TEST ENVIRONMENT FOR PERFORMANCE EVALUATION OF AN INTERNET RADIO

David Melendi, Manuel Vilas, Xabiel G. Pañeda, Roberto García, Víctor G. García
Computer Science Department, University of Oviedo
Campus Universitario de Viesques. Sede Departamental Oeste, 33204 Xixón-Gijón, Asturies

Keywords: Audio, Evaluation, Internet, Live, Multimedia, Radio, Streaming.

Abstract: This paper presents a test environment designed to improve Internet radio services through the evaluation of different service features. The environment comprises the generation of audio streams, the delivery of those streams through different communication networks, and the access of final users to those contents. A broad set of service architectures can be emulated, and several network configurations can be deployed using the available communication devices. It is also possible to simulate users' behaviour thanks to a workload generator that can be configured using statistical information obtained from real service access data. A case study is also presented, where a glimpse of the possibilities of the environment can be caught. This test environment will allow service administrators or research teams to predict what will happen in a real service if its configuration is modified, or if user behaviour changes. Furthermore, managers will be capable of knowing whether an Internet radio service can be improved or not.

1 INTRODUCTION

The emergence of the World Wide Web has changed the Internet world. This service has become a powerful medium. Daily, millions of users in the world browse the web, produce an important number of accesses, and a huge volume of information is delivered to them. Like other communication enterprises, traditional Radio companies have discovered the Internet to be an important tool to reach millions of users in the world without any constraint related to commercial licenses, limited frequency spectrums, high infrastructure costs, etc. Now technology is not an issue for these companies. The bandwidth increase in subscribers' access capabilities, and the development of the streaming technology have given rise to the appearance of a new complementary service: the Internet Radio.

There are two types of audio services on the Internet: live-audio and jukebox (audio-on-demand). In jukebox services, the user requests the information at any time and the server delivers it exclusively. This system allows users to interact with information: Pauses, backward and forward jumps are allowed. Its behaviour is similar to a CD player, and it is the user who controls what to listen to in every moment: these are *user driven* services (Veloso, 2002). On the other hand, live-audio is more similar to a traditional radio service. Contents are received directly by a multimedia server, which broadcasts them straight out to the audience. In these services users cannot choose what to listen to, as there is a previously established programme. These are *object driven* services (Veloso, 2002).

Most radio services on the Internet are based on streaming technology. The advantages of audio streaming and the subscribers' expectations are important. However, this technology presents some problems. Although audio delivering is not so resource-consuming as video, it also requires a constant quality of service. What is more, live-audio services require greater transmission capabilities than jukebox services, due to the fact that they tend to suffer stress periods depending on programme evolution, known as *prime-time* periods. In these stress periods a wide range of users connect at the same time to the same source of contents. To maintain service quality under control, it is important to select the best configuration parameters possible. These parameters include, among others, the quality of the produced contents, the architecture of the service, the configuration of the network, etc.

Nowadays, all these parameters are usually established by service managers based on their own experience. These managers usually know what the most appropriate configuration is, what qualities must be generated, or where to place each service device. Nevertheless, each service is usually

Melendi D., Vilas M., Pañeda X., Garcia R. and Garcia V. (2005).
TEST ENVIRONMENT FOR PERFORMANCE EVALUATION OF AN INTERNET RADIO.
In *Proceedings of the Second International Conference on e-Business and Telecommunication Networks*, pages 183-190

different to the rest: there are problems which can be found in one service that would never affect others. Furthermore, experience is knowledge based on the past, but technology is constantly evolving and producing solutions to solve old problems or to improve previous solutions. Now the problem is to answer the following question: how can managers test these technology evolutions knowing what will happen in a real service? It is very risky to make changes in a working environment, as it can lead to unexpected situations.

A solution for this issue may be to have a reduced copy of the real service under a controlled environment. This copy could be used to carry out small tests. But some questions go beyond the possibilities of such a copy: How does the service behave under extreme conditions? How can real users affect service performance? How do network devices affect this type of services? What impact do routing protocols have in heavy loaded radio services? If a radical change is carried out in the service, how will it behave? All these questions can only be answered if a complete and flexible test environment is available to service managers.

In this paper, a test environment for performance evaluation of Internet radio services is presented. The main aim of the presented environment is to help service managers in planning, deploying, configuring and improving live-audio services. Furthermore, the paper has followed an interesting practical approach, based on the evaluation and improvement of a real Internet Radio service.

The problem of delivering live streams to end-users with a certain degree of quality has long been studied. There are abundant papers focused on improving service architectures for live streaming, such as (Jonas, 1998), (Dutta, 1999), (Chawathe, 2000), (Deshpande, 2001), (Padmanabhan, 2002), (Dutta, 2001) or (Melendi, 2004). Other papers, such as (Holbrook, 1999) or (Banerjee, 2002), try to improve these services taking advantage of technologies such as multicast. Others, such as (Bolot, 1993), (Jian, 2000), or (Mena, 2000) study the effects of network conditions in streaming services and other types of applications. Users' behaviour in streaming services has also been studied in papers like (Chesire, 2001) or (Veloso, 2002). A flexible test environment could help service managers to test the conclusions offered by all these papers for a given service.

The rest of the paper is organized as follows: Section 2 provides a detailed description of an Internet radio service. The presented test environment is set out in section 3. Section 4 offers a practical case study in the proposed environment, where the impact of a network configuration option on radio services is analyzed. Finally, conclusions are presented in section 5.

2 SERVICE DESCRIPTION

There are mainly two possibilities while developing an Internet radio service: to capture and distribute live audio streams, or simulate live contents using stored information.

Usually, when a live audio stream is captured to be distributed over the Internet, the service provider has an already existing transmission or the necessary resources to generate one. This type of Internet radio service is the most suitable for traditional radio companies. These companies have an available audio stream and require a secondary distribution channel to reach a wider audience. As shown in Figure 1, in this type of services two main software components are used: a production tool and a multimedia server. The production tool, also known as *producer*, must capture live contents, probably using the sound card of the machine where it is being executed. While it is capturing those contents, the *producer* applies the proper data conversions using a specific codec and a pre-established set of quality parameters. Once the conversions have been made, the *producer* delivers the obtained audio packets to the multimedia server. This server is in charge of processing customers' requests and delivering them the audio stream generated by the *producer*.

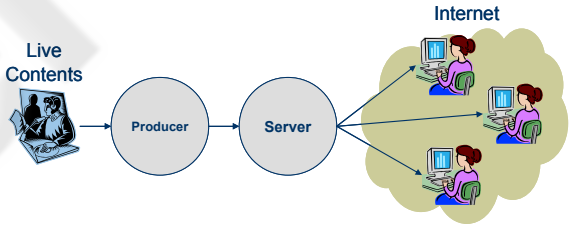

Figure 1: Radio Service with Live-Contents

On the other hand, if stored information is used, the service provider only needs a set of stored files coded with the necessary quality. These files are later sequenced to form a continuous audio stream. This type of services is based on the use of two software modules: a *simulated live transfer agent*, or *slta*, and a multimedia server. The *slta* generates a continuous audio stream by sequencing several stored multimedia files. This stream is then sent to the multimedia server that will deliver the generated contents to final users. In order to know which files must be sequenced, the *slta* uses a text file as input data. This file, also called *playlist*, has the path and the names of the files that must be used, alongside the meta-data that will be provided with the audio

stream to final users. Figure 2 shows the main components in an Internet radio service with stored contents.

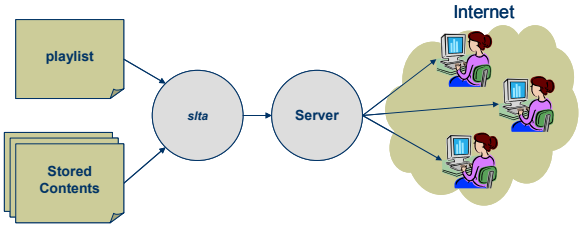

Figure 2: Radio Service with Stored-Contents

It must be taken into account that these service architectures are deployed on a communication network. While installing an Internet radio on a network, several alternatives are possible. All the needed software components can run on a single machine, or can be installed in dedicated computers. If dedicated computers are used, these computers can be in the same network, or in different networks interconnected by several devices. Furthermore, depending on the number of connected customers, advanced architectures can be used. These architectures are mainly based in the use of redundant systems, and intermediate devices such as multimedia *proxies*.

3 ENVIRONMENT DESIGN

The test environment has been designed to emulate the behaviour of a broad set of service architectures. It allows service administrators or research teams to deploy a simple radio service, test that service, modify its configuration or architecture, and draw conclusions about the impact of those changes. Furthermore, several network configurations can be tested using the available communication devices. Figure 3 shows the equipment installed in this environment.

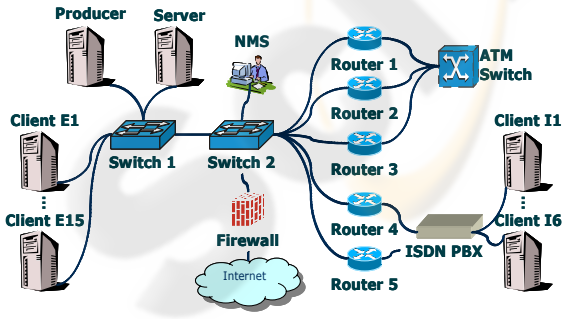

Figure 3: Environment Physical Architecture

Providing a real service, using this test environment will allow service administrators to predict what will happen if a certain change is applied, and whether the service can be improved or not. In order to meet these expectations, the test environment provides mechanisms to emulate the main steps on every Internet radio: content generation, content delivery and content access.

3.1 Content generation

In a real service, two possible sources can be used to deploy an Internet radio: an existing radio transmission, or a set of stored files sequenced to obtain a continuous audio stream. Given the fact that the source of audio may be irrelevant –it is not affected by computing or networking issues-, the test environment uses stored files to generate the contents that will be broadcast.

Although it is possible to use a common production tool, in its default configuration, this test environment uses a *simulated live transfer agent* or *slta*. This agent generates a continuous audio stream by sequencing several multimedia files. The audio stream is then sent to a multimedia server that will deliver the generated contents to final users.

Service administrators will code the multimedia files using the quality parameters they want to test. Once the final files have been generated, the administrators will only need to edit the *playlist* and write the names of the files on it. If new quality parameters are required, the contents must be recoded and the old files need to be replaced.

The environment, shown in Figure 3, has a machine running Linux called *Producer*, where the *slta* is being executed and the multimedia files must be stored. This *slta* produces an audio stream and sends it to the multimedia server executed in the machine named *Server*.

Moreover, the environment allows service administrators to test advanced configurations in this stage: redundancy and file sharing. Redundant data sources can be used installing more *Producer* machines with their correspondent *slta* application. These machines could use their own audio files –stored locally-, or could use a common file-system through *NFS*. So it is possible to test just software redundancy –using file sharing– or a complete redundancy duplicating everything: files and programmes.

3.2 Content delivery

Once we have generated the necessary contents, it is time to deliver those contents to final users.

The *slta* generates an audio stream that is sent to a multimedia server. In the test environment shown in Figure 3 a streaming server has been installed on a machine running Linux called *Server*. The task of

this server is to attend users' requests and to redistribute the audio stream under their commands.

Furthermore, the environment permits the testing of advanced configurations in this stage such as redundancy and intermediate devices. Redundant servers can be installed in order to attend users' requests. This is possible by adding new *Server* machines to the environment. In the simplest case, servers will inform users about other servers that manage the same audio stream. If a problem is detected in the main server, users will try to plug into the redundant server.

A more complex scenario uses several intermediate devices or *proxies*. Combining proxies with redundant servers, users will first go to the proxy and it will redirect their requests to the proper server. If poor performance periods are detected in one of the servers, the proxy can redirect users to a more efficient server. This proxy configuration is called *pass-through* mode: proxies are only redirecting devices. Another way of using proxies is a *splitting* mode, where proxies act as servers for final users. The original servers send the audio streams to proxies, and the proxies redistribute those contents among users. Again, this service configuration can be tested: new machines can be added to act as redundant servers or as proxies.

Apart from servers and proxies, another important element in any distributed service is the communication network. Here we can find two different scenarios for delivering audio streams to users: an Intranet Scenario and an Internet Scenario.

In an Intranet Scenario, the audience connects to the service using the same network where all the devices are installed. The size of this network varies from an office LAN to a network operator's WAN. Nevertheless, the owner of the network can usually control three critical points: available resources –for users and service devices-, service architecture, and network configuration.

On the other hand, in an Internet Scenario, the audience connects to the service using different networks and technologies. The owner of the network where service devices are installed does not control everything that happens in the service, so he would have difficulty in solving the problems that may arise while delivering the contents to final users.

To emulate these two scenarios, several network devices have been installed in the test environment shown in Figure 3. The installed equipment is broadly used in IP networks and allows service administrators to test radio services using different backbone and access technologies, where different routing protocols can be found. Almost every network device is from Cisco, mainly due to its position in the market and to the flexibility of its operating system *CISCO IOS*, which provides a broad range of communications techniques and standard protocols.

Two switches have been installed to give network access to every device in the test environment. These switches have 24 ports each and are interconnected using a *trunking* protocol, acting as one switch of 48 ports of 10/100 Mbps. Several *VLANs* can be created in these switches. Each *VLAN* has a set of ports assigned to the switches, so the devices plugged into those ports are working as if they were in different networks.

In order to interconnect the *VLANs*, several routers are available in the test environment. Furthermore, different models have been installed. Simpler routers can emulate small office connections, with firewall capabilities, QoS policies based in *WFQ* or *RSVP*, *NAT* and *PAT* translation, etc. Others are appropriate for larger companies or small network operators, supporting up to 70 heterogeneous network modules and up to 225 Kpps. All these routers have been equipped with different modules including *Fast Ethernet*, *ISDN*, *ATM* and serial communications.

The test environment also includes an *ATM* switch. This equipment simulates the behaviour of a network backbone that uses this technology. Several routers have been connected to this switch, emulating the connections that large enterprises contract to network operators under guaranteed bandwidth policies, connections between different network operators, or connections between network segments of the same operator.

An *ISDN* and *PSTN PBX* device has also been installed in the test environment. It allows routers to receive connections from users that access through *ISDN* or *PSTN* connections. Depending on the phone number provided by those users, the *PBX* redirects their calls to the proper router, simulating the behaviour of different *ISPs*.

A firewall has been installed in the system. In Figure 3 this firewall is restricting the access to the test environment from an external network. Nevertheless, if different networks are configured in the test environment, this firewall can be moved to apply restrictions in any of the existing connections. Furthermore, the chosen device can apply advanced firewall techniques working with *statefull filtering*. It is able to support 128,000 concurrent connections with a maximum throughput of 188 Mbps, and it can apply *NAT* and *PAT* conversions.

3.3 Content access

The last phase in every test environment is that of content access. In some way, customers' accesses

must be simulated in order to produce requests in the system. The proposed environment uses a workload generation tool, designed to request the available radio contents in the service.

This workload generation tool has been designed with three main goals: to simulate a high number of simultaneous accesses, to behave as real users do, and to work in a distributed fashion. With these goals in mind, the workload generator has been structured in three different modules: a coordinator module, a user module, and a player module. The structure of the generator is shown in Figure 4.

The coordinator module is in charge of generating a unique time line for a group of users involved in the emulation, and launching all the instances for those users. Several coordinator modules can be executed simultaneously, in order to control the workload process in different computers. Thanks to an initial interchange of control messages, the different coordinator modules are able to establish the reference start time of the emulation. After this initial coordination, these modules create one independent thread for each simulated user. Once the threads have been launched, the coordinators only have to wait until these threads reach their end. The number of users involved in the emulation, and the duration of the workload generation are the main parameters used in the configuration of coordinator modules.

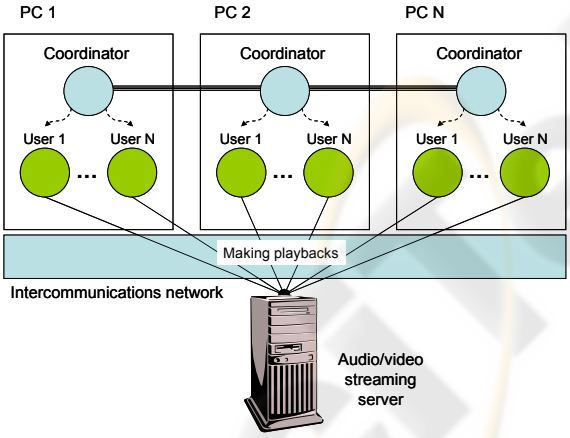

Figure 4: Workload Generator Structure

The user module emulates the behaviour of a single user. Using a pseudorandom process generation mechanism configured with information obtained from real services, the user module decides when to make a request to the server, the length of the request, and finally, the number, length and start time of the pauses to be carried out. All these parameters are taken from an input *XML* file where service managers can adjust the behaviour of the users using information obtained from their actual services. If this information is not available, works like (Chesire, 2001) or (Veloso, 2002) propose interesting characterization profiles which can be utilized to fulfil this configuration task. In this *XML* file several statistical distributions are configured with their main parameters. The duration distribution sets the length of the requests, while the inter-request time distribution sets the time between requests generated by the same user. Interaction distributions permit the generation of pauses in certain points of the reproductions. Also, if several audio channels are available, the popularity distribution sets which channels are most suitable to be chosen by this user. Although other distributions can be used, the loader calculates popularity using a zipf-like distribution; managers only need to set the proper θ parameter in this configuration file.

Once the user module has calculated all the necessary values, it passes all this information to the player module. The player module is in charge of establishing the connections with the streaming server. It is triggered by the user module, and it connects to the streaming server, requests a resource, negotiates session parameters, and receives the audio stream sent by the server. This module is based on the *HelixDNA* (RealNetworks, n.d.). This technology is the result of the collaborative effort of RealNetworks, independent developers and other leading companies, and offers the first open multi-format platform for digital creation, media delivery and playback. The developed player module acts as if it were the player of a real user. It does the same operations, apart from the audio and video rendering, which are not relevant for this environment and can limit the number of concurrent users per computer that can be emulated. Once the player has received a data packet from the server, it collects the necessary information and throws the packet away.

As shown in Figure 4, one exact replica of the workload generator runs on each of the computers used in the tests.

3.4 Test evaluation

Once the tests have been executed, it is time to analyze the obtained data and draw the proper conclusions. There are several data sources that can be used in order to extract detailed information of the executions.

Every service device has a log utility that can be used to retrieve information. Log files in multimedia services usually provide detailed information about users, resources, requests, and data packet transmission (RealNetworks, 2002). Users' information includes their identification –IP address and player ID– and their environment specification –

e.g. type, release, language, and id of the player, type and version of the operating system, etc.–. Resource information includes the URI of the stream, CPU throughput, memory consumption, license utilization, etc. Requests information comprises start and end dates and times, protocol, the amount of delivered data –in seconds and bytes–, the average bitrate, interactions made by the user, etc. Data packet transmission information includes number of data packets, disordered and lost packets, early and late packets, resent packets, failed resends, etc. An analysis tool such as our *Fesoria* (Pañeda, 2003) can be installed in the environment in order to analyze the log files provided by the multimedia server or any other service device.

It is also possible to retrieve information from the network devices installed in the environment. In Figure 3 a machine called *NMS* or *Network Management Station* collects information from network devices thanks to the *SNMP* protocol. While the tests are running, this computer polls routers, switches and firewalls to obtain traffic statistics and performance information. It is possible to track network latency, packet loss, traffic, bandwidth, CPU loads, etc.

The workload generator also provides detailed information about the incidents that occurred during the executions. Its user module generates a trace file with relevant information. This trace file holds information about significant events that have happened during the playbacks, and final statistics of the reproductions. An example of trace file follows:

```
<trace id_user="0" delay="32">
  <playback num="0">
      <url time="1104915121">
         rtsp://192.168.100.3/radio.rm
      </url>
      <begin>1104915122</begin>
      <playing>1104915122</playing>
      <buffering>1104915122</buffering>
      <playing>1104915125</playing>
      <stop>1104915132</stop>
      <statistics>
         <normal>277</normal>
         <recovered>8</recovered>
         <received>285</received>
         <lost>0</lost>
         <late>0</late>
         <bw>225000</bw>
      </statistics>
      <end>1104915132</end>
  </playback>
</trace>
```

Trace files are *XML* documents, and include the types of events and the times when they occur. Taking into account that the end of a given event corresponds to the beginning of the following one, the duration of those events can also be known. E.g. when the buffering process ends, the playback process starts, and when the latter ends, the stop event happens. These files are easily readable and can be used by analysis tools in order to generate detailed reports of the emulations. These trace files can also be installed on a web server in order to publish the results and see them on-line. An *XSL* template has been designed to generate *SVG* graphs with the results of the simulations. Figure 5 shows an example, where packet statistics are exposed.

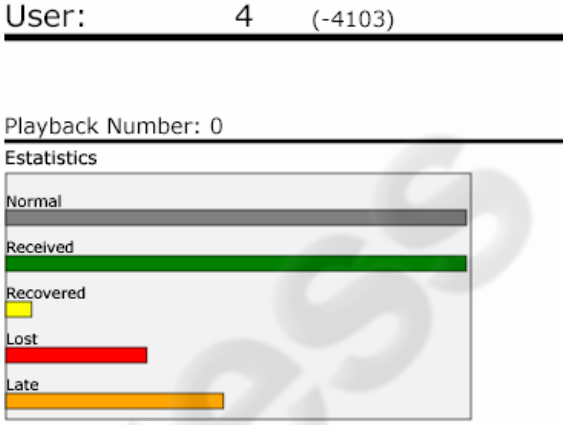

Figure 5: *SVG* graph with packet statistics

Service managers can analyse the obtained results through these graphs in a more intuitive way.

4 CASE STUDY

The digital news service **Asturies.Com** available from the domain www.asturies.com offers an Internet Radio service with considerable success. This service has been operating since April 2004 and offers a 24 hour channel of contents closely linked with Asturian culture: traditional and modern music, special programmes, news, jingles, etc.

By installing the Asturies.Com service in the test environment, we are able to estimate the maximum amount of users with a certain behaviour that can be supported by its current configuration and the increase in the number of users if configuration changes are applied to the service.

4.1 Service description

The Asturies.Com radio service is based on the distribution of stored contents. This service has an architecture similar to that shown in Figure 2. Several files are stored daily in the main server. In this machine, both an *slta* module and a multimedia

server have been installed. The *slta* generates a continuous stream of audio using the available files, and sends that stream to the multimedia server, in charge of distributing the contents among service users. The *Helix Universal Server* has been installed as multimedia server, and a license that allows up to 10 Mbps of transmission rate has been acquired. No intermediate devices have been installed.

The server machine has been installed in the network of the cable operator **Telecable**, under a *housing* contract. A service level agreement of 5 guaranteed Mbps has been established with this operator, in order to attend users' requests.

The utilized contents are coded using RealNetworks' *surestream* technology, which allows service manages to include several qualities in the same audio file. The responsibility of choosing the correct quality from the file is delegated to the multimedia server, which selects the most suitable depending on the quality of the connection with each client. Asturies.Com uses 11 kbps, 16 kbps and 20 kbps as reference qualities to produce the audio contents.

After almost one year of life, the service has registered thousands of users' accesses. During this period of time, we have been analyzing all these accesses, and have extracted several conclusions about the behaviour of the users of the service. Although the main goal of this paper is not to offer an extensive study on users' behaviour it is important for the case study to know certain details of this behaviour.

The Asturies.Com radio service presents the interesting characteristic that the length of the requests can be represented using an exponential distribution with parameter μ=1,470.15 seconds. This feature is shown in Figure 6 and it will be used to configure the workload generator during the tests.

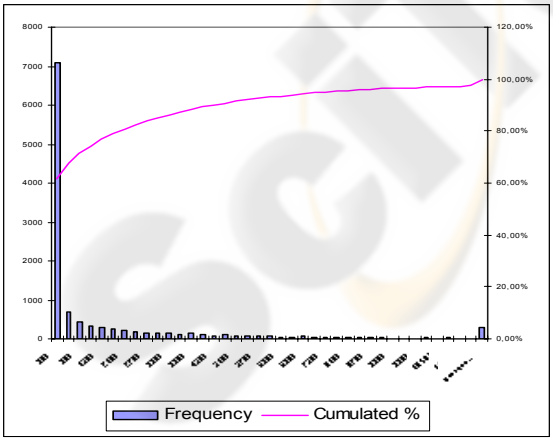

Figure 6: Length of reproductions

4.2 Case study description

Given the Asturies.Com service description and its users' behaviour, we would like to know the maximum amount of users that could be supported by its current service configuration.

There are principally two limits in the current configuration that can influence the number of accepted users: the contracted bandwidth and the license of the *Helix* server. Depending on the number of users, more bandwidth can be requested from the network operator, and a higher license can be acquired in order to increase the server limit. Nevertheless, all these changes in the service are expensive and must be studied in detail. In order to know what changes should be carried out, it is important to estimate the number of users that will be supported by the service. An accurate estimation is essential to calculate the investment profitability.

The service configuration of Asturies.Com Radio has been deployed in the test environment, and the workload generator has been configured for six different workloads. Each workload simulates a different amount of users behaving as shown in Figure 6 during a period of one hour. These users can make successive requests depending on the inter-request time obtained from an exponential distribution with parameter μ=600 seconds.

4.3 Results

The results obtained after running the tests are shown in Table 1 and the evolution in the bit-rate consumed in the server is shown in Figure 7.

Table 1: Maximum bit-rate values (Mbps)

Users	Max. Bitrate	Users	Max. Bitrate
100	1.71	500	8.37
200	3.18	600	9.75
300	4.87	700	11.4
400	6.62		

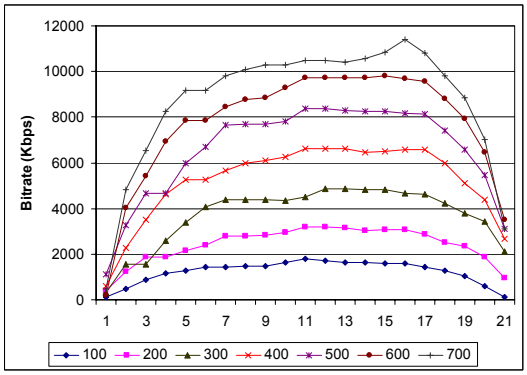

Figure 7: Bit-rate evolution

With the current limit of 5 Mbps of contracted upload capacity, 300 users can be processed by the service. On the other hand, if more bandwidth is contracted and the current server license is kept, a maximum of 600 users can be accepted by the service.

During the tests no performance issues were detected in the main server, reaching a maximum value of 5 % of CPU load.

5 CONCLUSION

The configuration and deployment of an Internet radio is a complex process, due to the high resource consumption of these services, the difficulty of transmitting continuous information over a shared data network, and the heterogeneity of cases that can be found in users' access connections. Nowadays, the configuration of one of these services is mainly based on managers' experience. However, a test environment flexible enough to predict what will happen if changes are applied to actual services, is a powerful tool for these administrators.

The presented test environment allows administrators to emulate the behaviour of almost any real radio service: different network configurations can be tested, different service architectures can be used, it is possible to work with the quality of contents, etc. This test environment can help managers to attain a service of quality, increasing its performance and profitability, at the same time as customers' satisfaction.

6 FUTURE WORK

Now that a test environment has been designed, it is time to analyse the behaviour of service users. This analysis will permit the configuration of the workload generator in order to behave as users do.

Once the behaviour of users has been analyzed, several studies will be undertaken in order to estimate the impact of network configurations on multimedia services: transport and routing protocols, network address translations, proxies, firewalls, etc.

ACKNOWLEDGEMENT

This research has been financed by the operator **Telecable de Asturias S.A.U.** and the newspaper **La Nueva España** within the *Media XXI II* project, and the **Spanish National Research Program** within *INTEGRAMEDIA* project (TSI2004-00979). This research has also received the cooperation of **Asturies.Com** and its Internet radio service.

REFERENCES

Banerjee, S., et al, 2002. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM 2002*.

Bolot, J., 1993. Characterizing End-to-End Packet Delay and Loss in the Internet. In *Proceedings of ACM SIGCOMM Conference 1993.*

Chawathe, Y., 2000. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*, Ph.D. Dissertation, UC at Berkley, U.S.A.

Chesire, M., et al, 2001, Measurement and Analysis of a Streaming-Media Workload. *Proceedings of USITS'01*

Deshpande, H., et al, 2001. *Streaming Live Media over a Peer-To-Peer Network*, Standford University, U.S.A.

Dutta, A., et al, 1999. MarconiNet – An Architecture for Internet Radio and TV Networks. In *Proceedings of NOSSDAV99 Conference*.

Dutta, A., et al, 2001. Streaming Architecture for Next Generation Internet. In *Proceedings of ICC 2001*.

Holbrook, M., et al, 1999. IP Multicast Channels: EXPRESS support for large-scale single source applications. In *Proceedings of ACM SIGCOMM'99 Conference.*

Melendi, D., et al, 2004. Deployment of Live-Video Services Based on Streaming Technology over an HFC Network. In *Proceedings of ICETE 2004 Conference.*

Mena, A., et al, 2000. An empirical study of real audio traffic. In *Proceedings of Infocom 2000.*

Jian, W., et al, 2000. Modeling of Packet Loss and Delay and Their Effect on Real-Time Multimedia Service Quality. In *Proceedings of NOSSDAV 2000.*

Jonas, K., et al, 1998. Get a KISS – Communication Infrastructure for Streaming Services in a Heterogeneous Environment. In *Proceedings of ACM Multimedia 98*.

Padmanabhan, V. N., et al, 2002. Distributing Streaming Media Content Using Cooperative Networking. In *Proceedings of ACM NOSSDAV 2002*.

Pañeda, X.G., et al, 2003. Analysis Tool for a Video-on-Demand Service based on Streaming Technology. *Lecture Notes in Computer Science, LNCS2720*

RealNetworks, The Helix™ DNA Client. Retrieved from https://helixcommunity.org/2002/intro/client

RealNetworks, 2002. *Helix Universal Server Administration Guide*, RealNetworks, Inc.

Veloso, E., et al, 2002. A Hierarchical Characterization of a Live Streaming Media Workload. In *Proceedings of ACM SIGCOMM Conference 2002*.