



TrustNode: Beginners Tutorial

December 20, 2019

Robin Kutzner, Marian Ulbricht

InnoRoute GmbH,
Marsstrasse 14a
80335 Munich, Germany

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Setup for basic tests | 3 |
| 2.1 | Setting up the Terminal connection | 4 |
| 2.2 | Setting up the Raspberry Pi | 4 |
| 2.3 | Setting up the TrustNode | 5 |
| 2.4 | Usefull Scripts | 8 |
| 2.4.1 | TN_beep | 8 |
| 2.4.2 | TN_fan.sh | 8 |
| 2.4.3 | TN_text2display.sh | 8 |
| 2.4.4 | TN_buttons.sh | 8 |
| 2.4.5 | TN_statistics.sh | 9 |
| 2.4.6 | TN_fpga_status.sh | 9 |
| 2.4.7 | TN_sys_ctrl_status.sh | 10 |
| 2.4.8 | TN_phy_force_linkspeed.sh | 10 |
| 3 | Webinterface | 10 |
| 3.1 | TrustNode statistics | 10 |
| 3.1.1 | Inport-Outport | 10 |
| 4 | Using the FlowCache | 11 |
| 4.1 | Setup configuration | 11 |
| 4.2 | Creating TNflowtable rules | 11 |
| 4.3 | Matching packet flow | 12 |
| 4.4 | Using an external controller | 15 |
| 5 | Time-Sensitive Networking | 19 |
| 5.1 | GCL entries | 19 |
| 6 | NETCONF | 21 |
| 7 | Setup advanced testbed | 22 |
| 7.1 | Preparation | 22 |
| 8 | Precision Time Protocol | 24 |
| 8.1 | Test PTP functionality | 24 |
| 8.2 | Start PTP on reboot | 25 |
| 8.3 | Test the synchronisation | 25 |

This document gives an introduction into the first steps with the TrustNode.

- Please send your Questions to ulbricht@innoroute.de

For the following tutorials, additional devices with the following capabilities will be needed:

- management wifi network and wifi router with running DHCP server
- two Raspberry Pi¹
- power supply for the Raspberry Pi ²
- TrustNode
- Boot data media for the TrustNode
- USB Network adapter
- some Ethernet cables
- microUSB cable
- free 1G Ethernet port
- full administration rights
- a Debian based operating system is recommended

Figure 1 gives an overview of the tutorial setup. The Raspberry Pi are connected via Ethernet cables to the physical ports 0 and 1 on the front side of the TrustNode.

An additional PC is used for the terminal connection and must be connected via an microUSB cable to the backside of the TrustNode(see Figure 1).

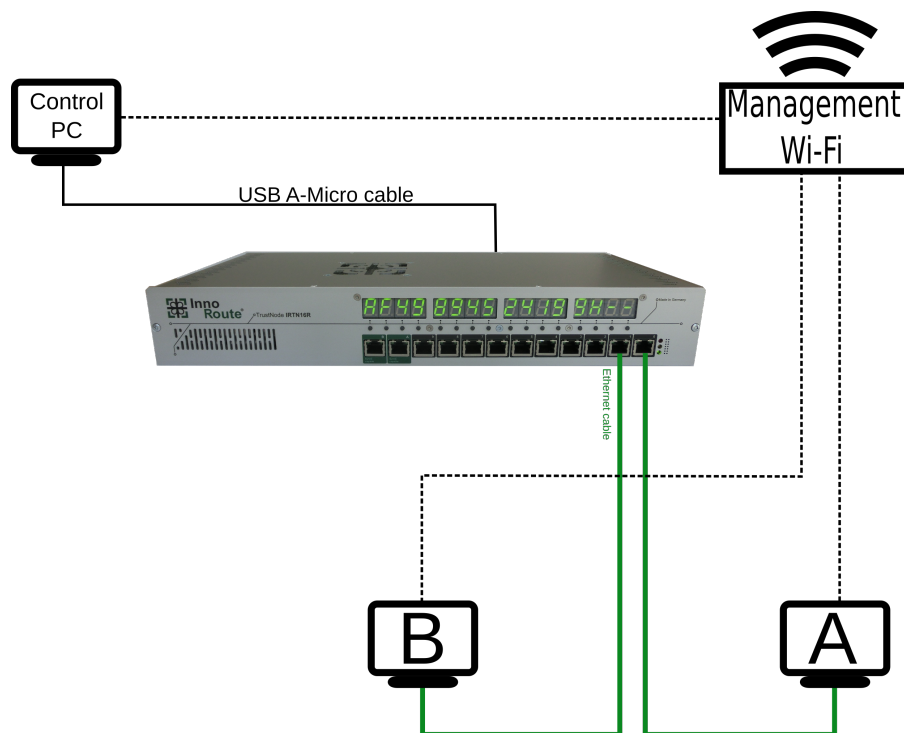


Figure 1: Tutorial setup

¹Raspberry Pi 3B+ is not supported

²should deliver at least 2A

2.1 Setting up the Terminal connection

To have access to the TrustNode root console, start your favourite terminal application³ on the ControlPC.

Select the USB serial connection which is installed if you plug⁴ in the USB-cable into the TrustNode. (normally `/dev/ttyUSB0`) Apply the connection parameters 115200 Baud, 8 data-bits, no parity-bit, 1 stop-bit. After pressing `[↵]`, the prompt *Trustnode login* should be visible. Login with the username *root* and the password *innoroot*. The welcome screen should be visible, as shown in Listing 1.

```
1 | _____
2 | | _ _ | | _ _ \ | _ _ |
3 | | | | _ _ _ _ _ | | _ _ | | _ _ |
4 | | | | ' _ \ | ' _ \ / _ \ | _ _ \ | | | _ _ \
5 | _ | | | | | | | | ( ) | | \ \ ( ) | | | | | _ _ \
6 | | _ _ | | | | | | \ _ _ / | | \ \ _ _ / \ _ _ , \ \ _ _ |
7 | www.trustno.de
8 | Trustnode v1.5
```

Listing 1: Welcome screen

2.2 Setting up the Raspberry Pi

Connect the two Raspberry Pi with the management-wifi-network and assign static IP-addresses to the wire interface of both Raspberry Pi by executing

```
1 | sudo ifconfig eth0 [IPaddress]
```

Then connect from the ControlPC to one of the Raspberry Pi using *SSH* and ping the other Raspberry Pi, using the IP address you just assigned.

³e.g. putty: <http://www.putty.org/>

⁴Plug-in careful and use a strain-relief for free hanging cables.

2.3 Setting up the TrustNode

Plug the USB network adapter in one of the USB ports at the back of the Trustnode. Then connect the USB network adapter with the management-Wifi Router using an Ethernet cable.

If the TrustNode is connected correctly, you will be able to open the GUI of the TrustNode by opening the Internet Browser and connecting to the TrustNode's IP address.

Sign in with username *root* and the password *innoroot*.

The Statistics section (*TrustNode* -> *TrustNode statistics*) shows a Transition diagram of the data flow.

Pinging one of the connected Raspberry Pi from the other one, you should see a dataflow from hardware port 0 to the virtual port 16, port 17 to port 1 and vice versa. (see Figure 2). As written in the manual, all port numbers below 16 are physical ports handled by FPGA ports beginning from 16 are virtual ports which are visible from CPU side.

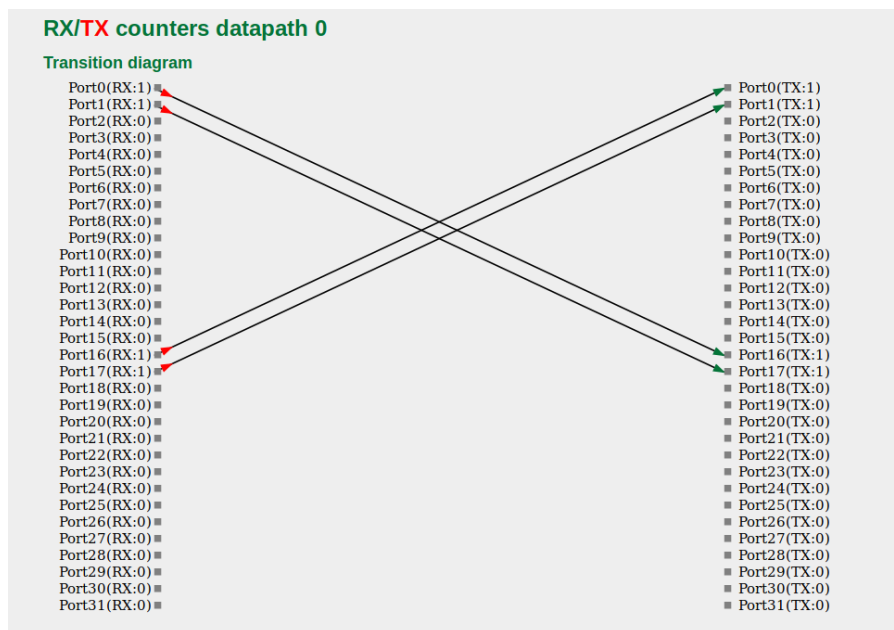


Figure 2: Transition diagram

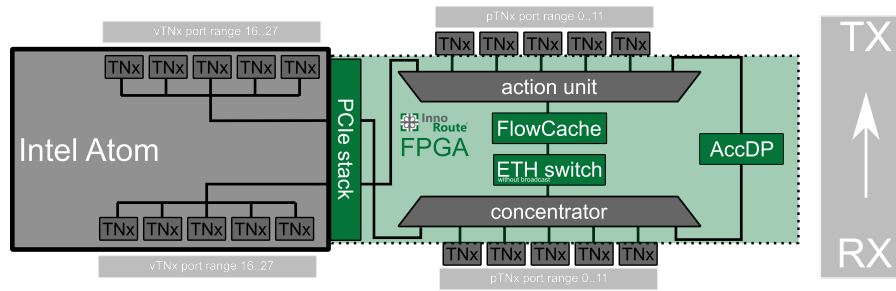


Figure 3: TN concept

Figure 3 shows the packet routing inside the TrustNode. Now, the packets are flowing through the Intel Atom Central Processing Unit (CPU).

```
1 TN_eth_switch.sh 1
```

This command activates the Ethernet switch of the TrustNode (see ETH switch in Figure 3), directing the flowing packets directly from the hardware inport to the hardware output, without passing through the CPU.

Figure 4 shows the datapath for packets flowing, while the ethernet switch is activated.

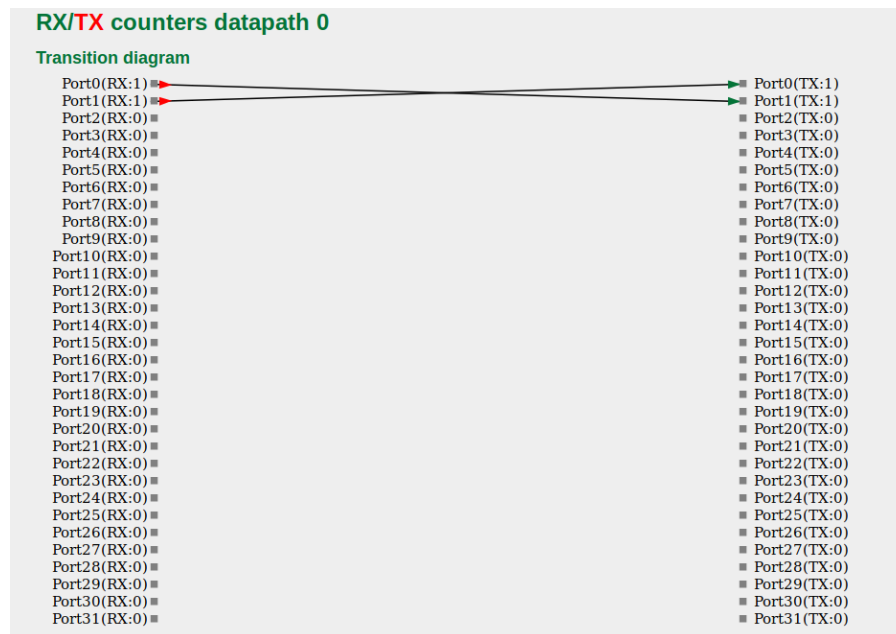


Figure 4: Transition diagram with activated Ethernet switch

In comparison to Figure 2, the packets are now flowing directly from the inport to the output, bypassing the CPU and the virtual ports 16 and 17.

Further Information are listed in the Ethernet switch statistics section of the TrustNode GUI.

```
1 | TN_eth_switch.sh 0
```

This command turns the ethernet switch back off, letting the packets flow through the CPU again.

```
1 | tcpdump -i TN0
```

This command shows a statistics of packets flowing through the CPU, when the ethernet switch is disabled.

2.4 Usefull Scripts

This section is about some usefull scripts for the first steps with the TrustNode

2.4.1 TN_beep

```
1 | TN_beep.sh
```

This command lets the TrustNode make an acoustic signal.

2.4.2 TN_fan.sh

```
1 | TN_fan.sh [ value ]
```

This command let you set the fan speed manually. Set a value between 100 (maximum) and 0 (minimum).

2.4.3 TN_text2display.sh

```
1 | TN_text2display.sh [ text ]
```

This command displays a text at the display on the front side of the Trustnode. You can use small and capital letters, but also numbers 0-9.

2.4.4 TN_buttons..sh

```
1 | TN_buttons.sh [ argument ]
```

This command displays the identification number of the buttons at the front side of the TrustNode. Executing the command without giving any argument, will show the help page. For using it, attach any character to the command.

For the best results, it is necessary to press the button and execute the command at the same time or press and hold down the button just befor executing the command.

```
1 | TN_statistics.sh [argument]
```

This command displays important statistics.

You can use the following arguments for displaying the statistics:

- *io_counters* for information regarding the counters about sending and receiving ports inside the FPGA
- *port_counters* for information about the packets flowing at each RX and TX port
- *bad_counters* for information about bad packets
- *flow_counters* for information about the packets per FlowID
- *phy_counters* for information about the physical layout chips (PHY)
- *-help* or *-usage* for further information

These statistics and further information can also be found at the Web-GUI of the Trustnode at *TrustNode -> TrustNode statistics*.

```
1 | TN_fpga_status.sh [argument]
```

This command displays information about the FPGA. Valid arguments are 0,1,2,3.

Use the argument *0* for standard amount information about the FPGA, like status-information, features and parameters, status and events, as well as the MMI status.

Use the argument *1* for minimum amount of information about the common MMI status.

Use the argument *2* for maximum amount of information output.

Use the argument *3* for error status only.

2.4.7 TN_sys_ctrl_status.sh

```
1 TN_sys_ctrl_status.sh
```

This command displays several outputs of the system controller.

2.4.8 TN_phy_force_linkspeed.sh

```
1 TN_phy_force_linkspeed.sh [argument]
```

With this command, you can set the linkspeed of the PHYs manually.

Use the argument *0* to set the PHY speed to 10 Mbps

Use the argument *1* to set the PHY speed to 100 Mbps

Use the argument *2* to set the PHY speed to 1000 Mbps

You can see the current PHY speed in the *PHY and MAC states* section of the Web-GUI of the TrustNode (*TrustNode -> TrustNode statistics*)

Webinterface

3.1 TrustNode statistics

To access the statistics section, you have to log into the UserInterface of the TrustNode and go to *TrustNode -> TrustNode statistics*.

3.1.1 Inport-Output

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.

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.

Furthermore, the Inport-Output-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.

For further information see the usermanual.

This section shows the configuration of the FlowCache⁵ for forwarding packets between the Raspberry Pi.

4.1 Setup configuration

For this section, you should use the setup from the previous part of this tutorial (see Section 2) with disabled Ethernet switch.

4.2 Creating TNflowtable rules

After the configuration is done, ping one Raspberry Pi by the other. Now switch to the Web-UI of the TrustNode and take a look in the Inport-Outport statistics. It should look like in Figure 2.

Now set up a Flowtable rule, which outsources the routing between the ports, affected by rule, to the FPGA, packets will not pass the CPU, but are routed through the hardware only. There are four different types of rules. Each type of rule comes with different parameters, it is mandatory to use all parameters of the concerned rule in order to create a correctly working rule. You can find detailed information about the different rules and parameters by in the help section of the TNflowtable command (*TNflowtable -help*)

Now create a new Flowtable Rule for data flow from port 1 to port 0 and one for the ping return from port 0 to port 1 by executing the commands:

```
1 TNflowtable add -I1 -O0
2 TNflowtable add -I0 -O1
```

This command creates a new rule, sending packets from directly from inport 1 directly to outport 0 and the other way around. If you now take a look at the inport-outport statistics section, you will see that the packets are flowing from port 1 to port 0 (see Figure 5).

For deleting the rule, you just created, just execute the command

```
1 TNflowtable del -I1 -O0
2 TNflowtable del -I0 -O1
```

After deleting the Flowtable rule, you will see, that the packet flow returned to it's previous state. Rebooting the TrustNode will result in a deletion of all rules.

⁵Note: The capabilities of the flowcache depends on the used system image and on the loaded bitstream. Some Images contain only a light version of the flowcache with a subset of matching fields. The [TNflowtable](#) tool will inform if a rule doesn't fit to the system capabilities. To find out the supported rules, run [TNflowtable -help](#).

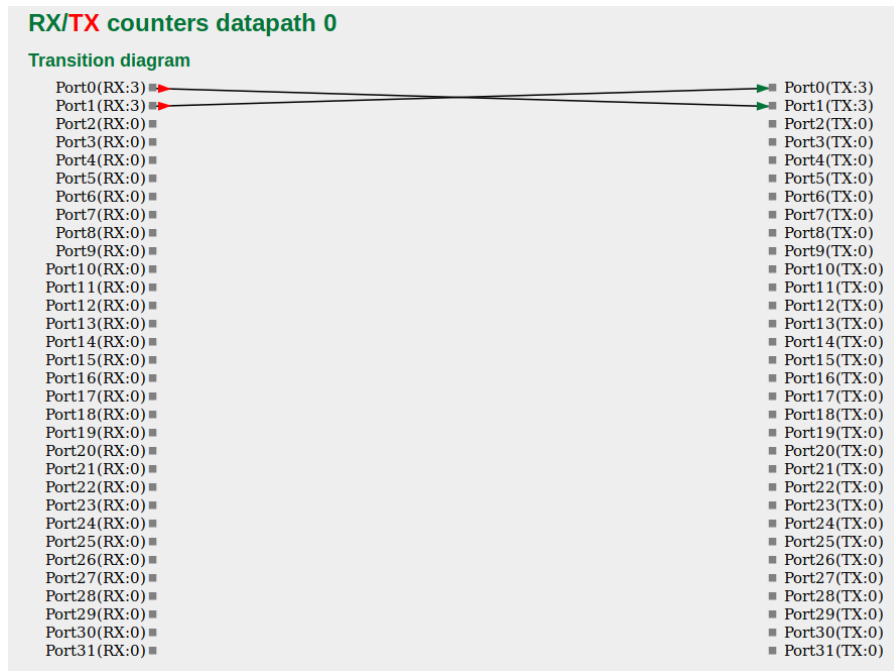


Figure 5: Flowtable Rule 1

4.3 Matching packet flow

In Section 4.2, we treated with matching all packets from one port, to another port.

In this section, we will take a look at matching specific packets. After having deleted all rules, created previously (rebooting the TrustNode is recommended), transfer this [File](#) to the root directory of one of the connected Raspberry Pi ⁶, using SCP. In pursuance of getting the best possible results, stop pinging the other Raspberry Pi now.

After having transfered the file sucessfully, execute the following command on the Raspberry Pi⁷.

```
1 | sudo tcpreplay -i eth0 --loop=0 tutorial1.pcap
```

If the command was executed correctly, you should see that the lights on both ethernet ports are flashing quickly. Now switch over to the Web-UI of the TrustNode and take a look at the Transition diagram of the statistics page.

Figure 6 shows the Transition diagram of the packets, now flowing. As you can see, packets are flowing from port 0 to the virtual port 16 and then from virtual port 17 to port 1. To send these specific packets directly from port 0 to port 1, create a Type 4 Flowtable rule.

For further information about the Type 4 Flowtable rule and the arguments needed, see *TNflowtable -help*.

```
1 | TNflowtable add -A6 -E0x800 -P6 -T0.0.0.4 -O1
```

⁶For getting the same results, as they are described in the tutorial, use the RaspBerry Pi attached to port 0

⁷tcpreplay is required

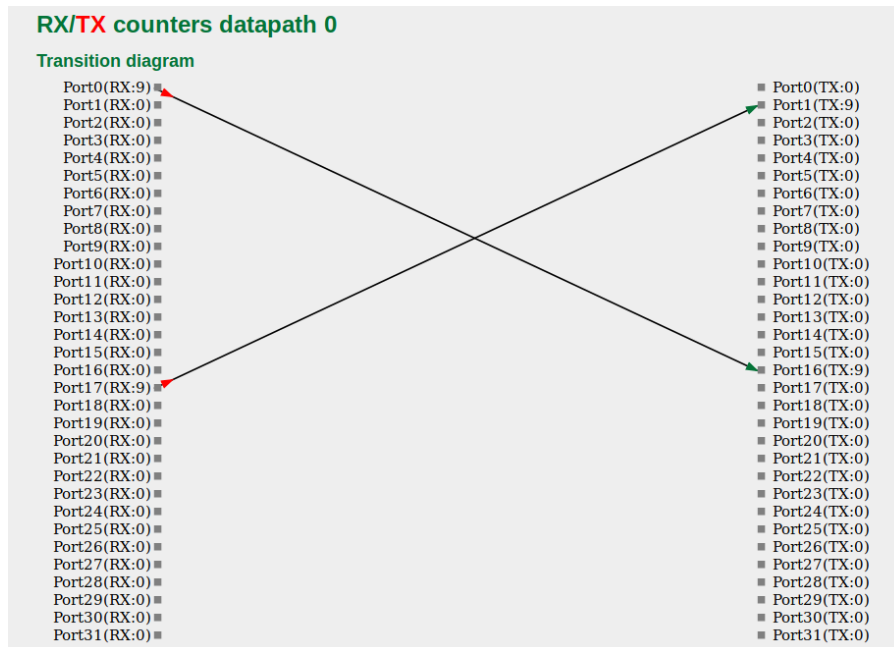


Figure 6: Transition Diagram without Flowtable rule

If you now take a look at the statistics page again, you will see that the packets are now flowing directly from port 0 to port 1, passing the only the hardware. The virtual ports 16 and 17 are not used anymore.

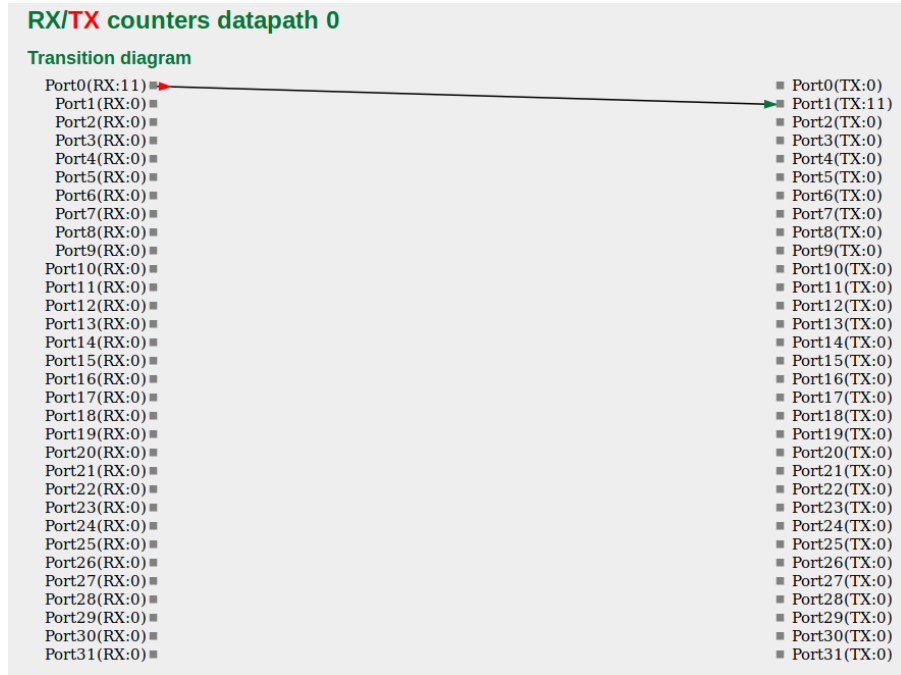


Figure 7: Transition Diagram with Flowtable Rule 4 applied

In comparison the the Type 1 rule (see Section 4.2), Rule 4 only concerns some, specific packets, while Rule 1 affects all packets, passing the concerned port. To illustarte this difference, start a ping with the other RaspBerry Pi, while Rule 4 is applied TCPReplay is still running. You should see that the packets, affected by Rule 4 are still routed through the hardware and don't pass the virtual ports, while the other packets (ping), which are not affected by Rule 4, still pass the virtual ports 16 and 17.

Depending on your hardware, it is possible that only Ruletype 1,2,3,4 are supported, you can check this by looking it up in the *TN_fpga_status.sh* (see Section 2.4.6). If "EMA=on", all Ruletypes are supported, otherwise only Ruletype 1-4 are supported.

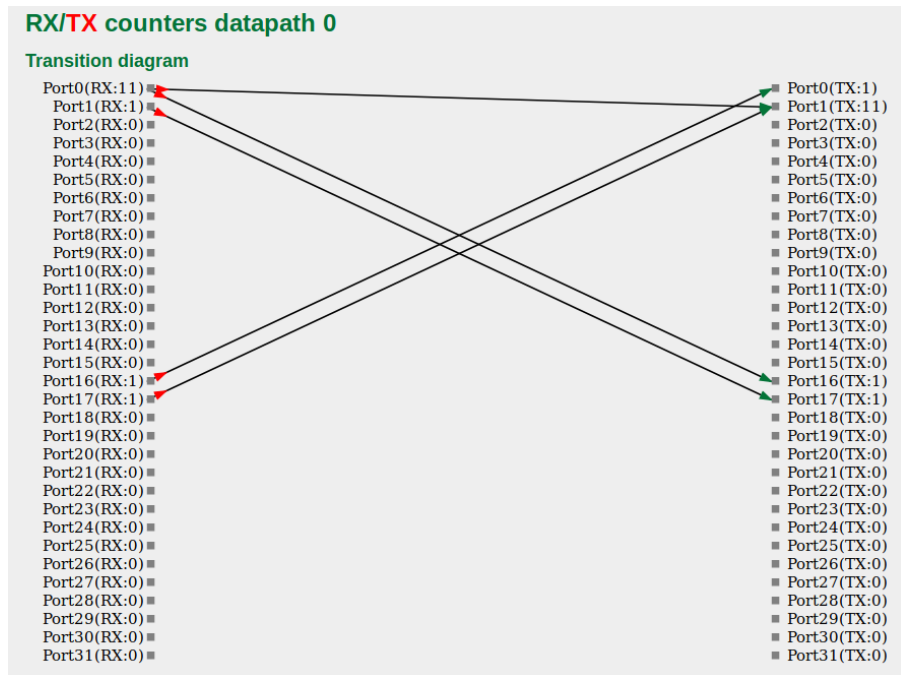


Figure 8: Transition Diagram with Rule 4 applied and Ping

4.4 Using an external controller

After having Section 4.3 completed, we will now go on with using the PC as an external controller for directing packets. For this section of the Tutorial, a Python installation and different packages are required to be installed on your PC. They can be installed by using the following commands:

```
1 sudo apt install python3-dev
2 pip install wheel
3 pip install flask
4 pip install ncclient
5 pip install ryu
```

After having Python and Ryu installed, extract the content of this [Directory](#).

Now, note down your PC's IP-Address in the management-wifi-network and navigate to the TrustN-ode_ryu_example directory and start the *startcontroller.sh*-Script.

Enter the following command on the TrustNode

```
1 TN_ovs_set_ctl.sh [PC's IP]
```

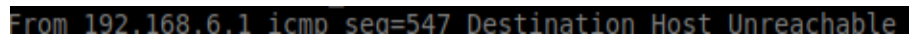
Now, ping one RaspBerry Pi from the other one (for optimal results, use the RaspBerry Pi attached to port 1 for pinging). The PC now acts as an external controller, routing the packets, using the OVS. You should see statistics about the packet flow at the console of your PC. Executing the command


```
1 | ovs-ofctl dump-flows TNbr
```

on your TrustNode, will show you the applied flow rules at the moment. You can delete these rules by typing in:

```
1 | ovs-ofctl del-flows TNbr
```

Either an external controller or flow rules are required in order to route the packets correctly. If you stop the *start_controller.sh* task on your PC now, you will see that the packets from the one Raspberry Pi don't reach the other. This is caused by the absence of flow rules and the controller. In this situation, the TrustNode isn't routing the packets at all. As a result, you should see that the destination host is unreachable for the Raspberry Pi (see Figure 9).



```
From 192.168.6.1 icmp seq=547 Destination Host Unreachable
```

Figure 9: Pinging the Raspberry Pi without using an external controller and no flow rules

The statistics section of the TrustNode-UI also shows that there is no packet flow from port 1 to port 0 at the moment.

Let's now set a flow rule for directing the packets.

To do this, execute the following command:

```
1 ./add_flow_type[ type ]. sh
```

There are 4 tutorial-scripts included for creating a new rule. The 4 different scripts belong to the 4 different flowtypes, supported by the Hardware, for further information see *TNflowtable -help*. For now, we will use script 1, which creates a rule, routing the packets directly from port 1 to port 0. Feel free to take a look into the source-code of the script but note that the ports are numbered starting by 1, while the TrustNode's ports are numbered starting by 0, so output 2 is port 1 of the TrustNode. If you now take a look at the rules for the flow (*ovs-ofctl dump-flows TNbr*), you will see the rule, just created (see Figure 10), you can see the current packet-routing at the statistics-page of the TrustNode too (see Figure 11). Packets from port 1 are now flowing directly to port 0 without passing the virtual ports 16 and 17.

```
root@TrustNode:~# ovs-ofctl dump-flows TNbr
cookie=0x0, duration=17.963s, table=0, n_packets=1, n_bytes=60, priority=1,in port=TN1 actions=set queue:2,output:TN0
```

Figure 10: Flow rule type 1

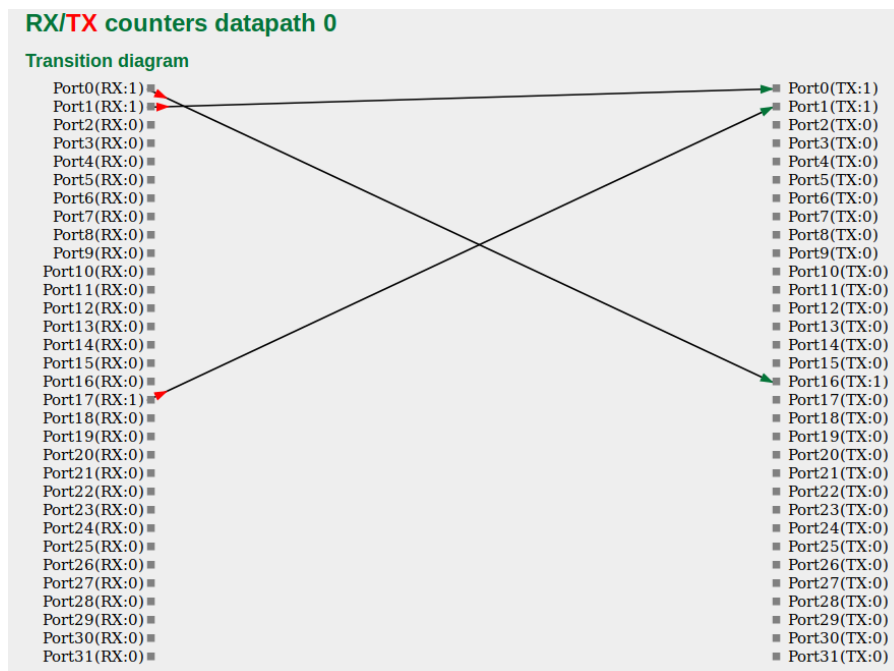


Figure 11: Transition Diagram after adding flow type 1

Now open the source-code of the *add_flow_type1.sh* and swap the values of the *inport* and *outport* and execute the script again. Note that all supported flow rules are applied to the hardware too. *TNflowtable print -c10* shows these flows too.

You can now see that packets are flowing from port 1 to port 0 and back. Take a look into the *ovs-ofctl dump-flows TNbr* and transition diagram again. You will see that packets are now routed directly from port 1 and 0 and back, without passing the virtual ports 16 and 17 at all.

You can see the flow matches, you just saw in the *TNflowtable* script, in the *Flow match counters* section of the TrustNode-UI too (see Figure 12).

| Flow match counters | | | |
|---------------------|------------------------------|-----|------------|
| | DP0 | DP1 | Total |
| RX | FlowID257: 1 FlowID258: 1 | | RXTotal: 2 |
| TX | FlowID257: 1 FlowID258: 1 | | TXTotal: 2 |

Figure 12: Flow match table with flow type 1 applied

As you saw earlier, there is no packet routing happening at all, if there is no external controller connected and no flow rules applied. To guarantee a basic routing of packets, set the fail-mode to standalone by executing the command

```
1 ovs-vsctl set --fail-mode TNbr standalone
```

The packet routing should now work without having a controller connected.

Restarting the TrustNode results in a deletion of all flow rules, applied earlier in the FPGA as well as in the OVS, but the controller's address won't be deleted. To delete the controller connection, use the command

```
1 ovs-vsctl del-controller TNbr
```

The script *remove_flows.sh*, included in the *tutorial-ryu* directory, also deletes the flows at *ovs-ofctl* and in the *TNflowtable*.

This section is about Time-Sensitive Networking and explains basic features and commands. In order to pursue the best possible results, please delete all rules, created previously (rebooting the TrustNode is highly recommended).

5.1 GCL entries

For this section of the tutorial, the basic Tutorial setup from Section 2 is required. In addition, please note the instructions given in Section 4.3.

Now, start the TCPReplay on the Raspberry Pi ⁸ by executing the command

```
1 | sudo tcpreplay -i eth0 --loop=0 tutorial1.pcap
```

Well if the packets are flowing, create a Type 4 Flowtable Rule and add a *-q2* argument to the end of the command. This causes the packets to be sorted into queue 2. The TrustNode has got 8 queues for every outgoing port.

```
1 | TNflowtable add -A6 -E0x800 -P6 -T0.0.0.4 -O0 -q2
```

You won't notice any changes by now, except that the packets are routed directly to port 1 and are now stored in queue 2.

To control the gates of the queues, you need to set two entries at the Gate Control List (GCL).

```
1 | TNtsnctl Change_entry -i0 -S0x01 -I500000000 -a2 -P0
```

This command creates the first entry at the GCL. Use the argument *-i[number]* to set the index number of the entry.

-S0x01 means that queue 0 is open. Use the argument *-S[number]* to open one or more of the gates of specific queues. The numbers are binary coded, so for instance *0x02* would open queue 1 and *0x03* both, the first and second queue.

The argument *-I[time]* represents the time, the concerned entry is active. The time is stated in nanoseconds, so we now have created an entry which will be active for 0.5 seconds.

The argument *-a[amount]* represents the amount of entries, you want to create. In this example, 2 entries are gonna be created.

The argument *-P[portnumber]* defines the port, affected by the entry. It is possible to create entries for every port independently.

So now create a second entry for port 0 by executing the command

⁸The Raspberry Pi, attached to port 0

```
1 | TNtsnctl Change_entry -i1 -S0xff -I500000000 -a2 -P0
```

As you see, this is entry number 2 (see *-i1*) which opens up all gates (see *-S0xff*), including the gate of queue 2.

You can now apply both entries by executing the command

```
1 | TNtsnctl apply -b0 -d0 -P0 -C0
```

The basetime is given in the argument *-b*. For this example, a basetime of 0 is used, which represents the earliest possible date. To use to current time, see *phc_ctl /dev/ptp0*.

The argument *-d* is the *cycle_time_extension* (see IEEE802.1Qbv). If you have software-version 896 (from 03.05.2019) or later installed, the argument *-C0* isn't necessary for applying the rules.

You should now see that the Physical Layer Chip (PHY)-light-emitting diode (LED) is flashing for one half of a second, while it doesn't flash at all for the other half of a second. For the time, entry 2 (with the parameter *-S0xff*) is active, all gates (including the gate of queue 2) are open, so the packets of the Raspberry Pi are flowing through the open gate of queue 2. During the other half of the second, the gate of queue 0 is open, but the gate of queue 2 is closed, so there are no packets flowing through queue 2 for this period of time.

If you apply other runtimes for the TAS entries, please keep in mind that the durations of the entries shouldn't be shorter than the runtime of one packet, but in length, they're limited by the size of the buffer. For further information about the TAS, see IEEE802.1Qbv

Please note that all packets from the Central Processing Unit (CPU) are sorted into queue 0. Applying a rule, which closes the gate of queue 0 could lead to serious problems, especially if you are using the time synchronization feature. That's why you should only apply rules to the GCL which will not affect queue 0 (value should be at least 0x01)

This section of the tutorial explains the basic usage and features of NETCONF. You will learn how to use NETCONF to apply flowtable- and Time Aware Shaper (TAS) rules. To use the NETCONF feature properly, you need to install the *ncclient* packet for Python. To do this, execute the following command on your PC ⁹:

```
1 pip install ncclient
```

You will also need the scripts, included in this [directory](#).

Now unzip the directory and open the *testsuite/ncclient/applyconfig.py* script with an text editor of you choice and change the *Host* value (line 4) to the IP address of you TrustNode.

Then send and apply the Extensible Markup Language (XML) file to the TrustNode, by executing the command ¹⁰:

```
1 python applyconfig.py ../xml/flowcache.xml
```

The *flowcache.xml* file has now been transmitted and applied to the TrustNode. If you take a look into the source-code of the *flowcache.xml* file, you will see that it creates a type 1 flowtable rule, sending all packets from port inport 1 to output 0 via queue 1.

For further information about the XML commands, see the *TrustNode-WebUI -> TrustNode -> TrustNode netconf*.

The *tas.xml* file, included in the xml-directory, automatically applies the 2 queue rules, mentioned in Section 5.1 for queue 1. To transfer and execute this XML-file, use the command

```
1 python applyconfig.py ../xml/tas.xml
```

Keep in mind that this script doesn't include the *TNflowtable* command for sorting the packets into queue 1. In order to combine these two scripts, create a new XML file, named *combined.xml* and copy and paste the whole content of the *tas.xml* file. Now switch to the *flowcache.xml* file and select everything between the opening and the ending *TNflowtable* brackets. Copy the selected code and paste it right after the closing *TNtas* bracket (*</TNtas>*).

If you now execute the python script again, you should see that the TrustNode behaves like after having applied both scripts, mentioned earlier, seperately. The PHY LED should now only flash for half a second. For further information see Section 5.1.

⁹make sure, Python2.7 is installed on you PC

¹⁰use Python2.7

For the following tests a more advanced test setup is used. As shown in Figure 13 two TrustNodes are connected with two Ethernet cables. One for management/synchronisation one for data.

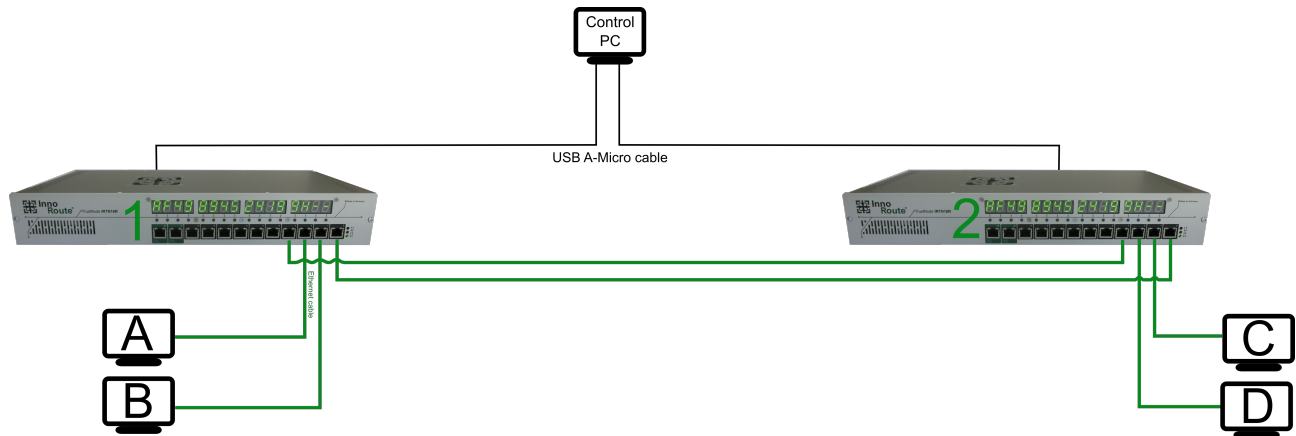


Figure 13: Advanced test setup

Connection list:

- TN1:port0 ↔ TN2:port0
- TN1:port1 ↔ Client-B
- TN1:port2 ↔ Client-A
- TN1:port3 ↔ TN2:port3
- TN2:port1 ↔ Client-C
- TN2:port2 ↔ Client-D

For the following tests the minimal version of the TrustNode image has to be v1.5. The bitstream used must have enabled TSN.

7.1 Preparation

To use the interface TN3 for management and synchronisation messages, the Interface has to be removed from the internal OVS bridge:

```
1 ovs-vsctl del-port TNbr TN3
```

Listing 2: Remove TN3 from TNbr

Run Listing 2 on both TrustNode devices.

Now the IP- and MAC address configuration of TN3 needs to be adapted. Edit the file `/etc/config/network` to apply to following changes:

```
1 config interface 'TN3'
2     option 'ifname' 'TN3'
3     option 'macaddr' '11:11:11:11:11:11'
4     option 'proto' 'static'
5     option 'auto' '1'
6     option ipaddr '192.168.3.1'
7     option netmask '255.255.255.0'
```

Listing 3: TrustNode 1 adapted network config

```
1 config interface 'TN3'
2     option 'ifname' 'TN3'
3     option 'macaddr' '22:22:22:22:22:22'
4     option 'proto' 'static'
5     option 'auto' '1'
6     option ipaddr '192.168.3.2'
7     option netmask '255.255.255.0'
```

Listing 4: TrustNode 2 adapted network config

Apply the settings in Listing 3 and Listing 4 to both TrustNodes. Reboot both devices using `TNchangemod 8` and test the connection using the `ping` tool.

Note: *All changes in `/etc/config/network` and to the Open vSwitch (OVS) are persistent and exist still after a device reboot. All following changes to the TAS and FlowCache will be reset by reloading the bit stream to the FPGA which will happen on every reboot. To store this changes and load the configuration after power on, changes have to be loaded on every startup, e.g. by adding an additional startup script to `/etc/init.d/TN_start`.*

This Section describes the setup of the Precision Time Protocol (PTP) for synchronising the clocks of both TrustNodes.

8.1 Test PTP functionality

To enable the PTP clock synchronisation via port3, run the command showed in Listing 5 on both TrustNodes.

```
1 ptp4l -m -q -i TN3 -f /usr/share/InnoRoute/ptp.conf
```

Listing 5: PTP test comand using the default configuration

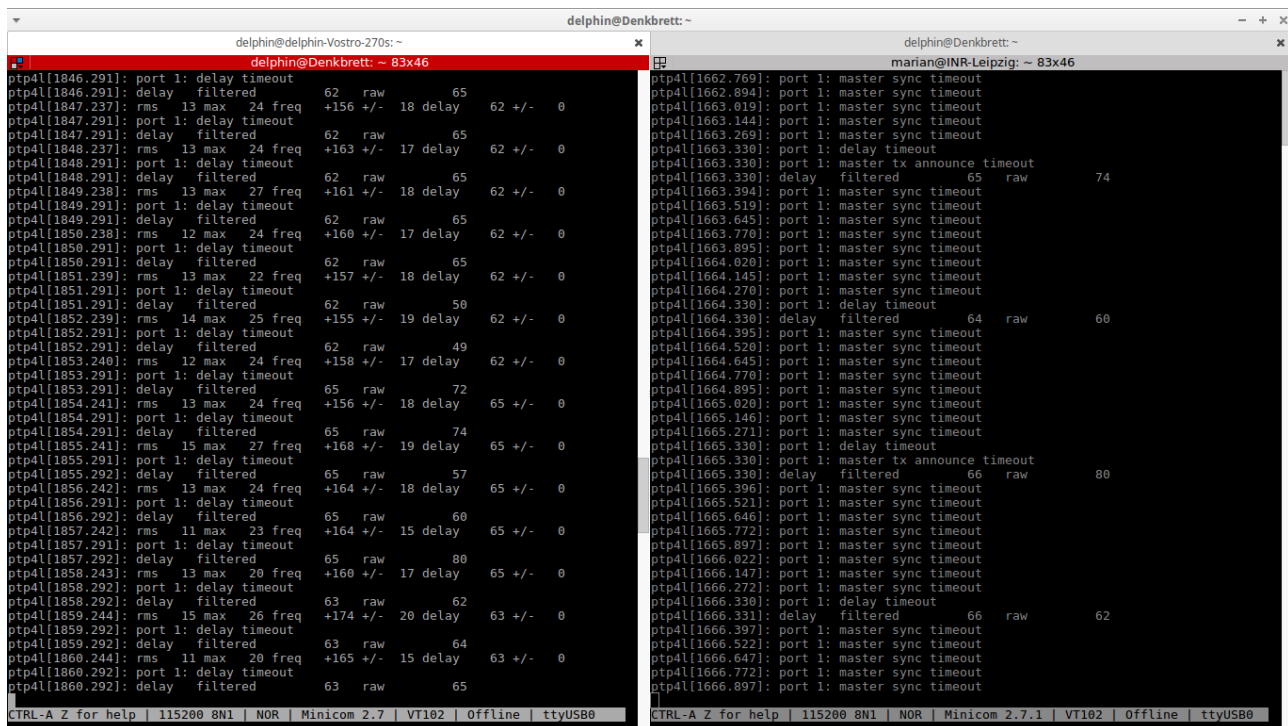


Figure 14: Time synchronisation test on both TrustNodes

Figure 14 shows the `ptp` output on both TrustNodes. One of both devices is automatically selected as master Device. To force one device to be the slave the `-s` Flag can be added to the `ptp4l` command. The configuration file `/usr/share/InnoRoute/ptp.conf` contains standard configurations to start `linuxptp` in OSI layer 2 (L2) P2P mode, the settings could be adapted to tune the synchronisation.

8.2 Start PTP on reboot

To start ptp on reboot we add `ptp4l -q -i TN3 -f /usr/share/InnoRoute/ptp.conf &` to the end of the start-section of `/etc/init.d/TN_start`. To test the settings, reboot both devices using `TNchangemod 8`. After reboot, port3 should show continuous activity. The command `phc_ctl /dev/ptp0 get` can be used to verify the current value on both ptp clocks.

8.3 Test the synchronisation

This is a simple test to verify the synchronisation behaviour. If you have a net-analyser device you can use this for testing, otherwise follow the following steps:

1. remove the cable on port3 of the TrustNodes
2. reboot one of both devices using `TNchangemod 8`
3. run `phc_ctl /dev/ptp0 get` on both devices simultaneous¹¹
4. insert the cable connecting both TrustNodes on port3
5. check again both timestamps using `phc_ctl /dev/ptp0 get`

Without the synchronisation, both clock should be far away, because the FPGA internal timer starts on 0, every time the bitstream is loaded. Using the PTP synchronisation the timestamps should be more nearly equal¹².

¹¹the Linux package *terminator* can be used to launch commands on several shells at the same time: <https://wiki.ubuntuusers.de/Terminator/>

¹²Using this measurement method maximum an 0.1s resolution can be achieved, to verify more accurate you have to use special measurement equipment.

CPU Central Processing Unit

GCL Gate Control List

L2 OSI layer 2

LED light-emitting diode

OVS Open vSwitch

OSI Open Systems Interconnection

PHY Physical Layer Chip

PTP Precision Time Protocol

TAS Time Aware Shaper

XML Extensible Markup Language