DAWD #9

The Tourist Guide to Ardour Data Structures

Second Things First

- IS-A
- If Foo IS-A Bar, then Foo was derived from Bar
- HAS-A
- If Foo HAS-A Bar, then Foo has at least one member that is a Bar
- HAS-A-PTR
- If Foo HAS-A-PTR, then Foo has at least one member that points to (“references”) a Bar

This Pointer Stuff

- If `foo` points at `bar`, what happens when/if bar is deleted?

```
Track* foo = bar;

delete bar;

foo->set_record_enabled (true); // what happens?
```

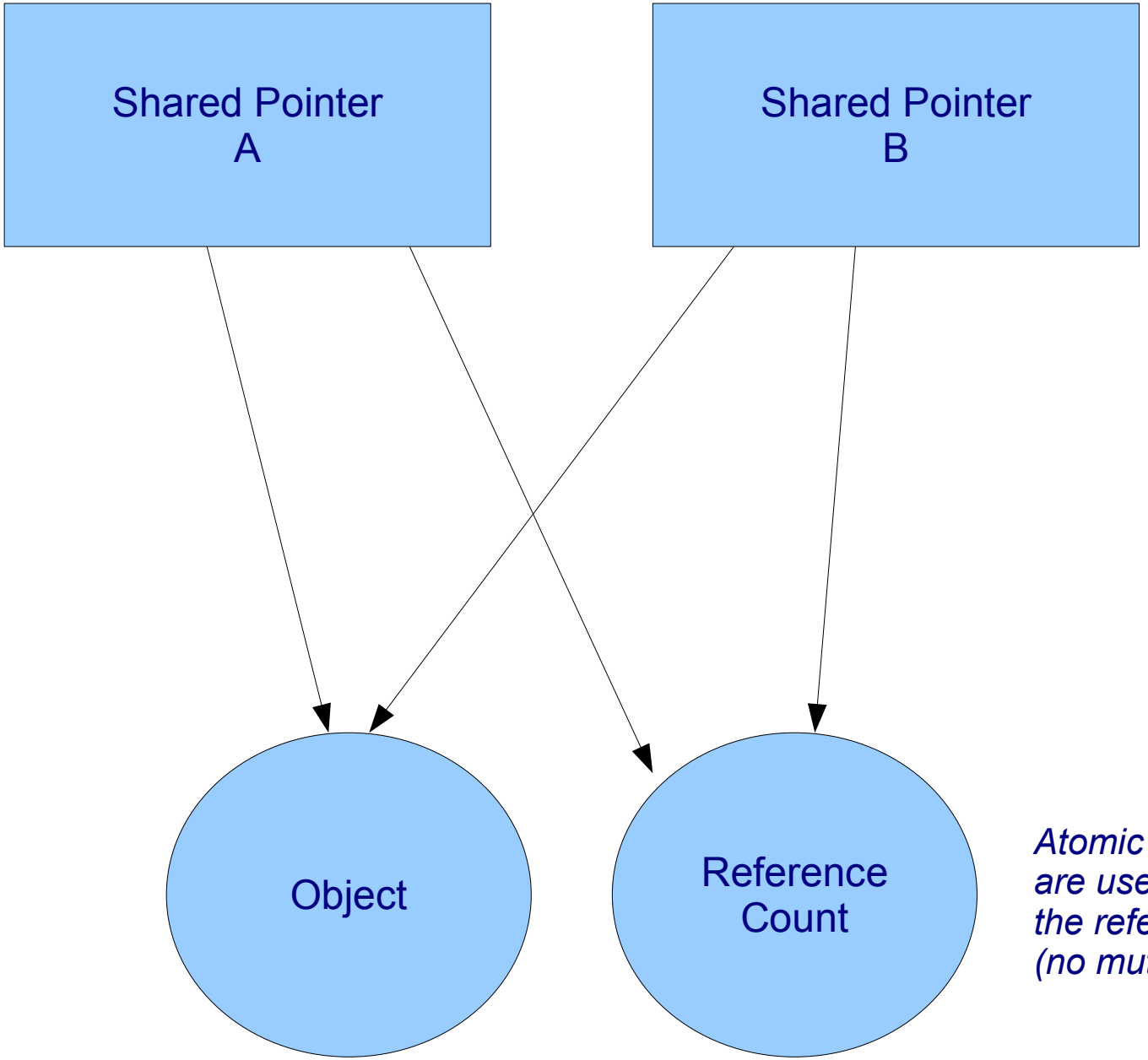
This Pointer Stuff 2

- We want `foo` to know that `bar` was deleted OR
- We want to stop `bar` from being deleted as long as `foo` points at it

Shared Pointers

- A pointer with a reference counter
- Every time a copy of the pointer is made (or assigned), the reference count increases
- Every time a pointer is deleted, the reference count is deleted
- The pointed-at object is deleted when the reference counter goes to zero
- `boost::shared_ptr<T>`

```
Foo* foo = new Foo ();
return shared_ptr<Foo> (foo);
```



Atomic operations are used to change the reference count (no mutexes/locks)

But ...

- Anonymous notifications

GOOD:

```
sigc::signal<void> SomethingChanged;

SomethingChanged.connect (mem_fun (aFoo, &Foo::handle_changes));
```

BAD:

```
sigc::signal<void> SomethingChanged;

SomethingChanged.connect (bind
    (mem_fun (aFoo, &Foo::handle_changes), someSharedPtr));
```

Why Is This Bad?

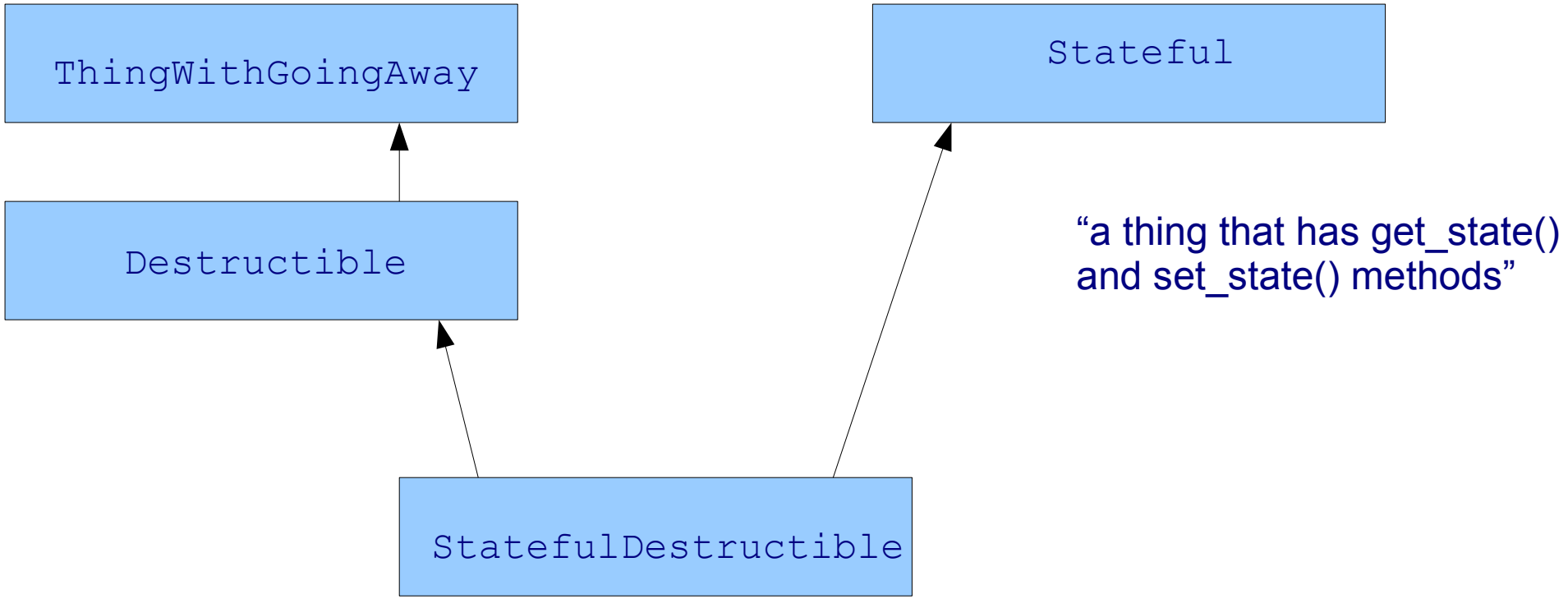
- The connection is stored in a “hidden” data structure (the “closure”)
- There is now a extra copy of the shared pointer
- We can delete every other instance of the shared pointer, but the reference count will never go to zero ....
- Object is never deleted!

```
struct hiddenImplicitClosure_1829291 {
        Foo* theFoo;
        void (Foo::*)(void) fooMethod;
        shared_ptr<Something> theExtraArgument;
}
```

How to solve this?

- Never use shared pointers when connecting to notifications of some kind
  - Use “weak pointers”
- Still need to solve the general issue – how do we arrange for destruction of objects in an MVC system that uses shared pointers?
- Shared pointer design assumes that objects will be deleted when the time is “right”
- MVC tends to require references from V/C to M
  - How do we (e.g.) delete a track and actually make sure it gets deleted?

The Basic Object Model



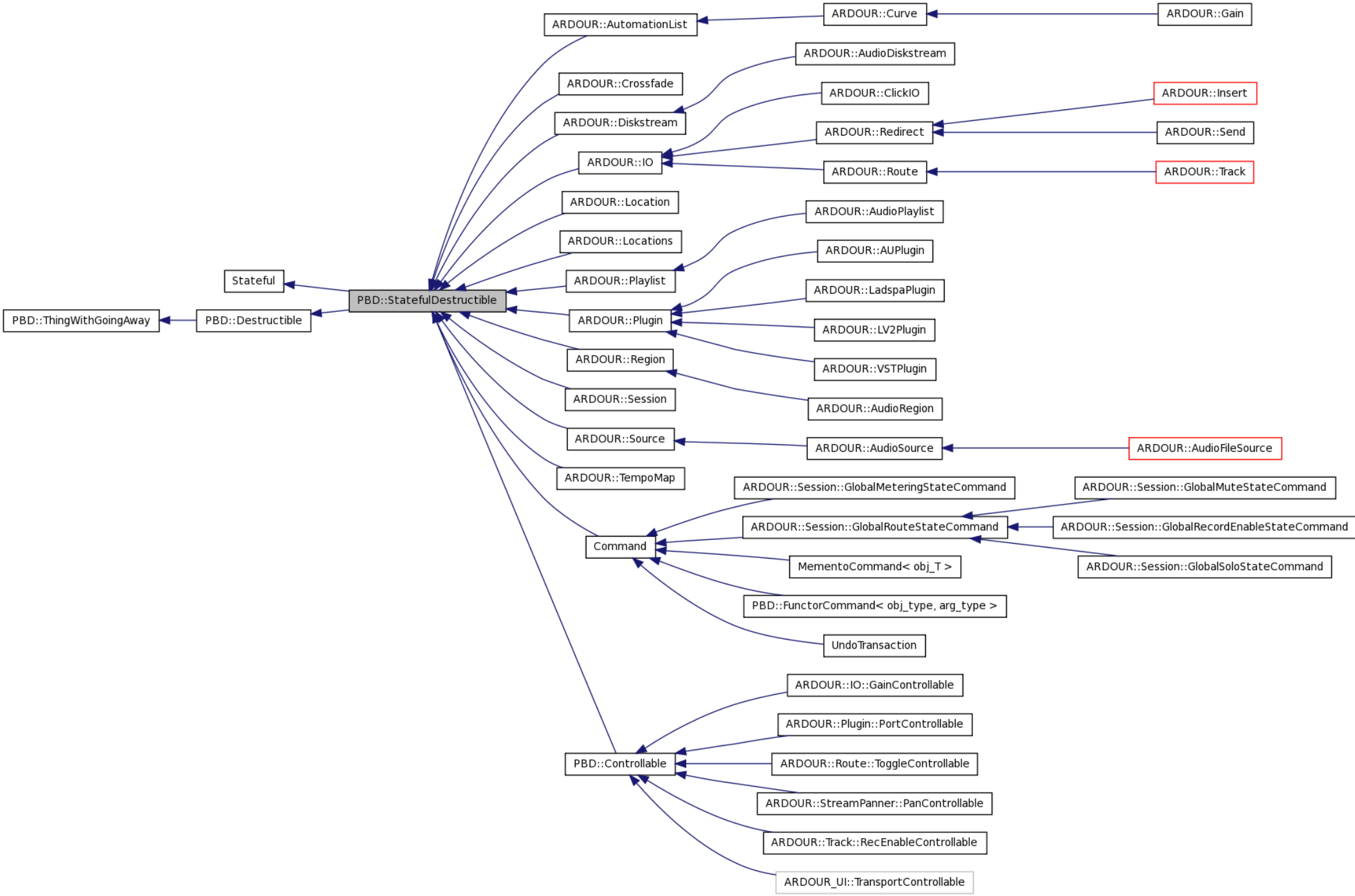
"a thing that has state, and signal that can be emitted when
it is destroyed, and whose destruction will cause it be disconnected
from all sigc++ signals it is connected to"

How To Destroy Something

- Add a method: `drop_references()`
  - This method emits `GoingAway`
- Anyone with shared_ptr's to this object must be connected to GoingAway
- On receipt of signal, destroy all shared_ptrs to the object
- At this point, the reference count should be close to zero
  - Delete your own shared_ptr
- Reference count goes to zero, object is deleted

One more thing: Shiva

- Sometimes we need to “couple” object destruction together
- “If A is destroyed, destroy B; if B is destroyed, destroy A”
- This is hard to do with either A or B
- Introducing Shiva, the Hindu god of chaos and destruction
- A “shiva” is an object that notices the destruction of 1 or both of two other objects; destroys the other object, and then destroys itself


ARDOUR::AutomationList
ARDOUR::Curve
ARDOUR::Gain
ARDOUR::AudioDiskstream
ARDOUR::Crossfade
ARDOUR::ClickIO
ARDOUR::Insert
ARDOUR::Diskstream
ARDOUR::Redirect
ARDOUR::Send
ARDOUR::IO
ARDOUR::Route
ARDOUR::Track
ARDOUR::Location
ARDOUR::AudioPlaylist
ARDOUR::Locations
ARDOUR::AUPlugin
Stateful
ARDOUR::Playlist
ARDOUR::LadspaPlugin
PBD::StatefulDestructible
PBD::ThingWithGoingAway
PBD::Destructible
ARDOUR::Plugin
ARDOUR::LV2Plugin
ARDOUR::Region
ARDOUR::VSTPlugin
ARDOUR::AudioRegion
ARDOUR::Session
ARDOUR::Source
ARDOUR::AudioSource
ARDOUR::AudioFileSource
ARDOUR::TempoMap
ARDOUR::Session::GlobalMeteringStateCommand
ARDOUR::Session::GlobalMuteStateCommand
ARDOUR::Session::GlobalRouteStateCommand
ARDOUR::Session::GlobalRecordEnableStateCommand
Command
MementoCommand< obj_T >
ARDOUR::Session::GlobalSoloStateCommand
PBD::FunctorCommand< obj_type, arg_type >
UndoTransaction
ARDOUR::IO::GainControllable
ARDOUR::Plugin::PortControllable
PBD::Controllable
ARDOUR::Route::ToggleControllable
ARDOUR::StreamPanner::PanControllable
ARDOUR::Track::RecEnableControllable
ARDOUR_UI::TransportControllable

Ardour Data Structures

- Top Down
- Session is the top

ARDOUR::Session

- Fundamentally, a Session is just a collection of other things
- Routes (tracks, busses)
- Locations
- Tempo map
- Non-JACK-related, per-session parameters
- Undo/Redo history
- MIDI ports
- Slaves
- Transport state & controls

ARDOUR::AudioEngine

- Fundamental abstraction of audio I/O and audio processing
- Hides JACK (could hide ASIO, CoreAudio)
- Session HAS-A-PTR to AudioEngine (and vice versa)
- Owns all “ports” (JACK, MIDI, private ardour-only ports)
- Provides transport time information

ARDOUR::IO

- An abstract class for objects that do input and output (audio and/or MIDI)
- Each IO has:
- Input port(s)
- Output port(s)
- Gain
- Panning (distributing signals to the output ports)
- Automation state
- Metering

ARDOUR::Route

- The basis of all signal flow in Ardour
  - Route IS-A ARDOUR::IO
  - Session HAS-A-PTR to all Routes
- A Route adds the following to ARDOUR::IO
  - Redirects
  - Mute, Solo

ARDOUR::Redirect

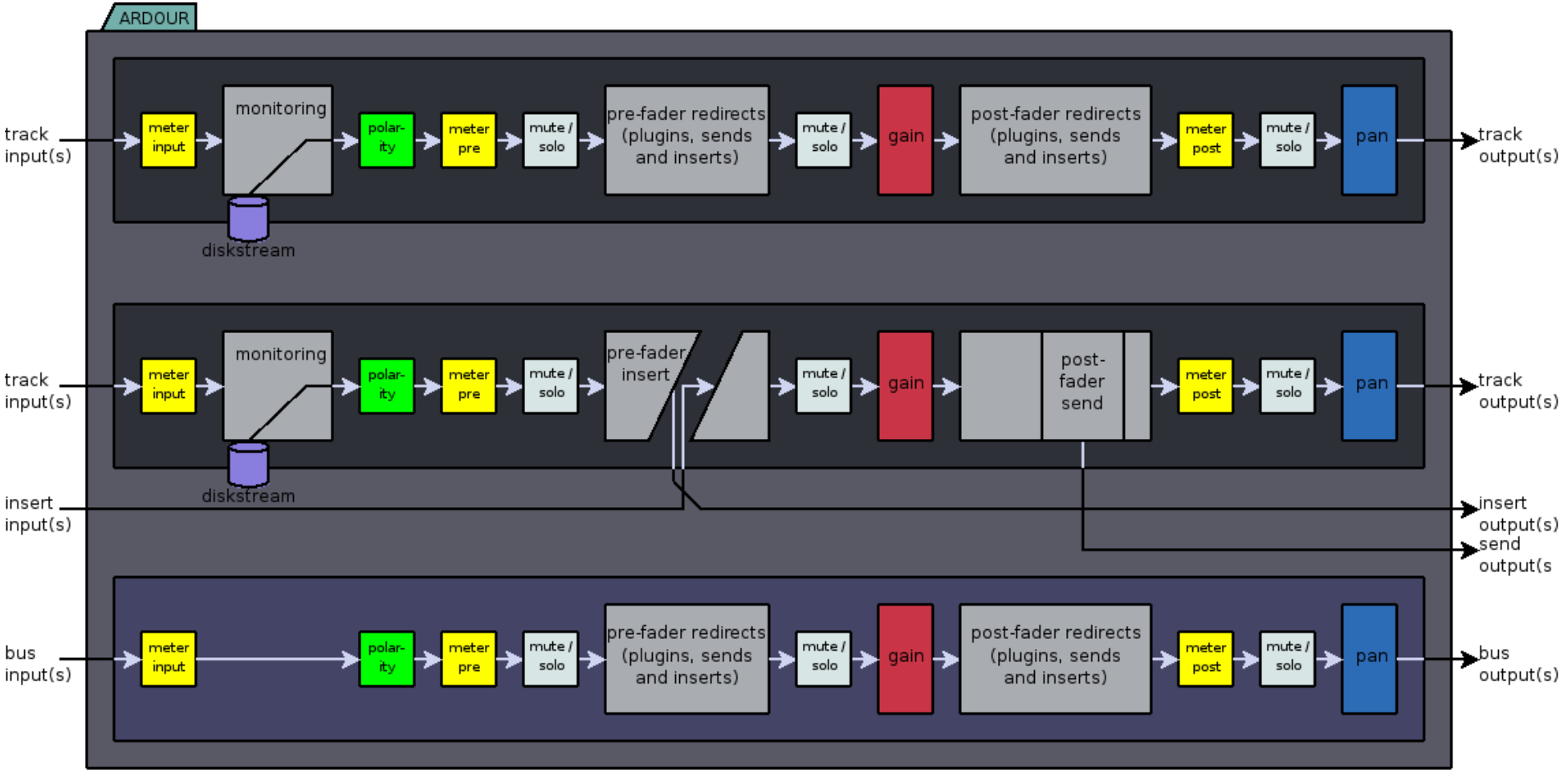
- Normal signal flow through a Route is:
- `Input -> gain -> pan -> output`
- Anything which changes this a “redirect”
- Abstract class
- 3 concrete classes: PluginInsert, PortInsert, Send
- Inserts deliver the signal to some object, then insert the output from the object back into the Route
- Sends have no effect on signal flow within the Route
- Has an anonymous pointer to a “GUI”

ARDOUR::Bus

- Doesn't exist
- A “bus” literally IS-A ARDOUR::Route (not derived, just a Route)

ARDOUR::Track

- Track IS-A Route
- Track HAS-A Diskstream
- Almost identical to Route but adds:
  - Record-enable
- Playback data can come from disk, not just input ports


ARDOUR
track input(s)
meter input
monitoring
diskstream
polar-ity
meter pre
mute / solo
pre-fader redirects (plugins, sends and inserts)
mute / solo
gain
post-fader redirects (plugins, sends and inserts)
meter post
mute / solo
pan
track output(s)
track input(s)
meter input
monitoring
diskstream
polar-ity
meter pre
mute / solo
pre-fader insert
mute / solo
gain
post-fader send
meter post
mute / solo
pan
track output(s)
insert input(s)
insert output(s)
send output(s
bus input(s)
meter input
polar-ity
meter pre
mute / solo
pre-fader redirects (plugins, sends and inserts)
mute / solo
gain
post-fader redirects (plugins, sends and inserts)
meter post
mute / solo
pan
bus output(s)

ARDOUR::PluginInsert

- PluginInsert IS-A Insert IS-A Redirect
  - PluginInsert HAS-A Plugin(s)

ARDOUR::Plugin

- Abstract class
- Defines **(virtual)** interface
- Get/set parameters
- Configure number of input/output signals
- Get the name and other information
- `connect_and_run` ()
- Automation of parameters

Actual Plugin objects

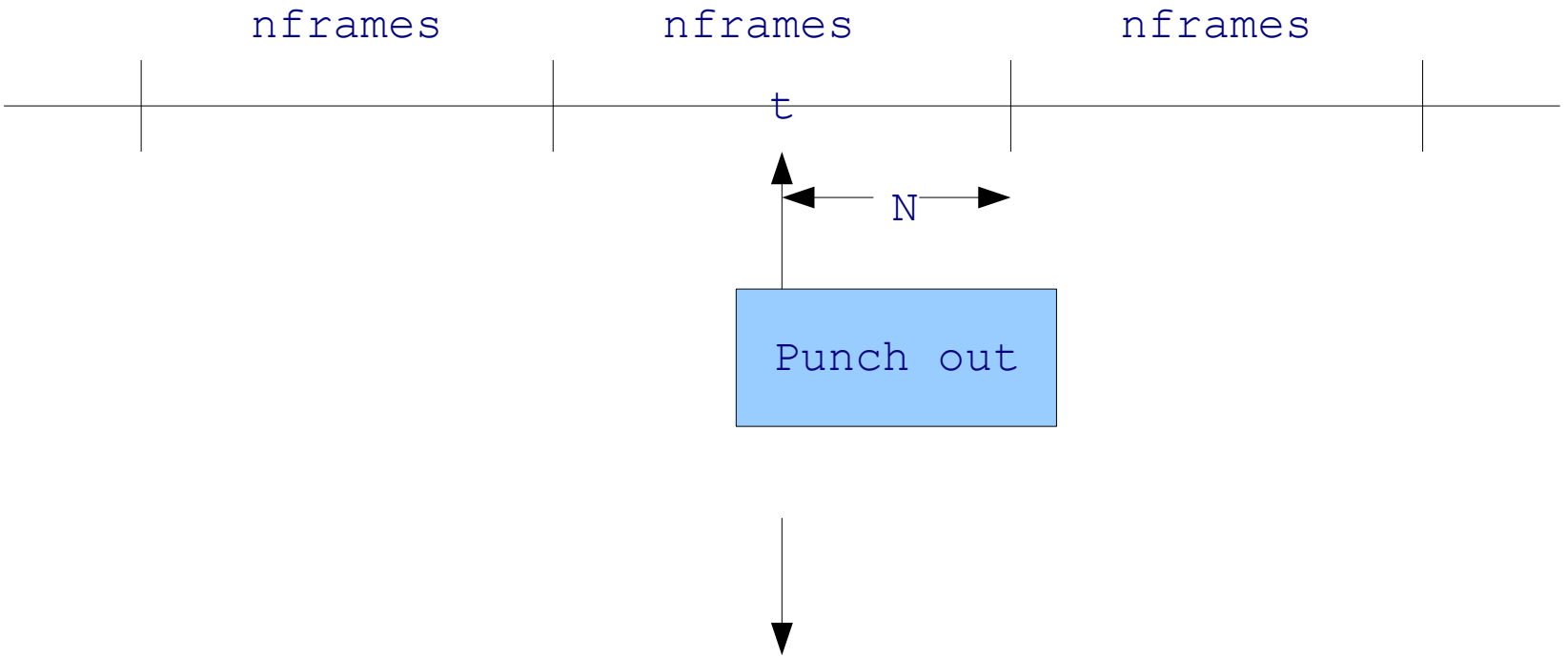
- ARDOUR::LadspaPlugin
- ARDOUR::AUPlugin
- ARDOUR::VSTPlugin
- Each one implements the interface (API) defined by ARDOUR::Plugin

Automation

- ARDOUR::ControlEvent: time (fractional samples) value (double-precision float)
- ARDOUR::AutomationList HAS-A list of ControlEvent
- Some AutomationLists HAVE-A ARDOUR::Curve
- Curve is an interpolator
- Changing automation data means modifying the contents of an AutomationList, so AutomationList provides cut/copy/delete/paste/move methods

2 Kinds of Automation

- 1) Sample-accurate “streaming” automation
  - Relies on Curve::get_vector()
  - Interpolates from the ControlEvents and returns a vector of values that can be applied per-sample
  - Used for gain & pan automation
- 2) Event automation
  - These are handled by putting them in the Session “event list”.
  - Noticed during the JACK process() callback
  - Subdivide the process() callback into N parts, “implement” the event in between
  - Used for transport control & plugin automation


nframes
nframes
nframes
t
N
Punch out
Process nframes-N
do punch out stuff
Process N frames

Controllable

- Anything that can be controlled by something else
  - 3 key methods:
    - `get_value()`
    - `set_value()`
    - `can_send_feedback()`

Actual Controllables

- IO::GainControllable
- Plugin::ParameterControllable
- Route::ToggleControllable
- StreamPanner::PanControllable
- Track::RecEnableControllable
- Each one has a `set_value()` method that does something different

MIDIControllable

- HAS-A Controllable
- HAS-A MIDI port
- Methods to start/stop “learning”
- Method to send feedback
- All Ardour plugin parameters are Controllables

ARDOUR::Diskstream

- A way to move data to/from disk
  - Big (lock-free) ringbuffers
    - HAS-A Playlist

ARDOUR::Playlist

- Playlist HAS-A list of Regions
- Session HAS-A list of Playlists
- Has methods to:
- Read (virtual)
- cut/copy/paste/partition/split
- Find regions at ...
- Note: data type agnostic
- AudioPlaylist handles audio data via read()

ARDOUR::Region

- An object defining data to be played back
- Session HAS-A list of pointers to Regions
- Region members: start,position, length
- Other properties: opaque, muted, sync position
- Methods to change length, positions
- Note: a Region defines its own position in a playlist, which means that edit methods in Playlist have to set this
- Abstract class, data type agnostic
- AudioRegion handles audio data

ARDOUR::AudioRegion

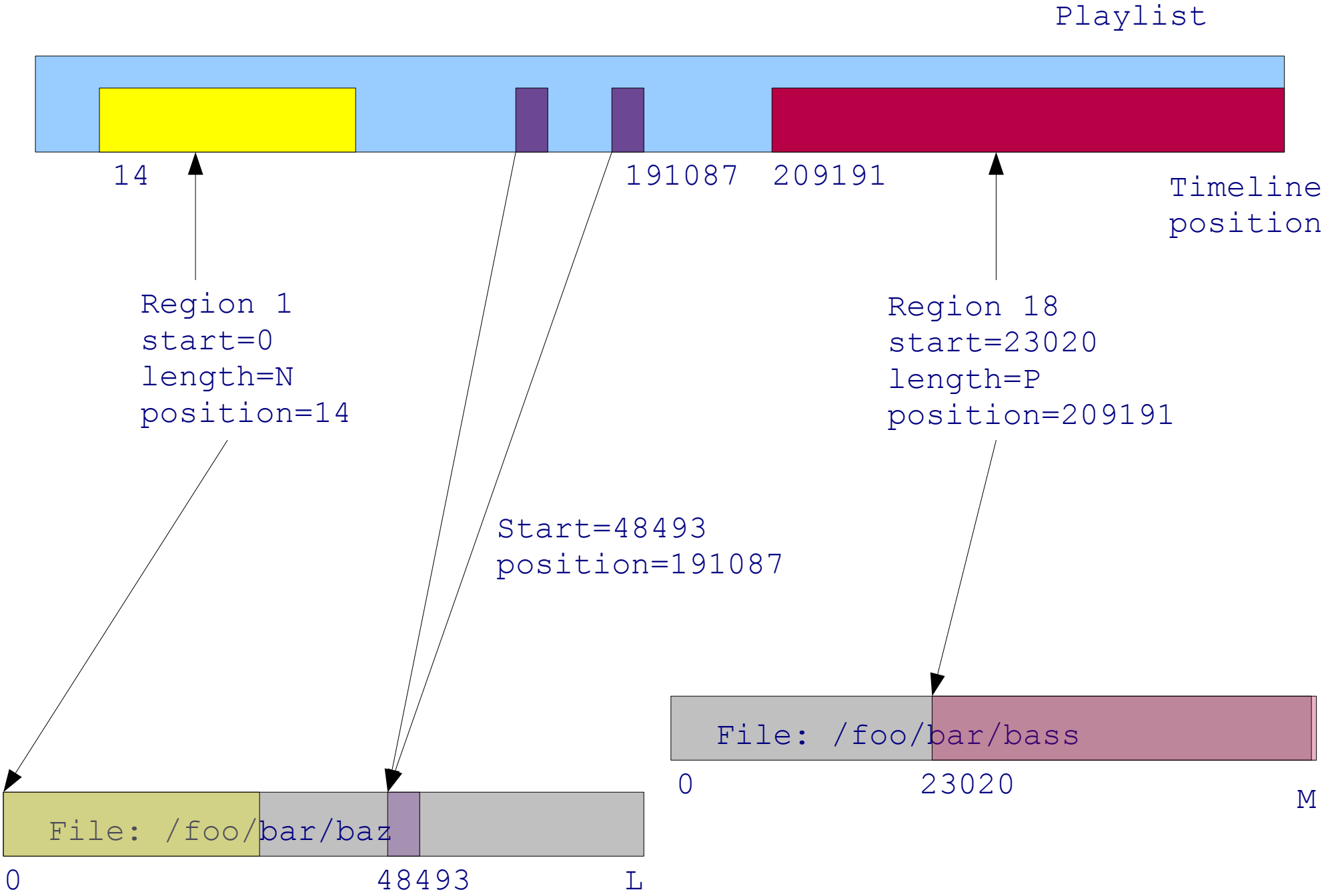
- AudioRegion IS-A Region
- AudioRegion HAS-A Source
- Adds gain envelope, fade in/out curves
- Methods to normalize, apply destructive processing
- Constructed via RegionFactory because we only want to refer to regions via shared pointers

ARDOUR::Source

- Abstract class
- A place to read data from (and possibly write it too)
- Normally disk files but not required
- Data type agnostic
- AudioSource handles audio data

ARDOUR::AudioFileSource

- Abstract class
- Defines API for file based audio I/O (i.e. virtual read/write methods)
- Handles peak file construction and access
- Concrete derived classes: SndFileSource, CoreAudioFileSource
- These use other libraries (libsndfile, ExtAudioFile) to actually implement audio file I/O as necessary


Playlist
14
191087
209191
Timeline position
Region 1
start=0
length=N
position=14
Region 18
start=23020
length=P
position=209191
Start=48493
position=191087
File: /foo/bar/baz
0
48493
L
File: /foo/bar/bass
0
23020
M