Common API for Transparent Hybrid Multicast draft-irtf-samrg-common-api - Status Update –

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

History of the Draft

- Version 00/01 presented @ IETF 76, Hiroshima (Nov. 2009)
- Adopted as WG document @ IETF 79, Beijing (Nov. 2010)
- Update version 03: Major update on namespace and mapping + Many clarifications and editorial improvements
- Currently published version: 04
  - Considers comments by WeeSan and Mario Kolberg
  - Section “Illustrative Example” + “Implementation” added
- RGLC February 2012
  - Several comments by John and Sebastian
  - Discussions about MTU issues

Comments by John (1)

- Please add language that specifies whether group names are intended to be unique (by namespace domain, by all domains, across all active sessions, across all past present or future sessions, other)?
  - Group Name is globally unique by all domains
- Is the application responsible for selecting group names and determining uniqueness?
  - Group Names can be chosen by the end user, which component ensures uniqueness is out of scope of the document
- Clarification of service discovery: Implementation-specific, addressed in another spec., or use existing mechanisms

Comments by John (2)

- Scheme extension: Is “sips” a possible scheme?
  - Yes.
- Is the `receive` call blocking?
  - Yes, in the reference implementation.
- How is mobility across multicast domains supported?
  - Hiding mobility by Group Names
- Support of multi-resolution multicast and XCAST
  - Next slides …

How does the API cover multi-resolution Multicast?

- General objective of multi-resolution multicast: Consider heterogeneous end devices
- Idea: Utilize the mapping from names to addresses
- Option 1: Different *addresses* for different quality
  - Domain-specific mapping
- Option 2: Pre-defined *naming* syntax
  - Example for a layered video:
    $EL_j.Q_i$`.blockbuster` denotes each individual layer

Support of XCAST?

- XCAST idea: Source explicitly pre-defines receiver
- Perspective of the API
  - XCAST is another distribution technology: new Interface
  - scheme://group@instantiation – instantiation denotes one or multiple *sources* – one may encode receivers using an explicit XCAST scheme
  - Receiver Ids are technology specific which would contradict abstraction approach of names

Comments by Sebastian

- At the moment error handling is quite limited and just indicates success or failure of send/recv operation. From a developers perspective this might be a little coarse, as there is no information on what to do about the error. A common problem would be the **size of data (message),** that an application would like to send within one send call.
  - Next version of the draft includes error codes for send/receive calls that indicates that message is too long

MTU Issues

- Current version does not provide mechanisms to handle MTU problems

Challenge:

- API considers a multi-technology scenario
  - Different technologies have different MTU sizes
  - MTU size may change over time
  - Interfaces may change dynamically (mobility)
- No more complexity to the application programmer

A **too** Simple Approach

- Each interface signals currently supported MTU
- Programmer configures MaxMessageSize with socket creation

Problems:

- No appropriate interface(s) available
- Change of path MTU during transmission
  - Interfaces will be disabled
  - Prevents data delivery

Proposal

- Middleware implementation guarantees a maxMessageSize to the application programmer
- Fragmentation will be implemented by technology modules if necessary
- No incompatiblity to native IP clients due to realistic values for maxMessageSize

Consequences - Draft Changes

Call extensions:

- getMaxMsgSize() as Service Call
- getAtomicMsgSize() as Socket Option

Editorial extensions:

- Separate MTU Section that discusses the general problem scope