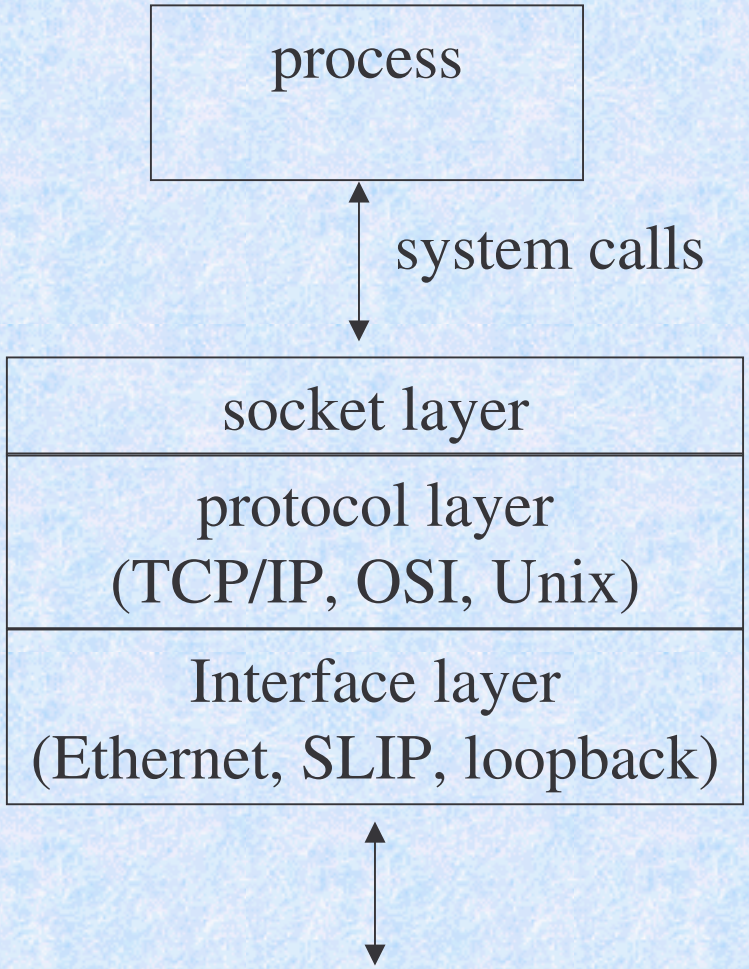
Kernel Code Directory Structure

Under /usr/src/sys

- **dev/fxp – Intel fxp network card driver**
- i386 – intel specific
- kern – general kernel stuff that does belong else where
- **net – general networking stuff**
- netiso – OSI protocols
- **netinet – TCP/IP protocols**
- pci – general pci support functions
- sys – header files
- nfs – NFS

Network Code Organization



7: application
6: presentation
5: session

4: transport
3: network

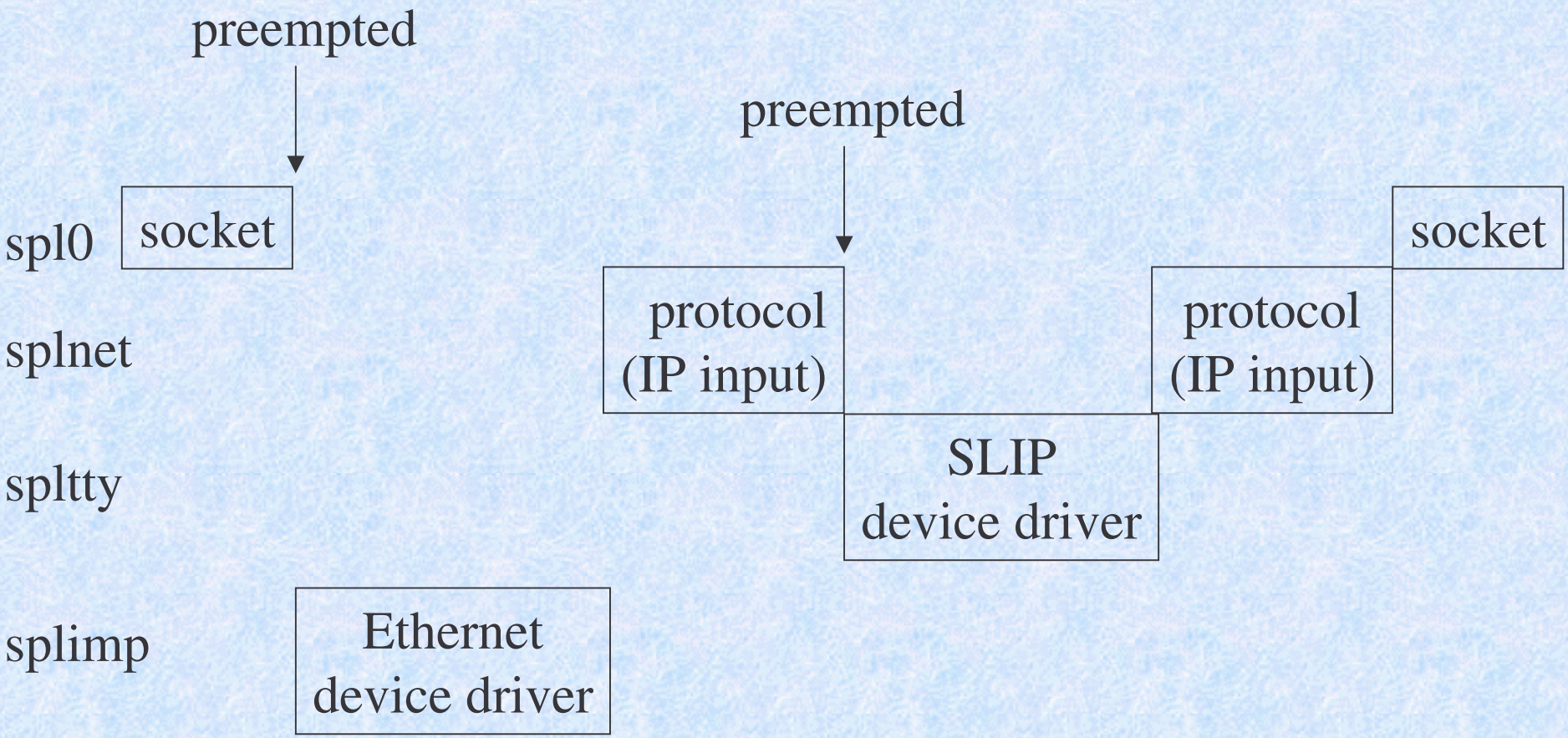
2: data link

1: physical

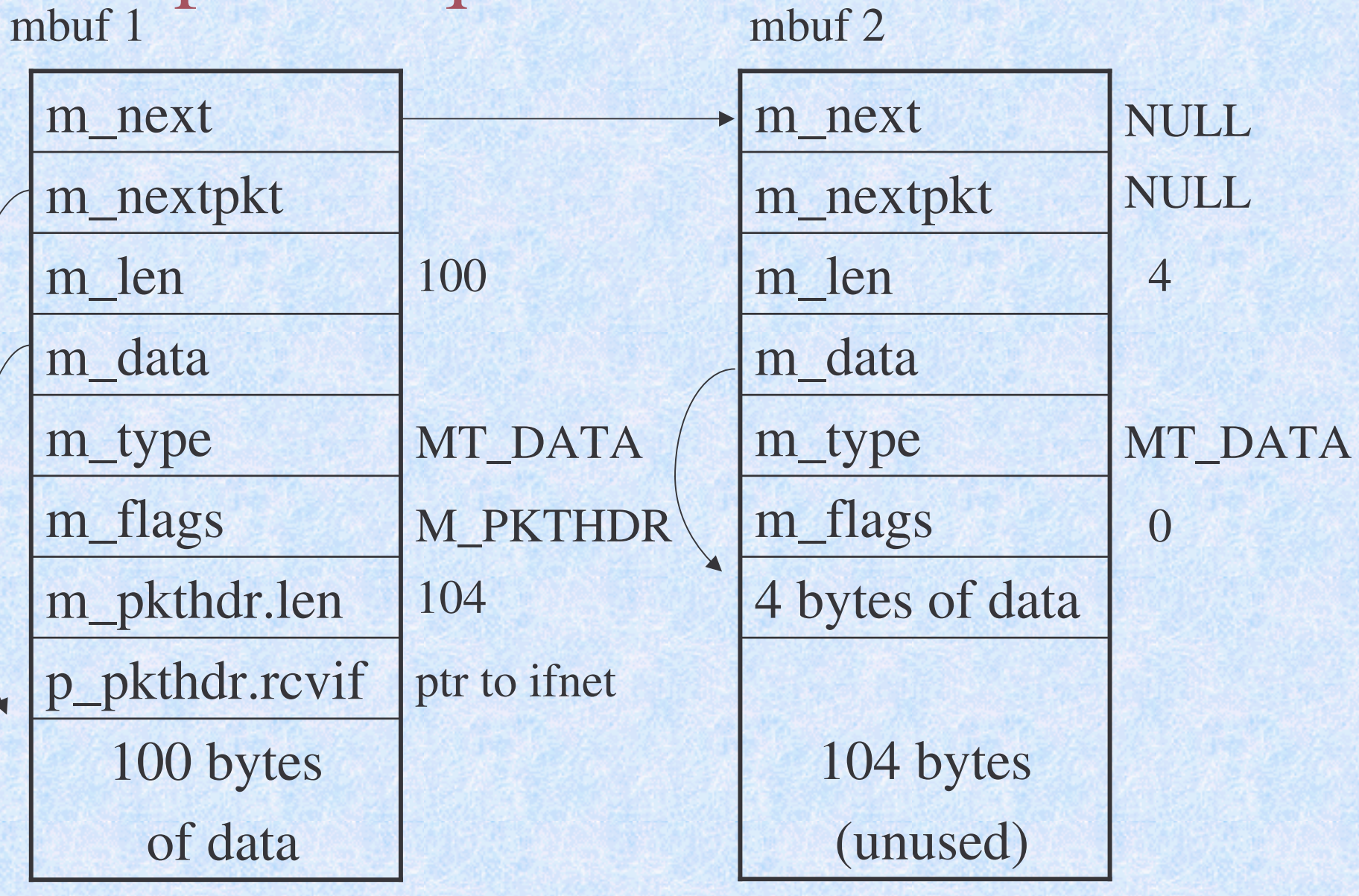
Set Priority Level Commands

spl0	normal operating mode, nothing blocked (lowest priority)
splsoftclock	low-priority clock processing
splnet	network protocol processing
spltty	terminal I/O
splbio	disk and tape I/O
splimp	network device I/O
splclock	high priortiy clock processing
splhigh	all interrupts blocked (highest priority)
splx(s)	return to level s

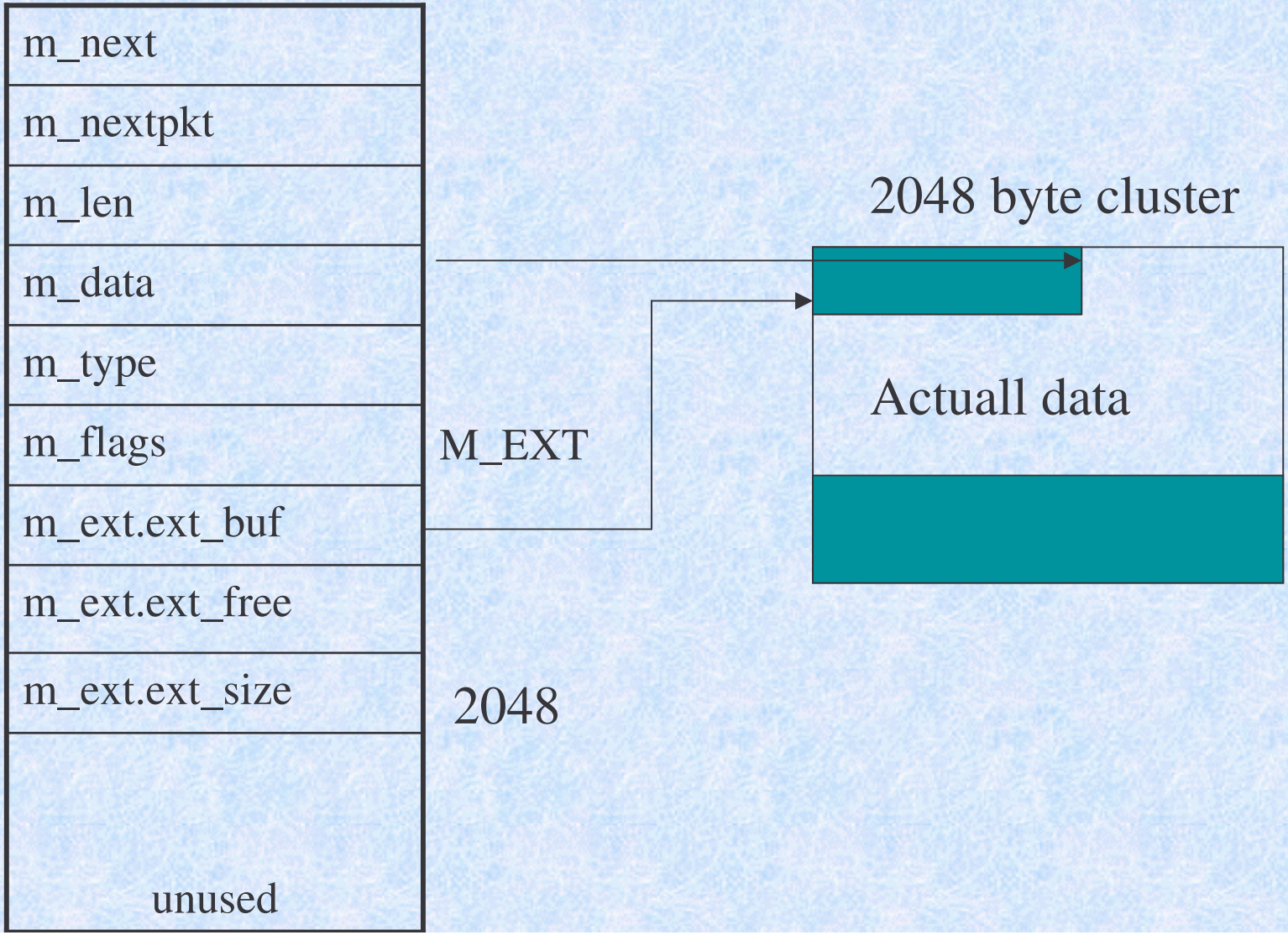
SPL Examples



packet queue data structure



External mbuf structure



Mbuf functions

m_adj(*m, len)	remove len bytes from start of m if len>0 or abs(len) bytes from end of m if len<0
m_cat(*m, n)	concatenate mbuf n to mbuf m
mbuf *m_copy(*m, offset, len)	copy len bytes from offset in mbuf m and return new mbuf
m_free, m_freem(*m)	free one mbuf or mbuf chain
mbuf *m_get(wait, type)	get new mbuf

by *m I mean struct mbuf *

Mbuf functions cont'

m_pullup(*m, len)	arrange data contiguously in memory
M_PREPEND(*m, len, wait)	prepend len bytes to start of m
mtod(*m, type)	cast the mbuf * to new type

ifnet structure

- Each interface has a struct ifnet holding its data.
- All struct ifnets are connected in a linked list.
- ifnet functions:
  if_init – initialize the interface
  if_output – queue outgoing packets for transmission.
  if_start – initiate transmission of packets.
  if_ioctl – the interface ioctl function
- Also hold interface name, flags, statistics, etc…

Incoming Ethernet Packet Handling

- Driver interrupt routine (fxp_intr) getting the packet from the hardware into an mbuf chain. calling ethernet driver
- Ethernet driver validate the packet. Look at layer 3 type, put packet on appropriate queue, and initiate appropriate software interrupt
- Layer 3 interrupt routine get packet from queue. Do the layer 3 job.

Incoming Packet – Network Card Driver

- Uses DMA to get incoming packet data into kernel memory.
- fxp_intr:
  Uses preallocated mbuf pool to build an mbuf chain holding the packet.
  Doing checksum validation.
  Calling the input function of the right interface.

src/sys/dev/fxp/if_fxp.c

Incoming Packet - Ethernet

```
void
ether_input(struct ifnet *ifp, struct ether_header *eh, struct mbuf *m)
        m->m_pkthdr.rcvif = ifp;
        eh = mtod(m, struct ether_header *);
        m->m_data += sizeof(struct ether_header);
        m->m_len -= sizeof(struct ether_header);
        m->m_pkthdr.len = m->m_len;

       ether_demux(ifp, eh, m);
```

src/sys/net/if_ethersubr.c

Incoming Packet – Ethernet (cont')

```
void ether_demux(ifp, eh, m)
        ether_type = ntohs(eh>ether_type);
        switch (ether_type) {
        case ETHERTYPE_IP:
                schednetisr(NETISR_IP);
                inq = &ipintrq;
                break;
          .
          .
          .
        (void) IF_HANDOFF(inq, m, NULL);
```

Incoming Packet - IP

ipintr – the IP interrupt routine

```
while(1) {
        s = splimp();
        IF_DEQUEUE(&ipintrq, m);
        splx(s);
        if (m == 0)
            return;
        ip_input(m);
}
```

src/sys/netinet/ip_input.c

Incoming Packet – IP (cont')

```
void ip_input(struct mbuf *m)

    IPFIREWALL and DUMMYNET hooks

    if (m->m_len < sizeof (struct ip) &&
            (m = m_pullup(m, sizeof (struct ip))) == 0) {

                ipstat.ips_toosmall++;
                return;
     }

    ip = mtod(m, struct ip *);

    validation checks on IP header
```

Incoming Packet – IP (cont’)

IP routing handling

IP fragments reassembly

m_adj to remove the IP header

inetsw[ip_protox[ip->ip_p]].pr_input)(m, off, nh);*)

Outgoing packet - UDP

```
static int udp_output(struct inpcb *inp, struct mbuf *m
    , sturct sockaddr *addr struct mbuf *control, struct thread *td) {

      M_PREPEND(m, sizeof(struct udpiphdr) + max_linkhdr, M_DONTWAIT);

      ui = mtod(m, struct udpip hdr *);

      Filling UDP header fields, pseudo IP header fields

      error = ip_output(m, inp->inp_options, NULL, ipflags,
                                    inp->inp_m_options, inp)

  }
```

src/sys/netinet/udp_usrreq.c

Outgoing packet - IP

```
int ip_output(struct mbuf *m, struct mbuf *opt, struct route *ro,
                    int flags, struct ip_moptions *imo, struct inpcb *inp) {

    ip = mtod(m, struct ip *);

    Multicast handling

    Routing handling – finding out which interface we go out from

    Cut to pieces smaller than the interfaces's MTU

    For (; m; m = m0) {

        m0 = m->m_nextpkt

        m->m_nextpkt = 0;

        error = *ifp->if_output(ifp, m, dst, ro->ro_rt);
    }

}
```

src/sys/netinet/ip_output.c

Outgoing packet - Ethernet

```
int ether_output(struct ifnet *ifp, struct mbuf *m,
                 struct sockaddr *dst, struct rtentry *rt0) {

    switch (dst->sa_family) {

    case AF_INET:

        error = arpresolve(ifp, rt0, m, dst, edst);

      …
    }
    M_PREPEND(m, ETHER_HDR_LEN, M_DONTWAIT);

    eh = mtod(m, struct ether_header *);

    IFQ_HANDOFF(ifp, m, error);

}
```

src/sys/net/if_ethersubr.c

```
#define IFQ_HANDOFF(ifp, m, err)
do {
    int len;
    short mflags;
    len = (m)->m_pkthdr.len;
    mflags = (m)->m_flags;
    IFQ_ENQUEUE(&(ifp)->if_snd, m, err);
    if ((err) == 0) {
        (ifp)->if_obytes += len;
        if (mflags & M_MCAST)
        (ifp)->if_omcasts++;
        if (((ifp)->if_flags & IFF_OACTIVE) == 0)
            if_start(ifp);
    }
} while (0)
```

src/sys/net/if_var.h

Outgoing Packet – Network Card Driver

- fxp_start found in src/sys/dev/fxp/if_fxp.c
- While there are packets in the send queue do:

  get a packet off the queue
  hand it to the hardware to transmit

References

- The Design and Implementation of the 4.4 BSD Operating System
  McKusick, Bostic, Karels, Quarterman
- TCP/IP Illustrated Volume 2
  Stevens and Wright
- FreeBSD source code