MLDv2 Protocol Design,
Implementation and Evaluation
for Source-Specific Multicast
over IPv6

Hitoshi Asaeda, INRIA
Shinsuke Suzuki, Hitachi Ltd.
Today’s Presentation

- Source-Specific Multicast (SSM)
- Multicast Listener Discovery version 2 (MLDv2) specification
- Host-side kernel implementation of MLDv2
  - MSF implementation and its evaluation
Multicast Communication Deployment

- **PIM-SM/MSDP/MBGP**
  - Current Inter-domain multicast routing protocols for IPv4
  - Any-Source Multicast (ASM) support (aka (*,G) join/leave)

- **Scalability**
  - 3rd-party dependency problem
  - Traffic concentration problem
  - Flood MSDP Source Active (SA) message

- **Complexity**
  - Manage both RPT and SPT
  - Need MSDP peer RPF check
Motivation

- If we can construct/maintain only SPT, multicast routing tree becomes pretty simple!
Source-Specific Multicast (SSM)

- If we consider inter-domain multicast service, ...
  - One-to-many or few-to-many communication model is feasible
  - Each source address can be recognized beforehand
- If each source address is known by each multicast listener, ...
  - SPT can be constructed directly
    - No need to create and maintain RPT
    - RP and MSDP are not required by routing protocols
    - ASM’s complexity and scalability problems are eliminated

Source-Specific Multicast (SSM) is the solution
SSM Deployment

- **Host-side implementation**
  - **Purpose**
    - Specification of interesting source address(es) as well as multicast address (aka (S,G) join/leave)
    - Summarizing and reporting (S,G) information
  - **Kernel implementation**
    - IGMPv3 for IPv4 and MLDv2 for IPv6
    - MSF APIs
  - **Application implementation**
    - MSF APIs

- **Router-side implementation**
  - **Purpose**
    - Translation of reported (S,G) information
    - Constructing SPT from initial phase
  - **Routing protocol implementation**
    - IGMPv3 for IPv4 and MLDv2 for IPv6
      (Currently, PIM-SSM is only the routing protocol to support SSM architecture)
MLDv2 Specification

- MLDv2 for IPv6
  - draft-vida-mld-v2-06.txt

- Main functions of MLDv2
  - Source address filtering
  - New API -
    - IPMulticastListen(socket, interface, multicast-address, filter-mode, source-list)
  - New type of Query and Report message
  - Robustness variable
  - Version compatibility mode
  - New destination address of Report message
  - No Report message surpression mechanism
  - etc.
Source Address Filtering

- Filter-mode is either "INCLUDE" or "EXCLUDE"
  - INCLUDE indicates that reception of packets sent to the multicast address is requested only from the specified source addresses.
  - EXCLUDE indicates that reception of packets sent to the multicast address is requested from all source addresses except the specified source addresses.

- Socket state
  - (i, m, filter-mode, source-list)

- Interface state
  - (m, filter-mode, source-list)
  - Initial interface state is (null, INCLUDE, null)
Interface State Transition

- Interface state is calculated as below:
  - If all sockets request a filter-mode of INCLUDE, then interface state is INCLUDE with the union source lists.
  - If any sockets request a filter-mode of EXCLUDE, then interface state is EXCLUDE with the intersection of all EXCLUDE source lists subtracting the union of all INCLUDE source lists.

- Action on change of interface state:

<table>
<thead>
<tr>
<th>Old State</th>
<th>New State</th>
<th>State-Change Record Sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCLUDE(A)</td>
<td>INCLUDE(B)</td>
<td>ALLOW(B-A), BLOCK(A-B)</td>
</tr>
<tr>
<td>EXCLUDE(A)</td>
<td>EXCLUDE(B)</td>
<td>ALLOW(A-B), BLOCK(B-A)</td>
</tr>
<tr>
<td>INCLUDE(A)</td>
<td>EXCLUDE(B)</td>
<td>TO_EX(B)</td>
</tr>
<tr>
<td>EXCLUDE(A)</td>
<td>INCLUDE(B)</td>
<td>TO_IN(B)</td>
</tr>
</tbody>
</table>
Multicast Source Filter (MSF) APIs

- Socket Interface Extensions for Multicast Source Filters
  - draft-ietf-magma-msf-api-03.txt

- Used to change a socket state

- Basic API
  - Used with setsockopt()
  - Used to join/leave a single channel

- Advanced API
  - Used with ioctl()
  - Used to join/leave a single or multiple channel(s), e.g., ((S1,S2,S3),G1)
  - Used to change a filter-mode of socket state without leaving joined channel
Basic API Usage

A part of a sample application code

```c
bcopy(&grp, &gsr.gsr_group, grp.sin6_len);
bcopy(&src, &gsr.gsr_source, src.sin6_len);

if (setsockopt(socket, IPPROTO_IPV6, MCAST_JOIN_SOURCE_GROUP,
               (char *)&gsr, sizeof(gsr)) < 0)
    perror("cannot listen group");
```
Advanced API Usage

- A part of a sample application code

```c
if ((gf = malloc(GROUP_FILTER_SIZE(numsrc))) == NULL) 
    perror("memory allocation error");

bzero(gf, GROUP_FILTER_SIZE(numsrc));
gf->gf_interface = index;
gf->gf_fmode = mode;
gf->gf_numssrc = numsrc;
bcopy(&grp, &gf->gf_group, grp.sin6_len);
for (i = 0; i < numsrc; i++)
    bcopy(&src[i], &gf->gf_slist[i], src[i].sin6_len);

if (ioctl(socket, SIOCSMSFILTER, gf) != 0) 
    perror("cannot listen group");
```
MSF Kernel Implementation

- Socket state migration and interface state transition
- State-Change report transmission
Source Address Comparison

- **Linear search algorithm**
  - **Advantage**
    - Easy to understand the logic
    - Easy to maintain the code
  - **Disadvantage**
    - May cause low performance if the number of source addresses is highly increased
      - It is quite rare that so many multicast applications use a same multicast address for each different channel.
        - e.g., (S1,G1), (S2,G1), (S3,G1), ... (S1000,G1), ...
Implementation Evaluations

- Evaluation on NetBSD-current (1.6l)
  - 1GHz Pentium III PC with 512MB memory

- Conditions are ...
  - The number of sampling data is 100 for each request
  - Multicast address - fixed
  - Source address - randomly created
Average Response Time of Basic API

- E.g., Request to change INCLUDE with null source address to INCLUDE with one source address
- Response time is proportional to the number of source addresses of the list

<table>
<thead>
<tr>
<th>Request</th>
<th>Average (micro sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN(0) -&gt; EX(0)</td>
<td>965</td>
</tr>
<tr>
<td>IN(0) -&gt; IN(1)</td>
<td>1381</td>
</tr>
<tr>
<td>IN(1) -&gt; IN(2)</td>
<td>1247</td>
</tr>
<tr>
<td>IN(2) -&gt; IN(3)</td>
<td>1291</td>
</tr>
</tbody>
</table>
Average Response Time of Advanced API

- Advanced API requires complex implementation
  - Need source address validation/duplication check before merging source address list etc.
  - First entry shows an initial INCLUDE join request in which five source addresses were specified.
  - Second entry and third entry are for the same request, but third one uses Basic API

<table>
<thead>
<tr>
<th>Request</th>
<th>Average (micro sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN(0) -&gt; IN(5)</td>
<td>2256</td>
</tr>
<tr>
<td>IN(5) -&gt; EX(1)</td>
<td>1637</td>
</tr>
<tr>
<td>IN(5) -&gt; EX(1)(*)</td>
<td>1512</td>
</tr>
</tbody>
</table>
Available MLDv2 Implementations

- **Kernel**
  - NetBSD-current
    - IGMPv3 implementation is prerequisite
      (Enabling IGMPv3 is NOT required.)
  - FreeBSD-4.7, NetBSD-1.6, OpenBSD-3.2
    - [http://www.kame.net](http://www.kame.net)

- **Router**
  - pim6sd
  - Hitachi GR2000
    - [http://www.internetworking.hitachi.com](http://www.internetworking.hitachi.com)
  - 6Wind
    - [http://www.6wind.com](http://www.6wind.com)

- **Applications and utilities**
  - mcastread
    - [http://www.kame.net](http://www.kame.net)
Thank you.