The Auxiliary Bus

Bi-directional event handlers

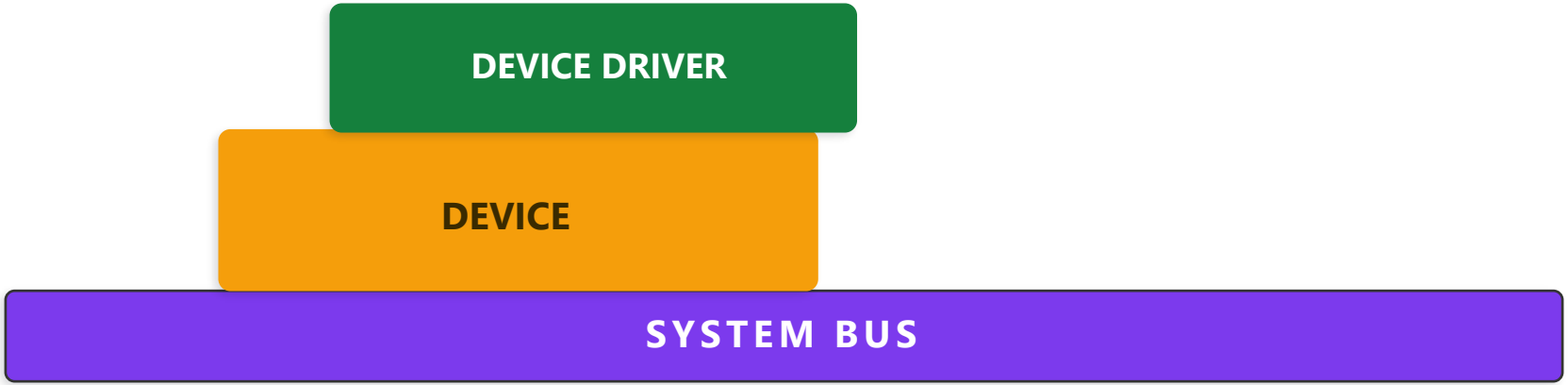
Netdev Conference 0x1A

Start with the System Bus

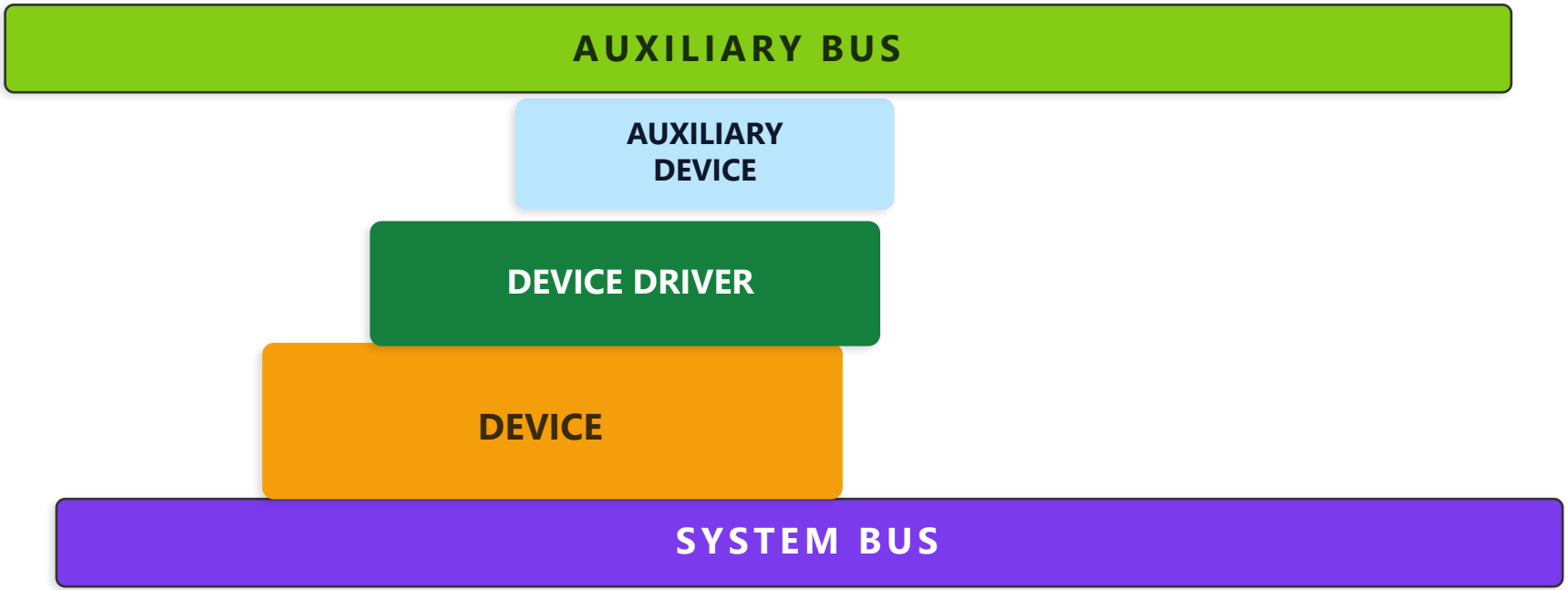
SYSTEM BUS

A Device Attaches to the Bus

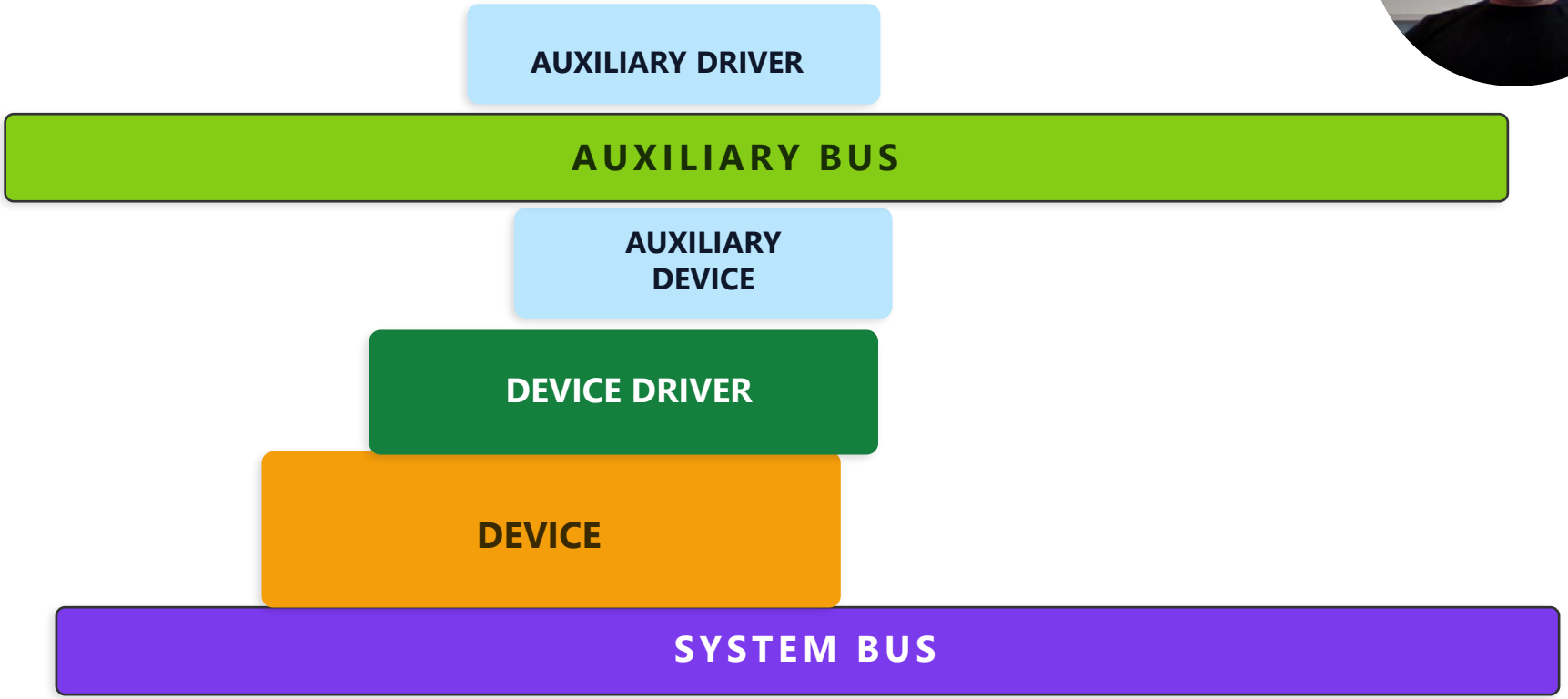
The Device Driver Binds



Split Out an Auxiliary Device



The Auxiliary Bus & Driver



COMMON HEADER FILE

Auxiliary Device Container

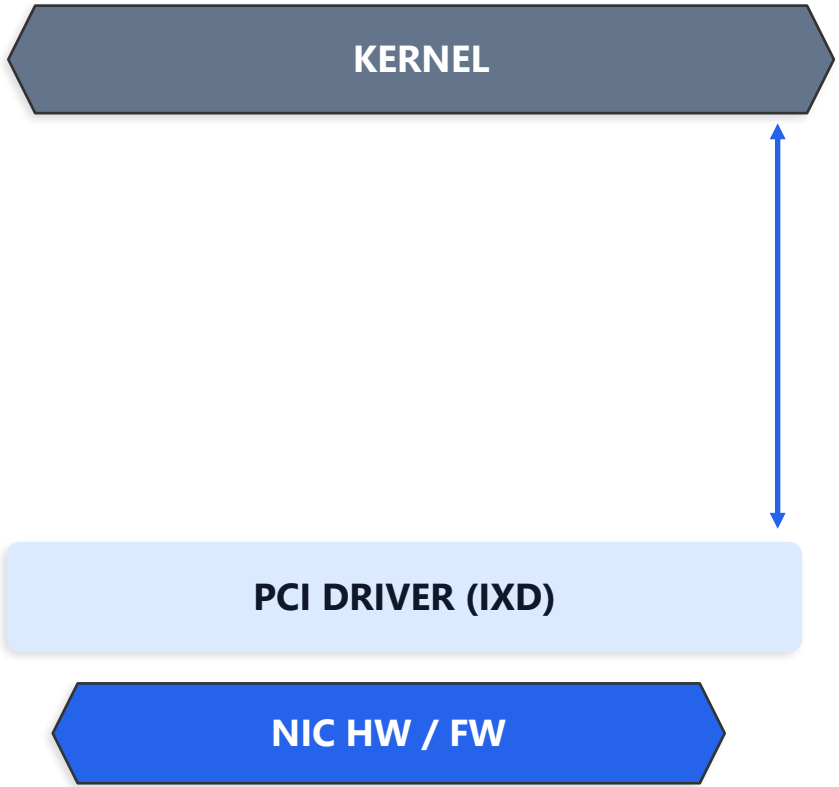
```
/* COMMON HEADER FILE in include/linux/ ... */

my_auxiliary_info {
    struct info_about_device my_struct;
}

my_auxiliary_device {
    struct auxiliary_device adev;
    struct my_auxiliary_info info;
};
```

COMMON HEADER FILE

Auxiliary Driver Container

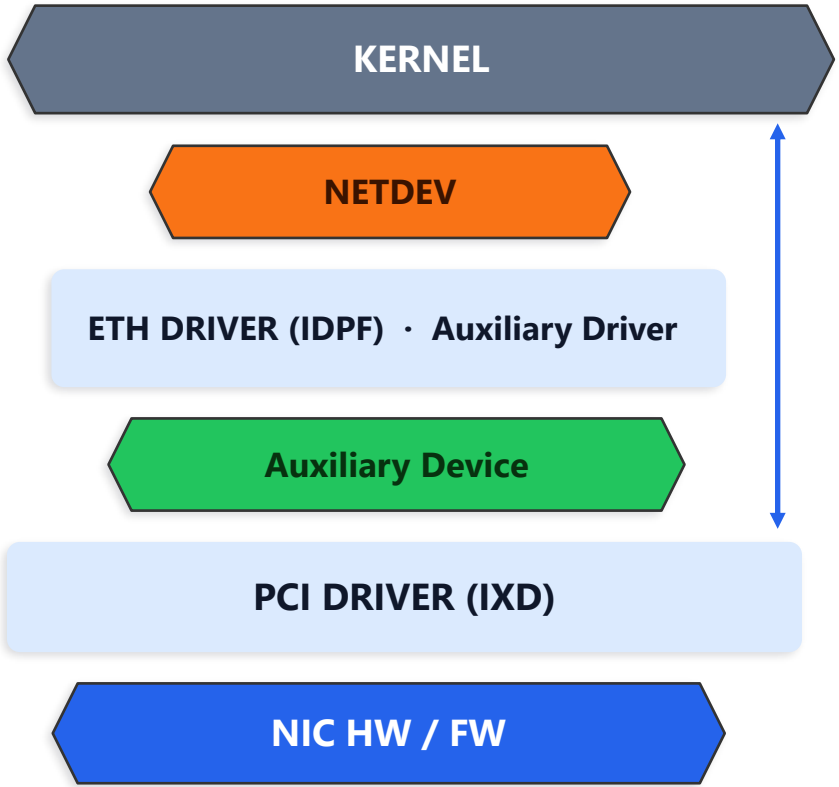
```
/* COMMON HEADER FILE in include/linux/ ... */
my_auxiliary_info {
    struct stuff_about_device my_struct;
}
struct my_aux_event {
    void *event_data;
    enum my_aux_event_codes event_code;
};
my_auxiliary_device {
    struct auxiliary_device adev;
    struct my_auxiliary_info info;
};
my_auxiliary_driver {
    struct auxiliary_driver adrv;
    int (*event_handler)(struct my_auxiliary_info *info,
                         struct my_aux_event *event);
};
```

Example: NIC Driver Stack



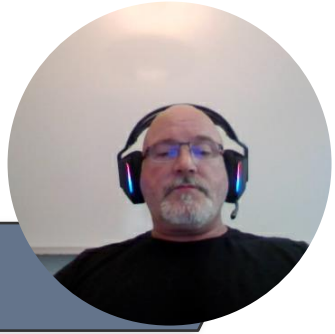
APPLIED: NIC DRIVER STACK

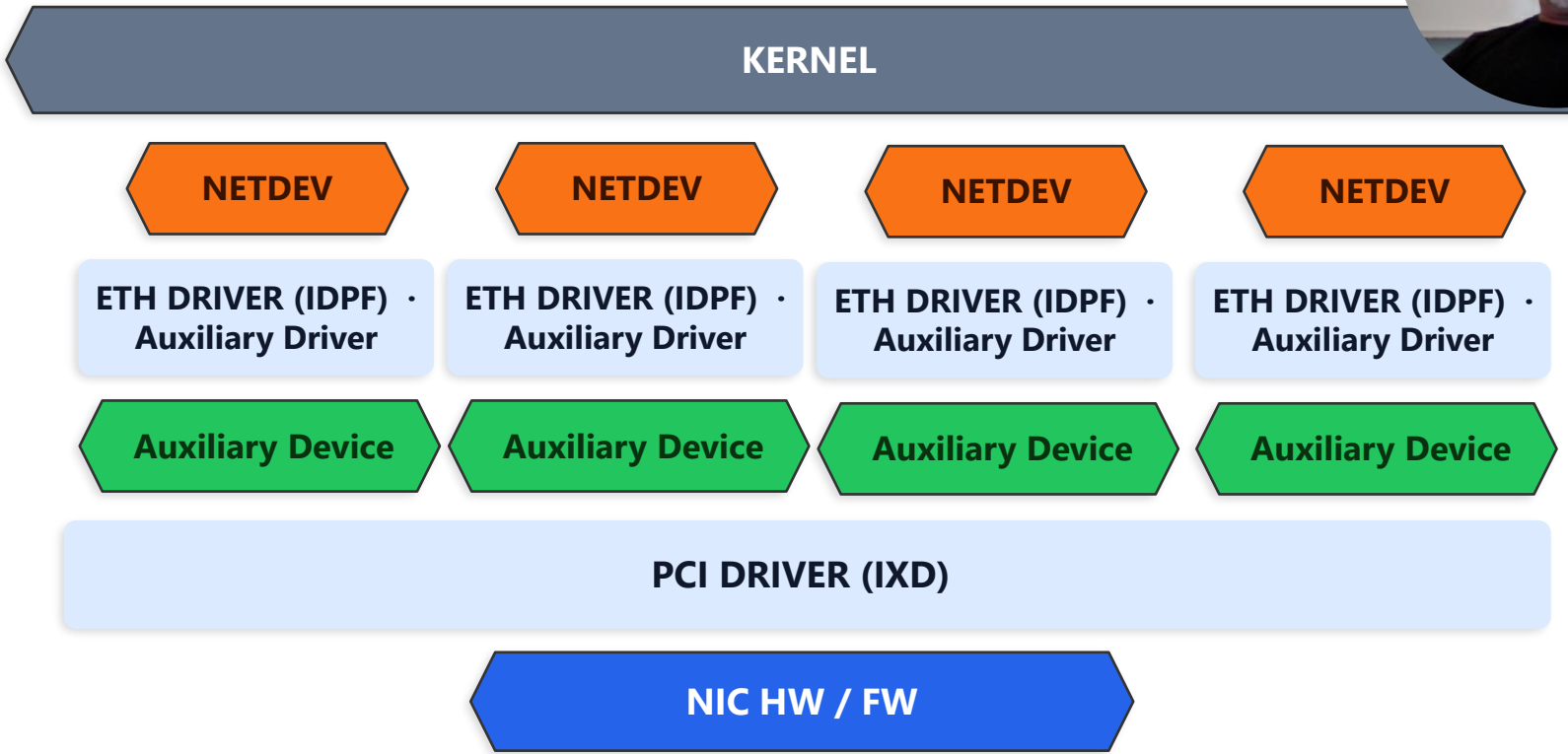
IDPF as an Auxiliary Driver



APPLIED: NIC DRIVER STACK

IDPF as an Auxiliary Driver





Managed by PCI Driver

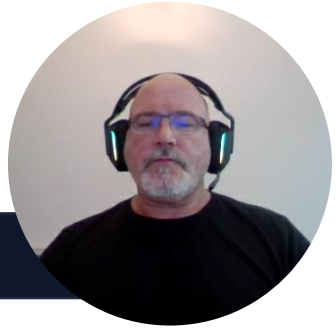
- Communication with FW
- SW DB of scheduling tree
- FLR/DLR flows
- Coodination between auxiliary devices
- Global Resources

Managed by Auxiliary Driver

- NetDev resources
- Local queue representation

APPLIED: NETWORK SHAPING

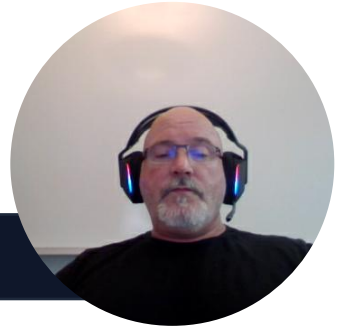
Shaping a Queue from Userspace



```
#> netshaper set dev eth0 handle scope queue id 3 bw-max 1gbit
```

APPLIED: NETWORK SHAPING

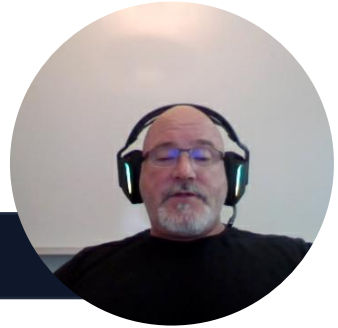
The Command Reaches NETDEV



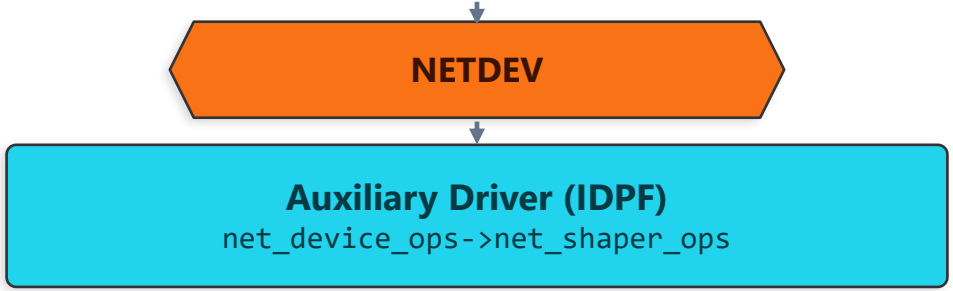
```
#> netshaper set dev eth0 handle scope queue id 3 bw-max 1gbit
```

APPLIED: NETWORK SHAPING

NETDEV Dispatches to the Auxiliary Driver

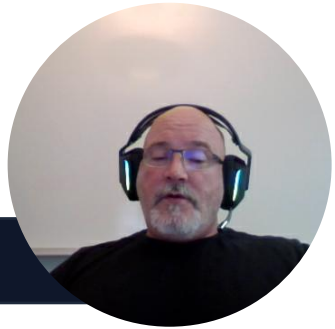


```
#> netshaper set dev eth0 handle scope queue id 3 bw-max 1gbit
```

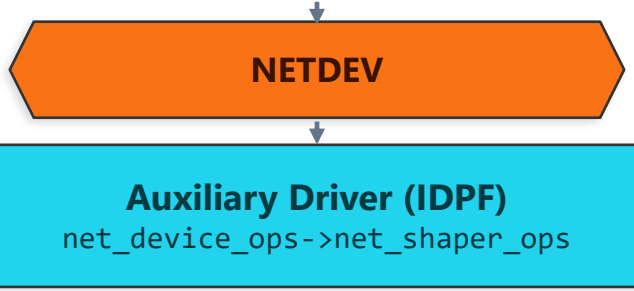


APPLIED: NETWORK SHAPING

Need to Program the HW Scheduling Tree



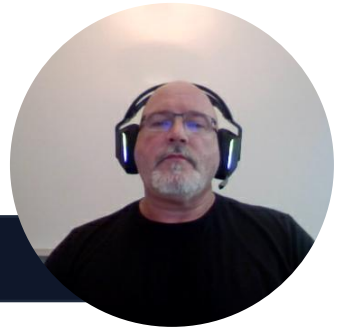
```
#> netshaper set dev eth0 handle scope queue id 3 bw-max 1gbit
```

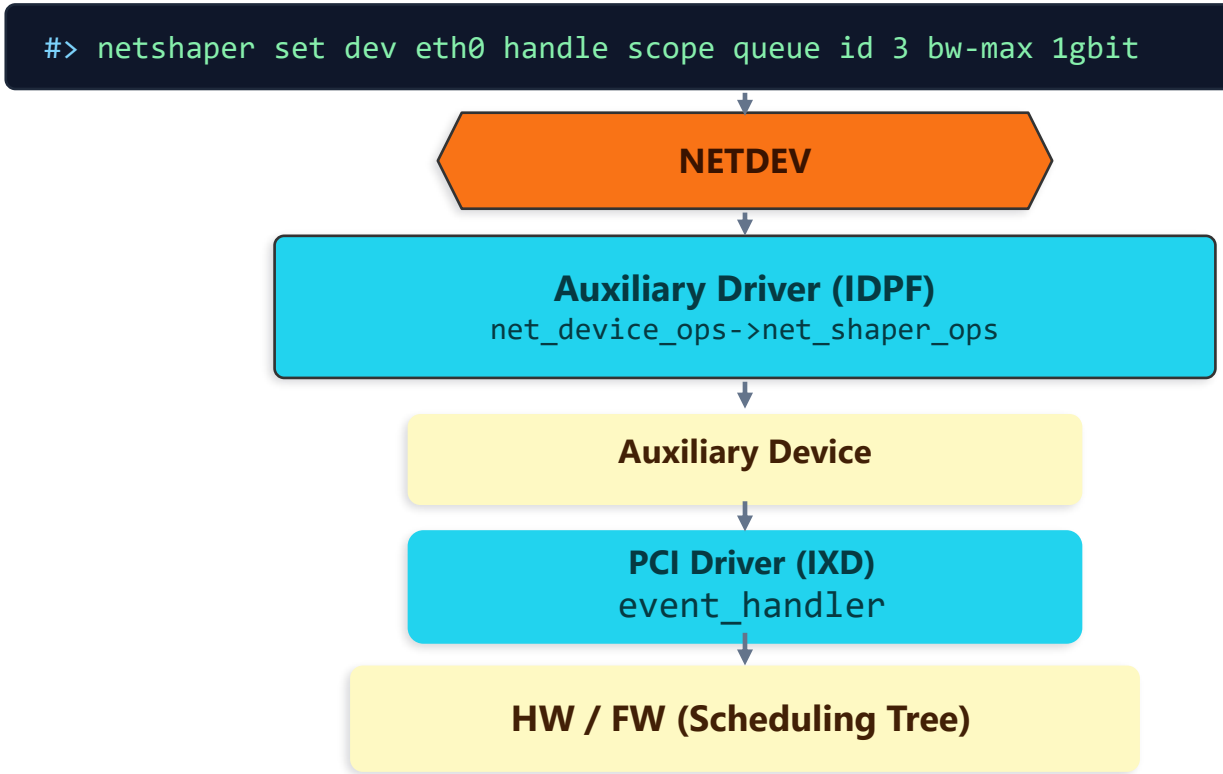


HW / FW (Scheduling Tree)

APPLIED: NETWORK SHAPING

Full Path Through the Auxiliary Device





Common Header with PCI Event Handler

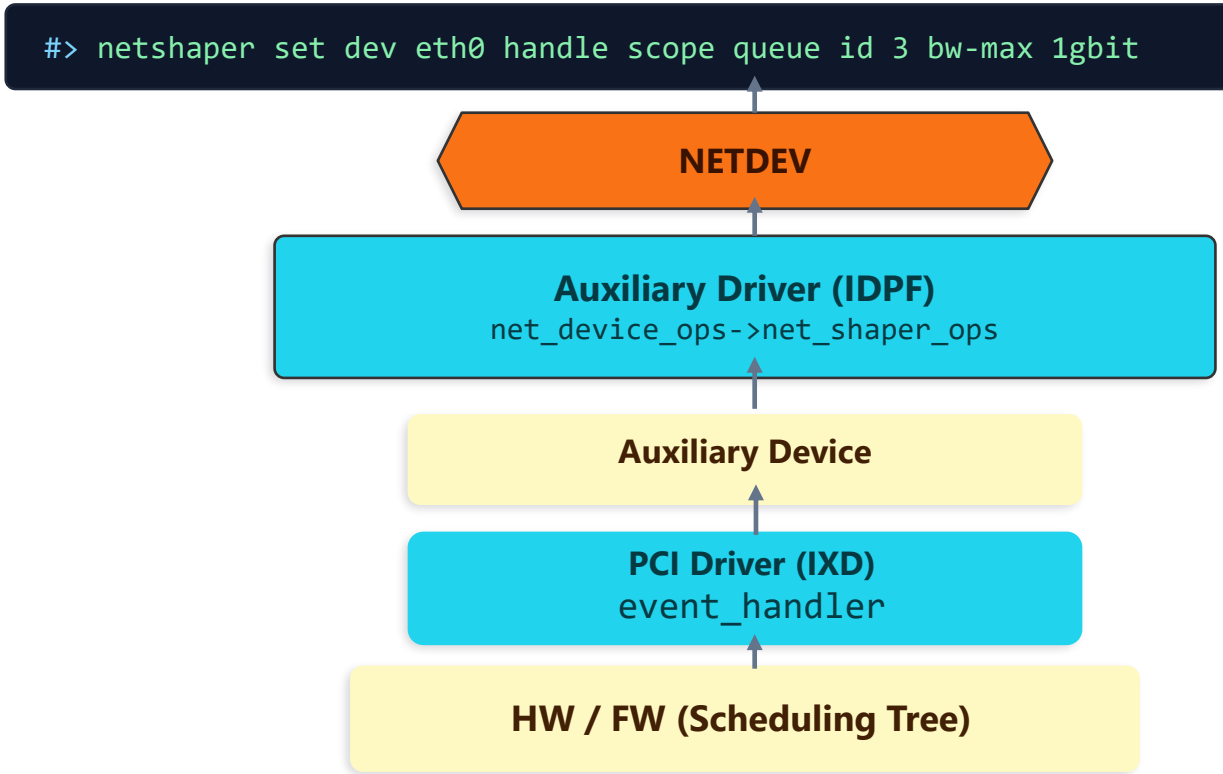
```
/* COMMON HEADER FILE in include/linux/ ... */
my_auxiliary_info {
    struct stuff_about_device my_struct;
}
struct my_aux_event {
    void *event_data;
    enum my_aux_event_codes event_code;
};
my_auxiliary_device {
    struct auxiliary_device adev;
    struct my_auxiliary_info info;
    int (*event_handler)(struct my_auxiliary_info *info,
                         struct my_aux_event *event);
};
my_auxiliary_driver {
    struct auxiliary_driver adrv;
    int (*event_handler)(struct my_auxiliary_info *info,
                         struct my_aux_event *event);
};
```

Event Codes & Rate-Limit Payload

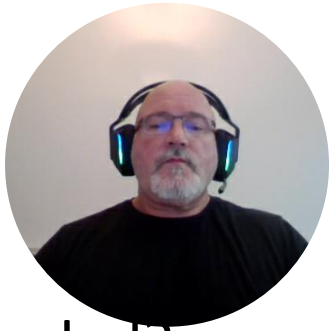
```
enum my_aux_event_codes {
  /* PCI driver -> aux driver */
  MY_AUX_ETH_EVENT_LINK_CHANGE,
  MY_AUX_ETH_EVENT_RESET_INITIATED,
  MY_AUX_ETH_EVENT_RESET_COMPLETE,
  /* aux driver -> PCI driver */
  MY_AUX_PCI_EVENT_REQ_HARD_RESET,
  MY_AUX_PCI_EVENT_SET_Q_RL,
};
```

```
struct my_aux_event_set_q_rl {
    u32 resource_id;
    u32 q_idx;
    u64 bw_min;
    u64 bw_max;
    u64 burst;
    u32 priority;
    u32 weight;
};
```

Return Result back to User Space



OPEN QUESTIONS



- Keep handler in container structs or move into auxiliary_device and auxiliary_driver?
- Discoverability – Anything beyond shared header needed? Capabilities?