Improving P2P Streaming in Wireless Community Networks[☆]

Luca Baldesi[a], Leonardo Maccari[a,*], Renato Lo Cigno[a]

[a] *Dipartimento di Ingegneria e Scienza dell'Informazione (DISI)*
Università di Trento, Via Sommarive 9, Povo, Trento, Italy

Abstract

Wireless Community Networks (WCNs) are bottom-up broadband networks empowering people with their on-line communication means. Too often, however, services tailored for their characteristics are missing, with the consequence that they have worse performance than what they could. We present here an adaptation of an Open Source P2P live streaming platform that works efficiently, and with good application-level quality, over WCNs. WCNs links are usually symmetric, or at least asymmetry is not by design as it is in ADSL, and a WCN topology is local and normally flat (contrary to the the global Internet), so that the P2P overlay used for video distribution can be adapted to the underlaying network characteristics. We exploit this observation to derive overlay building strategies that make use of cross-layer information to reduce the impact of the P2P streaming on the WCN while maintaining good application performance. We experiment with a real application in real WCN nodes, both in the Community-Lab provided by the CONFINE EU Project and within an emulation framework based on Mininet, where we can build larger topologies and interact more efficiently with the mesh underlay, which is unfortunately not accessible in Community-Lab. The results show that, with the overlay building strategies proposed, the P2P streaming applications can reduce the load on the WCN to about one half, also equalizing the load on links. At the same time the delivery rate and delay of video chunks are practically unaffected.

Keywords: Wireless Community Networks' P2P Live Streaming; Video Streaming; Topology Optimization; Cross-Layer Design; Performance Evaluation.

[☆]This work was partially funded by the European Unions 7-th Programme for research, technological development and demonstration under grant agreement No. 288535 "CONFINE" – Open Call 1 project "OSPS". During the work preparation, Leonardo Maccari has been partially supported by the project "Wireless Community Networks: aspetti giuridici, sociologici e tecnologici di un nuovo fenomeno d'aggregazione sociale" funded by *Fondazione CaRiTro*.

[*]Corresponding author

Email addresses: `luca.baldesi@unitn.it` (Luca Baldesi), `leonardo.maccari@disi.unitn.it` (Leonardo Maccari), `renato.locigno@disi.unitn.it` (Renato Lo Cigno)

1. Introduction

Wireless Community Networks (WCNs) are infrastructures built by the people for the people [1], with a bottom-up approach bringing broadband services to communities, rural and urban as well. They have became an important reality in the landscape of telecommunication and ICT services, and in some cases involve hundreds or thousands of nodes and people[1]. They spread over large regions and some effort is currently spent in planning a sort of interconnecting federation [2]. The continuous growth of these networks and the services experimented on them represent a major swerve from the dominating paradigm of consolidation and centralization of the Internet. The increasing traffic generated by their users requires careful studies on the resources involved and the performance achievable by applications, with a special focus on services that have been ostracized in the commercial Internet. Among these, P2P applications, divested from their illegal aura, suite very well the WCN philosophy (delivering user-generated contents to other users) and also its internal, distributed structure, and can constitute killer-applications. Indeed, a P2P distributed architecture realised through a mesh overlay can exploit the WCN link characteristics like the almost symmetric bandwidth and a high throughput. On the contrary, applications based on the client-server approach cannot scale without a large amount of centralized resources because they do not take advantage from the underlaying network structure.

One of the P2P applications that raises most interest in a WCN is live streaming, which can have a relevant impact on the community. 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 traditional media do.

Even if the P2P architecture is distributed, the application must manage network resources accurately to avoid starvation and to properly balance the load in the underlying physical network. Our early works [3] and [4] report the feasibility of P2P live streaming in WCNs, but also highlight possible limitations affecting these kind of networks, and describe possible tuning needed to maximize the distributed content quality, while limiting the network load.

This paper extends the work in [4], proposing and evaluating strategies for the management of the overlay topology and the content distribution in P2P live streaming. The key contributions of this paper can be summarized in the following bullets.

- The proposed strategies are aware and respectful of the network resources in a WCN, and autonomously adapt to them to achieve optimal performance with minimal resource use and maximal fairness.
- Several tests are performed on Community-Lab [2]. Community-Lab is a WCN testbed made up of real community network nodes, it allows the execution of experiments on different European WCNs testing the behaviour of real-world

[1]Notable examples in Europe only include: the Guifi.net community in Spain `http://guifi.net/`; AWMN Community,the Athens wireless metropolitan network `http://awmn.net/content.php`; the Ninux networks in Italy `http://ninux.org`; the Funkfeuer free networks in Austria `http://funkfeuer.at/index.php?id=42&L=1`.

applications. Measures and consequent conclusions taken on Community-Lab are particularly relevant as proof of concept of the feasibility in real WCNs, and give high confidence on the final reception quality of the content distribution.

- Additional tests have been run on Mininet [5]. Mininet is a network emulator we adapted to emulate actual WCNs. Our adaptation relays on the statistical data we collected from the Ninux WCN. This emulator allows the analysis of complex techniques like cross-layer optimization that are not feasible in Community-Lab.

The rest of the paper is structured as follows: Sect. 2 reviews the relevant literature and explains the goal of this paper; Sect. 3 describes P2P architecture and strategies and PeerStreamer, the platform we use for our tests; Sect. 4 is devoted to describe the strategies we propose to create an optimal overlay; Sect. 5 summarizes what are, at the application level, the strategies adopted to schedule video chunks transmissions; Sect. 6 presents the tools and infrastructures we use for testing our strategies; Sect. 7 reports the results we obtained from the tests and, finally, conclusions are given in Sect. 8.

2. State of the Art and Contribution of the Paper

Cross-layer optimization has been shown to be a key factor for real-time content distribution in wireless mesh networks [6] [7]. However, the approaches described in [6] [7] rely on advanced MAC/PHY protocols (such as IEEE 802.11ae) and the resulting algorithms have currently not been implemented. The multicast distribution on Wireless Mesh Networks is addressed in [8] and [9] but the suggested solutions rely on proposed advanced features for the data-link layer. [10] presents a collaborative algorithm for video streaming and cross-layer optimization, but it requires that all the WMN, including the user number and the load, is under full control. Video streaming in WMN is also addressed in [11] which reports a study on video streaming quality using different interference-aware metrics.

Only 4.5% of the papers analysed in [12] validates their findings with experiments, but reproducing the proposed techniques and the experimental scenario is almost impossible. In previous works [3][4] we tested PeerStreamer on WCNs and we highlighted the need of a network aware strategy to create an efficient overlay that can cope with packet losses in WCNs. The repeatability of the tests performed is granted by the open source nature of PeerStreamer and the real world testbed Community-Lab.

Building efficient overlays is a well studied topic, and efficient distributed algorithms to build unstructured peer overlays already exist [13] [14]. These algorithms focus on placing the peers with more network resources close to the source to enhance the content distribution performance. They behave well on the Internet, where network resources are not equally distributed. However, they do not take into account the network load and the fairness. Other works like [15] and [16], build efficient overlays based on network coordinates. Again, these methods are tailored for the Internet, hence they do not take into account network load and link fairness.

As already mentioned the application we use is based on PeerStreamer, an open source P2P live streaming platform briefly described in Sect. 3. The architecture of PeerStreamer is described in [17], while its capability to adapt to different network

scenarios is studied in [18]. Additional details and related literature can be found on the home page of PeerStreamer[2].

2.1. Goals of This Paper

Compared with the discussed literature, this paper takes a different global perspective, which is enabled by two facts. First of all we exploit the PeerStreamer platform that enables the customization, including cross-layer optimization, of P2P streaming and its adaptation to specific environments. Second, we exploit the knowledge of mesh networking and routing protocols to introduce simple, but smart solutions that enhance the streaming performance and use fairly the network resources.

The goals and contributions of this paper are thus unfurling in two directions. First of all we demonstrate the feasibility of a traditional "killer application" as live video streaming in a fully distributed fashion without any support from content distribution systems or large centralized facilities. Our effort lays the bases for the future development of live event streaming and video conference applications without the need of a cloud-based infrastructure. In doing this we also show that WCNs, far from being only "geek games", can support advanced and reliable services like video streaming. The availability of symmetric, medium to high bandwidth links[3] can compensate for the lack of a structured backbone, provided that the services, as we did for video streaming, are conceived and customized for this architecture. Second, we show that P2P live video streaming can be adapted to be respectful of network resources (reducing the number of times a video chunk is transmitted in the WCN) even in complex environments. This is a key concern for people that operate a WCN.

3. PeerStreamer: Assumptions and Notation

In this section we introduce and discuss the notation used, the key hypotheses for the theoretic analysis, and some general assumptions on P2P live streaming. We will also review the basic principles of PeerStreamer.

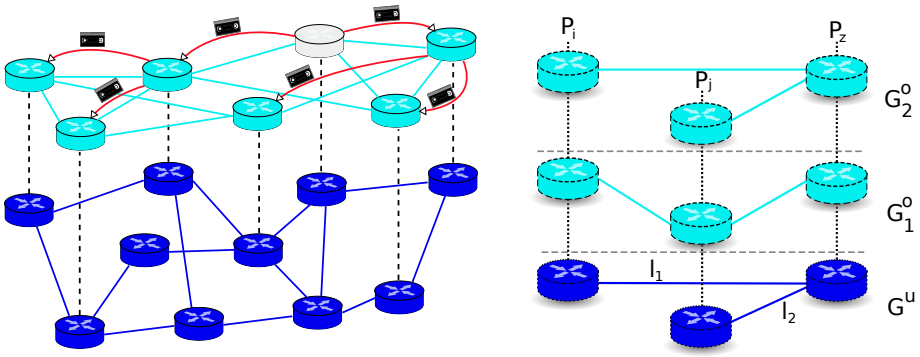
We call the WCN network the *underlay* to distinguish it from the *overlay* network built by PeerStreamer. The underlay is supposed to be connected and we assume each node knows the next hop and the distance (the hop count or the weighted hop count depending on the routing protocol) to any other node. This is true for any proactive routing protocol. We model the underlay as a graph

$$G^u = (R, L^u)$$

where R is the set of routers present in the WCN and L^u is the set of wireless links that connect them. The underlay is represented by blue (dark grey) nodes in Fig. 1a.

[2] `http://peerstreamer.org`

[3] Symmetry of links can be verified on the web sites of community networks, that often export the characteristics of their topology; moreover, the symmetry of the transmission equation with respect to the receiver and transmitter characteristics indicate that asymmetric links sometimes reported in 802.11 networks are due to pathological situations as the presence of a strong noise source close to one of the antennas.



(a) Wireless community network.

(b) Two possible overlays for a portion of an underlay.

Figure 1: WCNs with P2P overlay: wireless routers (in blue), P2P overlays (in cyan), video source (in grey). Red arrows represent the video distribution.

Some nodes of the underlay run an instance of PeerStreamer and are called *peers*. Each peer P_i at time t chooses a subset of the other peers as a set of neighbors that we call $N_i(t)$. Every P_i exchange video frames only with peers in $N_i(t)$, thus the union of all the $N_i(t)$ and the related links defines the network topology of the application, also represented as a graph and called the *overlay* (represented in cyan (light gray) in Fig. 1a). Let us introduce some definitions for the overlay graph: let $S(t)$ be the set of all peers at time $t \in [0, T]$. $S = \{\text{peer } P : \exists\, t \in [0, T] \text{ such that } P \in S(t)\}$ is the set of all peers and P_i the i-th peer of S. The subscript i is used only to distinguish peers from one another, the order of the peers is not relevant.

Definition 1. *The overlay built by PeerStreamer is a time-varying directed graph:*

$$G^o(t) = (S, L^o(t))$$

where S is the set of peers and

$$L^o(t) = \{(P_i, P_j) : P_j \in N_i(t)\}$$

is the set of edges that connect a peer to its neighbors.

In large networks made of thousands of users, peers constantly enter and exit the network and the overlay needs to be reorganized to follow these modifications. In this paper instead we consider a small amount of peers so that we can drop the dependency on t and assume $S(t) = S$. This does not mean our approach is not dynamic, if $S(t)$ changes, the choice of $N_i(t)$ changes and $G^o(t)$ varies; however, we're dropping the time variability to make the cross-layer design problem tractable, leaving the investigation on peers' churn for future works.

PeerStreamer can be installed directly on the wireless routers provided they have a Unix-like operating system, since its memory fingerprint and CPU occupation is

minimal [19][4]. PeerStreamer was developed with a function separation between the routing algorithm and streaming engine on one side, and content I/O on the other, the two connected via a standard network socket so that the video playback can be done in a different host than the one where PeerStreamer is running: users can run PeerStreamer on their wireless router and watch the videos on their laptop.

The cross-layer overlay design requires that the peers can map other peers on the routers where they are running. Indeed, this simply means that PeerStreamer has access to the system routing table, which is a reasonable assumption as the most popular routing protocols used in WCNs offer an interface that other programs can use to access the full dataset of information that the routing protocol owns. Since in PeerStreamer the peers are identified using their IP address and UDP port, it is easy to match a peer with the corresponding router.

The main difference between the overlay and the underlay is that the underlay is determined by the network topology, on which PeerStreamer does not have control, while the overlay is generated by PeerStreamer. The two layers are not independent: an edge in the overlay corresponds to a sequence of hops in the underlay so it is useful to express this relationship explicitly: sp_{ij} is the sequence $\{l_1^u ... l_n^u\}$ of links in L^u that form the shortest path of length n between the routers where peers P_i and P_j are installed.

Definition 2. *We call the* ***cross-layer graph incidence*** *of the path $P_i \rightarrow P_j$ the array A^{ij} of size $\|L^u\|$ made of elements:*

$$A_k^{ij} = \begin{cases} 1 & l_k^u \in sp_{ij} \\ 0 & l_k^u \notin sp_{ij} \end{cases}$$

A^{ij} is a convenient way of representing how the overlay influences the underlay. For instance the $l1$-norm $|A^{ij}|_1$ is the number of links in the underlay that are traversed by sp_{ij}. Moreover A^{ij} vectors that refer to distinct paths are summable as they have the same number of elements by construction. This will be useful to define impact metrics.

4. Building an Optimal Overlay

The overlay is built in a distributed way, each P_i choses $N_i(t)$ independently. Building the overlay is one of the major task in a P2P application [14]. PeerStreamer topology management is based on the newscast+ [20] protocol that uses low-throughput signalling messages that are exchanged by the peers to share their local knowledge of S. Without entering into details, it is sufficient to say that newscast+ periodically provides each peer with a partial list of all the peers forming the overlay. The choice of the gossipping algorithm is not critical with an overlay made of just a few tens of nodes: in just a few message exchanges each peer knows the entire S.

At regular intervals, P_i selects N_N peers from the latest received list of peers to populate its neighborhood $N_i(t)$. N_N is a key parameter to analyse the network dynamics and is investigated in Sect. 7. In PeerStreamer the overlay graph is forced to be an

[4]Technical information on the efficiency, the portability and in general the technical features of PeerStreamer can be found at the home page `http://peerstreamer.org`.

undirected N_N-regular to favour peer data exchange. This is realised through a simple protocol: if peer P_i adds P_j to its neighbourhood $N_i(t)$, it sends an advertisement message to P_j, which will add P_i in its neighbourhood $N_j(t)$. Conversely, if P_i removes P_j from $N_i(t)$, it notifies P_j that will removes P_i from $N_j(t)$. To guarantee a good connectivity of a random N_N-regular graph it is generally required that $N_N > \log_2(\|S\|)$.

Algorithm 1: Neighborhood refresh algorithm with newscast+ for peer P_i

Data: $N_i(t)$: set of peers in the neighbourhood at time t; S_t: newscast+ snapshot of peer at time t; N_N: target size of the neighborhood

```
N_i(t + 1) ← 70% of peers in N_i(t) randomly choosen
if ||N_i(t + 1)|| < N_N then
    move (N_N − |N_i(t + 1)|) random peers from S_t to N_i(t + 1)
end
```

Algorithm 1 describes a simple procedure to build a random N_N-regular overlay assuming the simple messaging depicted above is used. A random overlay construction is generally robust to peer churning, and it guarantees good connectivity properties, but is not network-aware: it is a reasonable compromise as long as information on the underlay is difficult to get, as in the Internet. In a WCN instead, it is possible to collect a good knowledge of the underlay, and exploit this information to build an overlay that reduces the use of the network resources. To clarify the relationship between underlay and overlay, consider Fig. 1b, which represents a portion made of three wireless routers of a larger underlay, each one with a running PeerStreamer instance: P_i, P_j and P_z. Two possible overlays are depicted, G_1^o (that does not mirror the underlay) and G_2^o. The distribution of chunks follows the overlay, in the case of G_1^o: $P_i \rightarrow P_j \rightarrow P_z$, so that chunks are sent once on l_1 and twice on l_2. In the case of G_2^o instead the chunks follow the path $P_i \rightarrow P_z \rightarrow P_j$ and both l_1 and l_2 are used only once for the transmission of the data. G_2^o thus uses the resources of the underlay more efficiently.

This example shows that different choices of G^o produce a different impact on the resources of G^u. To quantify this impact in general topologies we need to introduce two metrics: the network impact I^o and the fairness of the links occupation F^o.

Definition 3. *The network impact of an overlay G^o on an underlay G^u is defined using the cross-layer incidence, and it is the sum of all the occurrences of an underlay link being used to build the overlay:*

$$I^o = \sum_{i=1}^{\|S\|} \sum_{P_j \in N_i} |A^{ij}|_1$$

I^o counts how many times links in the underlay are used by peers in their interactions.

If each peer receives roughly the same amount of traffic from each of its neighbors (which is a realistic assumption), I^o will be proportional to the total number of packets exchanged in G^u for the entire video streaming. I^o can be used to compare the impact that different overlays built from the same set of peers S have on the underlay, it is an

a-posteriori measure of the efficiency of a given G^o. If we look back at Fig. 1b we can easily compute that the overlay G^o_1 has $I^o = 3$ while G^o_2 has $I^o = 2$.

The network impact does not consider how evenly the load is distributed on the links of the underlay, just the total amount of traffic. We use the Jain's fairness index [21] to evaluate and compare how evenly the underlay is exploited. Given a normalized vector $X = \{x_1 ... x_n\}$ the Jain fairness is:

$$J(X) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

The maximum value is 1, obtained when all the x_i are equal, and the minimum is $\frac{1}{n}$ when all the values except one are zero. We want to apply the fairness metric to measure how even is the distribution of the load on the links of the underlay given a certain overlay. Indeed, computing the fairness index on the full underlay can be misleading, since not all the underlay is used by the P2P application, just think at what happens when the underlay is the entire Internet! To make the fairness index comparable and meaningful regardless of the size of S and of the underlay, we introduce the notion of *useful underlay* $\widetilde{G}^u$.

Definition 4. *Given an underlay network G^u and a set of peers S the useful underlay $\widetilde{G}^u$ is the subgraph of G^u that includes all and only the links and nodes interested by a full mesh overlay among all the peers of S:*

$$\widetilde{G}^u = (\widetilde{R}, \widetilde{L}^u); \;\; \widetilde{L}^u = \bigcup_{P_i, P_j \in S} sp_{i,j}; \;\; \widetilde{R} = G^u \setminus \Omega$$

where Ω is the set of all nodes in G^u that are not connected by any link in $\widetilde{L}^u$.

We can now use $\widetilde{G}^u$ to compute the fairness indexes:

Definition 5. *Given a set of peers S, the fairness of an overlay G^o relative to the underlay $\widetilde{G}^u$ is:*

$$F^o = J(A_{G^o})$$

where

$$A_{G^o} = \sum_{i=1}^{\|S\|} \sum_{P_j \in N_i} A^{ij}$$

is the vector obtained summing all the cross-layer graph incidence vectors A^{ij} computed in $\widetilde{G}^u$.

Still F^o computed for different sets S or on different G^u are hardly comparable, as the distribution of the peers on routers and the topological constraints of the G^u can influence the achievable fairness. It would be useful to be able to use a fairness index normalized to the intrinsic properties of S and G^u. To this purpose we define the *intrinsic fairness* of the underlay F^u of S as the fairness index computed when the overlay is a full mesh.

Definition 6. *The intrinsic fairness of S given G^u is:*

$$F^u = J(A_{G^f})$$

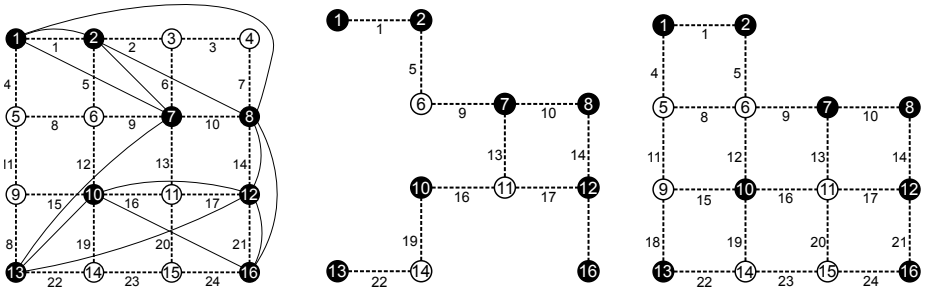
where A_{G^f} is the vector obtained summing the cross-layer graph incidence vectors A^{ij} relative to all the couples of peers (P_i, P_j) on $\widetilde{G}^u$.

Finally we can define a normalized or relative fairness φ that can be used to compare without biases the fairness of overlays obtained with different peer sets S on different networks G^u:

$$\varphi = \frac{F^o}{F^u}$$

Fig. 2 visually clarifies the notation we introduced so far. Fig. 2a represents a sample underlay graph, a set of peers and a potential choice for the overlay. Fig. 2b represents the $\widetilde{G}^u$ generated using the links that belong to sp_{ij} from every P_i to every $P_j \in N_i(t)$. Fig. 2c represents $\widetilde{G}^u$, the graph that is generated using the links that belong to sp_{ij} from every P_i to every P_j, used to compute the intrinsic fairness. Note that with such a regular topology there can be multiple choices of $\widetilde{G}^u$ and we choose one that minimises the number of nodes in $\widetilde{G}^u$ to increase the clarity of the example.

Table 1 reports the computation of all the A^{ij} for the overlay considered in Fig. 2b. Summing all the vectors and removing the zero elements we obtain A_{G^o} and we can finally compute $I^o = 27$ and $F^u = 0.88$. To compute φ one would use the graph in Fig. 2c, complete Table 1 with the A^{ij} between the couple of peers that are not direct neighbors in G^o, sum them all and remove the 0-values to compute A_{G^f}, then compute the intrinsic fairness F^u as in definition 6 and finally φ.



(a) Underlay G^u and overlay G^o (b) Portion of G^u used by G^o (c) The corresponding $\widetilde{G}^u$

Figure 2: An example of G^u (dashed lines and empty circles) and overlay G^o (solid lines and circles) with $N_N = 3$ (a); the portion of G^u interested by G^o (b); and the useful underlay $\widetilde{G}^u$ corresponding to G^o (c).

4.1. Network-aware Overlay Formation

One goal of this paper is to guide the choice of $N_i(t)$ so that the resulting overlay uses the underlay resources efficiently and fairly. The first component to achieve this is to define a suitable metric to change the peer selection to a ranking-based criterion. Algorithm 2 reports a suitable procedure to be used instead of the one in Algorithm 1.

A^{ij}	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	l_{11}	l_{12}	l_{13}	l_{14}	l_{15}	l_{16}	l_{17}	l_{18}	l_{19}	l_{20}	l_{21}	l_{22}	l_{23}	l_{24}
$A^{1,2}$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$A^{1,7}$	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$A^{1,8}$	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$A^{2,7}$	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$A^{2,8}$	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$A^{7,13}$	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0
$A^{8,12}$	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
$A^{8,16}$	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
$A^{10,13}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
$A^{10,12}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
$A^{10,16}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
A_{G^o}	3				4				4	2			1	2		3	2		2		2	2		

Table 1: The computation of A_{G^o} for $\widetilde{G}^u$ of Fig. 2b.

Algorithm 2: Optimized neighborhood refresh algorithm for peer P_i

Data: $N_i(t)$: set of peers in the neighbourhood at time t; S_t: newscast+ snapshot of peer at time t; N_N: target size of the neighborhood

1 $S \leftarrow N_i(t) \cup S_t$
2 $O =$ `OrderPeers`$(S, metric)$ /* ascend-order S according to "metric" */
3 $N_i(t+1) \leftarrow$ first N_N elements of O

We propose two different metrics for the `OrderPeers`$(S, metric)$ function in Algorithm 2. The first one is the weighted Hop Count Metric (H_c):

$$H_c(P_i, P_j) = |W \circ A^{ij}|_2$$

where W is the vector of the weights that the routing protocol assigns to each link, $\circ$ is the Hadamard product, i.e., the component-wise multiplication of all the elements in matrices with the same dimension, and $|\cdot|_2$ is the $l2$ or Euclidean norm of the vector. For simplicity we use a simple hop-count metric, so that $W = \vec{1}$, but results can be easily extended to more complex metrics. H_c is extremely simple to compute, and normally all the information needed for the computation of H_c are already present in the routing table of the router where P_i resides. The H_c metric helps avoiding situations like the one of overlay G_1^o in Fig. 1b in which P_i does not realize that P_z is on the path to reach P_j, so that any packet sent to P_j will reach also P_z. If P_i chooses the neighbors based on the proximity, P_i will first choose neighbors that do not have any other node on the path to reach them, which does not completely prevent, but surely lowers, the chances of creating inefficient overlays like G_1^0 of Fig. 1b.

We can conjecture that among all the possible choices for the G^o, H_c will generate one that most resembles the underlay. However, H_c will prefer peers that are particularly central in the underlay topology (i.e., that have a low average distance from all the other peers), which may introduce an unfair distribution of the choice and an unfair use of the underlay resources. In some cases it may be more efficient to choose a neighbor that is farther than another, but in an area of the underlay that is less used by PeerStreamer.

The second metric we propose is the Equalized Hop Count metric (E_{H_c}), which targets this issue while still trying to discourage communication between distant peers. E_{H_c} is computed by P_i as:

$$E_{H_c}(P_i, P_j) = |W \circ (A^{ij} + B)|_2$$

where B is an array defined as:

$$B = \sum_{i=1}^{\|S\|} \sum_{j=i+1}^{\|S\|} A^{ij}$$

Note that the second summation starts from $i + 1$ in order not to count each path twice, one for each direction, and that in the transitory phase in which a peer still does not know the whole S, it can approximate B with the information on the peers it is aware of. E_{H_c} needs more information to be computed than H_c as P_i needs to know all the shortest paths between any couple of known peers to compute B. This is possible, for instance, using the OLSR protocol in which the full network topology is known to each router, but it is time consuming; however, B can be cached and needs to be computed only when the network topology changes.

E_{H_c} introduces a key advantage over simple H_c: it adds a weight to the links that are more likely to be chosen given the underlay topology. If a peer is very central in the topology, the introduction of B will increase the weight of links around it and will penalise it compared to peers that are more peripheral. We will see that this has a direct effect on the fairness of the distribution of the load on the underlay.

The heuristics and metrics we have exposed here are rooted in the concepts of betweennes and centrality in a graph, that we extensively studied with the OLSR protocol in [22].

5. Video Distribution

In live streaming the source is a special peer P_s that provides the media content. In PeerStreamer, P_s concatenates one or more frames (audio and video separately) into chunks and injects them in the overlay. Chunks are numbered; C is the set of all chunks generated by P_s and C_h is the h-th chunk. For each chunk C_h there is a delivery deadline dl_h related to the playout time, after which C_h is no more useful and it's "lost" for the application. After its generation at time t, a chunk C_h is pushed in the overlay: P_s selects a peer $P_i \in N_s(t)$ and pushes C_h to it. C_h can be injected in m multiple copies to different peers in $N_s(t)$ in order to speed up the dissemination process. Chunks' reception is always acked (by all peers) for reliability and to avoid duplicate transmissions. Acks timeout is irrelevant for performance, but we use it to estimate the reliability of the logical link; it is set to 10 s. At regular time intervals τ_o, every P_i "offers" a window of size n_w of its most recent chunks to one $P_j \in N_i(t)$. P_j "selects" a set of size sc of them so that every C_h percolates in the overlay using an *offer/select* protocol. Actual transmissions of a chunk is initiated as soon as P_j selects it, or as soon as the ack of another chunk tells the application that the number of parallel transmissions is below the target level. This is fundamental to control the

overall average transmission rate, as UDP does not have rate or window control. No chunk is received twice, so the distribution of each chunk forms a tree on top of the overlay, as depicted in Fig. 1a.

The selection and building of the neighborhoods $N_i(t)$ is in any case mediated by the topology manager of PeerStreamer, which guarantees that $N_i(t)$ and $N_j(t)$ are never completely overlapped if $i \neq j$ so that the probability of "storm requests" to a single peer is negligible.

5.1. Peer and chunk selection

We refer to *peer and chunk selection* as the procedure used by P_i to choose $P_j \in N_i(t)$, offer it a set of chunks and let P_j select the *sc* of them he needs more. Many works have been published studying strategies to disseminate chunks in a P2P overlay on the Internet [23, 24, 25, 26, 27]. PeerStreamer has been designed to work on the Internet and we consider the "benchmark" scheduler one that selects uniformly at random in $N_i(t)$ the peer P_j to send the offer to, while P_j selects the most recent chunk that it does still not possess. This is called a "random – latest-useful" (R-LU) chunk selection and with some assumptions it can be considered optimal for live streaming [28]. R-LU performs quite well on the Internet; however, WCNs are not the Internet, and even if R-LU continues to be a benchmark, typical adaptation procedures based on the Round Trip Time (RTT) measures might not work, as RTT in a WCNs is not stable and it is not dominated by propagation, but can be highly variable [3]. Moreover in WCNs, the total number of peers can be expected to be relatively small (tens to hundreds of users, not thousands to millions), and this can also affect the adaptation strategies, making "wide Internet" approaches suboptimal. The experiments presented in this paper deal with the behavior of the distribution as a function of five different parameters:

- N_N: the target dimension of the outgoing neighborhood;
- m: the number of copies of each chunk injected by P_s into the overlay;
- *sc*: the number of chunks that a peer is allowed to select from each offer it receives (specified in the offer);
- *fa*: the number of audio frames that are assembled into a single audio chunk. In our experiments the video is H.264 encoded, the encoder generates video frames of variable dimension and audio frames of fixed size (207 bytes), which are injected separately by the source. A low *fa* produces a higher number of chunks, hence a higher impact on the network but a shorter delay in the data reception.
- P_s seeding strategy: Au or Aw. When the source P_s generates a new chunk it must decide to which peer $P_i \in N_s(t)$ it will be sent. The choice can be random with a uniform distribution (Au), or it can follow a weighted distribution that takes into account the local communication quality between P_s and P_i with the goal of making a more reliable chunk injection in the overlay (*Aw*). This is an adaptation we introduce only in the source, and it is better explained in the rest of this section.

The strategy used to inject the first chunk in the overlay is particularly important. As a clarifying example consider the case in which $m = 1$; if a chunk is lost at the first hop, it is lost for all the peers. As we expect that the number of peers in a WCN is small, the source can keep a statistic of their connectivity "quality" in order to avoid peers with bad connectivity, for instance leaf nodes in the underlay with a very weak link. Obviously P_s can not compensate the poor network performance of the underlay, so we designed a simple strategy to prevent P_s to choose such nodes. Note however that the strategy applies to the video source only, so peers with bad connectivity are not excluded from the overlay.

Let $w_i(t)$ be the weight associated by P_s to $P_i \in N_s(t)$ at time t. When a previously unknown peer P_i enters $N_s(t)$ its weight $w(i)$ is set to 1. P_s selects $P_i \in N_s(t)$ at time t with probability:

$$p_i^s(t) = \frac{w_i(t)}{\sum_{j:P_j \in N_s(t)} w_j(t)} \tag{1}$$

With Au strategy $w_i(t) = 1$ and P_s chooses $P_i \in N_s(t)$ with uniform probability.

With Aw the weight of P_i equals the rate of chunks that have been so far correctly acknowledged by P_i as a passive a posteriori measurement of link quality. For each $P_i \in N_s(t)$, P_s computes $w_i(t)$ as a moving average with an exponential decay of the successful chunk delivery rate. If P_i has been selected by P_s for chunk injection at time t_1, P_s first sends the chunk, and then updates the previous weight measured at time t_0:

$$w_i(t_1) = \begin{cases} \alpha + w_i(t_0)(1-\alpha) & \text{if ack received by } P_s \\ w_i(t_0)(1-\alpha) & \text{if a timeout expires} \end{cases} \tag{2}$$

Currently we have set $\alpha = 0.01$ and a timeout of 10s.

6. Experimental set-up

We performed experiments using two different platforms, the Community-Lab offered by the CONFINE project and a local emulation realised using the Mininet framework. In this section we briefly review both the platforms. The terms node and router are synonymous and identify a network node in the WCN, which is obviously also a router; they are used interchangeably in the following. The term *research device* (or RD) are instead the general "containers" of the Community-Lab, so the term is uses only in Community-Lab experiments.

6.1. Community-Lab

In order to have a reasonably controlled environment, and the possibility to run experiments through a standard and centralized interface, we run the experiments on the facilities provided by the CONFINE EU project: the Community-Lab.

Community-Lab [2] is a testbed that interconnects nodes from different real WCNs, intended to investigate problems and solutions for WCNs in realistic scenarios. Currently the CONFINE project involves different communities spread across Europe: Guifi.net, AWMN, FunkFeuer and Ninux.org, placed respectively in Spain, Greece,

Austria and Italy. All these networks can be considered "in production", meaning that thousands of users daily access their resources.

Community-Lab is made of special nodes called *research devices*, which are directly connected to WCN nodes, so they actually communicate through the WCN. Each research device can instantiate multiple linux containers called *slivers*, each of which belongs to a group called a *slice*. Researchers can create a slice and the related slivers by selecting the desired research devices and then run their experiments on all the slivers with a single command. More details on how Community-Lab can be used to run experiments with real P2P applications can be found in [4].

6.1.1. Experiments Performed on Community-Lab

We use Community-Lab to fulfill two goals. The first one is to verify that P2P live streaming is feasible in a real WCN. The second one is to find a set of configuration parameters for PeerStreamer that results in high quality streaming, in terms of number of frames delivered and playout delay. These parameters are used to perform topology-based optimizations in Mininet. We performed our tests in two islands of the Community-Lab: Guifi.net and AWMN. Since Community-Lab is still an ongoing project, the number of slivers in the other islands is still too small to be representative for our experiments, and also at the time of the experiments it was not yet possible to interconnect different Community-Lab islands, thus the experiments run on Community-Lab must be considered as a feasibility study and not as a performance evaluation one.

Every experiment is composed of several runs (normally between 10 and 30 depending on the stability of results) to ensure meaningful averages, each of which lasts ten minutes; we analyse the central five minutes of the runs in order to avoid data related to transient behaviors. The data of the experiments is averaged over all the successful runs. This technique ensure that overall each presented performance figure correspond to several hours (5 minutes times the number of runs) of streamed video, while different runs with different seeds and initial conditions accounts for network variability that would otherwise be very difficult to observe in a single long run. This is a standard methodology in the evaluation of streaming applications [29, 30].

The video we distribute is a re-encoding at 24 fps (video frames) and average bit rate of 300 kbit/s (including both audio and video) of Big Buck Bunny[5]. τ_o is set to guarantee that on average the offer rate is slightly larger than the frame rate, so the distribution is sustainable and the signalling overhead is minimal: a much smaller τ_o easily guarantees a better distribution but at the price of many refused offers, increasing the overhead.

The experiments explore the influence of the five parameters discussed in Sect. 5.1. N_N is varied from 5 (a value dangerously small) to 20 which, with the number of nodes we can deploy, means nearly a full mesh. Since $\|S\|$ is small we explore $m \in \{1, 3\}$ only; as m increases, the dissemination speeds up but it requires more and more resources at the source node P_s; as $m \to |S|$, the distribution process degenerates to a "multiple unicast" scenario as in cloud-based streaming. P_s can emit chunks either uniformly

[5] http://www.bigbuckbunny.org/

toward any peer (Au) or following Eq. 2 (Aw). Finally, the number of chunks that can be selected for each offer is increased, $sc \in \{1, 3, 5\}$, and the number of audio frames per chunk is also increased, $fa \in \{1, 5\}$.

6.2. Mininet

Mininet is a lightweight emulator of complex networks with a customizable topology. It is a python project realised under the BSD Open Source license that takes advantage of the contextualization features of the Linux kernel. While Community-Lab is a useful testbed to investigate real WCN scenarios, at its current state it provides little information on the underlay graph. We believe that cross-layer optimization can have a great role in the peer-to-peer and WCN context, thus we developed a framework using Mininet as an emulator of real-life WCNs capable of granting the access to information from the routing layer.

Mininet allows the specification of topologies, links' capacities, delays, and loss rates. We based our framework on real data from [31]. For this work we use the topology extracted from the Italian Ninux WCN with 126 nodes. For each link we can also access the ETX metric, that represents the average number of transmissions to successfully deliver a packet over a link [32]. We use a threshold of 5 retransmission (as suggested by the IEEE802.11 standard) so that we approximated the probability of loosing a packet on link l as $(1 - \frac{1}{ETX_l})^5$.

We currently do not have data regarding the bandwidth and the delay of each Ninux link, however, we use data from the analysis of a portion of the Guifi WCN [33]. For the available capacity we set a maximum of 10 Mbit/s for link. This value is an underestimation for modern WCNs, but already largely exceeds the need of the live streaming application we use.

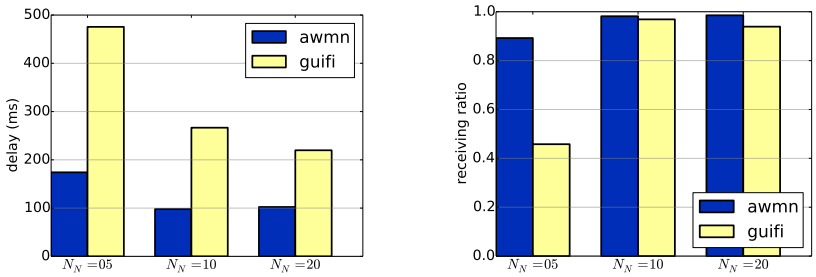
Mininet, through netem, allows the specification of a statistical model for the links' delays. Again, from the analysis of the data available from the authors of [33] we extracted the empirical distribution of delays and used it as input for Mininet. The distribution is long tailed with an average of 1.49 ms and a standard deviation of 2.82 ms, which is perfectly compatible with the use of 802.11n devices with sector antennas.

Finally we slightly modified Mininet in order to have a deeper control on IP address assignments. Our patches and the whole system is publicly available[6].

7. Experimental Results

We present results from several measurement campaigns, divided between the experiments on the Community-Lab, with the number of peers limited by the research device availability, and without the possibility of cross-layer optimization, and the experiments on Mininet, where these limitations are not present.

[6] netem can be found at:
`http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`
the modification and patches we implemented with the entire emulator are in the community network emulator project available at: `https://ans.disi.unitn.it/redmine/projects/`



(a) Average chunk delay.

(b) Average chunk receiving ratio.

Figure 3: Chunk level performance in the two WCNs as a function of N_N $m = 1$.

7.1. Testing PeerStreamer on the Community-Lab

The measurement campaign we present consists of about one month of experiments, with roughly 100 hours of actual video distribution, and several Gbytes of logged data analysed to select the most meaningful results. Experiments presented refer to the Guifi.net and AWMN Community-Lab islands. We also report an extract of the experiments performed for [4], necessary to frame the problem and introduce the cross-layer optimization, please refer to that paper for more details.

The number of available slivers is 10–12 in AWMN and 24–28 in Guifi.net, and we try to guarantee that for each experiment type the number of slivers is consistent across different runs, but slivers' availability is not always granted, so different experiments can have a slightly different number of slivers.

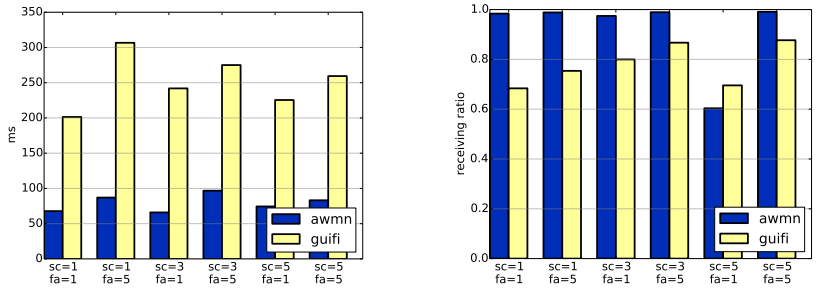
7.1.1. Role of the neighborhood size

Fig. 3 presents the chunk delivery rate and average chunk delay as a function of N_N, revealing that N_N plays a major role. Since the overlay over AWMN is composed of 11 peers only, the performance for $N_N = 10$ and $N_N = 20$ is almost identical, as the overlay is always a full mesh. Fig. 3 shows that in Guifi.net $N_N = 5$ is definitely too small and performance is unacceptable, even if $N_N > \log_2(\|S\|)$.

These results indicate that for these small overlays N_N has a much higher impact than what was measured in [18]. $N_N = 10$ produces delivery of almost 100% of chunks with a delay lower than 300 ms. If not otherwise stated the other experiments are run with this neighborhood size.

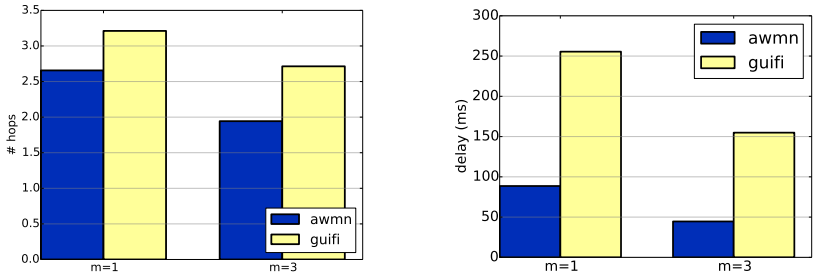
7.2. Chunks Transactions

These experiments were done on Community-Lab too. Fig. 4 shows the effects of chunk transaction dynamics, varying *sc* and *fa* in the offer/select protocol. A larger *fa* improves the receiving ratio as a consequence of the reduction of the total number of chunks per second, which requires a lower number of messages to be exchanged. Lowering the exchanged packets lowers the loss probability and improves the overall data dissemination. As expected a larger *fa* increases the delivery delay, but the effect is tolerable and remains within the limits of a live service. *fa* is the number of audio frames packed in a chunk, thus increasing *fa* increases the packetization delay of audio frames, and the playout delay.



(a) Average chunk delay. (b) Average chunk receiving ratio.

Figure 4: Chunk level performance in the two WCNs, varying *sc* and *fa* with $N_N = 10$, $m = 1$.



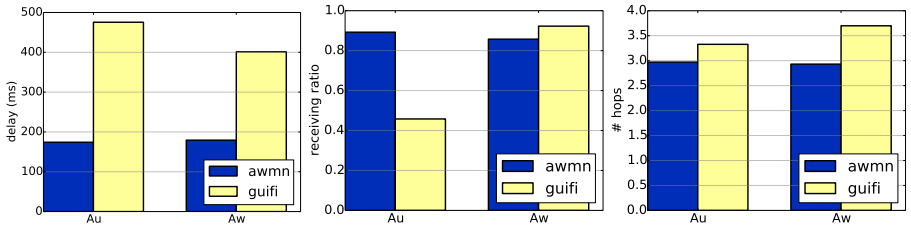
(a) Average number of hops per chunk. (b) Average chunk delay.

Figure 5: Chunk level performance in the two WCNs, varying m; N_N =10, sc =3, and fa =5.

The impact of *sc* on chunk losses is a bit more complex. Chunk delivery improves with *sc* for guifi.net, while in AWMN it is almost insensitive to *sc*. A high *fa* should increase the chances that a peer can retrieve all the chunks he needs in due time; however, the behavior as a function of the parameters is very complex and non-linear. For instance the combination $sc = 5, fa = 1$ lead to a performance degradation, possibly because the presence of many audio chunks together with the possibility of selecting many chunks per offer makes the dwell time of some chunks in the offer-select chunk-buffer too small, and they become unavailable for part of the overlay. A complete sensitivity analysis on all the parameters space to find an optimal set is extremely time consuming and out of the scope of this paper. The impact of *sc* on delay is instead negligible.

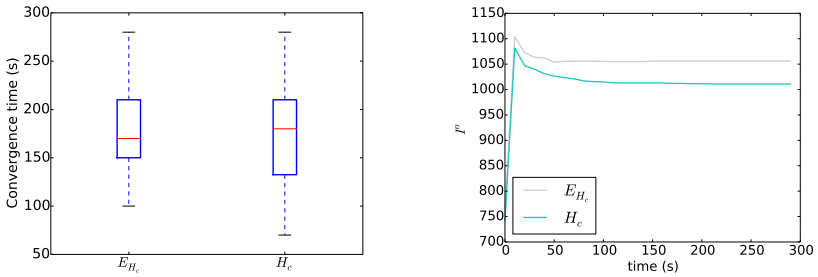
7.2.1. Chunks Injection Multiplicity and Push Strategy

If P_s increases the number of injected copies the distribution process is improved, at the cost of more networking resources for P_s. Fig. 5 reports the average chunk delay and the average number of hops of delivered chunks. The fraction of received chunks is practically one for all experiments. Increasing m from 1 to 3 the number of average hops needed to disseminate the chunks from the source to the peers decreases. The result is expected, and it is a consequence of the fact that three peers have the newly created chunk at the same time, so the distribution tree has a smaller diameter. As a



(a) Average delay. (b) Average delivery ratio. (c) Average number of hops.

Figure 6: Chunk level performance in the two WCNs, $N_N = 5$, $sc = 3$, and $fa = 5$.



(a) Boxplots of the convergence time. (b) Evolution of the network impact.

Figure 7: G^o convergence time using H_c and E_{H_c} metrics.

consequence also the delay decreases remarkably in relation to the smaller number of dissemination steps required.

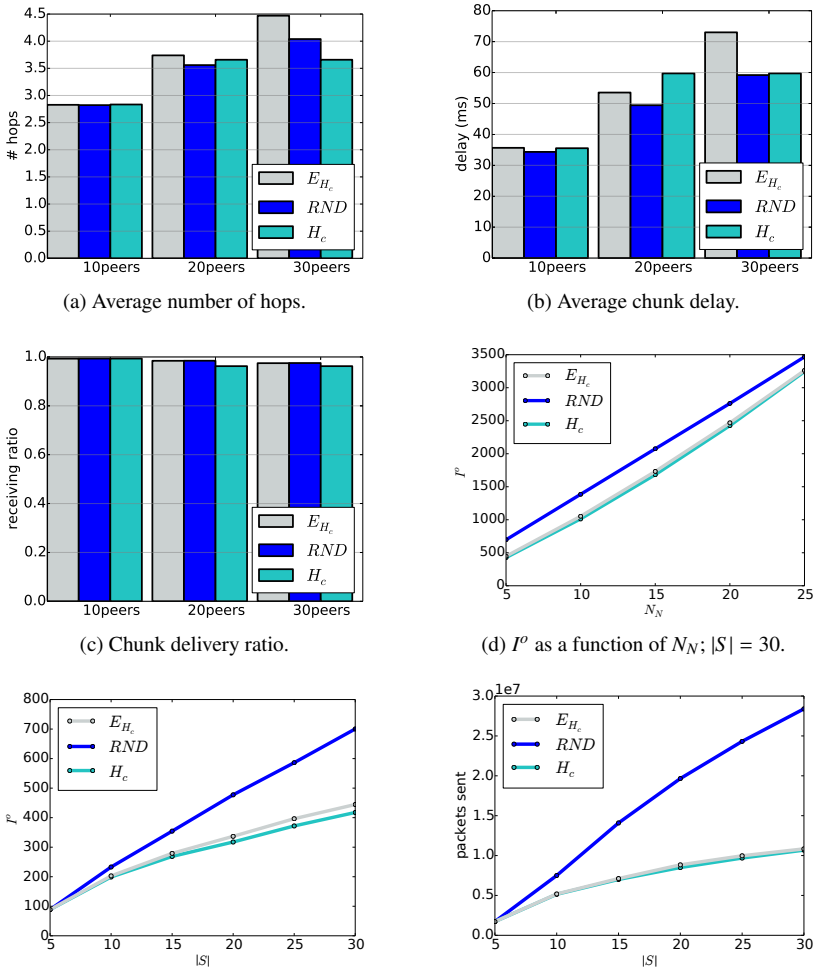
Fig. 6 shows that for Guifi.net both delay and loss improve remarkably with the Aw strategy. AWMN instead does not seem to be affected by the strategy. The reason for this difference is that the underlay of AWMN performs in average much better than Guifi so the Aw strategy does not offer relevant gains, as all the weights in Eq. (2) are roughly one. Please refer to [4] for more details on the underlay performance.

It is interesting to notice from Fig. 6 that the average number of hops increases with the Aw strategy. The reason is that chunks are injected less uniformly in the overlay, so that, on average, they have to be redistributed more times to reach all the peers, but this effect does not influence the delivery delay, as better transmission conditions imply that the chunks diffuse quickly without the need of retransmissions.

7.3. Topology cross-layer optimization on Mininet

Given the results obtained in the previous experiments, when not diversely specified the experiments in this section will use the following set of parameters: $N_N = 5$, $|S| = 30$, $fa = 5$, $sc = 5$, with default strategy Aw. The tests with Mininet focused on the metrics introduced in Sect. 4.1 using the real topology of the Ninux WCN. The first element analysed is the convergence time of G^o using the H_c and E_{H_c} metrics. The experiment involved thirty runs with $|S| = 30$ and $N_N = 5$ (we keep N_N intentionally small in order to highlight the improvement over the results shown in the Sect. 7.1.1).

A dynamic random strategy for the neighborhood selection imply that the overlay never stabilises; instead, H_c and E_{H_c} tend to create a more stable topology (albeit it can dynamically change when needed) with only minor fluctuations. Fig. 7 shows the average time for convergence to a stable topology using I^o as indicator of topology stabilization. Fig. 7a shows that in our emulations it takes in average between 2 and 3 minutes to reach a stable topology. Actually, Fig. 7b shows that after only 15-20 seconds the algorithm converges to an overlay very close to the stable one, which is a perfectly compatible with live streaming, considering that in real situations not all the peers are switched on at the same time.



(a) Average number of hops.

(b) Average chunk delay.

(c) Chunk delivery ratio.

(d) I^o as a function of N_N; $|S| = 30$.

(e) I^o as a function of $|S|$; $N_N = 5$.

(f) No. of frames transmitted in G^u.

Figure 8: Performance of the cross-layer optimization in the Ninux topology.

Let's now consider the application performance. Fig. 8a reports the average number

of hops in the overlay with a growing number of peers. It shows that H_c and E_{H_c} tend to increase the number of hops compared to the random strategy since they introduce a bias towards closer neighbors. Such bias decreases the use of resources in the underlay but can generate a G^o with a larger diameter compared with random choice. Fig. 8b shows that longer paths in the overlay do not actually produce a much higher delay (10ms are negligible in video live streaming), since we privilege shorter paths in the underlay (and thus with less probability of packet loss). This is confirmed observing Fig. 8c which shows that the chunk delivery rate is between 96% and 99.5%.

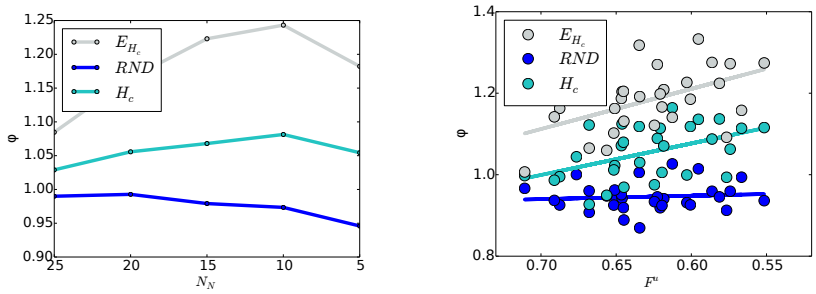
Fig. 8d shows I^o for G^o obtained with H_c and E_{H_c} compared with random choice increasing N_N with $|S| = 30$. With any strategy I^o increases with N_N, since the number of addends of the second summation of Def. 3 increases with N_N. Both H_c and E_{H_c} outperform the random strategy. This is even clearer in Fig. 8e that shows that the difference increases with the number of peers when $N_N = 5$. Fig. 8f shows that the number of frames transmitted with the random neighborhood choice increases nearly linearly with the number of peers, which is coherent with the expectation that the length of sp_{ij} remains roughly constant independently from the peers density. Using the cross-layer approximation, instead, the number of frames transmitted in G^u increases following a sub-linear curve, as the average length of sp_{ij} decreases thanks to the intelligent choice of G^o. We mentioned in Sect. 4 that we expect I^o to be proportional to the total number of frames transmitted in the G^u. We see from Fig. 8e and Fig. 8f that the curves are very similar with a difference probably due to packet retransmissions, that the I^o metric can not take into account.

The analysis of the relative fairness φ in Fig. 9 highlights the different behavior of H_c and E_{H_c}. As shown in Fig. 9a, with E_{H_c} φ computed with any neighborhood size is always significantly higher than the φ computed with H_c. The difference between the curves has a complex relationship with N_N. For large N_N the optimized algorithms have less effect (the overlay degenerates to a full mesh and no optimization is possible), for small N_N the difference also decreases since the overlay graph becomes sparse and there is less room for optimization. With intermediate values of N_N, E_{H_c} successfully exploits the underlay characteristics obtaining a greater fairness. At the cost of a minimal increase in the impact I^o, as shown in Fig. 8e, E_{H_c} can gain up to 20% in fairness compared to H_c.

Fig. 9b shows another interesting property of our metrics. In this case, 30 experiments have been performed varying S and keeping $N_N = 5$ with the three strategies. We computed φ and ordered the results according to the value of F^u. The graph shows that the more the underlay is unfair, the more the overlay is *relatively* fair. The lines are the best linear fit of the data points and show the clear different behavior of the random strategy compared to the cross-layer optimization. This means that our strategies, whose goal is to distribute the load evenly in the underlay, work particularly well when the underlay is skewed and not regular, or when the distribution of peers in G^u leads to a $\widetilde{G}^u$ with some nodes that are very central and thus potentially overloaded. This is a key advantage of our cross-layer optimization in challenging network scenarios.

7.4. Higher bit rate video encoding

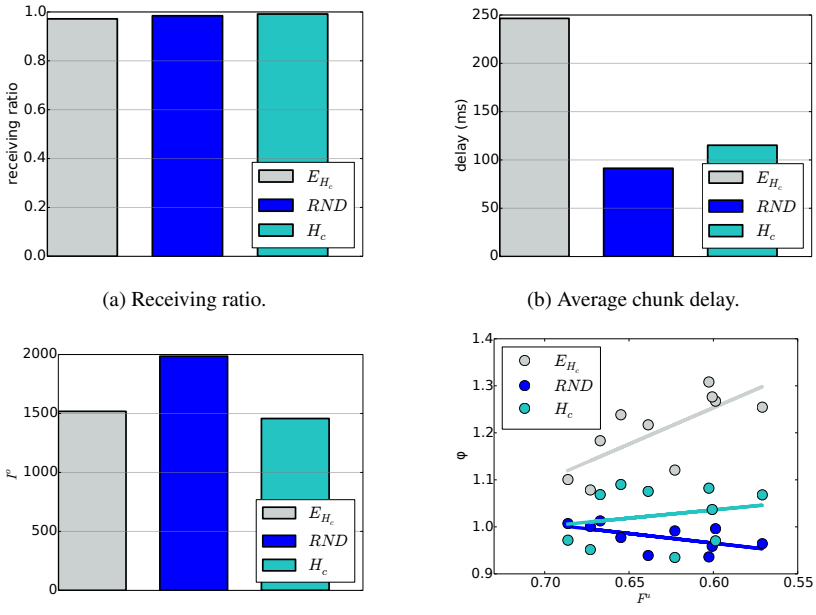
In the initial analysis we have experiments with 300 kbit/s streams, which is a low quality streaming in either CIF or QCIF format. The idea was to have a low-impact



(a) φ versus N_N; $|S| = 30$.

(b) φ versus F^u with $N_N = 5$.

Figure 9: Analysis of the relative fairness φ in the Ninux topology.



(a) Receiving ratio.

(b) Average chunk delay.

(c) Network impact.

(d) Neighbourhood fairness.

Figure 10: Loss and delay with 30 peers in the Ninux topology streaming a file at 1 Mbit/s.

application, meaning that adding one stream would be barely noticeable by the WCN, but most of all to avoid congesting the Community-Lab, which is more resource constrained. Fig. 10 shows instead the results with Mininet for a 1 Mbit/s stream, which can be considered, with today encoding techniques, standard TV quality, in the Ninux topology with 30 peers in the overlay. The reported results are the chunk receiving ratio and delay, together with the network impact I^o and the neighborhood fairness. These results show that a stream with standard quality is also sustainable and, in general, it does not affect the operations of the WCN, as links capacity in Ninux is normally well above 10 Mbit/s.

7.5. Lessons learned

The results we presented have shown that P2P live streaming is possible within WCNs. The code we developed is available as research code through the PeerStreamer web site, and custom applications can be implemented and optimized for any given WCN following the guidelines and discussions in this paper. Self-adaptation and optimization of the code, so that it can be "blindly" installed without networking skills is beyond the scope of this paper, and is definitely subject of further research.

8. Conclusions

Live video distribution is one of the applications that WCNs supports with difficulty, because the interconnection bandwidth toward the Internet to exploit cloud-based systems is often scarce, and because normally there are no resource-rich data centers inside the WCN, which is normally built bottom-up and with a flat architecture.

The use of P2P solutions matches well the characteristics of the WCNs, as they do not require a single node or data center with a lot of resources that distributes to everybody, but every node contributes to the distribution.

This paper presented the proof-of-concept that P2P live streaming is feasible in WCNs adapting the well known PeerStreamer platform to run on WCNs, at least with small to medium distribution overlays. To adapt PeerStreamer to WCNs we have devised new heuristics and strategies that exploit WCNs characteristics. Using the CONFINE testbed Community-Lab, the experiments have shown that PeerStreamer can work in a real WCN without overloading the WCN. This is guaranteed by Community-Lab itself, that isolates the experiments in such a way that if they run it means that the normal operation of the underlying WCN is not affected. As the delivery rate is close to one with very low delay we can conclude that the received video quality remains equal to the transmitted one. Using the Mininet emulation framework, it has been shown that PeerStreamer can build its own distribution overlay optimizing its configuration on the dynamic WCN characteristics. This feature is obtained with a cross-layer approach, letting PeerStreamer access the routing tables of nodes in the WCN.

The results are encouraging and show that the video distribution can be efficient and resource-aware, supporting the distribution of local events to the community on the resources owned and deployed by the same community.

Bibliography

[1] P. Frangoudis, G. Polyzos, V. Kemerlis, 'Wireless community networks: an alternative approach for nomadic broadband network access', IEEE Communications Magazine 49 (5) (2011) 206–213.

[2] A. Neumann, I. Vilata, X. León, P. E. Garcia, L. Navarro, E. López, 'Community-Lab: Architecture of a community networking testbed for the future Internet', in: IEEE 8th Int. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob 2012), Barcelona, SP, Oct 2012, pp. 620–627.

[3] L. Baldesi, L. Maccari, R. Lo Cigno, 'Live P2P Streaming in CommunityLab: Experience and Insights', in: 13th IEEE/IFIP Mediterranean Ad Hoc Networking Workshop (MedHocNet 2014), Piran, SLO, June 2014, pp. 23–30.

[4] L. Baldesi, L. Maccari, 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 2014), Larnaca, Cyprus, Oct 2014, pp. 33–39.

[5] B. Lantz, B. Heller, N. McKeown, 'A network in a laptop: rapid prototyping for software-defined networks', in: 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HOTNETS IX), Monterey, Canada, Oct. 2010, pp. 19:1–19:6.

[6] Y. Andreopoulos, N. Mastronarde, M. van der Schaar, 'Cross-layer optimized video streaming over wireless multihop mesh networks', IEEE Jou. on Selected Areas in Communications (JSAC) 24 (11) (2006) 2104–2115.

[7] H.-P. Shiang, M. Van Der Schaar, 'Multi-user video streaming over multi-hop wireless networks: a distributed, cross-layer approach based on priority queuing', IEEE Jou. on Selected Areas in Communications (JSAC) 25 (4) (2007) 770–785.

[8] R. Campos, C. Oliveira, J. Ruela, 'WiFIX+: a multicast solution for 802.11-based wireless mesh networks', in: 8th IEEE/IFIP Int. Conf. on Wireless On-Demand Network Systems and Services (WONS 2011), Bardonecchia, Italy, Jan. 2011, pp. 179–186.

[9] A. Russo, R. L. Cigno, I. Rubin, 'Protocol independent multicast: From wired to wireless networks', in: IEEE Int. Conf. on Computing, Networking and Communications (ICNC 2013), San Diego, CA, US, Jan. 2013, pp. 610–615.

[10] N. Mastronarde, D. S. Turaga, M. Van Der Schaar, 'Collaborative resource exchanges for peer-to-peer video streaming over wireless mesh networks', IEEE Jou. on Selected Areas in Communications (JSAC) 25 (1) (2007) 108–118.

[11] V. C. Borges, M. Curado, E. Monteiro, 'The impact of interference-aware routing metrics on video streaming in Wireless Mesh Networks', Elsevier Ad Hoc Networks 9 (4) (2011) 652–661.

[12] S. Mantzouratos, G. Gardikis, H. Koumaras, A. Kourtis, 'Survey of cross-layer proposals for video streaming over mobile ad hoc networks (MANETs)', in: IEEE Int. Conf. on Telecommunications and Multimedia (TEMU 2012), Heraklion, Grece, July 2012, pp. 101–106.

[13] R. J. Lobb, A. P. Couto da Silva, E. Leonardi, M. Mellia, M. Meo, 'Adaptive overlay topology for mesh-based P2P-TV systems', in: 18th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Williamsburg, VA, USA, June 2009, pp. 31–36.

[14] S. Traverso, L. Abeni, R. Birke, C. Kiraly, E. Leonardi, R. Lo Cigno, M. Mellia, 'Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems',

in: 12th IEEE Int. Conf. on Peer-to-Peer Computing (P2P'XII), Tarragona, SP, Sept 2012, pp. 13–24.

[15] P. Pietzuch, J. Ledlie, M. Mitzenmacher, M. Seltzer, 'Network-aware overlays with network coordinates', in: Workshops of the 26th IEEE Int. Conf. on Distributed Computing Systems (ICDCS), Lisbon, PT, July 2006, pp. 1–7.

[16] R. Cox, F. Dabek, F. Kaashoek, J. Li, R. Morris, 'Practical, distributed network coordinates', ACM SIGCOMM Computer Communication Review 34 (1) (2004) 113–118.

[17] R. Birke, E. Leonardi, M. Mellia, A. Bakay, T. Szemethy, C. Kiraly, R. Lo Cigno, F. Mathieu, L. Muscariello, S. Niccolini, J. Seedorf, G. Tropea, 'Architecture of a Network-aware P2P-TV Application: the NAPA-WINE Approach', IEEE Comm. Mag. 49 (6) (June 2011) 154–163.

[18] S. Traverso, L. Abeni, R. Birke, C. Kiraly, E. Leonardi, R. Lo Cigno, M. Mellia, 'Neighborhood Filtering Strategies for Overlay Construction in P2P-TV Systems: Design and Experimental Comparison', IEEE/ACM Trans. on Networking 99 (3) (June 2015) 741–754.

[19] L. Abeni, C. Kiraly, A. Russo, M. Biazzini, R. Lo Cigno, 'Design and Implementation of a Generic Library for P2P Streaming', in: ACM Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking, Florence, Italy, 2010, pp. 43–48.

[20] N. Tölgyesi, M. Jelasity, 'Adaptive Peer Sampling with Newscast', in: 15th Int. Euro-Par Conf. on Parallel Processing (Euro-Par), Delft, The Netherlands, 2009, pp. 523–534.

[21] R. Jain, D.-M. Chiu, W. R. Hawe, 'A quantitative measure of fairness and discrimination for resource allocation in shared computer system', Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, USA, 1984.

[22] L. Maccari, R. Lo Cigno, 'Betweenness estimation in OLSR-based multi-hop networks for distributed filtering', Elsevier Journal of Computer and System Sciences 80 (3) (May, 2014) 670–685.

[23] S. Traverso, C. Kiraly, E. Leonardi, M. Mellia, 'A performance comparison of hose rate controller approaches for P2P-TV applications', Elsevier Computer Networks 69 (2014) 101–120.

[24] A. Russo, R. Lo Cigno, 'Delay-Aware Push/Pull Protocols for Live Video Streaming in P2P Systems', in: IEEE Int. Conf. on Communications (ICC'10), Cape Town, SA, May 2010, pp. 1–5.

[25] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, A. Twigg, 'Epidemic live streaming: optimal performance trade-offs', in: ACM SIGMETRICS, Annapolis, MD, USA, June 2008, pp. 325–336.

[26] Y. Liu, 'On the minimum delay peer-to-peer video streaming: how realtime can it be?', in: 15th ACM Int. Conf. on Multimedia, Augsburg, DE, Sept 2007, pp. 127–136.

[27] R. Lo Cigno, A. Russo, D. Carra, 'On some fundamental properties of P2P push/pull protocols', in: 2-nd IEEE Int. Conf. on Communications and Electronics, Hoi An, Vietnam, June 2008, pp. 67–73.

[28] L. Abeni, C. Kiraly, R. Lo Cigno, 'On the Optimal Scheduling of Streaming Applications in Unstructured Meshes', in: IFIP Networking, Aachen, Germany, May 2009, pp. 117–130.

[29] J. Klaue, B. Rathke, A. Wolisz, 'EvalVid - A Framework for Video Transmission and Quality Evaluation', in: Proc. of the 13th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (Tools 2003), LNCS 2794, Springer, Urbana, IL,

USA, Sept. 2003, pp. 255–272.

[30] L. Abeni, C. Kiraly, R. Lo Cigno, 'Effects of P2P Streaming on Video Quality', in: IEEE Int. Conf. on Communications (ICC'10), Cape Town, SA, May 2010, pp. 1–5.

[31] L. Maccari, R. Lo Cigno, 'A week in the life of three large Wireless Community Networks', Elsevier, Ad Hoc Networks 24, Part B (2015) 175–190.

[32] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris, 'A High-throughput Path Metric for Multi-hop Wireless Routing', Springer Wireless Networks 11 (4) (July 2005) 419–434.

[33] L. Cerdà-Alabern, A. Neumann, P. Escrich, 'Experimental Evaluation of a Wireless Community Mesh Network', in: 16th ACM Int. Conf. on Modeling, Analysis & Simulation of Wireless and Mobile Systems (MSWiM), Barcelona, Spain, 2013, pp. 23–30.