The Message Bus

Colin Perkins (c.perkins@cs.ucl.ac.uk)
Jörg Ott (jo@tzi.uni-bremen.de)
Dirk Kutscher (dku@tzi.uni-bremen.de)

draft-ietf-mmusic-mbus-transport-00.*
draft-ietf-mmusic-mbus-semantics-00.*
Motivation

• Provide a means for “local” coordination between conferencing applications
• To allow simple, ad-hoc cooperation between entities
• To help modular system design
• To allow integration of tools
System Model

A user’s representation in conferences:

• Multiple tools run by a single user
  – user interface, media engines, control, etc.

• Need not all be on a single host

• Composition yields conferencing system
  – unified appearance to the user
  – coherent reaction to user input
System Model

Media Engine

H.323 Engine

Mbus

Media Engine

SIP Engine

Common User Interface

Controller module
Addressing

• By media type
• By module type: parts of a tool
• Specific applications
• Specific instances
  – Multiple names per application possible
  – Process ≠ Mbus entity
• Wildcarding allowed (‘*’)  
→ (audio engine rat 4099@10.0.0.1)
Transport Mechanisms

- Link local multicast or directed unicast
- Reliability achieved via retransmission
  - for fully qualified addresses only
  - restricted msg scope makes this easy
- Per-source ordering property
  - because it’s link local
  - single unack’ed message per destination
Transport Channels

• Link local multicast address (or unicast address of destination)
• Well-defined base port per user
  – for cross conference interactions
• Separate port per conference
• Collision avoidance between users
  – hash functions + message digest
Security

• Digital signatures to prevent spoofing
• Encryption for privacy
• Secure end system assumed, protecting against outsiders
  – store the hash- and encryption-keys in a local file or the registry
  – may require manual configuration
Message Format

• Common format for all messages

<Message Digest>
mbus/1.0 <seq#> <ts> <type> <src> <dst> <acklist>
cmd (args)
cmd (args)
...

**ts:** time stamp

**type:** reliable/unreliable

**acklist:** piggybacked ACKs
Command Format

• Commands are split into categories
  – Specific to media, protocol, etc.

• Structure in command names indicates categories
  
  \textit{rtp.source.mute, audio.input.gain}

• Self-describing commands
  – allows processing/ignoring regardless of the entity/ies a command was sent to
Message Classes

• Currently defined
  MBUS.*
  RTP.*
  AUDIO.*
  CALL-CONTROL.*

• Further classes envisioned
  – security
  – control (capabilities?, floor?, policy?)
Example

314159fe3289e2718289834eba43746823...
mbus/1.0 42 1998 U
  (audio engine rat 1234)
  (audio ui meccano-ui 1235)
  (27 28)
audio.output.powermeter (75)
rtp.source.active (jo@10.0.0.2 100)
rtp.source.active (csp@10.0.0.1 100)