Technische Universität Dresden

6G-life

BoostState: Low-Latency State Transfer in Edge Cloud Using Programmable Data Planes

Tung V. Doan , Mahdi Attawna, Frank H.P. Fitzek, Giang T. Nguyen
Technische Universität Dresden, Germany

The ACM/IRTF Applied Networking Research Workshop 2026 (ANRW'26)
Vienna, July 20, 2026

Motivation
The Growing Importance of Stateful Network Functions

- Network functions (NFs), as plentiful as routers and switches, recently gained tremendous interest in edge clouds

- Real-world NFs inherently require **stateful processing**: Rely on **states** (i.e., historical processing values) to handle packets


Firewall


Traffic monitor


NAT

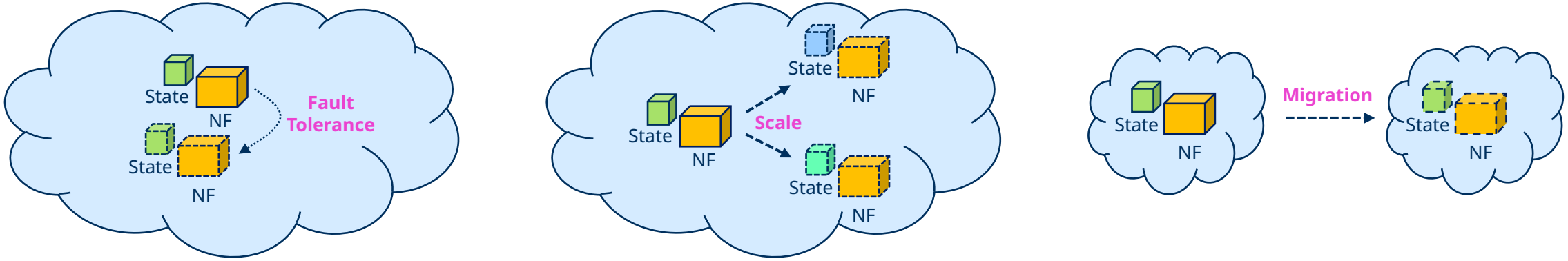
- Emerging workloads (e.g., AI) further accelerate the shift toward stateful processing

- Early attention from the standardization community

IETF92 - IRTF Open Meeting and ANRP Awards
Source: https://www.youtube.com/watch?v=4rhvmTWStRk

- **Stateful NFs,** as reported, utilize **60%** memory and contribute to **43%** of high-severity incidents in the network

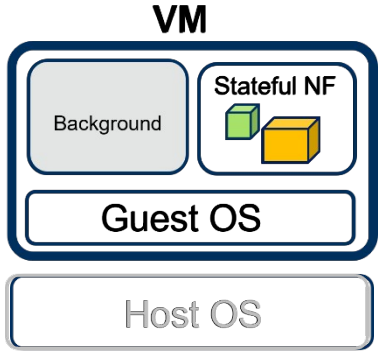
Motivation
Managing Stateful Network Functions



Managing stateful NFs requires **state transfer** to ensure **state consistency**

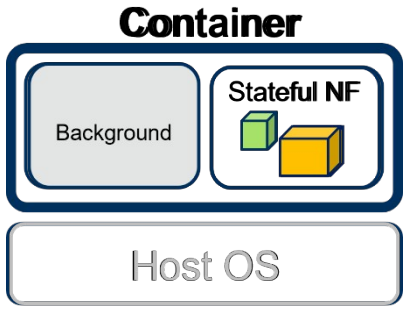
Motivation
Managing Stateful Network Functions

Managing stateful NFs requires **state transfer** to ensure **state consistency**



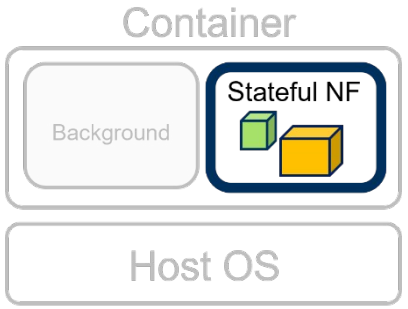
Transfer VM state

(e.g., **tens of seconds** [1])



Transfer container state

(e.g., **few seconds** [1])



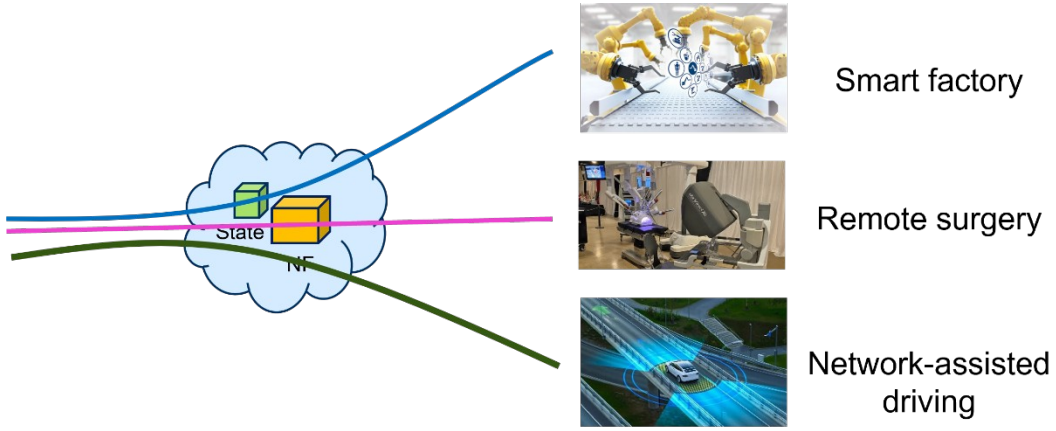
Transfer NF state

(e.g., **milliseconds** [1])

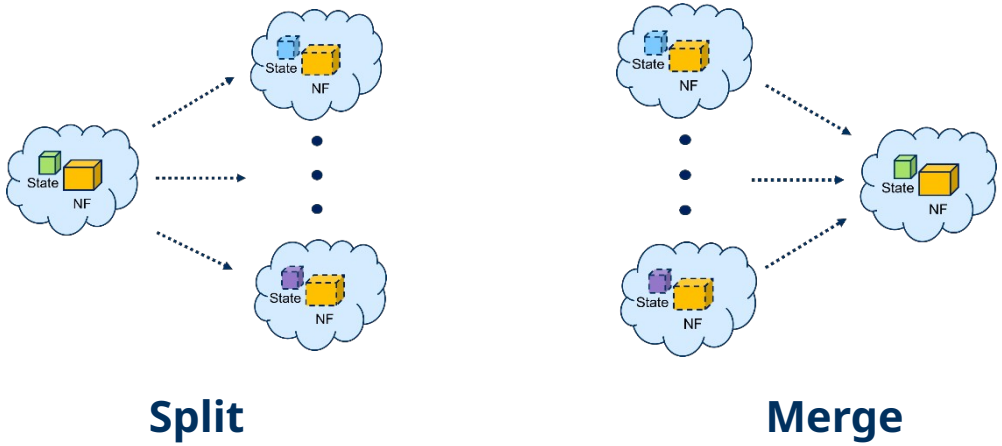
But… this comes with a new challenge

[1] Source: https://www.bell-labs.com/institute/publications/itd-19-59189l/#gref

Motivation
Stateful Network Functions in Edge Clouds



Stateful NFs supporting emerging use cases with **ultra-low latency**



Stateful NFs demand much **more complex state transfer**

State transfer becomes more challenging in edge clouds

BoostState: Design and Implementation
Design Requirements

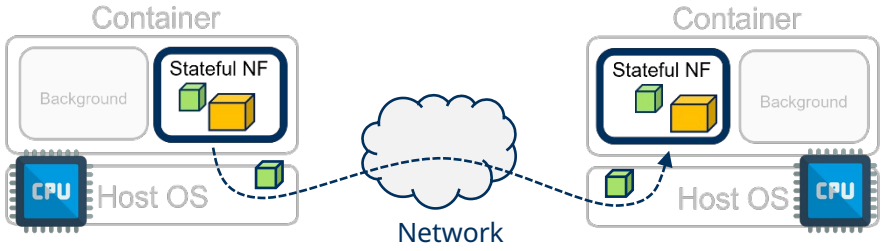
R1 **Low-latency** state transfer

R2 **Preserve** conventional deployment of stateful NFs in cloud environments

R3 **Low overhead** to stateful NFs

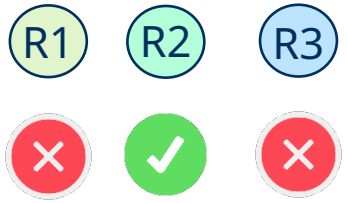
BoostState: Design and Implementation
Idea and Design Principles

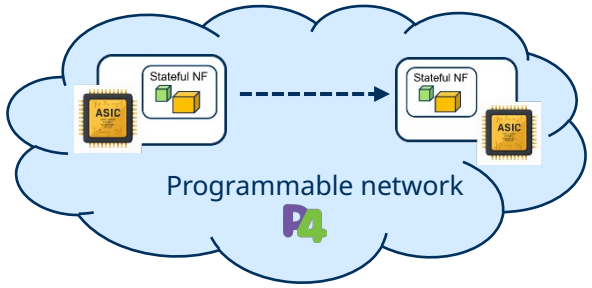
- **Assumption**: State transfer between stateful NFs is inherently performed over **the network**
- **Idea**: leverage programmable data planes to accelerate state transfer



Conventional state transfer
(**NFs on CPU hosts**)

OpenNF (SIGCOMM'14), FTC (SIGCOMM'20), StateOS (NOMS'24), StateProc (ACCESS'26)



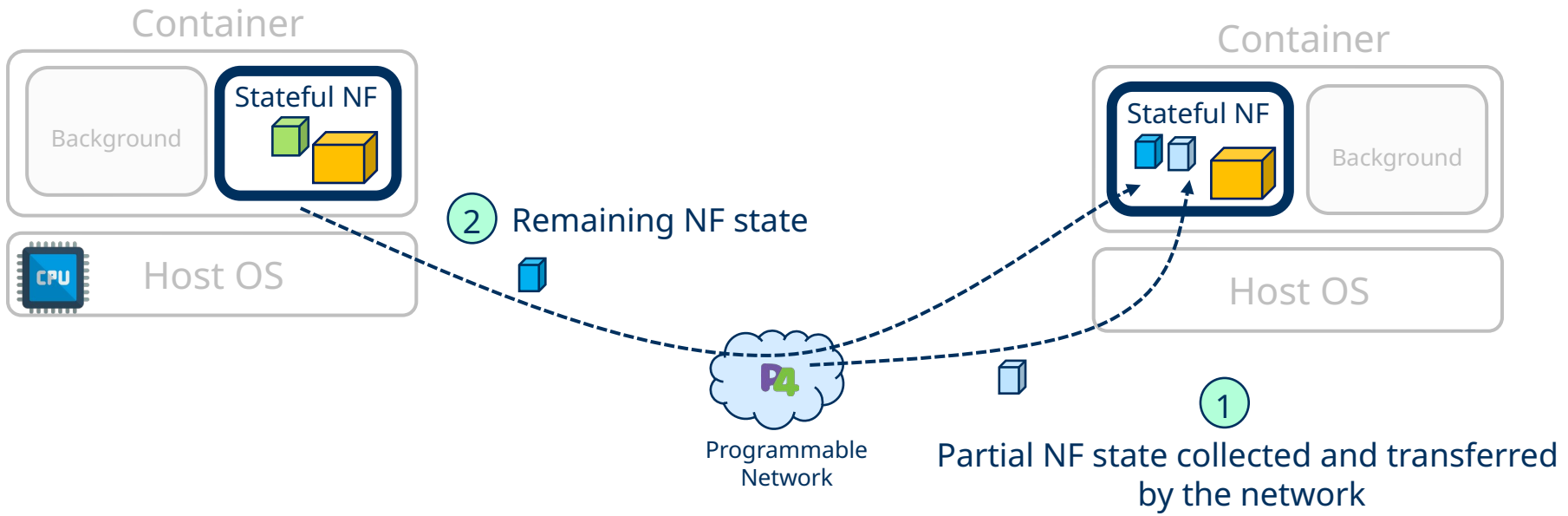


Recent State Transfer
(**NFs on programmable data planes**)

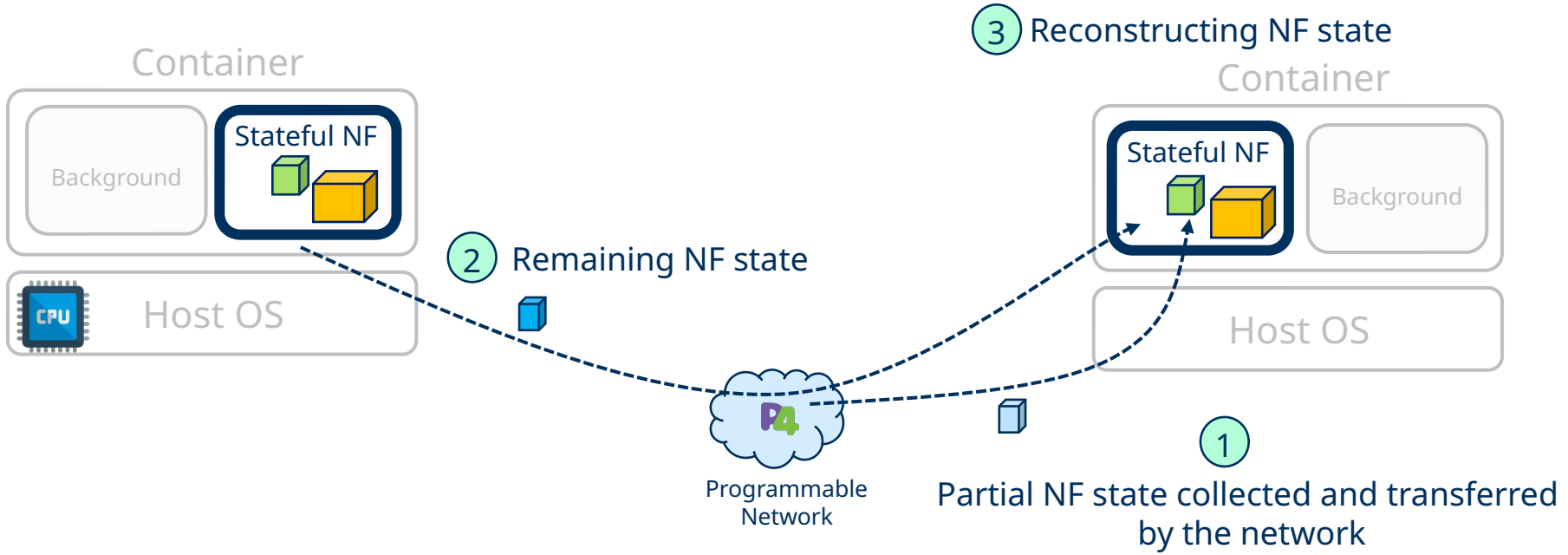
P4NFV (CNSM'18), SwiSh (NSDI'22), StateP4 (NetSoft'23)

BoostState's Idea:
Hardware-Software Codesign

BoostState: Design and Implementation
Overview

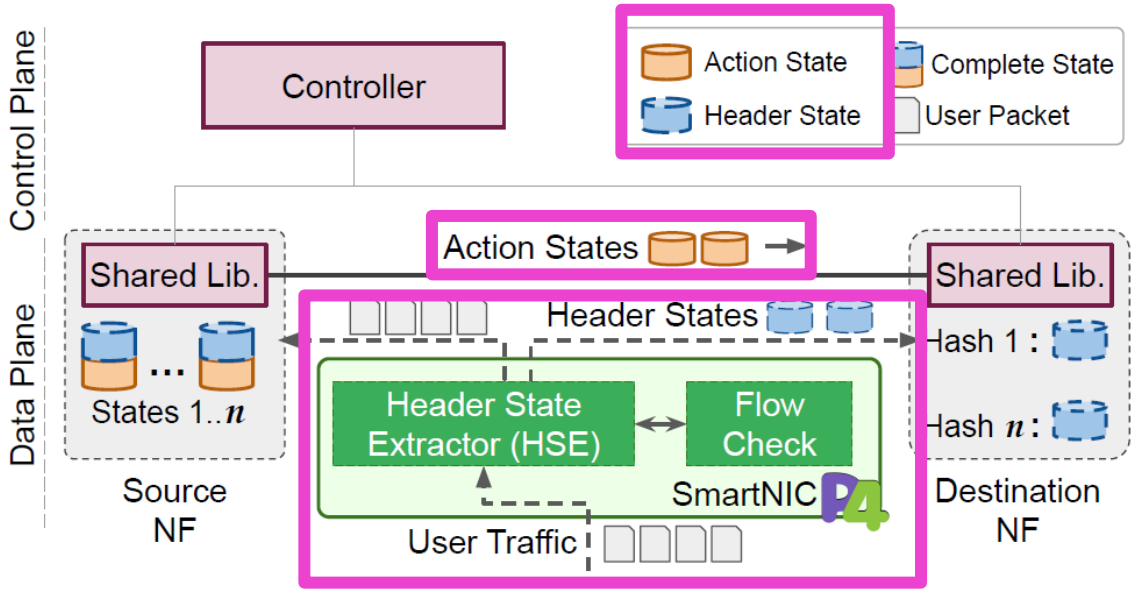


BoostState: Design and Implementation
Overview



BoostState: Design and Implementation
Detailed Design

- *BoostState* partitions NF state into:
  - **Header state**: 5-tuple packet information
  - **Action state**: actions (e.g., drop) taken on a packet
- Programmable data planes (e.g., SmartNIC) **collect and transfer header state** to destination NF
- Source NF **transfers action state** to destination NF, which then reconstructs the NF state



BoostState design

BoostState: Design and Implementation
Implementation

- P4 programs on Netronome SmartNIC to accelerate state transfer
- An x86 server equipped with an Intel Xeon E5-2630 v3 processor operating at 2.40 GHz and 128 GB of RAM
- Portable library to extract and reconstruct NF state
- Enhancements such as non-blocking state reception

Agilio® CX 2x40GbE SmartNIC

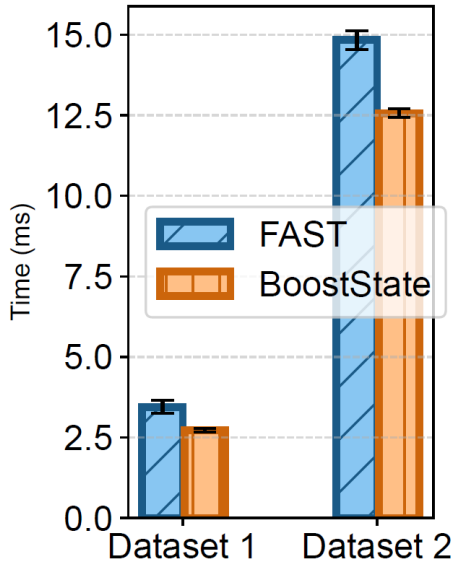
BoostState: Design and Implementation
Evaluation

- Examined scenarios:
  - **Split**: state transfer from one NF to multiple NFs
  - **Merge**: state transfer from multiple NFs to one NF

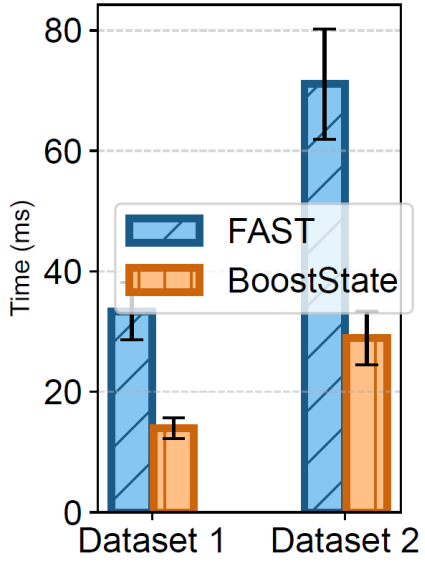
- State operations
  - **Scaling**: state transfer triggered by traffic growth
  - **Fault tolerance**: periodic state replication to maintain a synchronized backup NF

- State transfer latency includes:
  - **Serialization**: encoding state at the source NF
  - **Transmission**: transferring state over the network
  - **Deserialization**: decoding state at the destination NF.

Performance Evaluation
Merge Scenario: Scaling Operation

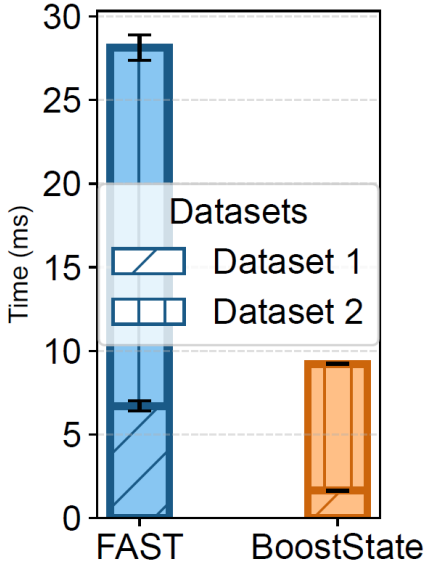


Serialization



Transmission

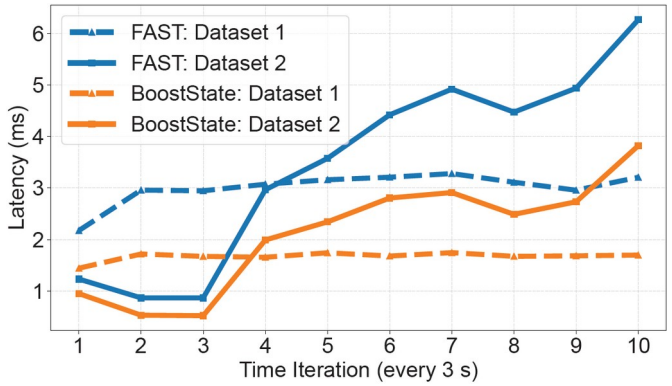
60% reduction



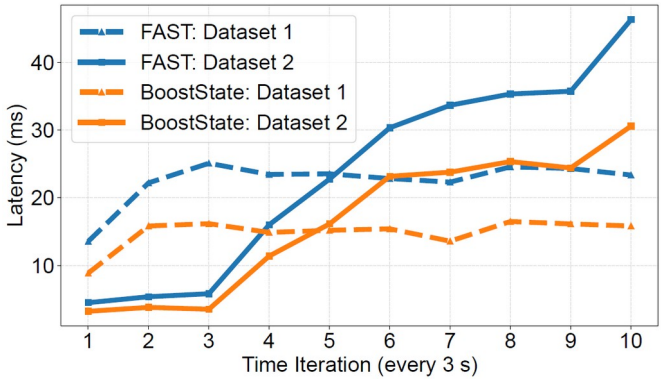
Deserialization

BoostState significantly outperforms the state-of-the-art by leveraging programmable data planes

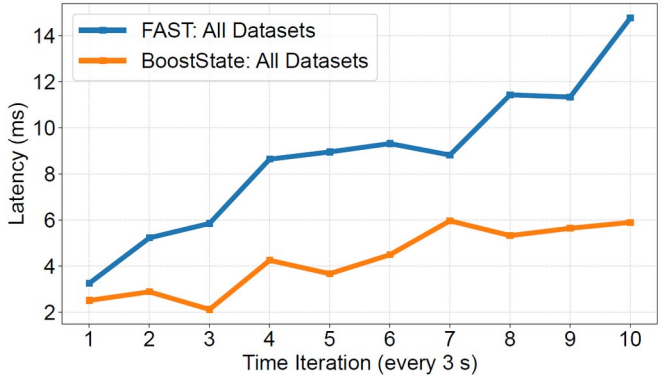
Performance Evaluation
Merge Scenario: Fault Tolerance Operation



Serialization



Transmission



Deserialization

BoostState significantly outperforms the state-of-the-art by leveraging programmable data planes

Conclusion

- We presented *BoostState* that leverages programmable data planes to accelerate state transfer in edge clouds
- *BoostState* preserves conventional cloud-based NF deployments while providing low-latency state transfer
- We implement *BoostState* using P4 language on a Netronome SmartNIC
- Results show that *BoostState* reduces state transfer latency compared to prior work, including up to **60% lower transmission latency**

Thank you for your attention!