2009 International Conference on Communication Software and Networks

Implementation of Real-time Network Extension on Embedded Linux

Yuan Tian[1,2]
1. Institute of Optics and Electronics, Chinese Academy of Science
2. Graduate University of the Chinese Academy of Science
Chengdu , China
ioecas@gmail.com

Guoqiang Ren
Institute of Optics and Electronics, Chinese Academy of Science
Chengdu , China
ioeren@163.com

Qinzhang Wu
Institute of Optics and Electronics, Chinese Academy of Science
Chengdu , China
ioeren@163.com

***Abstract*—Linux over the past few years has gained in popularity as the operating system for embedded networking equipment. Its reliability, low cost and undisputed networking capabilities made it one of the most popular choices for the networking devices. But traditional software network interfaces in Linux do not deliver satisfactory real-time performance. Hence alternative efficient real-time interfaces are required in network monitoring, distributed systems, real-time networking and remote data acquisition applications. So it is necessary to modify the original Linux to meet the real-time requirement. This paper describes the implementation of real-time network extension based on embedded Linux. Compared with different solution of achieving real-tine ability on Linux system, Xenomai and Rtnet have been chosen in our system. Finally, the real-time performance test has been carried out on embedded Linux network system. The test results indicate that, through applying Xenomai and Rtnet on embedded Linux, the hard real-time requirement can be met in our system.**

Keywords: Xenomai;Rtnet;Embedded Linux;Real-time

I. INTRODUCTION

PowerPC440 embedded processor was used to control and manage the entire system in hardware system. PowerPC440 is designed specifically to address high-end embedded applications and provides a high-performance, low-power solution which is able to interface to a wide range of peripherals. With on-chip power management features and intrinsically lower power dissipation, this embedded processor is suitable for embedded applications. This chip contains a high-performance RISC processor core, DDR SDRAM controller, PCI-X bus interface, Ethernet interface, control for external ROM and peripherals, DMA with scatter-gather support, serial ports, IIC interface, and general purpose I/O. Figure 1 illustrates the architecture of the hardware system based on PowerPC440[1].

Today, the use of Linux in embedded systems become more popular, because of its proven networking capabilities and well suiting for networking applications and services that require high reliability and high availability [2]. So Linux was chosen as operating system in our designing. However, Linux is not a real-time operating system. There are usually two approaches to make Linux real-time. The one is using a second kernel to schedule real-time tasks: solutions include

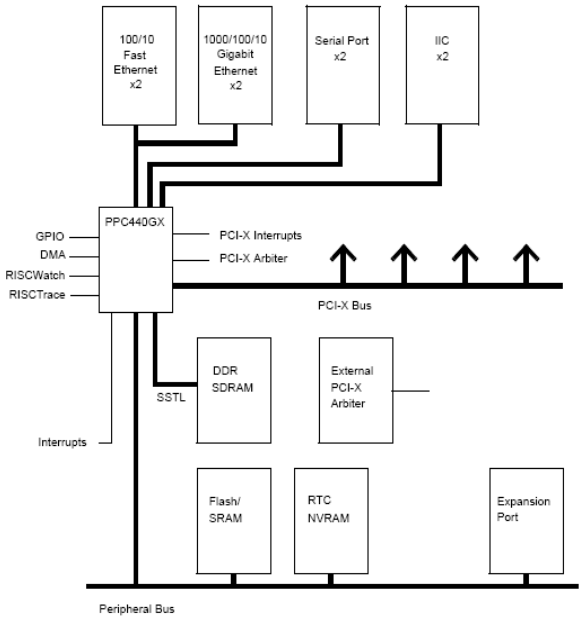

Figure 1. Architecture of the hardware system

Xenomai/ADEOS, RTLinux and RTAI, etc. The other is improving Linux kernel itself with regards to preemption, low latency, etc. In order to implement hard real-time network, Xenomai/ADEOS was chosen as real-time operating system mentioned in first method. Xenomai is a real-time development framework cooperating with the Linux kernel. It implements a micro-kernel with real-time scheduler. Xenomai's real-time nucleus and Linux kernel are in two ADEOS domains. Xenomai runs in a higher priority domain than Linux kernel. It also implements different APIs providing real-time services, like creating real-time tasks, timers, semaphores [3].

RTnet is an open source hard real-time network protocol stack for Xenomai and RTAI (real-time Linux extensions).It makes use of standard Ethernet hardware and supports several popular card chip sets, including Gigabit Ethernet. RTnet implements UDP/IP, ICMP and ARP in a deterministic way. It provides a POSIX socket API to real-time user space processes and kernel modules. Access to

978-0-7695-3522-7/09 $25.00 © 2009 IEEE
DOI 10.1109/ICCSN.2009.119

nondeterministic media is managed by the pluggable RTmac layer and the actual control discipline. As default for Ethernet, a Time Division Multiple Access (TDMA) discipline is provided.

In this paper, we describe the implementation of real-time network extension based on embedded Linux. The remainder of this paper is organized as follows. In section 2, the procedure of setting up bootloader and Linux kernel is present. Section 3 describes the architecture of Xenomai and how to use it to realize real-time kernel. Then RTnet based on Xenomai is introduced in section 4. Finally, we conclude in section 5.

II. SETTING UP BOOTLOADER AND LINUX KERNEL

Though the bootloader runs for a very short time during the system's startup and is mainly responsible for loading the kernel, it is a very important system component. It is a special task for embedded Linux systems, because the bootloaders used in such systems are either completely different from those used in common systems or, even when they are the same, are configured and operated in very different ways.

U-Boot, the universal bootloader, is arguably the richest, most flexible, and most actively developed open source bootloader available. U-Boot is based on the PPCBoot and ARMBoot projects. The version of U-Boot we used is 1.1.4 [4].

There are a lot of examples of evaluation board in U-Boot, but our hardware system is different from them. So some codes in U-Boot should be changed to adapt our system. Three main aspects of code should be modified. First, some addresses of peripheral hardware are needed to change. Second, the parameters to initialize SDRAM are adjusted. Finally, some device drivers should be rewrite. After finishing the three steps, U-Boot commands can be used to check system information, for example printenv, flinfo etc.

As the same with U-Boot, there are also lots of examples for different evaluation board in Linux source. The codes of an evaluation board which is similar with our hardware system are chosen. Then make some modification in Linux kernel source which is similar in U-Boot. After compiling the kernel, a uImage in the kernel source can be gained. Now U-Boot is used to download the uImage and startup system.

One of the last operations conducted by the Linux kernel during system startup is mounting the root file system. The root file system has been an essential component of Linux systems from the start. At the stage of test, network file system is used as root file system. The host nfs server is enabled and export a directory as root file system to target system. Then the target can mount this directory and share the file with host. After finishing testing, U-Boot, kernel and file system should be put into flash in three different partitions. In order to implement flash partition, Linux's MTD subsystem is used. JFFS2 is chosen as root file system. Though JFFS2 doesn't achieve compression ratios as high as CRAMFS, it has to maintain space for garbage collection and metadata structures that allow file system writing. JFFS2 provide power-down reliability and wear-leveling, which are very important characteristics for devices that rely on flash storage. The creation of a JFFS2 image is fairly simple:

```
mkfs.jffs2 -r rootfs/ -o images/rootfs-jffs2.img
```

Once you create the JFFS2 image, you can write it to its designated MTD device. But you first need to erase the MTD device where the image will be placed:

```
eraseall /dev/mtd0
```

With the MTD device erased, copy the JFFS2 image to the MTD partition:

```
cat images/rootfs-jffs2.img > /dev/mtd0
```

Now, mount the copied file system to take a look at it. If your target had previously been using an nfs-mounted root file system, you are now ready to boot it using the JFFS2 file system as its root file system. Thus the entire system set up automatically and successfully.

III. XENOMAI EVALUATION

Xenomai is a new real-time operating system emulation framework based on Linux. It aims at providing a consistent framework that helps implementing real-time interfaces and debugging real-time software on Linux. Xenomai comes with a growing set of emulators of traditional RTOS APIs, which ease the migration of applications from these systems to a Linux-based real-time environment.

It was designed with the goal to help application designers using traditional and proprietary real-time operating systems to move as smoothly as possible to a Linux based execution environment. Xenomai relies on the common features and behaviors found between many embedded traditional RTOS, especially from the thread scheduling and synchronization standpoints. These similarities are exploited to implement a nucleus exporting a set of generic services. These services grouped in a high-level interface can be used in turn to implement emulation modules of real-time application programming interfaces, which mimic the corresponding real-time kernel APIs [5].

A. Architecture of Xenomai

To make Xenomai's tasks hard real-time in Linux a Real-Time Application Interface co-kernel is used. It allows running real-time tasks seamlessly aside of the hosting Linux system while the tasks of the regular Linux kernel can be seen as running in a low-priority mode. The Real-Time Application Interface co-kernel shares hardware interrupts and system-originated events like traps and faults with the Linux kernel using the Adaptive Domain Environment for Operating Systems (Adeos) layer, which in turn ensures Real-Time Application Interface low interrupt latencies. The entire architecture of Xenomai is shown in Figure 2. Adeos is an event pipe line. The purpose of Adeos is to provide a flexible environment for sharing hardware resources among multiple operating systems, or among multiple instances of a single operating system. It has been ported to PowerPC. ADEOS is also known as "I-pipe", it delivers system events (interrupts, exceptions, system calls) in a timely and prioritized manner, along a "pipe line" of domains.

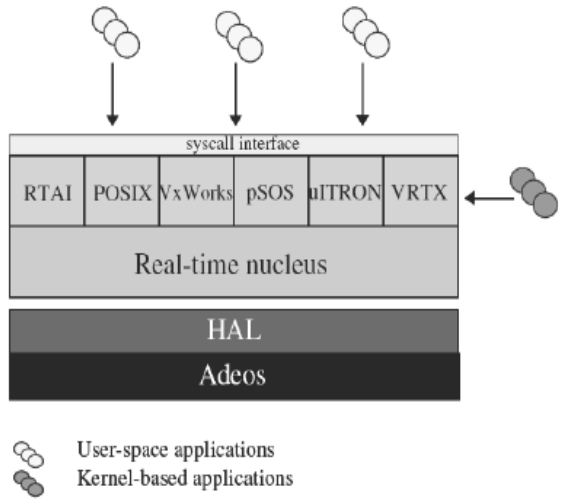

Figure 2. Architecture of Xenomai

B. *Apply Adeos and Xenomai Kernel Patch to the Linux Kernel*

Run the Xenomai configuration script to apply the kernel patch [6].

```
xenomai-root/scripts/prepare-kernel.sh
--arch=ppc --adeos=$adeos-patch-dir/adeos-ipipe-2.6.19.patch --linux=$linux-tree
Configure and build kernel:
make menuconfig
make
```

Build Xenomai applications and install the applications to Linux file system directory. This will build Xenomai in FDPIC format.

```
Xenomai-root/configure --host=ppc
make install
```

C. *Performance of Xenomai*

After finishing installation of Xenomai, now we can use the tools in Xenomai to test the system latency. This latency example program features a periodic real-time thread measuring its scheduling latency over periods of 100 us. Each second, results are posted by the sampling thread to a display thread, which in turn outputs the latency figures to the screen using standard Linux services. This illustrates the seamless migration of Xenomai's real-time threads between the Xenomai and Linux schedulers, in order to perform every system call issued by xeno-enabled Linux tasks from the proper execution context. The latency involves handling timer interrupt and current thread scheduled to run. Test results are shown in two different situations. One is without system load in table 1, the other is in heavy load in table 2. All results are in microseconds.

TABLE I. LATENCY WITHOUT LOAD

Min	Avg	Max	Overrun
-4.226	-3.574	6.385	0

TABLE II. LATENCY WITH HEAVY LOAD

Min	Avg	Max	Overrun
-4.211	-3.521	13.935	0

Two tables show that the maximum latency is 13.935us even though in heavy load situation, which doesn't overrun the real-time demand.

IV. RTNET EVALUATION

With the goal to provide a widely hardware independent and flexible real-time communication platform, RTnet project has been founded in 2001 at the University of Hannover, based on ideas and source code of a previous effort to provide deterministic networking. RTnet is a purely software-based framework for exchanging arbitrary data under hard real-time constraints. The available implementation is founded on Linux with the hard real-time extension RTAI [7].

A. *RTnet Stack*

RTnet implements a real-time capable IP-Protocol stack. Thus was done by porting the robust UDP/IP stack of the Linux kernel to RTAI. Furthermore the Address Resolution Protocol (ARP) that is normally dynamic is replaced by a static solution. The Media Access Control (MAC) sublayer from standard Ethernet (which is nondeterministic because of the stochastic media access mechanism CSMA/CD) is replaced by:(1) a token based MAC variant or a (2) Time Division Multiple Access (TDMA) approach. Because of the modularity of Rtnet and the available source code it is possible to implement a different mechanism for media access. A wide difference to the above-described Powerlink protocol is, that no dedicated protocol controller is needed. The RT capabilities are based on the real-time functionality of RTAI. Rtnet has been designed to lend real-time networking functionality to both the RTAI and Xenomai environments. It provides a customizable and extensible framework for hard real-time communication over Ethernet and other transport media.

The design of the RTnet stack as depicted in Figure 3 was inspired by the modulised structure of the Linux network subsystem. It aims at scalability and extensibility in order to comply with the different requirements of application as well as research scenarios. RTnet's software approach addresses both the independence of specific hardware for supporting hard real-time communication and the possibility to use such hardware nevertheless when it is available. Furthermore, it enables the integration of various other communication media beyond Ethernet [8].

B. *Installation of RTnet and Driver Porting*

Now the installation of RTnet will be explained and the test result of it will also be shown later. Firstly, RTnet source code should be downloaded [9]. The instructions are given below.

```
cd to a preferred directory (e.g. /usr/src)
tar xvjf /usr/src/rtnet-0.9.10.tar.bz2
cd rtnet
```

make menuconfig (run "make help" for further information)

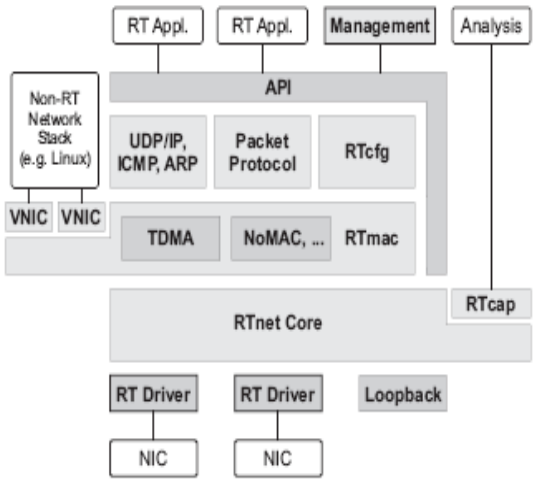

Figure 3. RTnet Stack

Set at least the real-time extension installation path and the path RTnet is to be installed to. You could also check other options if they fit your requirements. Help is available for each feature.

make

make install

Because Ethernet media access controllers in PowerPC aren't supported by RTnet currently. So EMAC driver should be ported from Linux to RTnet. A list is shown below about porting Ethernet device driver to RTnet.

- Add the following fields to private data:
  struct rtskb_queue skb_pool;
  rtdm_irq_t irq_handle;
- Initialize skb pool in probe or init function:
- Replace struct net_device with struct rtnet_device
- Replace netif_stop_queue with rtnetif_stop_queue
- Replace struct sk_buff with struct rtskb
- Replace netif_rx with rtnetif_rx
- Revise the xmit routine
- Modify interrupt handler
- Replace alloc_etherdev with the following lines
- Replace dev_alloc_skb(size) with dev_alloc_rtskb (size, &<priv>->skb_pool)

Ten point just be shown about porting, but they are not enough. It is recommended to take a look at existing drivers in RTnet source if some steps remain unclear.

C. RTnet Testing

After installations of RTnet, there are still several steps should be taken before testing. First, shutdown the network device which shall become part of the RT-network and remove its driver module (this also means that it must not be compiled into the kernel!). Second, Load required real-time modules (xeno_hal, xeno_nucleus, xeno_rtdm). Third, Check /etc/rtnet.conf and adapt at least the following parameters: RT_DRIVER, IPADDR, TDMA_MODE, TDMA_SLAVES. Run /sbin/rtnet start. Run rtping <remote-host> or load an application module.

Round Trip Time (RTT) is used to check the performance of RTnet. The command of rtping is executed to compute RTT. The test results with different data size are shown in table 3. All results are in microseconds.

TABLE III. RTT WITH RTNET

Byte	Min	Avg	Max
50	89	96	113
100	90	105	117
200	115	125	134
400	147	161	183
800	224	235	283
1000	263	270	278
1460	344	353	364

The information from the table above shows that real-time network requirement can be satisfied with RTnet.

V. CONCLUSIONS

In this paper Xenomai and RTnet based on embedded Linux are introduced. They are adaptable and extensible frameworks for real-time network over standard Ethernet. We extensively evaluated the performance of them. The test results show that this scheme can completely satisfy the requirement of real-time network. We believe this work is important to the implementation of distributed real-time systems, fieldbus coupling devices, low-cost real-time network analysers, etc [10]. Future work will further optimize network performance using advanced interrupt handling techniques.

REFERENCES

[1] AMCC Applied Micro Circuits Corporation, "PPC440GX Embedded Processor User's Manual", Revision 1.04-October 3, 2005

[2] Apostolos N. Meliones, Stergios D. Spanos, "Performance Analysis of Embedded Linux ATM for MPC8260 and Derivatives", Computers and Communications, in: Proceedings 11th IEEE Symposium, 26-29 June 2006 , pp.101 – 108

[3] A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, C. Taliercio, "Performance comparison of VxWorks, Linux, RTAI and Xenomai in a hard real-time application", IEEE-Trans. Nucl. Sc., submitted for publication.

[4] Karim Yaghmour, Building Embedded Linux Systems, O'Reilly & Associates, Inc, Sebastopol, 2003

[5] Philipper Gerum, "The Xenomai Project-Implementing a RTOS emulation framework on GNU/-Linux", White Paper, France, April 2004

[6] Philipper Gerum, "The Xenomai Project-Implementing a RTOS emulation framework on GNU/-Linux", White Paper, France, April 2004

[7] Hanssen, F, Jansen, P.G, Scholten, H, and Mullender, S, "RTnet: a distributed real-time protocol for broadcast-capable networks", Autonomic and Autonomous Systems and International Conference on Networking and Services, Joint International Conference, 23-28 Oct. 2005 , pp.18

[8] J. Kiszka, B.Wagner, Y. Zhang, J.F. Broenink, "RTnet—a flexible hard real-time networking framework", in: Proceedings of the 10th

IEEE International Conference on Emerging Technologies and Factory Automation, Catania, Italy, 2005.

[9] RTnet Home Page [Online], http://www.rtnet.org.

[10] R. Felton, K. Blackler, S. Dorling, O. Hemming, “Real-time plasma control at JET using an ATM Network”, in: Proceedings of the 11th IEEE-NPSS Real Time Conference, Santa Fe NM, USA, 1999.