

Running a JustPlay Environment

Day 1, Session 2

July 2006

Quanta JustPlay Seminar

Justin Mazzola Paluska

jmp@mit.edu

Assumed Knowledge

- Familiarity using and configuring Python
 - <http://docs.python.org/tut/tut.html> is a great tutorial to start with!
- Familiarity with standard POSIX tools
 - bash
 - make
- Familiarity with version control
 - We use SVN, which is very much like CVS.

Conventions

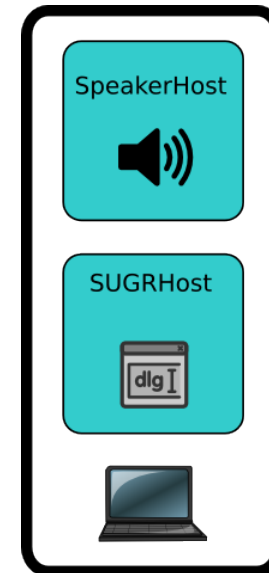
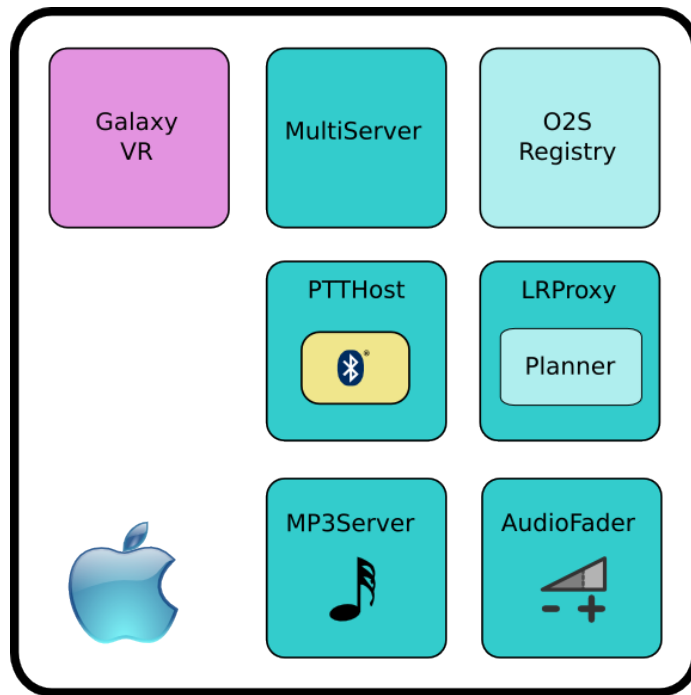
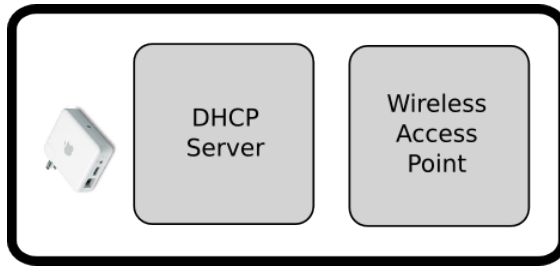
- `$VARIABLE` – something to be filled in
- All commands to be typed at the shell will be prefixed with “%” and in a grey box

```
%echo Hello World  
Hello World  
%ls *.py  
bootstrap_technique.py  lr_proxy.py
```

- Most examples assume you are in the root O2S directory
 - o2s-snapshot-2006-07 for this seminar

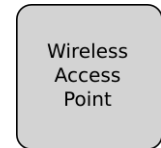
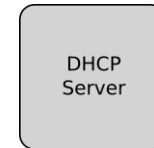
Background Information

JustPlay Demo



JustPlay Component Categories

- Required Infrastructure

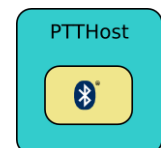
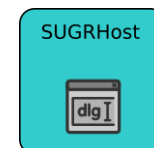
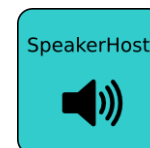
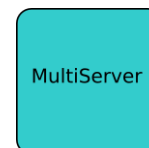
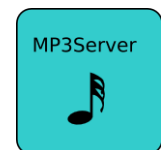
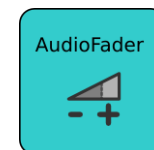


- O2S Infrastructure



- O2S Client Software

- Some clients have additional requirements



Required Infrastructure

- IP Network and interfaces
 - including DHCP
- OS with Python on all O2S devices
 - GNU/Linux
 - including embedded GNU/Linux systems as long as the platform has Python
 - Mac OS X
 - Windows with Cygwin

O2S uses Python

- We implemented most of our code in Python because Python code is easy and quick to write.
 - Python code is portable.
 - The Python interpreter is portable.
 - The Python interpreter is slow and big.
- nPops uses standard XML-RPC and sockets, so it is possible to port O2S nPops to other platforms.
 - There are limited implementations of O2S in other languages.

Required O2S Infrastructure

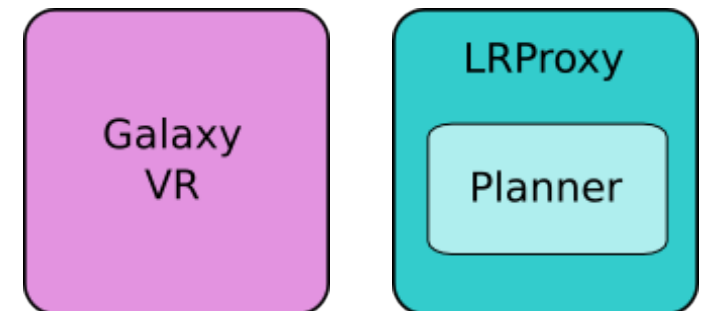
- O2S Registry
 - Can run on any machine on the same broadcast network as the clients
 - The O2S Registry is trusted code
 - If the Registry dies, so does your environment
 - Code is well-tested and verified
 - Should be on a “stable” machine



O2S
Registry

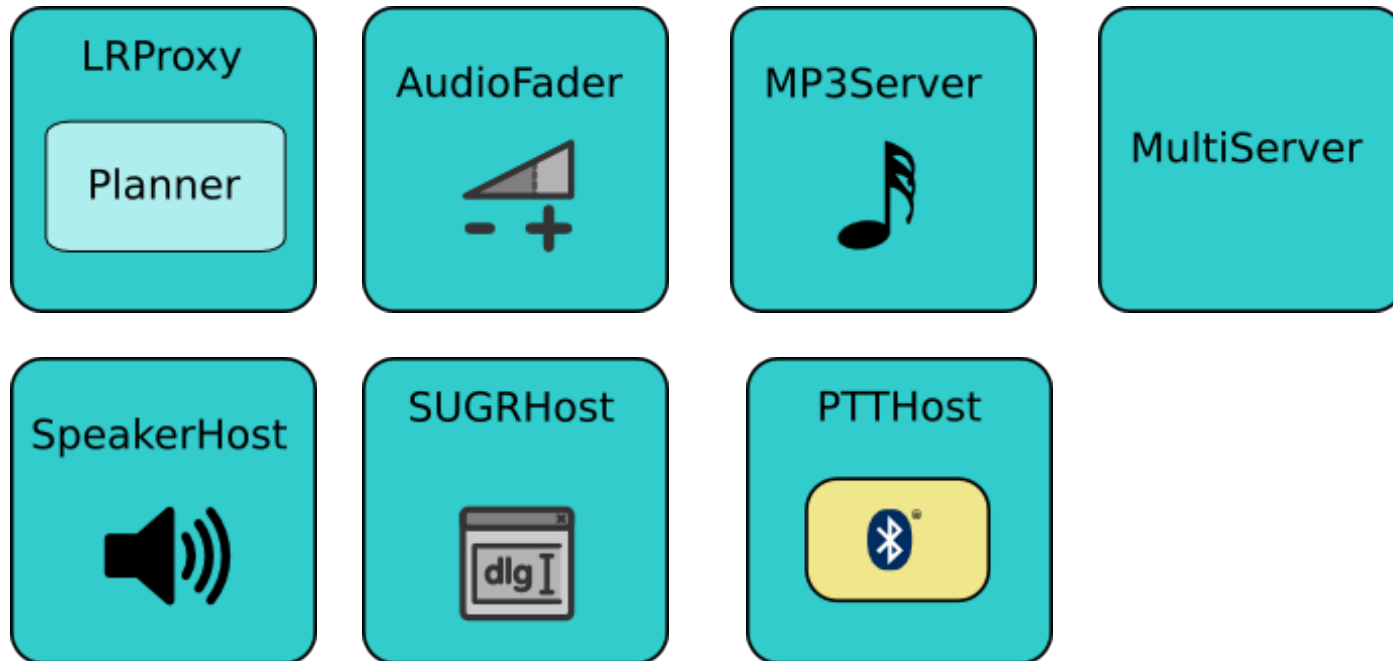
Optional JustPlay Software

- All used in the demo, but not strictly necessary.
- Galaxy Voice Recognition
- Planner
 - Embedded in the LRProxy of the Demo



Clients

- We used many clients
 - making a client is relatively easy



Clients – Music



Stream music from MP3 files to an output connector.



Stream music from an input connector to an output connector.

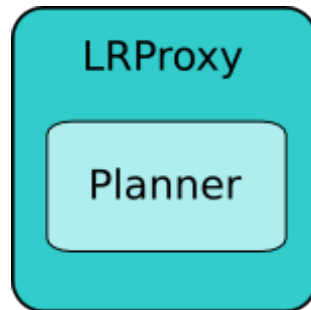
- Modifies stream in flight (volume)
- Aids hotswapping



Play music received from an input connector on a hardware music device

- Use hints in connector metadata to alter output volume and mix media streams.

Clients – Support Clients



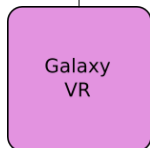
Interprets Voice Recognizer Events from the MultiServer

- Asserts or retracts PlayMusic goals based on the user's utterances
- Adopts SUGRHosts and PTTHosts



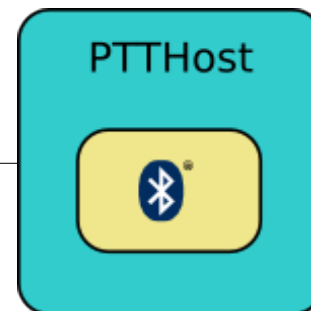
Galaxy Voice Recognition Interface

- Receive speech on an input connector
- Send back text as Events
- Communicates with Galaxy with non-O2S XML-RPC



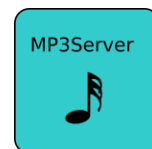
GUI Interface

- Sends button click events
- Records speech to an output connector



Bluetooth interface to Python on the Nokia Phone.

- Embeds an MP3Server to stream music.
- Records speech to an output connector



Clients

- Most clients are portable
- But some are not:
 - SUGRHost can't record audio on OS X
 - SpeakerHost doesn't work on OS X
 - PTTHost requires Bluetooth – only tested on OS X

Running JustPlay (Basics)

Running JustPlay – Prerequisites

- A full installation of Python
 - <http://www.python.org>
 - Linux: 2.3 or 2.4
 - Mac: 2.4.3 Universal Python
 - Windows: cygwin python
- Subversion
 - <http://subversion.tigris.org/>
 - Access to O2S source code

Running JustPlay – Initial Setup

- Check out the O2S snapshot:

```
%svn checkout \  
    svn://o2s.csail.mit.edu/branches/o2s-snapshot-2006-07
```

- Tell Python where to find O2S's modules:

```
%export PYTHONPATH=$(O2S_DIR)/lang-support/python
```

Running JustPlay – Registry

- After setup, the first O2S component you'll need is a registry.

```
%cd $(O2S_DIR)
%make registry
Starting a registry (UDP):
(cd system/registry; make registry)
make[1]: Entering directory `/home/jmp/.unison-
root/CSAIL/o2s/system/registry'
python -u registry.py
Registry started at ('', 0)
[Registry] Your Keep Alive Port: 33178
```

Running JustPlay – Registry

- “make registry” brings up a Registry that other O2S components will discover as “JustPlay”
 - It is a bad idea to have multiple registries with the same name.
 - Set the O2S_BEACON_NAME environment variable to use another name:

```
%export O2S_BEACON_NAME=PleaseJustWork  
%make registry
```

O2S Debugging Tools

- o2s.py – command line control and debugging tool
- o2watch.py – GUI entity control and debugging tool
 - both live in the o2s/scripts directory
- These tools both search for a registry then allow you to act on the Entities found in the registry

Debugging Tools – o2s.py

```
%python scripts/o2s.py who
(Discovering for 10.00 seconds. Please be patient.
  found 140.112.28.46 = (JustPlay, 58665, 140.112.28.46)
REGISTRY LIST:  [('140.112.28.46', 'JustPlay', 58665,
'140.112.28.46')]
Registry beacons heard:
      140.112.28.46: 58665          140.112.28.46 JustPlay

-----
o2s command utility.  (version $Revision: 3706 $)

* Querying Registry at: 140.112.28.46:58665

RID      Name      Type      Request Addr      Subs      Health
---      ----      -

```

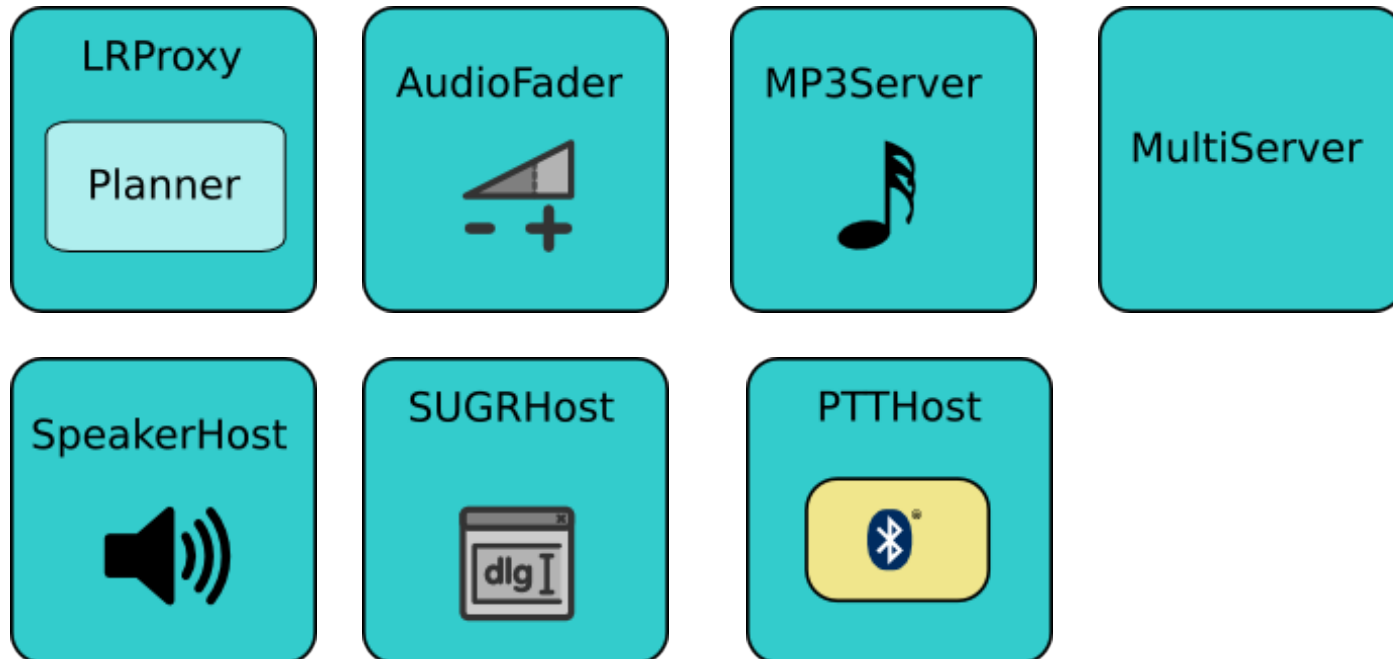
Debugging Tools – o2watch.py

```
%python scripts/o2watch.py &
[3] 26483
(Discovering for 10.00 seconds. Please be patient.
  found 140.112.28.46 = (JustPlay, 58665, 140.112.28.46)
REGISTRY LIST:  [('140.112.28.46', 'JustPlay', 58665,
'140.112.28.46')]
Registry beacons heard:
      140.112.28.46: 58665           140.112.28.46 JustPlay
```



Running O2S Entities

- So far, we're just running a registry and some tools.
- Entities do all of the interesting work in O2S



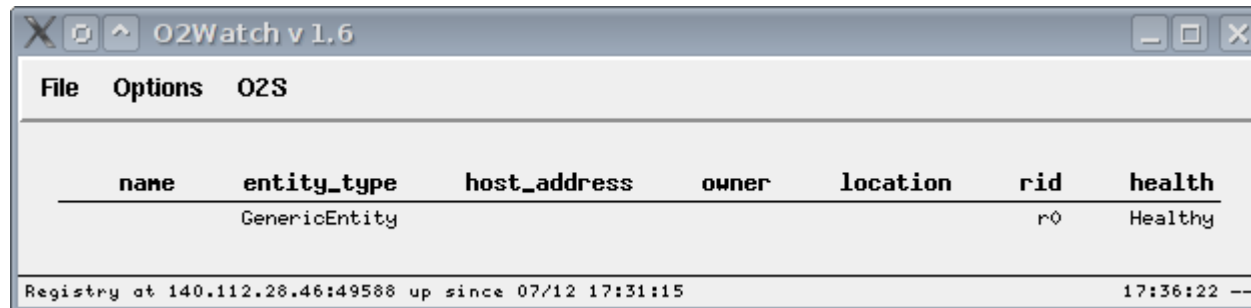
Starting an Entity

- Entities are just Python programs, started like any other Python program.
 - On startup, an Entity will search for a Registry and register with it.
 - The new Entity can now be located by other Entities using the same registry.

Starting an Entity

- The simplest entity is the GenericEntity:

```
%python system/entity.py &
```



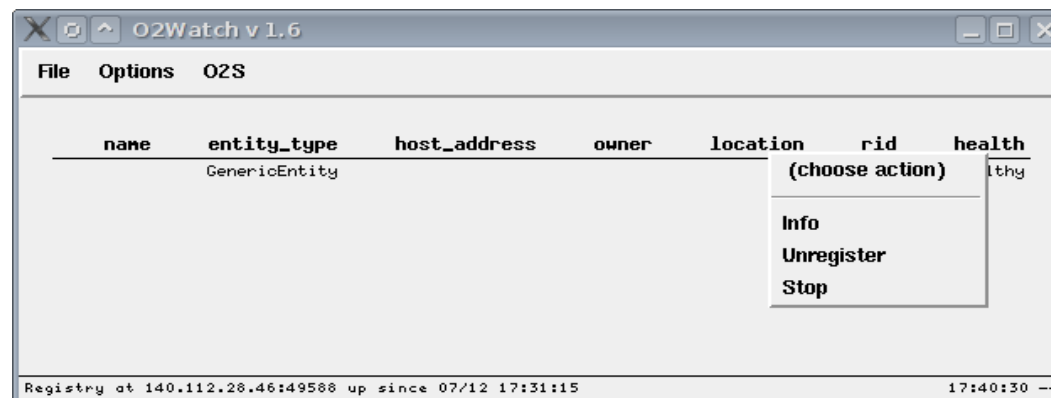
The screenshot shows a window titled "O2Watch v 1.6" with a menu bar containing "File", "Options", and "O2S". Below the menu bar is a table with the following columns: name, entity_type, host_address, owner, location, rid, and health. The table contains one row with the value "GenericEntity" under the entity_type column and "r0" under the rid column, with the health status "Healthy". At the bottom of the window, a status bar displays "Registry at 140.112.28.46:49588 up since 07/12 17:31:15" and the time "17:36:22 --".

name	entity_type	host_address	owner	location	rid	health
	GenericEntity				r0	Healthy

Registry at 140.112.28.46:49588 up since 07/12 17:31:15 17:36:22 --

Stopping An Entity

- kill it with shell commands
 - not clean, but the system detects that the Entity has died
- kill it with O2S debugging tools:
 - menu item in o2watch
 - `o2s.py --rid XXX kill`



Review of Debugging Tools

- The debugging tools are more useful when your environment has running Entities.
- o2s.py has many subcommands
 - who – list currently registered Entities
 - kill – kill an Entity
 - killall – kill all registered Entities
 - kill_reg – kill the registry
 - info – get information about an Entity

Choosing a Specific Registry

- Entities listen for Registry broadcasts for 10s.
 - Entities expect only 1 registry.
 - If there are multiple registries, an Entity chooses which to use at random.
- Set the `O2S_REGISTRY_ADDRESS` environment variable to force an Entity to choose a specific Registry
 - Bonus: it's faster too!
 - Also affects the O2S debugging tools

O2S Configuration System

- O2S can be configured using
 - environment variables
O2S_LOGGING=warning
 - command-line options
--o2s-logging=warning
 - configuration files
LOGGING=warning
 - /etc/o2s – global settings
 - ~/.o2s – user settings

Common Configuration Options (Clients)

- O2S_USER – exported username
 - Default: \$USER (on Linux/Mac OS)
- O2S_LOGGING – control the amount of debugging output
 - Default: “warning”
- O2S_REGISTRY_ADDRESS – the registry to use
 - Default: “*” – use any registry

Common Configuration Options (Infrastructure)

- `O2S_BEACON_NAME` – the address the registry should broadcast
 - Default: JustPlay
- `O2S_REGISTRY_SERVE_PORT` – use a specific port
 - Default: random port
- `O2S_GALAXY_REC` – hostname:port of the Galaxy server.
 - Default: “localhost:9000”

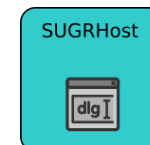
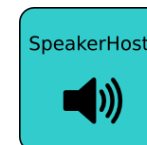
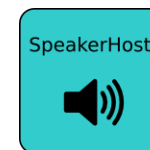
Running JustPlay (the Demo)

Optional Python Modules

- pyserial – used for Bluetooth
 - Pure Python; just download and install
 - <http://pyserial.sourceforge.net/>
- pymad – MP3
- audio_scale – fast volume scaling
 - GumStix are too slow to change audio volume using only Python.

JustPlay Python C Modules

- Only required by certain Entities, mostly having to do with audio
- C modules must be compiled for each platform
- pymad – MP3 decoder
 - <http://spacepants.org/src/pymad/>
- audio_scale
 - in o2s/hosts/soundio/audio_scale



Building audio_scale

- Requires a C compiler and Python
- First, build soundcard_params.py

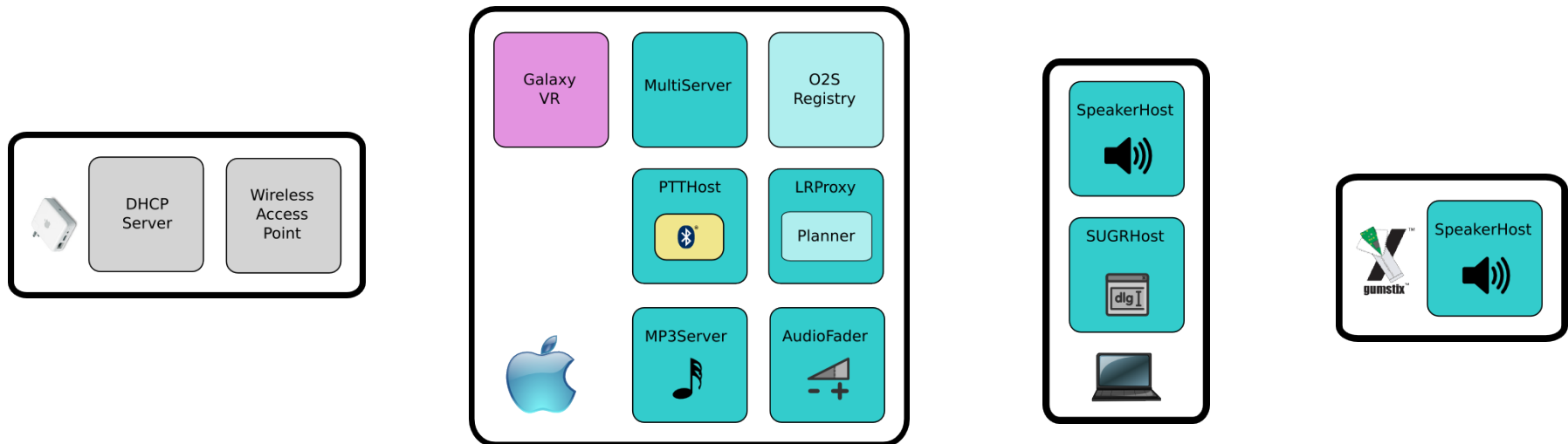
```
%cd hosts/soundio  
%make soundcard_params.py
```

- Then, build the audio_scale module:

```
%cd hosts/soundio/audio_scale  
%make
```

Bringing Up Your Own Demo

- Compile / install the relevant components you need on each piece of hardware, start the relevant software, then “just play”



Starting JustPlay Clients

SUGRHost



```
%cd hosts/nsugr  
%python tk_sugr.py
```

AudioFader



```
%cd playground/fader  
%python audio_fader.py
```

Starting JustPlay Clients



```
%cd playground/music  
%python mp3server $MUSIC_DIR
```

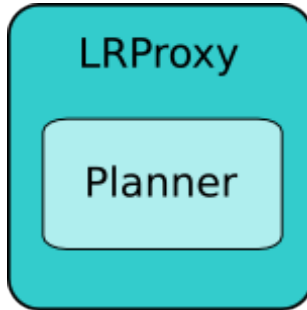
- \$MUSIC_DIR is any directory with .mp3 files



```
%cd hosts/speakerhost  
%python [options] speakerhost.py
```

- The options characterize the attached speaker
 - --wattage, --make, --model
 - Speaker characteristics used by the Planner

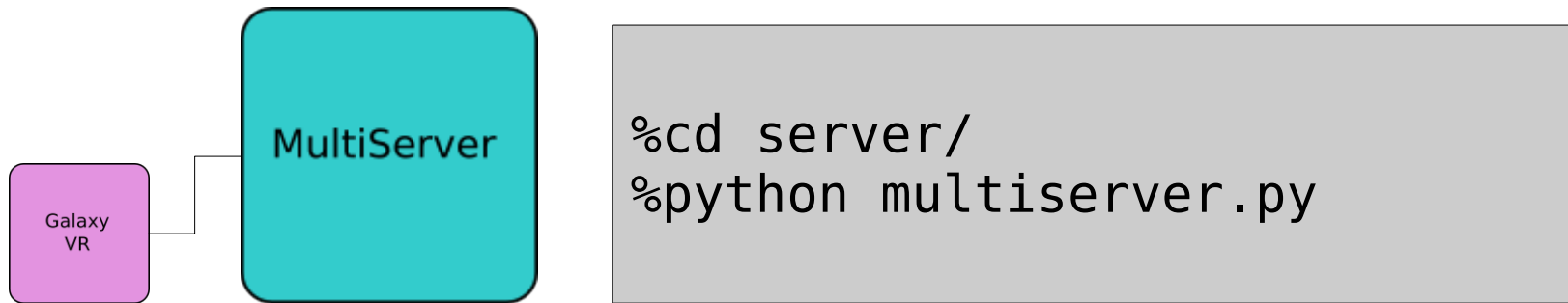
Starting JustPlay Clients



```
%cd playground/lrproxy  
%python [options] lr_proxy.py
```

- LRProxy can also take arguments to specify what Planner “visualizer” should be run.
 - “--inspector=console_viz” prints to the screen
 - “--inspector=sugr_viz” allows inspection from o2watch.py
- The embedded Planner needs to know what Techniques to use:
 - set TDB_PATH environment variable
 - JustPlay audio techniques are in the playground/audiohotswap directory.

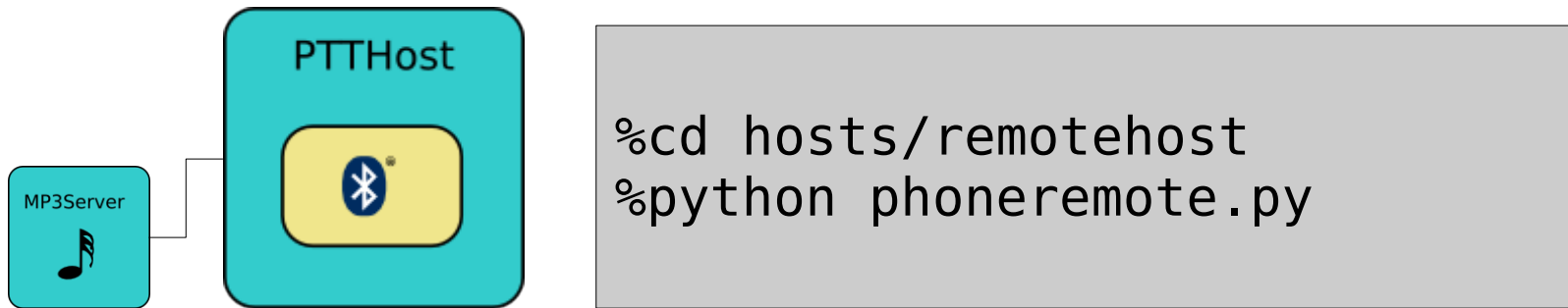
Starting JustPlay Clients



- If you want to use VR with the MultiServer, you must start Galaxy **before** starting the MultiServer
- Galaxy only runs on the Mac, but if you have compiled it on the Mac, run it with:

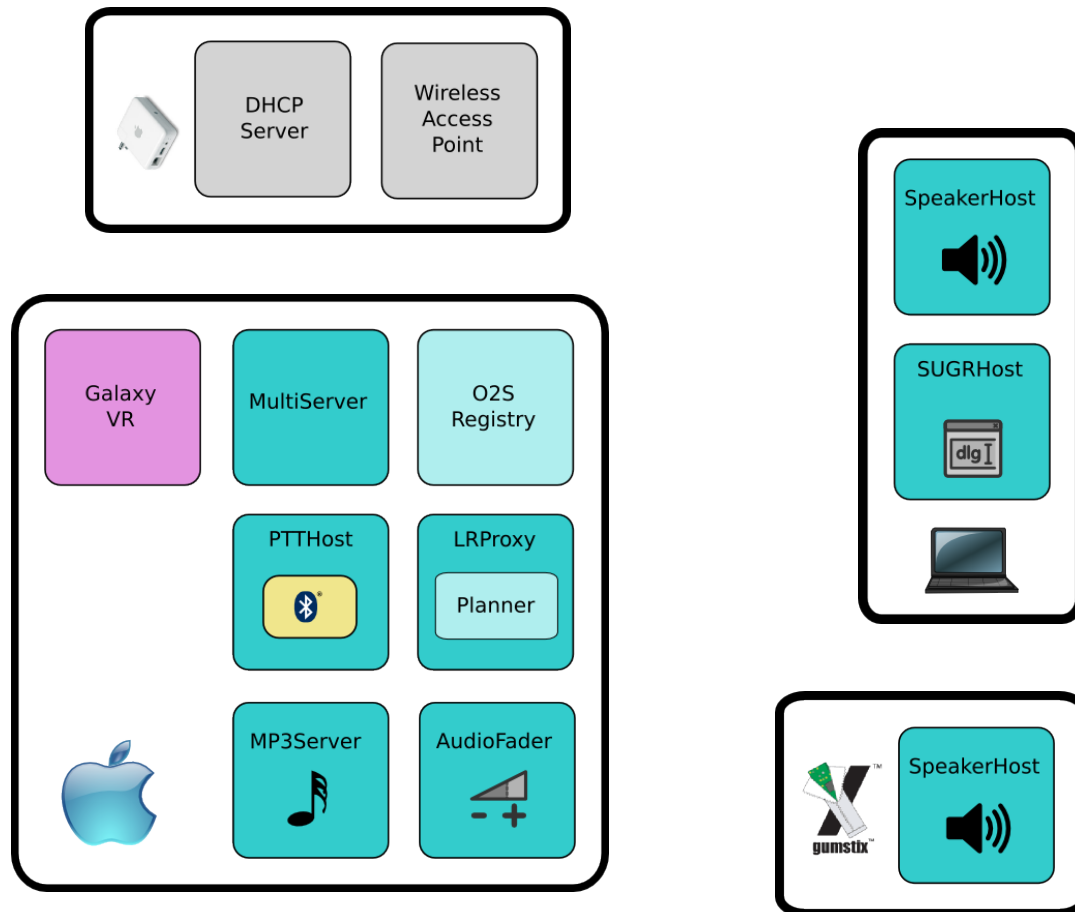
```
%cd $(GALAXY_DIR)  
%./run
```

Starting JustPlay Clients



- Requires pyserial and additional Bluetooth setup
- Requires O2S client software on the Nokia Phone
- Only regularly used on the Mac, but reportedly works on Windows too.

Questions?



Day 1, Session 2
July 2006
Quanta JustPlay Seminar

Justin Mazzola Paluska
jmp@mit.edu