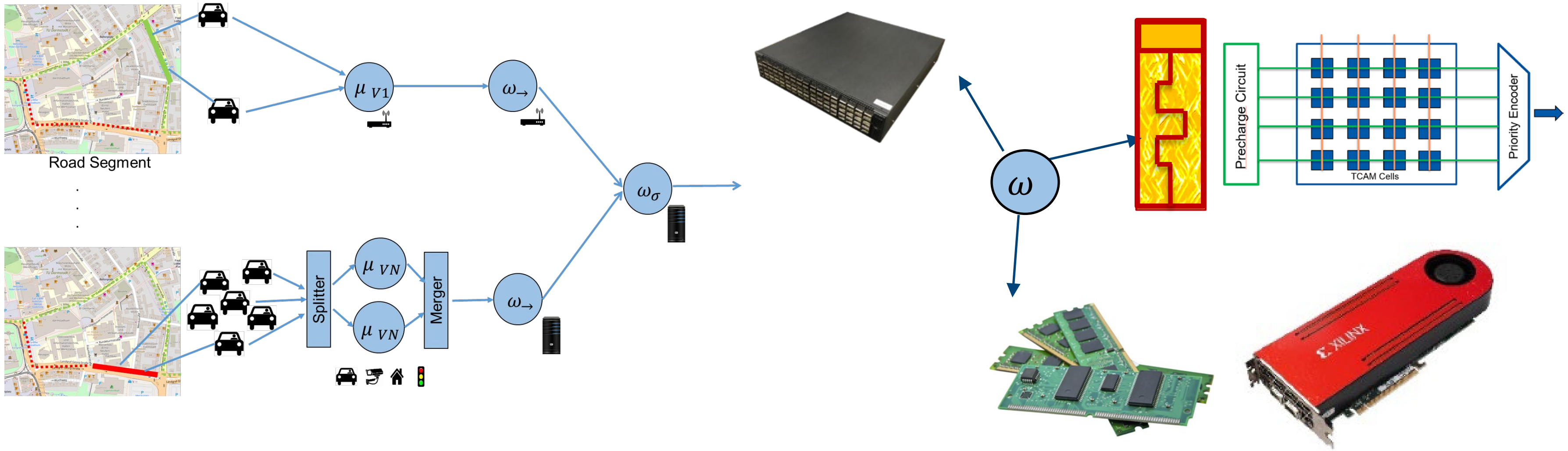
Accelerating the performance of distributed stream processing systems with in-network computing

Boris Koldehofe

@DEBS 2013



30-Jun-23

Short Introduction
Boris Koldehofe

Distributed and Operating Systems Group

Technical University of Ilmenau

Research

- Distributed data analytics
- Computer system principles
- Reliability and security

Specific Focus

- Distributed Event-based systems (DEBS)
- In-Network Computing

Data Driven Applications

Nowadays everywhere!

- Autonomous driving, smart factories, smart cities, telemedicine, and many more

MAPE loop of IoT services:

- **M**onitor and **A**nalyze “Things”
- **P**lan and **E**xecute Processes

Insights into data key to adapt applications

- Billions of things
- Exabytes of context knowledge




But Performance and Low Latency is not straight forward!

Outline

Why low latency response?

The Bottleneck in Data Movements

In-Network Computing Technologies accelerating performance

Examples in the context of Distributed Event-Based Systems

Conclusion

Low Latency responses

Often relates to highly accurate time stamps of events

Manufacturing process

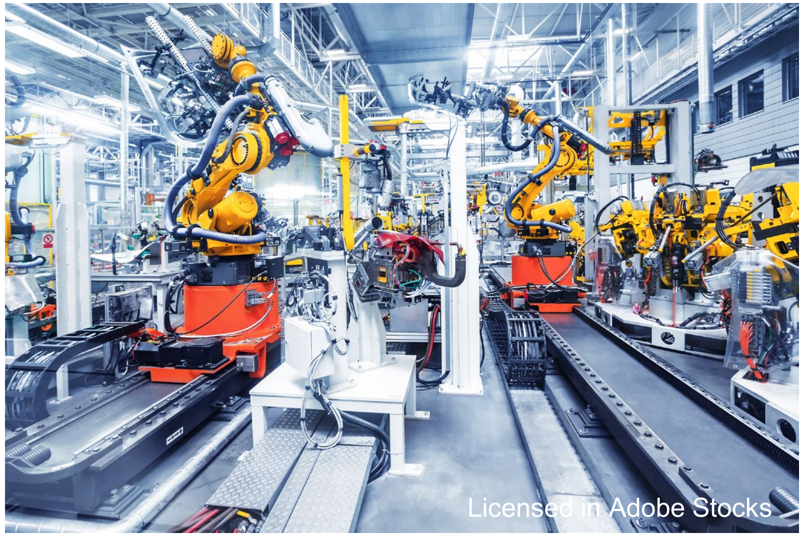
- Understand correct position over time
- Low Jitter in Communications

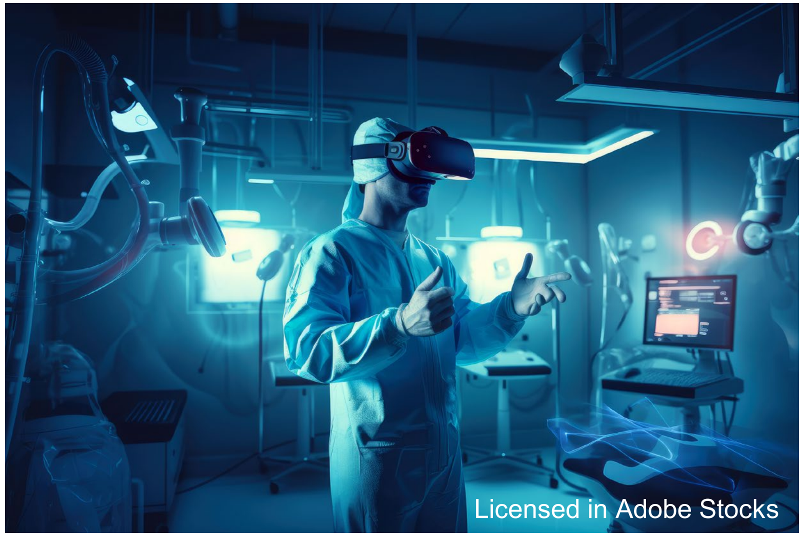
Telemedicine

- Understand situations with very low reaction time

Financial applications

- Algorithmic trading
- Very low responses in detecting and analyzing packets
  - See DEBS 2020 Grand Challenge





Improving Timestamp Accuracy
Technological developments

5G and even 6G Campus networks

- Goal interconnect processors fast
- 100µs - 1ms delays, high mobility

TSN

- Real-time guarantees for industrial applications

Edge Computing

- Offload Computations

Accelerators

- Computation
- I/O
- Protocols / Architectures

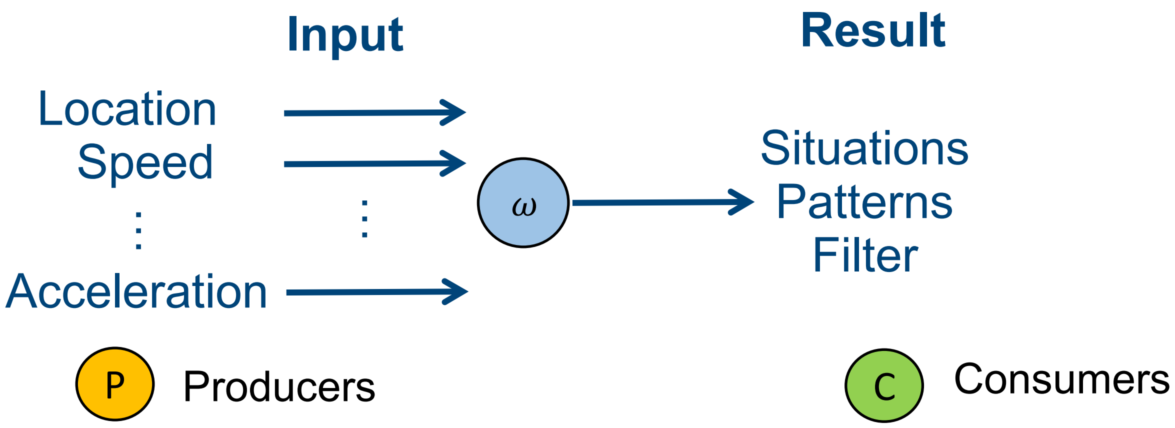
Timestamp inaccuracy	Location Inaccruracy
1s	10m
1ms	10cm
1µs	0.1mm

Moving Object of 36km/h

DEBS / Real-Time Analytics

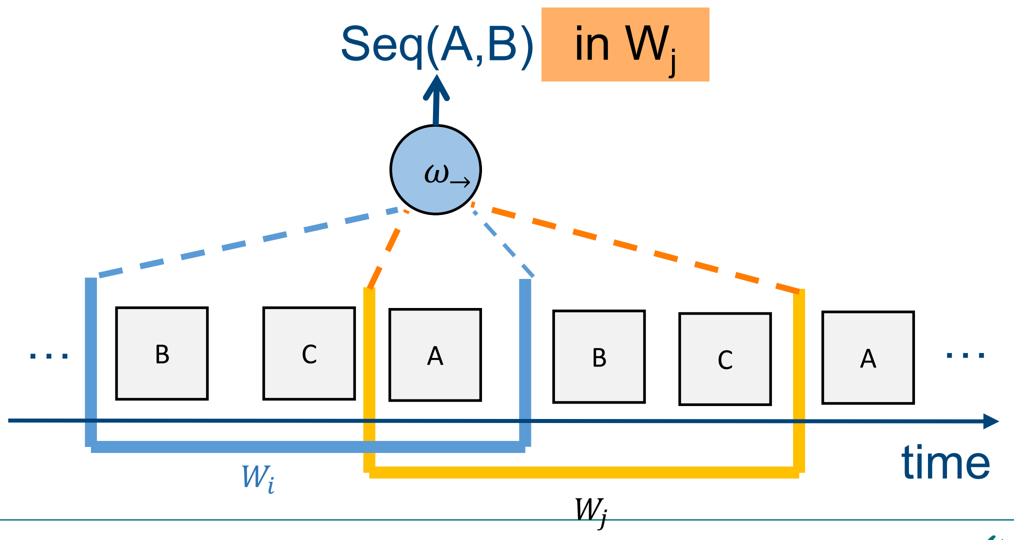
Correlations on data stream

- With low end to end delay
- High accuracy detection



Paradigm:

- *Operators* identify pattern on partial data stream: *window*
- E.g. *CEP operator*, Filter, Neural Network, Deep Learning Model



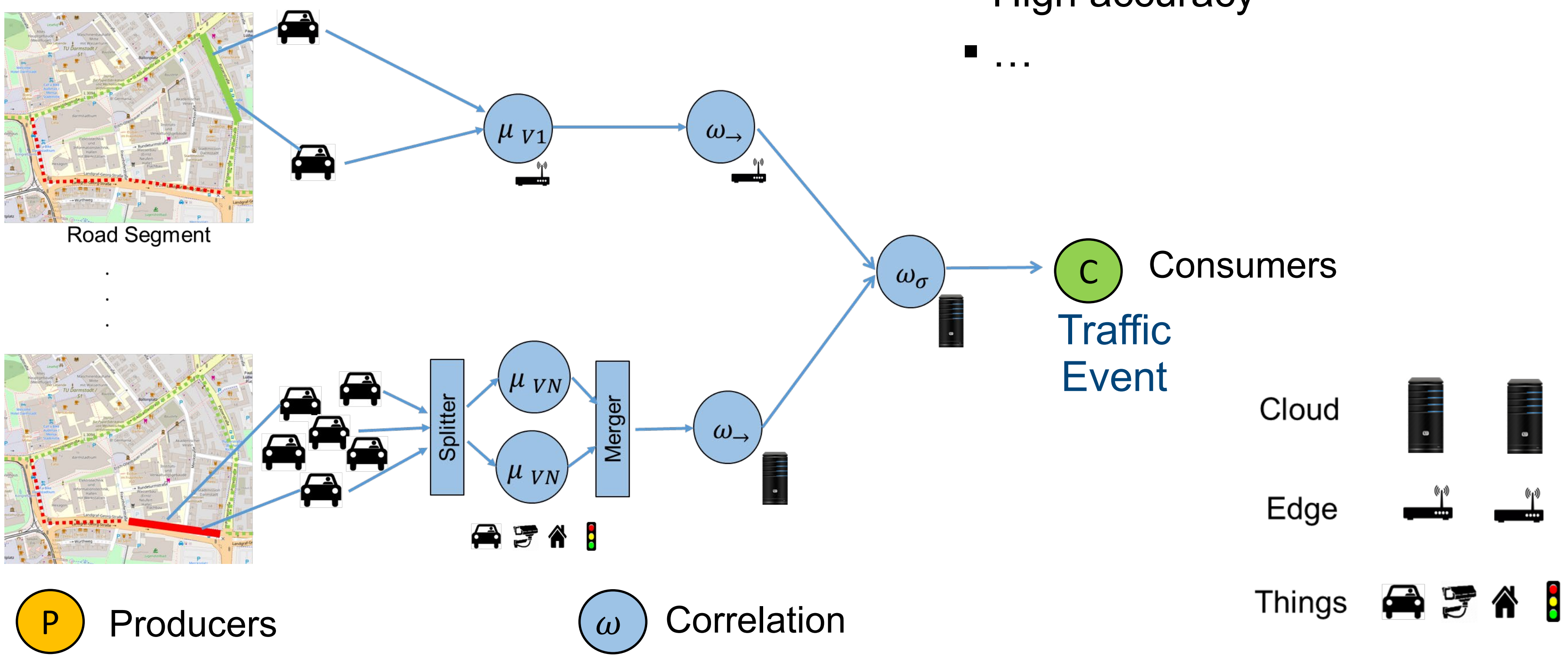
Distributed Real-Time Analytics

Execute operator network on a distributed infrastructure

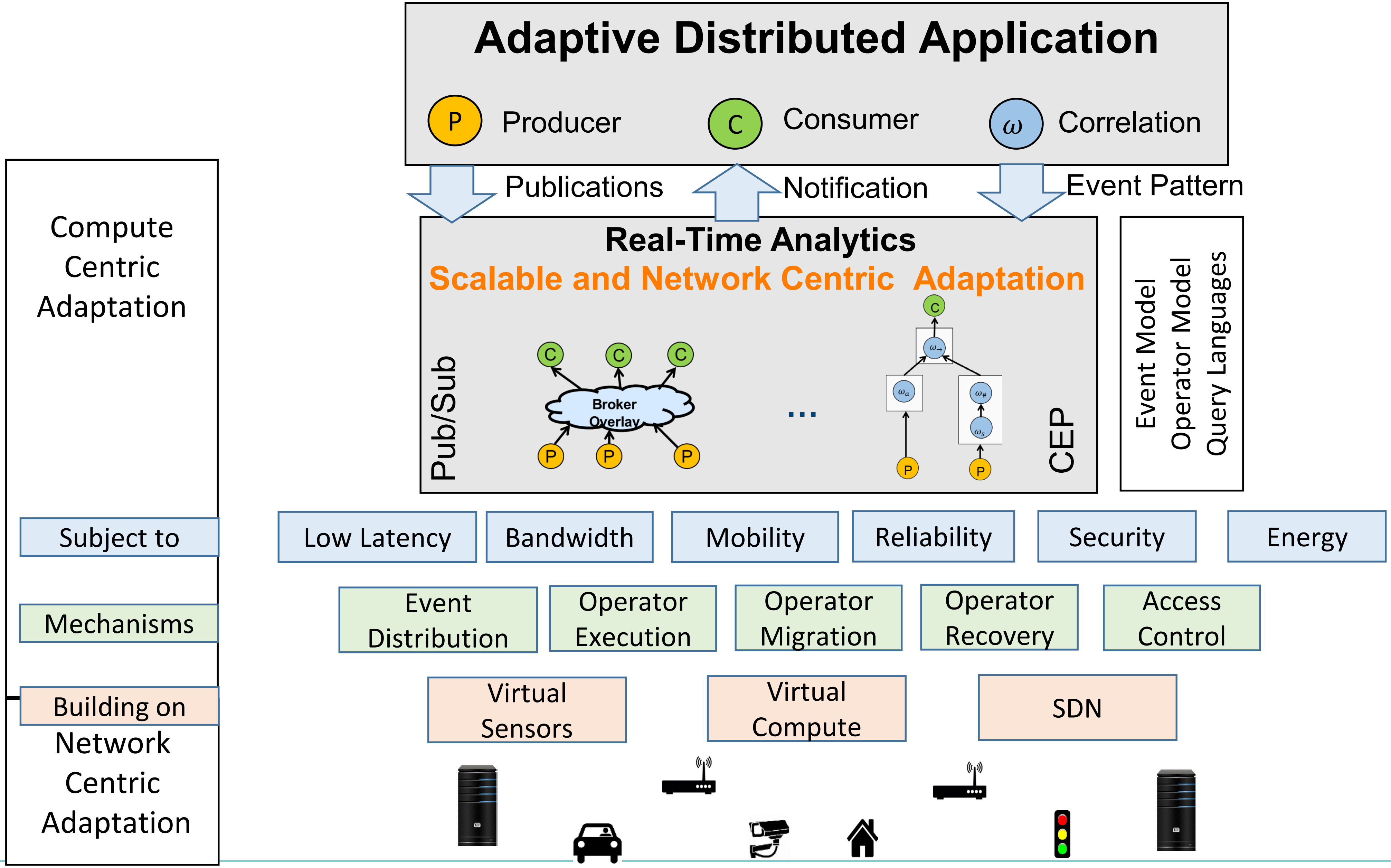
Increase Scalability and Performance

Optimization subject to potentially conflicting goals

- Decoupling producers and consumers
- Low end-to-end delay
- High accuracy
- …



The SPIRIT of science TECHNISCHE UNIVERSITÄT ILMENAU

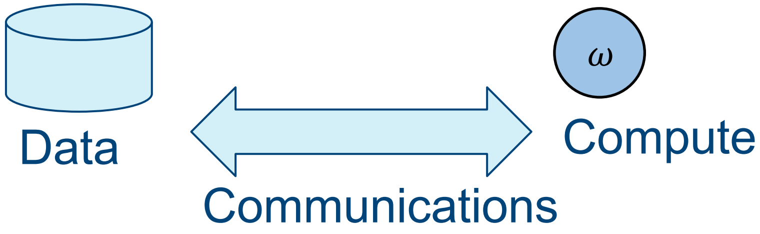

Adaptive Distributed Application
P
Producer
C
Consumer
ω
Correlation
Publications
Notification
Event Pattern
Real-Time Analytics
Scalable and Network Centric Adaptation
Pub/Sub
C
C
C
Broker Overlay
P
P
P
...
C
P
P
CEP
Event Model
Operator Model
Query Languages
Compute Centric Adaptation
Subject to
Low Latency
Bandwidth
Mobility
Reliability
Security
Energy
Mechanisms
Event Distribution
Operator Execution
Operator Migration
Operator Recovery
Access Control
Building on
Virtual Sensors
Virtual Compute
SDN
Network Centric Adaptation

Meeting Performance of Time Sensitive Distributed Applications

Cyberphysical application

- Low latency?
- Predictable performance?

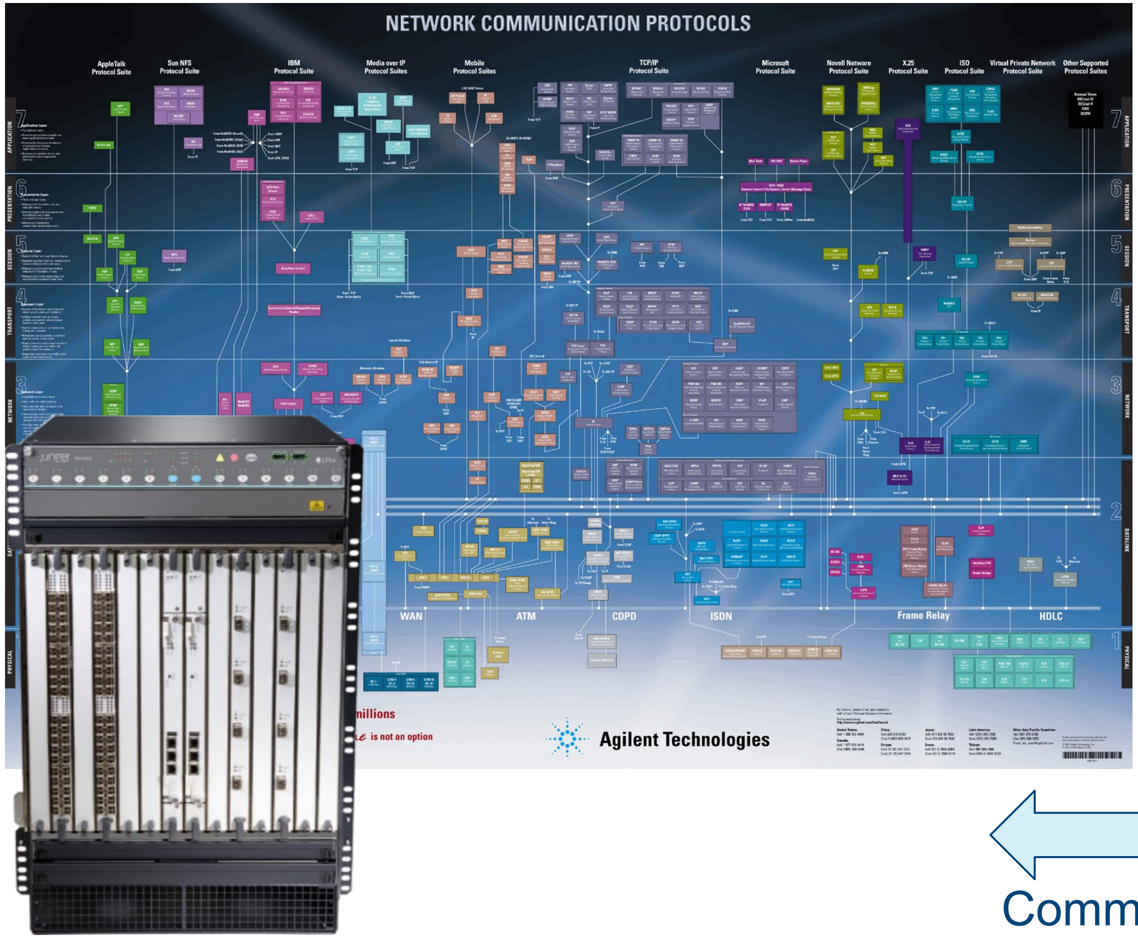
Bottlenecks in data movement and processing



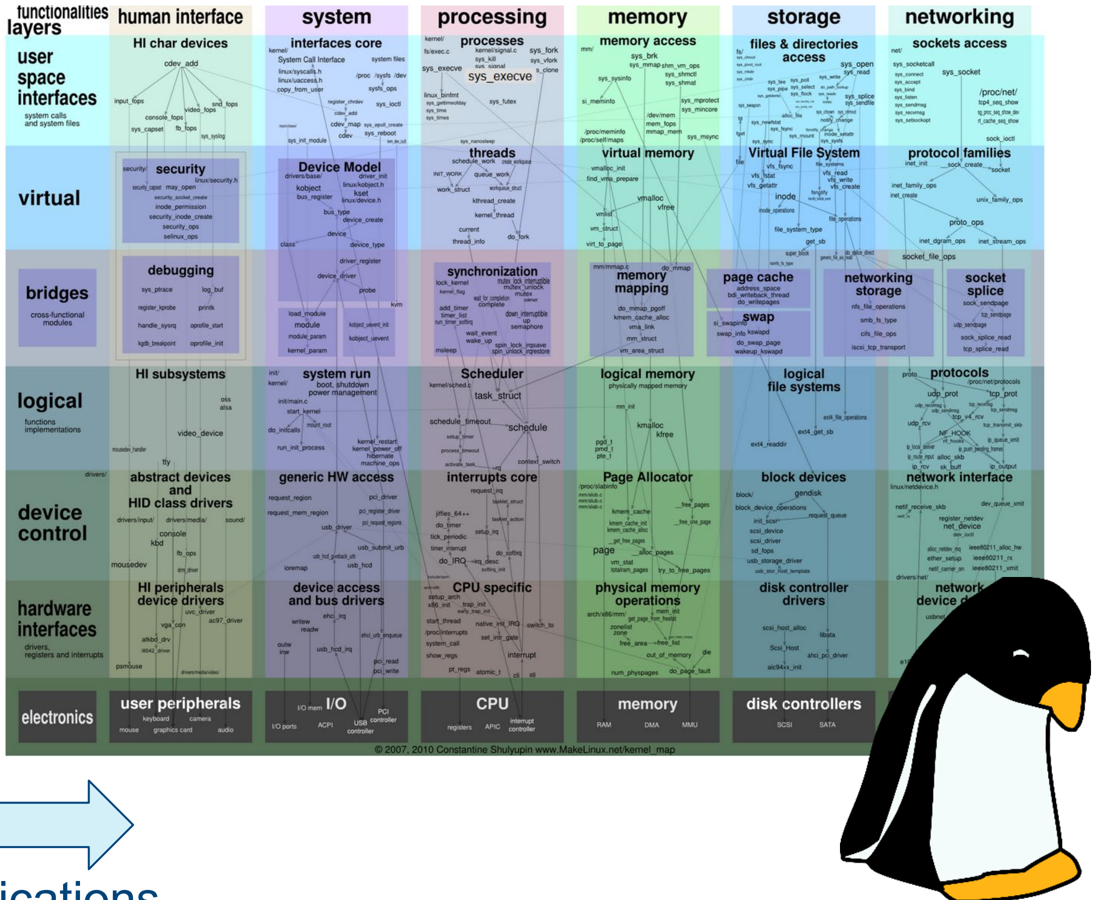
Requires much more flexibility in using mechanisms of the distributed infrastructure!

Lack for Flexibility: Communication Protocols and Operating Systems

Hardcoded in network appliances



Time for data to bypass the kernel



Communications

The SPIRIT of science
TECHNISCHE UNIVERSITÄT ILMENAU

Ingredients for Increased Flexibility

Programable hardware

- P4 Switches
- NetFPGA

New networking paradigms

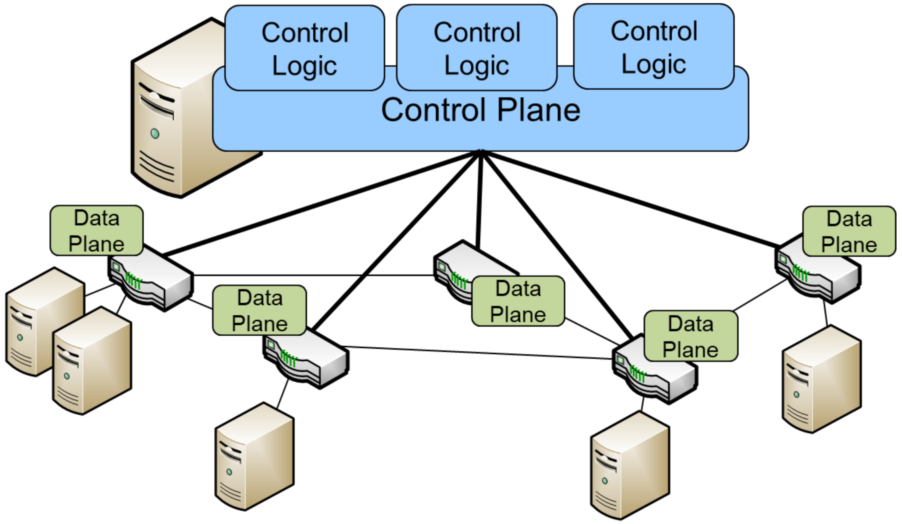
- Software-Defined Networking
- Network Function Virtualization

Significant changes in the infrastructure

- Edge Data Center
- Technologies & Concepts
  - DPDK, P4, OpenFlow, RDMA

Barefoot Tofino

FPGA



Enabler for in-network computing!

In-Network Computing

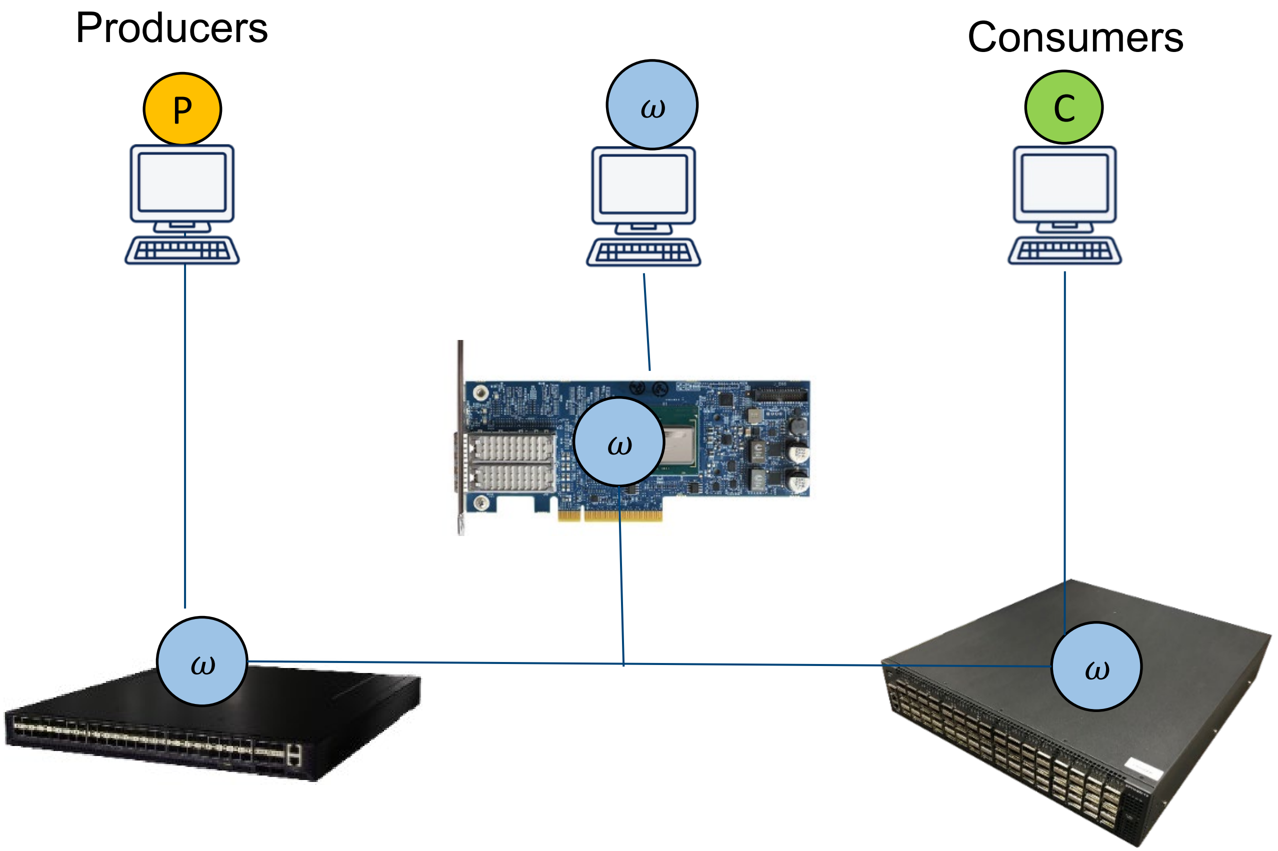
Idea enable computations on the data path

Traditionally,

- Packet header processing,
  - e.g., routing, firewall, packet classification, load balancing, deep packet inspection

Often

- Match/action pipeline model of networking hardware
- Management interface, specific programming interfaces, …



Evolution: In-Network Computing resources

Towards flexible, high performance, and energy-efficient in-network computing

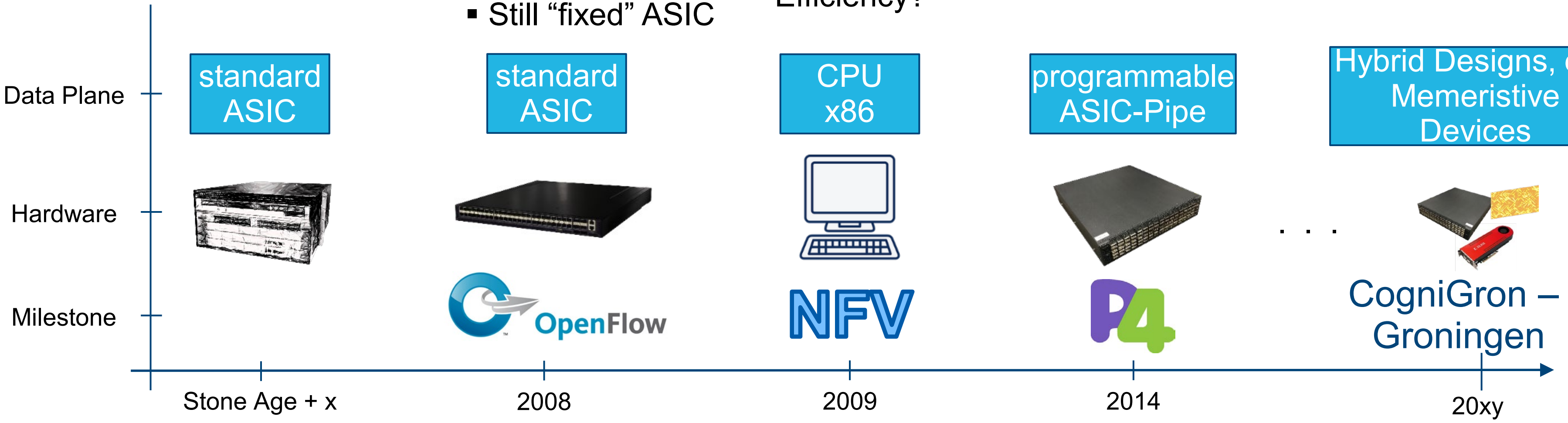
- "Blackbox"
- Vendor-lock

- "Blackbox" → "Whitebox"
- Software-defined Networking
- Controller Interface
- Still "fixed" ASIC

- "Why hardware?"
- Just do it in C!
- Standard Server
- DPDK
- Performance?
- Efficiency?

- Programmable ASIC for packet processing
- Very high bandwidth
- Limited flexibility

- Energy Efficient Switching?
- Computational Intelligence inside the network?

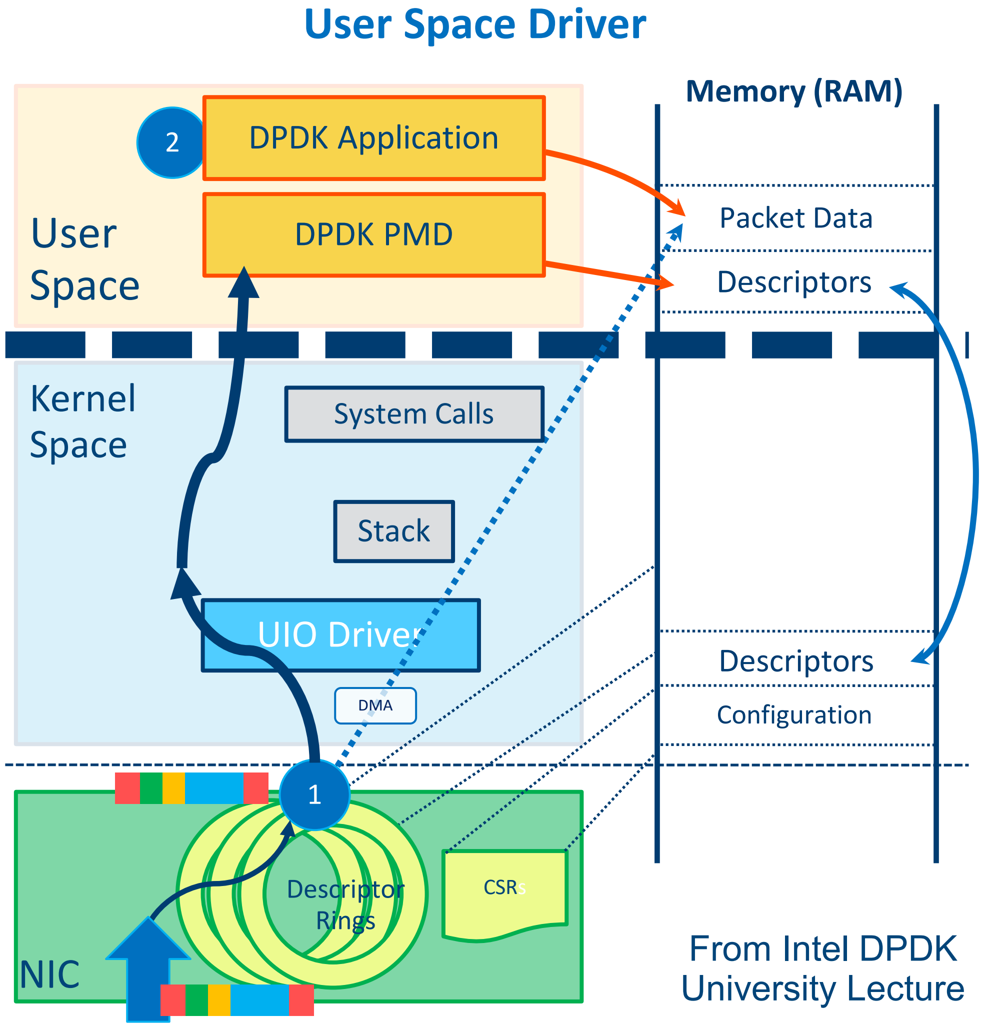


ASIC = „fixed silicon chip for special purpose, e.g. packet switching"

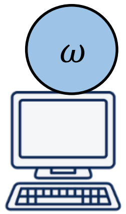
Performance Acceleration via INP

INP resources can reduce the time to move data, e.g.

- DPDK: circumvent OS
- OS Kernel: Enhance Communication Protocol
- NIC: process ahead of OS
- Switch : closer to producer/ consumer

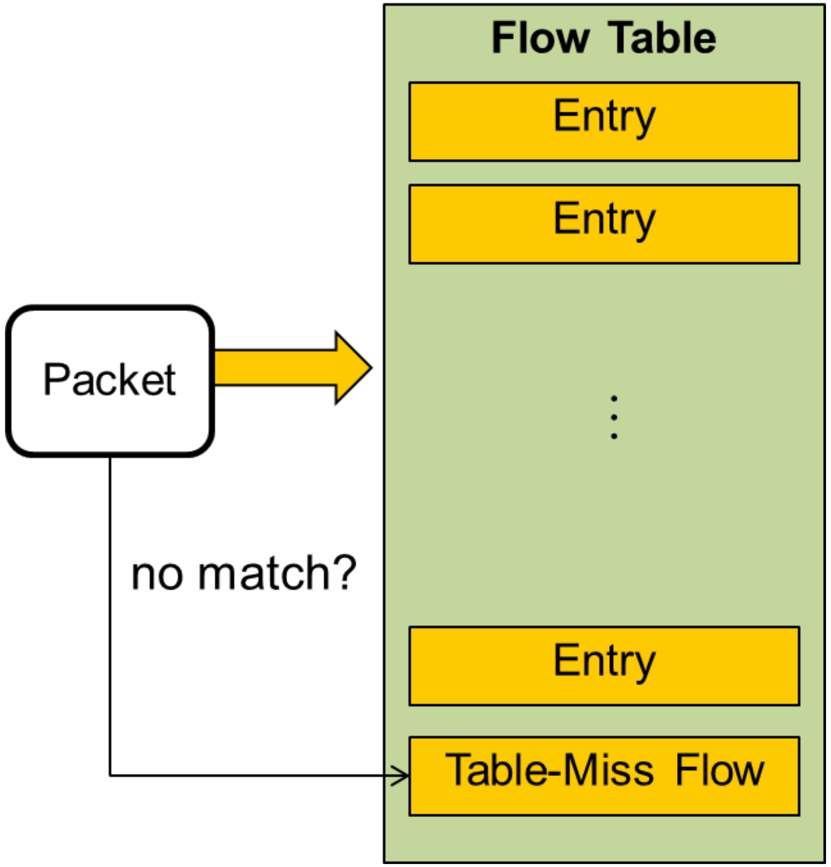


From Intel DPDK University Lecture



Performance Acceleration via INP

INP resources can reduce the time to move data, e.g.
- DPDK: circumvent OS
- OS Kernel: Enhance Communication Protocol
- NIC: process ahead of OS
- Switch : closer to producer/ consumer

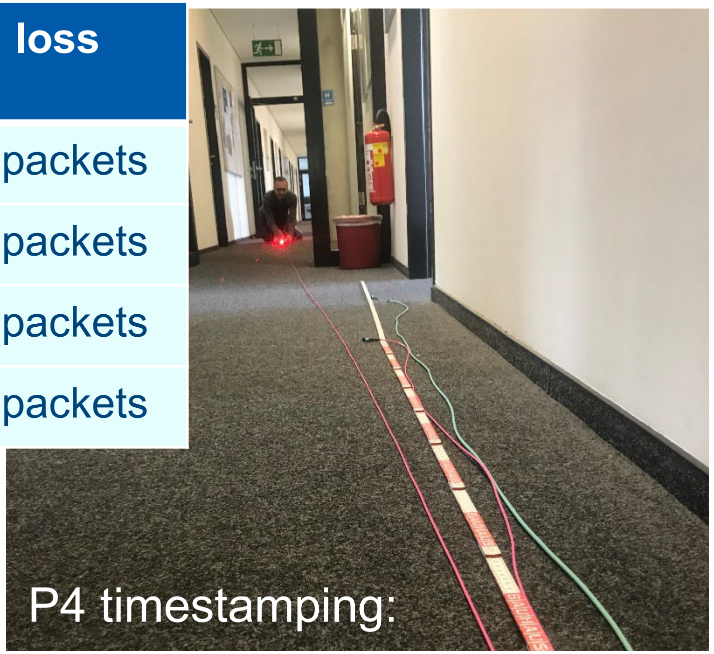


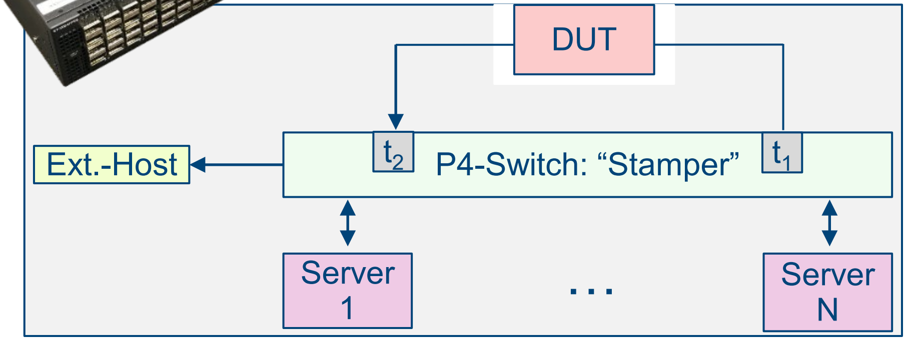
INP resources can accelerate the processing time
- Efficient Matching : TCAM
- Transformation and routing

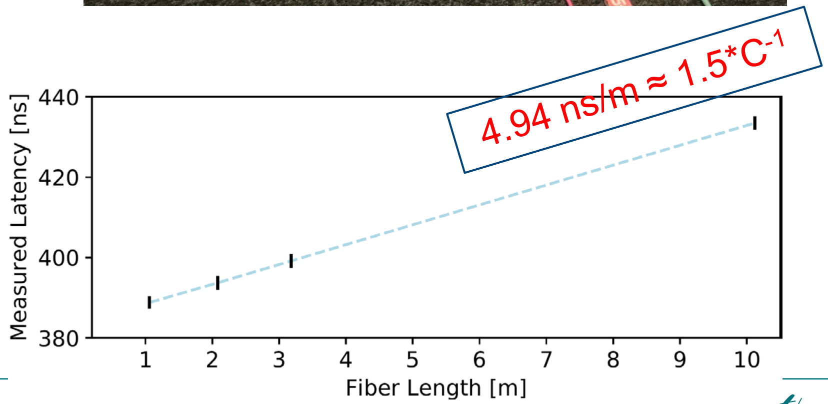
INP enables dynamic exchange of functionality

Is INP = Low Latency?
High Performance Packet Analytics in P4STA

	actual fiber length	avg. latency	std. dev.	loss
1 m	1.06 m	107.830 ns	1.46 ns	0 packets
2 m	2.08 m	112.850 ns	1.61 ns	0 packets
3 m	3.18 m	118.336 ns	1.52 ns	0 packets
10 m	10.12 m	152.883 ns	1.61 ns	0 packets







Kundel, Siegmund, Blendin, Rizk, Koldehofe. **P4STA: High Performance Packet Timestamping with Programmable Packet Processors.** In Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2020)

Challenges in using them for Real-time analytics

Specific domain specific programming models

- OpenFlow, P4, Verilog

Breaking distribution transparency

- E.g., applications does not work on byte streams, but packets!

Increased heterogeneity

Headers may leak information on the packet content

Outline

Why low latency response?

The Bottleneck in Data Movements

In-Network Computing Technologies accelerating performance

Examples in the context of Distributed Event-Based Systems

Conclusion

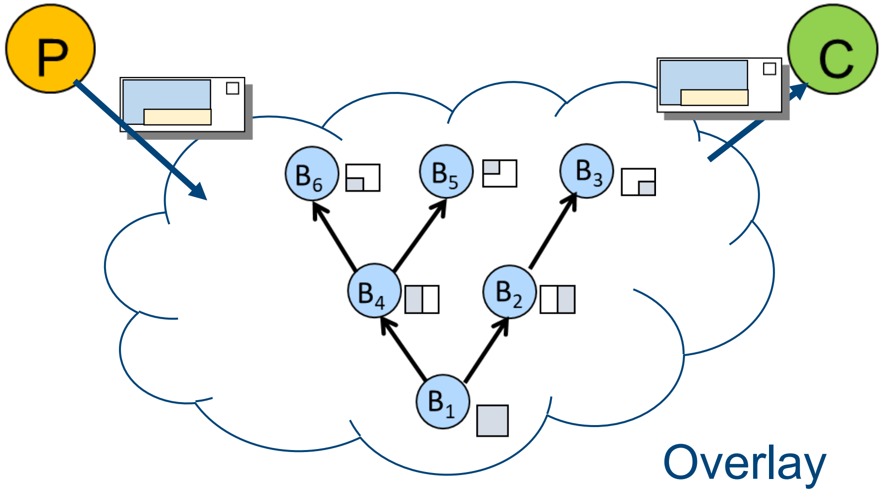
Publish/Subscribe and Performance

Efficient distribution by means of overlays

Bandwidth efficient overlays

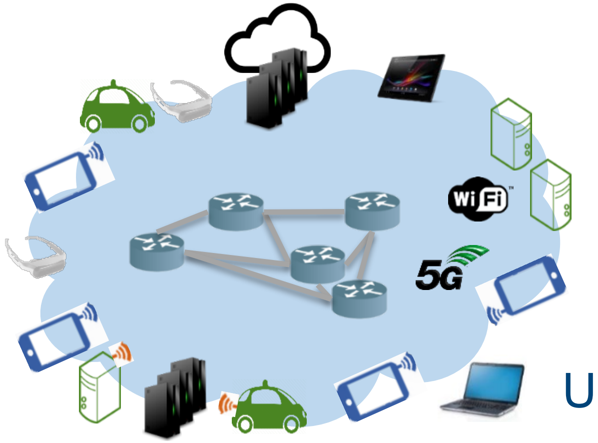
BUT big performance gap

- Overlay
- Underlay



Overlay

vs

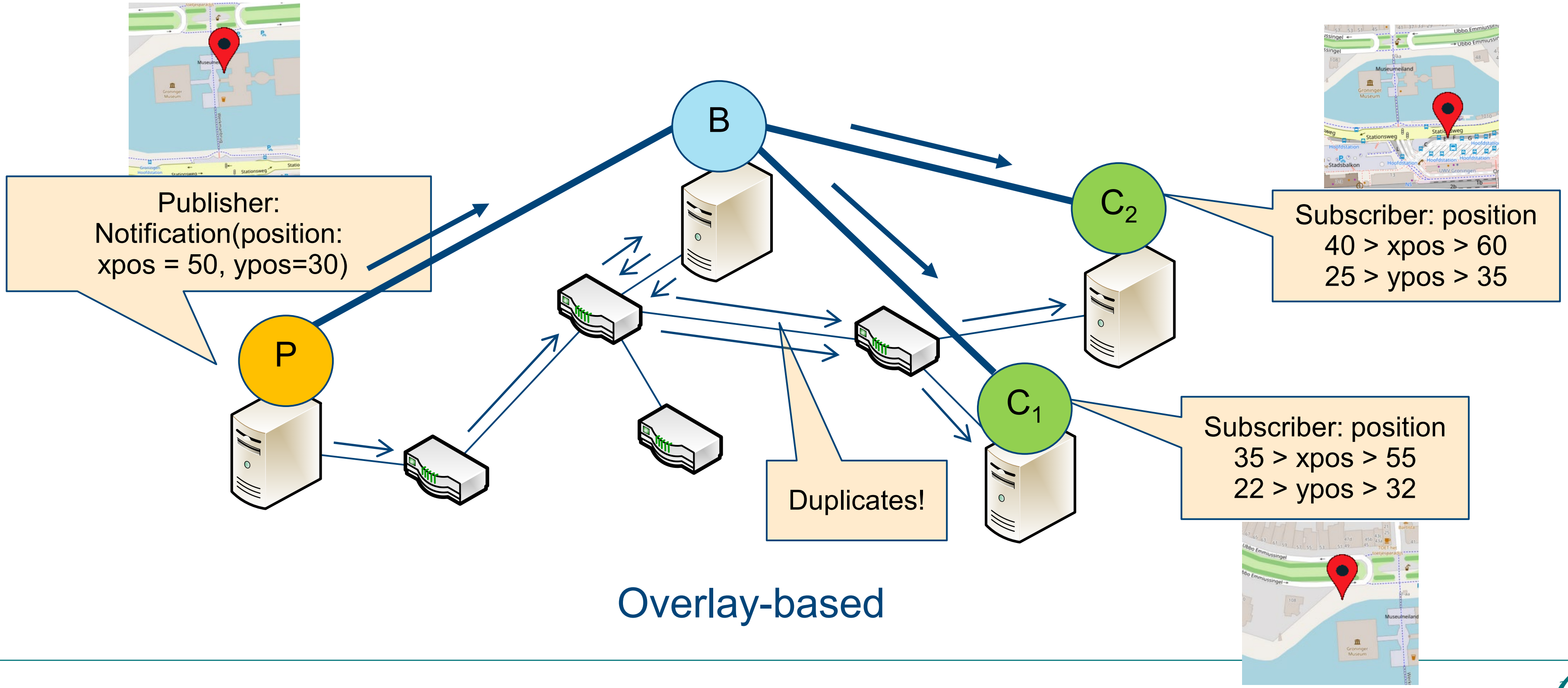


Underlay

High Performance Publish/Subscribe: Basic Idea

Reduce the overhead:

- Message duplications
- Matching subscriptions at the hardware

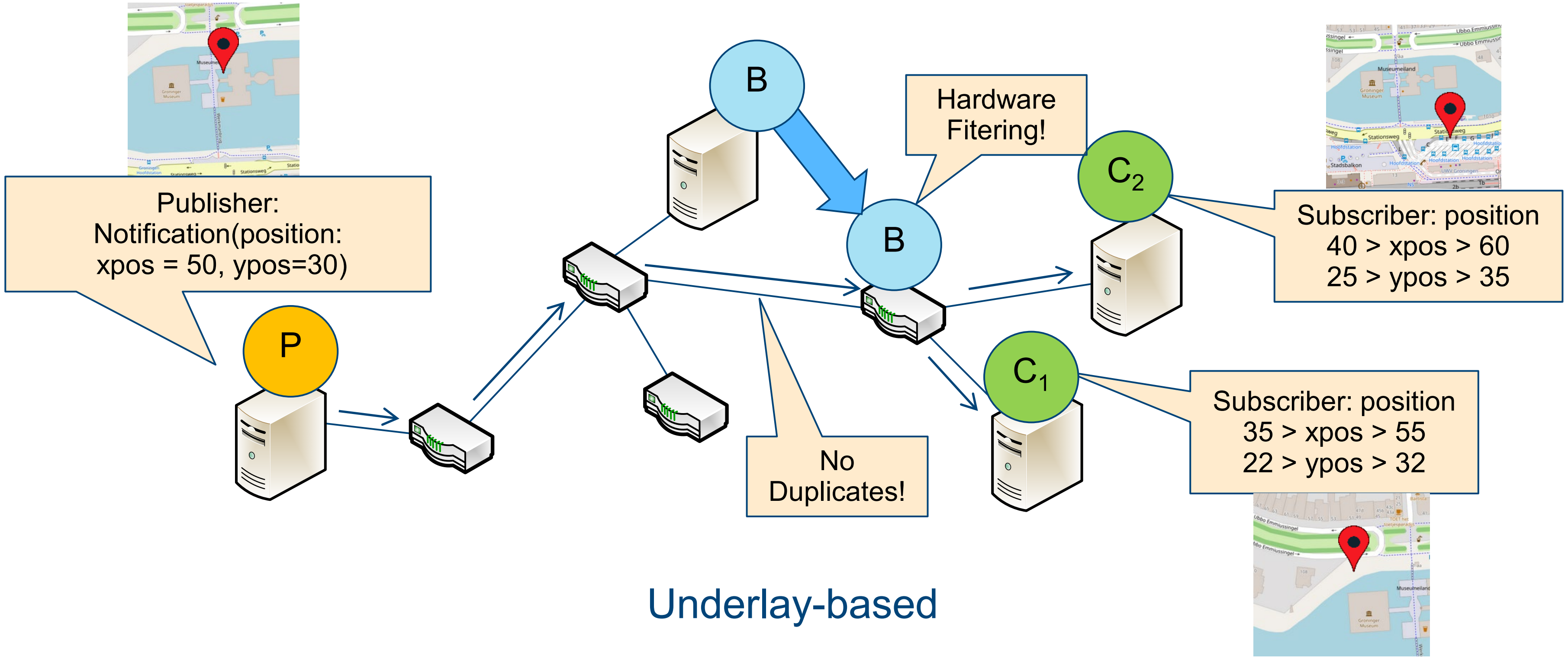


Overlay-based

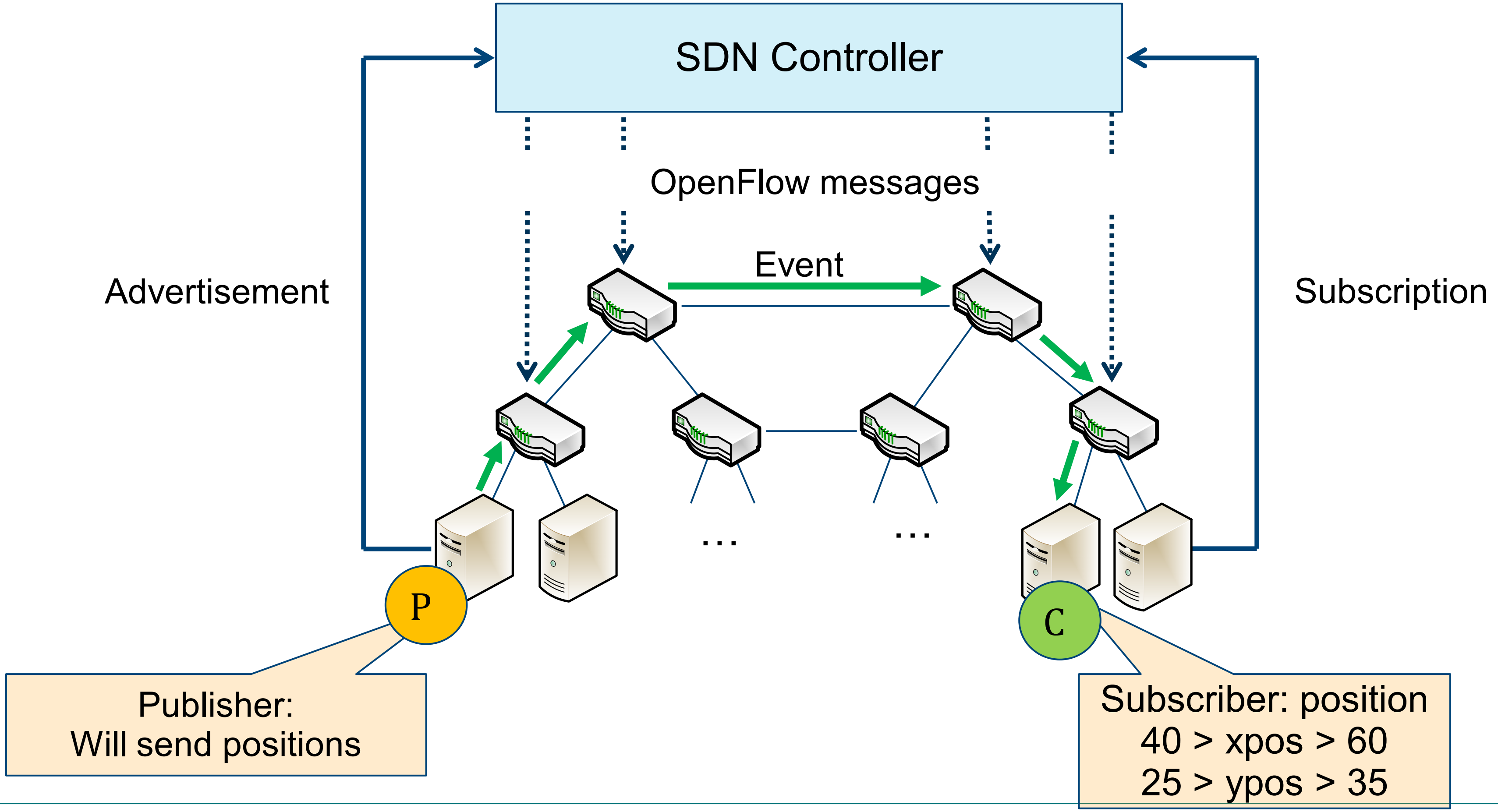
High Performance Publish/Subscribe: Basic Idea

Reduce the overhead:

- Message duplications
- Matching subscriptions at the hardware



SDN-based Publish/Subscribe Middleware



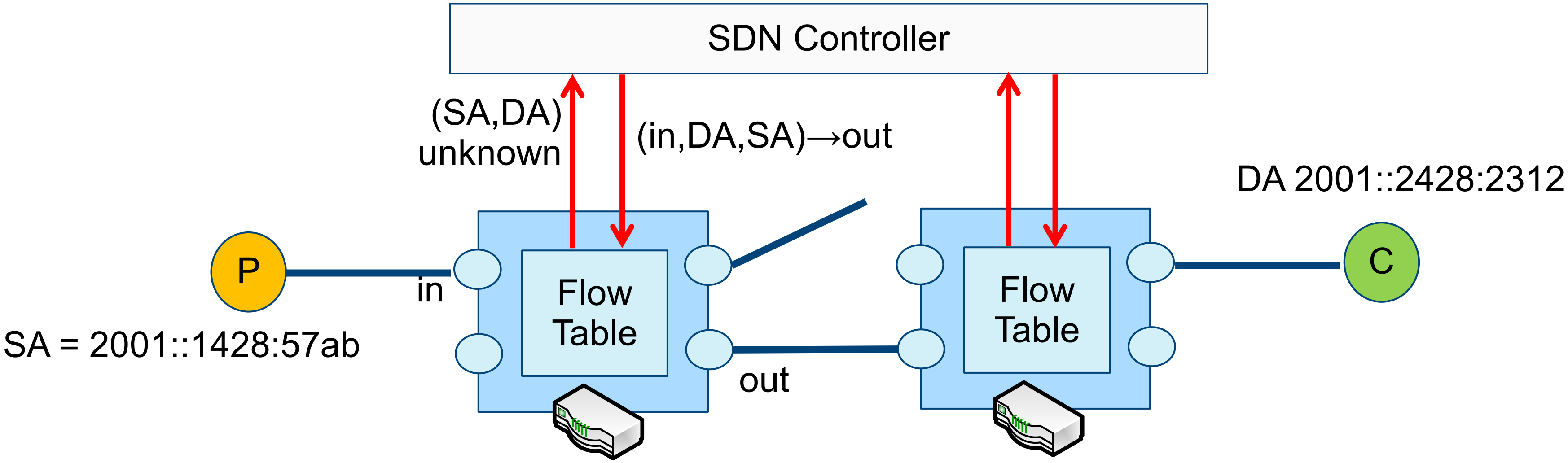
Bowmik, Tariq, Koldehofe, Kohler, Dürr, Rothermel. **High Performance Publish/Subscribe Middleware in Software-defined Networks.** IEEE Transactions on Networking (ToN), 2016.

Configuration Based on OpenFlow

Forwards packets from *in ports* to *out ports* by means of flow table, e.g.,

In port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Prot	Src	Dst

Controller can add, change and remove flow entries using OpenFlow

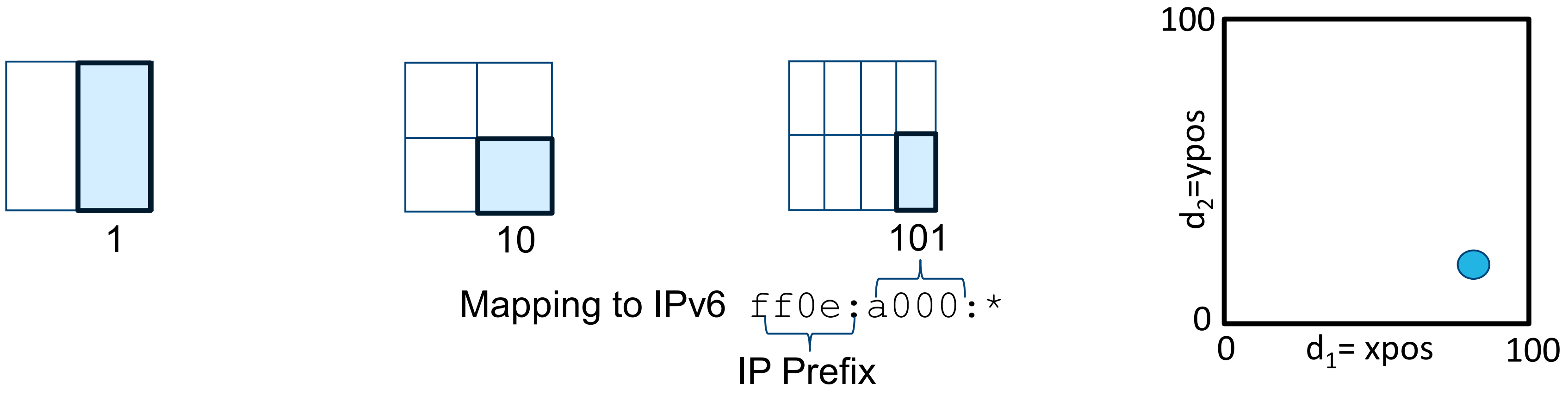


RQ: How to represent and match content-based subscriptions, e.g. in OpenFlow?

Bowmik, Tariq, Koldehofe, Kohler, Dürr, Rothermel. **High Performance Publish/Subscribe Middleware in Software-defined Networks.**
IEEE Transactions on Networking (ToN), 2016.

Subscription and event matching in flow table

1. **Generate binary representation based on spatial indexing**
2. **Map binary representation to IPv6 Multicast address**
   - Coexistence with other services



Approach overview

Subscription/Advertisement

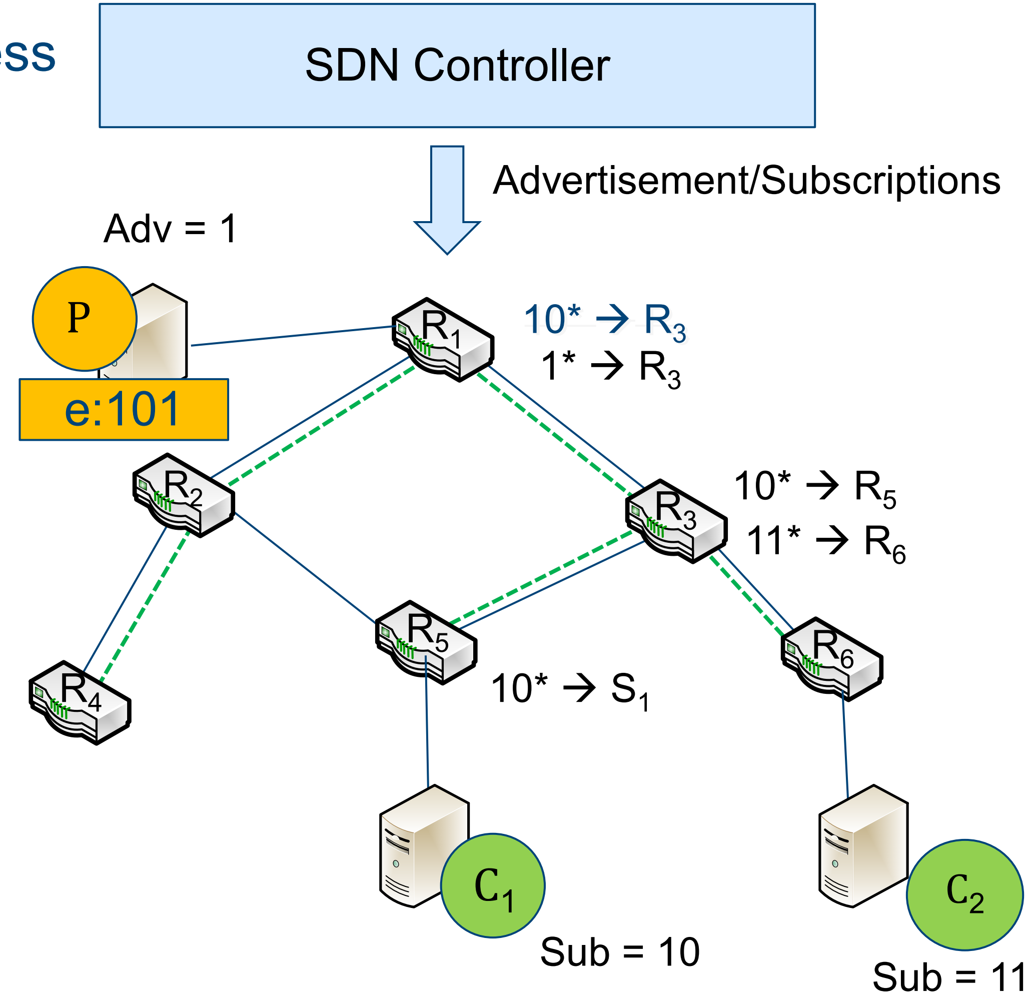
- Sent to controller with predefined IP address

Controller optimizes topology

- Establish paths between publishers and subscribers
- Paths are established along a tree

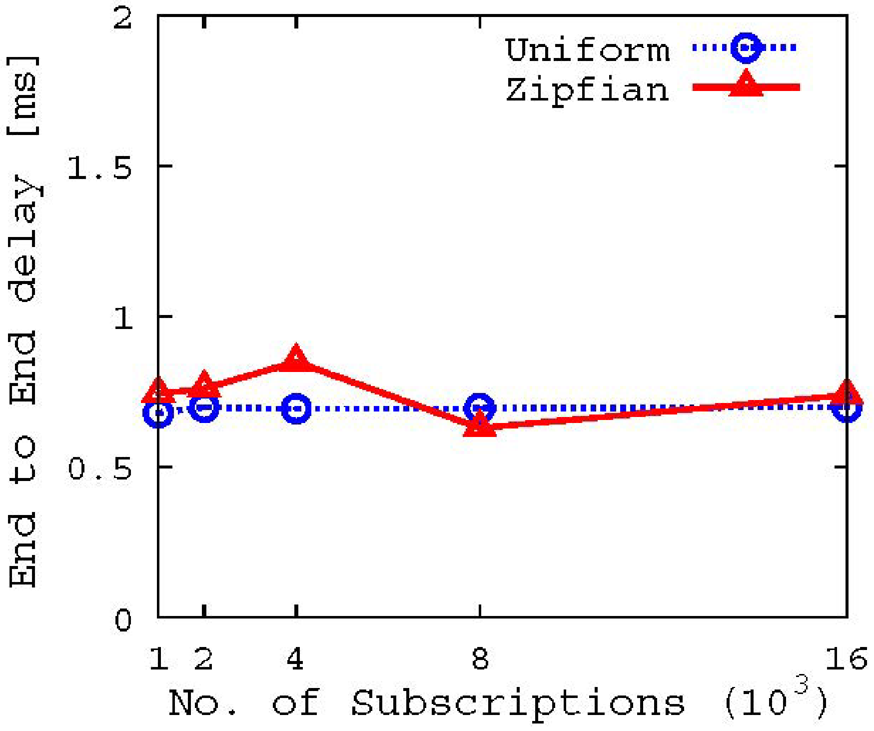
Events

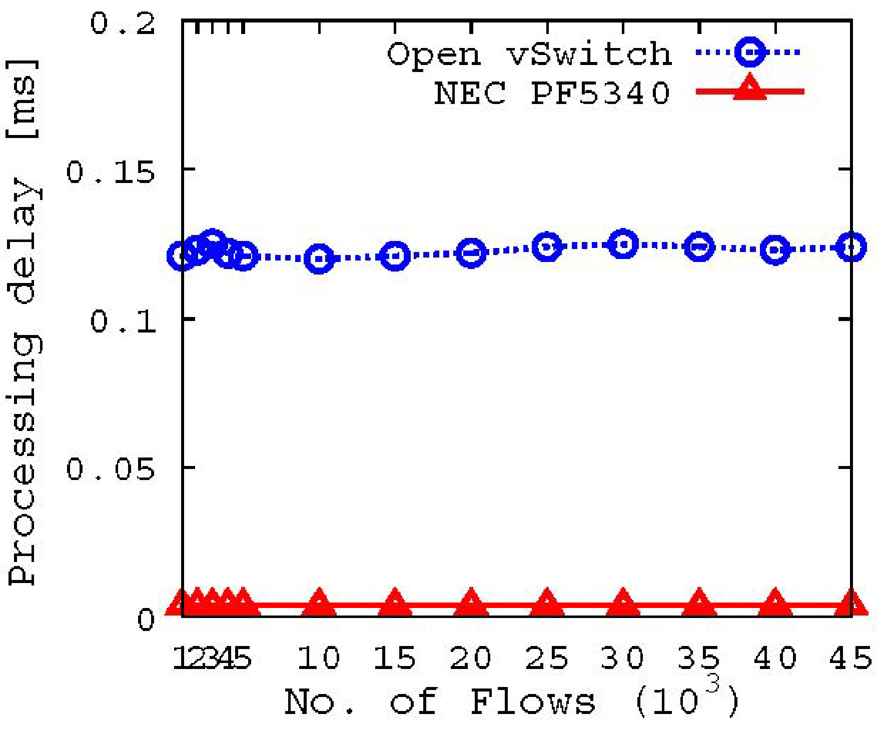
- Directly sent to the network
- $IP_{Prefix} \circ$ bit string



Bowmik, Tariq, Koldehofe, Kohler, Dürr, Rothermel. **High Performance Publish/Subscribe Middleware in Software-defined Networks.**
IEEE Transactions on Networking (ToN), 2016.

Result: Forwarding performance





Hierarchical fat-tree topology

10 Open vSwitches and 8 end-hosts

10,000 events

Properties

OpenFlow-based Management enables expressive subscription management

But requires from every publisher/ subscriber

- Understand the encoding

Relied on specific Header Fields!

But would work in general using a big field or mask

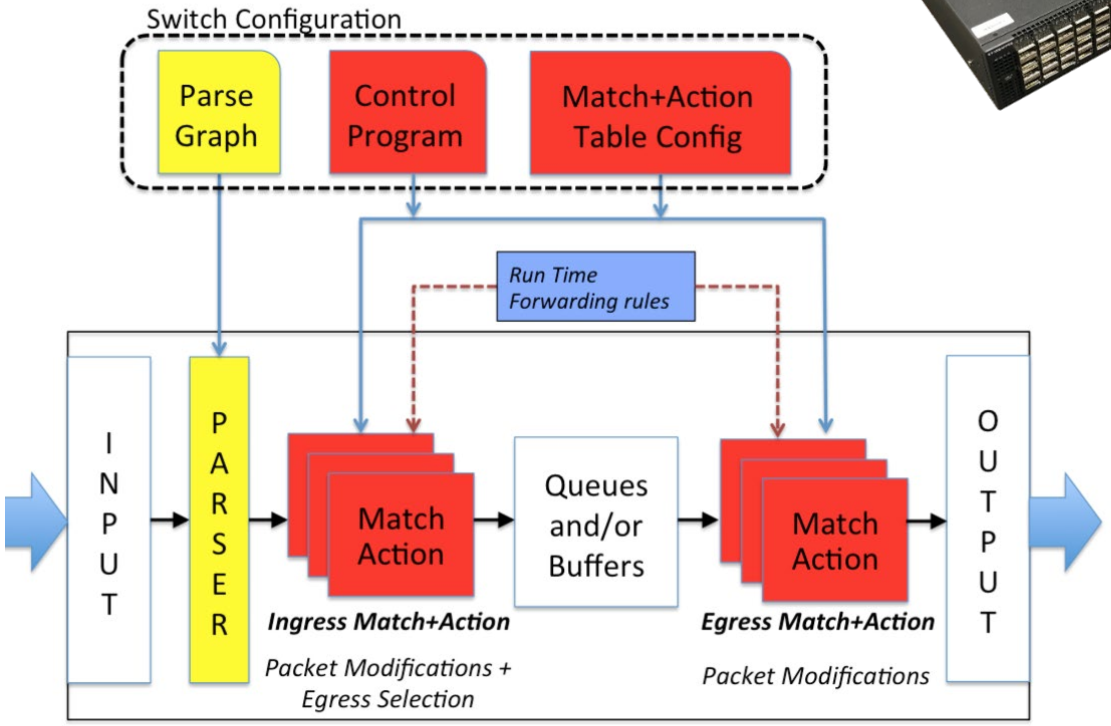
More Complex, state-full not considered!

Extending to P4 based INP

- **P4 supports Programming Reconfigurable Match Action Pipeline**
- **Define own Protocol Headers to be used by DEBS**
- **Define Matching Operations for specific Header fields**

```
typedef bit<32> timestamp_t;
typedef bit<16> type_t;
typedef bit<8>  attribute_t;
typedef bit<8>  value_t;

header event_h {
    type_t type; /* example: weather. */
    timestamp_t  timestamp; /* event occurance time. */
    attribute_t  attribute; /* example: humidity, temperature. */
    value_t  value; /* example: 45% humidity and 23 degrees celsius temperature. */
}
```



Source p4.org

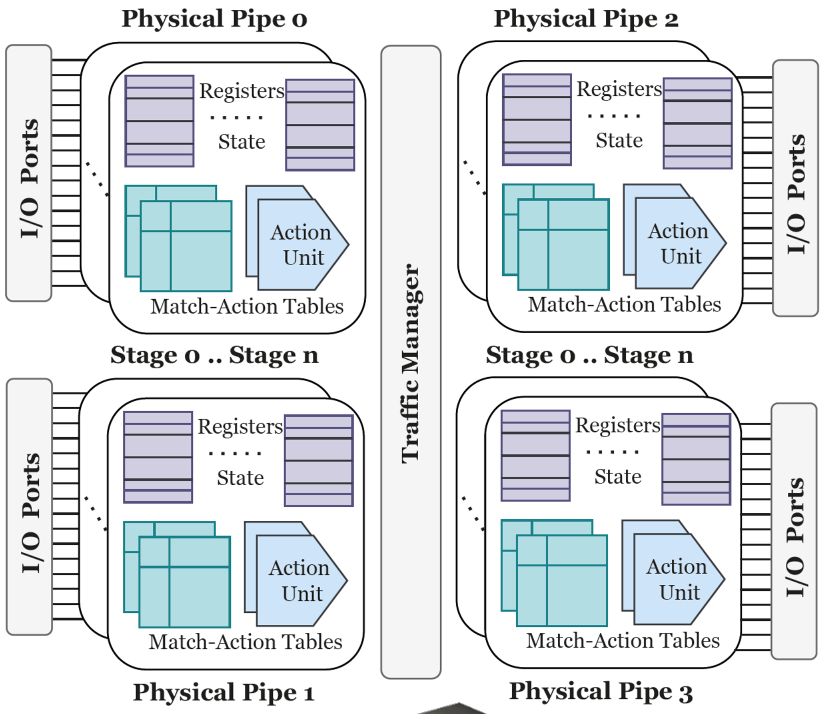
P4: Enhancing Stateful Operations

Limited Support for Stateful Operations

Many pitfalls:

- No sharing of registers between different stages of the pipeline
- Exclusive read or write operations
- Packet cannot iterate over all registers

However, can be used to model for specific platforms stateful CEP operators!



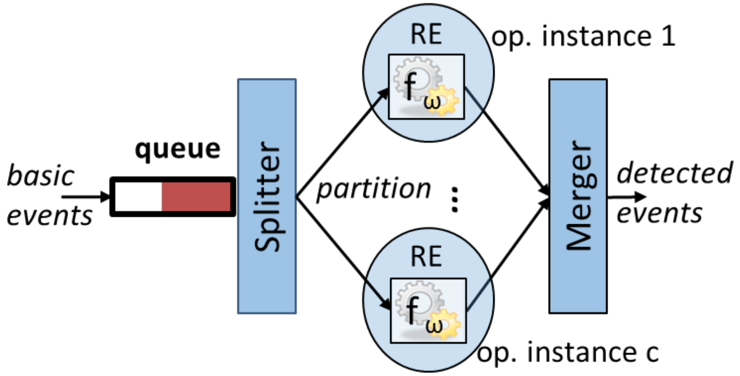
Kohler, Mayer, Dürr, Maaß, Bhowmik, and Rothermel. *P4CEP: Towards In-Network Complex Event Processing.* In Proceedings of the 2018 Morning Workshop on In-Network Computing (NetCompute ‘18, pp. 33–38. https://doi.org/10.1145/3229591.3229593

Supporting parallel operator execution with P4

Operator Parallelization is a common method in DEBS

Splitter:

- Partition streams in independent processable windows
- Operator instances return results to the merger
- Merger coordinates streams



Processing rate of the splitter is the bottleneck in scaling operators

Can be done already on the path between producers and consumers

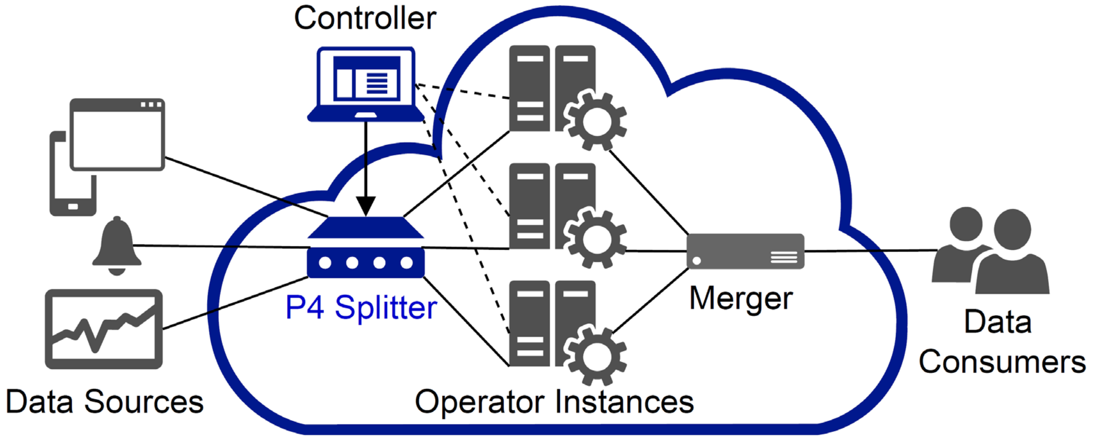
Ruben Mayer, Boris Koldehofe, Kurt Rothermel. **Predictable Low-Latency Event Detection with Parallel Complex Event Processing.** Internet of Things Journal, vol. 2(14), pp. 274–286. 2015. IEEE.

P4 Splitter: Window Operators

Idea: perform stream partitioning via INP

Problems:

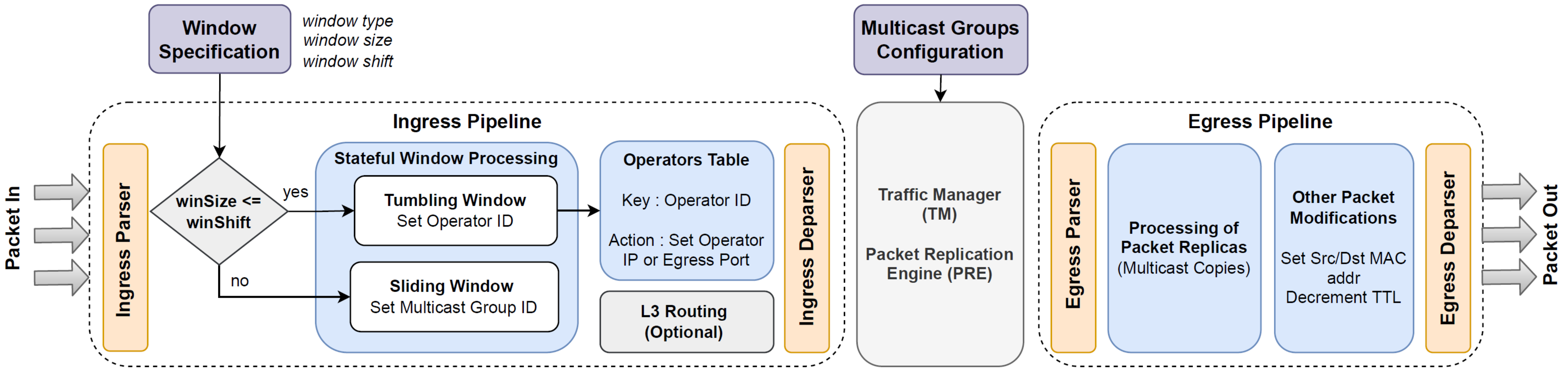
- Dynamically expressing multiple distinct window semantics for operators
  - Time-based, Count based, …
- Needs to be performed in line-rate with
  - Match Action Logic
  - Registers state



Bochra Boughzala, Christoph Gärtner, and Boris Koldehofe. **Window-based Parallel Operator Execution with In-Network Computing.** Proceedings of the 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), pp. 91–96, ACM press.

Basic Idea / Procedure

1. **Each stream identified by an id → Matching events**
2. **Window specifications can be dynamically added/removed**
   - with respect to a unique stream id (Dynamically Matching rules)
3. **Window state is captured via registers**
4. **Incoming events trigger updates to window state (dependent on window)**
5. **Will be added to a multicast group that sends an event packet to all destinations**

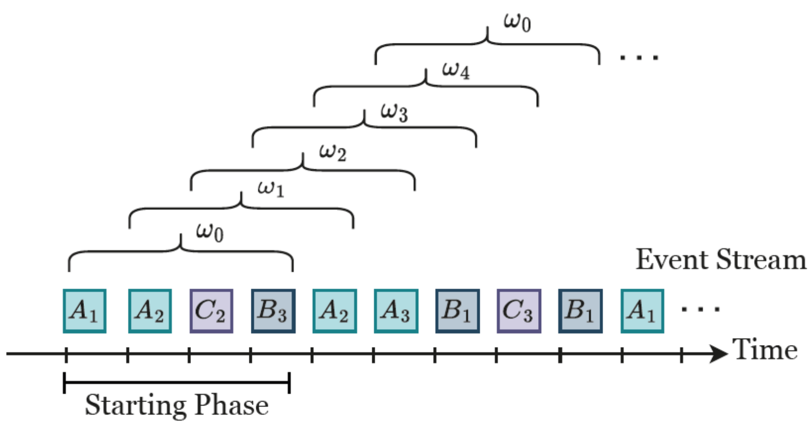


Example

Specifying a window semantics:

- e.g. Count-based Sliding Window
  - stream ID idx
  - parallelism degree N
  - window size n
  - window shift δ, $\delta \leq n$ (counting)

idx	latest Op.	$Overlap$	Op. Inst
0x23	ω_0	0	ω_0
0x23	ω_1	1	ω_0, ω_1
0x23	ω_2	2	$\omega_0, \omega_1, \omega_2$
0x23	ω_3	3	$\omega_0, \omega_1, \omega_2, \omega_3$
0x23	ω_4	3	$\omega_1, \omega_2, \omega_3, \omega_4$
0x23	ω_0	3	$\omega_2, \omega_3, \omega_4, \omega_0$
0x23	ω_1	3	$\omega_3, \omega_4, \omega_0, \omega_1$
0x23	ω_2	3	$\omega_4, \omega_0, \omega_1, \omega_2$



Round-robin load-balancing over 5 operator instances with n = 4, δ = 1.

Bochra Boughzala, Christoph Gärtner, and Boris Koldehofe. **Window-based Parallel Operator Execution with In-Network Computing.** Proceedings of the 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), pp. 91–96, ACM press.

Challenge: How to evaluate such a system?

Although P4 facilitates programming INP hardware, it is time consuming

One device costs ~5000-10000€

Huge Development effort

Don’t expect large scale comparison or a baseline comparison with Apache Flink

In general baseline comparisons

- Being faster neither straight forward nor very insightful

Bochra Boughzala, Christoph Gärtner, and Boris Koldehofe. **Window-based Parallel Operator Execution with In-Network Computing.** Proceedings of the 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), pp. 91–96, ACM press.

What did we evaluate:

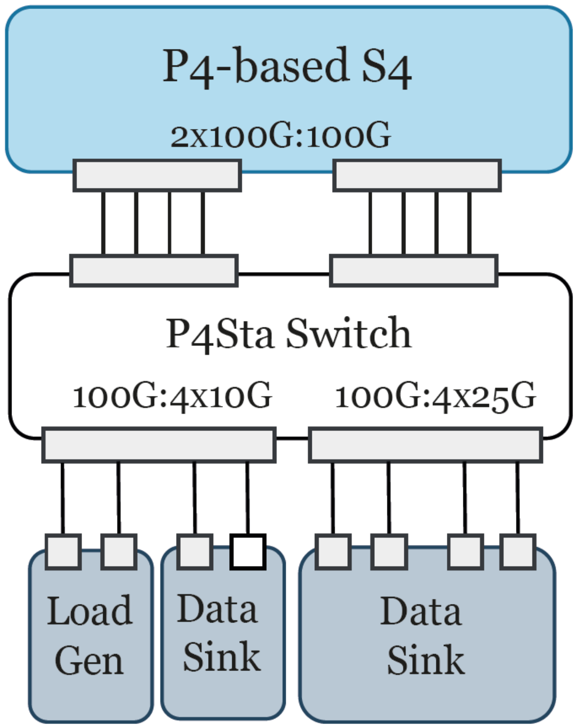
P4STA for Packet generation and validation

What is the latency introduced by a INP

Measure feasible throughput

Measure resources

- How many streams, operators, and windows can be supported



Some Findings
Throughput and Low Latency

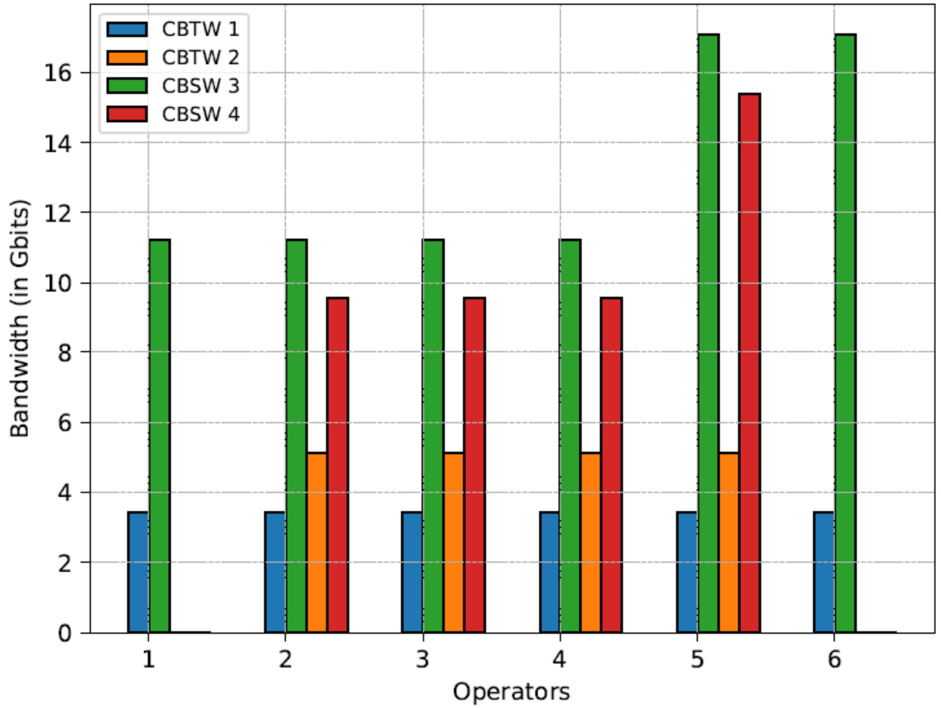
High Throughput low and stable latency

Throughput depends on window semantics

- Load generator is a bottleneck
- Higher parallelization degree and overlap increases bandwidth

Latency

- Independent of count-base vs time-based



Measurement	CBTW	CBSW	TBTW	TBSW
Average Latency	1.76 μs	1.86 μs	1.75 μs	1.87 μs
Minimum Latency	1.72 μs	1.8 μs	1.72 μs	1.83 μs
Maximum Latency	1.8 μs	1.92 μs	1.8 μs	1.93 μs

Stream	$\Sigma = (n, \delta)$	N
CBTW1	$\Sigma = (n = 100, \delta = 100)$	$N = 6$
CBTW2	$\Sigma = (n = 10, \delta = 10)$	$N = 4$
CBSW1	$\Sigma = (n = 5, \delta = 1)$	$N = 6$
CBSW2	$\Sigma = (n = 3, \delta = 1)$	$N = 4$

Bochra Boughzala, Christoph Gärtner, and Boris Koldehofe. **Window-based Parallel Operator Execution with In-Network Computing.** Proceedings of the 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), pp. 91–96, ACM press.

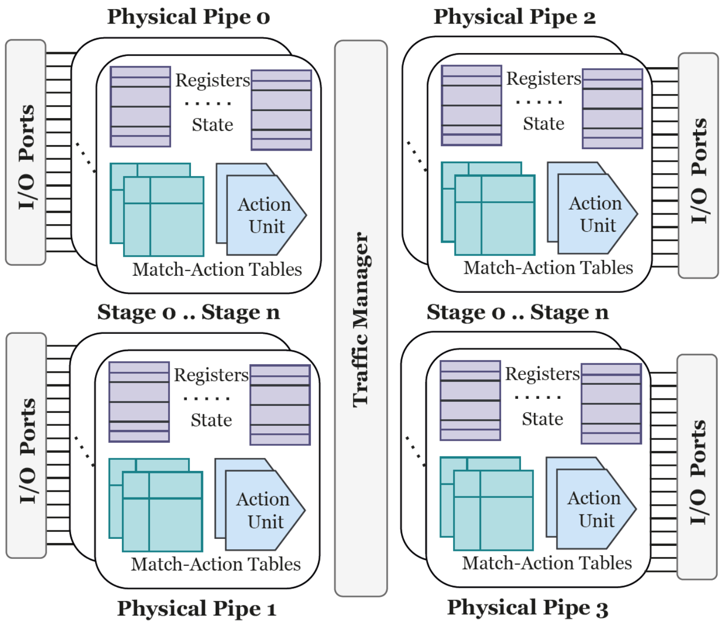
Some findings
Resource Usage

Resource Usage Determines Scalability

Comprises

- Stages, Tables, and Register Arrays
- Tofino1 has max 12 stages

Resource	CBTW	CBSW	TBTW	TBSW
Stages	6	6	7	8
Match Tables	12	12	18	20
Registers Arrays	3	2	6	7



- Could deploy with line rate-performance
  - Count-Based windows : 457k operators, 286k concurrent streams
  - Time-based windows: 362k operators and up to 65k streams

Bochra Boughzala, Christoph Gärtner, and Boris Koldehofe. **Window-based Parallel Operator Execution with In-Network Computing.** Proceedings of the 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), pp. 91–96, ACM press.

Interesting Approaches in INP for Data Driven Applications

Networking Community is working on many abstractions for Stateful INP

Challenge: understand practicality and applicability in Middleware services

RMT
[SIGCOMM' 2013]
Match-Action Tables

Domino
[SIGCOMM 2016]
Stateful Atoms

PiFo
[SIGCOMM 2016]
Queue Management

FloBlaze
[NSDI-2019]
FSAs

FloBlaze
[NSDI-2019]
FSAs

Taurus
[ASPOLOS 2022]
Map Reduce

Thoanos
[SIGCOMM 2022]
Map Reduce

Adapted from Vishal Shrivastav presentation at SIGCOMM

But also very interesting work in distributed computing!

- E.g. " P4xos: Consensus as a Network Service", IEEE/ACM Transactions on Networking, 2020.

Bochra Boughzala, Christoph Gärtner, and Boris Koldehofe. **Window-based Parallel Operator Execution with In-Network Computing.** Proceedings of the 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), pp. 91–96, ACM press.

Everything on Performance?

Not really!

Data movements are the cause for high energy efficiency!

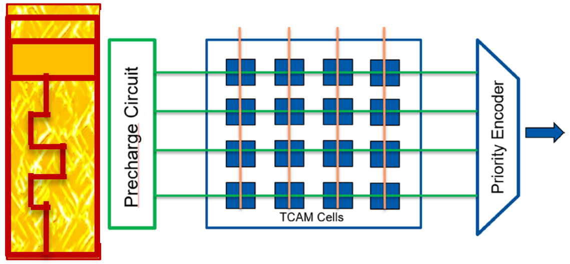
Moving to sustainable computing components!

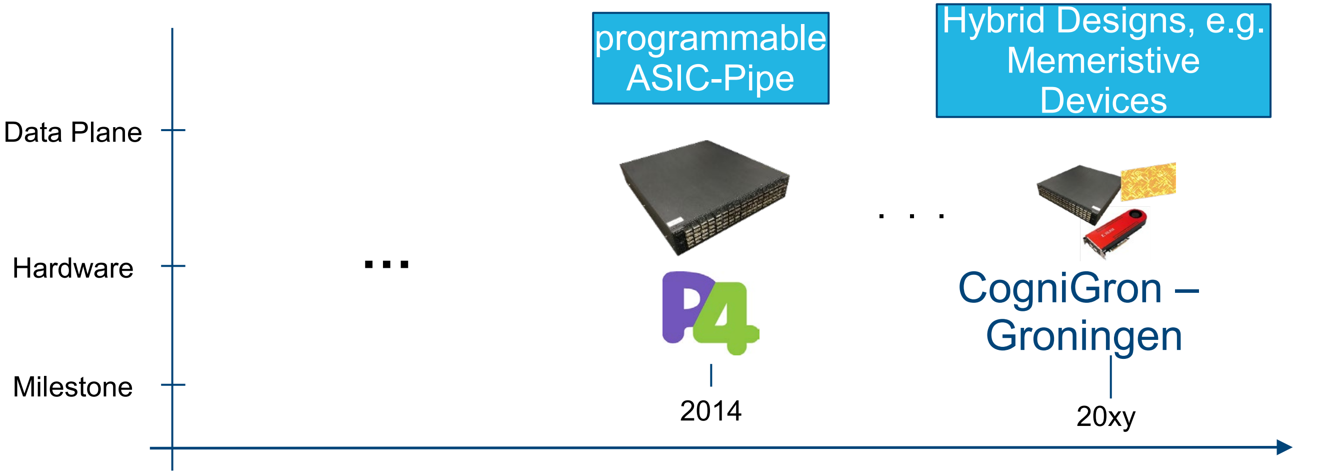
- Programmable ASIC for packet processing
- Very high bandwidth
- Limited flexibility

- Energy Efficient Switching?
- Computational Intelligence inside the network?

Recent example

- **TCA*m*M^CogniGron: Energy Efficient Memristor-Based TCAM for Match-Action Processing**





ASIC = „fixed silicon chip for special purpose, e.g. packet switching"

Saad Saleh, Anouk S. Goossens, Tamalika Banerjee, and Boris Koldehofe. **TCA*m*M^*CogniGron*: Energy Efficient Memristor-Based TCAM for Match-Action Processing.** In Proceedings of the 7th International Conference on Rebooting Computing (ICRC 2022). IEEE, 2022.

Conclusion

Distributed Real-Time Analytics is a fundamental and challenging paradigm in the Internet of Things

Accelerators based on In-Network Computing

- Reduce performance bottlenecks
- Utilize the Distributed Infrastructure more efficient

Distributed systems mechanisms

- Flexible usage of heterogeneous resources
- No single mechanism fits them all

Future Research:

- Better understanding of
  Distributed Computing + In-Network computing
- Energy-efficiency of In-Network Computing

Questions

Ralf Kundel and Fridolin Siegmund and Rhaban Hark and Amr Rizk and Boris Koldehofe. Network Testing Utilizing Programmable Networking Hardware. IEEE Communications Magazine, 7 pages, IEEE 2022.

Bowmik, Tariq, Koldehofe, Kohler, Dürr, Rothermel. High Performance Publish/Subscribe Middleware in Software-defined Networks. IEEE Transactions on Networking (ToN), 2016.

Bochra Boughzala, Christoph Gärtner, and Boris Koldehofe. Window-based Parallel Operator Execution with In-Network Computing. Proceedings of the 16th ACM International Conference on Distributed and Event-based Systems (DEBS '22), pp. 91–96, ACM press.

Prof. Dr. Boris Koldehofe

Technische Universität Ilmenau
Department of Computer Science and Automation
Distributed Systems and Operating Systems Group