In-Network Splitter Function Enabling Highly-Parallel Event Stream Processing

Bochra Boughzala[1], Boris Koldehofe[2]

[1] University of Groningen, The Netherlands.

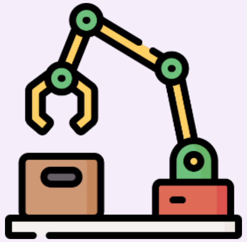
[2] Technical University Ilmenau, Germany.

4th GI/ITG KuVS Expert Discussion "Network Softwarization"

3/4 April 2025 (Online)

Modern Data Analytics Workloads

Data Sources



Industrial Automation

IoT, Sensing devices

Autonomous Driving

Role of Softwarized Networks ?

Real-time constraints
Performance, scalability requirements

Complex Event Processing (CEP)

Apache Flink

APACHE kafka™
A distributed streaming platform

Extraction of actionable insights on rapidly evolving situations

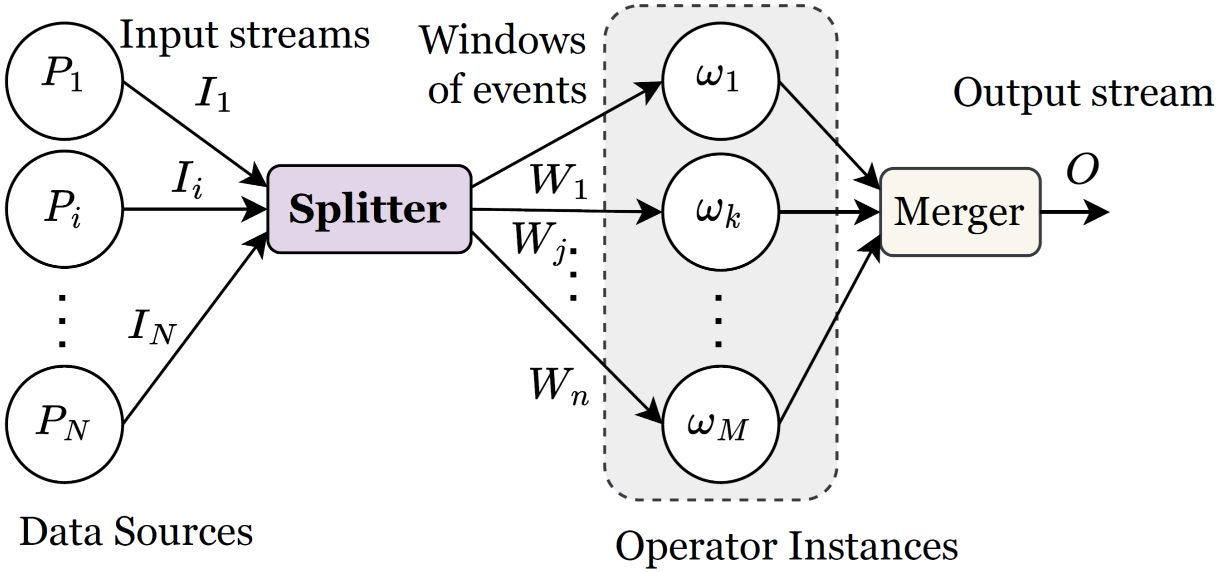
Image source: https://www.flaticon.com/

Parallel Stream Processing

Problem : Exponential Data Growth requires higher Throughput!

Solution : Data Parallelism

Splitter-Merger Architecture



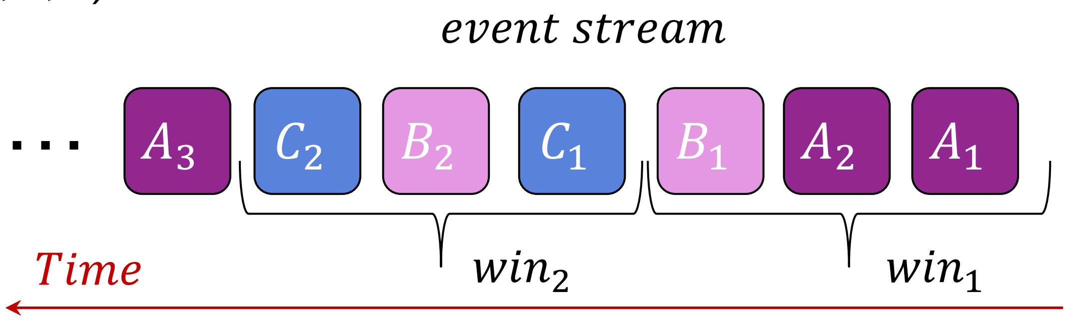
The Splitter Function

Window types in Streaming Systems

- Count-based windows
- Time-based windows
- Sliding windows (overlapping)
- Tumbling windows (non-overlapping)

Examples

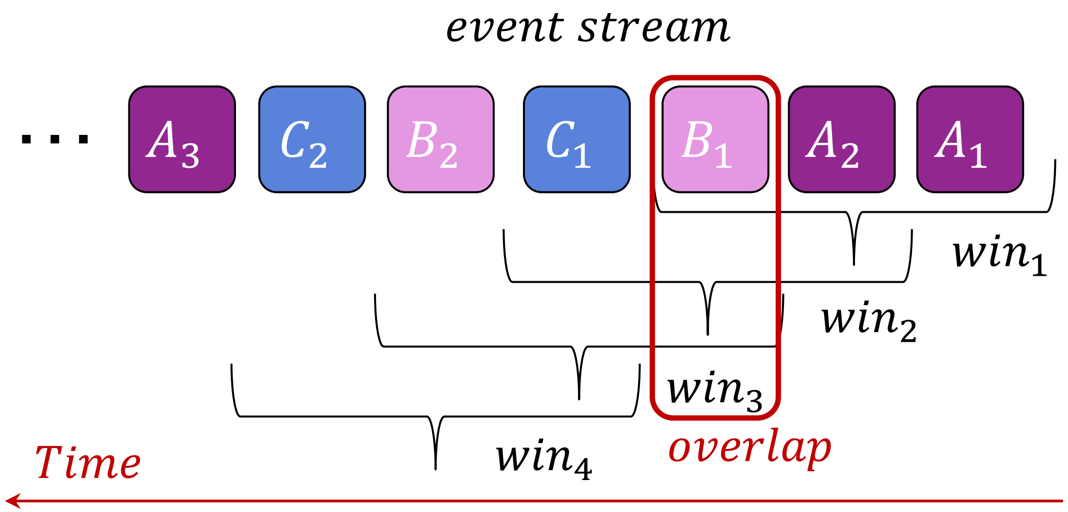
- *Count-based tumbling window* $\Sigma\ (n = 3, \delta = 3)$
- *Event pattern detection (A,B,C) → false*



The Splitter Function

Examples

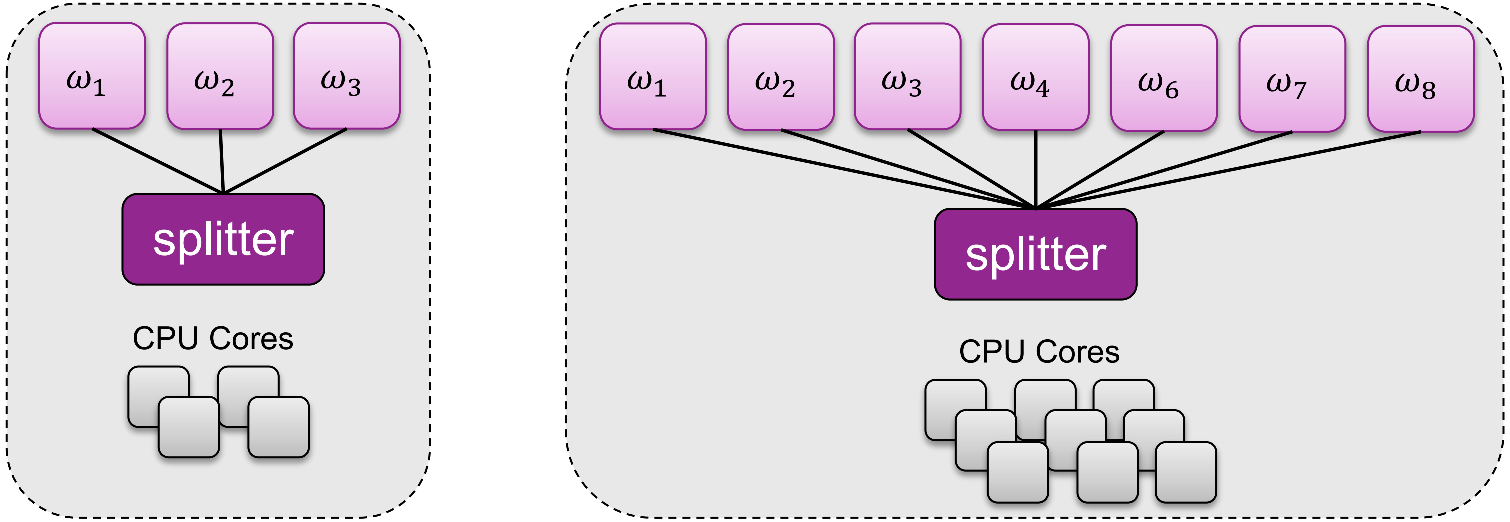
- *Count-based sliding window* Σ $(n = 3, \delta = 1)$
- *Event pattern detection (A,B,C) → true (window 2)*



The Performance of the Splitter

Scaling the throughput of the system

- Increase the degree of parallelism ~32 (maximum number of operator instances)
- Upper bound on the number of CPU cores in a single machine.

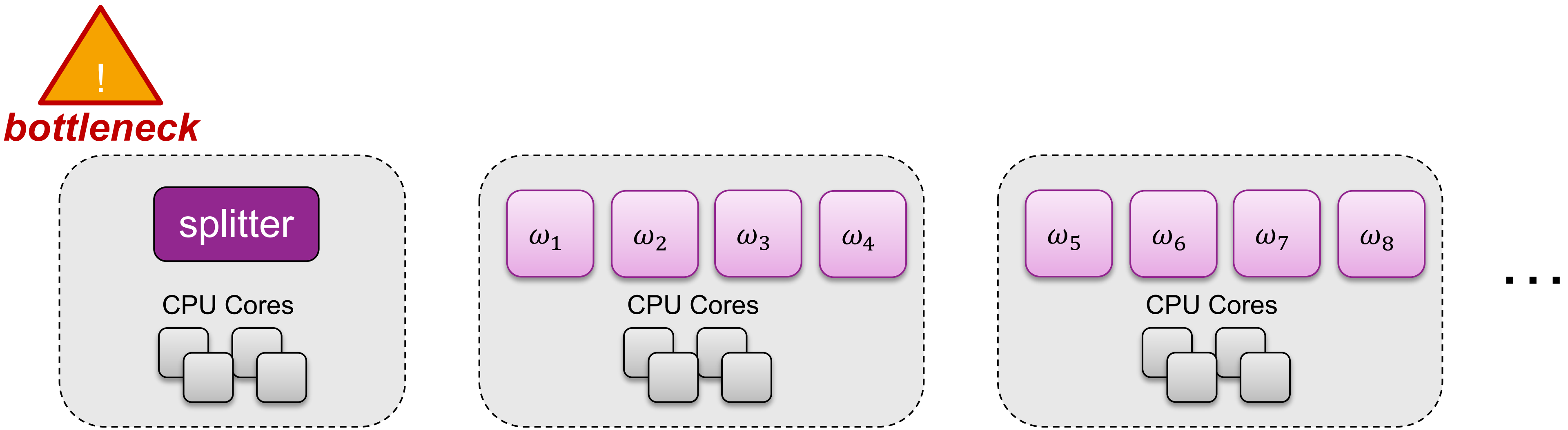


more and more CPU cores

The Performance of the Splitter

Distributed Execution

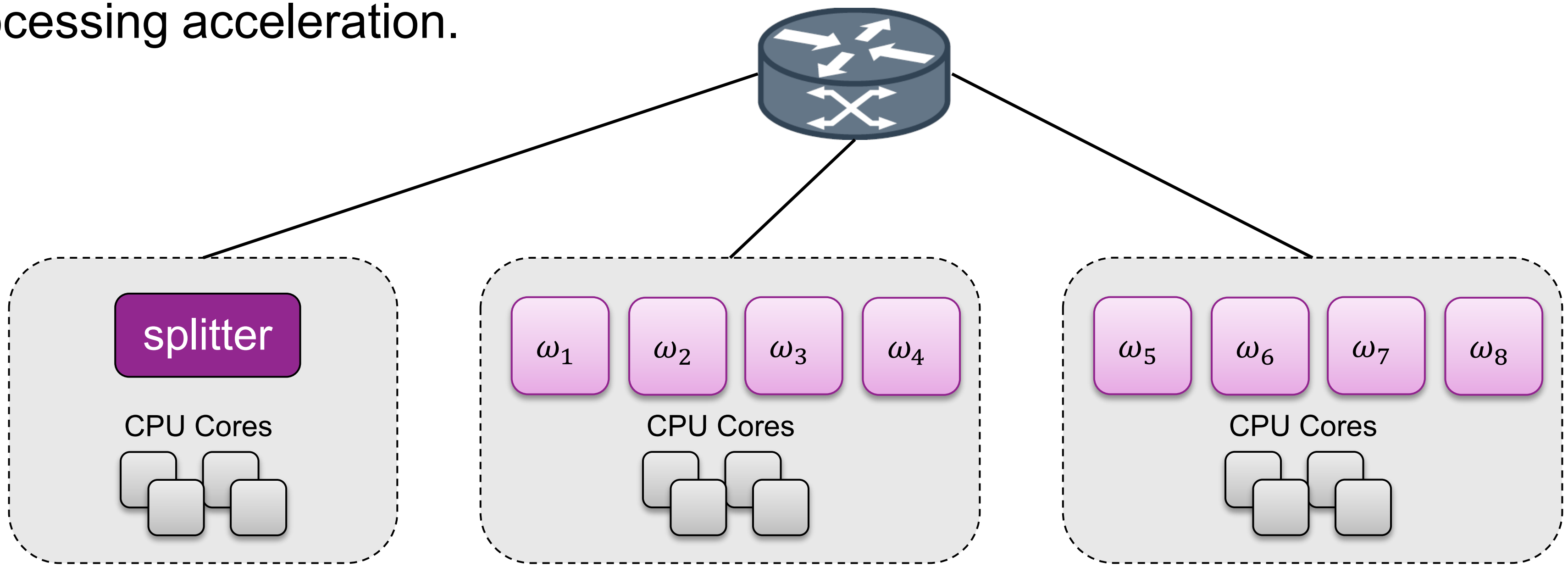
- Degree of parallelism is limited by the splitter performance



In-Network Computing

Emerging *in-network* programmable switches in modern Cloud infrastructure

- In-network programming model, e.g., P4.
- Splitter execution in the communication path.
- Stream processing acceleration.



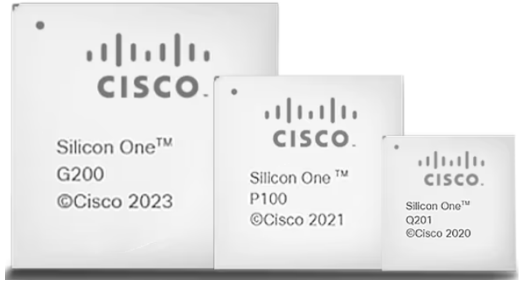
university of groningen

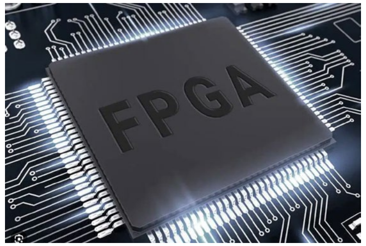
In-Network Computing

Performance Benefits

- Specialized hardware accelerators
  - High-speed Packet processors, TCAMs, …
    - Cisco Silicon One G200 51.2 Tbps
    - Intel Tofino 2 12.8 Tbps

- Less communication steps
  - Reduce latency
  - Save network bandwidth





Problem Description

Research Goal : Network-centric Data-analytics

- Leveraging in-network computing capabilities for speeding up and scaling parallel stream processing.

Research Questions

- *How to ensure the consistent and atomic transmission of windows of events as units by the splitter switch ?*

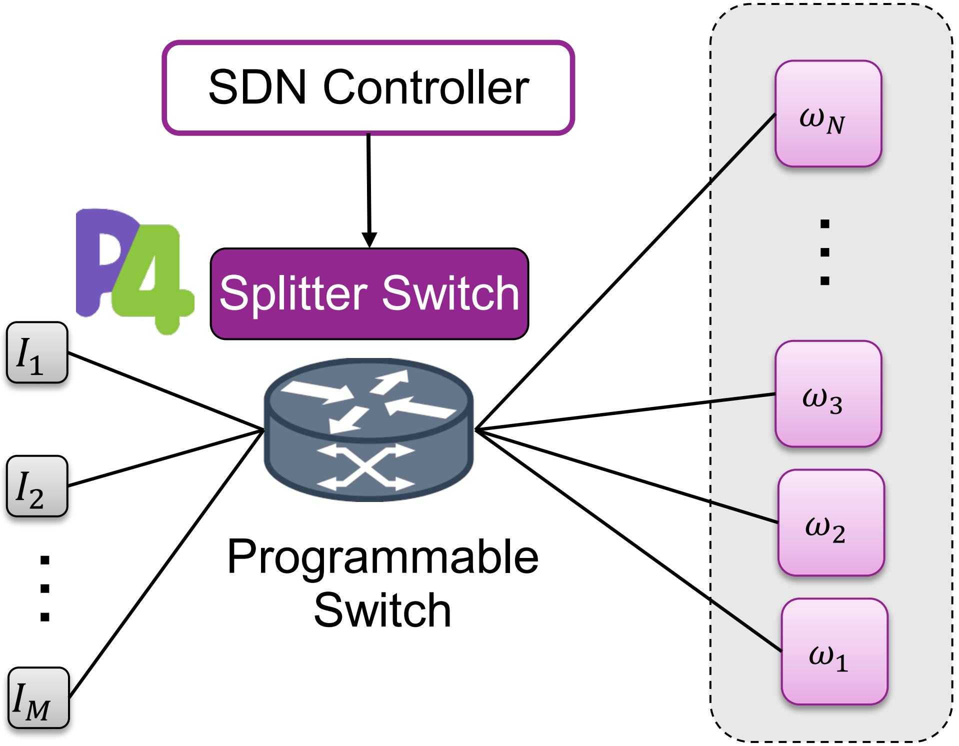
- *How to manage the window state and its persistent mapping to the correct operator(s) instance(s) while processing multiple event streams ?*

In-Network Data Parallelization Framework

In-Network Splitter Function

- Window-based Stream Splitting Logic
  - Various Windowing Semantics in the Data Plane
    1. Count-based Tumbling windows
    2. Count-based Sliding windows
    3. Time-based Tumbling windows
    4. Time-based Sliding windows

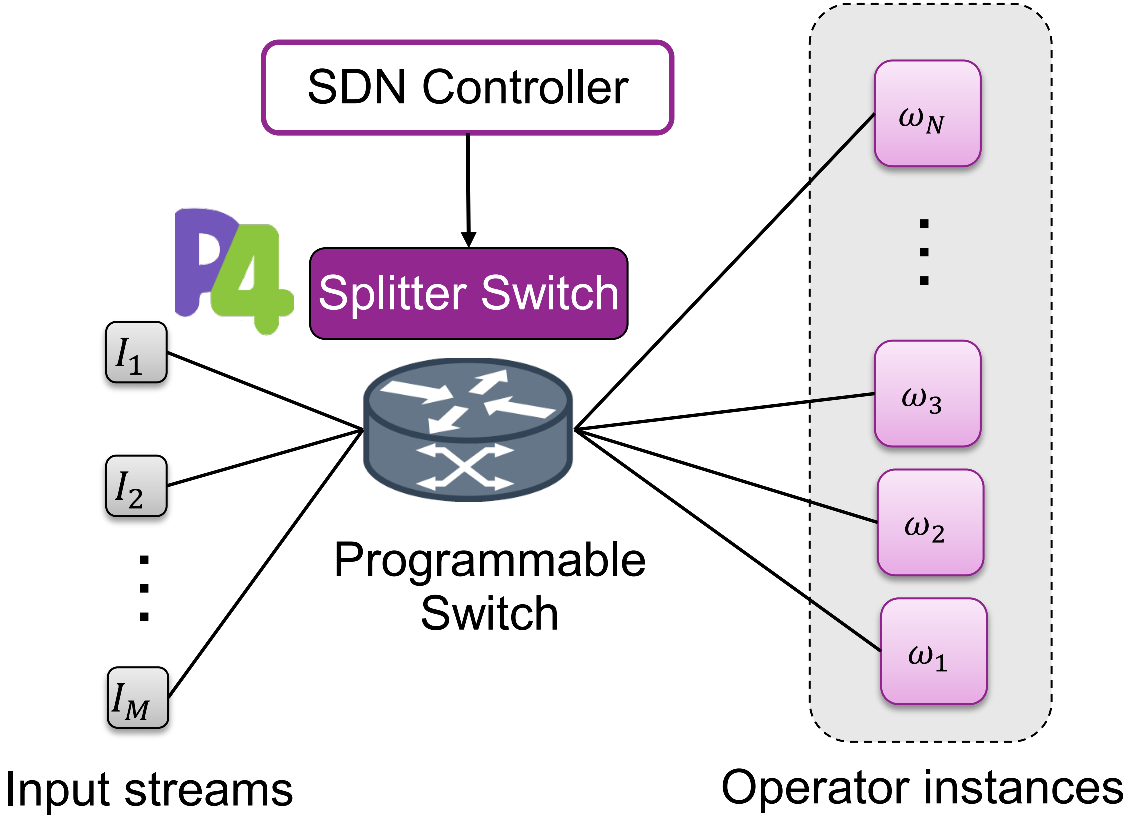
- Window Scheduling Logic
  - Load-balancing
  - Scheduling mechanism : round-robin



In-Network Data Parallelization Framework

Control plane interface

- Runtime configuration of splitter switch
  - Per input-stream window specification and parallelism degree
- Dynamic updates



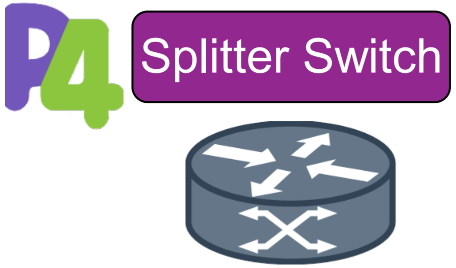
P4-enabled Splitter Switch

Key challenges

- Events from same window to be sent ***consistently*** to the same operator
- Windows to be sent ***atomically*** as units

State Management in the Data Plane

- Input stream to ingress port mapping
  - Unique Stream ID to each input stream used as register $\boldsymbol{index}$
- Retrieve configuration from match-action tables (Σ, N)
- State variables using registers and stateful ALUs
  - Tracking **position, operator ID,** current **overlap.**

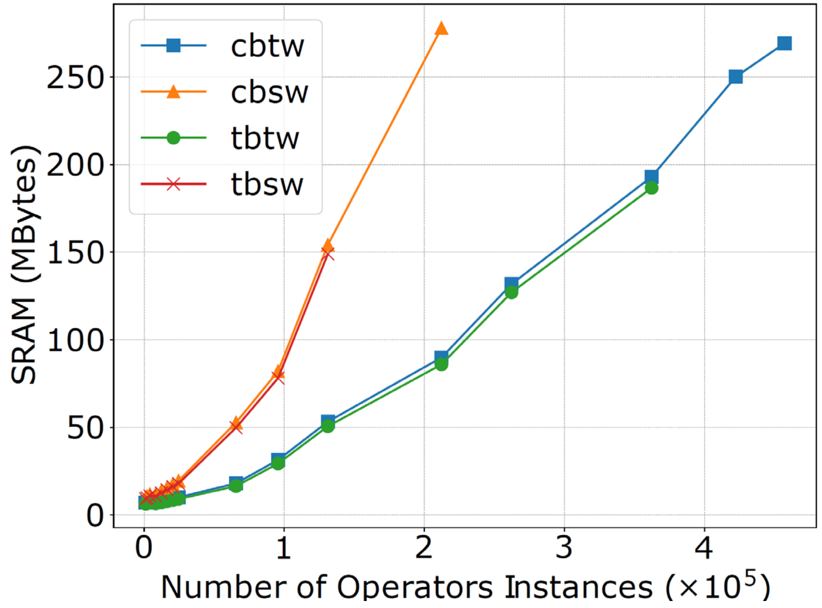


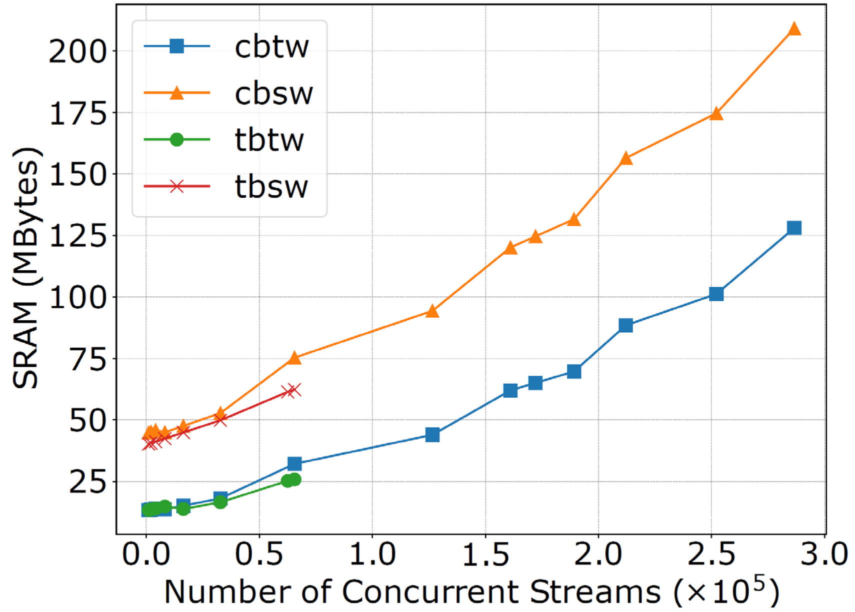
Programmable Switch

Key Findings

Scalability vs. Hardware Resources Consumption

- Note : Splitter switch standalone
- Scalability w.r.t. Parallelism Degree (fixed $32k$ data streams)
- Scalability w.r.t. Concurrent Data Streams (fixed $64k$ Operator Instances)

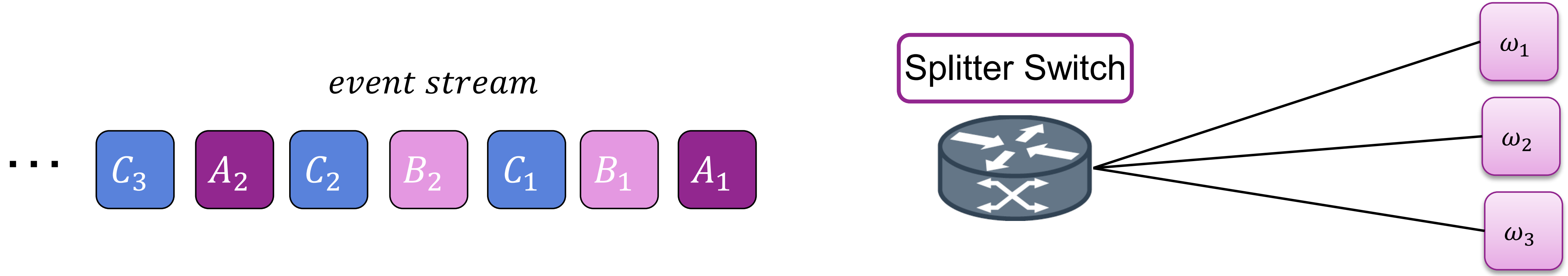




Handling Out-Of-Order Events

Order is important in Complex-Event Processing

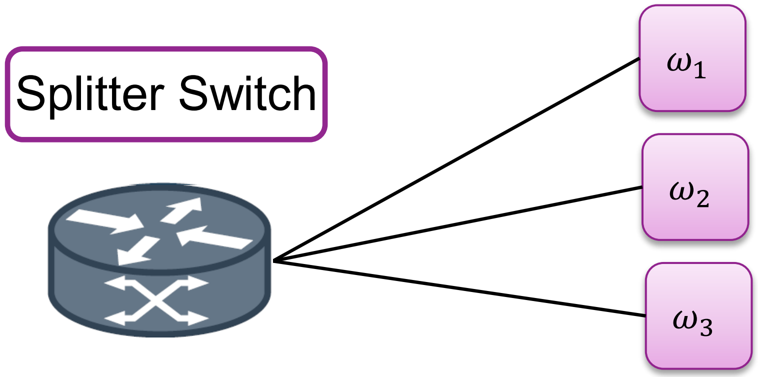
- Packet recirculation as a reordering mechanism
  - Detects a gap in sequence numbers $(e_1, t_1), (e_2, t_2), (e_3, t_3)$
  - Maximum number of recirculation attempts



Handling Single Point of Failure (SPoF)

Open challenge : what happens if the splitter switch crash ?

- Replication of the splitter switch
- Primary-secondary model (backup switches)



Conclusion & Future Work

In-network data parallelization framework for speeding up and scaling window-based parallel operator execution

- P4-based Stateful and Scalable Splitter Switch (P4SS [2], S4 [3])
- Different scaling capabilities depending on the window model

Conclusion & Future Work

Future Work

- Integration with stream processing engines, e.g., Apache Flink.

- Evaluation with real world dataset and concrete data-analytics applications.
  - Potential challenges : false positive/false negatives

- Varying assumptions.
  - One event per packet vs. multiple events per packet

References

[1] Boughzala, B. and Koldehofe, B., 2021, June. **Accelerating the performance of data analytics using network-centric processing**. In Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems (pp. 192-195).

[2] Boughzala, B., Gärtner, C. and Koldehofe, B., 2022, June. **Window-based parallel operator execution with in-network computing**. In Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems (pp. 91-96).

[3] Boughzala, B. and Koldehofe, B., 2024, June. **In-Network Management of Parallel Data Streams over Programmable Data Planes**. In 2024 IFIP Networking Conference (IFIP Networking) (pp. 50-58). IEEE.

Thank you for your attention.