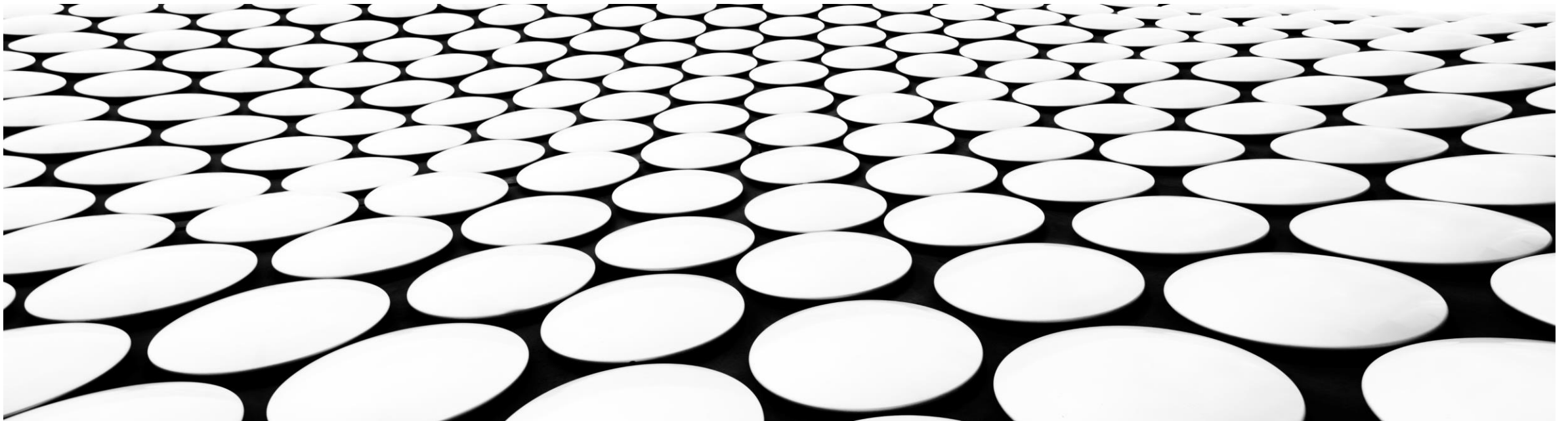




ON THE DESIGN OF FAST AND SCALABLE NETWORK APPLICATIONS THROUGH DATA STREAM PROCESSING

AUTHORS: ALESSANDRA FAIS, STEFANO GIORDANO, GREGORIO PROCISSI

PRESENTER: ALESSANDRA FAIS



OUTLINE

- Problem
- Main goals
- Background on Data Stream Processing
- State-of-the-art
- Proposed framework: fast and scalable continuous monitoring and data analysis in modern networks
 - Architecture
 - Research challenges
- Conclusion

PROBLEM: MANAGING MODERN NETWORKS

- Modern networks
 - Accommodate a variety of services on the same infrastructure
 - Adapt to requests for service (de-)activation
 - Always guarantee Quality of Service (QoS) requirements
- Network operators need
 - Rapid and easy network (re-)configuration
 - Continuous real-time network monitoring for detecting
 - Security issues
 - Performance degradation

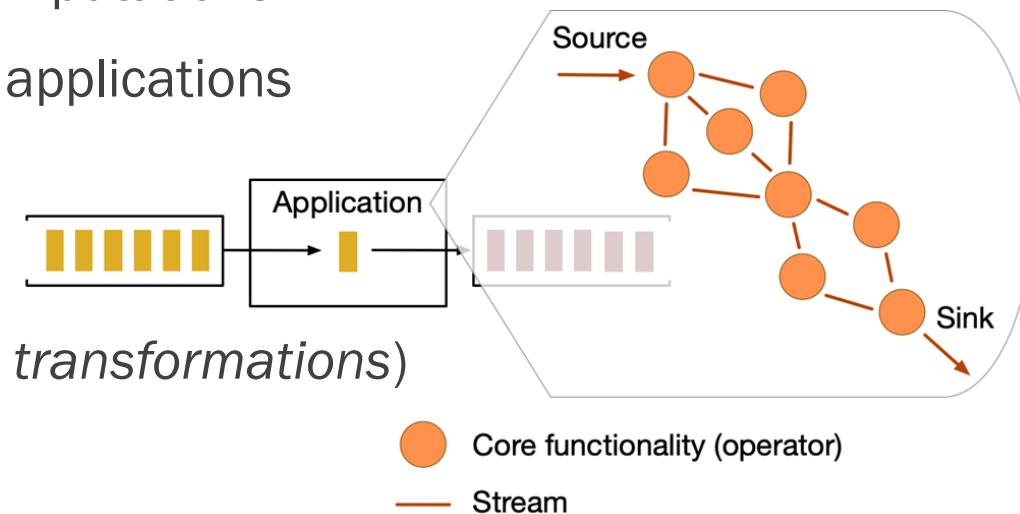
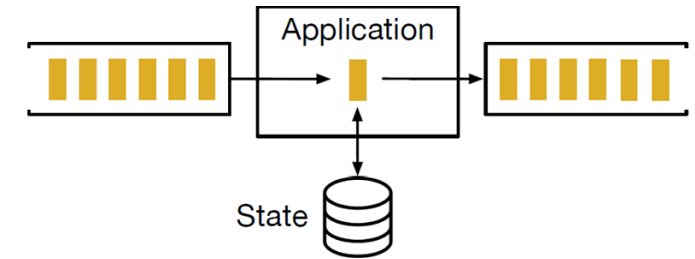
GOAL: IMPROVING CONTINUOUS MONITORING AND DATA ANALYSIS IN MODERN NETWORKS

Develop a framework that:

- Offers **high-level programming abstractions**
 - *Ad hoc* abstractions for network programming low-level aspects
 - Implement complex analysis queries in a simplified way
- Implements **fast and scalable processing**
 - Real-time continuous monitoring applications approached as **stream analytics problems**
 - Use Data Stream Processing to improve **scalability and performance**
 - Support complex real-time analysis
 - Support processing over huge amount of data

BACKGROUND: DATA STREAM PROCESSING (DASP)

- Need for continuous and real-time analysis of streaming data
 - Networking, smart cities, smart mobility, smart logistics, ...
- DaSP frameworks
 - Simplify the implementation of efficient streaming computations
 - Offer abstractions for implementing general-purpose applications
- Adopted model: **data-flow graph**
 - Stream pipelines of core functionalities (*operators* or *transformations*)



BACKGROUND: DATA STREAM PROCESSING FRAMEWORKS

Characterized by

- Exposed API
 - Level of abstraction (hide complexities of **stream handling** and **parallelism**)
- Runtime and targeted systems
 - Big Data Stream Processing frameworks
 - Java-based, target distributed systems
 - Other solutions
 - C++17-based, target single multicore machines with GPU support



STATE-OF-THE-ART SOLUTIONS

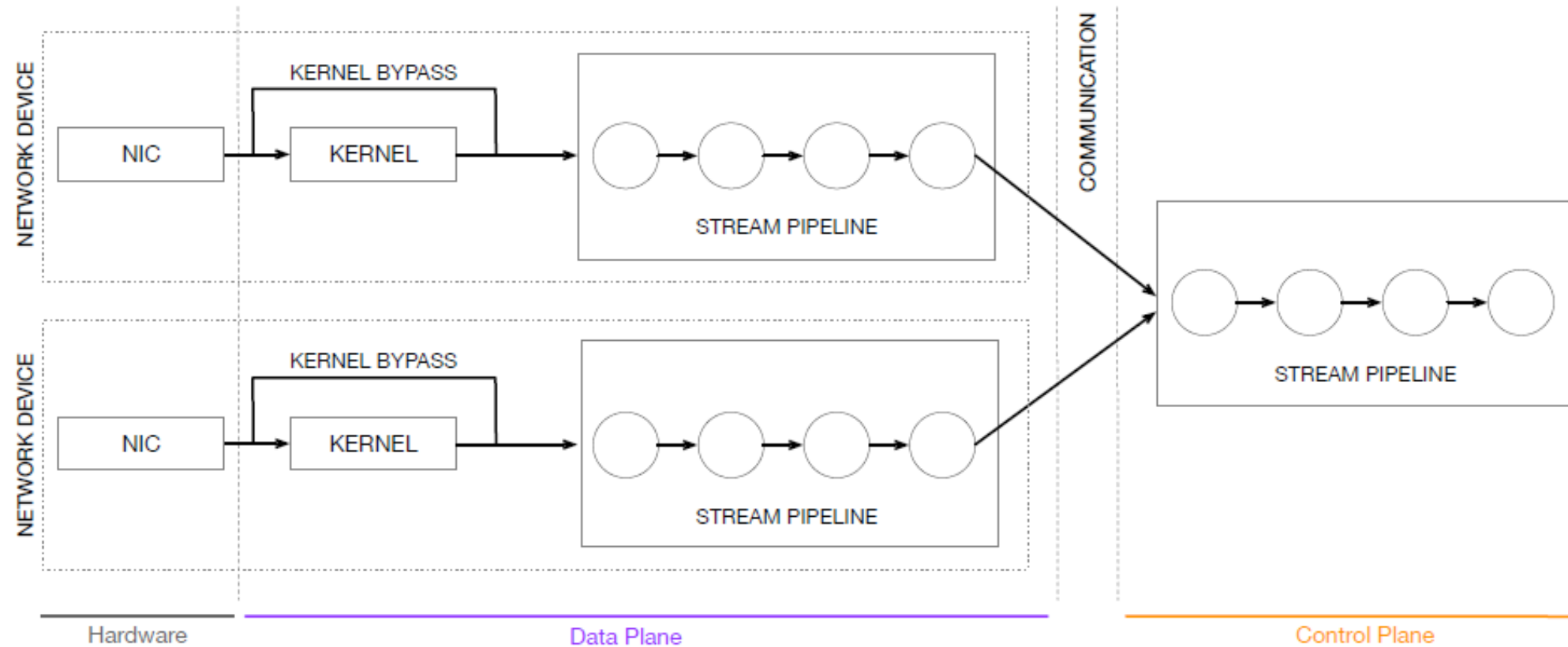
TRADEOFF:

- A. Implement very complex queries with proper abstractions at a lower performance
- B. Trade expressiveness to achieve high performance and scalability

Two main classes of solutions:

- Class A
 - Solutions relying on stream processing only
- Class B
 - Programmable switches alone
- Some new proposal tries to unify **expressiveness** and **performance**
 - Programmable switches + stream processing solutions

PROPOSED FRAMEWORK: HIGH-LEVEL VIEW OF THE ARCHITECTURE



RESEARCH CHALLENGES (1/3)

- Select the most suitable DaSP frameworks to implement stream pipelines

OBSERVATION: pipelines at data and control planes handle different data rates


Put this in relation with the achievable performance for the available DaSP frameworks

- **DASP FRAMEWORK FOR DATA PLANE STREAM PIPELINE:** WindFlow or similar
 - Offer high performance to sustain high data rates
- **DASP FRAMEWORK FOR CONTROL PLANE STREAM PIPELINE:** any DaSP framework
 - Java-based frameworks are enough for lower rates + support for pipeline distribution

RESEARCH CHALLENGES (2/3)

➤ New abstractions to deal with network related aspects

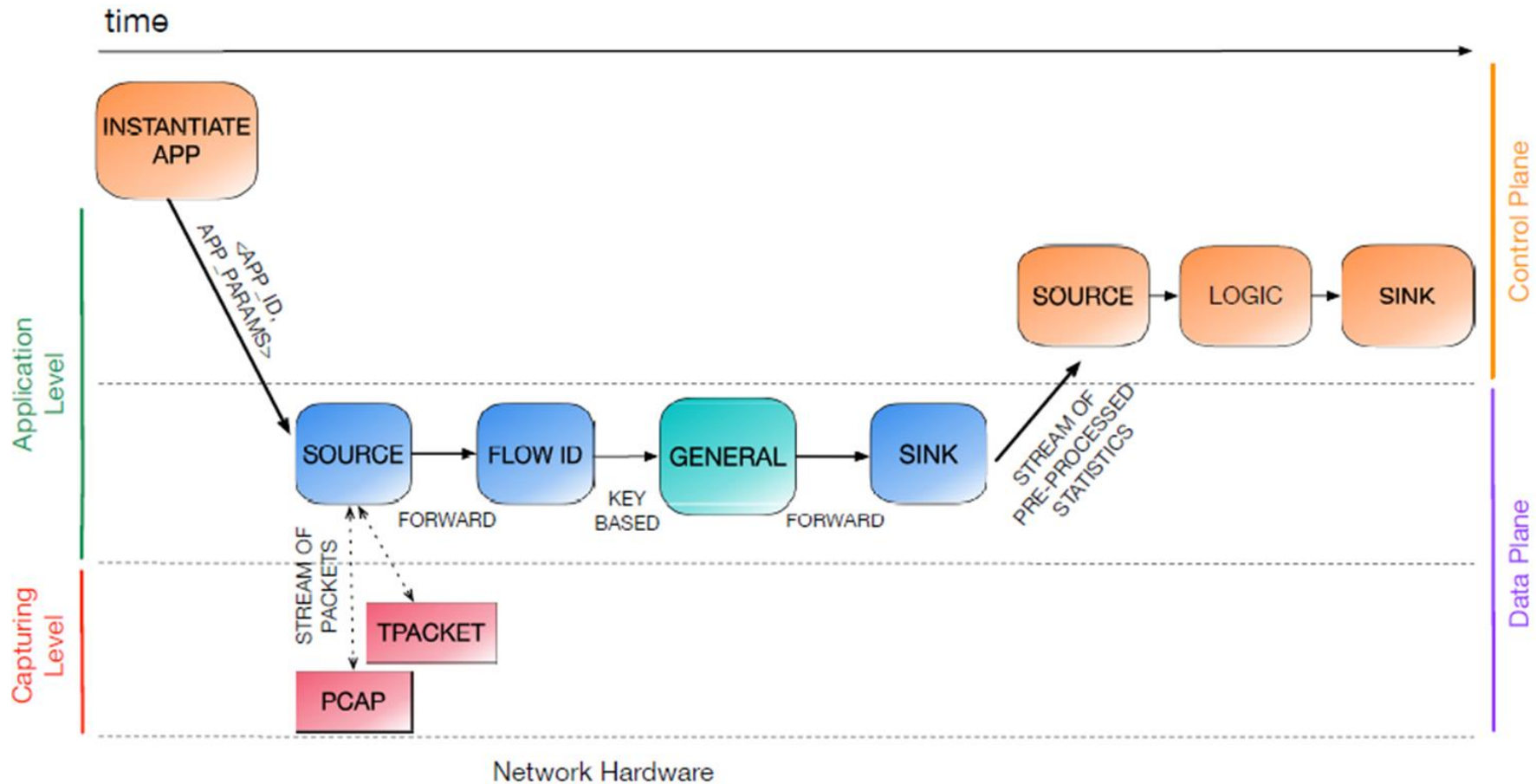
1. SOURCE operators for feeding data plane stream pipelines with packets from NICs

- Hide complexities related to usage of high-performance packet capturing engines
 - New SOURCE entities integrated in WindFlow
 - Capture packets
 - Translate packet data-type to WindFlow *tuple* format
-  ☒ LIBPCAP
☒ TPACKET (vers. 3)
☐ XDP/eBPF
☐ ...

2. Balance parallelism and offloading at both capturing and processing levels

- Avoid contention on the same CPU cores
- Maximize performance by finding optimal trade-offs

AN INITIAL PROOF-OF-CONCEPT APPLICATION



RESEARCH CHALLENGES (3/3)

- Design the communication between data and control planes

OBSERVATION: minimize the amount of data exchanged and the communication delay

- **DATA PLANE STREAM PIPELINES:** produce aggregated statistics on the processed packets
 - Small amount of short tuples to send
- **CONTROL PLANE STREAM PIPELINES:** possibly join several streams of statistics
 - Minimize distance between all the communicating entities
 - New strategies for optimal deployment of stream pipelines onto physical nodes

FUTURE WORK...

CONCLUSION

A new framework to implement fast and scalable applications for **real-time continuous network monitoring and data analysis**.

- High-level interface
 - ❖ Simplify the implementation
 - ❖ Encourage network programmers to use the proposed solution
- Two-layer structure based on stream pipelines at both data and control planes
 - DaSP frameworks to simplify pipeline implementation guaranteeing required performance
 - Run on general-purpose nodes (multicores + co-processors) with optimized resource usage
 - ❖ Accommodate higher number of processing applications on the same infrastructure
 - ❖ Promptly detect changed network conditions
 - ❖ React to guarantee QoS constraints