Multi Middleware Coexistence on Realtime Linux using TSN connectivity

TSN/A Conference 2022

Kurt Kanzenbach

Linutronix GmbH

28.09.2022

LINUTRONIX
LINUX FOR INDUSTRY

Agenda

1. Motivation
2. Multi Middleware
3. Evaluation
4. Conclusion

Motivation

Motivation

- Mainline Linux introduced real time networking and TSN functionalities
- Network stack utilizes:
  - Precision Time Protocol (PTP)
  - eXpress Data Path (XDP)
  - Queuing disciplines (Qdiscs), Filters, Steering
- How well do these perform for real world use cases?
- Possibilities and limitations of running multiple middlewares at the same time?

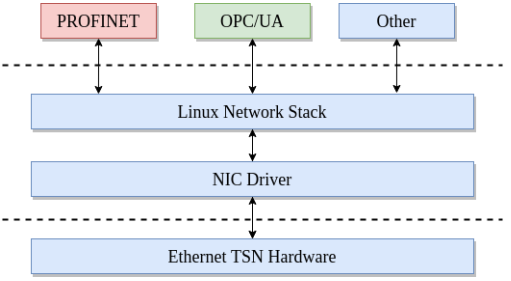


Figure: Multi Middleware

TSN Testbench

Objective

- **Aim is evaluation of manufacturer's hardware as well as underlying drivers and Linux network stack itself**
- **Deliver Ethernet TSN HW/driver evaluation application, not conformance testing**
- Independent of certain hardware/drivers by using standard mainline network and configuration interfaces
- Usable for regression testing in case of kernel updates
- Includes protocols such as PROFINET and OPC/UA PubSub

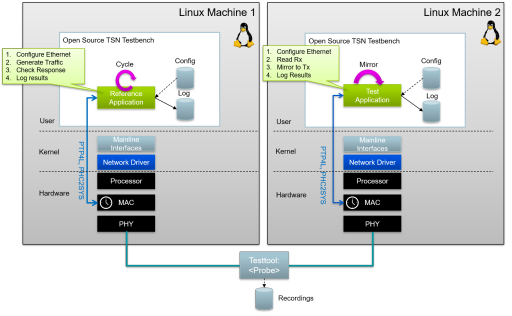


Figure: Testbench Overview

LINUTRONIX
LINUX FOR INDUSTRY

Agenda

1. Motivation
2. Multi Middleware
3. Evaluation
4. Conclusion

Challenges for Multi Middleware

Challenges for Multi Middleware

- **Addressing:**
  - Which Ethernet frame belongs to which application instance?
- **Sequencing:**
  - How to ensure correct frame sequence between different applications?
- **Network configuration:**
  - Who configures the network interface?
- **Resource separation:**
  - Can applications influence each other?

Approach

Approach taken

- Different TSN Testbench instances simulate different middlewares such as PROFINET and OPC/UA PubSub
- Challenges addressed:
  - Addressing by using unique tupels of { MAC DA, EtherType, VLAN ID }
  - Sequencing by either using Tx Launch Time or Scheduled Traffic or both
- Network configuration is out of scope. Other projects investigate this:
  - **Avnu detd:** `https://github.com/Avnu/detd/`
  - **ControlTSN:** `https://www.accesstsn.com/`

Testbench Architecture

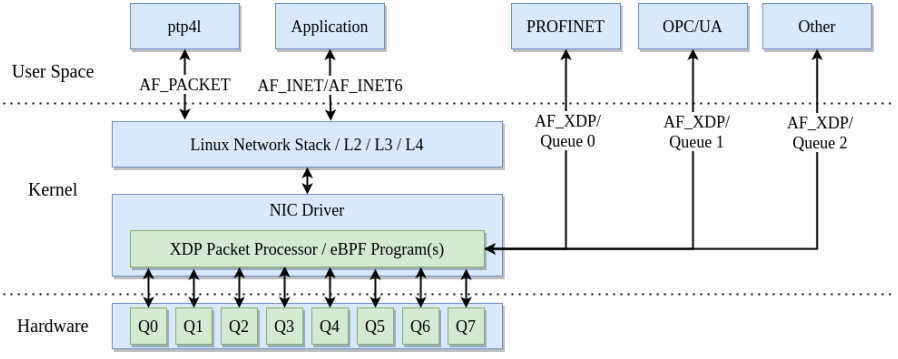


Figure: Testbench Architecture

Testbench Configuration

Testbench Configuration

- PROFINET or OPC/UA PubSub
- Cycle time and scheduling behavior
- Number and size of frames
- Thread options such as CPU pinning and priorities
- Socket options such as priorities or interface (XDP, RAW, ...)
- Logging and Debugging options

Sample OPC/UA configuration file snippet

```
Application:
  ApplicationClockId: CLOCK_TAI
  # ApplicationBaseStartTimeNS: 0
  ApplicationBaseCycleTimeNS: 500000
  ApplicationTxBaseOffsetNS: 400000
  ApplicationRxBaseOffsetNS: 100000
  ApplicationXdpProgram: xdp_kern_opcua_vid100.o
GenericL2:
  GenericL2Name: OpcUa
  GenericL2XdpEnabled: False
  GenericL2TxTimeEnabled: True
  GenericL2Vid: 100
  GenericL2Pcp: 6
  GenericL2EtherType: 0xb62c
  GenericL2NumFramesPerCycle: 32
  GenericL2PayloadPattern: |
    OpcUaPayloadPattern
  GenericL2FrameLength: 128
  GenericL2TxQueue: 7
  GenericL2SocketPriority: 7
  GenericL2Interface: eno1
```

Agenda

1. Motivation
2. Multi Middleware
3. Evaluation
4. Conclusion

Test Equipment

Test Equipment

- Two machines running Linux v5.15-LTS with PREEMPT_RT
- Intel TSN NICs: i225 and stmmac
- Fraunhofer IOSB INA *TSN Monitor* with Oscilloscope
- Network TAP (Profishark)

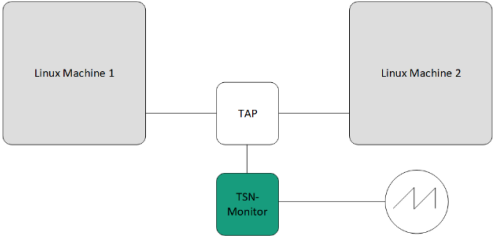


Figure: Test Setup

Functionality Tests

Functionality Tests

- Testbench constantly measures Ethernet Frame Round-Trip-Times
- Detects frame losses/damages and out-of-order scenarios
- *TSN Monitor* visualizes Ethernet data stream on wire

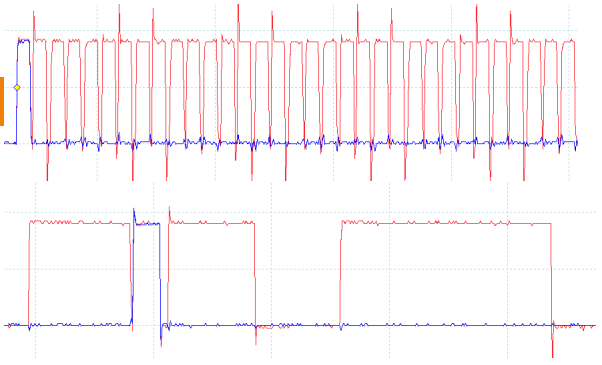


Figure: Interpacket Gap and Frame Preemption

Linux Sockets

- AF_XDP Sockets:
  - Memory management in application
  - Zero Copy available
  - Requires a single eBPF program per network interface
  - Limited by number of hardware queues and driver support
  - No support for Tx Launch Time
  - Performance example, 16 Frames per cycle:
    - Tx time: 22 µs
    - Rx time: 18 µs
- AF_PACKET Sockets:
  - Standard Linux socket
  - System call driven
  - Requires eBPF filters for multiple application instances
  - Not limited by queue usage
  - Tx Launch Time and packet scheduling in general usable
  - Performance example, 16 Frames per cycle:
    - Tx time: 107 µs
    - Rx time: 100 µs

Results regarding the Challenges

	AF_XDP Sockets	AF_PACKET Sockets
Addressing	eBPF XDP programs	eBPF filters
Sequencing	Scheduled Traffic	Tx Launch Time
Network configuration	Tricky	Less tricky
Resource Separation	TBD	TBD
Performance	Adequate	Poor
Queue usage	Limited by HW queues	Arbitrary

Table: Results

LINUTRONIX
LINUX FOR INDUSTRY

Agenda

1. Motivation
2. Multi Middleware
3. Evaluation
4. Conclusion

Outlook and Next Steps

Outlook and Next Steps

- Get feedback from community
- Release TSN Testbench as Open Source under permissive license
- Host required
- Perform additional testing in terms of resource separation and security aspects
- Plugin to network interface configuration

Current Linux limitations

Current Linux limitations

- XDP is bound to limitations such as hardware queue usage
- XDP bypasses all Linux scheduling mechanisms
- Orchestration of multiple applications is tricky
- Possible optimizations for Linux with PREEMPT_RT:
  - Soft interrupt handling
  - AF_XDP busy polling

Conclusion

Conclusion

- Running multiple middlewares is possible
- There are limitations in today's Linux ecosystem and TSN hardware, which can and should be improved in upcoming versions
- XDP is recommended for low cycles times and high throughput, but has limitations in multi middleware scenarios
- High demand for unified and central network interface configuration for end devices
- There will be a Open Source Testbench to evaluate and optimize your own TSN HW and drivers. Use and contribute to it!

Thank you for your attention.

Linutronix GmbH

kurt.kanzenbach@linutronix.de
Bahnhofstraße 3
88690 Uhldingen-Mühlhofen

