FOCES: Detecting Forwarding Anomalies in Software Defined Networks

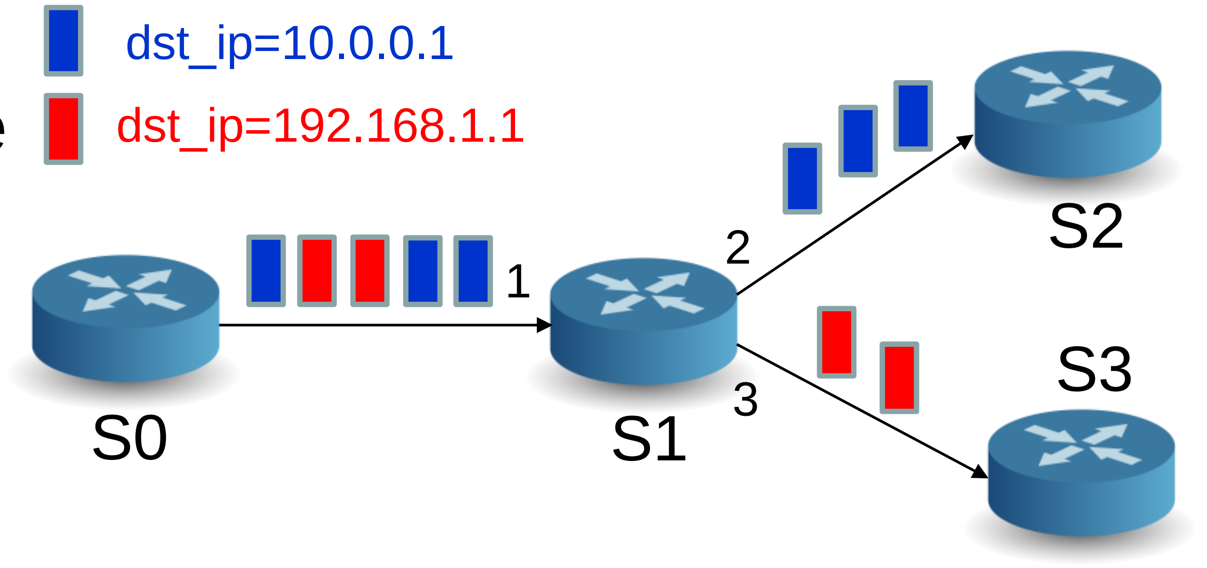
Peng Zhang[1], Shimin Xu[1], **Zuoru Yang[1]**, Hao Li[1] , Qi Li[2] , Huanzhao Wang[1] , Chengchen Hu[1]

[1]Xi’an Jiaotong University
[2]Tsinghua University

ICDCS 2018

Introduction

- Forwarding abstraction of Software Defined Network (SDN)
  - Match-Action model



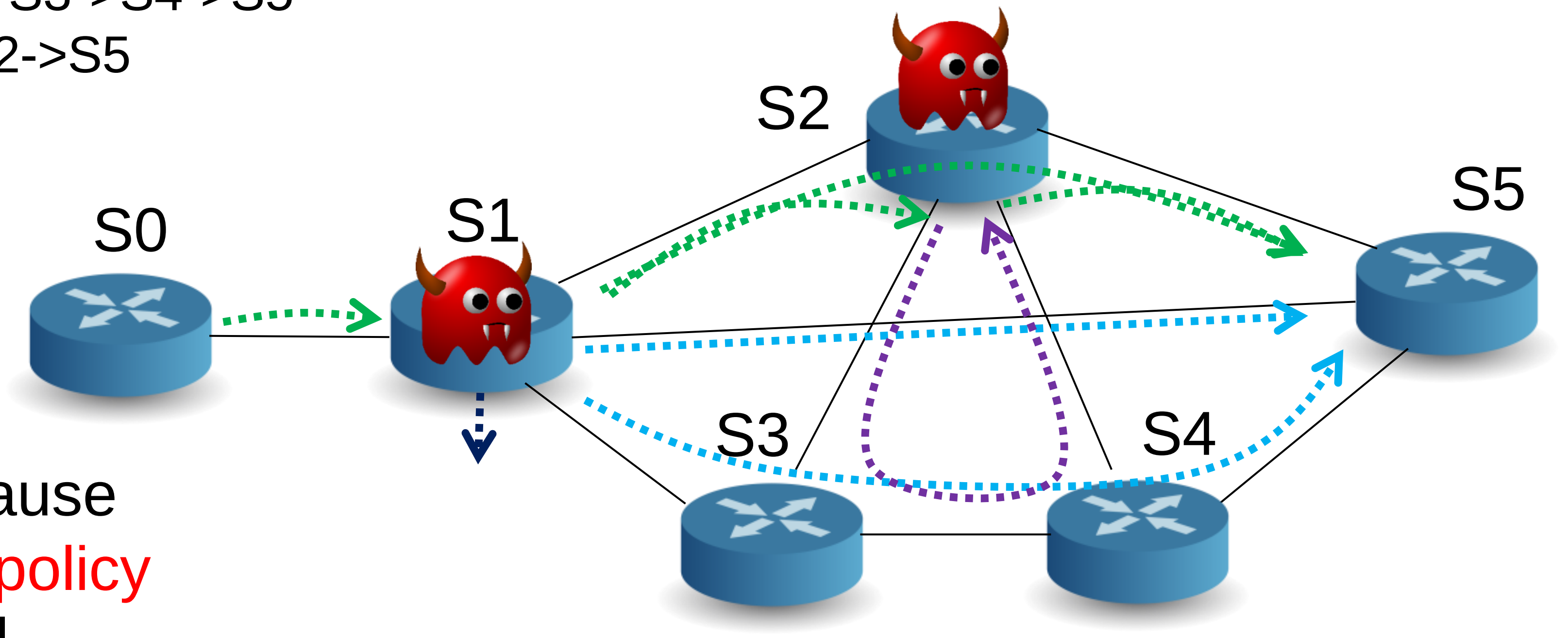
- A packet:
  - 1. match a rule in switches
  - 2. execute the related actions
  - 3. update the counter

S1's forwarding table

	Match	Action	Counter
Rule 1	dst_ip=10.0.0.0/8	forward to port 2	3
Rule 2	dst_ip=192.168.0.0/16	forward to port 3	2
…	…	…	

Forwarding Anomaly

- Normally, Forwarding anomaly can be classified into three types:
  - Early Drop: S1->⊥
  - Switch Bypass: S1->S5, S1->S3->S4->S5
  - Detour: S1->S2->S3->S4->S2->S5



- Forwarding anomaly can cause violation of critical security policy
  - Flow may bypass the firewall
  - ….

Countermeasures

- Rule Dumping
  - Read all the forwarding rules from suspicious switches, and checks the integrity of them
  - Limitation: compromised switches can easily bypass the detection by just reporting the original rules

- Path Validation
  - Each switch imprints packets with signature, so that the destination switch can check whether the path traversed by a packet is correct.
  - Limitation: need to modify switches to support cryptographic operations, high overhead.

Intuition of Statistics Verification

- Packets leave traces (i.e., counters) when they are forwarded along their paths

- If we know how packets SHOULD be forwarded, then we can have constraints on counters of different switches

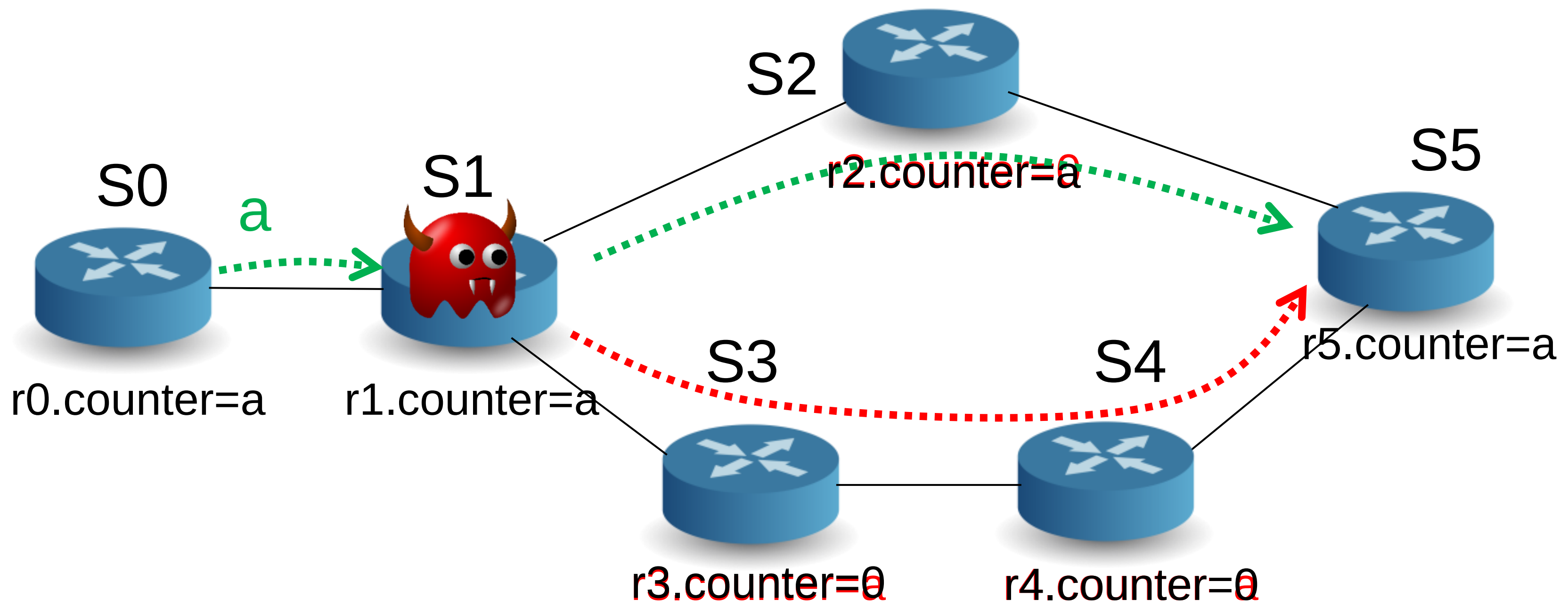
- If the packets deviate from their paths, then the constraints shall be violated.

Toy Example

- We know the path should be:

$$S0 \xrightarrow{r0} S1 \xrightarrow{r1} S2 \xrightarrow{r2} S5 \xrightarrow{r5}$$

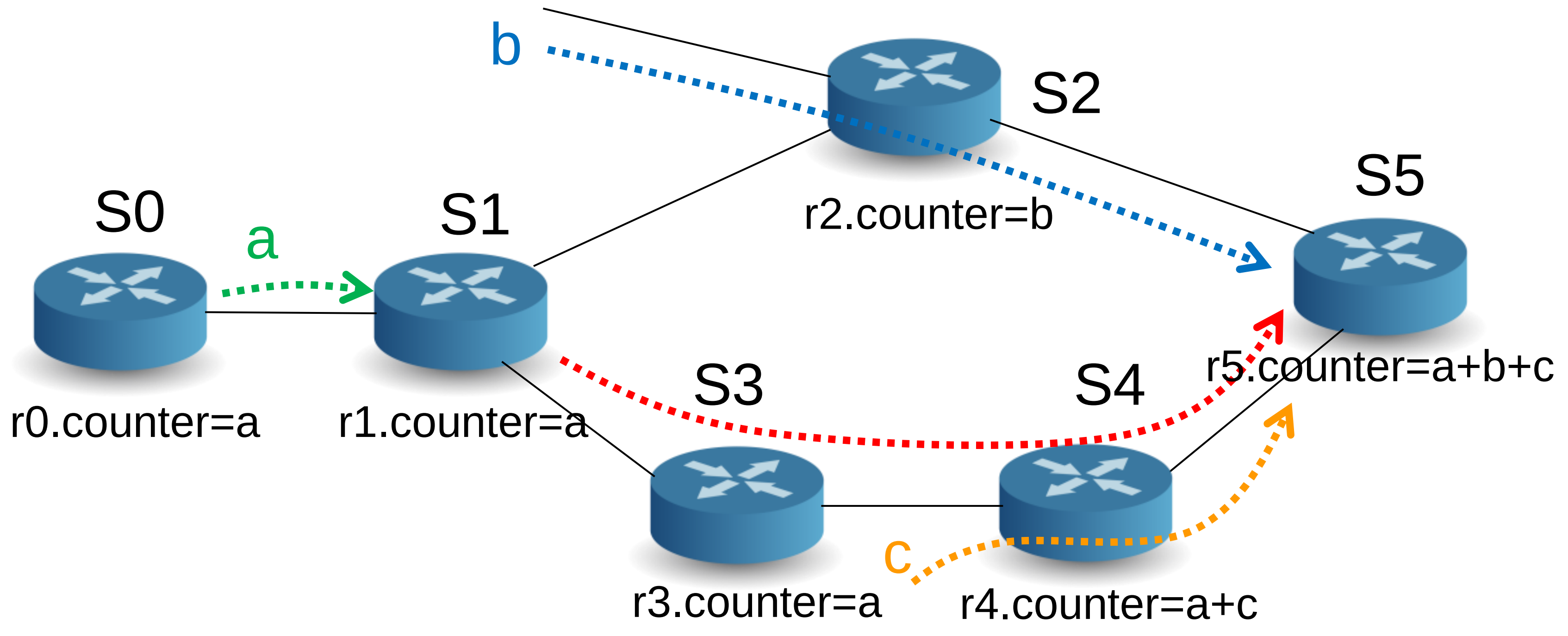
- r0, r1, r2, r5 should have the same counter value, and r3, r4 should have zero counter value. (“Flow Conservation Principle”)



Toy Example

- In real network, there are more than one flows, and each rule may match multiple flows.

e.g., each counter may aggregate multiple flows(wildcard)



The motivation of this work

- ➢ All the previous statistics verification tools check whether the counters of a individual flow conform to the flow conservation principle.

- ➢ However, applying the flow conservation principle for each individual flow has two serious limitations:
  - Limited Detection Scope: miss some forwarding anomalies happening to flows that are not check.
  - High Flow Table Overhead: install dedicated rules to collect the statistics of a specific flow.

An Open Question: **Can we extend the flow conservation principle from *individual* flows to a *network* of flows?**

Outline

- Overview
- FOCES: Theoretical Construction
- FOCES: Make it work
- Implementation & Evaluation

FOCES: FlOw Counter Equation System

- All the flows in the network: $f_1, f_2, \cdots, f_n$
- All the rules in the network: $r_1, r_2, \cdots, r_m$
- Define the **Flow Counter Matrix (FCM)** $H_{m \times n}$ as:

$$H_{i,j} = \begin{cases} 1 & \text{if flow } j \text{ hits rule } i \\ 0 & \text{otherwise} \end{cases}$$

- Let the counter of rule r_i be y_i , and $Y = (y_1, y_2, \cdots, y_m)^T$
- Let the volume of flow f_i be x_i , and $X = (x_1, x_2, \cdots, x_n)^T$
- If there are no forwarding anomalies:

$$H \cdot X = Y$$

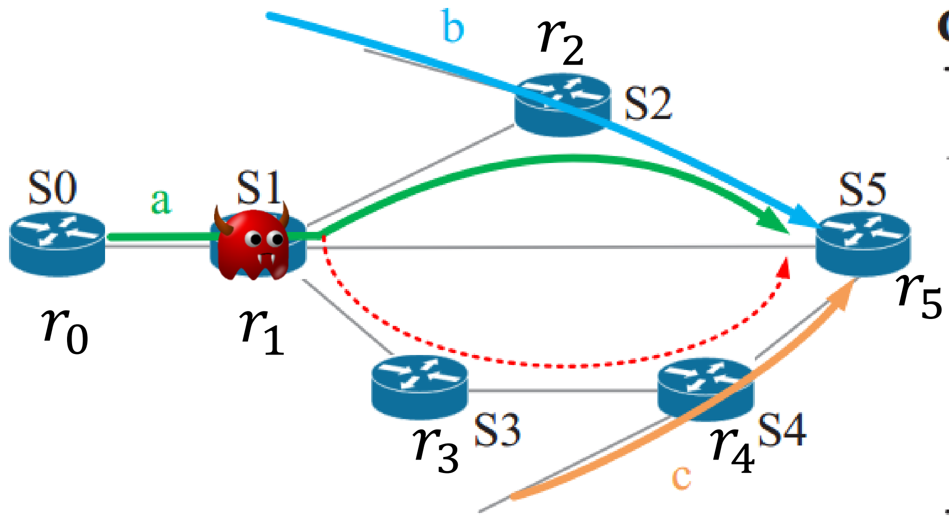
FOCES: FlOw Counter Equation System

- When there are forwarding anomalies, the real FCM will be $H' \neq H$, and the real counter vector will be $Y' = H' \cdot X$.

- However, we do not know either H' or X, but it is expected that $H \cdot \bar{X} = Y'$ should probably has no solutions if $m > n$, when it is a over-determined equation system

- The least square solution will be $\bar{X} = (H^T H)^{-1} H^T Y'$ and we should have $\square = | Y' - H\bar{X} | \neq 0$ (the standard to judge the anomaly)

For Example



Counter Constraints

Rule	Counter	observed counters
r0	a	a
r1	a	a
r2	a+b	b
r3	0	a
r4	c	a+c
r5	a+b+c	a+b+c

$$\begin{array}{c|ccc} & f_1 & f_2 & f_3 \\ \hline r_0 & 1 & 0 & 0 \\ r_1 & 1 & 0 & 0 \\ r_2 & 1 & 1 & 0 \\ r_3 & 0 & 0 & 0 \\ r_4 & 0 & 0 & 1 \\ r_5 & 1 & 1 & 1 \end{array} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ a \\ a+b \\ 0 \\ c \\ a+b+c \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ a \\ b \\ a \\ a+c \\ a+b+c \end{pmatrix}$$

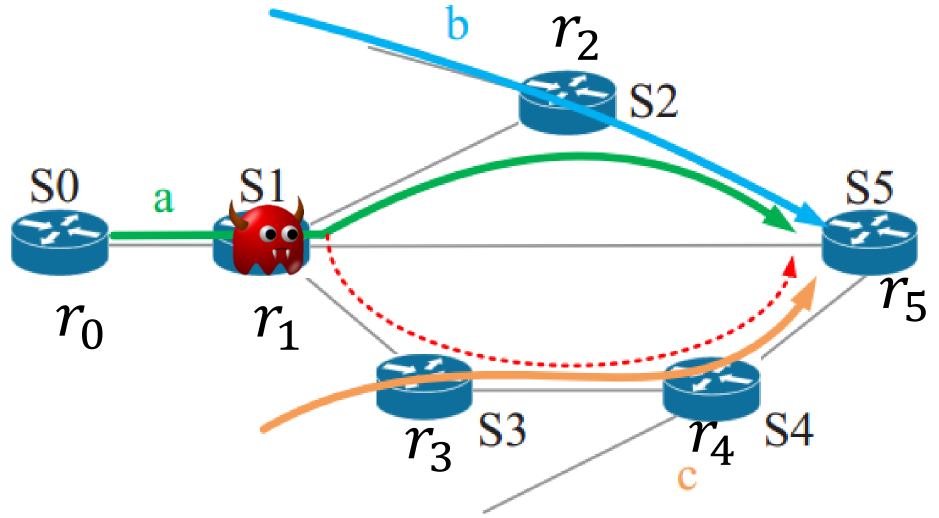
$$\boxed{\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a \\ a \\ b \\ a \\ a+c \\ a+b+c \end{pmatrix}}$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 8 \end{pmatrix} \Rightarrow \square = \begin{pmatrix} 0 & 0 & 0 & \boxed{3} & 0 & 0 \end{pmatrix}^T \neq 0 \Rightarrow$$ Forwarding Anomaly

Does this method always work?

Unfortunately, No

For Example



Counter Constraints

Rule	Counter	observed counters
r0	*a*	*a*
r1	*a*	*a*
r2	*a+b*	*b*
r3	*0*	*a+c*
r4	*c*	*a+c*
r5	*a+b+c*	*a+b+c*

$$\begin{array}{c|ccc} & f_1 & f_2 & f_3 \\ \hline r_0 & 1 & 0 & 0 \\ r_1 & 1 & 0 & 0 \\ r_2 & \boxed{1} & 1 & 0 \\ r_3 & \boxed{0} & 0 & \boxed{1} \\ r_4 & \boxed{0} & 0 & 1 \\ r_5 & 1 & 1 & 1 \end{array} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ a \\ \boxed{a+b} \\ \boxed{c} \\ \boxed{c} \\ a+b+c \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ \boxed{0} & 1 & 0 \\ \boxed{1} & 0 & 1 \\ \boxed{1} & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} a \\ a \\ \boxed{b} \\ \boxed{a+c} \\ \boxed{a+c} \\ a+b+c \end{pmatrix}$$

$$\boxed{\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a \\ a \\ b \\ a+c \\ a+c \\ a+b+c \end{pmatrix}}$$

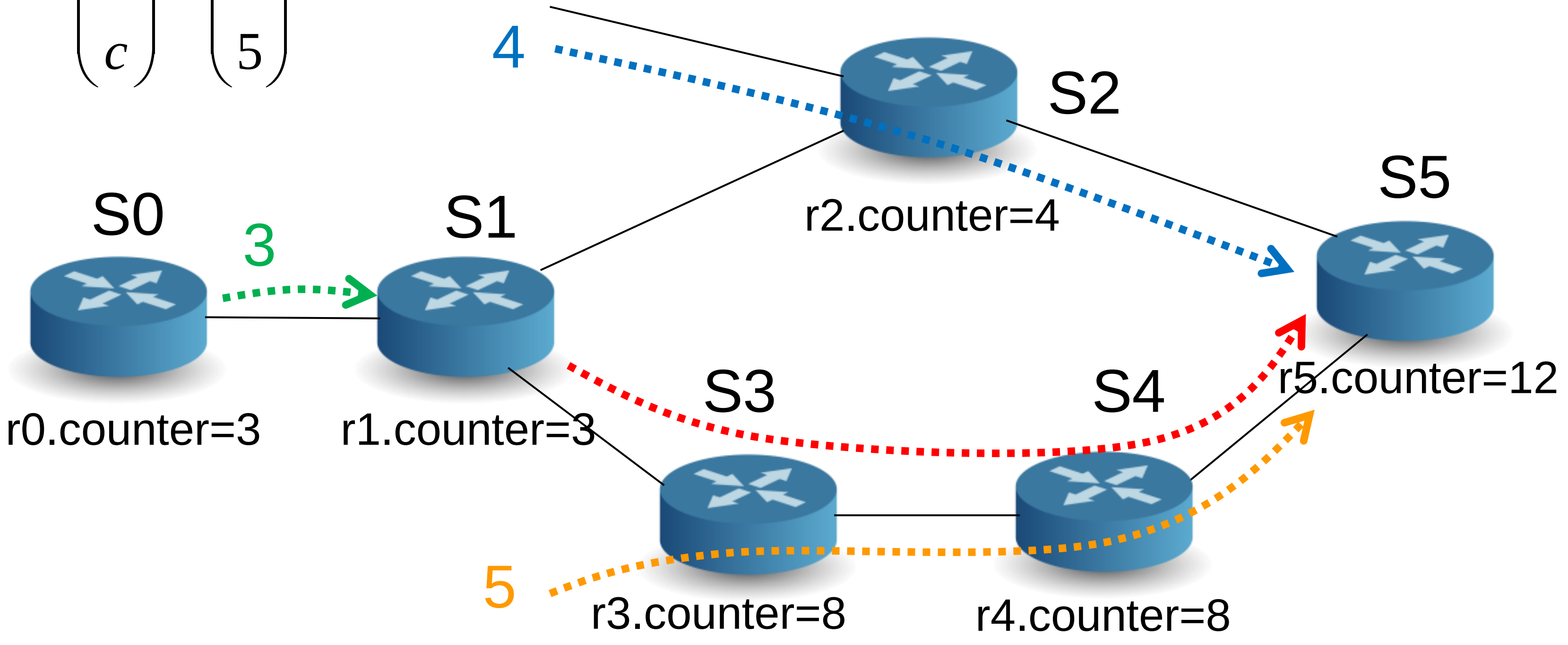
$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 8 \end{pmatrix} \Rightarrow \square = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T \Rightarrow$$

Normal (Wrong Result)

The Reason of this Failure

➢ The observed counters in this example

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix}$$

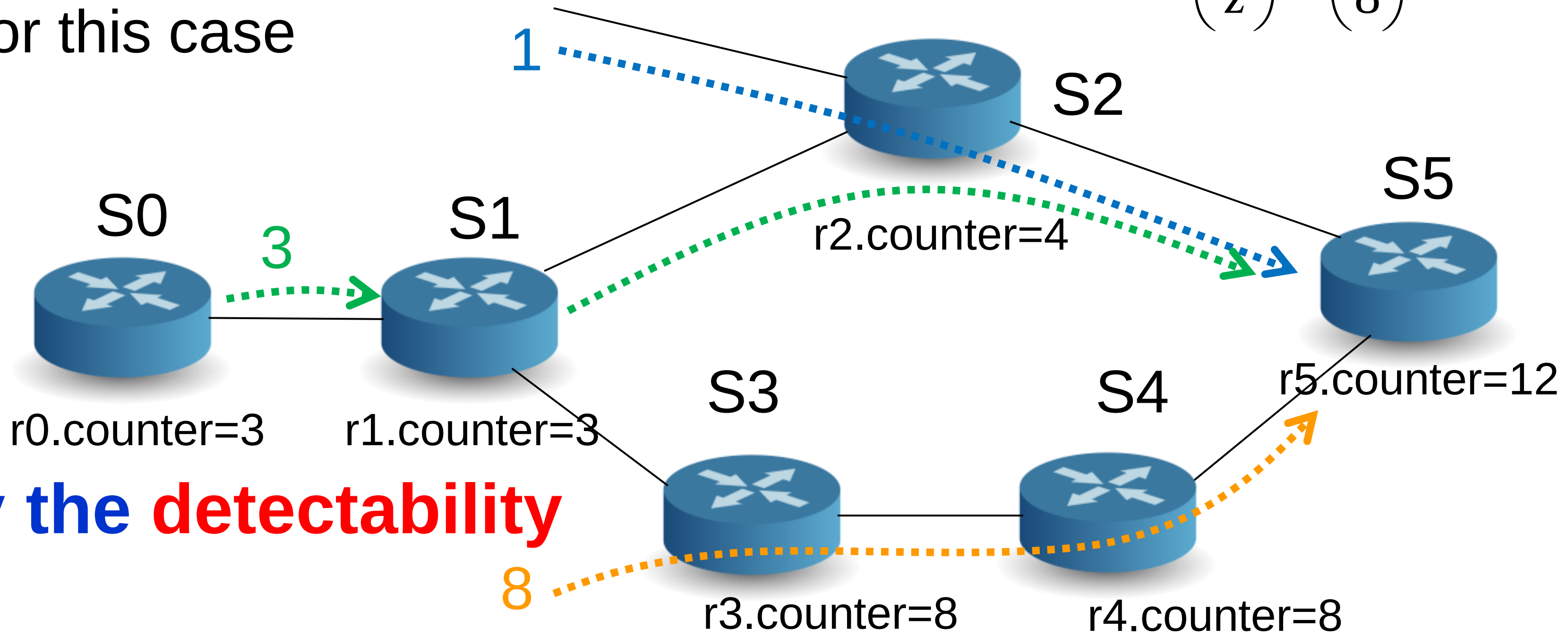


The Reason of this Failure

- The estimated counters in this example
  - same as the observed one

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 8 \end{pmatrix}$$

- Finding:
  - FOCES cannot work for this case



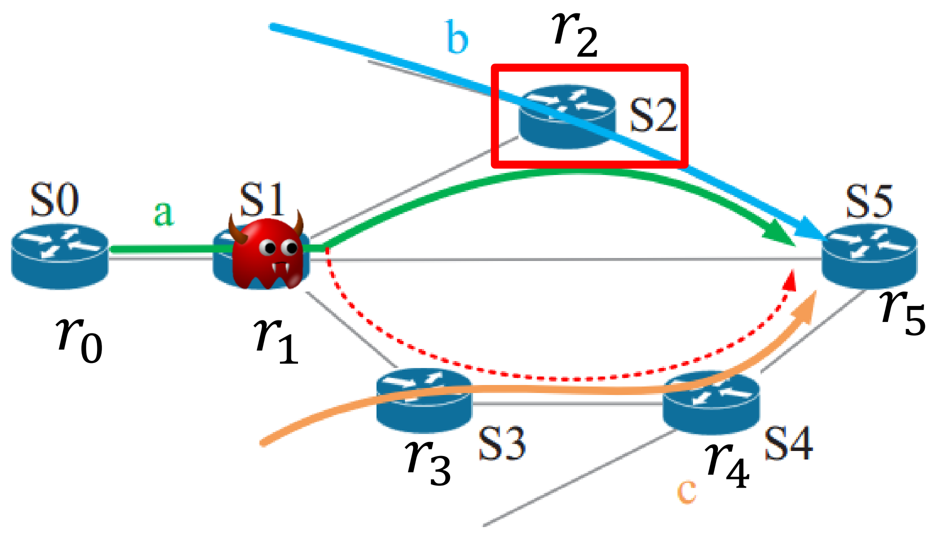
Question: **How to identify the detectability of a given case?**

Analysis on Detectability

- **Theorem 1**: If $h_i \rightarrow h_i'$ is undetectable if and only if h_i' lies in the linear subspace generated by $h_1, h_2, \cdots, h_n$
  - Theorem 1 is different to apply in real network
  - Its algorithm is complex

- **Theorem 2**: If $h_i \rightarrow h_i'$ is undetectable if and only if there is a switch S whose RBG $G_S^H(V_{in}, V_{out}, E)$ contains a loop
  - reduce Theorem 1 to the problem of finding loops in a bipartite graph

Review the Failure Example

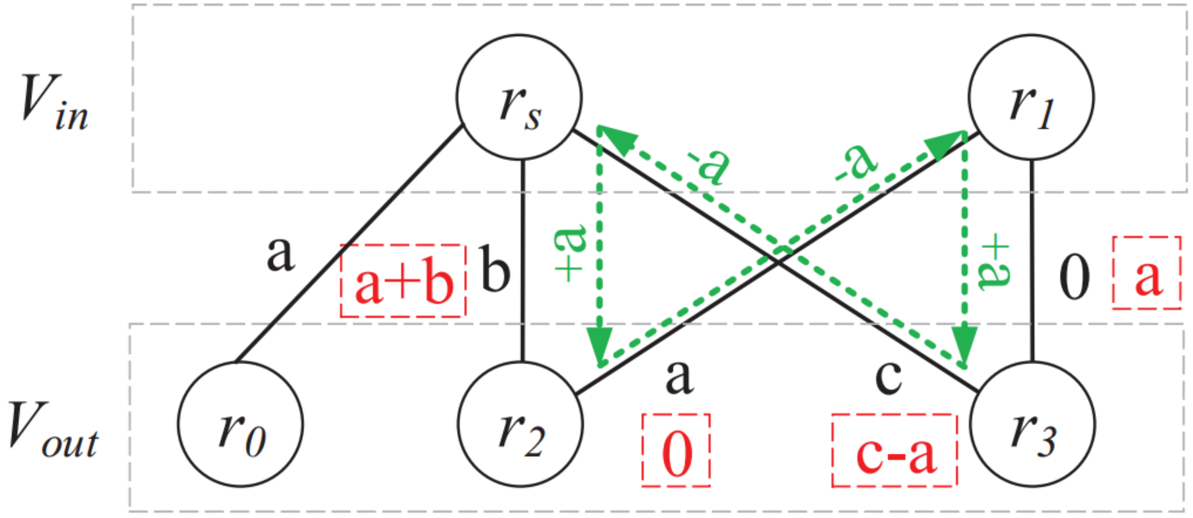


Counter Constraints

Rule	Counter	observed counters
r0	*a*	*a*
r1	*a*	*a*
r2	*a+b*	*b*
r3	*0*	*a+c*
r4	*c*	*a+c*
r5	*a+b+c*	*a+b+c*

➢ The Rule Bipartite Graph (RBG) of S2

- a loop marked in green dashed lines



Outline

Make FOCES Work in Realistic Settings

➢ Noises:

- Packet losses
- Out-of-sync counter

➡ $\square = | Y' - H\bar{X} | \neq 0$

➢ Scalability:

- Calculating the inverse of FCM is expensive when there are a large number of rules and flows

➡ Hard to apply in large scale network

Threshold-based Detection Algorithm

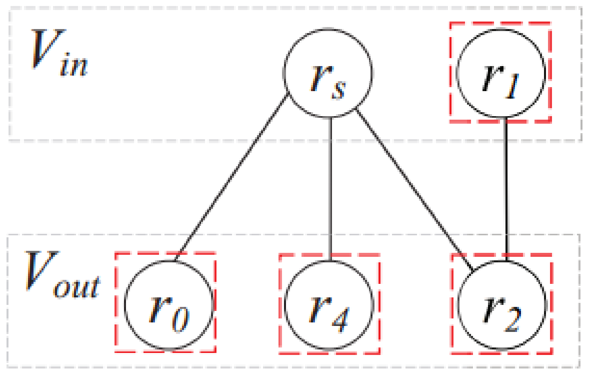
- Basic Idea: define the *anomaly index (AI)* to measure the possibility of forwarding anomaly, and eliminate the impact of such noises

- Anomaly Index: $\frac{Err_{\max}}{Err_{med}}$
  - the ratio of **Maximum** and **Median** of all elements in□
  - When there are forwarding anomalies, the AI should be very large ("majority good" assumption)

- Detection Threshold: T
  - $AI > T$: forwarding anomalies
  - $T = 4.5$ is the default detection threshold ("three-sigma rule" in probability theory)

Making FOCES Scalable

- Basic Idea: make FOCES scalable by reducing the computation time
  - It is inspired by the Rule Bipartite Graph (RBG)
  - Shrank the scale of FCM
- FCM Slicing:
  - extract the sub-FCM corresponding to the RBG.
  - Sub-FCMs are much smaller than the original FCM, it reduces the computation time.
- For Example:



Rule Bipartite Graph for *S2*

$$H = \begin{array}{c c} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} & \begin{array}{c} r_0 \\ r_1 \\ r_2 \\ \\ r_4 \\ \\ \end{array} \\ \begin{array}{ccc} f_1 & f_2 & f_3 \end{array} & \end{array} \rightarrow H(S_2) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Making FOCES Scalable

- **Theorem 3**: If a forwarding anomaly $h_i \to h_i'$ is detectable (without slicing), then it is still detectable when using slicing.
  - using slicing is equivalent to the baseline method in detecting forwarding anomalies

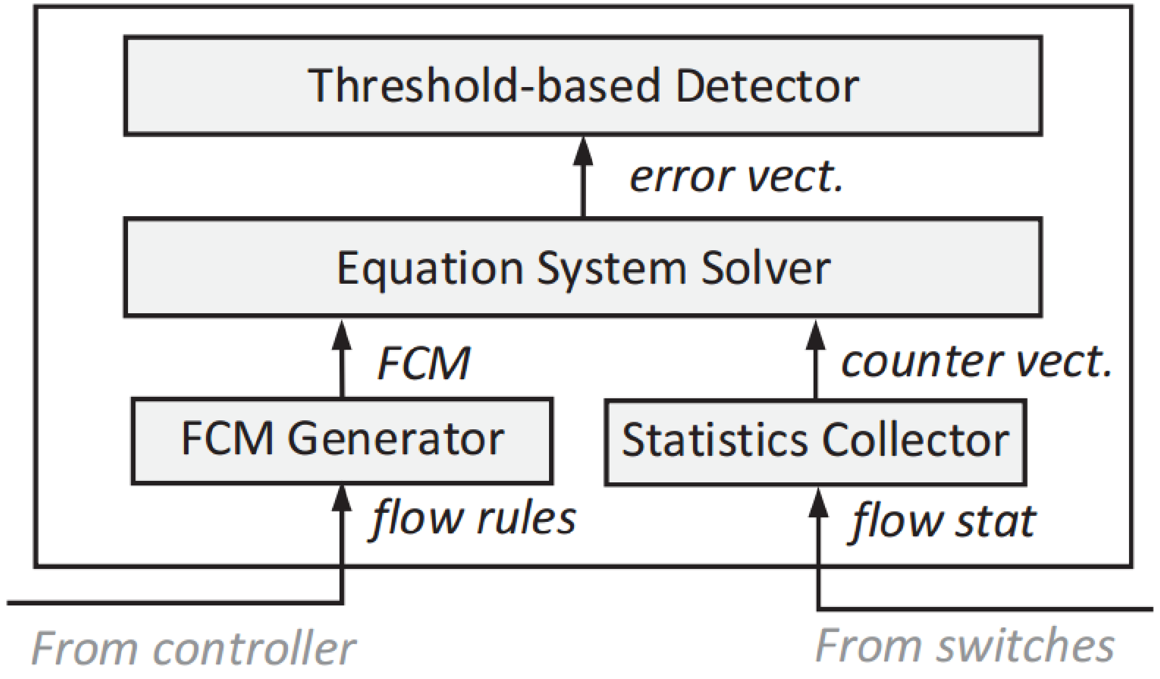
- Analysis on Computation Complexity Reduction:
  - without slicing: $O(N^3)$ *N* is the size of the FCM (approximately equals to that of Matrix Inversion)
  - with slicing: $O(N^{2.3})$

Outline

- Overview
- FOCES: Theoretical Construction
- FOCES: Make it work
- Implementation & Evaluation

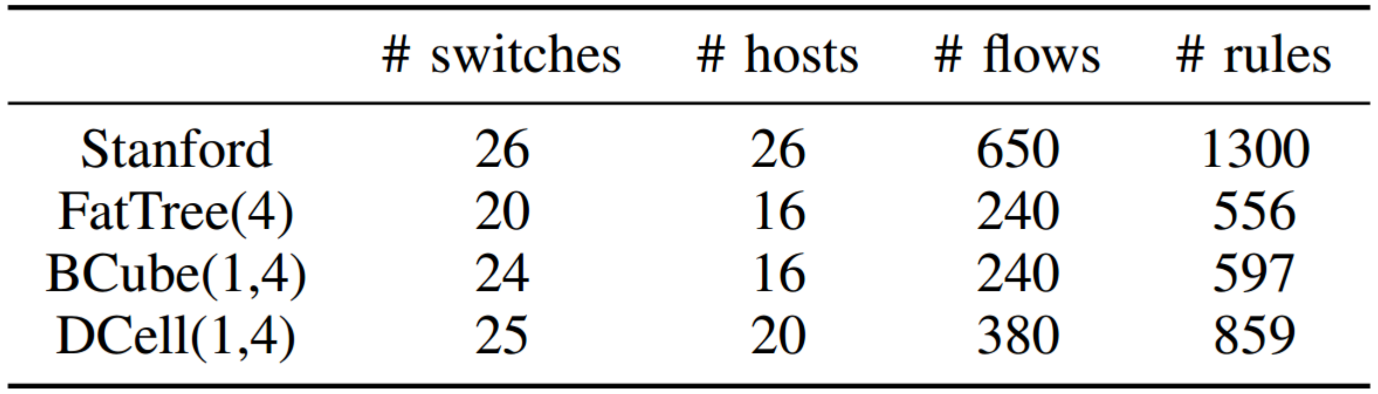
Implementation

- FOCES prototype:
  - 1500 LOC in Python
  - FCM Generator**:** ATPG, Floodlight REST API
  - Statistics Collector**:** Floodlight REST API, parse counters
  - Equation System Solver**:** "*NumPy*" library, "*sparse*" library of python



Experiment Setup

- SDN Controller: Floodlight v2.1
- Network: Mininet + Open vSwitches
- Topologies: Stanford, FatTree(4), BCube(1, 4), DCell(1, 4)

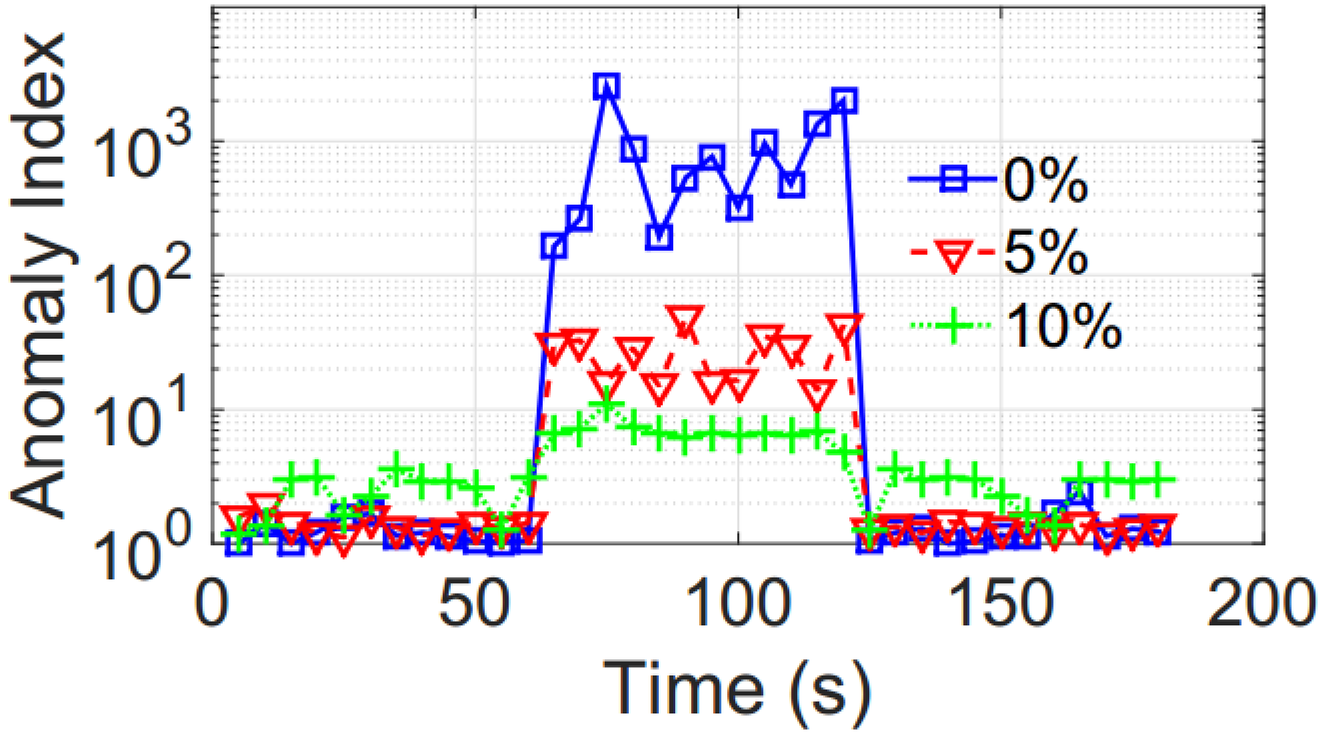


	# switches	# hosts	# flows	# rules
Stanford	26	26	650	1300
FatTree(4)	20	16	240	556
BCube(1,4)	24	16	240	597
DCell(1,4)	25	20	380	859

Functional Test

- Setting
  - BCube(1, 4)
  - Packet Loss Rates: 0%, 5%, 10%
  - Modify a rule: 60s-120s

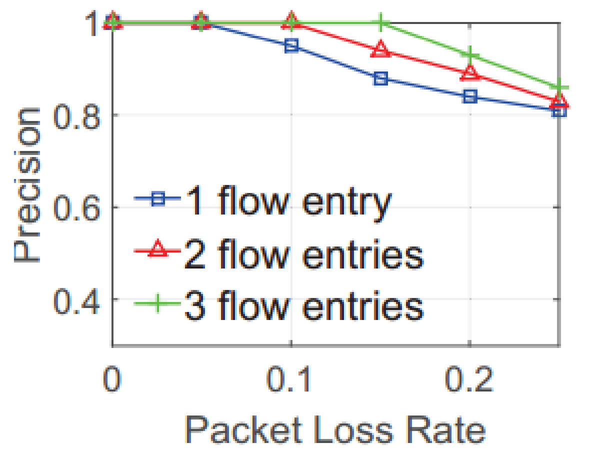
- Finding
  - AI quickly goes beyond the threshold, when forwarding anomalies happen.
  - Normal and anomaly cases become less distinguishable when packet loss rates increase



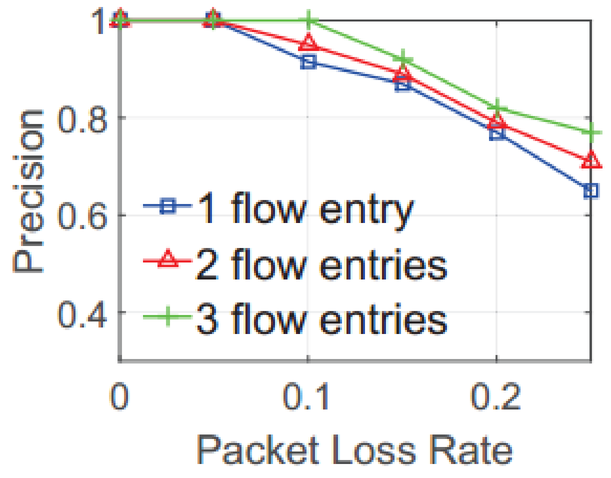
Detection Precision vs. Number of Anomalies

- Detection Precision $\frac{TP}{TP+FP}$
  - Randomly modify 1, 2, and 3 rules.
  - Detection threshold: 3.5

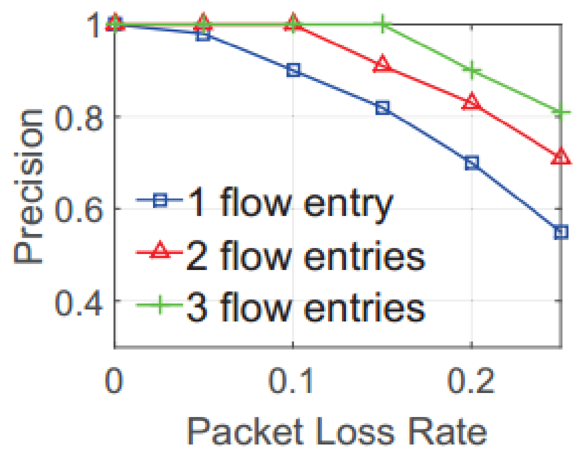
- Findings
  - Precision increases when more rules are modified.
  - Packet loss rate < 10%: precision > 90%



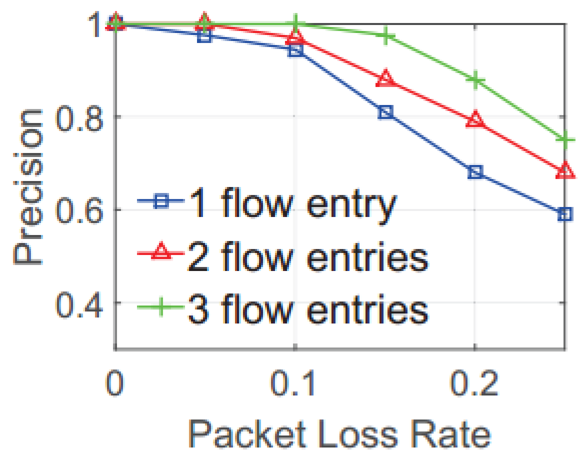
(a) Stanford



(b) FatTree(4)



(c) BCube(1,4)

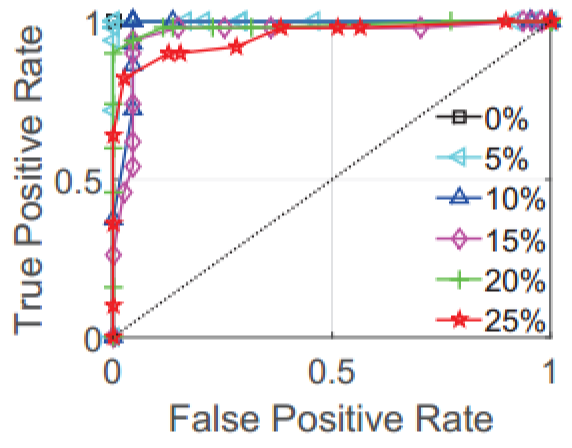


(d) DCell(1,4)

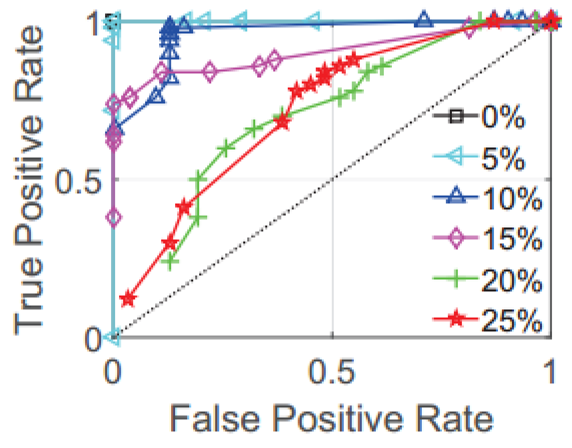
Detection Accuracy vs. Detection Threshold

- The Recover Operating Characteristic (ROC) Curve
  - Detection threshold: 1 ~ 100
  - Packet loss rates: 0% ~ 25%

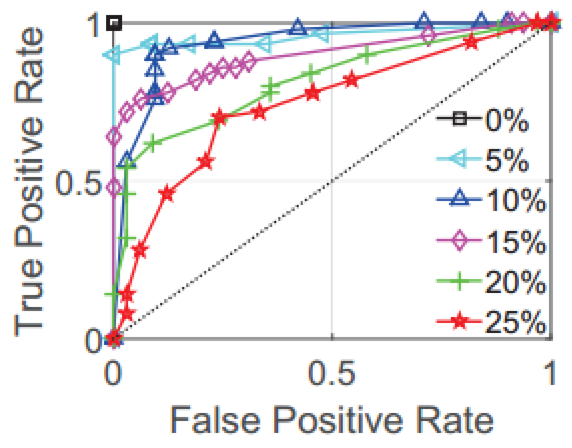
- Findings
  - Accuracy of FOCES is little affected: Packet Loss Rate < 10%
  - Best detection threshold: around 4.5
  - Best performance: TP rate nearly 100% and a FP rate around 4.3%.



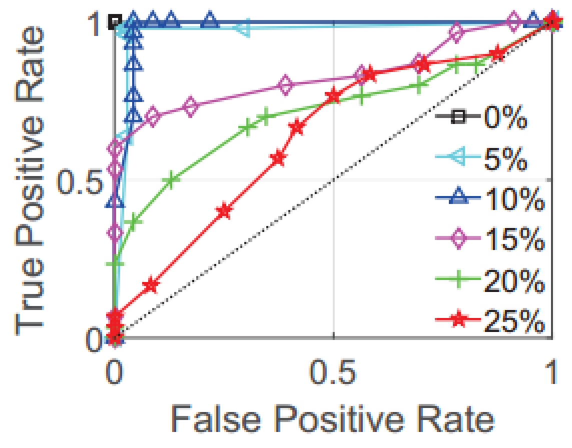
(a) Stanford



(b) FatTree(4)



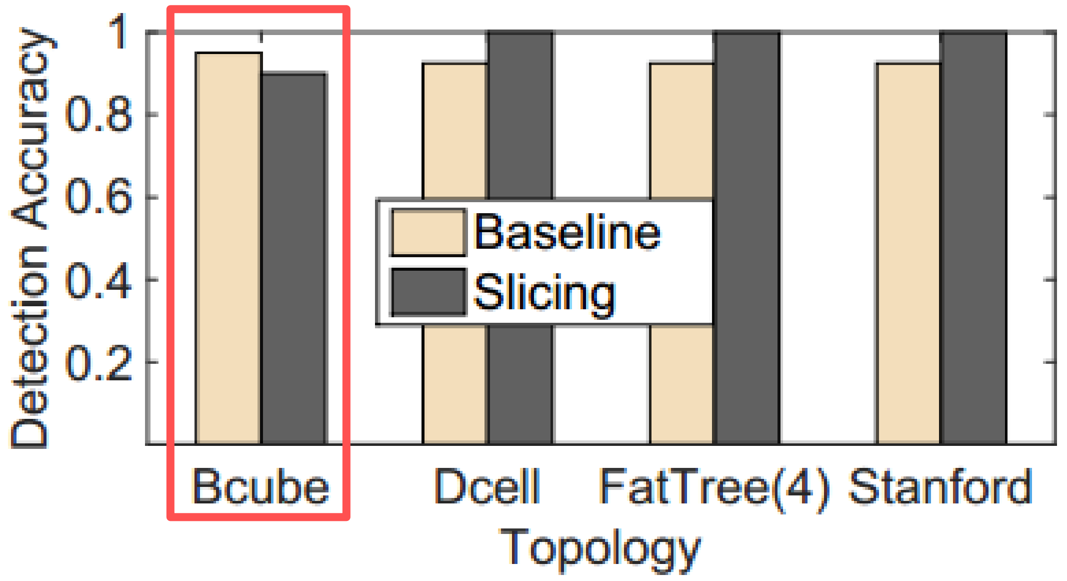
(c) BCube(1,4)



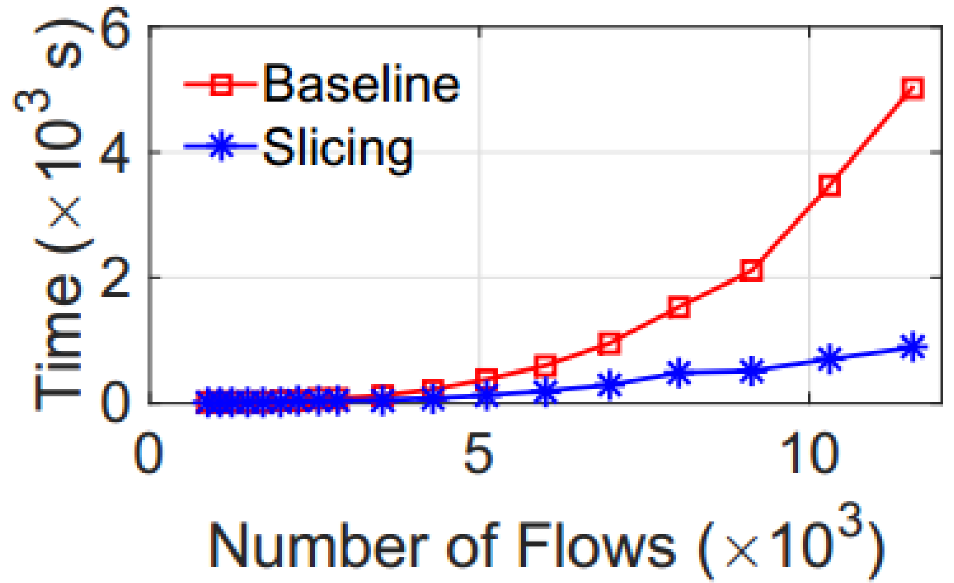
(d) DCell(1,4)

The Effectiveness of Slicing

- Detection Accuracy $\frac{TP+TN}{N+P}$
  - Slicing can achieve an even better detection accuracy, except for BCube(1, 4) topology.



- Computation time:
  - Topology: FatTree(8)
  - Computation time: slicing grows much slower than without slicing.
  - Reduction of computation overhead: nearly 80%



Conclusions

- Study how to extend flow conservation principle from *individual* flows to a *network* of flows and how to use it detect forwarding anomalies
- Design and analyze FOCES from both **theoretical** and **practical** perspectives
- Build FOCES prototype and conduct extensive experiments on Mininet with four topologies
  - Empirical results match theories
- Future Work:
  - The localization of the compromised switch (we have just finished it)