mISDN continued

Karsten Keil
SuSE Labs
SUSE Linux Products GmbH

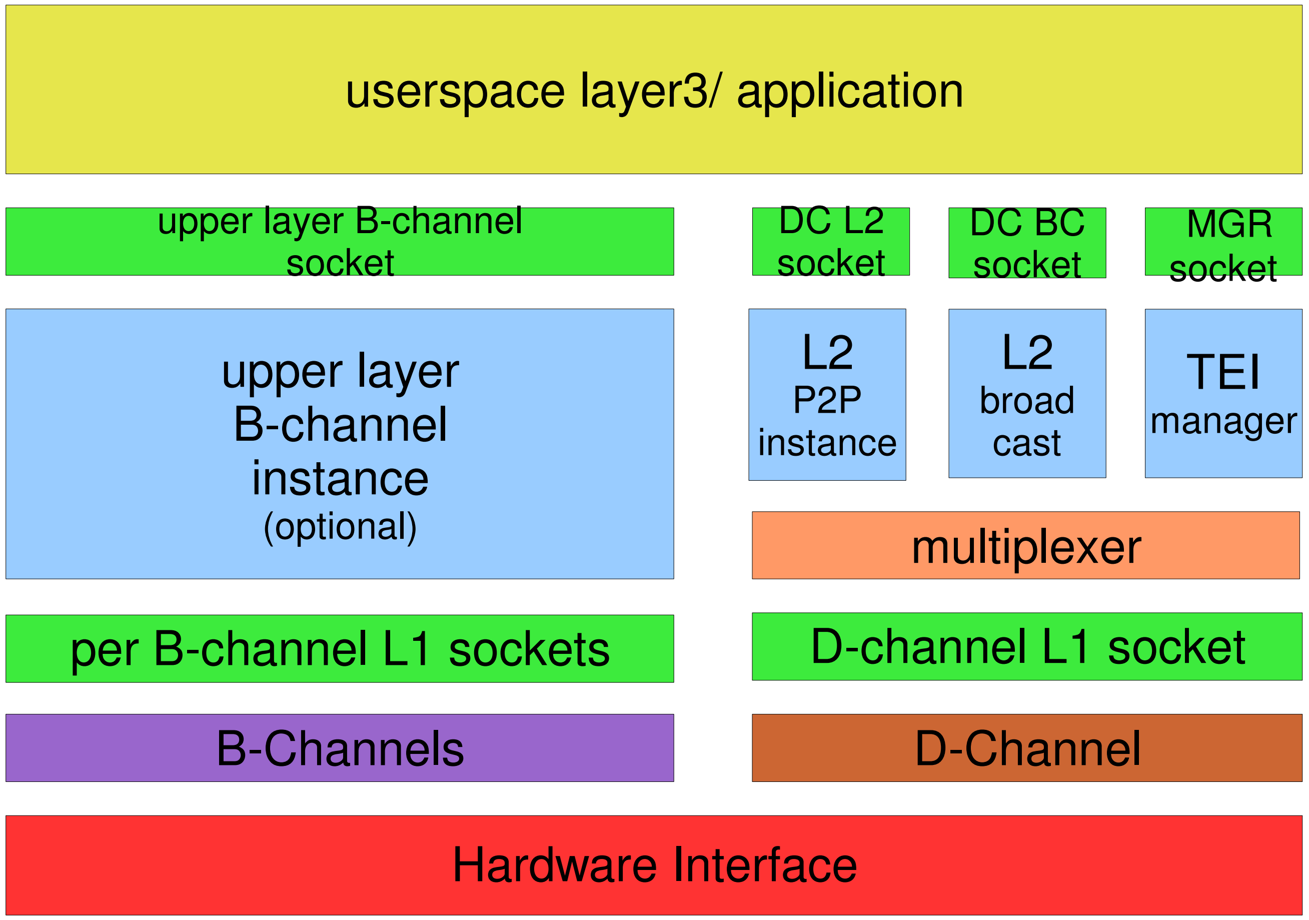
History

- mISDN was introduced on Linux Kongress 2004
- First version with major design flaws
- Redesign

Redesign

- Minimum protocol functions in kernel
- Use socket interface
- Control via standard socket operations and IOCTL
- Static D-channel stack
- Mode selection on runtime

mISDN new structure



Kernel/user space API

- Simple socket calls
- Different protocols address different levels/modules
- link to a device and channel via the bind address

```
struct sockaddr_mISDN {
    sa_family_t    family;
    unsigned char  dev;        /* device number */
    unsigned char  channel;   /* channel number 0 for D channel */
    unsigned char  sapi;       /* SAPI D-channel only */
    unsigned char  tei;        /* TEI D-channel only */
};
```

Kernel/user space API

- The D-channel can be accessed on Layer1 level (e.g. for logging or testing)
- But applications should use the Layer2 interface
- TEI management is included in Layer2

```
sock = socket(PF_ISDN, SOCK_DGRAM, ISDN_P_LAPD_TE);

l2addr.family = AF_ISDN;
l2addr.dev = 0;
l2addr.channel = 0;
l2addr.sapi = 0;
l2addr.tei = 127;

ret = bind(sock, (struct sockaddr *)&l2addr, sizeof(l2addr));

ret = sendto(sock, buf, len, 0, (struct sockaddr *)l2addr, sizeof(l2addr));

alen = sizeof(l2addr);
ret = recvfrom(sock, buf, blen, 0, (struct sockaddr *)&l2addr, &alen);
```

Kernel/user space API

- B-channels can stack additional modules between the card driver socket and the user space socket (e.g DSP functions)
- The function/module is selected via the protocol
- The channel is selected via the address

```
Bsock = socket(PF_ISDN, SOCK_DGRAM, ISDN_P_RAW);

l2addr.family = AF_ISDN;
l2addr.dev = 0;
l2addr.channel = 2;
l2addr.sapi = 0;
l2addr.tei = 0;

ret = bind(Bsock, (struct sockaddr *)&l2addr, sizeof(l2addr));

ret = sendto(Bsock, buf, len, 0, (struct sockaddr *)l2addr, sizeof(l2addr));

alen = sizeof(l2addr);
ret = recvfrom(Bsock, buf, blen, 0, (struct sockaddr *)&l2addr, &alen);
```

Message format

Primitive 32 bit	Identifier 32 bit	Payload data 0 to n bytes

- Primitive is the type of the message
- Identifier maybe used on types which need an answer to identify the origin message, it may be contain address informations as well
- Not all messages have payload data

Applications

- misdn_log
- misdn_bridge
- l1oipctrl
- Linux Call Router (lcr)

Wireshark demo

Linux Call Router live demo

Novell®