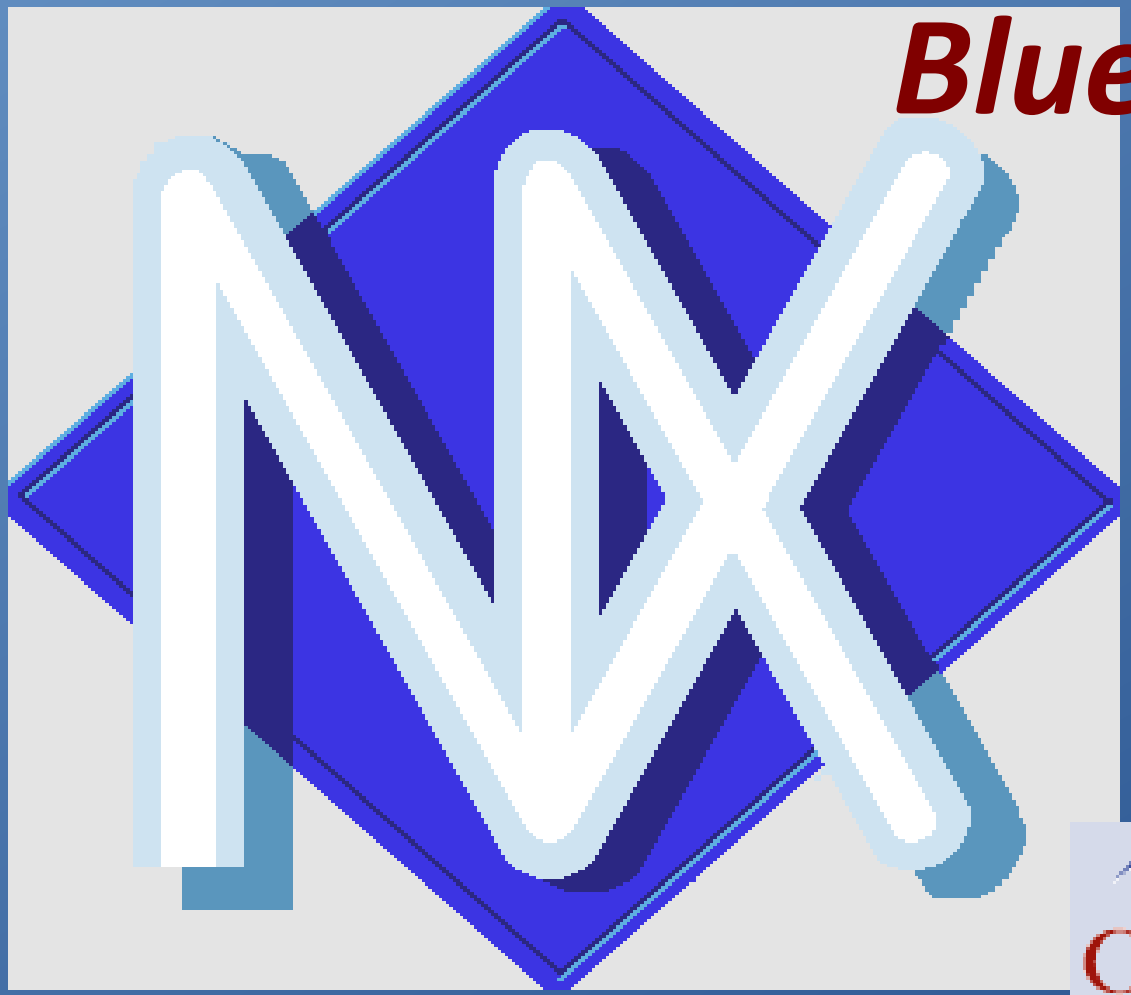
NuttX RTOS

Wifi/ Bluetooth



Gregory Nutt

NuttX Wireless Subsystem

Objective: Support ALL radios via Network Layer using standard, BSD socket interfaces (like Linux).

Simple, Low Cost Radios

- Use a simple character interface, *not part the wireless subsystem*
- cc1101,NRF2410L, LoRA (SX127X)

IEEE 802.15.4

- *Extensive* development; widespread usage; world class support
- Independent MAC and radio layers
- Supported by IPv6 / 6LoWPAN, AF_IEEE802 raw sockets, and character driver backdoor
- MRF24J40, XBee

Other Packet Radios

- Higher end, non-standard radios capable of networking
- Supported by IPv6 / 6LoWPAN, AF_PKTRADIO raw sockets
- STMicro Spirit radios

NuttX Wireless Subsystem (Continued)

WiFi (IEEE 802.11)

- *Not a highly developed sub-system*
- Controlled via Linux compatible IOCTLs
- Simple to port Linux WiFi utilities such as WAPI
- Standard BSD INET socket interface, including raw sockets.
- IEEE 802.11 stack currently part of radio driver
- BCM43362, BCM43428 with SDIO interface, FullMAC
- No SoftMAC (Partial port of NetBSD soft MAC available)

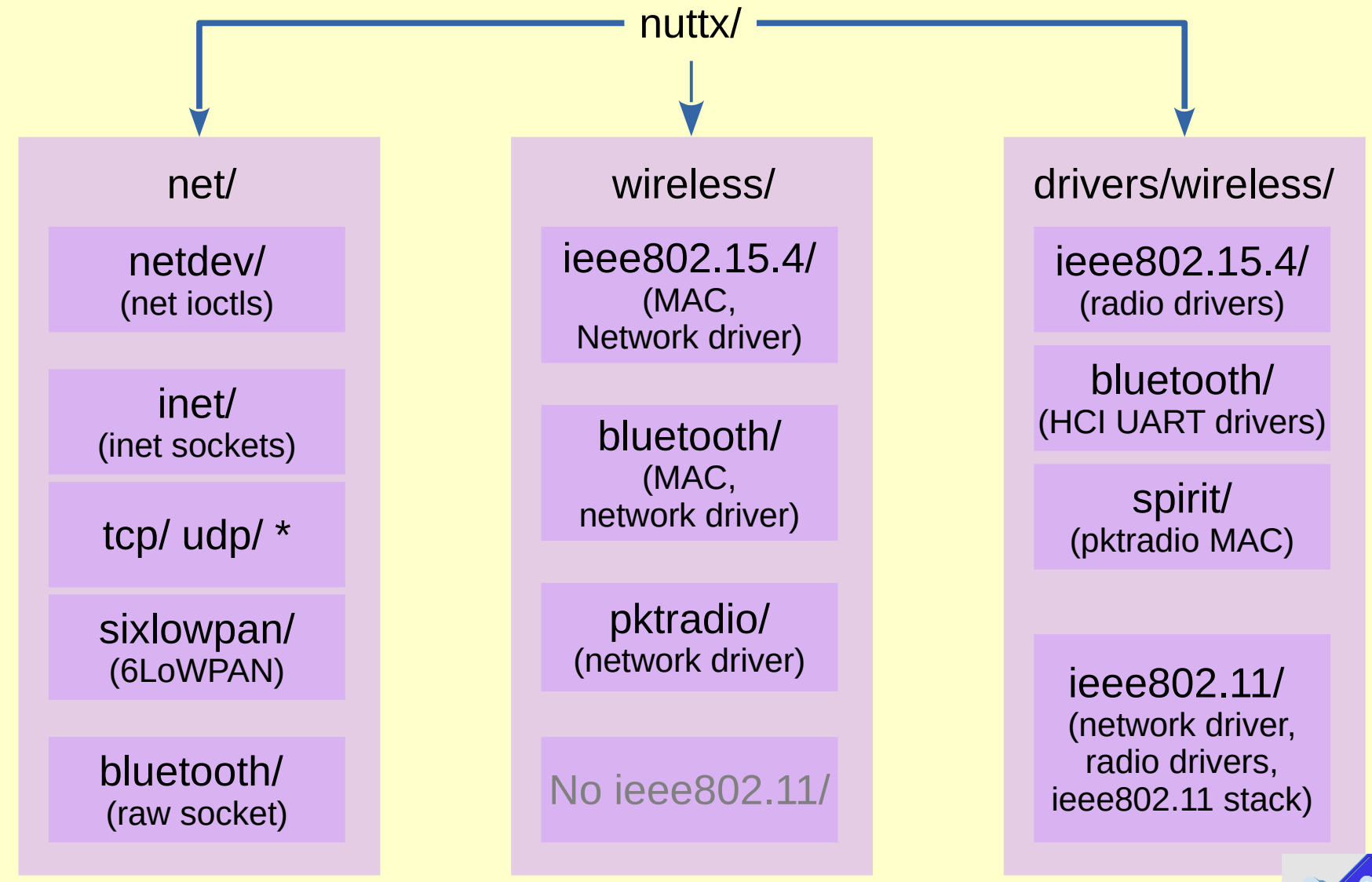
Bluetooth

- BLE 4.0 Stack leveraged from Intel BSD Zephyr release
- Current (Apache) Zephyr is Bluetooth/BLE 5.0
- AF_BLUETOOTH raw socket interface.
- Could support 6LoWPAN on LDAP
- HCI UARTs: TI CC2564, Laird BT860 (Cypress CYW20704) , BCM4348A1, BCM4343xm generic HCI-UART

WiFi Modules

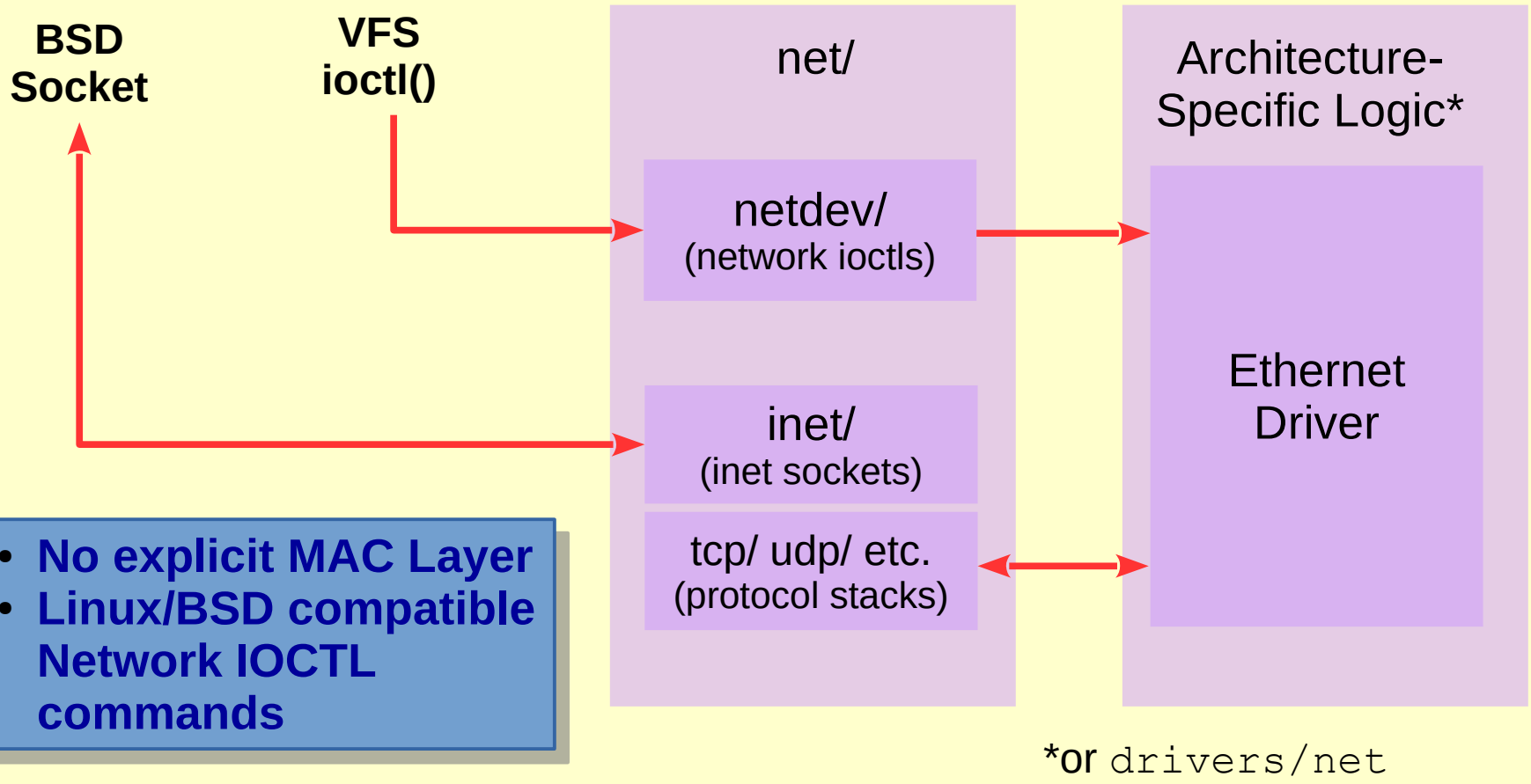
- With TCP/UDP stack on-module
- Support via *USRSOCK*, user space sockets
- GS2200M

Wireless Networking Directory Structure



*Not shown: ARP, ICMPv6, other protocols

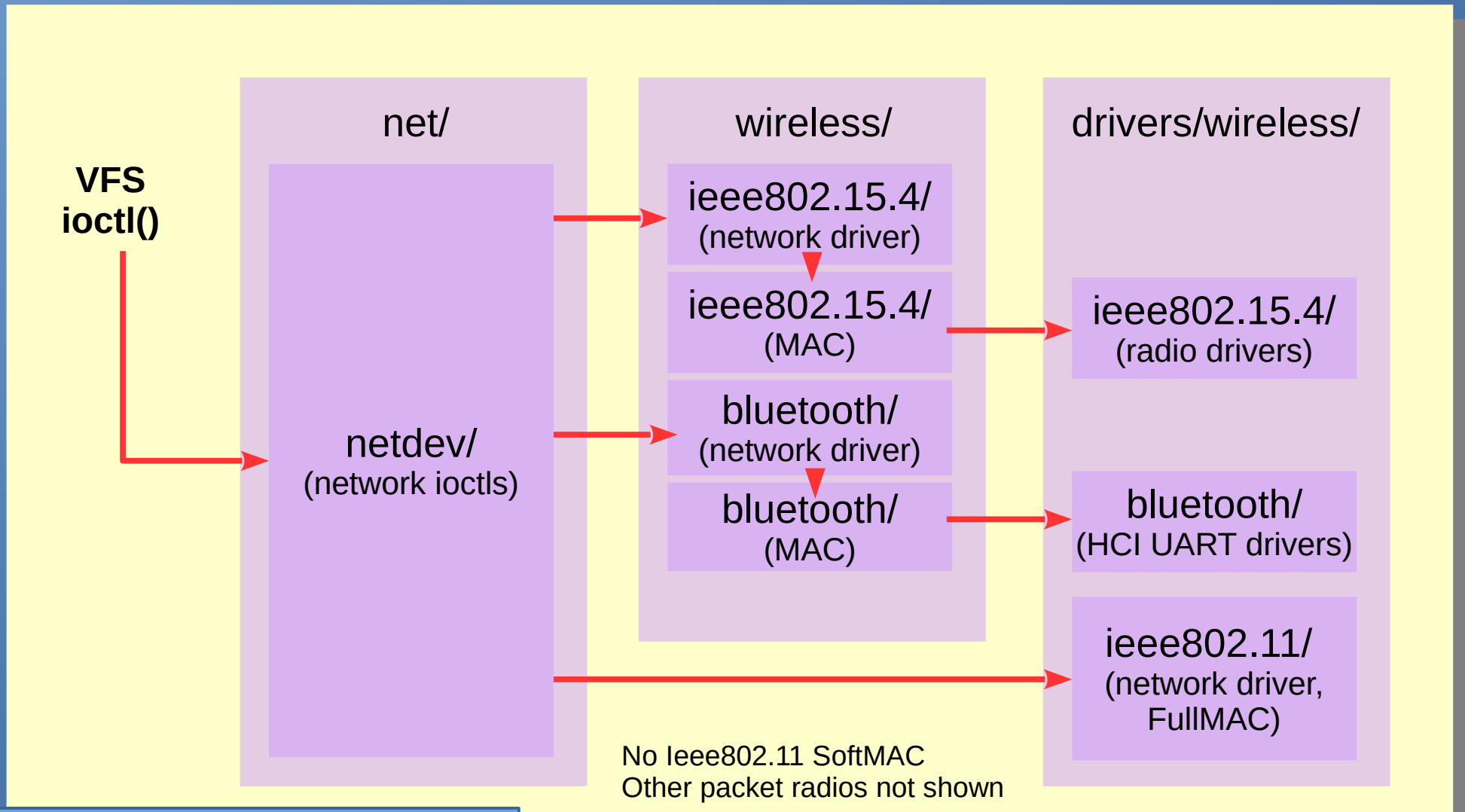
Simplest Data Flow Case: Wired Ethernet



- **No explicit MAC Layer**
- **Linux/BSD compatible Network IOCTL commands**

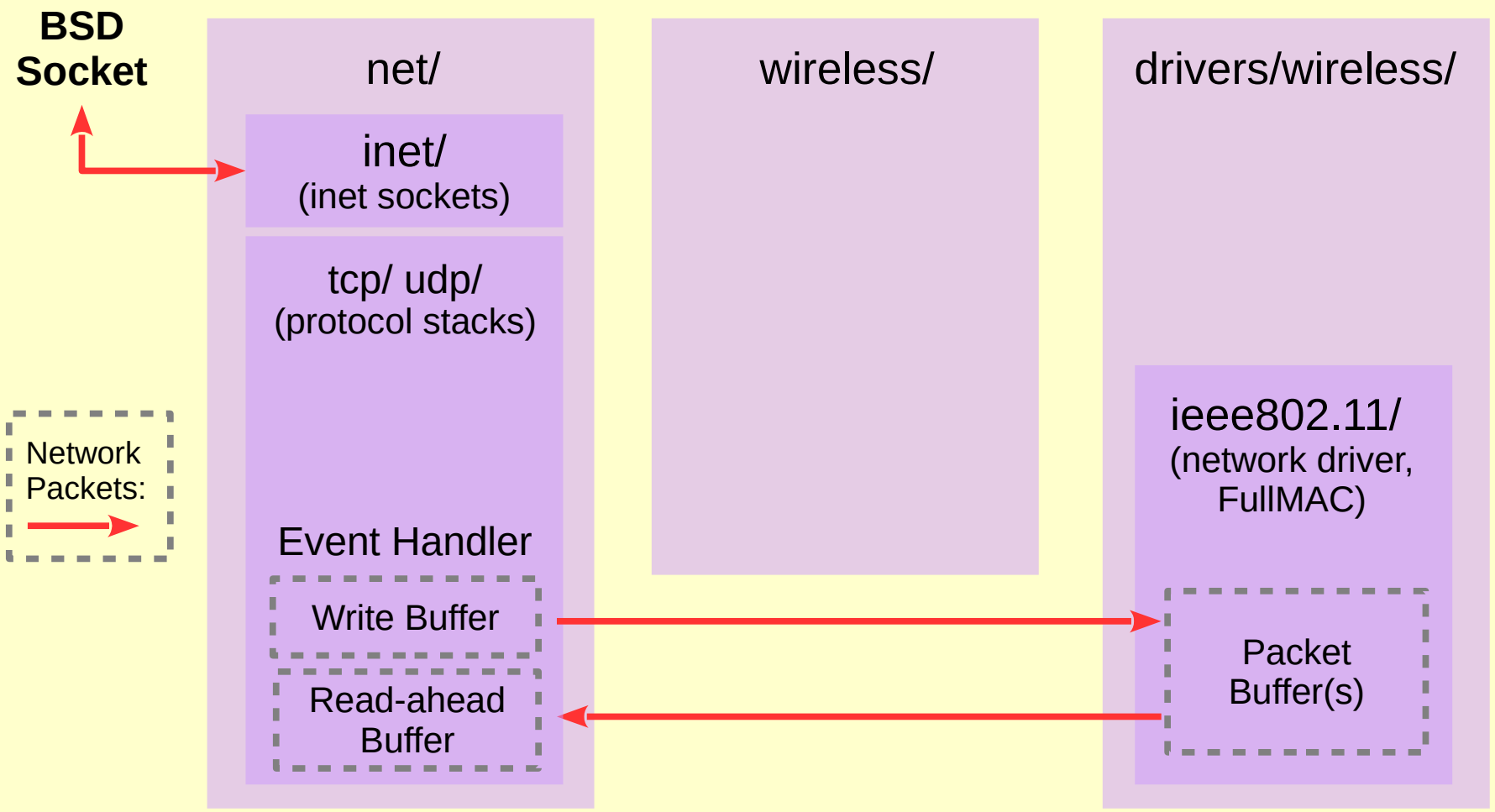
Network driver interface provides common Inteface between stacks and Layer2

IEEE802.15.4/WiFI/Bluetooth Network IOCTLs



Semi-standard, Linux/BSD compatible Wireless IOCTL commands

WiFi Network Packet Transfers

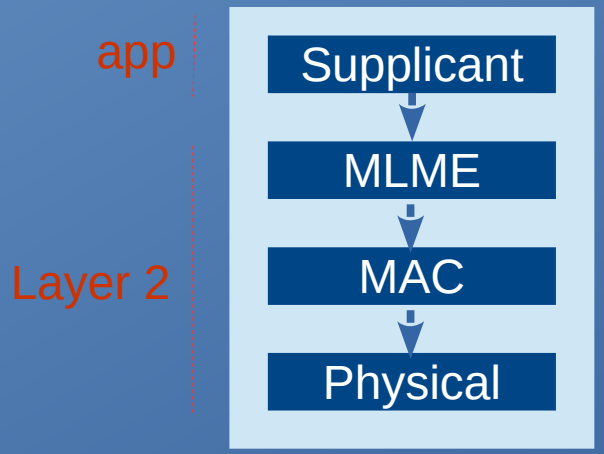


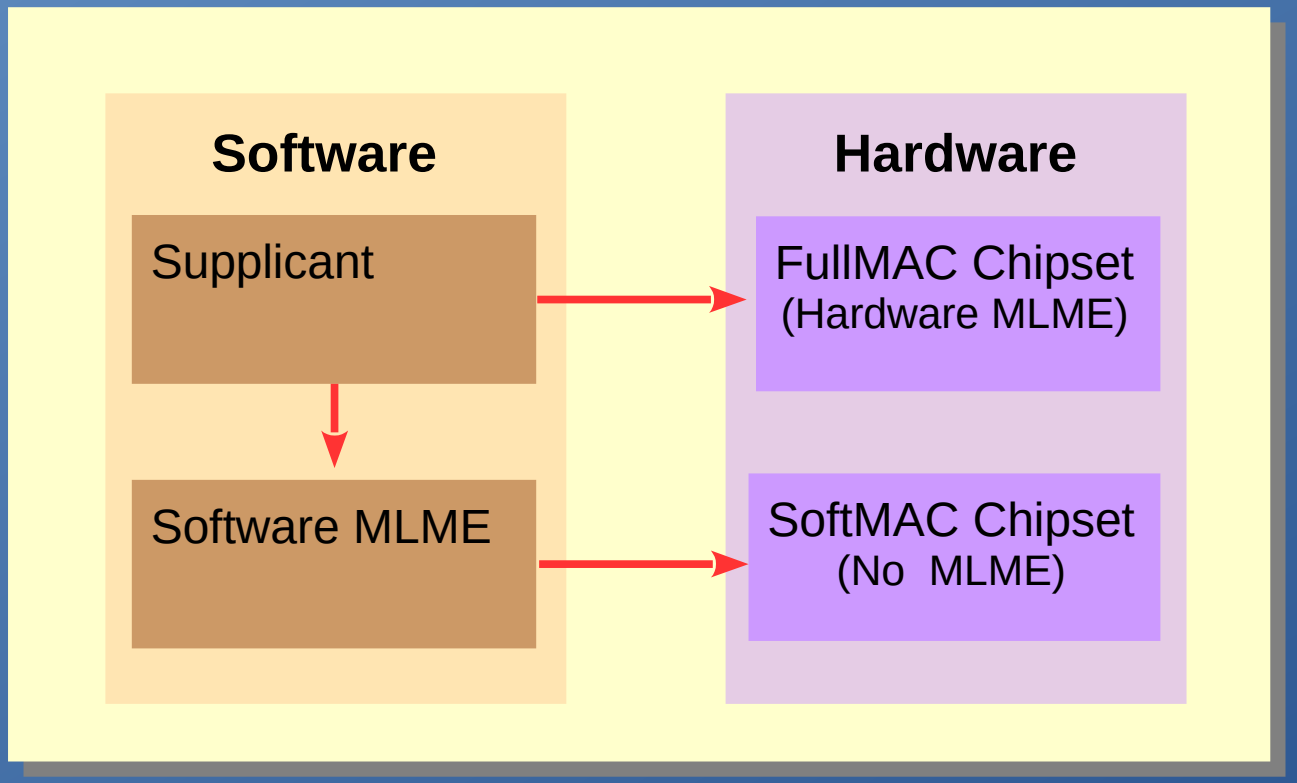
Not shown: ARP, ICMPv6, other protocols

WiFi SoftMAC vs FullMAC Chipsets

MAC
- Media Access Control
- Part of OSI Layer 2

MLME
- MAC Sublayer Management Entity
- Also Layer 2



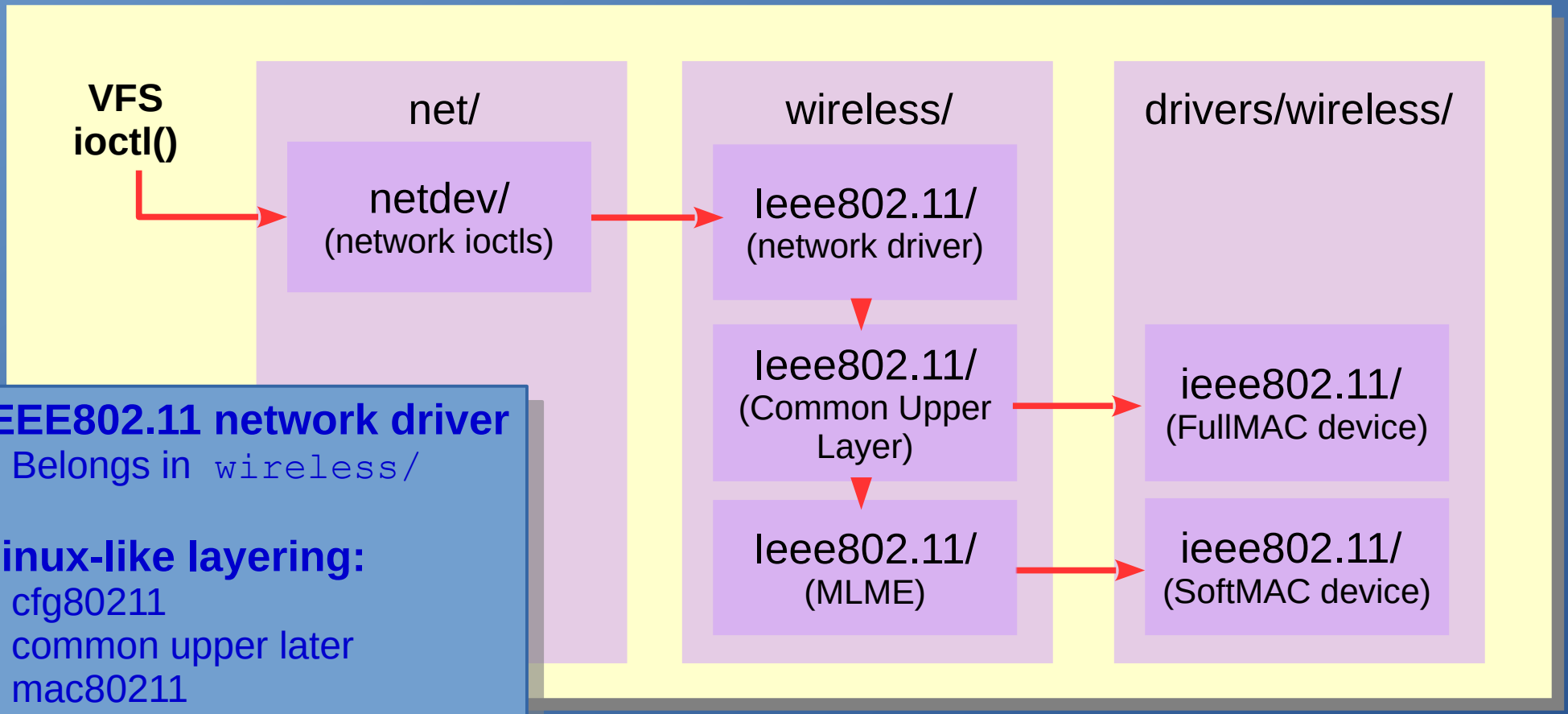


MLME is the management entity where the MAC state machines reside.

IEEE 802.11 SoftMAC / FullMAC Architecutre

(Future Development)

Supplicant



IEEE802.11 network driver

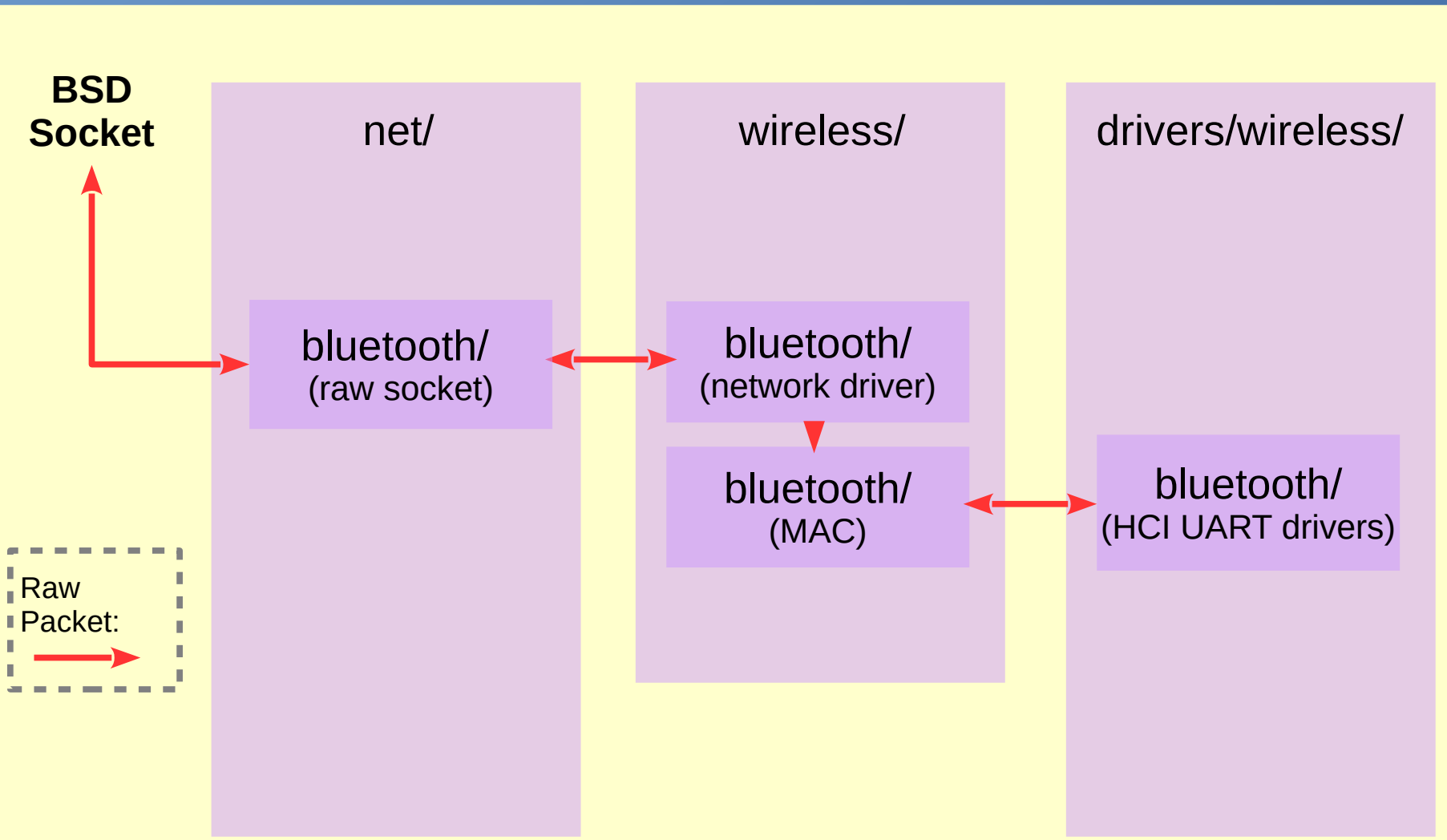
- Belongs in `wireless/`

Linux-like layering:

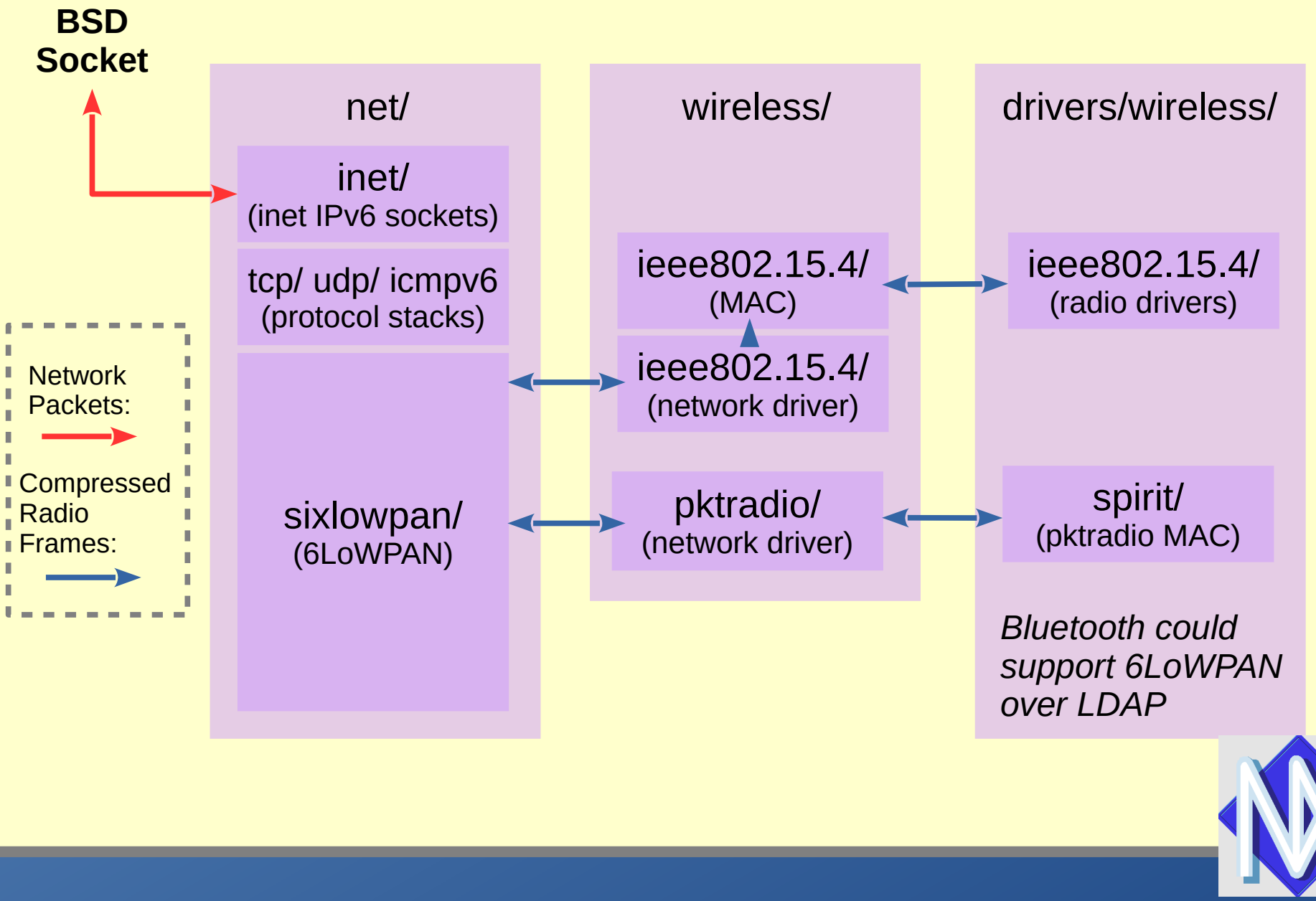
- cfg80211
  common upper later
- mac80211
  SoftMAC MLME

Partial port of NetBSD SoftMAC:
https://github.com/nuttx/nuttx_ieee80211

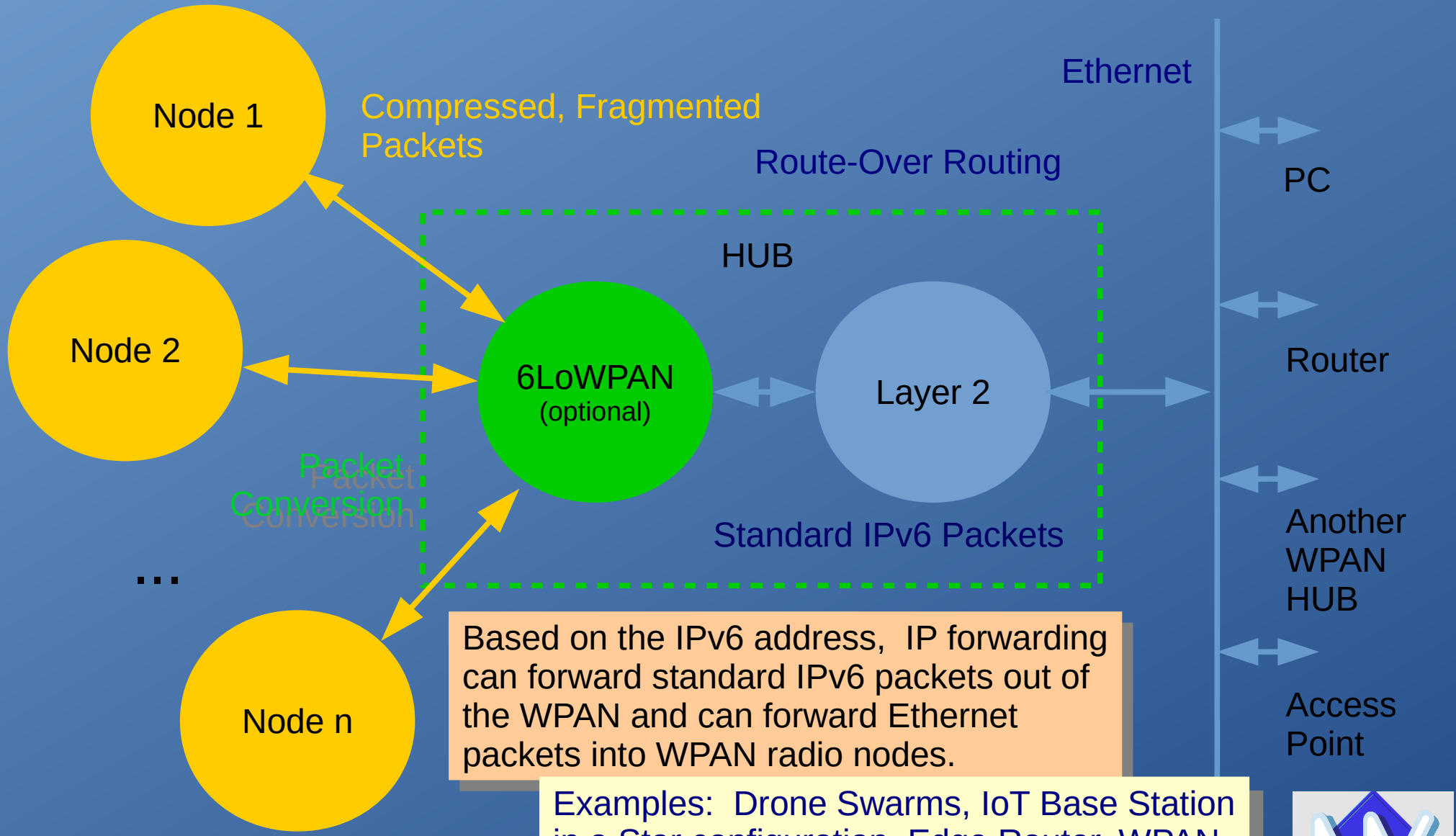
Bluetooth Transfers



IEEE802.15.4/PktRadio 6LoWPAN Transfers



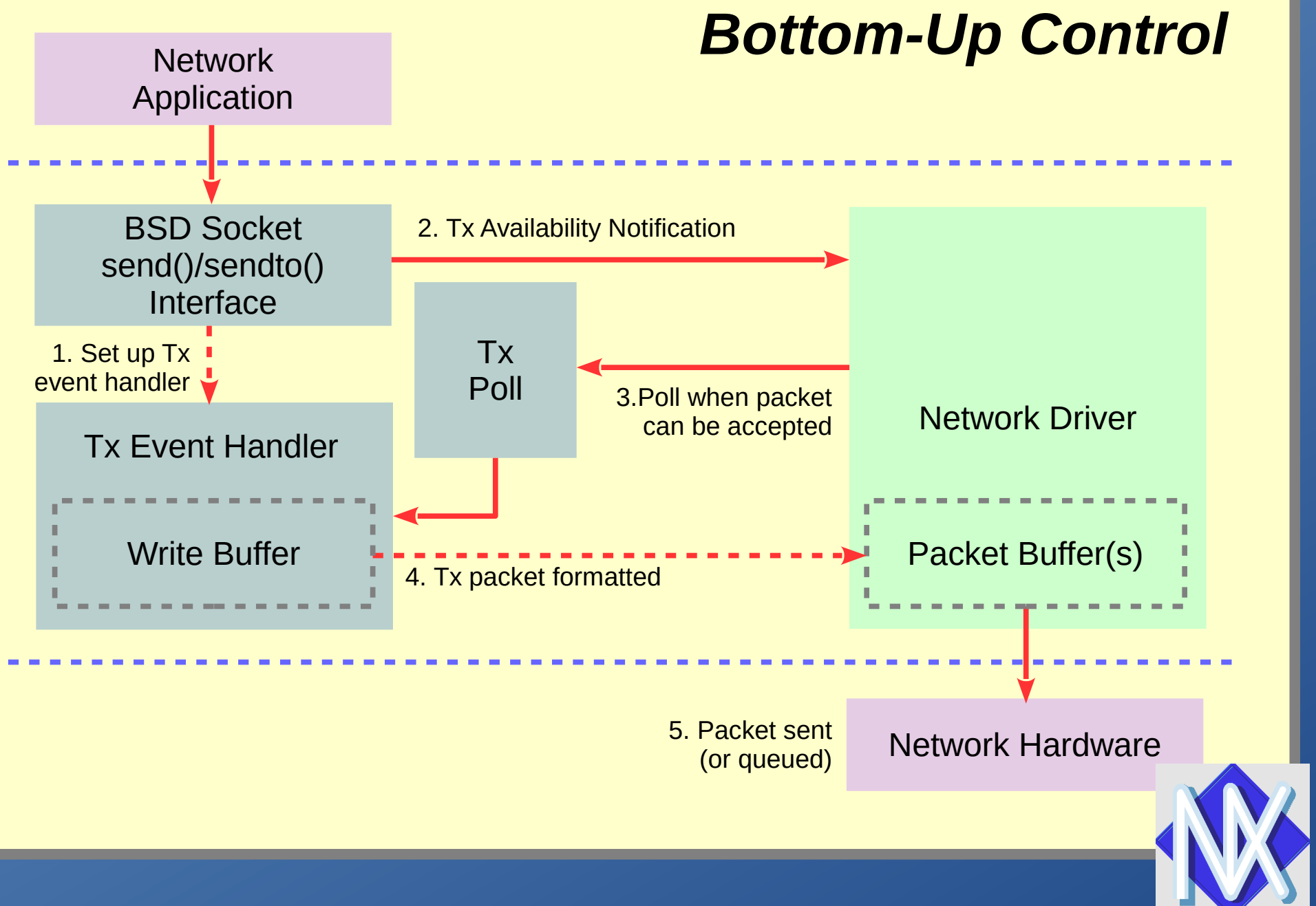
IEEE802.15.4/6LoWPAN WPAN Gateway
IP Forwarding



Based on the IPv6 address, IP forwarding can forward standard IPv6 packets out of the WPAN and can forward Ethernet packets into WPAN radio nodes.

Examples: Drone Swarms, IoT Base Station in a Star configuration, Edge Router, WPAN debug

Tx Event Handler “Rendezvous”



Network Driver Interface

- **`struct net_driver_s` defined in `include/nuttx/net/netdev.h`**
- **Provided by network driver all each call into the network stack**

“Bottom-Up Control”

- Network driver controls many events. *uIP* heritage.
- *The NuttX stack is a*n original work.
- It is ***not*** *uIP* but uses the TCP state machine and checksum
- algorithms from *uIP*

Network Driver Interface

Driver calls into Network

Rx Packet Input

- IPv4, IPv6 packet input
- 6LowPan frame input
- Raw packet tap

Tx Polling

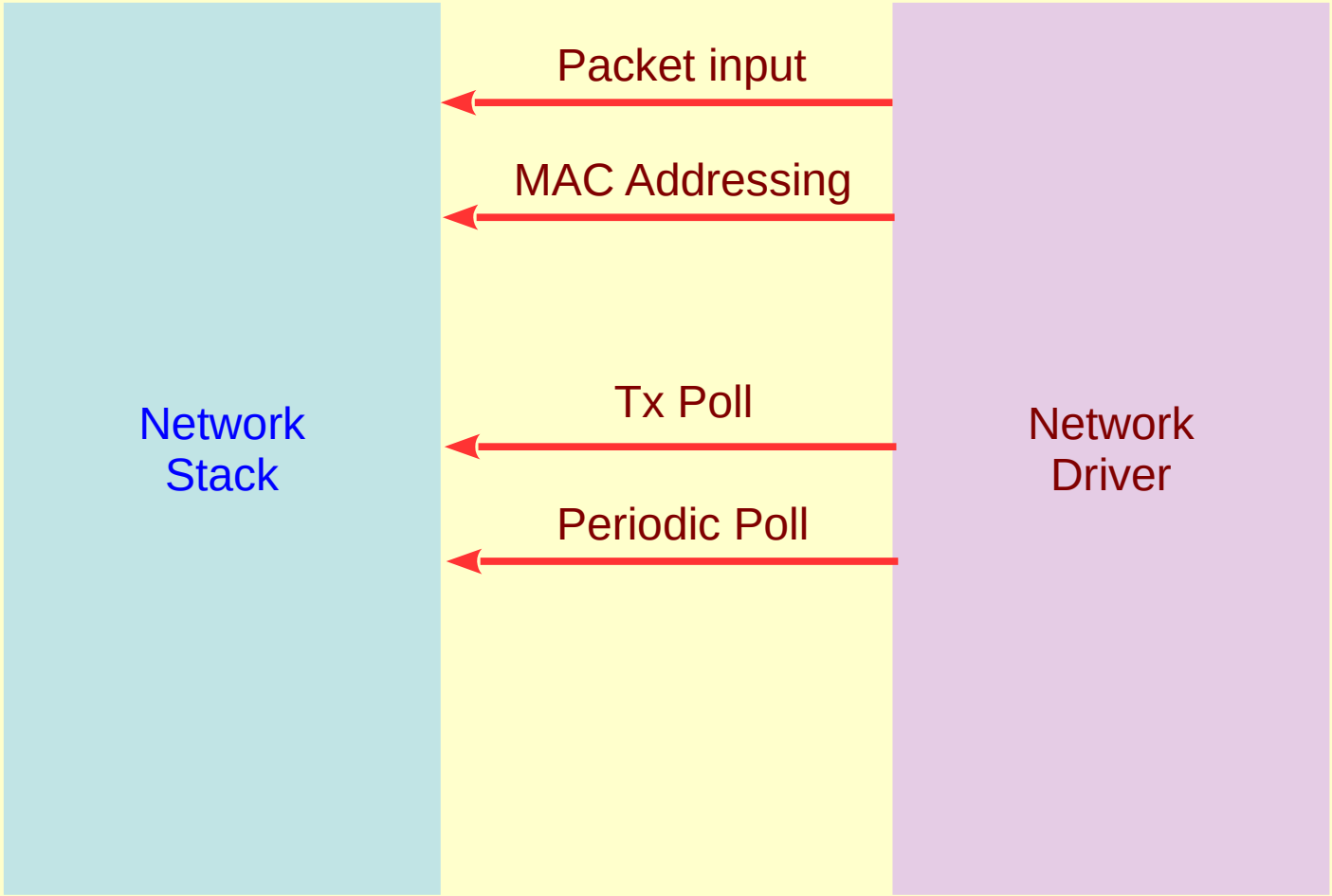
- Poll for Tx Packets
- Periodic Poll (TCP Protocol, socket option time outs, TCP Keepalive, etc.)

MAC Addressing

- ARP/IPv6 Neighbor Table Access
- Other protocols
- Needed for destination MAC address in outgoing Tx packets

Network Driver Interface

Driver calls into Network



struct net_driver_s instance passed with each call

Network Driver Interface

`struct net_driver_s` Content (simplified)

Network layer callbacks into Driver* device index

Data Link Layer 2 Information
- Data link layer protocol, header length
- MAC address

Network Layer 3 Information
- IP Addresses, router IP address, subnet mask
- Multicast group information

Packet Buffer
- Packet buffer, packet size, maximum packet size

Statistics

Network layer callbacks into Driver*

Network Driver Interface

Network Callbacks into Driver

***Network layer callbacks into Driver**

- Interface Up/Down
- Tx data available notification
- Address filtering
- Forwarded IOCTL commands*

Addition 6LoWPAN layer callbacks into Radio Drivers

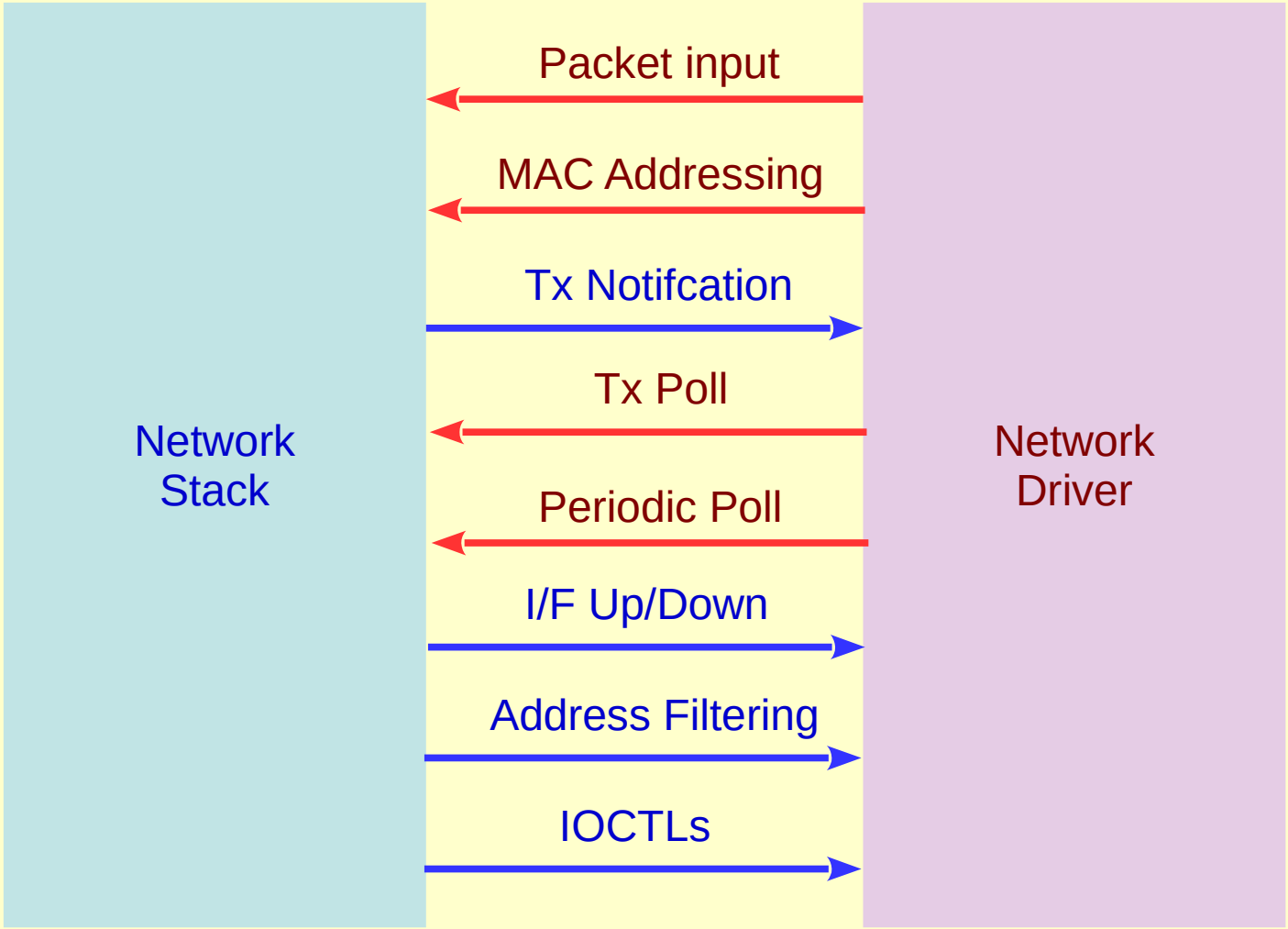
- Radio Frame MAC header length
- Outgoing Radio Frames
- Radio Properties

*** IOCTL Commands**

- May be handled at different levels
- Socket level IOCTL commands handled in net/ logic
- Unhandled IOCTL command forwarded to Network Driver
- Unhandled IOCTL command forwarded to MAC layer
- Unhandled IOCTL command forwarded to radio driver
- Unhandled IOCTL commend generates error code

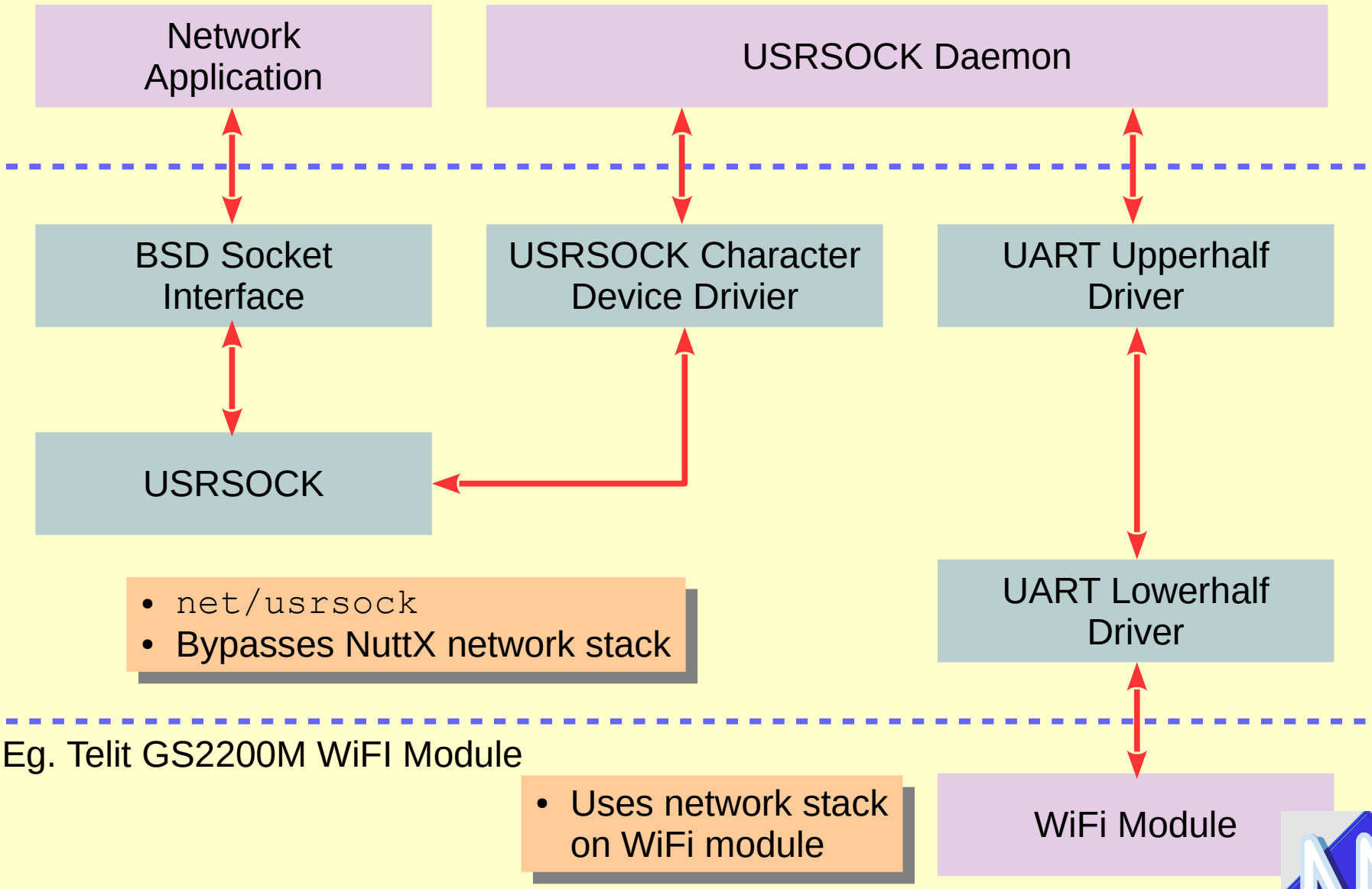
Network Driver Interface

Network calls into Driver



Plus 6LoWPAN Radio Driver Calls

WiFi Modules / Userspace Sockets



WiFi/Bluetooth Tools

WiFi Tools: Interface via Linux-compatible Network IOCTLs

WAPI

- Used to manage 802.11 Network
- Subset: *help, show, scan, ip, mask, freq, essid, mode, ap, bitrate, txpower*

Bluetooth Tools:

- Interface via Network IOCTLs
- Most derive from NetBSD

btsak

- Bluetooth Swiss Army Knife
- Top level commands: *help, info, features, scan, advertise, security, gatt*
- Gatt commands: *exchange-mtu, mget, discover, characteristic, descriptor, dget, read, read-multiple, rget, write, wget*

IEEE 802.15.4 Tools

Interface via

- BSD Socket IOCTLs or
- IEEE802.15.4 Backdoor Serial driver

IWPAN

- Similar to WAPI
- Inspired by iwpan on Linux
- Use to manage IEEE 802.15.4 PAN
- Radio settings: *cca, chan, devmode, eaddr, panid, promisc, saddr, txpwr*
- MAC commands: *assoc, disassoc, get, gets, poll, rxenab, scan, set, start, sync*

I8sak

IEEE802.15.4 Swiss Army Knife
Commands: *help, acceptassoc, assoc, blaster, get, poll, regdump, reset, scan, set, sniffer, startpan, tx*

I8shark

IEEE 802.15.4 Wireshark Adaptor
Packet capture and analysis

Future IoT Components

Kernel- vs. Application-Space Components

- No new, non-standard OS interfaces; Must use existing POSIX interfaces: BSD socket interface is the preferred interface

New Network Stack Components

- Most elements may require extensions to networking, such as:
- New socket address families under `net/`
- New MAC implementations under `wireless/`

New IoT Applications

Integrated IoT Applications and Network

- Example, mesh routing requires integration of application-level routing and network stack routing information for discovery and mesh maintenance
- Recommended Solution: *Netlink sockets.*

Future IoT Components – Netlink Sockets

Netlink Sockets

- Superficially compatible with Linux Netlink sockets
- Provide application access to internal network services
- For example: Access to internal routing and node discover for IoT meshes
- Status: Socket layer fully implemented/verified. Currently no network services.