Routing and Policy

Pulseaudio workshop

02/11/2012 Copenhagen

About Murphy ...

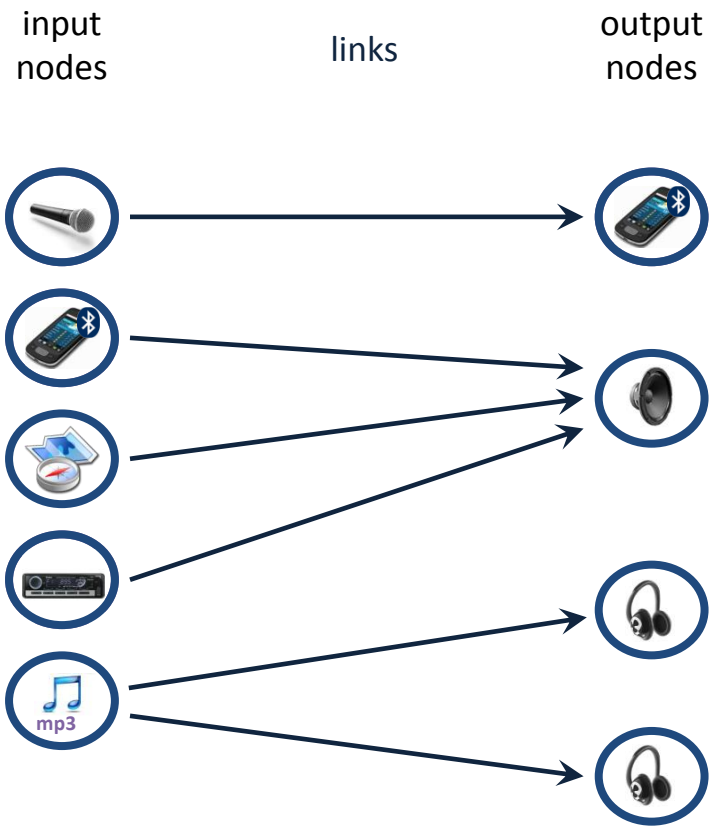
- Murphy is a open source policy engine
- 01.org/murphy
- One of the key task for Murphy is audio management (routing, volume limitation etc.)
- From audio perspective Murphy consists of
  - daemon
  - pulseaudio module for audio domain control
- Pulseaudio module works standalone without the daemon
- We would like either parts or the whole module to be part of upstream Pulseaudio

Example use cases

- The driver listens to radio
- Voice guided navigaton 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 drivers phone is ringing and the incoming call is accepted
    - The radio continues in the background with low volume
    - The phone discussion is routed to the front speakers and the built-in mic
    - the occasional navigation instructions are also routed to the front speakers

Logical Model for audio routing

Example

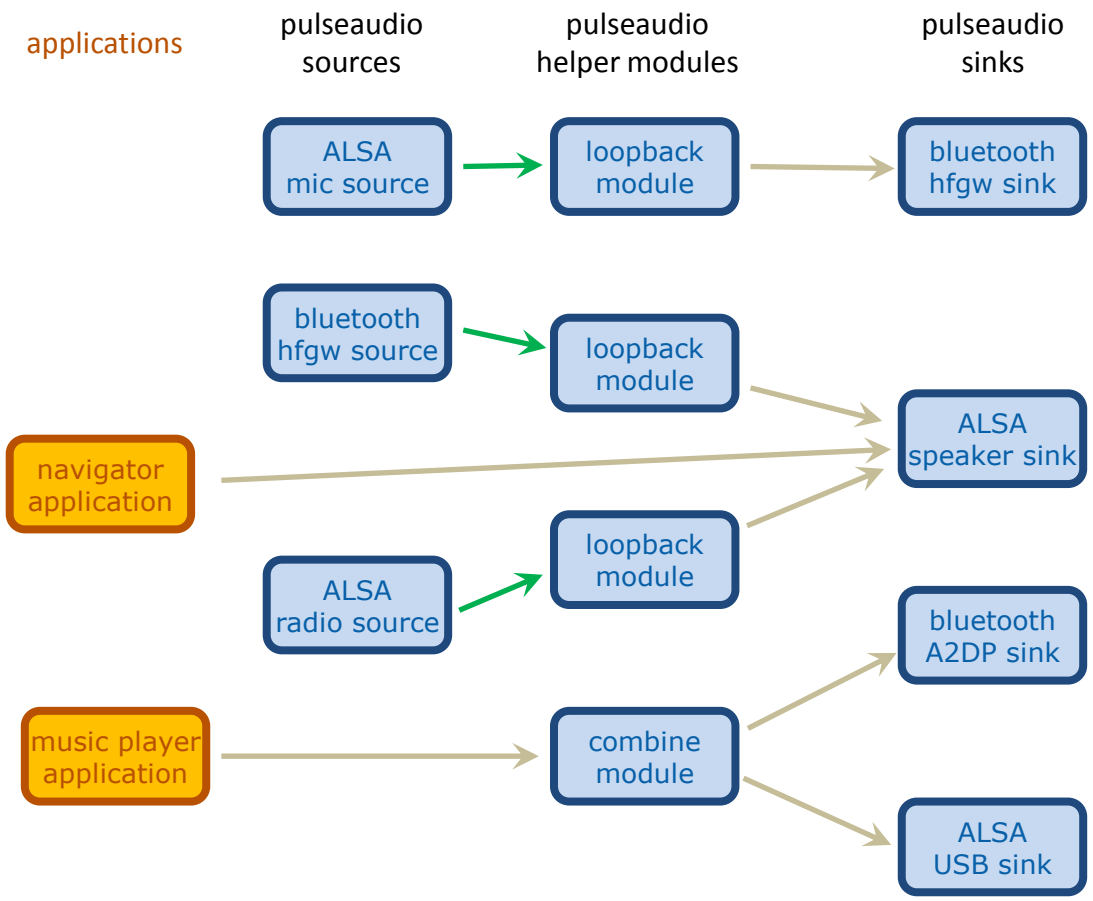


Logical Model

- switching matrix
  - nodes – routing endpoints
  - nodes dynamically appear/disappear (as bluetooth, USB and other accessories connect/disconnect)
  - input & output nodes
- input & output nodes can be freely linked (1:1, n:1, 1:n)
- link constrains
  - mutually exclusive bluetooth profiles (eg. for headsets either A2DP or handsfree)
  - possible HW limitations ie. ports
- explicit and default links
  - explicit - user requested link
  - default – produced by Murphy
    - if no explicit link was requested
    - configurable rules
- mapping to the GenIVI audio model
  - input node -> GenIVI source
  - output node -> GenIVI sink
  - link -> GenIVI connection

Implementation in PulseAudio

Example

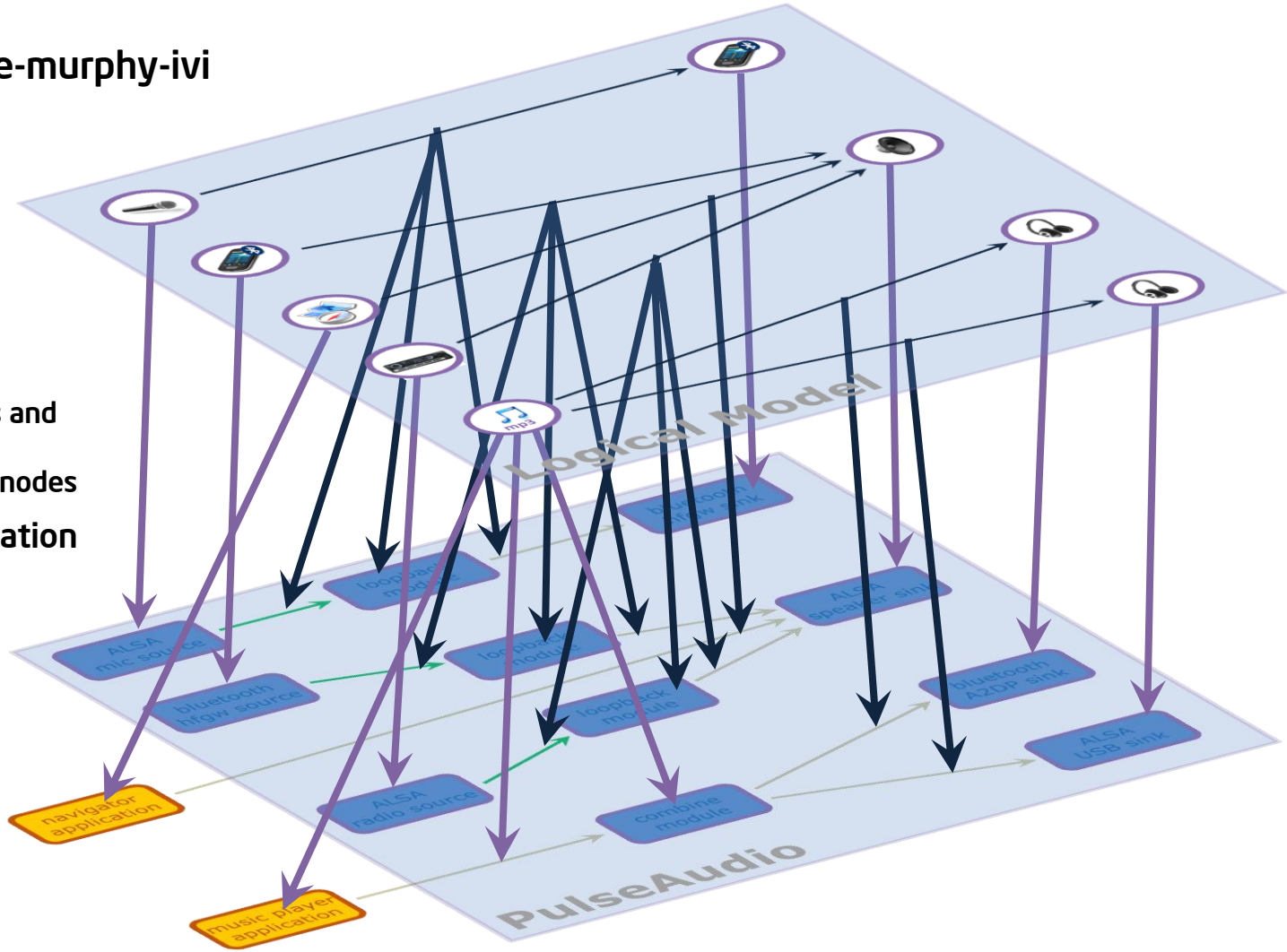


Implementation

- Mapping
  - logical input node
    - pulseaudio source
    - pulseaudio source-output (green arrows)
  - logical output node
    - pulseaudio sink
    - pulseaudio sink-input (red arrows)
- Helper modules for links
  - loopback module
    - to link a sink to a source
  - combine module
    - to link a sink-input to multiple sinks

Mapping the LogicalModel to PulseAudio

- **Mapping is in module-murphy-ivi pulseaudio module**
- **Logical model automatically built**
  - **plug'n play**
  - **dynamic**
    cards, sinks, sources and streams tracked to maintain the logical nodes
  - **no extra configuration needed**
- **Helper modules**
  - **automatically loaded/unloaded**



Nodes

- Routing endpoints
  - both available and potential sinks/sources
    - single sink with a speaker and headset port is two logical node
    - BT card with a2dp and HFP profiles makes two logical node regardless that only one of the sinks are available
  - both the sink-input of an MP3 player and the source of a CD player makes a logical source node
- Properties
  - implementation (ie. device or stream)
  - direction (ie. input or output)
  - internal/external (ie. alltime connected/hotplug)
  - type
    - for devices: speaker, headphone, headset, hfgw etc
    - for streams: navigator, camera, game, phone, browser, player etc
  - channels
  - etc ...
- Mapping of logical Nodes to PA objects
  - device/input => source
  - device/output => sink
  - stream/input => sink-input
  - stream/output => source-output

Explicit v.s. default routes

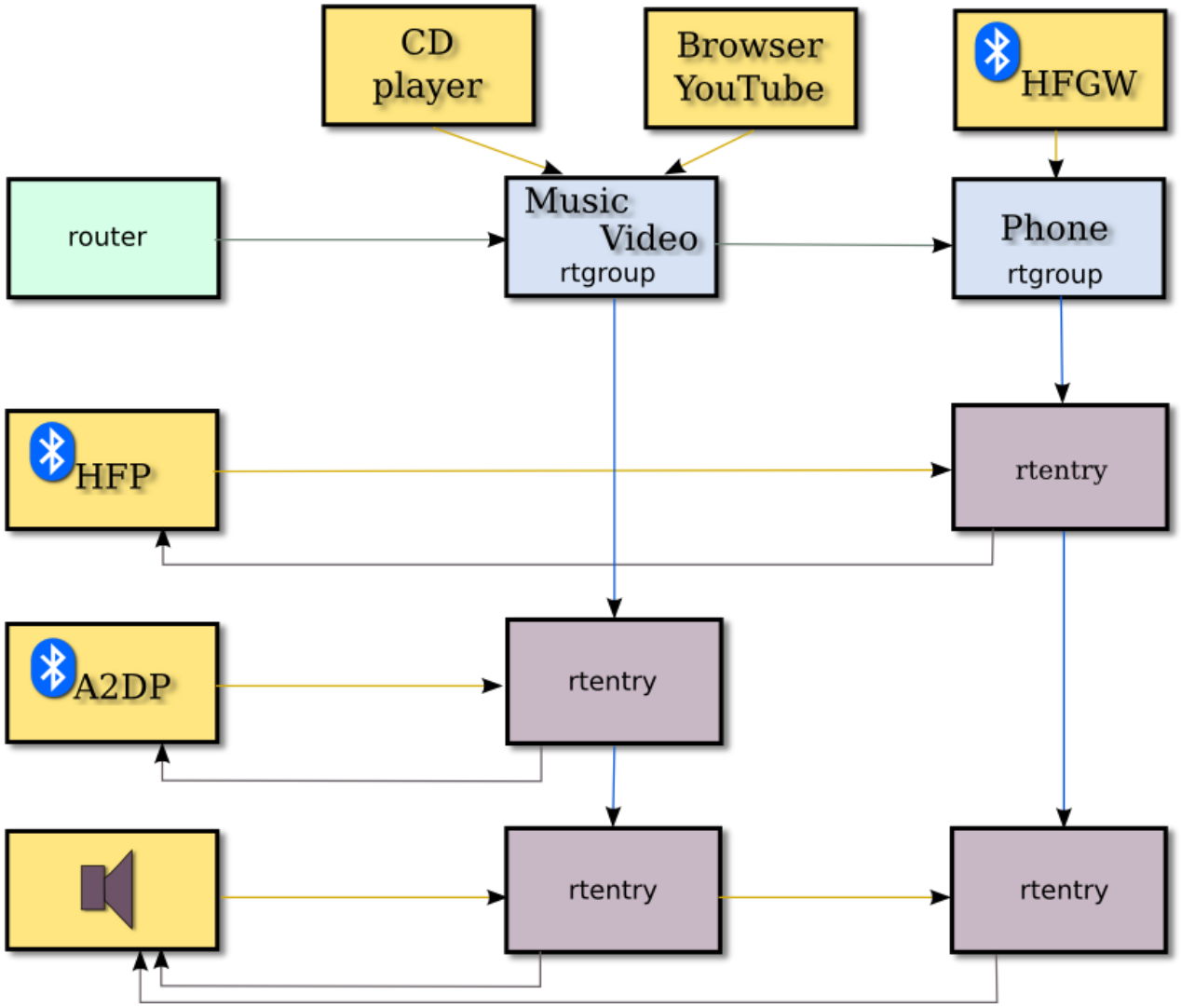
- explicit routes
  - user requested routes
    - via the extended native API
    - by setting the target sink at PA stream creation
  - a source node can have 0+ explicit routes
  - static
    - eg. connecting new headsets will not effect existing explicit routes
- default routes
  - automatic at stream creation
  - class based
    - classification is based on the ***media.role*** stream property
  - can be converted to explicit routes
  - 0 or 1 default route
  - dynamic,
    - eg. connecting a headset might change default routes

Priority based routing with conflict resolution

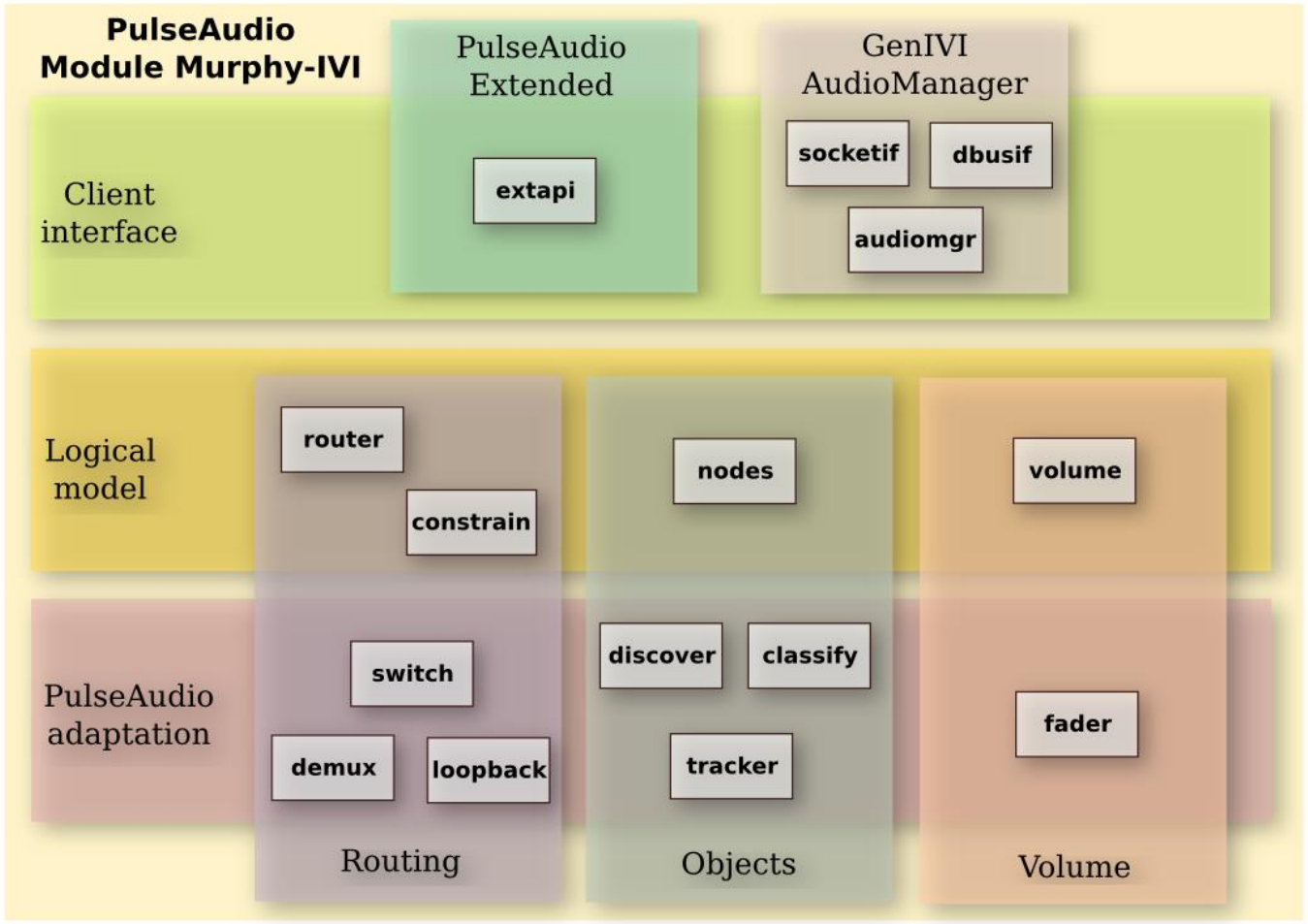
- priorities
  - explicit routes have always priorities over default routes
    - user requests are always honored
    - among explicit routes recent ones have priorities over older ones
  - default routing use class based stream priorities
    - coming from the ***media.role*** property
    - automatic priority assignement
    - class based routing target lists

- conflict resolution
  - higher priority routes might disable lower priority ones
  - walking through on streams in decreasing priority order to make the routing decisions
  - in case of conflicts
    - explicit routes are disabled
    - for default route the next available target on the routing target list

Default routing



Components


PulseAudio
Module Murphy-IVI
PulseAudio
Extended
extapi
GenIVI
AudioManager
socketif
dbusif
audiomgr
Client
interface
Logical
model
router
constrain
nodes
volume
PulseAudio
adaptation
switch
demux
loopback
discover
classify
tracker
fader
Routing
Objects
Volume

What could go upstream ?

- routing infrastructure
  - independent infrastructure
  - can be used for traditional routing ie. to manage PA objects directly
  - can be used with logical model
- logical model + switching fabric + protocol extension
  - part of the infrastructure
  - independent, optional layer (eg. as it is now)
- volume limit

Proposed features

- Multiple double linked list
  - somewhat similar what the Linux kernel uses
- Combine -> infrastructure
- Support for module devel package
  - protocol extension issues
- Zone property
- Infrastructure for method calls between PA modules