Multicast TV Channels over Wireless Neutral Access Networks: Proof of Concept and Implementation

Lorenz Klopfenstein, Andrea Seraghiti, Alessandro Bogliolo
University of Urbino
Urbino, Italy 61029
Email: {*lorenz.klopfenstein, andrea.seraghiti, alessandro.bogliolo*}*@uniurb.it*
Stefano Bonino, Andrea Tarasconi
Essentia S.p.A
Reggio Emilia, Italy 42124
Email: {*stefano.bonino, andrea.tarasconi*}*@essentia.it*

Abstract—**IP traffic trends and forecasts suggest that television over IP (IPTV) will be the killer application for next-generation networks (NGNs). There is, however, a chicken and egg situation between the deployment of broadband access networks and the diffusion of high quality multimedia services, such as high definition television, which actually impairs the development of NGNs. As a matter of fact, most existing access infrastructures are still under-provisioned and they provide no suitable support to the widespread diffusion of global Internet TV channels, while wireless access networks are sprouting worldwide to provide nomadic connectivity and to bridge digital divide in sparsely-populated regions. In this scenario, IPTV services are delivered only within the walled gardens of biggest wireline operators and they are far away from reaching the critical mass required to attract investments in NGNs. This paper proposes technological and architectural solutions which enable the widespread diffusion of Internet TV channels over existing wireless access networks, thus overcoming the deadlock between services and infrastructures and paving the way to NGNs. In particular, the paper addresses both technological and market issues and presents the results of laboratory tests and proof-of-concept experiments conducted within the wireless campus of the University of Urbino. Finally, the paper outlines the implementation of the key components of the proposed architecture as addins of** *openBOXware***, an open source platform for the development of bandwidth-aware multimedia applications over IP.**

Keywords-**Internet TV; IPTV; Multicast; Radio broadcast; Proxy.**

I. INTRODUCTION

Internet protocol television (IPTV) is expected to be the killer application for next-generation Internet [1], [2] for two main reasons. First, because it encompasses high-bandwidth multimedia services which cannot be delivered over today's network infrastructures with sufficient *quality of experience* (QoE) [3], thus prompting for the development of *next generation networks* (NGNs). Second, because IPTV is expected to inherit the popularity of traditional broadcast TV, thus driving the market penetration of NGNs and providing the business opportunities needed to motivate the investments they require.

This vision is further supported by IP traffic statistics [4] and video traffic forecasts [5], which predict that consumer IP traffic will account for 87% of the overall aggregate traffic in 2014 and almost 60% of this share will be taken by Internet video streaming and download. In 2011 IPTV services delivered within operators' networks are expected to account for more than 40% of the overall IP traffic [6].

In spite of the soundness of these arguments, the positive feedback loop between infrastructures and applications is hard to be triggered since none of the two elements can leap forward by itself. Hence, the broadband market suffers from a stagnation which is caused both by the lack of investments (access networks are under-provisioned and there are market-failure areas still affected by infrastructural digital divide) and by the lack of demand (users are aware that existing infrastructures are unsuitable to deliver high-quality multimedia services, so that such services do not create a new demand for network connectivity) [7], [8].

IPTV was born in a scenario characterized by insufficient access infrastructures managed according to monolithic business models and flat-fee access rates. In this context, wireline operators may offer high-value services (including IPTV) within their own walled gardens in order to increase their revenues, but they are not motivated to promote global high-bandwidth applications (such as Internet TV) which can be accessed by their customers through flat-fee connections without generating additional revenues [7].

Both the IPTV services delivered within the walled garden of some operator (literally called IPTV) and the Internet TV networks available worldwide are targeted only to Internet users. This trivial observation has two (less trivial) consequences: first, IPTV cannot be sold by itself to people who don't want to subscribe for an Internet connection [9], [8], second, IPTV users are assumed to be accustomed to Internet browsing. Hence, neither the commercial models nor the usage patterns of IPTV [10] resemble those of traditional television: while broadcast TV is a mass medium per excellence, IPTV is an interactive entertainment service which

fully exploits the two-way unicast nature of IP networks to comply with Internet user's personal wishes.

Although the capability of supporting personal *on-demand services* (ODSs), such as video on demand and networked video recording, grants a competitive advantage to IPTV, ODSs raise scalability issues when the number of simultaneous users increases, while the significant difference in the usage experience slows down the convergence between broadcast television and IPTV.

A. The underlying network model

Starting from the observation that monolithic network models are inadequate to solve digital divide issues, to enable competition, and to promote innovation [7], regulations have been introduced in many countries to facilitate competition by forcing incumbent operators to allow new-entrants to use their infrastructures by means of local-loop unbundling, line sharing, bistream access, and convenient wholesale offers. Although such initiatives have contributed to increase broadband penetration [11], they haven't changed significantly the relationship between end-users and operators, since the new entrants have usually adopted the same market strategies of the incumbents. On the contrary, a drastic change in the value chain of broadband access could be induced by exploiting the separation between network connectivity and service provisioning which is inherent in the TCP/IP stack. The concepts of *open access network* (OAN) and *neutral access network* (NAN) have been recently introduced to this purpose.

OANs [12], [13] aim at enabling competition among service providers (SPs) on top of a shared infrastructure which acts as a transparent broker (Internet service providers are nothing but a special category of SPs). End-users connect to the shared infrastructure and register with the SP of choice, which has his/her own edge router directly connected to a common *operator-neutral* backbone [14].

NANs [15] are a special category of OANs conceived to trigger the positive externality which is a determinant of success and sustainability for a communication network: the larger the number of users in a network the greater the added value for each of them. According to the NAN model, a sizeable set of services have to be delivered within the access network and made available to the users before they register with any Internet SP. End-users establish commercial relationships with SPs, that in turn pay a share of their revenues to the NAN organization [12], [16] for the resources used by their customers.

NANs have the potential to reach higher penetration than traditional access networks because they are open to users who are not subscribers and who possibly pay only for the services they really benefit from. The larger the appeal, the popularity, and the usability of such services, the larger the diffusion and the penetration of the access network which

makes them available. Mass media are the ideal services to be used as icebreakers for the diffusion of NANs.

The advent of wireless access technologies (WiFi, Hiperlan, WiMAX) and the allocation of significant public funding to address digital divide have prompted for the deployment of sustainable wireless access networks aimed at maximizing public utility. OAN and NAN models are particularly suited to this purpose. Although the feasibility of HD video streaming over wireless access networks has been recently assessed [17], [18], current wireless technologies are not ready to support high performance multicast services.

Based on the above arguments, this paper explores the feasibility of wireless NANs delivering IPTV services which retain the key features of free-on-air broadcast TV channels. Both architectural and technological issues are addressed and a working prototype is built to make a proof of concept. Section II presents the proposed architecture, Section III outlines the implementation details and the technical choices of the experiment conducted within the wireless campus of the University of Urbino [19], Section IV presents preliminary experimental results that demonstrate the feasibility of the proposed approach, Section V discusses its implementation on top of *openBOXware*, an open-source platform for the development of distributed multimedia applications [20]. Conclusions are drawn in Section VI.

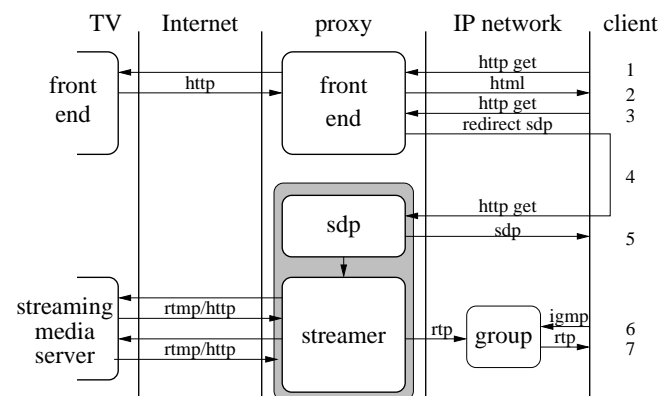## II. PROPOSED ARCHITECTURE

Figure 1. Schematic representation of the unicast-to-multicast proxy gateway.

The key components of the proposed architecture are the unicast-to-multicast TV proxy (hereafter called *proxy* for brevity) and the *base station* (BS), which grants wireless access to the *consumer premise equipment* (CPE).

A. Unicast-to-Multicast Conversion

The proxy acts as the TV head-end of an IPTV system, in that it receives, transforms and distributes TV contents to the subscribers. Its main functionalities can be described by referring to Figure 1, which provides a schematic representation of the overall software architecture used to make a

live Internet TV channel available to a multicast group: the Internet TV platform is represented to the left, the client is represented to the right, and the proxy is in between.

The Internet TV platform (left-most side of Figure 1) is composed of a front-end (i.e., a web server allowing end-users to browse live TV channels and gain access to on-demand services by means of their own web browsers) and a content delivery network, CDN (i.e, a network of streaming media servers distributed worldwide to reduce latency, increase quality of service, and perform Internet traffic optimization by providing some form of multicast support). At the application level, the stream is usually transferred across the Internet either in HTTP or in RTMP over TCP. When the end-user selects a content on the front-end of the Internet TV network, the client is transparently redirected to the most convenient streaming server of the CDN. A live TV channel can either be viewed as a continuous single stream which uses the same encoding, or, in some cases, it can be viewed as a playlist of heterogeneous contents (possibly encoded in different formats) which are concatenated at run time. In some Internet TV platforms the transition between subsequent contents requires the client to issue a request for the new content. Such a request is automatically triggered by the client software at the end of each stream in order to be transparent to the end-user.

Similarly to the Internet TV platform, the proxy is divided into a front-end and one or more streamers. The front-end is composed of a set of web pages that allow the end-user to browse and to choose a multicast TV channel among the ones made available by the proxy. When the client issues an HTTP-get request to the front-end (step 1 in Figure 1), the web page is dynamically composed, by taking channel data directly from the Internet TV platforms they belong to, and returned to the client (step 2). When the user selects a channel, the request (step 3) is redirected (step 4) to the corresponding *session description protocol* (SDP) file, which contains information about the streaming protocol adopted, the format and encoding of the data, and the address and port of the multicast group associated with the channel. For load-balancing reasons, the SDP files of different channels can be hosted by different servers. At the same time, a *streamer* is started for the requested channel, producing a stream that matches the parameters contained in the SDP file. The server returns the SDP file (step 5) to the client, which issues an *internet group management protocol* (IGMP) request (step 6) to the corresponding multicast group. Finally, the client starts receiving the RTP audio/video streams (step 7) generated by a *streamer* and sent to the multicast group. The streamer is a process which runs on the proxy to perform the unicast-to-multicast transformation. To this purpose, it acts as a client for the streaming server of the Internet TV channel of choice, it possibly performs data transcoding when needed, and it generates the audio/video RTP streams. In case of a TV channel composed of multiple files to be concatenated at

run time, this operation is transparently performed by the streamer, which produces a continuous stream. The streamer associated with a given channel is launched upon reception of the first request of the corresponding SDP file. Once the streamer is active, no further actions are required to serve subsequent requests to the same channel.

B. Wireless Multicast

In principle, multicast transmission is the most efficient solution for one-to-many communication over an IP network for four main reasons: i) the sender doesn't need to know the receivers, ii) the server sends each packet only once, iii) network nodes take care of replicating the packets only when it is striclty required to reach different users, iv) data packets are not delivered to receivers who don't want to receive them. Since each data packet is sent at most once across each link regardless of the number of receivers that will be reached through that link, bandwidth requirements become independent of the number of users. Due to the massive replication possibly required at the nodes, however, multicasting moves the bottleneck from bandwidth to computational requirements, which scale almost linearly with the number of users.

In principle, data replication issues could be locally overcome in a wireless network by exploiting the inherent broadcasting capabilities of a radio signal propagating over the air. By combining IP multicasting with radio broadcasting, both bandwidth and computational costs would become independent from the number of receivers.

Unfortunately, the radio multicast solutions available today within the IEEE 802.11 framework suffer from heavy limitations which come from the unicast-oriented standardization and implementation choices made over the years. In particular, most of the improvements introduced for enhancing the efficiency of unicast radio packets (such as *fast-frame*, *compression*, *Wireless Multimedia Extension*, and radio QoS) do not apply to broadcast/multicast radio packets, the only exceptions being the *raw rate* speed setting and the *no-ack* option introduced in IEEE 802.11e. Similarly, the strategies adopted by chipset vendors have sacrificed multicast/broadcast radio performance for the sake of unicast performance and cost reduction. For instance, top range chipsets provide four hardware queues for unicast traffic, while only one queue is available for broadcast/multicast traffic (one additional queue is usually dedicated to service packets). As a consequence, typical wireless point-to-multipoint base stations can provide limited performance and quality of service to multicast traffic, which competes with higher priority service radio packets.

The wireless equipment adopted in our setup have been specifically optimized at the *hardware abstraction level* (HAL) to overcome the above-mentioned limitations.

III. IMPLEMENTATION

This section outlines the implementation of a prototype of the architecture presented in Section 2.

A. Multicast proxy implementation

The proxy presents the list of available TV channels in a standard HTML webpage, served by ASP.NET (Mono) on an Apache server. The list is built by merging the lists of channels offered by remote Internet TV providers (retrieved by means of web service calls to their APIs) together with the list of possible local channels based upon media stored inside the proxy itself. The list is displayed and browsed by the client by means of any common web browser.

The client may request a specific TV channel by accessing a web URI using the browser. When such a request reaches the proxy, it checks whether the requested channel is already being streamed. This is done through a database that keeps tracks of the streaming processes, the channels they are handling, their parameters, and the streaming server they are running on. If no streamer is currently active, a new process is launched on a streaming server, which starts retrieving the desired channel and forwarding it to a multicast group picked by the proxy. Finally, the proxy server generates the SDP announcement containing the information required to retrieve the audio/video data streams and to decode them correctly.

Such an SDP announcement is served back to the requesting client using the "application/sdp" MIME type, so that the client, if properly configured, will automatically start an application capable of decoding and playing back the streams as declared by the announcement (*VLC Media Player* is used to this purpose in our setup). An SDP announcement is a standard text file containing the full description of a "media streaming session". At its most basic configuration, it can simply contain the IP address the stream is directed to and its components, which can be any number of video and audio streams, and their respective port numbers. Additionally, the announcement can also contain advanced details about each stream's encoding, which will be used by the client-side decoder to decide on how to interpret the incoming data.

Each streamer running on a streaming server is an instance of a mixed-mode .NET assembly running on *Mono* and relying on a native streaming back-end implemented in C++ using the *GStreamer* multimedia framework. The streamer determines the URI of the original resource associated with the TV channel requested by the client. The URI may refer to a local file, to a HTTP resource (as used by *Youtube*), to an Adobe RTMP stream (used by *StreamIt.it*), to a Microsoft Media Server (MMS) stream (used by *Rai.tv*), or to any other supported stream. As soon as the original location of the resource is ascertained, the native streamer is launched. The streamer relies either on built-in functionalities of *GStreamer* or on custom elements integrated with it in order to retrieve the stream and decode it. For instance, streaming through the Adobe RTMP protocol is not currently built into any mainstream *GStreamer* distribution, so that the specific support was provided by an in-house plug in.

The original audio and video streams are demuxed and then processed independently: each stream is decoded and converted as needed and then re-encoded using the encoding of choice for its final delivery. In our implementation, video data is always decoded and re-encoded using MPEG4-AVC/H.264 compression at a constant bitrate. Audio data is mixed down to stereo (in case of multichannel audio), resampled at 44.1 kHz, and finally encoded using the *Advanced Audio Coding* (AAC) scheme. In our setup the *x264* encoder has been used for video data and the *faac* encoder for audio data, each used through their respective *GStreamer* elements.

Note that the video compression step during transcoding also inserts SPS/PPS headers (i.e., *sequence/picture parameter set*) in the raw H.264 data stream, in order to ease the synchronization of client-side players[1]. These additional parameter headers inside the data stream contain information about compression, resolution and other properties that can be used by the decoder to synchronize to the stream and to properly decode the contents. This is needed mainly because video formats can vary abruptly between TV channels and even between single files belonging to the same channel, possibly causing the video player to lose sync and display corrupted data. Sometimes an abrupt change in resolution, if not handled during transcoding, can even crash some less resilient decoders on the receiving end. In case of TV channels with varying compression schemes and resolutions, one or more scaling and resizing steps can be added to the media pipeline in order to provide a more uniform output stream at the cost of a loss in picture quality, in addition to SPS/PPS headers (this feature was not used in our proof of concept). On the other hand, transcoding at runtime can be entirely avoided when channels are based on streams with pixel resolutions and encoding known beforehand. In this case, encoding information can be directly added to the SDP announcement (this may be the case especially for local files streamed from the proxy itself, which can be adequately prepared before streaming).

It is also worth noting that compression levels can be adapted to the available wireless bandwidth, trading off picture and sound quality, and even video resolution, for the number of simultaneous TV channels. On the other hand, wireless interference can be compensated by using less aggressive compression modes. This can be done, for instance, by targeting a lower H.264 encoding profile level, which issues more I-frames and avoids complex prediction techniques, thus easing the player's work to re-sync upon losing a packet. Those parameters can be either statically

---

[1]This option is known as "byte-stream" in the x264 encoding options.

configured in the proxy (as in our implementation) or dynamically adapted at runtime according to channel quality, number of channels or wireless traffic statistics.

B. Wireless equipment

The solution adopted for our experiments is based on the Essentia WFL R108F25x(B) radio module equipped with an Atheros AR5414 radio chipset. Hardware and protocol limitations have been overcome by means of multicast-oriented optimizations performed at the HAL, in the operating system (i.e., Essentia OpenWifless ESS OS), and in the device drivers. In particular, multiple multicast queues have been implemented and the IEEE 802.11e protocol has been extended in order to provide wireless multimedia extension (WMM) and QoS support to broadcast/multicast traffic.

As a result, the performance achieved by the current implementation of the radio module on multicast traffic over a point-to-point link equals that of unicast traffic. In addition, the radio protocol provides to multicast traffic full IEEE 802.1Q-2005 QoS support.

IV. EXPERIMENTAL RESULTS
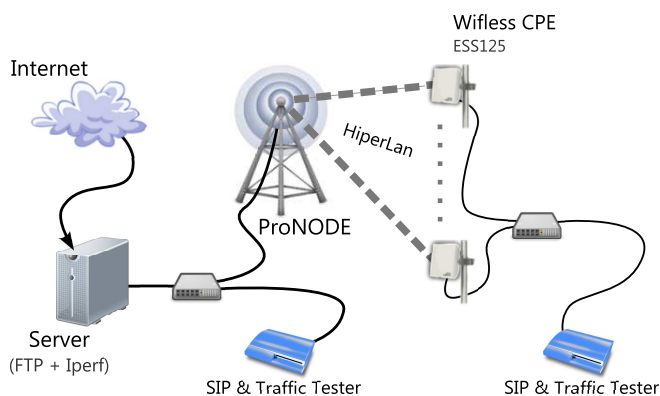
A. Lab experiments

Figure 2. Schematic representation of the experimental setup described in Section IV-A.

Extensive experiments were performed to assess the multicast performance of the wireless equipment and its scalability with the number of clients in a point-to-multipoint setup typical of a metropolitan access network. To this purpose, both the BS (namely, an Essentia Wifless ESS 24562 base station) and the CPEs (namely, Essentia Wifless 125 CPEs) adopted the enhanced radio module described in the previous subsection. An Agilent N2X traffic tester was used for traffic generation and performance measurements. In particular, the tester was used to simulate both a server connected to the BS, and 50 clients connected to 50 CPEs (VLAN tagging was used to connect the 50 CPEs to the tester across the same ethernet link). Two types of traffic were simulated concurrently: a multicast downlink uncompressable stream from

the server to all the clients and a unicast full-duplex UDP traffic between the server and each client. The experimental setup is depicted in Figure 2.

The aim of the experiment was to test the maximum achievable multicast bandwidth in presence of a background unicast traffic, and the scalability of the solution in terms of bandwidth and computational resources. Hence, a constant flow of 100kbps uplink and 100kbps downlink was generated for each client, summing up to 5Mbps uplink and 5Mbps downlink of unicast backgroud traffic across the BS. Then the multicast downlink traffic was increased while monitoring link quality and BS resource usage with 1 and 50 CPEs.

CPEs	Multicast traffic [Mbps]	Delay [ms]	Jitter [ms]	PLR %	CPU usage %
1	16.0	8	0.1	0	30
50	16.0	10	0.1	0	33
50	16.2	60	0.2	0	33
50	16.4	520	0.8	0	33
50	16.6	900	1.0	1	33

Table I
PERFORMANCE ACHIEVED BY A 24562 BS CONNECTED TO 50 CPES FOR DIFFERENT MULTICAST TRAFFIC LOADS ADDED TO A BACKGROUND OF FULL-DUPLEX UNICAST TRAFFIC. DATA REFER TO A RADIO CHANNEL OF 20MHZ AT 5.5GHZ, OPERATED AT A 24MBPS MULTICAST RAW SPEED.

The results reported in Table I show that the multicast performance of the BS does not depend on the number of CPEs connected. The marginal increase of CPU usage from 30% to 33% when connecting 50 CPEs is due to the additional 49 full-duplex unicast streams to be handled in our setup. Hence, as expected, multicast transmission does not suffer from scalability issues, the only physical limitation being the AR51414 HS encryption engine, which can handle up to 120 concurrent CPEs with no performance loss, and up to 512 CPEs per sector with performance degradation.

As for the limiting multicast bandwidth, no performance degradation occurs until 16Mbps of uncompressable traffic, while the degradation observed over this value is due to the limited computational resources of the CPEs. The maximum multicast traffic achieved by equipping CPEs with the same processor used in the BS was of 37Mbps per sector for a multicast raw speed of 54Mbps.

Additional tests were performed to determine the effect of modulation *raw speed* on the trade-off between multicast throughput and RF sensitivity of the Essentia WFL R108F25X(B) radio module. The experimental results are reported in Table II. Since the same modulation has to be imposed on all the CPEs associated with the same multicast group (or otherwise data replication would be necessary), the modulation raw speed can be used to span the trade off between the number of TV channels that can be delivered per sector (i.e., bandwidth or throupghput, column 4 in Table II) and the radius of the coverage area of the BS (i.e., sensitivity, column 2 in Table II). In particular, the coverage

Modulation Raw Speed	Sensitivity [dbm]	Mode HD/FD	Throughput [Mbps]	Jitter [ms]	Delay [ms]
54	-71	HD	37.0	0.5	3
		FD	19.5	0.5	4
48	-76	HD	34.0	0.1	5
		FD	18.2	0.1	5
36	-82	HD	27.2	0.1	4
		FD	14.5	0.1	6
24	-85	HD	19	0.1	5
		FD	10.0	0.1	7
18	-88	HD	15.0	0.1	6
		FD	7.5	0.1	4
12	-90	HD	10.5	0.1	7
		FD	5.1	0.1	8
9	-90	HD	7.4	0.1	7
		FD	3.8	0.1	6
6	-92	HD	5.2	0.1	9
		FD	2.5	0.1	5

Table II
EFFECT OF MODULATION ON THROUGHPUT AND SENSITIVITY OF THE
ESSENTIA WFL R108F25X(B) RADIO MODULE.

radius of the radio link doubles every 6dB of system gain (or sensitivity) if the same antennas are used. Hence, a 10X extension of the coverage radius (which corresponds to a 100X extension of the coverage area) could be achieved by changing the modulation from 54Mbps to 6Mbps, at the cost of reducing the number of TV channels per sector.

As a final remark, it is worth noting that multicast/broadcast packets are usually not retransmissible because of the lack of acknowledge in the protocol. This is an advantage in terms of protocol overhead (which ultimately has a benefit in terms of throughput) and a disadvantage in terms of QoS (which ultimately impacts the limiting distance of the radio link). In order to overcome this possible limitation, the CPEs adopted in our setup implement the *no-ack* option which allows one or more of them to inform the BS of the loss of a multicast packet, asking for retransmission. The overhead of the *no-ack* option ranges from 5% to 20% in case of CPEs operated with poor SNR.

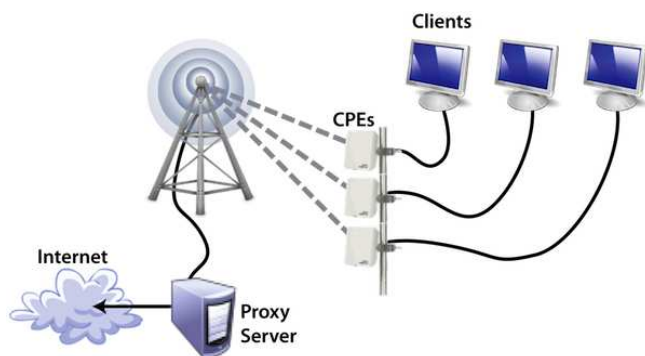### B. Proof-of-concept experiments

Figure 3. Schematic representation of the experimental setup described in Section IV-B.

A proof-on-concept experiment was conducted in Urbino (Italy) in May 2010 during a public event organized in the conference hall of the Ducal Palace. The setup is schematically represented in Figure 3.

Both the front-end and the streamer of the unicast-to-multicast proxy were executed on a 2.4GHz Intel Core 2 Quad with 4GB of RAM running Ubuntu 10.04. The proxy was installed within the intranet of the University of Urbino and connected to an Essentia Wifless ESS 24562 BS (belonging to the same LAN) installed on the roof of a building in front of the event site. Backhauling was granted by the Internet gateway of the University of Urbino, providing up to 150Mbps of Internet bandwidth. 10 PCs running *Opera Web Browser* and *VLC Media Player* on top of *Windows XP* were used as clients. Each of them was connected to an Essentia Wifless 125 CPE installed right outside the conference hall. Both *RAI Radiotelevisione Italiana* (http://www.rai.tv/) and *Streamit.it* (http://www.streamit.it/) were used as Internet TV providers. In addition, high-definition 1080p contents were streamed directly by the proxy. The performance of the BS (in terms of multicast and unicast throughput, packet loss, CPU, and memory usage) was monitored every 15s.

The experiment was used to demonstrate the feasibility of real-time transcoding and multicast transmission of HD Internet TV channels over a wireless link. In particular, it was clearly shown that bandwidth and computational resources increase linearly with the number of users watching unicast TV channels, while they are independent of the number of users watching a multicast TV channel.

A representative test was performed by comparing network behavior and device load when switching from watching the same TV channels through direct unicast connections to watching it as a shared multicast stream routed through the proxy server. As expected, the number of unicast connections has a linear impact on network traffic and device load, while the number of clients associated with the same multicast group deon't have any measurable effect neither on bandwidth occupation nor on CPU/memory usage on the BS. The same results were obtained both by streaming local resources from the proxy server and by transcoding live TV channels through the streamer.

The final stress test of the setup was executed in the following configuration: 2 clients watching the same multicast HD channel compressed at a constant rate of 5Mbps and streamed directly from the proxy, 2 clients watching the same *standard-definition* (SD) TV channel provided by *RAI Radiotelevisione Italiana* and transcoded at run time by the multicast proxy, 1 client watching a unicast SD channel provided by *Streamit.it*, 1 client watching a unicast HD channel provided by *Streamit.it*, 1 client watching the multicast version of a *Streamit.it* SD channel, 1 client watching the multicast version of a *Streamit.it* HD channel, and 2 clients watching a multicast HD content local to the proxy and compressed at a constant rate of 2Mbps. Figure 4

shows the 10 clients and the traffic/CPU/memory monitors as they appeared during the experiments.

Figure 4. Picture taken during the proof of concept experiment described in Section IV-B.

In summary, there were 2 unicast streams taken from the Internet, 3 Internet TV channels taken from the Internet and made available in multicast by the proxy, and 2 HD multicast streams directly generated by the proxy. Moreover, one of the clients watching a multicast TV channel was also connected to *YouTube* in order to generate unicast traffic across the same wireless link. The overall multicast traffic was of 12Mbps with a peak of unicast downlink traffic of 7Mbps. No packet loss was observed and the CPU of the BS was used only at 32%.

This final configuration also demonstrates that runtime transcoding of multiple standard and high definition video streams can be realistically done with consumer-grade hardware, without any quality degradation nor any noticeable delay for the clients.

V. OPEN SOURCE FRAMEWORK

The results of the proof-of-concept experiment prompted for the development of *openBOXware*, a portable, open-source framework providing a suitable abstraction for building bandwidth-aware client-server multimedia applications [20].

The final purpose of the framework is to offer a common development platform that can be adopted at various points of the scenario described in this paper, including content providers, proxy servers and receiving client end-points. Multimedia contents can be added and managed with ease at any stage by exploiting the abstraction and the flexibility offered by the plug-in system of *openBOXware*.

A. OpenBOXware

The software stack of *openBOXware* is similar to the one adopted in the original implementation of the unicast-to-multicast proxy described in Section III-A. Multimedia handling and streaming is based on the *GStreamer* framework,

while the basic runtime support is offered by *Mono*, the open-source implementation of *.NET*, which provides the language runtime, the type system, and the virtual runtime environment where the framework runs independently of the underlying hardware (*Mono* runtime is currently able to be compiled and run on many different platforms, including the most common *x86* and *ARM* architectures). The hardware abstraction features of *Mono* also provide the support for a dynamic plug-in system whose components can be easily distributed and run across different devices without need for recompilation.

In addition to what was originally implemented for the proof of concept, the framework also provides: a graphical user interface based on *Qt* widget toolkit, which enables the implementation of client-side applications, a complete *HTTP web server*, which allows the creation of web pages or web services, and a *Universal Plug and Play* (UPnP) module, which enables seamless interactions with other UPnP devices possibly available on the same local network.

All components and services that the framework implements and supports are exposed through an abstract API. Multimedia functionalities are provided by *pipelines*, which can be instantiated both by the framework and by its add-ins. Each pipeline is responsible for streaming an incoming data source to a target data sink. Transcoding is automatically handled by the framework as long as the source and sink components are properly described. Users of the framework never have to directly interact with the underlying *GStreamer* implementation, dealing only with abstract high-level descriptions of incoming and outgoing streams.

The API and the add-in system enable third-party developers and end-users to create their own plug-ins (or *extension points*) which can exploit all the multimedia handling capabilities of *openBOXware*. The system provides for three kinds of extension points:

- *Application*, the most powerful kind of extension point. Like the others it can be individually started and terminated, but it is the only one capable of interacting directly with the end-user through a custom graphical user interface. The extension point obtains a graphical context in which it can render itself, providing a custom-tailored user experience while still being able to integrate and exploit any other *openBOXware* functionality.

- *MediaSource*, used to expose a collection of media elements to the rest of the framework. Examples of media elements include Internet TV channels, web radios, online multimedia contents, or directories of resources local to the device or made available by other UPnP devices. These elements are then shared with the framework and with other applications in such a way that they can be explored as a hierarchical tree of multimedia resources. Each media element contains an abstract description of the resource it rep-

resents, specifying how the stream must be retrieved and how it must be decoded by the framework. *Open-BOXware* provides built-in support to the most common resources, including HTTP, TCP and UDP streams. Common multimedia streaming protocols like Adobe RTMP, Microsoft MMS, and RTSP are supported as well. The actual contents of the multimedia stream can be detected automatically by the framework or can be defined explicitly by the MediaSource developer. Once the resource is fully described through the API, the framework can automatically handle all needed transcoding and the user has only to deal with the abstract media element object.

- *MediaTarget*, used to expose a local or remote end-point, capable of receiving a multimedia stream (encoded in a certain format with specific properties) and playing it back. A single media target is composed of a video target, an audio target and a muxing target, which together define how the final stream must be encoded by the framework to match the specification. The muxing target determines how the audio and video sub-streams are combined together in a data stream that can be parsed by the receiving end (common supported container formats include Matroska, MP4 and FLV). Video and audio streams are encoded separately using the specification provided by the respective sub-targets: encoding format, bitrate, and any other property of the final data stream.

Any application can bind a MediaSource to a MediaTarget in a pipeline and start streaming data from the source to the target while the framework automatically takes care of the transcoding steps possibly required to adapt the stream to the target of choice.

B. Multicast TV on top of openBOXware

OpenBOXware is a general handler of multimedia flows streamed from heterogeneous sources to both local and remote targets. The capability of handling multiple simultaneous pipelines makes it suitable for *openBOXware* to be installed not only at the receiving end point, but also at the initial and intermediate nodes of a content delivery network. In particular, *openBOXware* provides a common framework for implementing all the software elements of the proof-of-concept outlined in Section IV-B: both the proxy server and the client-side application receiving the multicast stream could be implemented as *openBOXware* Applications, while the heterogeneous Internet TV sources could be implemented as independent MediaSources to be streamed to specific multicast groups, implemented as a MediaTargets.

A possible implementation of the proof of concept architecture on top of *openBOXware* is outlined in the rest of this section.

All the TV channels and media contents available are exposed as independent MediaSources to the instance of the framework running on the proxy server. The hierarchy of media elements is made available to the proxy front-end Application which makes use of the built-in web server in order to publish dynamic pages which allow end-users to browse the registered media sources by means of a simple web browser. The framework relies on an extensible list of content providers to make any new content available as soon as the corresponding add-in is installed on the proxy server.

Similarly to what happened in the original proof-of-concept implementation, when a play request reaches the web server (i.e., the front-end), the proxy application creates a pipeline for the selected media element and starts streaming it to a multicast group address. Clients may then receive the audio and video streams by interpreting the SDP descriptor file that is returned via HTTP by the proxy.

Although a standard web browser and a common media player can be used to select and receive contents from the proxy, specific client-side Applications can be developed on top of *openBOXware* to offer to the end-user custom interfaces and advanced features to browse and watch TV channels streamed through the proxy server. In our implementation, the server exposes an additional interface, other than the one dedicated to web browsers, in the form of a *REST* web service that responds with *JSON* encoded data. The JSON file is parsed and interpreted by the client application, which provides the user interface and the access control mechanisms. When a channel (or a media element) is selected, an HTTP request is issued that determine the web service running on the proxy to launch the stream and return the SDP file describing the streaming session the client can tune into. On the client side, the streams are received through a *GStreamer* pipeline and decoded locally.

VI. CONCLUSIONS

This paper has presented real-world experiments demonstrating the feasibility of multicast transmission of HD TV channels over the wireless point-to-multipoint connections often adopted in metro networks to bridge digital divide and to provide nomadic connectivity. The proposed solution makes use of two key components: a proxy, which acts as a client for the unicast TV channels available on the Internet and makes them locally available at specific multicast groups, and enhanced wireless equipment providing adequate support to multicast multimedia communication within the IEEE 802.11 framework.

The results of the proof-of-concept experiments outlined in this paper prompt for the development of new application-level solutions and market strategies aimed at enabling massive diffusion of Internet TV channels [21].

According to the neutral-access-network model [15], the diffusion of a popular and well understood application (such as television) over IP networks could play a fundamental

role to enhance broadband penetration and to motivate investments in next-generation networks.

A suitable support for the development of advanced IPTV delivery systems over managed IP networks is provided by *openBOXware*, an extensible open-source multimedia framework specifically conceived to bridge the gap between the proof-of-concept presented in this paper and real-world multicast IPTV applications [20].

REFERENCES

[1] A. Seraghiti, L. Klopfenstein, S. Bonino, A. Tarasconi, and A. Bogliolo, "Multicast TV Channels over Wireless Neutral Access Networks," in *Proceedings of Conference on Evolving Internet (INTERNET-10)*, 2010, pp. 153–158.

[2] Y. Xiao, X. Du, J. Zhang, F. Hu, and S. Guizani, "Internet Protocol Television (IPTV): The Killer Application for the Next-Generation Internet," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 126–134, 2007.

[3] R. Jain, "Quality of Experience," *IEEE Multimedia*, vol. 11, no. 1, pp. 95–96, 2004.

[4] Akamai, "The State of the Internet: 3rd Quarter, 2010 Report," *Akamai White Paper*, vol. 3, no. 3, 2011.

[5] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2009-2014," *Cisco White Paper*, 2010.

[6] ——, "Global IP Traffic Forecast and Methodology," *Cisco White Paper*, 2008.

[7] Broadband Working Group, "The Broadband Incentive Problem," *MIT Communications Futures Program White Paper*, September 2005.

[8] J. E. Prieger and W.-M. Hu, "The broadband digital divide and the nexus of race, competition, and quality," *Information Economics and Policy*, vol. 20, no. 2, pp. 150 – 167, 2008.

[9] K. Flamm and A. Chaudhuri, "An analysis of the determinants of broadband access," *Telecommun. Policy*, vol. 31, no. 6-7, pp. 312–326, 2007.

[10] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching Television Over an IP Network," in *Proceedings of IMC-08*, 2008, pp. 71–83.

[11] S. Turner, "Broadband Reality Check II: The Truth Behind America's Digital Decline," *Free Press*, 2006.

[12] R. Battiti, R. Lo Cigno, M. Sabel, F. Orava, and B. Pehrson, "Wireless LANs: From WarChalking to Open Acess Networks," *Mobile Networks and Applications*, vol. 10, pp. 175–287, 2005.

[13] J. Barcelo, J. Infante, and M. Oliver, "wireless Open Access Networks: State-of-the-Art and Technological Opportunities," in *Proceedings of INTERNET-09*, 2009.

[14] V. Kordas, E. Frankenberg, S. Grozev, B. L. N. Zhou, and B. Pehrson, "Who Should Own, Operate and Maintain an Opeartor Neutral Access Network?" in *LANMAN2002: IEEE Workshop on Local and Metropolitan Area Networks*, 2002.

[15] A. Bogliolo, "Introducing Neutral Access Networks," in *Proceedings of Int.l Conference on Next Generation Internet Networks (NGI 2009)*, 2009.

[16] J. C. Francis, N. Elnegaard, T. G. Eskedal, and R. Venturin, "Business Opportunities of Open Broadband Wireless Access Networks," in *Proceedings of Int.l Workshop on Broadband Wireless Access for Ubiquitous Networking*, 2006.

[17] M. Garcia, J. Lloret, M. Edo, and R. Lacuesta, "IPTV Distribution Network Access System Using WiMAX and WLAN Technologies," in *Proceedings of UPGRADE-CN*, 2009, pp. 35–44.

[18] L. Cai, Y. Luo, S. Xiang, and J. Pan, "Scalable modulation for scalable wireless videocast," in *Proceedings of IEEE Infocom-10*, 2010.

[19] A. Bogliolo, "Urbino Wireless Campus: A Wide-Area University Wireless Network to Bridge Digital Divide," in *Proceedings of AccessNets-07*, 2007, pp. 1–6.

[20] A. Bogliolo *et al.*, "The openBOXware project," http://www.openboxware.net/trac/, 2010.

[21] A. Bogliolo, "Video logs of Wireless Multicast TV experiments," http://www.youtube.com/view_play_list?p=824B2CA538D70C6E, 2010.