On Naming, Addressing, and Programming in Multicast*

Matthias Wählisch
Freie Universität Berlin & link-lab
Institut für Informatik
Email: waehlisch@ieee.org

Thomas C. Schmidt, Sebastian Meiling
HAW Hamburg
Dept. Informatik
Email: t.schmidt@ieee.org,
sebastian.meiling@haw-hamburg.de

I. Introduction

Emerging popular applications like IPTV and MMORGs urge the need for multicast. However, there is no global multicast deployment available today. At the same time many traditional and mobile systems would largely benefit from ease and efficiency of a network group service. A uniformly available multicast service will offer the chance to replace proprietary workarounds deployed in manifold ways.

Developers of applications for group communication need to decide on using IP or overlay multicast, multipoint unicast, or a reflecting server at the time of coding. In contrast, they may implement dynamic handling of different multicast technologies with respect to possible multicast deployment states, which increases complexity. In fact, application developers want to create group applications that run, without considering the specific multicast service below. The concepts of naming and addressing must be considered to abstract the identification of applications from data distribution paths. Multicast names are necessary to leverage technology–agnostic multicast programming. They can be used in a common group communication API, which has recently been defined in [1].

In this paper, we analyze the challenges of naming and addressing in global, heterogeneous multicast communication. Section II presents the problem space followed by Section III, which introduces a novel naming scheme. Finally, Section IV gives a conclusion and outlook.

II. Problem Space of Naming and Addressing in Multicast Communication

The general concept of naming and addressing distinguishes between names, which identify entities, and addresses, which are used to locate these entities. Such separation introduces abstraction between the layer that uses names and the layer that uses addresses. This gives higher flexibility. For example, the forwarding may change without modifying the name (e.g., required in Mobile IP), and names may be aggregated to a single address (e.g., implemented in LISP). However, it shifts the cost of complexity to the mapping problem. Each name needs to be mapped to the corresponding address, and vice versa to regain identification.

*This work is part of the H∀Mcast project (http://hamcast.realmv6.org), which is funded by the German Federal Ministry of Education and Research. H∀Mcast is part of G-Lab (http://www.german-lab.de).

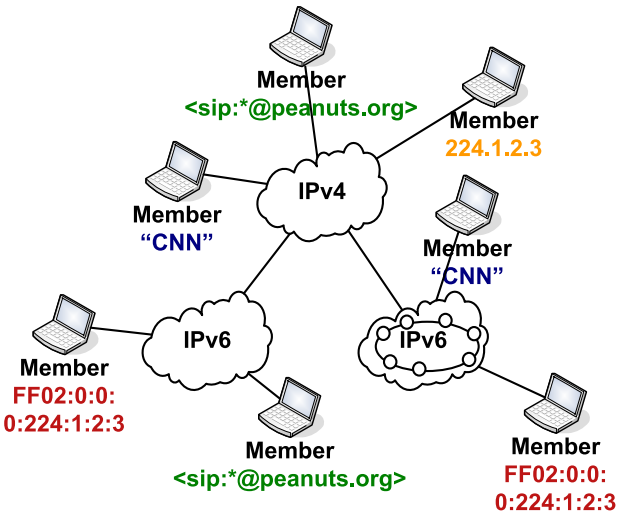

Fig. 1. Multicast diversity

In general, multicast differs from unicast in the following principle way:

- A multicast address does not represent a routing endpoint.
- There are several multicast flavors, i.e., Any Source Multicast (ASM) and Source–Specific Multicast (SSM).
- There is no feedback channel between receiver and source.

Multicast deployment occurs on several layers (e.g., IP and application layer multicast) and in different technologies (e.g., IPv4 and IPv6), cf., Figure 1. Domains may run the same technology, but remain isolated, or they may run distinct technologies, but host members of the same group. There is no relationship between multicast groups in the particular instance. In contrast to unicast, a multicast source serves per se multiple receivers. The routing infrastructure cannot identify the same multicast content that is forwarded based on different (technology–specific) addresses. Consequently, multiple distribution trees are constructed, which transmit content in parallel although traffic may be merged.

Following the observation by John Day "Multicast addresses is a set of distributed application names" [2, p. 329]. Using any application name on the socket layer to identify multicast content, however, would complicate the mapping process. Names

that equal syntactically may belong to different namespaces and thus applications. The current multicast socket API does not distinguish between naming and addressing. Naming can easily be implemented by introducing a new identifier, which is independent of the address. The identifier will be handled by the socket layer. In higher programming languages such as Java, this is realized by passing the host name to the socket.

In unicast, a DNS name is used as indirection to hide current technologies. The name may be associated with multiple addreses, which are resolved during run–time. Based on a simple probing mechanism, the receiver may check which address (and technology respectively) is currently available at the source. In multicast, however, it is common practice that addresses are not stored in the DNS. In addition, multicast receivers cannot probe for current multicast technologies at the source as there is no feedback channel. As a consequence, the receiver would subscribe to all addresses, which may result in redundant traffic in case of ASM. A simple DNS–based naming, thus, is not feasible for multicast to cope with different distribution techniques.

A very simplified option to represent the name is its encoding as string without characterizing special parts. The string must be parsed. This plain syntax relocates the mapping problem completely to the socket layer. Typically, multicast groups are named with respect to the current application type, which introduces a context. For example, a multicast group in SIP differs from an overlay identifier. A mapping function may benefit from the meaning of the multicast group if it includes an indication on the distribution technology available.

The meaning of the group name can be regained by the identification of the corresponding namespace. The association of the group name with the namespace allows for context–dependent mapping as well as the distinction of syntactically equal group names. In general, the namespace can be used to indicate the context of distribution, or the type of signaling used for multicast transport. Application programmers apply high–level meta data types for easier implementation and robustness, which implicitly determine the namespace, but must be defined in advance. This is inflexible. A generic data type that includes namespace awareness is a URI.

III. A Naming Scheme for Multicast

To encode multicast group names, we propose the following URI scheme:

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

The *scheme* refers to the specification of the assigned identifier (e.g., ip or sip), *group* denotes the group ID, *instantiation* identifies the entity that generates the instance of the group, *port* identifies a specific application, and *sec–credentials* are optional to implement security. Valid group IDs will be ip://224.0.1.1:4000 and sip://snoopy@peanuts.com, for example.

There is no need to distinguish between different IP versions based on the scheme identifier. IP addresses are denoted using a self–consistent syntax that allows for distinction (e.g., colons and square brackets in IPv6).

The proposed syntax of the group name provides a consistent term for ASM as well as SSM groups on the application layer. For example, *ip://224.10.20.30@1.2.3.4:5000/groupkey* describes a SSM group name, where *ip://224.10.20.30:5000/groupkey* denotes a source–independent multicast group.

The syntax also allows for flexible namespace support. The group name is defined by the multicast source. A multicast source is enabled to indicate a preferred forwarding scheme using a namespace that corresponds to a distribution technology.

Along this line, a multicast application developer opens a URI–aware multicast socket without predefining the distribution technology. In the case of a receiver subscription, for example:

```
ms=createMSocket()
ms.join(URI(
  "ip://224.10.20.30@1.2.3.4:5000/groupkey"
))
```

The socket can be used for another multicast group, as well:

```
ms.join(URI(
  "sip://hypnotic-talks@psychic.org"
))
```

IV. Conclusion & Outlook

Multicast introduces a new complexity with respect to its implementations at different layers and technologies. It is a characteristic example for service pluralism as we may face in future Internet scenarios.

In this paper, we shortly reviewed the challenges and requirements to abstract the interface between application programmer and network stack. We introduced a new naming scheme for multicast to implement technology–agnostic applications. This is part of a common multicast API [1], which we actively discuss in the IETF/IRTF.

Currently, we develop a hybrid multicast network stack that implements the presented naming and addressing for multicast. In future work, we will elaborate a lightweight mapping mechanism between names and addresses for group communication.

References

[1] M. Waehlisch, T. Schmidt, and S. Venaas, "A Common API for Transparent Hybrid Multicast," IETF, Internet-Draft – work in progress 02, March 2010.

[2] J. Day, *Patterns in Network Architecture*. Upper Saddle River, NJ, USA: Prentice Hall, 2008.