Integrating Sound and Multi-Media with X
MAS: The Media Application Server

Mike Andrews and Leon Shiman
05 September 2002
Tested Release Candidate 1
“Whirled Peas”

Target: January 1, 2003

for integration into X11R6.7, March 2003
The MAS Architecture is Different

- MAS is a sound and multimedia server
- Networking and RTP are fundamental
- More peer-to-peer than client-server
- Our core dependency is ANSI C89
- MAS anticipates the need for very complex assemblages
- Single-threaded, centralized scheduler
- Timing and performance monitoring are integral
- Required support for accessibility
The MAS Process
The MAS Architecture

- **Devices** are dynamically loaded MAS plug-ins
- Data is communicated between devices using MAS **ports**
- Work is done by **actions** that devices implement.
- Actions are triggered by the **scheduler**.
- The **assembler** manages the scheduler’s **event queue** for the set of devices in an **assemblage**.
- The **scheduler** acts on the queue of **events**.
- An **application** is a combination of devices and client processes.
The solution needs to consider X11’s features:

- Operable on many platforms
- Network transparent
- Open source core, allowing for proprietary releases
- Developed within a standards body
History and Goals

- Scale to the smallest and the biggest
- Everything is an extension - dynamic loading from the outset
- Functionality can scale as you need it
- Separation of data and control
- Enable very simple application construction out-of-box
- It needs to have good documentation.
The User Interface

GNOME, KDE, CDE...

MAS is agnostic.
MAS: Syntax and Semantics

server

client

process

application
Accessibility Requirements

- Controllable, low latency
- Stopping utterances quickly
- Multiplexing, with priorities
- High audio quality
- Small memory footprint
- Format independent media handling
- Synchronization of multiple media streams
Synchronization and Latency
A Basic Requirement

- ABC, CBS, NBC say no worse than audio 33ms lagging behind video or 16ms leading video
- EIA/TIA say -40ms, +25ms.
- Possible POTS delay: 50ms
- Satellite phones: 250ms
- Multiple audio sources, same room: discernible echoes after 55ms, reverberation between 10ms and 55ms, phase distortion less than 10ms.
The Real Time Protocol (RTP)
RFC 1889, January 1996

- It’s almost an Internet Standard
- Carries time-sensitive information: audio, video, events, etc.
- Timestamping in Network Time Protocol (NTP) format
- ... and, simultaneously, with media-specific clocks
- There’s no reason to re-invent the wheel!
RTP and MAS

- MAS uses RTP for all communications.
- The transport that carries RTP is unspecified. Currently, we’re running RTP over UDP, TCP, and UNIX-domain sockets.
- MAS control packets have their own RTP payload type.
- Media formats use pre-defined RTP profiles, when applicable. Several defined in RFC 1890, others in subsequent “RTP Profile” RFC’s.
- MAS doesn’t use RTP in a multicast mode, or even typically in a multi-participant session mode.
RTP and MAS

Remote machine
- MAS
- RTP/UDP
- RTP/TCP
- Pipe
- MAS client

Local desktop
- microphone
- speaker
- local sound interface
- RTP/UDP
- RTP/TCP

Digital audio/loss-tolerant media
Control

Linux-Kongress 2002
RTP and MAS

- Loss-tolerant digital audio or other media data.
  - Preferred transport
  - RTP + MAS extensions w/flow control
    - [RTP (a)]
    - UDP
  - Minimal RTP + MAS extensions w/framing
    - [RTP (b)]
    - TCP, UNIX socket, pipe, or other reliable stream.

- Scheduler protocol
- Assembler protocol
- plugin 1 protocol
- plugin 2 protocol
- ...
The MAS Protocol
Or, really, protocols...

- All entities in the system read and write structures with broken-out RTP headers and packet payloads
- We use a simple public interface for packing and unpacking
- But, you can use whatever you want.
- Device APIs and devices simply need to speak the same protocol.
- MAS comes with pre-defined protocols for its fundamental devices - CODECs, sources, filters, etc.
- These can be extended, or simply not used.
Simple “thru” Devices

- ** endian** - convert multi-byte linear audio samples to and from little, big, and host endian formats
- **codec_u-law** - convert 16-bit linear audio samples to and from the logarithmic, 8-bit u-law format.
- **channelconv** - convert mono to stereo and back.
- **squant** - quantize samples to other sample resolutions (e.g. 16-bit to 8-bit) with selectable noise shaping.
- **srate** - resample linear audio at a different sampling rate (e.g. 44100 Hz to 48000 Hz), selectable quality.
Virtually Expanding Devices

- **mix** - Mixes N linear audio streams down to one audio stream.
- **split** - Generates N audio streams from one audio stream.
The Network Device

- Moves data to and from network ports and MAS ports
- Connects to other servers
- Accepts connections from MAS client applications and other MAS processes
A Complete CODEC Has Three Devices

MPEG-1 Audio (including layer 3)

- **mp1a_source** - reads from a local file, stuffing an integral number of MPEG frames in each packet. It understands the inter-frame timing relationships.

- **mp1a_codec** - decodes incoming frames to their original format (e.g. 44kHz, 16-bit, linear, stereo) or encodes linear audio to the selected MPEG-1 format.

- **mp1a_sink** - receives packetized MPEG-1 frames, writes them to disk.
**ppm** - (Peak Program Meter) is a “thru” device that posts audio signal peak information to a source port - it can be connected to a client via the net device or to another device in the server. Adjustable integration and decay time.

**visual** - X11 visualization device, with spectrum analyzer mode. Linked directly with Xlib, it opens a window on the local X server.
Control-Only Devices

- **tag** - given an mp3 filename, supplies the ID3 tag metadata.

- **interact** - organize an N-way conference: receives join or leave actions from participants, manages connections to everyone’s mixers from everyone’s sinks.
N-Way Conferencing With a Hub Server
N-Way Conferencing (continued)
N-Way Conferencing (continued)
MAS Interfaces

for application and device programmers

- Application Programmer’s Interface (API)
- Device Programmer’s Interface (DPI)
- Device or device-class specific APIs
The MPEG 1 audio source device (mas_source_mp1a):

- Calls MAS DPI functions inside the server: `masd_post_data, masd_get_state`...
- Implements functions (actions) required by MAS: `mas_dev_init_instance, mas_dev_configure_port`...
- Implements MAS standard actions that are understood by the MAS API: `mas_get, mas_set`.
- Implements MAS device-class standard actions that are understood by the mas source device API: `mas_source_play, mas_source_stop`...
Metering Assemblage
peak program meter & sample rate converter
Two Application Assemblage
with a function generator
Security Concerns

- Open audio device for read/write
- File access: reading for streaming servers, writing for recording
- Real-time scheduling
- Trusted plug-in device signing
- TCP and UDP networking concerns
- The usual string problems
MAS Protocol Tunneling
Avoiding New Problems

“X works, but I don’t get any sound!”

• The user’s X11 client and server are on opposite sides of a firewall.

• The user’s X11 TCP/IP connection is tunneled through a secure shell connection.

• The user’s X11 TCP/IP connection is proxied with LBX.
MAS Protocol Tunneling
Solution

- Extensible connection fallback mechanism.
- If X11 is present, MAS can use XClientMessages to proxy some control information using the X server.
- The MX extension will allow the X server to proxy more data for MAS.
- Best case: TCP control connection, UDP or TCP for media transport. Can be tunneled through SSH.
- Middle: using the MX extension to carry MAS protocol over X11/TCP.
- Worst: ClientMessages for control/media connection.
MAS Protocol Tunneling

Remote machine

MAS

X11
X11
XClientMessages/TCP

Pipe

optional
LBX proxy

X11

Firewall

Local network

Local desktop

X Server

X11
XClientMessages/TCP

port 6000-6063

Display, mouse, keyboard, etc.

MAS

local sound interface

speaker

microphone

MAS client with
X11 GUI

Local network

 MAS Protocol Tunneling

Remote machine

MAS

X11
XClientMessages/TCP

Pipe

optional
LBX proxy

X11

Firewall

Local network

Local desktop

X Server

X11
XClientMessages/TCP

port 6000-6063

Display, mouse, keyboard, etc.

MAS

local sound interface

speaker

microphone

MAS client with
X11 GUI
The Development Process

- X.Org is partially supporting our work on the X11 tunneling mechanism and some other parts of our development effort.
- X.Org has played a fundamental role in the development of the MAS standard.
- Sun has independently supported several parts of our design and development effort.
- We’ve set up a project website and will be setting up a device registry and repository.
- We’re continuing to develop the MAS core, devices, and applications.
Credits

Mike Andrews
Garrett Banuk
Kaleb Keithley
Denis Marcin
Silvio Neef
Brian O’Brien
Leon Shiman
Ruth Shiman-Hackett

and the X.Org Audio Task Force Members
We Will Discuss

- History and Requirements
- The MAS Architecture
- Accessibility
- The Real Time Protocol
- Plug-in Devices and Assemblages
- MAS/X11 Interoperability
- Benchmarks
- The Development Process
- The Release Timetable
This is a Project, not a Product

After many legal issues, it’s open!

The project website:
http://www.MediaApplicationServer.net/

Development mailing list:
mas-devel@alex.shiman.com