A Common API for Transparent Hybrid Multicast

(draft-waehlisch-sam-common-api-02)

Matthias Wählisch, Thomas C. Schmidt

Stig Venaas

{waehlisch, t.schmidt}@ieee.org,
stig@cisco.com

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Freie Universität Berlin

link-lab

CISCO

Recall (1): Problem Statement

- Group communication is implemented on different layers and is based on different technologies
  - This results in several forwarding paths and varying group addresses (namespaces)

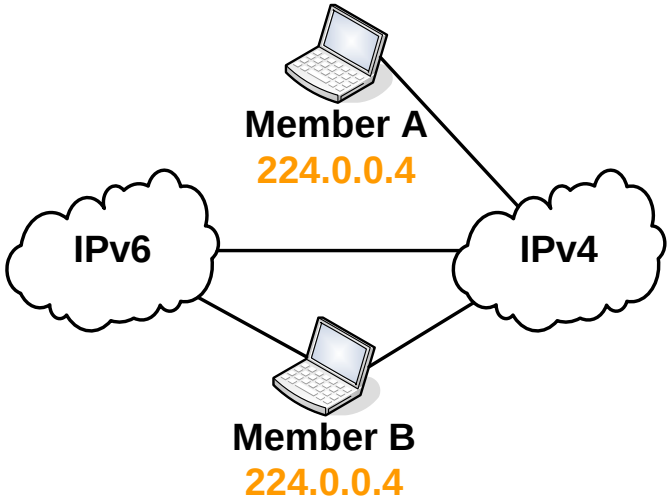
Objectives:

1. Enable any application programmer to implement independently of underlying delivery mechanisms
2. Make applications efficient, but robust w.r.t. deployment aspects

Recall (2): What is the Draft About?

- The current draft provides
  - a common multicast API on app. layer that abstracts group communication from distribution technologies
  - abstract naming and addressing by multicast URIs
  - mapping between naming and addressing
  - definition of protocol interaction to bridge multicast data between overlay and underlay

Example (1)



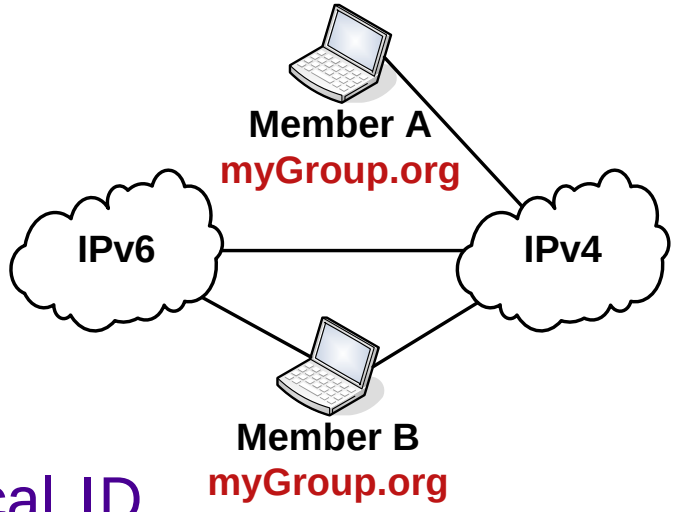
Current multicast API in Java:

- App. A and B decide on technology during programming
- MulticastSocket sock = new MulticastSocket(mcPort);
- sock.joinGroup(InetAddress.getByName("224.0.0.4"));

What happens underneath?

- Group management and data forwarding is based on “224.0.0.4”

Example (2)



Common multicast socket API:

- App. A and App. B subscribe to logical ID
- MulticastSocket sock = new MulticastSocket(mcPort);
- sock.joinGroup("myGroup.org");

What happens underneath?

- Mapping of "myGroup.org" on a technology identifier
- Group management and forwarding is based on a separate technology-specific identifier

Why do we Focus on Multicast?

- IP layer multicast is not globally deployed
- Several technologies around to bridge inter-domain deployment problem
  - application layer multicast, ...
- Common high level API abstraction is required
  - to produce code that runs in any environment
  - to construct hybrid solutions

Status

- Version 00/01 presented at IETF 76, Hiroshima
- Current version: 02
- Individual feedback received on version 02
  - Feedback by the RG is highly appreciated!
- Work on prototype implementation (C++)

Changes from Version 01 to 02 (1)

1. Document restructured to clarify the realm of document overview and specific contributions such as naming and addressing
2. A clear separation of naming and addressing was drawn. Multicast URIs have been introduced
   - Now, namespace is bound to Group ID
3. Clarified and adapted the API calls

Changes from Version 01 to 02 (2)

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

Terminology

- **Group Name**: application identifier that is used by applications to manage a multicast group
- **Group Address**: routing identifier that is used to distribute multicast data
- **Interface**: forwarding instance of a distribution technology on a given node

Group Name (1)

- Applications subscribe to Group Name(s)
- Group Communication stack maps Name to Group Address

How do we encode the Group Name?

- Wise choice is important for mapping function
- Variant A: Applications use pure string representation
- Variant B: Applications use data type that reflects namespace

Group Name (2)

- Typically, library that implements API provide high level data types
- Using such data type would implicitly determine the namespace
- A meta-data type that reflects identifier + namespace is an URI

Proposed URI Scheme

- `scheme "://" group "@" instantiation ":" port "/" sec-credentials`
- `scheme`: specification of assigned ID
- `group`: identifies the group
- `instantiation`: ID of the entity that generates the instance of the group
- `port`: ID of a specific application at a group instance
- `sec-credentials`: used for optional authentication
- Example: ipv4://224.0.0.22@1.2.3.4:5000/groupkey

Socket Options

- `getInterfaces(out Interface[] i)`
  - Returns a list of all available multicast comm. interfaces
- `addInterface(in SocketHandle h, in Interface i)`
  - Adds a distribution channel to a socket
- `delInterface(in SocketHandle h, in Interface i)`
  - Removes an interface from the socket
- `setTTL(in SocketHandle h)`
  - Defines maximum hop count
- Something else?

Open Issues

- Definition of mapping scheme including ASM/SSM consideration
- More detailed description of the URI scheme usage
  - Use case for security credentials
- Description of corporate usage of current and common group communication API
- Consensus that technology discovery out-of-scope?
- Regarding implementation: Are there any convenient functions that you would like to have?

Conclusion

- Clear separation between Group Name and Address
- Group Name data type: URI
- Next step: defining mapping mechanism
- More feedback is needed by RG members!