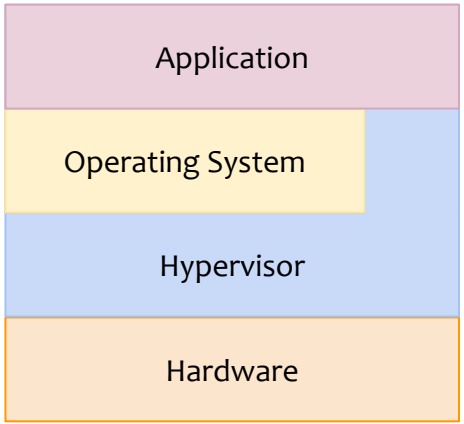
Network Function Virtualization with UniBPF

Paul Zhang
Advisor: Dr. Masanori Misono & Ilya Meignan--Masson
Systems Research Group
https://dse.cit.tum.de/

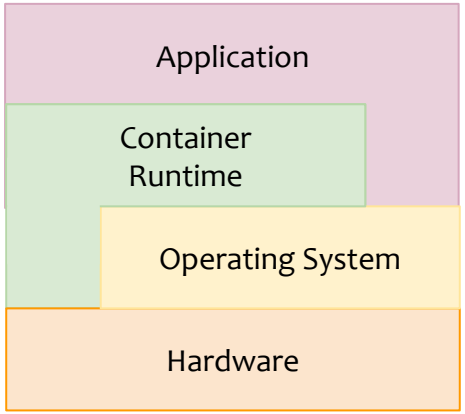
01.06.2024 - 01.10.2024

Unikernels

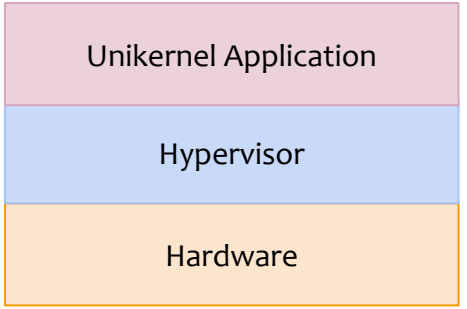
Reference:
- Welcome to Unikraft's Documentation



Traditional VMs
- heavyweight
- very secure



Containers
- lightweight
- least secure



Unikernels
- lightweight
- very secure

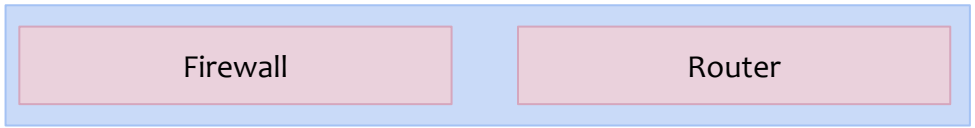
Network Function Virtualization

Reference:

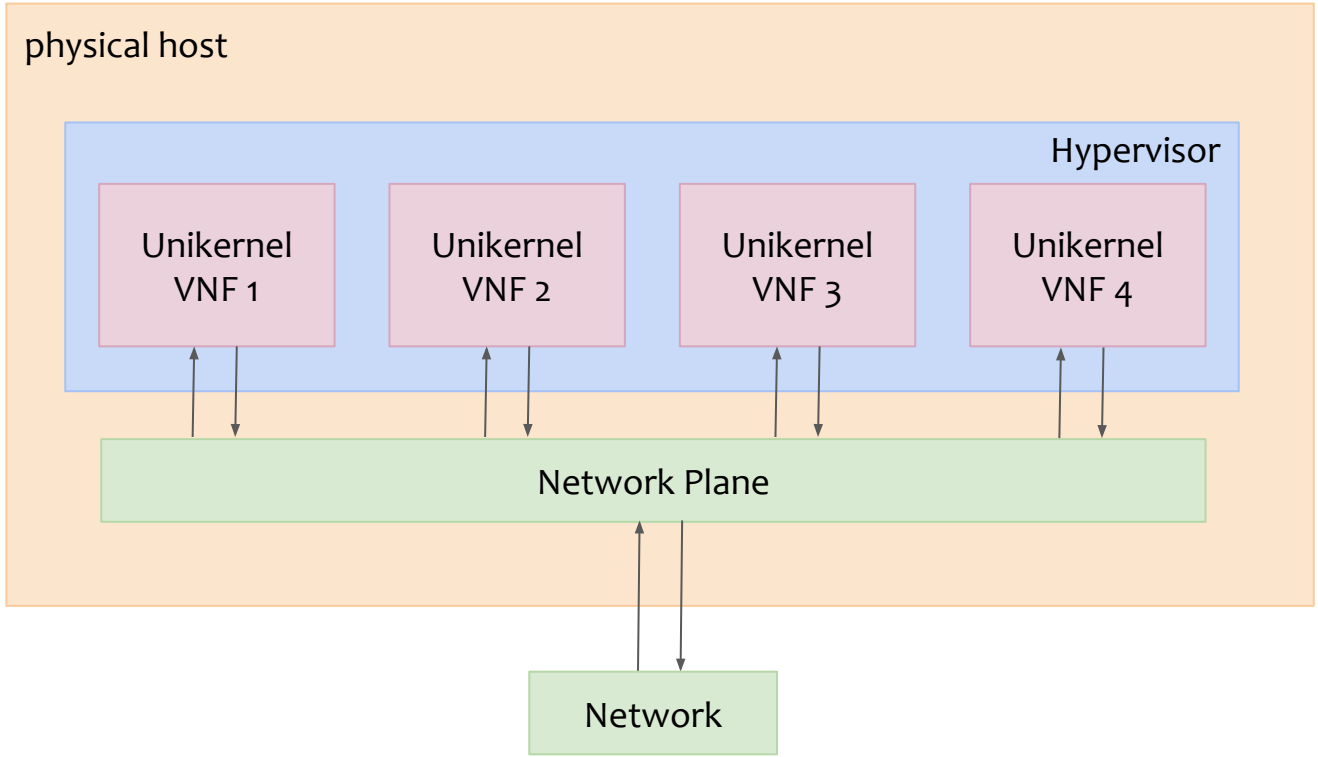
- What is Network Functions Virtualization (NFV)? | VMware Glossary

Traditional Networks: physical middleboxes **per function**

NFV: functions are **virtualized**

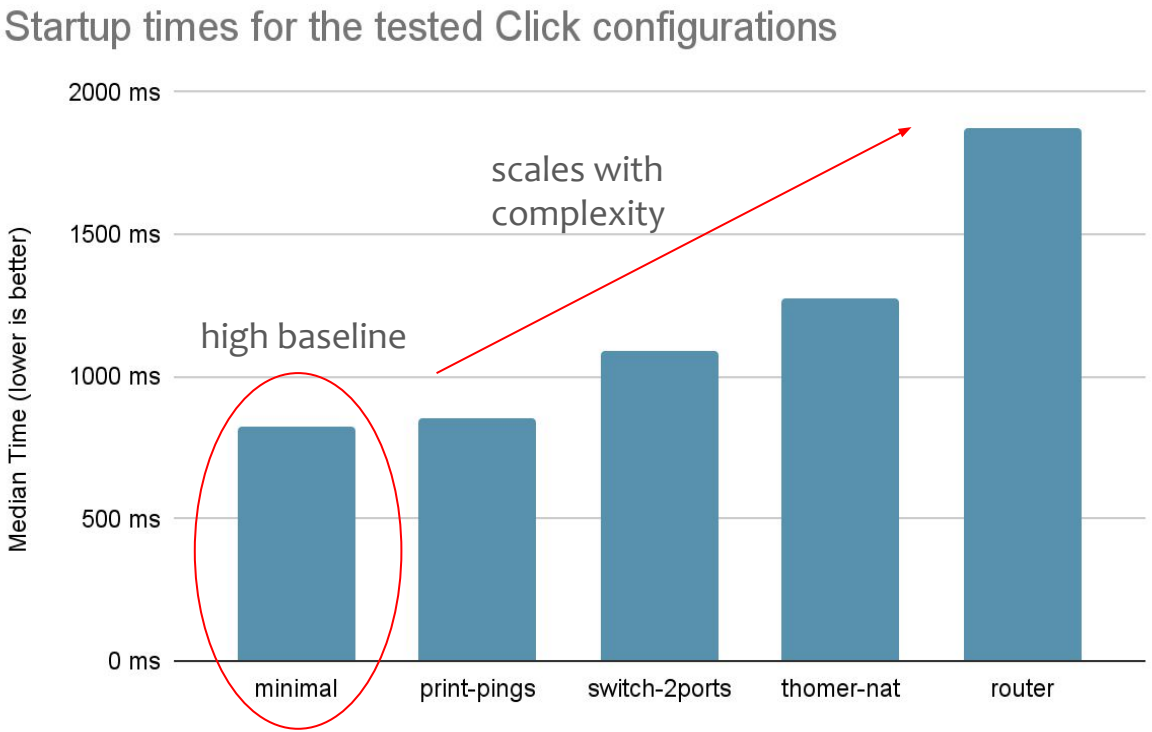


NFV with unikernels



Research Gap: Fast reconfiguration

- VNF updates need restarts => introduces downtime
- Sometimes state needs to be retained



Problem statement

How to allow secure reconfiguration of Unikernel-based VNFs *fast* and *without losing state*?

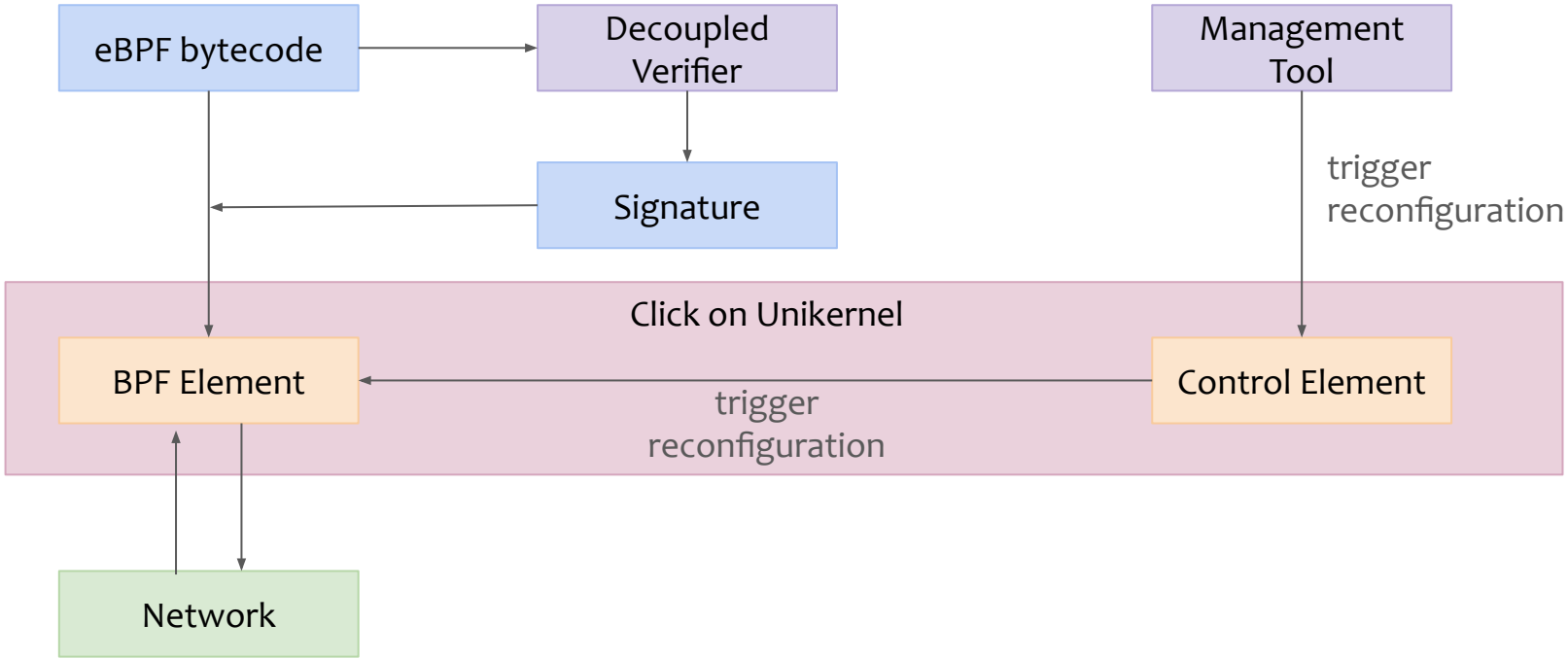
Our framework…

- … should be *reconfigurable*
- … should be *flexible*
- … should be *performant*
- … should be *compatible*
- … should be *secure*

Click Modular Router

- Each Click Element performs a specific networking task such as filtering, routing, inspection, …
- Multiple Click Elements are composed together, achieving the desired functionality

Overview


eBPF bytecode
Decoupled Verifier
Management Tool
Signature
trigger reconfiguration
Click on Unikernel
BPF Element
Control Element
trigger reconfiguration
Network

Outline

- ~~Motivation & Background~~
- **Design**
- Implementation
- Evaluation
- Summary

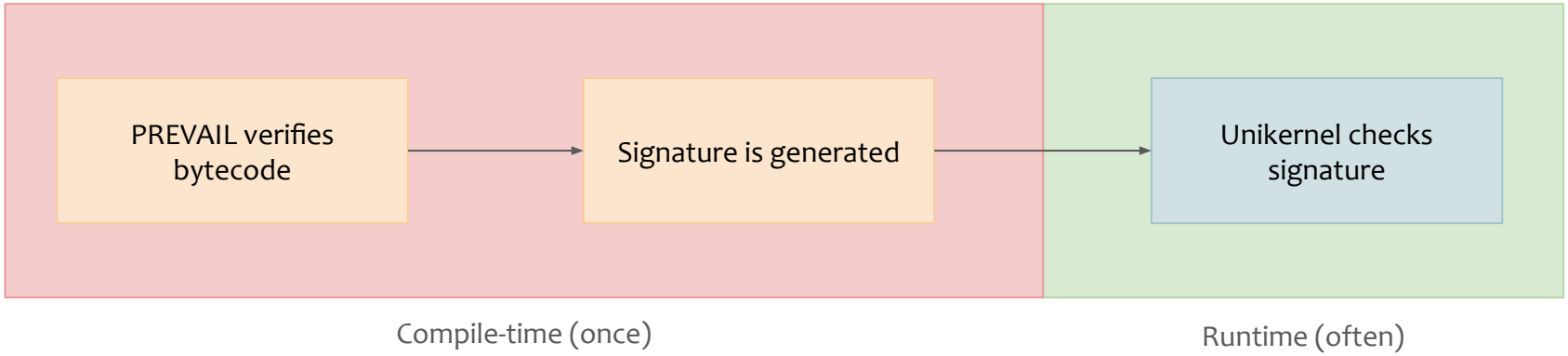
Design challenges

- **Challenge 1:** Lengthy & complex eBPF verification
  => Decoupled eBPF verification

- **Challenge 2:** Integrating eBPF with Click
  => eBPF-based Click Elements

- **Challenge 3:** Retaining state when updating BPF programs
  => Live reconfiguration persisting state of BPF Maps

Challenge 1: Decoupled eBPF Verification

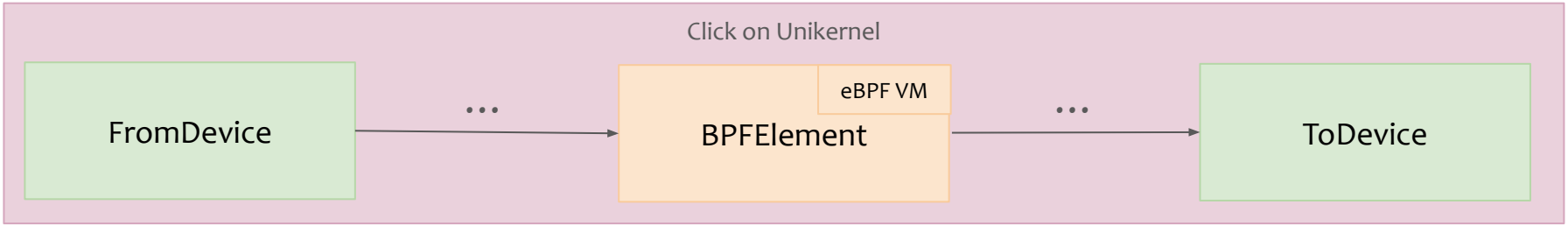


- Verification takes between 91% and 99% of BPF program load time
  => Generate a cryptographic signature verifying the verification!

Reference:

- Enabling eBPF on Embedded Systems Through Decoupled Verification

Challenge 2: Integrating eBPF into Click



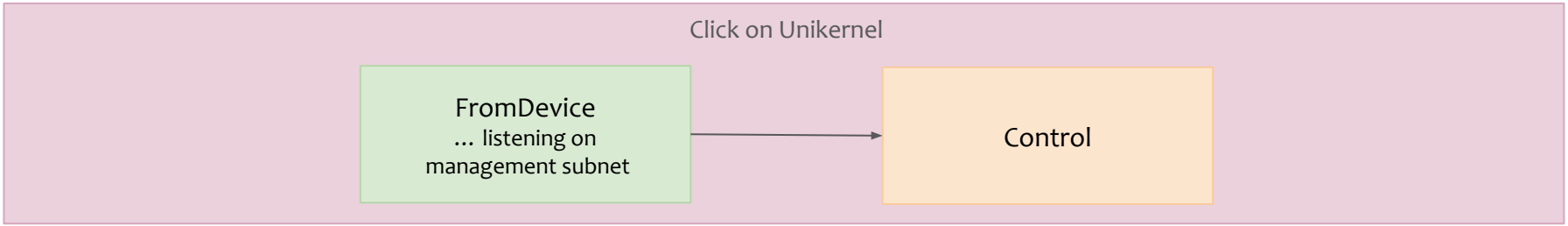
- We add BPF-based elements to Click
  - **BPFFilter:** Filtering packets
  - **BPFClassifier:** Classifies packets (1 input, N outputs)
  - **BPFRewriter:** Rewrites packet contents

Providing BPF-based alternative elements

Middlebox	Key Click Elements	BPF Replacements
Load balancer	RatedSplitter, HashSwitch	BPFClassifier
Firewall	IPFilter	BPFFilter
NAT	[IP|UDP|TCP]Rewriter	BPFRewriter
DPI	Classifier, IPClassifier	BPFClassifier
Tunnel	IPEncap, IPsecESPEncap	BPFRewriter
Multicast	IPMulticastEtherEncap, IGMP	BPFRewriter
BRAS	PPPControlProtocol, GREEncap	BPFRewriter
Monitoring	IPRateMonitor, TCPCollector	
DDoS prevention	IPFilter	BPFFilter
IDS	Classifier, IPClassifier	BPFClassifier
IPS	IPClassifier, IPFilter	BPFClassifier, BPFilter
Congestion control	RED, SetECN	BPFFilter, BPFRewriter
IPv6/IPv4 proxy	ProtocolTranslator46	BPFRewriter

Adapted from ClickOS Paper

Challenge 3: Live Reconfiguration



- Control Element processes all packets forwarded from the FromDevice
  - FromDevice listens on a device bound to a management subnet
  - Passes few preprocessing steps
  - Control element parses & dispatches live reconfiguration signal
- BPF Elements stop processing, load the new BPF bytecode & resume processing

Outline

- ~~Motivation & Background~~
- ~~Design~~
- **Implementation**
- Evaluation
- Further Ideas

Implementation

- eBPF interface modeled after Linux
  - BPF maps & helpers
  - Program interface similar to XDP
- Used following projects
  - **Unikernel:** Unikraft
  - **eBPF interpreter:** ubpf
  - **eBPF verifier:** PREVAIL
- For benchmarks, custom harness and criterion

Outline

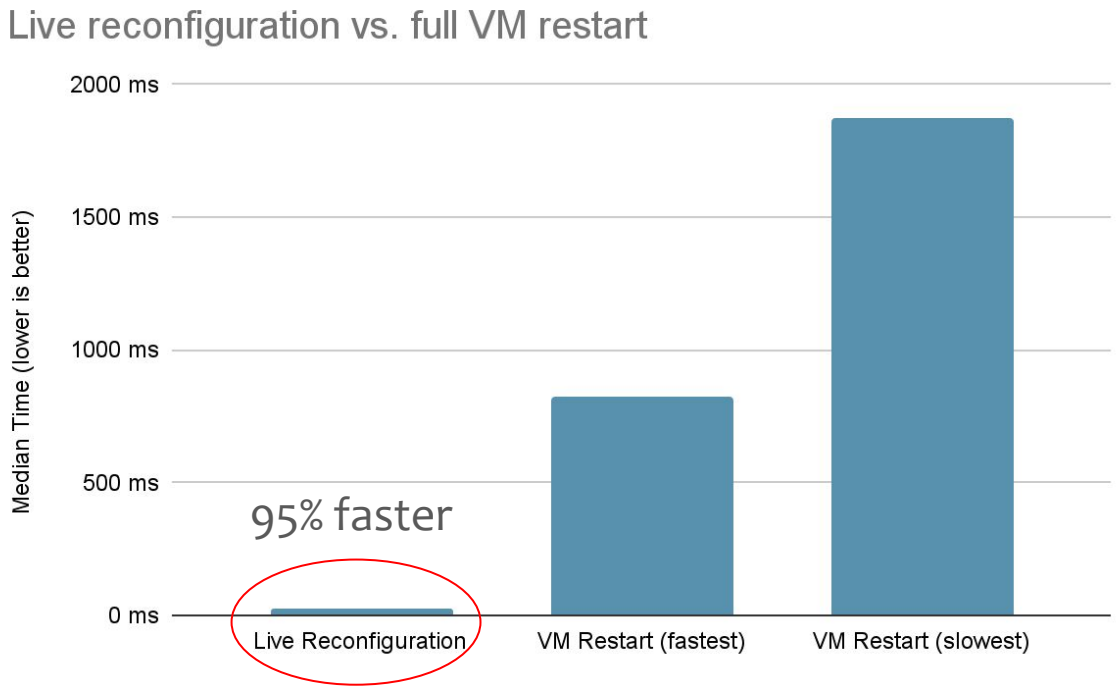
- ~~Motivation & Background~~
- ~~Design~~
- ~~Implementation~~
- **Evaluation**
- Summary

Evaluation: Overview

- Qualitative Evaluation
  - Ease of Development
  - Flexibility
  - **State Migration**
  - Learning Curve
  - Usability
- Quantitative Evaluation
  - Startup Time
  - **Throughput**
  - **Reconfiguration Time**
  - Memory Usage
  - Latency

Evaluation: Reconfiguration

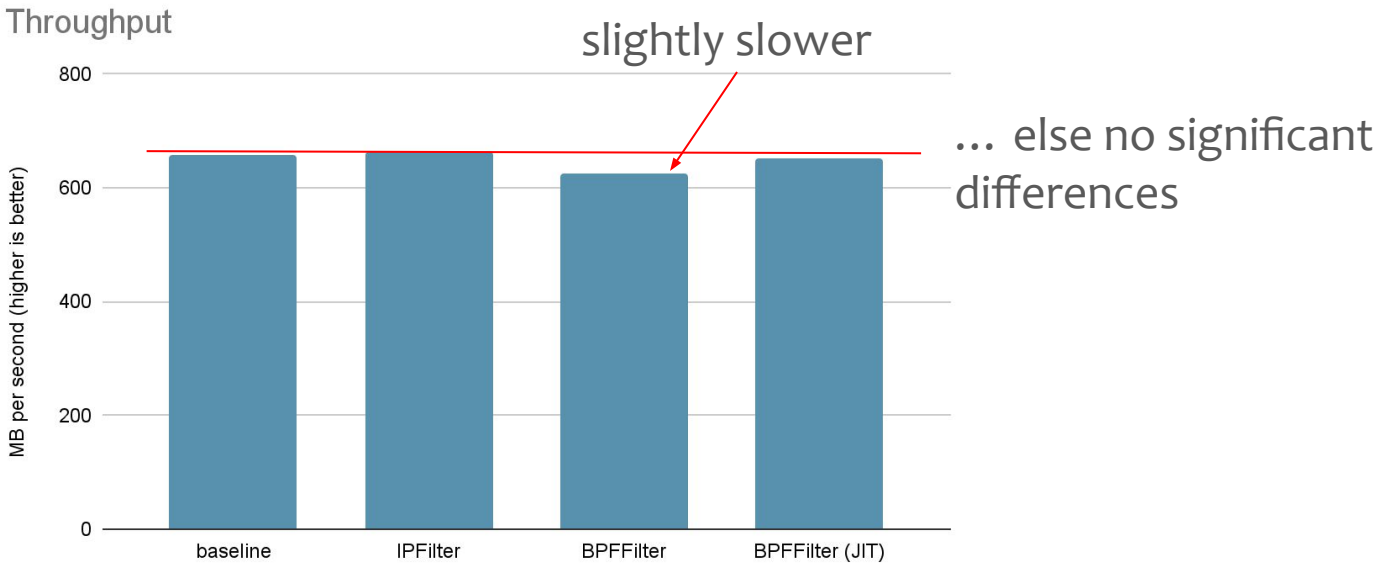
- Reconfiguration takes ca. **20 ms**
- … while retaining state as shown in the State Migration evaluation



Evaluation: Throughput

Reference:
- ClickOS and the Art of Network Function Virtualization

- No significant impact of BPFFilter on packet throughput
  - JIT performs slightly better
- Overall system performance can be improved
  - ClickOS achieved up to 9.68 GB/s



Outline

- ~~Motivation & Background~~
- ~~Design~~
- ~~Implementation~~
- ~~Evaluation~~
- **Summary**

Summary

In this project we have achieved

- eBPF-based NFV framework built on top of Click
  - BPF Elements allow **flexibility** in building different VNFs
  - Decoupled verification allows **fast startup** and guarantees **safety**
  - Live reconfiguration allows **up to 95% faster** updates
    - … while persisting state
- Competitive performance baseline
  - With improvements in networking layer, viable for production use cases