

# mISDN continued

---

Karsten Keil  
SuSE Labs  
SUSE Linux Products GmbH



**Novell.**

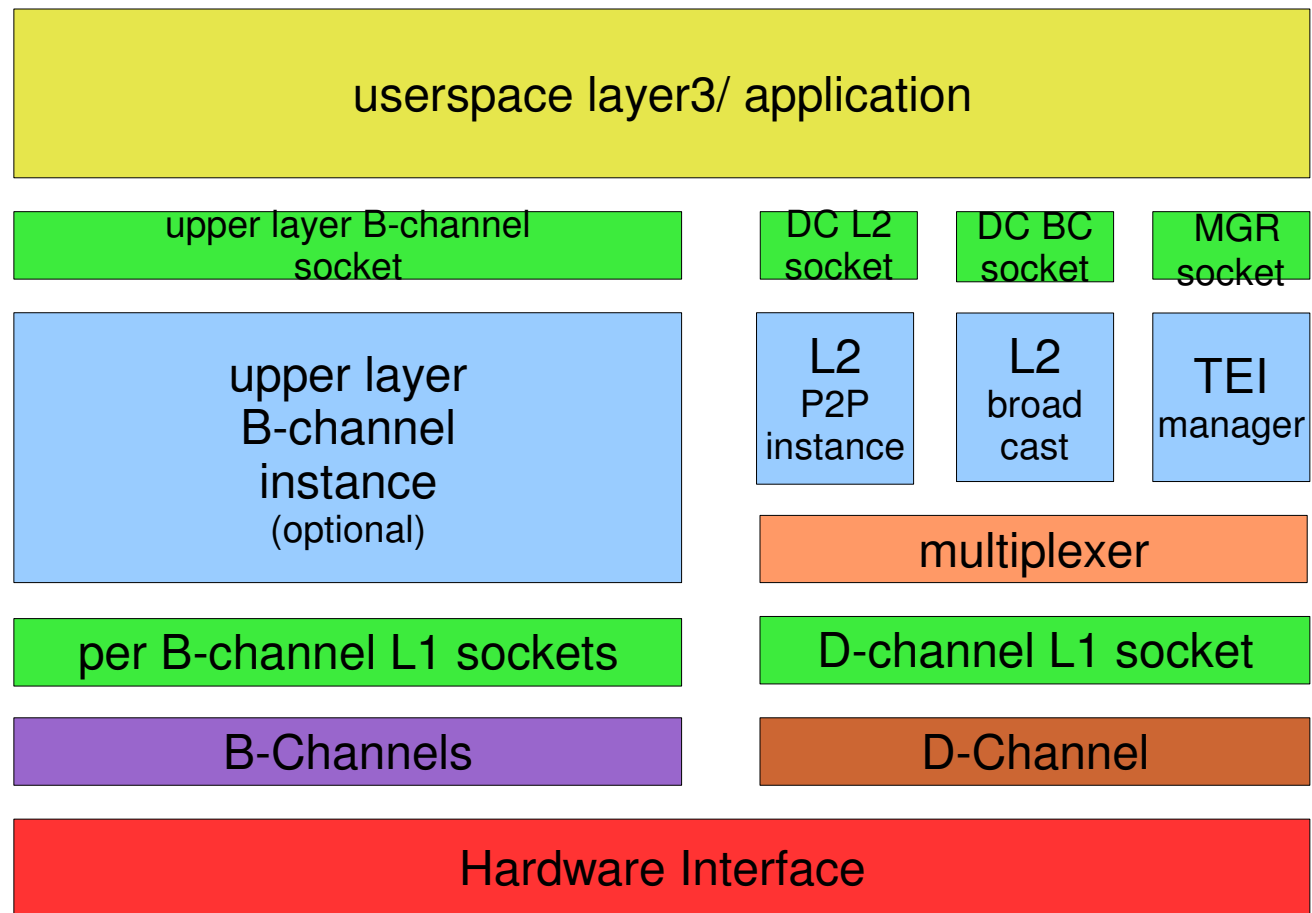
# History

- 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®

## General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. Novell, Inc., makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All Novell marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.5/>.

For other licenses contact author.



**Novell.**