TrustNode:
Descriptions and Interfaces
Technical Description

December 20, 2019

Marian Ulbricht

InnoRoute GmbH,
Marsstrasse 14a
80335 Munich, Germany

Contents

About this document

This document describes functions which are unique for the TrustNode device. It will not describe linux and networking basics. Please check also the TrustNode tutorial for initial questions. As part of our maintenance contract, our team will be happy to answer any configuration questions.

First Steps

Congratulations you are owning a TrustNode (TN), the fast, flexible FPGA-based routing platform made in Germany. In this section you will find any information to setup the device.

1.1 Flash the boot medium

If the TN is delivered without boot medium or you have to perform a full software update, flashing a USB-device or SD-card is needed. The archive where you found this manual contains also a compressed OS-image trustnode*combined-uefi.img.gz.

1.1.1 Linux – USB stick flashing

Congratulations, nearly done, just[1] type:

```
gunzip -c trustnode*combined-*.img.gz | sudo dd bs=1M of=/dev/XXX && sudo sync
```

If you need further information how to select the target device: see section 1.1.2.

1.1.2 Windows – USB stick flashing

First you have to extract the trustnode*combined-*.img.gz using you favourite tools e.g. 7zip. Now you can use a USB-stick flashing tool like ImageUSB[2] or Win32DiskImager[3] to get the image to USB-stick or SD-card.[4]. If done, unmount[5] the medium.

Note: Windows will not be able to read the data written to the device. Windows GUI tools will not be able to delete the UEFI-boot partition if written to the bootmedium the first time. To reflash a bootmedium see here or install a Linux operating system and see Section 1.1.1.

[1] We assume you are know what you are doing InnoRoute is not responsible for any data loss on you system.
[2] `http://www.chip.de/downloads/ImageUSB_61096110.html`
[3] `https://sourceforge.net/projects/win32diskimager/`
[4] This will delete any data on the chosen medium.
[5] Be sure that all data is written from the write buffer to the device itself.

1.1.3 UEFI-image BIOS settings

To boot-up the UEFI based images correctly, the following steps have to be applied to the BIOS-setup-utility of the CPU:

- After powerup press [ESC] and choose the "setup utility"
- Exit → load_optimal_defaults
- Advanced → consoleredirection → enable, 115200n8 (important!, if you forget, you will be blind)
- advanced → PCI → disable PCIe rootport 2-4
- boot → boot-type:uefi
- boot → set ACPI version to 4.0
- exit → save changes

Note: The UEFI image boot currently only from USB memorystick, SD-card is not supported.

1.2 Console connection

The root-shell is accessible over the microUSB port on the backside.[6] In some cases you have to install the FTDI-comport drivers[7]. The connection parameters are 115200 Baud 8N1, as printed on the backside of the TN.

1.3 SSH connection

SSHd is running at port 22, the root login is not permitted.

1.4 Login

Two logins are arranged:

- **root**:innoroot
- **TNuser**:innoroute

[6] Plug-in careful and use a strain-relief for free hanging cables.
[7] `http://www.ftdichip.com/Drivers/VCP.htm`

1.5 Firstboot

To setup your board with the default settings, run:

```
/etc/init.d/TN_firstboot start
```

The firstboot-script is automatically launched if a new image is used the first time. The output is written to /usr/share/InnoRoute/firstboot.log

1.5.1 RSA key

A 4096bit RSA-key is automatically generated from the firstboot script and stored in /root/.ssh .

1.6 Using the Build environment

To generate a own TN firmware image, a build environment including a setup script is available here: `https://github.com/InnoRoute/TrustNodeWRT`.

1.6.1 Build environment setup

Please use an Debian[8] based host system with minimum 30 GB free space. The included setup script will download all required packages and setup an openwrt-based build environment. This will need some time, depending on the amount of cores in your build-system.

```
git clone https://github.com/InnoRoute/TrustNodeWRT TrustNodeWRT
cd TrustNodeWRT
./makeTrustNodeWRT.sh
#enter your root pwd (we need to install packages before)
#wait (get some coffe)
```

[8] suggestion: `https://xubuntu.org/getxubuntu/`

1.6.2 Build environment usage

Basic commands:

```
make menuconfig #shows the config menu, to select packages
make kernel_menuconfig #shows the config menu for the kernel
make -j8 V=99 #generated an image with a lot of log messages using 8 threads
```

If the build is successful, the imagefile can be found under openwrt/bin/TrustNode-glibc/, precompiled packages can be found in openwrt/bin/TrustNode-glibc/packages. (see Section 1.1.1 for flashing the Image.)

1.6.3 Updating the InnoRoute packages

To update the InnoRoute packages from the code-repository use the OpenWRT update environment:

```
cd openwrt
scripts/feeds update InnoRouteTN -a
scripts/feeds install -p InnoRouteTN -a
scripts/feeds install -p InnoRouteTN TrustNode
```

1.6.4 Include the packet repository into OpenWRT

This step is normaly done by the TrustNodeWRT-setup-script. To include the InnoRoute package repository into an other openwrt instance, add the following line to the file openwrt-folder/feeds.conf.default:

```
src-git InnoRouteTN https://github.com/InnoRoute/packages.git
```

Now run the following commands from the OpenWRT-folder:

```
cd openwrt
scripts/feeds update InnoRouteTN
scripts/feeds install -p InnoRouteTN -a
scripts/feeds install -p InnoRouteTN TrustNode
make menuconfig
```

The new packages are now available in the *InnoRoute* section.

2.1 FPGA programming

The FPGA can be programmed via ftdi-JTAG or via ftdi-optomode(faster):

```
xc3sprog -c ftdi -v -p0 file.bit
TN_opto_prog file.bit
```

After programming the FPGA more information are available using the the status scripts described in Section 5.2.

2.2 Bitstream structure

Figure 1 gives an overview about the structure and implemented components in the TN bitstream[9].

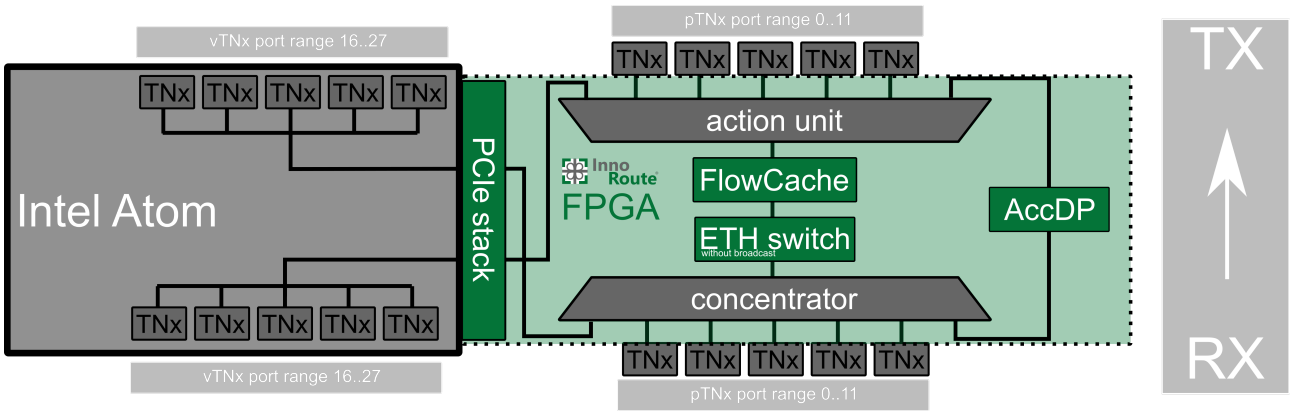

Figure 1: TrustNode bitstream structure

As shown in Figure 1, all physical ports of the device are available as virtual ports inside the Linux operating system. Packets which are forwarded to the Central Processing Unit (CPU) passing the whole chain of TN functions e.g. FlowCache and Ethernet switch before they are available on the internal TX ports. The FlowChache is chained after the Ethernet switch which means that this component can override decisions made by prior blocks. The acceleration datapath (AccDP) as a component which is not cut trough enabled, is also connected to internal ports of the design.

[9]This means the common bitstream, other implementation might contain more or a subset of the features.

Trustnode PCIe-MMI-Driver

The memory mapped interface to the TN FPGA core is accessible using the following command:

```
TNbar1 offset value
TNbar1 0x123 #reads value at address 0x123
TNbar1 0x123 0xAA #sets value at address 0x123 to 0xAA
```

The address register descriptions is part of the Hardware manual which is delivered with the bitstreams.

Trustnode PCIe-Ethernet-Driver

4.1 /proc filesystem configuration

/proc/TrustNode/TNsend2cpu set the ll flag in every TX-descriptor and send the packets trough the FPGA back to the CPU

/proc/TrustNode/TN_TXdbg enables debug messages in TX path

/proc/TrustNode/TN_RXdbg enables debug messages in RX path

The log messages can be viewed with tail -f /var/log/messages.

5.1 Linux packet management

The installed operation system uses the opkg packet-management system to easily update binary packets. InnoRoute provides the following packet feeds[10] which can be used within opkg.

```
src/gz TrustNode_sys http://www.innoroute.com/trustnode/openwrt-pkg/packages
src/gz TrustNode_packages http://www.innoroute.com/trustnode/openwrt-pkg/x86_64/packages
src/gz TrustNode_base http://www.innoroute.com/trustnode/openwrt-pkg/x86_64/base
src/gz TrustNode_routing http://www.innoroute.com/trustnode/openwrt-pkg/x86_64/routing
src/gz TrustNode_luci http://www.innoroute.com/trustnode/openwrt-pkg/x86_64/luci
src/gz TrustNode_telephony http://www.innoroute.com/trustnode/openwrt-pkg/x86_64/
telephony
```

Use opkg update to pull the actual packet lists from the server. See also Section 5.2 for an automatic upgrade of all packages.

5.2 Usefull scripts

This section describes some of the scripts located in /usr/share/InnoRoute/. Passing no parameter to the scripts will not launch any action but will display the help. The scripts uses the Memory Mapped Interface (MMI) described in Section 3.

TN_beep.sh: Makes a acoustic signal.

TN_pkg_upgrade_all.sh: Update package lists and installs updates for binary packages from InnoRoute's server.

TN_buttons.sh: Polls status of frontpanel buttons.

TN_clock.sh: Switches system clock source between Temperature-Controlled Crystal Oscillator (TCXO) and Phase Lock Loop (PLL).

TN_eth_switch.sh: Disable or enable the hardware ethernet switch.

TN_fc.sh: Disable or enable FlowCache modules.

TN_fpga_config_jtag.sh: Load an bitstream via Joint Test Action Group (JTAG) interface.

TN_fpga_config_lpc.sh: Load an bitstream via Low Pin Count (LPC) interface.

[10] For image versions 1.3 and higher this package sources are already installed.

TN_fpga_config_opto.sh: Load an bitstream via Future Technology Devices International (FTDI)-fast-serial interface.

TN_fpga_loopback.sh: Change Field Programmable Gate Array (FPGA) default packet handling.

TN_fpga_status.sh: Show status and statistics.

TN_fan.sh: Set fan speed.

TN_fpga_status.sh: Display status information about loaded bitstream.

TN_front_display.sh: Write data to front display.

TN_gpio.sh: Write data to general purpose input/output (GPIO) header(inside the case).

TN_inbterrupt.sh: Read out and reset current interrupt status.

TN_interrupt_mask.sh: Read or write the interrupt mask.

TN_mmi_status.sh: Display MMI transfer statistics.

TN_mute.sh: Disable beep.

TN_nw_bridge.sh: Add an brctl based bridge over several ports. (be aware of Open vSwitch (OVS))

TN_led.sh: Write data to internal and Alaska-Physical Layer Chip (PHY)-LEDs.

TN_mac.sh: Control the PHY chips via Management Data Input/Output (MDIO).

TN_nw_silent.sh: Silent Dynamic Host Configuration Protocol (DHCP)-client - and outgoing Address Resolution Protocol (ARP) messages.

TN_pcie.sh: Rescan Peripheral Component Interconnect Express (PCIe) bus and display extended device information.

TN_phy_dump.sh: Display status information about the PHY chips via MDIO.

TN_phy_access.sh: Control the PHY chips via MDIO.

TN_phy_examples.sh: Example for MDIO write.

TN_phy_init.sh: Initialising all PHYs via MDIO.

TN_phy_force_linkspeed.sh: Force the PHY chips to advertise only one linkspeed.

TN_phy_interrupt.sh: Read and reset the PHY interrupt status.

TN_phy_loopback.sh:	Enable loopback in the PHY.
TN_phy_reset.sh:	Hard-resetting the Ethernet PHYs.
TN_pll_dump.sh:	Dumping PLL register contents.
TN_pll_status.sh:	Display PLL satus information.
TN_rgmii_phase.sh:	Setting board-specific Reduced Gigabit Media Independent Interface (RGMII) input delays.
TN_rgmii_status.sh:	Display RGMII status information.
TN_sys_ctrl_pmod.sh:	Setting System Controller Peripheral Module (PMod) connection.
TN_sys_ctrl_status.sh:	Display status information from system controller.
TN_statistics.sh:	Display statistic counters.
TN_sysctl_pmod.sh:	Change the system controller PMod settings.
TN_sysctl_status.sh:	Readout system controller status.
TN_time.sh:	Set HW-clock.
TN_usb.sh:	Display attached Universal Serial Bus (USB) devices.
TN_version.sh:	Display Hardware version information.

5.3 PLL

The AD9558 PLL chip can be accessed via SM-BUS /dev/i2c-0. The device address is $0x69$. InnoRoute provide a special tool to configure the PLL with a predefined configuration file. The configuration file can be created using the AD-Tool[11]. A example file can be found in /usr/share/InnoRoute/. The following command loads the *.stp config file to the PLL.

```
INRpllload STP-file
#use parameter E to write the settings to the PLL-eeprom
```

[11] http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-ad9558.html

5.4 Sensors

Temperature, voltage and fan-speed probes are accessible via the lm-sensors-tool. The following command will list the available values of all sensors:

```
sensors
```

To change the sensors settings like max_temp or crit_temp, modify the file /usr/share/InnoRoute/INRsensor.conf and write the values to the hardware using the following command:

```
sensors -c /usr/share/InnoRoute/INRsensor.conf -s
```

5.5 Watchdog

The hardware watchdog is accessible via /dev/watchdog0 .

5.6 I2C

Native I2C access is actually not provided by the firmware of the CPU-module.

5.7 SM-Bus

The SM-Bus is accesable via /dev/i2c-0

5.8 Real Time Clock

The RTC is accesable via /dev/rtc0. Use the following comands to configure:

```
hwclock -w  #write systeme time into HW-clock
date -s "2013-11-19 15:11:40" #set system-time
```

5.9 FTDI eeprom

To initialise the ftdi-eeprom use the following command:

```
INR_ftdi_eeprom -p 0x6010 -v 0x0403 -M
```

5.10 Network configuration

In default state, the device is configured as DHCP client at interface eth0[12]. The configuration can be changed by manipulating the file /etc/config/network. If a bitstream with PCIe-network-interfaces is loaded the TN interface driver is loaded automatically on start up and the Interfaces TN0 to TNx will appear automatically. Image versions higher or equal then v1.0 OVS is used for internal network configuration, see Section 6.1 for more information.

5.11 Software packet management

For packet management the opkg packet manager is used to install or update an provided *.ipkg file use the following command:

```
opkg install <PACKETFILE>
```

[12] which can be a external attached USB network adapter

5.12 FPGA mode changing

The FPGA can be configured with several bitstreams to achieve different behaviours. This configuration is handled using the software TNmodchange, the last used configuration is automatically saved and loaded at next bootup. Please use the following command to change the FPGA configuration[13]:

```
TNchangemod <modus>
    # 0:disabled
    # 1:6Tree
    # 2:Acceleration Datapath
    # 3:Ethernet swiching
    # 4:Displaytest and loopback
    # 5:Atom-FPGA control interface
    # 6:Flowcache
    # 7:Combined Bitstream
    # 8:User Bitstream
```

The actual configuration is stored in the file /usr/share/InnoRoute/TNmod.conf, changes will applied after reboot.

5.12.1 User defined bitstream

TNmodchange 8 provides an special option to load a user defined bitstream from the first partition of the USB-bootmedium. Before selection this option, the *.bit file hast to be saved as tn_user.bit at the FAT32 partition of the boot medium.

[13] This is an example configuration, the bitstreams included in your image may be different.

5.13 FlowCache

The Flowcache is a part of the InnoRoute datapath (DP) which supports flow-classification in hardware. The hardware implementation depends on several tables for full and wildcard based matching L2 and L3 flows. A structural description of the FlowCache and its components can be found in the FlowCache documentation. A overview over the internal components structure is provided in Figure 1. The software tool TNflowtable helps to address the different flow table implementations and automatically select the best table for the provided information. Available flowtables:

EMH action table: Matching for full L2 or L3 field stack.

EMH hash table: Hash table for EMH rule table.

EMH collision table: Backup table for EMA hash collisions.

EMA action table: Matching for selectable L2 or L3 field stack.

EMA hash table: Hash table for EMA rule table.

Action table: Table for storing Actions.

Mastertable: Softwaretable, flows are automatically distributed to the hardware tables.

Supported match fields are:

- MAC_SRC
- MAC_DST
- VLAN
- ETHERTYPE
- IPv4_SRC
- IPv4_DST
- L4_PROTO
- TCP/UDP_SRC
- TCP/UDP_DST
- VLAN_PRIO
- TOS
- PHY_INPORT

The following commands are used to configure the Flowtables:

```
TNflowtable RuleT_EMH_add [options] #add to EMH rule table
TNflowtable HashT_EMH_add [options] #add to EMH hash table
TNflowtable CollT_EMH_add [options] #add to EMH collision table
TNflowtable RuleT_EMA_add [options] #add to EMA rule table
TNflowtable HashT_EMA_add [options] #add to EMA hash table
TNflowtable ActT_add [options] #add to actiontable
TNflowtable add [options] #add to mastertable and add to EMA, EMH or EMH_CollT and ActT
TNflowtable print -i2 -c10 #print mastertable from item  2 to 12
TNflowtable --help #show all options
```

Adding a flow to the Mastertable with TNflowtable add -C5.6.7.8 -T1.2.3.4 -O1 will automatically configure the hardware Tables to match both IP-address fileds and forward the packet to output port #1. To use the flowcache, the associated bitstream has to selected as described in Section 5.12.

5.13.1 Resetting the FlowCache

If for some reasons all flows have to been wiped from the hardware flow tables:

```
TNchangemod 7 #reload the hardware bitstream
rm /tmp/INR_FC* #remove all shadow memory files
```

5.13.2 FlowCache kernelmodule

comming soon...

5.13.3 FlowCache example commands

Table 1: FlowCache example commands

Command	**Match**	**Action**
TNflowtable add -I1 -O2	all packets from port #1	forward to port #2
TNflowtable del -I1 -O2	all entry's matching the pattern	delete entry
TNflowtable add -I0 -O16	all packets from port #0	forward to CPU port TN0
TNflowtable ActT_add -i0x155 -b1	all packets not matching a other[14] rules in the FlowCache[15]	drop (set bad=1)
TNflowtable ActT_del -i0x155	action table entry 0x155	delete entry
TNflowtable ActT_print -i0x155 -c10	print 10 entrys of the action table beginning with position 0x155	
TNflowtable add -T192.168.0.1 -O16	all packets with dst_ip=192.168.0.1	forward to port #16 (Port #16 is the TN0 interface inside Linux.)

[15] This entry can be used to overrule decisions made by the hardware Ethernet switch.

[15] -i specifies the position in a table to start with the search for a free position. To address a special table ID e.g. 0x155 ensure with the print command that table position 0x155 is free. Otherwise the entry will be stored at the next free space which can be e.g. 0x116.

5.13.4 Disable hardware Ethernet switch

Use the script TN_eth_switch.sh to disable the hardware Ethernet switch. Alternatively this result can be achieved using the following FlowCache rules:

```
TNflowtable add -I0 -O16
TNflowtable add -I1 -O17
TNflowtable add -I2 -O18
TNflowtable add -I3 -O19
TNflowtable add -I4 -O20
TNflowtable add -I5 -O21
TNflowtable add -I6 -O22
TNflowtable add -I7 -O23
TNflowtable add -I8 -O24
TNflowtable add -I9 -O25
TNflowtable add -I10 -O26
TNflowtable add -I11 -O27
```

5.13.5 Modular feature concept

The FlowCache consist of tree different modules (Linear search, EXACT MATHING WITH HARD-CODED FIELDS (EMH), EXACT MATHING WITH ARBITRARY FIELDS (EMA)) which can be en- or disabled during compilation of the VHDL sourcecode. The actual configuration can be readout with the following command: TN_fpga_status.sh 0 | grep Flow. The flow fields considered by the EMH module can be easyly adapted by minor changes in the VHDL and C sourcecode, several versions of this module are available, the command TNflowtable –help show the actual configuration of the rule types 1..4. Using TNflowtable -v | grep EMH_hash_revision the EMH version of the software module is displayed.

5.14 TNflowdump

The tool TNflowdump implements a lightweight connection between the OVS flowtables and the flowcache which is described in Section 5.13. The python script TNdumpflow.py is located in /usr/share/InnoRoute/scripts/. The script listen to all changes applied to a specified OVS bridge and adopt all changes to the entries of the flowcache. The monitored bridge can be specified in the configuration file /usr/share/InnoRoute/TNflowdump.conf. TNflowdump can be activated/deactivated using the init script /ect/init.d/TN_flowdump.

5.15 Time Aware Shaper

This section describes the configuration of the Time Aware Shaper (TAS).

NOTE: This feature is not available in the default bitstreams, but can be included after licensing the hilscher IPcore.

The userspace tool TNtsnctl is used to access all Time-Sensitive Networking (TSN) related registers and tables. According to "A Performance Study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv Enhancements"[1] the following tables are accessible per port:

- Queue priority list
- Gate Control list
- Time Gate Control list

5.15.1 General options

The general parameters can be used to address single ports or entry's in a list. The -m option transform every output into an machine readable form using JavaScript Object Notation (JSON).

```
General options:
-i, --ID=                    ID of entry
-m, --machine                Output in machine readable notation.
-P, --PORT=                  TrustNode port to configure
-v, --verbose                Produce verbose output
-c, --COUNT=                 Number of items to process
```

5.15.2 List options

The three lists can be accessed using the commands:

- TNtsnctl QueuePrio_list_print -i<start> -P<PORT> -c<count>
- TNtsnctl QueuePrio_list_change -i<ID> -P<PORT> -p<priority>
- TNtsnctl GateControl_list_print -i<start> -P<PORT> -c<count>
- TNtsnctl GateControl_list_change -i<ID> -P<PORT> -S<gate_state>
- TNtsnctl TGateControl_list_print -i<start> -P<PORT> -c<count>
- TNtsnctl TGateControl_list_change -i<ID> -P<PORT> -I<interval>

```
List options
-I, --INTERVAL=              Interval entry of list
-p, --QUEUE_PRIO=            Gate state of entry
-S, --GATE_STATE_VECTOR=     Gate state of entry
```

5.15.3 Configuration parameters

All other per port TSN configuration can be accessed via the config space using:

- TNtsnctl config_print -P<PORT>
- TNtsnctl config_change -P<PORT> <parameters to change>

```
General TSN config:
-a, --ADMIN_GCL_LEN=          The length of the gate list to be programmed.
-b, --ADMIN_BASE_TIME=        The base time for the gate list to be programmed.
-C, --ADMIN_CYCLE_TIME=       The admin cycle time constant for the gate list to
                              be programmed.
-d, --ADMIN_CYCLE_TIME_EXT=   The admin cycle time extension constant for the
```

```
                                gate list to be programmed.
-e, --CONFIG_CHANGE_TIME=       The calculated config change time. Assumption is
                                that calculations done in software.
-f, --CYCLE_START_TIME=         The cycle start time for the control list. When
                                the list pointer is set to zero.
-g, --GATE_ENABLE=              The gate enable bit. The gate control list is only
                                active if enabled, otherwise all traffic goes through.
-h, --CONFIG_CHANGE=            The config change command. Issued after
                                configuring the gate control list and admin registers.
```

5.15.4 Applying the configuration to the hardware

The TSN functions can be fully configured using the commands in the sections before. Using the command TNtsnctl apply automates this process, as described in [1][p. 23 ff.]. If no port is specified, the configuration is applied to all ports.

5.15.5 User-friendly configuration options

To reduce the risk of potential misconfiguration, user friendly commands are provided to access the gate control list. To apply two gate control list entries with the gate states 0x01 and 0xff on port0 use the following commands:

```
TNtsnctl Change_entry -i0 -S0x01 -I500000000 -a2 -P0
TNtsnctl Change_entry -i1 -S0xff -I500000000 -a2 -P0
TNtsnctl apply -C1000000000 -b0 -d0 -P0
```

The parameters provided to these commands are checked on plausibility and corrected in case of misconfiguration.

6.1 Open vSwitch

The Open vSwitch, running on the Atom processor, provides a open flow interface. Further documentation is available under `http://openvswitch.org` . To add a OpenFlow controller to the preconfigured OVS run the following command:

```
ovs-vsctl set-controller TNbr tcp:<IP_ADDR>
```

Note that the hardware Ethernet switch have to be overwritten or disabled to forward all traffic to the OVS, see Section 5.13.4.

6.2 Rapid Spanning Tree Protocol

The Rapid Spanning Tree Protocol (RSTP) is supported by OVS, a complete list of commands can be found here: `http://www.openvswitch.org/support/dist-docs/ovs-vsctl.8.txt`.

6.2.1 Enable Rapid Spanning Tree Protocol

```
ovs-vsctl set Bridge TNbr rstp_enable=true
```

6.2.2 Set Rapid Spanning Tree Protocol priority

```
ovs-vsctl set Bridge TNbr other_config:rstp-priority=28672
```

6.2.3 Set Rapid Spanning Tree Protocol port priority

```
ovs-vsctl set Port TNx other_config:rstp-port-priority=32
```

6.2.4 Set Rapid Spanning Tree Protocol bridge address

```
ovs-vsctl set Bridge TNbr other_config:rstp-address=00:aa:aa:aa:aa:aa
```

Note: If several TrustNodes are connected, each need to have a distinct bridge ID.

6.2.5 Print Rapid Spanning Tree Protocol status

```
TN_ovs_rstp_info.sh
```

6.3 Per packet queuing and policing 802.1Qci

The Qci core can be controlled using the tool TNqcictl.

```
root@TrustNode:~# TNqcictl --help
```

6.4 Simple Network Management Protocol

Simple Network Management Protocol (SNMP) is active, for secure configuration edit /etc/config/snmpd.

6.5 Link Layer Discovery Protocol

Link Layer Discovery Protocol (LLDP) is active, for configuration edit /etc/config/lldpd.

6.6 NetConf interface

The netopeer2 server is listening on port 830, providing an NetConf interface. The neconf interface description is embedded into the webUI, described in Section 7.2. The detailed configuration could be manipulated using the standard NetConf- and sysrepo-tools.[16]

[16] see `http://www.sysrepo.org/static/doc/html/bin_page.html`

6.6.1 Persistent storage

The tool TN_config.sh provides an mechanism to store NetConf Extensible Markup Language (XML) configuration in an persistent way. Stored configurations will be loaded automatically after every bitstream reprogramming.

```
root@TrustNode:~# TN_config.sh
TrustNode sysrepo persistent config tool, please specify action:
    [save]: saves actual config to be loaded after startup and FPGA reporogramming
    [load]: load stored configuration
    [clear]: clear stored configuration
    [print]: print saved config
    [running]: print actual running config
```

7.1 Statistics

7.1.1 Inport-Outport

The Input-Output section displays a transition diagram for datapath 0 and 1, consisting of a list of RX Ports 0-31, including physical ports 0-15 and virtual ports 16-31 as well as a list of physical TX Ports 0-15 and virtual TX Ports 16-31.

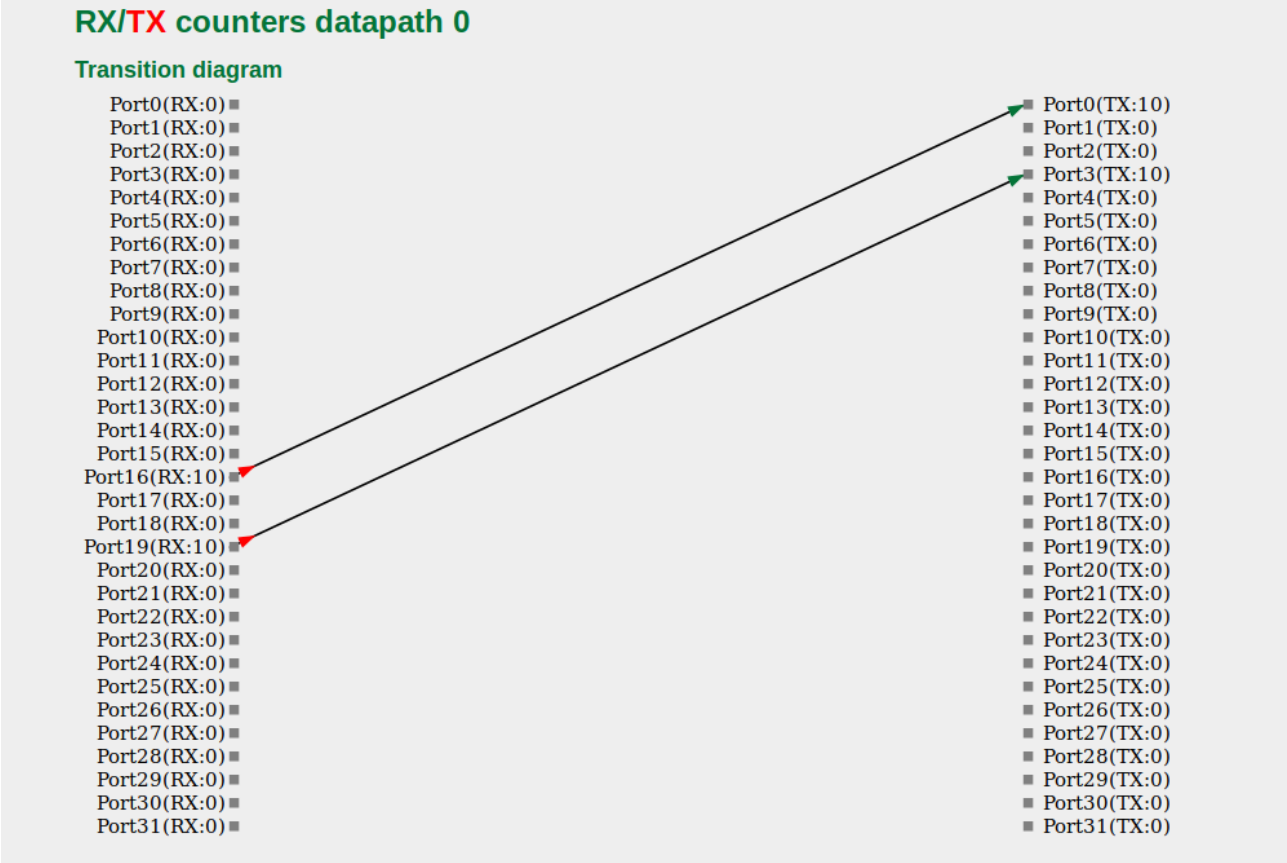

Figure 2: Inport-Outport Transition diagram for datapath 0

Arrowheads symbolize a data flow from the RX Port to the appropriate TX Port.
Incoming RX arrowheads are marked green, while outgoing TX arrowheads are marked red.
Missing arrowheads signal a packet loss. The numbers behind the Port designations indicate the packets flowing per second. "0" means, there are no packets flowing. For instance, there are 10 packets flowing from each Port19(RX) and Port 16(RX) to Port0(TX) and Port3(TX) (see Figure 2).

Furthermore, the Inport-Outport-Statistics page, displays a transition matrix, with TX-Portnumbers vertically and the RX-Portnumbers horizontally. Numbers indicate the packets flowing per second, "-" means, there are no packets flowing at the moment.

7.2 NetConf description

This page provides information about the NetConf interface data structure. The Yet Another Next Generation (YANG)-model can be downloaded on this page.

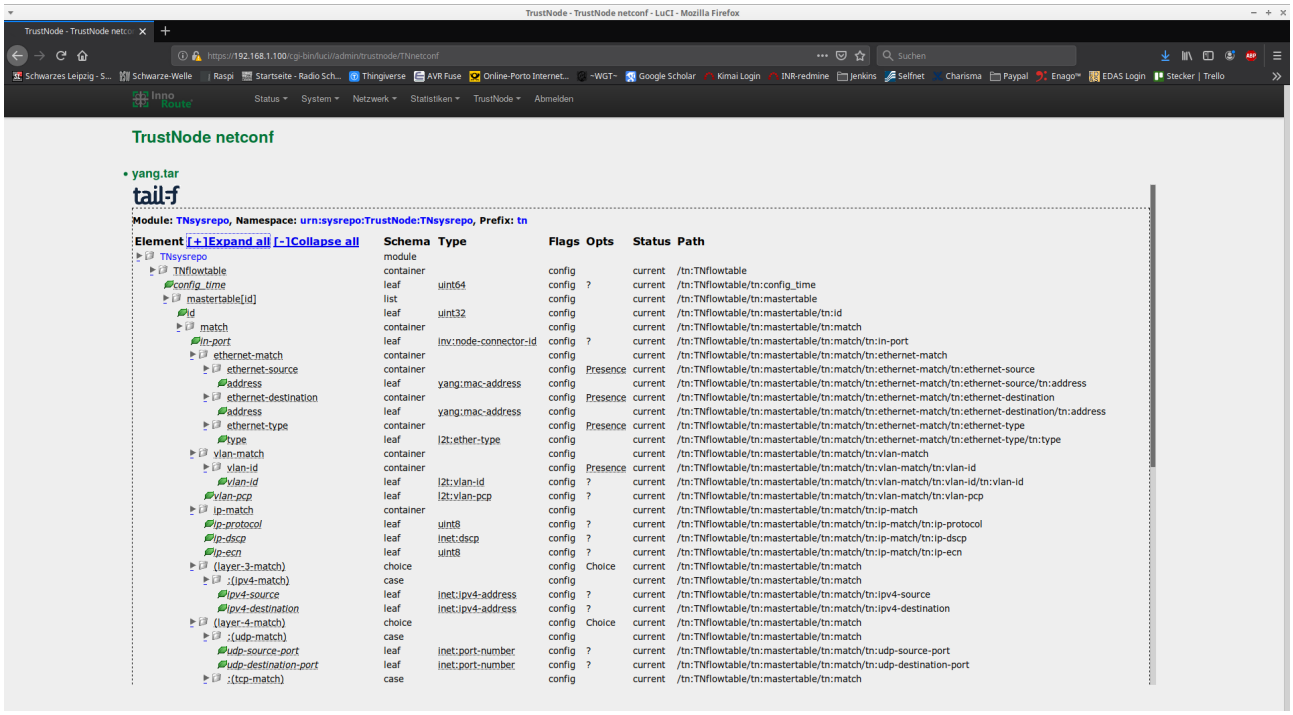

Figure 3: Interactive YANG-model representation at the webUI.

Acronyms

AccDP acceleration datapath

ARP Address Resolution Protocol

CPU Central Processing Unit

DP datapath

DHCP Dynamic Host Configuration Protocol

EMH EXACT MATHING WITH HARDCODED FIELDS

EMA EXACT MATHING WITH ARBITRARY FIELDS

FPGA Field Programmable Gate Array

FTDI Future Technology Devices International

GPIO general purpose input/output

JSON JavaScript Object Notation

JTAG Joint Test Action Group

LPC Low Pin Count

L2 OSI layer 2

L3 OSI layer 3

LLDP Link Layer Discovery Protocol

MMI Memory Mapped Interface

MDIO Management Data Input/Output

OVS Open vSwitch

OSI Open Systems Interconnection

OS Operating System

PCIe Peripheral Component Interconnect Express

PHY Physical Layer Chip

PMod Peripheral Module

PLL Phase Lock Loop

RSA Rivest, Shamir und Adleman

RSTP Rapid Spanning Tree Protocol

RGMII Reduced Gigabit Media Independent Interface

RTC Real Time Clock

SDN Software Defined Network

SNMP Simple Network Management Protocol

SD Secure Digital

TAS Time Aware Shaper

TCXO Temperature-Controlled Crystal Oscillator

TSN Time-Sensitive Networking

TN TrustNode

UEFI Unified Extensible Firmware Interface

USB Universal Serial Bus

XML Extensible Markup Language

YANG Yet Another Next Generation

References

[1] T. Wan and P. Ashwood-Smith. "A Performance Study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv Enhancements". In: *2015 IEEE Global Communications Conference (GLOBECOM)*. 2015, pp. 1–6. DOI: 10.1109/GLOCOM.2015.7417599.