Source Specific Multicast (SSM) with IPv6

Stig Venaas
UNINETT
Trondheim NO 7465
Norway
venaas@uninett.no

Tim Chown
School of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, United Kingdom
tjc@ecs.soton.ac.uk

Abstract

Source-Specific Multicast (SSM) is a more recent form of IP Multicast compared to the traditional Any Source Multicast (ASM) model. In making a fundamental assumption that there is only a single source in a specific Multicast group, simplifications can be made that lead to a new form of multicast that should be easier to deploy than ASM. The 6NET and Euro6IX projects have been at the forefront of SSM testing and deployment. In this paper we describe the results of work undertaken in these projects.

1. Introduction

The work described in this paper was carried out within the 6NET [1] and Euro6IX [4] projects, featuring partners that included Southampton, UNINETT, RENATER, PSNC, Strasbourg (ULP), GRNET (from 6NET) and UMU, UPM, Consulintel, T-Systems, TILAB (from Euro6IX).

Source-Specific Multicast (SSM) is defined within the Internet Engineering Task Force (IETF). Some of the key IETF documents are the SSM Architecture [7], IPv6 Multicast Deployment Issues [10], an Overview of SSM [13], Socket Interface Extensions for Multicast Source Filters [14], Unicast-prefix-based IPv6 Multicast Addresses [6] and Using IGMPv3 and MLDv2 For SSM [8].

2 The SSM Model

The traditional multicast service model allows for applications to join a group (G) in order to receive any data sent to the group. A source simply sends IP datagrams to the multicast IP address of G. An application does not need to know which sources are available. This model is called any-source multicast (ASM).

In contrast, the Source-Specific Multicast (SSM) service model is based on applications joining so-called channels rather than groups. A channel is a pair (S,G) consisting of a unicast source address S and a multicast group address G. For each source S and group G the application is to receive data from, it must join the channel (S,G). This means that the application explicitly specifies the sources, and hence it must know the source addresses.

2.1 Routing

In IPv6 the reserved address range for SSM is FF3x::/32 [6]. The forwarding of multicast packets in itself is no different for SSM than for ASM. The routing is different though. The most popular multicast routing protocol is PIM-SM. When a host joins a multicast group, a multicast tree (a so called shared tree) is built from a router in the network configured to be a Rendezvous Point (RP) towards the receiver. When a source starts sending to the group, the packets are sent from a router on the source's local network. Initially the multicast packets are actually encapsulated into unicast packets and sent as so called PIM register messages from the router next to the source to the RP.

In this way the packets are forwarded from the source to the receivers via the RP. However, once the multicast packets reach the last-hop router (the last router before a receiving host), the last-hop router can shortcut this by building so-called shortest-path trees towards the sources. The last-hop router learns the source addresses by looking at the source addresses of the multicast packets.

One problem is the routers knowing where the RP is for a group. One solution is to use BSR for RP-to-group mappings, but this is very hard to do throughout the Internet. For IPv4 there is also MSDP which allows for different parts of the Internet to use different RPs for the same group, by exchanging source information between RPs. MSDP is not available for IPv6, and there are no plans to make it so. Instead there is a solution (developed in 6NET) that embeds RP-addresses into multicast addresses [12].

PIM-SM can also provide for the SSM service. When a host joins a channel (S,G) the last-hop router immediately

knows the source address S, so without waiting for multicast packets, it can immediately start building the shortest-path tree towards the source. In order to provide SSM service only, no RPs are necessary. This basically removes all the complexities. There is no need for BSR, MSDP, etc., and no shared trees, no PIM register with encapsulation and decapsulation. This means that an SSM-only network is much easier to manage. When using PIM-SM, SSM and ASM can co-exist in the same network.

2.2 Communication between routers and hosts

For IPv4 and IPv6, there are protocols IGMP and MLD respectively that are used between routers and hosts for signalling which groups the host wants to receive data for. The latest versions of these protocols, IGMPv3 and MLDv2 [15] also let the host specify specific sources. In addition to SSM, they also allow a host to join a group as in ASM, but blocking certain sources.

For just sending SSM multicast the above protocols are not needed. A host does not in any way need to join and receive multicast in order to send.

2.3 RTCP with SSM

Many ASM applications use RTP together with RTCP feedback from receivers. The RTCP feedback is sent to the group so that all receivers learn of each other. Both the identity of other receivers and information on reception quality. Also how much bandwidth a receiver uses to send reports is calculated based on the total bandwidth used for reports from other receivers. The more receivers there are, the less bandwidth should each one use. Doing this with SSM is difficult. A new IETF work [3] describes how one might do RTCP with SSM.

2.4 Further Comparison with ASM

SSM gives some protection against rogue sources sending to a multicast group. A content provider can announce it's channels (S1,G), ..., (Sn,G) and an application joining those channels, will not receive packets sent to G with other source addresses than S1, ..., Sn. Also note that due to the so-called RPF check used in multicast distribution, there is protection against source address spoofing.

Since one joins channels rather than groups, it's not a problem if two different sources use the same group for two different sessions. One only needs to ensure uniqueness of the group used within the host.

All the advantages of SSM follow from the fact that the application explicitly specifies the source addresses. This is also the main problem with SSM. There are several applications where it is difficult to know in advance what sources should be joined.

2.5 Source Discovery with SSM

For the traditional multicast, ASM, where one simply joins a group, one has SAP (Session Announcement Protocol) for announcing available multicast sessions. For global scope one has one specific group, one for IPv4 and one for IPv6, for announcements. Thus an application can join this group, and see what global sessions are available.

In the absence of an ASM service one should try to come up with some replacement for SAP. A naive way of doing SAP with SSM is to have a single (S,G) instead of a single group G, and unicasting announcements to S which then resends them as multicast. Such an approach would not scale though. What one usually does for SSM is to simply list the session information on a web page, e.g. as SDP. Clearly a more elegant solution is needed.

2.5.1 SSM Source Discovery Protocol (SSMSDP)

Strasbourg (ULP) proposed and implemented a solution called SSMSDP which stands for Source Specific Multicast Source Discovery Protocol. This solution aims at being as general as possible, to provide an easy transition for applications based on the Any Source Multicast service model.

Each emulated ASM group is identified by a control channel (C, G) where C is the address of a controller. The controller can be implemented in any host, for example on the host of the group creator. The control channel is announced by exterior means, in the same way as an ASM group address. Hosts wanting to participate in the emulated ASM group join the control channel. A source Si regularly unicasts a source announcement (Si, Gi) to C. C regularly sends list of active sources on channel (C,G). Receivers receive source announcements on (C,G) and join each (Si,Gi).

2.5.2 SMSDPifier

The SSMSDP protocol implementation is based on a library of functions which immediately return, and execute the protocol in the background. Knowing that, it is easy to derive an application catching the old ASM setsockopt joining/leaving primitives and replacing them with SSMSDP setsockopt primitives. This enables the possibility to port any ASM applications on top of the SSM service without re-compiling or patching.

3 The SSM API

RFC3678 [14] specifies an API for specifying multicast source filters. This API is used by SSM applications to tell the operating system which channels (S,G) they want to join. It specifies both an IPv4 specific API and a protocol independent one. We suggest that one always uses the

protocol independent one. It has both a basic API that allows one to (for example) add new sources one at the time, and an advanced API where one can specify the entire list of sources with just one call. In addition to SSM mode where one joins channels (S,G), it also supports ASM mode. One then joins a group G, and can optionally block certain sources one doesn't want to receive from. The RFC also has an appendix on an ioctl() based API, but one should avoid that; it is only included for historical purposes.

4 SSM in Hosts and Routers

There is now quite wide support for SSM (MLDv2) in hosts and routers. For routers, these include BSD, Cisco (Cisco 7200 routers as of at least 12.3(4)T), 6WINDGate (SixOS 6121 version 6.6.1), Hitachi (Ver.07-03).

For hosts, Linux and BSD have support. For Linux, the 2.6.8 kernel and beyond have support; earlier versions can be used only if the multicast interface is explicitly specified in (S,G) joins. Since MLDv2 is needed only for routers and receivers any recent IPv6 enabled Linux should be suitable for transmitting SSM packets.

Unfortunately GNU libc, the C standard library commonly used with Linux, does not yet include the critical structure definitions and constants needed to compile an SSM application written to the recommended version neutral APIs. This makes development of SSM applications on Linux slightly more difficult.

There is as yet no MLDv2 support in Windows, or thus on most PDA platforms. A new Solaris 10 beta will soon add MLDv2 support (at the time of writing).

5 Applications

There are currently very few SSM-enabled IPv6 applications; we focused on MAD-FLUTE, and also used DVTS, a high quality video application (not reported in this paper).

5.1 MAD-FLUTE

MAD/TUT is a project [9] at Tampere University of Technology in which a team has been implementing protocols for reliable content delivery over multicast networks. The new protocol introduced in MAD-FLUTE is "FLUTE - File Delivery over Unidirectional Transport" [11]. In these protocols reliable delivery is ensured despite the lack of any back channel by the use of forward error correction (FEC).

The MAD-FLUTE implementation is a piece of portable software written in C and distributed under the GNU General Public License. It supports both IPv4 and IPv6 operation and works on a many operating systems including Microsoft Windows and a variety of Unix-like systems. It streams one or more files from a sender to either a single receiver over unicast UDP or to any number of receivers using multicast UDP. A FLUTE sender can specify an XML catalogue of files to be sent, or use a simple wildcard expression. The transfer rate and error correction can be adjusted to suit the network environment in which it is used. The latest versions of this software also implement a new Internet Draft "SDP Descriptors for FLUTE" [5] for session discovery.

6 Porting MAD-FLUTE to IPv6 SSM

The University of Southampton undertook the work of adding IPv6 SSM support to MAD-FLUTE on Linux. The protocol independent API was used. MAD-FLUTE itself is modular, and already had IPv4 SSM support so only a single module was altered at first, the library which implements ALC. Only approximately 100 lines of code were written or changed at this point.

As mentioned above, the GNU C library provided with Linux does not include the structures and data constants described in RFC 3678. Similar structures are defined by the Linux kernel, but it would not be wise to try to use these from an ordinary user space application like MAD-FLUTE. These internal structures are subject to change at any time without notice by the kernel developers, meaning that on future Linux systems the software could malfunction or fail to compile at all.

On the other hand lifting the definitions directly from the RFC could potentially lead to problems with size, ordering and layout of structures. In fact this was tried, and some critical parameters were re-ordered in the group membership API, resulting in a non-working application.

After experimentation a compromise was reached in which a C header file was written which contains a subset of the definitions from RFC 3678 (just enough to properly implement MAD-FLUTE's requirements) carefully edited to match the current Linux kernel ordering and layout rules. Assuming that the GNU C library eventually adopts either the original kernel structures (thereby blessing them as a userspace API) or takes a similar path to the one used here and invents its own equivalent structures with the same order and layout this solution will remain source and binary compatible indefinitely.

The port, together with some minor bug fixes, were contributed to MAD/TUT and were integrated into the official distribution starting with MAD-FLUTE 1.0.

7 Application testing

The MAD-FLUTE application was tested successfully across the 6NET core network (with Cisco GSRs) between partners. In Euro6IX, pan-network deployment is still ongoing.

7.1 MAD-FLUTE

UNINETT has a mirror of all the IETF's Internet drafts. Every night FTP is used to synchronise UNINETT's mirror with the IETF site. After this is done, an XML file is created listing all the new drafts. This XML file is used by the MAD FLUTE application, describing the files to be streamed.

At 11 every morning Central European time (CET/CEST) FLUTE streams the new files:

```
flute -S -a:IP6 -T:40 -m:ff3e::d3af:1
-s:2001:700:e000:0:204:75ff:fee4:423b
-X:100 -x:1 -l:1280 -f:/tmp/fdt.xml
```

The arguments are as follows: "-S" act as source; "-a" use IPv6; "-T" hop limit 40; "m" send to group ff3e::d3af:1; "-s" specifies the source address to use; "-X" sets FEC ratio, 100% means sending twice the amount of data, allowing 50% of the packets to be lost; "-x" sets the encoding to Reed Solomon; "-l" sets the length to 1280 to make sure there is no fragmentation; "-f" specifies the XML file describing the data set. The FEC ratio can be set much smaller, usually the amount of packet loss is very small.

At the receiving side, one should use something like:

```
flute -A -a:IP6 -M -m:ff3e::d3af:1
-s:2001:700:e000:0:204:75ff:fee4:423b
```

The arguments are as follows: "-A" for receive automatically; "a" use IPv6; "-M" for SSM; "-m" for the group; "-s" for the source.

The receiver will when started, run until it receives a transmission and then stop. By starting the receiver every morning before the transmission starts, or simply running it in a loop; one will get all the new files, and one can maintain a mirror of the IETF site. The only issue is that there is no way to remove files that are deleted at the IETF site.

For testing there is also a transmission to the group ff3e::d3af:2 which is repeated continuously until 10am the next morning, when it starts over with new files. This is useful because at any time people can test that SSM and their FLUTE application works.

8 Problems encountered

Many hurdles were overcome in this work, e.g.:

- Ensuring all SSM systems use the IANA-approved ICMP value of 143 (and not 206).
- Adjusting the MTU accordingly for tunnelled links.
- Mis-handling of hop-by-hop headers (required in MLDv2) by the pf firewall, as of FreeBSD 5.2.

9 Conclusions and Further Work

The results of using IPv6 SSM are encouraging, and further details are now written up[2]. However, there are a number of open areas for study and effort, including:

- New source discovery methods
- Developing SSM interactive applications
- Wider OS support for MLDv2 (WinXP, PDA, ...)
- Wider application support
- Support for MLD switch snooping
- Analysis of pros and cons for SSM/ASM

Southampton plans an advanced MP3 jukebox application based around MAD-FLUTE distribution. Other applications proposed include software distribution mirrors, and distribution of presentations for conferences. ULP plans to continue to test the MLDv2 API, but under Linux, and to further develop mflute, an application that builds upon the FLUTE protocol in a similar way to MAD.

References

[1] 6NET Project. http://www.6net.org/.

[2] 6NET Project Deliverable 5.9: IPv6 and SSM Cookbook, November 2004. http://www.6net.org/publications/deliverables/D5.9.pdf.

[3] J. Chesterfield. RTCP Extensions for Single-Source Multicast Sessions with Unicast Feedback. IETF Internet Draft (work in progress), October 2004. draft-ietf-avt-rtcpssm-07.

[4] Euro6IX Project. http://www.euro6ix.org/.

[5] SDP Descriptors for FLUTE. IETF Internet Draft (work in progress), September 2004. draft-mehta-rmt-flute-sdp-01.

[6] B. Haberman. Unicast-prefix-based IPv6 Multicast Addresses. IETF Standard, August 2002. RFC3306.

[7] H. Holbrook. Source-Specific Multicast for IP. IETF Internet Draft (work in progress), September 2004. draft-ietf-ssm-arch-06.

[8] H. Holbrook. Using IGMPv3 and MLDv2 for Source-Specific Multicast. IETF Internet Draft (work in progress), October 2004. draft-holbrook-idmr-igmpv3-ssm-08.

[9] MAD FLUTE Project. http://www.atm.tut.fi/mad/.

[10] IPv6 Multicast Deployment Issues. IETF Internet Draft (work in progress), September 2004. draft-ietf-mboned-ipv6-multicast-issues-01.

[11] T. Paila. FLUTE - File Delivery over Unidirectional Transport. IETF Standard, October 2004. RFC3926.

[12] P. Savola. Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address. IETF Internet Draft (work in progress), July 2004. draft-ietf-mboned-embeddedrp-07.

[13] An Overview of Source-Specific Multicast (SSM). IETF Standard, July 2003. RFC3569.

[14] D. Thaler. Socket Interface Extensions for Multicast Source Filters. IETF Standard, January 2004. RFC3678.

[15] R. Vida. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. IETF Standard, June 2004. RFC3810.