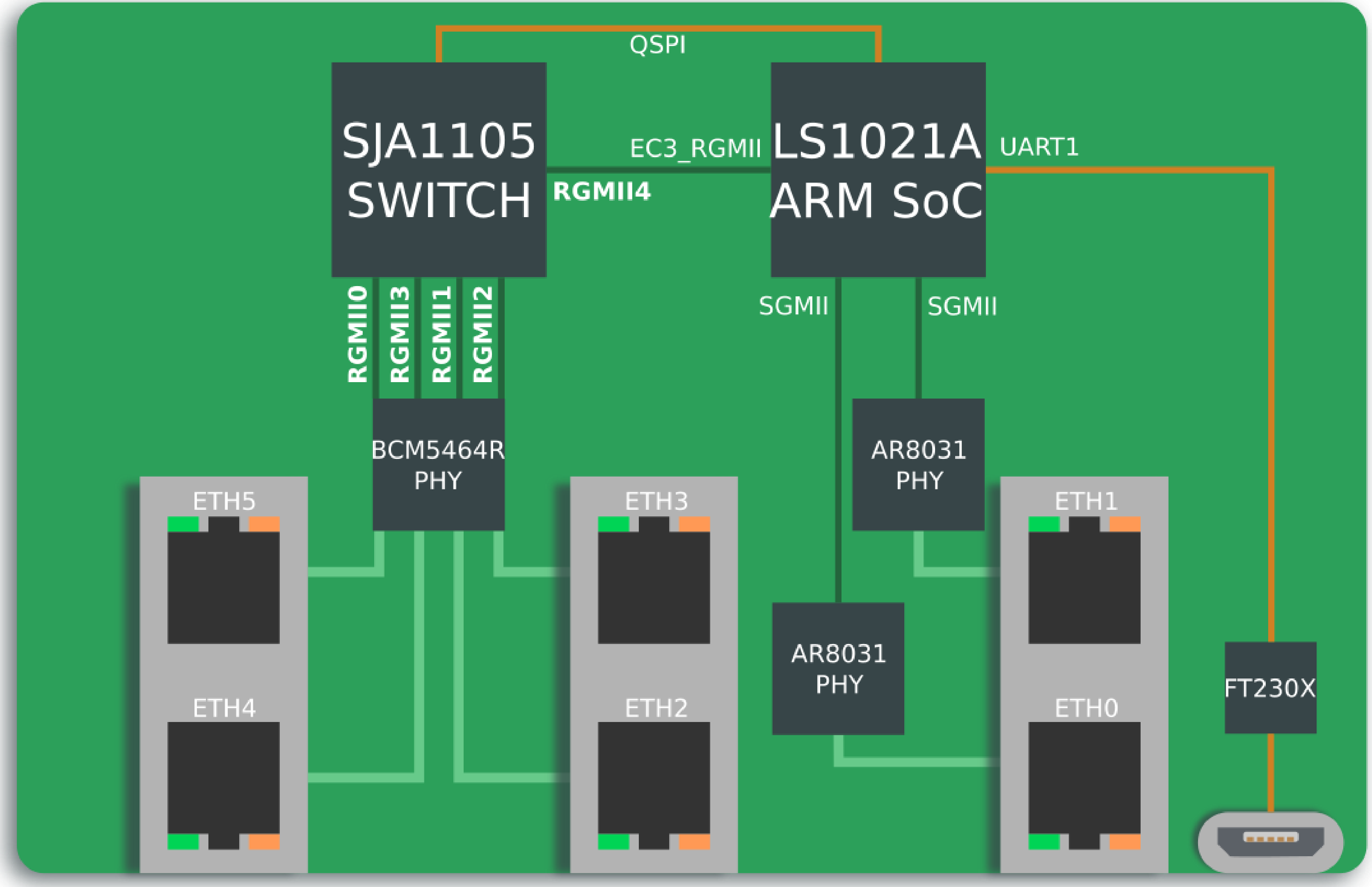
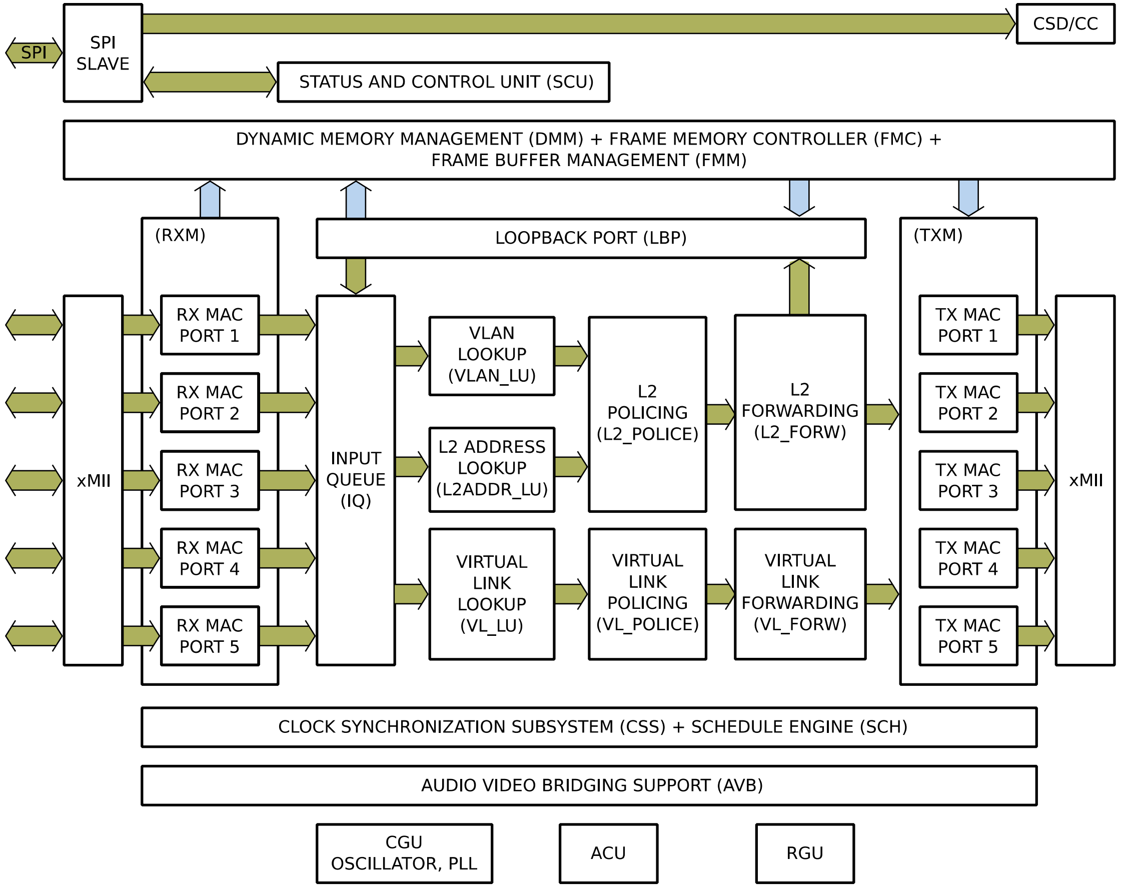
LS1021ATSN Board Overview



NXP SJA1105TEL Overview

- 5 port Ethernet switch for Automotive
  - MII, RMII, RGMII, 10/100/1000 Mbps, full duplex only
  - Store-and-forward architecture
  - 1024 entry MAC address learning table (static or dynamic)
  - Handles frames up to 2KB in size
  - Frame retagging, mirroring and replication
  - Support for 802.1Q VLAN frames and L2 QoS
  - Ingress and egress timestamping per port
  - Hardware forwarding for 1588v2 one-step sync messages
  - Ingress rate limiting (per-stream policing): 802.1Qci
  - Statistics for transmitted, received, dropped frames, buffer load
  - Time-aware traffic shaping: 802.1Qbv

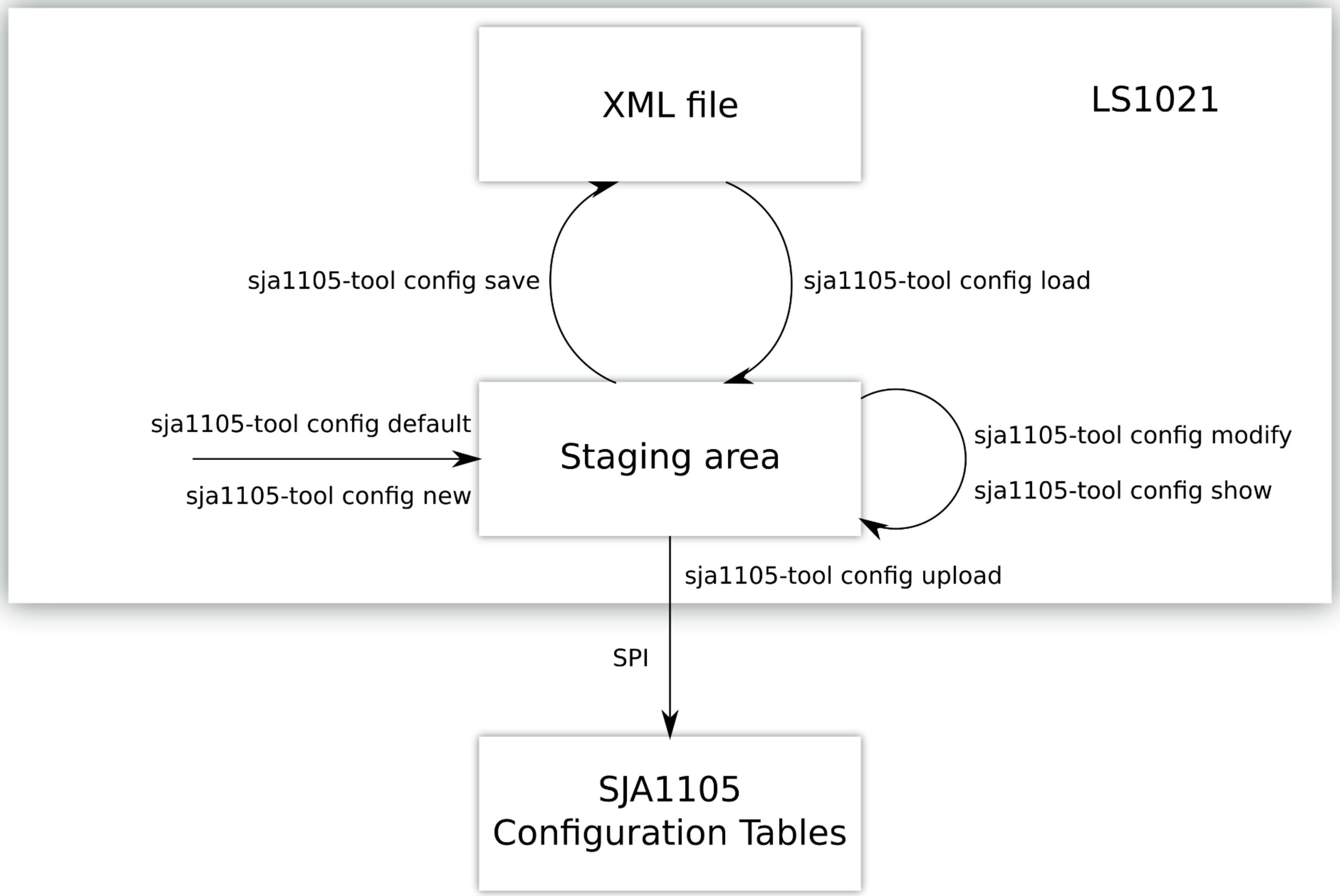

SPI
SPI SLAVE
CSD/CC
STATUS AND CONTROL UNIT (SCU)
DYNAMIC MEMORY MANAGEMENT (DMM) + FRAME MEMORY CONTROLLER (FMC) + FRAME BUFFER MANAGEMENT (FMM)
(RXM)
LOOPBACK PORT (LBP)
(TXM)
xMII
RX MAC PORT 1
RX MAC PORT 2
RX MAC PORT 3
RX MAC PORT 4
RX MAC PORT 5
INPUT QUEUE (IQ)
VLAN LOOKUP (VLAN_LU)
L2 ADDRESS LOOKUP (L2ADDR_LU)
VIRTUAL LINK LOOKUP (VL_LU)
L2 POLICING (L2_POLICE)
VIRTUAL LINK POLICING (VL_POLICE)
L2 FORWARDING (L2_FORW)
VIRTUAL LINK FORWARDING (VL_FORW)
TX MAC PORT 1
TX MAC PORT 2
TX MAC PORT 3
TX MAC PORT 4
TX MAC PORT 5
xMII
CLOCK SYNCHRONIZATION SUBSYSTEM (CSS) + SCHEDULE ENGINE (SCH)
AUDIO VIDEO BRIDGING SUPPORT (AVB)
CGU OSCILLATOR, PLL
ACU
RGU

Block diagram of SJA1105TEL

SJA1105-Tool Overview

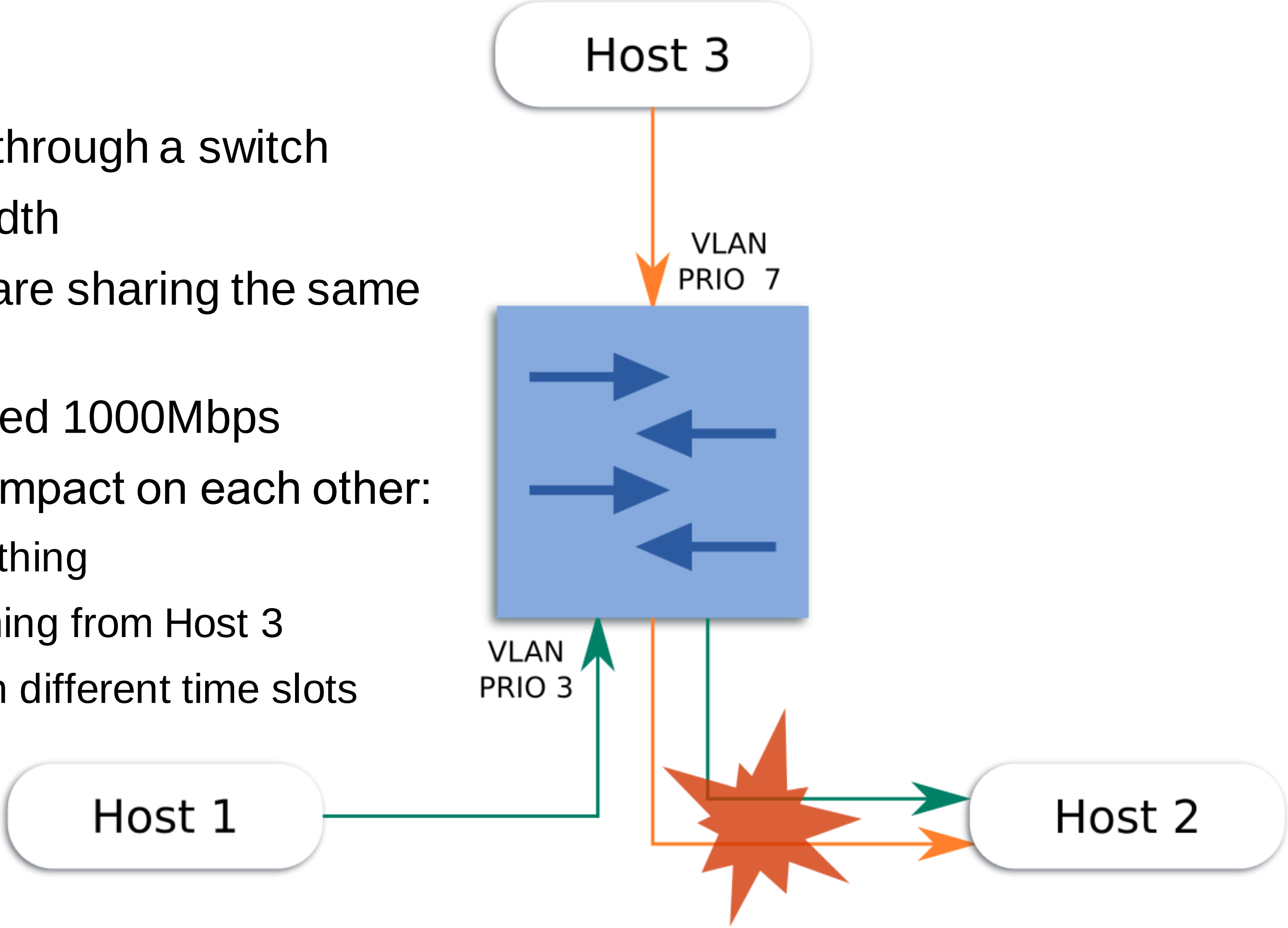
- Linux userspace application for configuring the NXP SJA1105
- The tool supports:
  - Importing a configuration for the SJA1105 switch from an XML file
  - Exporting the current SJA1105 configuration as an XML file
  - Uploading the current SJA1105 configuration to the switch through its SPI interface
  - Inspecting the current SJA1105 configuration
  - On-the-fly modification of the current SJA1105 configuration through command line or scripting interface
- The tool accepts shorthand versions of commands as long as they are unambiguous
  - `sja1105-tool configure show l2-policing-table` can be written as `sja1105-tool c sh l2-pol`

SJA1105-Tool Overview

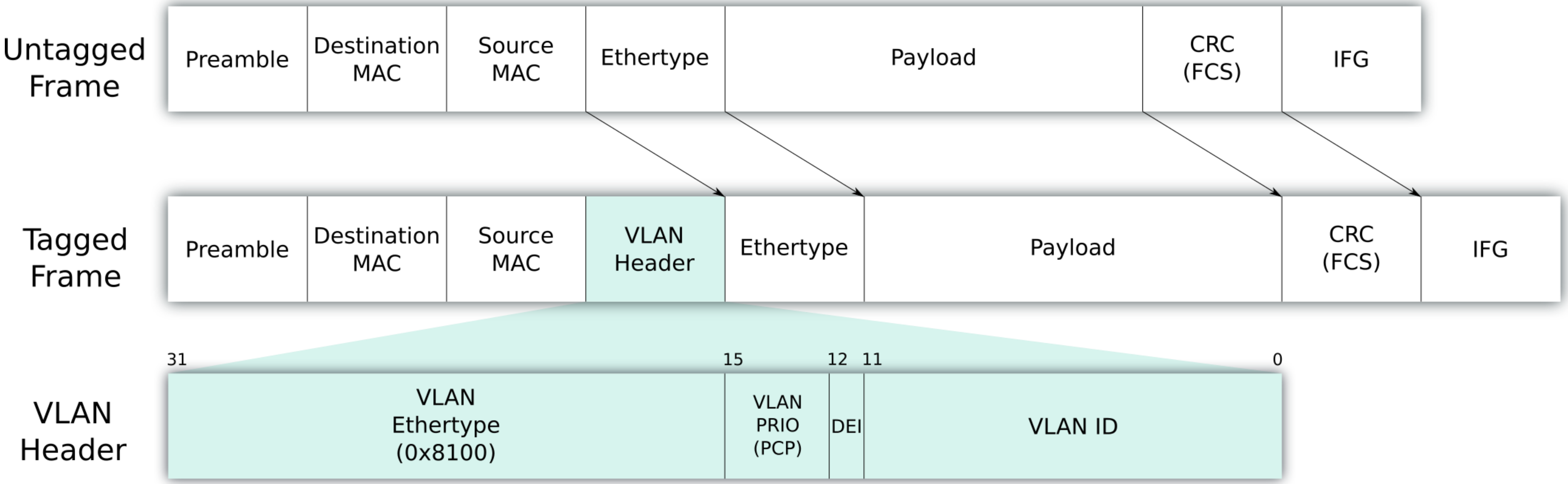


TSN Demo

- Overview
  - 3 host Linux machines connected through a switch
  - 2 TCP flows competing for bandwidth
  - Flows bottlenecked because they are sharing the same link towards Host 2
  - Combined throughput cannot exceed 1000Mbps
  - 3 approaches to isolate the flows’ impact on each other:
    - *Standard* switch configuration: do nothing
    - *Ingress Policing*: rate-limit traffic coming from Host 3
    - *Time Gating*: schedule the 2 flows on different time slots



VLAN Essentials

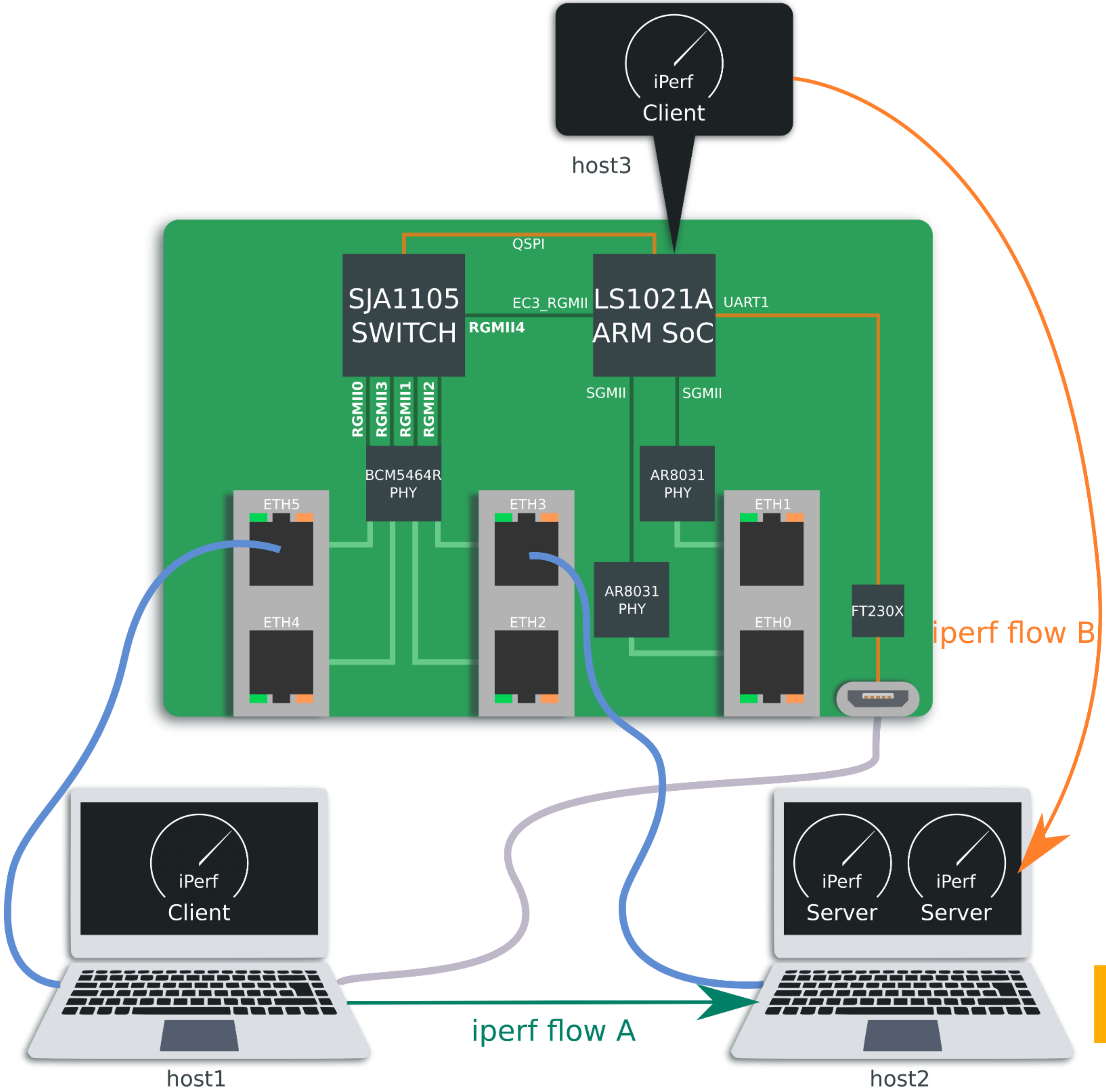


Use of VLAN tags in the demo

- SJA1105 has 3 main stages in its packet processing pipeline:
  - Ingress
  - Forwarding
  - Egress
- To distinguish between different flows, we configure the switch to assign a default (“native”) VLAN header on frames in the ingress stage
- We remove the VLAN tag on the egress stage
- The connected hosts (Host 1, Host 2, Host 3) are oblivious to this VLAN tagging
  - The switch receives untagged frames on ingress
  - The switch sends untagged frames on egress
  - The VLAN tag is only considered during the forwarding stage
- Based on the default VLAN tagging, the flows get differentiated treatment
  - In the policing configuration, one of the flows is rate-limited on the ingress port
  - In the scheduling configuration, each flow gets its own time slot allocated

Host identities

- Host 1
  - Ethernet port ETH5
  - Switch port RGMII **0**
- Host 2
  - Ethernet port ETH3
  - Switch port RGMII **2**
- Host 3
  - LS1021
  - Switch port RGMII **4**



Demo Preparation – iPerf server

- Host 2: Take note of the IP address given by the LS1021 DHCP server:

```
$ ip addr show dev eth0
```

- Start listening on the default TCP port 5001:

```
$ iperf -s
```

Demo Preparation – iPerf flow from Host 1 to Host 2

- Host 1: Run this command to start a TCP iPerf flow towards Host 2 for 100 seconds:

```
$ iperf -c <host-2-ip-address> -i 0.5 -t 100 -f m | tee
/dev/tty | 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"'
```

Demo Preparation – iPerf flow from Host 3 to Host 2

- Host 1: Run this command remotely on Host 3 to start a TCP iPerf flow towards Host 2 for 100 seconds:

```
$ unbuffer ssh -tt root@172.15.0.1 "iperf -c <host-2-ip-
address> -i 0.5 -t 100 -f m" | tee /dev/tty | 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"'
```

Demo Preparation – iPerf3 UDP alternative

- If UDP mode is used, then more relevant statistics can be gathered
  - Jitter = (TRi – TRi-1) - (TSi – TSi-1) = variation of timestamp difference between consecutive packets, at sender vs at receiver
  - https://tools.ietf.org/html/rfc1889#appendix-A.8
  - 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 in they are just reordered)
  - Bandwidth no longer depends on the TCP congestion window size
- But these statistics must now be collected at the server side

Demo Preparation – iPerf3 UDP alternative

- Start two iPerf servers on Host 2:

```
$ mkfifo bw_pipe lost_pipe jitter_pipe
$ unbuffer iperf3 -s -p 5201 –f m | tee /dev/tty | awk -v
host=host1 –f iperf.awk
$ unbuffer iperf3 -s -p 5202 –f m | tee /dev/tty | awk -v
host=host3 –f iperf.awk
```

- Necessary files (`show` and `iperf.awk`) can be found here:

http://sw-stash.freescale.net/scm/dnind/ls1021atsn-demo, branch `embedded-world-show`

Demo Preparation – iPerf3 UDP alternative

- You can also view the statistics graphically with Gnuplot on Host 2:

```
$ ./show bw_pipe 0 1000 "Bandwidth (Mbps)" "UDP throughput for iPerf flows towards Host 2"
```

```
$ ./show lost_pipe 0 100 "Percentage" "UDP Packet loss or reordering"
```

```
$ ./show jitter_pipe 0 1 "Jitter (ms)" "Jitter for UDP iPerf flows"
```

Demo Preparation – iPerf3 UDP alternative

- Host 1:

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

- Host 3:

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

Standard configuration – Default Built-in config

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

```
$ sja1105-tool config default ls1021atsn
$ sja1105-tool config upload
```

- The built-in standard configuration is the same as the one supplied in the rootfs:

```
$ sja1105-tool config save standard.xml
$ sja1105-tool config load /etc/sja1105/standard-config.xml
$ sja1105-tool config save supplied-standard.xml
$ diff –s standard.xml supplied-standard.xml
# Files are identical
```

Standard configuration – Ingress Policer

- The Policer inside SJA1105 is implemented as a Token Bucket Shaper
  - 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
- 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)
- Can also police based on maximum frame size
- L2 Ingress Policer is **“deactivated”**
  - This means that RATE and SMAX are set to maximum (0xFFFF, 0xFA00) for all entries

Standard configuration – Ingress Policer

```
[root@ls1021atsn ~] # sja1105-tool conf show l2-pol
L2 Policing Table: 40 entries
|| Entry 0:             || Entry 1:             || Entry 2:             || Entry 3:             || Entry 4:             ||
|| SHARINDX  0x0        || SHARINDX  0x1        || SHARINDX  0x2        || SHARINDX  0x3        || SHARINDX  0x4        ||
|| SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     ||
|| RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     ||
||                      ||                      ||                      ||                      ||                      ||
|| Entry 5:             || Entry 6:             || Entry 7:             || Entry 8:             || Entry 9:             ||
|| SHARINDX  0x5        || SHARINDX  0x6        || SHARINDX  0x7        || SHARINDX  0x8        || SHARINDX  0x9        ||
|| SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     ||
|| RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     ||
||                      ||                      ||                      ||                      ||                      ||
|| Entry 10:            || Entry 11:            || Entry 12:            || Entry 13:            || Entry 14:            ||
|| SHARINDX  0xA        || SHARINDX  0xB        || SHARINDX  0xC        || SHARINDX  0xD        || SHARINDX  0xE        ||
|| SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     || SMAX      0xFFFF     ||
|| RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     || RATE      0xFA00     ||

....
```

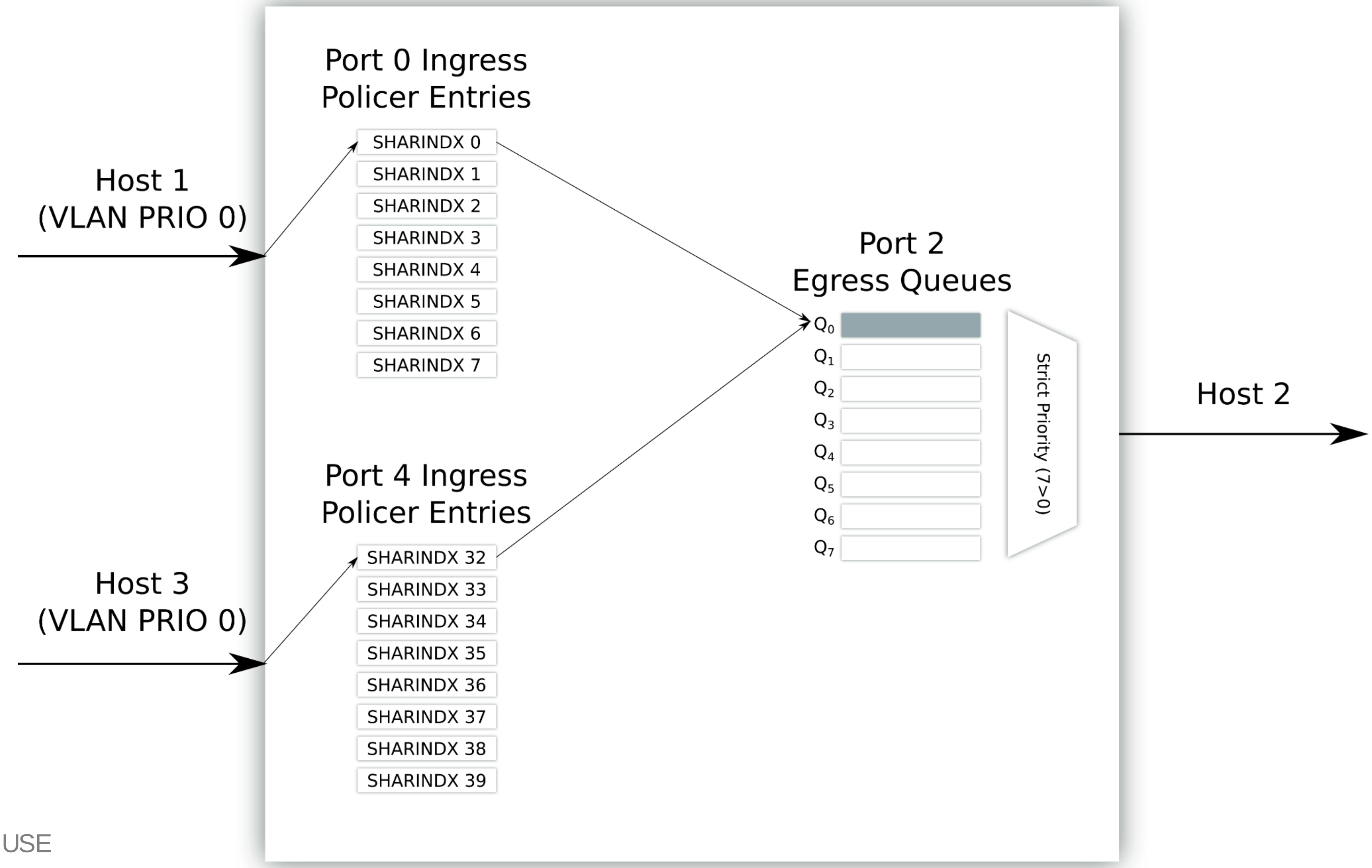
Standard configuration – Native VLAN assignment

- 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
    - This case applies to the demo
- VLAN priorities are taken into consideration for the L2 Forwarding stage
- By default all ingress ports get VLAN priority 0 (best-effort)
- Default behavior is to remove VLAN tags from packets on egress

Standard configuration – Native VLAN assignment

```
[root@ls1021atsn ~] # sja1105-tool conf show mac-configuration-table
MAC Configuration Table: 5 entries
|| Entry 0:                     || Entry 1:                             ||
|| VLANPRIO   0x0               || VLANPRIO   0x0                       ||
|| VLANID     0x0               || VLANID     0x0                       ||
||                              ||                                      ||
|| Entry 2:                     || Entry 3:                             ||
|| VLANPRIO   0x0               || VLANPRIO   0x0                       ||
|| VLANID     0x0               || VLANID     0x0                       ||
||                              ||                                      ||
|| Entry 4:                     ||
|| VLANPRIO   0x0               ||
|| VLANID     0x0               ||
```

Standard configuration – Queuing Diagram



Standard configuration – Conclusions

- Individually, Host 3 gets around 600Mbps and Host 1 gets 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)

Standard Prioritized configuration

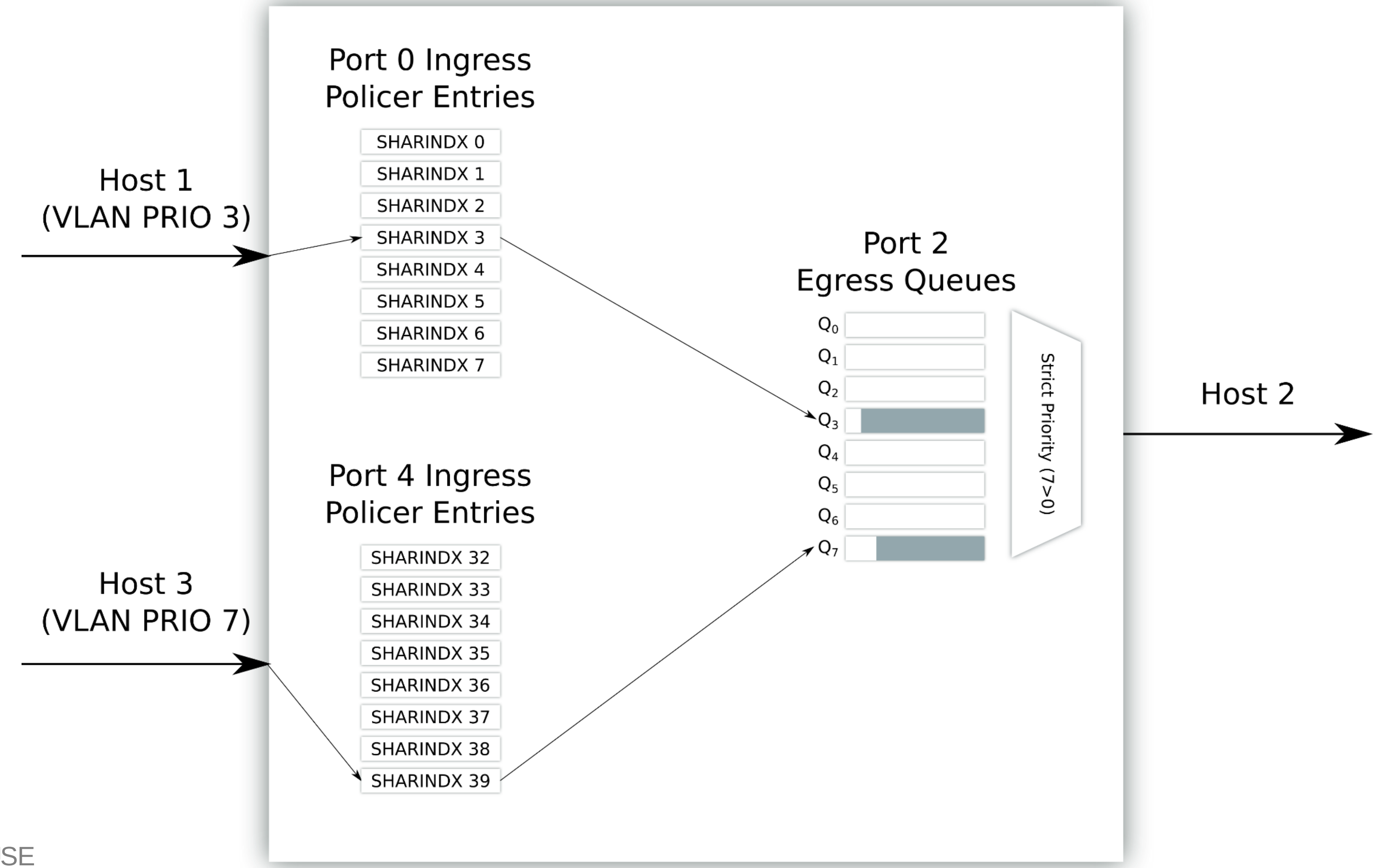
- Assign native VLAN priority 3 to Host 1 and priority 7 to Host 3
  - This is done on a per-ingress port basis (Host 1 -> Port 0, Host 3 -> Port 4)
- On the egress port 2, if Host 3's queue is not empty, the switch will always prefer to send packets from that instead of Host 1's queue

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

Standard Prioritized configuration – Native VLAN assignments

```
[root@ls1021atsn ~] # 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               ||
```

Standard Prioritized configuration – Queuing Diagram



Standard Prioritized configuration - Conclusions

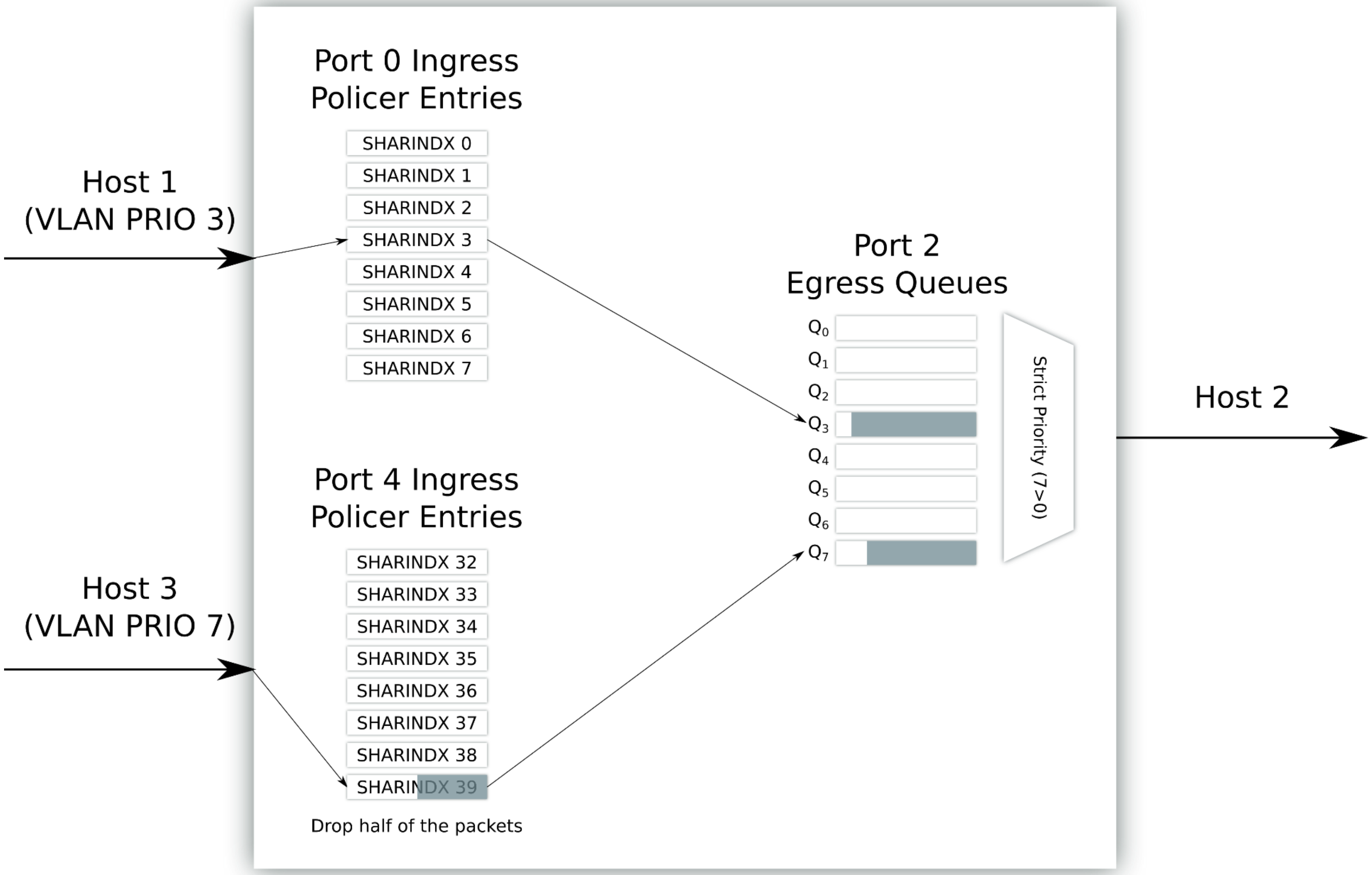
- Host 3 has higher priority, it always gets its 600Mbps
- Host 1 can only get the remaining 400Mbps

Policing configuration

- We can apply rate limiting on Host 3 so Host 1 can get more than 400Mbps
  - Set the *RATE* value for Host 3 to 32000 (half of the maximum value)

```
$ sja1105-tool config load prioritizing.xml
$ sja1105-tool config mod l2-policing-table[39] rate 32000
$ sja1105-tool config save policing.xml
$ sja1105-tool config upload
```

Policing configuration – Queuing Diagram

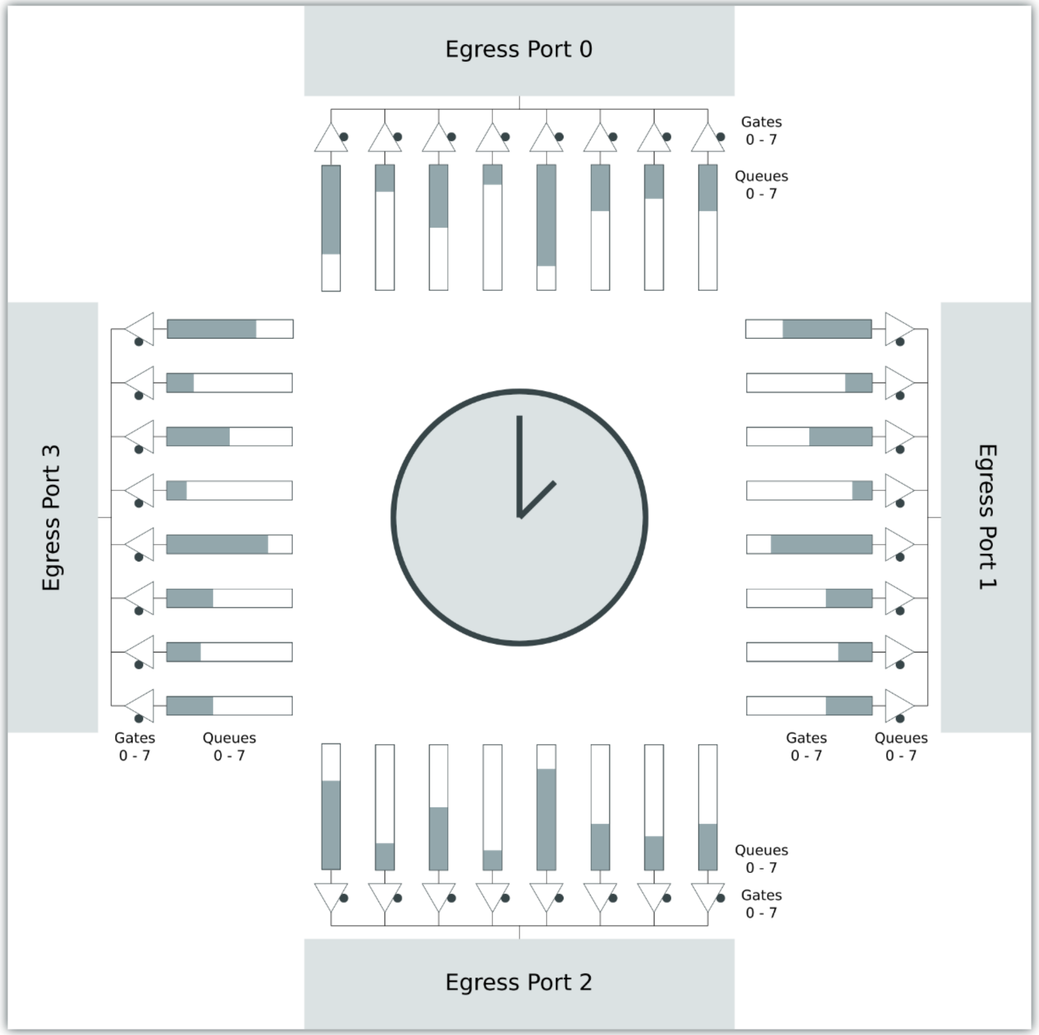


Policing configuration - Conclusions

- Through a combination of prioritization and policing, we can obtain the desired bandwidth allocation for both Host 1 and Host 3 (500-500)
- This is done by dropping half of the packets from Host 3
  - This may not always be desirable

Scheduling configuration: Theory

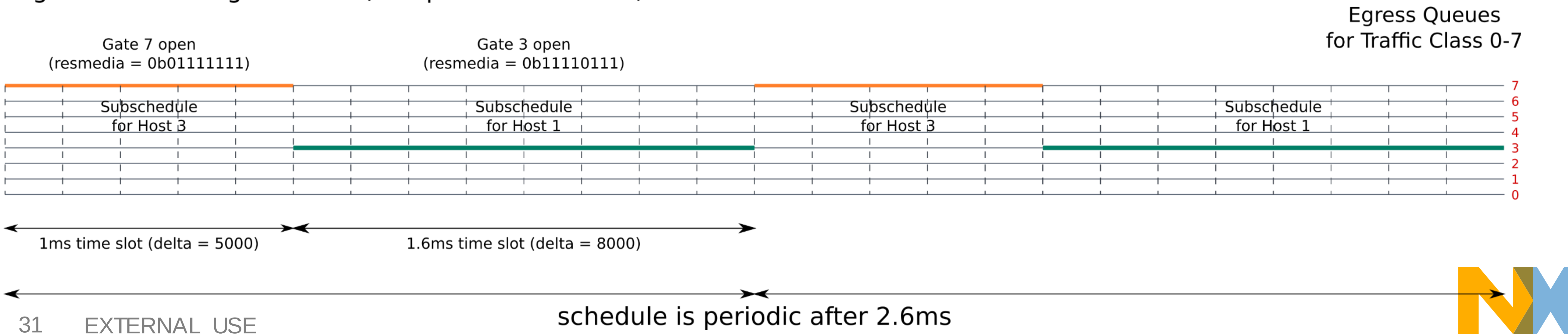
- The Time-Aware Scheduler works by following the guidelines in 802.1Qbv
  - The 5 Egress Ports each have 8 Gates, which can be open or closed
  - Each Gate has 1 Queue associated with it
  - Whenever a Gate is open, packets from that Queue can be sent out the wire
  - An internal clock generates ticks each 200ns
  - At each tick, a new time slot can be created, where some Gates can be opened and some can be closed
- Can be configured to work effectively like TDM (Time Division Multiplexing) for Ethernet



Scheduling configuration: Theory

- The user defines how many clock ticks each time slot takes
  - The individual time slots are called *subschedules*
- Once the Time-Aware Scheduler goes through each time slot in a round-robin fashion, it starts over again periodically
  - A complete period of *subschedules* is called a *schedule*
- On Egress Port 2 (toward Host 2), create a subschedule for VLAN PRIO 7 and one for PRIO 3
  - Host 1 is completely isolated from Host 3
  - Minimal interference, best utilization of bandwidth

Egress Scheduling at Port 2 (destports = 0b00100)



Scheduling configuration: Theory

- 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)
- 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
- We want a single schedule with 2 subschedules
  - Create the 2 subschedules in schedule-table[0] and schedule-table[1]
  - Set the entry point to schedule-table[0] in the schedule-entry-points-table
  - Set the ending point to schedule-table[1] in the schedule-parameters-table

Scheduling configuration: Commands

- Schedule table

```
$ sja1105-tool conf load prioritizing.xml
# Create two subschedules
$ sja1105-tool conf mod schedule-table entry-count 2
# Set both subschedules to be active on egress Port 2 (towards Host 2)
$ for i in 0 1; do sja1105-tool conf mod schedule-table[$i] destports 0b00100; done
$ 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)
$ sja1105-tool conf mod schedule-table[0] resmedia 0b01111111
# Configure first subschedule to keep only gate 3 open (coming from Host 1)
$ sja1105-tool conf mod schedule-table[1] resmedia 0b11110111
# Allocate a 5000*200ns = 1ms time slot for first subschedule (Host 3)
$ sja1105-tool conf mod schedule-table[0] delta 5000
# Allocate a 8000*200ns = 1.6ms time slot for second subschedule (Host 1)
$ sja1105-tool conf mod schedule-table[1] delta 8000
```

Scheduling configuration: Commands

- Schedule table – check if correct

```
$ [root@ls1021atsn ~] # 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       0x1388             || DELTA       0x1F40             ||
||                                ||                                ||
```

Scheduling configuration: Commands

- Schedule Entry Points table

```
# Create an entry in the Schedule Entry Points table
$ 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
```

Scheduling configuration: Commands

- Schedule Entry Points table – check if correct

```
[root@ls1021atsn ~] # sja1105-tool conf show schedule-entry-
points-table

Schedule Entry Points Table: 1 entries
|| Entry 0:                       ||
|| SUBSCHINDX 0x0                 ||
|| DELTA      0x0                 ||
|| ADDRESS    0x0                 ||
||                                ||
```

Scheduling configuration: Commands

- Schedule Parameters table

```
$ sja1105-tool conf mod schedule-parameters-table entry-count 1
$ 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).
```

Scheduling configuration: Commands

- Schedule Parameters table – check if correct

```
[root@ls1021atsn ~] # sja1105-tool conf show schedule-parameters-table
Schedule Parameters Table: 1 entries
|| Entry 0:                                           ||
|| SUBSCHEIND [0x1 0x1 0x1 0x1 0x1 0x1 0x1 0x1 ]      ||
||                                                    ||
```

Scheduling configuration: Commands

- Schedule Entry Points Parameters table

```
$ 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
$ sja1105-tool conf mod schedule-entry-points-parameters-table[0] clksrc 1
# Wrap everything up
$ sja1105-tool config save scheduling.xml
$ sja1105-tool config upload
```

Scheduling configuration: Commands

- Schedule Entry Points Parameters table – 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.
```

Scheduling configuration: Conclusions

- 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

UDP iPerf3 notes

- Client connections can also be started from Windows (Host 1)
- Only the iPerf server (Host 2) needs to run GNU/Linux for Gnuplot
- Need to run 2 separate iPerf servers (port 5201 and 5202) on Host 2
- Reasons for packet loss reported at the iPerf server:
  - Dropped by the Ingress Policer of SJA1105
    - `sja1105-tool status port 4` – counter **`N_POLERR`** – seen in policing-config.xml
  - Tail drop at egress queue towards Host 2
    - `sja1105-tool status port 2` – counter **`N_QFULL`** – seen in {standard,scheduling}-config.xml
  - Dropped at protocol level by iPerf server
    - Out of order packets: `src/iperf_udp.c: iperf_udp_recv()`
    - Server receive timeouts? Seen in scheduling-config.xml

UDP iPerf3 notes

- In scheduling-config.xml, packets coming from Host 3 can arrive:
  - Early: if they were sent in the time slot where Gate 7 was open on SJA1105
  - Late: if they were sent outside of that time slot
- iPerf3 server experiences timeouts (?) and declares late packets as lost
  - Only experimental conclusion so far
- 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

UDP iPerf3 notes

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

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

- Tradeoff is that jitter increases
- This is a non-problem if traffic is always sent in-band
  - Synchronization needed between egress qdisc in Linux TC and the SJA1105 switch
  - LS1021 kernel must always know when its gate is open on the TSN switch, and only send out packets in that interval

Final conclusion

- After running this guide, inside the home directory /home/root there will be 4 XML files:
  - standard.xml
  - prioritizing.xml
  - policing.xml
  - scheduling.xml
- The walkthrough must be followed only once. Afterwards, a specific config can be loaded like this:

```
$ sja1105-tool config load standard.xml
$ sja1105-tool config upload
```

NXP

SECURE CONNECTIONS
FOR A SMARTER WORLD