MAS: The Media Application Server

Mike Andrews and Leon Shiman

5 April 2002

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

We Will Discuss

- History and Goals
- The MAS Architecture
- Accessibility
- The Real Time Protocol
- Plug-in Devices and Assemblages
- MAS/X11 Interoperability
- Benchmarks
- The Development Process

History and Goals
In the Beginning

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

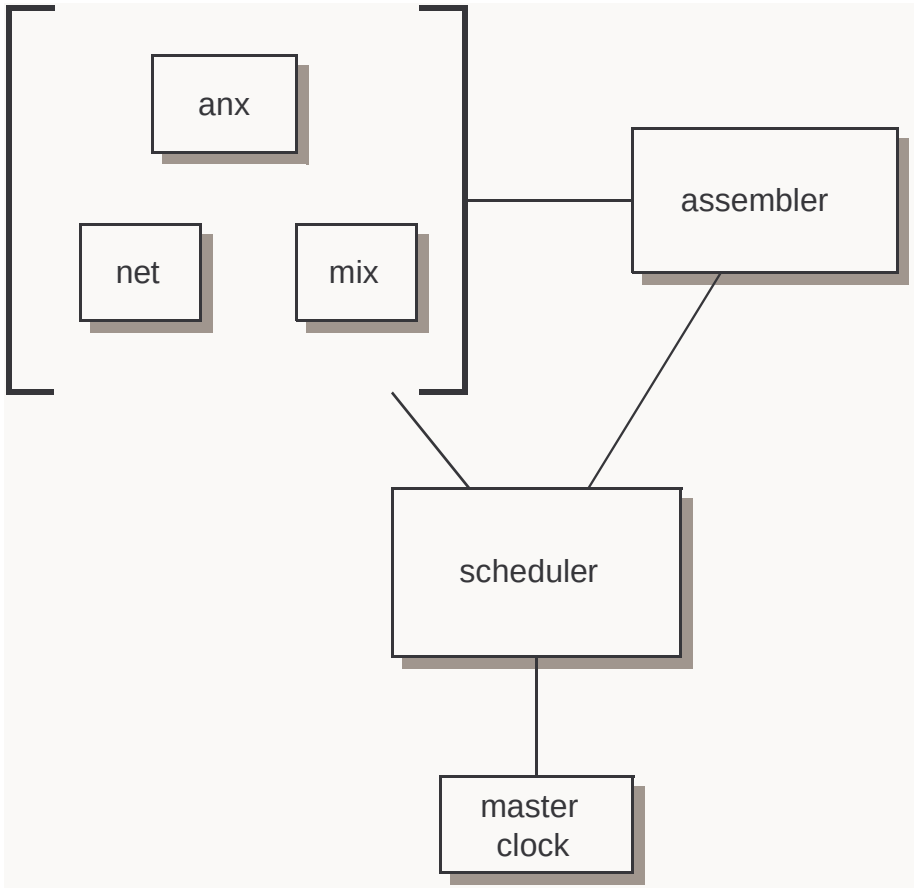
- Scale to the smallest and the biggest
- Everything is an extension - dynamic loading from the outset
- Functionality can scale as you need it
- Separation of data and control
- Enable very simple application construction out-of-box
- It needs to have good documentation.

The Desktop

GNOME, KDE, CDE?

We’re agnostic.

The MAS Model



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 program and sets of devices, called **assemblages**.
- The scheduler acts on a queue of **events**.

Accessibility
Requirements

- Controllable, low latency
- Stopping utterances quickly
- Multiplexing, with priorities
- High audio quality
- Small memory footprint
- Synchronization of multiple streams

Synchronization and Latency

- 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: discernable echoes after 55ms, reverberation between 10ms and 55ms, phase distortion less than 10ms.

The MAS Architecture
Distinctions

- Networking and RTP are fundamental
- Client-server and server-server
- Our core dependency is ANSI C89
- We allow for arbitrarily complex assemblages
- Timing and performance monitoring are integral

Interfaces

- Application Programmer’s Interface (API)
- Device Programmer’s Interface (DPI)
- Common Interfaces
- Individual device APIs

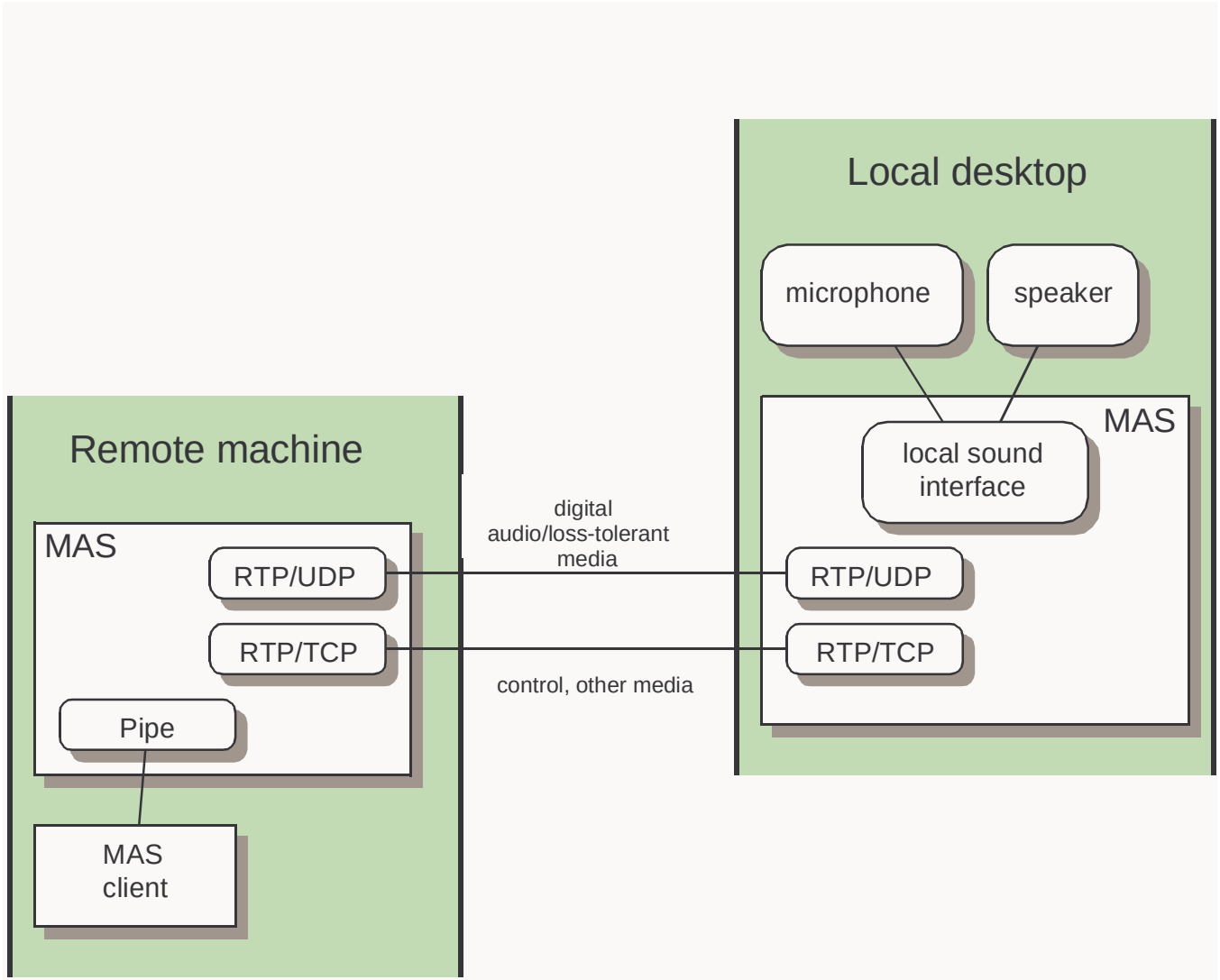
The Real Time Protocol (RTP)
RFC 1889, January 1996

- 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!

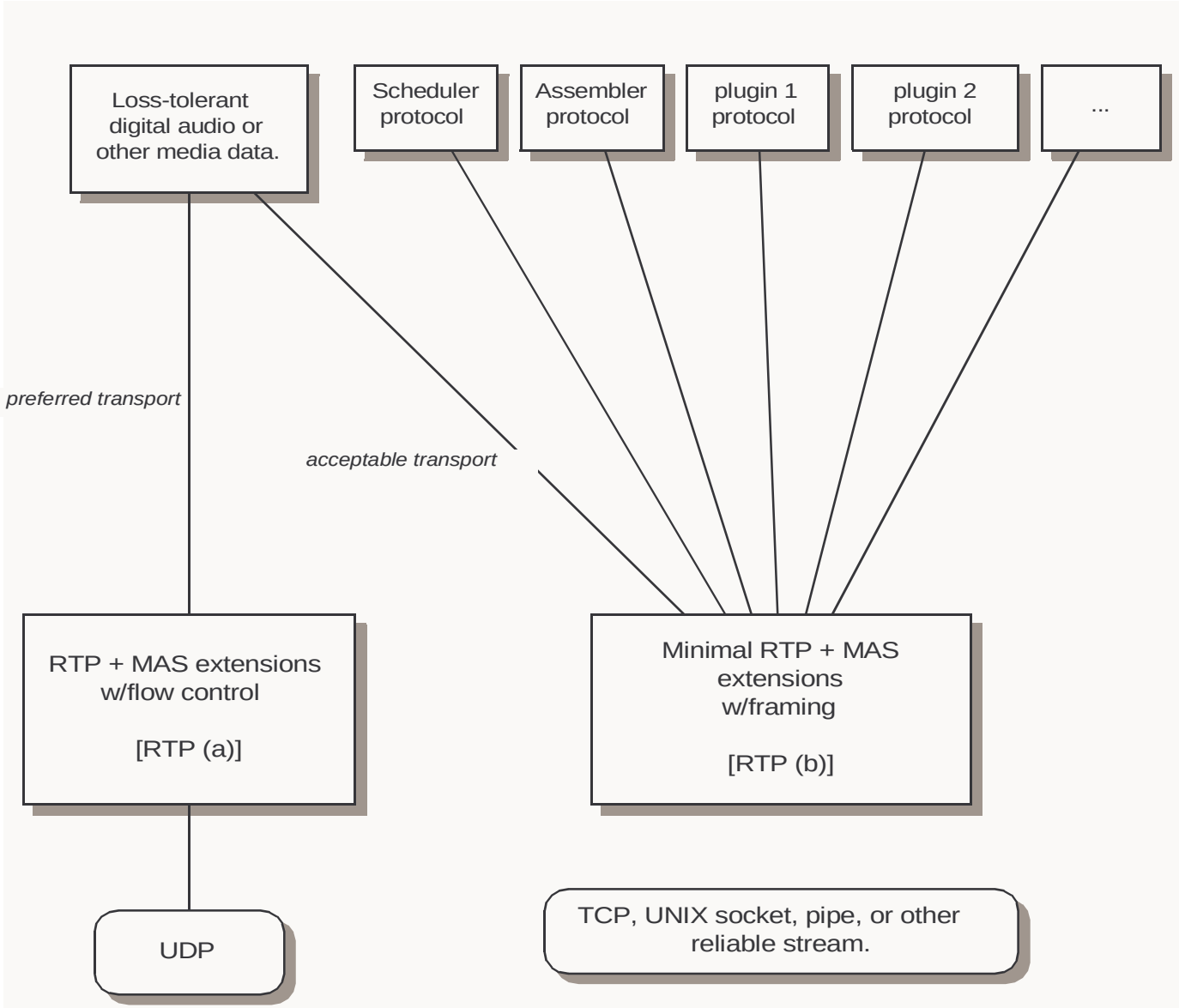
RTP and MAS

- 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


Local desktop
microphone
speaker
MAS
local sound interface
RTP/UDP
RTP/TCP
digital audio/loss-tolerant media
control, other media
Remote machine
MAS
RTP/UDP
RTP/TCP
Pipe
MAS client

RTP and MAS


Loss-tolerant digital audio or other media data.
Scheduler protocol
Assembler protocol
plugin 1 protocol
plugin 2 protocol
...
preferred transport
acceptable transport
RTP + MAS extensions w/flow control
[RTP (a)]
Minimal RTP + MAS extensions w/framing
[RTP (b)]
UDP
TCP, UNIX socket, pipe, or other reliable stream.

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.

Net Device

- Moves data between network ports and MAS ports
- Connects to other servers
- Accepts connections from clients and servers

MPEG Audio

- Split into two devices: file reader (source) and CODEC.
- The source reads from a local file and stuffs an integral number of MPEG frames in a packet.
- The CODEC decodes incoming frames to their original format (44kHz, 16-bit, linear, stereo)

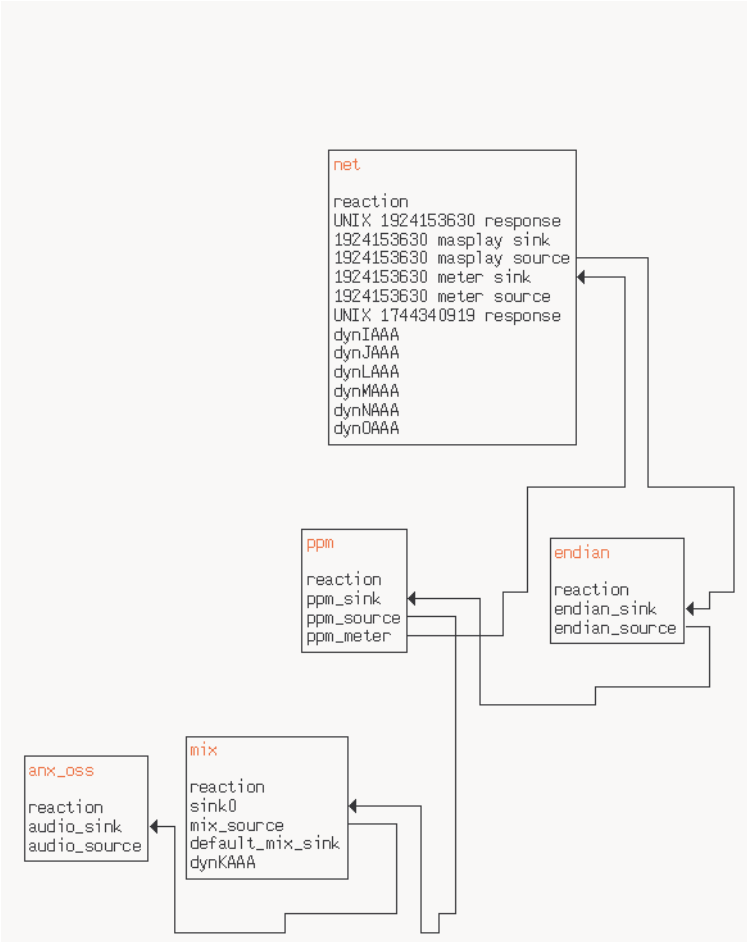
Peak Program Meter

- Detects peaks in the audio signal (there is an IEC spec. for this)
- Sends peak information out a source port that can be connected to the net device to send back to a client, or to another device in the server.
- Adjustable integration and decay time.

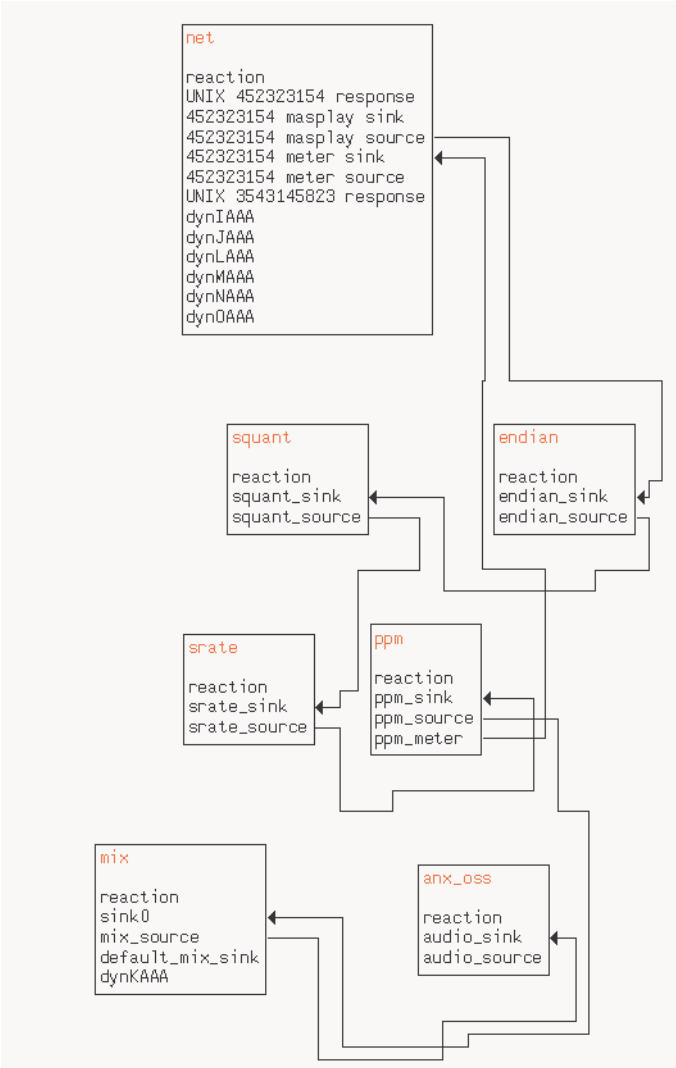
Mixer

- Mixes multiple sources of audio together to form a single stream.
- Has a “replicating” sink port that can be connected to multiple times, creating a new port to replace it.

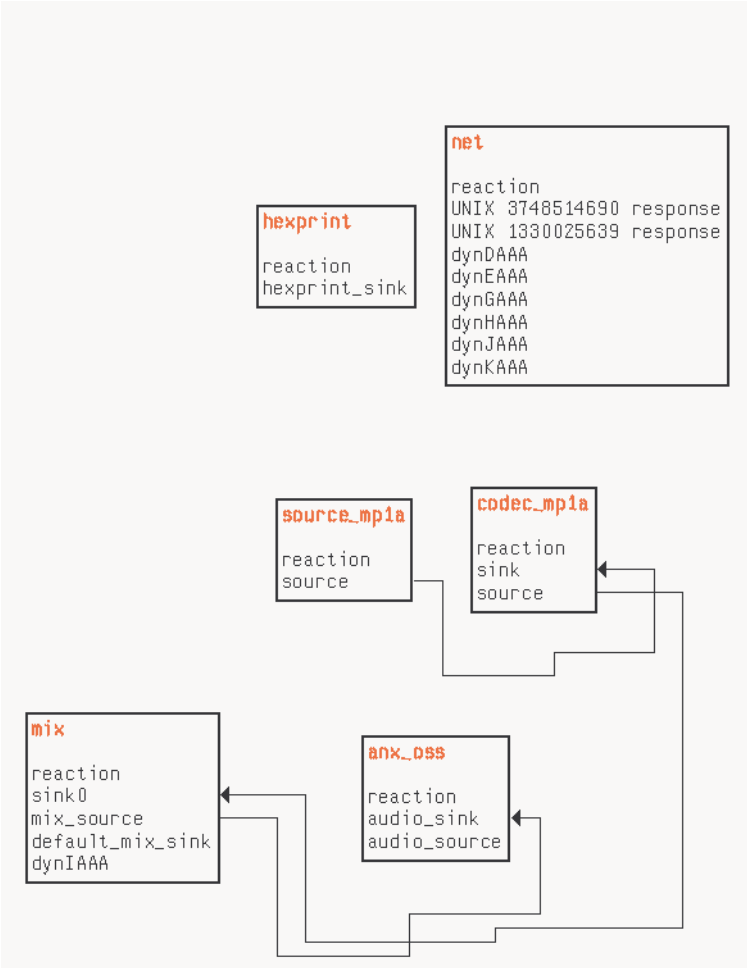
Peak Program Meter
In Use



Sample Rate Converter and Quantizer
In Use



MPEG Audio Source/CODEC
In Use



Benchmarks

```
library file: libmas_srate_device.so
          name: srate
     interface: srate
       summary: MAS Sample Rate Converter
   description:
        vendor: X.org
    vendor_url: http://www.x.org/
     copyright: Copyright 2001, Shiman Associates, Inc.
       license: MIT
    build_date: Nov  2 2001
    build_host: code1.shiman.com
       version: 1.0.0
       actions: mas_dev_init (0x401ca890)
                mas_dev_configure_port (0x401caaf0)
                mas_dev_show_state (0x401ca984)
                mas_srate_convert (0x401cad08)
  action stats: mas_dev_init
                     sum: 43.000000
                   count: 1
```


```
                mean: 43.000000
                 min: 43.000000
                 max: 43.000000
           mas_dev_configure_port
                 sum: 74.000000
               count: 2
                mean: 37.000000
                 min: 35.000000
                 max: 39.000000
           mas_dev_show_state
                 sum: 0.000000
               count: 0
                mean: 0.000000
                 min: 300000000.000000
                 max: 0.000000
           mas_srate_convert
                 sum: 2281145.000000
               count: 7585
                mean: 300.744232
                 min: 9.000000
                 max: 902.000000
static ports: ("srate_sink", MAS_SINK, mas_srate_cmatrix)
           ("srate_source", MAS_SOURCE, mas_srate_cmatrix)
```


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

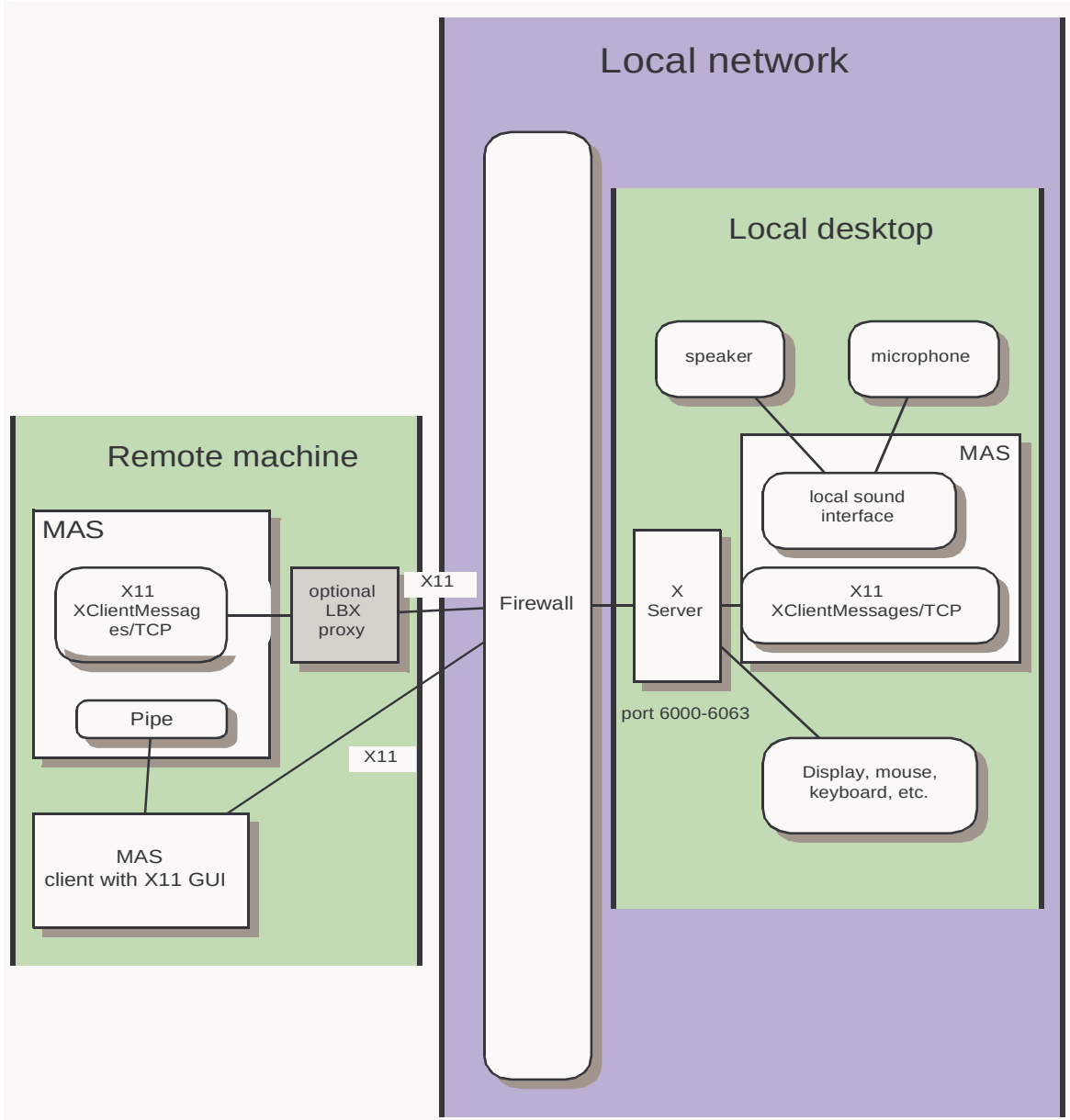
"X works, but I don't get any sound!"

- 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



The Development Process
You, us, and X.org

- X.org is funding us to work on the X11 tunneling mechanism.
- We have a proposal in X.org now for funds to support project management, documentation, and porting.
- We’re setting up a device registry/repository.
- We’re going to continue to develop the MAS core, devices, and applications on our own hook.
- We need application and plug-in developers!

Credits

Mike Andrews

Garrett Banuk

Kaleb Keithley

Denis Marcin

Silvio Neef

Leon Shiman

and the X.org Audio Task Force Members