The Broadway Audio System

Ray Tice

ray@x.org

Mark Welch

mw@x.org

X Consortium Inc.

Introduction: Problem Statement

- Lack of standardized interfaces to audio services
- Lack of standardized network protocols

Goals

- Platform independence
- Hardware independence
- Network transparency

Non-goal

- Be all things to all people

Targeted applications

- Basic record and playback
- Audio on the web
- Playing synchronized audio/video clips
- Teleconferencing
- Unix and Windows

Agenda

- Overview
- Time model
- API & code examples
- Security model
- Event model
- Extensions

Achieve goals by

- Requiring core set of services
- Allowing expansion by layering and extension
- Providing conveniences for targeted applications (make simple things simple)

Leverage existing and future technologies...

- Compression formats
- File formats
- Multicasting
- Time-critical transport
  - RTP
  - “Open” RealAudio

...without requiring them

Technologies included

- Device sharing
- Rate & format conversion
- Explicit time model (from AF)
- Synchronization
- Caching of sound clips

Technologies excluded in first release

- Generalized DSP and filtering
- Complex synchronization constructs
- Generalized analog routing and filtering

Design philosophy

- Keep learning curve low
  - Make simple things easy
  - Make complex things possible
- Organize
  - Shield novices from complexity
  - Help experts find things

Architecture

- Client-server, like X11
- Connection based -- uses ICE protocol
- Defined C API, like Xt
  - Low-level and convenience services
- Defined network protocol

Library responsibilities

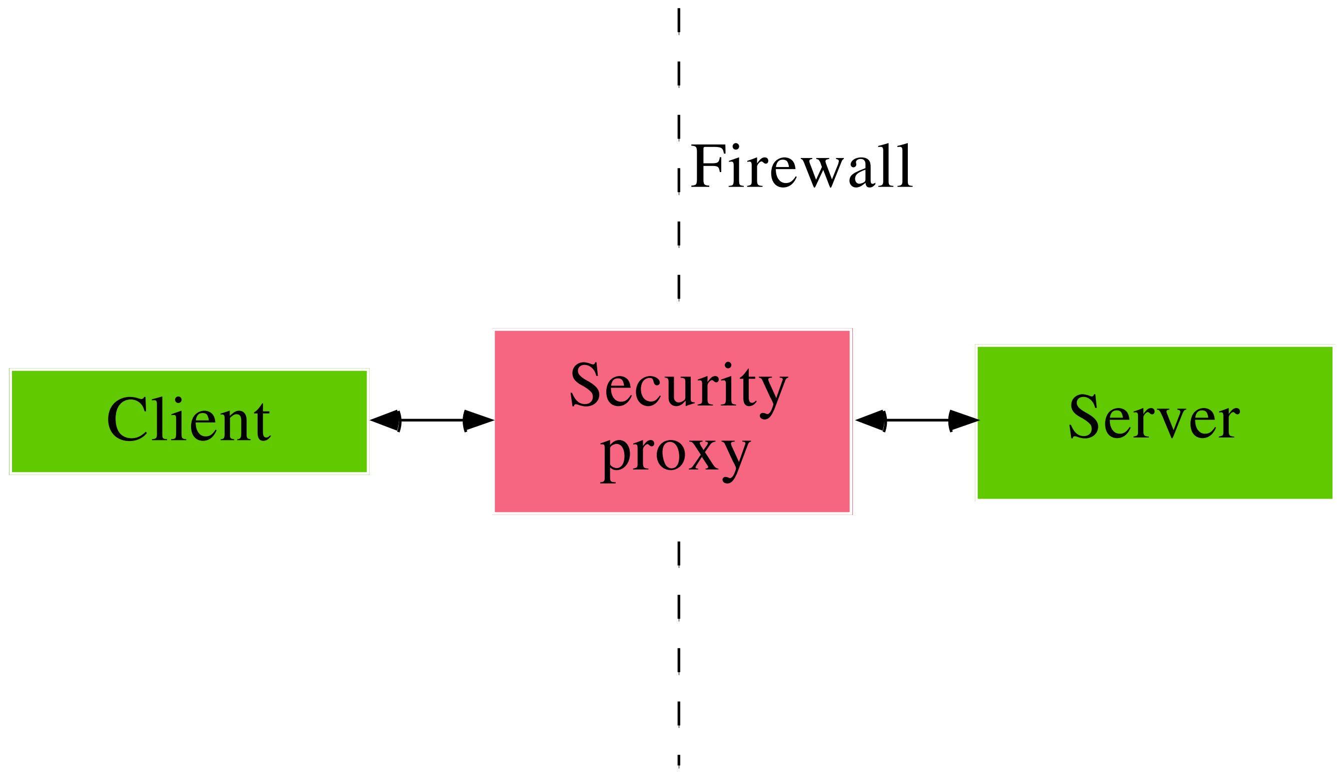
- Interface with application
- Send requests from client to server
- Receive events from server and dispatch
- Handle file formats
- Move audio data to server

Server responsibilities

- Provide hardware independent programming model
- Share audio services among clients
- Perform rate & format conversion
- Synchronization
- Access control

Internet considerations

- Security proxy handles firewall crossing



Object model

- Model resource allocation in server -- more formal about objects than X11
- Organize related attributes

Object model

- Class (single inheritance)
- Object
- Operations: Create, Destroy, Get, Set, Read, and Write

Analysis of object model

- Works over client/server connection
- Reduces interface size by using canonical requests
- Applicable to similar problems
- Tuned to meet our specific needs

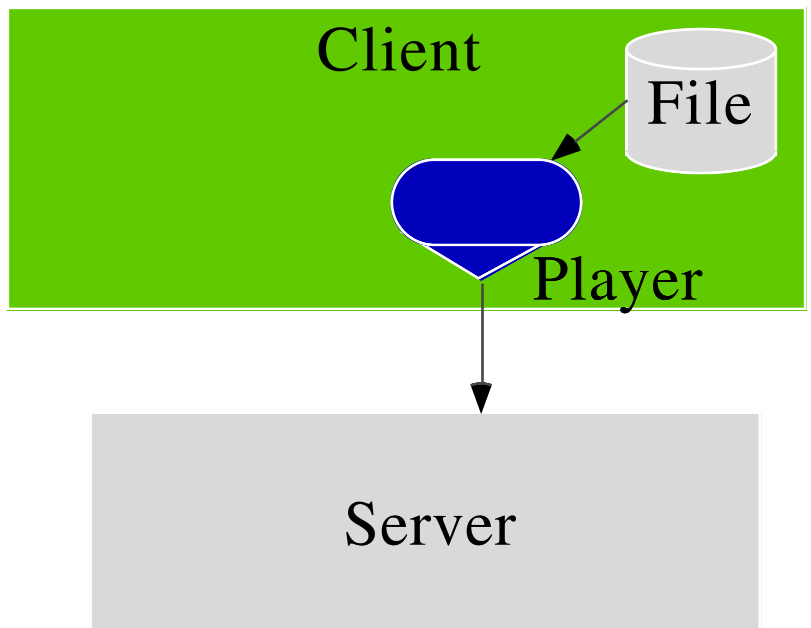
Client side overview

- File objects
- File players
- Others

Example: Playing from file

- Open audio services
- Create file object (given file name)
- Create player object (given file object)
- Call main loop

Example: Playing from file



Server side: Devices

- Represents a piece of input or output hardware (microphone or speaker and the corresponding ADC or DAC)
- Does not represent signal processing (DSP)
- Unidirectional

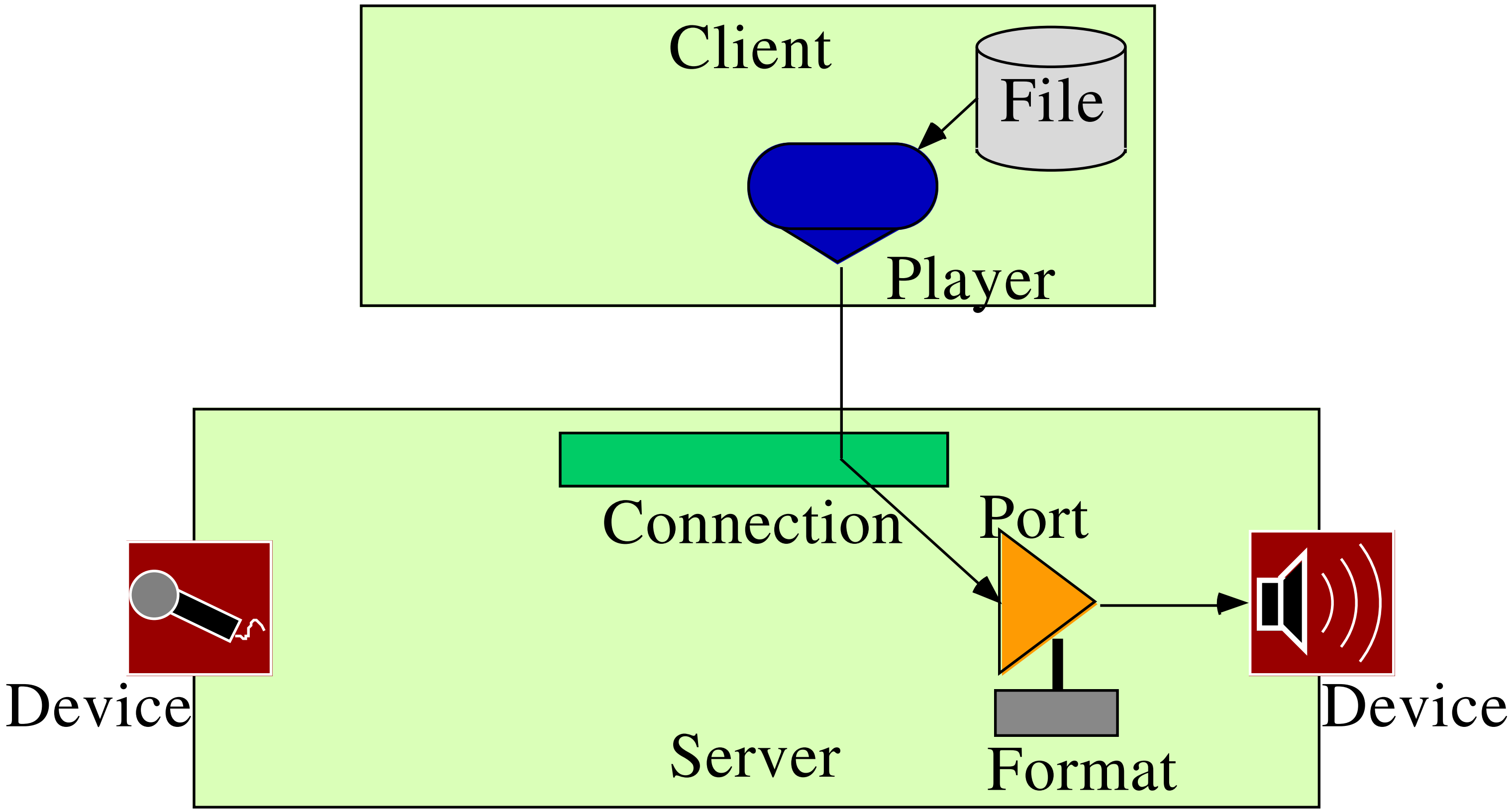
Ports

- Client access to buffers/buckets/devices
- Format conversion (decompression / resample / etc.)
- Synchronize source to device time
- Over/underflow policy

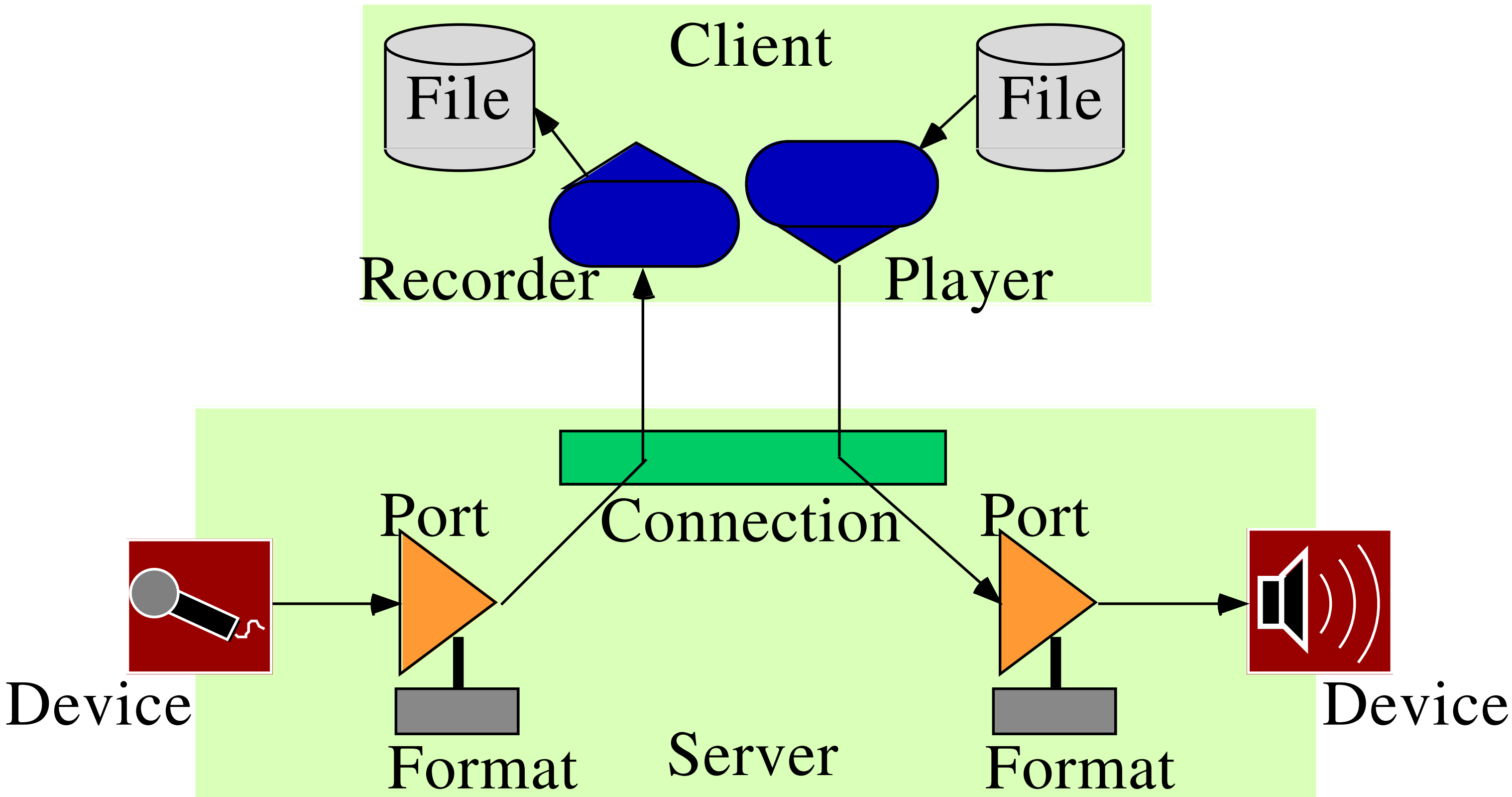
Formats

- Define characteristics of audio data:
  - Encoding (linear, u-law, a-law, etc.)
  - Sample rate (8000, 44100, 48000, etc.)
  - Other attributes (sample size, number of channels, endian)

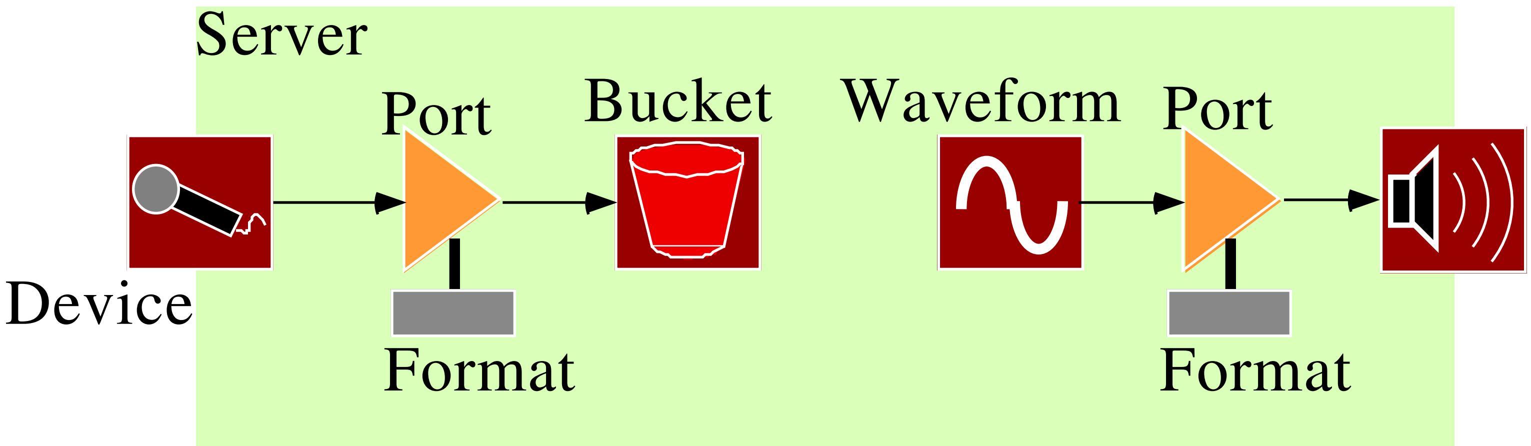
Example: Playing from file



Example: Record/playback



Buckets & waveforms



Other housekeeping

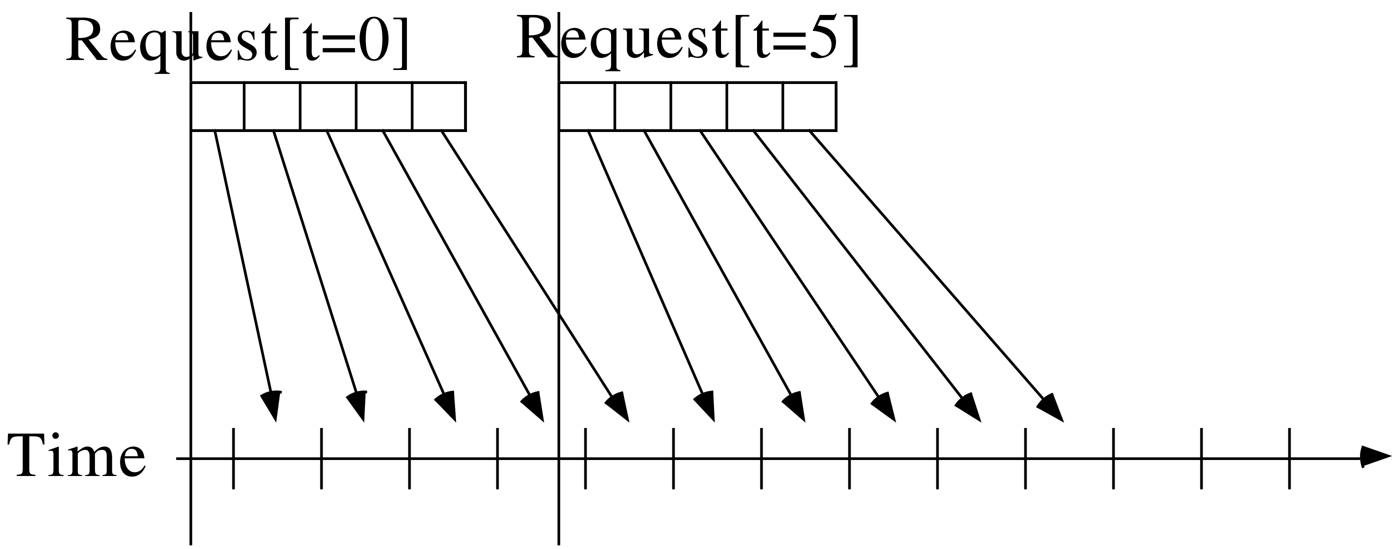
- Security
- Connection management

Time model

- A timeline has 1 tick per sample
- Separate timelines for the incoming data stream and device
- The port object resolves the difference in timelines
  - rate conversion
  - adding or dropping samples

Time model – Transport delay

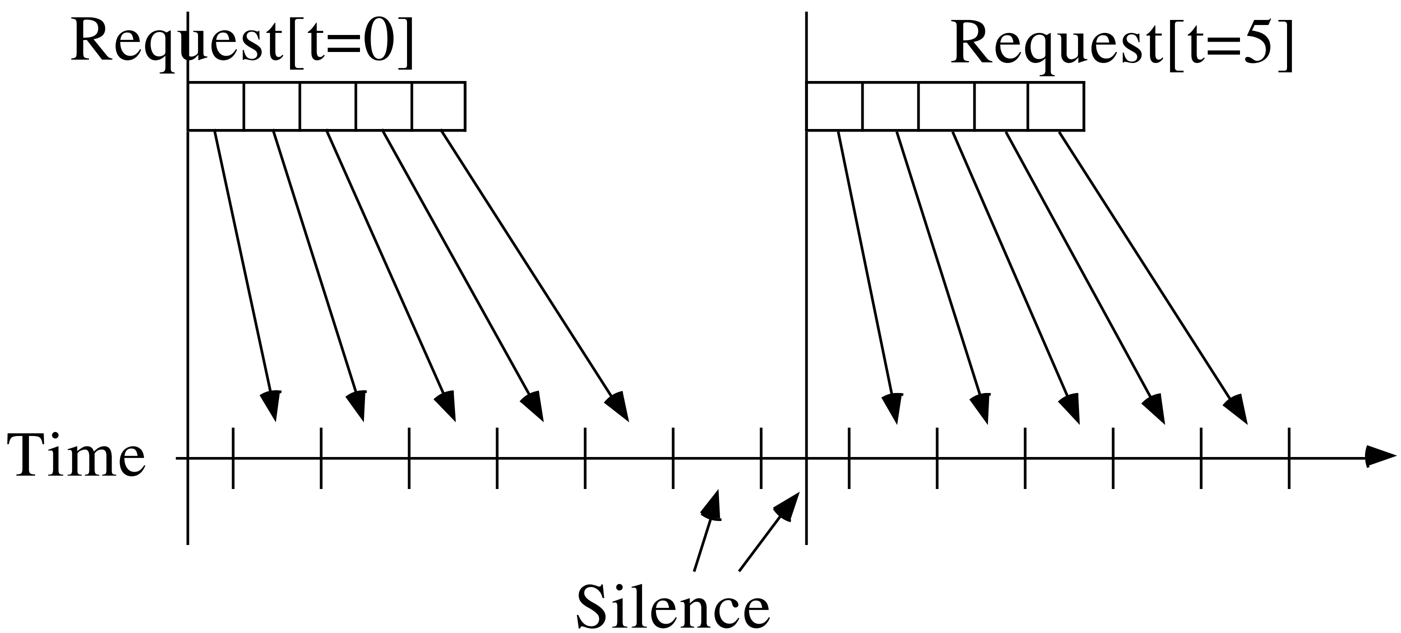
- Normally, requests arrive ahead of the time at which they need to be played



Time model – Transport delay

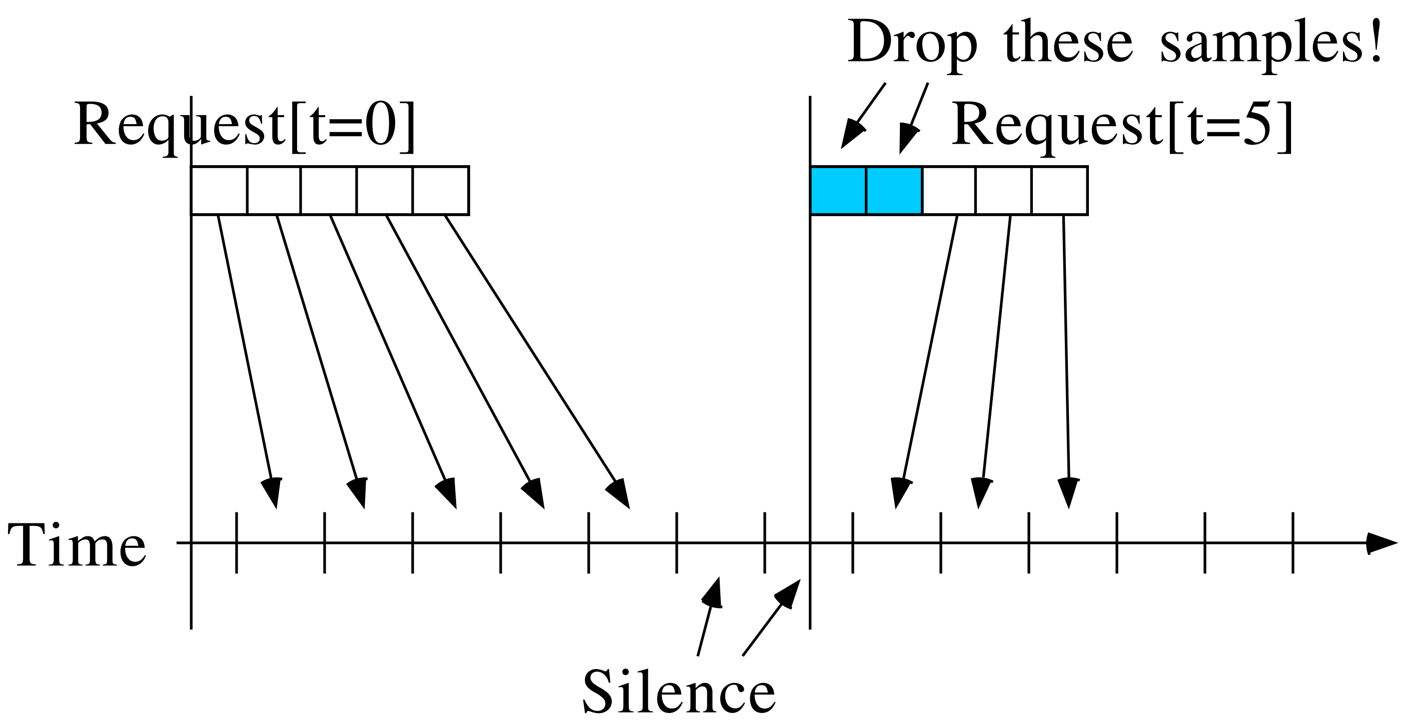
- Two ways of handling transport delay:

  1. Stalling



Time model – Transport delay

2. Dropping samples from incoming data



Synchronization with other media

- Low level in first release
- Video or graphics are synchronized to audio
- Inside X server, audio timeline presented as an XSync counter
- More complex synchronization possible in the future

API

- Most API encapsulated in 8 basic calls
- Objects referenced by tag, not pointer
  - Hides implementation, storage details
  - Objects in either client or server

API: Connections

- XaOpenAudio: open a connection

```
connection = XaOpenAudio(networkID,
    argcInOut, argvInOut,
    errStringLength, errString);
```

- XaCloseAudio: shut down a connection

```
XaCloseAudio(connection);
```

API: Object manipulation

- XaCreate: create an object

```
tag = XaCreate(connection,
    objectClass, attribute, value,
    ... );
```

- XaDestroy: destroy an object

```
XaDestroy(connection, tag);
```

API: Object manipulation

- XaSet: set attribute value(s) on an object

```
XaSet(connection, tag,
   asyncCallbackFunc,
   asyncCallbackData, attribute,
   value, ...);
```

- XaGet: get attribute value(s) from an object

```
XaGet(connection, tag,
   asyncCallbackFunc,
   asyncCallbackData, attribute,
   &value, ...);
```

API: Audio data transfer

- XaRead: read audio data from a port

```
XaRead(connection, port, atTime,
   refTime, minBits, maxBits,
   &bitsRead, &leftPad, buff);
```

- XaWrite: write audio data to a port

```
XaWrite(connection, port, atTime,
   refTime, numBits, leftPad, buff);
```

API: Synchronous vs. asynchronous

- Synchronous: blocks until a reply is received

```
XaGet(connection, object, NULL,
   NULL, ...);
```

- Asynchronous: returns immediately, calls callback when reply is received

```
XaGet(connection, object,
   callbackFunc, callbackData, ...);
```

API: Other calls

- Locating objects/atoms
  - XaFindAtom: Find a name
  - XaFindObject: Find an object

API: Other calls

- Manage the event queue
  - XaPending: Check for pending events
  - XaSync, XaCheckpoint: Return when pending requests are completed
  - XaFlush: Flush the event queue
  - XaMainLoop: Main loop for audio-only apps

Example: Play samples from buffer

```
conn = XaOpenAudio("", NULL, NULL,
   256, errString);

format = XaFindObject(conn,
   XaCformat, XaAname, XaNformatCD);

port = XaCreate(conn, XaCport,
   XaNinputBuffer, XaTnone, XaNformat,
   format, NULL);

XaWrite(conn, port, 0, XaNnowTime,
   numBits, leftPad, audioBuff);

XaClose(conn);
```

Example: Play samples from file

```
conn = XaOpenAudio("", NULL, NULL,
   256, errString);

file = XaCreate(conn, XaCfile,
   XaApathName, 3/users/mw/
   myFile.au2);

player = XaCreate(conn,
   XaCfilePlayer, XaNfile, file,
   NULL);

XaMainLoop();
```

Security

- Broadway security
- Security per connection (trusted/untrusted or other access group)
- Access control attributes on each connection/device
- Access settings: none, query, read-only, read-write

Audio Manager

- May have a GUI
- Not a window manager for audio
  - Doesn’t run continually
  - Presence is not required
- Sets preferences, persistent state
- Configures access for devices/etc.

Events

- Dispatched by event handler objects
- Create / change / destroy object events
- Plays well with Xt
  - XaMainLoop or XtMainLoop
  - Xa provides function and file descriptor for use with Xt event functions
  - Xa equivalents to Xt basic event commands

Creating an event handler (API)

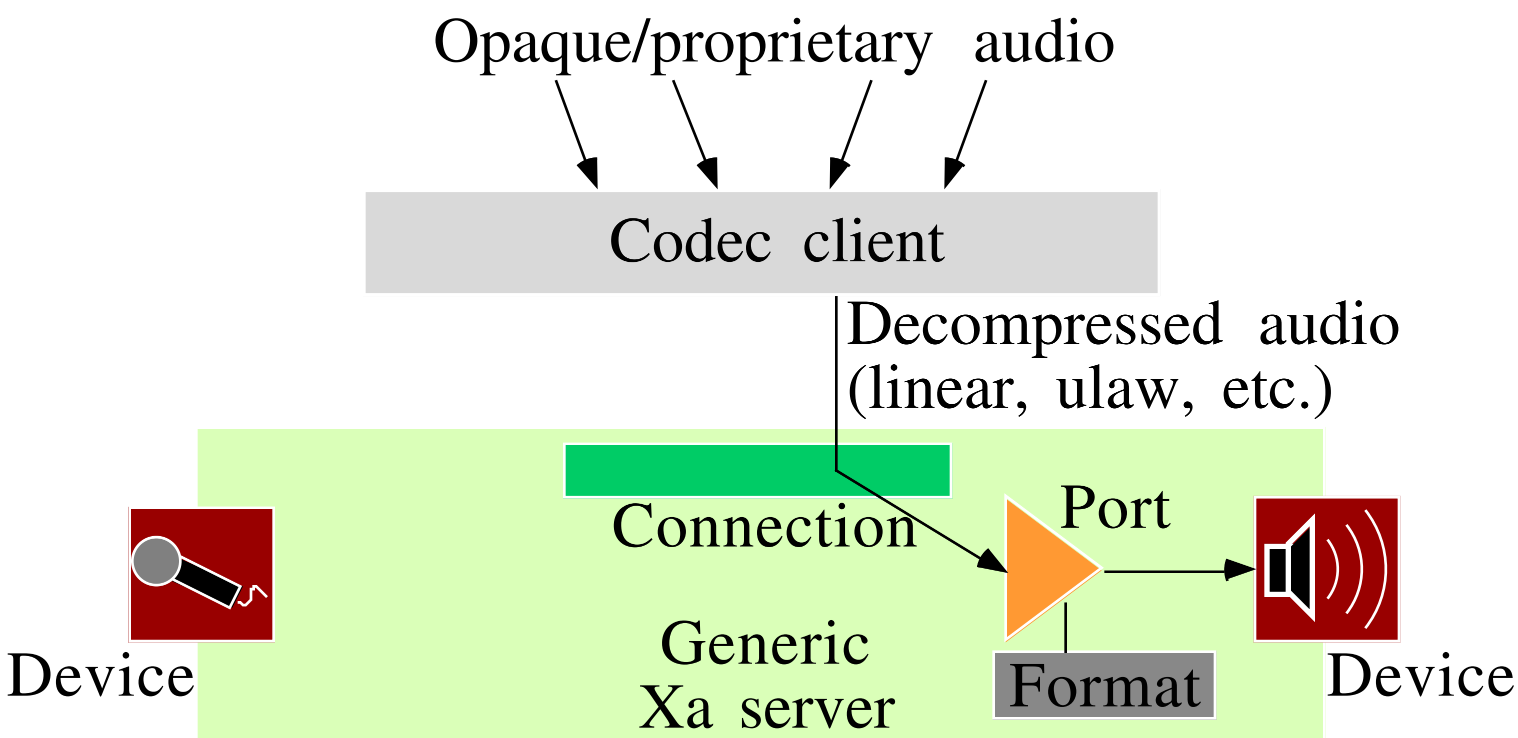
```
handler = XaCreate(connection,
   XaCeventHandler,
   XaNhandlerProc, (XaArgVal) myFunc,
   XaNhandlerData, (XaArgVal) myData,
   XaNtargetObject,
   myFavoriteClassObject,
   XaNeventType, XaAcreate,
   NULL);
```

Extensions

- May define new protocol requests
- May add new classes
- New devices
  - Multicast/radio
  - Telephony
- New format encodings...

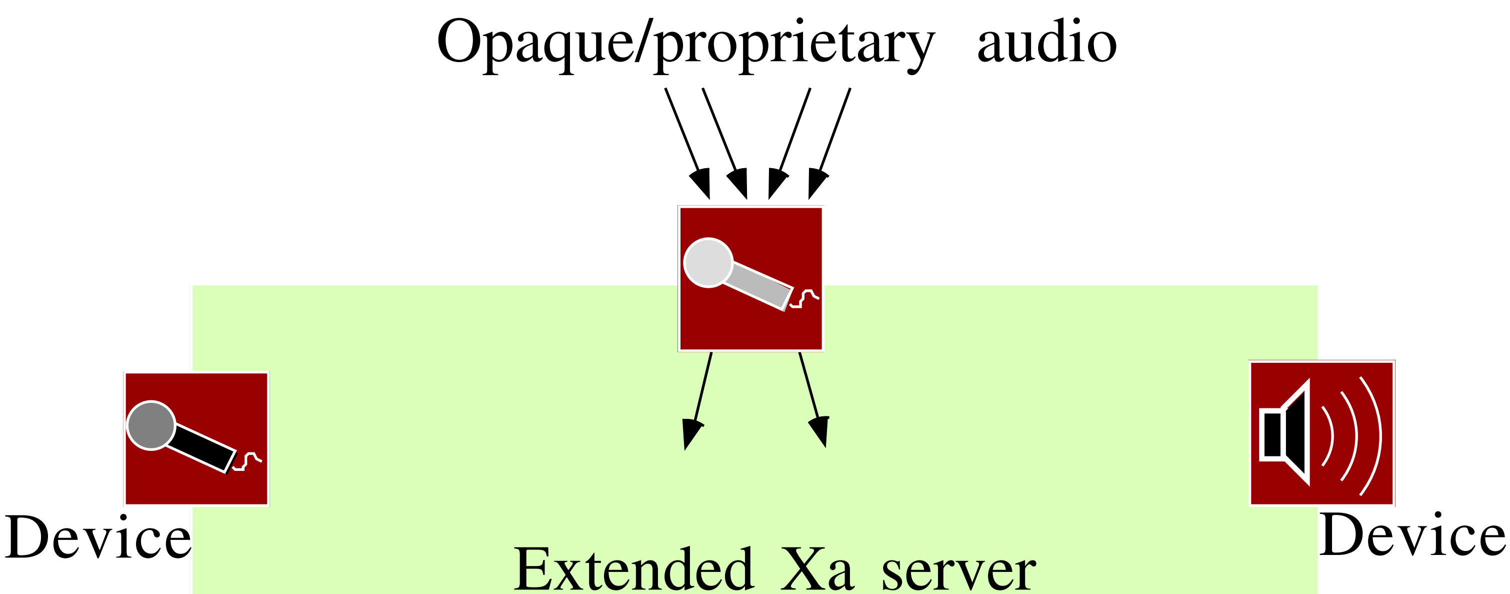
Implementing new encodings

- Method 1: Xa client



Implementing new encodings

- Method 2: Custom device



Futures

- Flows
- Synchronization/mixing of multiple input streams
- Perform complex processing on streams
- Additional synchronization facilities

Status

- In active development
  - Library & protocol specs
  - Library, server software
- Consortium standard in time for Broadway
- Implementation release with Broadway

The Broadway Audio System

Ray Tice (ray@x.org)

Mark Welch (mw@x.org)

URL for slides:

ftp://ftp.x.org/contrib/conferences/XTech96/audio_slides.ps