Chair of Network Architectures and Services
School of Computation, Information and Technology
Technical University of Munich

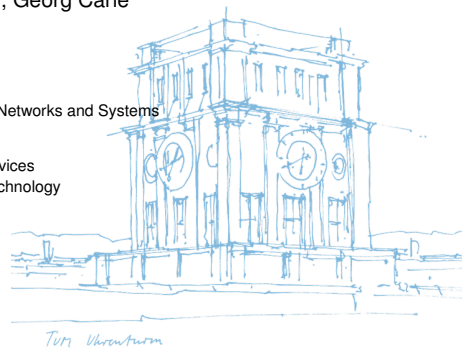
Never Miss Twice – Add-on-Miss Table Updates in Software Data Planes

Manuel Simon, Sebastian Gallenmüller, Georg Carle

Thursday 30th November, 2023

Academic Salon on High-Performance and Low Latency Networks and Systems

Chair of Network Architectures and Services
School of Computation, Information and Technology
Technical University of Munich



State Keeping in Data Planes

- 6G aims for low-latency but high-resilient communication
- State keeping is essential for many applications
- *Registers* (*arrays*) are unstructured memory areas accessible by indices
  - may be fragmented in memory
  - no matching support
  - limited functionality
- In *tables*, structured state can be accessed by sophisticated key matching
- State is often kept by the control plane which decreases performance for state-heavy applications
- We implemented state keeping via *tables* directly in the data plane

P4

- P4 [1] is a domain-specific language for SDN data planes
- In P4, *registers* are changeable within the data plane, *tables* only by the control plane

→ Updatable table entries would increase performance

→ In **previous work** implemented them for the P4 software target *T4P4S* using an `@__ref` annotation [4]

P4

- P4 [1] is a domain-specific language for SDN data planes
- In P4, *registers* are changeable within the data plane, *tables* only by the control plane

→ Updatable table entries would increase performance

  → In **previous work** implemented them for the P4 software target *T4P4S* using an `@__ref` annotation [4]

  → **Here**, we present add-on-miss insertions to tables [3]

P4

- P4 [1] is a domain-specific language for SDN data planes
- In P4, *registers* are changeable within the data plane, *tables* only by the control plane

→ Updatable table entries would increase performance

  → In **previous work** implemented them for the P4 software target *T4P4S* using an `@__ref` annotation [4]

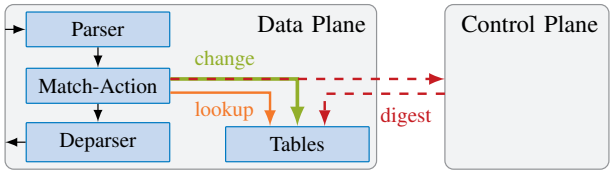
  → **Here**, we present add-on-miss insertions to tables [3]

T4P4S

- *T4P4S* [5] is a hardware-independent transpiler from P4 to C code linked with DPDK developed by ELTE
- The Data Plane Development Kit (DPDK) is an open-source framework enabling fast packet processing in user space
- DPDK performs Receive Side Scaling (RSS) to split traffic among several *lcores*/threads

Table Updates

Digest - Current P4 Way



Current State

- For changes in match-action tables, the data plane has to send a digest to the control plane
  - in *T4P4S*: the controller is a separate process, communication via a socket (low round-trip time (RTT))
- Controller requests data plane to update the table

→ Digest-based approach introduces overhead

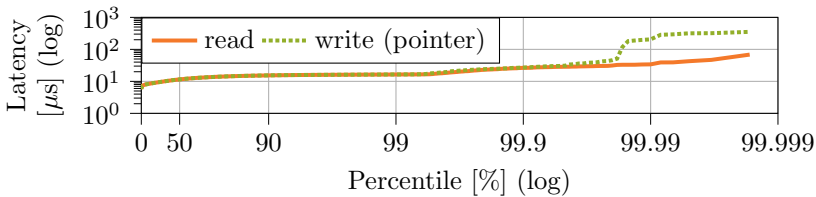
Approaches

- **Digest:** introduces a sleep of 1 second or 1 RTT
  - ⇒ impractical for frequent updates
- **Add-On-Miss:** direct update in the data plane
  - ⇒ avoids the detour over the controller
  - ⇒ improves performance

Previous Work – Changeable Table Entries

- In previous work[1], we implemented updatable table entries
  - `@__ref` annotation to declare parameters as references
- Replaced table architecture for synchronization
- Analyzed different synchronization and storage designs

⇒ Table entry updates possible at line-rate



[1] M. Simon, H. Stubbe, D. Scholz, S. Gallenmüller, and G. Carle: High-Performance Match-Action Table Updates from within Programmable Software Data Planes, *EuroP4 '21* [4]

Add-On-Miss – Implementation

- Upcoming P4 Portable NIC Architecture (PNA) defines new table property: `add_on_miss` and new extern for `exact` matches

Add-On-Miss – Implementation

- Upcoming P4 Portable NIC Architecture (PNA) defines new table property: `add_on_miss` and new extern for `exact` matches

```
table forward {
        actions= {forward, add}
        key = {hdr.eth.srcAddr: exact;}
        add_on_miss = true;
        default_action=add;
}
```

```
action forward(bit<48> dstMac) {
        ...
}

action add() {
        bit<48> dstMac = 0xffffffffffff;
        add_entry<forward_params_t>
                ("forward", {dstMac});
}
```

Add-On-Miss – Implementation

- Upcoming P4 Portable NIC Architecture (PNA) defines new table property: `add_on_miss` and new extern for `exact` matches

```
table forward {
        actions= {forward, add}
        key = {hdr.eth.srcAddr: exact;}
        add_on_miss = true;
        default_action=add;
}
```

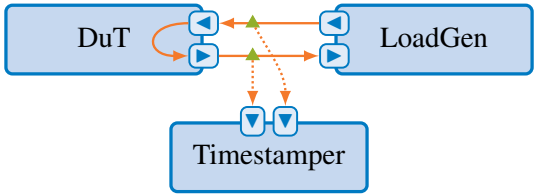
```
action forward(bit<48> dstMac) {
        ...
}

action add() {
        bit<48> dstMac = 0xffffffffffff;
        add_entry<forward_params_t>
                ("forward", {dstMac});
}
```

- For the implementation of them in *T4P4S*, we profit from the adaptions to the synchronization mechanism of the tables done in previous work

Evaluation

Setup



DuT

- Intel Xeon D-1518 2.2 GHz, 32 GB RAM
- Latency optimized *T4P4S*
- `add_on_miss` activated

LoadGen

- MoonGen [2] is used to generate traffic
- Contains key and value of new entry
- Packet size 84 B

Timestamper

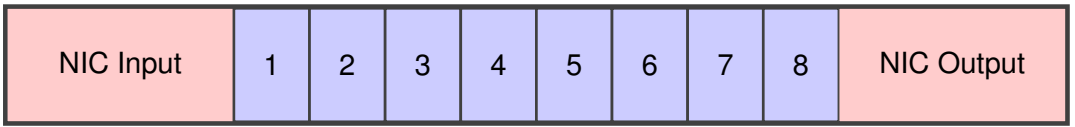
- Packet streams duplicated using optical splitter
- Timestamps each packet incoming packet
- Resolution: 12,5 ns

Evaluation

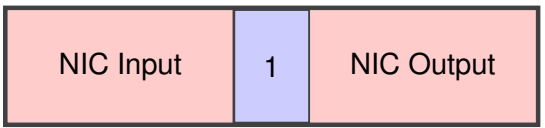
Batched processing

- NIC I/O has nearly constant overhead
- One packet is processed after another

Throughput-optimized → larger batch size

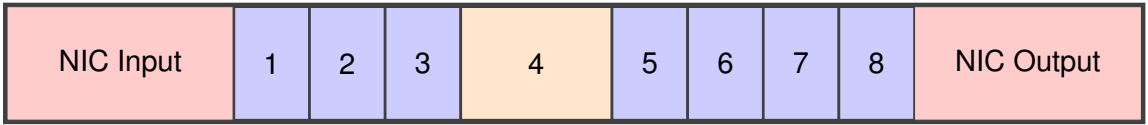


Latency-optimized → minimal batch size



Evaluation

Throughput-optimized → larger batch size



→ Throughput measures average cost per packet

→ Ideal to measure the maximum performance

Latency-optimized → minimal batch size

→ Latency measures single cost for each packet

→ Ideal to measure cost of different operations

Evaluation

Approach

P4 program

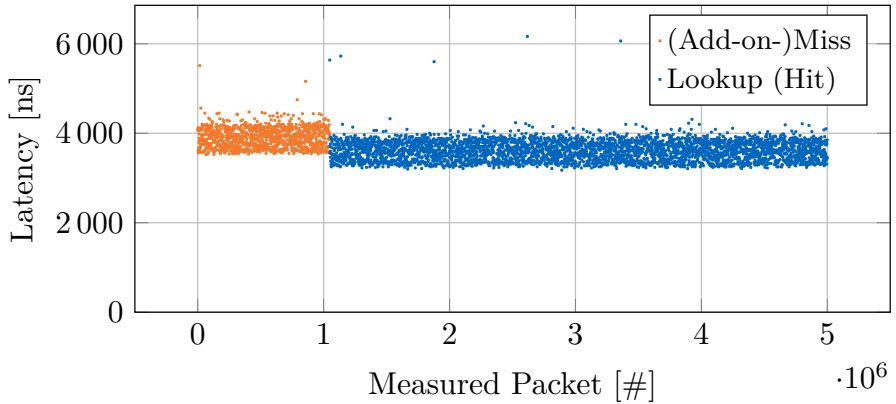
- Each packet contains key and value for a new table entry
- P4 programs contains lookup to this *one* table
- Forward all packets back

Two phases

- Key cycle pseudo-randomly through $[0, 2^{20}]$ several times
- *First phase*: only insertions are performed
- *Second phase*: mainly lookups are performed; some insertions are done with different rates

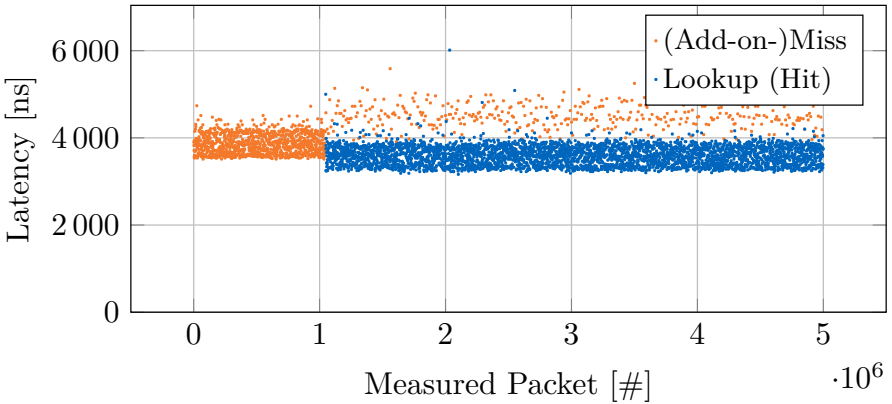
Evaluation

- *First phase*: 2^{20} packets triggering an insertion
- *Second phase*: $\approx 4M$ packets trigger lookup of previously inserted packets



Evaluation

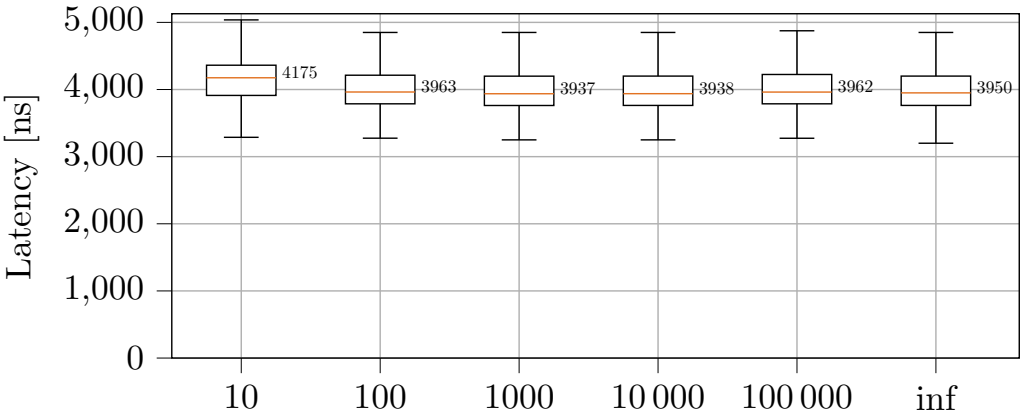
- *First phase*: 2^{20} packets triggering an insertion
- *Second phase*: $\approx 4M$ packets trigger lookup of previously inserted packets
  - But every 10 000-th packet triggers additional insertion



Evaluation

- Different rate of insertions during *second phase*

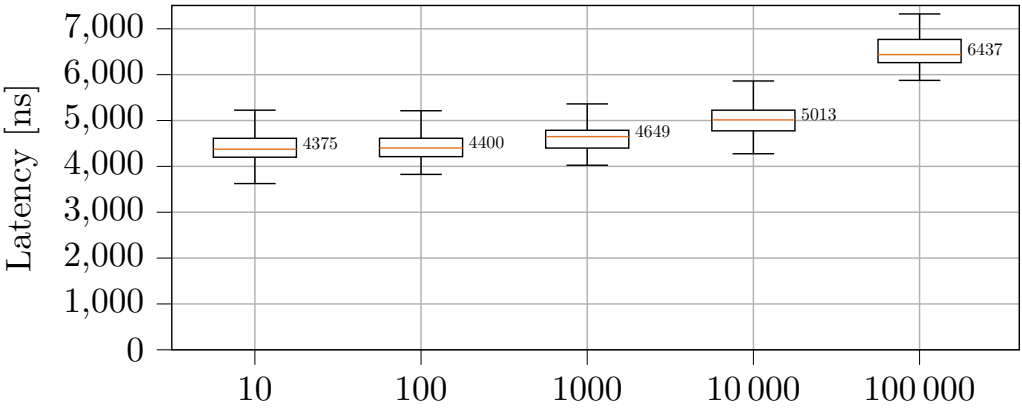
⇒ Median mixed (i.e. insertions & lookups) latency decreases with increasing rate



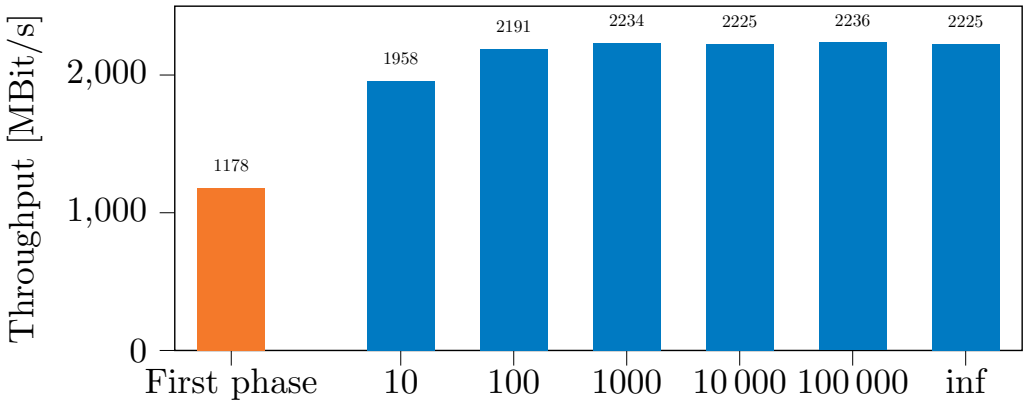
Evaluation

⇒ Insertion latency increases with increasing rate (up to 47 %)

⇒ Worse branch prediction



- 84 Byte Packets



Conclusion

- Adding state to the P4 data plane increases number of possible low-latency applications
  - Updatable Table Entries[2]
  - Add-On-Miss Insertions[3]
- Add-on-Miss insertions enable cheap insertions w.r.t. latency

[2] M. Simon, H. Stubbe, D. Scholz, S. Gallenmüller, and G. Carle: High-Performance Match-Action Table Updates from within Programmable Software Data Planes, *EuroP4 '21* [4]

[3] M. Simon, S. Gallenmüller, and G. Carle: Never Miss Twice – Add-on-Miss Table Updates in Software Data Planes, *WueWoWas '23* [3]

Conclusion

- Adding state to the P4 data plane increases number of possible low-latency applications
  - Updatable Table Entries[2]
  - Add-On-Miss Insertions[3]
- Add-on-Miss insertions enable cheap insertions w.r.t. latency

- Is this a step backwards in SDN ?

[2] M. Simon, H. Stubbe, D. Scholz, S. Gallenmüller, and G. Carle: High-Performance Match-Action Table Updates from within Programmable Software Data Planes, *EuroP4 '21* [4]

[3] M. Simon, S. Gallenmüller, and G. Carle: Never Miss Twice – Add-on-Miss Table Updates in Software Data Planes, *WueWoWas '23* [3]

Conclusion

- Adding state to the P4 data plane increases number of possible low-latency applications
  - Updatable Table Entries[2]
  - Add-On-Miss Insertions[3]
- Add-on-Miss insertions enable cheap insertions w.r.t. latency

- Is this a step backwards in SDN ?
  - ⇒ **No**, local and global state may work hand-in-hand
  - ⇒ PNA proposal comes from the P4 community
  - ⇒ PNA brings P4 to the NIC of the end-host where state is required anyways

[2] M. Simon, H. Stubbe, D. Scholz, S. Gallenmüller, and G. Carle: High-Performance Match-Action Table Updates from within Programmable Software Data Planes, *EuroP4 '21* [4]

[3] M. Simon, S. Gallenmüller, and G. Carle: Never Miss Twice – Add-on-Miss Table Updates in Software Data Planes, *WueWoWas '23* [3]

Bibliography

[1] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker.
P4: programming protocol-independent packet processors.
Comput. Commun. Rev., 44(3):87–95, 2014.

[2] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle.
Moongen: A scriptable high-speed packet generator.
In *Proceedings of the 2015 Internet Measurement Conference*, IMC '15, page 275–287, New York, NY, USA, 2015. Association for Computing Machinery.

[3] M. Simon, S. Gallenmüller, and G. Carle.
Never Miss Twice - Add-On-Miss Table Updates in Software Data Planes.
In *KuVS Fachgespräch - Würzburg Workshop on Modeling, Analysis and Simulation of Next-Generation Communication Networks 2023 (WueWoWAS'23)*, page 5, 2023.

[4] M. Simon, H. Stubbe, D. Scholz, S. Gallenmüller, and G. Carle.
High-performance match-action table updates from within programmable software data planes.
In *ANCS '21: Symposium on Architectures for Networking and Communications Systems, Layfette, IN, USA, December 13 - 16, 2021*, pages 102–108. ACM, 2021.

[5] P. Vörös, D. Horpácsi, R. Kitlei, D. Leskó, M. Tejfel, and S. Laki.
T4p4s: A target-independent compiler for protocol-independent packet processors.
In *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–8. IEEE, 2018.

Chair of Network Architectures and Services
School of Computation, Information and Technology
Technical University of Munich

Never Miss Twice – Add-on-Miss Table Updates in Software Data Planes

Manuel Simon, Sebastian Gallenmüller, Georg Carle

Thursday 30th November, 2023

Academic Salon on High-Performance and Low Latency Networks and Systems

Chair of Network Architectures and Services
School of Computation, Information and Technology
Technical University of Munich