APKeep: Realtime Verification for Real Networks

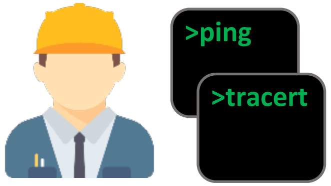
Peng Zhang[*], Xu Liu[*], Hongkun Yang[+], Ning Kang[*], Zhengchang Gu[*], Hao Li[*]

[*] *Xi'an Jiaotong University*, [+]*Google*

Background

Network outages are common

human misconfiguration, software bugs, etc.



Post-effect troubleshooting is slow

manually find the root cause after outages using simple tools

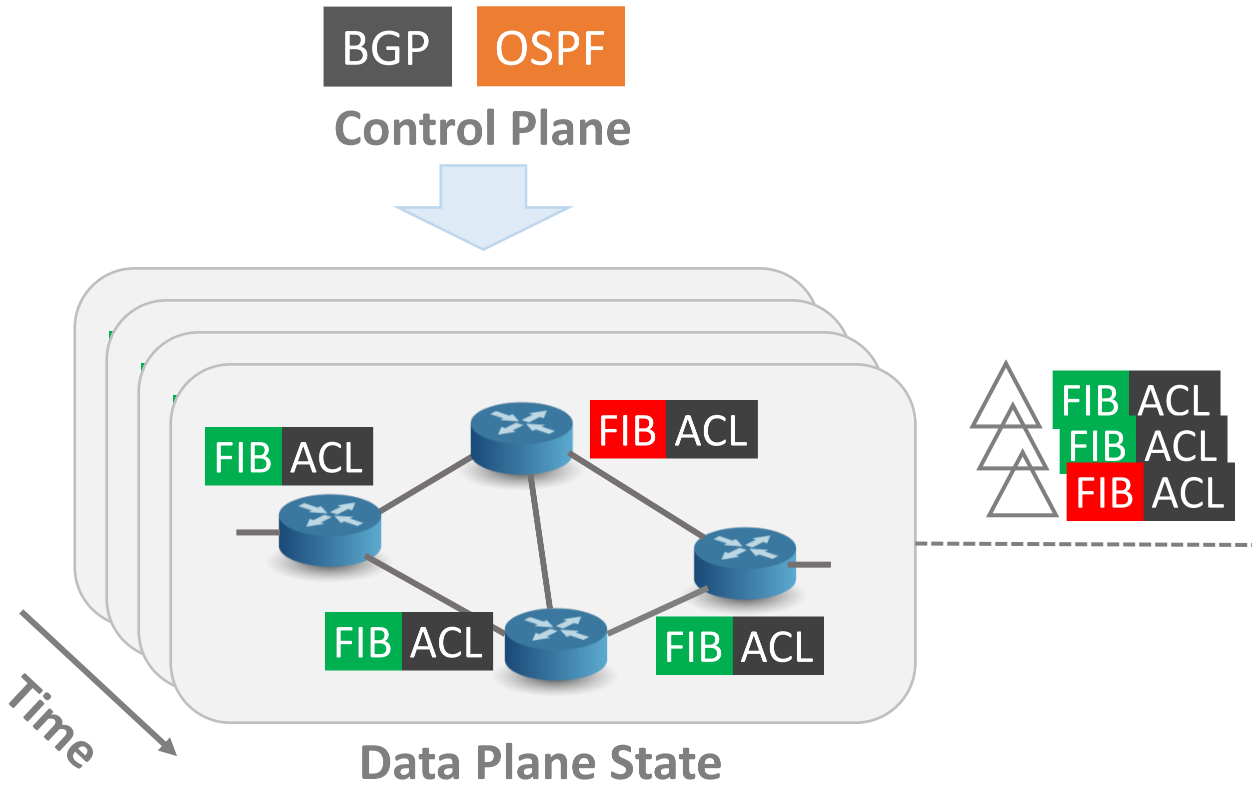
The cost can be quite expensive

service down for hours/days, heavy loss of revenue

Network Verification

automatically check network correctness with formal methods

Realtime Network Verification



Realtime Network verification

VeriFlow [NSDI'13], NetPlumber [NSDI'13],
AP Verifier [ICNP'13], Delta-net[NSDI'17]

Check correctness of data plane state after each update in real time

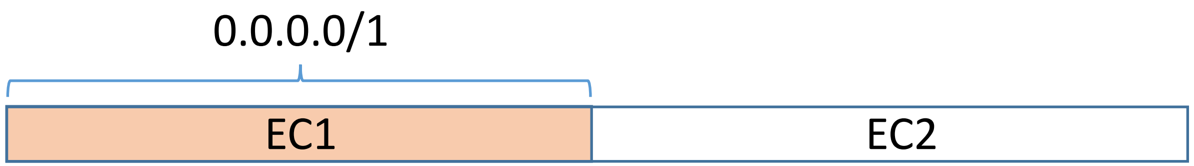
correctness: reachability, no loops, no blackholes, isolation, …

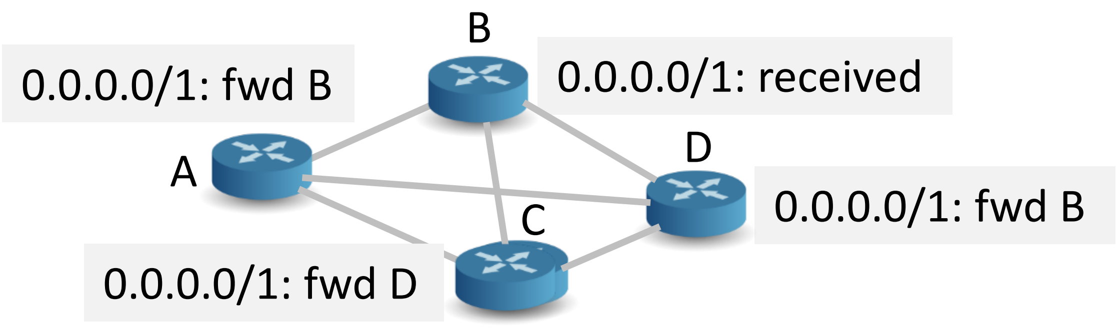
real time: verify each update in small amount of time (<1ms)

Preliminary to Realtime Network Verification

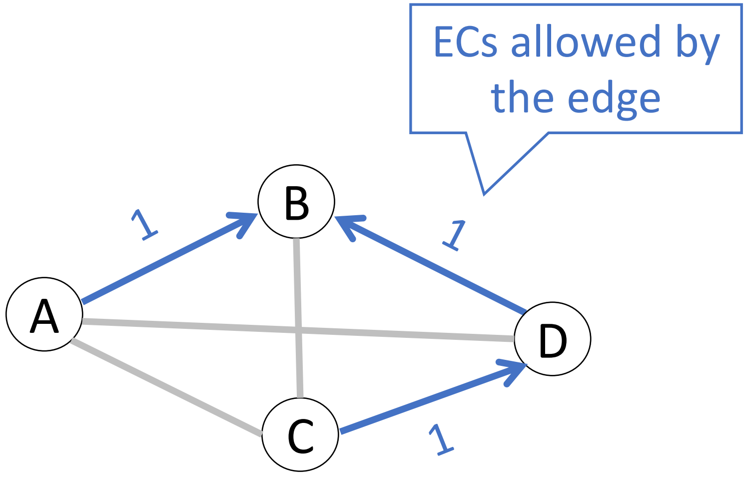
Equivalence Class (EC): a set of packets with the same forwarding behavior

IP header space





Data plane state



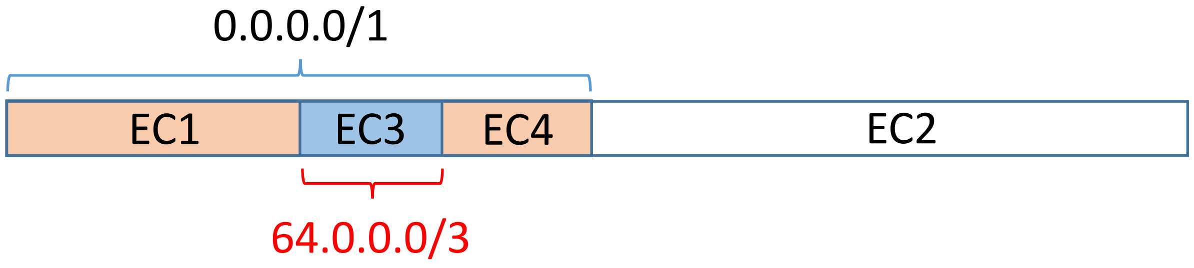
Network model

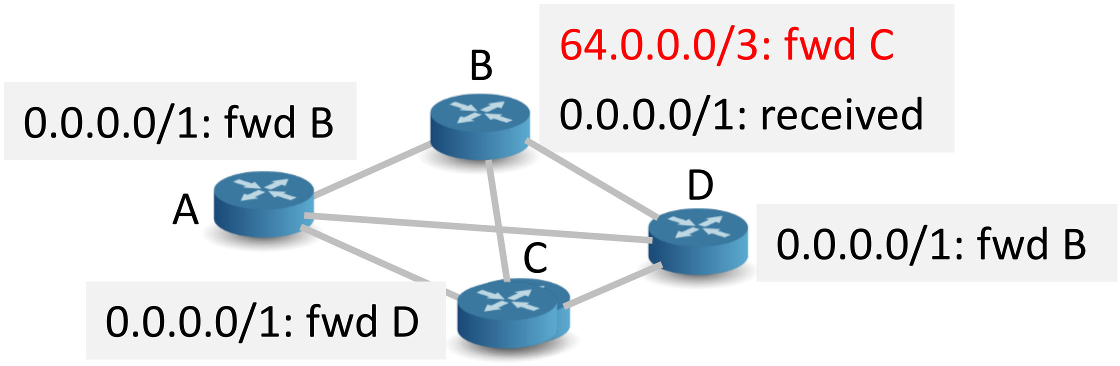
Preliminary to Realtime Network Verification

Incremental update and verification [VeriFlow, NSDI'13] [AP Verifier, ICNP'13] [Delta-net, NSDI'17]

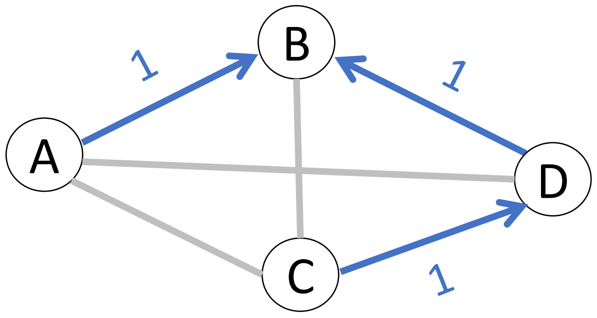
Update the ECs

IP header space





Data plane state



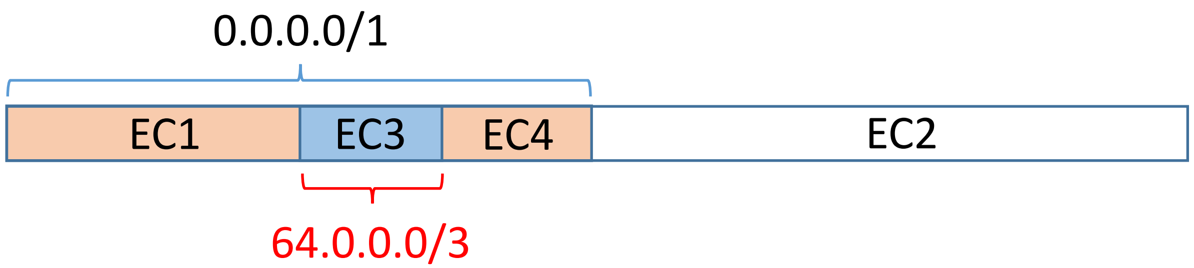
Network model

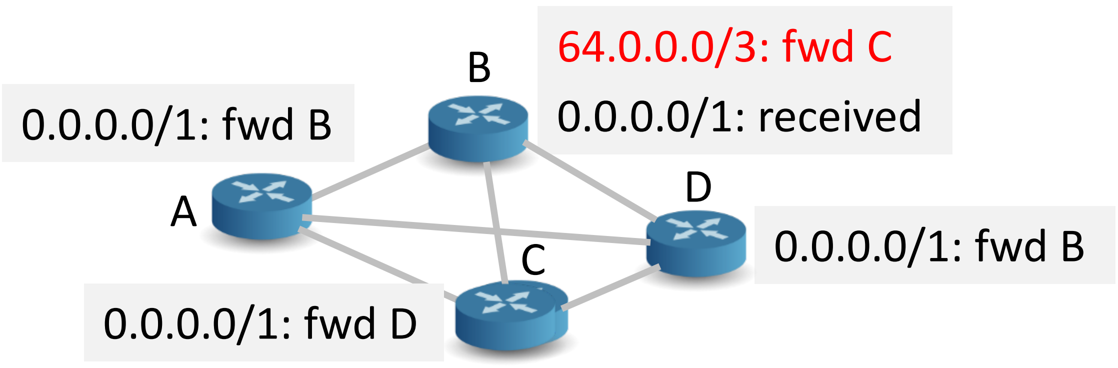
Preliminary to Realtime Network Verification

Incremental update and verification [VeriFlow, NSDI’13] [AP Verifier, ICNP’13] [Delta-net, NSDI’17]

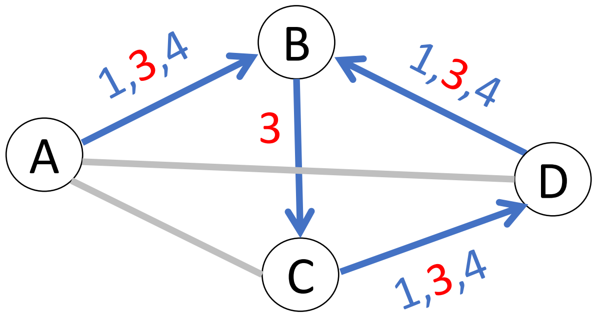
Update the ECs >> Update the model

IP header space





Data plane state



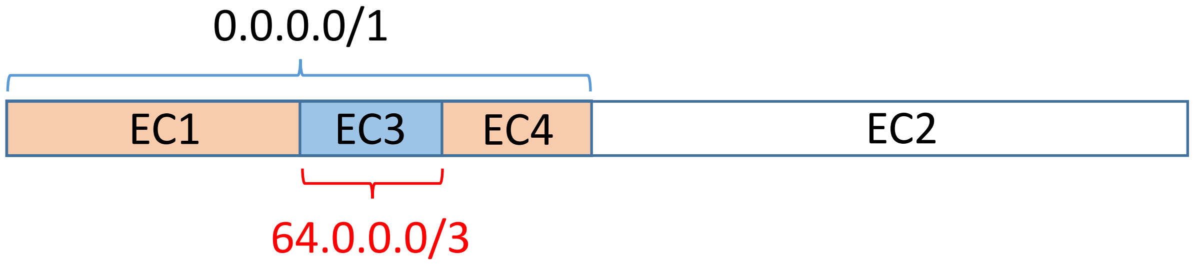
Network model

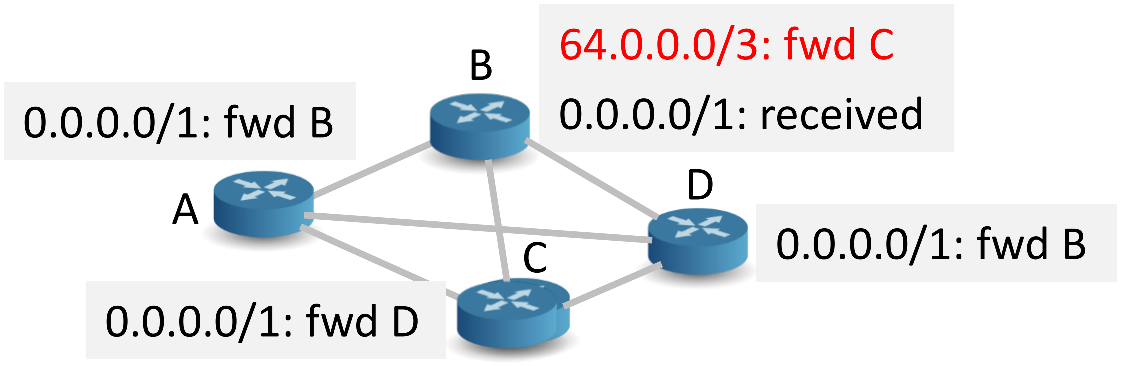
Preliminary to Realtime Network Verification

Incremental update and verification [VeriFlow, NSDI'13] [AP Verifier, ICNP'13] [Delta-net, NSDI'17]

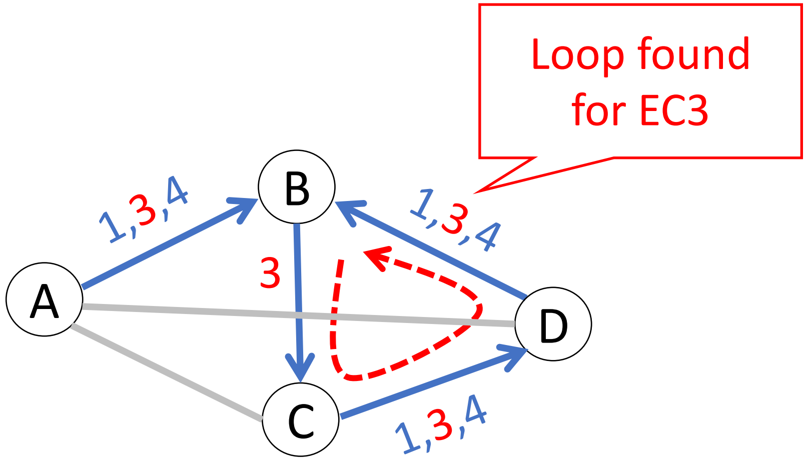
Update the ECs >> Update the model >> Check properties

IP header space



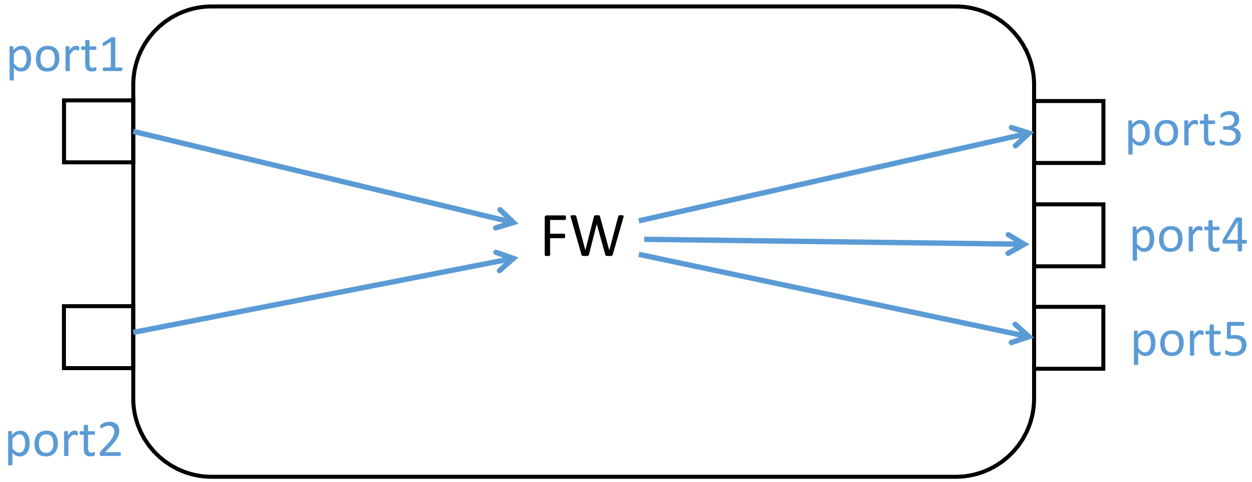


Data plane state



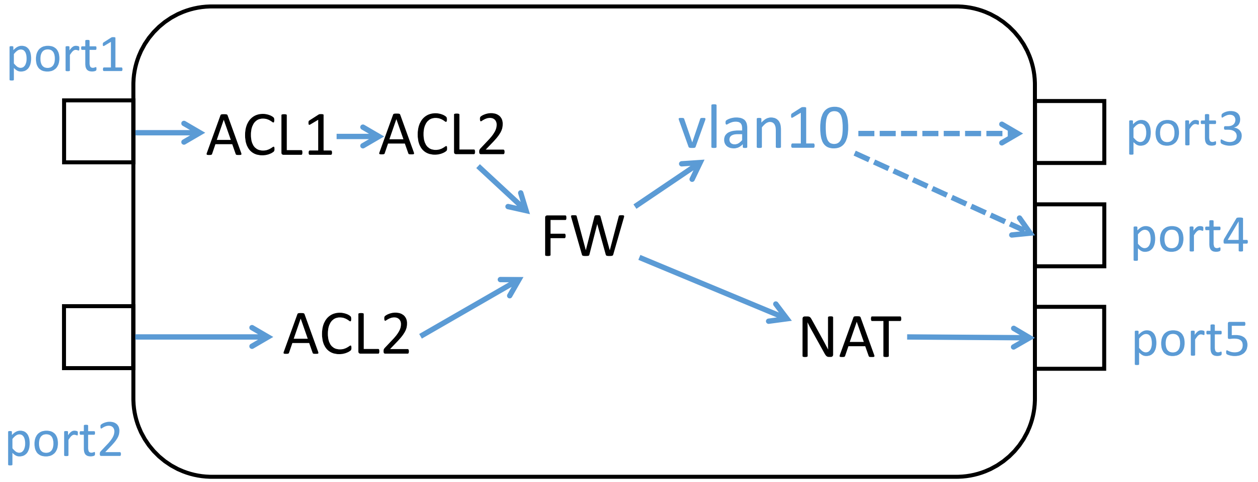
Network model

Realtime Verification for “Real” Networks



FW rules:
dstIP=192.168.0.0/16 port5
dstIP=192.168.10.0/24 VLAN10
… …

Realtime Verification for “Real” Networks



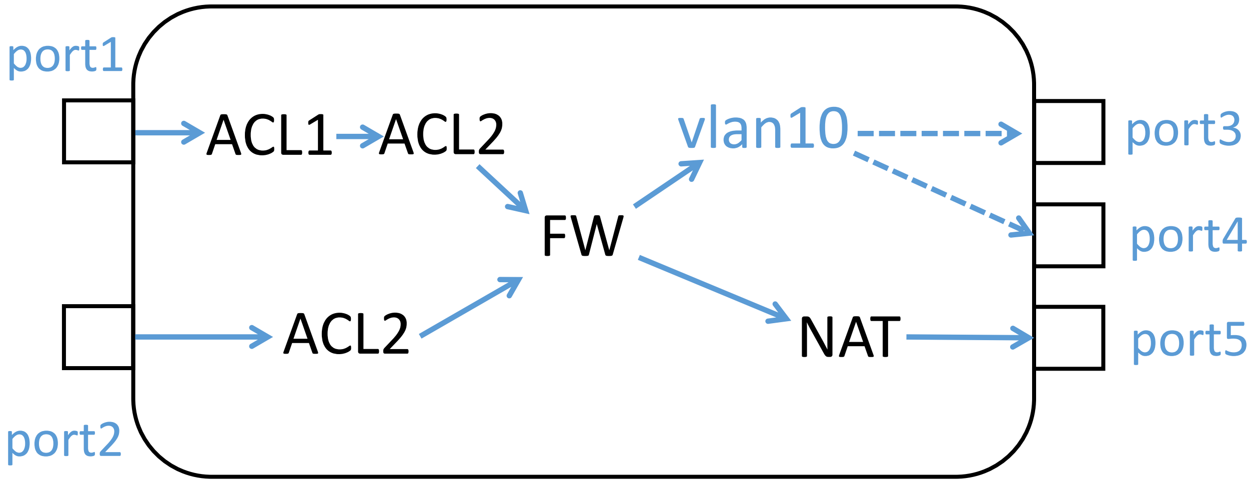
FW rules:
dstIP=192.168.0.0/16 port5
dstIP=192.168.10.0/24 VLAN10
… …

Various functionalities beyond forwarding

- filtering (ACL), rewriting (NAT), traffic policy, …

Requirement 1: Network model should be **expressive** of common functionalities

Realtime Verification for "Real" Networks



FW rules:
dstIP=192.168.0.0/16 port5
dstIP=192.168.10.0/24 VLAN10
… …

ACL1 rules:
dstIP=10.0.0.0/16 **dstPort**=22 permit
dstIP=10.0.1.0/24 **srcIP**=10.0.2.0/24 **dstPort**=80 deny
… …

Various functionalities beyond forwarding

- filtering (ACL), rewriting (NAT), traffic policy, …

Requirement 1: Network model should be **expressive** of common functionalities

Multiple fields other than IP prefix

- 5-tuples used by ACL, traffic policy, NAT, etc.

Requirement 2: Update of ECs should be **scalable** for multi-field rules

Scalability Issue due to Multi-Field Rules

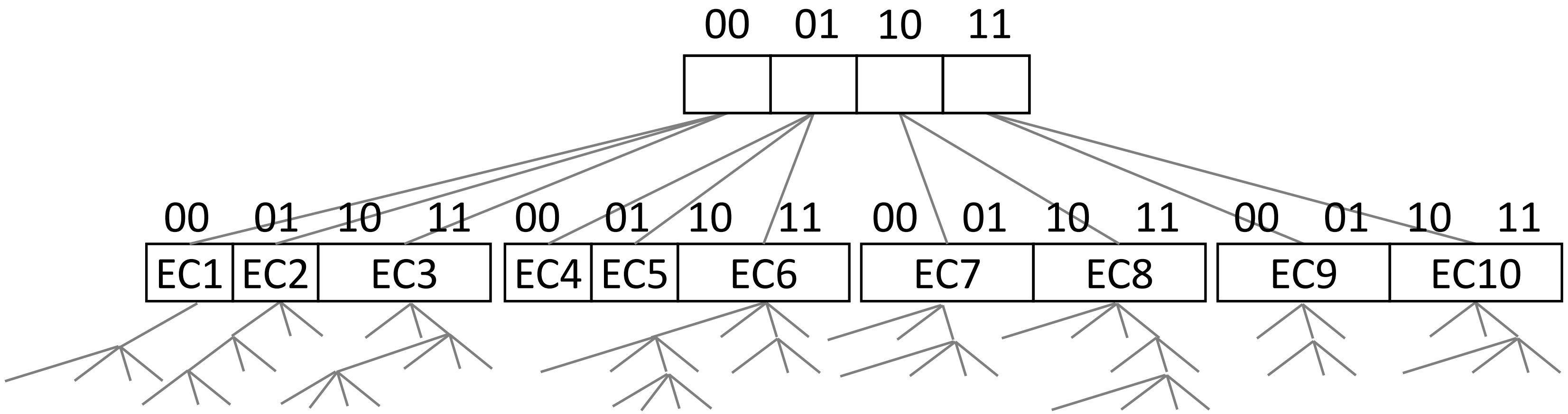
(1) ECs based on Ranges: fast for single-dimensional forwarding rules

Forwarding rules

R1. **dstIP**=00: forward port2
R2. **dstIP**=10: forward port2

ACL rules

R3. **dstIP**=0$*$, **dstPort**=0: deny
R4. **dstIP**=$**$, **dstPort**<2: permit



Network	#fw rules	#acl rules	**# of ECs**
Stanford	3.84×10^3	686	**15,100,968**
Purdue	3.52×10^6	2707	**>104,743,229**

Explosion of ECs

- Memory overflow
- Long verification time

Scalability Issue due to Multi-Field Rules

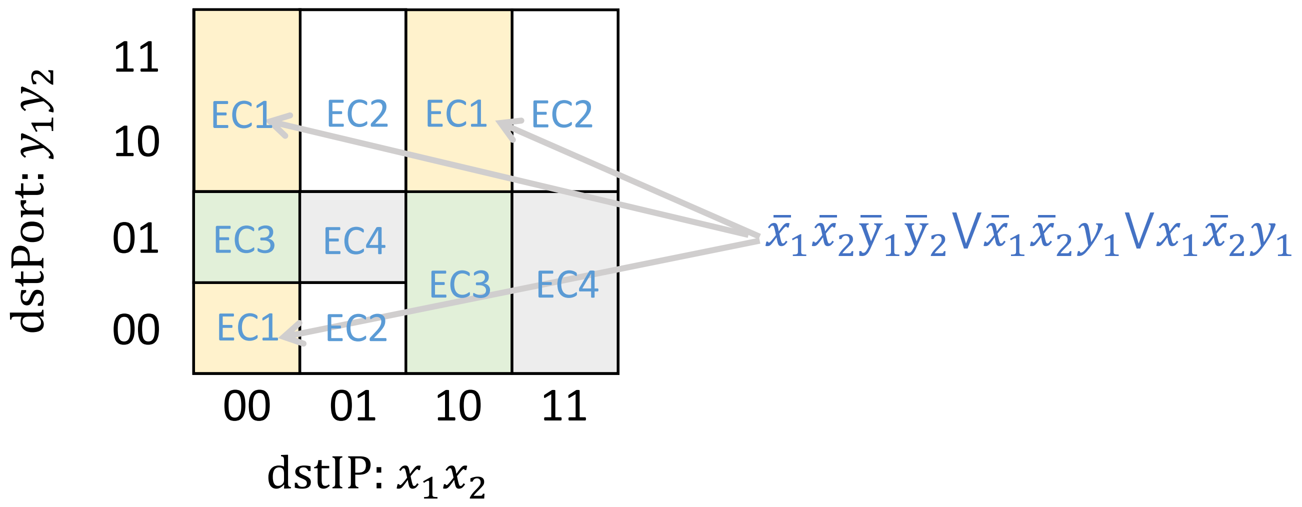
(2) ECs based on Atomic Predicates [AP Verifier, ICNP'13]: minimum # of ECs

Forwarding rules

R1. **dstIP**=00: forward port2
R2. **dstIP**=10: forward port2

ACL rules

R3. **dstIP**=0∗, **dstPort**=0: deny
R4. **dstIP**=∗∗, **dstPort**<2: permit



Network	#fw rules	#acl rules	# of ECs
Stanford	3.84×10^3	686	**515**
Purdue	3.52×10^6	2707	**4160**

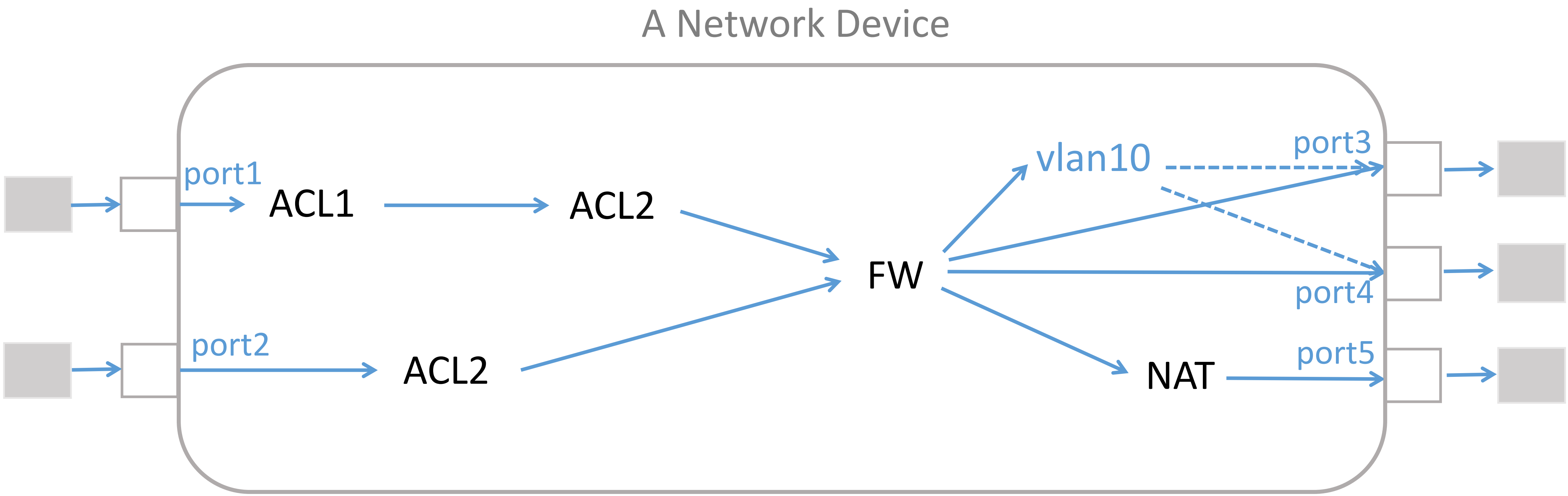
challenging to update atomic predicate fast

- An update potentially affects all atomic predicates
- Checking all atomic predicates is expensive (~10ms)

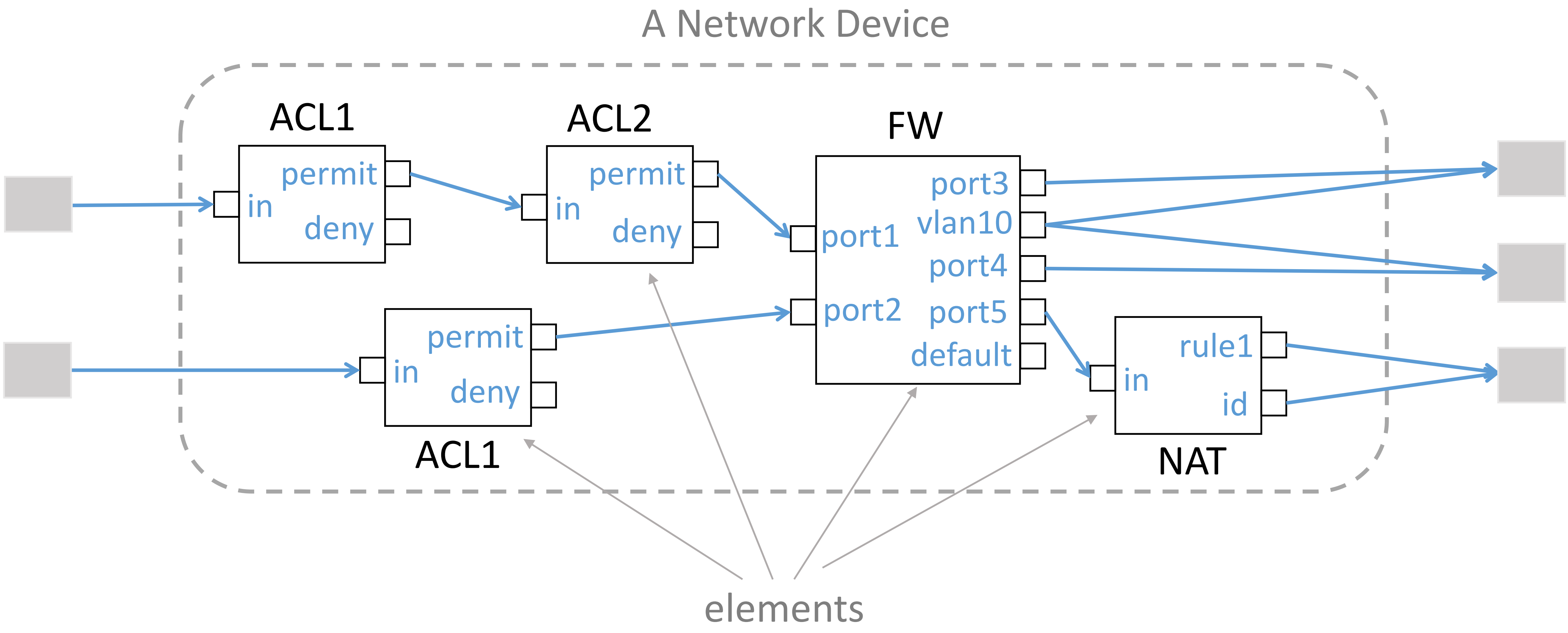
APKeep

- Modular Network Model
- Scalable Update of ECs

The Modular Network Model



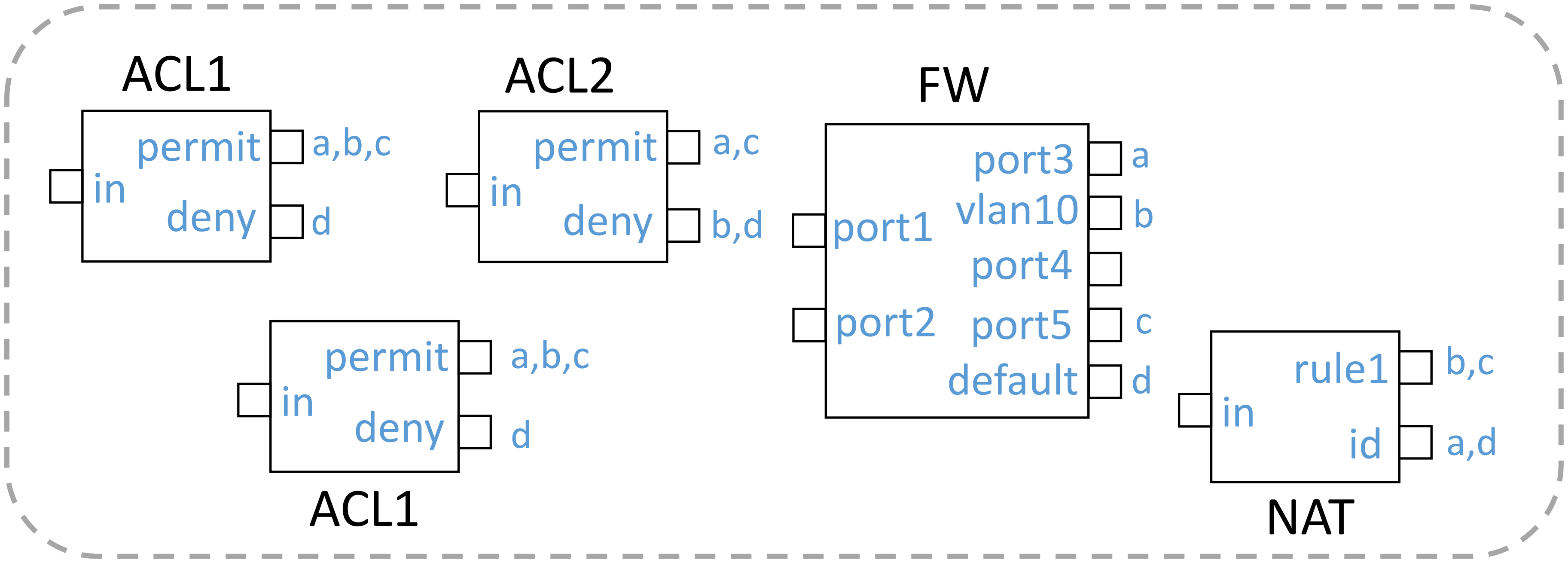
The Modular Network Model



APKeep

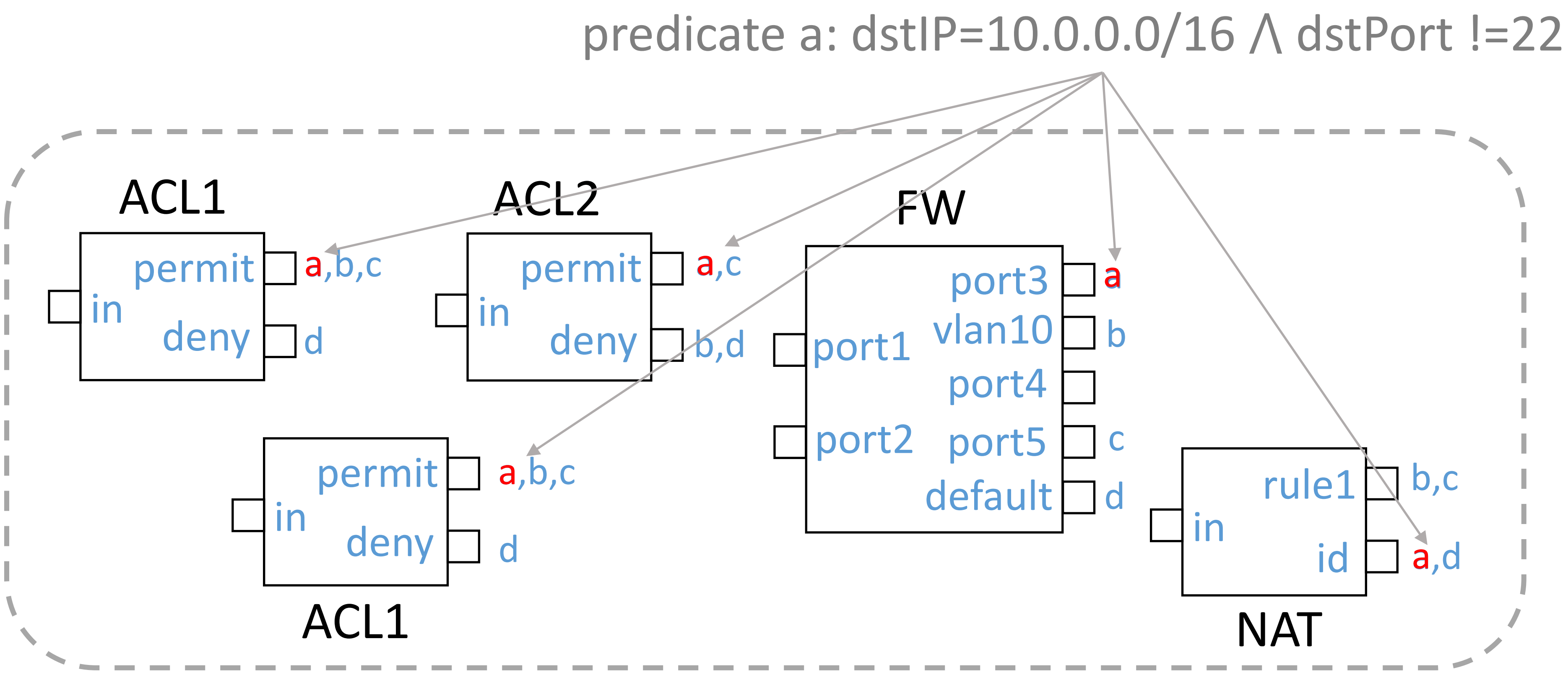
- Modular Network Model
- Scalable Update of ECs

Equivalence Class in Modular Network Model



The model supports general representation of EC

Equivalence Class in Modular Network Model

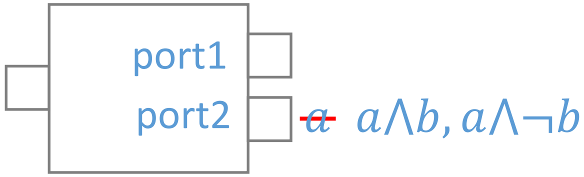


The model supports general representation of EC

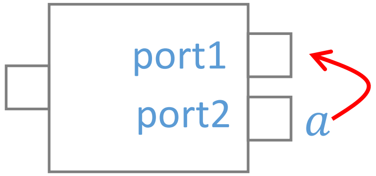
Fast Update of Minimum Number of ECs

APKeep **fast** updates **the minimum number** of ECs **(atomic predicates)** with three operations

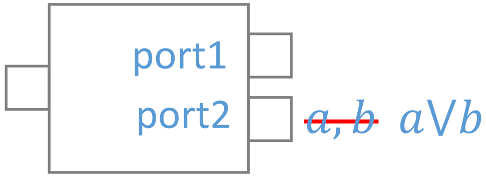
Split a predicate*



Transfer a predicate



Merge predicates



*Inspired by AP Verifier to compute atomic predicates

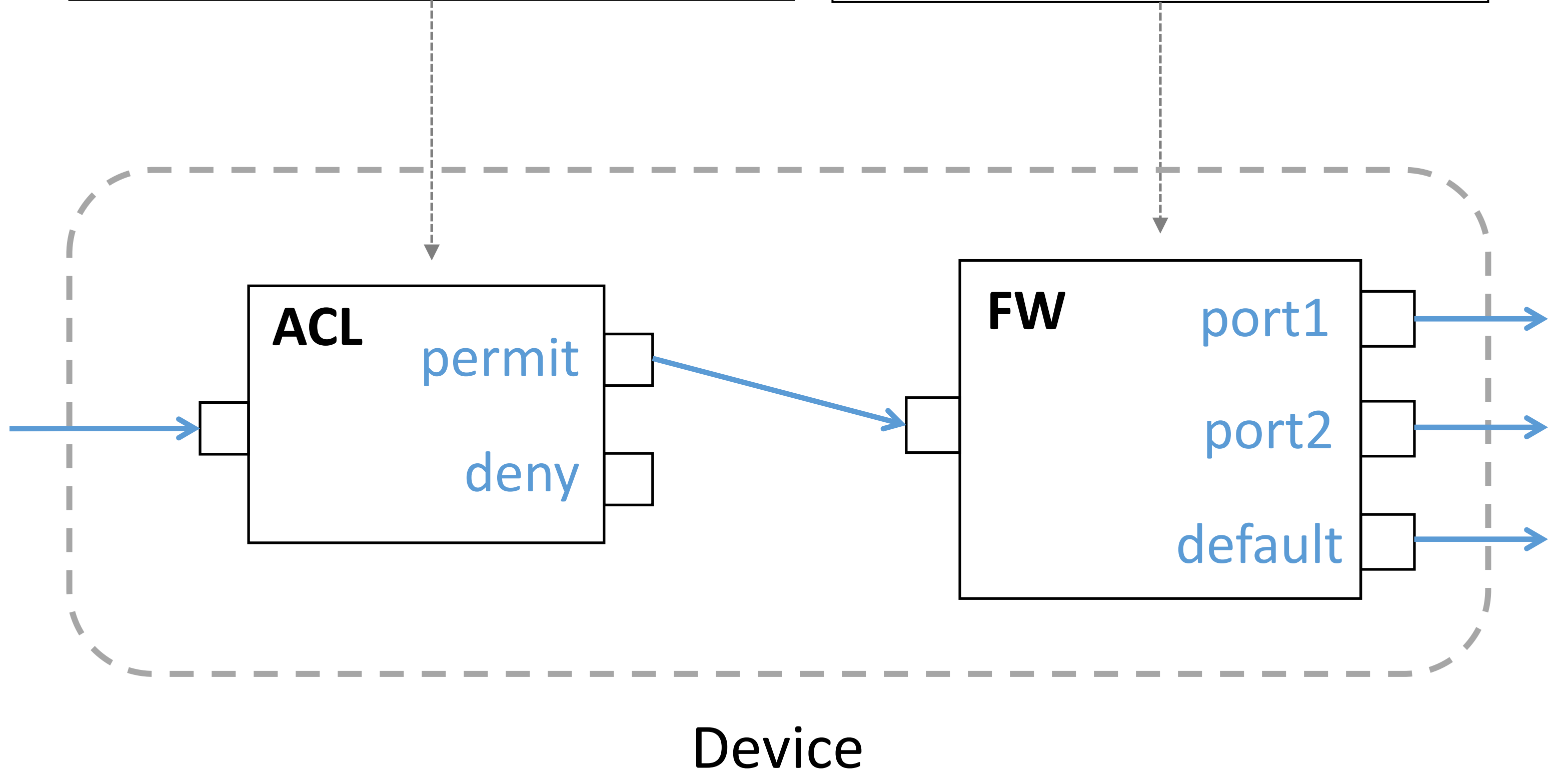
Example

ACL rules

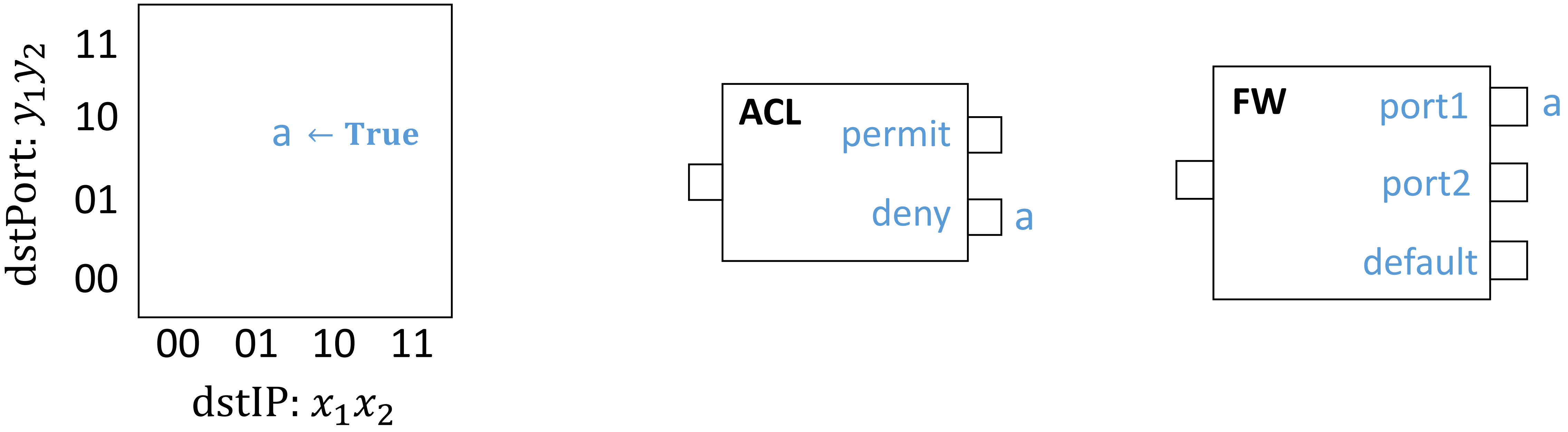
R1. dstIP=0*, dstPort=0: deny
R2. dstIP=**, dstPort<2: permit

Forwarding rules

R3. dstIP=00: forward port2
R4. dstIP=10: forward port2



Initial State without Rules

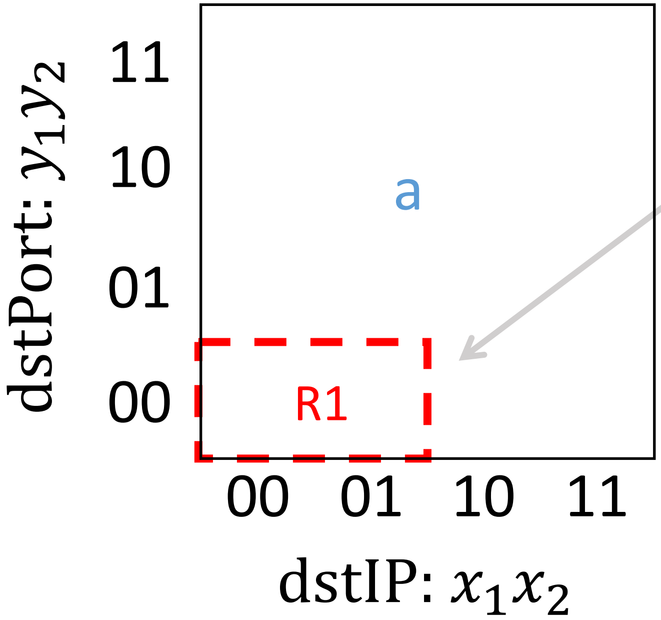


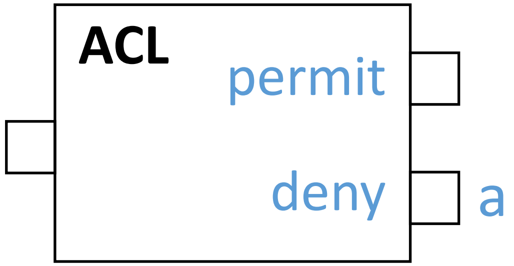
Initial State without Rules

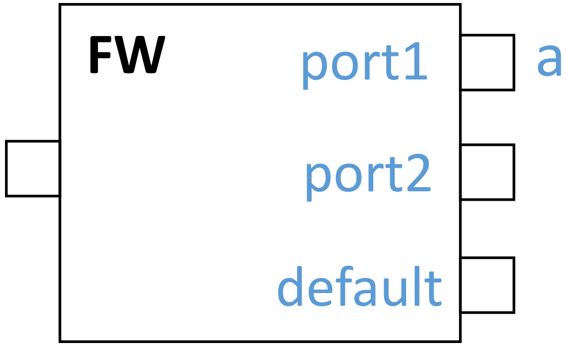
ACL rules

R1. dstIP=0∗, dstPort=0: deny

No behavior change:
no need to update



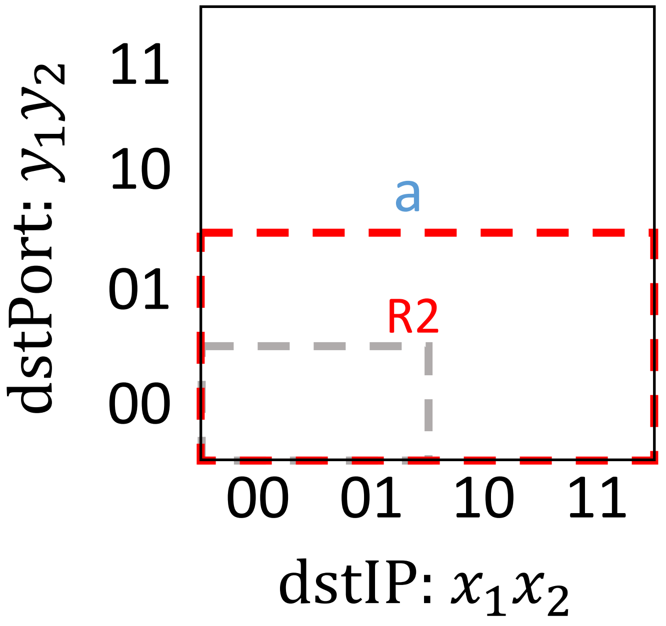


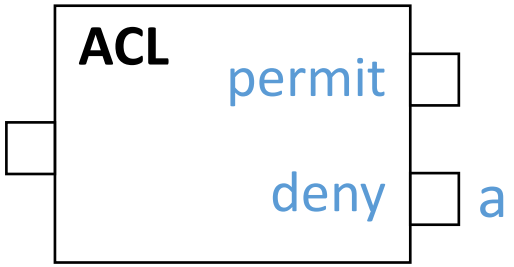


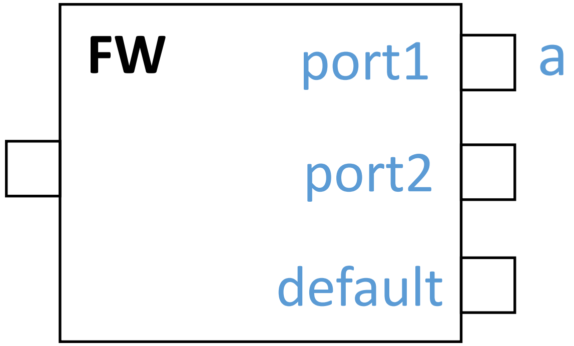
Splitting and Transferring Predicates

ACL rules

R1. dstIP=0∗, dstPort=0: deny
R2. dstIP=∗∗, dstPort<2: permit





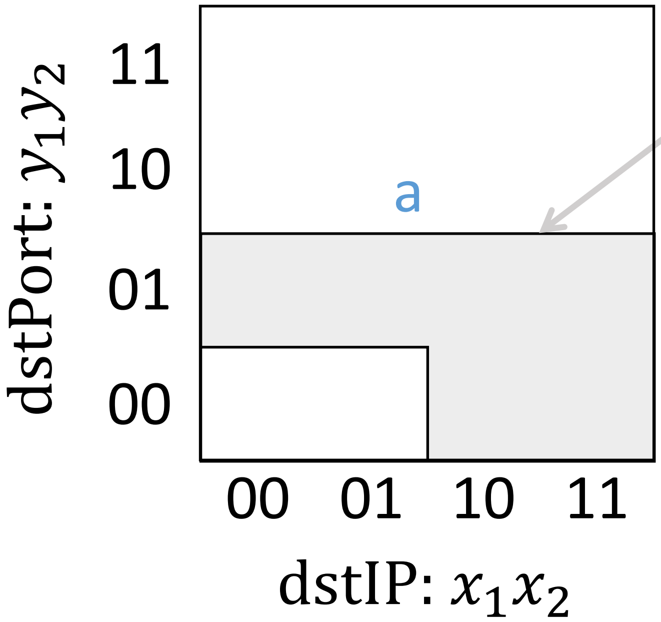


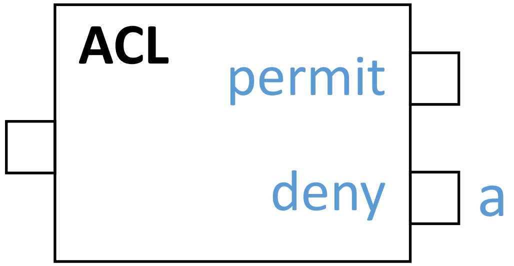
Splitting and Transferring Predicates

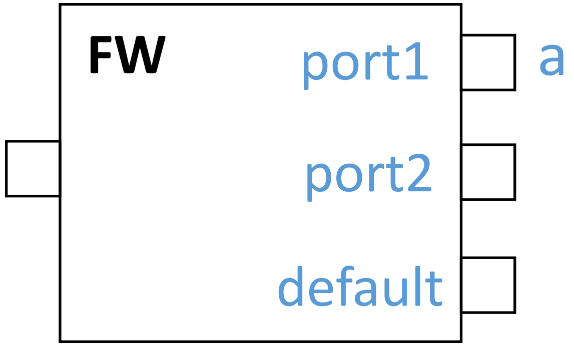
ACL rules

R1. dstIP=0∗, dstPort=0: deny
R2. dstIP=∗∗, dstPort<2: permit

Part of a changes behavior from *deny* to *permit*



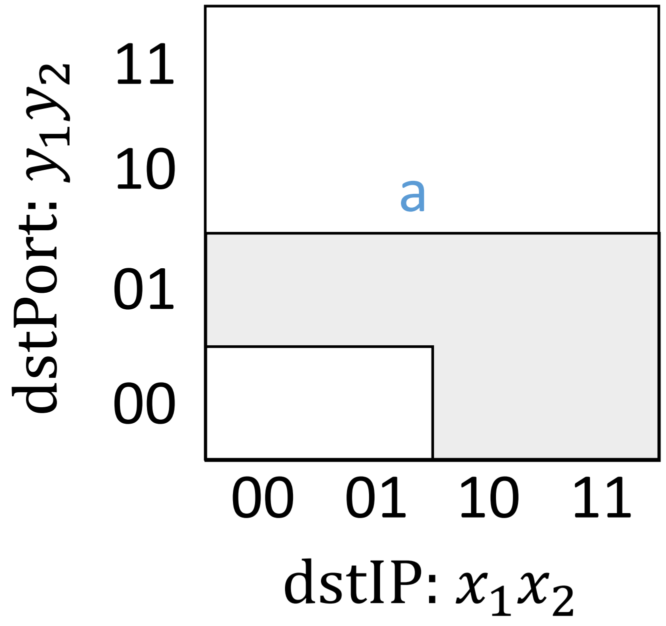


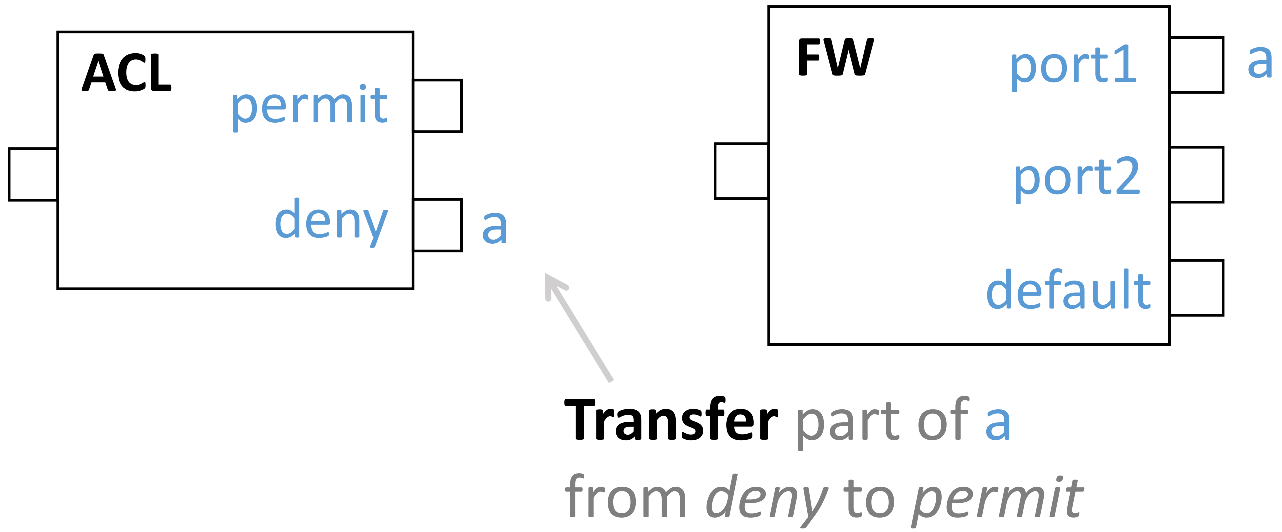


Splitting and Transferring Predicates

ACL rules

R1. dstIP=0∗, dstPort=0: deny
R2. dstIP=∗∗, dstPort<2: permit

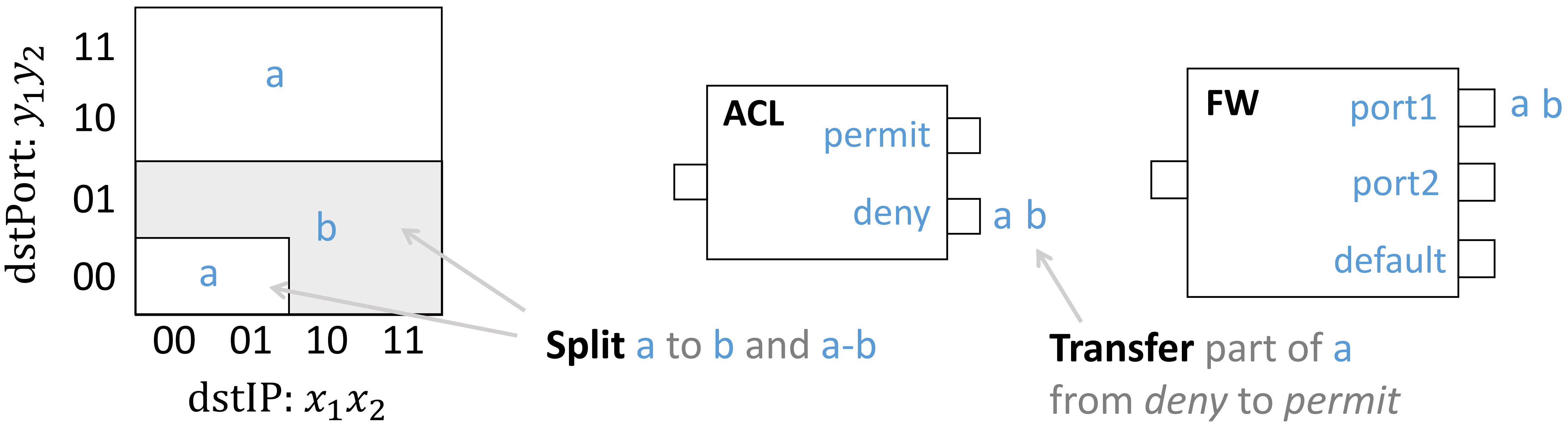




Splitting and Transferring Predicates

ACL rules

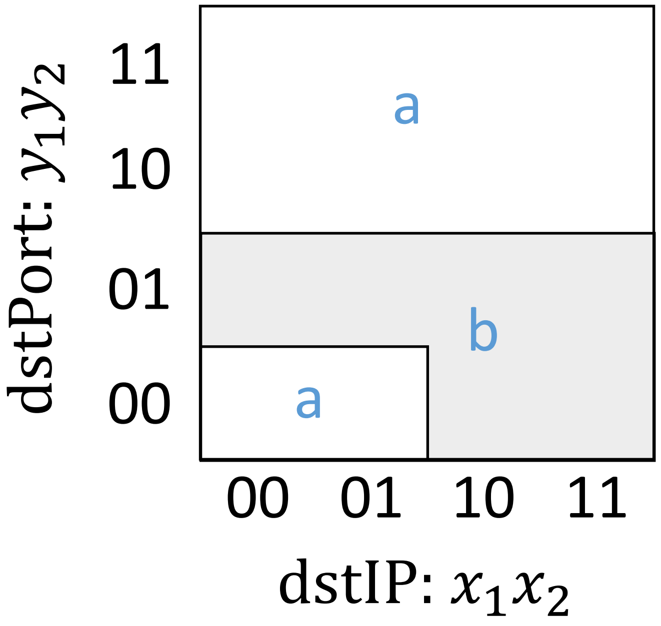
R1. dstIP=0∗, dstPort=0: deny
R2. dstIP=∗∗, dstPort<2: permit

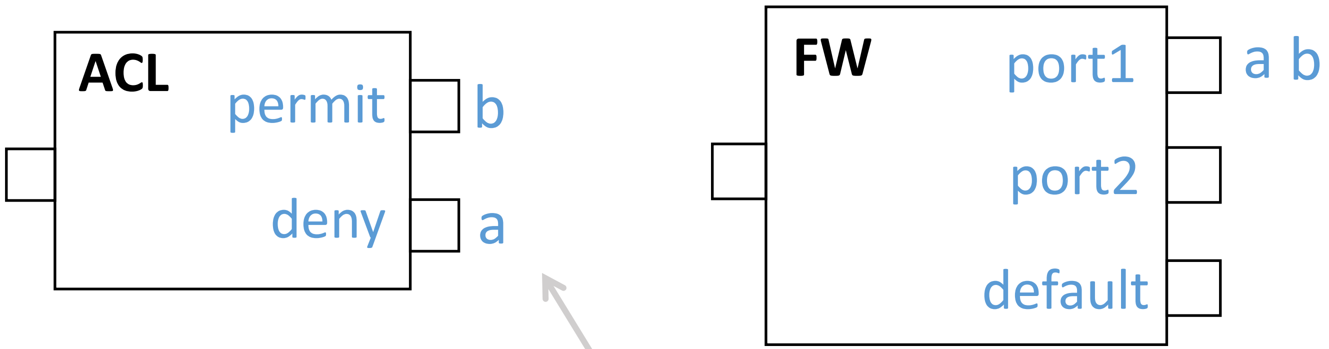


Splitting and Transferring Predicates

ACL rules

R1. dstIP=0∗, dstPort=0: deny
R2. dstIP=∗∗, dstPort<2: permit





Transfer part of a
from *deny* to *permit*

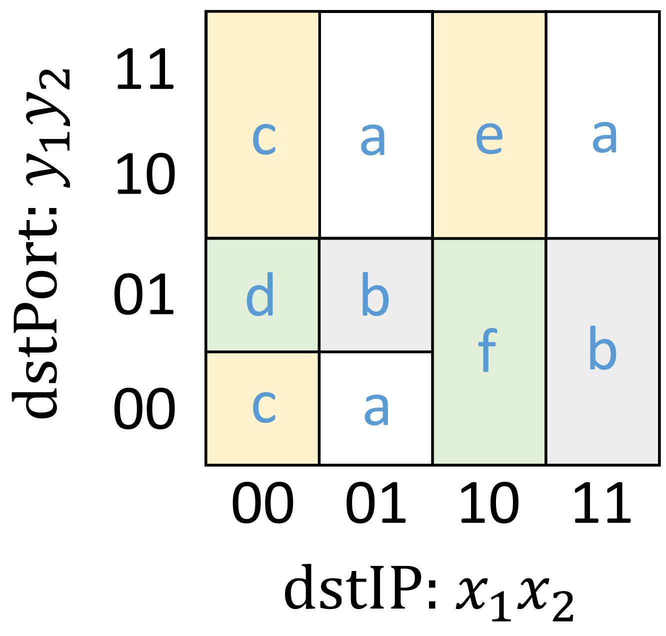
Merging Predicates

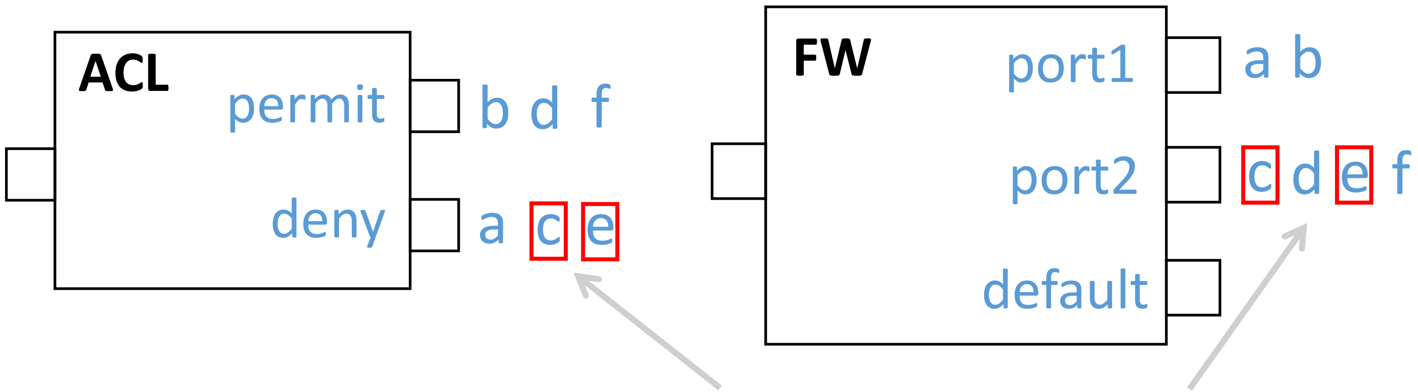
ACL rules

R1. dstIP=0∗, dstPort=0: deny
R2. dstIP=∗∗, dstPort<2: permit

Forwarding rules

R3. dstIP=00: forward port2
R4. dstIP=10: forward port2





c and e have the same forwarding behavior

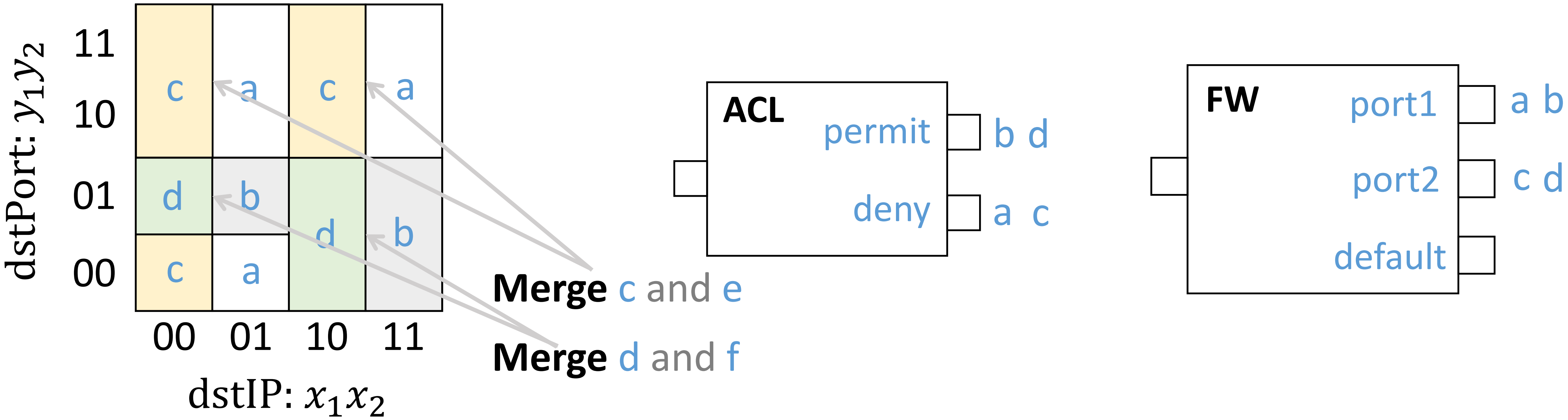
Merging Predicates

ACL rules

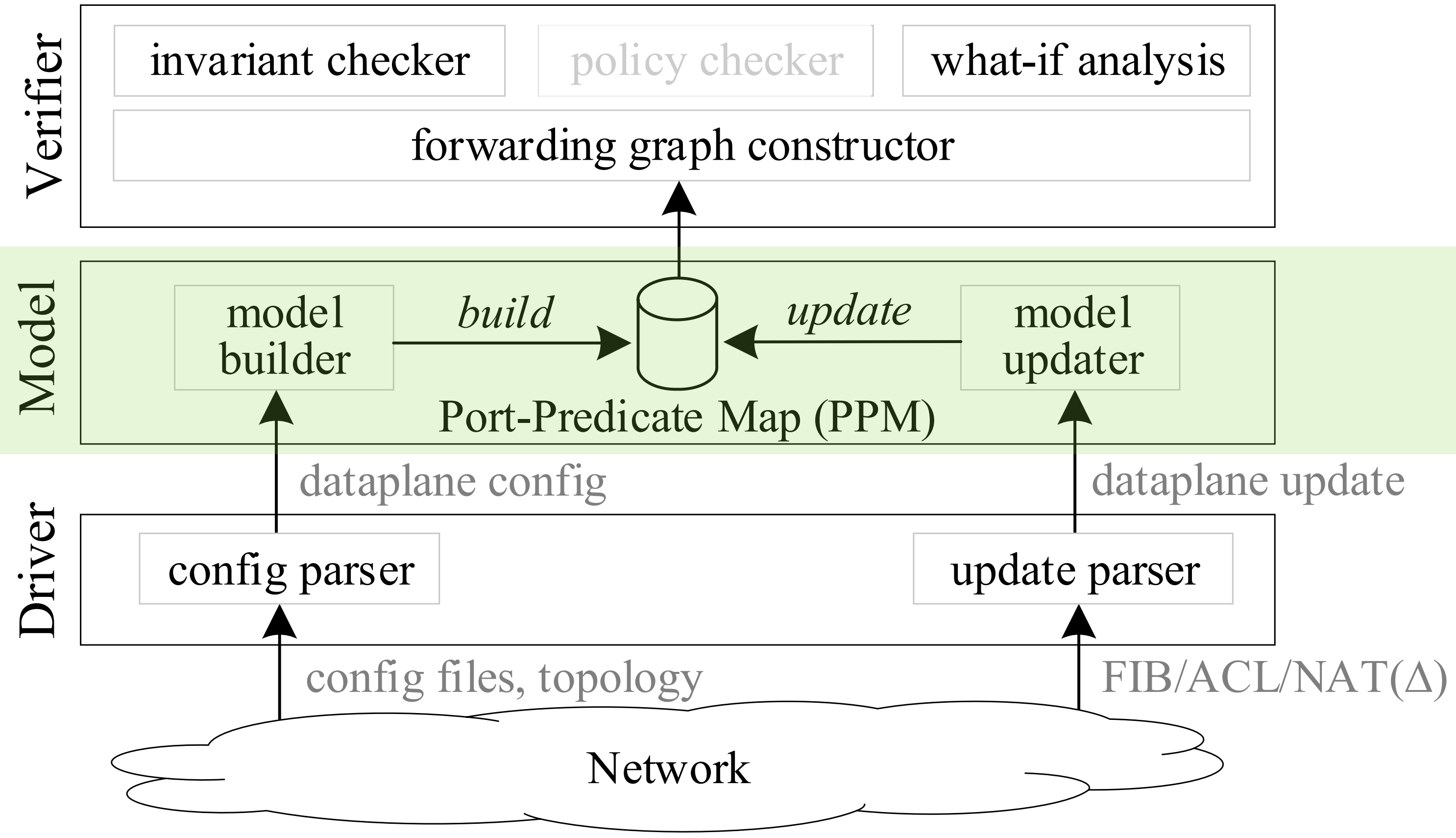
R1. dstIP=0*, dstPort=0: deny
R2. dstIP=**, dstPort<2: permit

Forwarding rules

R3. dstIP=00: forward port2
R4. dstIP=10: forward port2



System Implementation



Verification applications running on the model (policy checker still under development)

Using **AP Verifier** and **APT** to encode match fields and packet rewriting action with BDD (BDD library: **JDD**)

Parsing config files and rule updates into vendor-neutral format

AP Verifier and **APT** are open source, available at:
http://www.cs.utexas.edu/users/lam/NRL/Atomic_Predicates_Verifiers.html

Evaluation – Dataset

8 Datasets from Stanford, Internet2, Purdue, and Delta-net

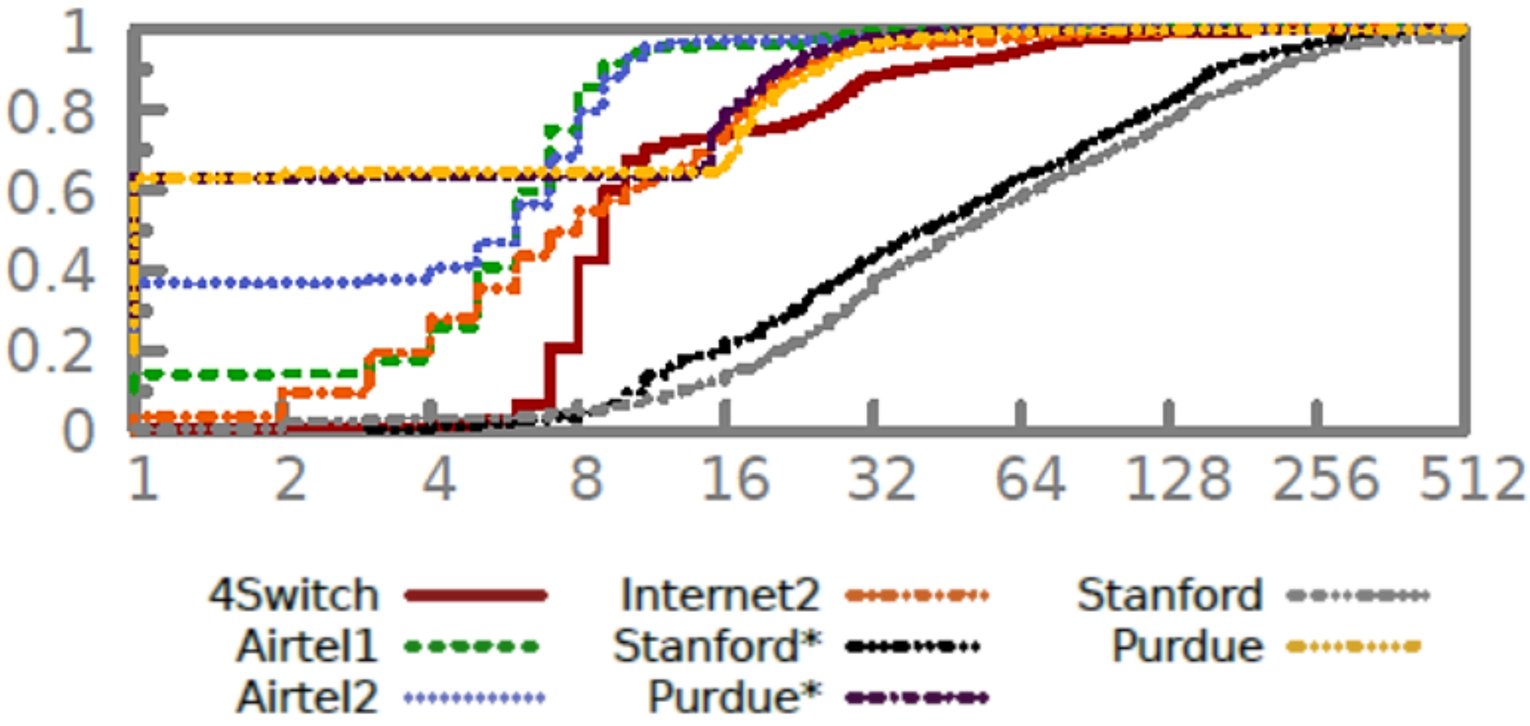
- 6 datasets with only IP forwarding rules
- 2 datasets with both IP forwarding rules and ACL rules

Network	Nodes	Links	Forwarding rules	ACL rules	Updates
Airtel1	68	260	6.89×10^4	0	1.42×10^7
Airtel2	68	260	9.84×10^4	0	5.05×10^8
4Switch	12	16	1.12×10^6	0	1.12×10^6
Internet2	9	56	1.26×10^5	0	2.52×10^5
Stanford*	16	74	3.84×10^3	0	7.68×10^3
Purdue*	1,646	3,094	3.52×10^6	0	7.04×10^6
Stanford	124	182	3.84×10^3	686	9.05×10^3
Purdue	2,159	3,607	3.52×10^6	2,707	7.05×10^6

IP forwarding rules only

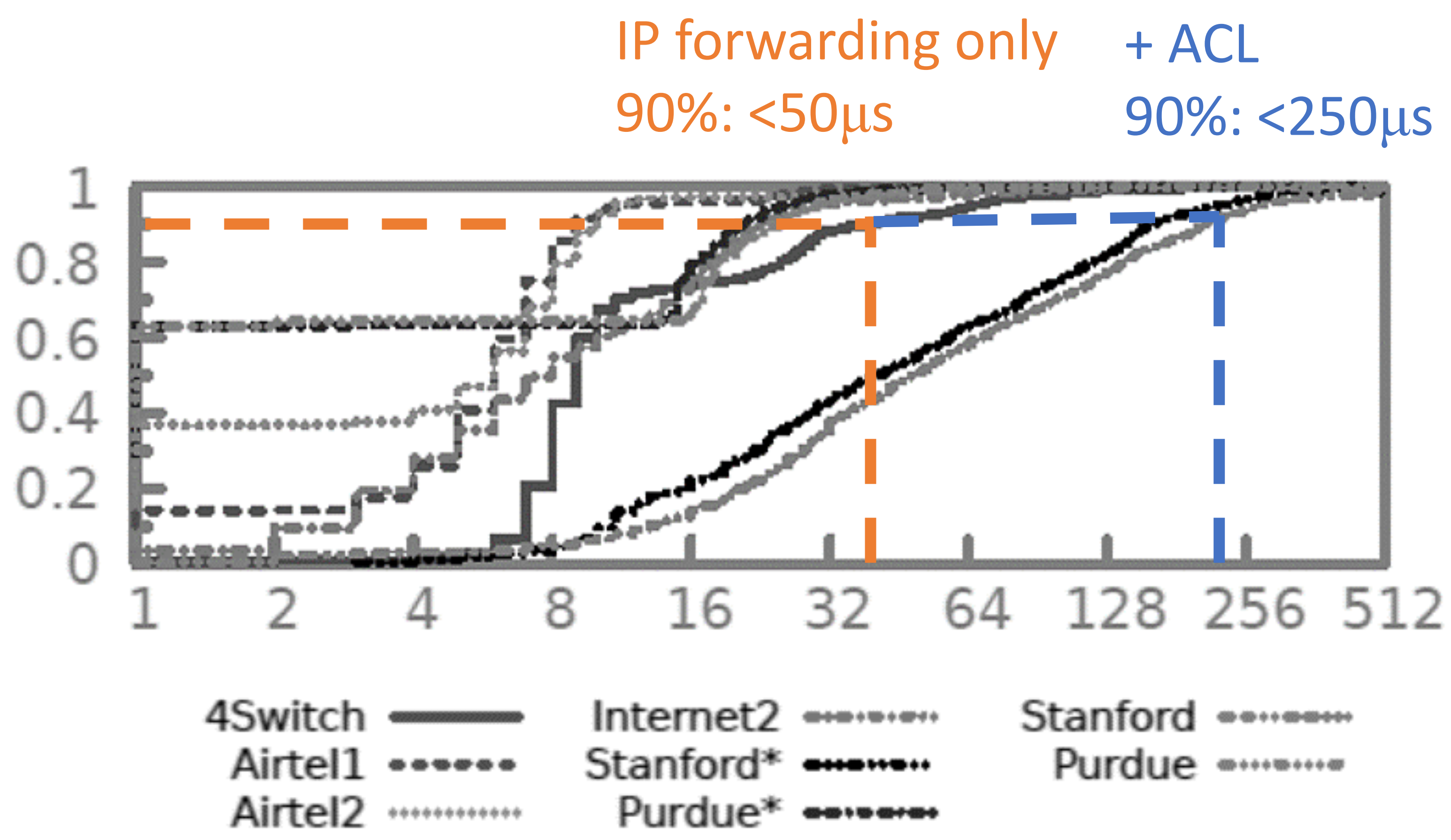
IP forwarding rules
+ ACL rules

Evaluation – Verification Speed



Verification: checking loops after each update. **Setting**: Linux desktop with 3.0GHz Intel Core i5 CPU and 32GB RAM

Evaluation – Verification Speed



Verification: checking loops after each update. **Setting**: Linux desktop with 3.0GHz Intel Core i5 CPU and 32GB RAM

Evaluation – Verification Speed

Our multi-field extension of Delta-net

Network	Average time (μs)					
	AP Verifier	VeriFlow	NetPlumber	Delta-net^{MF}	APKeep^{-}	APKeep
Airtel1	80	59	3,804	3	5	**7**
Airtel2	135	48	TO	4	4	**6**
4Switch	5,316	2,706	19,678	4	2,190	**21**
Internet2	1,660	144	2,123	3	9	**12**
Stanford*	1,953	468	8,700	9	98	**94**
Purdue*	777	648	MO	15	2	**9**
Stanford	2,072	4.8×10^6	9,532	MO	3.1×10^5	**127**
Purdue	TO	TO	MO	MO	MO	**13**

Timeout: >24 hours

Memory overflow: >32GB

Verification: checking loops after each update. **Setting**: Linux desktop with 3.0GHz Intel Core i5 CPU and 32GB RAM

Conclusion

APKeep: checking correctness of data plane with **real** devices in **real** time

- Modular network model: expressive and extensible for real network devices
- Scalable update of ECs: fast updating the minimum number of ECs (<1ms)

Future work

- Checking operator intent beyond reachability
- Parallelizing the update of predicates

Thanks for your attention

Peng Zhang
p-zhang@xjtu.edu.cn