this paper.

The contribution of this paper lies in describing how a modern, open source P2P streaming implementation matches perfectly the ethical goals and technical characteristics of CNs and how PeerStreamer distribution core could be integrated in the CN routing nodes, while deploying the graphical user interface on users PCs, laptops, or even a modern TV set. The Italian CN of Ninux, based in Florence gives a perfect chance to test with PeerStreamer. It is a small scale network with a lively community ready to include new applications in its network infrastructure to attract new people, and it is in direct contact with ARCI[6], a large Italian ONG (Non-Governmental Organization) that is interested in using Ninux to enlarge the audience of the live events it organises. We are actively working to implement the described architecture and we plan to present experimental results from this implementation in a near future.

The rest of the paper is organized as follows, in Sect. 2 we introduce Ninux and Ninux Florence, in Sect. 3 we describe PeerStreamer and in Sect. 4 we explain the possible design we will use. Finally we draw the conclusions.

2. NINUX: THE LARGEST ITALIAN CN

Ninux[7] is the largest Italian CN, with about 300 nodes installed in the whole Italian peninsula. Ninux was bootstrapped in Rome more than ten years ago as a small experiment led by local hackers, and today it is composed of various Ninux *islands*, among which the Rome island contains roughly 2/3 of the total number of nodes. The islands are not directly connected, and each local community is independent, all of them embrace the Ninux Manifesto that is largely based on the Pico Peering agreement used by several CNs[8]. Ninux was born to build a cooperative, neutral network with universal access. Ninux is not a cooperative ISP, it doesn't have a formal association that represents it, nevertheless through its members it achieved several goals, it is a recognized Autonomous System, and it is currently part of the Italian network of research (GARR), through which it is connected to the Internet Exchange Point of Rome. Ninux has been already the subject of some research works [9, 8], and it currently participates to the CONFINE FP7 FIRE project[9], the largest European research project that studies community networks.

[5]See `http://www.napa-wine.eu/cgi-bin/twiki/view/Public` for a full description of the project, and PeerStreamer homepage `http://peerstreamer.org` for the details on the software development

[6]The Italian acronym stands for Associazione Ricreativa e Culturale Italiana, `http:\\www.arci.it`

[7]See `http:\\www.ninux.org` and `\http:\\www.firenze.ninux.org` for the Florence island

[8]See `http://www.picopeer.net/`

[9]`http://confine-project.eu`

2.1 Ninux in Florence

The Florence portion of Ninux is the second largest of the whole Ninux network after the main island of Rome. It was started in late 2012 and, at the time of writing, it is comprised of 21 nodes and it is animated by a stable community of about ten people. The network nodes are installed primarily in private houses, a few more are hosted by associations. Fig. 1 shows the current state of the network, green nodes are active ones, orange nodes are "*potential nodes*", that is, locations in which people have manifested the interest of placing a new node but could not join yet. At the current state there are few applications used by the network community. People use the network to share files, to access remote servers they control in different locations, or to share an ADSL connection between two houses. There is an experimental voice over IP server running.

The community is lively, and often participates at public events to promote the network and to attract participants coming from heterogeneous experiences. One of the goals of Ninux in Florence, indeed, is to quickly move from a geek-only experiment (that is the genesis of many CNs) to a network involving a wider segment of population. One way of achieving this goal is to focus on providing cheap Internet access, and turn the network in a cooperative ISP, which attracts the still many digitally divided people. On the other hand, the experience of Ninux and other CNs tell that those who are interested in cheap Internet access may not share the network principles and will most likely become *clients* of the network, instead of participants. Ninux Florence tries to avoid such drawback. In fact, there is no focus on Internet access and the community tries to involve new people interested in the funding values of the network itself. Of course the more applications will be deployed in the network, the easier it will be to attract external people.

Ninux Florence has a physical headquarter that is hosted in a building managed by the ARCI ONG. ARCI is one of the largest Italian ONGs, with more than one million people affiliated and almost 5000 points of presence in Italy. The building that hosts the weekly meeting of Ninux is called *exfila* and is located in the south-east part of the city, pointed by a red arrow in Figure 1. Among other things, exfila is used to organize concerts and events, as it features a small auditorium. Ninux has an active node in exfila.

The idea described in this paper comes from the need of the Ninux community to increase the number of the internal applications, together with the availability of contents that ARCI generates in exfila, such as concerts and seminars, and that would be very hard if at all possible to publish live on the Internet using the ADSL connection present in exfila. Being able to stream the live video events in the Ninux network would be beneficial also for ARCI: it would achieve a larger audience, and, in the future it would make it possible to connect other ARCI buildings to Ninux in order to receive the live events from exfila. Furthermore, ARCI hosts an FM radio in one of his buildings, so it may be possible to stream events from exfila to the radio station, and from there stream the audio in FM. Conversely, live contents generated by the radio could be spread also on the CN, as audio is a sub-part of video and PeerStreamer can be easily adapted.

3. P2P STREAMING AND PeerStreamer

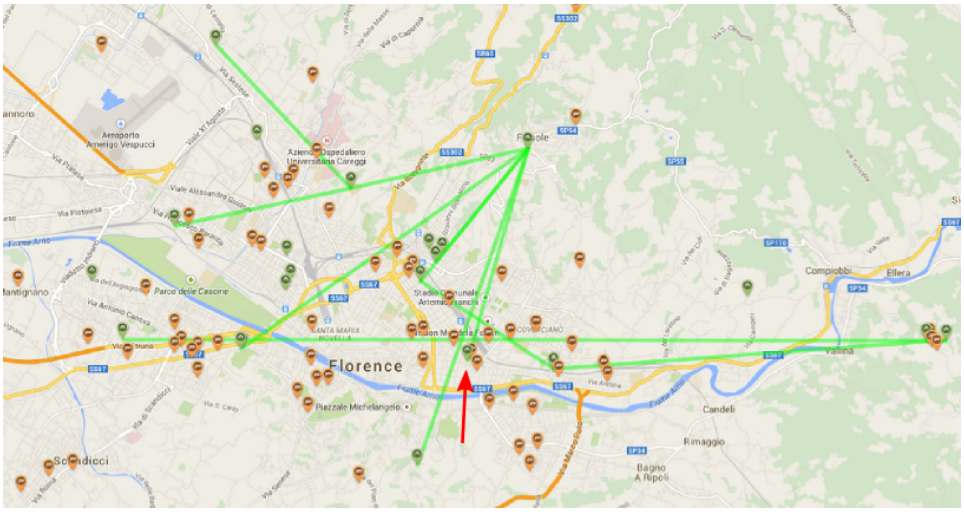

Figure 1: The topology of the Ninux network of Florence, the red arrow points to the headquarter of Ninux Florence.

P2P live streaming matches both the technical constraints and the philosophy of a CN. From a technical point of view, centralized cloud-based streaming does not suit a CN, since every video stream is independently generated for each client. If the streaming service is placed outside the CN, then the connection that the CN uses to access the Internet will be overloaded by the potentially tens of clients that request a unique video stream. Even if the source is placed inside the CN, the scalability can not be granted because the links connected to the source node will be soon overloaded when the number of clients increases. PeerStreamer instead subtracts load from the source and distributes the impact on various network links, so that the video distribution becomes sustainable even increasing the number of clients. Its application to CNs has already been tested and showed promising results [3, 2]. Under a social point of view, live broadcasting of community user ideas and events fosters the opinion exchange within the community and can be a means for information propagation in a way much more tied to the citizens with respect to what usually media do. It can be used to stream concerts, seminars, lessons with a completely bottom-up approach without the need to involve any external cloud-based or commercial platform.

3.1 PeerStreamer

PeerStreamer is an open source platform for live peer-to-peer video streaming. It has been developed as part of the European NAPA-WINE project and is capable of exploiting a peer-to-peer mesh overlay to deliver media contents optimizing the receiving delay. PeerStreamer is composed of different customizable parts.

Much of the interest around PeerStreamer is due to its capacity of scaling over thousands of peers keeping a very low latency in the packet delivery. Such characteristic was proved during a test campaign conducted on PlanetLab [6], a research network worldwide spread that encourages the development of new emerging services. Thanks to PeerStreamer scalability, as the number of peers grows, the delivery delay of each packets grows only logarithmically. In practice even with more than two thousands peers the transmission only requires a few seconds to reach the farthest peer: in television broadcasting this is defined as *live*.

This important feature is achieved through the epidemic data diffusion scheme implemented in PeerStreamer. The data source will fragment the video stream in basic units ("chunks") and inject in the overlay only few copies of each chunk. The receiving peers will redistribute those packets to other peers and so on until every peer have received its own copy of the chunks [4].

Other peer-to-peer applications are designed to maximize the expected throughput and are not suitable for live streaming due to the time constraints of the overall transmission on the overlay. PeerStreamer instead is intended to minimize the receiving delay.

PeerStreamer can be logically split in three main components; the *source tools* which inject the media contents in the overlay, the *streamer* which is the actual responsible of setting up the peers overlay and spread the contents across the network, and the *player application* which takes the video chunks from the local streamer and display the video to the user. The resulting PeerStreamer logical overlay is depicted in Fig. 2.

3.2 The Web Interface

PeerStreamer supports the separation of the streaming function, the software component that takes care of trading video chunks, from the playback function that visualizes the video stream. The second component can be substituted by any compatible players, such as the widely used Video Lan Client (VLC)[10]. One architectural choice that is currently under design is the introduction of a web-based interface which has been proved to be generally more appealing for the users and could grant a higher degree of flexibility.

Currently, each streamer can manage only one video stream. As a future development, we can imagine that multiple streams,

[10]See `http:\\www.videolan.org`

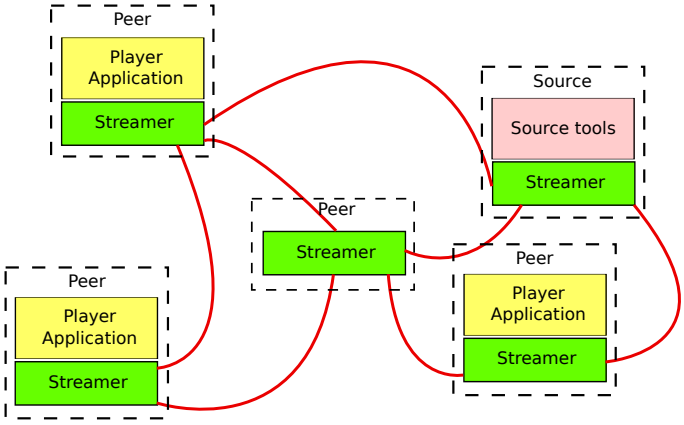

Figure 2: PeerStreamer network architecture. The source tools inject the video packets in the overlay, the peer-to-peer overlay is composed of the streamers and the player applications perform the playout. Not all the peers need to implement the player.

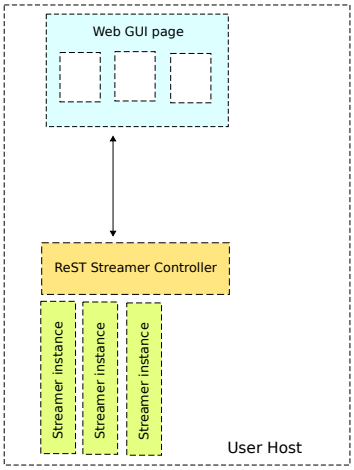

Figure 3: Possible PeerStreamer architecture.

generated from multiple sources at the same time could be running on the same network. We thus include in the architecture a controller of the various streamer instances. The interactions with the controller could be based on the web protocol (HTTP) with Representational State Transfer (ReST), a lightweight and efficient architectural style that grants a high degree of interoperability with other web applications. A ReST oriented architecture is sketched in Fig. 3. Each user has one (or more than one) software daemon running on his host (the streamer) responsible of the reception and sharing of video frames. The web interface allows, through HTTP transactions, the creation of new streamers to access to other videos and the playback on the web interface itself, using the video plugin already installed in the user's browser. Potentially many playback objects can fit in the web interface, a typical application could be a video conference.

Such architecture allows running the streamer in a different physical host than the playback function and gives the chance to realize many different set-ups. Using a web interface would introduce also the possibility to easily mesh-up and integrate other web-based services. For instance it could be possible to add *social* features like a messaging chat shared among the users or introduce the possibility of showing material related to the video.

4. PEERSTREAMING IN NINUX

In this section we describe four potential configurations based on the PeerStreamer architecture for the Ninux Florence CN, as well as for any other CN. The configurations are listed in growing complexity, in each one we give for granted the presence of a video source in the network and we describe only the configuration of a single client. All configurations are compatible and can be mixed in different clients in the network.

The first configuration is the simplest and the one that is already possible given the current capabilities of PeerStreamer. It embeds all the components in the user PC, which needs to be able to access the IP address class of the CN.

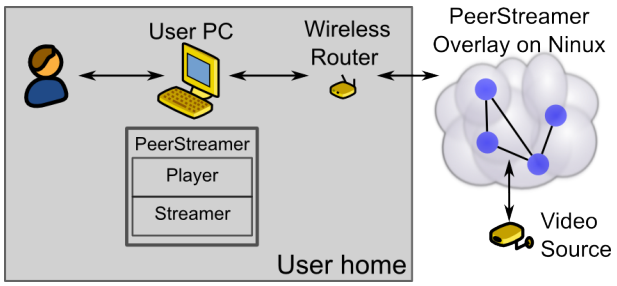

Figure 4: Configuration 1, all the software is included in user's PC.

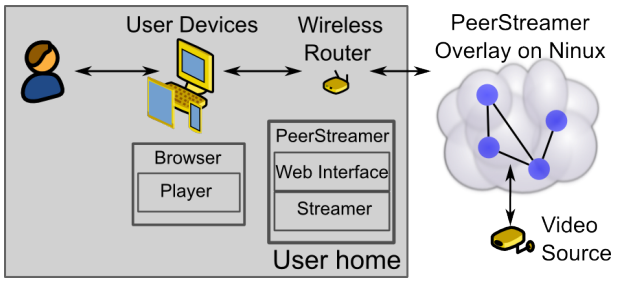

Figure 5: Configuration 2, PeerStreamer is embedded in the wireless router used to access the Ninux network, user accesses video via browser.

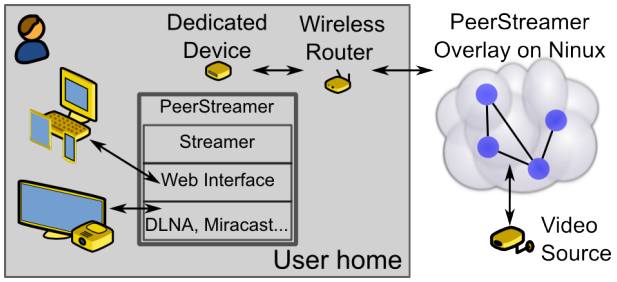

Figure 6: Configuration 3, PeerStreamer runs on a dedicated device, and other peripherals access the stream via standard protocols.

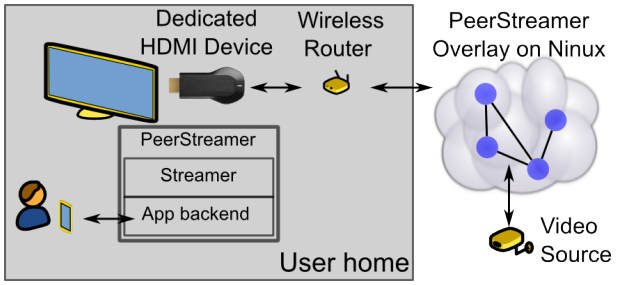

Figure 7: Configuration 4 PeerStreamer runs on a dedicated device connected via HDMI to a smart-TV. Such device is controlled via a dedicate mobile App.

Fig. 4 depicts such configuration that is normally used when PeerStreamer operates over the Internet, it is the simplest one and can be used by the user without noticing if the source is in Ninux or on the Internet, assuming Internet access is available.

A second configuration is the one depicted in Fig. 5 which splits the role of the streaming function from the role of the playback. The streamer and the controller can be placed directly on the wireless device used to access Ninux (the Ninux node) and the user can access the stream pointing any web browser to the web server in the node. The advantage of such configuration is that the streaming engine can be embedded directly in the Ninux nodes when they are installed, so that every node in the network comes with PeerStreamer preinstalled. Moreover, multiple devices (PCs, tablets, smartphones) can access the video at the same time. The controller will then be accessed by the web interface to instruct the streamer to participate to the diffusion of a video from a certain source. Once this is done every device in the house of the user can access the video. Note that the streaming engine is a very lightweight application written in C language, with a minimal memory fingerprint, so it can run even in a constrained environment such as wireless access point.

To realize such configuration some development is needed: the controller and the web interface need to be realized. Moreover, PeerStreamer needs to be ported to the operative systems and hardware of common wireless router (such as Tp-Link or Ubiquiti routers, that are the most used in Ninux and use embedded ARM processors). PeerStreamer currently supports x86 platforms, the GNU/Linux, Windows and MacOS operating systems.

A third configuration, that would require less effort would be to use a separated device in the home network to run PeerStreamer and is reported in Fig. 6. Currently, cheap (less than 100€) x86 devices are present in the market and could be used as a small peer-to-peer media center. The benefit of such approach would be that the dedicated device, being a more computationally powerful platform without any other concurring tasks could offer the video streaming in different formats, in order to be accessible to various devices. Among these, we can imagine that protocols such as DLNA or Miracast could be used to serve smart-TVs or smart-projectors, so that the user could enjoy the video even more comfortably.

Such approach is perfectly compatible with the one used by the CLOMMUNITY project[11], which develops the Cloudy linux distribution[12] for usage in CNs. PeerStreamer is already one of the services introduced in Cloudy, which could be a building block for the dedicated media center. Still, the control of the streamer (to join a certain swarm) would take place from a web page via a web browser

Finally, the forth configuration, depicted in Fig. 7 pushes even further on user friendliness. The dedicated hardware device is one of the embedded computers with HDMI output (on the model of Google Chromecast or the Intel "Compute Stick"). Such devices are multi-core platforms with Wi-Fi support of the size of a USB stick that can be plugged directly into the HDMI plug of a TV. The functional model is the same of the previous configuration but the video will be directly transferred to the TV. For commodity, such device would not be controlled via a web interface, but with a mobile phone App connected to the home Wi-Fi. Such configuration would be the easiest to use, and implies the development of various components that are currently not available.

5. CONCLUSIONS

This paper described a work-in-progress project that merges the outcome of a previous research project with the needs of an existing community network. We outlined how P2P streaming can be a killer (in all senses) application for CNs, and how the Florence Ninux community can benefit from such technology, being in direct connection with the ARCI ONG, which can offer live streaming of local contents. We described potential architectural models some of which we will test with the joint effort of all the contributors of this paper: academia, Ninux and ARCI. Since PeerStreamer is an open source software such models can be freely replicated also in other CNs.

6. REFERENCES

[1] L. Abeni, C. Kiraly, and R. Lo Cigno. On the Optimal Scheduling of Streaming Applications in Unstructured Meshes. In *NETWORKING '09: 8th Int. IFIP-TC 6 Networking Conference; LNCS 5550*, pages 117–130. Springer-Verlag, 2009.

[11] `http://clommunity-project.eu/`
[12] See http://cloudy.community/

[2] L. Baldesi, L. Maccari, and R. Lo Cigno. Improving P2P streaming in community-lab through local strategies. In *10th IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 33–39, 2014.

[3] L. Baldesi, L. Maccari, and R. Lo Cigno. Live P2P streaming in CommunityLab: Experience and insights. In *13th IEEE Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pages 23–30, 2014.

[4] R. Birke, C. Kiraly, E. Leonardi, M. Mellia, M. Meo, and S. Traverso. A delay-based aggregate rate control for p2p streaming systems. *Computer Communications*, 35:2237 – 2244, 11/2012 2012.

[5] R. Birke, E. Leonardi, M. Mellia, A. Bakay, T. Szemethy, C. Kiraly, R. Lo Cigno, F. Mathieu, L. Muscariello, S. Niccolini, J. Seedorf, and G. Tropea. Architecture of a network-aware p2p-tv application: The napa-wine approach. *IEEE Communications Magazine*, 49:154–163, 06/2011 2011.

[6] 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.*, 33(3):3–12, July 2003.

[7] R. Lo Cigno, A. Russo, and D. Carra. On Some Fundamental Properties of P2P Push/Pull Protocols. In *2nd IEEE Int. Conf. on Communications and Electronics (ICCE 2008)*, pages 67–73, June 2009.

[8] L. Maccari. An analysis of the Ninux wireless community network. In *9th IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–7, 2013.

[9] L. Maccari and R. Lo Cigno. A week in the life of three large Wireless Community Networks. *Ad Hoc Networks*, 24:175–190, 2015.

[10] S. Traverso, L. Abeni, R. Birke, C. Kiraly, E. Leonardi, R. Lo Cigno, and M. Mellia. Neighborhood Filtering Strategies for Overlay Construction in P2P-TV Systems: Design and Experimental Comparison. *IEEE/ACM Transactions on Networking*, on-line(99):1–14, March 13 2014.