Linux Networking Overview

主讲：孟宁
电话：0512-68839302
E-mail: mengning@ustc.edu.cn
主页: http://staff.ustc.edu.cn/~mengning
地址：苏州工业园区独墅湖高等教育区仁爱路166号明德楼A302室

2009年12月
Structure of the Linux kernel
IEEE 802 in the Linux network architecture
LLC protocol control information

<table>
<thead>
<tr>
<th>LLC variant</th>
<th>MAC and LLC frame formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>802.3:</td>
<td>Preamble</td>
</tr>
<tr>
<td>802.2:</td>
<td>Preamble</td>
</tr>
<tr>
<td>802.2/SNAP:</td>
<td>Preamble</td>
</tr>
</tbody>
</table>

- 802.2/SNAP: 0xAA 0xAA Ctl OUI PID
Structure of socket buffers
struct sk_buff
{
    struct sk_buff *next, *prev;
    struct sk_buff_head *list;
    struct sock *sk;
    struct timeval *stamp;
    struct net_device *dev, *rx_dev;

    union /* Transport layer header */
    {
        struct tcphdr *th;
        struct udphdr *uh;
        struct icmphdr *icmph;
        struct igmphdr *igmph;
        struct iphdr   *ipiph;
        struct spxhdr  *spxh;
        unsigned char  *raw;
    } h;

    union /* Network layer header */
    {
        struct iphdr   *iph;
        struct ipv6hdr *ipv6h;
        struct arphdr  *arph;
        struct ipxhdr  *ipxh;
        unsigned char  *raw;
        __u32
        atomic_t
        priority;
    } nh;

    union /* Link layer header */
    {
        struct ethhdr *ethh;
        unsigned char *raw;
    } mac;

    struct dst_entry *dst;
    char            cb[48];
    unsigned int    len, csum;
}  

volatile char used;
unsigned char is_clone, cloned, pkt_type, ip_summed;
users;
protocol, security;
*head, *data, *tail, *end;
(*destructor)(struct sk_buff *);
Packet queues in the Linux kernel
struct sk_buff_head
{
    struct sk_buff *next;
    struct sk_buff *prev;
    __u32 qlen;
    spinlock_t lock;
};
the packet buffers across the protocol hierarchy
The structure of a network device interface

- **Network devices** (adapter-independent)
- **Network driver** (adapter-specific)
- Abstraction from adapter specifics
A network adapter uses an interrupt to send messages.
Linking net_device structures

- dev_base

- net_device
  - name: eth0
  - state
  - next
  - priv
  - Hardware
  - MAC layer
  - Network layer
    - open
    - stop
    - hard_start_xmit

- local2
  - private driver structure
    - adapter2_open()
    - adapter2_stop()
    - adapter2_start_xmit()

- net_device
  - name: eth1
  - state
  - next
  - priv
  - ...

- Hardware

- Network layer
  - open
  - stop
  - hard_start_xmit
  - ...

- local1
  - private driver structure
    - adapter_open()
    - adapter_stop()
    - adapter_start_xmit()
Activity forms in the Linux network architecture
谢谢大家！