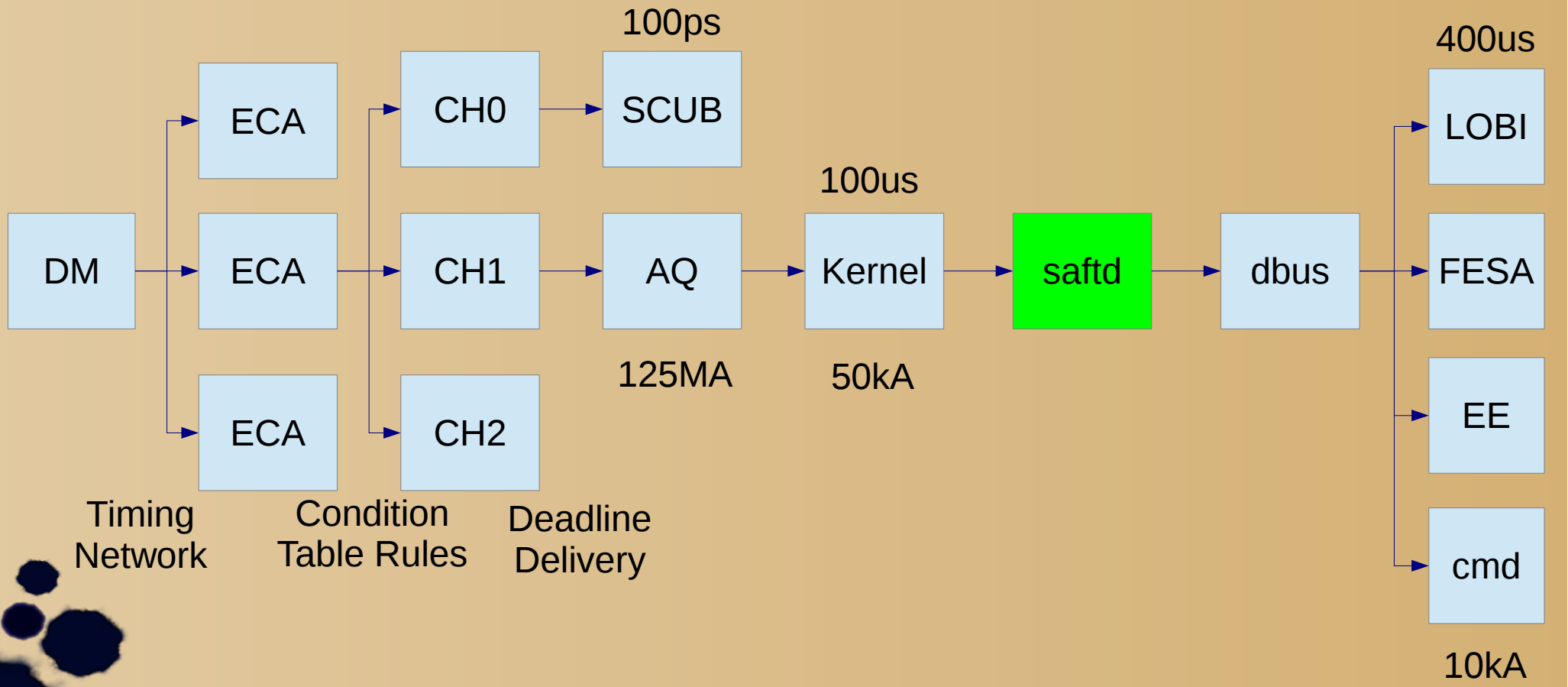
Simple API for Timing (saftlib)

https://github.com/terpstra/saftlib

Recap: Timing System

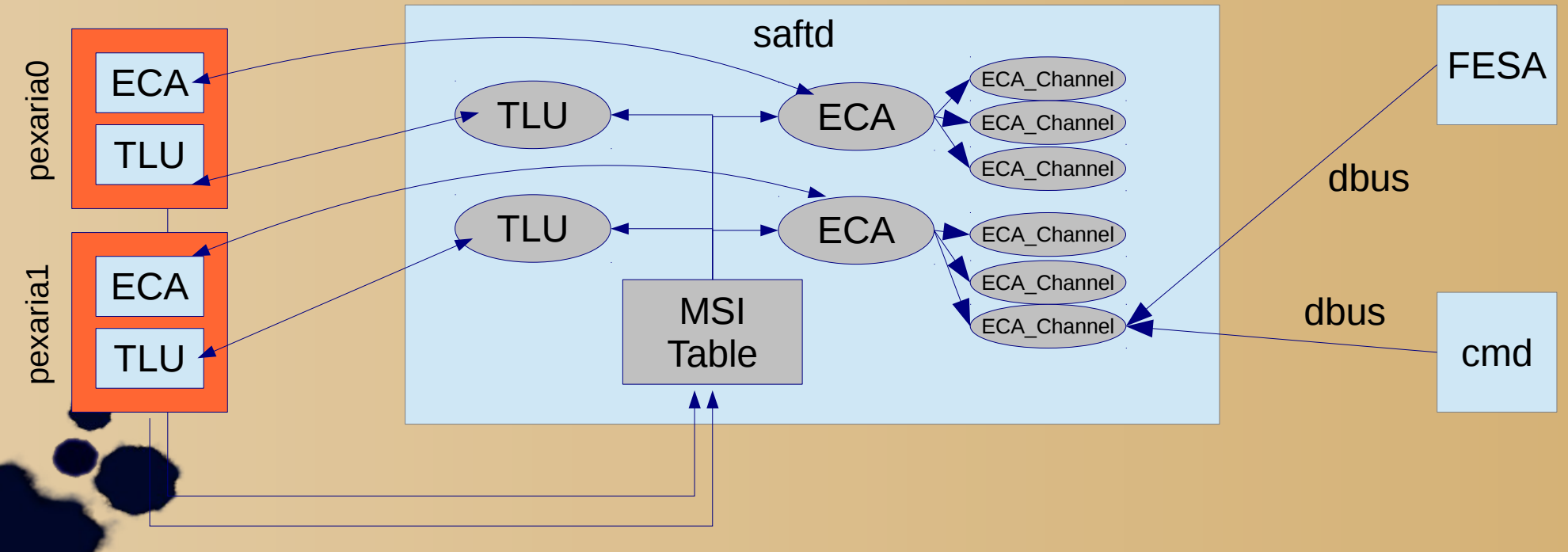
- Data master sends event messages describing future work
- Receiver h/w matches events against locally configured conditions
- Local actions are generated by receiver and scheduled for delivery
- Actions are delivered at hardware layer through channels
- One channel (the action queue) feeds software via interrupt
- ***Actions are distributed to interested client programs (saftlib)***
- FESA/etc executes non-real-time code in response to actions

Recap: Timing System



Saftd: One shared object per controlled item

- Hardware is discrete and must be shared; e.g., table entries combined
- MSI must be demultiplexed in software; multiple devices+listeners



D-Bus Basics

- Object-Oriented – provides a collection of objects

- Each d-bus client program has an address - like an IP (:1.22)
  - Some clients have well-known names – like DNS (de.gsi.saftlib)
- Objects have a path – like a file (/de/gsi/saftlib/ECA_pex0_0)
- Objects implement interfaces - like Java (de.gsi.saftlib.ECA_Channel)
- Interfaces have: methods, properties, and signals

- Method calls need all of the above: address + path + interface + method

Client programs communicate via proxies

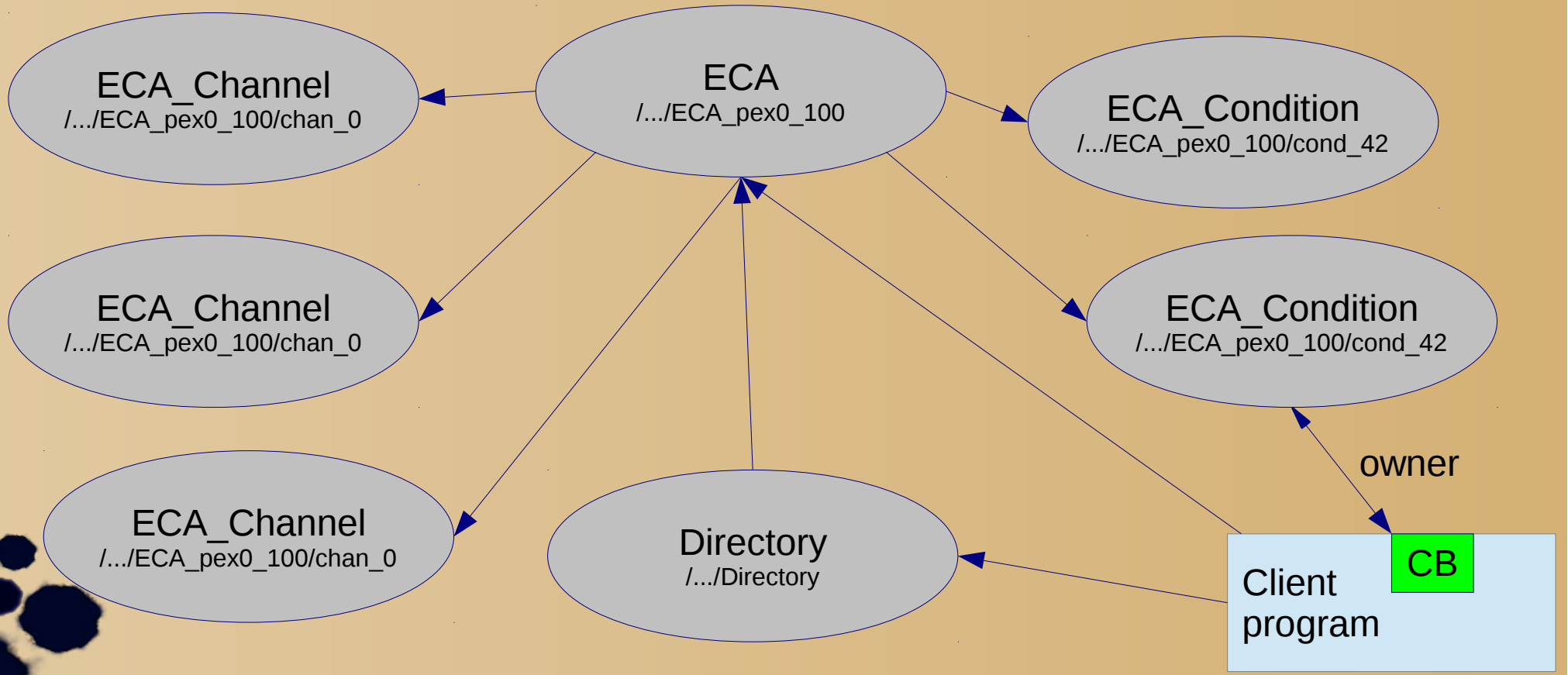
- The real objects reside inside saftd
- Client programs access the objects via IPC – dbus

- auto proxy = saftlib::ECA_Proxy::create(path); // path from directory
- proxy->CurrentTime(time); // method call
- proxy->getFree(); // property access
- proxy->Overflow.connect(my_callback); // signal

ECA in Saftlib

- ECA Event fields
  - Event ID : command executed by data master
  - Parameter : payload/context of event for timing receivers
  - Time : time of execution
- Client programs create conditions
  - Matches when event ID falls within specified range/prefix
    - Produces an action with specified delay (execution = time+offset)
  - Can be tied to a software callback (dbus signal)
  - Can output a tag to a hardware channel
    - Tags indicates what the hardware should do

ECA Objects appear under /de/gsi/saftlib/



Saftlib has your back

- Saftlib's ECA driver promises the following:
  - Software conditions never conflict (for one event)
    - All requested conditions will be honoured via conceptual multicast
  - Hardware conditions never conflict (for one event)
    - The second application in a conflict never succeeds at NewCondition
  - Every application is isolated; only resource exhaustion is shared
    - Conditions for application that crash are automatically purged
  - All possible hardware failure modes can be monitored via signals

The (for one event) clause is why less DM events = better

New plugins are easy

- Saftlib is The Place to put drivers for shared resources

1. Create an interface XML file (dbus IDL)
   - Generates documentation, header, proxy, and service stub
2. Implement the methods in the interface via overload
   - Just some etherbone cycles...
3. Add a probe function to find your Wishbone core
   - Saftlib core manages IRQs, devices, drivers, and d-bus export for you

Complete TLU driver is just 193 lines of code

Questions?

https://github.com/terpstra/saftlib