Stateless Network Functions: Breaking the Tight Coupling of State and Processing

Murad Kablan, Azzam Alsudais, Eric Keller
University of Colorado

Franck Le
IBM

（NSDI 2017）

Network Functions

Firewall

Load balancer

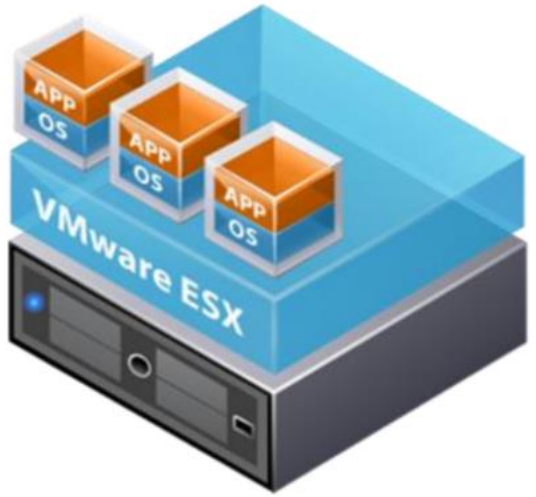
NAT

Intrusion Prevention

To protect and manage the network traffic

NFV(Network Function Virtualization)

Hardware Network Functions



Software Network Functions
(Virtual Machines)

Moving hardware-based network functions into common-used hardware (like commodity servers) running as software in virtual environment such as VMs and Containers.

Goals of NFV

- Instantly deploy the network functions compared with hardware architecture.(Instant Deployment)
- Elastically scale the network functions on demand. (Seamless Scalability)
- Quickly recover from failure.(Failure Resiliency)

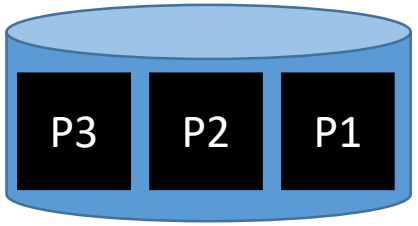
Challenges for stateful VNF

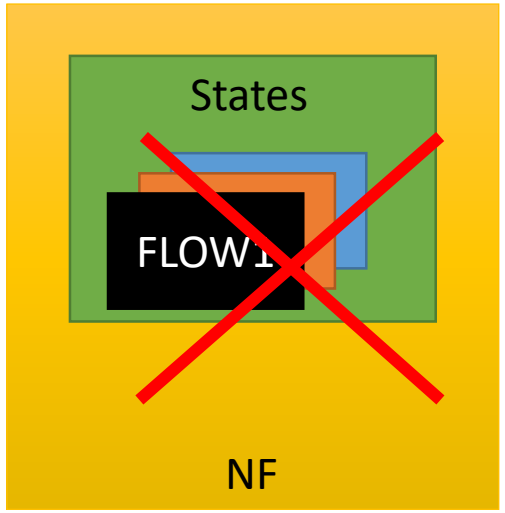
- For stateful virtualized network functions, there is always a tight coupling between their states and processing logic.

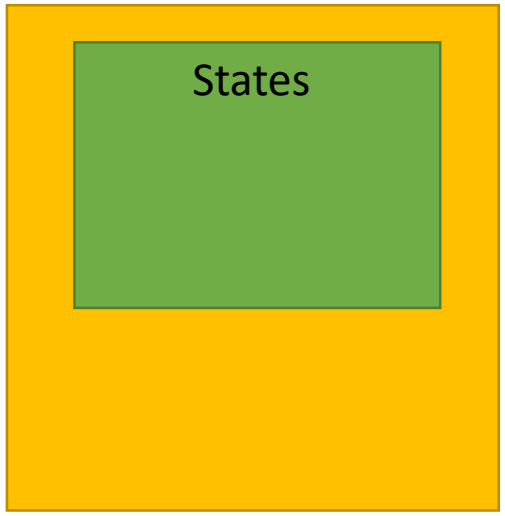
- Firewall: connection tracking information.
- Load balancer: mapping to backend server.

- These states are usually updated for every packet-processing.

Challenge1: Failure

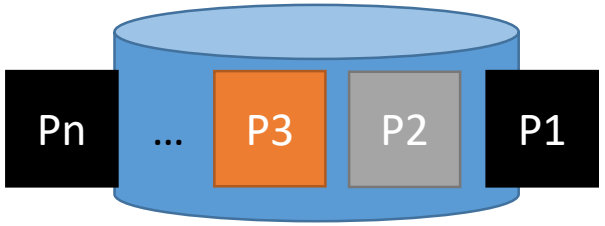


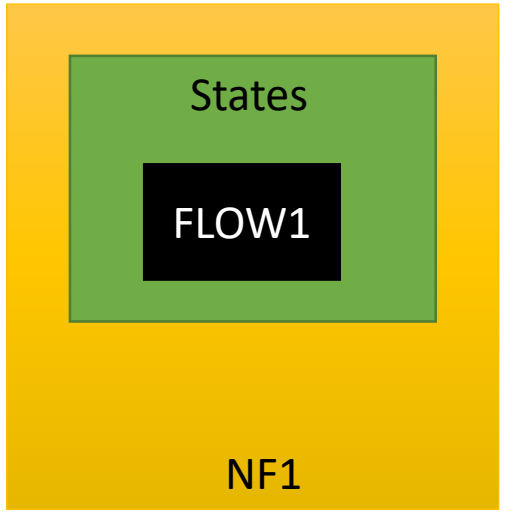


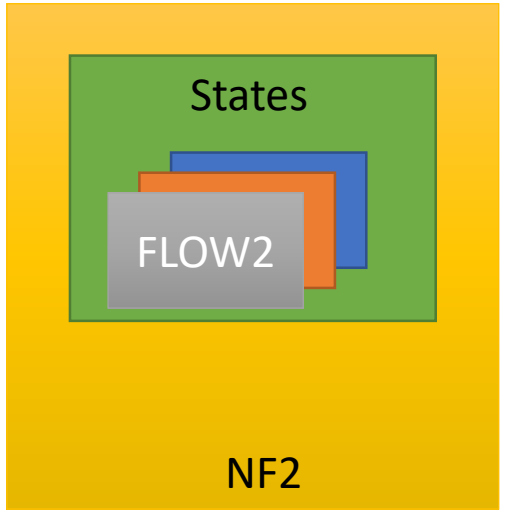


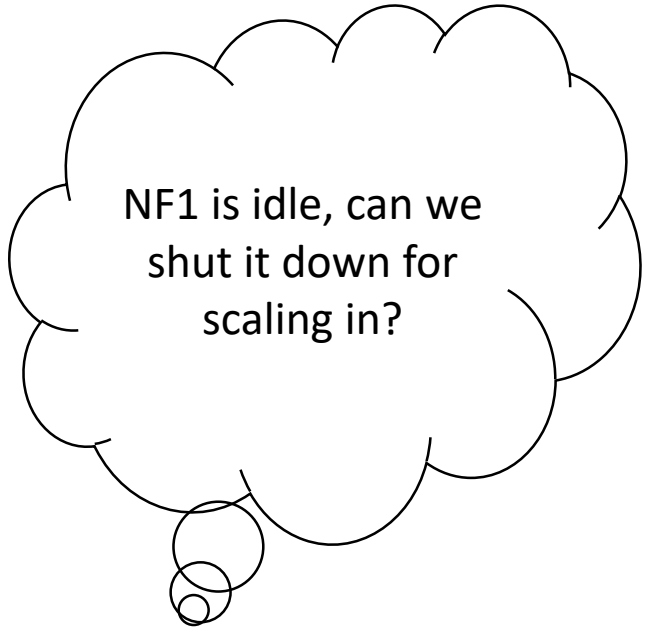
Can’t find the flow

Challenge2: Scale in



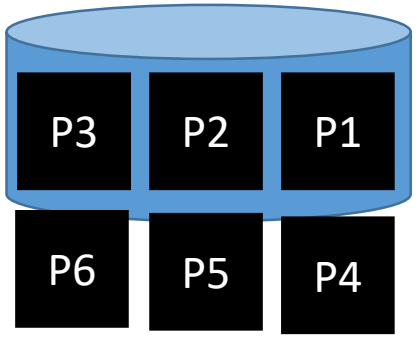


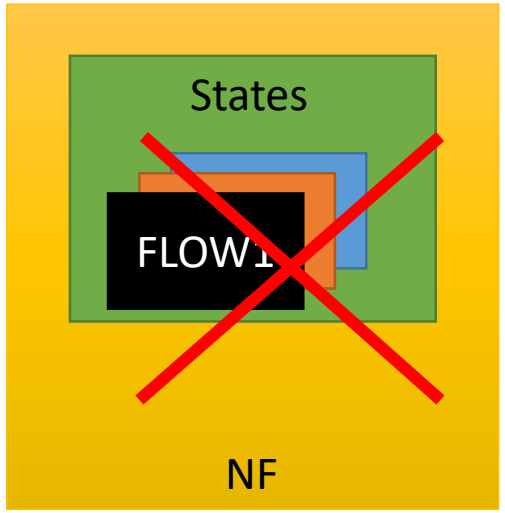


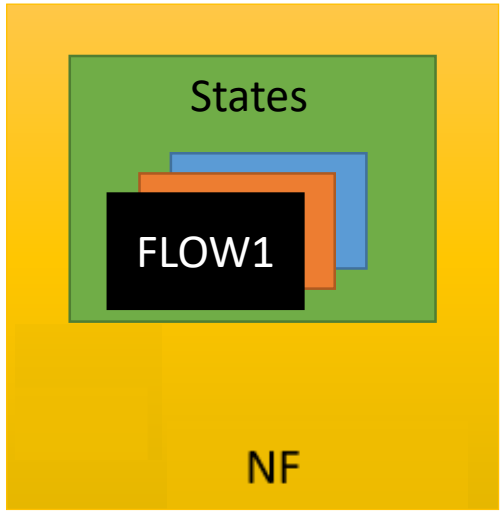


Current solutions

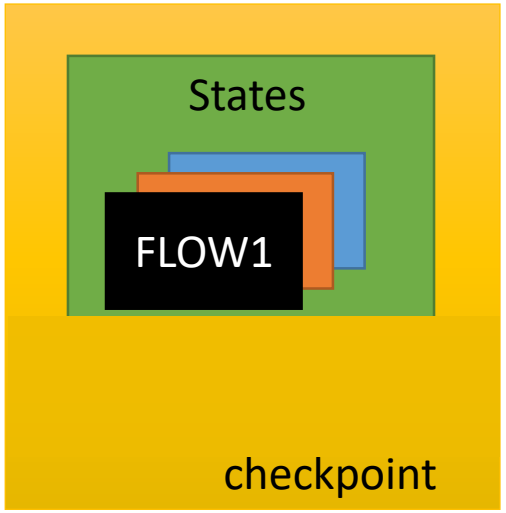
- Checkpoint for failure





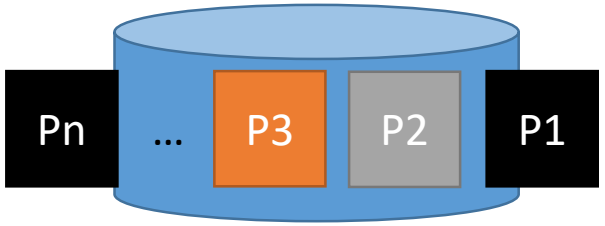


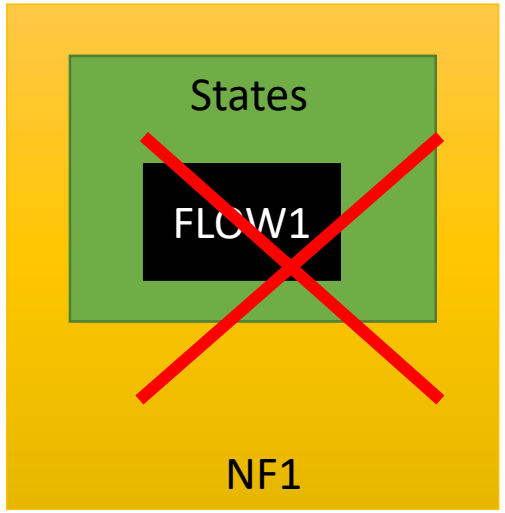
checkpoint

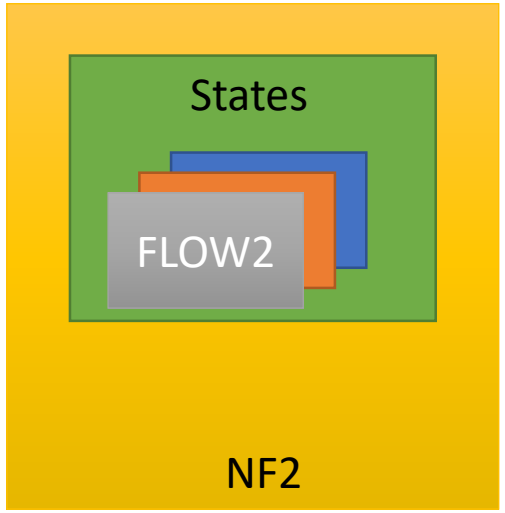


Current solutions

- State migration for scaling







Limitation of current solutions

➢Checkpoint

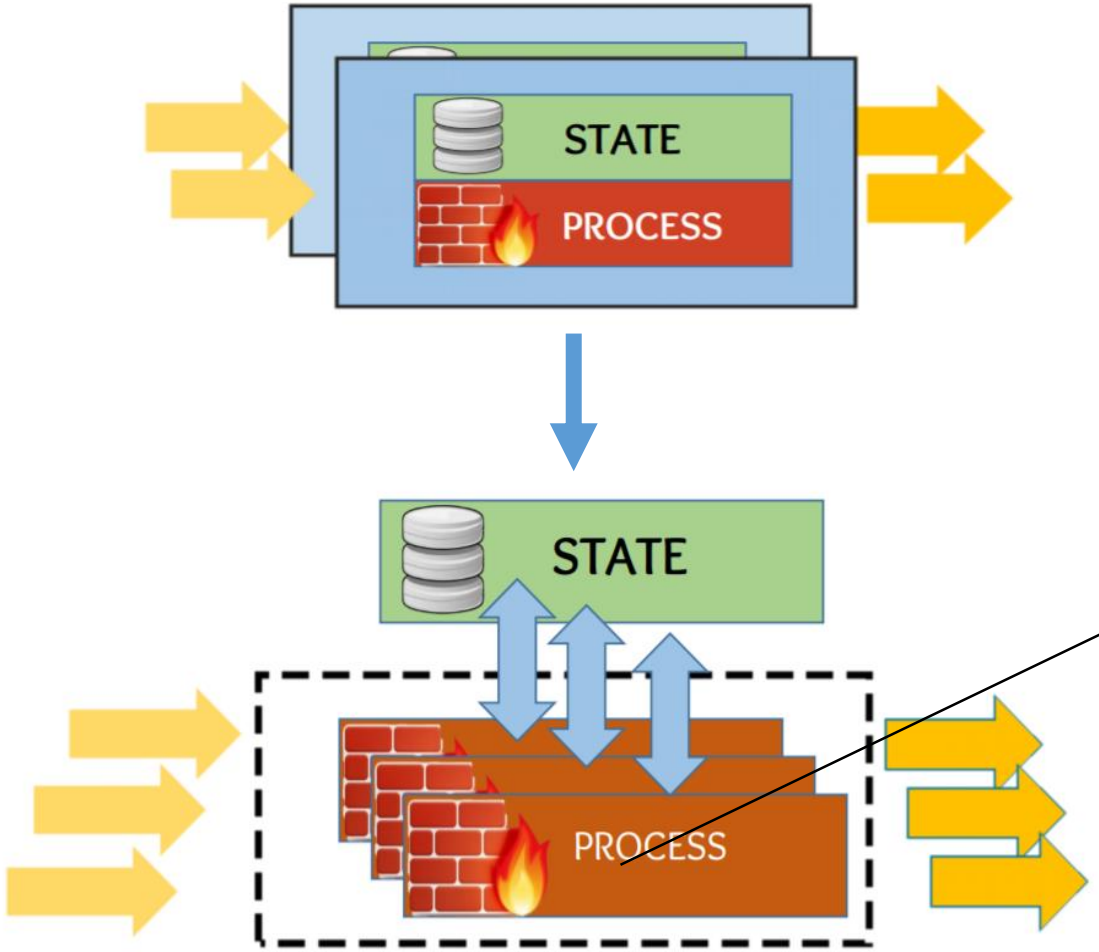
- High packet latency.
- Long recovery time (time since last checkpoint).

➢State migration

- High overhead to migrate state.

Is there a way to achieve both failure resilience and elastic scalability without sacrificing performance?

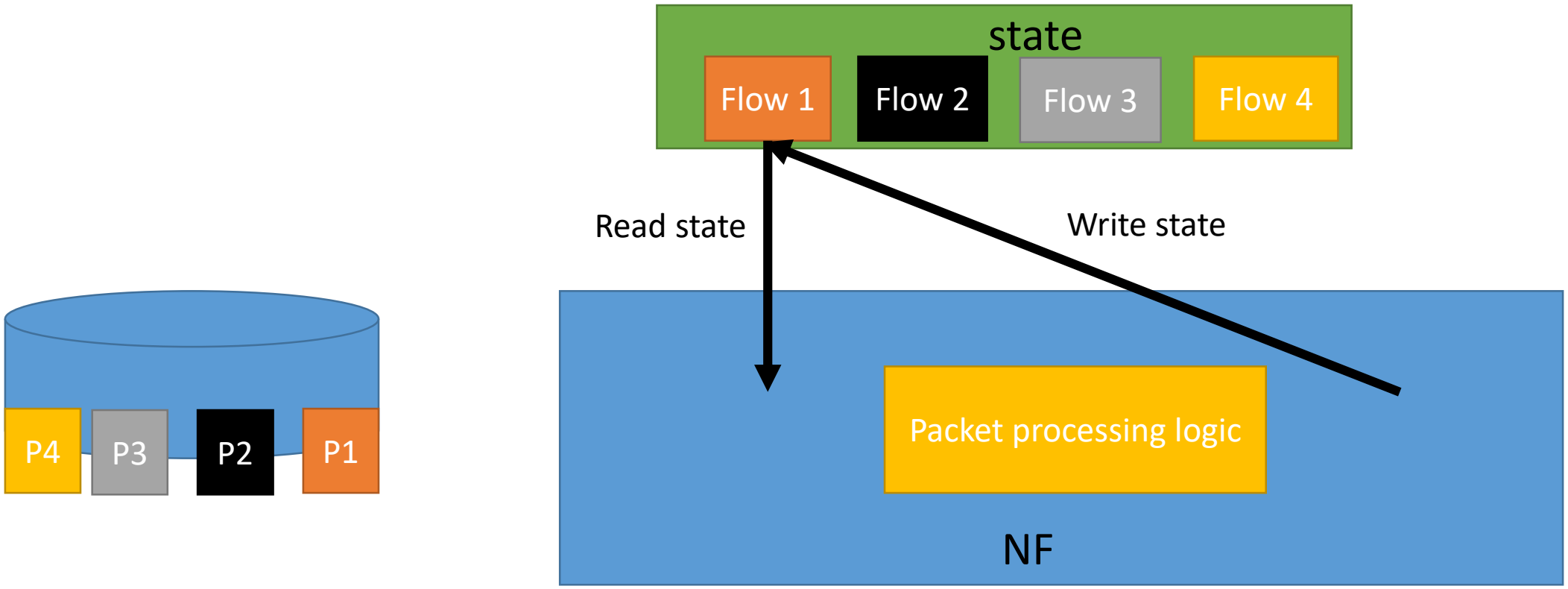
Main idea of the paper



To decouple the state from the processing.

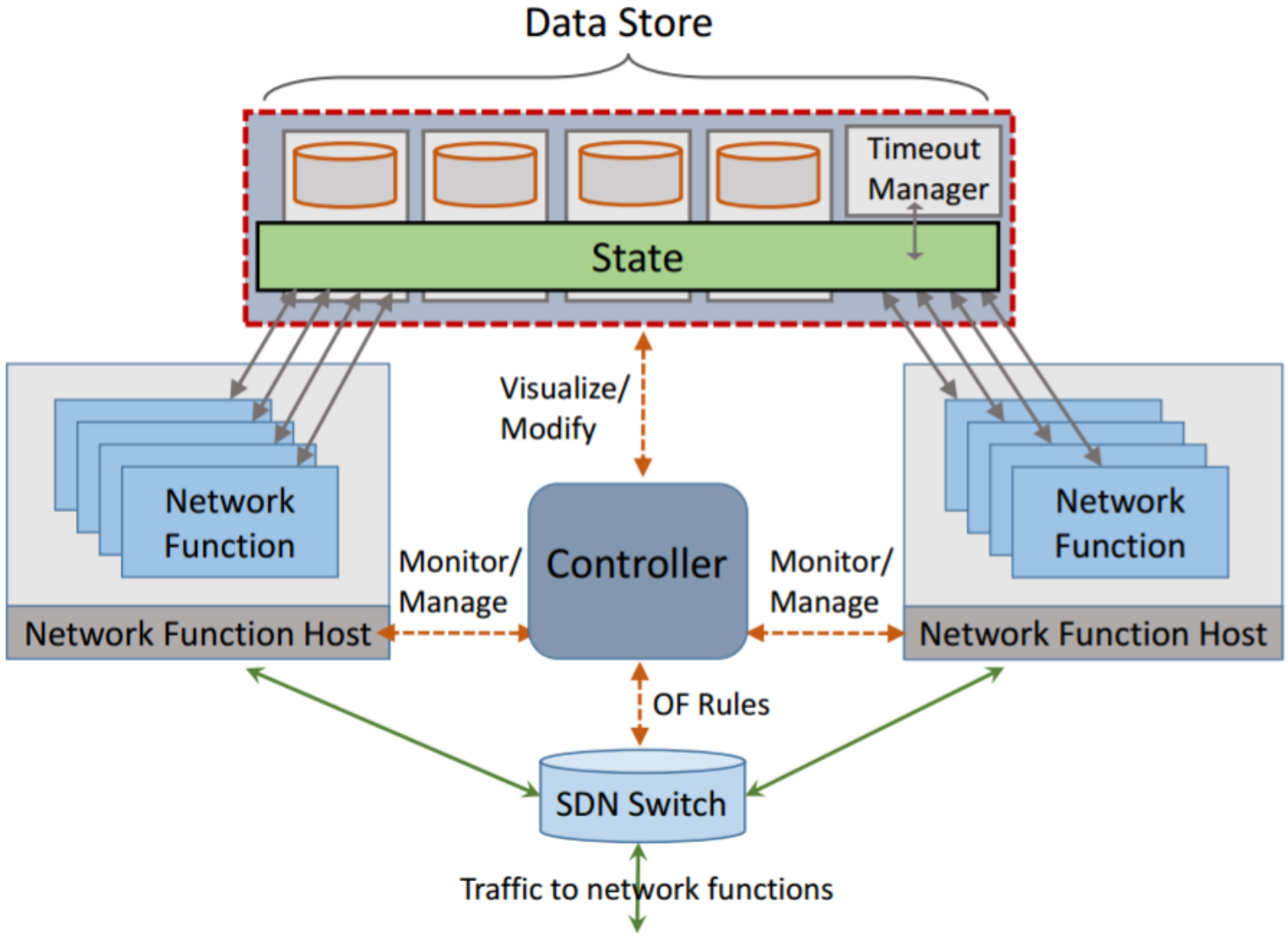
Just like stateless network functions.

Processing steps of “Stateless NF”



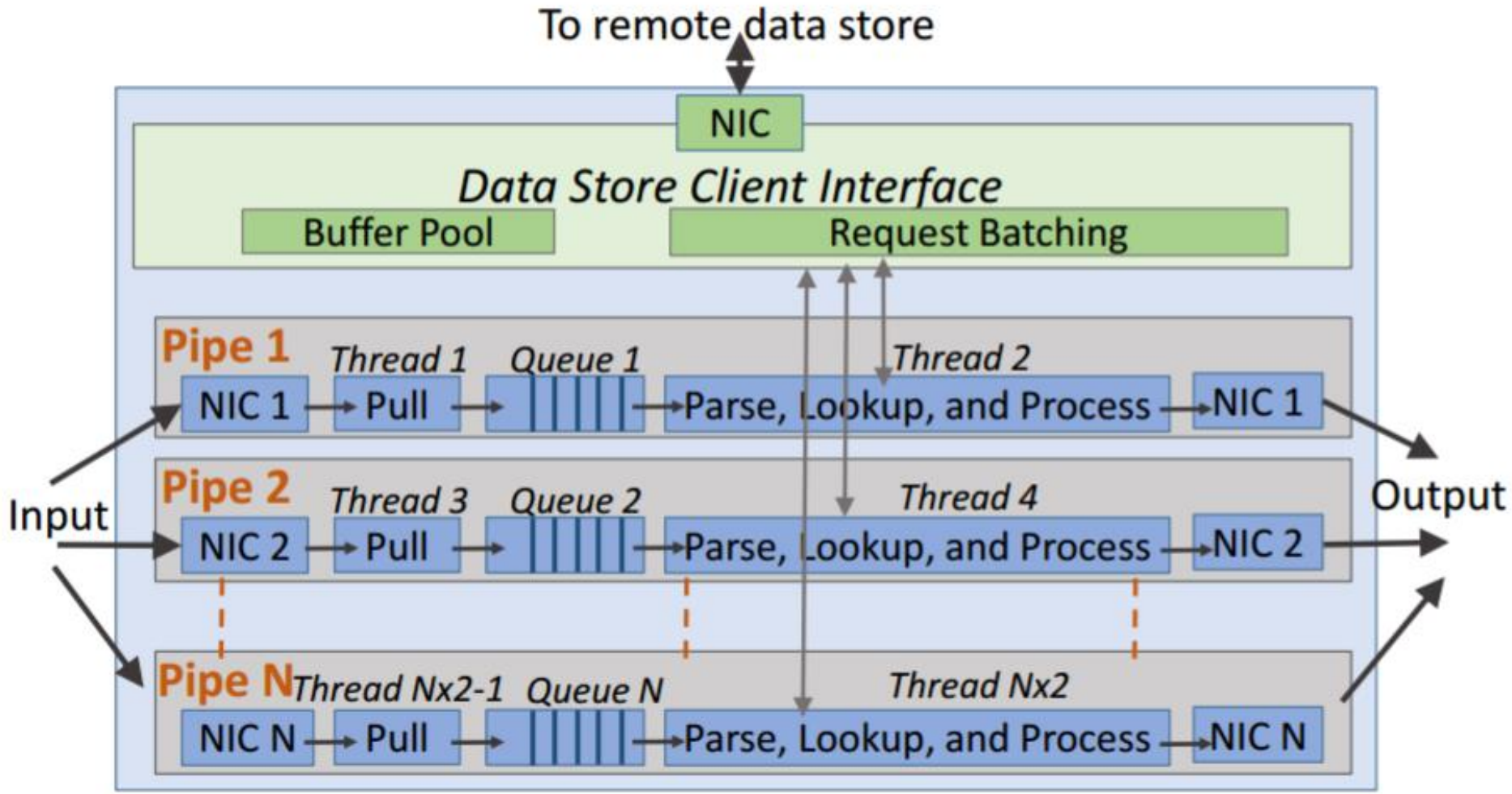
Use hash key to read from and write to remote state data store. The key is generated by 5 tuples of a flow.

Whole architecture of Stateless NF System



Architecture of Stateless NF

- Deployable packet processing pipeline.
- High-performance network I/O.
- Optimized data store client interface.



Evaluation

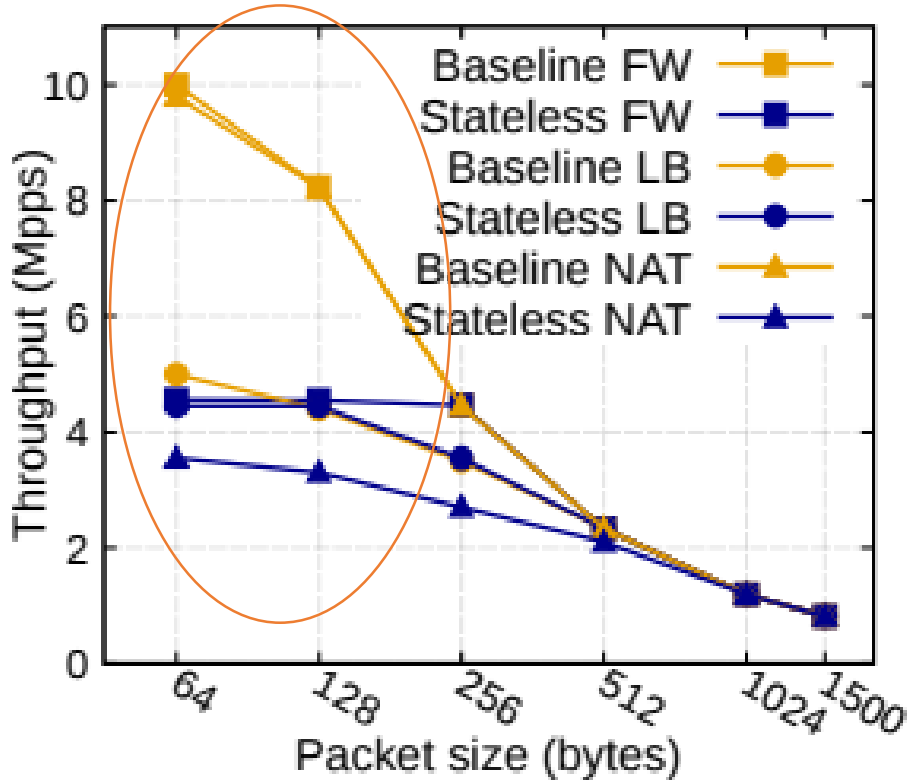
➢Setup

- 6 servers(2 for NF, 2 for RAMCloud, 1 for traffic generation, 1 for controller).
- 2 switches(an Infiniband Switch for connecting NF and RAMCloud, a SDN Switch for traffic).

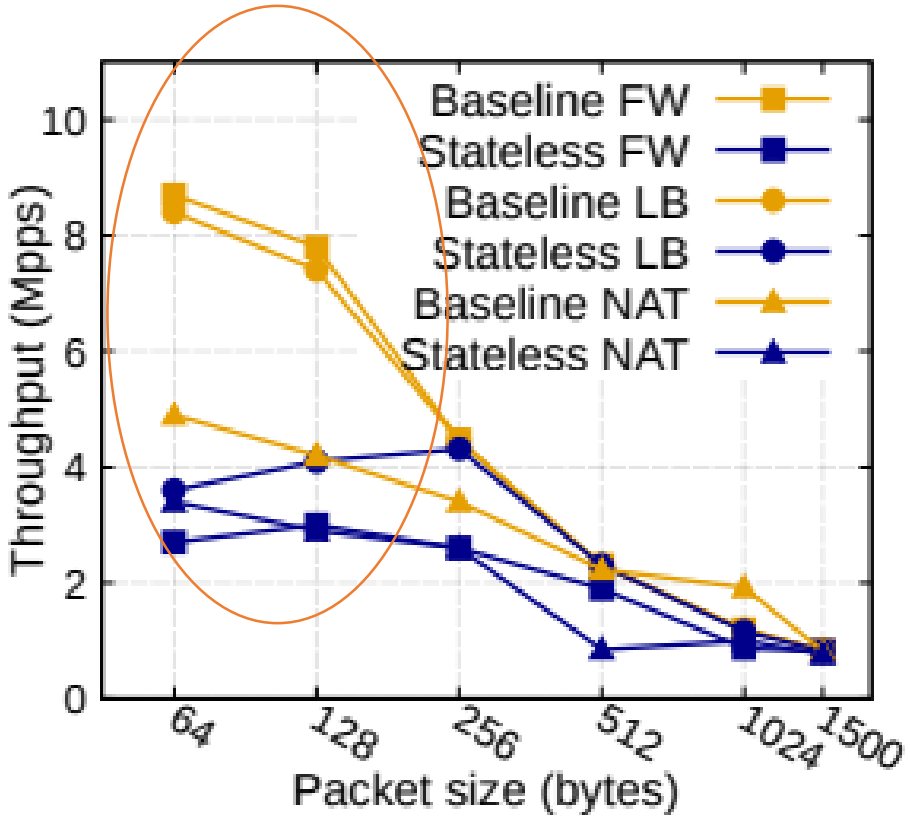
Throughput

A single RAMCloud server is able to handle around 4.7 Million lookups/sec.

Raw packet



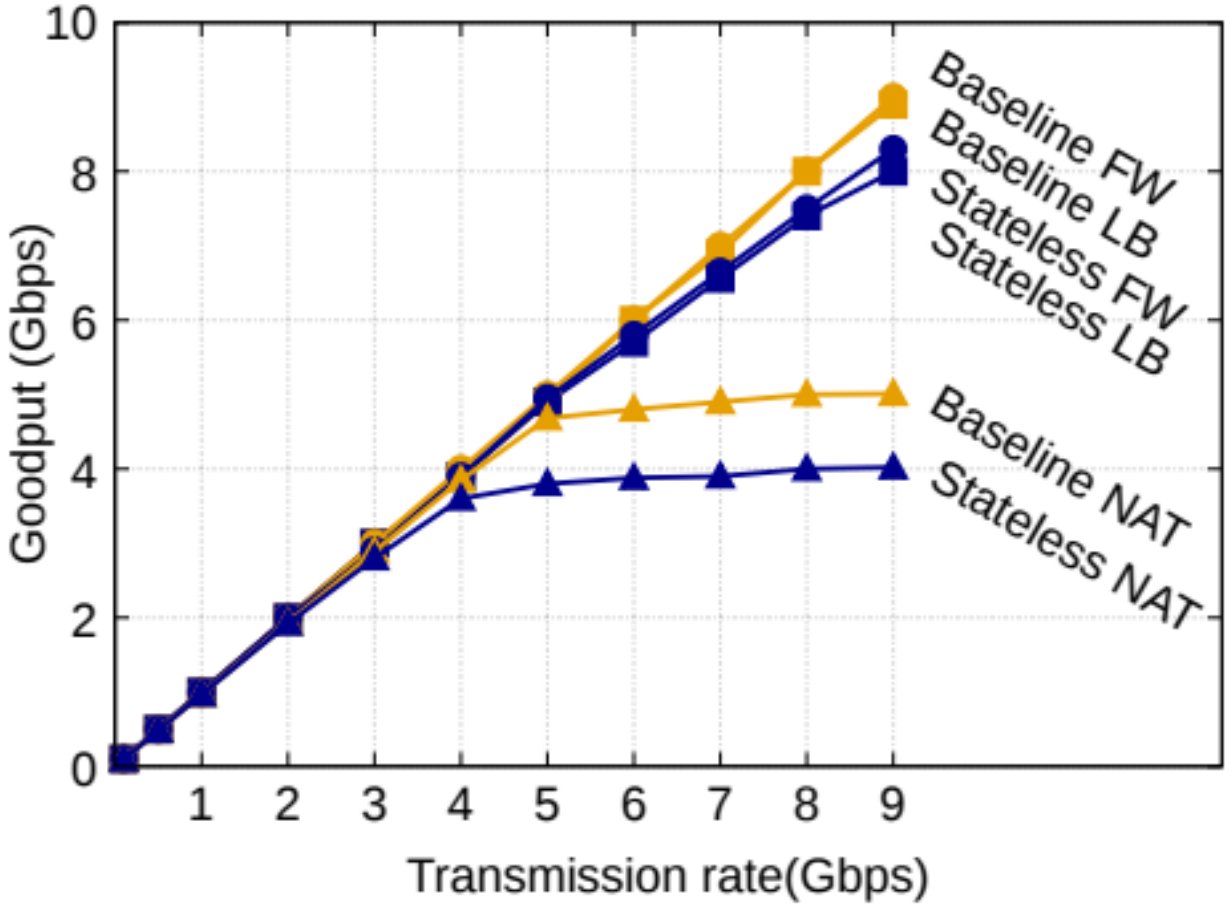
(a) Long flow case



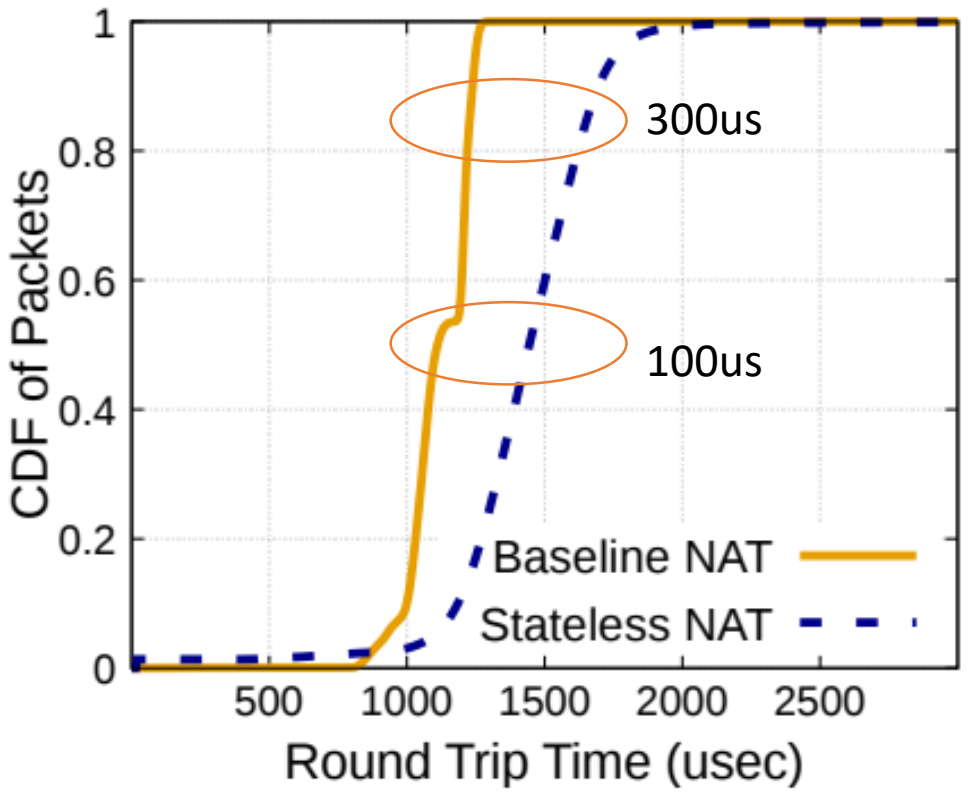
(b) Short flow case

Throughput

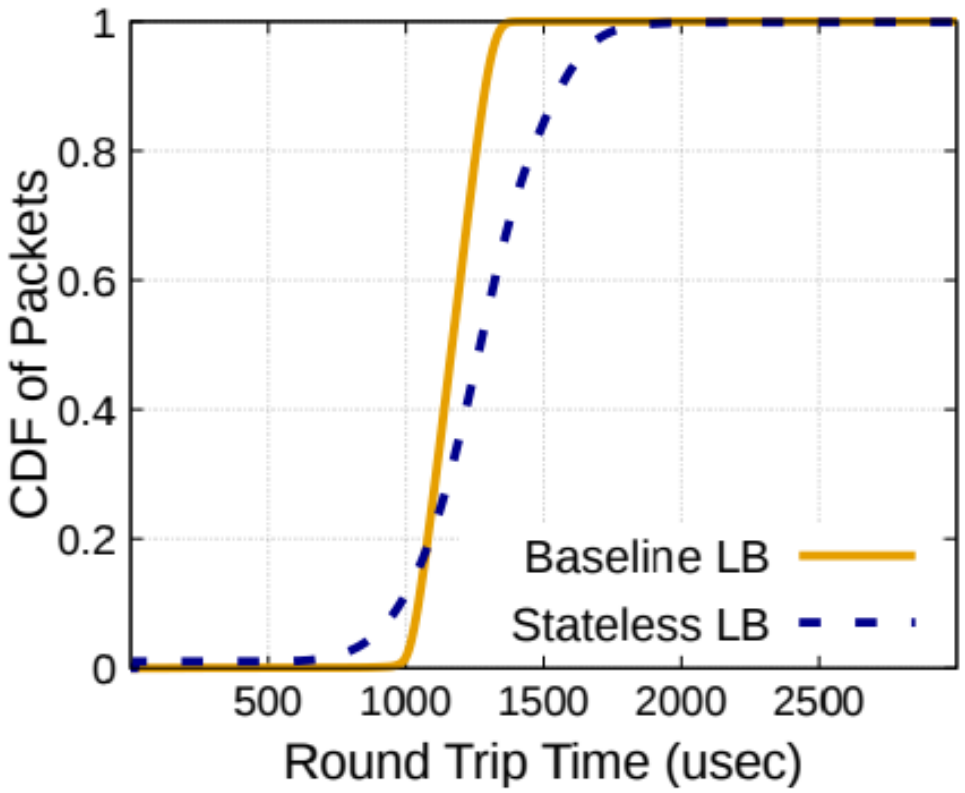
Enterprise trace



Latency

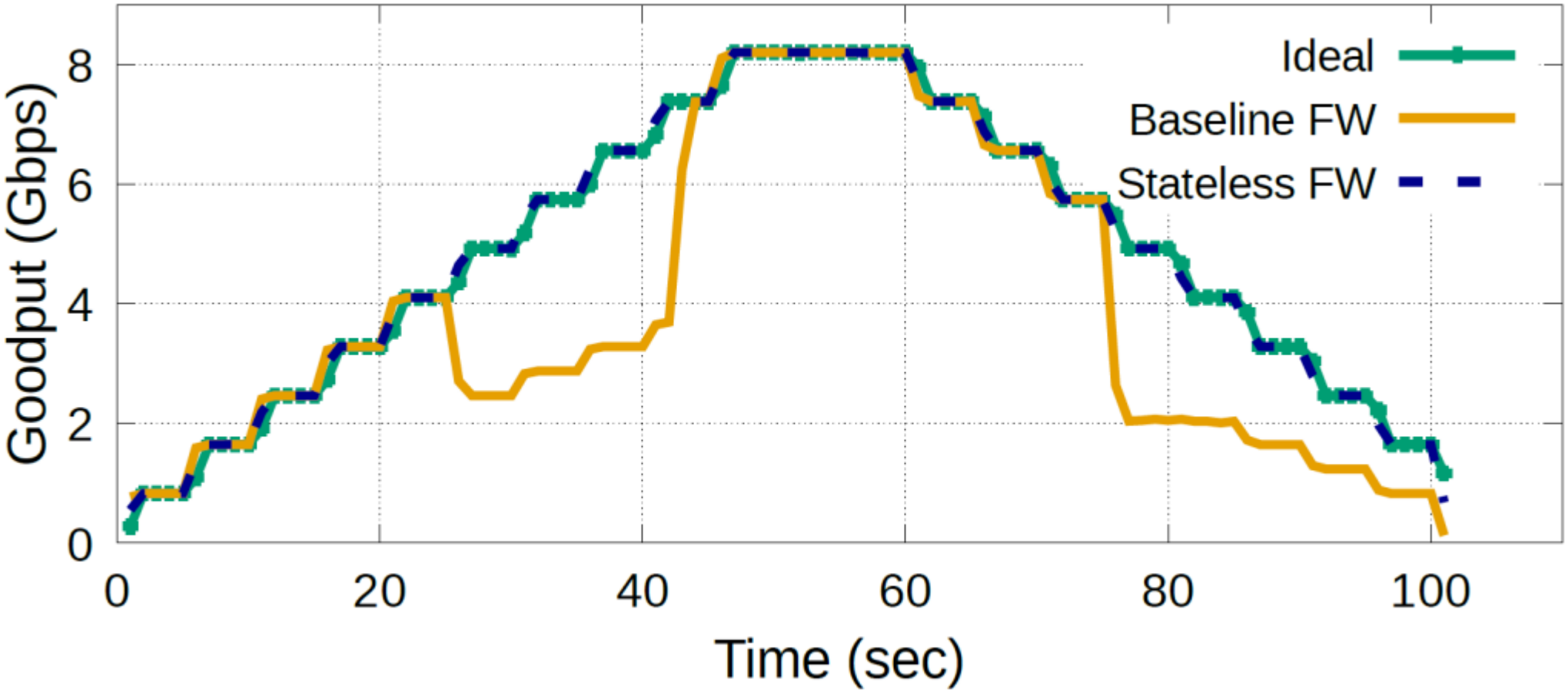


(a) NAT



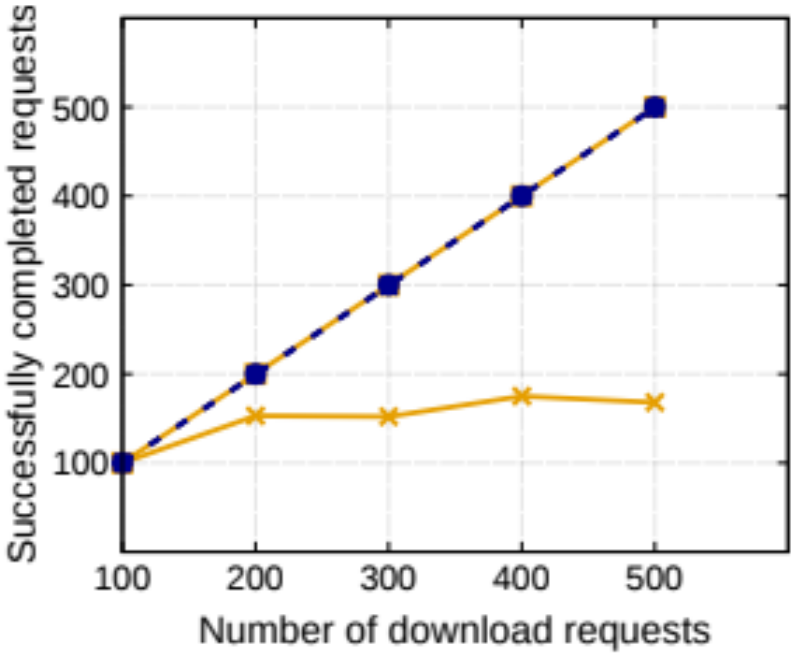
(b) Load Balancer

Scaling in and out

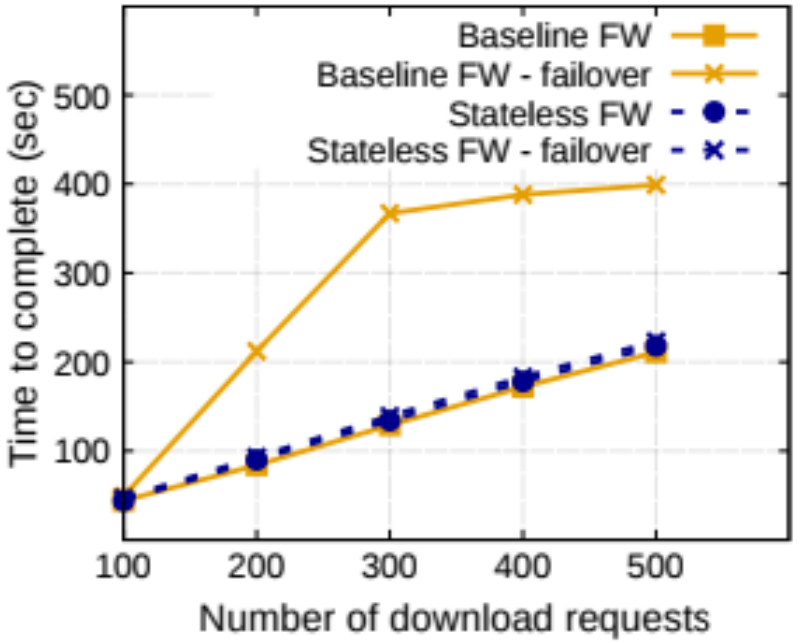


Failure Resilience

To download up to 500 20MB files in a loop of 100 concurrent http downloads through the firewall.



(a) Completed requests



(b) Time to complete requests

Summary

➢Pros

- It decouples the existing design of network functions into a stateless processing component.
- It utilizes DPDK for high performance network I/O.
- It nearly matches the ideal performance when scaling and recovering.

➢Cons

- It doesn’t consider the shared flow state.
- The performance of the system highly depends on the performance of the remote system.