

# About the challenges and successes with Xenomai as RTOS for Routers with integrated Ethernet Switches

Andreas Glatz  
andreasglatz@ruggedcom.com

*Ruggedcom Inc.  
Concord, ON, Canada*

## Outline

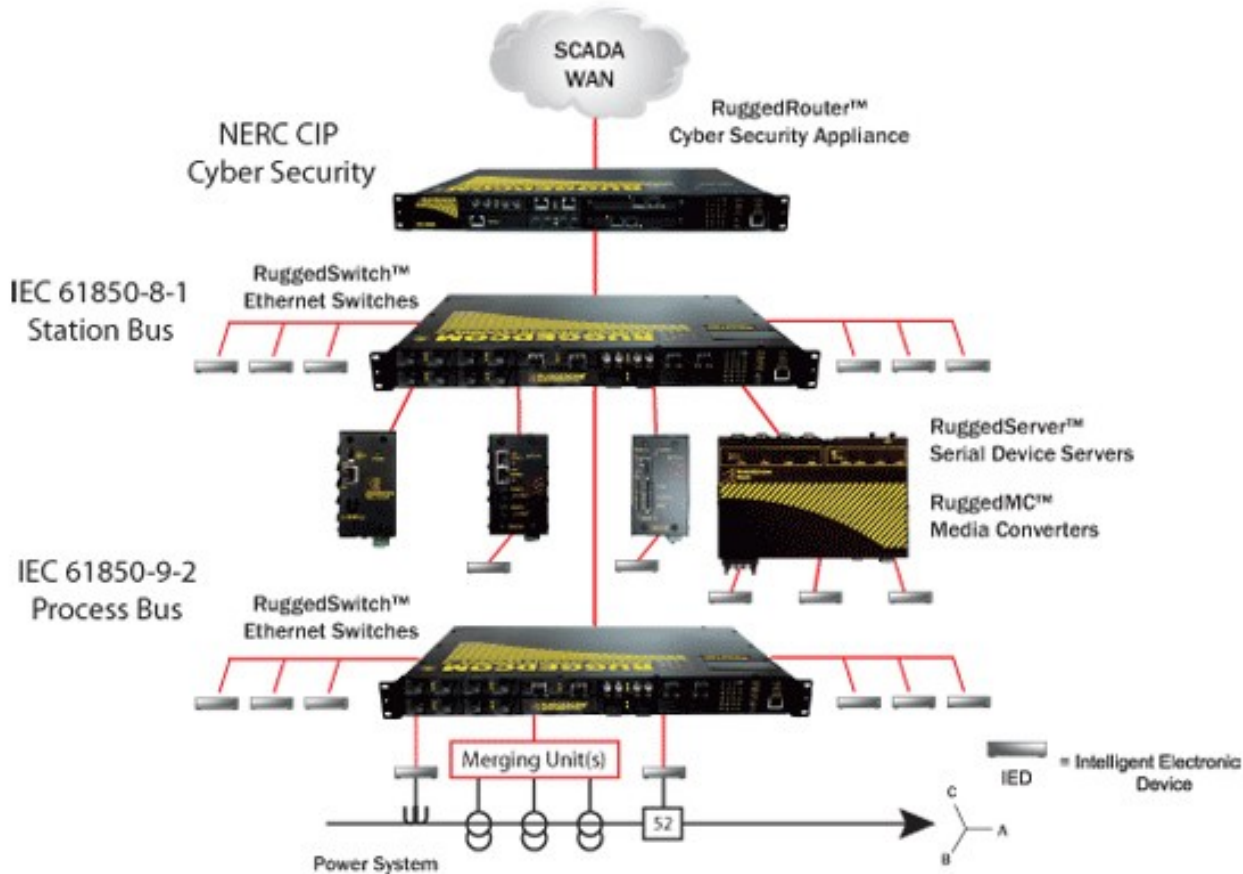
- **Company facts**
- **Motivation for using Xenomai**
- **Development environment**
  - **IDEs**
  - **Debugging**
  - **Tracing**
- **Outlook**

## Company Facts

- **Founded 2001 by Marzio Pozzuoli (President and CEO) near Toronto, Ontario, Canada**
  - **Public: TSX:RCM**
  - **~ 200 employees**
- **Designs and manufactures Layer2 and Layer3 networking devices for harsh electrical and climatic environments**
- **Key markets**
  - **Electrical Utility, Transportation, Industrial, Military**
- **Strong customer base all around the world**
  - **ABB, GE, Siemens, Hydro One, National Grid, Alcoa, Chevron, Boeing, Lockheed Martin, US Navy**

## Company Facts (cont'd)

- Switches and Routers in a power grid substation



## Company Facts (cont'd)

- RuggedSwitch (RSG2300)



- Rugged Operating System (ROS)
  - Well developed and tested Embedded C++ code (no rtti, no exceptions, no standard C++ library!)
  - Coldfire running traditional RTOS
- Supports advanced layer2 features
  - RSTP, VLAN, Link aggregation, ...

## Company Facts (cont'd)

- RuggedRouter (RX1100)



- Rugged Operating System on Linux (ROX)
  - Well tested standard (Quagga, Webmin, ...) and custom applications
  - x86 running Debian Linux
- Supports Layer3 (IP) features
  - Firewall, Routing protocols (OSPF, ...), DHCP Agent, Traffic prioritization, NTP Server, ...

## Motivation for using Xenomai

- **NEW: RuggedBackbone**
- **Router with integrated Switch**
  - Should have all features of a RuggedSwitch and a RuggedRouter
  - New hardware with highest port density so far
- **Obvious strategy**
  - Reuse as much software as possible
  - Combine IP from previous products



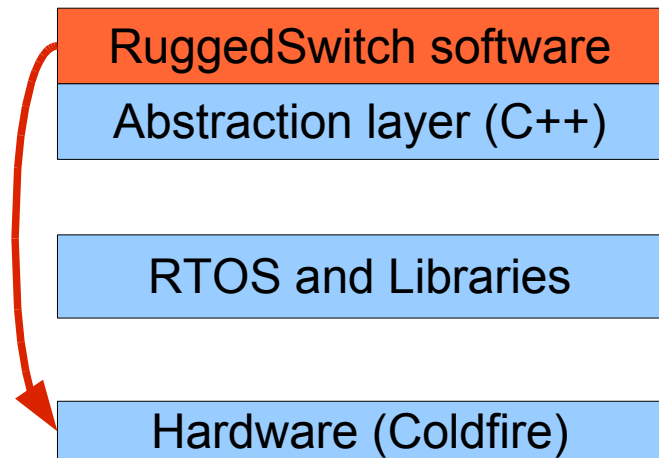
## Motivation for using Xenomai (cont'd)

- **Problems**
  - **Poor hard real-time capabilities of Vanilla Linux Kernel**
  - **Commercial RTOSes available but without rich feature set as offered by Linux**
- **Our solution: Xenomai**
  - **RuggedRouter software runs in Linux domain**
  - **RuggedSwitch software runs in Xenomai domain: preemption of any Linux activity (including routing)**

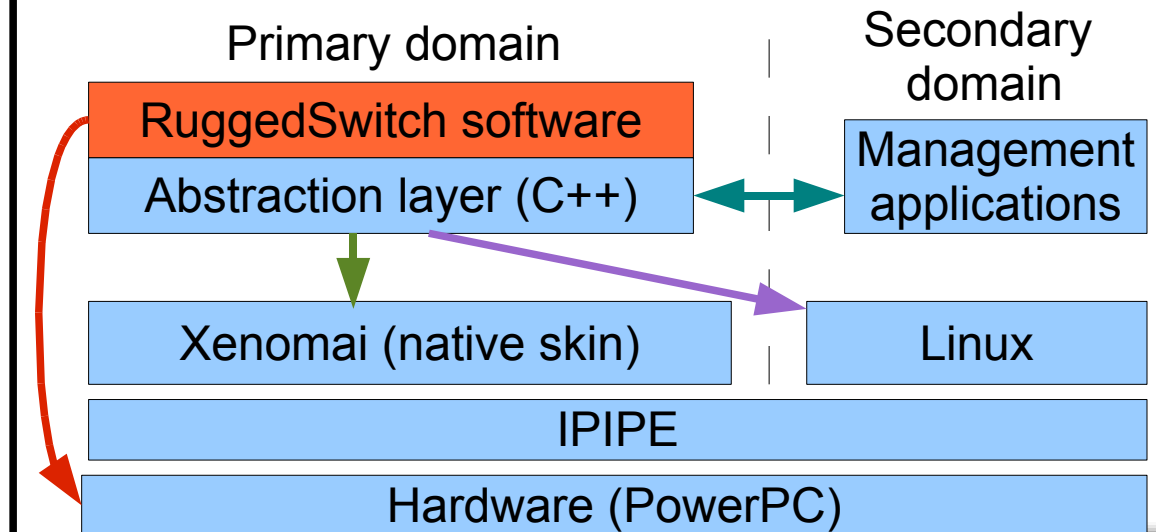
## Motivation for using Xenomai (cont'd)

- **Goal:** Run exactly the same RuggedSwitch software on traditional RTOS and Xenomai
- **Gain:** Reduces required amount of manpower for maintenance and testing to a minimum

### RuggedSwitch



### RuggedBackbone



## Development Environment

- For the past 5 years:
  - RuggedSwitch software developers used a very mature IDE (with integrated Remote Debugger, Profiler, ...)
- Alternatives for Xenomai (Linux) Software development?
  - 'Traditional' Linux development environment
    - Vim, Emacs, Gdb, Gcc, Make, ...
  - IDEs available on Linux (most mature)
    - Eclipse, Kdevelop, ...

## Linux IDEs

- Comparison of most mature IDEs

	Eclipse	Kdevelop
Current version	Galileo	3.5.5
Dependencies	Sun's J re	KDE libraries
Responsiveness	acceptable	very fast
Gdb frontend	good (native gdb plugin), buggy (DS F plugin)	primitive
Indexer	close to acceptable (build-in)	very good (ctags)
Visual appearance and configurability of Editor	very good	good to very good
CVS, SVN, ... integration	acceptable	acceptable
3 <sup>rd</sup> party plugin providers	Windriver, Abatron, Lttnng, Intel, IB M, ...	no plugins

## Remote Debugging

- Remote debuggers

Debugger	Type	User interface	User-space		Kernel-space	
			real-time	non real-time	real-time	non real-time
<b>gdb</b>	Software	Text, Eclipse, Kdevelop	<b>gdbserver</b> (problems with Xenomai 2.4.9)	<b>gdbserver</b>	<b>kgdb</b> (only x86 and ttyS)	<b>kgdb</b> (ttyS, Ethernet)
<b>B DI2000</b>	Hardware		practically not available because B DI doesn't know anything about processes / threads	yes		
<b>Lauterbach</b>		Motif based GUI, Eclipse plugin (?)	yes			
<b>Windriver</b>		Eclipse plugin	Problems installing it because it depends on specific RedHat version			

## Remote Debugging (cont'd)

- **Problems with remote debuggers for Xenomai (Linux)**
  - Sometimes single stepping doesn't work because gcc doesn't generate proper debug information
  - Combined kernel- and user-space debugging with BDI2000: BDI catches user-space breakpoints
  - $\geq$ Linux-2.6.30: gdbserver freezes when debugging code where a thread is created inside of another thread

## Tracing

- **Simple tracing**
  - `rt_printf()`
  - **GPIOs, Time Stamp Counter (TSC) and Oscilloscope**
- **Advantage**
  - **Most accurate way of tracing as it is taylorred for a specific piece of code**
- **Problems**
  - **Many development cycles to find the right spot to place the probes**
  - **Often applies to only one piece of code**

## Tracing (cont'd)

- **Tracers**
  - Compared to commercial solutions, support for tracing Xenomai applications is very basic
  - Tracing markers/probes are placed in kernel code
  - Development is slowly moving towards markers/probes in user-space applications

Tracers	Kernel version	Principal of operation	Works with Xenomai	Interface	Trace duration
I-pipe	Comes with Xenomai	Instrumentation of every kernel function	yes	/proc	Short
Lttng	Since 2.6.12	Static markers, Kprobes	yes	Trace gets written to a file by a User-space daemon	Long
Ftrace	Since 2.6.27	Instrumentation of kernel functions	N/A	/proc	Short
SystemTap	Relies on Kprobes	Kprobes	N/A	Typically printk	Varies

## Tracing (cont'd)

- **LTTng**
  - **~10min tracing with 5 markers armed produced about 1GB of data (78.350.043 events)**
    - `xn_nucleus_sched_switch()`
    - `xn_nucleus_irq_enter()`
    - `xn_nucleus_irq_exit()`
    - `xn_nucleus_irq_enable()`
    - `xn_nucleus_irq_disable()`
  - **Analysis of textual output: No lttv version which can produce a graphical representation of the trace**
  - **It took some time to find the right size of the kernel tracing buffers so that no events are lost**

## Tracing (cont'd)

- Sometimes tracing events are not ordered by time
- Sample output:

```
xn_nucleus_sched_switch: 146.891552548 (huhu/cpu_0), 0, 0, , , 0, 0x0, MODE_UNKNOWN  
{ thread_out = 0xe1046848, thread_out_name = "StartTask", thread_in = 0xc03b5db8,  
thread_in_name = "ROOT" }
```

```
xn_nucleus_irq_exit: 146.891556943 (huhu/cpu_0), 0, 0, , , 0, 0x0, MODE_UNKNOWN { irq =  
43 }
```

```
xn_nucleus_irq_enter: 146.891566528 (huhu/cpu_0), 0, 0, , , 0, 0x0, MODE_UNKNOWN { irq =  
43 }
```

```
xn_nucleus_irq_exit: 146.891576218 (huhu/cpu_0), 0, 0, , , 0, 0x0, MODE_UNKNOWN { irq =  
43 }
```

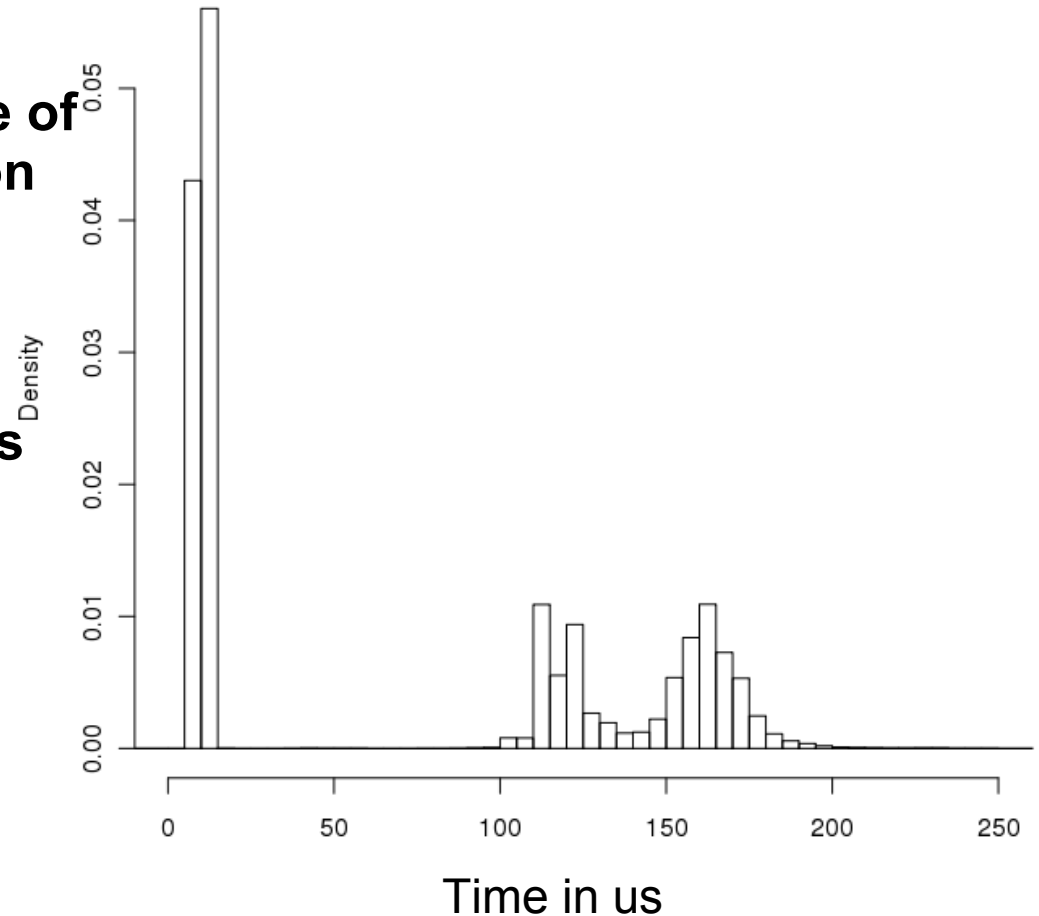
```
xn_nucleus_irq_enter: 146.891743633 (huhu/cpu_0), 0, 0, , , 0, 0x0, MODE_UNKNOWN { irq =  
43 }
```

```
xn_nucleus_irq_disable: 146.891756788 (huhu/cpu_0), 0, 0, , , 0, 0x0, MODE_UNKNOWN { irq  
= 43 }
```

```
xn_nucleus_irq_disable: 146.891759338 (huhu/cpu_0), 0, 0, , , 0, 0x0, MODE_UNKNOWN { irq  
= 43 }
```

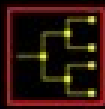
## Tracing (cont'd)

- UEC ISR duration histogram
  - derived from LTTng trace of RuggedSwitch application running in kernel-space
  - Tx ISR duration is ~10us
  - Rx ISR duration varies between 120us and 160us
- Tracing overhead:
  - 10%- 25% increase in duration
  - Double peak in RxISR duration histogram



## Outlook

- **Open issues**
  - **Possibility to have tasks which have a lower priority than Linux**
  - **Support for debugging distributed architectures**
  - **Xenomai plugin for LTTV**
- **Xenomai/SOLO**
  - **Runs on top of PREEMPT\_RT patched kernel**
  - **Provides native Xenomai API**
  - ***Possible solution to first issues***



**Danke fuer Ihre  
Aufmerksamkeit**