Open Industrial Linux User Guide Release v0.2

Table of Contents

1 Introduction

1.1 About OpenIL

The OpenIL project (“Open Industry Linux”) is designed for embedded industrial usage. It is an integrated Linux distribution for industry.

OpenIL is built on buildroot (version 2016.11.2) project and provides packages for industry market.

- **Focus on industry:** OpenIL provides key components for industry usage, for example, Time sensitive network (TSN), Netconf, IEEE 1588 and Xenomai.
- **Ease of use:** OpenIL is a tool that simplifies and automates the process of building a complete Linux system for an embedded system, using cross-compilation. It follows the buildroot project rules. For more buildroot information, please refer to the page:

  https://buildroot.org/
- **Extensibility:** OpenIL provides capabilities of industry usage and standardized Linux system packages. And user can also easily replicate the same setup on customized packages and devices.
- **Lightweight:** OpenIL only includes necessary Linux packages and industry packages in order to make the system more lightweight to adapt industry usage. User can customize the package via configuration file
- Open Source: OpenIL is an open project. Anyone can participate the OpenIL development in Open Source community.

1.2 OpenIL Organization

OpenIL follows the Buildroot directory structure like below. The second and third lines of the directory are generated during compilation.

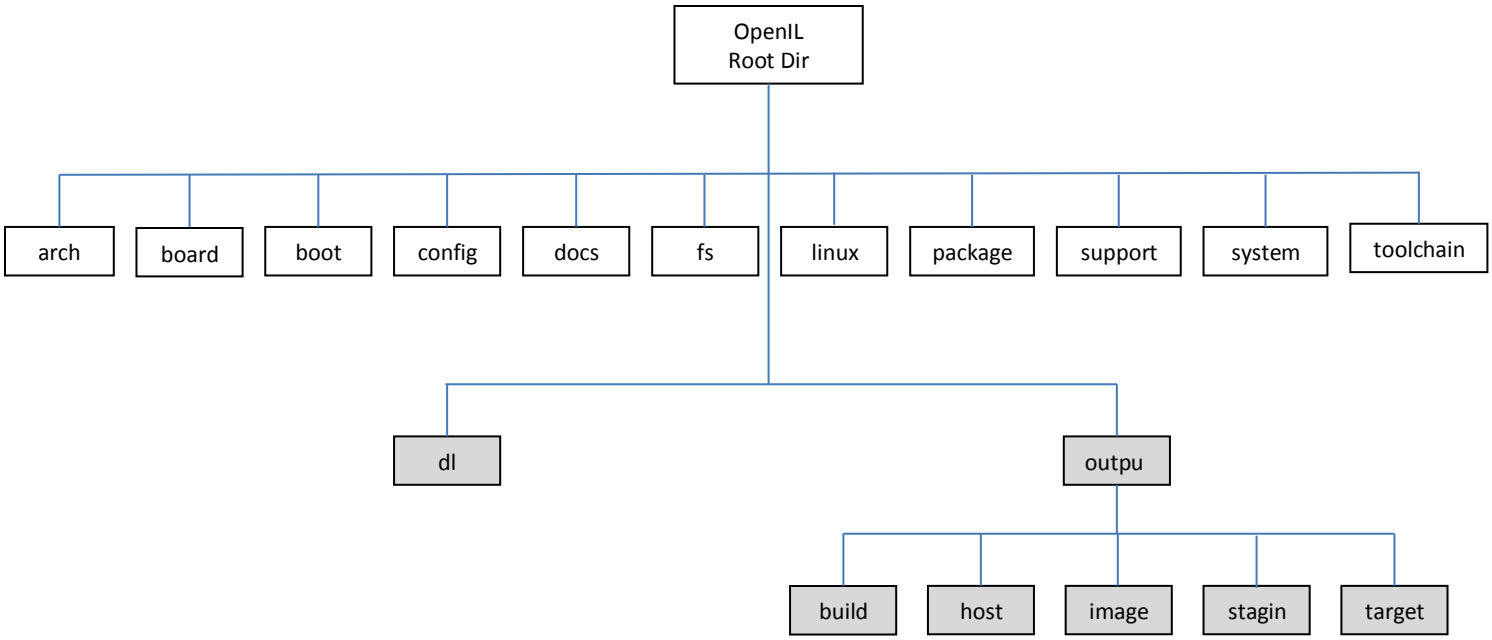

Figure 1-1 OpenIL Structure

Source directories:

- arch: files defining the architecture variants (processor type, ABI, floating point, etc.)
- toolchain: packages for generating or using toolchains.
- system: the rootfs skeleton and options for system-wide features.
- linux: the linux kernel package.
- package: all the user space packages (1800+).
- fs: logic to generate filesystem images in various formats.
- boot: bootloader packages.
- configs: default configuration files for various platforms.
- board: board-specific files (kernel configurations, patches, image flashing scripts, etc.)
- support: misc utilities (kconfig code, libtool patches, download helpers, and more.)
- docs: documentation.

Build directories:

- dl: where all the source tarballs are downloaded into.
- output: global output directory.
- output/build: where all source tarballs are extracted, the build of each package takes place.
- output/host: contains both the tools built for the host and the sysroot of the toolchain.
- output/staging: just a symbolic link to the sysroot, i.e. to host/<tuple>/sysroot/ for convenience.
- output/target: the target Linux root filesystem, used to generate the final root filesystem images.
- output/images: contains all the final images: kernel, bootloader, root filesystem and so on.

1.3 Host System Requirements

OpenIL is designed to build in Linux systems. And the following host environments have been verified to build the OpenIL.

- CentOS Linux 7 (Core)
- CentOS release 6.7 (Final)
- Ubuntu 16.10
- Ubuntu 16.04

1.4 Industrial Features

- Netconf/Yang
- Netopeer
- TSN
- IEEE 1588
- Xenomai Cobalt mode

1.5 Supported Platforms

Platform	Architecture	Config file in OpenIL	Boot
ls1021atsn	ARM v7	configs/nxp_ls1021atsn_defconfig	SD card
ls1043ardb (32-bit)	ARM v8	configs/nxp_ls1043ardb-32b_defconfig	SD card
ls1046ardb (32-bit)	ARM v8	configs/nxp_ls1046ardb-32b_defconfig	SD card
ls1012ardb (32-bit)	ARM v8	configs/nxp_ls1012ardb-32b_defconfig	QSPI

Table 1-1 Supported Platforms

1.6 Platform Default Compilation Settings

Platform	Toolchain	libc	Init system
ls1021atsn	gcc 5.x	glibc 2.23	BusyBox
ls1043ardb (32-bit)	gcc 5.x	glibc 2.23	BusyBox
ls1046ardb (32-bit)	gcc 5.x	glibc 2.23	BusyBox
ls1012ardb (32-bit)	gcc 5.x	glibc 2.23	BusyBox

Table 1-2 Default Setting

2 Getting Start

At the completion of this section, you should get the OpenIL source code, build and program the NXP platform images simply and run the OpenIL system on the platforms.

2.1 Getting OpenIL

OpenIL release is made every few months. Release number follow the format YYYYMM, for example 201705. Release tarball are available at:

https://github.com/openil/openil

To follow development, you can make a clone of the Git repository. Refer to the command:

```
$ git clone https://github.com/openil/openil.git
$ cd openil
# checkout to the v0.2 release tag
$ git checkout OpenIL-201705 -b OpenIL-201705
```

2.2 OpenIL Quick Start

The steps below build the NXP platform images with OpenIL quickly. And you should NOT ignore the important notes in the following section.

2.2.1 Important Notes

- Build everything as a normal user. There is no need to be root to configure and use OpenIL. By running all commands as a regular user, you protect your system against packages behaving badly during compile and installation.
- Not use make -jN to build OpenIL: top-level parallel make is currently not supported.
- The PERL_MM_OPT issue. you may encounter the error information of the PERL_MM_OPT when run make command in some host Linux environment like this:

```
You have PERL_MM_OPT defined because Perl local::lib
is installed on your system. Please unset this variable
before starting Buildroot, otherwise the compilation of
Perl related packages will fail
make[1]: *** [core-dependencies] Error 1
make: *** [_all] Error 2
```

To resolve it, just unset the PERL_MM_OPT.

```
$ unset PERL_MM_OPT
```

2.2.2 Building the Final Images

For the NXP platforms supported by OpenIL, the default configuration files can be found in directory “configs”.

Platform	Config file in OpenIL
ls1021atsn	configs/nxp_ls1021atsn_defconfig
ls1043ardb (32-bit)	configs/nxp_ls1043ardb-32b_defconfig
ls1046ardb (32-bit)	configs/nxp_ls1046ardb-32b_defconfig
ls1012ardb (32-bit)	configs/nxp_ls1012ardb-32b_defconfig

Table 2-1 Default Configuration

Those “configs/nxp_xxxx_defconfig” files include all the necessary U-Boot, Kernel configurations and application packages for filesystem. Based on the files without any changes, you can build a complete Linux environment for the target platforms.

To build the final images for one NXP platform (i.e. ls1021atsn), simply run:

```
$ cd openil
$ make nxp_ls1021atsn_defconfig
$ make
# or make with a log
$ make 2>&1 | tee build.log
```

Note: “make clean” should be implemented first before any other new compilation!

The “make” command will generally perform the following steps:

- Download source files (as required and just at the first time);
- Configure, build and install the cross-compilation toolchain;
- Configure, build and install selected target packages;
- Build a kernel image, if selected;
- Build a bootloader image, if selected;
- Create a root filesystem in selected formats.

After the correct compilation, you can find all the images for the platform at “output/images”.

```
images/
├── boot.vfat
├── ls1021a-tsn.dtb                --- dtb file for ls1021atsn
├── rootfs.ext2
├── rootfs.ext2.gz
├── rootfs.ext2.gz.uboot           --- ramdisk can be used for debug
├── rootfs.ext4.gz -> rootfs.ext2.gz
├── rootfs.tar
├── sdcard.img                     --- entire image can be programmed into the SD card
├── uboot-env.bin
├── u-boot-with-spl-pbl.bin        ---    uboot image for ls1021atsn
└── uImage                         ---    kernel image for ls1021atsn
```

For more detailed OpenIL configuration, you can refer to the Buildroot document.

https://buildroot.org/downloads/manual/manual.html#getting-buildroot

2.3 Bootup Board Quickly

Following steps to bootup the NXP platforms with the images built from OpenIL.

All the NXP platforms can be bootup from SD card or QSPI flash. After the compilation for one platform, sdcard.img or qspi.img will be generated in folder “output/images”.

Platform	Boot	Final Image	Board SW Setting
ls1021atsn	SD card	sdcard.img	SW2 = 0b’111111
ls1043ardb (32-bit)	SD card	sdcard.img	SW4[1-8] +SW5[1] = 0b'00100000_0
ls1046ardb (32-bit)	SD card	sdcard.img	SW5[1-8] +SW4[1] = 0b'00100000_0
ls1012ardb (32-bit)	QSPI	qspi.img	SW1 = 0b'10100110 SW2 = 0b'00000000

Table 2-2 Boards SW Setting

The flash image (sdcard.img/qspi.img) includes all the RCW, DTB, U-Boot, kernel, Rootfs, and necessary applications.

Note: make sure the board is set to bootup from SD card or QSPI by SW configuration. For all platform SW setting, please refer to the platform hardware setting chapter.

2.3.1 SD Card Bootup

For the platforms that can be bootup from SD card, just one step to program the sdcard.img.into SD card. Insert one SD card (at least 512M size) into any Linux host machine, then run:

```
$ sudo dd if=./sdcard.img of=/dev/sdx
# or in some host machine:
$ sudo dd if=./sdcard.img of=/dev/mmcblkx

# find the right SD Card device name in your host machine and replace the “sdx” or “mmcblkx”.
```

Then insert the SD card into the target board (switch the board boot from SD card first) and power on.

2.3.2 QSPI Bootup

For the platform which can be bootup from QSPI (ls1012ardb), following the below steps to program the qspi.img into QSPI flash.

Set the board boot from QSPI, then power on and enter the U-Boot command environment.

```
=>i2c mw 0x24 0x7 0xfc; i2c mw 0x24 0x3 0xf5
=>tftp 0x80000000 qspi.img
=>sf probe 0:0
=>sf erase 0x0 +$filesize
=>sf write 0x80000000 0x0 $filesize
=>reset
```

2.3.3 Startup the board

After the sdcard.img/qspi.img programming, startup the board and the following information should be printed out:

Figure 2-1 OpenIL System Startup

The system will be logged in automatically.

3 NXP OpenIL Platforms

3.1 Introduction

The OpenIL can support four NXP Layerscape ARM platforms: LS1012ARDB, LS1021ATSN, LS1043ARDB, LS1046ARDB. For more information about those platforms, please refer to the NXP website:

http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/qoriq-layerscape-arm-processors:QORIQ-ARM

Following the above chapter you can boot up the boards quickly with a complete SD card or QSPI image. This chapter describes how to deploy U-Boot, Linux kernel and root file system to the board . The guide starts with generic host and target board pre-requisites. This is followed by board-specifc configuration:

- Switch Settings
- U-Boot environment Variables
- Device Microcodes
- Reset Configuration Word (RCW)
- Flash Bank Usage

Note: This chapter is wrote to those who want to do more sub-system debugs, like U-Boot, kernel and so on. At the beginning, the board should can boot up and run in U-Boot command environment.

3.2 LS1021ATSN

The LS1021A Time-Sensitive Networking (TSN) Reference Design is a platform that allows developers to design solutions with the new IEEE Time-Sensitive Networking (TSN) standard. The board includes the QorIQ Layerscape LS1021A industrial applications processor and SJA1105T TSN switch. The LS1021ATSN is supported by an industrial Linux SDK with Xenomai real time Linux, which also provides utilities for configuring TSN on the SJA1105T.

With virtualization support, trust architecture secure platform, Gigabit Ethernet, SATA interface, and an Arduino Shield connector for multiple wireless modules, the LS1021ATSN is ready to support industrial IoT requirements.

3.2.1 Switch Settings

The following table lists and describes the switch configuration for LS1021ATSN Board.

Note: The OpenIL just support the SD card boot.

Platform	Boot Source	SW setting

LS1021ATSN	SD card	SW2 = 0b’111111

Table 3-1 LS1021ATSN SD Boot SW Setting

3.2.2 Target Images Update

To build the images for ls1021atsn platform:

```
$ cd openil
$ make nxp_ls1021atsn_defconfig
$ make
# or make with a log
$ make 2>&1 | tee build.log
```

- Programming U-Boot with RCW in SD card

  Power on the ls1021atsn board to U-Boot command environment, then:

```
=>tftp 81000000 u-boot-with-spl-pbl.bin
=>mmc erase 8 0x500
=>mmc write 0x81000000 8 0x500
#then reset the board
```

- Deploy kernel and Ramdisk from TFTP

  1. Set the U-Boot environment

```
=>setenv bootargs ‘root=/dev/ram0 rw ramdisk_size=40000000 console=ttyS0,115200’
=>saveenv
```

  2. Boot up the system

```
=>tftp 83000000 uImage
=>tftp 88000000 rootfs.ext2.gz.uboot
=>tftp 8f000000 ls1021a-tsn.dtb
=>bootm 83000000 88000000 8f000000
```

3.3 LS1043ARDB and LS1046ARDB

The QorIQ LS1043A/LS1046A reference design board is designed to exercise most capabilities of the LS1043A/LS1046A device; NXP's first quad-core, 64-bit ARM®-based processor for embedded networking and industrial infrastructure.

3.3.1 Switch Settings

The OpenIL just support SD card boot mode for ls1043ardb and ls1046ardb platforms.

Platform	Boot Source	SW setting
LS1043ARDB	SD card	SW4[1-8] +SW5[1] = 0b'00100000_0
LS1046ARDB	SD card	SW5[1-8] +SW4[1] = 0b'00100000_0

Table 3-2 LS1043ARDB/LS1046ARDB SD Boot SW Setting

3.3.2 Target Images Update

For LS1043ARDB and LS1046ARDB platforms, the OpenIL just support 32B system. To build the images for LS1043ARDB / LS1046ARDB platforms:

```
$ cd openil
$ make nxp_ls1043ardb-32b_defconfig
# or
$ make nxp_ls1046ardb-32b_defconfig
$ make
# or make with a log
$ make 2>&1 | tee build.log
```

- Programming U-Boot with RCW, FMan ucode and PPA in SD card
  Power on the LS1043ARDB / LS1046ARDB board to U-Boot command environment, then:

```
# programming U-Boot with RCW
=> tftpboot 82000000 u-boot-with-spl-pbl.bin
=> mmc write 82000000 8 800
# programming the FMan ucode
=> tftpboot 82000000 fsl_fman_ucode_ls1043_r1.0_108_4_5.bin
=> mmc write 82000000 820 50
# programming the PPA firmware
```

```
=> tftpboot 82000000 ppa.itb
=> mmc write 82000000 2800 36


#then reset the board
```

- Deploy kernel and Ramdisk from TFTP
  1. Set the U-Boot environment

```
=>setenv bootargs "root=/dev/ram0 earlycon=uart8250,mmio,0x21c0500 console=ttyS0,115200"
=>saveenv
```

  2. Boot up the system

```
# for ls1043ardb
=>tftp a0000000 kernel-ls1043a-rdb-aarch32.itb
# for ls1046ardb
=>tftp a0000000 kernel-ls1046a-rdb-aarch32.itb


=>bootm a0000000
```

3.4 LS1012ARDB

The QorIQ LS1012A processor delivers enterprise-class performance and security capabilities to consumer and networking applications in a package size normally associated with microcontrollers. Combining a 64-bit ARM®v8-based processor with network packet acceleration and QorIQ trust architecture security capabilities, the LS1012A features line-rate networking performance at 1W typical power in a 9.6mm x 9.6mm package.

The QorIQ LS1012A reference design board (LS1012A-RDB) is a compact form-factor tool for evaluating LS1012A application solutions. The LS1012A-RDB provides an Arduino shield expansion connector for easy prototyping of additional components such as an NXP NFC Reader module.

3.4.1 Switch Settings

The LS1012ARDB platform can boot just from QSPI Flash.

The table below lists the default switch settings and the description of these settings.

Platform	Boot Source	SW setting

LS1012ARDB	QSPI Flash 1	SW1 = 0b'10100110 SW2 = 0b'00000000
	QSPI Flash 2	SW1 = 0b'10100110 SW2 = 0b'00000010

Table 3-3 LS1012ARDB QSPI Boot SW Setting

3.4.2 Target Images Update

For LS1012ARDB platform, the OpenIL just support 32B system. To build the images for LS1012ARDB platforms:

```
$ cd openil
$ make nxp_ls1012ardb-32b_defconfig
$ make
# or make with a log
$ make 2>&1 | tee build.log
```

- Programming U-Boot, RCW and PPA in QSPI
  Power on the LS1012ARDB board to U-Boot command environment, then:

```
# programming U-Boot
=>i2c mw 0x24 0x7 0xfc; i2c mw 0x24 0x3 0xf5
=>tftp 0x80000000 <u-boot_file_name>.bin
=>sf probe 0:0
=>sf erase 0x100000 +$filesize
=>sf write 0x80000000 0x100000 $filesize
# programming RCW
=>i2c mw 0x24 0x7 0xfc; i2c mw 0x24 0x3 0xf5
=>tftp 0x80000000 <rcw_file_name>.bin
=>sf probe 0:0
=>sf erase 0x0 +$filesize
=>sf write 0x80000000 0x0 $filesize
#programming PPA
=> tftp 0x80000000 ppa.itb
=> sf probe 0:0
```

```
=> sf erase 0x500000 +$filesize
=> sf write 0x80000000 0x500000 $filesize
#then reset the board
```

- Deploy kernel and Ramdisk from TFTP
    1. Set the U-Boot environment

```
=>setenv bootargs ‘ttyS0,115200 root=/dev/ram0 earlycon=uart8250,mmio,0x21c0500'
=>saveenv
```

    2. Boot up the system

```
=>tftp a0000000 kernel-ls1012a-rdb-aarch32.itb
=>bootm a0000000
```

4 Industrial Features

4.1 NETCONF/YANG

- NETCONF v1.0 and v1.1 compliant (RFC 6241)
- NETCONF over SSH (RFC 6242) including Chunked Framing Mechanism
- DNSSEC SSH Key Fingerprints (RFC 4255)
- NETCONF over TLS (RFC 5539bis)
- NETCONF Writable-running capability (RFC 6241)
- NETCONF Candidate configuration capability (RFC 6241)
- NETCONF Validate capability (RFC 6241)
- NETCONF Distinct startup capability (RFC 6241)
- NETCONF URL capability (RFC 6241)
- NETCONF Event Notifications (RFC 5277 and RFC 6470)
- NETCONF With-defaults capability (RFC 6243)
- NETCONF Access Control (RFC 6536)
- NETCONF Call Home (Reverse SSH draft, RFC 5539bis)
- NETCONF Server Configuration (IETF Draft)

4.2 TSN

On the LS1021ATSN platform, TSN features are implemented as part of the SJA1105TEL Automotive Ethernet L2 switch.

- MII, RMII, RGMII, 10/100/1000 Mbps
- IEEE 802.1Q: VLAN frames and L2 QoS
- IEEE 1588v2: Hardware forwarding for one-step sync messages
- IEEE 802.1Qci: Ingress rate limiting (per-stream policing)
- IEEE 802.1Qbv: Time-aware traffic shaping
- Statistics for transmitted, received, dropped frames, buffer load
- TTEthernet (SAE AS6802)

4.3 Xenomai

Xenomai is a Free Software framework adding real-time capabilities to the mainline Linux kernel. Xenomai also provides emulators of traditional RTOS APIs, such as VxWorks® and pSOS®. Xenomai has a strong focus on embedded systems, although Xenomai runs over mainline desktop and server architectures as well.

Xenomai 3 is the new architecture of the Xenomai real-time framework, which can run seamlessly side-by-side Linux as a co-kernel system, or natively over mainline Linux kernels. In the latter case, the mainline kernel can be supplemented by the PREEMPT-RT patch to meet stricter response time requirements than standard kernel preemption would bring.

One of the two available real-time cores is selected at build time. The dual kernel core is codenamed Cobalt, the native Linux implementation is called Mercury.

Xenomai can help you in:

• Designing, developing and running a real-time application on Linux.

• Migrating an application from a proprietary RTOS to Linux.

• Optimally running real-time applications alongside regular Linux applications.

Xenomai features are supported for ls1021atsn, ls1043ardb and ls1046ardb.

More information can be found at Xenomai official website: http://xenomai.org/

4.3.1 Xenomai running mode

The dual kernel core is codenamed Cobalt, the native Linux implementation is called Mercury. Both Mercury and Cobalt are supported.

4.3.2 Running Xenomai Mercury

Xenomai Mercury provide the following API reference:

Test programs:

- latency

Xenomai timer latency benchmark, user manual can be found at:
http://www.xenomai.org/documentation/xenomai-3/html/man1/latency/index.html

- cyclictest
  Xenomai high resolution timer test, user manual can be found at:
  http://www.xenomai.org/documentation/xenomai-2.6/html/cyclictest/index.html

Utilities:

- xeno
  Wrapper for Xenomai executables, user manual can be found at:
  http://www.xenomai.org/documentation/xenomai-2.6/html/xeno/index.html
- xeno-config
  Display Xenomai libraries configuration, user manual can be found at:
  http://www.xenomai.org/documentation/xenomai-2.6/html/xeno-config/index.html

4.3.3 Running Cobalt mode

Xenomai Cobalt provides many APIs to do test.

1> Clocktest
Using test program clocktest provided by Xenomai to test timer APIs.
There are three kinds of timere sources:
CLOCK_REALTIME, CLOCK_MONOTONIC, CLOCK_HOST_REALTIME.

Check timer with clock name CLOCK_REALTIME:

```
$ clocktest –C 0
```

Check timer with clock name CLOCK_MONOTONIC:

```
$ clocktest –C 1
```

Check timer with clock name CLOCK_HOST_REALTIME:

```
$ clocktest –C 32
```

2> The interrupt handle by Cobalt
IFC and e1000e interrupt have been handled by Cobalt kernel.

```
$ cat /proc/xenomai/irq
```

Note: for e1000e test case, the Linux kernel standard network stack was used instead of rtnet stack.

3> Cobalt IPIPE tracer
The following options are available at kernel configuration time:

- “Enable tracing on boot” CONFIG_IPIPE_TRACE_ENABLE
  Defines if the tracer is active by default when booting the system or shall be later

enabled via /proc/ipipe/trace/enable. Specifically if function tracing is enabled, deferring to switch on the tracer reduces the boot time on low-end systems.

- “Instrument function entries” CONFIG_IPIPE_TRACE_MCOUNT

  Trace each entry of a kernel function. Note that this instrumentation, though it is the most valuable one, has a significant performance impact on low-end systems (~50% larger worst-case latencies on a Pentium-I 133 MHz).

- “Trace IRQs-off times” CONFIG_IPIPE_TRACE_IRQSOFF

  Instrument each disable and re-enable of hardware IRQs. This allows to identify the longest path in a system with IRQs disabled.

- “Depth of trace log” CONFIG_IPIPE_TRACE_SHIFT

  Controls the number of trace points. The I-pipe tracer maintains four ring buffers per CPU of the given capacity in order to switch traces in a lock-less fashion with respect to potentially pending output requests on those buffers. If you run short on memory, try reducing the trace log depth which is set to 16000 trace points by default.

- “Use vmalloc’ed trace buffer” CONFIG_IPIPE_TRACE_VMALLOC

  Instead of reserving static kernel data, the required buffer is allocated via vmalloc during boot-up when this option is enabled. This can help to start systems that are low on memory, but it slightly degrades overall performance. Try this option when a traced kernel hangs unexpectedly at boot time.

4> Latency of timer IRQ

```
$ latency -t 2 -T 60
```

Note: The location of 'latency' may be different from version to verion. Currently it locates in /usr/bin.

5> Latency of task in Linux kernel

```
$ latency -t 1 -T 60
```

6> Latency of task in user space

```
$ latency -t 0 -T 60
```

7> Smokey to check feature enabled

```
$ smokey --run
```

8> Thread context switch

```
$ switchtest -T 30
```

9> “xeno-test”
By default, the load command is "dohell 900", which will generate load during 15 minutes.

Step #1: Prepare one storage disk and ethernet port connected server, for example:
$ fdisk /dev/sda
$ mkfs.ext2 /dev/sda1
$ mount /dev/sda1 /mnt
$ ifconfig <nw port> <ip addr>

Step #2: $ cd /usr/xenomai/bin
Step #3: $ sudo ./xeno-test -l "dohell -s <server ip> -m /mnt"

4.4 IEEE 1588

4.4.1 Introduction

IEEE Std 1588-2008 (IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems) defines a protocol enabling precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects.

The 1588 timer module on NXP QorIQ platform provides hardware assist for 1588 compliant time stamping. Together with a software PTP (Precision Time Protocol) stack, it implements precision clock synchronization defined by this standard. Many open source PTP stacks are available with a little transplant effort, such as linuxptp which are used for this release demo.

4.4.2 PTP device types

There are five basic types of PTP devices, as follows:

a) Ordinary clock

A clock that has a single Precision Time Protocol (PTP) port in a domain and maintains the timescale used in the domain. It may serve as a source of time, i.e., be a master clock, or may synchronize to another clock, i.e., be a slave clock.

b) Boundary clock

A clock that has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain. It may serve as the source of time, i.e., be a master clock, and may synchronize to another clock, i.e., be a slave clock.

c) End-to-end transparent clock

A transparent clock that supports the use of the end-to-end delay measurement mechanism between slave clocks and the master clock.

d) Peer-to-peer transparent clock

A transparent clock that, in addition to providing Precision Time Protocol (PTP) event transit time information, also provides corrections for the propagation delay of the link connected to the port receiving the PTP event message. In the presence of peer-to-peer transparent clocks, delay measurements between slave clocks and the master clock are performed using the peer-topeer

delay measurement mechanism.

e) Management node

A device that configures and monitors clocks.

Note: Transparent clock, is a device that measures the time taken for a Precision Time Protocol (PTP) event message to transit the device and provides this information to clocks receiving this PTP event message.

4.4.3 linuxptp stack

This software is an implementation of the Precision Time Protocol (PTP) according to IEEE standard 1588 for Linux. The dual design goals are to provide a robust implementation of the standard and to use the most relevant and modern Application Programming Interfaces (API) offered by the Linux kernel. Supporting legacy APIs and other platforms is not a goal.

- Supports hardware and software time stamping via the Linux SO_TIMESTAMPING socket option.
- Supports the Linux PTP Hardware Clock (PHC) subsystem by using the clock_gettime family of calls, including the new clock_adjtimex system call.
- Implements Boundary Clock (BC) and Ordinary Clock (OC).
- Transport over UDP/IPv4, UDP/IPv6, and raw Ethernet (Layer 2).
- Supports IEEE 802.1AS-2011 in the role of end station.
- Modular design allowing painless addition of new transports and clock servos.

4.4.4 Quick start guide

Hardware requirement

- Two boards for basic master-slave synchronization
- Three or more boards for BC synchronization

Software requirement

- Linux BSP of Industry Solution Release
- PTP software stack

Ethernet interfaces connection for master-slave synchronization

Connect two Ethernet interfaces between two boards in back-to-back way. Then one board will be master and the other will be slave when they synchronize. Both the master and the slave are working as OCs.

Ethernet interfaces connection for BC synchronization

Three boards are required at least for BC synchronization. When three boards are used for BC synchronization. Assume board A will work as BC with two PTP ports. Board B and board C works as OCs.

Board	Clock Type	Interfaces used
A	BC	Interface 1, Interface 2.
B	OC	Interface 1
C	OC	Interface 1

Table 4-1 Ethernet Interfaces Connection for BC

1. Connect board A interface 1 to board B interface 1 in back-to-back way.
2. Connect board A interface 2 to board C interface 1 in back-to-back way.

For example, LS1021ATSN BC synchronization connection is as below figure.

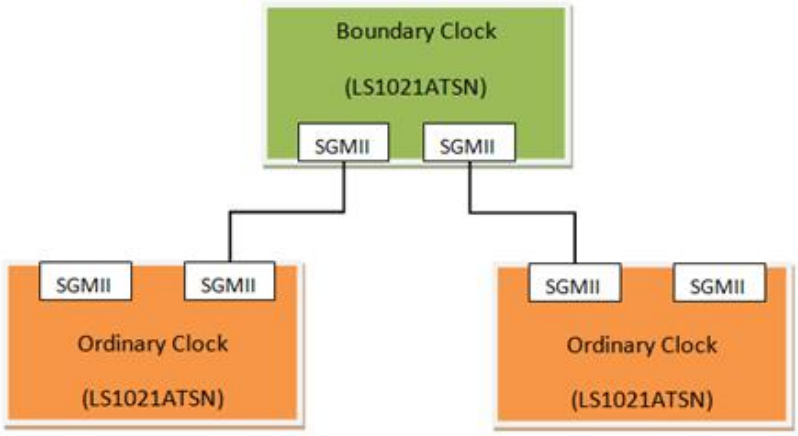

Figure 4-1 LS1021ATSN BC Synchronization

PTP stack startup

Before starting up kernel to run PTP stack, make sure there is no MAC address conflict in the network. Different MAC addresses should be set for each MAC on each board in u-boot. For example,

Board A:

```
=> setenv ethaddr 00:04:9f:ef:00:00
=> setenv eth1addr 00:04:9f:ef:01:01
=> setenv eth2addr 00:04:9f:ef:02:02
```

Board B:

```
=> setenv ethaddr 00:04:9f:ef:03:03
=> setenv eth1addr 00:04:9f:ef:04:04
=> setenv eth2addr 00:04:9f:ef:05:05
```

Board C:

```
=> setenv ethaddr 00:04:9f:ef:06:06
=> setenv eth1addr 00:04:9f:ef:07:07
=> setenv eth2addr 00:04:9f:ef:08:08
```

Linuxptp stack supports both OC and BC. It is included in SD card images of LS1021ATSN, LS1043ARDB, LS1046ARDB built by buildroot. Below command is used for running it. The master could be selected by software BMC (Best Master Clock) algorithm.

```
$ ptp4l -i eth0 -p /dev/ptp0 -f /etc/ptp4l_default.cfg –m
```

If the board is used as BC with several PTP ports, the '-i ' argument could point more interfaces. Such as below command for two PTP ports BC.

```
$ ptp4l -i eth0 -i eth1 -p /dev/ptp0 -f /etc/ptp4l_default.cfg –m
```

Note: The interface name and PTP device name in commands should be changed accordingly.

4.4.5 Known issues and limitations

1. For LS1021ATSN, the linuxptp stack only supports LS1021A Ethernet interfaces. It couldn't be used for SJA1105 switch Ethernet interfaces.

2. Packet loss issue could be observed on LS1021ATSN SGMII interfaces connected in back-to-back way. The root cause is that the PHY supports IEEE 802.11az EEE mode by default. The low speed traffic will make it go into low power mode. It affects 1588 synchronization performance greatly. Below workaround could be used to disable the feature.

```
$ ifconfig eth0 up
$ ethtool --set-eee eth0 advertise 0
$ ifconfig eth0 down
$ ifconfig eth0 up
```

3. The ptp4l stack may report getting tx timestamp timeout, but this rarely appears. This is not a bug. The stack tries to get tx timestamp after send message, but it couldn't get it if the driver hasn't completed tx timestamp processing in time. Just increasing tx_timestamp_timeout and re-running the stack will resolve this problem .

ptp4l[574.149]: timed out while polling for tx timestamp

ptp4l[574.152]: increasing tx_timestamp_timeout may correct this issue, but it is likely caused by a driver bug

4.4.6 Long term test results

Linuxptp

Connection: back-to-back master to slave

Configuration: Sync internal is -3

Test boards: two LS1021ATSN boards, one as master and another one as slave

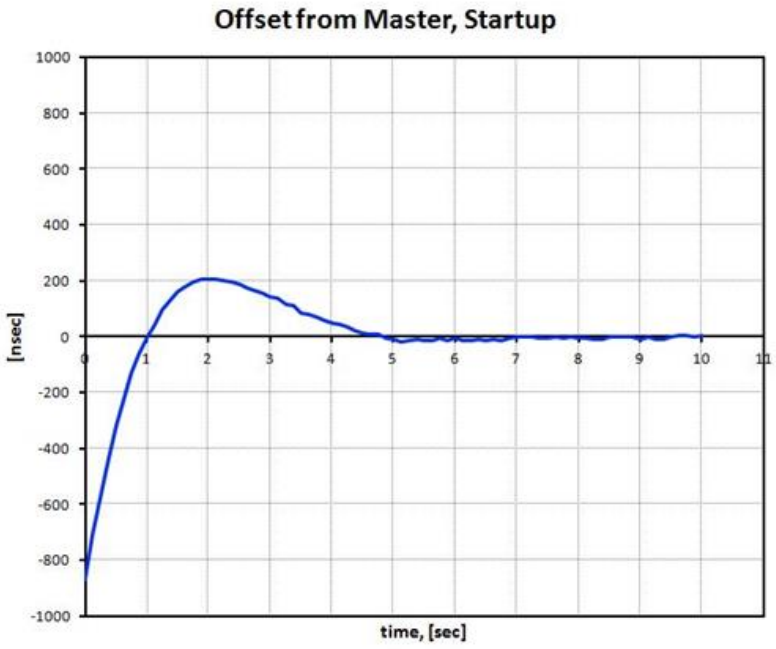

Figure 4-2 Offset from Master in Startup Stage

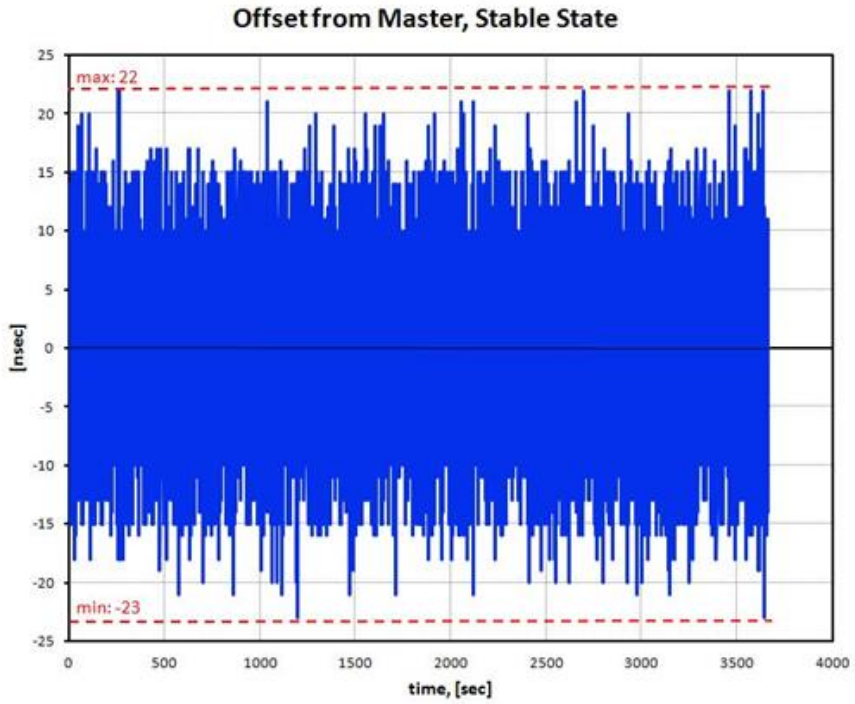

Figure 4-3 Offset from Master in Stable Stage

5 NETCONF/YANG

5.1 Overview

The NETCONF protocol defines mechanism for device management and configuration retrieval and modification. It uses a remote procedure call (RPC) paradigm and a system of exposing device (server) capabilities, which enables a client to adjust to the specific features of any network equipment. NETCONF further distinguishes between state data, which are read-only, and configuration data, which can also be modified. Any NETCONF communication happens on 4 layers as shown in the Table 5.1. XML is used as the encoding format.

1	Content	Configuration data	Notification data
2	Operations	<edit-config>	
3	Messages	<rpc>, <rpc-reply>	<notification>
4	Secure	Transport SSH, TLS	

Table 5-1 The NETCONF Layers

YANG is a standards-based, extensible, hierarchical data modeling language that is used to model the configuration and state data used by NETCONF operations, remote procedure calls (RPCs), and server event notifications. The device configuration data are stored in the form of an XML document. The specific nodes in the document as well as the allowed values are defined by a model, which is usually in YANG format or possibly transformed into YIN format with XML-based syntax. There are many such models created directly by IETF to further support standardization and unification of the NETCONF interface of the common network devices. For example, the general system settings of a standard computer are described in the ietf-system model (rfc7317) or the configuration of its network interfaces defined by the ietf-interfaces model (rfc7223). However, it is usual for every system to have some specific parts exclusive to it. In that case there are mechanisms defined to enable extensions while keeping the support for the standardized core. Also, as this whole mechanism is designed in a liberal fashion, the configuration does not have to concern strictly network. Even RPCs additional to those defined by NETCONF can be characterized, thus allowing the client to request an explicit action from the server.

A YANG module defines a data model through its data, and the hierarchical organization of and constraints on that data. A module can be a complete, standalone entity, or it can reference definitions in other modules and sub-modules as well as augment other data models with additional nodes. The module dictates how the data is represented in XML.

A YANG module defines not only the syntax but also the semantics of the data. It explicitly defines relationships between and constraints on the data. This enables you to create syntactically correct configuration data that meets constraint requirements and enables you to validate the data against the model before uploading it and committing it on a device.

For information about NETCONF, see RFC 6241, *NETCONF Configuration Protocol.*

For information about YANG, see RFC 6020, YANG - *A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, and related RFCs.

5.2 Netopeer

Netopeer is a set of NETCONF tools built on the libnetconf (https://github.com/CESNET/libnetconf) library. It allows operators to connect to their NETCONF-enabled devices as well as developers to allow control their devices via NETCONF.

5.2.1 libnetconf

libnetconf is a NETCONF library in C intended for building NETCONF clients and servers. It provides basic functions to connect NETCONF client and server to each other via SSH, to send and receive NETCONF messages and to store and work with the configuration data in a datastore. libnetconf implements the NETCONF protocol introduced by IETF. libnetconf is

maintained and further developed by the Tools for Monitoring and Configuration department of CESNET.

For information about libnetconf see:

https://github.com/CESNET/libnetconf

https://rawgit.com/CESNET/libnetconf/master/doc/doxygen/html/index.html

5.2.2 Netopeer Server

Netopeer software is a collection of utilities and tools to support the main application, Netopeer server, which is a NETCONF server implementation. It uses libnetconf for all NETCONF communication. Conforming to the relevant RFCs2 and still being part of the aforementioned library, it supports the mandatory SSH as the transport protocol but also TLS. Once a client successfully connects using either of these transport protocols and establishes a NETCONF session, it can send NETCONF RPCs and the server will respond with correct replies.

Most of the standard capabilities mentioned in the RFC 6241 are implemented including validating the new configuration before applying it or being able to return to a previous configuration if the new one failed to be applied. Except these, Netopeer server supports additional features such as sending notifications on certain events to a client, provided that it subscribes to them, or access control, when every user has the available parts of the configuration for reading and for writing specified and cannot access any other.

As part of the Netopeer server, there is a set of the following tools:

- `netopeer-server` as the main service daemon integrating the SSH/TLS server.
- `netopeer-manager` as a tool to manage the `netopeer-server`'s modules.
- `netopeer-configurator` as a tool for the server first run configuration.

5.2.3 Netopeer Client

`Netopeer-cli` is a CLI interface allowing user to connect to a NETCONF-enabled device and to obtain and manipulate its configuration data.

This application is part of the Netopeer software bundle, but compiled and installed separately. It is a NETCONF client with a command line interface developed and primarily used for Netopeer server testing, but allowing all the standards and even some optional features of a full-fledged NETCONF client.

`netopeer-cli` serves as a generic NETCONF client providing a simple interactive command line interface. It allows user to establish a NETCONF session with a NETCONF-enabled device on the network and to obtain and manipulate its configuration data. `netopeer-cli` is limited to a single NETCONF connection at a time via a forward or a reverse (Call Home) connecting method.

5.3 sja1105 YANG Models Introduction

There are registers tables in the sja1105.

"schedule-table",

"schedule-entry-points-table",

"vl-lookup-table",

"vl-policing-table",

"vl-forwarding-table",

"l2-address-lookup-table",

"l2-policing-table",

"vlan-lookup-table",

"l2-forwarding-table",

"mac-configuration-table",

"schedule-parameters-table",

"schedule-entry-points-parameters-table",

"vl-forwarding-parameters-table",

"l2-address-lookup-parameters-table",

"l2-forwarding-parameters-table",

"clock-synchronization-parameters-table",

"avb-parameters-table",

"general-parameters-table",

"retagging-table",

"xmii-mode-parameters-table",

Each table owns their registers list. The sja1105 yang model try to add all the table entries elements as leaves.

module: sja1105	
Configuration	+--rw sja1105 +--rw schedule-table | +--ro name? string | +--rw entry* [index] | +--rw index uint32 | +--rw winstindex? string | +--rw winend? string

```
|    +--rw winst?         string
|    +--rw destports?     string
|    +--rw setvalid?      string
|    +--rw txen?          string
|    +--rw resmedia_en?   string
|    +--rw resmedia?      string
|    +--rw vlindex?       string
|    +--rw delta?         string
+--rw schedule-entry-points-table
| +--ro name?    string
| +--rw entry* [index]
|    +--rw index         uint32
|    +--rw subschindx?   string
|    +--rw delta?        string
|    +--rw address?      string
+--rw l2-address-lookup-table
| +--ro name?    string
| +--rw entry* [index]
|    +--rw index         uint32
|    +--rw vlanid?       string
|    +--rw macaddr?      string
|    +--rw destports?    string
|    +--rw enfport?      string
+--rw l2-policing-table
| +--ro name?    string
| +--rw entry* [index]
|    +--rw index         uint32
|    +--rw sharindx?     string
|    +--rw smax?         string
|    +--rw rate?         string
|    +--rw maxlen?       string
|    +--rw partition?    string
+--rw vlan-lookup-table
| +--ro name?    string
| +--rw entry* [index]
|    +--rw index         uint32
|    +--rw ving_mirr?    string
|    +--rw vegr_mirr?    string
|    +--rw vmemb_port?   string
|    +--rw vlan_bc?      string
|    +--rw tag_port?     string
|    +--rw vlanid?       string
+--rw l2-forwarding-table
| +--ro name?    string
| +--rw entry* [index]
|    +--rw index         uint32
|    +--rw bc_domain?    string
|    +--rw reach_port?   string
|    +--rw fl_domain?    string
|    +--rw vlan_pmap?    string
+--rw mac-configuration-table
```

```
      |  +--ro name?    string
      |  +--rw entry* [index]
      |     +--rw index         uint32
      |     +--rw top?          string
      |     +--rw base?         string
      |     +--rw enabled?      string
      |     +--rw ifg?          string
      |     +--rw speed?        string
      |     +--rw tp_delin?     string
      |     +--rw tp_delout?    string
      |     +--rw maxage?       string
      |     +--rw vlanprio?     string
      |     +--rw vlanid?       string
      |     +--rw ing_mirr?     string
      |     +--rw egr_mirr?     string
      |     +--rw drpnona664?   string
      |     +--rw drpdtag?      string
      |     +--rw drpuntag?     string
      |     +--rw retag?        string
      |     +--rw dyn_learn?    string
      |     +--rw egress?       string
      |     +--rw ingress?      string
      +--rw schedule-parameters-table
      |  +--ro name?    string
      |  +--rw entry* [index]
      |     +--rw index         uint32
      |     +--rw subscheind?   string
      +--rw schedule-entry-points-parameters-table
      |  +--ro name?    string
      |  +--rw entry* [index]
      |     +--rw index        uint32
      |     +--rw clksrc?      string
      |     +--rw actsubsch?   string
      +--rw l2-address-lookup-parameters-table
      |  +--ro name?    string
      |  +--rw entry* [index]
      |     +--rw index            uint32
      |     +--rw maxage?          string
      |     +--rw dyn_tbsz?        string
      |     +--rw poly?            string
      |     +--rw shared_learn?    string
      |     +--rw no_enf_hostprt?  string
      |     +--rw no_mgmt_learn?   string
      +--rw l2-forwarding-parameters-table
      |  +--ro name?    string
      |  +--rw entry* [index]
      |     +--rw index       uint32
      |     +--rw max_dynp?   string
      |     +--rw part_spc?   string
      +--rw general-parameters-table
      |  +--ro name?    string
```

	| +--rw entry* [index] | +--rw index uint32 | +--rw vllupformat? string | +--rw mirr_ptacu? string | +--rw switchid? string | +--rw hostprio? string | +--rw mac_fltres1? string | +--rw mac_fltres0? string | +--rw mac_flt1? string | +--rw mac_flt0? string | +--rw incl_srcpt1? string | +--rw incl_srcpt0? string | +--rw send_meta1? string | +--rw send_meta0? string | +--rw casc_port? string | +--rw host_port? string | +--rw mirr_port? string | +--rw vimarker? string | +--rw vimask? string | +--rw tpid? string | +--rw ignore2stf? string | +--rw tpid2? string +--rw xmii-mode-parameters-table | +--ro name? string | +--rw entry* [phy_mac] | +--rw phy_mac string | +--rw xmii_mode? string
State Data	+--rw version | +--ro deviceId? string | +--ro revision? string +--ro config-files | +--ro configFile* string +--ro ports +--ro port* string
RPC	rpcs: +---x save-local-config | +---w input | +---w configfile? string +---x load-local-config | +---w input | +---w configfile? string +---x load-default
Notification	

Table 5-2 sja1105 yang model tree

5.4 Install Netopeer-cli On Centos/Ubuntu

Centos 7.2:

Install the following packages:

```
$ sudo yum install libxml2-devel libxslt-devel openssl-devel libgcrypt dbus-devel doxygen libevent readline.x86_64 ncurses-libs.x86_64 ncurses-devel.x86_64 libssh.x86_64 libssh2-devel.x86_64 libssh2.x86_64 libssh2-devel.x86_64
```

pyang install:

```
$ git clone https://github.com/mbj4668/pyang.git
$ cd pyang
$ sudo python setup.py install
```

Pull, configure, make, and install libnetconf:

```
$ git clone https://github.com/CESNET/libnetconf.git
$ cd libnetconf
$ ./configure
$ sudo make
$ sudo make install
```

Pull netopeer and configure, make, and install cli:

```
$ git clone https://github.com/CESNET/netopeer.git
$cd netopeer/cli
$ ./configure
$ make
$ make install
```

Ubuntu 16.04:

Install the following packages:

```
$ sudo apt-get install libxml2-dev libxslt-dev libssl-dev libssh2-1-dev xsltproc doxygen pkg-config libtool-bin cmake libssh-dev libevent-dev libreadline-dev libcurl4-openssl-dev python-libxml2 autoconf
```

pyang install:

```
$ git clone https://github.com/mbj4668/pyang.git
$ cd pyang
$ sudo python setup.py install
```

Note:

There is a version issue for libssh install in the Ubuntu below version 16.04. Apt-get install libssh may get version 0.6.4. But libnetconf need a version of 0.7.3 or later. Remove the default one and reinstall by download source code and install manually.

```
$ dpkg -r libssh-dev
```

```
$ wget https://red.libssh.org/attachments/download/195/libssh-0.7.3.tar.xz --no-check-certificate
$ unxz libssh-0.7.3.tar.xz
$ tar -xvf libssh-0.7.3.tar
$ cd libssh-0.7.3/
$ mkdir build
$ cd build/
$ cmake ..
$ make
$ sudo make install
```

Pull, configure, make, and install libnetconf:

```
$ git clone https://github.com/CESNET/libnetconf.git
$ cd libnetconf
$ ./configure
$ sudo make
$ sudo make install
```

Pull netopeer and configure, make, and install cli:

```
$ git clone https://github.com/CESNET/netopeer.git
$ cd netopeer/cli
```

```
$ ./configure
$ make
$ sudo make install
```

5.5 Configuration

5.5.1 Netopeer-server

The `netopeer-server` is the NETCONF protocol server running as a system daemon. The `netopeer-server` is based on the libnetconf library. It provides an environment to run transAPI modules for configuration a specific device or application according to its data model.

OPTIONS

-d Run in daemon mode.

-h Show help.

-V Show program version.

-v level Set the verbosity level. Possible values are from 0 (default) to 3. This overrides any NETOPEER_VERBOSE environment variable.

5.5.2 Netopeer-manager

`netopeer-manager` provides access to the configuration of the `netopeer-server` modules. The `netopeer-server` modules extends its functionality to control another devices or applications via transAPI or just by storing configuration data.

OPTIONS

--help

Prints the generic description and a list of commands. Detailed description and list of arguments for the specific command are displayed by using --help argument of the command.

COMMANDS

add

Add a new `netopeer-server` module. Added module is enabled by default and it will be loaded by the `netopeer-server` during its next start.

add [--help] --name NAME (--model MODEL | --augment AUGMENT | --import IMPORT) [--transapi TRANSAPI] [--features FEATURE [FEATURE ...]] [--datastore DATASTORE]

--name NAME

Specifies the name of the `netopeer-server` module. The NAME is used as an identifier of the module in the `netopeer-server` configuration.

--model MODEL

Specifies path (absolute or relative) to the module's main data model in YIN format. In this option, the whole module configuration is created.

--augment AUGMENT

Specifies path (absolute or relative) to an augment model of the main data model in YIN format. This model is always appended at the end of the model list.

--import IMPORT

Specifies path (absolute or relative) to a model in YIN format that is imported by the main model. This model is always prepended at the beginning of the model list.

--transapi TRANSAPI

Optional parameter to specify path to the transAPI module related to the module's main data model. If the transAPI module is not specified, `netopeer-server` will allow the connection.

figuration data manipulation according to the data model, but the changes will not be applied to any device. This part of the process is handled just by the transAPI module.

--features FEATURE [FEATURE ...]

Data model can define various features that extend its basic functionality. By default, `netopeer-server` supposes all features to be disabled. This option explicitly specifies

List of features to enable. If one want to enable all features, value * can be used.

--datastore DATASTORE

Specifies path to the file where the configuration data will be stored. If not specified, datastore is implemented as empty and it will not able to store any configuration data.

list

Print the list of all `netopeer-server`'s modules.

list [--help] [--name NAME]

--name NAME

If specified, it is the name of the main `netopeer-server` module for which the list of extending data models will be printed.

rm

Remove the specified `netopeer-server` main module.

rm [--help] --name NAME [--model MODEL]

--**name** NAME

Specifies the name of the main `netopeer-server` module to remove.

--**model** MODEL

If specified, only this extending model will be removed instead of the whole module.

5.5.3 netopeer-cli

`netopeer-cli` is command line interface as NETCONF client. `netopeer-cli` serves as a generic NETCONF client providing a simple interactive command line interface. It allows user to establish a NETCONF session with a NETCONF-enabled device on the network and to obtain and manipulate its configuration data. `netopeer-cli` is limited to a single NETCONF connection at a time via a forward or a reverse (Call Home) connecting method.

COMMANDS

help

Display list of commands. --help option is also accepted by all commands to show detailed information about the command.

connect

Connect to a NETCONF server.

connect [--help] [--login username] [--port num] host

--login username

Specifies the user to log in as on the NETCONF server. If not specified, current local username is taken.

--tls

Use NETCONF over TLS transport instead of default SSH. Default client certificate and trusted CA directory are used for TLS handshake.

This option is available only when the `netopeer-cli` is compiled with configure's --enable-tls option.

--cert cert_path

Use a specific certificate for TLS handshake. cert_path specifies path to the client certificate in CRT format. If --key option is not specified, cert_path is expected to contain also the private key for the client certificate, in PEM format.

This option is available only when the `netopeer-cli` is compiled with configure's --enable-tls option.

--key key_path

Specifies path to the private key for the client certificate in KEY format. If not specified, cert_path is expected to contain also the private key for the client certificate, in PEM format.

This option is available only when the `netopeer-cli` is compiled with configure's --enable-tls option.

--trusted trusted_CA_store

Specifies path to a trusted CA certificate bundle in PEM format to be used exclusively for server verification for this connection instead of the default CA directory.

This option is available only when the `netopeer-cli` is compiled with configure's --enable-tls option.

--port num

Port to connect to on the NETCONF server. By default, port 830 for SSH or 6513 for TLS transport is used.

host

Hostname of the target NETCONF server.

disconnect

Disconnect from a NETCONF server.

commit

Perform NETCONF <commit> operation. For more details see RFC 6241 section 8.3.4.1.

copy-config

Perform NETCONF <copy-config> operation. For more details see RFC 6241 section 7.3.

copy-config [--help] [--defaults mode] [--source datastore | --config file] target_datastore

--defaults mode

Use :with-defaults capability with specified retrieval mode. For more details see RFC 6243 section 3 or WITH-DEFAULTS section of this manual.

--source datastore

Specifies source datastore for the <copy-config> operation. For description of the datastore parameter, see the DATASTORES section of this manual.

This option is available only when the `netopeer-cli` is compiled with configure's --enable-tls option.

--config file

Specifies path to a local file containing the complete configuration to copy. This option alternates the --source option.

target_datastore

Target datastore to be rewritten. For description of possible values, see the DATASTORES section of this manual.

delete-config

Perform NETCONF <delete-config> operation. For more details see RFC 6241 section 7.4.

delete-config [--help] target_datastore

t**arget_datastore**

Target datastore to delete. For description of possible values, see the DATASTORES section of this manual. Note, that the running configuration datastore cannot be deleted.

discard-changes

Perform NETCONF <discard-changes> operation. It reverts the candidate configuration to the current running configuration. For more details see RFC 6241 section 8.3.4.2.

edit-config

Perform NETCONF <edit-config> operation. For more details see RFC 6241 section 7.2.

edit-config [--help] [--defop operation] [--error action] [--test option] [--config file | --url URI] target_datastore

--defop operation

Specifies default operation for applying configuration data.

merge Merge configuration data at the corresponding level. This is the default value.

replace Edit configuration data completely replaces the configuration in the target datastore.

none The target datastore is unaffected by the edit configuration data, unless and until the edit configuration data contains the operation attribute to request a different operation. For more info, see the EDIT-CONFIG section of this document.

--error action

Set reaction to an error.

stop Abort the operation on first error. This is the default value.

Continue Continue to process configuration data on error. The error is recorded and negative response is returned.

Rollback Stop the operation processing on error and restore the configuration to its complete state at the start of this operation. This action is available only if the server

supports :rollback-on-error capability (see RFC 6241 section 8.5).

--test option

Perform validation of the modified configuration data. This option is available only if the server supports :validate:1.1 capability (see RFC 6241 section 8.6).

set Do not perform validation test.

test-only Do not apply the modified data, only perform the validation test.

test-then-set Perform a validation test before attempting to apply modified configuration data. This is the default value.

--config file

Specify path to a file containing edit configuration data. The content of the file is placed into the <config> element of the edit-config operation. Therefore, it doesn't have to be a well-formed XML document with only a single root element. If neither --config nor --url is specified,

user is prompted to write edit configuration data manually. For examples, see the EDIT-CONFIG section of this document.

--url URI

Specify remote location of the file containing the configuration data hierarchy to be modified, encoded in XML under the element <config> in the urn:ietf:params:xml:ns:netconf:base:1.0 namespace. Note, that this differs from file parameter, where the <config> element is not expected.

target_datastore

Target datastore to modify. For description of possible values, see the DATASTORES section of this manual. Note, that the url configuration datastore cannot be modified.

get

Perform NETCONF <get> operation. Receives both status as well as configuration data from the current running datastore. For more details see RFC 6241 section 7.7.

get [--help] [--defaults mode] [--filter [file]]

--defaults mode

Use :with-defaults capability with specified retrieval mode. For more details see RFC 6243 section 3 or WITH-DEFAULTS section of this manual.

--filter [file]

Specifies if the request will contain subtree filter (RFC 6241 section 6). The option is able to accept path to the file containing the filter specification. If the path is not specified, user is prompted to write the filter specification manually.

get-config

Perform NETCONF <get-config> operation. Retrieves only configuration data from the specified target_datastore. For more details see RFC 6241 section 7.1.

get-config [--help] [--defaults mode] [--filter [file]] target_datastore

--defaults mode

Use :with-defaults capability with specified retrieval mode. For more details see RFC 6243 section 3 or WITH-DEFAULTS section of this manual.

--filter [file]

Specifies if the request will contain subtree filter (RFC 6241 section 6). The option is able to accept path to the file containing the filter specification. If the path is not specified, user is prompted to write the filter specification manually.

target_datastore

Target datastore to retrieve. For description of possible values, see the DATASTORES section of this manual. Note, that the url configuration datastore cannot be retrieved.

get-schema

Perform NETCONF <get-schema> operation that retrieves specified data model used by the server. This operation is available only if the server implements the YANG Module for NETCONF Monitoring. The list of available schemas can be retrieved from /netconf-state/schemas subtree via the <get> operation. For more details see RFC 6022 sections 3.1 and 4.

get-schema [--help] [--version version] [--format format] identifier

--version version

Version of the requested schema.

--format format

The data modeling language (format) of the requested schema. Default value is yang.

identifier

Identifier for the schema list entry.

lock

Perform the NETCONF <lock> operation to lock the entire configuration datastore of a server. For more details see RFC 6241 section 7.5.

lock [--help] target_datastore

target_datastore

Target datastore to lock. For description of possible values, see the DATASTORES section of this manual. Note, that the url configuration datastore cannot be locked.

unlock

Perform the NETCONF <unlock> operation to release a configuration lock, previously obtained with the <lock> operation. For more details see RFC 6241 section 7.6.

lock [--help] target_datastore

target_datastore

Target datastore to unlock. For description of possible values, see the DATASTORES section of this manual. Note, that the url configuration datastore cannot be unlocked.

user-rpc

Send your own content in an RPC envelope. This can be used for RPC operations defined in data models not supported by the `netopeer-cli`.

user-rpc [--help] [--file file]

--file file

Specifies a file containing NETCONF RPC operation in XML format. Only the NETCONF <rpc> envelope is added to the file content and then it is sent to a server. If the option is omitted, user is prompted to type the RPC content manually.

verbose

Enable/disable verbose messages.

debug

Enable/disable debug messages. Available only if the `netopeer-cli` is compiled with configure's --enable-debug option.

quit

Quit the program.

DATASTORES

running

Base NETCONF configuration datastore holding the complete configuration currently active on the device. This datastore always exists.

startup

The configuration datastore holding the configuration loaded by the device when it boots. Only present on servers that implement :startup capability.

candidate

The configuration datastore that can be manipulated without impacting the device's current configuration and that can be committed to the running configuration datastore. Only

present on servers that implement :candidate capability.

url:URI

Refers to a remote configuration datastore located at URI. The file that the URI refers to contains the configuration data hierarchy to be modified, encoded in XML under the eleâ€

ment <config> in the urn:ietf:params:xml:ns:netconf:base:1.0 namespace. This datastore is available only on servers that implement :url capability.

5.5.4 Operation Examples

In the sja1105 yang model, user may set any register of sj1105 or a whole registers map by `netopeer-cli`.

Make sure the netconf server is running (check `netopeer-server` is running at ls1021atsn). If `netopeer-server` is not running, input command to let it run(verbose mode):

```
$ /usr/local/bin/netopeer-server -v 3
```

Test commands at netopeer-client(a CentOS/Ubuntu PC to run `netopeer-cli` as example):

Start client software

```
$ netopeer-cli
```

Connect the netconf server ls1021atsn board(use the ip on ls1021atsn, here 10.193.20.53 is example):

```
$ Netconf> connect --port 830 --login root 10.193.20.53
```

Get staus datas of server

```
$ netconf> get
```

User can get daters what he wants by --filter option

```
$ netconf> get --filter
```

Then input the filtered node you want to see in the text editor:

```
<sja1105>
    <ports/>
</sja1105>
```

Get config daters what he wants in running/candidate/startup space:

```
$ netconf> get-config running
```

```
$ netconf> get-config candidate
```

```
$ netconf> get-config startup
```

You can also add '--filter' parameter to filter what you want to see.

You can use 'discard-changes' command to copy data from 'running' to 'candidate' space.

```
$ netconf> discard-changes
```

Note:

It is recommend to run 'discard-changes' when the first time the board boot up. See chapter 5.6 Trouble shootings No. 5.

You can use ‘commit’ command to copy data from ‘candidate’ to ‘running’ space.

```
$ netconf> commit
```

User can edit any elements in the tables or multi tables’ elements for running/candidate/startup space.

```
$ netconf> edit-config candidate
```

‘candidate’ modification would only edit the candidate registers map. Not really modify into sja1105 true registers.

```
$ netconf> edit-config running
```

‘running’ modification would only edit the running registers map. It is really modifying sja1105 true registers.

```
$ netconf> edit-config startup
```

‘startup’ modification would only edit the startup registers map. Not really modify into sja1105 true registers.

Input below text into the editor as example and wait for got 'OK' message:

```
        <sja1105 xmlns="http://nxp.com/ns/yang/tsn/sja1105">
    <l2-policing-table>
        <entry>
          <index>1</index>
          <sharindx>0x1</sharindx>
          <smax>0x8000</smax>
          <rate>0xFA00</rate>
          <maxlen>0x5EE</maxlen>
          <partition>0x0</partition>
        </entry>
```

```
    </l2-policing-table>
</sja1105>
```

The example is trying to edit entry index 1 entry in l2-policing-table.

Refer the chapter 5.3 to check the yang model of sja1105 data design.

User can edit config a saved xml file with all of or part of the full yang model nodes list. You can refer the files under board/nxp/ls1021atsn/rootfs_overlay/etc/sja1105/*.xml for how to input edit-config contend in the cross compiler buildroot folder.

```
$ ls board/nxp/ls1021atsn/rootfs_overlay/etc/sja1105/*.xml
policing.xml   prioritizing.xml  standard.xml
```

Even you can check the server board to see what xml files has been saved in the server:

```
$ netconf> get --filter
<sja1105><config-files/></sja1105>
```

So as an example of the edit-config command would be(suppose we are under buildroot source code root folder):

```
$ netconf> edit-config --config board/nxp/ls1021atsn/rootfs_overlay/etc/sja1105/policing.xml
running
```

Use 'running' option to direct edit into physical registers of sja1105. Use 'candidate' option to edit the register into candidate space.

User can copy data between the running/candidate/startup by ‘copy-config’ or source xml file in the client:

```
$ netconf> copy-config --source running startup
```

Here is an example to copy the running space date to startup space.

Three more rpc calls are provided in the sja1105 yang model.

If user wants to save current registers mapping into a file (must extend with .xml. This ‘save-local-config’ command would save current ‘running’ space config registers value into the file.)

In the `netopeer-cli` shell, by ‘user-rpc’ command:

```
$ netconf> user-rpc
```

Input below text into the editor (nc_standard.xml is an example file name):

```
<save-local-config xmlns="http://nxp.com/ns/yang/tsn/sja1105">
   <configfile>
      nc_standard.xml
   </configfile>
</save-local-config>
```

Then a file nc_standard.xml would save to the /etc/sja1105/ in the server (board).

Note:

You can get all saved xml files by 'get' command by ‘config-files’ filter.

If user wants to load a xml to set the sja1105 registers into ‘running’ and ‘candidate’ space, user can use rpc command ‘load-local-config’. The xml file must list in the /etc/sja1105/. (Get file list by 'get' command and with ‘config-files’ filter.)

In the `netopeer-cli` shell, by ‘user-rpc’ command:

```
$ netconf> user-rpc
```

Input below text into the editor (policing.xml is an example name):

```
<load-local-config xmlns="http://nxp.com/ns/yang/tsn/sja1105">
   <configfile>
      policing.xml
   </configfile>
</load-local-config>
```

If user wants to load a default registers mapping mode (sja1105 as a normal switch mode).

In the `netopeer-cli` shell, by 'user-rpc' command:

```
$ netconf> user-rpc
```

Input below text into the editor (policing.xml is an example name):

```
<load-default xmlns="http://nxp.com/ns/yang/tsn/sja1105" />
```

5.6 Troubleshooting

1. Connect fail at client side:

```
libnetconf ERROR: Remote host key changed, the connection will be terminated!
libnetconf ERROR: Checking the host key failed.
```

Fixing:

The reason is that the SSHD key changed in server.
You need to input command 'quit' `netopeer-cli` first. Then remove *~/.ssh/known_hosts* and restart `netopeer-cli`.

2. Command error shows:

```
libnetconf ERROR: Input channel error (Socket error: Connection reset by peer)
user-rpc: receiving rpc-reply failed.


 Closing the session.
```

Fixing:
Lost connection in few minutes later if not to communicate with server.

Reconnect server.

3. Run command at `netopeer-server`(board):

```
[$ /usr/local/bin/netopeer-server -v 3
```

Got error:

```
netopeer-server[216]: sock_listen: could not bind "::0" port 830 (Address already in use)
netopeer-server[216]: Server is not listening on any address!
```

Fixing:
There should another `netopeer-server` is running. Use command to check:

```
$ ps | grep netopeer-server
```

If there is a `netopeer-server` process:

```
$ /usr/local/bin/netopeer-server -d
```

4. Terminate `netopeer-server`.

   The right way to terminate the server use command Ctrl ^ C or:

```
$ /etc/init.d/S90netconf stop
```

5. Get a empty candidate contend:

```
$ netconf> get-config candidate


 Result:
```

   Fixing:
   It is recommend to input a ‘discard-changes’ command when the first time for `netopeer-server` boot up and connected by client.

```
$ netconf> discard-changes
```

6. When you operate command ‘edit-config’ ‘get-config’ got error:

```
NETCONF error: operation-failed (application) - There is no device/data that could be affected.
```

Fixing:
The reason is that the server was not properly terminated. If you killed the server several times then all this is stored in the shared memory and that many times on next server startup the candidate datastore will not be replaced by the running datastore.
Do not uses kill -9 to terminate `netopeer-server` process. Please use Ctrl ^ C in the terminal with `netopeer-server` (or you can send SIGINT instead SIGKILL)

You should restart `netopeer-server` (Ctrl^ C or /etc/init.d/S90netconf restart)

```
$ /etc/init.d/S90netconf restart
```

Then in the client `netopeer-cli` soon after connected :

```
$ netconf> discard-changes
```

7. The `netopeer-server` default to set nothing registers of sja1105 chip when server startup. If user wants to let `netopeer-server` auto set a series registers in sja1105 at `netopeer-server` boot up time, you need to set the registers value to the startup space. Below contend introduce how to set ‘startup’ space.

   Edit ‘startup’ space configuration.

   ‘startup’ space configures are for setting sja1105 registers when `netopeer-server` start time.
   There are several ways to edit ‘startup’ space configures data. Here are the examples:

   From the ‘running’ space configure copy to ‘startup’ space:

   ```
   $ netconf> copy-config --source running startup
   ```

   Copy from client local file (policing.xml is example file name) to server ‘startup’ space:

   ```
   $ netconf> copy-config copy-config --config policing.xml startup
   ```

   Modify some sub-tree registers from client local file (policing.xml is example file name) to server ‘startup’ space:

   ```
   $ netconf> edit-config --config policing.xml startup
   ```

8. Validate command shows:

   ```
   NETCONF error: operation-not-supported (application) - Request could not be completed because the requested operation is not supported by this implementation.
   ```

Note:
'validate' command is not supported in current version.

9. Datastores dead lock

When the server crashes or is terminated with SIGKILL, it may happen that the internal datastore locks stay locked. In such a case, the next time the `netopeer-server` (or any other libnetconf based application) tries to access the configuration datastores, it freezes. To solve this problem, release the locks manually removing the /dev/shm/sem.NCDS_FLOCK_* files.

Refer the chapter 5.6 No. 4 for how to terminate the `netopeer-server` proper.

10. Modifying */etc/network/interfaces* probably cause the `netopeer-server` segment fault when `netopeer-server` starts if ietf-interfaces be selected in the buildroot. Use command to remove the ietf-interfaces model.

```
$ /usr/local/bin/netopeer-manager rm --name ietf-interfaces
$ /usr/local/bin/netopeer-manager list
```

6 1-Board TSN Demo

6.1 Introduction

Time Sensitive Networking (TSN) is an extension to traditional Ethernet networks, providing a set of standards compatible with IEEE 802.1 and 802.3. These extensions are intended to address the limitations of standard Ethernet in sectors ranging from industrial and automotive applications to live audio and video systems.

Traditional Ethernet applications must be designed very robust in order to withstand corner cases such as packet loss, delay or even reordering. TSN aims to provide guarantees for deterministic latency and packet loss under congestion, allowing critical and non-critical traffic to be converged in the same network.

On NXP platforms, TSN features are provided by the SJA1105TEL Automotive Ethernet switch present on the LS1021ATSN board. These hardware features can be used to implement the following standards:

- 802.1Qbv - Time Aware Shaping
- 802.1Qci - Per-Stream Filtering and Policing

6.2 Objectives

- Manage bandwidth problems related to network contention
- Demonstrate the features of the L2 Ingress Policer
- Create time slots for scheduled traffic
- Show the usage of the sja1105-tool

6.3 Demo overview

The TSN demo employs 3 Linux machines connected through the SJA1105 switch. Of these 3 machines, one is the LS1021 SoC and the other 2 should be laptops or PC's connected through Ethernet cables to the LS1021ATSN board. A diagram of the connections required is shown in Figure 6-1**Error! Reference source not found.Error! Reference source not found.Error! Reference source not found.Error! Reference source not found.**. 1000Mbps Ethernet ports are required on the 2 GNU/Linux laptops.

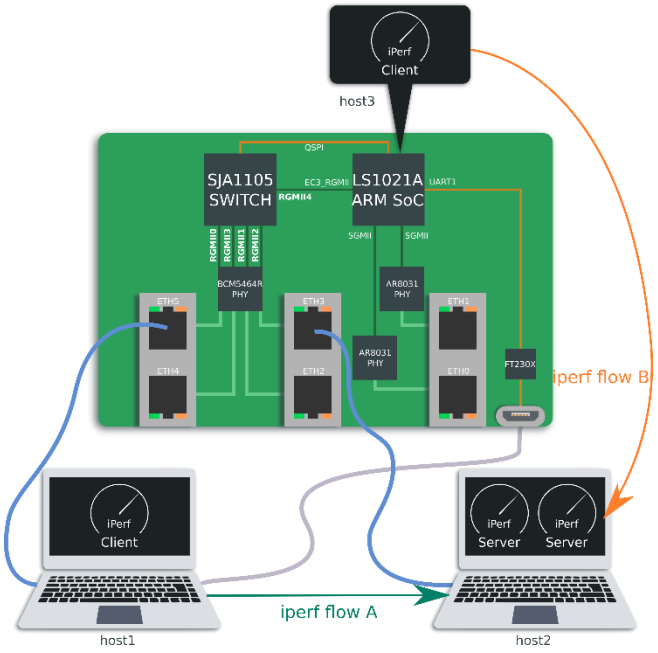

Figure 6-1. Diagram of connections of the laptops to the board and their role

Through the SJA1105 switch there are two TCP flows competing for bandwidth:

- An iPerf connection running from client Host 1 to server Host 2
- An iPerf connection running from client Host 3 to server Host 2

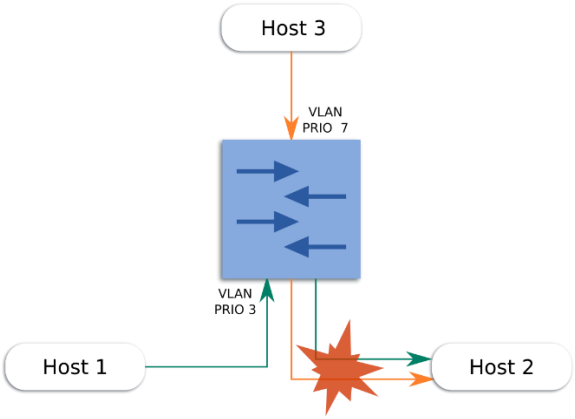

Figure 6-2. Flows directed from Host 1 and Host 3 towards Host 2 are bottlenecked at the switch egress interface

The schematic diagram of the two flows is shown in Figure 6-2. A correlation can be drawn between Figure 6-1 and Figure 6-2. The LS1021 (Host 3) and one of the Linux laptops (Host 1) are acting as iPerf clients, whereas the second Linux laptop (Host 2) is the iPerf server.

Because both these flows share the same link between the SJA1105 and Host 2, they are bottlenecked and competing for the 1000Mbps total bandwidth of that link. The demo shows 3 approaches to isolate the flows' impact on one another:

- Standard switch configuration: This is the behavior of traditional Ethernet switches.
- Ingress Policing: Rate-limit traffic coming from Host 3 (in order to protect the flow Host 1 - Host 2).
- Time Gating: Schedule the 2 flows on different time slots.

6.4 Use of VLAN tags in the demo

The 802.1Q standard specifies that VLAN-encapsulated Ethernet frames have an additional 4 octet header with the following fields:

- VLAN Ethertype: this must be set to 0x8100.
- VLAN Priority Code Point (PCP)
- Drop Eligibility Indication (DEI)
- VLAN ID

In the second and third approaches of the demo (*Ingress Policing* and *Time Gating*), the SJA1105 must distinguish between the two flows, in order to prioritize them. To do so, it uses VLAN tags, specifically the PCP (priority) field.

The SJA1105 switch has 3 main stages in its packet processing pipeline:

- Ingress
- Forwarding
- Egress

On the ingress stage, the switch is configured to assign a default ("native") VLAN header on frames, based on their incoming port.

Based on the default VLAN tagging, the flows receive differentiated treatment:

- In the policing configuration, one of the flows is rate-limited on the ingress port
- In the scheduling configuration, similar rate-limiting effect is reached by each flow getting its own time slot allocated for the forwarding and egress stages

On the egress stage, the default VLAN tag is removed, so the connected hosts (Host 1, Host 2, Host 3) are oblivious to this VLAN tagging.

6.5 Host preparation

6.5.1 Host 2 (Ubuntu 16.04)

```
[ubuntu] $ sudo apt-get update
[ubuntu] $ sudo apt-get install iperf iperf3 \
       screen openssh-client openssh-server feedgnuplot expect
```

6.5.2 Host 1 (Ubuntu 16.04 or 14.04 or Windows)

This machine must only act as an iPerf client. If running Ubuntu, you install iPerf this way:

```
[ubuntu] $ sudo apt-get update
[ubuntu] $ sudo apt-get install iperf iperf3
```

If running Windows, you can download iPerf2 and iPerf3 binaries from the https://iperf.fr website.

6.5.3 LS1021ATSN Environment

You can now introduce the SD card into the LS1021ATSN board. On the first boot you will have to configure the U-Boot environment variables through the serial connection:

```
[ubuntu] $ screen /dev/ttyACM0 115200
Hit any key to stop autoboot:  0
=> boot
# OpenIL will auto-login as root
# This command should load the standard TSN switch configuration
$ sja1105-tool config default -f ls1021atsn
# You can exit screen by typing Ctrl-a - K
```

As soon as the board boots up, a DHCP server and an SSH server should start on the LS1021. It should auto-assign an IP of 172.15.0.1 and give out IP addresses to any clients that connect through ports marked as ETH2, ETH3, ETH4 or ETH5 on the chassis. After the SJA1105 switch is initialized by running the command above over serial, then connecting an Ethernet cable to one of those ports should also allow SSH access to the LS1021:

```
[ubuntu] $ ssh root@172.15.0.1
```

6.6 Troubleshooting

6.6.1 SSH

If you get an authentication error from SSH, it might be solved by forgetting the board's SSH keys:

```
[ubuntu] $ ssh-keygen -f "~/.ssh/known_hosts" -R 172.15.0.1
```

6.6.2 DHCP

To diagnose DHCP failure to assign IP addresses to clients, you can run tcpdump on both the Linux PC and the LS1021 (connected over serial). This way you can find out where are the packets stuck, or who and why is dropping them.

```
[ubuntu/openil] $ sudo tcpdump -i <eth-port> -e \
            "icmp or arp or port 67 or port 68"
```

On the LS1021, `<eth-port>` should be replaced with eth2, the port connected to the SJA1105 switch. On regular desktop GNU/Linux distributions (Ubuntu, Fedora), connections (including DHCP requests) are managed by the `NetworkManager` service. However, you can either assign a static IP in the 172.15.0.0/24 range manually, or you may invoke a DHCP client manually like this and see what is the status:

```
[ubuntu] $ sudo dhclient -v <eth-port>
```

If the GNU/Linux PC does send DHCP packets (DISCOVER and/or REQUEST) but these packets don't appear in the `tcpdump` on the LS1021, there may be a problem with the SJA1105 switch configuration. Here are some commands you can run to inspect it (from the LS1021):

```
[root@openil] $ watch sja1105-tool status port
# a nicer way to display the same info as above
[root@openil] $ get-port-status.sh
[root@openil] $ watch sja1105-tool status general
# a nicer way to display the same info as above
[root@openil] $ get-general-status.sh
[root@openil] $ tail -f /var/log/messages
# this is to load a known-working config
[root@openil] $ sja1105-tool config default -f ls1021atsn
```

6.6.3 Layer 2 and Layer 1

If there seems to be no L2 connection between the GNU/Linux PC and the TSN board, you can add a static ARP entry (for debugging) towards the LS1021 like this:

```
[ubuntu] $ sudo arp -s 172.15.0.1 \
                "<mac-of-eth2-on-ls1021>" dev <eth-port>
```

WARNING: The default configuration of the SJA1105 switch is in fixed-speed 1000Mbps mode. At the same time, the default configuration of the BCM5464R PHY chip is in auto-negotiation mode, advertising 1000Mbps mode.

If you wish to connect a client that only supports 100Mbps Ethernet, you first have to reconfigure the link speed of the ports:

```
[root@openil] $ sja1105-tool config modify -f mac[i] speed 2
# Here is the mapping between i (SJA1105 RGMII ports)
# and actual port labels on the chassis:
# Chassis ETH2: Switch port RGMII 1
# Chassis ETH3: Switch port RGMII 2
# Chassis ETH4: Switch port RGMII 3
# Chassis ETH5: Switch port RGMII 0
# To LS1021:    Switch port RGMII 4
```

Note that in this mode, only the SJA1105 switch was reconfigured for 100Mbps Ethernet link speed. The BCM5464R PHY is not affected by this change, and still (falsely) advertises auto-negotiation at 1000Mbps. This is a known issue.

6.7 Demo Preparation

Traffic generation for this demo can be performed in a multitude of ways. Here, we are showing standard iPerf2 in TCP mode plus iPerf3 in UDP mode, which gives some more connection statistics.

6.7.1 iPerf2 in TCP mode

6.7.1.1 Server

Host 2:

```
# Take note of the IP address given by the LS1021 DHCP server
[host2] $ ip addr show dev eth0
# Start listening on the default TCP port 5001
[host2] $ iperf -s
```

6.7.1.2 Clients

Host 1:

```
[host1] $ chmod +x ./host1-client.sh  # See Attached files
[host1] $ ./host1-client.sh
```

Host 3:

```
[host1] $ chmod +x ./host3-client.sh  # See Attached files
# Yes, this is correct, the host3-client script invokes
# iPerf client on Host 3 remotely from Host 1 (over ssh).
[host1] $ ./host3-client.sh
```

6.7.2 iPerf3 in UDP mode

In this mode, some more statistics can be gathered, which may be more relevant for the characterization of traffic flows:

- Jitter = (TRi – TRi-1) - (TSi – TSi-1) = variation of timestamp difference between consecutive packets, at sender vs at receiver
- Loss rate = iPerf places sequence numbers in each sent UDP datagram, and counts all packets between the "expected" and the "received" sequence number as lost (even if they are just reordered)
- Bandwidth no longer depends on the TCP congestion window size

On the other hand, UDP mode requires that these statistics be collected at the server side.

6.7.2.1 Servers

Host 2:

```
# Start two iPerf servers on Host 2
[host2] $ mkfifo bw_pipe lost_pipe jitter_pipe
[host2] $ unbuffer iperf3 -s -p 5201 -f m | \
              tee /dev/tty | awk -v host=host1 -f iperf.awk
[host2] $ unbuffer iperf3 -s -p 5202 -f m | \
              tee /dev/tty | awk -v host=host3 -f iperf.awk
```

The necessary scripts for these commands (`show` and `iperf.awk`) can be found in 6.7.3 Attached Files.

Although the commands above are enough for the two iPerf3 servers to receive traffic, you can also view the servers' statistics graphically with `feedgnuplot` on Host 2:

```
[host2] $ ./show bw_pipe 0 1000 "Bandwidth (Mbps)" \
                  "UDP throughput for iPerf flows towards Host 2"
[host2] $ ./show lost_pipe 0 100 "Percentage" \
                  "UDP Packet loss or reordering"
[host2] $ ./show jitter_pipe 0 1 "Jitter (ms)" \
                  "Jitter for UDP iPerf flows"
```

6.7.2.2 Clients

Host 1:

```
[host1] $ iperf3 -c <host-2-ip-address> -p 5201 -u -b 0 -t 100
```

Host 3:

```
[host1] $ ssh -tt root@172.15.0.1 \
         "iperf3 -c <host-2-ip-address> -p 5202 -u -b 0 -t 100"
```

6.7.3 Attached Files

show:

```
#!/bin/bash

pipe=$1
ymin=$2
ymax=$3
ylabel=$4
title=$5

if [[ $# -lt 5 ]]; then
        echo "Usage: $0 <pipe> <ymin> <ymax> <ylabel> <title>"
        exit
fi

exec cat "${pipe}" | tee /dev/tty | \
```

```
        feedgnuplot --stream 0.5 --domain --dataid --exit --lines \
        --ymin "${ymin}" --ymax "${ymax}" --autolegend \
        --style host1 'linewidth 2 linecolor rgb "blue"' \
        --style host2 'linewidth 2 linecolor rgb "green"' \
        --timefmt "%H:%M:%S" --set 'format x "%H:%M:%S"' \
        --xlen 30 --xlabel "Time" --ylabel "${ylabel}" \
        --title "${title}"
```

iperf.awk:

```
/Mbits\/sec/ {
        # Extract loss percentage from within brackets
        gsub(/[\(#\%\)]/, "", $(NF));
        print strftime("%T"), host, $(NF-5) > "bw_pipe";
        print strftime("%T"), host, $(NF-3) > "jitter_pipe";
        print strftime("%T"), host, $(NF)   > "lost_pipe";
        fflush("bw_pipe"); fflush("jitter_pipe"); fflush("lost_pipe");
}
```

host1-client.sh:

```
#!/bin/bash

host2_address=172.15.0.2 # Change this accordingly
# Run this command to start a TCP iPerf flow towards Host 2 for 100 seconds
unbuffer iperf -c ${host2_address} -i 0.5 -t 100 -f m | \
          unbuffer -p tee /dev/tty | \
          unbuffer -p awk '$NF == "Mbits/sec" {print $(NF-1); fflush();}' |
\
          feedgnuplot --stream 0.5 --lines --exit --ymin 0 --ymax 1000 \
                      --xlabel "Time (1/2 seconds)" \
                      --ylabel "Bandwidth (Mbps)" \
                      --title "iPerf from Host 1 to Host 2" \
                      --xlen 30 --style 0 \
                      'linewidth 2 linecolor rgb "green"'
```

```
# NOTE: If you do not get any output from the command above,
# remove the two occurences of "unbuffer -p ", leaving lines 6-7 like this:
#          tee /dev/tty | \
#          awk '$NF == "Mbits/sec" {print $(NF-1); fflush();}' | \
```

host3-client.sh:

```
#!/bin/bash

# Host 3 IP is static, normally should not need to change this
host3_address=172.15.0.1
host2_address=172.15.0.2 # Change this accordingly

# Run this command remotely to start a TCP iPerf flow
# towards Host 2 for 100 seconds
unbuffer ssh -tt root@${host3_address} \
          "iperf -c ${host2_address} -i 0.5 -t 100 -f m" | \
          unbuffer -p tee /dev/tty | \
          unbuffer -p awk '$NF == "Mbits/sec" {print $(NF-1); fflush();}' |
\
          feedgnuplot --stream 0.5 --lines --exit --ymin 0 \
                      --ymax 1000 --xlabel "Time (1/2 seconds)" \
                      --ylabel "Bandwidth (Mbps)" --title \
                      "iPerf from Host 3 to Host 2" --xlen 30 \
                      --style 0 'linewidth 2 linecolor rgb "orange"'
# NOTE: If you do not get any output from the command above,
# remove the two occurences of "unbuffer -p ", leaving lines 11-12 like
this:
#          tee /dev/tty | \
#          awk '$NF == "Mbits/sec" {print $(NF-1); fflush();}' | \
```

6.8 SJA1105 configuration walkthrough

6.8.1 Managing configurations with the sja1105-tool

The `sja1105-tool` is a Linux userspace application for configuring the SJA1105 TSN switch. The tool supports:

- Importing a configuration for the SJA1105 switch from an XML file
- Exporting the current SJA1105 configuration as an XML file
- Uploading the current SJA1105 configuration to the switch through its SPI interface
- Inspecting the current SJA1105 configuration
- On-the-fly modification of the current SJA1105 configuration through command line or scripting interface

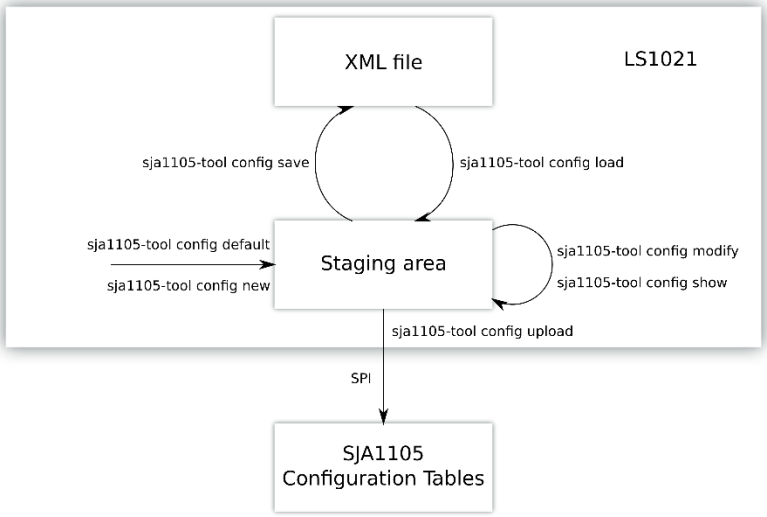

Figure 6-3. State machine view of the sja1105-tool commands and configuration formats

Because the physical SPI registers of the SJA1105 switch are write-only, a copy of them is kept in the staging area, which is effectively a file in the LS1021 OpenIL filesystem. The staging area keeps a binary image of the configuration to be uploaded over SPI using `sja1105-tool config upload`, and which can also be read back by the user with `sja1105-tool config show`.

More documentation on the sja1105-tool is distributed as man pages along with the source code:

```
[ubuntu:sja1105-tool] $ cd docs/man

# General command overview
[ubuntu:man] $ man -l ./sja1105-tool.1

# Detailed usage of sja1105-tool config
[ubuntu:man] $ man -l ./sja1105-tool-config.1

# Detailed usage of sja1105-tool status
[ubuntu:man] $ man -l ./sja1105-tool-status.1

# Detailed usage of sja1105-tool reset

[ubuntu:man] $ man -l ./sja1105-tool-reset.1

# File format for sja1105-tool configuration
[ubuntu:man] $ man -l ./sja1105-conf.5
```

```
# File format for XML switch configuration tables
[ubuntu:man] $ man -l ./sja1105-tool-config-format.5
```

6.8.2 Standard configuration

6.8.2.1 Commands

Host 3 (LS1021): prepare the SJA1105 switch with a sane, default built-in config:

```
[root@openil] $ sja1105-tool config default ls1021atsn
[root@openil] $ sja1105-tool config upload
[root@openil] $ sja1105-tool config save standard.xml
```

6.8.2.2 Ingress Policer

The L2 Ingress Policer inside the SJA1105 is implemented as a Token Bucket:

- Bucket max size (also known as burst size) is called SMAX (maximum is 0xFFFF)
- Bucket refill speed is RATE bytes per second (up to a maximum of 64000)
- Each ingress packet removes from the bucket a number of tokens equal to its length in bytes
- Can also police traffic based on maximum frame size

The Policing table has 45 entries:

- One for each Ingress Port x VLAN PRIO (5 x 8)
- One for Broadcast Traffic coming from each Ingress Port (5)

In the standard configuration, the L2 Ingress Policer is "**deactivated**". This means that RATE and SMAX are set to maximum (0xFFFF, 0xFA00) for all entries, so rate limiting can never occur at the maximum ingress rate of 1000Mbps.

This can be seen by looking at the `l2-policing-table` entries:

```
[root@openil] $ sja1105-tool conf show l2-pol
```

6.8.2.3 Native VLAN Assignments

These are configurable through the MAC Configuration Table (5 entries, one per port). Native VLAN tags are added only if the switch received the packets as untagged. The user can select whether the switch includes the VLAN tags in the egress packet or not. VLAN priorities are taken into consideration for the L2 Forwarding stage.

In the standard configuration, all ingress ports get by default VLAN priority 0 (best-effort) and all egress ports remove VLAN tags from packets.

6.8.3 Queuing Diagram

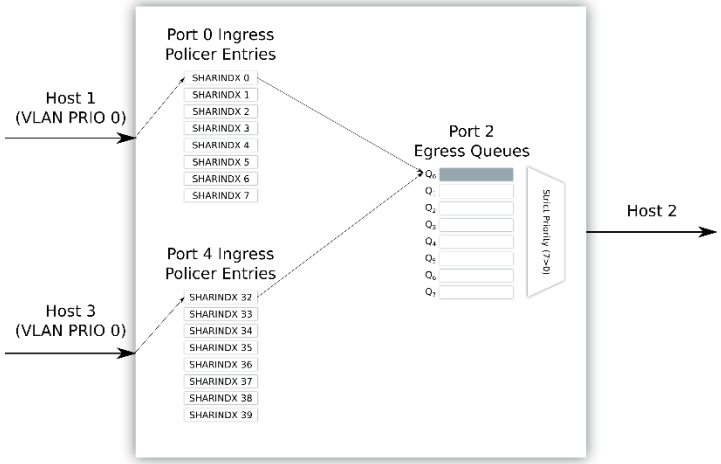

Figure 6-4. All traffic gets assigned to VLAN priority 0, causing contention on the same egress queue

6.8.3.1 Results

- Individually, both Host 3 and Host 1 get around 950Mbps
- Ran at the same time, Host 3 and Host 1 bandwidths oscillate
  - First Host 3 gets very low bandwidth (between 1 and 100Mbps)
  - After about 30s, the TCP congestion control algorithms reach to a steadier state
  - Bandwidth allocation is suboptimal (sum of the bandwidths is much lower than 1000Mbps)

6.8.4 Prioritizing configuration

We can assign native VLAN priority 3 to Host 1 and priority 7 to Host 3. This is done on a per-ingress port basis (Host 1 -> Port 0, Host 3 -> Port 4).

On the egress port 2, if Host 3's queue is not empty, the switch will always prefer to send packets from that instead of Host 1's queue.

6.8.4.1 Commands

```
[root@openil] $ sja1105-tool conf default ls1021atsn
[root@openil] $ sja1105-tool conf mod mac-config[0] vlanprio 3
[root@openil] $ sja1105-tool conf mod mac-config[4] vlanprio 7
[root@openil] $ sja1105-tool conf save prioritizing.xml
[root@openil] $ sja1105-tool conf upload
```

6.8.4.2 Native VLAN assignments

```
[root@openil] $ sja1105-tool conf show mac-configuration-table
MAC Configuration Table: 5 entries
|| Entry 0:                  || Entry 1:                  ||
|| VLANPRIO   0x3            || VLANPRIO   0x0            ||
|| VLANID     0x0            || VLANID     0x0            ||
||                           ||                           ||
|| Entry 2:                  || Entry 3:                  ||
|| VLANPRIO   0x0            || VLANPRIO   0x0            ||
|| VLANID     0x0            || VLANID     0x0            ||
||                           ||                           ||
|| Entry 4:                  ||
|| VLANPRIO   0x7            ||
|| VLANID     0x0            ||
```

6.8.4.3 Queuing Diagram

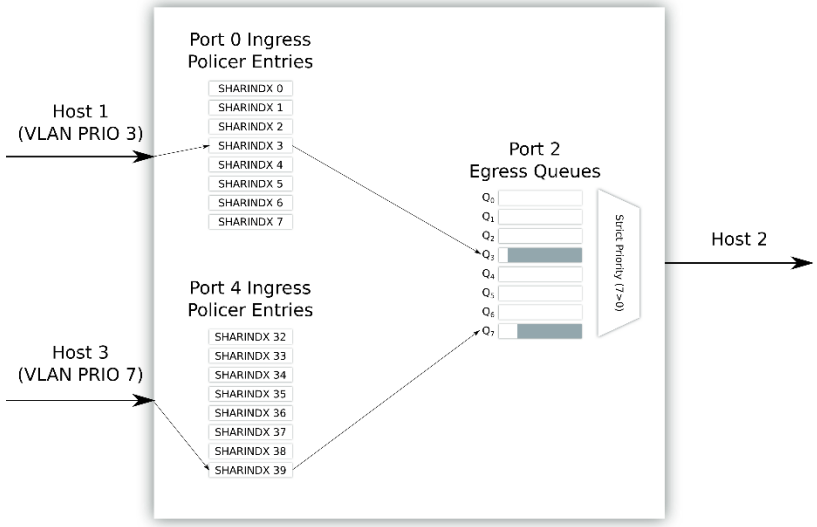

Figure 6-5. Host 3 traffic is assigned a higher VLAN priority than Host 1. On egress, Host 3 will dominate because of the strict priority queuing discipline of the switch.

6.8.4.4 Results

- Host 3 has higher priority, it always gets its 950Mbps
- Host 1 can only get the remaining bandwidth up to 1000Mbps, which is typically very low (under 50Mbps).

6.8.5 Policing configuration

Based on the prioritizing config, we can apply rate limiting on Host 3, so Host 1 is able to get more than 400Mbps. The choice of rate-limiting Host 3 instead of Host 1 is intentional, since Host 3 has a higher VLAN priority and will obtain its rate-limited slice of the bandwidth anyway.

6.8.5.1 Commands

```
# Set the RATE value for Host 3 to (0.6 * 64000) = 38400, for 600Mbps
[root@openil] $ sja1105-tool config load prioritizing.xml
[root@openil] $ sja1105-tool config mod l2-policing-table[39] rate 38400
[root@openil] $ sja1105-tool config save policing.xml
[root@openil] $ sja1105-tool config upload
```

6.8.5.2 Queuing diagram

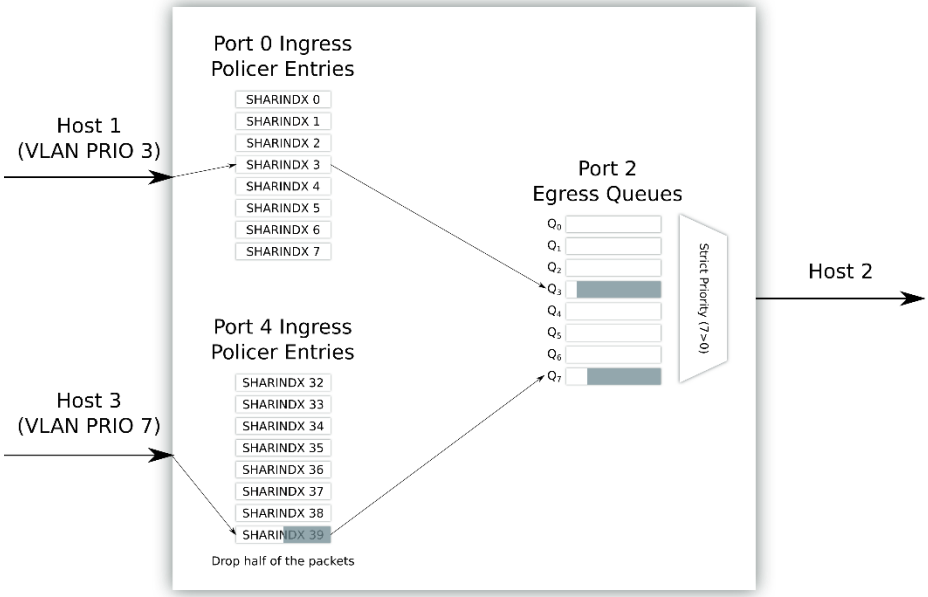

Figure 6-6. Same queues as with the prioritizing configuration. The higher-priority traffic is rate-limited to allow the lower-priority traffic to use more of the remaining space.

6.8.5.3 Results

- Through a combination of prioritization and policing, we can obtain the desired bandwidth allocation for both Host 1 and Host 3 (400-600)
- This is done by dropping part of the packets from Host 3 (which may not always be desirable)

6.8.6 Scheduling configuration

The Time-Aware Scheduler works by following the guidelines in 802.1Qbv:

- The 5 Egress Ports each have 8 Gates, which can be open or closed
- Each Gate has 1 Queue associated with it
- Whenever a Gate is open, packets from that Queue can be sent out the wire
- An internal clock generates ticks each 200ns
- At each tick, a new time slot can be created, where some Gates can be opened and some can be closed

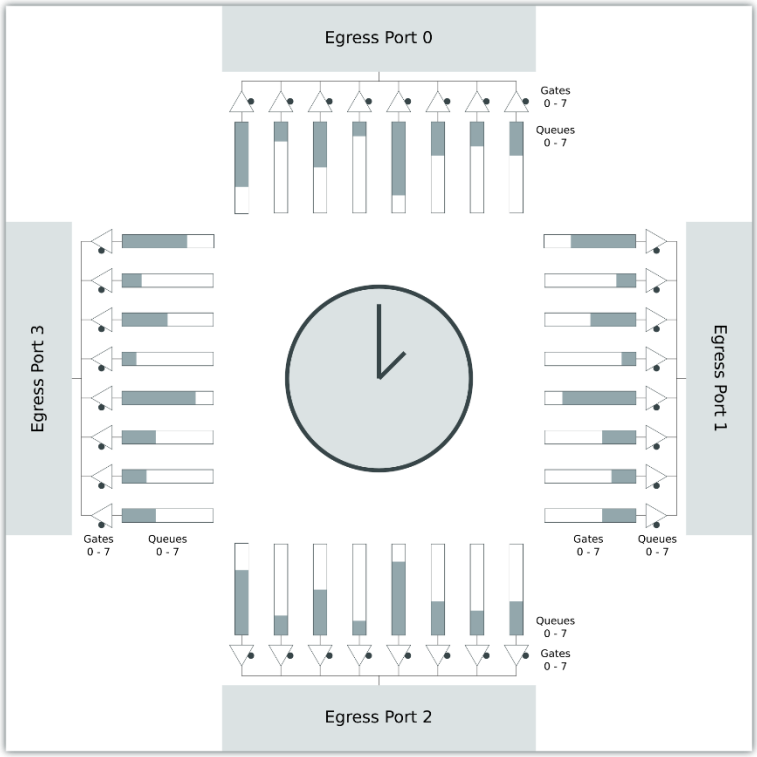

Figure 6-7. Structure of the Time Aware Scheduler

The user defines how many clock ticks each time slot (called subschedule) takes, and also which flows (identified by their VLAN PRIO bits) are allowed to dequeue packets on each time slot. Once the Time-Aware Scheduler goes through each time slot (subschedule) in a round-robin fashion, it starts over again periodically. A complete period of subschedules is called a schedule.

In the third approach of the demo, the Time Aware Scheduler is active for Egress Port 2. This is the link towards Host 2, where the contention between Flow 1 and Flow 2 happens. The SJA1105 switch is configured to create a subschedule for VLAN PRIO 0 and one for PRIO 3, each having an equal amount of time. This way, Flow 1 is completely isolated from Flow 2, and there is minimal interference between the two, which allows better utilization of bandwidth.

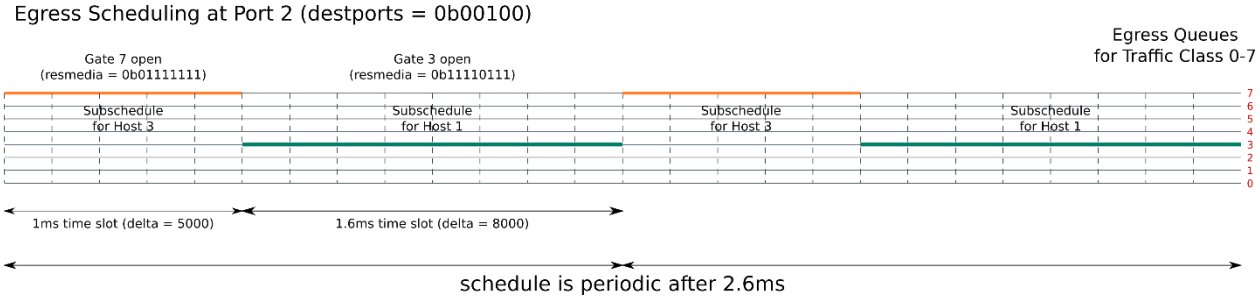

Figure 6-8. Time gating for Host 1 and Host 3, viewed on the time axis

6.8.6.1 Commands

The Schedule Table contains the definitions of all the subschedules:

- What egress ports is the subschedule active on
- Which gates (egress queues for traffic classes) should be open and which should close
- The duration of the subschedule, in 200ns increments

The Schedule Table does **NOT** define how the subschedules are linked together.

SJA1105 allows for a total of 8 simultaneous schedules (for this demo we shall only use 1).

```
[root@openil] $ sja1105-tool conf load prioritizing.xml
# Create two subschedules
[root@openil] $ sja1105-tool conf mod schedule-table entry-count 2
# Set both subschedules to be active on egress Port 2 (towards Host 2)
[root@openil] $ for i in 0 1; do sja1105-tool conf mod schedule-table[$i] \
                                 destports 0b00100; done
[root@openil] $ for i in 0 1; do sja1105-tool conf mod schedule-table[$i] \
                                 resmedia_en 1; done
# Configure first subschedule to keep only gate 7 open (coming from Host 3)
[root@openil] $ sja1105-tool conf mod schedule-table[0] resmedia 0b01111111
# Configure first subschedule to keep only gate 3 open (coming from Host 1)
[root@openil] $ sja1105-tool conf mod schedule-table[1] resmedia 0b11110111
# Allocate a 50000*200ns = 10ms time slot for first subschedule (Host 3)
[root@openil] $ sja1105-tool conf mod schedule-table[0] delta 50000
# Allocate a 80000*200ns = 16ms time slot for second subschedule (Host 1)
[root@openil] $ sja1105-tool conf mod schedule-table[1] delta 80000
# Check that we configured the schedule-table properly
[root@openil] $ sja1105-tool conf show schedule-table
```

```
Schedule Table: 2 entries
|| Entry 0:                        || Entry 1:                        ||
|| WINSTINDEX  0x0                 || WINSTINDEX  0x0                 ||
|| WINEND      0x0                 || WINEND      0x0                 ||
|| WINST       0x0                 || WINST       0x0                 ||
|| DESTPORTS   0x4                 || DESTPORTS   0x4                 ||
|| SETVALID    0x0                 || SETVALID    0x0                 ||
|| TXEN        0x0                 || TXEN        0x0                 ||
|| RESMEDIA_EN 0x1                 || RESMEDIA_EN 0x1                 ||
|| RESMEDIA    0x7F                || RESMEDIA    0xF7                ||
|| VLINDEX     0x0                 || VLINDEX     0x0                 ||
|| DELTA       0xC350              || DELTA       0x13880             ||
||                                  ||                                 ||
```

Each schedule has a starting point and an ending point, defined as indices to subschedules from the Schedule Table:

- The starting point is defined in the Schedule Entry Points table
- The ending point is defined in the Schedule Parameters table

Schedule Entry Points table:

```
# Create an entry in the Schedule Entry Points table
[root@openil] $ sja1105-tool conf mod \
                schedule-entry-points-table entry-count 1
# The default configuration will work:
# delta="0x0" address="0x0" => Start subschedule 0 at time 0

# Check if correct
[root@openil] $ sja1105-tool conf show schedule-entry-points-table
Schedule Entry Points Table: 1 entries
|| Entry 0:                        ||
|| SUBSCHINDX 0x0                  ||
|| DELTA      0x0                  ||
|| ADDRESS    0x0                  ||
||                                  ||
```

Schedule Parameters Table:

```
[root@openil] $ sja1105-tool conf mod schedule-parameters-table entry-count
1
[root@openil] $ sja1105-tool conf mod schedule-parameters-table[0] \
                                    subscheind "[1 1 1 1 1 1 1 1]"
# The Schedule Parameters table has a single entry.
# For each schedule i in (0..7), subscheind[i] is the last
# subschedule of that schedule. To be exact, it is an index
# of that subschedule in the Schedule table.
# Here we only care about the first "1" since only that refers
# to our schedule (0). All the other schedules must be configured
# such that they start and end at subschedule 1 (i.e. are disabled).

# Check if correct
[root@openil] $ sja1105-tool conf show schedule-parameters-table
Schedule Parameters Table: 1 entries
|| Entry 0:                                          ||
|| SUBSCHEIND [0x1 0x1 0x1 0x1 0x1 0x1 0x1 0x1 ]     ||
||                                                   ||
```

Schedule Entry Points Parameters table:

```
[root@openil] $ sja1105-tool conf mod \
                schedule-entry-points-parameters-table entry-count 1
# Set the scheduling engine to run in standalone mode
# (no clock correction/synchronization)
# Other options include SAE AS6802 or internal PTP clock
[root@openil] $ sja1105-tool conf mod \
                schedule-entry-points-parameters-table[0] clksrc 1

# Check if correct
[root@ls1021atsn ~] # sja1105-tool conf show \
                      schedule-entry-points-parameters-table
Schedule Entry Points Parameters Table: 1 entries
|| Entry 0:                       ||
|| CLKSRC    0x1                  ||
```

```
|| ACTSUBSCH 0x0                    ||
||                                  ||
# The Schedule Entry Points Parameters table contains
# only one entry, which is global for all schedules.
```

Wrap everything up:

```
[root@openil] $ sja1105-tool config save scheduling.xml
[root@openil] $ sja1105-tool config upload
```

6.8.6.2 Results

- Regardless of being run separately or simultaneously, Host 1 gets around 560Mbps and Host 3 gets around 360Mbps
- The bandwidth ratio is the same as the ratio between time slot durations (delta – 1ms and 1.6ms):
- 360/(360+560) = 5000/(5000+8000) ~= 0.4
- The Time Aware Scheduler of the SJA1105 allows for finer-grained control over bandwidth allocation

6.9 UDP iPerf3 notes

For the alternative UDP iPerf3 traffic generation mode, client connections can also be started from Windows (Host 1). In this case, only the iPerf server (Host 2) needs to run GNU/Linux for the Gnuplot real-time view of bandwidths.

Because iPerf3 refuses simultaneous connections to the same server, we need to run 2 separate servers (port 5201 and 5202) on Host 2.

The reasons for the packet drop reported at the iPerf3 server might include:

- Packets dropped by the Ingress Policer of the SJA1105: `sja1105-tool status port 4` (counter **N_POLERR**) – seen in policing.xml
- Tail drop at egress queue towards Host 2: `sja1105-tool status port 2` (counter **N_QFULL**) – seen in standard.xml and scheduling.xml
- Dropped at protocol level by the iPerf3 server
  - Out of order packets are reported as lost in UDP mode
  - Server receive timeouts – seen in scheduling.xml

In scheduling.xml, packets coming from Host 3 can arrive on the SJA1105:

- Early: if they were sent in the time slot where Gate 7 was open on SJA1105 (in-band)

- Late: if they were sent outside of that time slot (out-of-band)

iPerf3 server experiences timeouts and declares out-of-band packets as lost. This shows the opportunity to combine the traffic scheduling done by the switch with a more complex queuing discipline on the traffic senders, that is aware of the switch's time slots and only sends packets in-band.

WARNING: For the original time slot durations (1ms and 1.6ms), running both Host 1 and Host 3 at the same time only causes more congestion (which leads to *more timeouts*)

- Both iPerf servers (5201 and 5202) experience high packet loss
- Egress scheduler doesn't interrupt current sent frame when time slot expires
- 1ms is not enough guard time to completely separate the flows

If time slots are increased 10x (10ms, 16ms), this problem disappears:

```
[root@openil] $ sja1105-tool config modify schedule-table[0] delta 50000
[root@openil] $ sja1105-tool config modify schedule-table[1] delta 80000
[root@openil] $ sja1105-tool config upload
```

6.10 Final word

After running the steps in this guide, inside the OpenIL home directory there will be 4 XML files:

- standard.xml
- prioritizing.xml
- policing.xml
- scheduling.xml

The walkthrough must be followed step-by-step only once. Afterwards, a specific configuration can be loaded like this:

```
[root@openil] $ sja1105-tool config load standard.xml
[root@openil] $ sja1105-tool config upload
```

7 3-Board TSN Demo

7.1 Objectives

- Demonstrate per-stream filtering and policing
- Remote configuration over NETCONF
- Show a more advanced usage scenario, with cascaded rate-limiters and complex traffic identification

7.2 TTEthernet (SAE AS6802) features

7.2.1 Virtual Links

Virtual Links provide a way for the SJA1105 switch to identify and route streams of traffic based on more advanced criteria than just destination MAC. As such, a switch deeply embedded in the core of a network can distinguish between multiple flows headed towards the same destination.

Virtual Links can be unicast or multicast (but must have a single traffic source). They are especially useful in industrial networks, where all routes and MAC addresses are static and known a-priori.

7.2.2 Virtual Link Selection Criteria

There are two different lookup formats (criteria) for VL selection:

- If `general-parameters-table.vllupformat == 1`, then selection is done similarly to the AFDX/ARINC664 standard:

```
# Check if a frame matches Virtual Link Lookup entry i
foreach (i in vl-lookup-table.entry-count) {
    if ((dest_mac[47:16] & general-parameters-table.vlmask) ==
                           general-parameters-table.vlmarker) &&
        (dest_mac[15:0] == vl-lookup-table[i].vlid) &&
        (src_port       == vl-lookup-table[i].port)) {
            return true;
}
```

- If `general-parameters-table.vllupformat == 0`, selection is done like this:

```
# Check if a frame matches Virtual Link Lookup entry i
foreach (i in vl-lookup-table.entry-count) {
    if ((dest_mac          == vl-lookup-table[i].macaddr) &&
        (ingress_vlan_id   == vl-lookup-table[i].vlanid) &&
        (src_port          == vl-lookup-table[i].port) &&
        (ingress_vlan_prio == vl-lookup-table[i].vlanprior) {
            return true;
}
```

If using lookup format 0 (non-ARINC664), then VLs can further be classified as critical or not.

If `vl-lookup-table.iscritical == 1` then the VL can be:

- Rate Constrained, if `vl-policing-table.type == 1`
- Time Triggered, if `vl-policing-table.type == 0`

The VL Priorities are: Time Triggered VL > Rate Constrained VL > Best Effort Traffic

7.2.3 Rate-Constrained Virtual Links

Policing ensures containment of faults caused by end systems.

Deterministic behavior in an industrial network is a joint work between end systems and switches.

- End-Systems perform traffic **shaping** and Integrity Checking on each VL
- Switches performs traffic **policing** on each VL

Flows can be policed based on **Bandwidth Allocation Gap** (BAG) and **Jitter.**

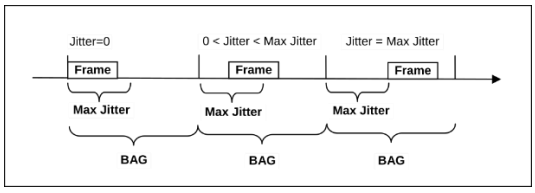

Figure 7-1. Policing on Rate-Constrained Virtual Links

Specifically, the VL Policer will accept ingress traffic at a maximum rate of $\frac{10000}{vl-policing-table.bag}$ packets per second. It will also drop packets which have a jitter higher than $vl-policing-table.jitter \times 10\mu s$.

7.2.4 Time-Triggered Virtual Links

Time-Triggered messages are treated as highest-priority by the SJA1105 switch. They are linked with the Schedule Table. For best-effort traffic, the mechanism of output triggering is through `resmedia`. For time-triggered VL's, it is through `txen`.

For a specific time slot (defined as a Schedule Table entry), when `txen` is set, then the time-triggered VL number `vlindex` is scheduled for output, on the egress priority queue that it has configured in `vl-forwarding-table.priority`. If the bit corresponding to `vl-forwarding-table[vlindex].priority` is also set as part of the `resmedia` bits of this entry, the time-triggered VLs always have higher egress priority over both rate-constrained and best-effort traffic, even in the same priority queue.

Currently the SJA1105 features for time-triggered VLs are not entirely exported to software.

7.2.5 Example usage of Virtual Links

In the current 3-board TSN demo, Virtual Links are used on Board 1, in both the policing and scheduling setup, to differentiate between flows F1 and F4. Both these flows originate on Host 1 and are seen by the SJA1105 switch on Board 1 as coming from port 4.

Differentiation (and thus different VLAN priority assignment) is done based on destination MAC address (which is Ubuntu for F1 and Board 3 for F4).

For the exact code that handles VLAN priority remapping, see the Attached Files chapter.

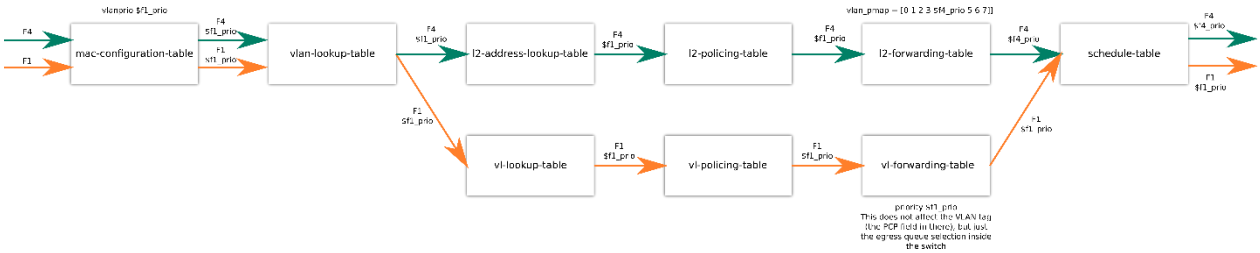

Figure 7-2. Example usage of Virtual Links to differentiate between F1 and F4 in the scheduling configuration on Board 1.

7.3 Network topology

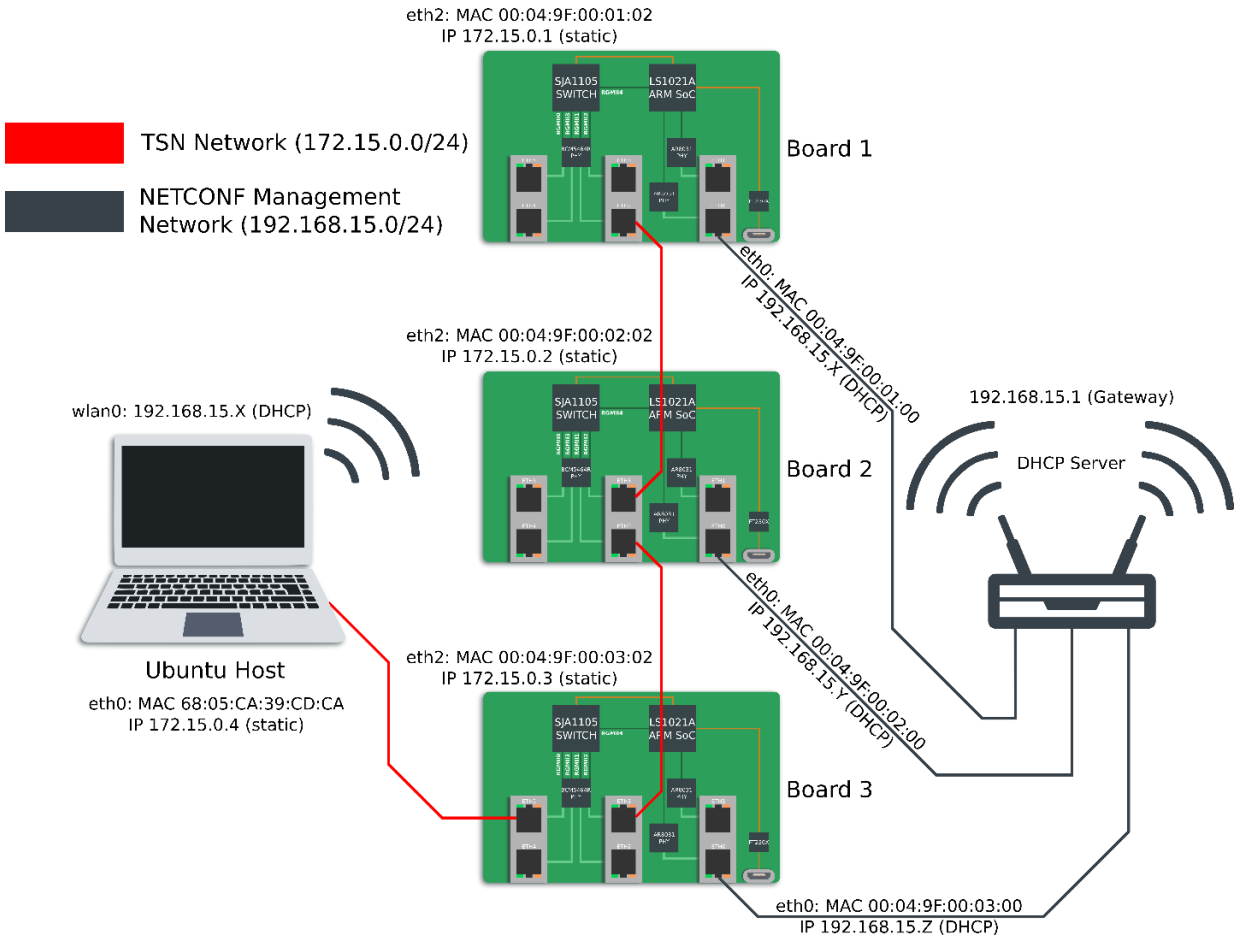

Figure 7-3. 3-board TSN network topology

7.4 Required parts

- 3 LS1021ATSN boards
- 1 PC or laptop running a GNU/Linux distribution (Ubuntu, CentOS etc), and having two separate network interfaces (either 1 Ethernet and 1 Wi-Fi, or 2 Ethernet ports)
- Either 1 Wi-Fi router or an off-the-shelf 4-port switch
- 6 or 7 Ethernet cables (depending if using Wi-Fi or not)
- 3 microUSB cables for serial access to TSN boards

7.5 Host preparation

7.5.1 Ubuntu PC

MAC address is assumed to be AA:BB:CC:DD:EE:FF. You can either adapt the scripts to match yours, or spoof your MAC address using NetworkManager:

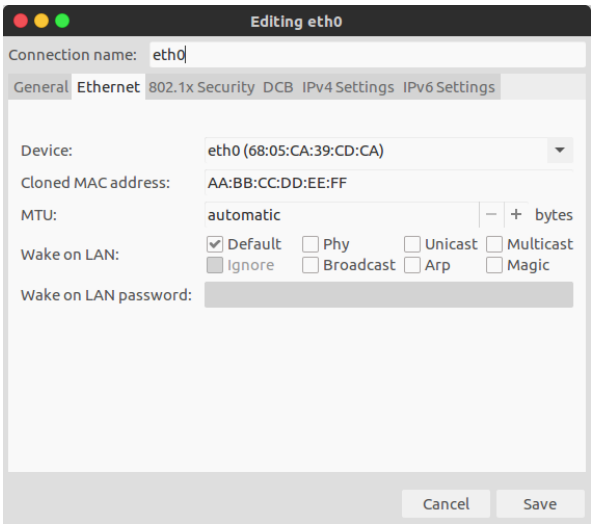

Figure 7-4. MAC address spoofing using GNOME NetworkManager

IP address must be assigned statically to 172.15.0.4.

```
[ubuntu] $ sudo ip addr link set dev eth0 down
[ubuntu] $ sudo ip addr flush dev eth0
[ubuntu] $ sudo ip addr add dev eth0 172.15.0.4/24
[ubuntu] $ sudo ip addr link set dev eth0 up
[ubuntu] $ sudo ip addr show dev eth0
```

This can also be done using NetworkManager:

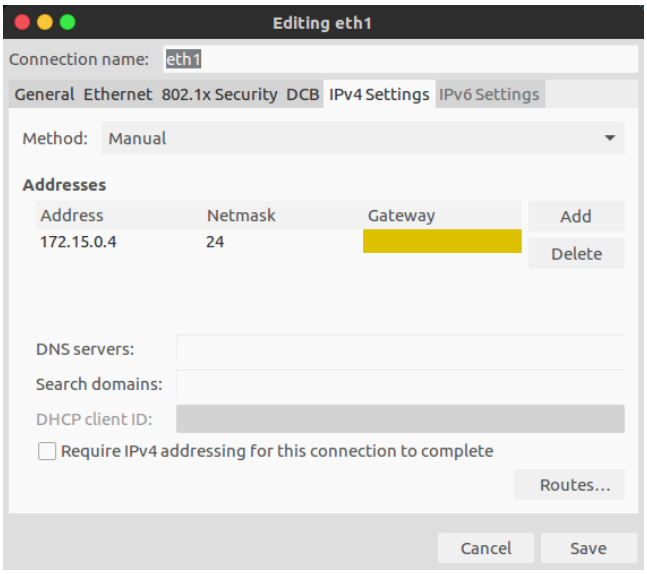

Figure 7-5. Static IP assignment using GNOME NetworkManager

After pressing the "Routes..." button, please make sure that the "Use connection only for resources on its network" option is checked.

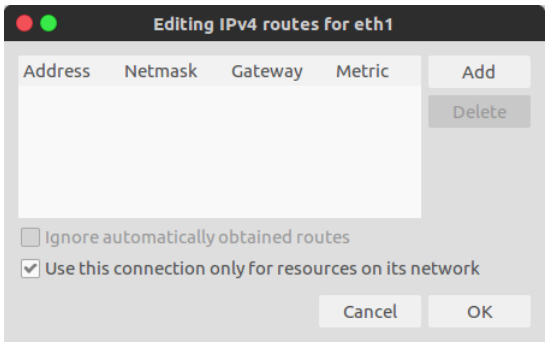

Figure 7-6. Tell NetworkManager not to use this interface as a gateway for general Internet traffic.

The other network interface (either `eth0` or `wlan0`) is expected to be connected through DHCP to the 192.168.15.0/24 network, and able to ping Board 1, Board 2 and Board 3.

```
[ubuntu] $ sudo arp-scan -l --interface [eth0 | wlan0]
Interface: [eth0 | wlan0], datalink type: EN10MB (Ethernet)
Starting arp-scan 1.8.1 with 256 hosts (http://www.nta-
monitor.com/tools/arp-scan/)
192.168.15.11   00:04:9f:00:01:00       Freescale Semiconductor
192.168.15.12   00:04:9f:00:02:00       Freescale Semiconductor
192.168.15.13   00:04:9f:00:03:00       Freescale Semiconductor
```

It is also a good idea to delete the list of known SSH servers that Ubuntu previously connected to:

```
# Avoid NETCONF connection issues
[ubuntu] $ rm -f ~/.ssh/known_hosts
```

7.5.2 LS1021ATSN Boards

7.5.2.1 MAC

Board 1:

```
setenv ethaddr  00:04:9F:00:01:00
setenv eth1addr 00:04:9F:00:01:01
setenv eth2addr 00:04:9F:00:01:02
saveenv
boot
```

Board 2:

```
setenv ethaddr  00:04:9F:00:02:00
setenv eth1addr 00:04:9F:00:02:01
setenv eth2addr 00:04:9F:00:02:02
saveenv
boot
```

Board 3:

```
setenv ethaddr  00:04:9F:00:03:00
setenv eth1addr 00:04:9F:00:03:01
setenv eth2addr 00:04:9F:00:03:02
saveenv
boot
```

7.5.2.2 IP

Board 1:

```
[openil] $ cat << EOF > /etc/network/interfaces
auto lo
```

```
iface lo inet loopback
        mtu 65535
auto eth2
iface eth2 inet static
        address 172.15.0.1
        netmask 255.255.255.0
auto eth0
iface eth0 inet dhcp
EOF
[openil] $ echo "board1" > /etc/hostname
[openil] $ cat << EOF > /etc/hosts
127.0.0.1       localhost
172.15.0.1      board1
172.15.0.2      board2
172.15.0.3      board3
172.15.0.4      ubuntu
EOF
[openil] $ reboot
```

Board 2:

```
[openil] $ cat << EOF > /etc/network/interfaces
auto lo
iface lo inet loopback
        mtu 65535
auto eth2
iface eth2 inet static
        address 172.15.0.2
        netmask 255.255.255.0
auto eth0
iface eth0 inet dhcp
EOF
[openil] $ echo "board2" > /etc/hostname
[openil] $ cat << EOF > /etc/hosts
127.0.0.1       localhost
```

```
172.15.0.1      board1
172.15.0.2      board2
172.15.0.3      board3
172.15.0.4      ubuntu
EOF
[openil] $ reboot
```

Board 3:

```
[openil] $ cat << EOF > /etc/network/interfaces
auto lo
iface lo inet loopback
        mtu 65535
auto eth2
iface eth2 inet static
        address 172.15.0.3
        netmask 255.255.255.0
auto eth0
iface eth0 inet dhcp
EOF
[openil] $ echo "board3" > /etc/hostname
[openil] $ cat << EOF > /etc/hosts
127.0.0.1       localhost
172.15.0.1      board1
172.15.0.2      board2
172.15.0.3      board3
172.15.0.4      ubuntu
EOF
[openil] $ reboot
```

7.5.2.3 Disable DHCP server

```
[root@board{1,2,3}] $ /etc/init.d/S80dhcp-server stop
[root@board{1,2,3}] $ mv /etc/init.d/S80dhcp-server /etc/init.d/dhcp-server-not-used
```

7.6 Traffic flows

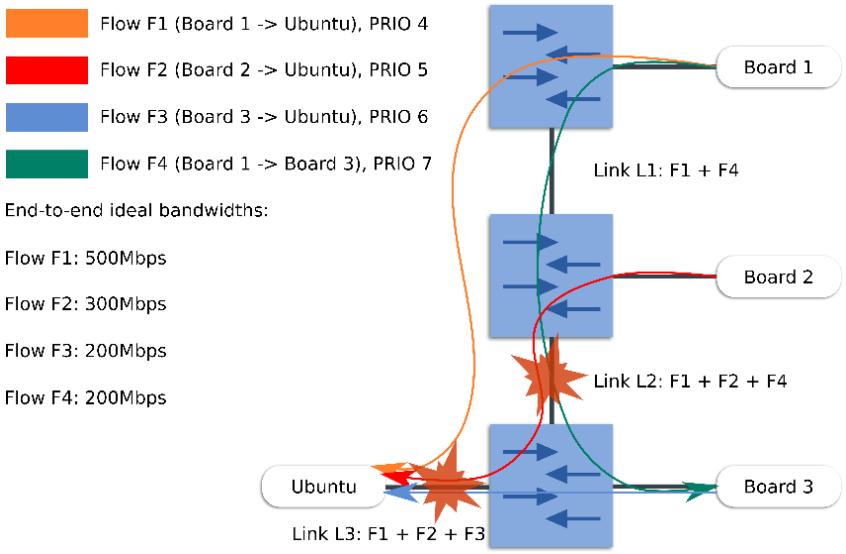

Figure 7-7. Diagram of traffic flows. On link L2 (between board2 and board3) there is contention between flows F1, F2 and F4. On link L3 (board3 towards Ubuntu) there is contention between F1, F2 and F3.

The purpose of this demo is to show how to control resource usage in an environment closer to real life, with multiple end systems (here we are considering the LS1021A SoC on Board 1, Board 2 and Board 3, as well as the Ubuntu PC connected to Board 3), as well as multiple TSN switches (the SJA1105 switches present in Board 1, Board 2 and Board 3).

The limits for traffic flows F1, F2, F3 and F4 have been chosen as such because of the inherent 1000Mbps limitation of links L2 and L3:

- $F1 + F2 + F4 \leq 1000\text{Mbps}$
- $F1 + F2 + F3 \leq 1000\text{Mbps}$

We also require that F1 is limited to 500Mbps.

VLAN priorities 4 for F1, 5 for F2, 6 for F3 and 7 for F4 have been intentionally chosen such that the highest bandwidth traffic (F1) gets the lowest VLAN priority, and the flows rate-limited at the lowest value (F3 and F4) get the highest VLAN priorities (6 and 7).

7.6.1 Policing configuration

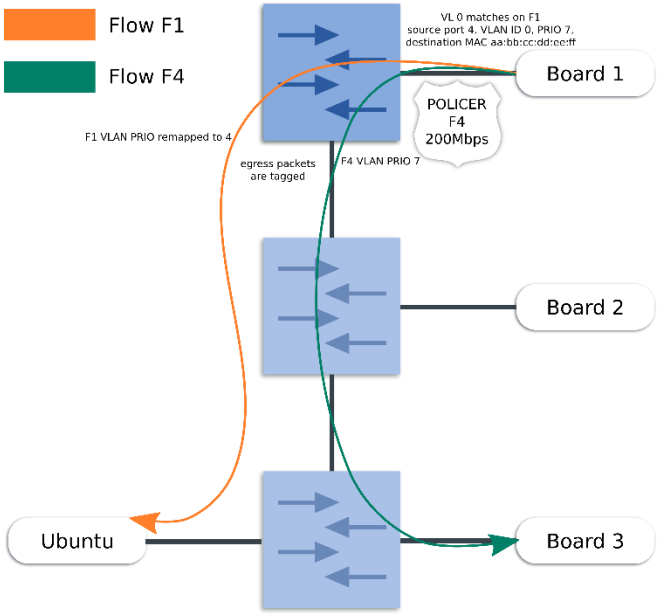

Figure 7-8. Actions to be done on Board 1: using Virtual Links, tag F1 with VLAN PRIO 4 and F4 with PRIO 7. Rate-limit F4 to 200Mbps.

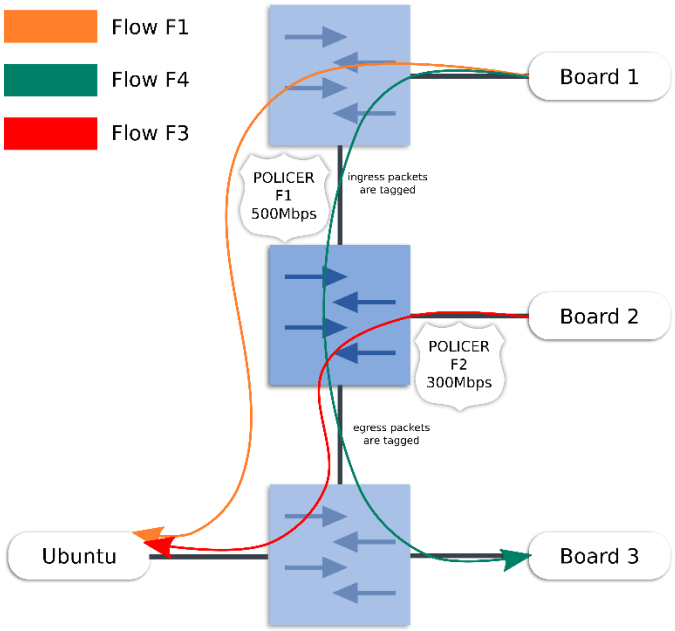

Figure 7-9. Actions to be done on Board 2: rate-limit F1 to 500Mbps and F2 to 300Mbps

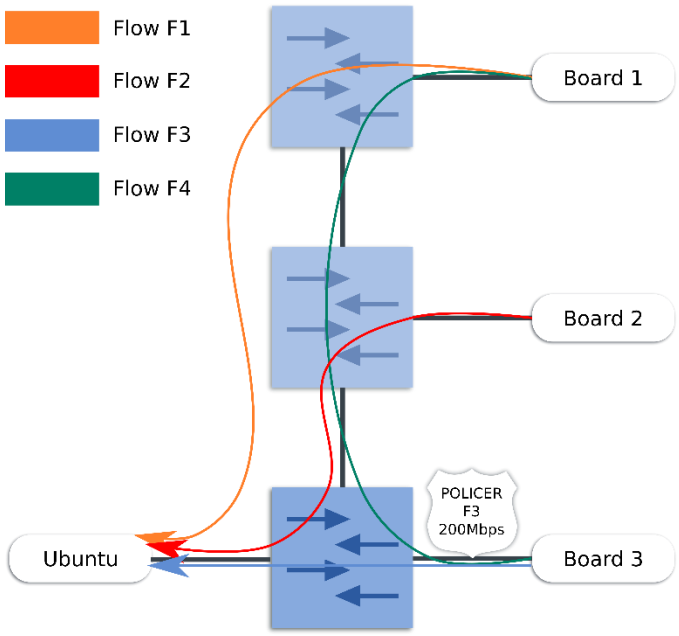

Figure 7-10. Actions to be done on Board 3: rate-limit F3 to 200Mbps.

7.6.1.1 Creating the configuration

From 0

Attached files copy `board1-policing.sh` on Board 1, `board2-policing.sh` on Board 2 and `board3-policing.sh` on Board 3. Also copy `definitions.sh` on all boards (this file can allow modification of some demo-related parameters). Then run these commands:

Board1:

```
[root@board1] $ chmod +x board1-policing.sh
[root@board1] $ ./board1-policing.sh
```

Board2:

```
[root@board1] $ chmod +x board2-policing.sh
[root@board2] $ ./board2-policing.sh
```

Board3:

```
[root@board1] $ chmod +x board3-policing.sh
[root@board3] $ ./board3-policing.sh
```

At the end, for each board there should appear a new configuration in `/etc/sja1105/`.

WARNING: If you are trying to change the VLAN Priorities in `definitions.sh`, the `board1-policing.sh` script does not take `f1_prio` and `f4_prio` in consideration for `vlan_pmap` remapping (line 34). This is a known issue just for the `board1-policing.sh` script, so please modify the `vlan_pmap` array manually if changing VLAN priorities.

7.6.2 Scheduling configuration

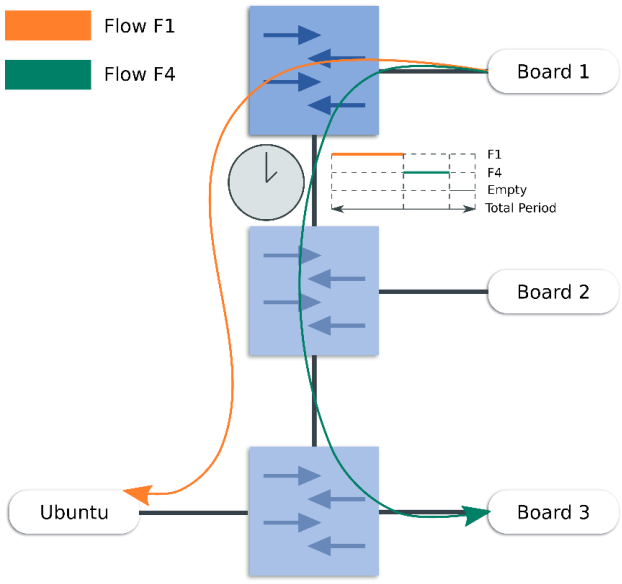

Figure 7-11. Actions to be performed on Board 1: Break up 10ms time intervals in such way that F1 traffic is only scheduled on egress L1 link 50% of the time (5ms) and F4 traffic 20% of the time (2ms). This results in rate-limiting F1 at 500Mbps and F4 at 200Mbps.

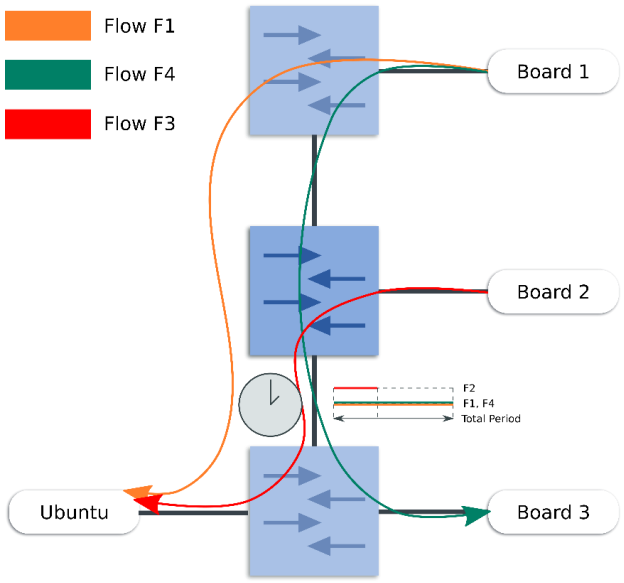

Figure 7-12. Actions to be performed on Board 2: Break up 10ms time intervals in such way that F2 traffic is only forwarded on egress L2 link 30% of the time. Give F2 traffic higher priority, to guarantee the bandwidth. Allow F1 and F4 traffic on egress at all times, but treat as background traffic. This is not to incur additional rate limiting.

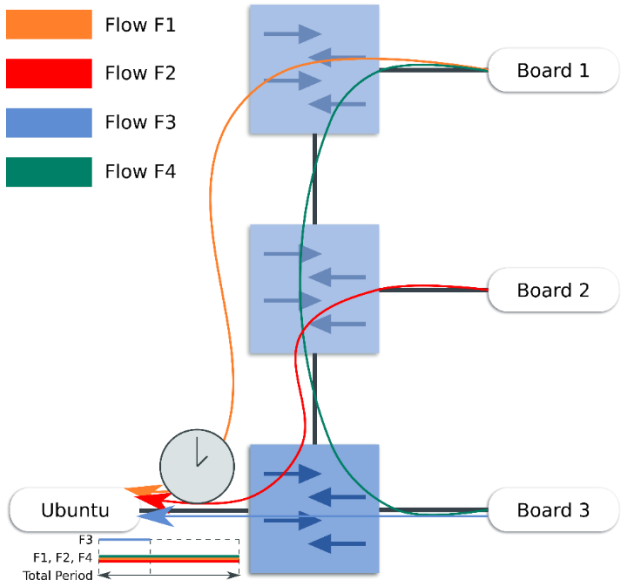

Figure 7-13. Actions to be performed on Board 3: Break up 10ms time intervals in such way that F3 traffic is only forwarded on egress L3 link 20% of the time. Give F3 traffic higher priority, to guarantee the bandwidth. Allow F1, F2 and F4 traffic on egress at all times, but treat as background traffic. This is not to incur additional rate limiting.

7.6.2.1 Creating the configuration

From **Error! Reference source not found. Error! Reference source not found.**, copy `board1-heduling.sh` on Board 1, `board2-scheduling.sh` on Board 2 and `board3-scheduling.sh` on Board 3. Also copy `definitions.sh` on all boards (this file can allow modification of some demo-related parameters). Then run these commands:

Board1:

```
[root@board1] $ chmod +x board1-scheduling.sh
[root@board1] $ ./board1-scheduling.sh
```

Board2:

```
[root@board2] $ chmod +x board2-scheduling.sh
[root@board2] $ ./board2-scheduling.sh
```

Board3:

```
[root@board2] $ chmod +x board3-scheduling.sh
[root@board3] $ ./board3-scheduling.sh
```

At the end, for each board there should appear a new configuration in `/etc/sja1105/`.

7.6.2.2 Known issues

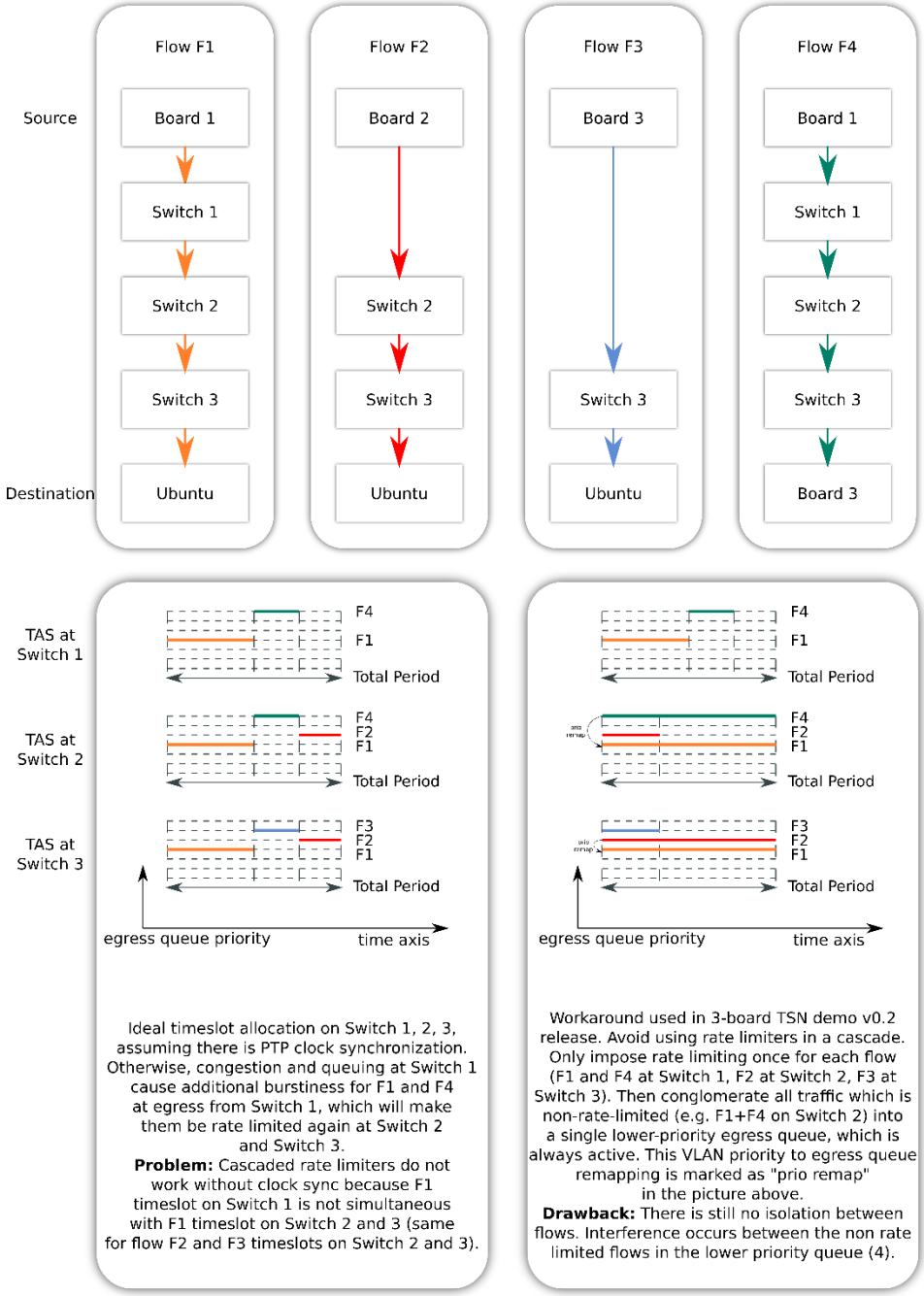

Figure 7-14. Limitation caused by the lack of clock synchronization between the 3 SJA1105 switches. As a workaround, time slots must be allocated in such a way that flow isolation is no longer possible.

7.7 NETCONF usage

Examples will be given only for Board 1 (IP 192.168.15.11). Please repeat all steps below for Board 2 and Board 3.

7.7.1 Creating a NETCONF session

Assume that the NETCONF IP addresses are as following:

- 192.168.15.11 for board1
- 192.168.15.12 for board2
- 192.168.15.13 for board3

You are advised to keep 3 connections open to all 3 boards, each in a separate window.

```
[ubuntu] $ netopeer-cli
netconf> connect --port 830 --login root 192.168.15.<board{1|2|3}-ip>
netconf> # press Enter for no password
# You may need to run this, in case you are using the candidate
# datastore (here we are not) and it becomes locked.
# See 5.6.5.
# netconf> discard-changes
```

7.7.2 Loading an existing XML configuration into the NETCONF datastore

After running the scripts in "**Error! Reference source not found. Error! Reference source not und.**" and "**Error! Reference source not found. Error! Reference source not found.**", on each board there should be a policing and a scheduling configuration file in XML format.

```
netconf> user-rpc
<!--#
  Type the content of a RPC operation.
  -->
<load-local-config xmlns="http://nxp.com/ns/yang/tsn/sja1105">
    <configfile>
       board1-policing.xml
    </configfile>
</load-local-config>
```

Running the command above will also apply the configuration.

7.7.3 Transferring the configurations to Ubuntu

Although this operation can also be performed using `scp`:

```
[ubuntu] $ scp root@192.168.15.11:/root/board1-{policing|scheduling}.xml .
[ubuntu] $ scp root@192.168.15.12:/root/board2-{policing|scheduling}.xml .
[ubuntu] $ scp root@192.168.15.13:/root/board3-{policing|scheduling}.xml .
```

It is also possible to transfer the configuration using NETCONF and `netopeer-cli`:

```
netconf> get-config running --out board1-policing.xml
```

If this operation ran successfully, there should be a new file in the current working directory on Ubuntu named `board1-policing.xml`, with the current contents of the datastore of the netopeer-server that we are connected to. Assuming you followed along over step **Error! eference source not found. Error! Reference source not found.**, this should match exactly the content of `/etc/sja1105/board1-policing.xml` on Board 1.

Please proceed and transfer the contents of all XML configurations to the Ubuntu PC.

At the end of this step, in the current working directory you should have:

- `board1-policing.xml`
- `board2-policing.xml`
- `board3-policing.xml`
- `board1-scheduling.xml`
- `board2-scheduling.xml`
- `board3-scheduling.xml`

7.7.4 Applying the configuration over NETCONF

The XML files correspond to the policing and scheduling configurations described in **Error! eference source not found.** and **Error! Reference source not found.**. They are following the syntax of the YIN model that can be found in `/usr/local/etc/netopeer/sja1105/sja1105.yin`, but are also compatible with the `sja1105-tool` backend which can load and export them natively.

Now that we also have the XML files generated with `sja1105-tool` locally on the Ubuntu PC, we can apply them remotely using NETCONF.

```
# Apply board1-policing.xml to the running datastore
# Configuration takes effect immediately
netconf> edit-config --config board1-policing.xml running
# Inspect the running datastore
```

```
netconf> get-config running
```

7.7.5 Running a configuration at startup

```
netconf> copy-config --source running startup
```

7.7.6 Viewing port statistics counters

The NETCONF protocol (and YANG data models) make a clear distinction between configuration data and state data. SJA1105 port counters are an example of state data exported by the `yang-sja1105` netopeer module. These can be very useful for debugging or investigating the traffic remotely.

```
# Get all configuration + state data
# get-config, by contrast, shows just configuration data
netconf> get
# Get just the port counters
netconf> get --filter
<sja1105>
    <ports/>
</sja1105>
```

7.7.7 Ending the NETCONF session

```
netconf> disconnect
```

7.8 Testing

For bandwidth testing, you have to use, one by one, these combinations of configurations:

- `board1-policing.xml` on Board 1, `board2-policing.xml` on Board 2, `board3-policing.xml` on Board 3
- `board1-scheduling.xml` on Board 1, `board2-scheduling.xml` on Board 2, `board3-scheduling.xml` on Board 3

Please refer to chapters **Error! Reference source not found. Error! Reference source not und.**, **Error! Reference source not found. Error! Reference source not found.**, or **Error! Reference source not found. Error! Reference source not found.**, on how to apply these.

7.8.1 Start the iPerf3 servers

```
[ubuntu] $ iperf3 -s -p 5201
[ubuntu] $ iperf3 -s -p 5202
[ubuntu] $ iperf3 -s -p 5203
[ubuntu] $ ssh -tt root@192.168.15.13 \
             "iperf3 -s -p 5201"  # board3
```

7.8.2 Start the iPerf3 clients

```
[ubuntu] $ ssh -tt root@192.168.15.11 \
         "iperf3 -c ubuntu -p 5201 -t 100"   # board1, flow F1
[ubuntu] $ ssh -tt root@192.168.15.12 \
         "iperf3 -c ubuntu -p 5202 -t 100"   # board2, flow F2
[ubuntu] $ ssh -tt root@192.168.15.13 \
         "iperf3 -c ubuntu -p 5203 -t 100"   # board3, flow F3
[ubuntu] $ ssh -tt root@192.168.15.11 \
         "iperf3 -c board3 -p 5201 -t 100"   # board1, flow F4
```

7.9 Attached files

definitions.sh:

```
#!/bin/sh

# Change this to match your MAC address
ubuntu_mac="aa:bb:cc:dd:ee:ff"
board1_mac="00:04:9F:00:01:02"
board2_mac="00:04:9F:00:02:02"
board3_mac="00:04:9F:00:03:02"
f1_prio=4
f2_prio=5
f3_prio=6
f4_prio=7
f1_rate_mbps=500
```

```
f2_rate_mbps=300
f3_rate_mbps=200
f4_rate_mbps=200
untagged_frame_size=1518
tagged_frame_size=1522
```

board1-policing.sh:

```
#!/bin/sh

# Chassis ETH2: Switch port RGMII 1
# Chassis ETH3: Switch port RGMII 2
# Chassis ETH4: Switch port RGMII 3
# Chassis ETH5: Switch port RGMII 0
# To LS1021:    Switch port RGMII 4
#
source ./definitions.sh

# Flows handled here:
# F1 - source board1, destination ubuntu
# F4 - source board1, destination board3
#
# Actions performed here:
# Rate-limit F4 at $f4_rate_mbps
f4_rate=$(echo "64000 * $f4_rate_mbps / 1000" | bc)

sja1105-tool config default ls1021atsn
# Initially, tag both F1 and F4 on ingress with F1's VLAN PRIO.
# Later, we will retag F4 with its own VLAN PRIO, at L2 forwarding.
sja1105-tool config modify mac-config[4] vlanprio $f1_prio
# Send VLAN-tagged traffic on the egress interface towards board2
sja1105-tool config modify vlan-lookup-table[0] tag_port 0b00010
# Treat F4 as regular L2, and apply ingress policing
```

```
# We still have to use the ingress VLAN prio here for F4,
# because it gets remapped only later, in the L2 Forwarding stage
# This does not apply to F1 because it is treated as VL
sja1105-tool config modify l2-policing-table[$((4*8 + $f1_prio))] rate $f4_rate

# Remap F4 from ingress VLAN prio 4 to egress VLAN prio 7.
# Unfortunately there is no easy way not to hardcode this
# and replace vlan_pmap[$f1_prio] with $f4_prio.
sja1105-tool config modify l2-forwarding-table[4] vlan_pmap "[0 1 2 3 7 5 6 7]"

# Save F1 from being treated as best-effort traffic,
# and therefore policed by the L2 Policer.
# Instead it is treated as a Rate-Constrained Virtual Link
# and has to pass its own BAG (Bandwidth Allocation Gap) check.
sja1105-tool config modify general-parameters-table vllupformat 0
sja1105-tool config modify vl-lookup-table entry-count 1
# VL 0 == flow F1 (source board1, dest ubuntu)
sja1105-tool config modify vl-lookup-table[0] iscritical 1
sja1105-tool config modify vl-lookup-table[0] macaddr   "$ubuntu_mac"
sja1105-tool config modify vl-lookup-table[0] vlanid     0
sja1105-tool config modify vl-lookup-table[0] vlanprio   $f1_prio
sja1105-tool config modify vl-lookup-table[0] port       4
# iscritical == 1, so this is unused
sja1105-tool config modify vl-lookup-table[0] destports  0

sja1105-tool config modify vl-forwarding-table entry-count 1
# VL 0 (flow F1)
# rate-constrained
sja1105-tool config modify vl-forwarding-table[0] type 0
# keep egress priority queue to 4
sja1105-tool config modify vl-forwarding-table[0] priority  4
sja1105-tool config modify vl-forwarding-table[0] partition 0
```

```
# forward to port 1
sja1105-tool config modify vl-forwarding-table[0] destports 0b00010

# Policer entry for VL 0 (F1). Defined as rate-constrained flow.
# Do not perform Bandwidth Allocation Gap (BAG) check.
f1_bag=0
sja1105-tool config modify vl-policing-table entry-count 1
# rate-constrained: make sure this is the same as in the fw-table
sja1105-tool config modify vl-policing-table[0] type     0
sja1105-tool config modify vl-policing-table[0] \
             maxlen   $untagged_frame_size
sja1105-tool config modify vl-policing-table[0] sharindx 0
sja1105-tool config modify vl-policing-table[0] bag      $f1_bag
sja1105-tool config modify vl-policing-table[0] jitter   0

# Make sure that L2 + VL traffic do not exceed maximum of 929 128-byte memory blocks.
# Use one partition for L2 traffic and one for VL traffic, since there are just 2 flows.
sja1105-tool config modify vl-forwarding-parameters-table entry-count 1
sja1105-tool config modify vl-forwarding-parameters-table partspc "[460 0 0 0 0 0 0 0]"
sja1105-tool config modify vl-forwarding-parameters-table debugen 0
sja1105-tool config modify l2-forwarding-parameters-table part_spc "[460 0 0 0 0 0 0 0]"

sja1105-tool config save /etc/sja1105/board1-policing.xml
# Uploading is not the primary concern of the script, but
# instead the generation of the config in xml format
sja1105-tool config upload
```

board2-policing.sh:

```
#!/bin/sh
```

```
# Chassis ETH2: Switch port RGMII 1
# Chassis ETH3: Switch port RGMII 2
# Chassis ETH4: Switch port RGMII 3
# Chassis ETH5: Switch port RGMII 0
# To LS1021:    Switch port RGMII 4
#
source ./definitions.sh

# Flows handled here:
# F1 - source board1, destination ubuntu
# F2 - source board2, destination ubuntu
# F4 - source board1, destination board3
#
# Actions performed here:
# Rate-limit F1 at $f1_rate_mbps
# Rate-limit F2 at $f2_rate_mbps
#
f1_rate=$(echo "64000 * $f1_rate_mbps / 1000" | bc)
f2_rate=$(echo "64000 * $f2_rate_mbps / 1000" | bc)

sja1105-tool config default ls1021atsn
# Make sure VLAN-tagged packets are not dropped by the ingress policer
sja1105-tool config modify l2-policing-table[$((8*2 + $f1_prio))] \
                           maxlen $tagged_frame_size
sja1105-tool config modify l2-policing-table[$((8*2 + $f4_prio))] \
                           maxlen $tagged_frame_size
# Send VLAN-tagged packets towards board3
sja1105-tool config modify vlan-lookup-table[0] tag_port 0b00010

# Flow F2 - source port 4
# Apply L2 ingress policing
sja1105-tool config modify mac-config[4] vlanprio $f2_prio
sja1105-tool config modify l2-policing-table[$((4*8 + $f2_prio))] rate
$f2_rate
```

```
# Also apply ingress policing on flow F1 - source port 2
sja1105-tool config modify l2-policing-table[$((2*8 + $f1_prio))] rate
$f1_rate

sja1105-tool config save /etc/sja1105/board2-policing.xml
# Uploading is not the primary concern of the script, but
# instead the generation of the config in xml format
sja1105-tool config upload
```

board3-policing.sh:

```
#!/bin/sh

# Chassis ETH2: Switch port RGMII 1
# Chassis ETH3: Switch port RGMII 2
# Chassis ETH4: Switch port RGMII 3
# Chassis ETH5: Switch port RGMII 0
# To LS1021:    Switch port RGMII 4
#
source ./definitions.sh

# Flows handled here:
# F1 - source board1, destination ubuntu
# F2 - source board2, destination ubuntu
# F3 - source board3, destination ubuntu
# F4 - source board1, destination board3
#
# Actions performed here:
# Rate-limit F3 at $f3_rate_mbps
#
f3_rate=$(echo "64000 * $f3_rate_mbps / 1000" | bc)

sja1105-tool config default ls1021atsn
# Make sure VLAN-tagged packets are not dropped by the ingress policer
```

```
sja1105-tool config modify l2-policing-table[$((8*2 + $f1_prio))] \
                           maxlen $tagged_frame_size
sja1105-tool config modify l2-policing-table[$((8*2 + $f2_prio))] \
                           maxlen $tagged_frame_size
sja1105-tool config modify l2-policing-table[$((8*2 + $f4_prio))] \
                           maxlen $tagged_frame_size

# Flow 3 - source port 4
# Apply L2 ingress policing
sja1105-tool config modify mac-config[4] vlanprio $f3_prio
sja1105-tool config modify l2-policing-table[$((4*8 + $f3_prio))] rate
$f3_rate

sja1105-tool config save /etc/sja1105/board3-policing.xml
# Uploading is not the primary concern of the script, but
# instead the generation of the config in xml format
sja1105-tool config upload
```

board1-scheduling.sh:

```
#!/bin/sh

# Chassis ETH2: Switch port RGMII 1
# Chassis ETH3: Switch port RGMII 2
# Chassis ETH4: Switch port RGMII 3
# Chassis ETH5: Switch port RGMII 0
# To LS1021:    Switch port RGMII 4
#
source ./definitions.sh

# Flows handled here:
# F1 - source board1, destination ubuntu. Virtual Link 0
# F4 - source board1, destination board3. Best effort
#
```

```
# Actions performed here:
# Create a time schedule with F1 allowed on first time slot,
# F4 allowed on second, and there is an extra empty time slot.
# Sum of all slots (the period of the schedule) is 10ms.
# The empty time slot is because f1_rate_mbps + f4_rate_mbps < 1000.
#
f1_slot_ms=$(echo "$f1_rate_mbps / 100.0" | bc -l)
f4_slot_ms=$(echo "$f4_rate_mbps / 100.0" | bc -l)
empty_slot_ms=$(echo "10 - $f1_slot_ms - $f4_slot_ms" | bc -l)
f1_delta=$(echo "($f1_slot_ms*5000)/1" | bc)
f4_delta=$(echo "($f4_slot_ms*5000)/1" | bc)
empty_delta=$(echo "($empty_slot_ms*5000)/1" | bc)
echo "F1 delta: $f1_delta, F4 delta: $f4_delta, empty delta: $empty_delta"

# Configuration begins here

sja1105-tool config default ls1021atsn
# Flows F1 and F4 - source port 4
# F1 will get special treatment as VL 0
sja1105-tool config modify mac-config[4] vlanprio $f1_prio
sja1105-tool config modify vlan-lookup-table[0] tag_port 0b00010
# Do the same retagging as in board1-policing.sh
# Remap ingress-to-egress VLAN tags $f1_prio (4) to $f4_prio (7)
# Applicable only for F4 (best-effort).
sja1105-tool config modify l2-forwarding-table[4] vlan_pmap "[0 1 2 3 7 5 6
7]"

sja1105-tool config modify general-parameters-table vllupformat 0
sja1105-tool config modify vl-lookup-table entry-count 1
sja1105-tool config modify vl-lookup-table[0] iscritical 1
sja1105-tool config modify vl-lookup-table[0] macaddr   "$ubuntu_mac"
sja1105-tool config modify vl-lookup-table[0] vlanid     0
sja1105-tool config modify vl-lookup-table[0] vlanprio  "$f1_prio"
sja1105-tool config modify vl-lookup-table[0] port       4
```

```
sja1105-tool config modify vl-forwarding-table entry-count 1
# rate-constrained
sja1105-tool config modify vl-forwarding-table[0] type 0
# do not overwrite f1 egress queue
sja1105-tool config modify vl-forwarding-table[0] priority $f1_prio
sja1105-tool config modify vl-forwarding-table[0] partition 0
# forward to port 1
sja1105-tool config modify vl-forwarding-table[0] destports 0b00010

# Nothing interesting happening here
# Rate-constrained VL but no BAG check
sja1105-tool config modify vl-policing-table entry-count 1
sja1105-tool config modify vl-policing-table[0] type     0
sja1105-tool config modify vl-policing-table[0] maxlen   $untagged_frame_size
sja1105-tool config modify vl-policing-table[0] sharindx 0
sja1105-tool config modify vl-policing-table[0] bag      0
sja1105-tool config modify vl-policing-table[0] jitter   0
sja1105-tool config modify vl-forwarding-parameters-table entry-count 1
sja1105-tool config modify vl-forwarding-parameters-table partspc "[460 0 0 0
0 0 0 0]"
sja1105-tool config modify vl-forwarding-parameters-table debugen 0
sja1105-tool config modify l2-forwarding-parameters-table part_spc "[460 0 0 0
0 0 0 0]"



sja1105-tool config modify schedule-table entry-count 3
# Egress port is 1 (towards board2)
sja1105-tool config modify schedule-table[0] destports 0b00010
sja1105-tool config modify schedule-table[1] destports 0b00010
sja1105-tool config modify schedule-table[2] destports 0b00010
sja1105-tool config modify schedule-table[0] resmedia_en 1
sja1105-tool config modify schedule-table[1] resmedia_en 1
sja1105-tool config modify schedule-table[2] resmedia_en 1
```

```
# TAS: Create timeslots for F1, F4 and an extra, empty timeslot because
#      F1 + F4 should be rate-limited at less than 1000Mbps total
sja1105-tool config modify schedule-table[0] resmedia $((0xFF & ~(1<<$f1_prio)))
sja1105-tool config modify schedule-table[1] resmedia $((0xFF & ~(1<<$f4_prio)))
sja1105-tool config modify schedule-table[2] resmedia 0b11111111
sja1105-tool config modify schedule-table[0] delta $f1_delta
sja1105-tool config modify schedule-table[1] delta $f4_delta
sja1105-tool config modify schedule-table[2] delta $empty_delta

# This is mostly boilerplate, but needs to be done
sja1105-tool config modify schedule-entry-points-table entry-count 1
sja1105-tool config modify schedule-parameters-table entry-count 1
sja1105-tool config modify schedule-parameters-table[0] subscheind "[2 2 2 2 2 2 2 2]"
sja1105-tool config modify schedule-entry-points-parameters-table entry-count 1
sja1105-tool config modify schedule-entry-points-parameters-table[0] clksrc 1

sja1105-tool config save /etc/sja1105/board1-scheduling.xml
# Uploading is not the primary concern of the script, but
# instead the generation of the config in xml format
sja1105-tool config upload
```

board2-scheduling.sh:

```
#!/bin/sh

# Chassis ETH2: Switch port RGMII 1
# Chassis ETH3: Switch port RGMII 2
# Chassis ETH4: Switch port RGMII 3
# Chassis ETH5: Switch port RGMII 0
# To LS1021:    Switch port RGMII 4
#
```

```
source ./definitions.sh

# Flows handled here:
# F1 - source board1, destination ubuntu.
#      VLAN prio $f1_prio, egress queue $f1_prio
# F2 - source board2, destination ubuntu
#      VLAN prio $f2_prio, egress queue $f2_prio
# F4 - source board1, destination board3
#      VLAN prio $f4_prio, egress queue $f1_prio
#
# Actions performed here:
# Create a time schedule where F1+F4 traffic is always permitted,
# and F2 traffic is only permitted to allow $f2_rate_mbps.
# Sum of all slots (the period of the schedule) is 10ms.
#
f2_slot_ms=$(echo "$f2_rate_mbps / 100.0" | bc -l)
other_slot_ms=$(echo "10 - $f2_slot_ms" | bc -l)
f2_delta=$(echo "($f2_slot_ms*5000)/1" | bc)
other_delta=$(echo "($other_slot_ms*5000)/1" | bc)
echo "Delta for all traffic: $f2_delta, delta for no F2: $other_delta"

# Configuration begins here

sja1105-tool config default ls1021atsn
# Must increase the size limit so vlan-tagged packets are
# not dropped by the policer
# Ingress trunk interface is port 2 (connected to board1), carrying flow F1
and F4.
# Frames on the way back (TCP ACKs) are not VLAN-tagged by the switch.
# So tagged packets are circulating from board1 -> board2, but untagged
# from board2 -> board1. Also see tag_port below.
sja1105-tool config modify l2-policing-table[$((2*8 + $f1_prio))] \
                    maxlen $tagged_frame_size
sja1105-tool config modify l2-policing-table[$((2*8 + $f4_prio))] \
```

```
                    maxlen $tagged_frame_size
# Make egress port 1 as trunk (connected to board3), carrying flows F1, F2,
F4.
sja1105-tool config modify vlan-lookup-table tag_port 0b00010
# Flow F2 - source port 4. Assign default VLAN priority $f2_prio
sja1105-tool config modify mac-config[4] vlanprio $f2_prio

# Remap VLAN priority $f4_prio (7) to egress queue $f1_prio (4).
# Applicable to flow F4. Does not change VLAN tag.
sja1105-tool config modify l2-forwarding-table[$((5+7))] vlan_pmap "[4 4 4 4 4
0 0 0]"

# Schedule table configuration

sja1105-tool config modify schedule-table entry-count 2
# Egress port is 1 (towards board3)
sja1105-tool config modify schedule-table[0] destports 0b00010
sja1105-tool config modify schedule-table[1] destports 0b00010
sja1105-tool config modify schedule-table[0] resmedia_en 1
sja1105-tool config modify schedule-table[1] resmedia_en 1
# TAS: Timeslot 0: All traffic allowed, F2 having priority
#      Timeslot 1: F2 traffic not allowed
sja1105-tool config modify schedule-table[0] resmedia \
             $((0xFF & ~(1<<$f1_prio) & ~(1<<$f2_prio)))
sja1105-tool config modify schedule-table[1] resmedia \
             $((0xFF & ~(1<<$f1_prio)))
sja1105-tool config modify schedule-table[0] delta $f2_delta
sja1105-tool config modify schedule-table[1] delta $other_delta

# This is mostly boilerplate, but needs to be done
sja1105-tool config modify schedule-entry-points-table entry-count 1
sja1105-tool config modify schedule-parameters-table entry-count 1
sja1105-tool config modify schedule-parameters-table[0] subscheind "[1 1 1 1 1
1 1 1]"
sja1105-tool  config  modify  schedule-entry-points-parameters-table  entry-count
```

```
1
sja1105-tool config modify schedule-entry-points-parameters-table[0] clksrc 1

sja1105-tool config save /etc/sja1105/board2-scheduling.xml
# Uploading is not the primary concern of the script, but
# instead the generation of the config in xml format
sja1105-tool config upload
```

board3-scheduling.sh:

```
#!/bin/sh

# Chassis ETH2: Switch port RGMII 1
# Chassis ETH3: Switch port RGMII 2
# Chassis ETH4: Switch port RGMII 3
# Chassis ETH5: Switch port RGMII 0
# To LS1021:    Switch port RGMII 4
#
source ./definitions.sh

# Flows handled here:
# F1 - source board1, destination ubuntu
#      VLAN prio $f1_prio, egress queue $f1_prio
# F2 - source board2, destination ubuntu
#      VLAN prio $f1_prio, egress queue $f1_prio
# F3 - source board3, destination ubuntu
#      VLAN prio $f1_prio, egress queue $f1_prio
# F4 - source board1, destination board3
#
#
# Actions performed here:
# Create a time schedule towards Ubuntu where F1+F2 traffic
```

```
# is always permitted, and another one where F3 traffic is
# only permitted enough to accomodate $f3_rate_mbps.
# Sum of all slots (the period of the schedule) is 10ms.
#
f3_slot_ms=$(echo "$f3_rate_mbps / 100.0" | bc -l)
other_slot_ms=$(echo "10 - $f3_slot_ms" | bc -l)
f3_delta=$(echo "($f3_slot_ms*5000)/1" | bc)
other_delta=$(echo "($other_slot_ms*5000)/1" | bc)
echo "Delta for all traffic: $f3_delta, delta for no F3: $other_delta"

sja1105-tool config default ls1021atsn
# Must increase the limit so vlan-tagged packets are not dropped by the
policer
sja1105-tool config modify l2-policing-table[$((2*8 + $f1_prio))] \
                    maxlen $tagged_frame_size
sja1105-tool config modify l2-policing-table[$((2*8 + $f2_prio))] \
                    maxlen $tagged_frame_size
sja1105-tool config modify l2-policing-table[$((2*8 + $f4_prio))] \
                    maxlen $tagged_frame_size

# Flow F3 - source port 4. Assign default VLAN priority $f3_prio
sja1105-tool config modify mac-config[4] vlanprio $f3_prio

# Remap VLAN priority $f2_prio (5) to egress queue $f1_prio (4).
# Applicable to flow F2. Does not change VLAN tag.
sja1105-tool config modify l2-forwarding-table[$((5+5))] vlan_pmap "[4 4 4 4 4
0 0 0]"

sja1105-tool config modify schedule-table entry-count 2
# Egress port is 0 (towards ubuntu)
sja1105-tool config modify schedule-table[0] destports 0b00001
sja1105-tool config modify schedule-table[1] destports 0b00001
sja1105-tool config modify schedule-table[0] resmedia_en 1
sja1105-tool config modify schedule-table[1] resmedia_en 1
```

```
# TAS: Timeslot 1: All traffic allowed, F3 having priority
#      Timeslot 2: F3 traffic not allowed
sja1105-tool config modify schedule-table[0] resmedia \
             $((0xFF & ~(1<<$f1_prio) & ~(1<<$f2_prio) & ~(1<<$f3_prio)))
sja1105-tool config modify schedule-table[1] resmedia \
             $((0xFF & ~(1<<$f1_prio) & ~(1<<$f2_prio)))
sja1105-tool config modify schedule-table[0] delta $f3_delta
sja1105-tool config modify schedule-table[1] delta $other_delta

# This is mostly boilerplate, but needs to be done
sja1105-tool config modify schedule-entry-points-table entry-count 1
sja1105-tool config modify schedule-parameters-table entry-count 1
sja1105-tool config modify schedule-parameters-table[0] subscheind "[1 1 1 1 1
1 1 1]"
sja1105-tool  config  modify  schedule-entry-points-parameters-table  entry-count
1
sja1105-tool config modify schedule-entry-points-parameters-table[0] clksrc 1

sja1105-tool config save /etc/sja1105/board3-scheduling.xml
# Uploading is not the primary concern of the script, but
# instead the generation of the config in xml format
sja1105-tool config upload
```

8 Known Issues

Item	Description
1	Need to define more relevant usage scenario for SAE AS6802 features (time-triggered, rate-constrained traffic). These are not used at the moment.
2	Sja1105-tool does not communicate with the BCM5464R PHY.
3	Flow contention on Switches 2 and 3 in 3-board TSN demo.
4	Xenomai PCIe interrupt support is only working with INTx mode.

Appendix A. Version Tracking

Date	Version	Comments	Author(s)
28/04/2017	0.2	Engineer release	NXP
26/05/2017	0.2	Customer release	NXP