openPOWERLINK over Xenomai

Pierre Ficheux (pierre.ficheux@smile.fr)

02/2017
• French embedded Linux developer, writer and teacher
• CTO @ Smile-ECS (Embedded & Connected Systems)
POWERLINK
Industrial bus

- Used to connect industrial devices in real time mode
- Main standards are both “serial” and/or Ethernet
  - CAN
  - MODBUS (-TCP)
  - Profinet
  - EtherCAT
  - AFDX (Avionics Full Duplex)
  - POWERLINK
- Ethernet is a standard
  - Easy to integrate, cheap hardware and good performances (standard CAN is 1 Mbps)
  - Homogeneous networking (routing, etc.)
  - No RT because of collision detection
### POWERLINK

- Deterministic Ethernet based industrial bus
- Originally created by B&R automation (Austria) in 2001
- Managed since 2003 by open organization EPSG (Ethernet POWERLINK Standardization Group)
- Leverage advantages of Ethernet for RT networking systems
- 1.8 M systems installed (2016)
- Min cycle time is 100 µs, 240 nodes on a single network
- Software only → works on standard Ethernet
- Avoid collisions thanks to a software only protocol :-)
- Open-source version (2.x) openPOWERLINK available from SourceForge
- Now used by B&R hardware (PLC)
POWERLINK frame

Powerlink frame “inside” Ethernet payload
POWERLINK protocol

- One “manager” node (MN) and several “controlled” nodes (CN)
- Cycle divided in 3 steps
  - MN synchronizes CNs with a SoC (Start of Cycle) frame which starts “isochronous phase” (RT)
  - CN receives \textit{PReq} (Poll Request) from MN, and replies with \textit{PRes} (Poll Response) and data (RT)
  - Last step is “asynchronous phase” (no RT) started with \textit{SoA}. Addressed node should answer \textit{ASnd}
- Standard IP-based protocols and addressing can be used during the asynchronous phase
POWERLINK protocol
• Open source implementation of POWERLINK stack
• BSD license
• Support for Linux, Windows, Xilinx/Altera FPGAs
• Official support for x86, ARM (Zynq)
• CMake based → CMAKE_TOOLCHAIN_FILE for cross-compilation
• Buildroot integration by Smile-ECS (latest stable 2.4.1)
• Building process:
  - Stack
  - Drivers (if necessary)
  - Demo applications MN/CN (console, Qt)
Architecture 1 (kernel)

- Application in user space
- Stack and drivers in kernel space
- High performance and precision
- Specific drivers (stack/src/kernel/edrv for Ethernet drivers)
  - About > 10 supported controllers
  - No Linux “mainlining”
- Hard to debug (kernel space)
• Move stack to user space
• Use *libpcap* library (tcpdump) to talk with standard Linux driver
• Proven solution
• Much easier to debug
• Based on Linux NIC drivers
• Better with PREEMPT_RT patch
• 100 \( \mu s \) jitter (only 40 \( \mu s \) in kernel)
Xenomai
OpenPOWERLINK / Xenomai
Linux/Xenomai & RT :-)

OpenPOWERLINK / Xenomai
• Using Linux as “RTOS” is very interesting
  – POSIX
  – Hybrid approach → UNIX + some RT tasks
  – Usable as a standard UNIX

• 2 solutions :
  – Upgrading Linux kernel RT performance
    (PREEMPT_RT, the “official” way)
  – Adding a RT “co-kernel” sharing hardware with Linux
    (RTLlinux, RTAI, Xenomai)
• Maintained by Thomas Gleixner
• Mostly used on x86 (but runs on most recent ARM, Nios2, Microblaze)
• Needs a mainline kernel (or something like)
• Very easy to install (just a kernel patch)
• Same programming API as standard kernel (user and kernel space)
• 50 μs jitter (x86/Atom), 150 μs on Raspberry Pi B+
• Currently used with openPOWERLINK
• Official project of Linux foundation since Oct. 2015!
• Adding co-kernel for RT tasks
  – RT subsystem inside kernel module(s)
  – Needs kernel patch for hardware resource (IRQ) virtualization

• Main projects
  – Kernel only (RTLinux, 1996) → “dead”
  – Kernel & (partially) user space (RTAI, 1998)
  – Full user space integration (Xenomai, 2001)

• 10 µs jitter on Atom/x86, 50 µs on Raspberry Pi B+
RTLinux architecture (kernel only)
• Created by Philippe Gerum
• Xenomai = realtime Linux subsystem
  – RT tasks in user space
  – RT driver API = RTDM for “Real Time Driver Model”
  – RT network stack = RTnet !
• Include “skins” for POSIX, VxWorks, VRTX, uITRON, pSOS+, etc.
• Runs on top of I-pipe (Interrupt pipeline)
  – Xenomai domain (RT)
  – Linux domain (No RT)
• Currently 2.6.5 and 3.0.3
• 3.0 uses co-kernel (Cobalt) or PREEMPT_RT (Mercury)
• GPL license (kernel), LGPL (user)
Xenomai 3 architecture (Cobalt)
- I-pipe = interrupt source for “domains” (Xenomai, Linux) based on ADEOS technology
- Highest priority to RT domain (Xenomai)
- Stalled/unstall domain instead of hardware CLI/STI
Xenomai in industry

- CANFestival (CANopen stack)
- PEAK System CAN boards drivers
- EtherCAT master
- RTDM SPI driver (i.MX28)
- BEREMIZ, integrated development environment for machine automation
- And much more...
openPOWERLINK over Xenomai
Proof of concept

- Started as 2 internships
  - Damien Lagneux for the first version on i.MX6 boards and Buildroot (Armadeus APF6, RIOTboard → RTnet support for “FEC” controller)
  - Geoffrey Bonneville for improvement and Android testing (AIOSP) on BeagleBone Black
- No Raspberry Pi because of USB based Ethernet (though there is Pi2 port for openPOWERLINK!)
- Using openPOWERLINK in Xenomai domain
- PREEMPT_RT / Xenomai comparison!
• Xenomai v2 contribution, merged with v3
• Based on RTDM (protocol device)
• Limited hardware support (dedicated driver API)
  – FEC, AT91, AM335x (BB Black, external contrib)
  – RTL8139, Natsemi, PCnet32
  – MPC8xxx

• Example session (BB Black)
  # insmod rtnet .ko
  # insmod rt_smsc.ko
  # insmod rt_davinci_mdio.ko
  # insmod rt_ticpsw.ko
  # insmod rtpacket.ko
  # insmod rtipv4.ko
  # rtifconfig rteth0 up 192.168.1.1
  # rtroute add 192.168.1.2 00:22:15:80:D5:88 dev rteth0
  # rtping 192.168.1.2
• One openPOWERLINK architecture is based on libpcap
• Libpcap (Linux) is based on “packet” protocol
• Xenomai RTnet stack includes packet socket support ("rtpacket" module)
• Porting libpcap to Xenomai is quite difficult!
• Hardware is limited by RTnet driver availability (but it's just an POC !)
• PCAP layer removed
• Sending / receiving packet (through packet socket) directly from/to the openPOWERLINK stack
• RTnet architecture is close to openPOWERLINK “kernel” architecture
- 1 i.MX6 board as MN
- 2 B&R modules as CN
- 1 B&R capture module (timestamping)
- 1 PC for saving frames
Xenomai solution is close to Linux “kernel” version of openPOWERLINK (architecture 1)

Based on Baumgartner/Schoenegger/Wallner papers (B&R)

Stress with cpuburn, dd, hackbench

Jitter for 500 μs cycle

Better than PREEMPT_RT!
Conclusion + future work

- Currently not stable enough for industrial use → stack debug and optimization
- Test with more CN
- Work to be done with B&R
  - mainlining in openPOWERLINK project
  - more test with available POWERLINK devices
- PREEMPT_RT version is fair enough for most projects
- Currently not a strategic market for Smile-ECS (except one customer)
Bibliography

- http://www.ethernet-powerlink.org
- http://openpowerlink.sourceforge.net/web
- https://lwn.net/Articles/572740
- http://www.beremiz.org
- “Introduction to RTnet”, P. Ficheux, Open Silicium #15 (french)
- “Improving openPOWERLINK with RTnet”, G. Bonneville, Open Silicium #19 (french)
- “openPOWERLINK et Xenomai” D. Lagneux (Internship report, french)
- “A(I)OSP, Android – Industrial - Open Source Project” G. Bonneville (Internship report, sfrench)