Audio Management in Tizen IVI

Jaska Uimonen

TIZEN™
DEVELOPER CONFERENCE
2013
SAN FRANCISCO

Introduction

Some personal data

- Working for Intel OTC in Finland
- Currently working with Tizen IVI profile
- Past work history in mobile development
- A Member of Murphy team http://01.org/murphy

TIZEN™ DEVELOPER CONFERENCE 2013 SAN FRANCISCO

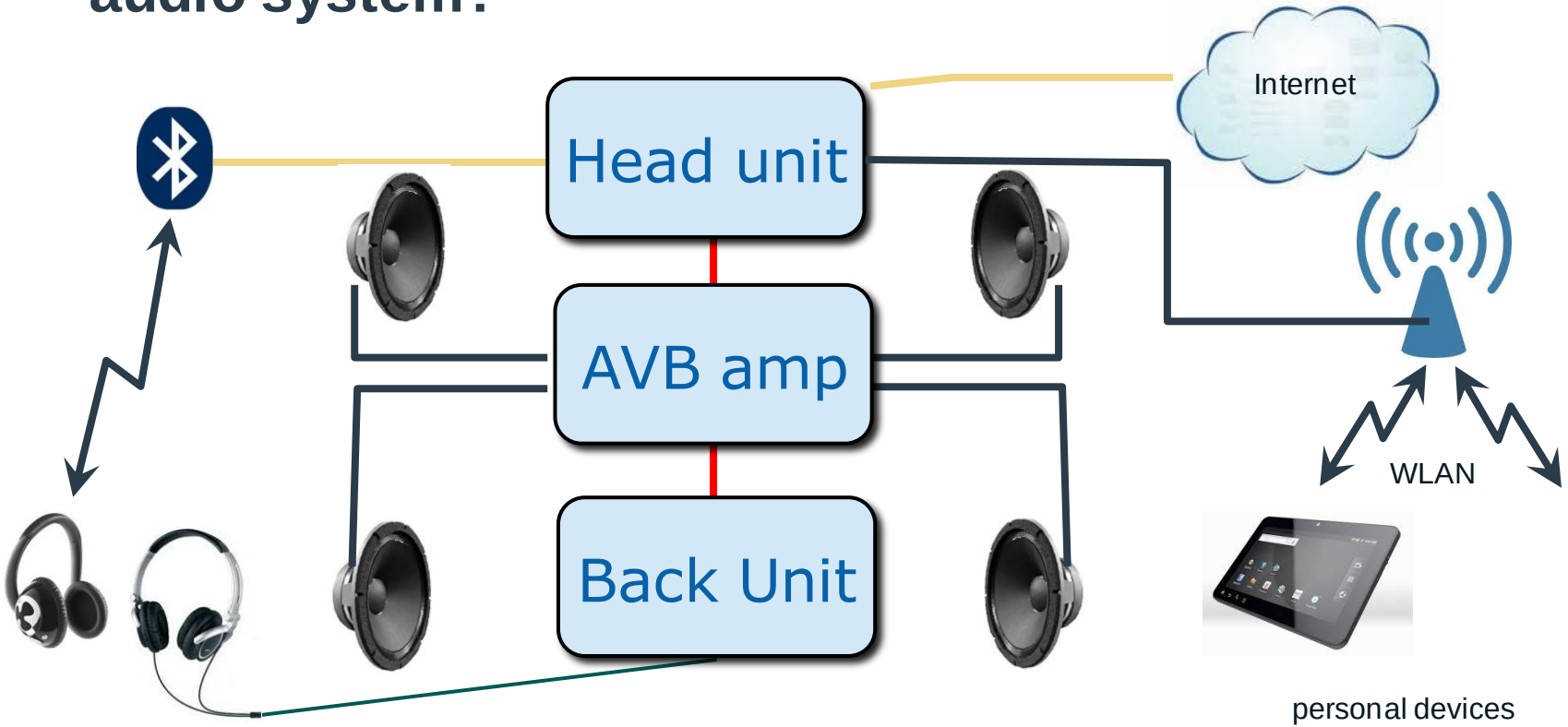
What audio management means in Tizen IVI?

- **Policy controlled routing**
  - Routing audio streams to their allowed destinations (or possibly to many destinations)
- **Policy controlled volume**
  - Volume ramping
  - Muting
  - Static volume change for audio stream's life time
- **Policy controlled stream pre-emption**
  - Stopping/pausing/killing and possibly restarting conflicting audio streams

TIZEN™ DEVELOPER CONFERENCE 2013 SAN FRANCISCO

Overview

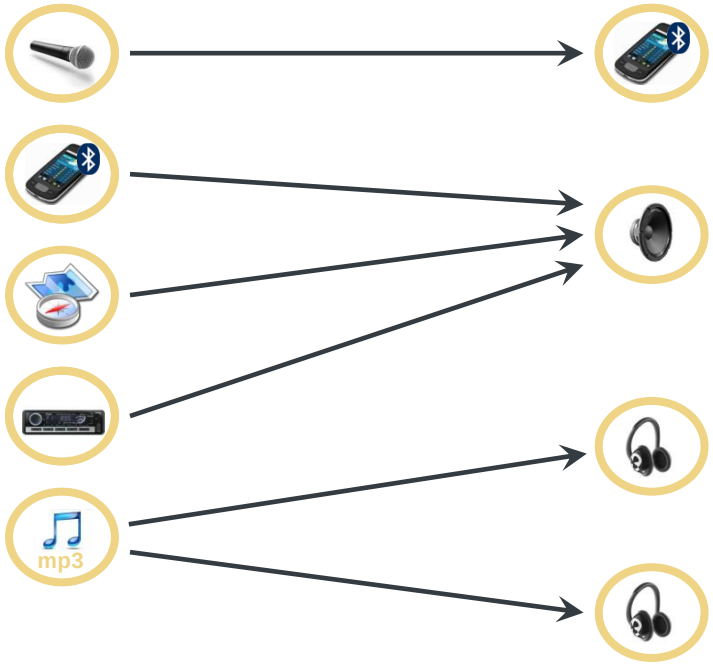
What kind of HW configuration could constitute an IVI audio system?



TIZEN DEVELOPER CONFERENCE 2013 SAN FRANCISCO

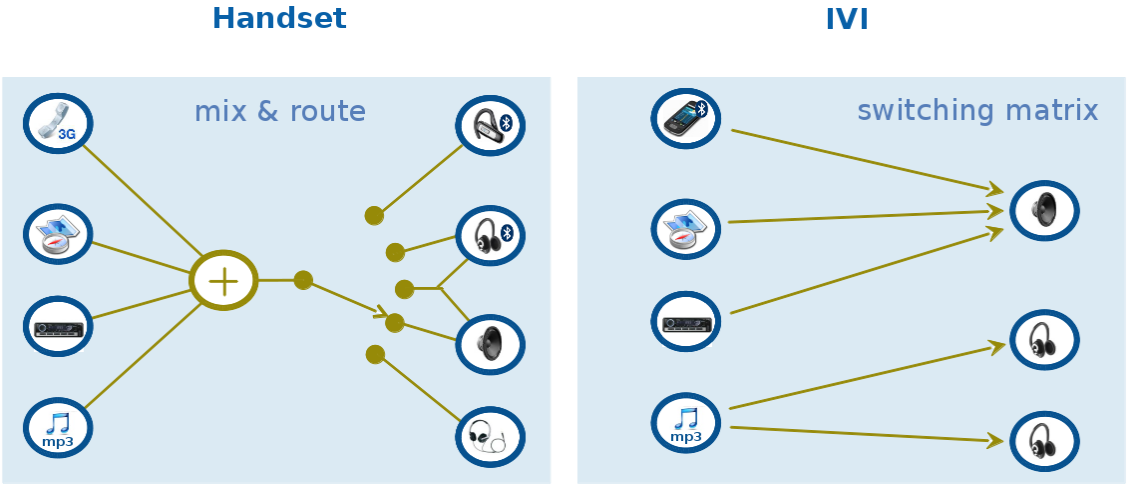
What kind of audio use cases we could have in IVI?

- The driver listens to radio
- Voice guided navigation is on
- Backseat passengers listen to the same mp3 music using headphones
- The drivers personal phone is connected to the car's handsfree gateway via bluetooth.
- The driver's phone is ringing and the incoming call is accepted



Conclusion: IVI and mobile audio systems are different

- Many simultaneously used outputs
- Possibly multiple users (in different zones)
- Possibly multiple computing units connected via network

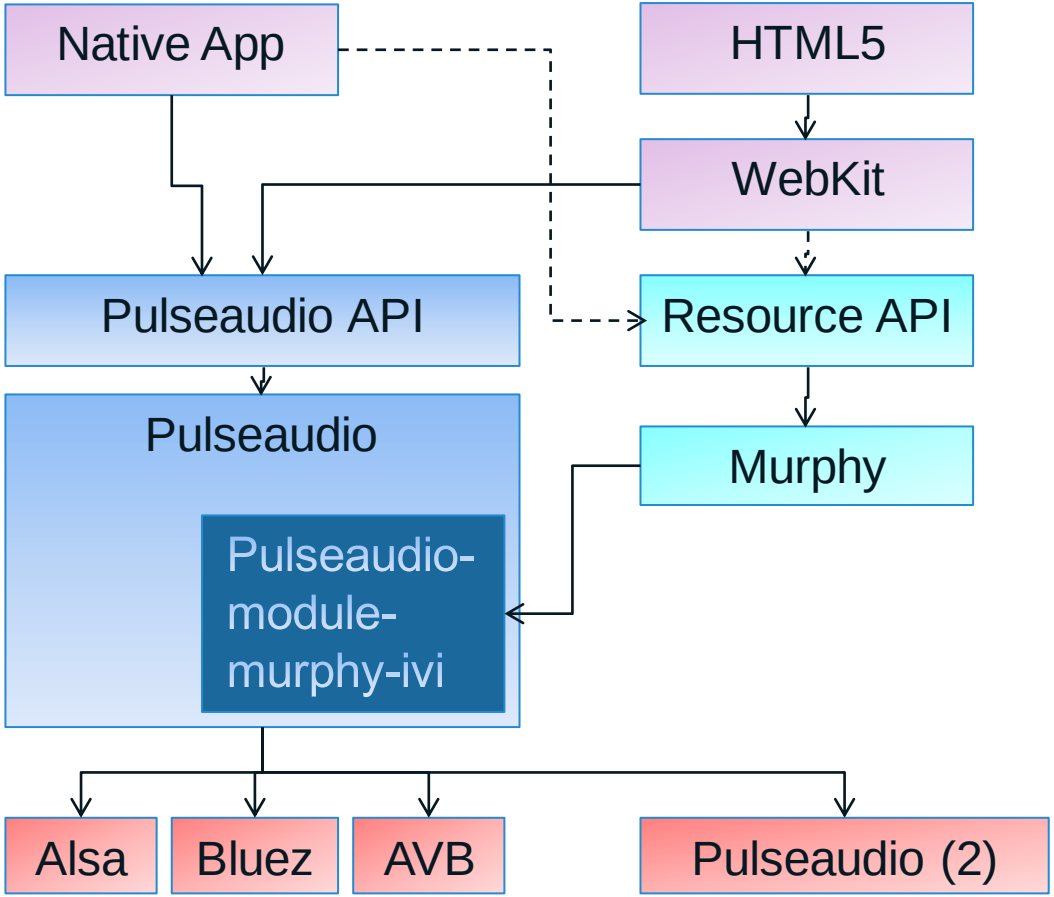


TIZEN DEVELOPER CONFERENCE 2013 SAN FRANCISCO

Design & Implementation

High-level SW design

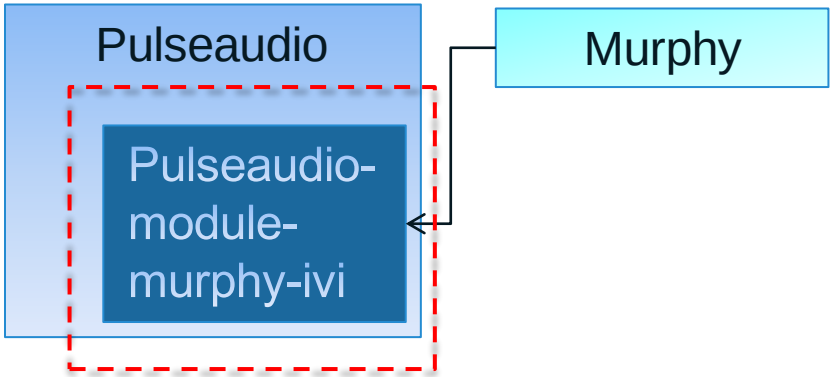
- Audio domains centralized in pulseaudio
- Independent audio management module inside pulseaudio
- Applied policies are based on stream tagging
- Basic html <audio> and <video> tags are integrated to the policy
- For more fine grained policy handling Murphy resource API can be used



TIZEN DEVELOPER CONFERENCE 2013 SAN FRANCISCO

Features in pulseaudio-module-murphy-ivi

- Routing with priority queues
- Volume control with constraints
- Configuration and scripting
- Resource allocation through Murphy



TIZEN DEVELOPER CONFERENCE 2013 SAN FRANCISCO

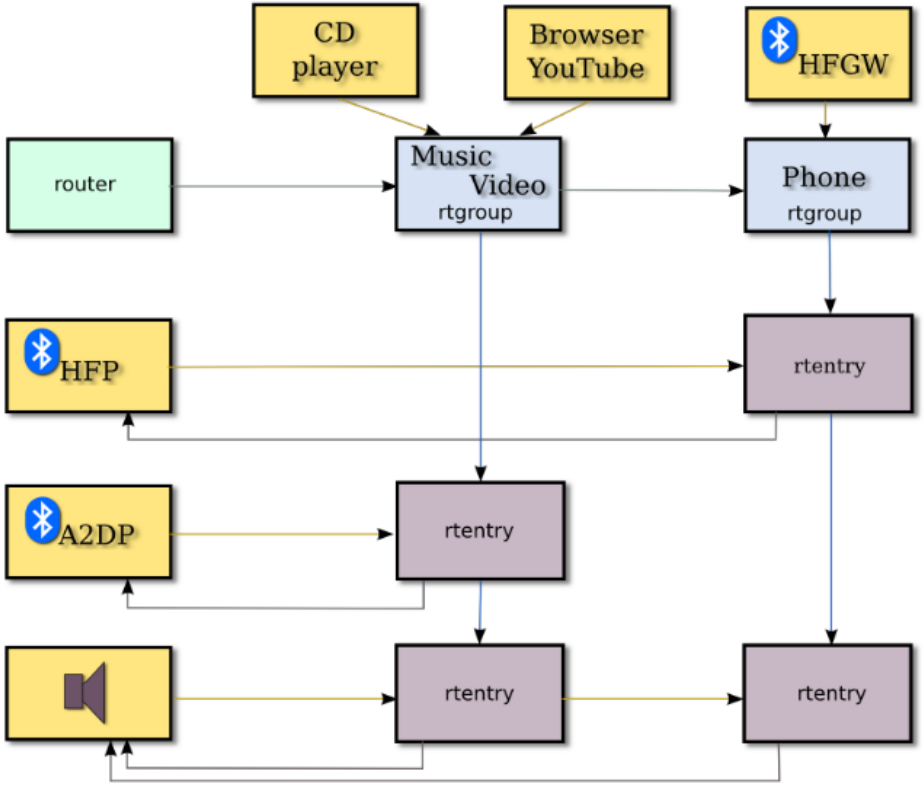
Routing

Nodes

- **New logical model in pulseaudio – Nodes**
  - Correspond to pulseaudio sinks and sources
  - Input and output nodes can be freely connected – for e.g. 1:N
  - Nodes are dynamically appearing and reappearing
  - Nodes might have HW limitations
- **Nodes can be used for *explicit* or *default* routing**
  - Explicit route is requested by the user
  - New events will not affect an explicit route
  - Default route is automatic and created at stream creation
  - Default route is dynamic and class based

Priority based conflict resolution

- Explicit routes have always priority over default route
- Default routes use class based stream priorities
- Class based routing target lists
- Walking through the streams in decreasing priority order to make the routing decision
- In case of conflict explicit routes are disabled and for default route the next available target on the list is chosen



TIZEN DEVELOPER CONFERENCE 2013 SAN FRANCISCO

Volume control

Volume control implementation

- **Volume control is based on an independent mechanism**
  - smooth volume ramp up/down of streams or entire devices
  - usual volumes left untouched
  - real enforcement mechanism, ie. clients can't override it
- **Control points**
  - Streams
  - Devices
- **Scriptable**
  - for configuration, ie. what and how to link together to achieve the desired effect

Volume constraints

- **Class based volume constraints**
  - For e.g. telephony stream is attenuating music but not navigator
- **Generic constraints**
  - Can be used for e.g. noise dependent master volume level
  - However generic limits can depend on device type or other conditions
  - From all the applicable generic limits the highest attenuation will be used
- **Constraints are combined**
  - If the **generic limit** would be **-12dB** and
  - The **class limit** would be **-20dB** then the
  - **Actual limit** for the stream would be **-32dB**

Configuration and scripting

Configuration and scripting

- **Proper configuration and scripting is an essential because**
  - We want to run same software in different verticals
  - Different manufacturers want different policies
  - Fast prototyping is essential for product programs
- **Pulseaudio-module-murphy-ivi configuration is done with Lua**
  - Configuration done at initialization, after that Lua is not executed
- **There is also possibility to script some of the functionality**
  - Lua is executed also at run time

LUA configuration example

- **Routing groups**
  - Default
  - Phone
- **Application classes**
  - Belong to a routing group
  - Have priorities

```
routing_group {
    name = "default",
    node_type = node.output,
    accept = builtin.method.accept_default,
    compare = builtin.method.compare_default
}

routing_group {
    name = "phone",
    node_type = node.input,
    accept = builtin.method.accept_phone,
    compare = builtin.method.compare_phone
}

routing_group {
    name = "phone",
    node_type = node.output,
    accept = builtin.method.accept_phone,
    compare = builtin.method.compare_phone
}

application_class {
    node_type = node.event,
    priority = 6,
    route = {
        output = routing_group.default_output
    },
    roles = { event = no_resource }
}

application_class {
    class = "phone",
    node_type = node.phone,
    priority = 5,
    route = {
        input = routing_group.phone_input,
        output = routing_group.phone_output
    },
    roles = { phone = no_resource, carkit = no_resource }
}
```

Connection to Murphy policy manager

Connection to Murphy database

- Module-murphy-ivi can subscribe to Murphy database events
- Based on the events LUA scripting or internal C functions can be invoked
- This way cross domain policies can be nicely handled
- Example would be speed dependent volume

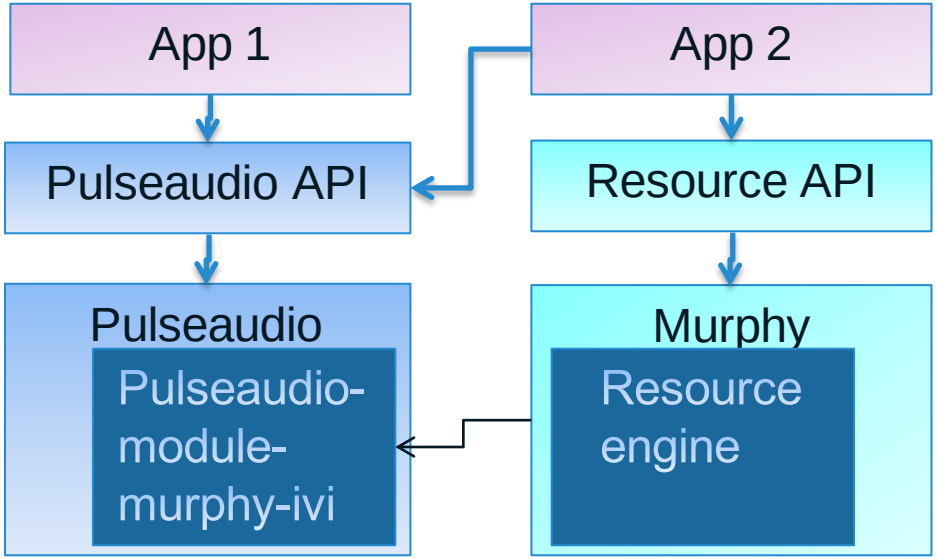
```
mdb.select {
          name = "vehicle_speed",
          table = "amb_vehicle_speed",
          columns = {"value"},
          condition = "key = 'VehicleSpeed'"
}

element.lua {
   name    = "speed2volume",
   inputs  = { speed = mdb.select.vehicle_speed, param = 9 },
   outputs = {  mdb.table { name = "speedvol",
                            index = {"zone", "device"},
                            columns = {{"zone", mdb.string, 16},
                                       {"device", mdb.string, 16},
                                       {"value", mdb.floating}},
                            create = true
                           }
            },
   oldvolume = 0.0,
   update  = function(self)
               speed = self.inputs.speed.single_value
               if (speed) then
                   volume = (speed - 144.0) / 7.0
               else
                   volume = 0.0
               end
               diff = volume - self.oldvolume
               if (diff*diff > self.inputs.param) then
                   print("*** element "..self.name.." update "..volume)
                   self.oldvolume = volume
                   mdb.table.speedvol:replace({zone = "driver", device = "speakers", value = volume})
               end
             end
}
```

```
mdb.import {
    table = "speedvol",
    columns = {"value"},
    condition = "zone = 'driver' AND device = 'speaker'",
    maxrow = 1,
    update = builtin.method.make_volumes
}
```


TIZEN™ DEVELOPER CONFERENCE 2013 SAN FRANCISCO

Resource allocation through Murphy

- Pulseaudio-module-murphy-ivi can reserve a resource for you if configured to do so
- Stream pre-emption works then automatically, although not so fine grained as through Murphy resource API



TIZEN DEVELOPER CONFERENCE 2013 SAN FRANCISCO

THANK YOU!

- Pulseaudio-module-murphy-ivi can be found from http://github.com/otcshare/pulseaudio-module-murphy-ivi
- Murphy and some documentation can be found from http://01.org/murphy