Indian Journal of Science and Technology, *Vol 9(17), DOI: 10.17485/ijst/2016/v9i17/93109, May 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Implementation of Xenomai Framework in GNU/Linux Environment to Run Applications in a Real Time Environment

K. Sripath Roy[1]* and K. Gowthami[2]

[1]Department of ECE, KL University, Vaddeswaram, Guntur - 522502, Andhra Pradesh, India;
koganti_sripathroy@kluniversity.in
[2]Department of ECM, KL University, Vaddeswaram, Guntur - 522502, Andhra Pradesh, India;
konagowthami93@gmail.com

Abstract

Objectives: The aim of this paper is to portray the achievement of new Linux conveyance taking into account Debian Linux and Xenomai Real-Time structure. **Methods/Analysis:** This acknowledgment is instigated through the esteemed requirement of the real-time systems in recent software technologies. The fundamental objective of such conveyance is to suggest classic Operating Systems (OS) which incorporate Xenomai base. **Findings:** Xenomai is a real-time evolution framework co-operating with GNU/Linux operating system providing a ubiquitous, device-agnostic, hard real-time support to user-area applications coherently integrating into GNU/Linux operating system environment. Generally the Xenomai technology first aims at aiding application designers depending on traditional RTOS to move effortlessly to a GNU/Linux-based execution environment, without rewriting their application entirely at no cost. **Improvements/Applications:** Porting of Xenomai to Debian based Operating System and evaluation of performance parameters like clock test, latency on a General Purpose Computer.

Keywords: Debian, GNU/Linux, Porting, Real-Time Systems, RTOS, Xenomai

1. Introduction

Real-time embedded software has been playing a vital part in the data innovation market. In the RTOS market, there have been some prevalent performers with industry-endorsed benchmarks. Here, we aid to the expanding significance of real-time Linux developments as each of it proposes a group of advantages. Xenomai real-time developments have the major circumstances to duplicate standard RTOS interfaces, consistent to non-real-time Linux. Such reception can be worth-while to the general framework. This paper demonstrates the significance of using Xenomai along with Debian for continuous working frameworks and on-going applications model[1].

A simpler migration path from conventional RTOS to GNU/Linux can favour a broad compliance of the latter as a real-time embedded platform. Taking into account the determinative similarities between conventional RTOS, Xenomai innovation directs to an industrious design non-partisan and non-specific emulation layer taking favourable circumstances from these comparisons. This emulation might direct to reduce the disparity between exceptionally divided RTOS world along with GNU/LINUX world. Xenomai is about setting aside a few minutes working framework APIs accessible to Linux-based platforms. When the objective Linux kernel can't meet the necessities with respect to the timing requirements, Xenomai can likewise supplement it for conveying stringent real-time guarantees based on authentic co-kernel technology[2].

**Author for correspondence*

Xenomai helps in:

- Constructing, creating and running a real-time application on Linux.
- Migration of an application from a conventional RTOS Environment to Linux.
- Ideally running RTOS applications (VxWorks, pSOS, VRTX, uITRON, POSIX) nearby local Linux applications.

The main objective is to give real-time abilities in view of some functionality traded by a theoretical RTOS core. In generic, Xenomai targets on resilience, adaptability and practicality rather than attempting to accomplish most reduced in fact plausible latencies like RTAI does.

2. Xenomai Technology and Adeos

2.1 Xenomai Technology

2.1.1 Standards

While discussing the standards of Xenomai it is decisive to have precise view of the architecture. Figure 1 demonstrates a reliable analysis about the architecture of complete structure. It resides of assorted distinct conceptual layers, which presents an excessive resilience[3].

On top of hardware Adeos nano kernel is working. The Hardware Abstraction Layer (HAL) above Adeos is the architecture-reliant part of Xenomai. While porting Xenomai to be runnable on another hardware platform, the HAL has to be acclimated, so that Adeos is made accessible to new design. The significant factor of the system is the complex RTOS kernel running adjacent to HAL. It agents a group of collective RTOS assistances, accepted by most of the systems. The benefits can either be approached through a Xenomai inherent API or by means of distinct RTOS-API's build adjacent to nucleus. These further added API skins for conventional RTOS make it possible to run traditional operating system above the Xenomai core[4]. Xenomai facilitates users to process their applications in Kernel as well as in User-Space. Running functions in the user-space persuades the reliability of the system.

2.1.2 Features

The following are the important features of the Xenomai-nucleus.

- Multiple processes support:
- Protective scheduling methods.
- 32-bit accumulation preferences.
- Round-robin scheduling processes with equal preference.
- Watchdogs for each thread.
- Five-dimensional state imitation for threads.
- Abutment for superior inheritance.
- Primitive synchrony backing:
- Supports precedence inheritance, application which is communal to scheduler code.
- Supports time-constrained delay and coercive deletion along waiter's activation.
- Timer and clock administration:
- Periodic and aperiodic conditions.
- Nucleus obviously switches from periodic jiffies to time-stamp conflicting values relying against timer performing mechanism.
- Constrained inferior-case time to start, stop and control timers.
- Basic allocation of memory:
- Dynamic memory allocation with real-time guarantees.

2.2 ADEOS

The Adaptive Domain Environment for Operating Systems (ADEOS) is a nano kernel HAL that function among computer hardware and operating system that runs on it. It contributes to an extendable and robust environment which allows the sharing of hardware resources amid numerous operating systems. It is a resource constructive layer available as Linux kernel patch. Figure 2 describes that it grants several distinct domains to exist together on same hardware. The domains do not see each other by themselves, but every domain sees Adeos. The domains contend each other for the altering of external and internal events.

2.2.1 Architecture

Adeos implements a sequence of signals. At each instance when a peripheral sends a signal, the disparate operating systems running in the machine are excited which in turn decides whether to handle, neglect, abandon, or abort the signal[5]. Signals that are discarded by an OS are passed to next OS in the chain. Terminated signals are not propagated to latter satges.

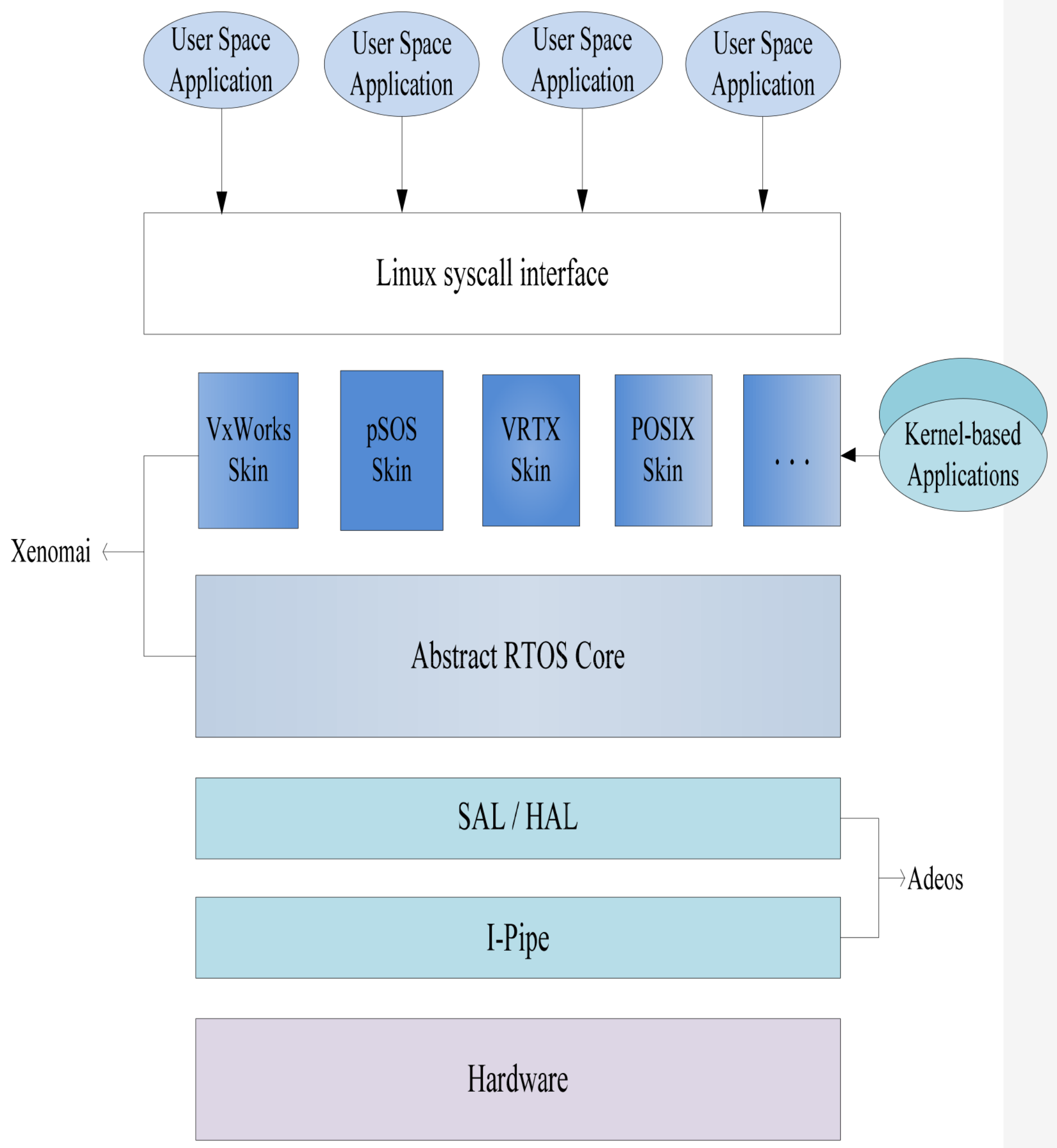


Figure 1. Xenomai architecture.

2.2.2 Adeos Interrupt Pipe

Another basic feature of Adeos is its capability to export a common API to client domains, which is autonomous of underlying CPU architecture. Each domain is adhered to a central data structure called "event-pipeline" or "I-Pipe", which offers the possibility to inform the domains for external interrupts, system calls pointed by Linux or other system events generated by kernel code as demonstrated in Figure 3. In order to dispatch external functions in a computed aspect, Adeos proposes the feasibility of halting the events. The I-Pipe propagates interrupts over distinct domains running on hardware[6]. As some domains choose to be the first to accept hardware interrupts, Adeos contributes the domains to have approach to priority interrupt dispatching.

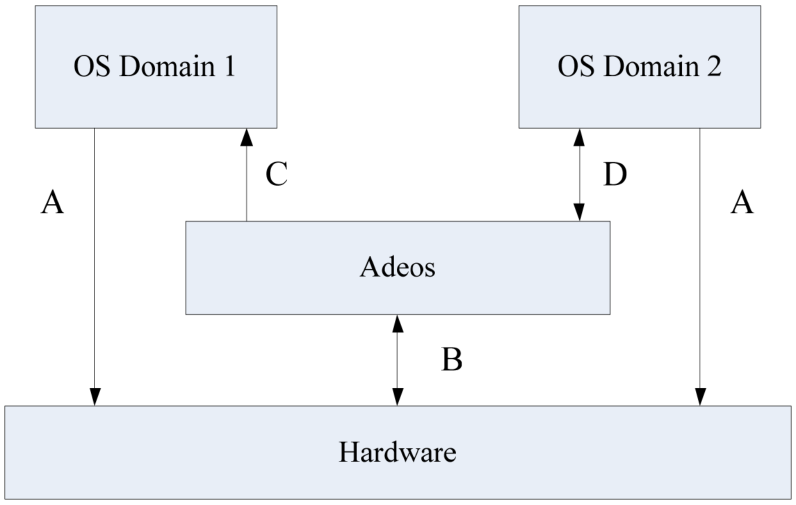


Figure 2. Adeos architecture.

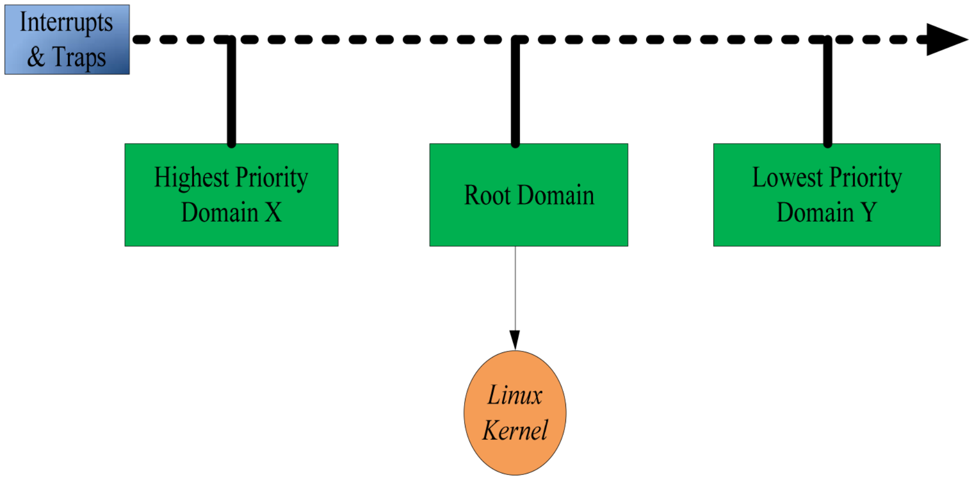


Figure 3. Adeos I-PIPE.

3. Migration from Xenomai 2 to Xenomai 3

The following are the issues considered when migrating from Xenomai 2 to 3.

- Xenomai is about RTOS APIs:
- Dual kernel is not by any means the only approach to real-time.
- Even real-time may not be required in some conditions.
- Kernel space is antagonistic to external APIs:
- Debugging is complicated.
- Extending APIs inflates the kernel.
- Kernel space is not a location for applications:
- This may bring about inaccurate software designs.

As the continuous PREEMPT_RT exertion conveys on the short and limited latency guarantee with a single kernel configuration on preferred hardware platforms for which this technology is accessible and sophisticated, it establishes opportunities to continue the applicability of Xenomai as a relocation tool, so that moving an application to such a system does not certainly involve porting the code over the POSIX API.

Xenomai 3 confiscated this opportunity, by enabling Xenomai APIs for dual kernel and inherent Linux configurations.

3.1 Xenomai 3 Architecture

Xenomai 3 is the new architecture which can run flawlessly side-by-side Linux as a co-kernel system like Xenomai 2, or inherently over mainline Linux kernels.

This new architecture presents two real-time cores chosen at build time.

3.1.1 Xenomai 3 Dual Kernel Configuration: Cobalt

This dual kernel named ***Cobalt*** is assembled into the Linux kernel as shown in Figure 4, handling all time-detracting activities like handling interrupts and setting up real-time threads. The Cobalt core has the greater priority over the inherent kernel activities. It is a significant modification of the Xenomai 2.x system. Cobalt implements the RTDM specification for integrating with real-time device drivers.

3.1.2 Xenomai 3 Single Kernel Configuration

The native Linux version, an embellished implementation of the experimental Xenomai work, is called *Mercury*. In this environment only a standalone employment of the RTDM specification in a kernel module is necessary, for integrating the RTDM compliant device drivers with native kernel as depicted in Figure 5.

This works with the addition of *Copperplate* interface, which arbitrates between the real-time API/emulator your application employs and the latent real-time core.

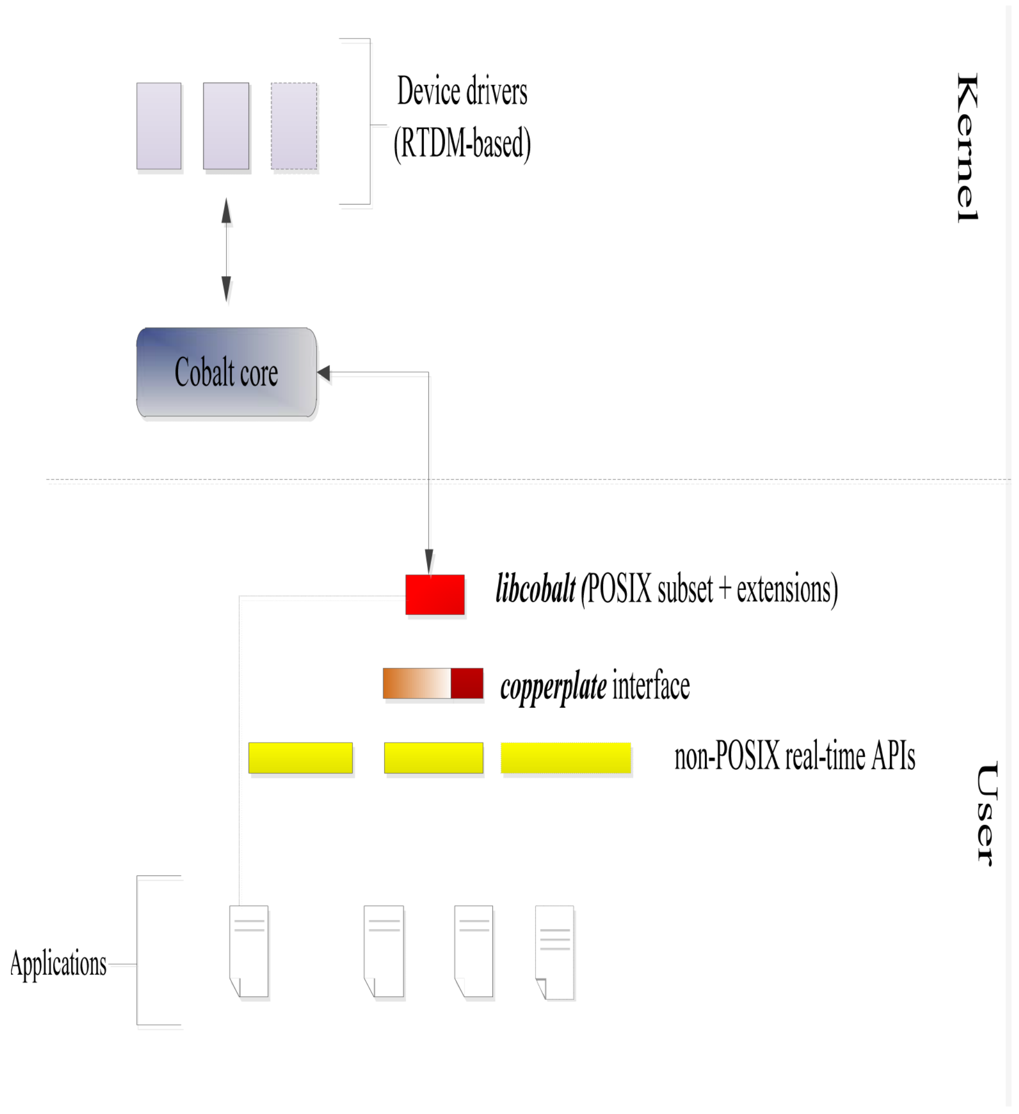


Figure 4. Xenomai 3 dual kernel.

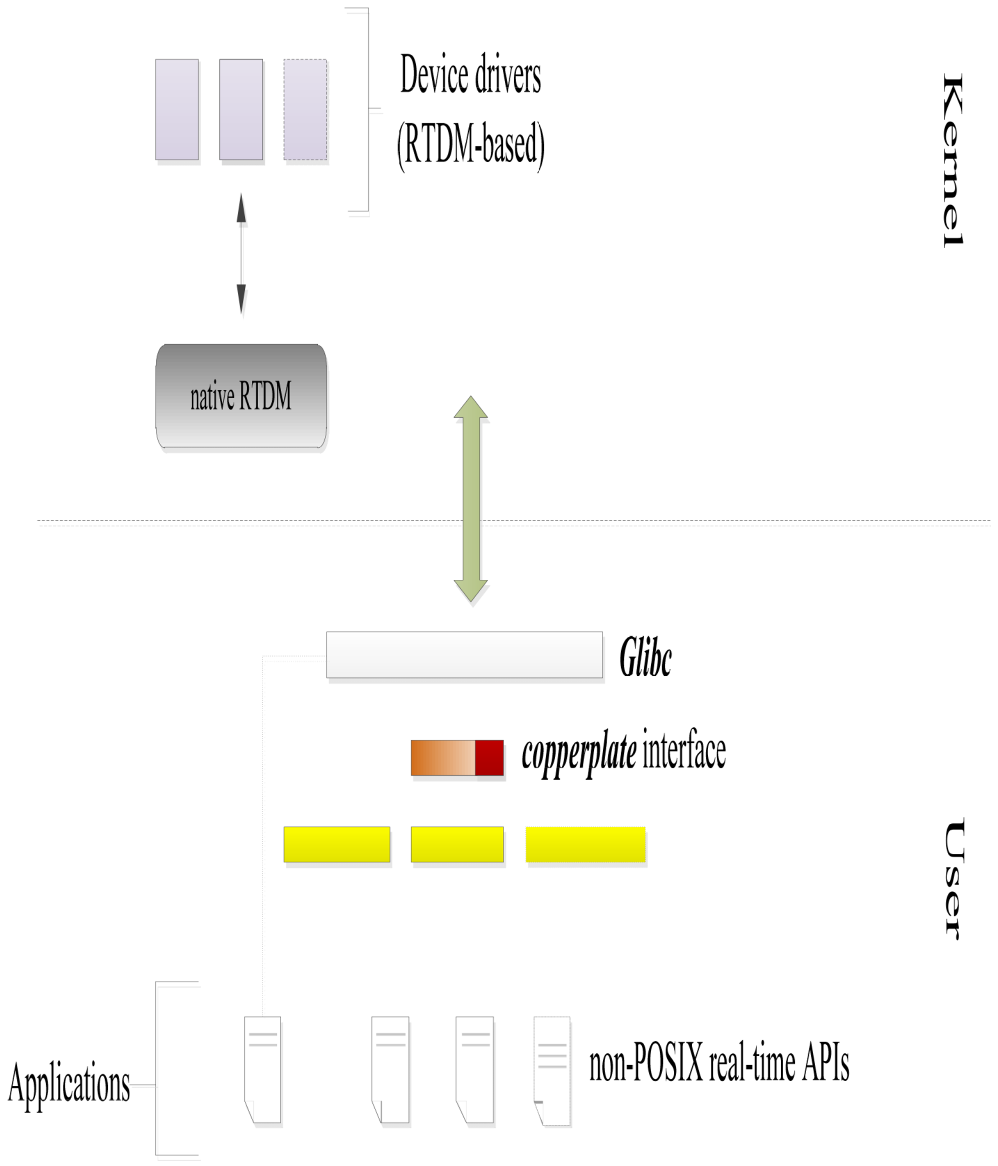


Figure 5. Xenomai 3 single kernel configuration.

3.1.3 Advantages of Xenomai 3 Compared to Xenomai 2

- Non-POSIX APIs available in native kernel configuration which can run over a single or dual kernel configuration inadequately and the same applies for applications built over them.
- Data transfer between real-time threads happens directly from user-space, even in multi-process applications.

4. Building Xenomai 3 on x86 Kernel

Xenomai follows a split source model by decoupling the kernel space abutment from user-space libraries. For this, kernel and client-space Xenomai segments are individually accessible under the kernel/ and lib/sub-trees. The kernel which executes the in-kernel bolster code is seen as an implicit expansion of Linux kernel. Cobalt kernel is prepared by building it as a component of destination kernel. To build it on an x86 kernel, we have to configure the kernel using *xconfig/menuconfig* and then build using [ARCH=i386] *bzImage modules.*

4.1 Essentials: Non-Specific Prerequisites (Both Cores)

- GCC must have bolster for legacy nuclear builtins (__sync structure).
- GCC ought to have a (rational/working) support for TLS ideally, in spite of the fact that this is not compulsory if working with – *disable-tls.*
- On the off chance that you plan to empower the client space registry support, then CONFIG_FUSE_FS must be empowered in the objective part running the constant applications.

4.2 Cobalt-Precise Essentials

- The kernel variant should be 3.10 or better.
- An Intrude on Pipeline (I-pipe) patch must be accessible for your destination kernel. Exclusive patches from the I-pipe-center arrangement are suitable, legacy patches from the adeos-ipipe arrangement are most certainly not.
- A Timestamp Counter (TSC) is required from running on a x86_32 equipment. Distinct with Xenomai 2.x, TSC-copying utilizing a PIT register is not accessible[7].

5. Results

Various test cases are run using this cobalt core on x86 architecture on Single CPU with 256 MB RAMS like clocktest, latency[8].

5.1 Clocktest

Clocktest is a measure of Xenomai series. For every CPU, it often produces a time offset, an accumulated value, the total warps and utmost warp with respect to microseconds. In this test, we have 3 cases i.e., real-time, monotonic, host_real-time. Figure 6 presents the output of clock test.

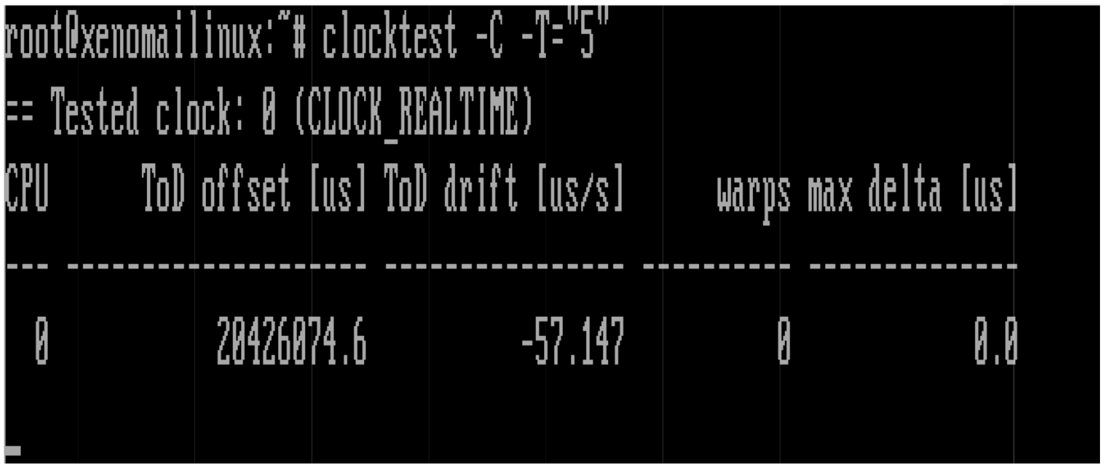
Default values:
Clock_Realtime = 0, Clock_monotonic = 1, Clock_Host_Realtime = 42.

5.2 Latency Test

Latency is a timer reference program. In this test we have several options to print histograms of latencies, dump histogram, to know test duration, data lines per header, etc. Figure 7 presents the output of latency test.

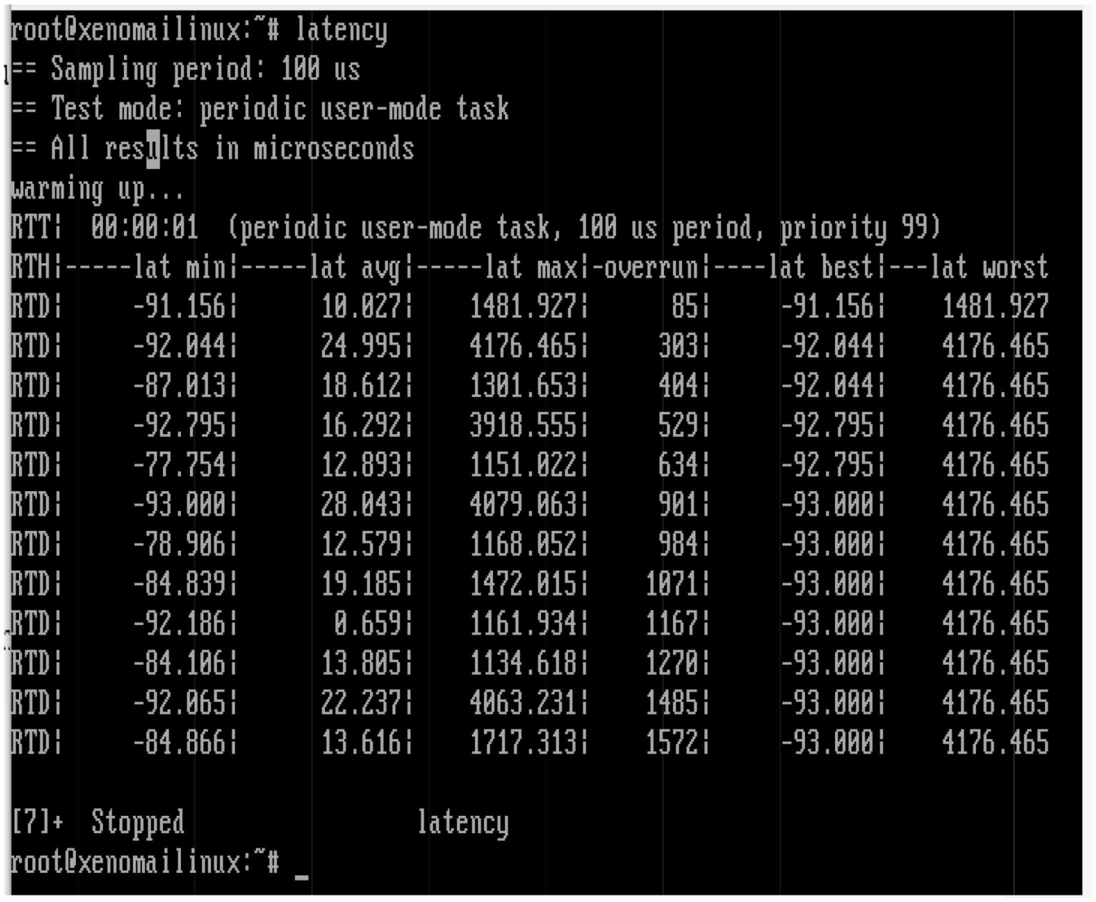
6. Conclusion and Future Work

In this paper, we have discussed the various features of Xenomai which is integrated with Linux in order to meet the requirements with respect to response time



```
root@xenomailinux:~# clocktest -C -T="5"
== Tested clock: 0 (CLOCK_REALTIME)
CPU      ToD offset [us] ToD drift [us/s]      warps max delta [us]
--- -------------------- ---------------- ---------- --------------
  0         20426074.6          -57.147          0            0.0
```

Figure 6. Clock-test of Xenomai.



```
root@xenomailinux:~# latency
== Sampling period: 100 us
== Test mode: periodic user-mode task
== All results in microseconds
warming up...
RTT|  00:00:01  (periodic user-mode task, 100 us period, priority 99)
RTH|-----lat min|-----lat avg|-----lat max|-overrun|----lat best|---lat worst
RTD|     -91.156|      10.027|    1481.927|      85|     -91.156|    1481.927
RTD|     -92.044|      24.995|    4176.465|     303|     -92.044|    4176.465
RTD|     -87.013|      18.612|    1301.653|     404|     -92.044|    4176.465
RTD|     -92.795|      16.292|    3918.555|     529|     -92.795|    4176.465
RTD|     -77.754|      12.893|    1151.022|     634|     -92.795|    4176.465
RTD|     -93.000|      28.043|    4079.063|     901|     -93.000|    4176.465
RTD|     -78.906|      12.579|    1168.052|     984|     -93.000|    4176.465
RTD|     -84.839|      19.185|    1472.015|    1071|     -93.000|    4176.465
RTD|     -92.186|       0.659|    1161.934|    1167|     -93.000|    4176.465
RTD|     -84.106|      13.805|    1134.618|    1270|     -93.000|    4176.465
RTD|     -92.065|      22.237|    4063.231|    1485|     -93.000|    4176.465
RTD|     -84.866|      13.616|    1717.313|    1572|     -93.000|    4176.465

[7]+  Stopped                 latency
root@xenomailinux:~# _
```

Figure 7. Latency test of Xenomai.

constraints[9,10]. And also Xenomai can supplement it to deliver inflexible real-time guarantees. Xenomai extends abutment for hard real-time applications due to which it can be used in several applications. And also Xenomai is built on beaglebone board to carry out some test cases. These tests are further used to run hard-real-time applications.

7. References

1. Xenomai implementing a RTOS emulation framework on GNU/Linux. Available from: https://xenomai.org/documentation/xenomai-2.0/pdf/xenomai.pdf
2. Start here. Introduction to Xenomai. Available from: https://xenomai.org/start-here/
3. Lal SV, Palaniappan R, Prakash V. Real time nursing management system for health care industry by using xenomai kernel. Indian Journal of Science and Technology. 2015 Aug; 8(20):1–10.
4. A quantitative comparison of realtime linux solutions. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.136.4601&rep=rep1&type=pdf
5. Life with Adeos. Available from: http://www.xenomai.org/documentation/branches/v2.0.x
6. Adaptive domain environment for operating systems; 2001. Available from: http://www.opersys.com/ftp/pub/Adeos/adeos.pdf
7. Migrating from Xenomai 2.x to 3.x. Available from: https://xenomai.org/introducing-xenomai-3/
8. Xenomai API reference. Available from: https://xenomai.org/api-reference.
9. Murikipudi A, Prakash V, Vigneswaran T. Performance analysis of real time operating system with general purpose operating system for mobile robotic system. Indian Journal of Science and Technology. 2015 Aug; 8(18):1–6.
10. Marieska MD, Kistijantoro AI, Subair M. Analysis and benchmarking performance of real time patch linux and xenomai in serving a real time application. 2011 International Conference on Electrical Engineering and Informatics (ICEEI); Bandung. 2011 Jul. p. 1–6.