\Orchestrating a brighter world NEC

Realizing Stateful Flow Processing with Forwarding Tables and In-Switch-Generated Messages

Peer Hasselmeyer, Andreas Ripke, Fabian Schneider
NEC Laboratories Europe

Acknowledgement – SSICLOPS & Beba

This presentation has received funding from the European Union's Horizon 2020 research and innovation programme 2014-2018 under grant agreements No. 644122 and No. 644866.

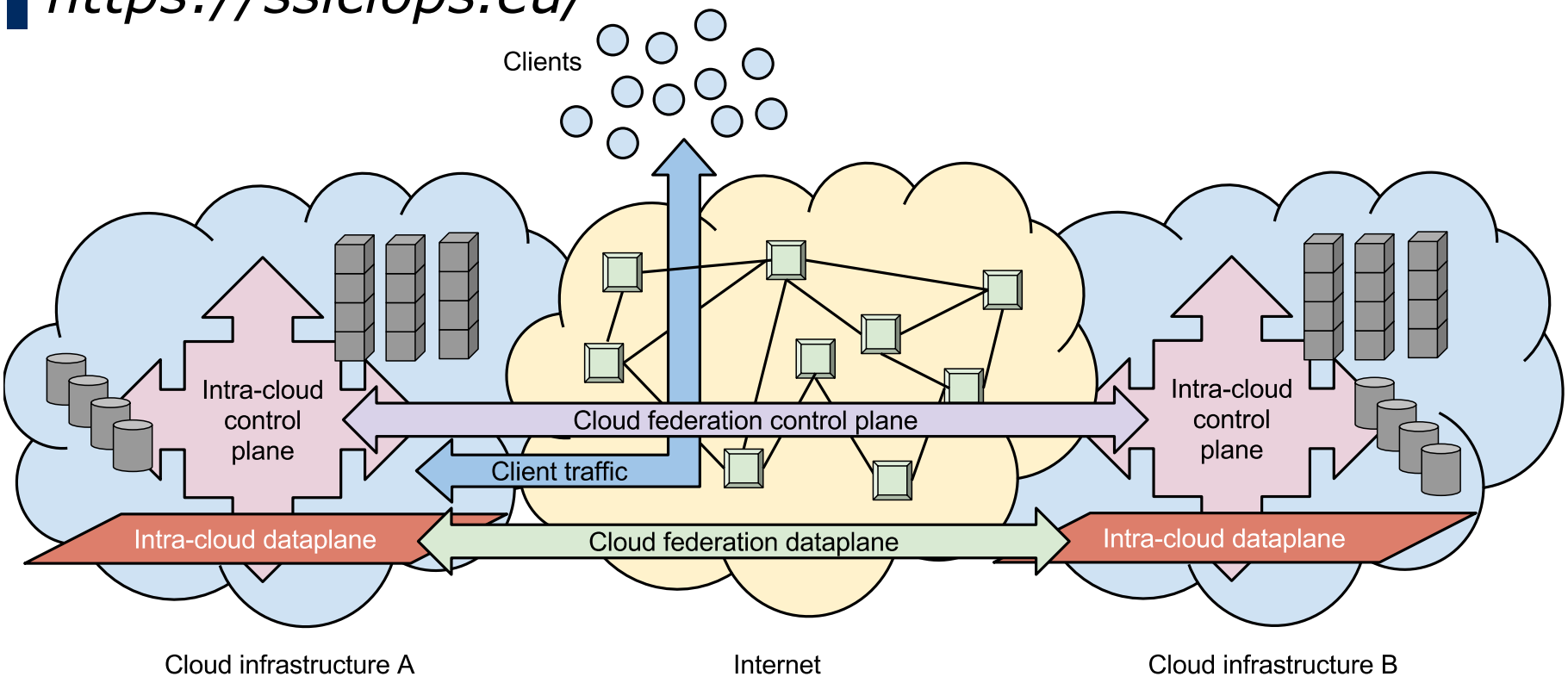
Disclaimer

This presentation reflects only the authors' views and the European Commission is not responsible for any use that may be made of the information it contains.

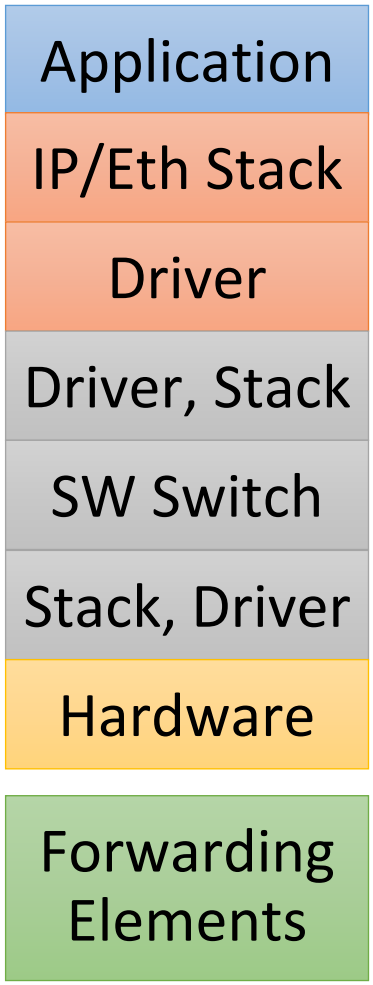
SSICLOPS H2020 Project

- "Scalable and Secure Infrastructures for Cloud Operations"
- Optimizing and securing packet transport in and across data centers
- *https://ssiclops.eu/*

SSICLOPS



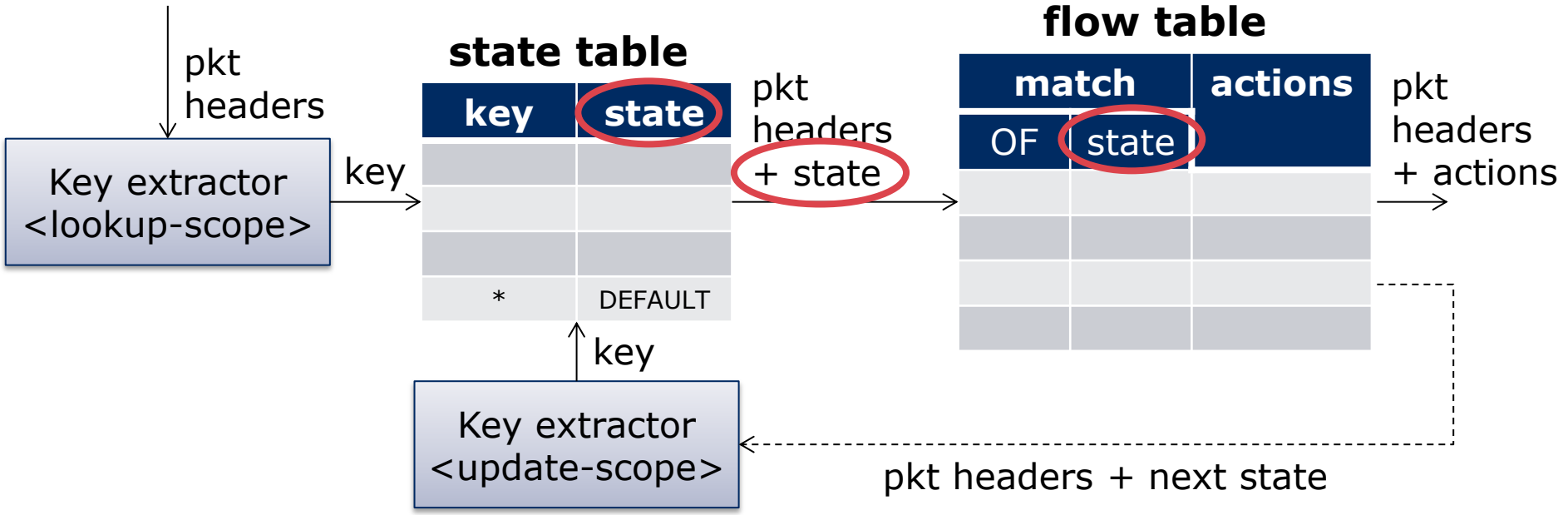
SSICLOPS H2020 Project



- Network stack includes various components: TCP/IP stack, congestion control, path management
- Existing work shows that drastic improvements are possible, including netmap, DCTCP, MPTCP, ...
- Includes SDN on various layers, including centralized control and towards forwarding elements

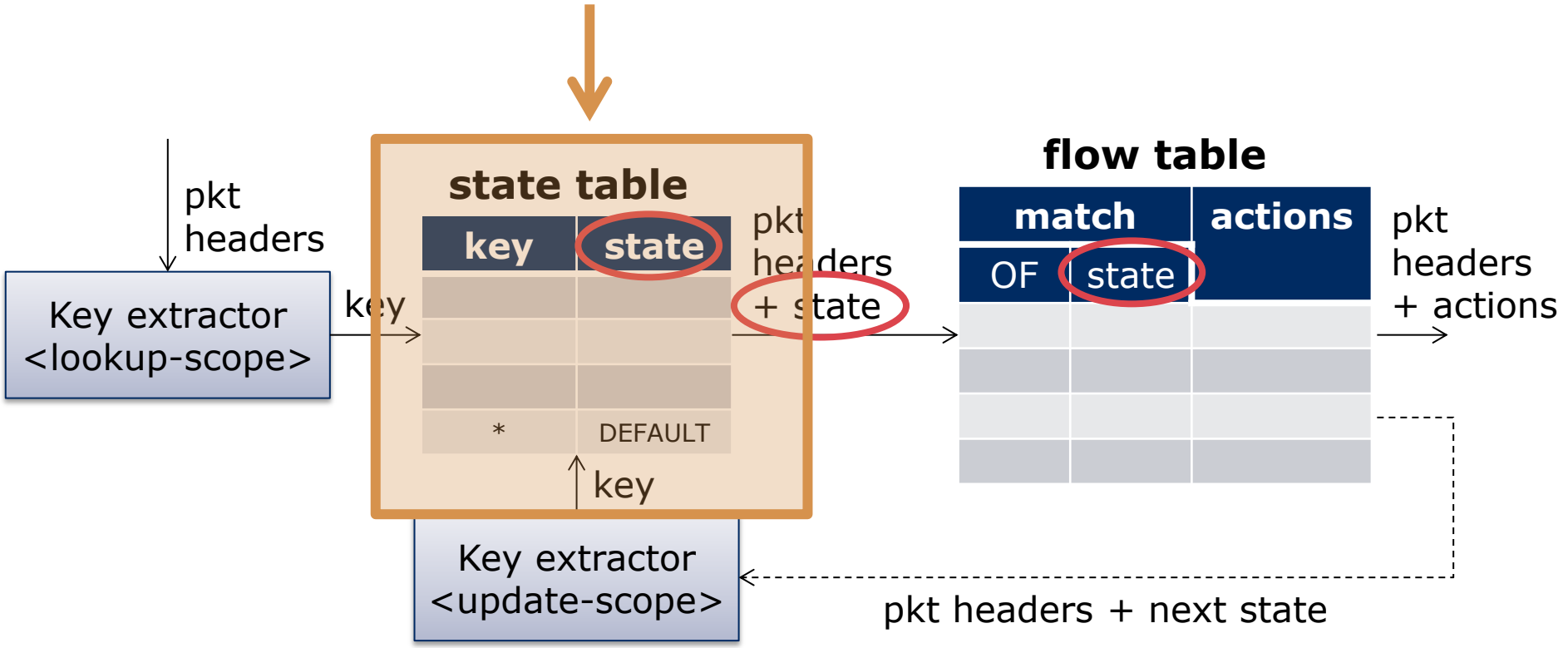
Stateful Flow Processing

- **Goal**: delegate processing from controller to forwarding elements (FEs)
- Keep state at FEs
- Forwarding behavior affected by state
- State affected by incoming packets



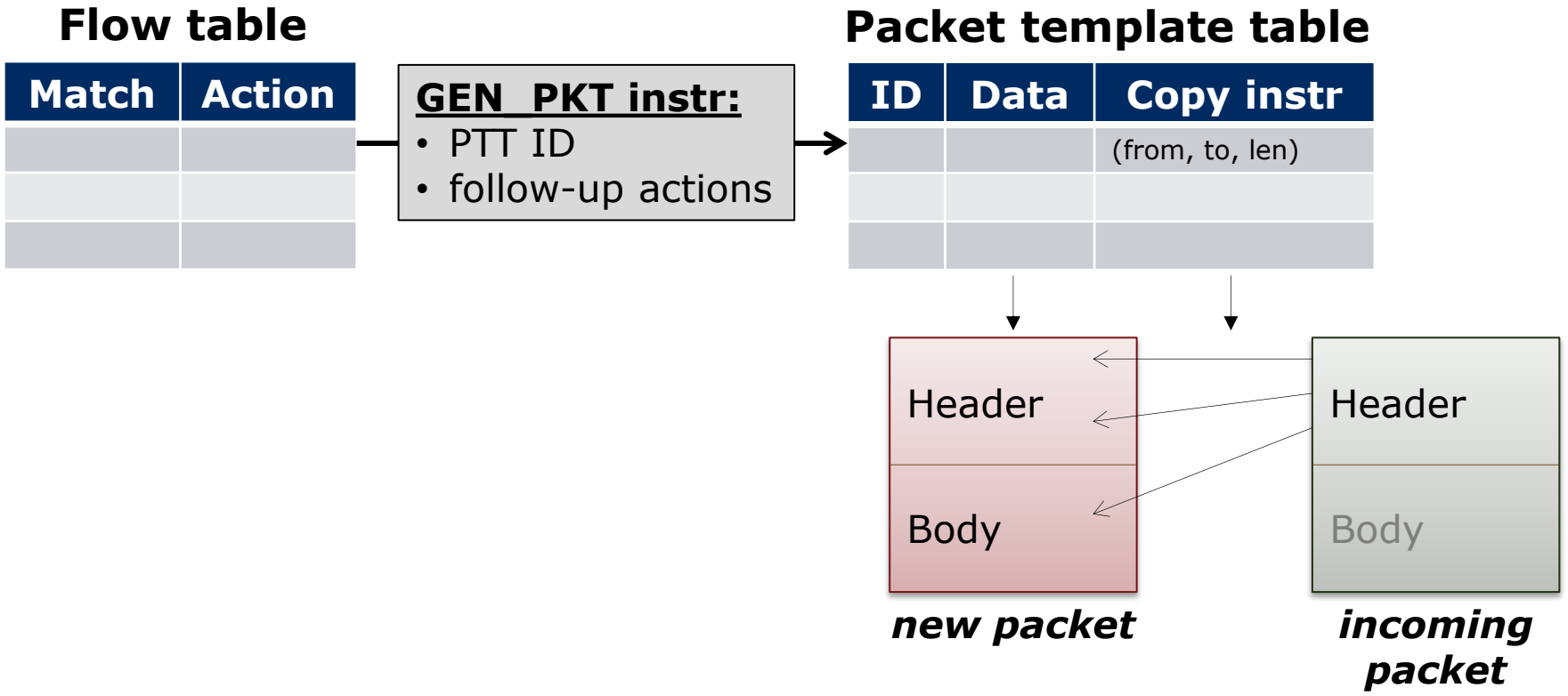
Stateful Flow Processing

- State keeping is typically done in a state table, which is additional to other FE components
- **Issue**: added components (TCAM) and complexity

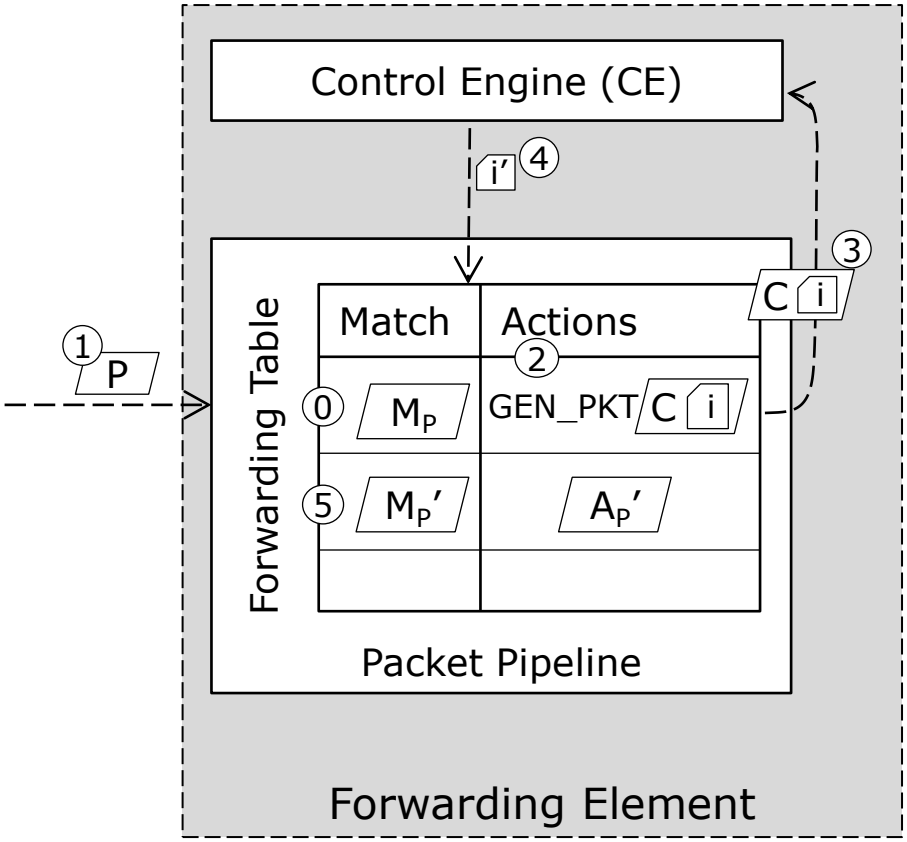


In-Switch Packet Generation

- **Idea**: let switches generate and send packets
- Based on controller instructions and incoming packets
- Can be realized as new OpenFlow action



Stateful Processing with Packet Generation



- **Idea**: get rid of separate state table by manipulating the forwarding table
- **Assumption**: packet generation capability available
- Switch sends OF control messages to itself
- Such messages update forwarding table to reflect state changes

Example: Stateful Firewall

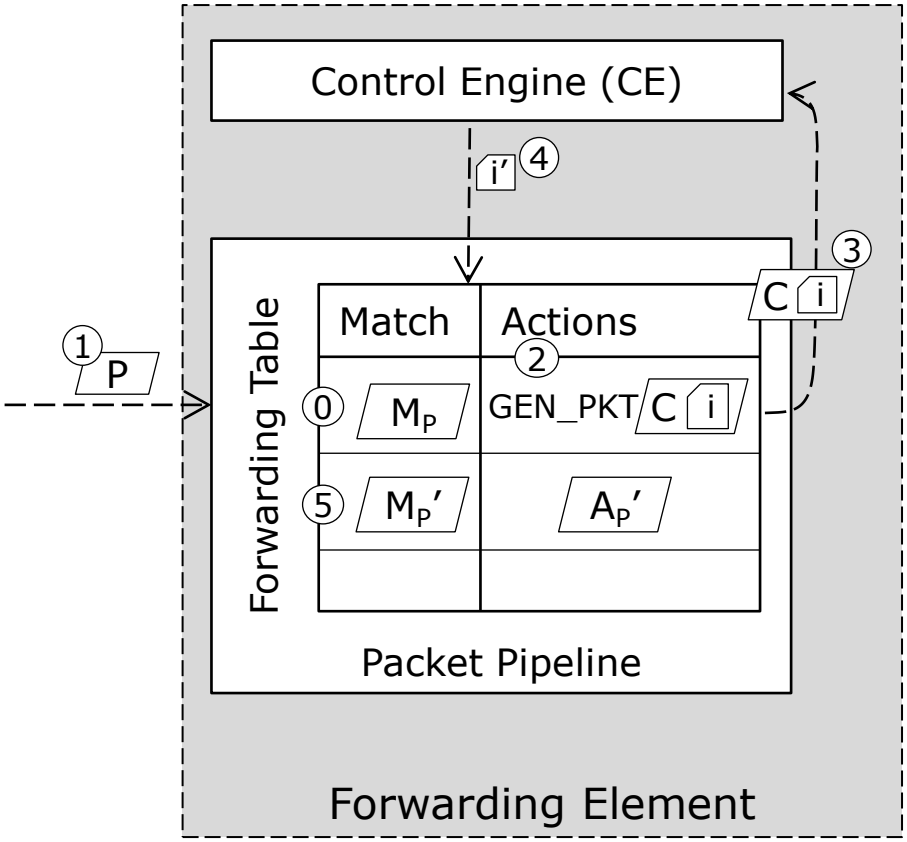
Flow Table

Match	Prio	Actions
in-port == *EXT*	low	drop
in-port == *INT*	low	output(*EXT*) & gen-pkt(id=*1,* output(*LO*)) & gen-pkt(id=*2,* output(*LO*))
src-ip==C, dst-ip==S, L4proto=*TCP*, src-port==*pC*, dst-port==*pS*,	high	output(*EXT*)
src-ip==*S*, dst-ip==*C*, L4proto=*TCP*, src-port==*pS*, dst-port==*pC*,	high	output(*INT*)

Packet Template Table

ID	Data	Copy instructions
1	src-ip=<switchIP>, dst-ip=<switchIP>, L4proto=UDP, body=flow_mod(*match*, action = output(*EXT*), prio = high)	In *match*: src-ip=**src-ip**, dst-ip=**dst-ip**, L4proto=**L4proto**, src-port=**src-port**, dst-port=**dst-port**
2	src-ip=<switchIP>, dst-ip=<switchIP>, L4proto=UDP, body=flow_mod(*match*, action = output(*INT*), prio = high)	In *match*: src-ip=**dst-ip**, dst-ip=**src-ip**, L4proto=**L4proto**, src-port=**dst-port**, dst-port=**src-port**

Implementation Aspects



Packet transfer

- *patch cable*: send out port and receive on another
- *loopback*: use an internal loopback port
- *internal*: transfer separate from regular data path

Connection

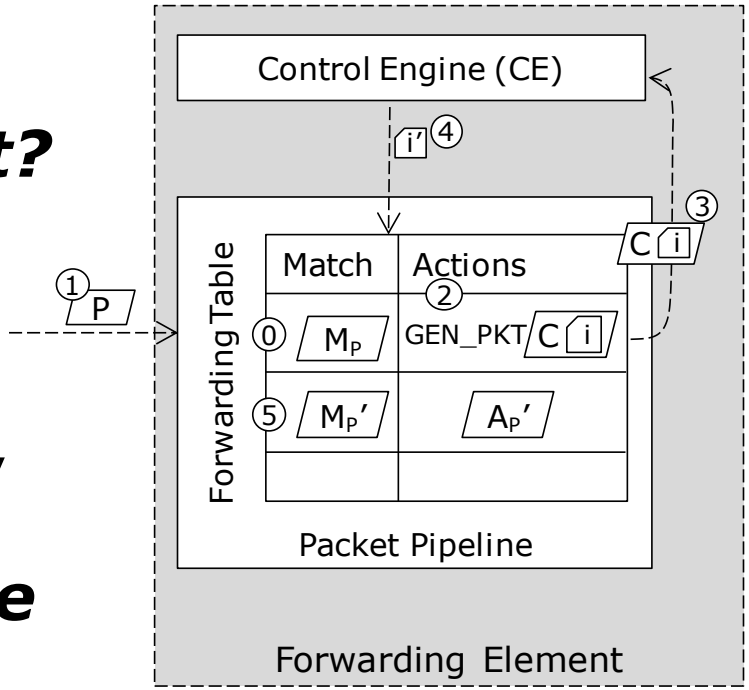
- use UDP, to avoid TCP overhead (connection setup, state management, flow control, ...)
- possible with OpenFlow's auxiliary connections

Summary

- Stateful processing can enhance programmability
- Implementation in hardware costly (money and complexity)
- Proposed to use in-switch packet generation to realize stateful flow processing by generating OF control messages

Discussion

- Proposal requires packet generation functionality in switches: ***how realistic is that?***
  - might need packet generation table
  - does not need to reside in TCAM, rather regular DRAM
- No added complexity on switch, but impact on control program unclear: ***is gain on switch side worth increased complexity of controller?***
- Processing of control messages needs time (slow path!) → state updates reflected with delay ("eventual consistency"): ***which applications can tolerate this? Which cannot?***




Orchestrating a brighter world
NEC