Testing and Verifying an IPv6 Based Multicast Network

Vilmos Bilicki
University of Szeged
Department of Software Engineering
6720 Szeged, Hungary
bilickiv@inf.u-szeged.hu

Abstract

Today we are witnessing the widespread introduction of the Triple Play Service. The main Television Broadcasting companies start, or plan to start, their IPTV services. The current data network infrastructure and the unicast communication paradigm are not effective for providing such services. The multicast data communication paradigm is as old as the World Wide Web but, even so, it has no wide acceptance or deployment. With IPTV this situation may change. Due to the small address space IPv4 cannot provide the necessary support for multicast communication. It may happen that multicast will be the main driving force behind the widespread use of the IPv6 protocol.

With the IPv6 multicast network there will be many new protocols and implementations. The testing of these implementations is not easy without protocol validation tools and, unfortunately, we were not able to find a freely available framework on the net for protocol validation.

Testing an already deployed network can provide valuable information about future situations and services. System administrators need a framework that can also supply them with traffic orchestration and measurement data. We were not able to find a framework that was easy to use and would allow us to create a distributed orchestrated testing procedure for an arbitrary network with arbitrary protocols.

These were some of the reasons for setting up our NetSpotter project. Here we will show the architecture and the services our framework currently offers. Then we will also present our measurements of the Linux IPv6 PIM-SM implementation called MRD6.

1. Introduction

The number of users with broadband Internet access is skyrocketing. According to estimates in [Rob05] and [Eni05], the number of users with broadband access in the U.S increased by 36% in 2004. Now almost 70% of all U.S home users have broadband connections. These users are a potential market for new services. One typical service package is the Triple Play service where the users get Internet access, Voice over IP and IPTV services on a single broadband connection. The IPTV service might become the "Killer Application" or the next big hit for widespread multicast usage. Here a large number of users will be connected to the same streaming channel and, in this case, the unicast data transfer model will not be an effective and scalable solution. Currently there are only a few applications which utilise multicast support of the network. These applications come mainly from the cluster or grid world like: Jboss [JBoss], the BEA Weblogic [Weblogic] cluster consistency provider framework or other distributed applications like our LanStore [Bil05] distributed storage.

It is well known that the IPv4 address space is a valuable resource. This is true for the Class D addresses as well. So it could happen that the Triple Play solutions will become the driving force behind IPv6. One of the most attractive features of the IPv6-based networks is their multicasting capability. Due to their large address space many addressing solutions can be applied. The use of scoped addresses is another potential area for efficient traffic engineering.

With efficient bandwidth usage we also get some challenges. In multicast routing a new approach was needed for loop avoidance. The large number of groups can be a critical issue as well. In contrast with web and email traffic, the VoIP and the IPTV services are sensitive to delay and jitter. The network operators should audit their networks to see how they can cope with the new challenges. The frequent testing of a network may give administrators some useful data and experience on making preparations for special situations that may arise.

2. Related work

A popular approach in network testing is one of using traffic generators. There are many interesting applications for traffic generation. But they are very simple approaches or they are not maintained. One of the best known freely available traffic generators is the D-ITG[D-ITG] package, which supplies the

user with a distributed testing capability. In its current state it is a miscellaneous collection of utilities. The distributed control of the agents is done with the help of a propriety protocol. The agents listen in on a specified port for instructions. One may write and implement software to control them. It supports many protocols (CP, UDP, ICMP, DNS, Telnet, VoIP (G.711, G.723, G.729, Voice Activity Detection, and Compressed RTP)). One advantage of this solution is the support of different probabilistic distributions for modelling different traffic scenarios. It also supports IPv6. The software package is written in C++ and it has been ported to both Linux and Windows. One can if one wishes use a Java-based GUI for managing a single agent. Compared to our approach where the user has the freedom to construct arbitrary packets, this one just has a fixed set of supported protocols. Our approach provides a message sequence chart editor where the user can specify arbitrary sequences and the task of synchronising the participants is the duty of the server In D-ITG the distributed testing scenarios may be defined in configuration files (without synchronisation) or they may be managed from a remote controller, but currently there is no tool comparable to our MSC editor for orchestrating different distributed traffic situations. We have not found any information about the support for IPv6 multicast testing on the net. The only suitable one we found was the software package DBeacon[DBeacon]. It was the only one available for this purpose. Although it is a very useful tool, it lacks a number of important features like membership testing and multipoint to multipoint testing. One can manually create arbitrary configuration files, but in this case the system administrator should do the work. It may be the best tool for a simple multicast network testing procedure where we are not actually interested in different traffic scenarios, but just want to know whether the network works or not.

3. Our solution

Our goal was to design and implement a general platform for network testing and protocol validation. To achieve this goal we set the following criteria for our framework:

- The user can define every bit of information of the sent and received packets.
- The user can define arbitrary sequences from previously defined set of messages.
- The user can define arbitrary scheduling for incoming and outgoing messages.
- The user can define a distributed scenario where there are several traffic sources and destinations arbitrarily located on the network.
- The system should be easy to use (user friendly).
- To reduce the burden of looking after a distributed system, it should be managed from one central point.

With this functionality we can not only test a system but we can also validate and check the conformance of different protocol implementations.

3.1 Architecture

To fulfil the above criteria we opted for a centralised solution. As the reader will notice in the picture there is a central server and an arbitrary number of agents.

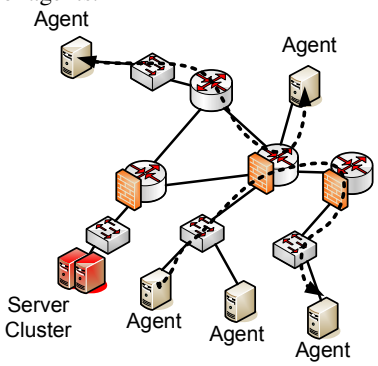

Figure 1.

The agents have their independent ability to execute the scenarios defined by the central server. They are the source and the destination of network traffic and they may be the sampling points too. In the central point of our framework there is a server where the user can orchestrate different traffic scenarios. As we may like to provide access to our system from different locations, and which may be separated by firewalls, we opted for a web based user interface. Due of special user interface requirements we implemented the interface as a Java Applet (Figure 2).

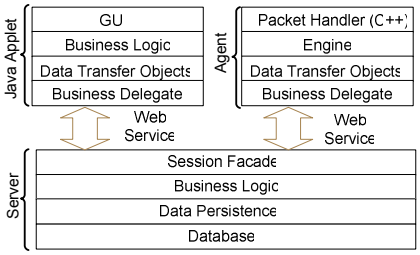

Figure 2.

In spite of the effectiveness of multicast communication, we decided to use unicast communication between the agents and the central server because of its simplicity and firewall friendliness. The agents may be placed on network segments that are separated from the central server by firewalls; hence we use web services as a communication channel between the central server and the agents. As we would like to test the network it may happen that there is no connection between the server and one or more agents. We

found a solution for this problem in the DBeacon software package where there is no central point and the whole system is built as a peer-to-peer solution. Owing to its complexity and unpredictable nature we later decided to reject this solution. To overcome the network failure between the server and the agent one can manually copy the scenario file to the failing agent. We do not require special purpose or dedicated machines for an Agent role. As they may function as normal desktops due to security constraints it is not a good idea if they act as servers. So the communication is effectively one way. The agents can access the central server but the central server cannot initiate communication. To ensure the manageability of the agents they are connected to the central server by a given schedule. The defining of this schedule is the duty of the central server.

For some measurements, scheduling is critical. Suppose, for instance, we would like to measure the delay between the sending and the receiving of a multicast RTP packet. As the clocks of the agent machines may not have been synchronised properly, we can not rely on them. But we can provide two solutions for this problem. An offline solution is one where the agent sends its local clock value to the central server during the to-do list download. The central server modifies the scheduling based on the difference between its clock and the agent's clock. This solution can be used in most situations, but when precise scheduling is needed and different clock speeds are not tolerated an online solution may be used. The agents connect to a special scheduler method which returns when all the agents have been connected and the clock on the central server hits a given value.

The central server could be a single point of failure, but as we would like to use this system for the continuous testing and monitoring of a network a failure of the system cannot be tolerated. Hence we designed and implemented a multilayer approach whose diagram is shown in Figure 2. Both the database layer and the business logic may be clustered. The logic is implemented as EJB 3.0 session beans. Some of them just have a Web Service interface for the agents and controlling Applet. We used POJO's to represent the data. The persistence of these objects is handled by the Application server.

3.2 Services

In this subsection we would like to describe the services provided by our framework and the way they were implemented by us.

3.2.1 Network handling

As the Java language is a high level language and the development cycle is shorter than that for an unmanaged environment, we implemented the client in this environment. The biggest challenge for us was raw network handling. The Java platform provides only high level network handling beginning with its capability for socket handling. As we would like to give the user the chance to define an arbitrary packet we extended the capabilities of the Java platform with a new API to handle raw network traffic. We implemented this functionality in C++ and ported it to the Linux and Windows platform. With this API one can send MLDv2[RFC3810] packets from a Windows box that does not have the capacity to handle MLDv2 packets, or one can send PIM-SM[PIM-SM] Hello messages from a machine which is not a router. The Java RTP stack can send IPv6 RTP packets only with a unicast source and destination addresses that have DNS entries. In some cases this is not available. With our solution the user can define RTP packets and handle them without relying on a DNS service.

3.2.2 Agents

The agents are installed on different machines in different parts the network, independently of the number of firewalls between the agents and the central server. The first task of the agent during the start-up procedure is to register itself on the central server. During this process the agent transfers all of its special properties to the server like the number of interfaces and the defined IP addresses. This data is refreshed only when needed. The user may group the agents and define specific properties for them (e.g. message sequences).

3.2.4 Templates

The freedom to define arbitrary messages is not of much value without an easy-to-use toolset. No one will define a message sequence one bit a time and calculate the checksums as well. Hence we designed and implemented a powerful template engine for this. The templates have the following properties:

- Inheritance
- Composition
- Auto fields
- Alias handling

With the help of inheritance one can define message families from less specific to the most specific messages e.g. IPv6 packet, IPv6 packet with UDP encapsulation, or an IPv6 packet with a UDP or RTP encapsulation. With the help of composition we can achieve the same results. With these solutions one can define message libraries and reuse them. And using auto fields one can define the content of a field to be filled by the GUI. The

checksum is a good example where the user may select the fields from which the checksum is calculated. The user may define friendly aliases and use them in the GUI instead of the long IPv6 addresses. Another example is when the user would like to set up a large message sequence and the difference between the preceding and subsequent message field can be defined as a logical expression. With these features a time consuming test case setup may be less monotonous for the user and be less error prone.

3.2.3 Sequence definition

To describe the message sequences we constructed an easy-to-understand XML syntax based on the ITU-T. Z.120 [Z120] message sequence chart recommendation. We then selected the most interesting subset of the functionality defined in Z.120 for the implementation. With the help of the GUI (shown in Figure 2) the user can define sequences for an arbitrary number of agents. These sequences are then stored in the database. When an agent downloads its own sequence, the server creates a customised sequence with synchronisation and collects the messages from the general sequence that are of interest to an agent. In this way the user is able to create complex scenarios and the agents will just receive the communication sequences they are involved in.

3.2.5 Probabilistic functions

We applied several well known probabilistic distributions that are used in telecommunication and traffic modelling fields. One can define the value of an auto field as an output of a probabilistic function.

3.2.6 Reporting

The user can define the interesting properties to measure during a test. This might be the measured traffic parameters like delay, jitter or the difference between the defined and the received message sequence. The result might be the whole received message sequence (without data). The results of a measurement are transferred to the central server after the measurement has been taken. On the server side one can use the visualisation framework to analyse the results.

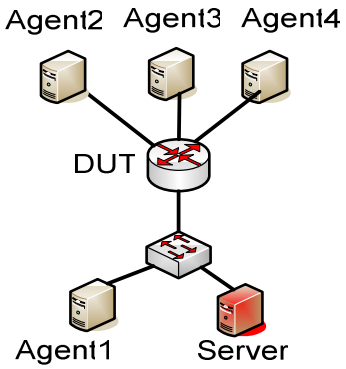

Figure 4.

4. Measurements

For a system administrator to guarantee the continuous operation of the managed network, a good knowledge of the capabilities of the network is needed. Our experience shows that a common solution used by most administrators is to monitor the network with the help of an SNMP based software package. This solution may provide same knowledge about the actual state of the network but, it cannot provide much information about the effects of planned or unplanned special events on the network. For example whether the company has decided to migrate the voice communication from the POTS to a VoIP solution based on the current network. Due to the undetermined nature of the network traffic, the complexity of the network and lack of detailed documentation about the capabilities of networking devices, the analytic approach for predicting the possible impact of the new network traffic in most cases cannot be used. A more popular and useable approach is to measure the network in different scenarios. Currently there are only basic devices available for this task. Most traffic generators can only be used with fixed configurations and as they intended to be desktop applications they are not meant to be used as distributed applications. The recommendations for system testing are mostly based on stress tests. We think that knowledge of the behaviour of the managed network in an everyday situation could be more important than during peak periods. In spite of well known theoretical models for various types of traffic we were not able to find any suggestions about the kind of measurements we should make.

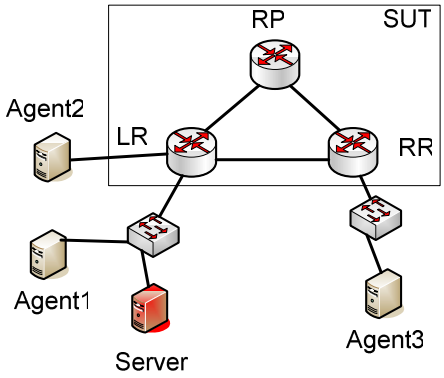

Figure 5.

4.1 IPv6 multicast measurements

Our original goal was to test the capabilities of the Linux IPv6 multicast router especially the PIM-SM implementation. The RFC 3918 [RFC3918] describes the methodology of IPv4 multicast testing and RFC 2432 [RFC2432] describes the terminology used in this field. These documents only specify a single source multiple receiver testing scenario. The [IPv6benchmarking] draft contains several additions to the benchmarking methodology which can be interesting for IPv6 benchmarking. Below we will show our results for IPv6 multicast group capacity and join delay in different traffic scenarios and network topologies.

4.1 The configuration used

We set up a sample configuration shown in the pictures 3 and 4 with Linux IPv6 PIM-SM [PIM-SM] routers and Linux-based clients for them. The machines had the following configuration:

- Software:
  - Debian Sarge
  - MRD6 0.9.5 PIM-SM implementation [MRD6]
  - Zebra Ripng as a unicast routing algorithm
  - Java 1.5_06

Table 1. contains the hardware specifications of the machines.

	Processor	Memory (MByte)	Network card (3Com 100MBit/s)
RP (rendezvous)	P4 1300 MHz	512	2
RL	P4 1300 MHz	512	4
RR	Celeron 600 MHz	256	3
Agent1	P4 1300 MHz	512	1
Agent2	Celeron 600 MHz	256	1
Agent3	Celeron 600 MHz	256	1

Table 1.

4.2 The number of supported channels

In the experiments our goal was to learn more about the dependence between the number of channels and the packet loss rate. Our tests were done with an equal number of packets (50000). For testing traffic we used an IPv6-based UDP packet of variable length and fixed content. The only varying parameter in the UDP was a serial value. On the receiver side the received serials were the result. Each MLDv2 packets contained 50 multicast addresses with exclude directive.
We conducted the measurement for both topologies (SUT and DUT, Figures 3,4). In both cases the traffic source was Agent3 and the traffic destination was Agent1. The number of received packets is shown in the Table 2.

N.Ch	64	512	1500
10	50000	50000	49200
100	49514	49664	43311
1000	46813	43808	41642
10000	n.a	n.a	n.a
60000	n.a	n.a /	n.a

Table 2.

Evaluation:
The system worked well up to 100 channels. With 1000 the packet loss rate increased, but only to about 2-10%. With a larger packet it was greater. If we injected the same traffic several times the packet loss rate decreased by 1-5%. We suppose the reason for this behaviour can be found in the FIB implementation. When we chose 10000 channels or more the system could not cope with it. The RR router processed about 4300 subscriptions and from these subscriptions only 3150 were registered on LR. We slowed down the subscription rate but the best result we were able to achieve was that of registering 5600 channels on RR and 2947 channels on LR. It was surprising to us that the RR started sending PIM-SM Join messages only after processing the majority of the MLDv2 Register messages, rather than in parallel. It seems that the MLDv2 handling task has higher priority than the PIM-SM signalling task. The multicast traffic for 10000 channels generated by Agent1 used about 60 MBit/s of bandwidth. Despite this low value the LR was totally overloaded during PIM-SM Register packet generation. From this experiment we may conclude that this system is well able to handle some 10-40 channels. Clearly the number of channels handled by the routers strongly affects the performance of a multicast network.
The DoS attack on a multicast network aided by a large number of multicast channels can pose a real threat. The real network traffic is not significant (in the case of MLDv2 Join packets, several tens of ICMPv6 packets), but the impact of this traffic might be devastating. So we need safeguards.

4.3 The channel join delay

Here we measured the channel join delay for different channel numbers. We measured the time between the last MLDv2 packet and the first arriving UDP packet in milliseconds. The results that we obtained are listed below.

N.Ch	64	512	1500
10	17	23	14
100	227	254	319
1000	3800	3700	4200
10000	72777	>70000	>70000
60000	>70000	>70000	>70000

Table 3.

Evaluation:
It seems that the delay is proportional with the number of channels. For larger packets the delay is larger but the difference is not significant.

5. Conclusions and future work

In this article we presented our new network testing and protocol validation framework. The strength of this framework lies both in its user friendly GUI and the support it provides for defining a network traffic from top to bottom. However, the framework is still incomplete. This was the reason why in our testing scenarios we used and described only some of its features. The first release of it is planned in Q3 2006. As we mentioned earlier, the current network testing scenarios are mostly concerned with benchmarking. We think that measuring a real network situation with a lot of agents can provide the same or more valuable data than that obtained from benchmarking. The probabilistic approach where the traffic parameters are defined in terms of known probabilistic functions will add new data to the network testing field.
Here we did not evaluate the protocol validation capability, but rather we measured the channel handling capabilities. But we think that the protocol validating capability should be widely used among network protocol implementers. During the testing phase it turned out that, based on RFCs, it is not a trivial task to fully specify a packet in detail. Hence we would like to define the most interesting protocols for our framework and we plan to make these sample configurations available on a community site.
In our experiments it turned out that the multicast network can be an easy target of a DoS attack. With a relatively small packet number a multicast network can be shut down.
The whole system was developed in the Java language so it is portable. The software package is available under a GNU GPL licence from http://netspotter.sf.net.

6. Acknowledgement

I would like to thank David P. Curley for checking this article from a linguistic point of view. I would also like express my gratitude to Sándor Dojcsák, Barnabás Tajti and Viktor Kertész for their creative ideas and the meticulous work done while implementing our new framework.

7. References

[Rob05] Rob McGann. Broadband: High Speed, High Spend. http://www.clickz.com/stats/sectors/broadband/article.php/3463191, January 2005

[Min05] Miniwatts Marketing Group. INTERNET USAGE STATISTICS - The Big Picture. http://www.internetworldstats.com/stats.htm , December 2005

[JBoss] Burke B., Labourey S. Clustering with JBoss 3.0. November 2002. http://www.onjava.com/pub/a/onjava/2002/07/10/jboss.html

[WebLogic] BEA. Using WebLogic Server Clusters. 2006. http://edocs.bea.com/wls/docs81/cluster/overview.html

[Bil05] V. Bilicki. LanStore: a highly distributed reliable file storage system, .NET Technologies'2005 conference proceedings, ISBN 80-86943-01-1, pp. 47-57, 2005

[D-ITG] Avallone S, Botta A., Emma D., Guadagno S, Pescap A. D-ITG V. 2.4 Manual. 2004. http://www.grid.unina.it/software/ITG/codice/D-ITG2.4-manual.pdf

[DBeacon] Santos H. dbeacon, a Multicast Beacon. http://artemis.av.it.pt/~hsantos/dbeacon/

[RFC3810] Vida R., Costa L. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. 2004. http://www.faqs.org/rfcs/rfc3810.html

[PIM-SM] Fenner B., Handley M., Holbrook H. Kouvelas I. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). 2006. http://tools.ietf.org/wg/pim/draft-ietf-pim-sm-v2-new/draft-ietf-pim-sm-v2-new-12.txt

[RFC3918] Stopp D., Hickman B., RFC 3918 - Methodology for IP Multicast Benchmarking, 2004, http://www.faqs.org/rfcs/rfc3918.html

[RFC2432] Dubray K., Terminology for IP Multicast Benchmarking, 1998, http://rfc.net/rfc2432.html

[IPv6benchmarking] Popoviciu C., Hamza A., Velde G., Dugatkin D., Kine B., IPv6 Benchmarking Methodology, 2006, http://www.ietf.org/internet-drafts/draft-popoviciu-bmwg-ipv6benchmarking-00.txt

[Z.120] ITU-T, Formal description techniques (FDT) – Message Sequence Chart, 1990, http://www.itu.int/ITU-T/studygroups/com10/languages/Z.120_1199.pdf

[MRD6] Santos H., MRD6 - an IPv6 Multicast Router, http://artemis.av.it.pt/~hsantos/dbeacon/