Supporting IP Multicast Streaming Using Overlay Networks

Marc Brogle, Dragan Milic, Torsten Braun
University of Bern
Institute of Computer Science and Applied Mathematics
Neubrueckstrasse 10
CH-3012 Bern, Switzerland
brogle|milic|braun@iam.unibe.ch

ABSTRACT

In this paper we present our solution for providing IP Multicast on end systems in the Internet. The goal of the proposed solution is not to replace IP Multicast, but to provide an IP Multicast interface to applications on end systems in the current Internet environment, where IP Multicast is not available. Our solution, called Multicast Middleware, is a software, which is based on using Application Level Multicast (ALM) for transporting IP Multicast traffic. The use of the Multicast Middleware is transparent for applications on end systems, since our Multicast Middleware uses a virtual network interface to intercept native IP Multicast communication. In this paper we also present a performance evaluation of our Multicast Middleware. The results of this evaluation show that our Multicast Middleware is able to provide high bandwidth throughput to applications. This makes our Multicast Middleware a viable solution for supporting multimedia streaming services, etc.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; C.2.5 [**Local and Wide-Area Networks**]: Internet

General Terms

Performance

Keywords

transparent overlay multicast, performance, peer-to-peer, multimedia streaming

1. INTRODUCTION

The multicast communication paradigm represents a communication where a small group of senders (usually one) is sending information to a large group of receivers. In the Internet this has been realized in the form of IP Multicast [8]. IP Multicast has been proposed and specified almost two decades ago. Unfortunately, even today IP Multicast has not been widely deployed in networks of commercial Internet service providers (ISP). Some reasons for this are:

- IP Multicast must be supported by all routers on the path from source to destination.
- Additional inter-ISP coordination is required (policy issues of inter-domain routing).
- IP Multicast routing can be very resource intensive.

As a transition between the Internet without IP Multicast and its full availability to the end-user, the MBONE [9] approach has been proposed. In MBONE, the Internet is considered as a set of isolated IP Multicast enabled islands. These "islands" are interconnected by an overlay network of tunnels. The overlay network is used to tunnel the IP Multicast traffic between the MBONE islands over parts of the Internet that support only unicast traffic. MBONE tunnels are implemented using the loose source routing (LSRR IP option) or by encapsulating IP Multicast packets in unicast packets. The drawback of this approach is that tunnels have to be set up manually. As a consequence the tunnel end-points must be permanently available and require fixed IP addresses. This prohibits most Internet end-users from using the MBONE, since usually the end-users are not permanently (e.g. modem users) and do not have fixed IP addresses assigned.
Although IP Multicast is not widely available, there exist numerous applications using it. The MBONE video conferencing tools for example include among others vat (visual audio tool), nv (Network video tool), vic (video conferencing tool), etc. The Access Grid Project, for example, offers the Multicast Application Sharing Tool (MAST). Microsoft research developed the advanced collaboration and interactive distance learning software called ConferenceXP. This software also uses IP Multicast to achieve an efficient communication between the collaborating parties. The video lan client (VLC) is an IP Multicast enabled video broadcasting and video playback tool, which we also used to test and evaluate our Multicast Middleware.
Since MBONE was not able to provide multicast communication to the Internet end-users, numerous solutions were proposed to address this problem. The rising popularity of Peer-to-Peer (P2P) networks lead to a revival of multicast in the form of Application Level Multicast (ALM) [2, 3, 6, 14, 20, 23–25]. ALM mechanisms use similar methods as IP Multicast for data dissemination, but move the replication of multicast data from routers to the end systems. The

Qshine'07, August 14-17, 2007, Vancouver , Canada

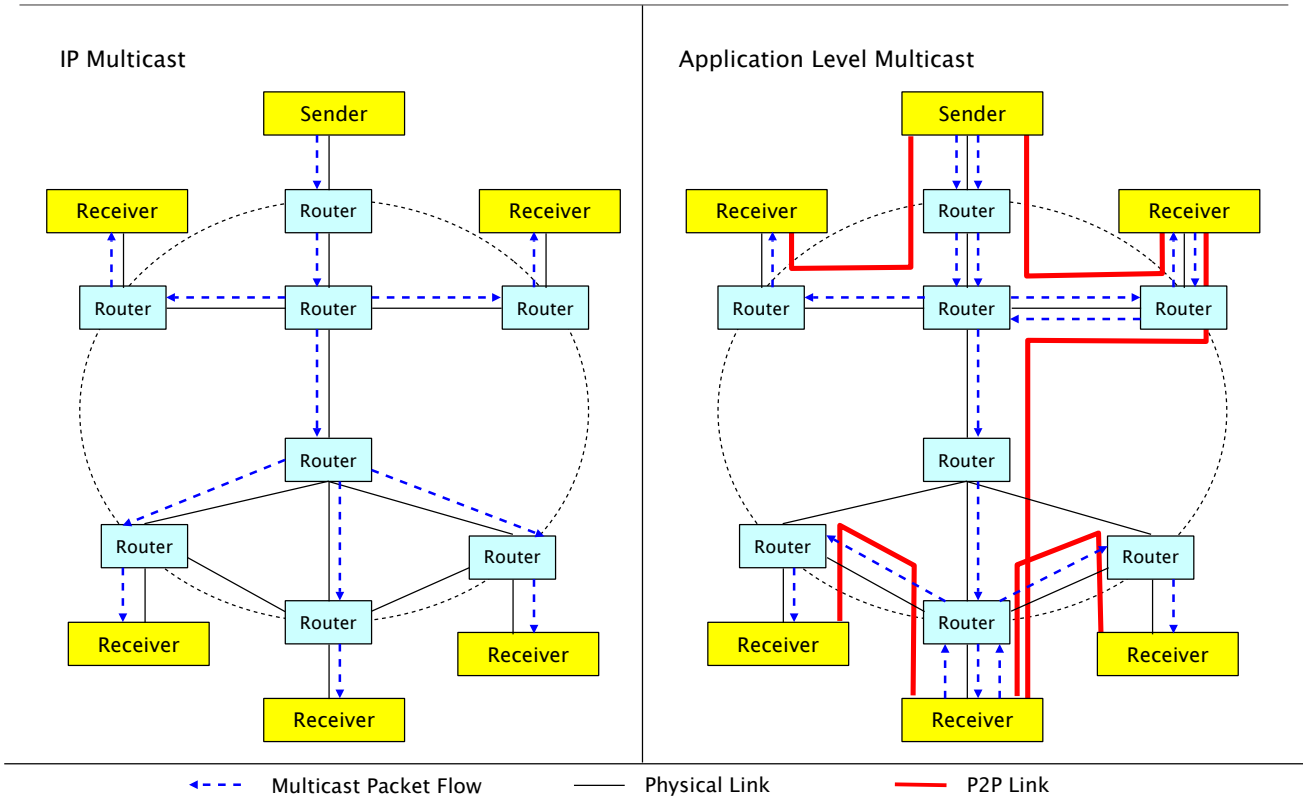

Figure 1: IP Multicast with data replication in the routers vs. Application Level Multicast

advantage of this approach is that ALM mechanisms do not require multicast support by the routers. On the other hand, the replication of data only on end systems is not as efficient as the replication in the routers. For example, the right side in Fig. 1 shows a typical ALM scenario, which uses only unicast communication between the end systems to enable multicast services. In comparison to IP multicast (presented on the left side in Fig. 1), there is some redundancy of the data that is sent over the physical links. The reason for this is that the replication of data is only done on end systems and not on routers. Although ALM mechanisms can never achieve the efficiency of IP Multicast regarding the usage of network resources, it is still much more efficient than unicast communication between the sender and all receivers.

The advantages of ALM mechanisms are, that they are designed to be self-organizing and fault-tolerant, which make them easier to deploy than the MBONE. This makes ALM mechanisms better candidates for deploying multicast services to end-users. The drawback of using ALM mechanisms is that the used protocols and APIs are not standardized, which makes application development dependent on specific ALM protocols. Another drawback is that existing IP Multicast enabled applications would have to be adapted to the specific API of ALM protocols. An overview and classification of peer-to-peer content distribution technologies can be found in [1].

To use the best of both worlds our solution uses an ALM for transporting multicast traffic over the Internet. At the same time it offers a standard IP Multicast interface to applications on end systems. This solution is based on a so-called Multicast Middleware. The Multicast Middleware enables the transparent use of ALM mechanisms for all IP Multicast enabled applications on end systems.

This is achieved by a virtual network interface intercepting and forwarding multicast packets to the Multicast Middleware. This mechanism can be used for high bandwidth and real time multimedia streaming as presented in [15], where we mainly described how the Multicast Middleware can be used for video streaming, but without using an elaborate P2P / ALM infrastructure and not having implemented Quality of Service (QoS) mechanisms. In this paper we investigate performance enhancements regarding multiple hops in an overlay network, discuss the usage of a widely used P2P / ALM infrastructure called Scribe / Pastry and briefly show how the different QoS mechanisms can be applied.

The remainder of the paper is organized as follows: In the next Section we describe communication obstacles in the Internet. Section 3 contains a description of our proposed Multicast Middleware. In Section 4 we present the performance evaluation results. In the last section we summarize results and give an overview of future work.

2. COMMUNICATION OBSTACLES IN THE INTERNET

Besides the lack of global IP Multicast support, there are further limiting factors for the communication in the Internet. The most severe are the existence of Firewalls and Network address translators (NATs).

Firewalls are network devices that filter portions of network traffic based on protocol header information and/or the data payload of IP packets. They are used to protect network devices within a private network from intrusions from the Internet. Firewalls normally limit the ability of hosts in the Internet to connect to hosts behind them. In some networks, they are also used to prevent communication of hosts behind the firewall with hosts in the Internet.
Such installations are used as a preventive measure against Trojan horses, Internet worms, etc. This also limits the use of P2P applications on hosts behind firewalls, since these applications assume universal connectivity between hosts.
The address space of the IP protocol version 4 is limited to 2^{32} addresses. In addition there are IP address ranges that cannot be used for end host addresses such as the address ranges reserved for private use, the IP Multicast address range, the network addresses, and the broadcast addresses. To overcome this limitation, NATs were introduced. NATs are network devices, which allow a whole private network to appear as one IP address on the Internet. The drawback of using NATs is that the peer-to-peer communication is affected, since the end systems behind a NAT are not able to accept any incoming connections. There are several proposals for solving this problem such as UPNP [22]. Nevertheless no universal solution for this problem is available currently. Another issue with NATs is that the idea of IP addressing, where each host has a unique address, is violated. The hosts behind a NAT usually have an IP address from a range of IP addresses are reserved for private use (10.0.0.0/8, 192.168.0.0/16 or 172.16.0.0/20), which means that there are potentially many hosts with the same IP address. This aspect of NATs also subverts different P2P node ID generation schemes, which assume that every host has a unique IP address.
All communication obstacles presented in this section also apply to IP Multicast. For an in-depth analysis of connectivity restrictions for overlay networks see [13].

3. MULTICAST MIDDLEWARE

3.1 Concept

We have developed the so-called Multicast Middleware [15], which provides an IP Multicast interface to applications on the end system and at the same time uses an ALM for transporting multicast traffic. This unique feature allows the users of end systems in the Internet to benefit from easy deployment of P2P based ALM mechanisms and still use existing IP Multicast enabled applications. To provide a platform independent solution, we have implemented our Multicast Middleware in Java.
The Multicast Middleware has been developed in the context of the European Framework Program 6 project called EuQoS [10]. The aim of the EuQoS is to enable Quality of Service (QoS) for end systems in heterogeneous networks. The task of the Multicast Middleware is to simplify the QoS provision for IP Multicast by mapping the multicast communication to unicast.
The most important feature of the Multicast Middleware is a transparent provision of an IP Multicast interface for end system applications. This is achieved by using a virtual network interface (e.g. TAP [21]) to communicate with the operating system on end systems. The virtual network interface is a special network device driver, which unlike a usual network device driver does not link to physical hardware such as an Ethernet card, but forwards the traffic to a user-space process. In our case this user space process is the Multicast Middleware. For the applications this virtual network interface acts like a "real" network interface. Data forwarding to the Multicast Middleware and processing is completely transparent to the operating system as well as to the involved applications. The Multicast Middleware uses the TAP mechanism to mimic an IP Multicast router attached to an Ethernet network, primarily by implementing the Internet Group Management Protocol (IGMP) [5, 11]. By setting appropriate routing table entries for IP multicast addresses, those packets are directed to the virtual network interface instead of the real physical network interface. All IP Multicast traffic will be redirected to the Multicast Middleware entity running on the end system. By doing so the Multicast Middleware is aware of every IP Multicast group, to which the end system is subscribed to. Applications on an end system with a running Multicast Middleware use the standard IP Multicast group management system calls. These calls are translated by the operating system into IGMP messages. The operating system communicates with our Multicast Middleware through the virtual network interface and views it as an IP Multicast router. Every IP Multicast packet leaving the end system is routed to our Multicast Middleware through the virtual network interface. Furthermore the Multicast Middleware is able to send IP Multicast traffic back to the end system through the same virtual network interface. The Multicast Middleware also participates in a P2P ALM mechanism. This ALM mechanism is used to transport the captured multicast traffic between the end systems. The described packet flow is depicted in Fig. 2.

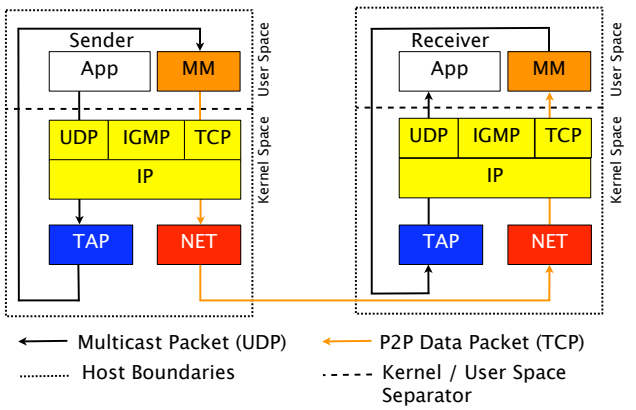

Figure 2: Packet flow between Applications and the Multicast Middleware

Each network interface requires at least one valid unicast IP address assigned to it. This is also the case for virtual network interfaces such as TAP. Since we assume that some of the end systems are behind a NAT, we have decided to assign a unique IP address from an IP address range reserved for private use (172.16.0.0/20) to the TAP device on each end system. The allocation of those IP addresses is done in a distributed manner. This avoids address clashes and ensures full time availability of the IP allocation service.
Our Multicast Middleware is not limited to use a specific

ALM. We are able to use any ALM as long as it provides standard multicast operations (joining and leaving a multicast group, receiving and sending multicast traffic).
Currently we are using Scribe [18] / Pastry [17] as ALM. Pastry is a decentralized, self-organizing and fault-tolerant overlay network. Scribe is a generic, scalable and efficient ALM running on-top of Pastry.

3.2 Scribe / Pastry

Pastry uses prefix matching routing [16] for delivering messages. Each host assigns itself a randomly chosen ID from a predefined ID space (typically a 128 bit value) when it joins the Pastry network. The IDs are equally distributed over the whole available ID space.
A simplified example is shown in Fig. 3. As we can see Pastry routing tries on each hop between source and destination to match one or more prefixes of the message's destination address. If no Pastry peer has been found at the destination address, the peer with the numerically closest ID to the destination address is responsible of handling the message.
Scribe builds on top of Pastry's routing and uses reverse-path forwarding trees to multicast messages. Each multicast group is represented by a topic ID. The host numerically closest to the topic ID becomes the root of the multicast distribution tree. All multicast messages are directly sent to this root node, which then multicasts the messages to all group members. To join a certain topic, a Scribe node sends a join message through the Pastry network.
As shown in Fig. 3, the message is routed to the root node through prefix matching. Each Scribe node visited on the path also joins the same topic and remembers, which direct child nodes have subscribed to this specific topic ID. If a node already has joined the same topic ID earlier, no additional join message will be sent towards the root node. Finally, when messages have to be multicast to all group members subscribed to a specific topic ID, the root node sends the message to its direct children, which then relay it to their direct children respectively. This is repeated until such a node has no more direct children to be served with the multicast message for the specific topic ID.
In our Multicast Middleware implementation we use one dedicated Scribe / Pastry overlay network per active IP Multicast group.
We also use one dedicated Pastry network to store the management information about active groups and free addresses from a private IP address range to be used by the virtual network interfaces.

3.3 Implementation

We have decided to use an open-source implementation of the Scribe / Pastry protocol called Freepastry [12]. Freepastry relies on Java object serialization, which is not optimal for transporting data over the Internet. The reason for this is that each time a Pastry message is de-serialized a new instance of a Java object is created. If we would be using Pastry messages to transport the IP Multicast packets, each time one packet is received from the Pastry network at least one Java object would be created. Due to automatic garbage collection mechanism in the Java virtual machine, these new objects would be de-allocated only when the heap of the Java virtual machine is full. Since the code execution of the Java virtual machine is paused during garbage collection, the packet delivery would be suspended and this would lead to increased packet delays or packet drops. We have therefore decided to use Scribe / Pastry only to construct the topology of the overlay network. For transporting IP Multicast traffic, we use an overlay network with the same topology, but using our own high performance optimized P2P protocol instead of the Java object serialization as used in Scribe / Pastry. This high performance optimized P2P is based only on copying buffers without generating any Java objects. Such a design allows us to provide high performance multicast data transfer between end systems in the Internet, which is crucial for multimedia applications such as video streaming, video conferencing, etc.
The Multicast Middleware also supports QoS for IP Multicast. To be able to provide QoS for ALM it is important that the construction of the overlay network is QoS-aware. In order to support QoS, we need a method for constructing QoS aware multicast trees. To achieve this we have proposed and implemented modifications of Scribe / Pastry, which are further described in [4]. These modifications allow the construction of QoS aware overlay networks. Using our modifications and extensions of Scribe / Pastry we are now able to enforce building multicast trees that take the QoS requirements and capabilities of peers into account. Our extended Scribe / Pastry implementation has been integrated into the Multicast Middleware.
For providing QoS guarantees for the P2P links, our Multicast Middleware relies either on using QoS offered by the transport network (as provided by EuQoS) or using a best-effort based on measurements and/or predictions [19]). In order to build the Multicast tree to support these QoS mechanisms, the construction of the Multicast tree has to take the QoS capabilities and requirements into account [4].

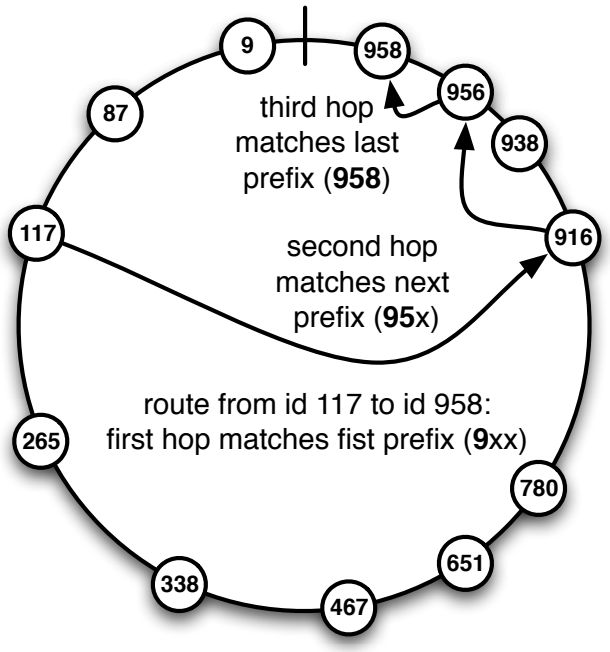

Figure 3: Simplified example of Pastry routing

4. PERFORMANCE EVALUATION

4.1 Measurement Scenarios

Usually P2P network implementations like Scribe / Pastry based on Java or other high level programming languages are not very appropriate for high-bandwidth streaming. This is i.a. due to the fact that object serialization is used for sending P2P messages through the overlay network. The object serialization implies usually a performance penalty due to the overhead of (de-)serializing objects. To overcome this problem we are using a custom binary signaling protocol and avoid creating new objects in most cases. This makes our Multicast Middleware usable for high bandwidth and real-time critical environments.

To construct an overlay network, we use the mechanisms of Scribe / Pastry. For the actual data transmission we are using our optimized messaging protocol, which has been thoroughly optimized for high bandwidth data dissemination scenarios. Our evaluation has shown that by using our optimized communication protocol performance improvements by one magnitude compared to the default P2P message protocol of Scribe / Pastry can be achieved.

Therefore, our performance evaluation is focusing on determining the maximum achievable bandwidth that can be processed between two peers. Furthermore we will also evaluate the performance of an example path from sender to receiver in an overlay network.

Since we use a Scribe / Pastry overlay network structure, which is know to scale very well with an increasing number of peers and group members [7, 17], we do not evaluate how well the overlay network performs in these terms. Due to limited bandwidth in distributed testbed environments such as Planetlab or the EuQoS testbed, it is not feasible to perform distributed high-performance measurements.

In order to evaluate the maximum bandwidth, which an instance of the Multicast Middleware can process, and to investigate the impact of forwarding IP Multicast traffic through an ALM, we have performed a series of tests with two different scenarios, which are shown in Fig. 4.

The first scenario is a P2P network consisting of only two end systems. To avoid the interference of packet generation to the performance of the Multicast Middleware, we generated the traffic on separate host and forwarded it through a gigabit Ethernet to the first peer. In the first peer, we used an Ethernet bridge functionality of the Linux kernel to interconnect the Ethernet interface with the virtual network interface. For the same reasons we built a similar scenario for capturing the traffic. The goal of this setup is to determine the maximum throughput of the best case in the P2P network, where IP Multicast traffic is tunneled directly from a sender to a receiver.

To determine the effect of chaining multiple peers to forward the traffic, we have designed a second scenario. In that scenario, we have tunneled the IP Multicast traffic through five peers in a chain. This simulates an example of a path taken on one branch of the multicast tree in an overlay network. The first and the last peer in this chain were to connected to a traffic generator respectively capture point in the same manner as in the first scenario.

We used the MGEN traffic-generating tool to generate and capture the IP Multicast traffic. For each scenario we generated 24 flows with different sending rates ranging from 11 to 241 Mbps in steps of 10 Mbps. Each flow was sent for

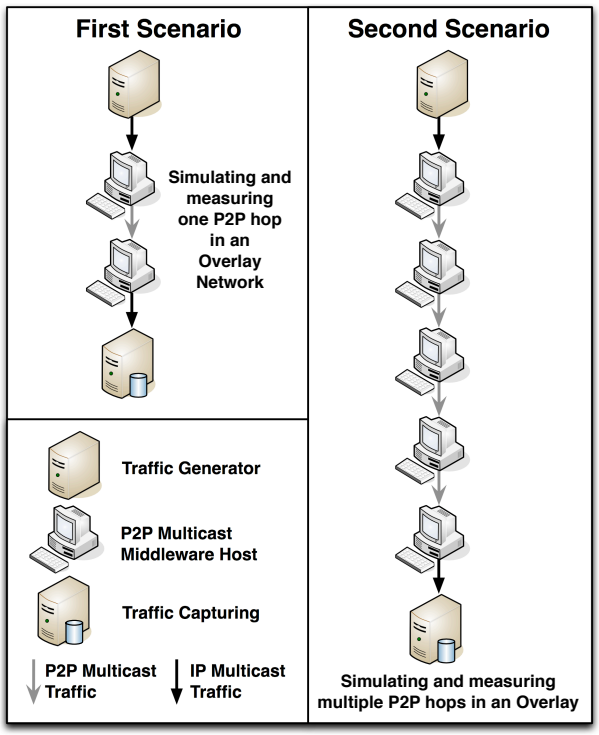

Figure 4: Scenarios for the Multicast Middleware performance evaluation

120 seconds. The payload of each packet consisted of 1024 bytes. The hosts used for both scenarios were identical in terms of hard- and software. Each of them had a Pentium D 3 GHz CPU with 1 GB of memory. The operating system installed was Fedora Core 5 Linux with kernel version 2.6.17. All hosts were interconnected via a gigabit Ethernet.

4.2 Measurement Results

Fig. 5 shows the captured bandwidth compared to the generated bandwidth for both scenarios.

The packet loss for the different transmission rates in both scenarios is shown in Fig. 6. The reason for packet loss is due to the incapability of at least one peer to process traffic at the given rate, which happens when the maximum processing capacity of an end system at a given time is reached. The packet loss for a bandwidth up to 100 Mbps is negligible for both scenarios. For transmission rates of more than 100 Mbps packet loss increases significantly. As shown in Fig. 6, the packet loss is less than 4% for a bandwidth up to 155 Mbps. Therefore, our Multicast Middleware should be able to support multimedia streaming up to 155 Mbps with acceptable packet loss.

Both figures show that there is no significant difference between the packet losses for both scenarios, which indicates that the impact of delivering the IP Multicast traffic through multiple peers is minimal. Fig. 5 also shows that the maximum bandwidth that an instance of a Multicast Middleware can deliver is 210 Mbps.

We also determine the jitter for both scenarios. Our results show that the jitter increases with the number of peers involved in transporting the IP Multicast traffic. For the first scenario the maximum jitter was below 15ms for a band-

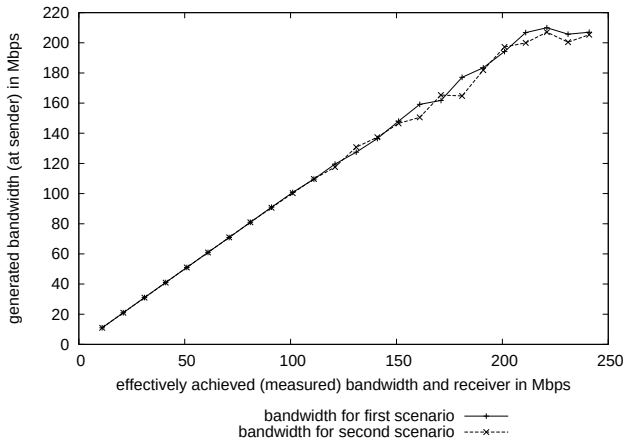

Figure 5: Generated and effectively achieved bandwidth of measurements for both scenarios

width up to 155 Mbps. For the second scenario the maximum jitter went up to 150ms, due to a few outliers. The mean delay was much lower.
For future evaluation we will compare native IP Multicast performance versus overlay multicast performance (using the Multicast Middleware) in simple, locally restricted but high-performance network environments and scenarios.

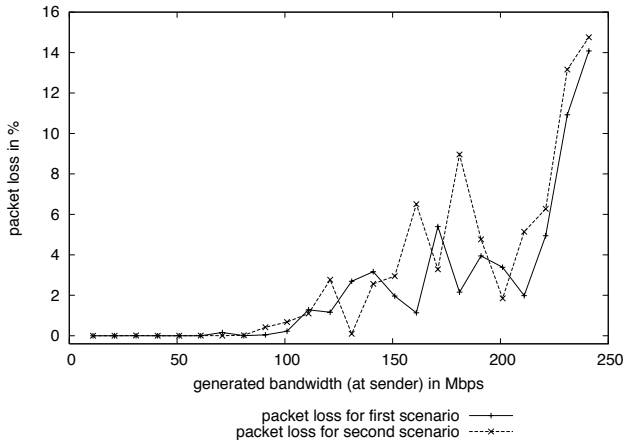

Figure 6: Packet loss measured for both scenarios

5. SUMMARY AND CONCLUSION

In this paper we have presented our solution named Multicast Middleware to provide IP Multicast on end systems over the Internet using ALM. We have described how a virtual network interface can be used to provide a transparent IP Multicast interface for applications on end systems. Furthermore we have sketched how our Multicast Middleware is used in the European FP6 EuQoS project to provide QoS for IP Multicast. Our performance evaluation of the Multicast Middleware implementation has shown that the Multicast Middleware can process up to 155 Mbps with an acceptable packet loss rate or any dramatic increase of the jitter. These results qualify our Multicast Middleware as a solution for multimedia streaming applications.

6. ACKNOWLEDGMENTS

This work was funded by the European Union 6th Framework Programme under contract IST FP6 IP 004503 EuQoS Integrated Project.

7. REFERENCES

[1] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proceedings of the ACM conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '02)*, pages 205–217, New York, 2002.

[3] A. Bozdog, R. van Renesse, and D. Dumitriu. Selectcast: a scalable and self-repairing multicast overlay routing facility. In *SSRS '03: Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems*, pages 33–42, New York, NY, USA, 2003. ACM Press.

[4] M. Brogle, D. Milic, and T. Braun. QoS enabled multicast for structured P2P networks. In *Workshop on Peer-to-Peer Multicasting at the 4th IEEE Consumer Communications and Networking Conference*. IEEE, January 2007.

[5] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3. RFC3376, October 2002.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03)*, pages 298–313, New York, 2003.

[7] M. Castro, P. Druschel, A. M. Kermarrec, and A. I. T. Rowstron. Scribe: a large-scale and decentralized application-level multicast infrastructure. *Selected Areas in Communications, IEEE Journal on*, 20(8):1489–1499, 2002.

[8] S. Deering. Host extensions for IP multicasting. RFC1112, August 1989.

[9] H. Eriksson. MBONE: the multicast backbone. *Commun. ACM*, 37(8):54–60, 1994.

[10] EuQoS project web site, available online: http://www.euqos.org.

[11] W. Fenner. Internet Group Management Protocol, Version 2. RFC2236, Novembe 1997.

[12] Free pastry web site, available online: http://freepastry.org.

[13] A. Ganjam and H. Zhang. Connectivity restrictions in overlay multicast. In *Proceedings of the 14th ACM international workshop on Network and operating systems support for digital audio and video (NOSSDAV '04)*, pages 54–59, New York, 2004.

[14] M. Kwon and S. Fahmy. Path-aware overlay multicast. *Comput. Networks*, 47(1):23–45, 2005.

[15] D. Milic, M. Brogle, and T. Braun. Video broadcasting using overlay multicast. In *ISM '05: Proceedings of the Seventh IEEE International Symposium on Multimedia*, pages 515–522, Washington, DC, USA, 2005. IEEE Computer Society.

[16] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environ ment. In *SPAA '97: Proceedings of the ninth annual ACM symposium on Paralle l algorithms and architectures*, pages 311–320, New York, NY, USA, 1997. ACM Press.

[17] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Nov. 2001.

[18] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. Scribe: The design of a large-scale event notification infrastructure. In J. Crowcroft and M. Hofmann, editors, *Networked Group Communication, Third International COST264 Workshop (NGC'2001)*, volume 2233 of *Lecture Notes in Computer Science*, pages 30–43, Nov. 2001.

[19] M. Scheidegger, T. Braun, and F. Baumgartner. Endpoint cluster identification for end-to-end distance estimation. In *International Conference on Communications, Istanbul, Turkey*. IEEE, June 2006. CD-ROM.

[20] A. Sobeih, W. Yurcik, and J. C. Hou. Vring: A case for building application-layer multicast rings (rather than trees). In *Proceedings of the The IEEE Computer Society?s 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, pages 437–446, Washington, DC, 2004.

[21] Universal TUN/TAP driver, available online: http://vtun.sourceforge.net/tun/.

[22] Upnp device architecture 1.0, upnp forum, version 1.0.1, available online: http://www.upnp.org.

[23] J. Zhang, L. Liu, C. Pu, and M. Ammar. Reliable peer-to-peer end system multicasting through replication. In *Proceedings of the Fourth International Conference on Peer-to-Peer Computing (P2P'04)*, pages 235–242, Washington, DC, 2004.

[24] R. Zhang and Y. C. Hu. Borg: a hybrid protocol for scalable application-level multicast in peer-to-peer networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 172–179, New York, 2003.

[25] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20, New York, NY, USA, 2001. ACM Press.