

7.5 Performance of DMA ring on real-time platform

RTAI version 3.1 with LXRT on Linux 2.4.24 is chosen as the real-time platform. This platform guarantees bounded task response jitter, therefore a small DMA buffer size of 64 was sufficient to tackle highest packet rate and yet manifest the best robustness. Both, "DMA ring on LXRT with PIT" and "DMA ring on LXRT with RTC" employs a DMA buffer size of 64 bytes. These two architecture manifest all the superior performances figures as "DMA ring on Redhat 8", like no packet loss, low CPU utilization and in addition it yields very low packet delivery latency jitter. The next sub-sections also demonstrate that reduction of memory utilization and packet delivery latency jitter is achieved without any extra cost. As there is no packet loss so that profile is skipped.

7.5.1 CPU utilization profile

Fig. 7.21 and 7.22 presents CPU utilization for "DMA ring on Redhat 8", "DMA ring on LXRT with PIT (8254) timer" and "DMA ring on LXRT with RTC timer". Fig. 7.22 presents the same plot for the packet rate range below 35 kpps. Smaller 64 byte packets allow highest possible packet rates, hence expose worst case behaviors.

Fig. 7.21: CPU utilization profile of DMA ring architectures
(64 Byte UDP packets)

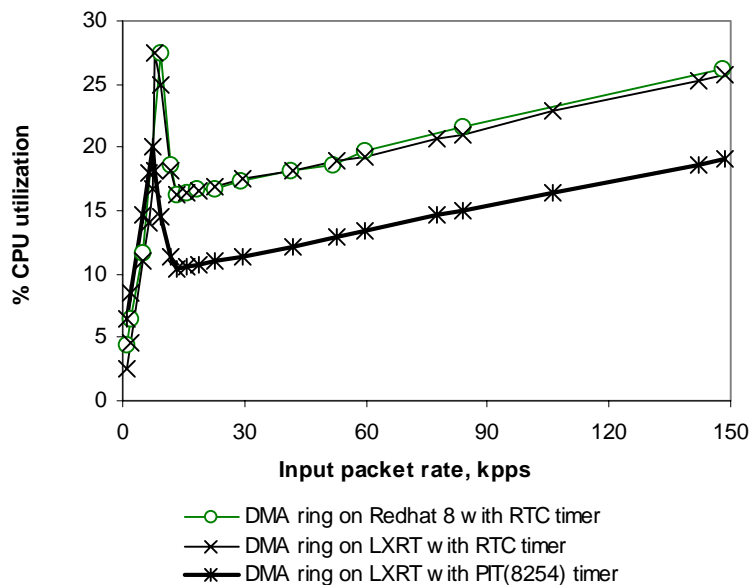
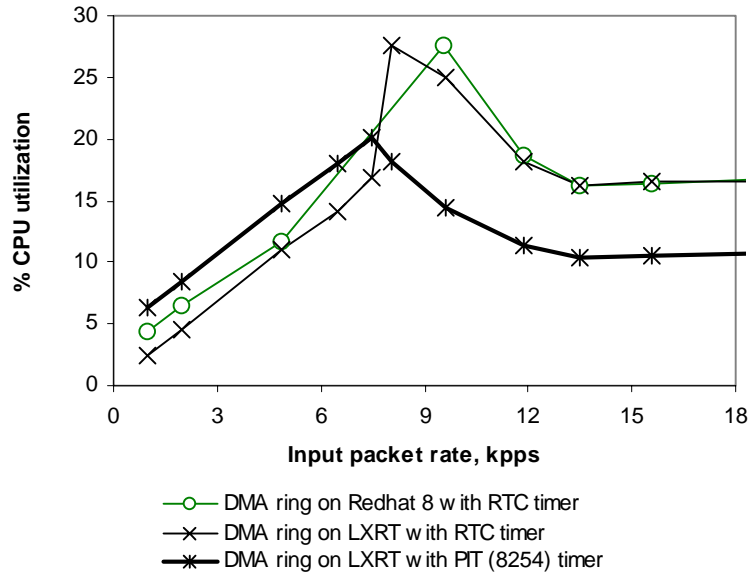


Fig. 7.22: CPU utilization profile of DMA ring architectures at lower packet rates
(64 Byte UDP packets)



The shapes of the CPU utilization profiles of DMA ring on LXRT are similar to that on Redhat 8. At lower packet rates when the DMA ring operates in interrupt mode, the CPU utilization of "DMA ring on LXRT with PIT (8254) timer" is slightly higher than other two architectures. This is because the PIT timer is operational as soon as the system starts, even when the system is not operating in polling mode. Some CPU resource is consumed by the RTAI co-kernel to service these PIT timer interrupts. Interrupt rate of PIT timer is set at 8.192 kHz which is sufficiently high to cause additional 3 to 4 % CPU utilization. In other two architectures, "DMA ring on LXRT with RTC timer" and "DMA ring on Redhat 8", the RTC timer, do not start till the packet rate cross the 8.333 kpps threshold. These architectures do not implement the PIT timer running at 122 microsecond periodicity, hence they do not have additional CPU utilization due to high frequency PIT timer interrupts. That explains 3 to 4 % higher CPU consumption in case of "DMA ring on LXRT with PIT (8254) timer" compared to other two architectures at range below 8 kpps.

On the other hand RTC timer in case of "DMA ring on Redhat 8" and "DMA ring on LXRT with RTC timer" starts beyond 8.333 kpps packet rate and cause additional 5 to 6% CPU consumption due to RTC interrupt servicing. From these figures it is evident that "DMA ring on LXRT with PIT (8254) timer" is the most efficient one when the entire operation range is considered. This establish that RTAI kernel timer is a better implementation choice compared to RTC timer. At all packet rates the "DMA ring on LXRT with PIT (8254) timer" consumes less than 17% of the CPU resources.

"DMA ring on Redhat 8" and "DMA ring on LXRT with RTC timer" are very similar, both uses RTC timer interrupts to pace the polling. Comparing the CPU utilization of these two, it can be concluded that CPU utilization does not deteriorate on using RTAI-LXRT over Linux.

7.5.2 Packet delivery latency profile

Table 7.9 compares the estimated average, 98th percentile and worst case statistics for packet delivery latencies of various DMA ring architectures at different packet rates (10⁶ samples).

Table 7.9: Packet delivery latency profile for DMA ring architecture
(for 64 Byte packets, figures in µsec)

Rank	Architecture	Statistics	2kpps	5kpps	18.5kpps	59kpps	148kpps
1.	DMA ring on LXRT with RTC timer	Avg.	14.4	14.4	61.4	61.4	61.4
		98 th	19.2	22.4	136.1	136.1	136.1
		Worst case	44.8	44.8	172.5	172.5	172.5
2.	DMA ring on LXRT with PIT(8254) timer	Avg.	14.4	14.4	61.4	61.4	61.4
		98 th	19.2	22.4	141.1	141.1	141.1
		Worst case	41.6	44.8	182.5	182.5	182.5
3.	DMA ring on Redhat 8	Avg.	14	14	61	61	61
		98 th	25	21	131	136	136
		Worst case	50	123	1935	2345	2345

For LXRT architectures the packet delivery latencies during interrupt operations could be directly measured by reading the NIC counter in user space (section 6.2.3), as these latencies were bounded within 183 microsecond. The poll period were measured by the same method as in "DMA ring on Redhat 8".

LXRT improves the worst case behavior by bounding the jitter, therefore for both LXRT architectures the jitter is low. But nothing can be concluded with sufficient confidence, about which LXRT architecture has lower packet delivery latency.

An actual feel of the packet delivery latency statistics can be obtained by observing the frequency distribution histograms for packet delivery latency and polling period jitter for these architectures.

Fig. 7.23 presents the frequency distribution of the packet delivery latency measured at 5 kpps packet rate on LXRT on the same receiver hardware (PII 333Mhz, Dell desktop). This same distribution is applicable for both LXRT architectures. The worst case interrupt latency observed is 44.8 microsecond.

Fig. 7.23: Frequency distribution of packet delivery latency in LXRT
(122 sec timer period, for RTAI 3.1 with Linux 2.4.24 on PII 333 Mhz)

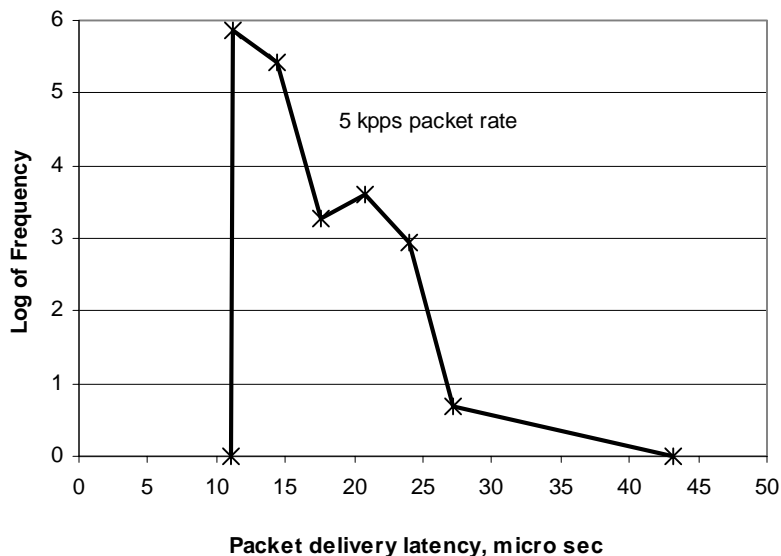


Fig. 7.24 presents the poll period of "DMA ring on LXRT with PIT(8254) timer" relative to that of "DMA ring on Redhat 8". The poll period for LXRT with PIT timer is bounded between 60 and 185 microseconds whereas for Redhat 8, it is distributed between 59 and 2345 microseconds.

Fig. 7.24: Frequency distribution of DMA ring poll period in Redhat 8 and LXRT

(for RTAI 3.1 with Linux 2.4.24 on PII 333 Mhz, Dell desktop)

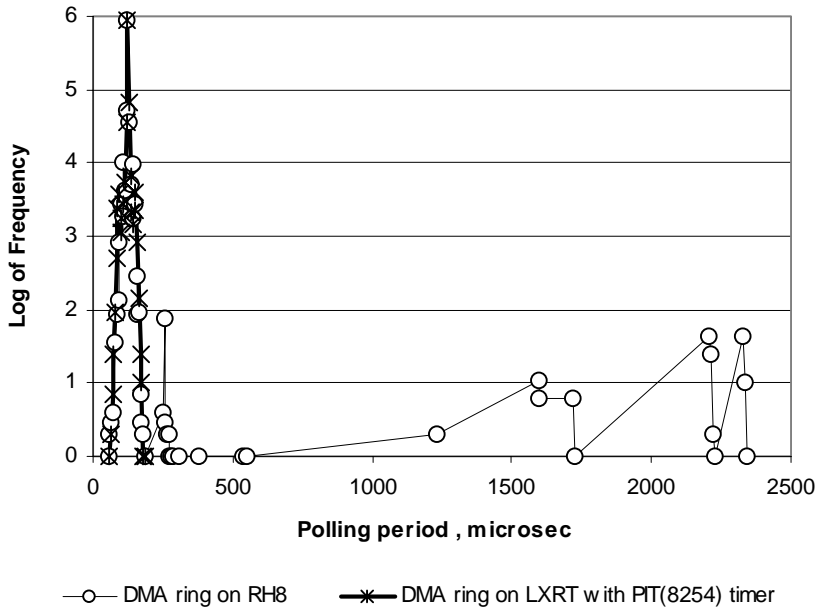
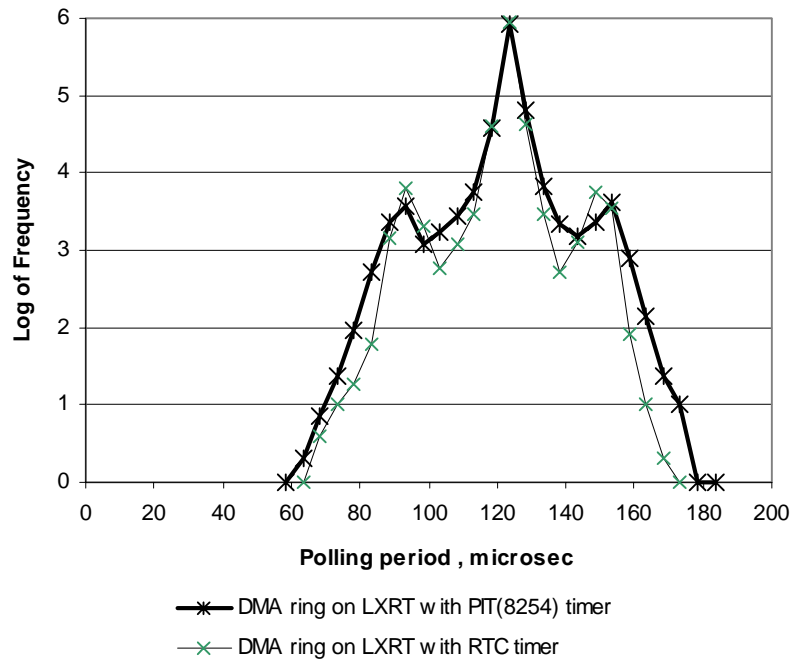


Fig. 7.25 compares the polling period distribution for "DMA ring on LXRT with PIT" with that of "DMA ring on LXRT with RTC" at 148 kpps (worst load scenario). Distribution for RTAI kernel timer based on PIT and RTC timer are very similar with worst case jitter of around 185 microseconds for both. The profile is distributed uniformly on both sides of the timer period at 122 microsecond. This comparison shows that in RTAI co-kernel for uni-processors, both schemes: polling paced by native RTAI timer (i.e. PIT in this case) and polling by an external hardware timer (i.e. the RTC timer in this case), yields similar advantage when jitter is concerned. This may not be the case

in RTAI co-kernels for SMP architectures, where native RTAI timers are differently implemented and may yield lower jitters than PIT and RTC.

Fig. 7.25: Frequency distribution of polling period paced by PIT and RTC timer in LXRT
(for RTAI 3.1 with Linux 2.4.24 on PII 333 Mhz, Dell desktop)



7.5.3 Memory requirement profile

On basis of memory utilization criteria, the architectures can be ranked as in Table 7.10 in decreasing order of performance i.e. on increasing order of memory resource utilization. Each packet buffer on the DMA ring in all the architectures were provisioned with 1536 Bytes (section 6.2.3). Both the LXRT architectures, i.e. "DMA ring on LXRT with PIT" and "DMA ring on LXRT with RTC" have same memory requirements, so both of them are represented by a single entry, "DMA ring on LXRT", in Table 7.10. DMA ring on LXRT does not require any socket buffer, hence with only 64 entries in the DMA buffer, it requires the least memory resource compared with Linux and NAPI.

Table 7.10: Memory utilization of DMA ring on LXRT and Redhat 8 compared with other low memory architectures

Rank	Architecture	DMA buffer size (Bytes)	Kernel packet queue and socket buffer size (Bytes)	Total memory requirement (Bytes)
1	DMA ring on LXRT	64*1536	-	98304
2	NAPI	128*1536	65536	262144
3	Linux	32*1536	300*1536 + 65536	575488
4	DMA ring on Redhat 8	2048*1536	-	3145728

7.5.4 Robustness

"DMA ring on LXRT with PIT timer" with DMA ring size of 64 can tolerate an additional load of more than 70%. With this load it pass all the robustness tests which "DMA ring on Redhat 8" passed, as depicted in section 7.2.5.

By using LXRT the DMA buffer size requirement is reduced, without sacrificing the robustness and other performance aspects. So it can be concluded that the apparent weakness of "DMA ring on Redhat 8" has been effectively addressed by LXRT.

7.6 Comparison of overheads in LXRT and Redhat 8

Average polling cycle invocation overhead for "DMA ring on LXRT with PIT" can be estimated from the dependency of CPU utilization on polling rate and packet rate, as done at the end of section 7.3.1. By regression, the dependency of CPU utilization on polling and packet rates is obtained as -

$$\eta = 8.128 * 10^{-6} * f_{Poll} + 0.746 * 10^{-6} * f_{PR} + 0.01 \dots\dots\dots \text{Eqn. 7.2}$$

where - η is the CPU utilization, presented as a fraction
 f_{Poll} is the polling frequency in Hz.
 f_{PR} is the incoming packet rate packets per second.

Comparing this relationship with Eqn. 5.4 of section 5.7, the average polling overhead is estimated as 8.128 microsecond. The average packet processing time is still the same 0.746 microsecond per packet as in Eqn. 7.1. The CPU utilization due to background tasks is only 1.0 % in LXRT whereas the same for Redhat 8 was 3.2 %. The polling overhead (8.128 microsecond) in LXRT is half of that in case of the Redhat 8 implementation (16.323 microsecond). This shows that RTAI's interrupt servicing and the `rt_sleep()` function call used to block and wait for the next polling period is more efficient than Linux's interrupt servicing, `ioctl()` and `wait_event_interruptible()` mechanisms. The CPU utilization due to back ground kernel tasks in LXRT is one third of that in case of Redhat 8. This may indicate that the RTAI scheduler is more efficient than the Redhat 8 Linux scheduler. So RTAI on Linux 2.4.24 may be a better choice in all respects compared to Redhat 8.

7.7 Summary

The key findings from the performance studies can be summarized as following -

- Packet loss in receiver is primarily due to buffer overflow.
- Low fixed rate polling yields better performance even though polling overhead may be quite high. This strategy amortizes polling overhead over many packets.
- Threshold based mode switching in a hybrid-interrupt-polling scheme allows optimization of the polling operation. Performance loss due to frequent mode switching is avoided.
- Reduced context switching, integrated minimal protocol processing, efficient border crossing, using a common staging area to avoid run time memory allocation and copy operation improves performance of a packet receiving architecture.

- The proposed "DMA ring" architecture has superior performance compared to existing solutions - NAPI, PFRING and Linux networking stack in terms of significantly lower CPU utilization, lower packet delivery latency and robustness (Table 7.11).

**Table 7.11: Superior performance of DMA ring:
Comparison with best of class solutions.**

Performance Measure	DMA ring on LXRT (PIT)	DMA ring on Redhat 8 (RTC)	Contemporary best of class
<i>No loss capacity</i>	> 148 kpps	> 148 kpps	PFRING 84 kpps
<i>Loss at 148 kpps</i>	0%	0%	PFRING + NAPI 29.40%
<i>CPU utilization</i>	< 17 %	< 28 %	All 100%
<i>CPU available for event Processing tasks</i>	> 70 %	70%	All 0%
<i>Packet delivery latency (micro sec) at 84 kpps</i>	Avg = 61 98th = 136 Worst case = 172	Avg = 61 98th = 136 Worst case = 2345	PFRING Avg = 240608 98th = 266081 Worst case = 267985
<i>Memory requirement (Bytes)</i>	98304	3145728	NAPI 262144
<i>Robustness: can handle continuous load over 1 billion packets</i>	Pass not a single loss	Pass not a single loss	No competing solution is available

- Minimizing protocol processing, efficient border crossing and polling can yield significant benefits at all packet rates. Shared DMA buffer render additional benefit only if the packet size is quite large.
- There is some cost associated with memory allocation, but it is not too high.
- If DMA ring is used on a GPOS like Redhat 8, then it requires more memory, however if it is implemented on a real-time platform, this weakness is addressed.

DMA ring on a real-time platform use the least amount of memory even less than half of what Linux networking stack requires.

- DMA ring on a real-time platform also bounds the worst case packet delivery latency to a small value.
- It is possible to optimize performance of DMA ring on a real-time platform by utilizing the native kernel timer to pace the polling instead of using external hardware timer interrupts. Choosing suspend-resume for task blocking and unblocking yields better DMA ring performance.
- Using a real-time platform with an additional kernel itself does not deteriorate the CPU performance of the system. There is no cost of implementing such real-time platform in the present problem context.
- In fact, the RTAI scheduler may be more efficient than Redhat 8 Linux scheduler.
- NAPI has the worst no loss packet capacity, it also deteriorates this performance aspect of "PFRING with NAPI" architecture.
- NAPI introduces high packet delivery jitters at high packet rates, it depreciates this performance aspect of the "PFRING with NAPI" architecture.
- NAPI suffer from livelock phenomena even though it was designed as a solution against it. This was because the NAPI polling is sustained only at very high packet rates, and because NAPI wastes much CPU resources due to frequent context switching, high polling overheads and kernel protocol stack inefficiencies.
- Even though NAPI was designed to employ low priority softirqd tasks to reduce CPU starvation of user space tasks, but user space tasks still starved. NAPI fails in both the design goals - livelock and user space starvation.

- In spite of these limitations of NAPI, it reduces packet losses in case of PFRING. PFRING with NAPI has far better throughput and lower packet loss compared to PFRING alone.
- PFRING collapses at very high packet rate.
- A user space mechanism can be better or as good as a kernel mechanism for network I/O processing if certain design pre-cautions are taken.