Ralf Kundel, Christoph Gärtner, Sukanya Bhowmik, Boris Koldehofe. *Flexible Content-based Publish/Subscribe over Programmable Data Planes.*
To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS), IEEE, 978-1-7281-4973-820, 2020.

Flexible Content-based Publish/Subscribe over Programmable Data Planes

Ralf Kundel*, Christoph Gärtner*, Manisha Luthra*, Sukanya Bhowmik†, Boris Koldehofe*
*Multimedia Communications Lab, Technische Universität Darmstadt, Germany
{*ralf.kundel, christoph.gaertner, manisha.luthra, boris.koldehofe*}*@kom.tu-darmstadt.de*
†IPVS Stuttgart, Germany
sukanya.bhowmik@ipvs.uni-stuttgart.de

***Abstract*—Publish/subscribe systems have to react fast on changes in their environment while handling many events with low end-to-end latency and high throughput. Moving the broker functionality of publish/subscribe systems to the underlying network layer reduces the path length of events and, in addition, forwarding benefits from powerful and programmable hardware. So far attempts of underlay publish/subscribe depend on a specific API of the network devices, e. g., the OpenFlow protocol, which have restrictions in dealing with dynamic devices and corresponding changes in the introduced attribute names for matching and filtering events.**

In this work, we focus on the next generation of network devices, which are envisioned to provide reconfigurable hardware components, specified by the open P4 description language. We introduce two new approaches that enable a flexible and generic attribute/value encoding, understandable by P4-capable packet processors, to benefit from the performance properties of hardware. Furthermore, the proposed approaches reduce the effort in encoding and decoding event messages.

***Index Terms*—Publish/Subscribe, P4, Dataplane, Offloading**

I. Introduction

Publish/subscribe is a key paradigm to establish scalable and robust communication between producers and consumers of information, especially, in dynamic environments like the IoT or mobile networks where hosts frequently join and leave the network. Therefore, it is of tremendous importance to enable a decoupled communication between producers and consumers, i. e., producers do not need to know their consumers and vice versa. Decoupled communication is traditionally accomplished by a broker-based overlay network which is in charge to forward events to interested subscribers [11]. Especially, content-based publish/subscribe allows to specify fine-grained subscriptions, e. g., by constraints on values of specific event attributes. While publish/subscribe systems in general focus on bandwidth efficiency, a fast adaptation of changing environments and low latency is required as well in more and more usecases, e. g., IoT. Therefore, existing work has focused on improving the performance of publish/subscribe systems by making the overlay topology more agnostic to the network. More recently proposed publish/subscribe systems move the broker functionality into the network benefiting from the hardware acceleration capabilities of the networked devices, e. g., by building on hardware acceleration with FPGA/GPUs [10], using SDN capabilities [2] and some even base on the data plane programming language P4 [14]. A key limitation of these approaches is that they build on a static encoding of attributes in the header fields. Dynamic IoT environments impose significant costs in reconfiguring systems in these environments, e. g., by updating data plane descriptions or encoding entire routing trees in events [14]. In this work, we intend to enable flexible attribute/value encoding while dealing with the challenges of (i) dynamic environments and (ii) addressing low latency applications.

By doing so, we build on a new generation of programmable network devices whose behavior can be reconfigured by the P4 description language. The main contributions of this paper are: (1) two approaches of flexible attribute/value encoding in packet headers which can be interpreted by programmable data planes and (2) a proof-of-concept P4 data plane implementation for these attribute encodings.

In the following, we first discuss background information and related work. In Section III, we introduce our system model and approach of encoding attribute/value pairs in parsable packet headers. Finally, in Section IV, we briefly evaluate our approach and conclude with Section V.

II. Background

This work focuses on facilitating publish/subscribe in the data plane using the open programming language P4 [3]. This language enables easy data plane programming in a C-like syntax which can be compiled to a hardware specific configuration file, e. g., network switches, SmartNICs and FPGAs. The language allows the definition of any packet header format which enables the introduction of new network protocols with ease. Furthermore, application specific logic with custom control flows and application specific match tables, exceeding the functionality of common switches, can be realized within the P4 program and capable hardware.

A. Related Work

Programmability on the networking devices and hardware acceleration has been investigated in existing work. For instance, Margara *et al.* [10] exemplified the potential of hardware acceleration for publish/subscribe with a multi-threaded and GPU-based broker implementation. Pleroma [2] is a content-based publish/subscribe approach using OpenFlow, a software-defined switch configuration protocol. The approach uses an encoding technique based on spatial indexing to map

Ralf Kundel, Christoph Gärtner, Sukanya Bhowmik, Boris Koldehofe. *Flexible Content-based Publish/Subscribe over Programmable Data Planes.*
To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS), IEEE, 978-1-7281-4973-820, 2020.

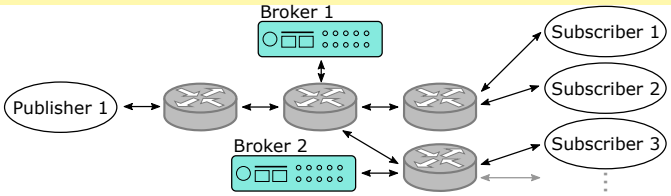

Fig. 1: Broker based overlay publish/subscribe system in a typical network environment.

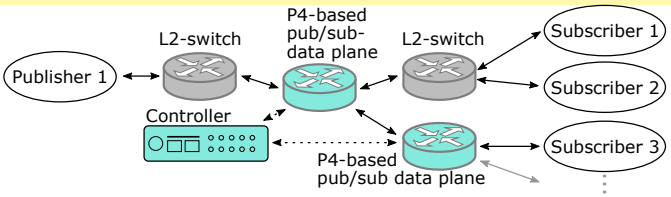

Fig. 2: Underlay publish/subscribe system with P4-capable switches instead of brokers.

advertisements, subscriptions and events to header fields supported by OpenFlow, e. g., the IPv6 multicast address range. The approach supports at line-rate performance content-based publish/subscribe, but (1) limits the number of attribute value pairs because of the fixed size header fields, (2) approximates only encoded values and (3) creates additional overhead for the publishers to encode the content attributes [1].

P4 is an emerging data plane programming language for networking hardware which gives flexibility to define own protocol independent header fields in favor of OpenFlow. Wernecke *et al.* [14] realize —like our work— content-based publish/subscribe with P4 by sacrificing a key property to scalable content-based publish/subscribe: the decoupling of publishers and subscribers. The header of each packet is required to carry the entire dissemination tree. Jepsen [5] *et al.* investigated applicability of P4 for publish/subscribe at $3.2Tbit/s$ at Barefoot Tofino ASIC with a fixed attribute/value encoding. With a special purpose compiler they created a custom P4 data plane for the given use-case and all packets have to match this fixed header format. P4 has also been investigated to accelerate other middleware paradigms such as complex event processing [7]. Similar concerns were raised as part of INetCEP [9] for information-centric networking.

Another promising alternative is to realize publish/subscribe over fully programmable hardware [6]. For example Tsoi *et al.* [12] discussed the usability of FPGAs, a very flexible technology with good performance but also high complexity.

To summarize, related work has shown up a huge potential to accelerate the performance publish/subscribe systems. Nevertheless, these approaches are very limited in flexibility in terms of dealing with dynamics in environment while providing low latency.

III. Content-based Subscription matching in Data Planes

In this section we describe flexible attribute/value pair encoding in packets and how to match them in a general way. The approach can be separated in three parts: (1) attribute/value encoding at the publisher, (2) parsing the attributes in the data plane and (3) matching the previously parsed values.

A. System Model

A publish/subscribe system is comprised of three main system entities: (1) *publishers*, (2) *subscribers*, and (3) *brokers* (cf. Figure 1). Publishers announce events they indicate to publish in form of advertisements, while subscribers announce their interest in form of subscriptions. The brokers of the publish/subscribe system forward published events to subscribers with a matching subscription.

We will use the content-based subscription model, i. e., events are represented by $< name, value >$ pairs, e. g., event $e = [< temp, 25 >, < hum, 30 >]$. Subscriptions and advertisements are represented as logical expressions over predicates on the attribute values. For example, a subscriber may require to receive all events matching $[temp < 30 \& hum > 15]$.

In contrast to traditional overlay-based publish/subscribe systems (cf. Figure 1), the broker functionality of this work is realized by P4 programmable switches (cf. Figure 2). Furthermore, the P4-brokers can co-exist with traditional L2-network elements. The figures illustrate the potential in reducing hops and benefiting from hardware-accelerated matching operations inside the P4-brokers. For instance, an event from *Publisher 1* to *Subscriber 2 & 3* has a reduced number of hops as the connections to and from *Broker 1 & 2* are not existent any more. In addition, we expect a lower latency and higher event rates as hardware typically performs much faster than software.

B. Flexible Attribute/Value Encoding in Packet Headers

One major point of P4-based publish/subscribe systems is the encoding of attributes and values, which we discuss in this section. The first and only assumption in our and any other Ethernet-based system is the existence of a valid Ethernet header in the beginning of a packet. By that, the L2 addresses can be extracted and the value of the *EtherType* field is known. Based on the *EtherType* value, the following header type, e. g., IPv4 or IPv6, can be identified and subsequently its fields can be extracted. Our approach works as part of the network layer and does not rely on IP protocols. To identify our introduced protocol by all system components, we use the *EtherType=0x9001*, which is currently unused.

As the goal of this work is a flexible attribute/value encoding, two header fields are required per attribute: (1) kind of attribute and (2) corresponding attribute value. Due to matching constraints of hardware architectures, our approach is limited to integer and fixed-point based values on most architectures. In the following we present two approaches of attribute encoding:

- **Attribute/Value pairs:** For each attribute a tuple of kind of attribute and value is encoded.
- **Bitmask:** One reserved bit per possible attribute indicates whether this is part of the event.

Ralf Kundel, Christoph Gärtner, Sukanya Bhowmik, Boris Koldehofe. *Flexible Content-based Publish/Subscribe over Programmable Data Planes.*
To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS), IEEE, 978-1-7281-4973-820, 2020.

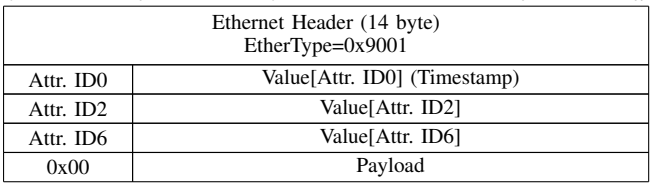

Fig. 3: ID-based attribute/value pair encoding.

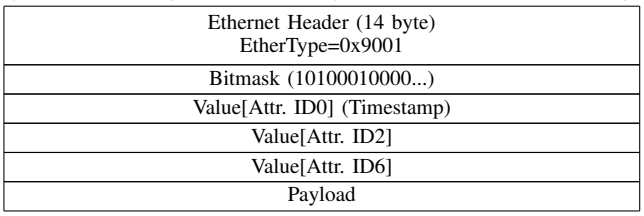

Fig. 4: Bitmask based attribute/value encoding.

Each attribute has a special meaning which is encoded by an unique attribute ID, e. g., the *air pressure* of an IoT sensor could be encoded with the $ID = 2$ as illustrated in Table I.

Attribute/Value Pairs: The first approach, encoding attribute/value pairs as tuples, is depicted in Figure 3. The *EtherType=0x9001* indicates that a publish/subscribe header follows the Ethernet header. The first bits of this publish/subscribe header indicate the kind of the first attribute by its ID. Hereafter, the corresponding value of this attribute is encoded. In the depicted example, the ID is *0*, which means, referring to Table I, that the first value describes the timestamp of the event. Afterwards, the next following bits indicate the next attribute, enabling a recursive attribute/value pair extraction. The final ID *0x00* indicates the end of the attribute stack, similar to stacked MPLS-labels. As every event contains at least one attribute and the ID *0* has to be encoded first or never, this can be used again to indicate the end of stack.

The parser-code, given in Listing 1, describes the recursive attribute/value pair extraction. It is important to note that initially the state `parse_first_id` (Line 12–Line 16) extracts only the first attribute ID. After that, in state `parse_event` (Line 17–Line 23), the attribute value of the previously parsed attribute ID will be extracted. Furthermore, the next attribute ID will be parsed always as well. Until the next ID is not zero ($0x00$ indicates end of attribute list), this extraction will be performed recursively.

The attributes are stored in a P4 header stack, similar to a C array data structure, and the value of attribute *i* is stored in the stack entry at position *i* independent of its packet's header position.

Bitmask: The second approach of representing the attributes and values is depicted in Figure 4. As before, the type field of the Ethernet header indicates that the following header contains a publish/subscribe specific header. As first part of this header, a bitmask represents which kind of attribute values are part of this event and consequently encoded in the following header fields. In the given example bitmask (`10100010000`), from left to right, the bits of Attribute 0, 2, and, 6 are set. Thus there will be three attribute values. After the bitmask, all attribute values are listed, sorted by their corresponding attribute ID, without any further additional information. In contrast to the first approach, the ordering of the values must be ascending to their attribute IDs as there is no explicit attribute ID before each value. Parsing this header structure is similar to the first approach: initially, the bitmask will be extracted. Afterwards, the values are extracted recursively based on a slicing bitmask which fade out the already extracted attributes and terminates with the last *high bit*.

Attribute	ID
timestamp	0
temperature	1
air pressure	2
...	...

TABLE I: Exemplary attribute encoding.

C. Subscription matching in programmable pipelines

The matching algorithm can be performed within one single generic P4-table as depicted in Listing 2. All attribute values, stored in the *event* header stack fields, and a bitmask of valid attributes form the input of this table. A ternary match operator on the bitmask allows a selection of interest attributes (Line 3). Longest prefix matches (lpm) enable a content-based range matching by creating multiple lpm flow rules (e.g., in Line 4). For example, the predicate $p2 = [16 \leq temp \leq 47]$ must be divided in two lpm-matchable flow rules ($[16 \leq temp \leq 31]$ and $[32 \leq temp \leq 47]$) because a single lpm match can not cover exactly this range. In $P4_{14}$, `range` matching can be used instead of `lpm` matching. Multiple overlapping flow rules cause no problem since each flow rule has a priority and by that only one entry of the table will match, ensured by the controller logic. It is important to note that our current approach matches on a P4 header stack which has valid and

```
1  typedef bit<8> nextId_t; nextId_t last;
2  header event_t {
3      bit<32> value;
4      nextId_t nextId;
5  }
6  state start {
7    packet.extract(hdr.ethernet);
8    transition select(hdr.ethernet.etherType) {
9      0x9001: parse_first_id;
10     default: accept;
11 }}
12 state parse_first_id {
13   packet.extract(hdr.event_prefix);
14   last = hdr.event_prefix.nextId;
15   transition parse_event;
16 }
17 state parse_event {
18   packet.extract(hdr.event[last]);
19   last = hdr.event[last].nextId;
20   transition select(last) {
21     0: accept;
22     default: parse_event;
23 }}
```

Listing 1: P4-pseudocode of extracting attribute/value pairs from packet header fields recursively.

Ralf Kundel, Christoph Gärtner, Sukanya Bhowmik, Boris Koldehofe. *Flexible Content-based Publish/Subscribe over Programmable Data Planes.*
To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS), IEEE, 978-1-7281-4973-820, 2020.

```
1  table event {
2    reads {
3      metadata.bitmask     : ternary;
4      hdr.event[0].value   : lpm;
5      ...
6      hdr.event[31].value  : lpm;
7    }
8    actions {
9      sendToGroup;
10 }}
```

Listing 2: Generic attribute/value matching in P4 data planes independent on header structure.

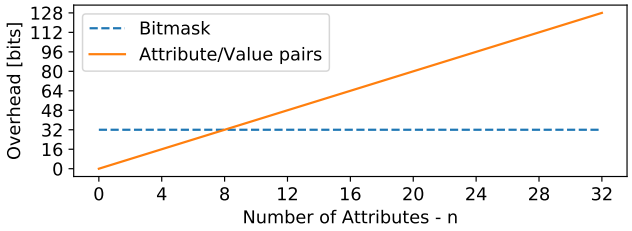

Fig. 5: Required bits for encoding n used attribute types for maximum allowed attributes $n_{max} = 32$ and $k = 4$ attribute ID bits (ID-based approach).

invalid entries if only a subset of the attributes are contained in the event. These fields are ignored by the match rule as the lpm match is "/0" which is equivalent to a wildcard(*) match. Forwarding is realized by assigning the packet to a multicast group in the action `sendToGroup` which contains all destinations for this event (Line 9).

D. Controller Centric Subscription Handling

The controller of the publish/subscribe system, handling advertisements and subscriptions, has to compute flow rules for every P4 switch in the system. This includes setting up all needed multicast groups and aggregation of subscriptions. As this work focus on the data plane feasibility, we will not discuss the control logic in detail.

E. Limitations of P4 Implementations

During our tests we observed some minor hurdles, e. g., the sample source code in Listing 1 addresses the header stack `event` with the previously extracted ID. However, regarding the P4 specification, this feature is optional and might not be supported. Tests with the *behavioral model 2 (bmv2)*, which is the reference implementation for P4, have shown that this is not supported. We tackled this issue by creating one parser state per attribute ID with a constant header stack address and moved the ID specific logic to the state transition logic.

Besides that, header fields like the attribute ID in Figure 3 might be restricted to a size which is byte aligned (8, 16, 24 bit).

As the attributes are extracted to a generic header stack, not all stack entries are valid. Applying this vector of headers to a table might cause, depending on the target, an undefined behavior. A workaround is using metadata stacks instead of header stacks which behave similar but are always valid.

Furthermore, physical scaling limitations of P4 architectures should be considered. The language does not limit the size of parsed headers, header stack data structures, and tables but hardware architectures have limited resources.

IV. Results & Discussion

We validated this approach by an implementation for the P4 reference switch *behavior model 2 (bmv2)*, embedded in a mininet topology consisting of 3 switches. The publishers and subscribers are realized with the python framework *Scapy*, allowing the easy creation and parsing of custom packet headers. This work focuses on the most common language P4, but we assume a feasibility in other upcoming languages, each driven by a single vendor, as NPL (Broadcom), MicroC (Netronome), and SDNet (Xilinx), as well. The source code is openly available for reproducing and continuing our work[1].

In this work, we proposed two different approaches of how to encode attributes in packet headers. The first approach (Figure 3) requires k bits, e. g., 8 bits, for each of the n_i attributes of event i to encode its type. The second approach requires n_{max} bits for the bitmask in every event independent of the number of attributes per event. Both, k and n_{max}, are parameters which can be configured. Figure 5 illustrates that above a certain average threshold (here: n=8) the bitmask approach performs better and vice versa.

We assume, based on existing work [8] [4], a forwarding delay of $\leq 1\mu s$ for P4-ASICs and $2\mu s - 15\mu s$ for P4-FPGAs/NPUs compared to $\sim 250ms$ for software based approaches [13] can be achieved. Event rates of upto 100 million events/s for NPUs/FPGAs and 1 billion for P4-ASICs are realistic.

Note: The flexibility of current programmable data planes allows only numeric matches, e. g., integer and fixed-point based. If at all, string and floating-point matching requires the use of P4 external functions and supported hardware, currently only FPGAs. Parsing string and floating-point based attributes and values is possible in P4 pipelines.

V. Conclusion & Outlook

With this work we have illustrated and analyzed the power of P4-programmable data planes in the context of content-based publish/subscribe. Compared to existing approaches, based on hard-coded header structures, the proposed approaches for attribute encoding provide high flexibility regarding diverse attribute/value sets of the advertised messages. Our analysis highlights the encoding overhead of the two approaches and point out the break-even point. Furthermore, adding newly used attributes does not require an update of the current system state, consisting of attribute ID mappings and flow entries. The feasibility of this approach was demonstrated by a prototypical implementation based on the P4 software switch *bmv2*. Next steps are (1) an in depth evaluation in a real P4-based network mesh and (2) improvement towards resource efficient match-tables and rules.

[1] https://github.com/ralfkundel/p4bsub/

Ralf Kundel, Christoph Gärtner, Sukanya Bhowmik, Boris Koldehofe. *Flexible Content-based Publish/Subscribe over Programmable Data Planes.*
To appear in the Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS), IEEE, 978-1-7281-4973-820, 2020.

ACKNOWLEDGMENT

This work has been supported by the German Research Foundation (DFG) as part of the project C2 within the Collaborative Research Center (CRC) 1053 MAKI. We thank our colleagues and reviewers for their valuable input and feedback.

REFERENCES

[1] S. Bhowmik, M. A. Tariq, J. Grunert, D. Srinivasan, and K. Rothermel, "Expressive content-based routing in software-defined networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2460–2477, Nov 2018.

[2] S. Bhowmik, M. A. Tariq, B. Koldehofe, F. Dürr, T. Kohler, and K. Rothermel, "High performance publish/subscribe middleware in software-defined networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1501–1516, Jun. 2017.

[3] 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," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[4] H. Harkous, M. Jarschel, M. He, R. Pries, and W. Kellerer, "Towards Understanding the Performance of P4 Programmable Hardware," in *Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2019.

[5] T. Jepsen, M. Moshref, A. Carzaniga, N. Foster, and R. Soulé, "Packet subscriptions for programmable asics," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, ser. HotNets '18. New York, NY, USA: ACM, 2018, pp. 176–183.

[6] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "Lipsin: Line speed publish/subscribe inter-networking," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, p. 195–206, Aug. 2009. [Online]. Available: https://doi.org/10.1145/1594977.1592592

[7] T. Kohler, R. Mayer, F. Dürr, M. Maaundefined, S. Bhowmik, and K. Rothermel, "P4cep: Towards in-network complex event processing," in *Proceedings of the 2018 Morning Workshop on In-Network Computing*, ser. NetCompute '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 33–38. [Online]. Available: https://doi.org/10.1145/3229591.3229593

[8] R. Kundel, L. Nobach, J. Blendin, H.-J. Kolbe, G. Schyguda, V. Gurevich, B. Koldehofe, and R. Steinmetz, "P4-bng: Central office network functions on programmable packet pipelines," in *15th International Conference on Network Service Management*. IEEE, Oct 2019.

[9] M. Luthra, B. Koldehofe, J. Höchst, P. Lampe, A. H. Rizvi, R. Kundel, and B. Freisleben, "Inetcep: In-network complex event processing for information-centric networking," in *Proceedings of 15th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, Sep 2019.

[10] A. Margara and G. Cugola, "High-performance publish-subscribe matching using parallel hardware," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 126–135, Jan 2014.

[11] M. A. Tariq, B. Koldehofe, G. G. Koch, I. Khan, and K. Rothermel, "Meeting subscriber-defined qos constraints in publish/subscribe systems," *Concurr. Comput. : Pract. Exper.*, vol. 23, no. 17, pp. 2140–2153, Dec. 2011. [Online]. Available: http://dx.doi.org/10.1002/cpe.1751

[12] K. H. Tsoi, I. Papagiannis, M. Migliavacca, W. Luk, and P. Pietzuch, "Accelerating publish/subscribe matching on reconfigurable supercomputing platforms," in *Many-Core and Reconfigurable Supercomputing Conference (MRSC), Rome, Italy*, vol. 3, 2010, p. 2010.

[13] S. Venkataraman, A. Panda, K. Ousterhout, M. Armbrust, A. Ghodsi, M. J. Franklin, B. Recht, and I. Stoica, "Drizzle: Fast and adaptable stream processing at scale," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: ACM, 2017, pp. 374–389.

[14] C. Wernecke, H. Parzyjegla, G. Mühl, P. Danielis, and D. Timmermann, "Realizing content-based publish/subscribe with p4," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2018, pp. 1–7.