

SAM Research Group
Internet-Draft
Intended status: Informational
Expires: January 12, 2012

M. Waehlisich
link-lab & FU Berlin
T C. Schmidt
HAW Hamburg
S. Venaas
cisco Systems
July 11, 2011

A Common API for Transparent Hybrid Multicast
draft-irtf-samrg-common-api-02

Abstract

Group communication services exist in a large variety of flavors, and technical implementations at different protocol layers. Multicast data distribution is most efficiently performed on the lowest available layer, but a heterogeneous deployment status of multicast technologies throughout the Internet requires an adaptive service binding at runtime. Today, it is difficult to write an application that runs everywhere and at the same time makes use of the most efficient multicast service available in the network. Facing robustness requirements, developers are frequently forced to use a stable, upper layer protocol controlled by the application itself. This document describes a common multicast API that is suitable for transparent communication in underlay and overlay, and grants access to the different multicast flavors. It proposes an abstract naming by multicast URIs and discusses mapping mechanisms between different namespaces and distribution technologies. Additionally, it describes the application of this API for building gateways that interconnect current multicast domains throughout the Internet.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 12, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Use Cases for the Common API	5
2. Terminology	6
3. Overview	7
3.1. Objectives and Reference Scenarios	7
3.2. Group Communication API & Protocol Stack	8
3.3. Naming and Addressing	10
3.4. Mapping	11
4. Common Multicast API	12
4.1. Notation	12
4.2. Abstract Data Types	12
4.2.1. Multicast URI	12
4.2.2. Interface	13
4.2.3. Membership Events	13
4.3. Group Management Calls	14
4.3.1. Create	14
4.3.2. Delete	14
4.3.3. Join	14
4.3.4. Leave	15
4.3.5. Source Register	15
4.3.6. Source Deregister	15
4.4. Send and Receive Calls	16
4.4.1. Send	16
4.4.2. Receive	16
4.5. Socket Options	17
4.5.1. Get Interfaces	17
4.5.2. Add Interface	17
4.5.3. Delete Interface	17
4.5.4. Set TTL	18
4.5.5. Get TTL	18

4.6. Service Calls	18
4.6.1. Group Set	18
4.6.2. Neighbor Set	19
4.6.3. Children Set	19
4.6.4. Parent Set	20
4.6.5. Designated Host	20
4.6.6. Enable Membership Events	21
4.6.7. Disable Membership Events	21
5. Functional Details	21
5.1. Namespaces	22
6. IANA Considerations	22
7. Security Considerations	22
8. Acknowledgements	22
9. Informative References	23
Appendix A. C Signatures	24
Appendix B. Practical Example of the API	25
Appendix C. Deployment Use Cases for Hybrid Multicast	26
C.1. DVMRP	27
C.2. PIM-SM	27
C.3. PIM-SSM	28
C.4. BIDIR-PIM	28
Appendix D. Change Log	29
Authors' Addresses	31

1. Introduction

Currently, group application programmers need to make the choice of the distribution technology that the application will require at runtime. There is no common communication interface that abstracts multicast transmission and subscriptions from the deployment state at runtime. The standard multicast socket options [RFC3493], [RFC3678] are bound to an IP version and do not distinguish between naming and addressing of multicast identifiers. Group communication, however, is commonly implemented in different flavors such as any source (ASM) vs. source specific multicast (SSM), on different layers (e.g., IP vs. application layer multicast), and may be based on different technologies on the same tier as with IPv4 vs. IPv6. It is the objective of this document to provide a universal access to group services.

Multicast application development should be decoupled of technological deployment throughout the infrastructure. It requires a common multicast API that offers calls to transmit and receive multicast data independent of the supporting layer and the underlying technological details. For inter-technology transmissions, a consistent view on multicast states is needed, as well. This document describes an abstract group communication API and core functions necessary for transparent operations. Specific implementation guidelines with respect to operating systems or programming languages are out-of-scope of this document.

In contrast to the standard multicast socket interface, the API introduced in this document abstracts naming from addressing. Using a multicast address in the current socket API predefines the corresponding routing layer. In this specification, the multicast name used for joining a group denotes an application layer data stream that is identified by a multicast URI, independent of its binding to a specific distribution technology. Such a group name can be mapped to variable routing identifiers.

The aim of this common API is twofold:

- o Enable any application programmer to implement group-oriented data communication independent of the underlying delivery mechanisms. In particular, allow for a late binding of group applications to multicast technologies that makes applications efficient, but robust with respect to deployment aspects.
- o Allow for a flexible namespace support in group addressing, and thereby separate naming and addressing/routing schemes from the application design. This abstraction does not only decouple programs from specific aspects of underlying protocols, but may

open application design to extend to specifically flavored group services.

Multicast technologies may be of various P2P kinds, IPv4 or IPv6 network layer multicast, or implemented by some other application service. Corresponding namespaces may be IP addresses or DNS naming, overlay hashes, or other application layer group identifiers like <sip:*@peanuts.org> but also names independently defined by the applications. Common namespaces are introduced later in this document, but follow an open concept suitable for further extensions.

This document also proposes and discusses mapping mechanisms between different namespaces and forwarding technologies. Additionally, the multicast API provides internal interfaces to access current multicast states at the host. Multiple multicast protocols may run in parallel on a single host. These protocols may interact to provide a gateway function that bridges data between different domains. The application of this API at gateways operating between current multicast instances throughout the Internet is described, as well.

1.1. Use Cases for the Common API

Four generic use cases can be identified that require an abstract common API for multicast services:

Application Programming Independent of Technologies: Application programmers are provided with group primitives that remain independent of multicast technologies and their deployment in target domains. They are thus enabled to develop programs once that run in every deployment scenario. The employment of group names in the form of abstract meta data types allows applications to remain namespace-agnostic in the sense that the resolution of namespaces and name-to-address mappings may be delegated to a system service at runtime. Thereby, the complexity is minimized as developers need not care about how data is distributed in groups, while the system service can take advantage of extended information of the network environment as acquired at startup.

Global Identification of Groups: Groups can be identified independent of technological instantiations and beyond deployment domains. Taking advantage of the abstract naming, an application is thus enabled to match data received from different interface technologies (e.g., IPv4, IPv6, or overlays) to belong to the same group. This not only increases flexibility, an application may for instance combine heterogeneous multipath streams, but also simplifies the design and implementation of gateways and translators.

Simplified Service Deployment through Generic Gateways: The common multicast API allows for an implementation of abstract gateway functions with mappings to specific technologies residing at a system level. Such generic gateways may provide a simple bridging service and facilitate an inter-domain deployment of multicast.

Mobility-agnostic Group Communication: Group naming and management as foreseen in the common multicast API remain independent of locators. Naturally, applications stay unaware of any mobility-related address changes. Handover-initiated re-addressing is delegated to the mapping services at the system level and may be designed to smoothly interact with mobility management solutions provided at the network or transport layer.

2. Terminology

This document uses the terminology as defined for the multicast protocols [RFC2710],[RFC3376],[RFC3810],[RFC4601],[RFC4604]. In addition, the following terms will be used.

Group Address: A Group Address is a routing identifier. It represents a technological specifier and thus reflects the distribution technology in use. Multicast packet forwarding is based on this ID.

Group Name: A Group Name is an application identifier that is used by applications to manage communication in a multicast group (e.g., join/leave and send/receive). The Group Name does not predefine any distribution technologies, even if it syntactically corresponds to an address, but represents a logical identifier.

Multicast Namespace: A Multicast Namespace is a collection of designators (i.e., names or addresses) for groups that share a common syntax. Typical instances of namespaces are IPv4 or IPv6 multicast addresses, overlay group IDs, group names defined on the application layer (e.g., SIP or Email), or some human readable strings.

Interface: An Interface is a forwarding instance of a distribution technology on a given node. For example, the IP interface 192.168.1.1 at an IPv4 host.

Multicast Domain: A Multicast Domain hosts nodes and routers of a common, single multicast forwarding technology and is bound to a single namespace.

Inter-domain Multicast Gateway (IMG): An Inter-domain Multicast Gateway (IMG) is an entity that interconnects different Multicast Domains. Its objective is to forward data between these domains, e.g., between IP layer and overlay multicast.

3. Overview

3.1. Objectives and Reference Scenarios

The default use case addressed in this document targets at applications that participate in a group by using some common identifier taken from some common namespace. This Group Name is typically learned at runtime from user interaction like the selection of an IPTV channel, from dynamic session negotiations like in the Session Initiation Protocol (SIP), but may as well have been predefined for an application as a common Group Name. Technology-specific system functions then transparently map the Group Name to Group Addresses such that

- o programmers are enabled to process group names in their programs without the need to consider technological mappings to designated deployments in target domains;
- o applications are enabled to identify packets that belong to a logically named group, independent of the interface technology used for sending and receiving packets. The latter shall also hold for multicast gateways.

This document considers two reference scenarios that cover the following hybrid deployment cases displayed in Figure 1:

1. Multicast Domains running the same multicast technology but remaining isolated, possibly only connected by network layer unicast.
2. Multicast Domains running different multicast technologies but hosting nodes that are members of the same multicast group.

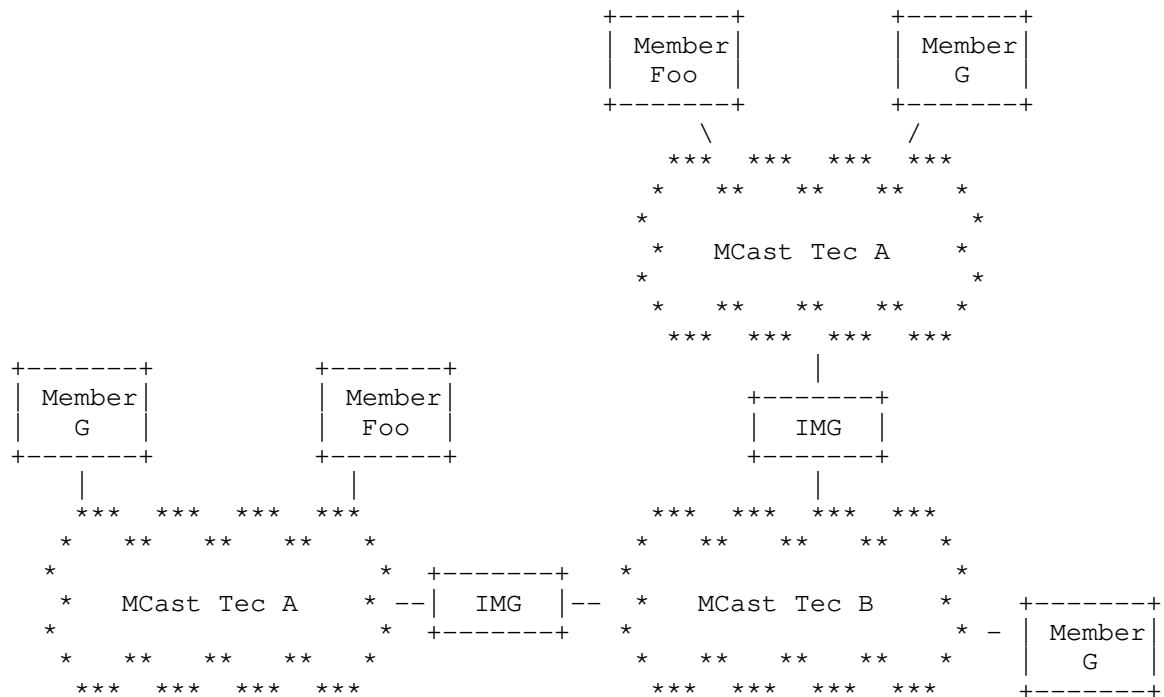


Figure 1: Reference scenarios for hybrid multicast, interconnecting group members from isolated homogeneous and heterogeneous domains.

It is assumed throughout the document that the domain composition, as well as the node attachment to a specific technology, remain unchanged during a multicast session.

3.2. Group Communication API & Protocol Stack

The group communication API consists of four parts. Two parts combine the essential communication functions, while the remaining two offer optional extensions for an enhanced monitoring and management:

Group Management Calls provide the minimal API to instantiate a multicast socket and to manage group membership.

Send/Receive Calls provide the minimal API to send and receive multicast data in a technology-transparent fashion.

Socket Options provide extension calls for an explicit configuration of the multicast socket such as setting hop limits or associated interfaces.

Service Calls provide extension calls that grant access to internal multicast states of an interface such as the multicast groups under subscription or the multicast forwarding information base.

Multicast applications that use the common API require assistance by a group communication stack. This protocol stack serves two needs:

- o It provides system-level support to transfer the abstract functions of the common API, including namespace support, into protocol operations at interfaces.
- o It bridges data distribution between different multicast technologies at the local host.

A general initiation of a multicast communication in this setting proceeds as follows:

1. An application opens an abstract multicast socket.
2. The application subscribes/leaves/(de)registers to a group using a Group Name.
3. An intrinsic function of the stack maps the logical group ID (Group Name) to a technical group ID (Group Address). This function may make use of deployment-specific knowledge such as available technologies and group address management in its domain.
4. Packet distribution proceeds to and from one or several multicast-enabled interfaces.

The multicast socket describes a group communication channel composed of one or multiple interfaces. A socket may be created without explicit interface association by the application, which leaves the choice of the underlying forwarding technology to the group communication stack. However, an application may also bind the socket to one or multiple dedicated interfaces, which predefines the forwarding technology and the namespace(s) of the Group Address(es).

Applications are not required to maintain mapping states for Group Addresses. The group communication stack accounts for the mapping of the Group Name to the Group Address(es) and vice versa. Multicast data passed to the application will be augmented by the corresponding Group Name. Multiple multicast subscriptions thus can be conducted

on a single multicast socket without the need for Group Name encoding at the application side.

Hosts may support several multicast protocols. The group communication stack discovers available multicast-enabled interfaces. It provides a minimal hybrid function that bridges data between different interfaces and Multicast Domains. Details of service discovery are out-of-scope of this document.

The extended multicast functions can be implemented by a middleware as conceptually visualized in Figure 2.

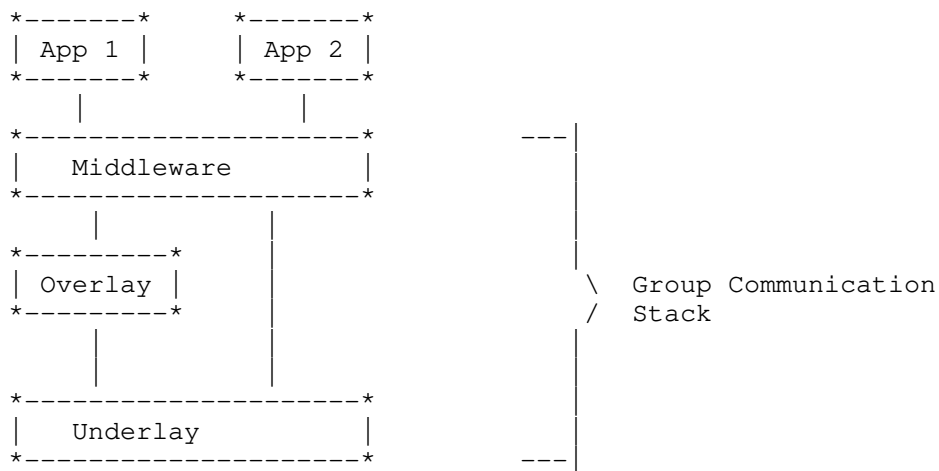


Figure 2: A middleware for offering uniform access to multicast in underlay and overlay

3.3. Naming and Addressing

Applications use Group Names to identify groups. Names can uniquely determine a group in a global communication context and hide technological deployment for data distribution from the application. In contrast, multicast forwarding operates on Group Addresses. Even though both identifiers may be identical in symbols, they carry different meanings. They may also belong to different namespaces. The namespace of a Group Address reflects a routing technology, while the namespace of a Group Name represents the context in which the application operates.

URIs [RFC3986] are a common way to represent namespace-specific identifiers in applications in the form of an abstract meta-data type. Throughout this document, any kind of Group Name follows a URI notation with the syntax defined in Section 4.2.1. Examples are,

ip://224.1.2.3:5000 for a canonical IPv4 ASM group,
sip://news@cnn.com for an application-specific naming with service
instantiator and default port selection.

An implementation of the group communication middleware can provide convenience functions that detect the namespace of a Group Name and use it to optimize service instantiation. In practice, such a library would provide support for high-level data types to the application, similar to the current socket API (e.g., `InetAddress` in Java). Using this data type could implicitly determine the namespace. Details of automatic namespace identification is out-of-scope of this document.

3.4. Mapping

All group members subscribe to the same Group Name taken from a common namespace and thereby identify the group in a technology-agnostic way.

Group Names require a mapping to addresses prior to service instantiation at an Interface. Similarly, a mapping is needed at gateways to translate between Group Addresses from different namespaces. Some namespaces facilitate a canonical transformation to default address spaces. For example, ip://224.1.2.3:5000 has an obvious correspondence to 224.1.2.3 in the IPv4 multicast address space. Note that in this example the multicast URI can be completely recovered from any data packet received from this group.

However, mapping in general can be more complex and need not be invertible. Mapping functions can be stateless in some contexts, but may require states in others. The application of such functions depends on the cardinality of the namespaces, the structure of address spaces, and possible address collisions. For example, it is not obvious how to map a large identifier space (e.g., IPv6) to a smaller, collision-prone set like IPv4.

Two (or more) Multicast Addresses from different namespaces may belong to

- a. the same logical group (i.e., same Group Name)
- b. different multicast channels (i.e., different Group Addresses).

A mapping can be realized by embedding smaller in larger namespaces or selecting an arbitrary, unused ID in the target space. The relation between logical and technical ID is maintained by mapping functions which can be stateless or stateful. The middleware thus queries the mapping service first, and creates a new technical group

ID only if there is no identifier available for the namespace in use. The Group Name is associated with one or more Group Addresses, which belong to different namespaces. Depending on the scope of the mapping service, it ensures a consistent use of the technical ID in a local or global domain.

4. Common Multicast API

4.1. Notation

The following description of the common multicast API is described in pseudo syntax. Variables that are passed to function calls are declared by "in", return values are declared by "out". A list of elements is denoted by <>. The pseudo syntax assumes that lists include an attribute which represents the number of elements.

The corresponding C signatures are defined in Appendix A.

4.2. Abstract Data Types

4.2.1. Multicast URI

Multicast Names and Multicast Addresses used in this API follow an URI scheme that defines a subset of the generic URI specified in [RFC3986] and is compliant with the guidelines in [RFC4395].

The multicast URI is defined as follows:

```
scheme "://" group "@" instantiation ":" port "/" sec-credentials
```

The parts of the URI are defined as follows:

scheme refers to the specification of the assigned identifier [RFC3986] which takes the role of the namespace.

group identifies the group uniquely within the namespace given in scheme.

instantiation identifies the entity that generates the instance of the group (e.g., a SIP domain or a source in SSM) using the namespace given in scheme.

port identifies a specific application at an instance of a group.

sec-credentials used to implement security credentials (e.g., to authorize a multicast group access).

4.2.2. Interface

The interface denotes the layer and instance on which the corresponding call will be effective. In agreement with [RFC3493] we identify an interface by an identifier, which is a positive integer starting at 1.

Properties of an interface are stored in the following struct:

```
struct if_prop {
    unsigned int if_index; /* 1, 2, ... */
    char        *if_name;  /* "eth0", "eth1:1", "lo", ... */
    char        *if_addr;  /* "1.2.3.4", "abc123", ... */
    char        *if_tech;  /* "ip", "overlay", ... */
};
```

The following function retrieves all available interfaces from the system:

```
getInterfaces(out Interface <ifs>);
```

It extends the functions for Interface Identification in [RFC3493] (cf., Section 4) and can be implemented by:

```
struct if_prop *if_prop(void);
```

4.2.3. Membership Events

A membership event is triggered by a multicast state change, which is observed by the current node. It is related to a specific Group Name and may be receiver or source oriented.

```
event_type {
    join_event;
    leave_event;
    new_source_event;
};

event {
    event_type event;
    Uri group_name;
    Interface if;
};
```

An event will be created by the middleware and passed to applications

that are registered for events.

4.3. Group Management Calls

4.3.1. Create

The create call initiates a multicast socket and provides the application programmer with a corresponding handle. If no interfaces will be assigned based on the call, the default interface will be selected and associated with the socket. The call may return an error code in the case of failures, e.g., due to a non-operational middleware.

```
createMSocket(in Interface <ifs>,  
              out Socket s);
```

The if argument denotes a list of interfaces (if_indexes) that will be associated with the multicast socket. This parameter is optional.

On success a multicast socket identifier is returned, otherwise NULL.

4.3.2. Delete

The delete call removes the multicast socket.

```
deleteMSocket(in Socket s, out Int error);
```

The s argument identifies the multicast socket for destruction.

On success the out parameter error is 0, otherwise -1.

4.3.3. Join

The join call initiates a subscription for the given group. Depending on the interfaces that are associated with the socket, this may result in an IGMP/MLD report or overlay subscription, for example.

```
join(in Socket s, in Uri group_name, out Int error);
```

The s argument identifies the multicast socket.

The group_name argument identifies the group.

On success the out parameter error is 0, otherwise -1.

4.3.4. Leave

The leave call results in an unsubscription for the given Group Name.

```
leave(in Socket s, in Uri group_name, out Int error);
```

The s argument identifies the multicast socket.

The group_name identifies the group.

On success the out parameter error is 0, otherwise -1.

4.3.5. Source Register

The srcRegister call registers a source for a Group on all active interfaces of the socket s. This call may assist group distribution in some technologies, the creation of sub-overlays, for example. Not all multicast technologies require this call.

```
srcRegister(in Socket s, in Uri group_name,  
            in Interface <ifs>, out Int error);
```

The s argument identifies the multicast socket.

The group_name argument identifies the multicast group to which a source intends to send data.

The ifs argument points to the list of interface indexes for which the source registration failed. A NULL pointer is returned, if the list is empty. This parameter is optional.

If source registration succeeded for all interfaces associated with the socket, the out parameter error is 0, otherwise -1.

4.3.6. Source Deregister

The srcDeregister indicates that a source does no longer intend to send data to the multicast group. This call may remain without effect in some multicast technologies.

```
srcDeregister(in Socket s, in Uri group_name,  
              in Interface <ifs>, out Int error);
```

The s argument identifies the multicast socket.

The group_name argument identifies the multicast group to which a source has stopped to send multicast data.

The `ifs` argument points to the list of interfaces for which the source deregistration failed. A NULL pointer is returned, if the list is empty.

If source deregistration succeeded for all interfaces associated with the socket, the out parameter error is 0, otherwise -1.

4.4. Send and Receive Calls

4.4.1. Send

The `send` call passes multicast data for a Multicast Name from the application to the multicast socket.

```
send(in Socket s, in Uri group_name,  
     in Size msg_len, in Msg msg_buf,  
     out Int error);
```

The `s` argument identifies the multicast socket.

The `group_name` argument identifies the group to which data will be sent.

The `msg_len` argument holds the length of the message to be sent.

The `msg_buf` argument passes the multicast data to the multicast socket.

On success the out parameter error is 0, otherwise -1.

4.4.2. Receive

The `receive` call passes multicast data and the corresponding Group Name to the application.

```
receive(in Socket s, out Uri group_name,  
        out Size msg_len, out Msg msg_buf,  
        out Int error);
```

The `s` argument identifies the multicast socket.

The `group_name` argument identifies the multicast group for which data was received.

The `msg_len` argument holds the length of the received message.

The `msg_buf` argument points to the payload of the received multicast data.

On success the out parameter error is 0, otherwise -1.

4.5. Socket Options

The following calls configure an existing multicast socket.

4.5.1. Get Interfaces

The `getInterface` call returns an array of all available multicast communication interfaces associated with the multicast socket.

```
getInterfaces(in Socket s,  
              out Interface <ifs>, out Int error);
```

The `s` argument identifies the multicast socket.

The `ifs` argument points to an array of interface index identifiers.

On success the out parameter error is 0, otherwise -1.

4.5.2. Add Interface

The `addInterface` call adds a distribution channel to the socket. This may be an overlay or underlay interface, e.g., IPv6 or DHT. Multiple interfaces of the same technology may be associated with the socket.

```
addInterface(in Socket s, in Interface if,  
              out Int error);
```

The `s` and `if` arguments identify a multicast socket and interface, respectively.

On success the value 0 is returned, otherwise -1.

4.5.3. Delete Interface

The `delInterface` call removes the interface `if` from the multicast socket.

```
delInterface(in Socket s, Interface if,  
              out Int error);
```

The `s` and `if` arguments identify a multicast socket and interface, respectively.

On success the out parameter error is 0, otherwise -1.

4.5.4. Set TTL

The `setTTL` call configures the maximum hop count for the socket a multicast message is allowed to traverse.

```
setTTL(in Socket s, in Int h,  
       in Interface <ifs>,  
       out Int error);
```

The `s` and `h` arguments identify a multicast socket and the maximum hop count, respectively.

The `ifs` argument points to an array of interface index identifiers. This parameter is optional.

On success the out parameter `error` is 0, otherwise -1.

4.5.5. Get TTL

The `getTTL` call returns the maximum hop count a multicast message is allowed to traverse for the socket.

```
getTTL(in Socket s,  
       out Int h, out Int error);
```

The `s` argument identifies a multicast socket.

The `h` argument holds the maximum number of hops associated with socket `s`.

On success the out parameter `error` is 0, otherwise -1.

4.6. Service Calls

4.6.1. Group Set

The `groupSet` call returns all multicast groups registered at a given interface. This information can be provided by group management states or routing protocols. The return values distinguish between sender and listener states.

```
struct GroupSet {
    uri group_name; /* registered multicast group */
    int type;        /* 0 = listener state, 1 = sender state,
                     2 = sender & listener state */
}

groupSet(in Interface if, out Int num_groups,
         out GroupSet <groupSet>, out Int error);
```

The if argument identifies the interface for which states are maintained.

The num_groups argument holds the number of groups in the groupSet array.

The groupSet argument points to a list of group states.

On success the out parameter error is 0, otherwise -1.

4.6.2. Neighbor Set

The neighborSet function returns the set of neighboring nodes for a given interface as seen by the multicast routing protocol.

```
neighborSet(in Interface if, out Int num_neighbors,
            out Uri <neighbor_address>, out Int error);
```

The if argument identifies the interface for which neighbors are inquired.

The num_neighbors argument holds the number of addresses in the neighbor_address array.

The neighbor_address argument points to a list of neighboring nodes on a successful return.

On success the out parameter error is 0, otherwise -1.

4.6.3. Children Set

The childrenSet function returns the set of child nodes that receive multicast data from a specified interface for a given group. For a common multicast router, this call retrieves the multicast forwarding information base per interface.

```
childrenSet(in Interface if, in Uri group_name,  
            out Int num_children, out Uri <child_address>,  
            out Int error);
```

The if argument identifies the interface for which children are inquired.

The group_name argument defines the multicast group for which distribution is considered.

The num_children argument holds the number of addresses in the child_address array.

The child_address argument points to a list of neighboring nodes on a successful return.

On success the out parameter error is 0, otherwise -1.

4.6.4. Parent Set

The parentSet function returns the set of neighbors from which the current node receives multicast data at a given interface for the specified group.

```
parentSet(in Interface if, in Uri group_name,  
          out Int num_parents, out Uri parent_address,  
          out Int error);
```

The if argument identifies the interface for which parents are inquired.

The group_name argument defines the multicast group for which distribution is considered.

The num_parents argument holds the number of addresses in the parent_address array.

The parent_address argument points to a list of neighboring nodes on a successful return.

On success the out parameter error is 0, otherwise -1.

4.6.5. Designated Host

The designatedHost function inquires whether the host has the role of a designated forwarder resp. querier, or not. Such an information is provided by almost all multicast protocols to prevent packet duplication, if multiple multicast instances serve on the same

subnet.

```
    designatedHost(in Interface if, in Uri group_name  
                  out Int return);
```

The if argument identifies the interface for which designated forwarding is inquired.

The group_name argument specifies the group for which the host may attain the role of designated forwarder.

The function returns 1 if the host is a designated forwarder or querier, otherwise 0. The return value -1 indicates an error.

4.6.6. Enable Membership Events

The enableEvents function registers an application at the middleware to inform the application about a group change. This is the result of receiver new subscriptions or leaves as well as the observation of source changes. The group service may call other service calls to get additional information.

```
    enableEvents();
```

Calling this function, the middleware starts to pass membership events to the application. Each event includes an event type identifier and a Group Name (cf., Section 4.2.3).

4.6.7. Disable Membership Events

The disableEvents function deactivates the information about group state changes.

```
    disableEvents();
```

On success the middleware will not pass membership events to the application.

5. Functional Details

In this section, we describe specific functions of the API and the associated system middleware in detail.

5.1. Namespaces

Namespace identifiers in URIs are placed in the scheme element and characterize syntax and semantic of the group identifier. They enable the use of convenience functions and high-level data types while processing URIs. When used in names, they may facilitate a default mapping and a recovery of names from addresses. They characterize its type, when used in addresses.

Compliant to the URI concept, namespace-schemes can be added. Examples of schemes and functions currently foreseen include

IP This namespace is comprised of regular IP node naming, i.e., DNS names and addresses taken from any version of the Internet Protocol. A processor dealing with the IP namespace is required to determine the syntax (DNS name, IP address version) of the group expression.

OLM This namespace covers address strings immediately valid in an overlay network. A processor handling those strings need not be aware of the address generation mechanism, but may pass these values directly to a corresponding overlay.

SIP The SIP namespace is an example of an application-layer scheme that bears inherent group functions (conferencing). SIP conference URIs may be directly exchanged and interpreted at the application, and mapped to group addresses on the system level to generate a corresponding multicast group.

Opaque This namespace transparently carries strings without further syntactical information, meanings or associated resolution mechanism.

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

This draft does neither introduce additional messages nor novel protocol operations.

8. Acknowledgements

We would like to thank the HAMcast-team, Dominik Charousset, Gabriel

Hege, Fabian Holler, Alexander Knauf, Sebastian Meiling, and Sebastian Woelke, at the HAW Hamburg for many fruitful discussions and for their continuous critical feedback while implementing API and a hybrid multicast middleware.

This work is partially supported by the German Federal Ministry of Education and Research within the HAMcast project, which is part of G-Lab.

9. Informative References

- [I-D.ietf-mboned-auto-multicast]
Thaler, D., Talwar, M., Aggarwal, A., Vicisano, L., Pusateri, T., and T. Morin, "Automatic IP Multicast Tunneling", draft-ietf-mboned-auto-multicast-11 (work in progress), July 2011.
- [RFC1075] Waitzman, D., Partridge, C., and S. Deering, "Distance Vector Multicast Routing Protocol", RFC 1075, November 1988.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.
- [RFC3678] Thaler, D., Fenner, B., and B. Quinn, "Socket Interface Extensions for Multicast Source Filters", RFC 3678, January 2004.
- [RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

- [RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.
- [RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.
- [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, August 2006.
- [RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, October 2007.

Appendix A. C Signatures

This section describes the C signatures of the common multicast API, which is defined in Section 4.

```
int createMSocket(uint32_t *if);

int deleteMSocket(int s);

int join(int s, const uri group_name);

int leave(int s, const uri group_name);

int srcRegister(int s, const uri group_name,
                uint_t num_ifs, uint_t *ifs);

int srcDeregister(int s, const uri group_name,
                  uint_t num_ifs, uint_t *ifs);

int send(int s, const uri group_name,
         size_t msg_len, const void *buf);

int receive(int s, const uri group_name,
            size_t msg_len, msg *msg_buf);
```

```
int getInterfaces(int s, uint_t num_ifs, uint_t *ifs);

int addInterface(int s, uint32_t if);

int delInterface(int s, uint32_t if);

int setTTL(int s, int h, uint_t num_ifs, uint_t *ifs);

int getTTL(int s, int h);

int groupSet(uint32_t if, uint_t *num_groups,
             struct groupSet *groupSet);

struct groupSet {
    uri group_name; /* registered multicast group */
    int type;        /* 0 = listener state, 1 = sender state,
                     * 2 = sender & listener state */
};

int neighborSet(uint32_t if, uint_t *num_neighbors,
               const uri *neighbor_address);

int childrenSet(uint32_t if, const uri group_name,
               uint_t *num_children, const uri *child_address);

int parentSet(uint32_t if, const uri group_name, uint_t *num_parents,
              const uri *parent_address);

int designatedHost(uint32_t if, const uri *group_name);
```

Appendix B. Practical Example of the API

```
-- Application above middleware:

//Initialize multicast socket;
//the middleware selects all available interfaces
MulticastSocket m = new MulticastSocket();

m.join(URI("ip://224.1.2.3:5000"));
m.join(URI("ip://[FF02:0:0:0:0:0:0:3]:6000"));
m.join(URI("sip://news@cnn.com"));

-- Middleware:

join(URI mcAddress) {
    //Select interfaces in use
    for all this.interfaces {
        switch (interface.type) {
            case "ipv6":
                //... map logical ID to routing address
                Inet6Address rtAddressIPv6 = new Inet6Address();
                mapNametoAddress(mcAddress, rtAddressIPv6);
                interface.join(rtAddressIPv6);
            case "ipv4":
                //... map logical ID to routing address
                Inet4Address rtAddressIPv4 = new Inet4Address();
                mapNametoAddress(mcAddress, rtAddressIPv4);
                interface.join(rtAddressIPv4);
            case "sip-session":
                //... map logical ID to routing address
                SIPAddress rtAddressSIP = new SIPAddress();
                mapNametoAddress(mcAddress, rtAddressSIP);
                interface.join(rtAddressSIP);
            case "dht":
                //... map logical ID to routing address
                DHTAddress rtAddressDHT = new DHTAddress();
                mapNametoAddress(mcAddress, rtAddressDHT);
                interface.join(rtAddressDHT);
                //...
        }
    }
}
```

Appendix C. Deployment Use Cases for Hybrid Multicast

This section describes the application of the defined API to implement an IMG.

C.1. DVMRP

The following procedure describes a transparent mapping of a DVMRP-based any source multicast service to another many-to-many multicast technology.

An arbitrary DVMRP [RFC1075] router will not be informed about new receivers, but will learn about new sources immediately. The concept of DVMRP does not provide any central multicast instance. Thus, the IMG can be placed anywhere inside the multicast region, but requires a DVMRP neighbor connectivity. The group communication stack used by the IMG is enhanced by a DVMRP implementation. New sources in the underlay will be advertised based on the DVMRP flooding mechanism and received by the IMG. Based on this the event "new_source_event" is created and passed to the application. The relay agent initiates a corresponding join in the native network and forwards the received source data towards the overlay routing protocol. Depending on the group states, the data will be distributed to overlay peers.

DVMRP establishes source specific multicast trees. Therefore, a graft message is only visible for DVMRP routers on the path from the new receiver subnet to the source, but in general not for an IMG. To overcome this problem, data of multicast senders will be flooded in the overlay as well as in the underlay. Hence, an IMG has to initiate an all-group join to the overlay using the namespace extension of the API. Each IMG is initially required to forward the received overlay data to the underlay, independent of native multicast receivers. Subsequent prunes may limit unwanted data distribution thereafter.

C.2. PIM-SM

The following procedure describes a transparent mapping of a PIM-SM-based any source multicast service to another many-to-many multicast technology.

The Protocol Independent Multicast Sparse Mode (PIM-SM) [RFC4601] establishes rendezvous points (RP). These entities receive listener and source subscriptions of a domain. To be continuously updated, an IMG has to be co-located with a RP. Whenever PIM register messages are received, the IMG must signal internally a new multicast source using the event "new_source_event". Subsequently, the IMG joins the group and a shared tree between the RP and the sources will be established, which may change to a source specific tree after a sufficient number of data has been delivered. Source traffic will be forwarded to the RP based on the IMG join, even if there are no further receivers in the native multicast domain. Designated routers of a PIM-domain send receiver subscriptions towards the PIM-SM RP.

The reception of such messages initiates the event "join_event" at the IMG, which initiates a join towards the overlay routing protocol. Overlay multicast data arriving at the IMG will then transparently be forwarded in the underlay network and distributed through the RP instance.

C.3. PIM-SSM

The following procedure describes a transparent mapping of a PIM-SSM-based source specific multicast service to another one-to-many multicast technology.

PIM Source Specific Multicast (PIM-SSM) is defined as part of PIM-SM and admits source specific joins (S,G) according to the source specific host group model [RFC4604]. A multicast distribution tree can be established without the assistance of a rendezvous point.

Sources are not advertised within a PIM-SSM domain. Consequently, an IMG cannot anticipate the local join inside a sender domain and deliver a priori the multicast data to the overlay instance. If an IMG of a receiver domain initiates a group subscription via the overlay routing protocol, relaying multicast data fails, as data are not available at the overlay instance. The IMG instance of the receiver domain, thus, has to locate the IMG instance of the source domain to trigger the corresponding join. In the sense of PIM-SSM, the signaling should not be flooded in underlay and overlay.

One solution could be to intercept the subscription at both, source and receiver sites: To monitor multicast receiver subscriptions ("join_event" or "leave_event") in the underlay, the IMG is placed on path towards the source, e.g., at a domain border router. This router intercepts join messages and extracts the unicast source address S, initializing an IMG specific join to S via regular unicast. Multicast data arriving at the IMG of the sender domain can be distributed via the overlay. Discovering the IMG of a multicast sender domain may be implemented analogously to AMT [I-D.ietf-mboned-auto-multicast] by anycast. Consequently, the source address S of the group (S,G) should be built based on an anycast prefix. The corresponding IMG anycast address for a source domain is then derived from the prefix of S.

C.4. BIDIR-PIM

The following procedure describes a transparent mapping of a BIDIR-PIM-based any source multicast service to another many-to-many multicast technology.

Bidirectional PIM [RFC5015] is a variant of PIM-SM. In contrast to

PIM-SM, the protocol pre-establishes bidirectional shared trees per group, connecting multicast sources and receivers. The rendezvous points are virtualized in BIDIR-PIM as an address to identify on-tree directions (up and down). However, routers with the best link towards the (virtualized) rendezvous point address are selected as designated forwarders for a link-local domain and represent the actual distribution tree. The IMG is to be placed at the RP-link, where the rendezvous point address is located. As source data in either cases will be transmitted to the rendezvous point address, the BIDIR-PIM instance of the IMG receives the data and can internally signal new senders towards the stack via the "new_source_event". The first receiver subscription for a new group within a BIDIR-PIM domain needs to be transmitted to the RP to establish the first branching point. Using the "join_event", an IMG will thereby be informed about group requests from its domain, which are then delegated to the overlay.

Appendix D. Change Log

The following changes have been made from
draft-irtf-samrg-common-api-01

1. Pseudo syntax for lists objects changed
2. Editorial improvements

The following changes have been made from
draft-irtf-samrg-common-api-00

1. Incorrect pseudo code syntax fixed
2. Minor editorial improvements

The following changes have been made from
draft-waehlich-sam-common-api-06

1. no changes; draft adopted as WG document (previous draft-waehlich-sam-common-api-06, now draft-irtf-samrg-common-api-00)

The following changes have been made from
draft-waehlich-sam-common-api-05

1. Description of the Common API using pseudo syntax added
2. C signatures of the Comon API moved to appendix

3. updateSender() and updateListener() calls replaced by events
4. Function destroyMSocket renamed as deleteMSocket.

The following changes have been made from
draft-waehlich-sam-common-api-04

1. updateSender() added.

The following changes have been made from
draft-waehlich-sam-common-api-03

1. Use cases added for illustration.
2. Service calls added for inquiring on the multicast distribution system.
3. Namespace examples added.
4. Clarifications and editorial improvements.

The following changes have been made from
draft-waehlich-sam-common-api-02

1. Rename init() in createMSocket().
2. Added calls srcRegister()/srcDeregister().
3. Rephrased API calls in C-style.
4. Cleanup code in "Practical Example of the API".
5. Partial reorganization of the document.
6. Many editorial improvements.

The following changes have been made from
draft-waehlich-sam-common-api-01

1. Document restructured to clarify the realm of document overview and specific contributions s.a. naming and addressing.
2. A clear separation of naming and addressing was drawn. Multicast URIs have been introduced.
3. Clarified and adapted the API calls.

4. Introduced Socket Option calls.
5. Deployment use cases moved to an appendix.
6. Simple programming example added.
7. Many editorial improvements.

Authors' Addresses

Matthias Waehlich
link-lab & FU Berlin
Hoenower Str. 35
Berlin 10318
Germany

Email: mw@link-lab.net
URI: <http://www.inf.fu-berlin.de/~waehl>

Thomas C. Schmidt
HAW Hamburg
Berliner Tor 7
Hamburg 20099
Germany

Email: schmidt@informatik.haw-hamburg.de
URI: <http://inet.cpt.haw-hamburg.de/members/schmidt>

Stig Venaas
cisco Systems
Tasman Drive
San Jose, CA 95134
USA

Email: stig@cisco.com

SAMRG
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2012

T. Li
Z. Sun
H. Wang
C. Jia
NUDT
July 12, 2011

P2MP Streaming Media Delivery Problem Statement
draft-litao-p2mpsm-d-problem-statement-01

Abstract

Currently, more and more Internet streaming services are getting increasingly popular among users. Recent large-scale deployed delay- and loss-sensitive services, such as IPTV, impose stringent requirements to the streaming media delivery. Streaming media service providers and operators have been struggling to make the QoE of the subscribers at a satisfactory level. However, most of the existing P2MP streaming media delivery technologies, such as IP multicast, are far from ideal in the aspects of scalability, reliability and stability. This document presents the problem statements in point to multipoint streaming media delivery, explains why network awareness and policy-based routing control should be urgently addressed for the point to multipoint streaming media delivery mechanism and introduces what should be further considered in the future.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference

material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 13, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Terminology	5
3. Problem Statement Scope	5
4. Architecture Requirements	7
4.1. Scalability	7
4.2. Incremental Deployment	7
5. Why IETF Needs to Develop Solutions Instead of Relying on Existing Technologies?	7
5.1. IP Multicast	7
5.2. RTP/RTCP Extensions of IP Multicast	8
5.3. Overlay (CDN P2P)	8
6. Security Considerations	9
7. IANA Considerations	9
8. Acknowledgments	9
9. References	9
9.1. Normative References	9
9.2. Informative References	10
Authors' Addresses	11

1. Introduction

Streaming traffic is among the fastest growing traffic on the Internet. As streaming media delivery has characteristics of long-live connection and high stable transmission rate, the Internet capacity is imposed with stringent requirements of high bandwidth, low delay and jitter, and low packet loss. The situation is further complicated by frequent load variations from the dynamic behavior of large asynchronous clients. The current Internet faces many challenges from the core to the edge, as its end-to-end design principle with best-effort service cannot be well suited for point to multipoint (P2MP) streaming media delivery application.

From an ISP's perspective, the Internet bandwidths are relatively scarce and precious resources, especially in the core network. When end-to-end unicast technology is used for the streaming media delivery over the Internet, a separated point-to-point connection (e.g., UDP/TCP connections) is employed between the sender and each receiver. It leads to the poor use of the available bandwidth due to the multiple copies of streaming media object on the same link. It is desirable for ISP that an efficient P2MP delivery mechanism is established for delivering copies of the streaming media data to multiple recipients at different locations, in order to minimize the amount of the required network resources (usually in terms of bandwidth).

From an end-user's perspective, quality of experience (QoE) is widely appreciated as an important subjective measurement for the streaming media delivery service, such as IPTV, VoD. Usually, the quality of service (QoS) metrics (such as network delay, jitter and packet loss), rather than QoE, is used for the objective measure of the streaming media delivery service. With the dramatic improvement in digital video quality (like High Definition) and the prevalence of streaming media applications, the Internet is facing with significant challenges in providing QoS insurance. It is essential to build an efficient P2MP streaming media delivery mechanism for an optimal end-user experience. The better the QoE is, the more likely the end-users subscribe streaming media service, which benefits both ISPs and the content providers.

From a service provider's perspective, a service network requires efficient and low-cost deployment, maintenance and management. However, the existing distributed routing models and protocols for P2MP streaming media delivery cannot provide enough support to fulfill the above requirements. It is crucial for network fundamental infrastructure to provide efficient protocol and mechanisms, such that the service providers can rapidly and automatically monitor, detect and troubleshoot performance issues in

the service networks.

The motivation of this document is to clarify the problems facing the P2MP streaming media delivery.

2. Terminology

Quality of Experience (QoE): The overall acceptability of an application or service, as perceived subjectively by the end-user[ITU-TP].

Quality of Service (QoS): The collective effect of performance which determines the degree of satisfaction of a user of the service[ITU-T].

Streaming Media Service Provider (SMSP): A company that offers delivering streaming media services such as providing video content, improving network performance, and enhancing transmission quality, etc.

3. Problem Statement Scope

There are a number of problems related to the P2MP streaming media delivery. The two major issues are listed below.

(1) The difficulty of network state information (NSI) acquisition.

Unpredictable behaviors of users present unusual challenges to the network delivering P2MP streaming media. The links and network elements (routers and switches) experience rapid and large-scale changes in bandwidth availability. Therefore, the congestion may occur frequently over time and space, which introduces delays and jitters to the flows of streaming media. That is adverse to the streaming media delivery due to its stringent demands on discontinuity-free and high stable transmission rate.

Network-aware streaming media delivery is an attractive approach to mitigate the problems. It plays a very important role in delivering timely and accurate information for supporting well-founded adaption decisions in monitoring, diagnostics and failure restoration. However, the exact information about the current condition of the network is hard to obtain for adapting the resource demands for QoS. Some NSI can be derived from the transport level or directly from the application level. NSI acquisition at application level is widely accepted as it can be easily implemented and deployed. Transport-level NSI acquisition can provide more accurate end-to-end

information while brings less overhead to the network. And, sophisticated analysis is possible by using the network tomography techniques to infer the lower-layer network state. However, rapidly and accurately locating the problem to the particular network node/link requires more network state information directly from the inside of the network. Some protocols in TCP/IP stack can implicitly gather the end-to-end network performance metrics only for support embedded control mechanism, such as TCP and RTP[RFC1889] [RFC3550]. Few of the protocols explicitly provide mechanisms, such as bandwidth and timestamp, for delivering the state information of network elements (such as routers, switches and links) to the application, which makes QoS assurance of the P2MP streaming media delivery difficult.

(2) The difficulty of policy-based control

Most of the P2MP streaming delivery systems employ pull-based service model based on IP multicast technology. In the model, the streaming data are transmitted along the delivery paths which are decided or calculated according to the distribution of users' requests. However, other service models, such as push-based and pull-and-push based, are also very useful in providing different featured services to the end-users. For example, the subscribed advertisements, video messages and news can be directly delivered to the specific end-users without explicit requests under push-based service model. The model improves initiative and flexibility of P2MP streaming media delivery service, and provides technical support for the implementation of value-added services. Pull-and-push based service model can be very helpful in reducing the delivery time derived from the dynamic changes of users' request. For example, in the of IPTV service the edge network, the channel zapping delay can be shortened, if the streaming data are pushed to some intermediate nodes before the users pull the content.

The different service models, i.e., pull-based, push-based and pull-and-push based model, can provide different optimized characteristic to the P2MP delivery service of streaming media. And, it will be significant if different service models can be implemented as different policies for the P2MP streaming media delivery. However, the difficulty is that most of current delivery mechanisms, such as IP multicast, do not separate the policy from the mechanism. This lack of flexibility, for example, prevents ISP/SP that would ideally prefer to choose some service model, but not others, in transmitting media data. It is essential for a protocol/mechanism of P2MP streaming media delivery to provide simple, stable and common interface to enable policy-based control. In this case, a framework accommodating different service models and different delivery policy can be constructed.

4. Architecture Requirements

4.1. Scalability

Scalability is one of the most critical issues in the deployment of P2MP streaming media delivery system. Many aspects of the system, such as routing protocols, including state information maintenance and address allocations, etc., should take the scalability into consideration [I-D.boudani-mpls-multicast-tree]. P2MP streaming media delivery suffers from the large number and stringent QoE requirement of end-users. Scalability can be evaluated not only in terms of the number of groups but also by the number of participants per group and by groups for which the set of participants changes often over time. A scalable P2MP streaming media delivery system should be simple to implement, robust, use minimal network overhead and consume minimal resources of routers/switches [Wong]. Usually, hierarchical architecture is a good design choice for tackling the scalability problem.

4.2. Incremental Deployment

Incremental deployment is an important architectural attribution required by the P2MP streaming media delivery mechanisms. The cost of deployment, running, and maintaining can be largely reduced, if the efficient mechanism for P2MP streaming media delivery supports incremental deployment. That means, new schemes and solutions should not modify the original fundamental infrastructure, and the update or the substitution can proceed smoothly. New mechanisms/protocols are expected to coexist and integrate with existing protocols, such as IP multicast and RTP, while offering more flexible deployment options.

5. Why IETF Needs to Develop Solutions Instead of Relying on Existing Technologies?

5.1. IP Multicast

IP multicast is the most effective technology to solve the bandwidth problem in streaming media transmission. It can reduce both network link cost and server bandwidth requirement for serving a large number of receivers simultaneously[Lao]. In IP multicast, the source sends only one copy of an addressed packet to a group of receivers to reduce the consumption of the network bandwidth. For P2MP distribution of data, Source Specific Multicast (SSM) [RFC4608] has been proposed to offer simplified unidirectional routing. When there is only one source of multimedia traffic, it is unnecessary to require the routing infrastructure to create either a full mesh of distribution trees or bidirectional connectivity among all group

members. Although the complete protocol architecture of IP multicast has already been constructed and standardized, it has not been widely deployed in the past years. One problem of IP multicast is the scalability. To support a large number of groups and users, routers in the network should keep all the states information of the active groups. The costs of maintaining the state information come in terms of memory at the routers, and processing of periodic control messages to maintain such state. The control message overhead and memory cost grow linearly with the number of multicast groups supported by the router. The second problem related the IP multicast technology is its lack of the supported features required of a robust commercial implementation[Diot] . The features include authentication and accounting, group management, security and network management, etc.

5.2. RTP/RTCP Extensions of IP Multicast

Although SSM technology may simplify routing for P2MP streaming data delivery, unidirectional communication makes it impossible for the source and other members in the group to obtain feedback and control information[Chesterfield] . An extension of RTCP has been proposed to enable SSM that do not have many-to-many communication capability to receive RTP data streams and to continue to participate in the RTCP by using multicast in the source-to-receiver direction and unicast to send receiver's feedback to the source on the standard RTCP port[RFC5760] . RTP/RTCP's original target was multiparty multimedia conferencing in multicast networks. RTP is designed for data transmission and RTCP is for the distribution of feedback and control information. RTP/RTCP provides SSM the ability for supporting the monitoring and locating the fault in the network. By utilizing network tomography technology, RTCP can provide a wealth of data for QoE monitoring, network management and fault diagnosis purposes [ACBegen] [Begen] .On the Scalability of RTCP-Based Network Tomography for IPTV Services. However, these end-to-end network state information are often insufficient to accurately identify the particular fault router/switch or link. The collection, analysis and processing of the NSI required by network tomography technology make real-time fault location and fast failure recovery hard to be implemented.

5.3. Overlay(CDN P2P)

Peer-to-peer (P2P) technology has been proposed for streaming media delivery aiming at the high cost of deployment and maintenance of the infrastructure-based approach [Kien]. P2P lets users end hosts forward video data to other users' end hosts in the downstream. The technology has the advantages of scalability and easy-to-development. It also can provide robust and resilient when the network failure occurs. However, P2P streams bring no profit to the ISPs and violate

the fairness principle for the business. ISP cannot benefit from offering service to the peers who wish to freely use network bandwidth resources. Moreover, the streaming media data delivery by P2P concerns the logic network rather than physical network states and topology. A packet may be sent on the same link more than once, which incurs inefficiency in utilizing the precious network resources.

In CDN architecture, the content is first distributed to CDN servers which are pre-placed in many regions. CDN can provide reliable delivery and cost-effective scaling. It can also provide high security for the contents stored. However, CDN costs more and provides less scalability.

The technology of hybrid CDN and P2P has been proposed for live streaming and VoD applications. The hybrid architecture is an effective way to reduce the cost of content distribution and guarantee the quality of transmission for P2MP streaming media service. However, hybrid CDN and P2P architecture requires dedicated dimensioning and design, as the different contribution policies, interrelated system parameters and their impacts on multiple performance metrics should be carefully determined for better performance.

6. Security Considerations

This document has no additional requirement for security.

7. IANA Considerations

This document has no additional requirement for IANA Considerations.

8. Acknowledgments

We would like to thank Jigang Wu and Guozhi Song for their valuable suggestions.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

- [ACBegen] Begen, A., Perkins, C., and J. Ott, "On the Use of RTP for Monitoring and Fault Isolation in IPTV".
- [Begen] Begen, A., Perkins, C., and J. Ott, "On the Scalability of RTCP-Based Network Tomography for IPTV Services", 3 2010.
- [Chesterfield]
Chesterfield, J. and E. Schooler, "An Extensible RTCP Control Framework for Large Multimedia Distributions", In Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications 2003, 3 2003.
- [Diot] Diot, C., Levine, B., and J. Lyles, "Deployment issues for the IP multicast service and architecture", IEEE Network 2000,14(1):78_8, 14(1):78_8 2000.
- [I-D.boudani-mpls-multicast-tree]
Boudani, A., "An Effective Solution for Multicast Scalability:The MPLS Multicast Tree (MMT)", draft-boudani-mpls-multicast-tree-06 (work in progress), October 2004.
- [ITU-T] "Telephone Network and ISDN Quality of Service, Network Management and Traffic Engineering: Terms and Definitions Related to Quality of Service and Network Performance Including Dependability", ITU-T Recommendation E. 800, Aug 1994.
- [ITU-TP] "Appendix I to P.10/G.100: Definition of QoE", ITU-TP.10/G. 100, Jan 2007.
- [Kien] Hua, K., Tantaoui, M., and W. Tavanapong, "Video delivery technologies for large-scale deployment of multimedia applications", 3 2004.
- [Lao] Lao, L., Cui, J., Gerla, M., and D. Maggiorim, "A Comparative Study of Multicast Protocols: Top, Bottom, or In the Middle?", In Proceedings of INFOCOM' 2006.
- [RFC1889] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

- [RFC4608] Meyer, D., Rockell, R., and G. Shepherd, "Source-Specific Protocol Independent Multicast in 232/8", BCP 120, RFC 4608, August 2006.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [Wong] Wong, T., Ed., "Multicast Forwarding and Application State Scalability in the Internet", phdthesis Wong:CSD-00-1114, December 2000, <<http://www.eecs.berkeley.edu/Pubs/TechRpts/2000/6423.html>>.

Authors' Addresses

Tao. Li
NUDT
47 Yanwachi St
Changsha, Hunan 410073
China

Phone: +86-13487568531
Email: taoli.nudt@gmail.com

Zhi Gang. Sun
NUDT
47 Yanwachi St
Changsha, Hunan 410073
China

Phone: +86-13875903721
Email: sunzhigang@nudt.edu.cn

Hui Wang
NUDT
47 Yanwachi St
Changsha, Hunan 410073
China

Phone: +86-13467578342
Email: wanghuinudt@gmail.com

Chun Bo. Jia
NUDT
47 Yanwachi St
Changsha, Hunan 410073
China

Phone: +86-13407318066
Email: jiachunbo1988@sina.com

SAMRG
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2012

Z. Sun
H. Wang
T. Li
J. Mao
NUDT
July 12, 2011

Labelcast Protocol
draft-sunzhigang-sam-labelcast-02

Abstract

This document presents a lightweight transport protocol-Labelcast, especially for long-lived connection video streams. This protocol provides a procedure for application programs to send media data to other programs. Like the UDP protocol, the Labelcast does not provide delivery and duplicate protection. However, it provides efficient support for the monitoring of video transmission quality. And, fast forwarding mechanism based on label is also supported by the protocol. Labelcast can coexist and integrate with existing mechanisms, such as IP multicast and RTP/RTCP, while offering more flexible deployment options and scaling to support a greater number of simultaneous multicast groups.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 13, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Ideas of Labelcast	5
2.1. Replacing UDP for video streaming delivery	5
2.2. Relationship to RTP	5
2.3. Relationship to IP multicast	6
2.4. Lightweight protocol	6
3. Labelcast Protocol	7
3.1. Labelcast header format	7
3.2. Forward label table	8
3.3. The Requirement for IP Protocol Fields	8
4. Requirement of Protocol	8
4.1. Processing Requirement	8
4.1.1. Building the Label Paths	8
4.1.2. Requirement of the Source	8
4.1.3. Processing Requirement of Labelcast Forwarding Nodes	9
4.1.4. Requirement of End Systems	9
4.2. Impact on protocol stack	9
4.2.1. Source serve	9
4.2.2. Client	10
4.2.3. Forwarding Node	10
5. Application Example	10
5.1. Point-to-Multipoint Data Distribution Based on Labels	10
5.2. Video-aware Network Processing	11
5.3. Detecting Network Status	12
6. Labelcast Deployment	12
6.1. Relationship to Common API	12
6.2. IP tunnel	12
6.3. Gateway	13
7. Security Considerations	13
8. IANA Considerations	13
9. Acknowledgments	13
10. References	13
10.1. Normative References	13
10.2. Informative References	13
Authors' Addresses	14

1. Introduction

Internet Protocol Television (IPTV) service has emerged as one of the most promising applications in the coming years. IPTV video content is delivered over IP networks and based on IP techniques. However, there is lacking efficient Internet protocols or mechanisms for IPTV services, especially between core network and access network, which is the bottleneck for IPTV data transmission. [I-D.litao-p2mpsmd-problem-statement]

Peer-to-Peer based on application layer technology just concerns the logic network rather than real network states and topology, and a great deal of redundant streams pass over the core Internet. Due to lack of administration, P2P reduces the ISPs' benefits and is not feasible to IPTV live broadcast systems. Pure clients P2P can't be further developed to the main delivery technologies.

IP multicast is insufficient to deploy largely-scale over the Internet because of their defects in scalability, user management, flow control, etc. The scalability of IP multicast is an important issue and this hampers its implementation. Routers keep every active group states and when the scale is increasing, the burden would be expanded. Another problem in IP multicast is the reliability. The UDP is used as the transport layer in IP multicast and the integrity of data will not be assured.

The transport layer protocols TCP/UDP, which are not designed for IPTV service at the beginning, can not achieve the motivation of effective video traffic transmission due to the characteristics of IPTV streams, such as long-time connection, high bandwidth consumption and so on.

TS over RTP/UDP RFC 1889 [RFC1889] are the most widely used transmission means for streaming media, however there are two problems. One problem is that RTP/UDP can not provide efficient Point-to-Multipoint IPTV distribution methods. Another is that RTP/UDP is in lack of support to monitoring the transfer quality of the video. Not only end systems are hard to realize monitoring towards their QoE (Quality of Experience), but also the intermediate routing nodes can't optimize QoS due to the lack of enough information. RFC 3550 [RFC3550]

Labelcast is a transport protocol supporting Point-to-Multipoint IPTV data distribution especially for the characteristics of long-lived connection, high bandwidth consumption and continuity. This protocol contains abundant fields that can support Point-to-Multipoint data transmission, and video quality monitoring both for intermediate routing nodes and end systems. Labelcast can efficiently meet the

requirements of the video quality transmission monitoring, failure detection and isolation of network failures.

2. Ideas of Labelcast

The basic ideas are as follows:

1. IPTV data distribution protocol based on Labelcast can support the stream characteristics of long-lived connection, high bandwidth consumption, real-time and continuity.
2. Simplify the implementation of Point-to-Multipoint data transmission by utilizing labels.
3. Reduce the processing overhead of intermediate routing nodes.
4. Support monitoring of video transmission quality.

2.1. Replacing UDP for video streaming delivery

UDP is an unconnected and unreliable transport protocol. In IPTV data transmission, only the destination port field of UDP header is functional, while the field of source port, length and checksum are not used. Labelcast defines some specific fields to support the features important to video traffic transmission, such as bandwidth and timestamp for monitoring. Labelcast sets up the transmission paths between source and receivers through label switching. The protocol supports for the packet priorities, bandwidth reservation, calculating time interval and packets loss rate, etc. Based on the state information, the video transmission quality can be monitored and the scheduling strategy can be optimized.

Besides, the evaluation of the video transmission quality at intermediate routing nodes and end systems can take advantage of MDI index RFC 4445 [RFC4445]. DF and MLR values are parts of MDI. DF value reflects the delay jitter of the video traffic and MLR value reflects the loss rates of the video traffic. The bandwidth of the transmission requirement, loss rate and the delay jitters can also be calculated by intermediate routing nodes, and the transmission QoS can be controlled.

2.2. Relationship to RTP

RTP is designed for supporting multimedia application, and it can not provide reliable insurance for sequenced data streams, nor flow or congestion control mechanism. These are all implemented by RTCP. RTCP sends periodically control messages to the group members. The

members can get the condition of network through feedback data. RTCP control flow increases linearly to the number of participants, and it consumes more bandwidth in larger groups. In this situation, Labelcast can replace UDP to work with RTP/RTCP as a transport layer protocol as it provide abundant information for video quality monitor, failure detection and flow control . Moreover, Labelcast is responsible for monitoring network nodes (routers/switches) in the Internet, while RTP/RTCP is responsible for monitoring end hosts. Labelcast and RTP/RTCP can work together to get the complete view of the whole network.

2.3. Relationship to IP multicast

IP multicast is considered as the most effective technology to solve the bandwidth problem in IPTV data transmission. However, IP multicast is not a network-aware mechanism and can not diagnose the transmission quality. Most streams are concentrated in the minority paths (the shortest ones). In IP multicast, the packets are forwarded through fixed paths, even if there is congestion between two nodes. Besides, IP multicast can not guarantee the quality of transmission as it uses UDP protocol in transport layer.

As a supplement of the IP multicast technology, Labelcast can co-exist with IP multicast in the operational network. Routers can be configured to decide whether a packet is forwarded by IP multicast or labelcast labels, with the infomation of protocol field in IP packet. In Labelcast, the forwarding is depend on the label field, and the packets are forwarded by the label switching. Label aggregation techniques can be used to reduce the forwarding states. IP multicast address is considered as different group ID in Labelcast. If the forwarding node is not a Labelcast node, the packets is forwarded based on the layer 3 processing.

In this way we are able to combine the two multicast technologies together even though it seems that they are conflicting with each other at the first glance. Labelcast can replace IP multicast if all the intermediate nodes support it. Compare to IP multicast, Labelcast can accelerate the forwarding speed and reduce the total routing overhead among the transmitting path. In a hybrid network of IP multicast and Labelcast, for example, IP tunnel mechanism or gateway mechanism can be utilized for the connection. Two solutions will be described later in chapter 6.

2.4. Lightweight protocol

Labelcast is a lightweight protocol with two reasons. Firstly, it can be configured by utilizing Openflow technique [McKeown], and the labels are allocated through an external controller. The way to

calculate the label can be either centralized or distributed.
 Secondly, only a Labelcast module is added into the router, and the protocol stack in the hosts is with minor change.

3. Labelcast Protocol

3.1. Labelcast header format

Labelcast header has the length of 8 bytes with seven fields which is illustrated in figure 1.

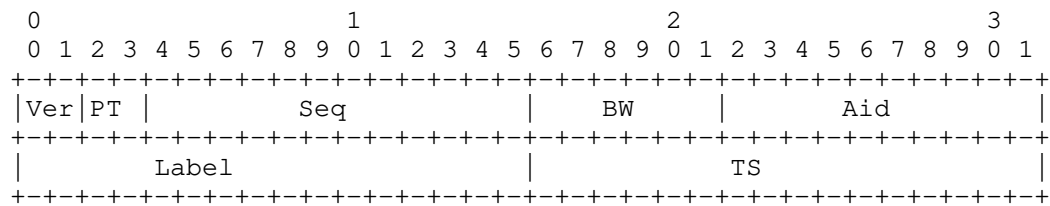


Figure 1: Head Format

(1) Ver [0:1] is the version of Labelcast protocol and it is defined "01" initially.

(2) PT [0:1] denotes the type of frame, which decides the priority to discard packets. whether the payload it carries is I frame or B frame or P frame reflect the importance of packets. I frames own the highest priority and P frames own the lowest. 11: having the highest discarding priority when there is a congestion (The first to be dropped). 00: having the lowest discarding level if there is a congestion (The last to be dropped). When network congestion occurs, flow control based on priorities can be realized.

(3) Seq [0:11] denotes the sequences of packets which is used for the detection of the packets loss. It denotes the sequences of packets in the flow. It is used to monitor the packets loss and order. Suppose the sequence field is k bits, the length of packets is n bytes, and the bandwidth of the flow is Mbps, then the period of the sequence is $T_{seq} = 2^k * n/M$.

(4) BW [0:5] denotes that the average bandwidth of flow is $BW * 128Kbps$. The maximum value is 8Mbps. BW which has the value of 0 denotes that bandwidth is unknown.

(5) Aid [0:7] denotes the application ID.

(6) Label [0:15] denotes labels related to the packets routing. The

processing actions of nodes are determined by this field. The nodes which support the label determine the next hop forwarding nodes by looking up the label table.

(7) TS [0:15] denotes the sending timestamp of packets, of which the unit is us(microsecond). It is used to account the delay between two nodes.

3.2. Forward label table

The forward label table is composed of four tuples, which include ingress port number, ingress label, egress port number and egress label. The label entry is modified by outside controller through control protocol.

In some special conditions, the tuple should be expanded. When there is a non-Labelcast node between two Labelcast nodes, it may receive two different video flows which have the same ingress port and label. However, the two different flows can not be distinguished by the above-mentioned forwarding label table. So some tuples are needed to identify flow ID, such as IP address of the source.

3.3. The Requirement for IP Protocol Fields

The value of Labelcast protocol which is identified in IP header field is 123.

4. Requirement of Protocol

4.1. Processing Requirement

4.1.1. Building the Label Paths

Similar to ATM and MPLS RFC 5332 [RFC5332], the virtual paths are established between the source and each receiver with extra means before IPTV data transmission. The extra means could be the signaling protocol of MPLS LDP RFC 3031 [RFC3031], or centralized control manners [I-D.kellil-sam-mtosp] of static calculation and configuration.

4.1.2. Requirement of the Source

(1) If source is unaware of the contents of applications, the Pri field is filled with 00 uniformly.

(2) Seq field could be filled when the packets are sent from source.

(3) BW field could be filled when the packets are sent from source.

(4) The Aid field should be filled according to the application requirements.

(5) Label field could be filled when the packets are sent from source.

(6) The TS field could be filled when the packets are sent from source.

4.1.3. Processing Requirement of Labelcast Forwarding Nodes

(1) Ver, Pri, Seq, BW and Aid can't be modified by Labelcast forwarding nodes.

(2) Label field should be modified according to the next hop distribution nodes. When packets arrive at the Label nodes, Label table is looked up at first. From the label table, the local processing actions and the next egress labels are known .

(3) According to the extra configure, TS field could be modified or keep unchanged by the forwarding nodes.

(4) The nodes which don't support Labelcast protocol can forward the packets based on IP address.

In order to prevent the initial virtual paths being deviated by IP forwarding from their inherent orientations, IP tunnels should be adopted. The packets are encapsulated with the address of next hop Labelcast nodes as the destination address, and then they can be sent to the next hop Label nodes directly.

4.1.4. Requirement of End Systems

(1) The end systems submit the payload of packets to different applications according to Aid field.

(2) The video transmission quality can be evaluated with BW, Seq, TS parameters etc.

4.2. Impact on protocol stack

4.2.1. Source serve

The communication interface will be negotiated between server and client before data transmission, Labelcast packets are identified by Aid. The session manager in Source Server adds the Labelcast module

and it can support TCP, UDP and Labelcast. Stream processor can provide RTSP/RTP/UDP/HTTP/Labelcast and it encapsulates the transport layer header with Labelcast protocol form.

4.2.2. Client

Client receives Labelcast packets with Raw Socket, it resolves Labelcast packets and sends the payload to the applications. Clients sample the video content and send the related information such as BW, Seq, TS to monitor.

4.2.3. Forwarding Node

When a intermediate Labelcast node receives the Labelcast packets, it mainly has three processes: 1. Modify the TTL options in the header and recompute the checksum of IP header. 2. Modify the timestamp of the header, and rewrite the local time. 3. look up the label table, get the next hop, and replace the label.

5. Application Example

5.1. Point-to-Multipoint Data Distribution Based on Labels

The label paths are composed of many labelcast nodes in series, and labels are assigned before the transmission of video traffic. The label tables are established according to their forwarding paths. The next hop Label nodes (one or multiple) can be determined by looking up the labels of packets in label table.

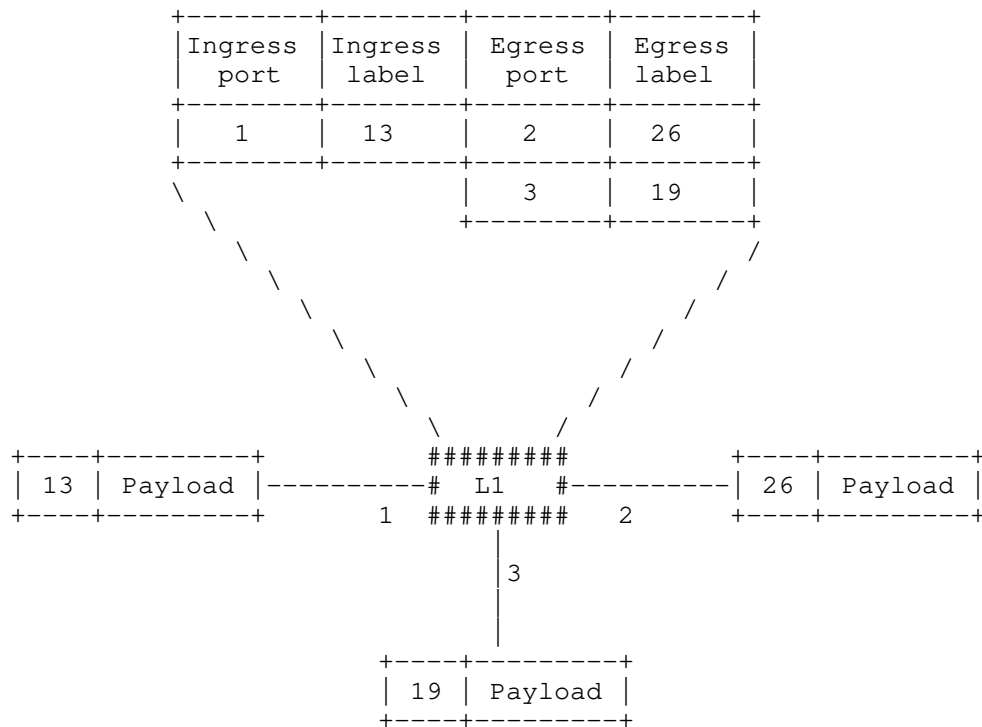


Figure 2: Label Processing

Figure 2 shows the label processing. When the packets with label 13 arrive at port 1 of Labelcast node L1, L1 looks up the label table, then determine the forwarding ports and their corresponding new labels. Since there are two forwarding ports for packets arriving port 1 with label 13, the packets are replicated at L1, and forwarded to port 2 and port 3. Before the forwarding process, the labels of packets are modified with new labels of 26 and 19 respectively.

5.2. Video-aware Network Processing

Labelcast protocol can simplify the video traffic identifications at network nodes and can guarantee application-aware QoS through the Pri field which is initialized at the source. The video transmission quality can be monitored through Bw, TS, Seq fields, and correspondingly the distribution paths are optimized by the monitoring results. For example, the arrival intervals can be calculated by the Bw and length of packets, and then the delay jitters of the successive arriving packets can be concluded. Based on this, the processing actions of forwarding nodes can be optimized.

5.3. Detecting Network Status

The characteristics of video transmission are high bandwidth-occupied, time-orderly, so they have high network requirement for Internet. For its continuity and time-orderly, video traffic itself is a good manner of network measurement, and network information is placed in the header of Labelcast protocol. Network status can be known by the Labelcast protocol. Through the analysis of TS and Seq fields, the performance of network can be acquired.

The timestamps are marked by each Label nodes passing by, and whether or not the upstream links are congested can be inferred through the delay jitters between packets. t_1, t_2 denote the timestamps of two packets with N intervals in previous hop Label node, and the timestamps of local Label node are t_3 and t_4 , then formula $a = (t_4 - t_3) / (t_2 - t_1)$ reflects the delay jitter of previous hop.

6. Labelcast Deployment

Labelcast requires minor modification to the underlay network which can be deployed gradually. Labelcast processing module can be added into router by ISP just like a value-added module.. Labelcast nodes can coexist with the normal IP nodes in the network, which is unlike of MPLS or IP multicast where all the routers must provide support.

6.1. Relationship to Common API

Considering a hybrid multicast network, a common multicast API [I-D.waehlich-sam-common-api] which is suitable for transparent communication in underlay and overlay, and grants access to the different multicast flavors. Common API offers calls to transmit and receive multicast data independent of the supporting layer and the underlying technological details. So the deployment of Labelcast will have little influence for users since it is transparent to application layer if with support of common API.

6.2. IP tunnel

IP tunnel can be used in a hybrid network where IP nodes and Labelcast nodes are connected with each other. If a Labelcast node A needs to go through an IP node B to another Labelcast node C, it will firstly look up the unicast or multicast table to get the forwarding port. Then, node A encapsulates the packet with the IP header of destination C and send it to B. Node B will forward the packet to C according to the IP route table. After node C receives the packet, it will remove the encapsulated IP header.

6.3. Gateway

Gateway can be used to connect an IP multicast network with a Labelcast network. It can map a UDP header to a Labelcast header for transmitting through these two different networks.

7. Security Considerations

The security issues brought by Labelcast is what we need take into consideration.

8. IANA Considerations

The value of Labelcast protocol which is identified in IP header field is 123.

9. Acknowledgments

The authors would like to acknowledge Dilife Tchnology, Inc. for help in Labelcast implementation of server and clients during project.

10. References

10.1. Normative References

- [RFC1889] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

10.2. Informative References

- [I-D.kellil-sam-mtoci]
Kellil, M., Janneteau, C., and P. Roux, "Multiparty Transport Overlay Control Protocol (MTOCP)",
draft-kellil-sam-mtoci-01 (work in progress), July 2010.
- [I-D.litao-p2mpsmd-problem-statement]
Sun, Z., Li, T., Wang, H., and C. Jia, "P2MP Streaming

Media Delivery", draft-litao-p2mpsmd-problem-statement-00 (work in progress), March 2011.

[I-D.waehlich-sam-common-api]

Waehlich, M., Schmidt, T., and S. Venaas, "A Common API for Transparent Hybrid Multicast", draft-waehlich-sam-common-api-01 (work in progress), October 2009.

[McKeown] McKeown, N., Anderson, T., and H. Balakrishnan, "OpenFlow: enabling innovation in campus networks.", Computer Communication Review, Vol38 2008.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC4445] Welch, J. and J. Clark, "A Proposed Media Delivery Index (MDI)", RFC 4445, April 2006.

[RFC5332] Eckert, T., Rosen, E., Aggarwal, R., and Y. Rekhter, "MPLS Multicast Encapsulations", RFC 5332, August 2008.

Authors' Addresses

Zhi Gang. Sun
NUDT
47 Yanwachi St
Changsha, Hunan 410073
China

Phone: +86-13875903721
Email: sunzhigang@nudt.edu.cn

Hui. Wang
NUDT
47 Yanwachi St
Changsha, Hunan 410073
China

Phone: +86-13467578342
Email: wanghuinudt@gmail.com

Tao. Li
NUDT
47 Yanwachi St
Changsha, Hunan 410073
China

Phone: +86-13487568531
Email: taoli.nudt@gmail.com

Jian Biao. Mao
NUDT
47 Yanwachi St
Changsha, Hunan 410073
China

Phone: +86-13618464415
Email: mjn198812@sina.com

