VPP PLUS Middlebox

Tobias Bühler - 30.01.2018

Cambridge Mami meeting

https://github.com/mami-project/vpp-plus

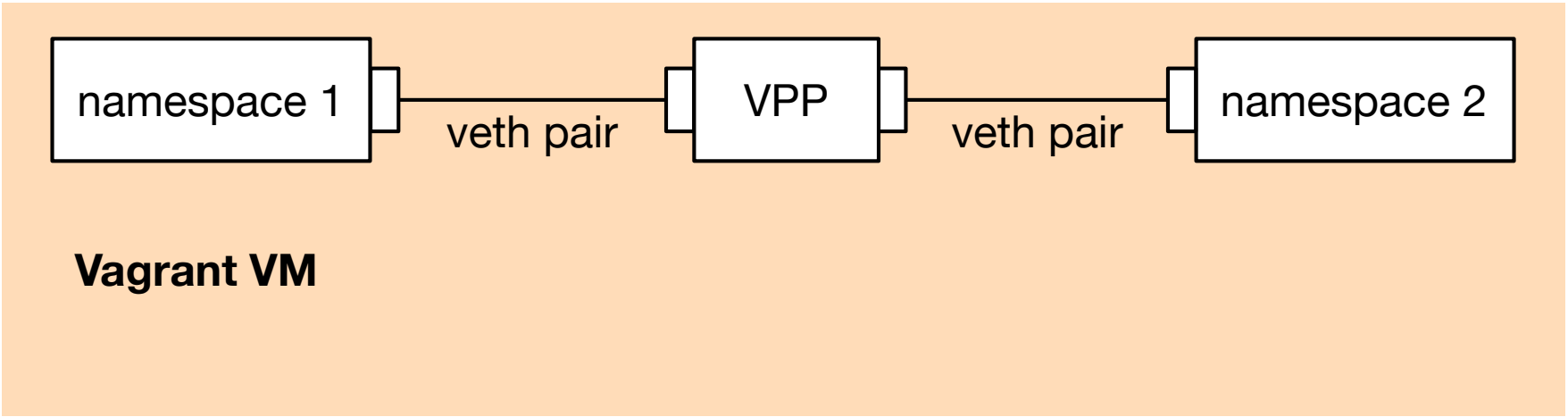
measurement and architecture for a middleboxed internet

measurement | architecture | experimentation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421.The opinions expressed and arguments employed reflect only the authors' view. The European Commission is not responsible for any use that may be made of that information.

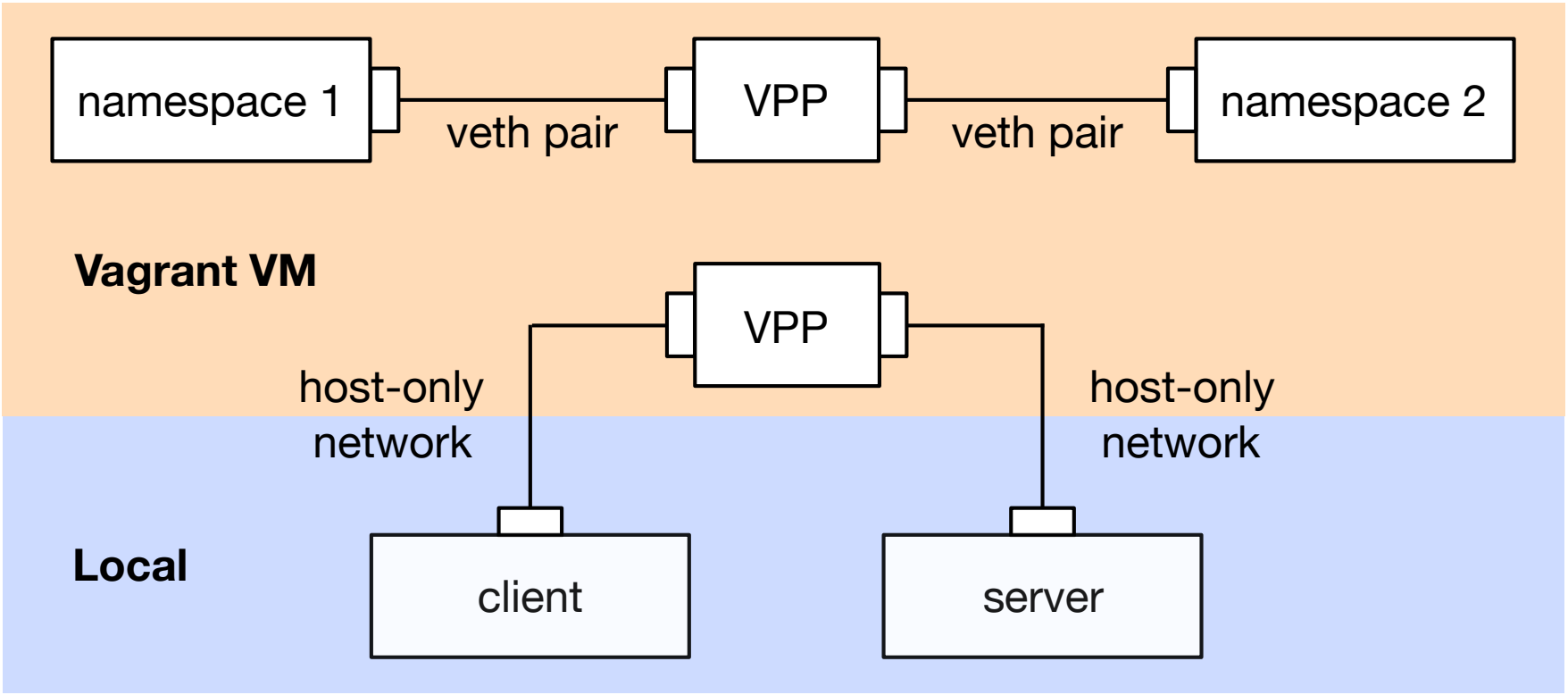
Current Setup

- VPP (17.10) is running in a Vagrant VM (Ubuntu 16.04)



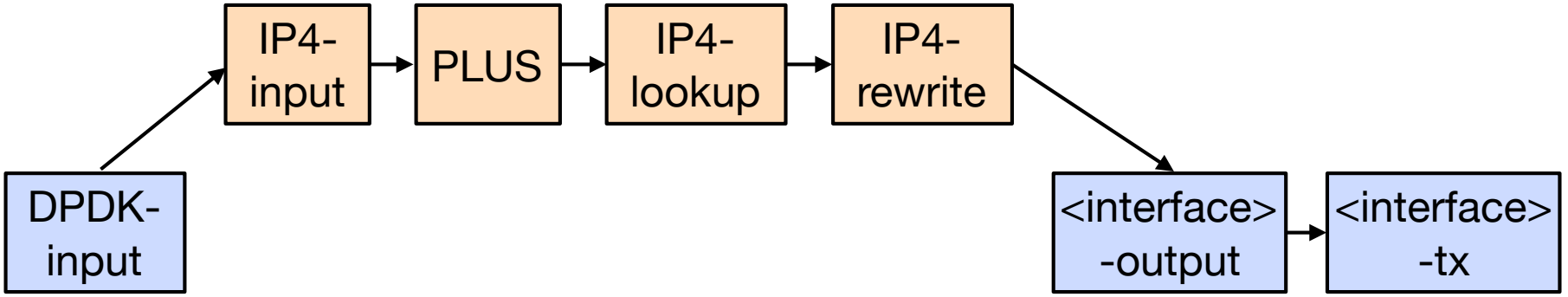
Current Setup

- VPP (17.10) is running in a Vagrant VM (Ubuntu 16.04)



VPP Tree - PLUS Node Placement

- Can easily be moved (plugin runs before/after …)
- Currently, we get only valid packets
- Before any forwarding decisions (important for e.g. QoS)

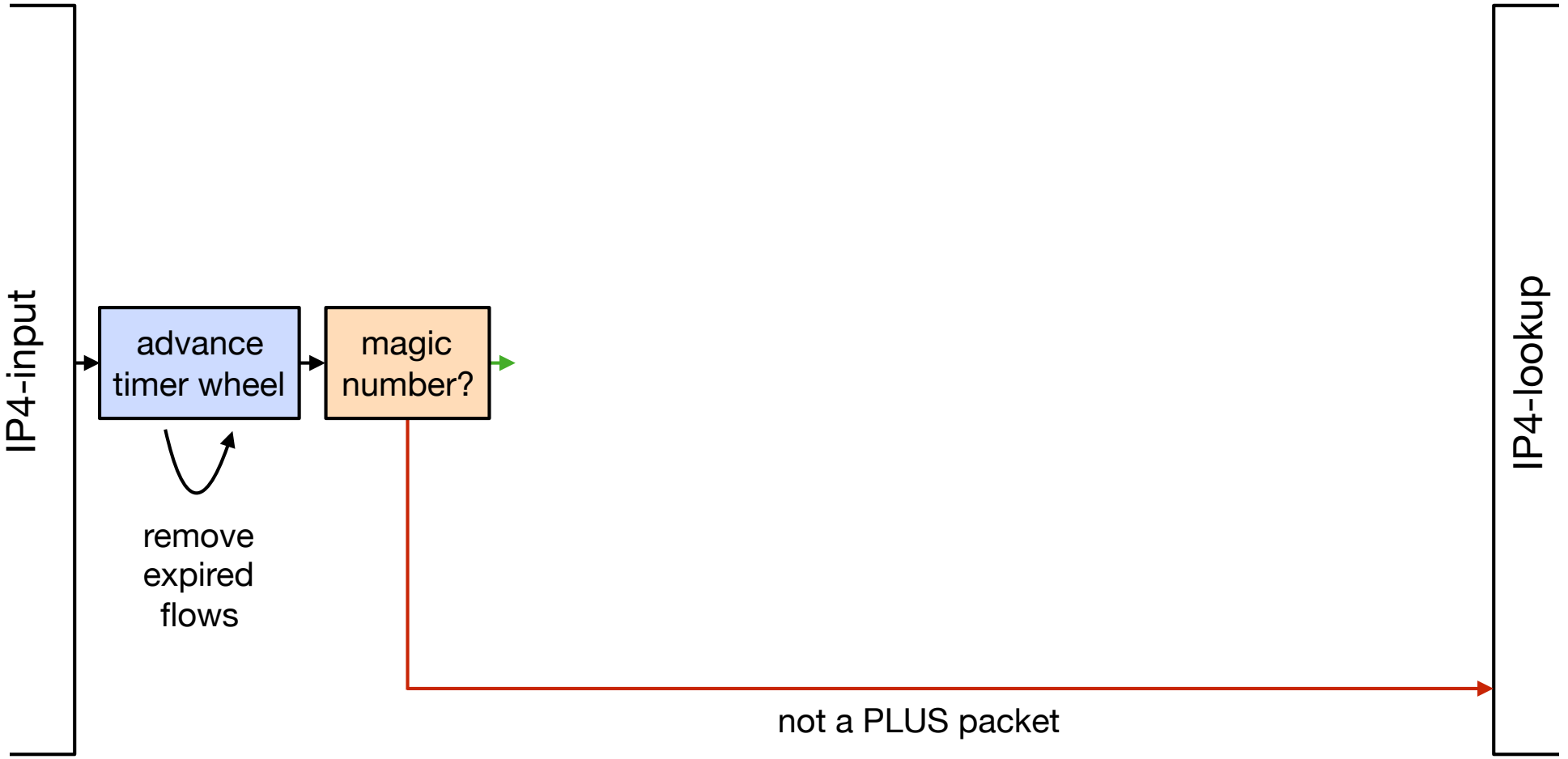


PLUS Node - Workflow

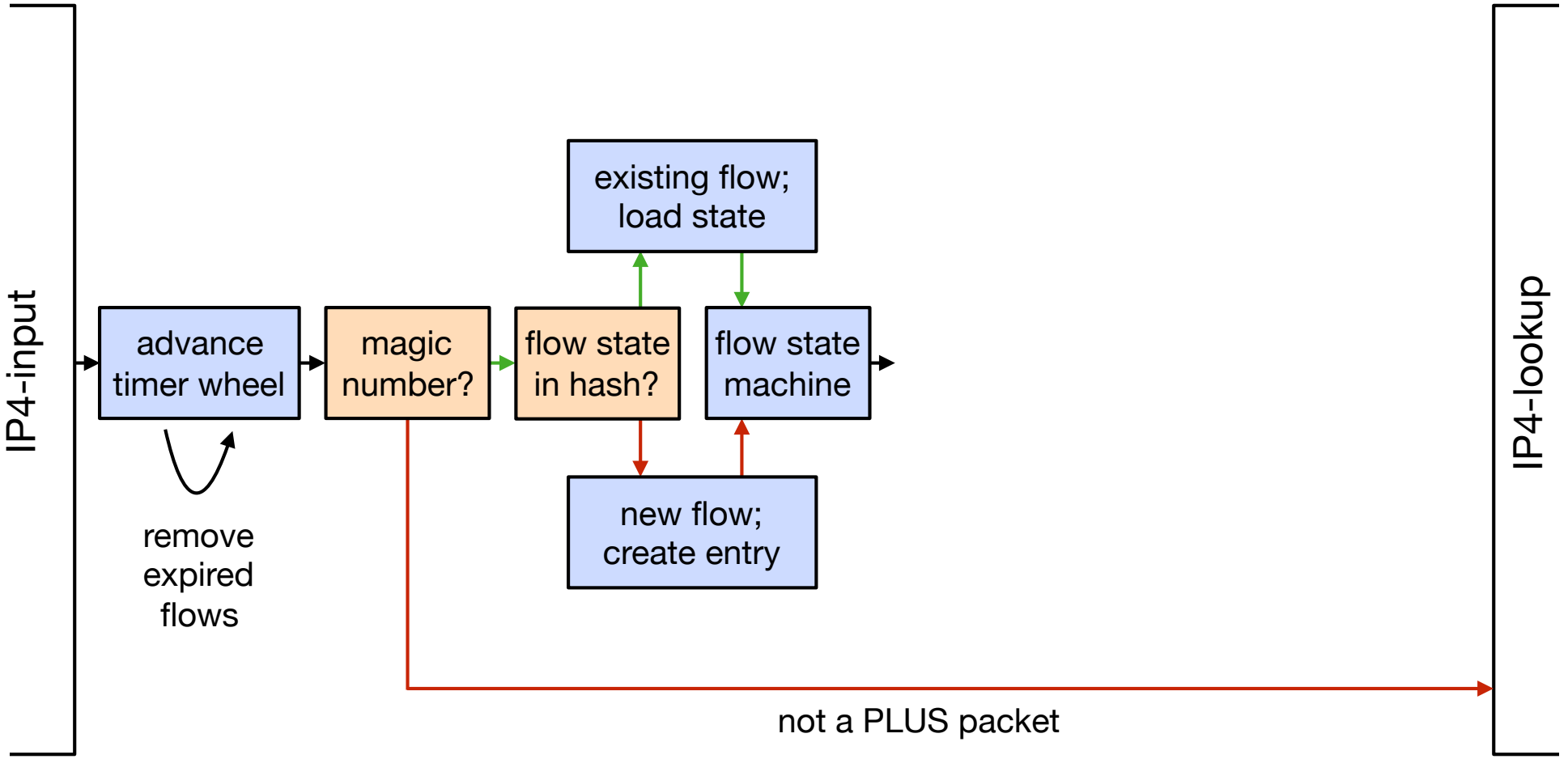
IP4-input

IP4-lookup

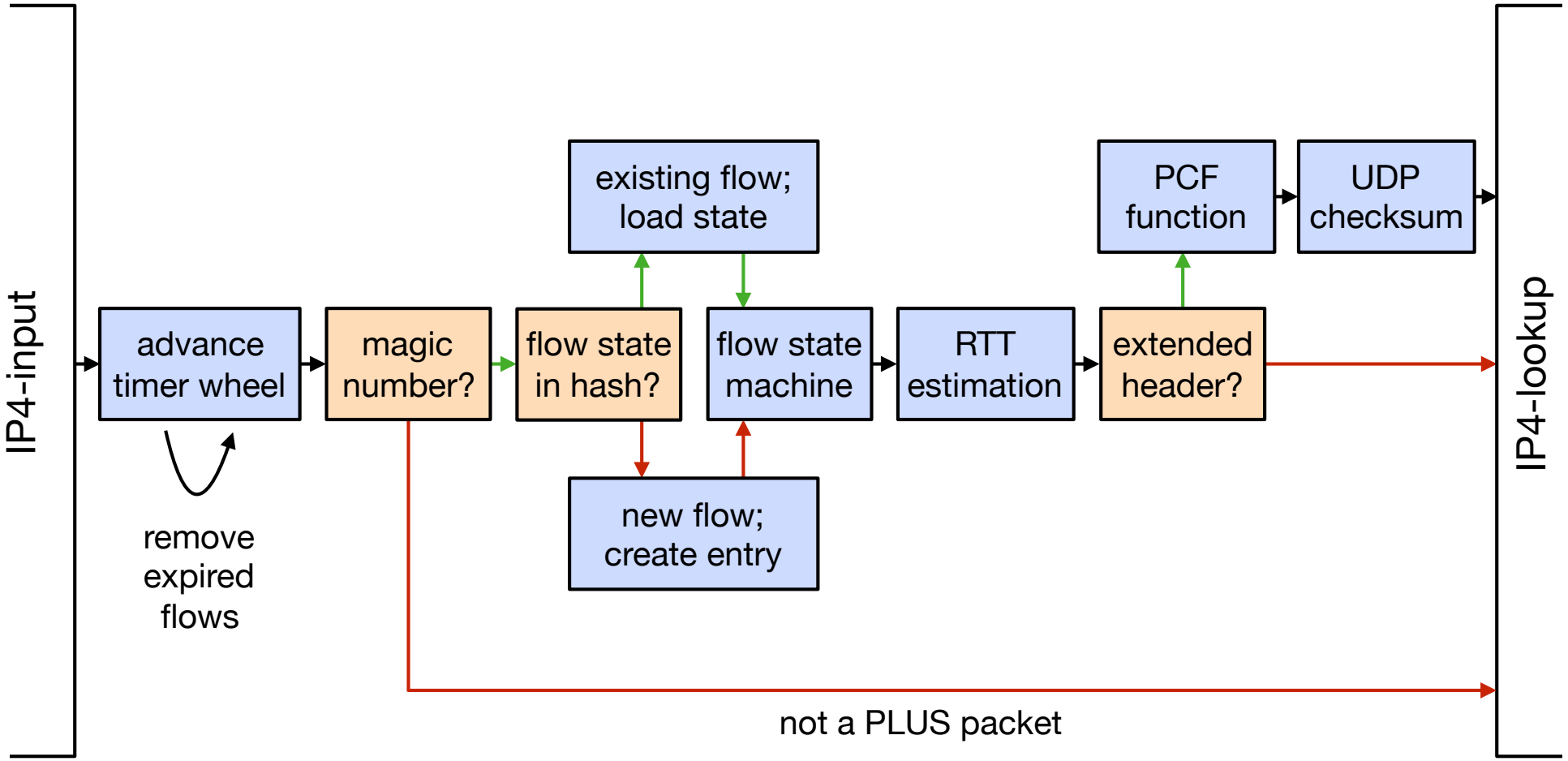
PLUS Node - Workflow



PLUS Node - Workflow



PLUS Node - Workflow

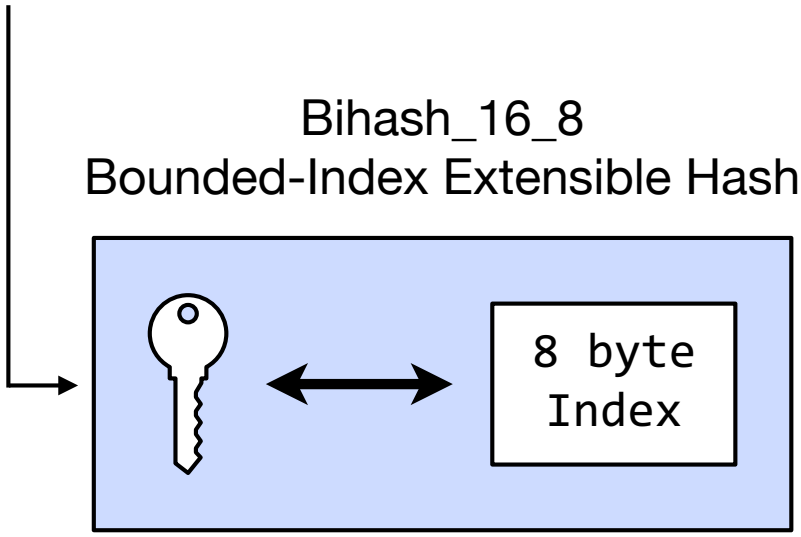


State Management

Key: 5-Tuple (IPv4) + CAT

src IP	XOR	dst IP		4 bytes
src port	XOR	dst port		2 bytes
protocol			(1)	2 bytes
CAT				8 bytes
				16 bytes

Bihash_16_8
Bounded-Index Extensible Hash



State Management

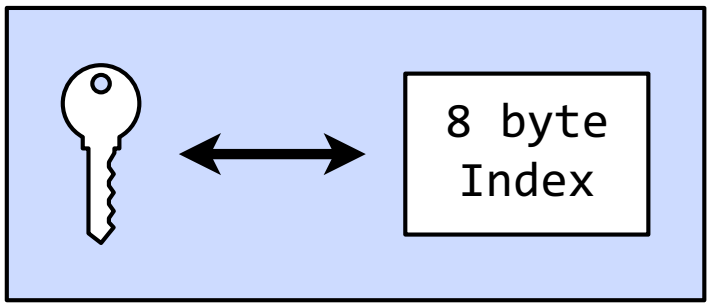
Key: 5-Tuple (IPv4) + CAT

src IP	XOR	dst IP	4 bytes
src port	XOR	dst port	2 bytes
protocol		(1)	2 bytes
CAT			8 bytes
			16 bytes

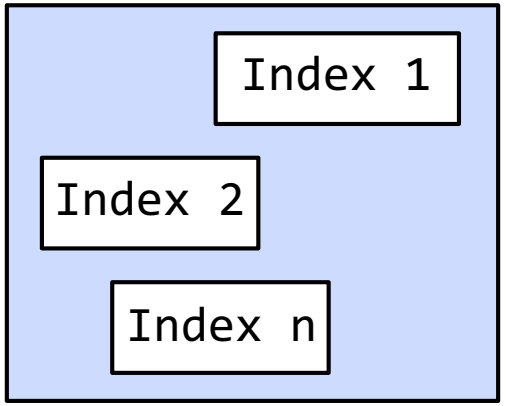
Flow state

```
State (UNIFLOW, …)
PSN and PSE
RTT estimation
Packet counter
...
```

Bihash_16_8
Bounded-Index Extensible Hash



Pool



Supported Features

- Activate each interface separately
- Full state machine with timers
- RTT estimation based on PSN/PSE pairs
- Optional header „**path accumulator**“:
  - PCF Type = 1, PCF Len = 1, PCF_II = 0
    PCF Value: counter increased by each PLUS MB
- Full flow output at any time
- Full packet traces

VPP in Action - Packet Trace

- Full packet trace through all VPP nodes

```
00:00:53:314457: ip4-input
  UDP: 192.168.100.1 -> 192.168.101.1
  …

00:00:53:314463: plus
  PLUS packet: CAT: 11154013587666973726, PSN: 434598561, PSE: 0
  Current state: UNIFLOW, stop bit: 0, extended bit: 1
  PCF type: 1, PCF len: 1, PCF II: 0, PCF hop count value: 1

00:00:53:314475: ip4-lookup
  fib 0 dpo-idx 3 flow hash: 0x00000000
  …
```

VPP in Action - Statistics

- At any time via CLI command: *sudo vppctl plus stat*

```
Total flows: 17, total active flows: 3
==================================================
Flow CAT: 11154013587666973726, observed packets: 648
Current state: ASSOCIATED, estimated RTT: .002135988s
==================================================
Flow CAT: 5834098447867823, observed packets: 2048
Current state: STOPPING, estimated RTT: .001189906s
==================================================
Flow CAT: 273818860780310528O, observed packets: 47
Current state: ASSOCIATED, estimated RTT: .009428921s
==================================================
```

Some Numbers

- Average time in PLUS node: **~3µs** (in VM on laptop)
  - Longest task: extended header with checksum update
- Currently support for **2'048** concurrent flows
  - Extensible: larger/multiple timer wheels, hashes & pools
- A lot of possibilities for code optimizations
  - E.g. double-loop, multiple threads
- Does not look too bad for a first version

Current Work

- Tests with the PLUS Go code from ZHAW (PLUS-lib)
  - Roman and Stephan
- Detected problems:
  - Go does not like virtual namespaces
  - Code bugs — UDP checksum was not always correctly updated after changes in extended header
  - Newest code not pushed to GitHub ;-)

Planned Experiments

- VPP setup on an ETHZ server (after SIGCOMM deadline)
- Tests with the plus-quic-go code:
  - https://github.com/mami-project/plus-quic-go
  - Webserver & crawler
  - Concurrent connections, RTT estimation, full load, …
- Support for LRS bits
  - Need matching setup/network
- RTT comparisons with ping/TCP (compare QUIC work)