Conflict Detection in Software-Defined Networks

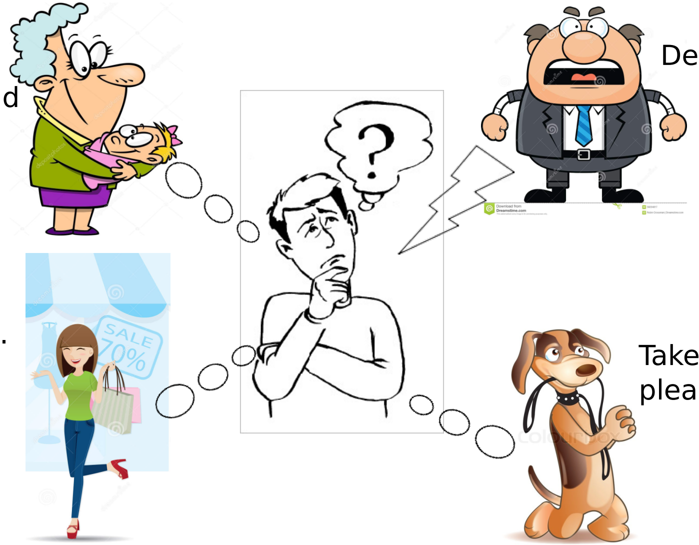
Author 1
author1@campus.lmu.de

Author 2
mnm-team.org/∼author2

Aufgabensteller: Prof. Dr. Dieter Kranzlmüller
Betreuer 1: 1. Supervisor
Betreuer 2: 2. Supervisor

July 21, 2022

Conflicts in Real Life



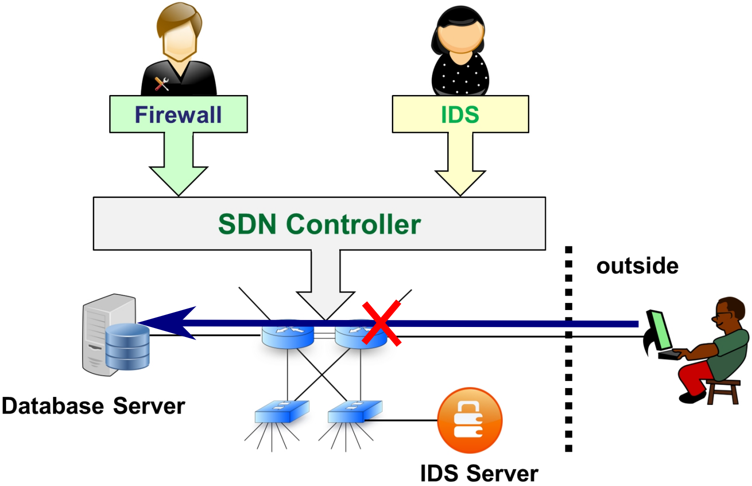
I want a grandchild

Deadline!

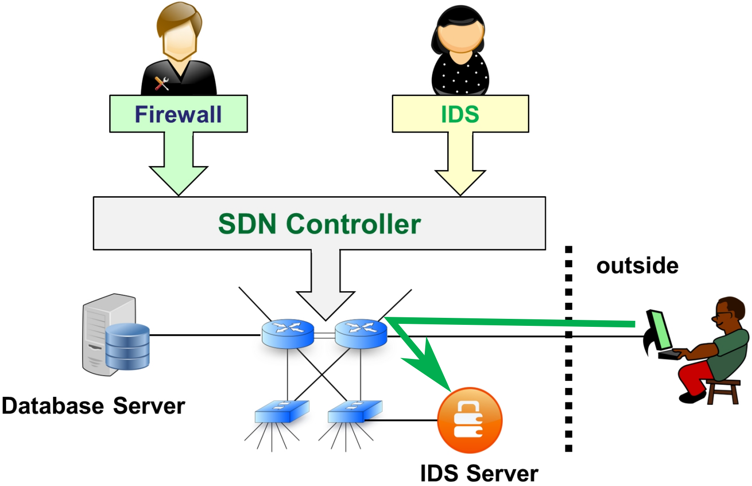
...

Take a walk, please

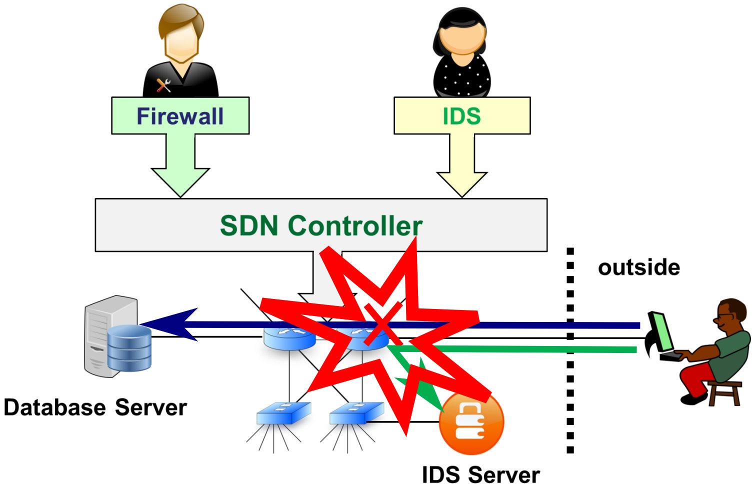
Conflicts in SDN



Conflicts in SDN



Conflicts in SDN

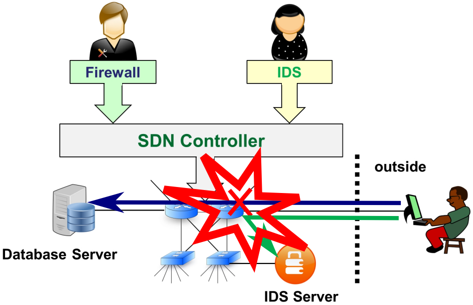


Conflicts in SDN

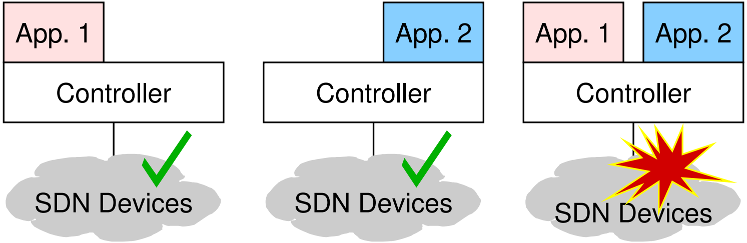
Possible consequences:

- Application's goals are not fulfilled
- Unexpected, unreliable network behaviour

⇒ Conflicts need to be detected and resolved



Conflict Definition

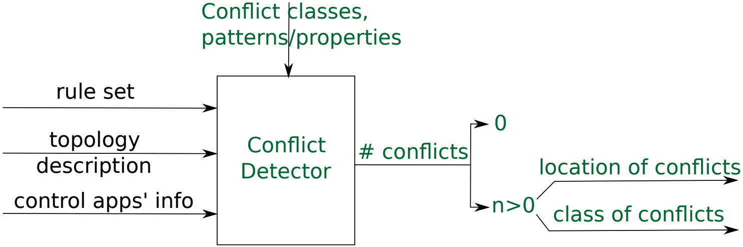


Research Questions

1. What is a suitable method to research conflicts in SDN?
2. How can conflicts between control applications be classified based on their rules (conflict classification)?

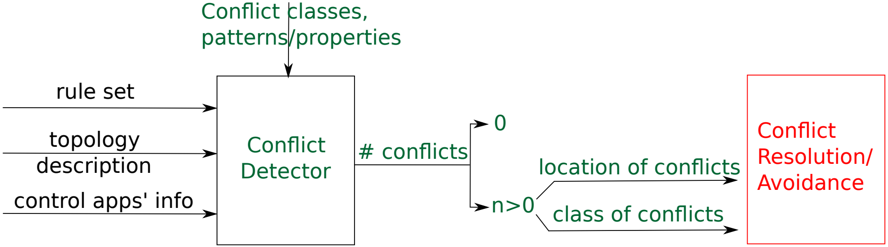
Research Questions

1. What is a suitable method to research conflicts in SDN?
2. How can conflicts between control applications be classified based on their rules (conflict classification)?
3. How many conflicts exist in a given rule set (conflict detection)?
   3.1 Which rules cause conflicts?
   3.2 To which class does each detected conflict belong?



Research Questions

1. What is a suitable method to research conflicts in SDN?
2. How can conflicts between control applications be classified based on their rules (conflict classification)?
3. How many conflicts exist in a given rule set (conflict detection)?
   3.1 Which rules cause conflicts?
   3.2 To which class does each detected conflict belong?



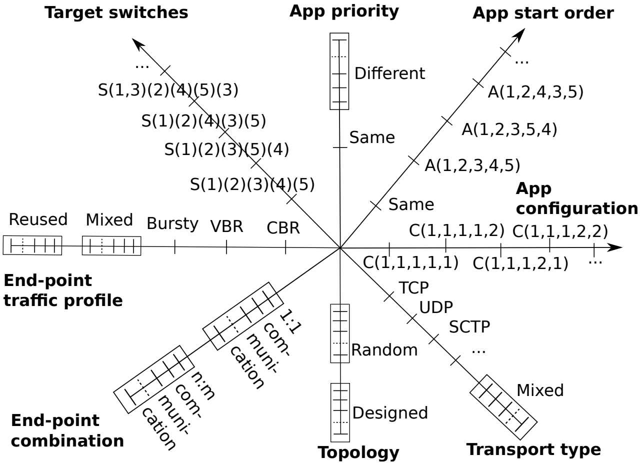
State-of-the-art

- Related work 1
- Related work 2
- . . .

Approach

1. Analytical approach
2. Experimental approach

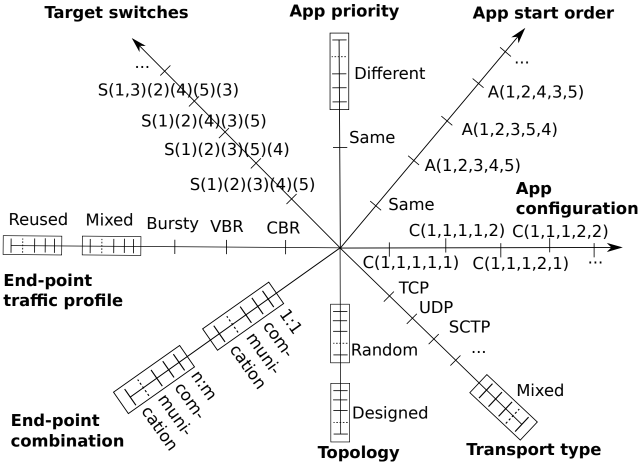
Space for Experiments



Control applications:

- Shortest Path First Routing (SPF)
- End-point Load Balancer (EpLB)
- Path Load Balancer (PLB)
- Firewall (FW)
- . . .

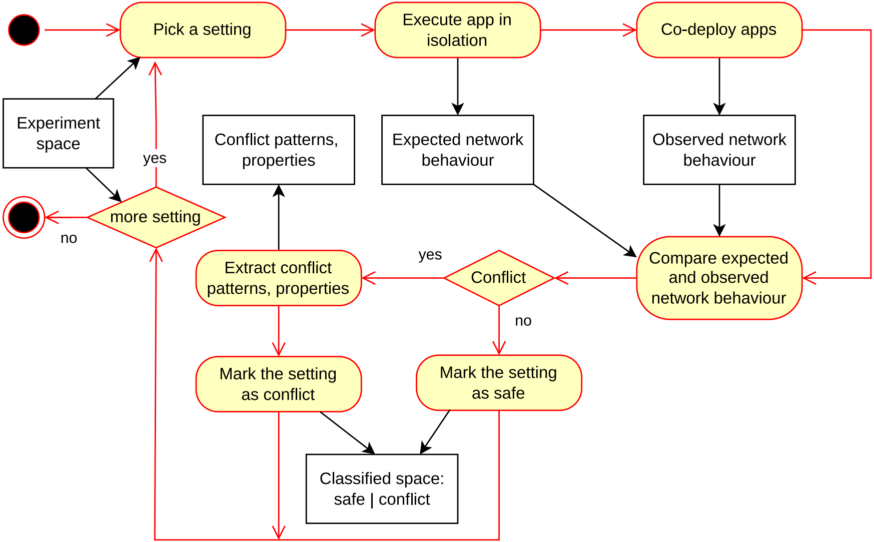
Space for Experiments



The number of experiments is immense

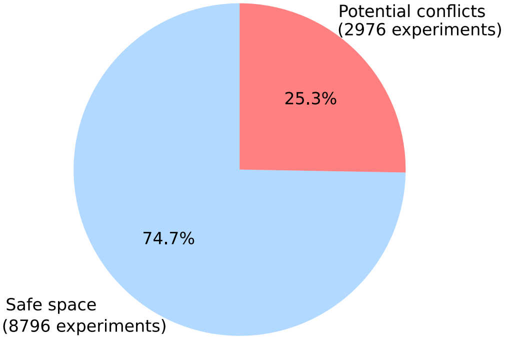
⇒ **restrict the space size and automate experiments**

Methodology for Experiments


Pick a setting
Execute app in isolation
Co-deploy apps
Experiment space
yes
Conflict patterns, properties
Expected network behaviour
Observed network behaviour
more setting
no
yes
Extract conflict patterns, properties
Conflict
Compare expected and observed network behaviour
no
Mark the setting as conflict
Mark the setting as safe
Classified space: safe | conflict

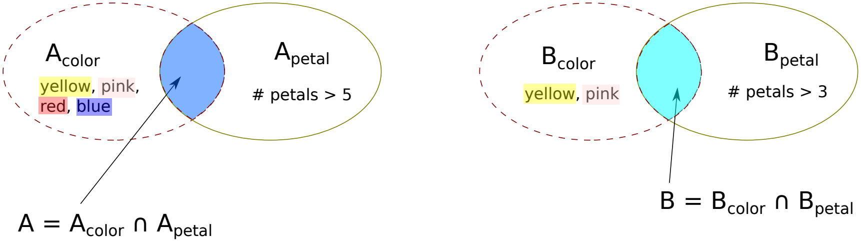
Explored Subspaces

# Topologies	12
# Applications	14
App. configuration	1 → 5
App. start order	same and different
App. priority	same and different
Target switches	1 → all
Ep. Traffic Profile	CBR and VBR
EP. Combination	unicast, multicast
Transport type	TCP, UDP
# Experiments	11,772



Dataset is available at
https://github.com/mnm-team/sdn-conflicts

Comparing Multi-Property Sets using $\cdot r$



A_{color}

yellow, pink, red, blue

A_{petal}

petals > 5

$A = A_{color} \cap A_{petal}$

B_{color}

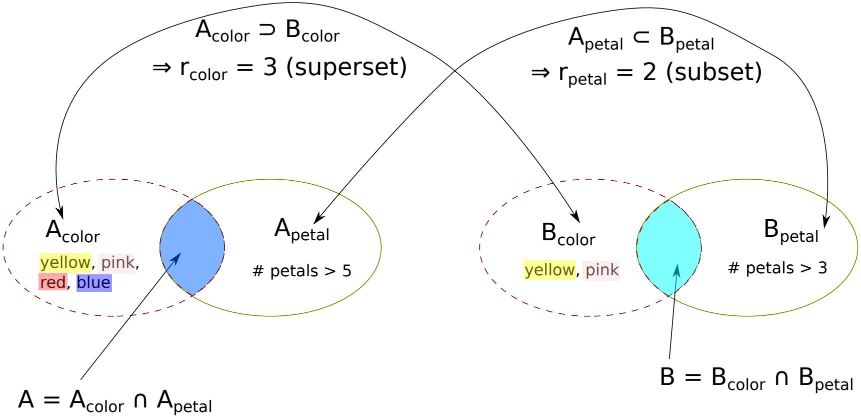
yellow, pink

B_{petal}

petals > 3

$B = B_{color} \cap B_{petal}$

Comparing Multi-Property Sets using $\cdot r$



$A_{\text{color}} \supset B_{\text{color}}$

$\Rightarrow r_{\text{color}} = 3$ (superset)

$A_{\text{petal}} \subset B_{\text{petal}}$

$\Rightarrow r_{\text{petal}} = 2$ (subset)

A_{color}

yellow, pink, red, blue

A_{petal}

petals > 5

B_{color}

yellow, pink

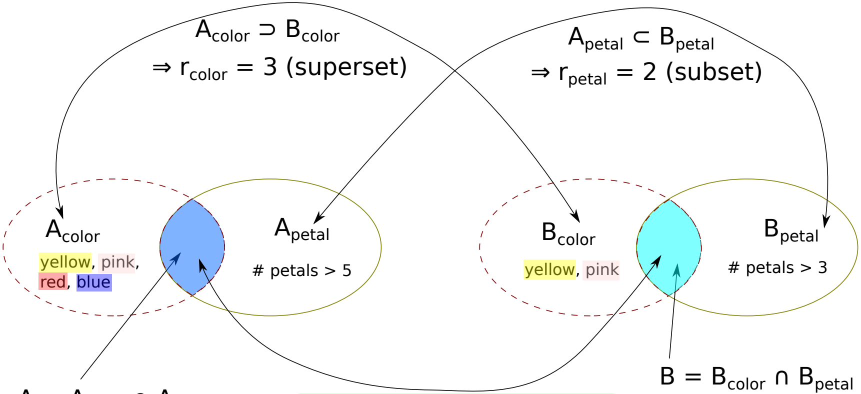
B_{petal}

petals > 3

$A = A_{\text{color}} \cap A_{\text{petal}}$

$B = B_{\text{color}} \cap B_{\text{petal}}$

Comparing Multi-Property Sets using $\cdot_r$



$A_{\text{color}} \supset B_{\text{color}}$

$\Rightarrow r_{\text{color}} = 3$ (superset)

$A_{\text{petal}} \subset B_{\text{petal}}$

$\Rightarrow r_{\text{petal}} = 2$ (subset)

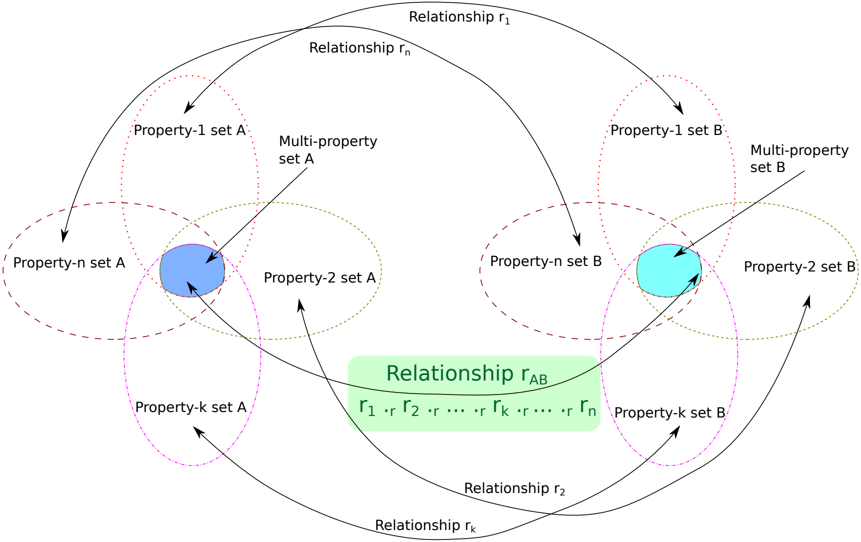
$A = A_{\text{color}} \cap A_{\text{petal}}$

$B = B_{\text{color}} \cap B_{\text{petal}}$

$r_{AB}=r_{\text{color}} \cdot_r r_{\text{petal}} = 3 \cdot_r 2 = 4$

$\Rightarrow$ A intersects B

Comparing Multi-Property Sets using $\cdot r$



Relationship r_1

Relationship r_n

Property-1 set A

Multi-property set A

Property-n set A

Property-2 set A

Property-k set A

Property-1 set B

Multi-property set B

Property-n set B

Property-2 set B

Property-k set B

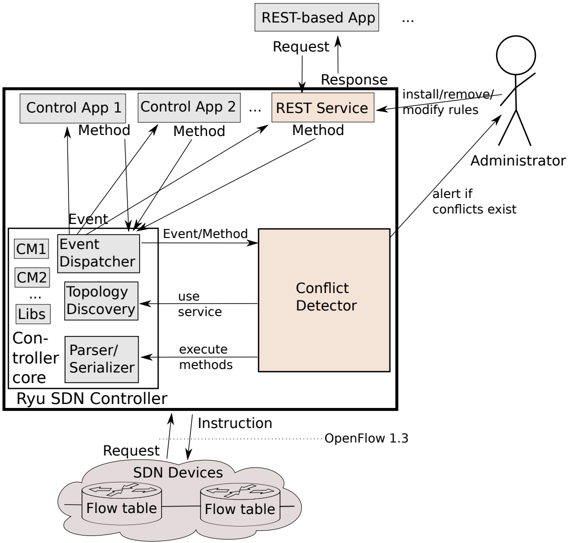
Relationship r_{AB}

$r_1 \cdot_r r_2 \cdot_r \dots \cdot_r r_k \cdot_r \dots \cdot_r r_n$

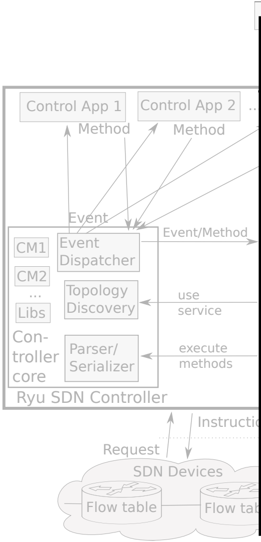
Relationship r_2

Relationship r_k

Conflict Detection Prototype



Conflict Detection Prototype



Conflict Detector

+ rule_database
+ local_conflict_rules
+ distributed_conflict_rules
+ hidden_conflict_rules
+ rule_graph
...

+ detect_distributed_conflicts()
+ detect_hidden_conflicts()
+ build_rule_graph()
...

Detected Results in Designed Cases

Rules are deployed with known conflicts

Conflicts detected by the prototype are then controlled manually

Results for both MWN and Stanford topologies:

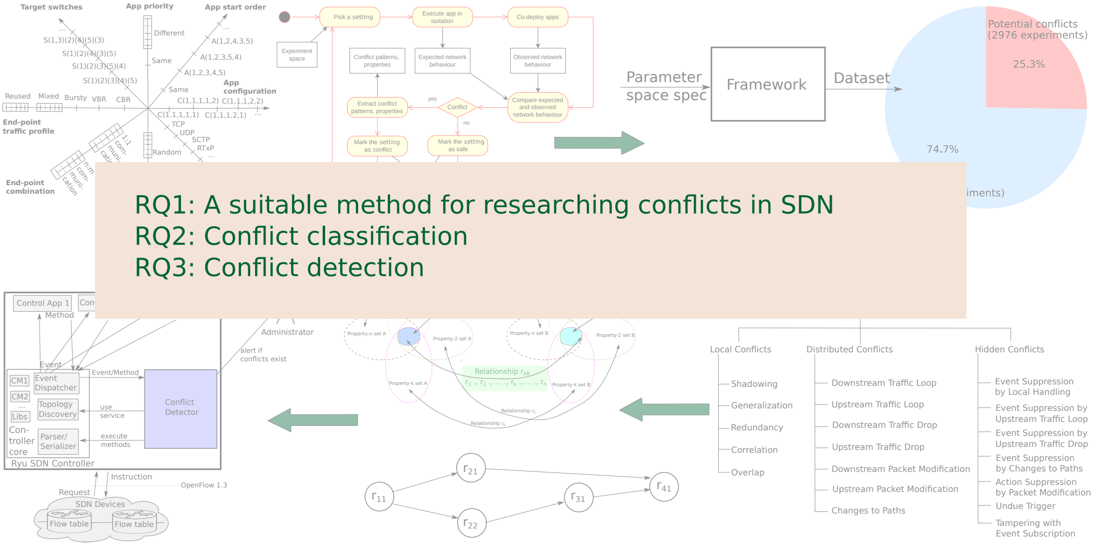
Test	Local conflicts					Traffic Loop	Traffic Drop	Hidden conflicts ESLH
	Shadowing	**Generalization**	**Redundancy**	**Correlation**	**Overlap**			
1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3
4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5

detected by the prototype / designed

ESLH: Event Suppression by Local Handling

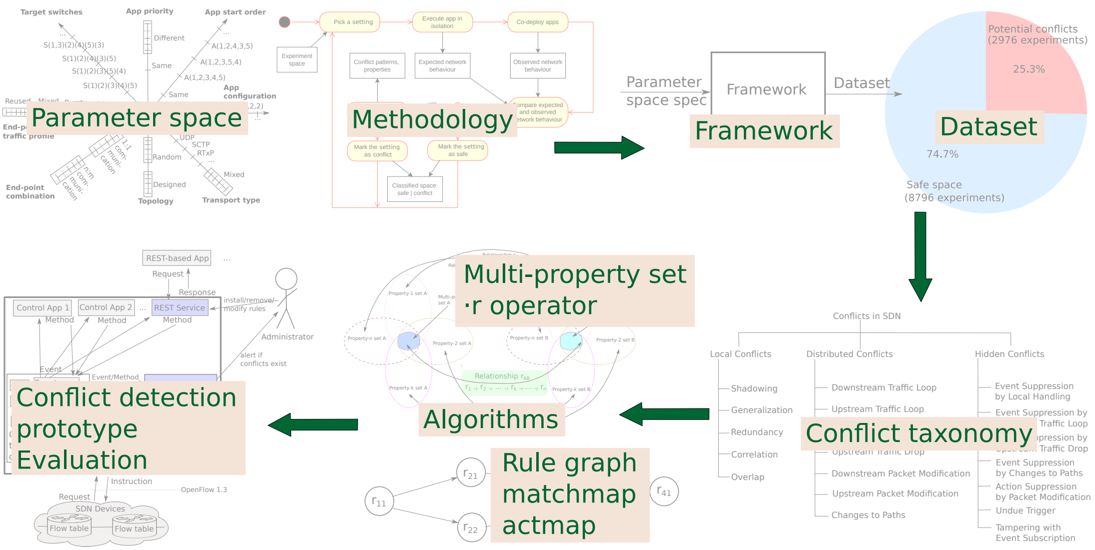
⇒ All conflicts are precisely identified

Conclusions



RQ1: A suitable method for researching conflicts in SDN
RQ2: Conflict classification
RQ3: Conflict detection

Conclusions



Parameter space

Methodology

Framework

Dataset

Conflict taxonomy

Multi-property set ·r operator

Algorithms

Rule graph matchmap actmap

Conflict detection prototype Evaluation

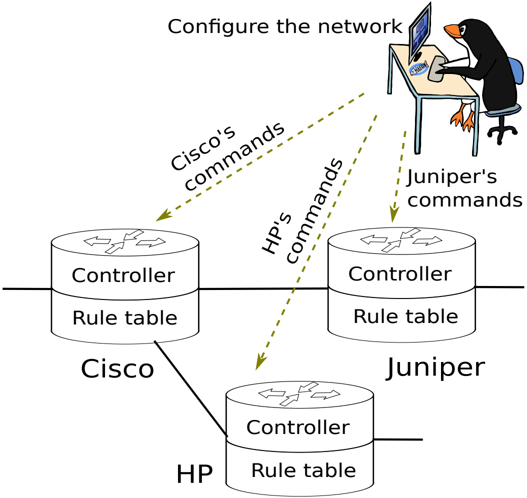
Prospects

- Future work 1
- Future work 2
- ...

Backup

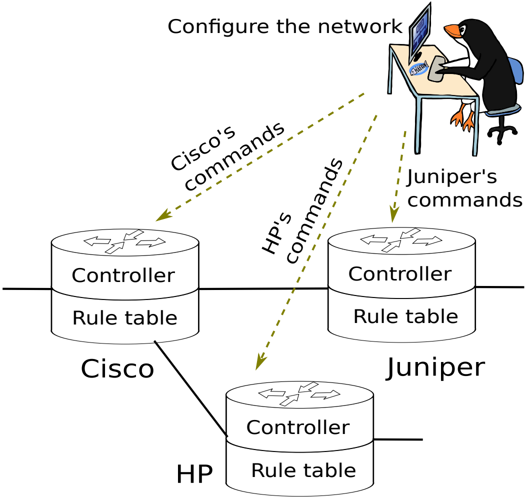
Traditional Networks and SDN

Traditional networks



Traditional Networks and SDN

Traditional networks



Configure the network

Cisco's commands

HP's commands

Juniper's commands

Controller

Rule table

Cisco

Controller

Rule table

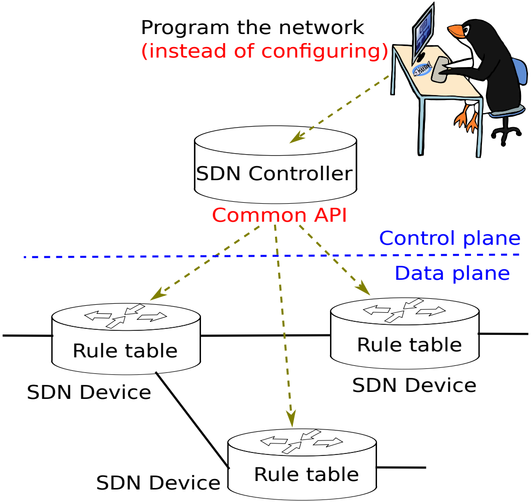
Juniper

Controller

Rule table

HP

Software-Defined Networks (SDN)



Program the network
(instead of configuring)

SDN Controller

Common API

Control plane

Data plane

Rule table

SDN Device

Rule table

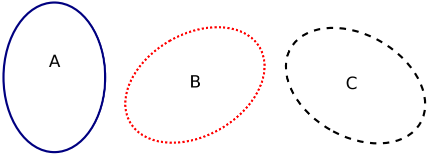
SDN Device

SDN Device

Rule table

Multi-Property Set

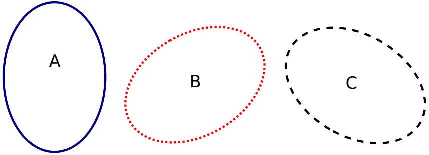
A = a set of flowers having **five petals**
B = a set of flowers with **red color**
C = a set of flowers being **scentless**



Question: S_{ABC} = a set of flowers having **five petals, red color** and being **scentless** = ?

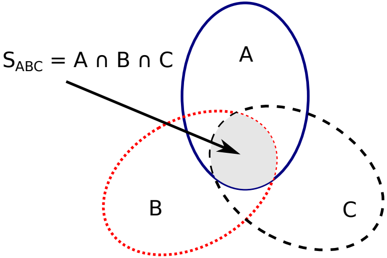
Multi-Property Set

A = a set of flowers having **five petals**
B = a set of flowers with **red color**
C = a set of flowers being **scentless**



Question: S_{ABC} = a set of flowers having **five petals, red color** and being **scentless** = ?

Answer: $S_{ABC} = A \cap B \cap C$



Match fields of SDN rules are multi-property sets, e.g.,
match={ip_src=192.168.1.1, ip_dst=192.168.1.2,
ip_proto=tcp, tcp_dst=80}

Normalizing SDN Rules by Matchmap, Actmap

Problem: diverse expressions of the match and action components of SDN rules complicate their automatic comparison based on multi-property set and $\cdot r$, e.g.,

rule 1's match: $\{$ *ip_src=192.168.1.1* , *tcp_dst=80* $\}$

rule 2's match: $\{$ *ip_dst=192.168.1.2* $\}$

Normalizing SDN Rules by Matchmap, Actmap

Problem: diverse expressions of the match and action components of SDN rules complicate their automatic comparison based on multi-property set and $\cdot r$, e.g.,

rule 1's match: $\{ ip\_src = 192.168.1.1,\ tcp\_dst = 80 \}$

rule 2's match: $\{ ip\_dst = 192.168.1.2 \}$

Solution: normalizing the match and action components via a common template to obtain their uniform **matchmap** and **actmap**, e.g.,

ip_src	ip_dst	tcp_dst

rule 1's **matchmap**: $\{ ip\_src = 192.168.1.1,\ ip\_dst = any,\ tcp\_dst = 80 \}$

rule 2's **matchmap**: $\{ ip\_src = any,\ ip\_dst = 192.168.1.2,\ tcp\_dst = any \}$

Results in Randomly Checking Cases (MWN)

The number of conflicts is unknown in advance

Random conflict samples from those identified by the detector are controlled manually

Test	App Priority	# rules	Local conflicts					Traffic Loop	Traffic Drop	HC ESLH
			Sha	Gen	Red	Cor	Ove			
1	(2,2,2,2)	790	0/0/0	0/0/0	0/0/0	27/10/10	0/0/0	0/0/0	0/0/0	60/10/10
2	(2,2,3,4)	803	0/0/0	0/0/0	0/0/0	26/10/10	0/0/0	0/0/0	0/0/0	60/10/10
3	(3,2,2,3)	816	0/0/0	0/0/0	0/0/0	27/10/10	0/0/0	0/0/0	0/0/0	60/10/10
4	(3,5,2,4)	789	0/0/0	0/0/0	0/0/0	25/10/10	0/0/0	0/0/0	0/0/0	59/10/10
5	(5,4,3,2)	791	0/0/0	0/0/0	0/0/0	24/10/10	0/0/0	0/0/0	0/0/0	60/10/10

Each cell shows *the number of conflicts detected by the prototype/ the number of conflicts selected randomly to control/ the number of correct conflicts confirmed based on the manual control*

Sha: Shadowing, Gen: Generalization, Red: Redundancy, Cor: Correlation, Ove: Overlap

HC ESLH: hidden conflict class Event Suppression by Local Handling.

⇒ All randomly checking conflicts are correct