Porting Linux IrDA
to the Windows Kernel

Jan Kiszka

Linux 2003 Conference
Edinburgh, Scotland
Overview

- IrDA Introduction
- Motivation: IrCOMM on Windows
- Architectures
- Analysis and Visualization
- Project Status
- Synergy Effects
- New Project Proposal
- Conclusion
IrDA

- Specification about how to exchange data using infrared beams
- Widely available: cell phones, PDAs, notebooks, PC adapters
- Half-duplex, serial, point to point transmission
- Typical baud rates from 9,6 Kbit/s up to 4 Mbit/s
- Why still IrDA when there is Bluetooth, etc.?
  - Hardware is lower cost and more common
  - Faster than Bluetooth (4 MBits/ vs. 1 MBit/s)
  - Intuitive user interface (point and shoot)
  - Security advantages: physically constrained link
  - No electromagnetic interference (802.11, cordless-phones, microwave oven, etc.)

*Infrared Data Association: industry group of more than 100 hardware and software vendors
*Baud rates:
  - 9,6 Kbit/s - discovery and link setup
  - 115 Kbit/s - RS232 IR adapters, small devices
  - 4 Mbit/s - USB adapters, notebooks
  - [16 Mbits/s - specified, but no hardware available]
• Physical Layer:
  - Interface between optical and electrical medium
  - Modulation

• IrLAP:
  - Frame wrapping
  - Link access control (master/slave-based when connected)
  - Device and link addressing
  - Discovery
  - Link (point-to-point) establishment with capability negotiation (baud rate, timing, frame size)
  - Error detection and frame repetition
  - Low level flow control

• IrLMP:
  - Service multiplexing over established links
  - Includes IAS

• IAP:
  - Used by Information Access Service (IAS)
  - Provides or retrieves information about available IrDA application protocols (service addresses, parameters)

• TTP:
  - Flow control at service level
  - Segmentation and reassembly

• OBEX: Exchanges files, phone book entries, etc.
• IrCOMM: Replacement of wired RS232 and Centronics
• IrLAN: LAN over infrared
• IrMC: Telephone specific services (phone book, calendar, messaging, voice, ...)
• IrXXX: Implement your own protocols (just like on top of TCP/IP)
**IrDA Status**

<table>
<thead>
<tr>
<th></th>
<th>Linux</th>
<th>Windows 2000/XP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Drivers</strong></td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td><strong>Basic Stack</strong></td>
<td>+...++</td>
<td>(+)</td>
</tr>
<tr>
<td><strong>OBEX</strong></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>IrLAN</strong></td>
<td>✔</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>IrNET (not IrDA)</strong></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>IrCOMM</strong></td>
<td>+...++</td>
<td>poor</td>
</tr>
</tbody>
</table>

• **Drivers:**
  - Linux support is constantly getting better (today less new IR chipsets)

• **Basic Stack:**
  - Linux stack very mature (e.g., it does not hang), memory leaks now fixed
  - Windows stack can hang in certain error situations, this could also be caused by user mode tools (which are definitely faulty)

• **IrLAN:**
  - Microsoft introduced IrNET (PPP-based) as a replacement of IrLAN (IR-to-Ethernet)

• **IrCOMM:**
  - Centronics support missing on Linux (minor importance)
  - And on Windows...?
The IrCOMM Issue

- Worked well with Windows 9x/ME
- NT4: no IrDA at all (commercial solution available)
- Minimal support on first Win2k version
  - No virtual COM port
  - Software should use socket API instead
    ➔ No cell phone modem support
    ➔ Hardly no PDA synchronization
    ➔ No control parameters (status lines, etc.)
- Win2k Service Pack 2 / WinXP
  - Cell phone modem support
  - Hidden virtual COM port
    (badly implemented, cell phones only)

Microsoft wanted software vendors to modify their programs, which didn't work as expected:
- Palm updated HotSync correctly
- Nokia bought third-party virtual COM port solution (very faulty)
- Many cell phone or sync tools never got updated

- End of 2000: First contact with IrDA on an internship (implemented simple IrDA/IrCOMM stack for 68K handheld)
- IrCOMM started as a free time project (I always wanted to write a Windows driver :-)
- First release 03/2001
IrCOMM2k – Porting Linux IrDA to the Windows Kernel

Jan Kiszka

IrCOMM2k Version 1

User Space

Application
(e.g. HyperTerminal)

System Service
ircomm2k.exe

WinSock API

WinSock API

MS-IrDA Stack

IR Adapter

Driver

COM Port Driver

ircomm2k.sys

COM Port Driver

/Device/Ircomm0

/Device/IrcommBridge0

Kernel Space

Hardware

COMx: COMMBRDG:

/Device/IrcommBridge0/Device/Ircomm0

 Straight forward approach: Use what Microsoft gave us (socket API for IrCOMM)
 Requires to disable built-in IrTRAN-P (picture transfer for digital cameras) support for incoming connections, which does not always work immediately
 Incoming status lines (CTS, DSR, RI, and DCD) can be statically set or cleared on virtual COM port
 It was a progress, but many programs still refused to work (PsiWin, cell phone tools, ...)
• Framework supporting structured development of interface drivers, filters, media translators, and protocols
• Defines set of available media types for upper and lower edges of drivers
• Specifies functionalities, which drivers have to provide
  - Packet handling
  - Miniport configuration objects (OIDs)
  - PnP and power management
• Provides library to access hardware, system resources, and other NDIS drivers
• Some strange limitations on miniport drivers (cannot easily access all kernel features), protocol drivers do not suffer from it
• Allows to control the binding order (user mode hook functions called during setup procedure)
NDIS and IrDA

- Additional NDIS medium “IrDA”
- Simple and well specified driver interface

**How to improve IrCOMM support?**

- Idea 1: Attach at TDI level
  - TDI details for IrDA not published
- Idea 2: Intermediate IrDA driver (filter)
  - Does not work with IrDA medium
  - But: Registration of parallel IrDA protocol driver possible
  - Manipulation of the binding order required
  - Full “shadow” IrDA stack is needed!
• NDIS binding manipulation:
  - MS-IrDA only binds to forward devices
  - IrDA2k only binds to real adapter devices
• One forward adapter per real IR adapter
• Several COM ports supported (shortcoming of the first version)
• Switching between IrDA2k and MS-IrDA:
  - Manually (using task bar menu of irmon2k)
  - Automatically: Cut off MS-IrDA as soon as a virtual COM port is opened, but try to send disconnect request/command first if MS-IrDA was connected
Components of Linux-IrDA

- Very well structured
- Almost a 1:1 implementation of the specification (especially state machines)
- Required for IrCOMM2k:
  - IrCOMM (without tty implementation)
  - Core layers (TinyTP, IAS, IrLMP, IrLAP)
  - Driver layer (requires adaption)
  - Most utils (without driver related crc* and wrapper*)
Porting the Driver Layer

- **Common functionality:**
  - Send / receive IrLAP payload data (no wrapping)
  - Report media busy
  - Set baud rate
  - Get adapter parameters (baud rates, timing, etc.)

- **Payload encapsulation**
  - Linux: skbuffs
  - Windows: NDIS packets
  - Mini skbuff inside private packet field
  - No fragmentation (not used by Linux-IrDA)
  - Every NDIS-IrDA packet must carry QoS parameters

```
Data

Buffer Discriptor

NDIS packet

Mini skbuff
```

- **skbuffs:**
  - Generic packet administration structure
  - Contains addresses and data pointers
  - Data can be fragmented

- **NDIS Packets:**
  - Equivalent on Windows (very similar)
  - Can include an additional private field
  - Preallocated pool required (can be extensible)

- **Mini skbuff contains only needed fields for IrDA**
Making Linux-IrDA Compile

- Provide new kernel header files
- Mappings
  - Memory allocation (size must be stored with every block!)
  - Spin locks (simple)
  - Timers (requires patches)
  - Work queues (mapped to a single semaphore)
- New message macros (IRDA_DEBUG etc.)
- Some additional fiddling
  - Many void #defines (__init, __attribute__, ...)
  - Very few patches (e.g., "#pragma pack" avoids structure alignment)
- Managed the include dependencies (hard work)

Maxim: Modify the Linux-IrDA source code as less as possible
  ➔ Easier updates to new kernel versions
- Timeouts: relative on Windows, absolute on Linux
  ➔ Remove “jiffies” from timeout calculations
- Work queues: only used for soft-IRQs of virtual IrCOMM ports (transmission complete, flow control events)
- Message macros (was a bit tricky):
  - VC++ does not support variable argument lists in macros!
  - Inline functions required
    - __FUNCTION__ not known to VC++, provides __FILE__ and __LINE__ instead
    - __LINE__ is an integer, cannot be concatenated to strings by the preprocessor (very annoying)
    - Do not want to modify every IRDA_DEBUG, map __FUNCTION__ to __FILE__
- Include dependencies:
  - Some kernel header files must include DDK headers, correct order is important
  - First ran into some circular dependencies
  - Include tree is not beautiful, but it compiles
Compiled Linux-IrDA!

- First test as a console application (irdalnx)
  - built-in ACTiSYS dongle driver
  - used old virtual COM port (ircomm2k.exe substitution)
- Finally integrated into irda2k.sys – and it worked!

• IrCOMM2k 2.0.0 enabled the first IR-based synchronization between PSION handheld and Windows 2000/XP
• All known cell phone tools work without problems (e.g. PEP2000, s25@once, Mobile Master, VisSie)
• First IrLPT server on Windows 2000 was implemented using IrCOMM2k 2.0.0 (captures printing of a mobile medical device)
Analyzing IrCOMM2k

- What to do with all the printks of Linux-IrDA?
  How to debug IrDA traffic, locally and on remote systems?

- Solution: new device for both debug messages and IrDA packets
- Simple console application (irda2kdump) prints messages and raw packets preserving the context
- Redirection to file enables simple remote diagnosis

  Manual packet parsing required

Both sent or received packets of IrDA2k and MS-IrDA can be captured!

Remote diagnosis:
- Is IrCOMM2k activated or is just MS-IrDA running?
- Does the IR device connect?
- Why does the connection break down?
- Is the remote device really IrDA compliant?
IrDA and Ethereal

- IrDA plugin for Ethereal
- No modification of Ethereal required
- First version by Shaun Jackman (2000, UDP-encapsulated)
- New approach: Extension replaces SunATM frame format
- Uses ATM pseudo header
  - Direction (received/sent)
  - Inserted log messages
IrDA and Ethereal (2)

- Offline capturing to libpcap file
  - Windows: irda2kdump -o <file>
  - Linux: irdadump -o <file>  
    (no log message support)
- Real-time capturing with modified WinPcap
- Modified libpcap on Linux?
- Full integration in Ethereal?
  - requires new file format
  - requires new packet type
- Offline capturing simply by pressing a button (Windows users...)
IrDA Monitor

- Yet simple monitoring (and configuration) tool irmon2k
- Link indication in task bar (“Am I already connected?”)
- Still only visions (a lot of work...):
  - Link parameter visualization (link speed, link quality, etc.)
  - Local and remote IAS browser
  - More configuration parameters (Linux-IrDA's /proc variables)
  - Portable implementation (using WxWindows or Java)
- Linux: GNOME IR monitor (girda)
  - Has to polls for discovery results every 2 seconds
  - Common event driven interface would be nice...
    (IrDA2k: blocking IOCTL of debug character device)

- Parameter visualization also includes alarms when link is interrupted
- Event IOCTLs of IrDA2k:
  - Wait for event (at the moment: connected/disconnected, IrCOMM2k active/inactive)
  - Get last event
- Linux-IrDA:
  - /dev/irnet can report events (but only IrNET related)
IrCOMM2k Status Overview

• Version 1.2.1:
  - stable (with known drawbacks)
  - support has been stopped
  - Thousands of downloads/month

• Version 2.0.0-alpha7:
  - Based on 2.4.20 + skb-leak patches
  - Stable driver core, but not bug-free
  - Difficult setup procedure

• CVS Version (06/2003):
  - Based on 2.5.67 + skb-leak patch

• New setup tool under development (Timothy Gack)

• Version 2 core status:
  - Some PSIONs (Symbian OS) do not connect reliably
  - Connection breakdown during long file transfers
  ➔ Unclear if bug in IrCOMM2k or Symbian OS (no test device available)
  ➔ Jean Tourrilhes: maybe due to IrTTP bug in 2.4.20, should be fixed in 2.5.67

• New contributors are always welcome!
Synergy Effects

- Several hidden bugs were discovered and fixed
- Common protocol analyzer
- Develop common GUI?
- Improve IrCOMM usability?

Let's think further:
- Bluetooth plugin for Ethereal?
- Linux Bluetooth for Windows?
  (An expert: “Linux currently provides the best implementation.”)
- TCP/IP with IPsec...?

Feedback to Linux-IrDA:
- Encouraged code revision to fix skb-leaks
- Solved very rare IrLAP connection problems
- Corrected IrCOMM status lines initialization

IrCOMM usability issue:
- Several applications listening on incoming IrCOMM connections, i.e. multiple COM ports are open
- Which port shall be used?
  (IrCOMM spec suggests IAS InstanceName attribute, but no device queries it)
- Solution: Assign devices to listening ports based on their nicknames and/or device addresses (similar approach in Linux-IrNET)
- Configuration interface required
Another Issue: IPsec clients

- How to communicate securely over insecure networks (scenarios: WLAN, road warrior access)?
  - IPsec tunnels
- Yet very poor usability, especially on client side
- Situation on Windows (client):
  - Only one full-featured client (SSH Sentinel, commercial)
  - Weak system integration (e.g., when changing networks)
  - Some stability and interoperability issues
  - Not easily pre-configured distributable (Version 1.3) – or not free
- Situation on Linux (client):
  - Missing DHCP-client support for IPsec
  - Scripts could be used for automated setup and profile switching
• Consequently applied abstraction: virtual NIC
• Virtual NIC contains all required components for IPsec tunnels
  - TCP/IP stack to contact VPN gateway independently
  - Filters to avoid illegal traffic forwarding
  - IPsec protocol (tunnel mode only)
• Use FreeS/WAN components (known to be very stable)
Conclusion

- IrCOMM2k provides missing virtual COM ports for Windows 2k/XP
- Based on almost unmodified Linux-IrDA stack
- Commonly usable components:
  - Ethereal IrDA plugin
  - GUI for configuration and monitoring [planned]
- Good example for software reuse in open source projects
- Reuse improves software quality
- Porting kernel components can be benefiting for both systems – and especially for the open source idea
- Suggested new project: IPsec client for Windows based on Linux implementation
Any Questions?

www.ircomm2k.de/english

Jan.Kiszka@web.de

Acknowledgments
I would like to thank Jean Tourrilhes for his very helpful comments on these slides.

Some more references
- Linux-IrDA:
  http://irda.sourceforge.net,
- Windows Platform SDK: IrDA
- Window DDK: Network Devices and Protocols (includes IrDA miniport drivers)
- Infrared Data Association:
  http://www.irda.org
- IrCOMM2k’s SourceForge site:
  http://sourceforge.net/projects/ircomm2k