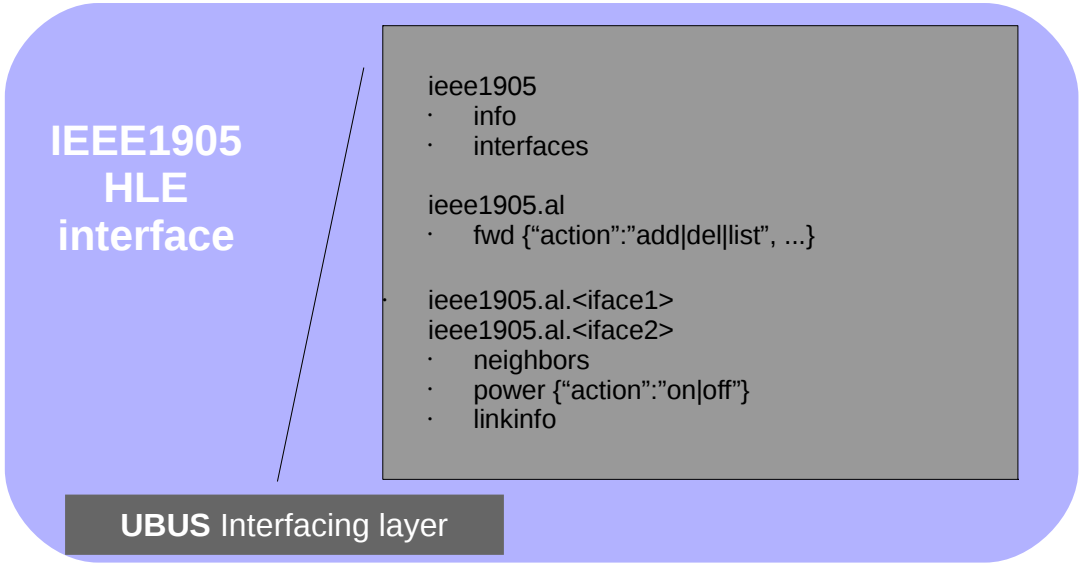
IEEE1905 + MultiAP-r2 architecture - IOPSYS

IOPSYS IEEE1905

- In a 1905 device, how HLE communicates with the 1905AL is implementation specific. It can be through any IPC mechanism.
- In iopsysWRT, HLE communicates to the IEEE1905 AL through UBUS.
  - IEEE1905 ALME SAP is implemented through ‘ieee1905*’ UBUS objects and methods.
  - The UBUS Interfacing Layer may invoke the underlying network interfaces’ specific APIs to complete a HLE request.
  - IEEE1905 ALME may create CMDU(s) if a specific HLE request requires it to do so. For example, topology query CMDU(s) may be generated by the ALME if HLE makes a get_neighbors UBUS request.



IOPSYS implements the HLE interface through ieee1905* UBUS object(s).

All communication between ieee1905 devices in a multi-AP network will happen through standard CMDUs.

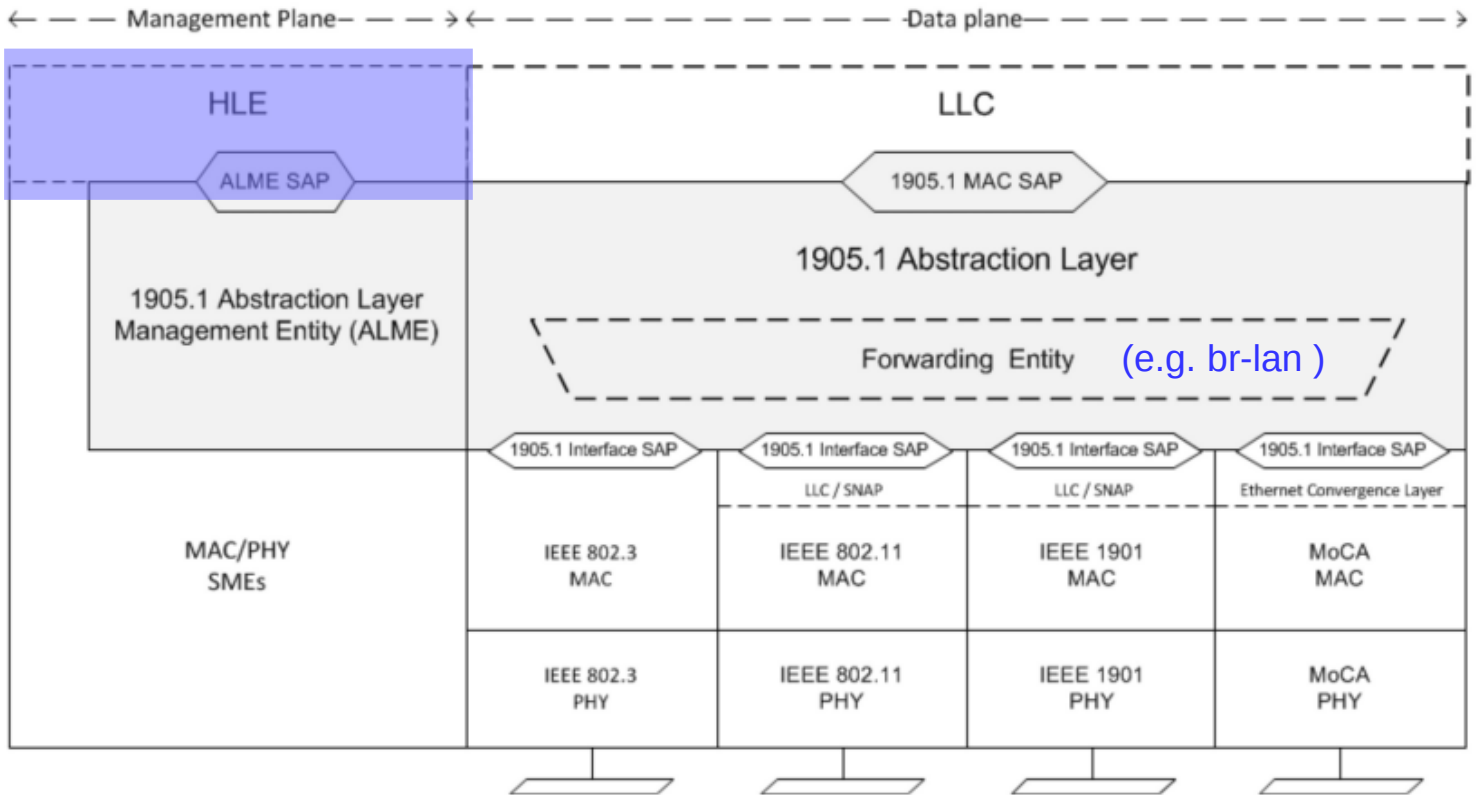


Figure 4-2—1905.1 abstraction layer model

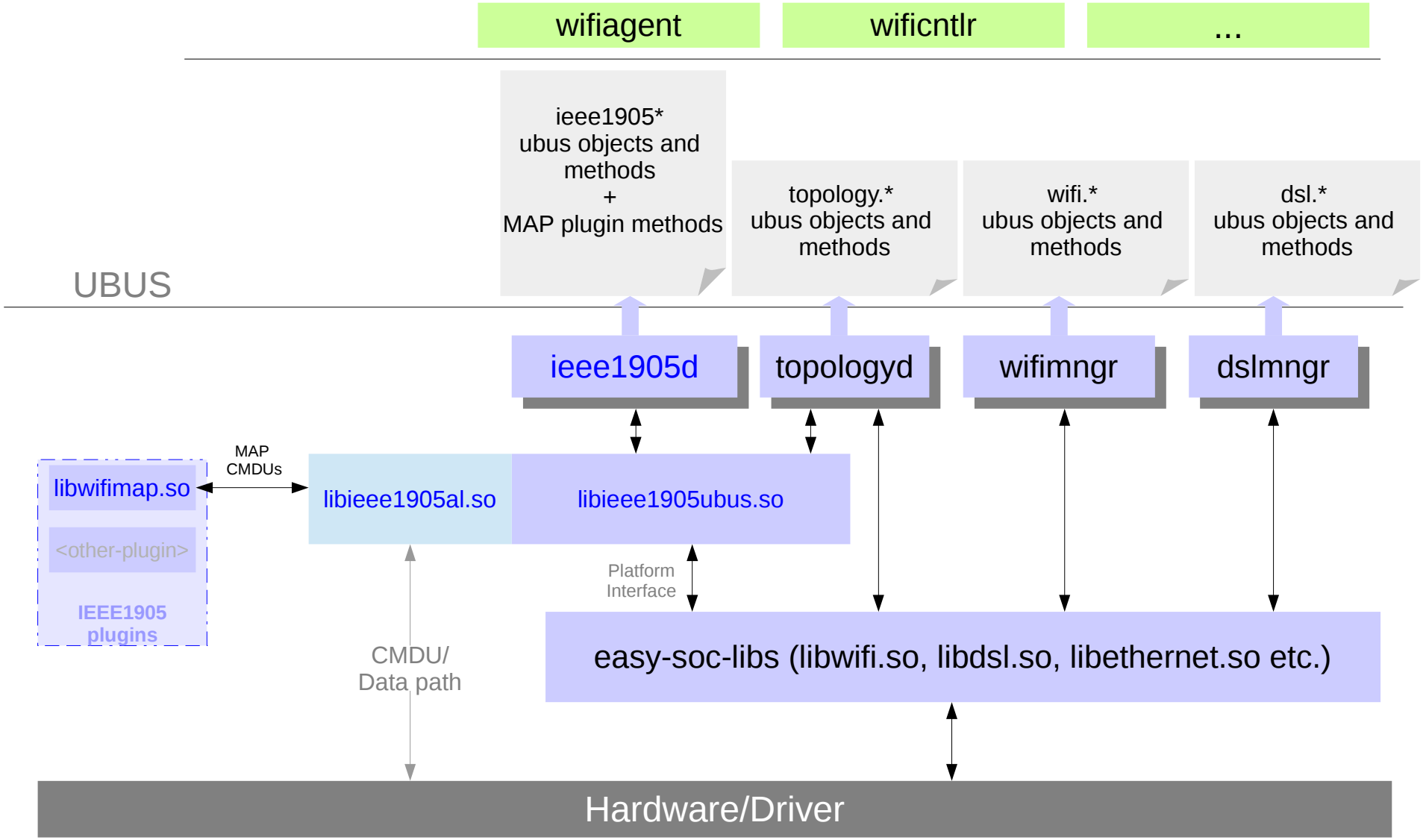
IOPSYS IEEE1905 stack

- IEEE1905 stack is implemented fully in the user space.
- The AL is implemented as a shared library (libieee1905al.so).
- The UBUS Interfacing Layer is also implemented as a shared library (libieee1905ubus.so).
- User daemon 'ieee1905d' is responsible to start/stop the IEEE1905 stack.
  - Script '/etc/init.d/ieee1905 start' starts the ieee1905 stack, and
  - '/etc/init.d/ieee1905 stop' will stop it.
- During startup, the 'ieee1905d' configures the 1905 AL with the network interfaces it reads from a UCI config file '/etc/config/ieee1905'. It then creates the corresponding ieee1905* UBUS objects.
- During exit, it unregisters the network interfaces from the 1905 AL and destroys the UBUS objects.
- Communication between HLE and the 1905 ALME happens through the ieee1905* UBUS objects and methods.

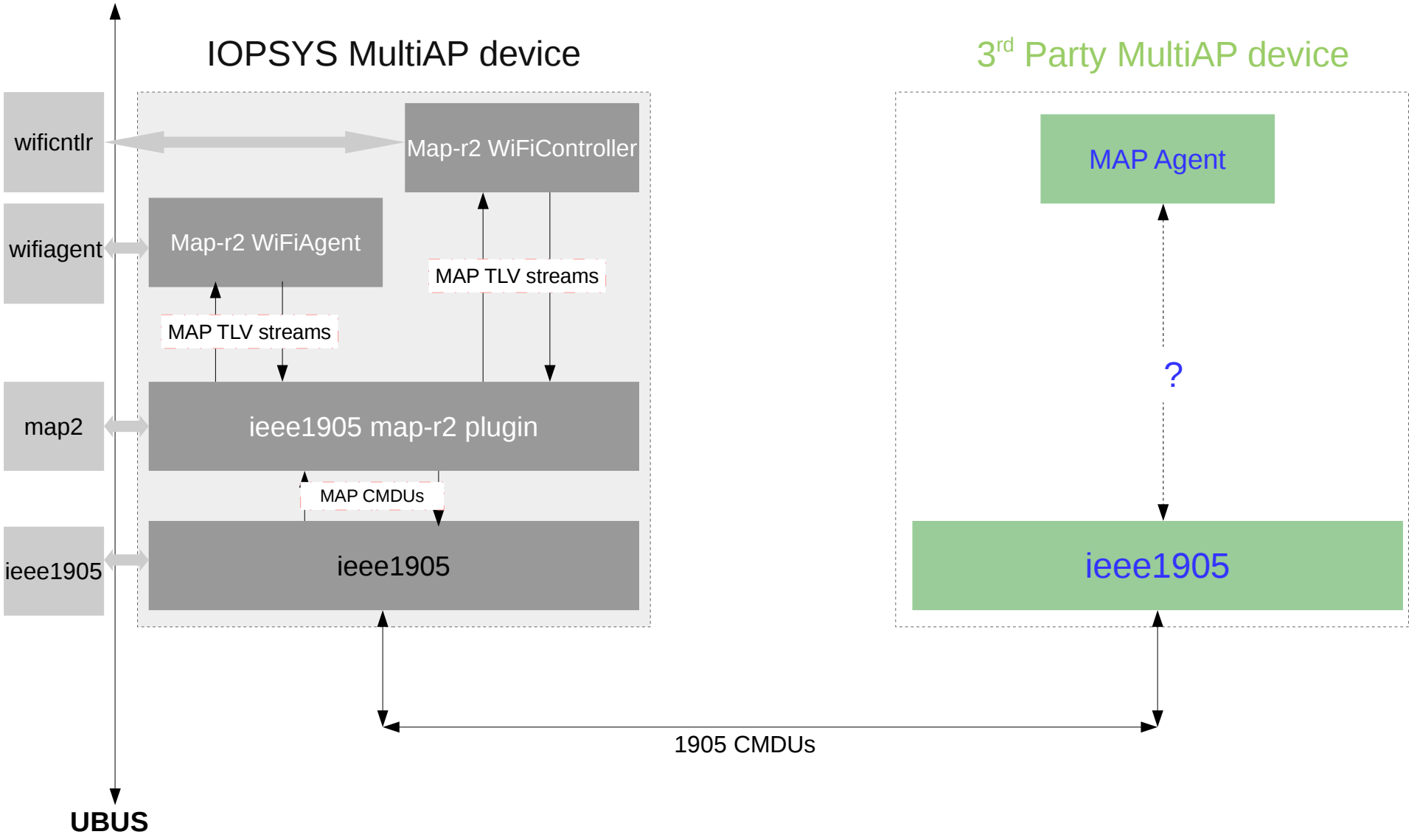
Multi-AP integration

- Multi-AP (MAP) is implemented in a shared library (libwifimap-2.so), separate from the IEEE1905 stack.
- ‘libwifimap-2.so’ exposes well defined APIs, which a MAP Agent can use to perform MAP functions and use cases.
- ‘libwifimap-2.so’ is a IEEE1905 plugin, which can expose additional MAP specific APIs through ‘libieee1905ubus.so’ over UBUS.

IEEE1905 + Multi-AP modules



IOPSYS vs. 3rd Party Multi-AP



Boot and startup - page1

Do_Init_wifiagent:

- /etc/init.d/wifiagent start
- Reads config file ("/etc/config/wifiagent") to know about -
  fh-iface, bk-iface, onboarding status, 1905al, etc.
- Call **Do_Cond_Init_ieee1905** if not already running.
  [NOTE: wificntlr may have started it, or the earlier wifiagent could have crashed].
- Register itself with "1905map2" plugin with **MAP_AGENT** role.
- Initializes its own core.
- Notifiy "1905map2" plugin that it is ready to process CMDUs.
- If *onboarded == false*,

  Then

  Call **Do_Onboarding_wifiagent**

  Else

  Call **Do_APAutoconfig_wifiagent**

Boot and startup - page2

Do_Onboarding_wifiagent:

- forall bk-iface,
  do
    If bk-iface == WIFI,
    Then
      Do_WPS(bk-iface)
    Else
      Update bk-iface as *onboarded = 1*
  done.

- Call **Do_APAutoconfig_wifiagent**

Do_APAutoconfig_wifiagent:

- forall fh-iface,
  do
    Send **ap_autoconfig_search**(fh-iface)
  done.

Do_Rxhandle_wifiagent:

- Verify CMDU type and call appropriate handler functions.
  [handler functions peform TLVs processing as per MAP2 Spec]

Boot and startup - page3

Do_Cond_Init_ieee1905:

- Get list of fh-iface and bk-iface which will be part of 1905 stack.
  [wifiagent updates the config file "/etc/config/ieee1905" after it knows about fh-ifaces and bk-ifaces from its conifg].
- Check for availability of plugins (t.x. 1905map2) and loads them.
- Create **1905*** ubus objects for 1905 stack management, control and status.
- Prepare 1905 AL, like setup rx handlers, msg-queues etc.
- Start 1905 AL.

Boot and startup - page4

Do_Init_wificntlr:

- /etc/init.d/wificntlr start
- Read config file ("/etc/config/wificntlr") to know about -
  fh-iface credentials, bk-iface credentials for supported wifi bands, data-elements collection interval, default policy for wifiagents in the network etc.
- Call **Do_Cond_Init_ieee1905** if not already running.
- Register itself with 1905map2 plugin with **MAP_CONTROLLER** role.
- Initialize its own core.
- Notifiy 1905map2 plugin that is is ready to process CMDUs.
- Call **Do_APAutoconfig_wificntlr**

Boot and startup - page5

Do_APAutoconfig_wificntlr:

- Send ap_autoconfig_search with supported role Registrar.
- Call **Do_Rxhandle_wificntlr**

Do_Rxhandle_wificntlr:

- Verify CMDU type and call appropriate handler functions.
  [handler functions peform TLVs processing as per MAP2 Spec]