P4-PSFP: P4-Based Per-Stream Filtering and Policing for Time-Sensitive Networking

Fabian Ihle, Steffen Lindner, Michael Menth

http://kn.inf.uni-tuebingen.de
Time-Sensitive Networking (TSN)

- Subset of standards for transmission in Ethernet based networks

**Features**
- No congestion-based packet loss
- Guaranteed upper bound for latency
- Co-existence with best-effort transmission

**How is this achieved?**
- Streams need to be admitted by the network before transmission (admission control)
- Network reserves resources for transmission
  - Time-based
  - Credit-based
"How can guarantees in TSN be enforced by the network even if individual streams do not adhere to the terms negotiated during admission control?"
Per-Stream Filtering and Policing (PSFP)

- Identifies and monitors streams regarding terms negotiated during admission control
  - Standardized in IEEE Std 802.1Qci

- PSFP acts on streams not adhering to their resource bounds
  - Streams otherwise consume resources reserved for other streams
  - Block out misbehaving / misconfigured streams from the network
  - Drop single violating frames
  - Alter priorities of frames

- TSN guarantees for other streams not at risk!
Per-Stream Filtering and Policing (PSFP)

Stream ID 1
- Priority 3
- Gate ID 1
- Meter ID 1
- Filters
- Counters

Stream ID 2
- Priority 3
- Gate ID 1
- Meter ID 2
- Filters
- Counters

Stream ID 3
- Priority *
- Gate ID 2
- Meter ID 1
- Filters
- Counters

Stream ID n
- Priority *
- Gate ID m
- Meter ID k
- Filters
- Counters

Stream filters

Gate ID 1
- closed
- IPV = 2

Gate ID 2
- open
- IPV = *

Gate ID m
- closed
- IPV = *

Stream gates

Meter ID 1

Meter ID 2

Meter ID k

Flow meters

Queuing

P4-PSFP: P4-Based Per-Stream Filtering and Policing for Time-Sensitive Networking
PSFP Components (1)

► Stream filter
  - Identify streams
    - Assign frame to a stream gate and a flow meter instance via IEEE Std. 802.1CB stream identification
  - Maximum frame size filter

► Stream gate
  - Monitors compliance of streams with negotiated time slices
  - Incoming frame is assigned to a time slice according to its ingress timestamp
  - Represented as stream Gate Control List (stream GCL)
    - Periodically repeated
    - Time slices with open / closed state
      * Open: Forward frame
      * Closed: Drop frame
    - Optional: Internal Priority Value (IPV)
      * Rewrite bridge internal priority
Flow meter
- Monitors compliance with bandwidth (credit-based)
- Token Bucket Policer (RFC2698)
  - Committed Information Rate (CIR)
  - Excess Information Rate (EIR)
- 2-rate 3-color marking
  - **Green**: Tokens from CIR → Forward
  - **Yellow**: Tokens from EIR → Mark *(DropEligibleIndicator)*
  - **Red**: No tokens left → Drop

All PSFP components provide mechanisms to permanently block a stream
Implementation Overview

P4-based PSFP ingress pipeline

1. **MAT_Stream_Identification**
   - Match keys: pkt.SrcMAC (ternary) & pkt.DstMAC (ternary) & pkt.VLAN_ID (exact) & ...
   - Action: 
     - stream_handle
     - stream_gate_id
     - flow_meter_id
     - max_sdu_exceeded_enable
     - close_gate_invalid_rx_enable
     - close_gate_octets_enable
     - mark_all_frames_red_enable
   - Parameters: 
     - assign_psfp_parameters

2. **MAT_Frame_Size_Filler**
   - Match keys: meta.stream_handle (exact) & meta.frame_size (range)
   - Action: 
     - forward
   - Parameters: 
     - -

3. **MAT_Stream_Gate_Control_List**
   - Match keys: meta.stream_gate_id (exact) & pkt.in_timestamp (range)
   - Action: 
     - assign_to_time_slice
     - gate_state
     - ipv
   - Parameters: 
     - set_color
     - drop_on_yellow
     - color_aware

Stream filter
Stream gate
Flow meter
Challenges – Periodicity

► Problems

▪ How to model periodicity of stream GCL?
▪ How to assign an absolute ingress timestamp $t_i$ to a relative position $t_i^{rel}$ in the stream GCL?
  – $t_i^{rel} = t_i \mod h$

No modulo operator available on Tofino!

► Solution

▪ Generate a frame $j$ every $h$ time steps and store the ingress timestamp $t_j^h$ in a register
  – $t_j^h$ reflects the timestamp of the last completed period
▪ $t_i^{rel} = t_i - t_j^h \rightarrow$ semantically a modulo operation!
Implementation Detail

1. Retrieve hyperperiod timestamp and calculate relative position in stream GCL.
2. Truncate timestamp and append frame size → Recirculate.
3. Do PSFP.
4. Generated packet → Store hyperperiod timestamp → Drop

P4-PSFP: P4-Based Per-Stream Filtering and Policing for Time-Sensitive Networking
Time Synchronization and Clock Drifts

Stream GCLs require a highly synchronized network! (order 10ths of µs)
- Synchronize via Precision Time Protocol (PTP)
  - Not supported on our Tofino!

Assume synchronized control plane → Implemented mechanism to sync. data plane to control plane
- Aggregation of all time inaccuracies $\Delta = \delta + \varepsilon_1 + \varepsilon_2$
  - Offset from control plane time $\delta$
  - Clock drifts $\varepsilon_1$
  -Delays $\varepsilon_2$ between stream GCLs of different ingress ports

P4-PSFP: P4-Based Per-Stream Filtering and Policing for Time-Sensitive Networking
Control plane continuously updates $\Delta$

- Writes $\Delta$ into MAT in data plane
- Data plane adjusts time stamps of frames in an atomic operation
- Over-/Underflow handling needed!
Evaluation – Time-based Metering and Periodicity

Traffic generator P4TG [2] feeds 100 Gb/s stream into P4-PSFP

- Apply time-based metering: 1-4-2-1 stream GCL
  - 100 µs open, 400 µs closed, 200 µs open, 100 µs closed
- Apply $\Delta$-adjustment of $\Delta^* = 300$ µs at $t_0$, revert at $t_1$

2 streams generated by P4TG
- 90 Gb/s each with dedicated ingress and recirculation port
  - Blue: stream to measure latency
  - Orange: interfering traffic
- Same egress port
  - Force congestion and queueing
- Continuous CBR traffic as we do not have TSN synchronized talkers

Method
- Measure latency of blue stream for different PSFP configurations
- Drop orange stream after PSFP processing
  - Orange frames occupy the queue but are dropped afterwards
Evaluation – Overloaded network

Latency constantly low for accurate stream GCL scheduling!

![Graphs showing latency over time for an overloaded network.](image-url)
## Evaluation – Scalability

<table>
<thead>
<tr>
<th>Stream identification function</th>
<th>Null stream</th>
<th>Source MAC</th>
<th>IP stream (ternary)</th>
<th>IP stream (exact)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet source address</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ethernet destination address</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>VLAN ID</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IP source address</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IP destination address</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DSCP</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Next Protocol</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Source port</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Destination port</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Max. number of stream identification entries</td>
<td>35840</td>
<td>4096</td>
<td>2048</td>
<td>32768</td>
</tr>
</tbody>
</table>
Conclusion

► First full-fledged implementation of PSFP on real hardware conform to IEEE Std 802.1 Qci
  ▪ May be used until implementation on TSN switches is available
  ▪ Open Source: https://github.com/uni-tue-kn/P4-PSFP

► Functionality of PSFP components could be verified in extensive evaluations
  ▪ Stream ID, credit-based and time-based metering, time synchronization capability

► P4-PSFP implementation effectively eliminates queueing in a congested network environment through highly accurate synchronization of stream GCLs

► P4-PSFP scales up to 35840 different streams

► Implemented concepts can be individually reused in other implementations
  ▪ Periodicity in the data plane
  ▪ Time synchronization where PTP is unavailable