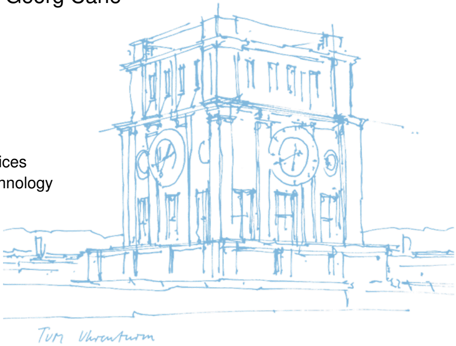
Never Miss Twice – Add-on-Miss Table Updates in Software Data Planes

Manuel Simon, Sebastian Gallenmüller, Georg Carle

Thursday 29th June, 2023

WueWoWas'23

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



Introduction

State Keeping in Data Planes

- 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 [2] 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 [5]

Introduction

Background

P4

- P4 [2] 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 [5]

  → For **this** publication, we implemented add-on-miss insertions to tables

Introduction

Background

P4

- P4 [2] 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 [5]

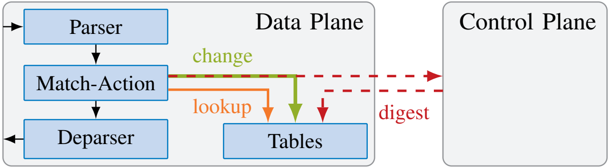
  → For **this** publication, we implemented add-on-miss insertions to tables

T4P4S

- *T4P4S* [6] 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

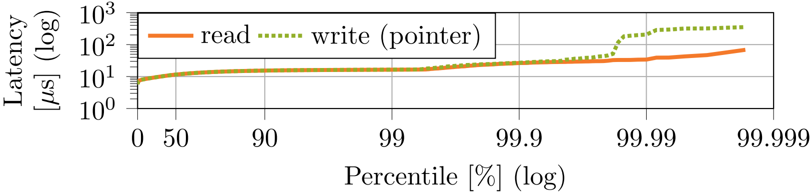
Related Work

- The upcoming Portable NIC Architecture (PNA) [1] will allow adding entries on lookup misses
- FlowBlaze [4] allows state updates in programmable data planes relying on registers
- SwiSh [7] implements a distributed state layer to programmable switches

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* [5]

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;
}
```

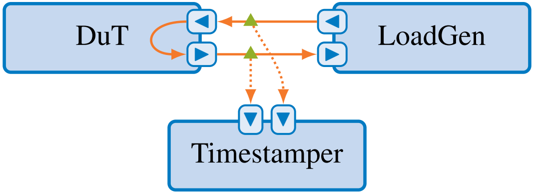
```
action forward(bit<48> dstMac) {
        ...
}

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 RAM
- Latency optimized *T4P4S* → batch size of one
- `add_on_miss` activated

LoadGen

- MoonGen [3] 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

Approach

P4 program

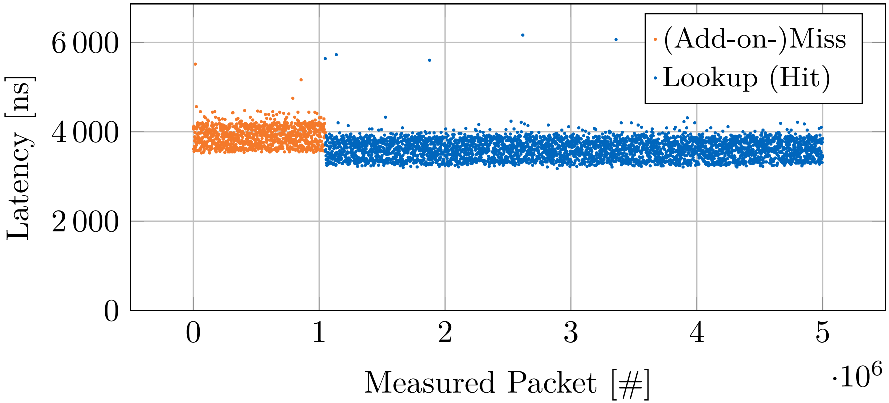
- Each packet contains key and value for a new table entry
- P4 programs contains lookup to this *one* table
- Forward all packets back

Two phases

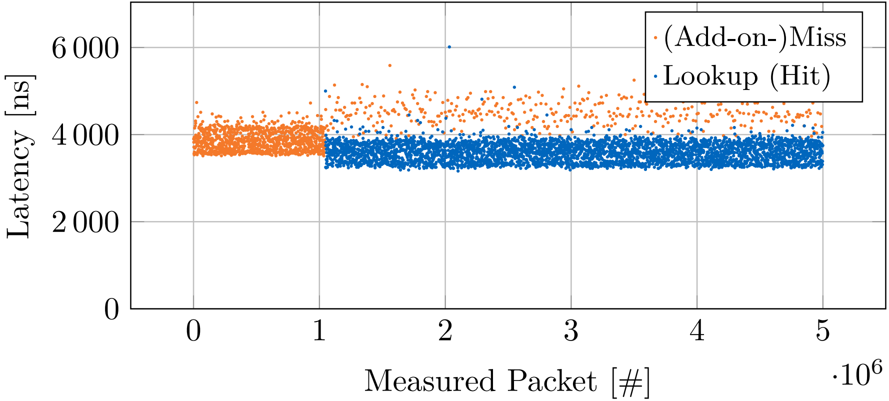
- Key cycle pseudo-randomly through $[0, 2^{20}]$ several times
- *First phase*: only insertions are performed
- *Second phase*: mainly lookups are performed; some insertions are done with different rates

Evaluation

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



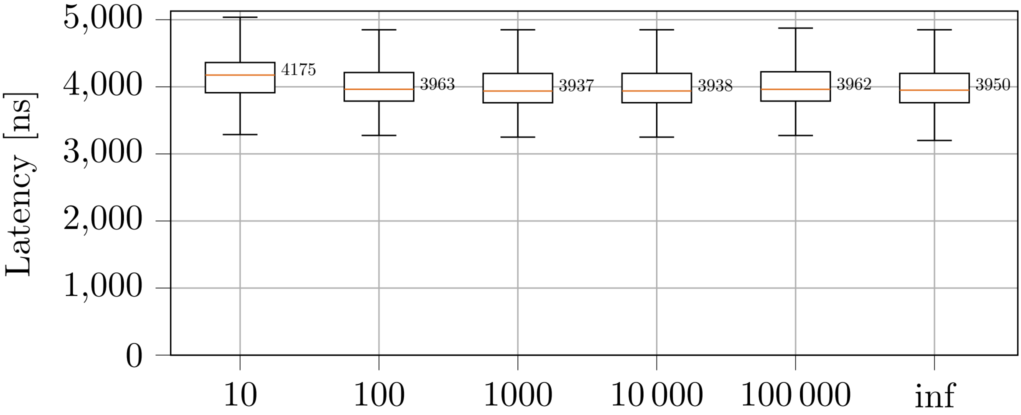
- *First phase*: 2^{20} packets triggering an insertion
- *Second phase*: $\approx 4M$ packets trigger lookup of previously inserted packets
  - But every 10 000-th packet triggers additional insertion



Evaluation

- Different rate of insertions during *second phase*

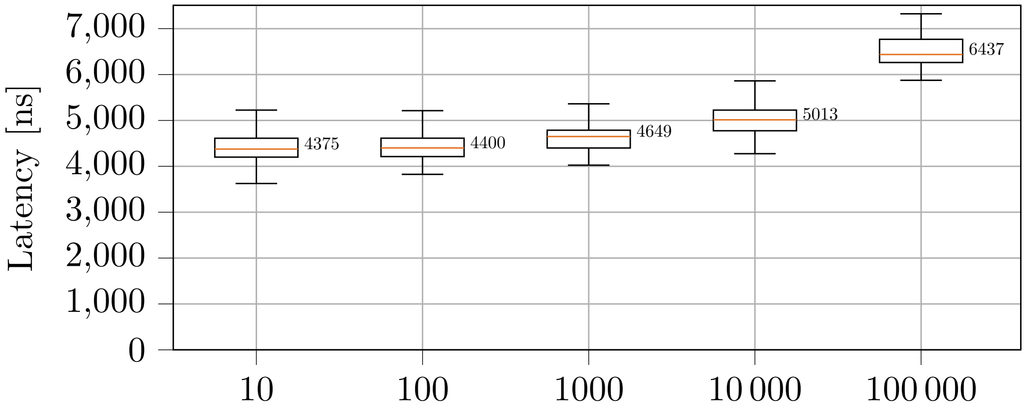
⇒ Median mixed (i.e. insertions & lookups) latency decreases with increasing rate



Evaluation

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

⇒ Worse branch prediction



Conclusion

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

Conclusion

- Adding state to the P4 data plane increases number of possible low-latency applications
  - Updatable Table Entries
  - Add-On-Miss Insertions
- Add-on-Miss insertions enable cheap insertions w.r.t. latency
- Is this a step backwards in SDN ?

Conclusion

- Adding state to the P4 data plane increases number of possible low-latency applications
  - Updatable Table Entries
  - Add-On-Miss Insertions
- 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

Bibliography

[1] P4 portable nic architecture (pna), version 0.5.
accessed: 2023-03-10.

[2] 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.

[3] 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.

[4] S. Pontarelli, R. Bifulco, M. Bonola, C. Cascone, M. Spaziani, V. Bruschi, D. Sanvito, G. Siracusano, A. Capone, M. Honda, et al.
Flowblaze: Stateful packet processing in hardware.
In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 531–548, 2019.

[5] 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.

[6] 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.

[7] L. Zeno, D. R. Ports, J. Nelson, D. Kim, S. Landau-Feibish, I. Keidar, A. Rinberg, A. Rashelbach, I. De-Paula, and M. Silberstein.
{SwiSh}: Distributed shared state abstractions for programmable switches.
In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 171–191, 2022.