XShot: Light-weight Link Failure Localization using Crossed Probing Cycles in SDN

Hongyun Gao, Laiping Zhao*, Huanbin Wang, Zhao Tian, Lihai Nie, Keqiu Li

TANKLab, Tianjin University

More links, more failures

- Networks grow rapidly in scale
  - Ten thousands of network devices
  - Hundred thousands of links
- Failures become common
  - Fail-stop failures
  - Partial failures
    - E.g., a faulty link dropping packets randomly

Home » Security Bloggers Network » How to Efficiently Onboard Thousands of Devices

How to Efficiently Onboard Thousands of Devices

by Jake Ludin on November 21, 2018

Each year, college campuses must navigate the trials associated with successfully connecting thousands of new students to the
vices in the past few

Infoblox research finds explosion of personal and IoT devices on enterprise networks introduces immense security risk

Infoblox
NEXT LEVEL NETWORKING

A quarter of US
has an IoT secur
20 percent of U
personal and Io

COMPLEX SYSTEMS

The New Laws of Explosive Networks

Researchers are uncovering the hidden laws that reveal how the Internet grows, how viruses spread, and how financial bubbles burst.

Severe service outages caused by failures

- It often takes hours or more to restore
- Huge economic losses and labor consumptions

Local News

Bell outage in Ottawa leaves customers without service

Bruce Deachman
Aug 06, 2020 • Last Updated 2 days ago • 1 minute read

Problems at Bell

Google's service outage today was caused by... a router failure (Updated)

29

CLOUD COMPUTING > PERFORMANCE MANAGEMENT

IBM Cloud Outage Causes Disruptions: Learning from the Failure

IBM Cloud suffered a multi-zone outage impacting its services. Here are steps and strategies organizations should take to limit cloud outage risk.

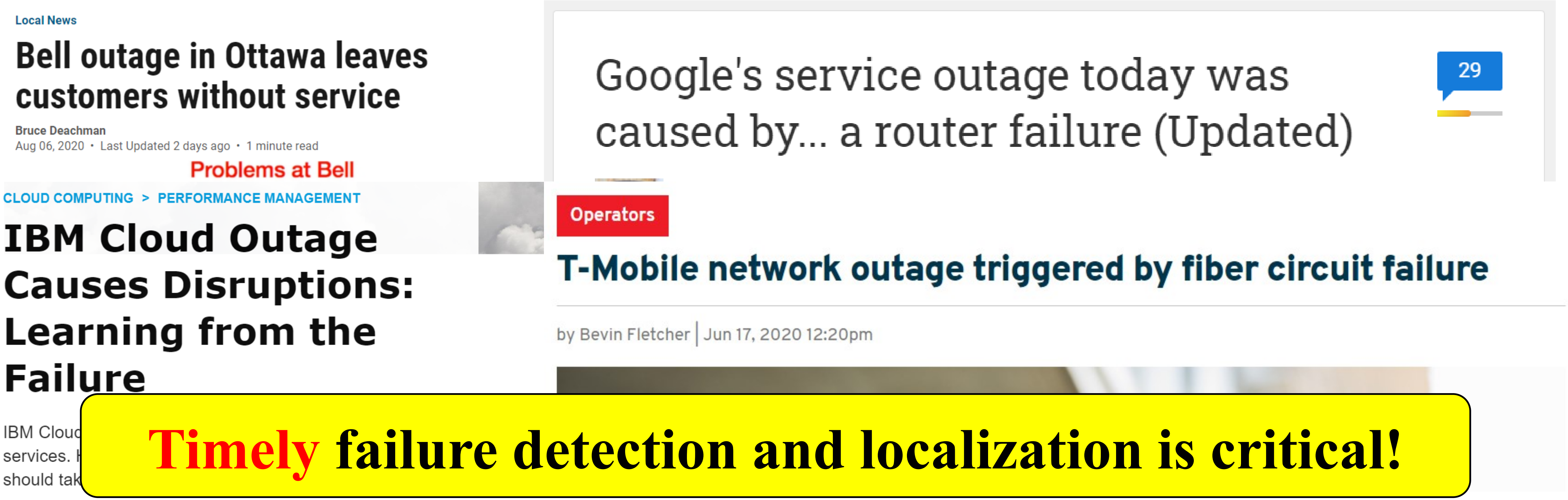
Operators

T-Mobile network outage triggered by fiber circuit failure

by Bevin Fletcher | Jun 17, 2020 12:20pm

Severe service outages caused by failures

- It often takes hours or more to restore
- Huge economic losses and labor consumptions


Local News

Bell outage in Ottawa leaves customers without service

Bruce Deachman
Aug 06, 2020 • Last Updated 2 days ago • 1 minute read

Problems at Bell

Google's service outage today was caused by... a router failure (Updated)

29

CLOUD COMPUTING > PERFORMANCE MANAGEMENT

IBM Cloud Outage Causes Disruptions: Learning from the Failure

IBM Clou
services. I
should tak

Operators

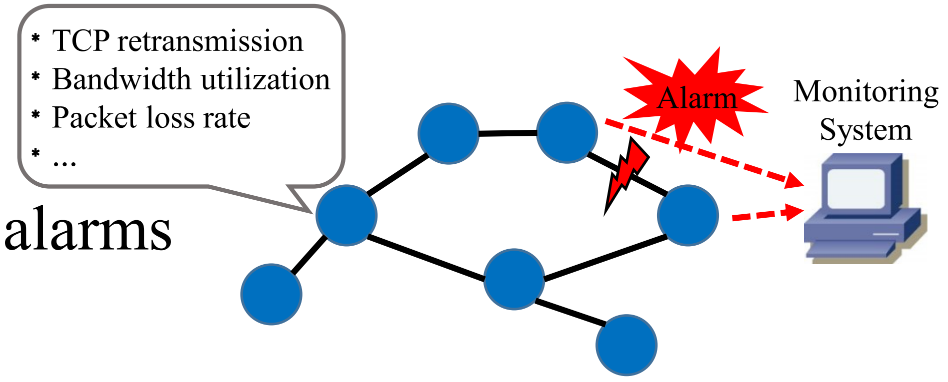
T-Mobile network outage triggered by fiber circuit failure

by Bevin Fletcher | Jun 17, 2020 12:20pm

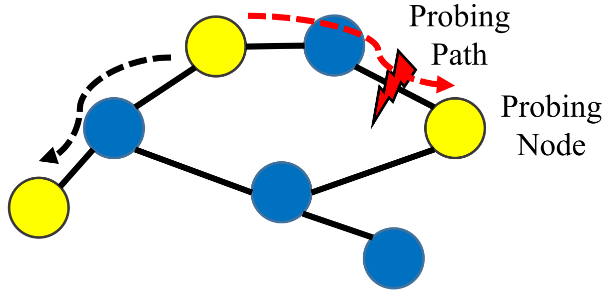
Timely failure detection and localization is critical!

Existing tools rely on network monitoring

- Passive monitoring
  - Use readily available metrics to generate failure alarms
  - The downside is alarm signals are often missed
    - Introduce many false alarms
    - Turn failure localization into a long-time lagging process
- Active probing
  - Inject probing packets to monitor the network status
  - But it cannot provide accurate failure position
    - Due to the unknown routing in traditional networks



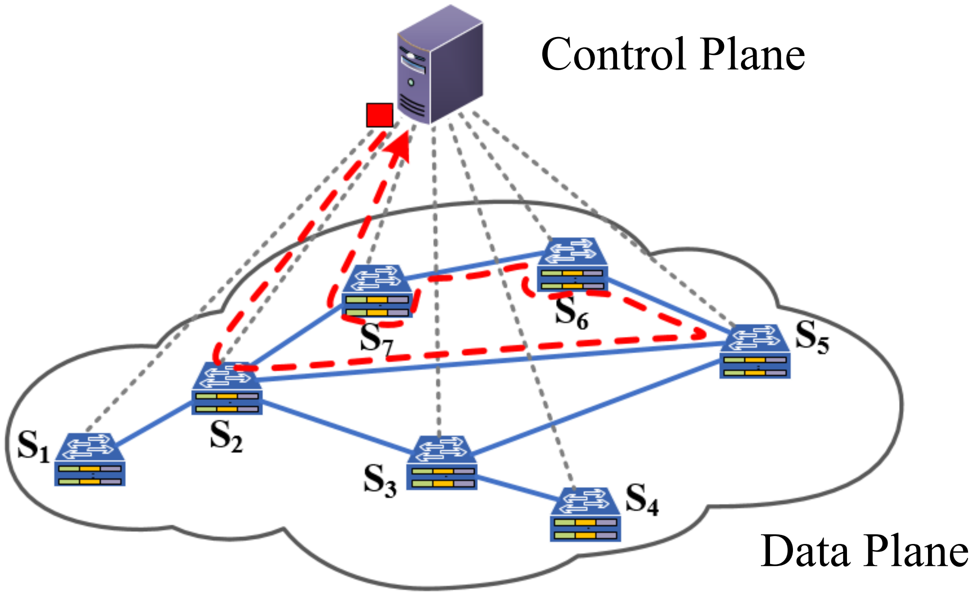
Passive monitoring



Active probing

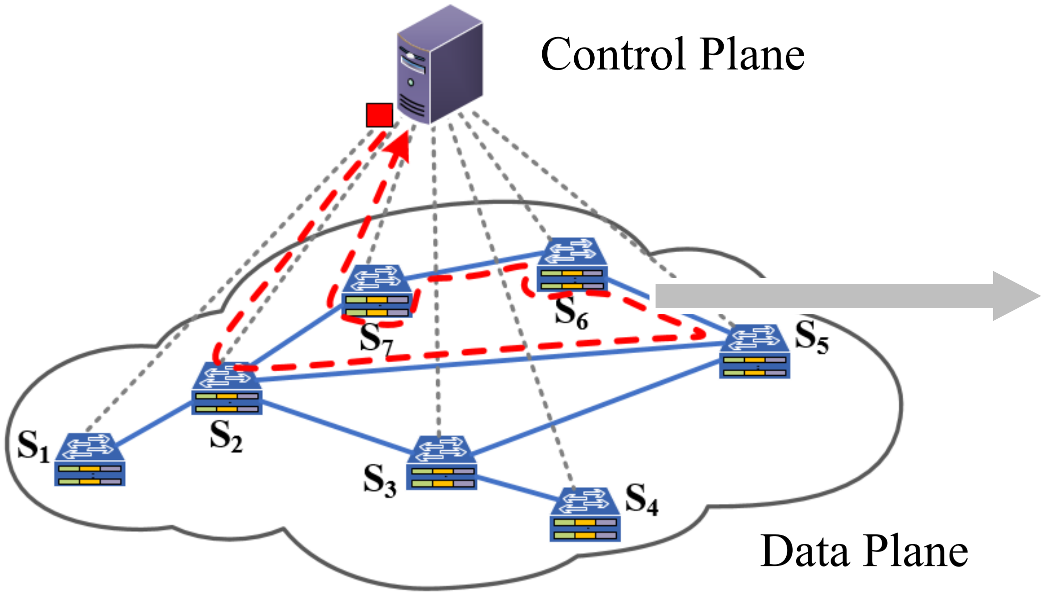
SDN opens up an opportunity

- It decouples the control plane from the data plane
- It routes packets on predefined paths



SDN opens up an opportunity

- It decouples the control plane from the data plane
- It routes packets on predefined paths



The predefined paths make it possible to localize the exact position of failures efficiently.

Connectivity verification is not enough

- Connectivity verification
  - Measure the up-or-down state of a path according to the receiving state of probing packets
  - Moreover, richer link metrics can be further derived through end-to-end performance measurements
- Although effective
  - Cannot distinguish fail-stop and partial failures
  - Incur high cost
    - Additional hardware monitors
    - Many probing packets and forwarding rules
    - Long probing time

Connectivity verification is not enough

- Connectivity verification
  - Measure the up-or-down state of a path according to the receiving state of probing packets
  - Moreover, richer link metrics can be further derived through end-to-end performance measurements
- Although effective
  - Cannot distinguish fail-stop and partial failures
  - Incur high cost
    - Additional hardware monitors
    - Many probing packets and forwarding rules
    - Long probing time

Probing packets impose a large communication load

Forwarding rules take expensive resources of TCAM

Our aim

- To pinpoint the exact faulty links in SDN in a more light-weight and quick manner
  - To save cost
    - Reduce the number of probing packets and forwarding rules
    - Need no additional hardware monitors
  - To distinguish fail-stop and partial failures

Major challenges

- How to formulate the probing cost in terms of packets and rules?
  - Probing packets and forwarding rules increase over the number of probing paths
  - To minimize the cost, the probing paths should be crafted carefully
- How to identify partial failures from noisy measurements?
  - Given the probing paths, the measured metrics are often noisy
  - It is difficult to recognize partial failures from noises

Our design: *XShot*

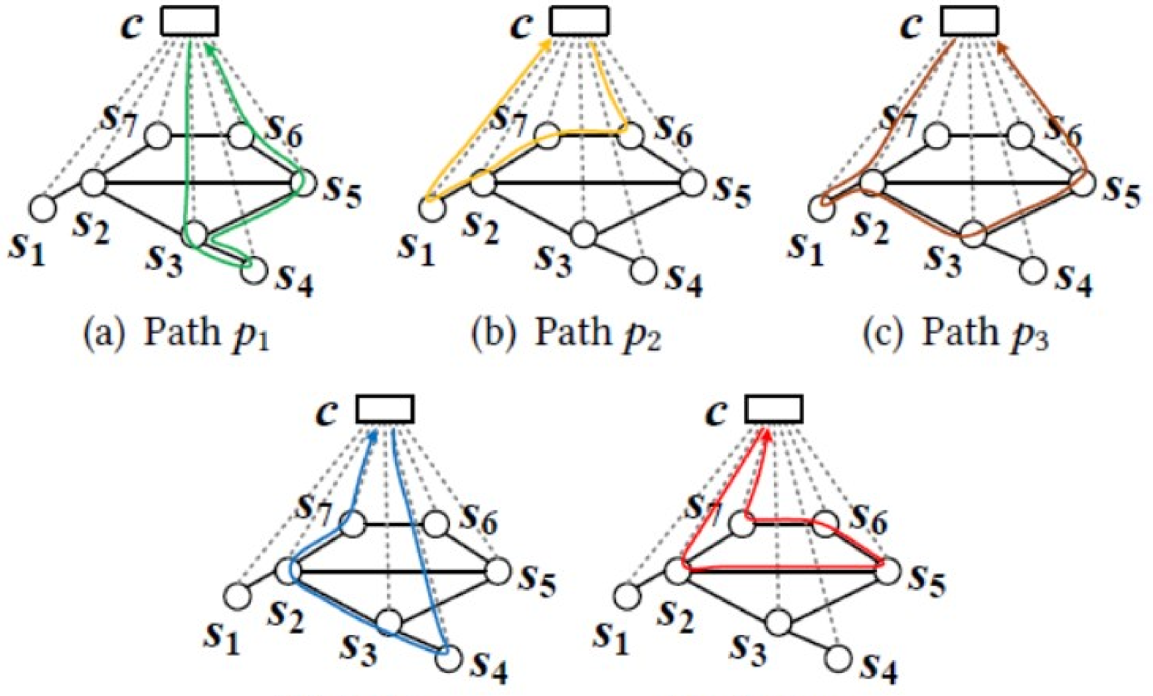
- A quick and light-weight failure localization system in SDN
  - Cross verification
    - A cross probing-based link failure localization method in SDN
  - ILP model
    - For minimizing the number and length of probing paths
  - ADW-Donut
    - A machine learning algorithm that learns to identify partial failures from noisy measurements

What is cross verification?

- A method to localize the faulty link within just one-round shot of crossed
  - Each link failure corresponds to one and only one binary code
  - The code is defined based on the probing results of crossed paths

Example: Probing solution for an SDN

- Five probing paths (i.e., cycles) with controller *c* as the only monitor
- Each link has a unique 5-bit failure code



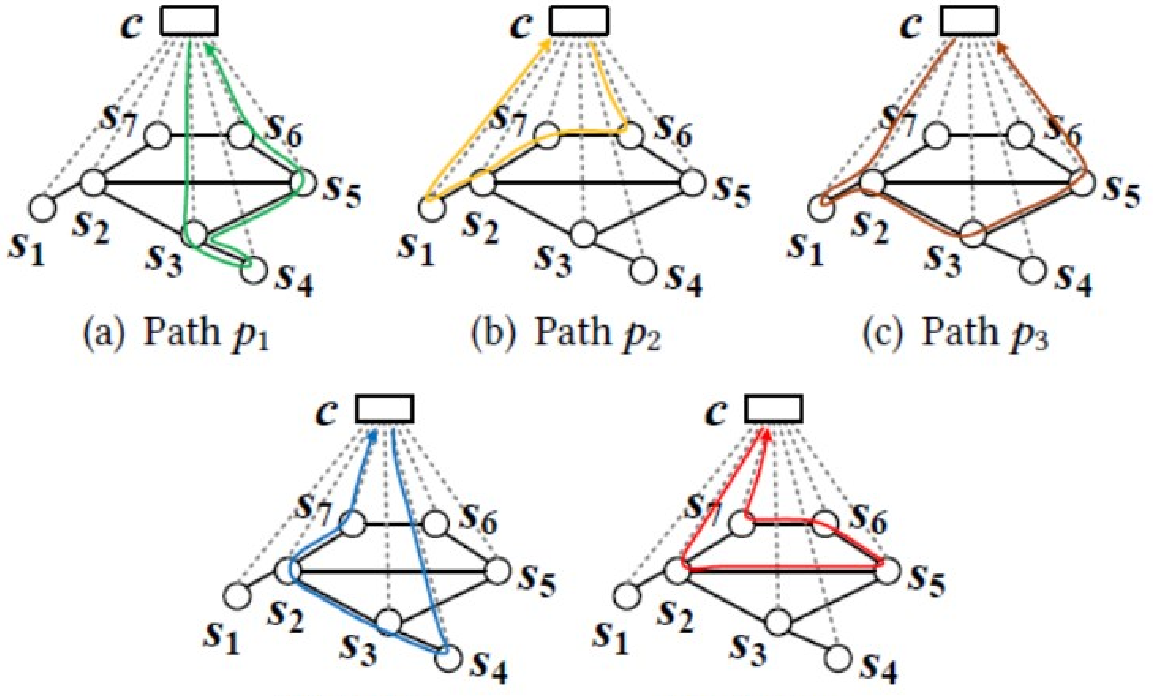
(a) Path p_1 (b) Path p_2 (c) Path p_3 (d) Path p_4 (e) Path p_5

Link	Binary Code					Link	Binary Code				
	p_1	p_2	p_3	p_4	p_5		p_1	p_2	p_3	p_4	p_5
(s_1, s_2)	0	1	1	0	0	(c, s_1)	0	1	0	0	0
(s_2, s_3)	0	0	1	1	0	(c, s_2)	0	0	1	0	1
(s_2, s_5)	0	0	0	0	1	(c, s_3)	1	0	0	0	0
(s_2, s_7)	0	1	0	1	0	(c, s_4)	0	0	0	1	0
(s_3, s_4)	1	0	0	1	0	(c, s_5)	0	0	1	0	0
(s_3, s_5)	1	0	1	0	0	(c, s_6)	1	1	0	0	0
(s_5, s_6)	1	0	0	0	1	(c, s_7)	0	0	0	1	1
(s_6, s_7)	0	1	0	0	1						

(f) Cross verification code for each link failure

Example: Probing solution for an SDN

- Five probing paths (i.e., cycles) with controller *c* as the only monitor
- Each link has a unique 5-bit failure code



(a) Path p_1 (b) Path p_2 (c) Path p_3

(d) Path p_4 (e) Path p_5

Link	Binary Code					Link	Binary Code				
	p_1	p_2	p_3	p_4	p_5		p_1	p_2	p_3	p_4	p_5
(s_1, s_2)	0	1	1	0	0	(c, s_1)	0	1	0	0	0
(s_2, s_3)	0	0	1	1	0	(c, s_2)	0	0	1	0	1
(s_2, s_5)	0	0	0	0	1	(c, s_3)	1	0	0	0	0
(s_2, s_7)	0	1	0	1	0	(c, s_4)	0	0	0	1	0
(s_3, s_4)	1	0	0	1	0	(c, s_5)	0	0	1	0	0
(s_3, s_5)	1	0	1	0	0	(c, s_6)	1	1	0	0	0
(s_5, s_6)	1	0	0	0	1	(c, s_7)	0	0	0	1	1
(s_6, s_7)	0	1	0	0	1						

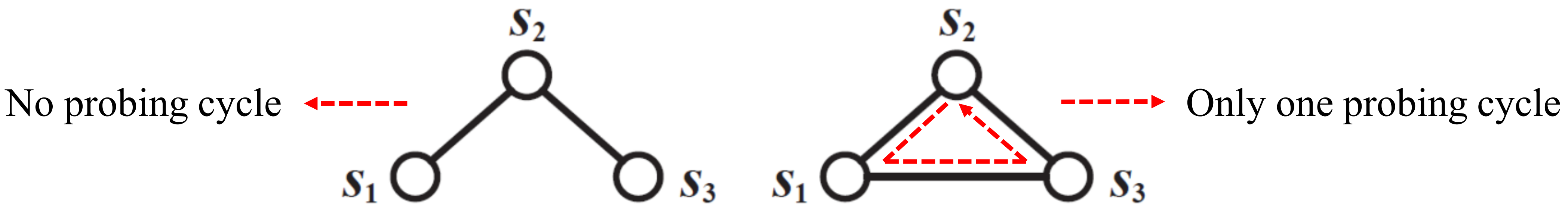
(f) Cross verification code for each link failure

Limitations of the existing cross verification

- In all-optical networks
  - A node can only be traversed at most once by each probing cycle
  - A link can only be traversed at most once by each probing cycle
    - This is because optical signals of the same wavelength can only be transmitted in one direction on each link

Limitations of the existing cross verification

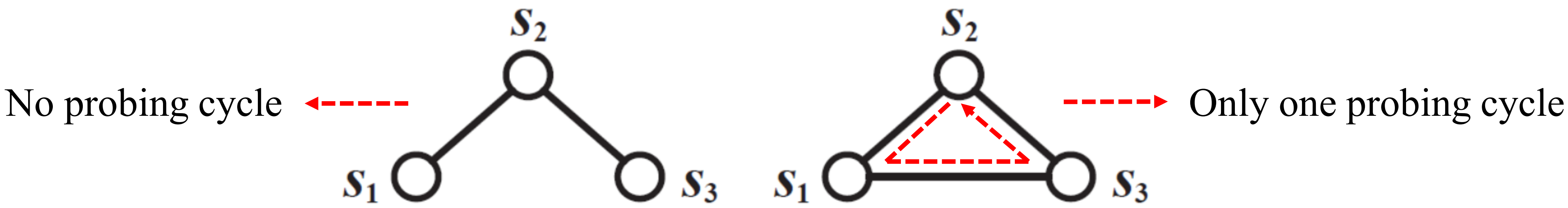
- In all-optical networks
  - A node can only be traversed at most once by each probing cycle
  - A link can only be traversed at most once by each probing cycle
    - This is because optical signals of the same wavelength can only be transmitted in one direction on each link
- "Failure localization" problem



(a) One-cut structure (b) Two-cut structure

Limitations of the existing cross verification

- In all-optical networks
  - A node can only be traversed at most once by each probing cycle
  - A link can only be traversed at most once by each probing cycle
    - This is because optical signals of the same wavelength can only be transmitted in one direction on each link
- “Failure localization” problem



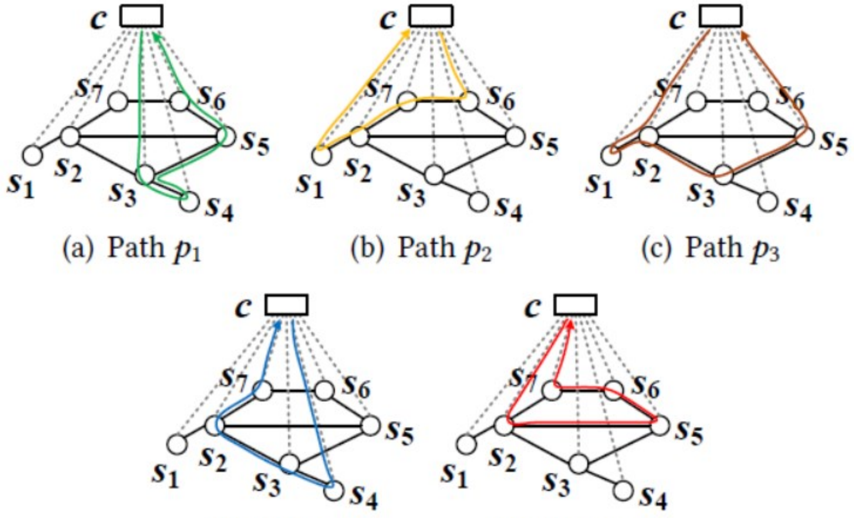
All links cannot be distinguished from each other.

Our cross verification

- In SDN networks
  - A node can be traversed multiple times by each probing cycle
  - ***Note***: A link can be traversed at most once in either direction by each probing cycle

Our cross verification

- In SDN networks
  - A node can be traversed multiple times by each probing cycle
  - *Note*: A link can be traversed at most once in either direction by each probing cycle



(a) Path p_1 (b) Path p_2 (c) Path p_3 (d) Path p_4 (e) Path p_5

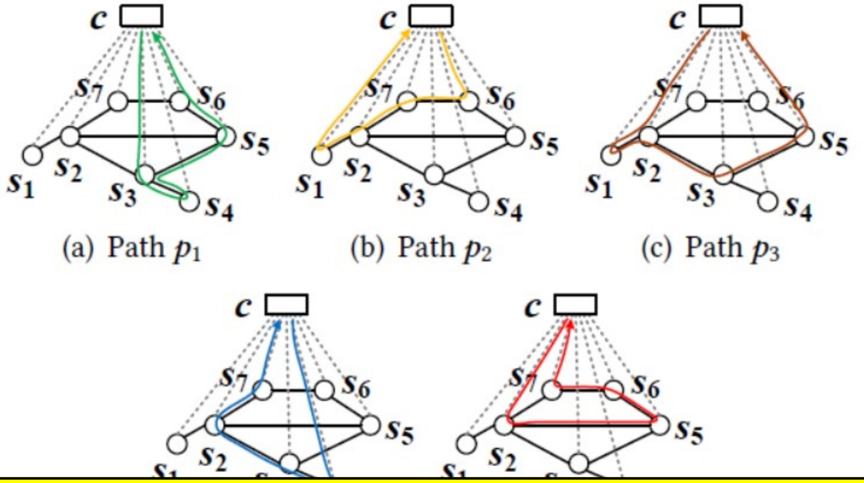
Link	Binary Code					Link	Binary Code				
	p_1	p_2	p_3	p_4	p_5		p_1	p_2	p_3	p_4	p_5
(s_1, s_2)	0	1	1	0	0	(c, s_1)	0	1	0	0	0
(s_2, s_3)	0	0	1	1	0	(c, s_2)	0	0	1	0	1
(s_2, s_5)	0	0	0	0	1	(c, s_3)	1	0	0	0	0
(s_2, s_7)	0	1	0	1	0	(c, s_4)	0	0	0	1	0
(s_3, s_4)	1	0	0	1	0	(c, s_5)	0	0	1	0	0
(s_3, s_5)	1	0	1	0	0	(c, s_6)	1	1	0	0	0
(s_5, s_6)	1	0	0	0	1	(c, s_7)	0	0	0	1	1
(s_6, s_7)	0	1	0	0	1						

(f) Cross verification code for each link failure

Example network with one-cut and two-cut links

Our cross verification

- In SDN networks
  - A node can be traversed multiple times by each probing cycle
  - *Note*: A link can be traversed at most once in either direction by each probing cycle



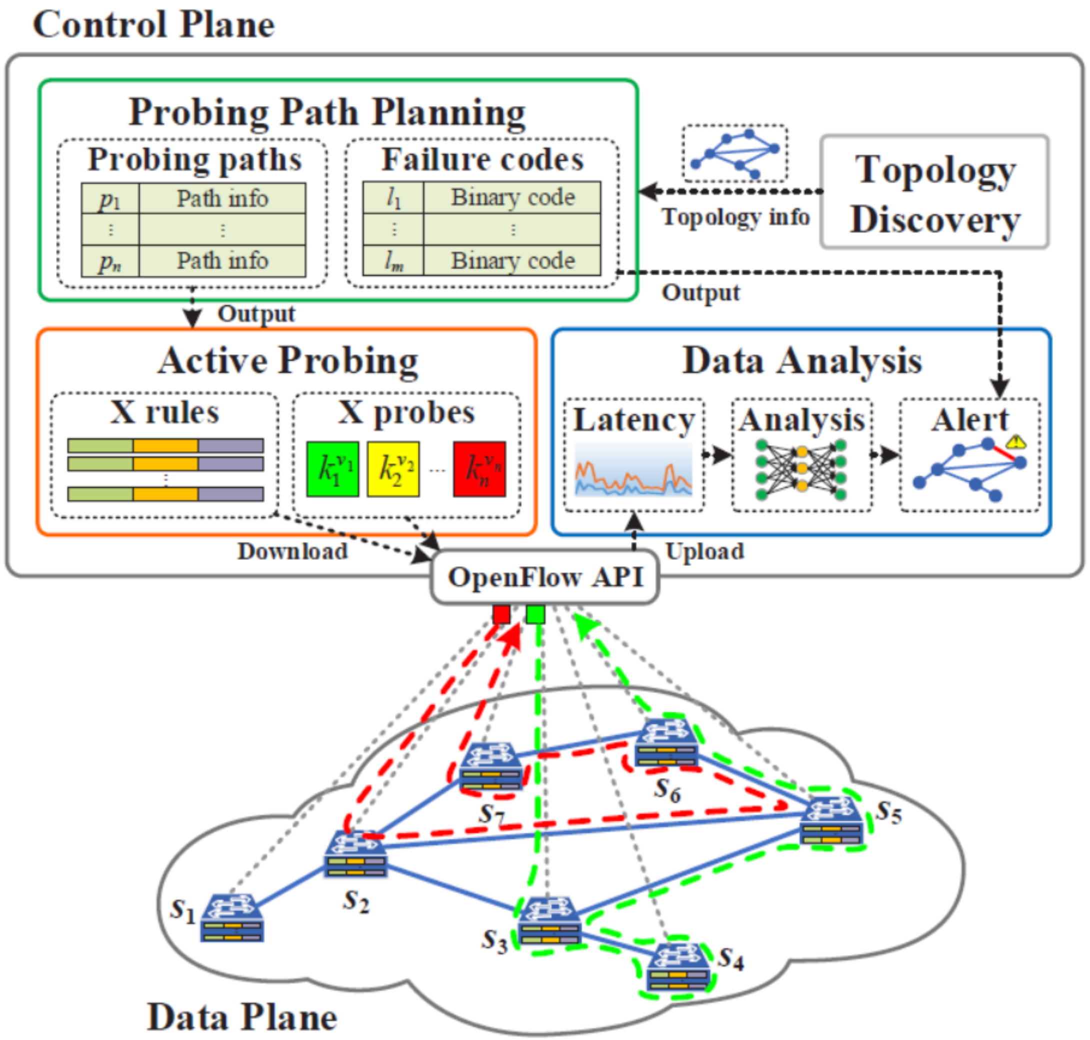
(a) Path p_1 (b) Path p_2 (c) Path p_3

Link	Binary Code					Link	Binary Code				
	p_1	p_2	p_3	p_4	p_5		p_1	p_2	p_3	p_4	p_5
(s_1, s_2)	0	1	1	0	0	(c, s_1)	0	1	0	0	0
(s_2, s_3)	0	0	1	1	0	(c, s_2)	0	0	1	0	1
(s_2, s_5)	0	0	0	0	1	(c, s_3)	1	0	0	0	0
(s_2, s_7)	0	1	0	1	0	(c, s_4)	0	0	0	1	0
(s_3, s_4)	1	0	0	1	0	(c, s_5)	0	0	1	0	0
(s_3, s_5)	1	0	1	0	0	(c, s_6)	1	1	0	0	0
(s_5, s_6)	1	0	0	0	1	(c, s_7)	0	0	0	1	1
(s_6, s_7)	0	1	0	0	1						

All links can be distinguished from each other.

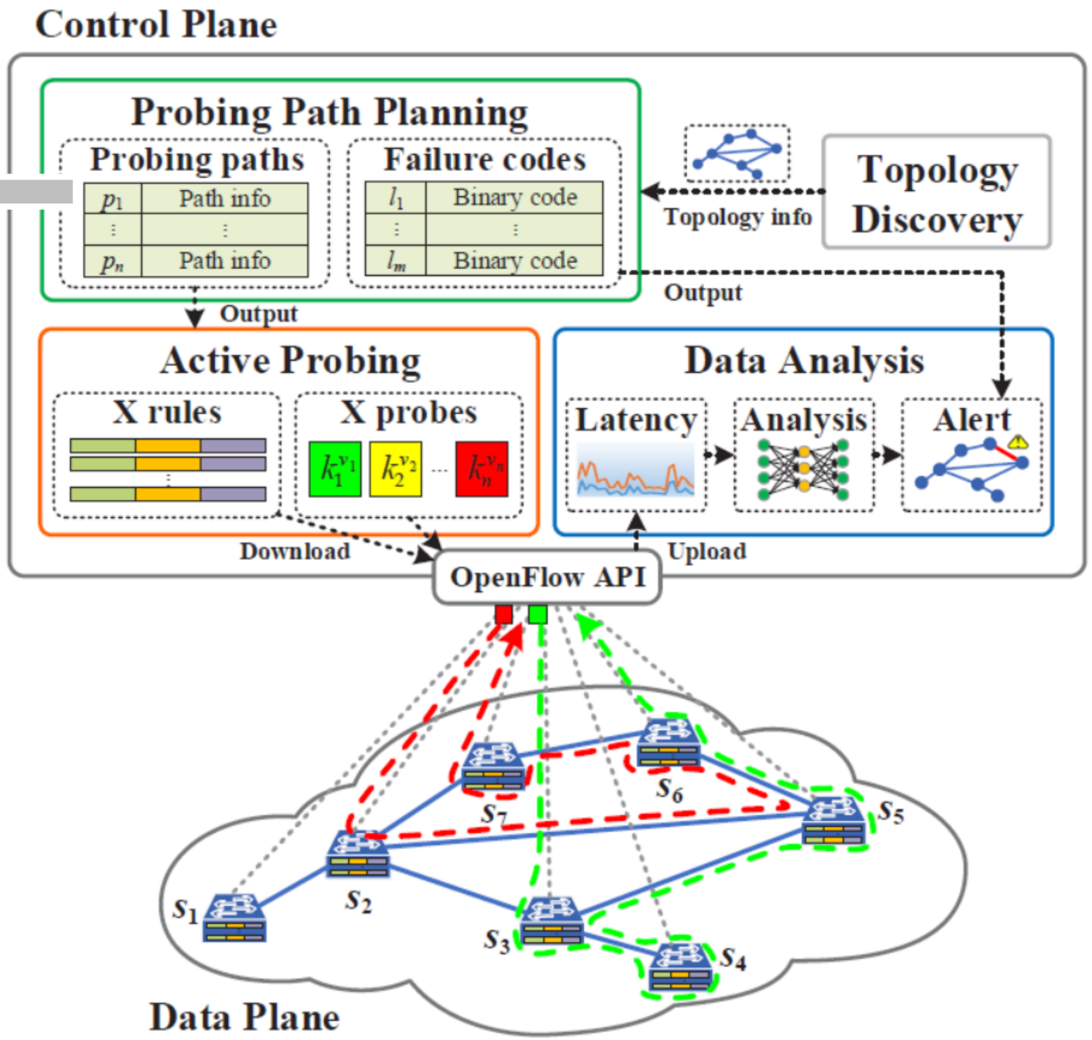
Overall design of *XShot*

- Three components
  - Probing path planning
  - Active probing
  - Data analysis



Overall design of *XShot*

Probing path planning: Given the network topology, it generates a probing solution consisting of probing paths and failure codes by ILP model



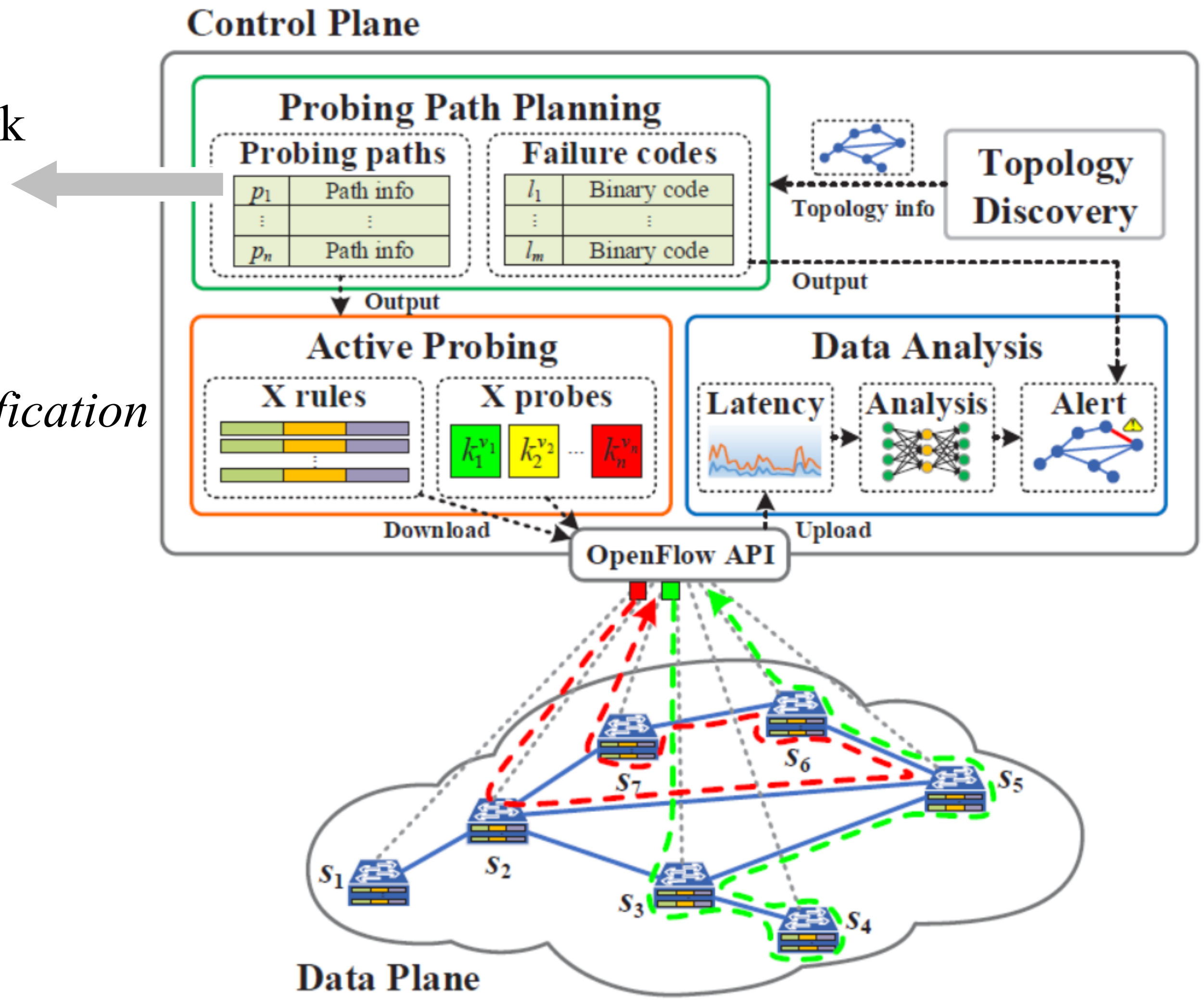
Overall design of *XShot*

Probing path planning: Given the network topology, it generates a probing solution consisting of probing paths and failure codes by ILP model

ILP model: Formulated based on *cross verification*

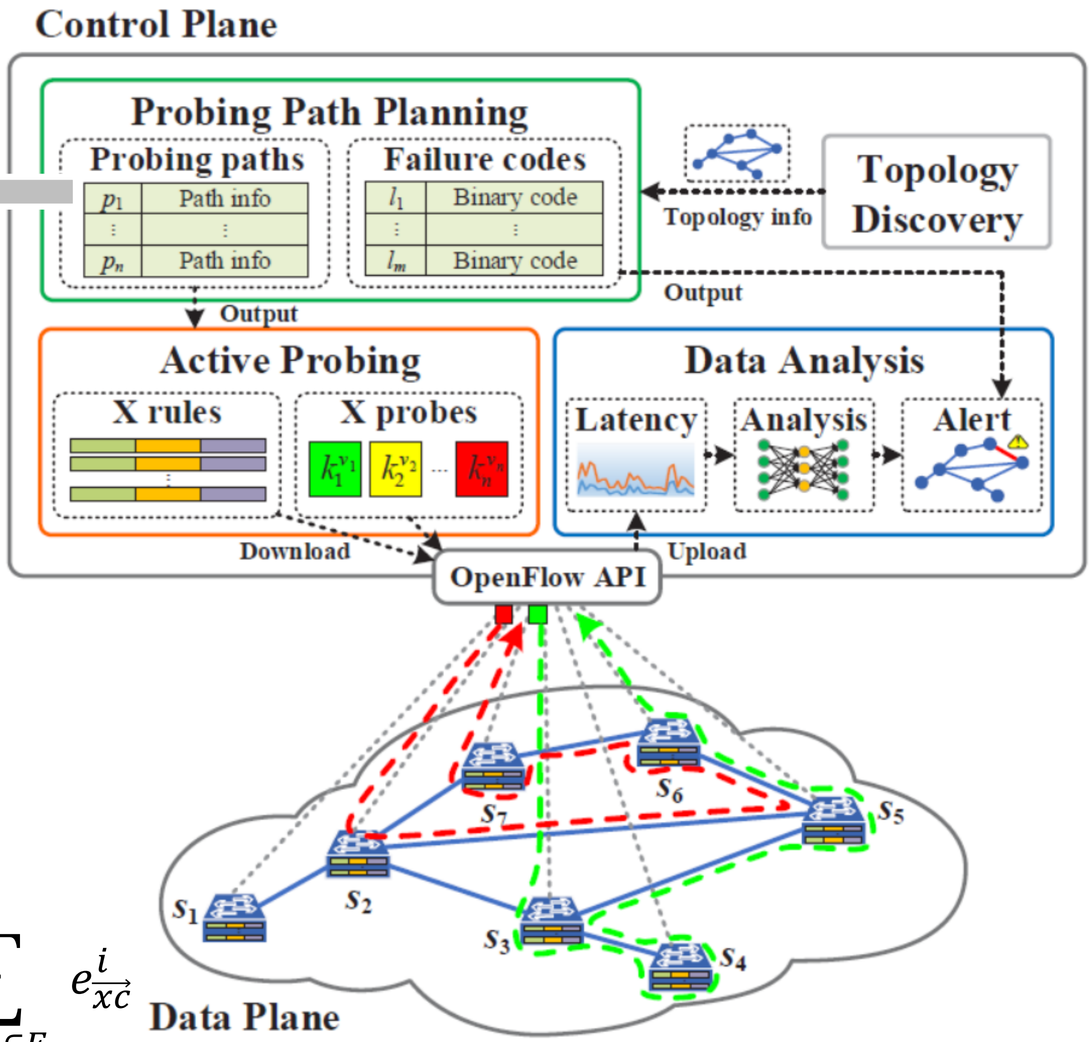
Objective:

$$min \quad \omega \times c_{pkt} + c_{rule}$$



Overall design of *XShot*

Probing path planning: Given the network topology, it generates a probing solution consisting of probing paths and failure codes by ILP model



ILP model: Formulated based on *cross verification*

Objective: A weight, *w*>1

$$\min \quad \omega \times c_{pkt} + c_{rule}$$

Probing packet cost:

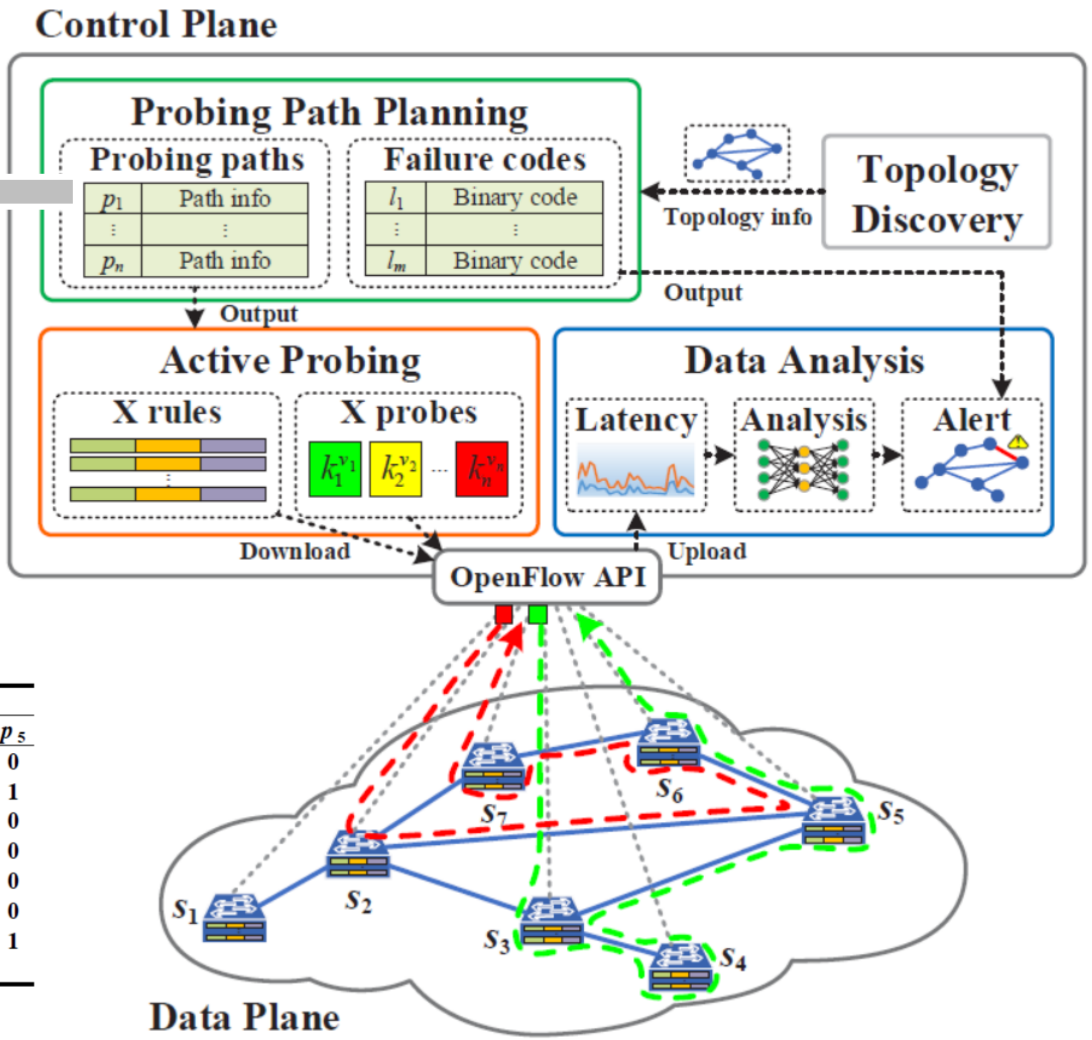
$$c_{pkt} = \sum_{i} \sum_{(c,y)\in E_c} e^{i}_{\overrightarrow{cy}}$$

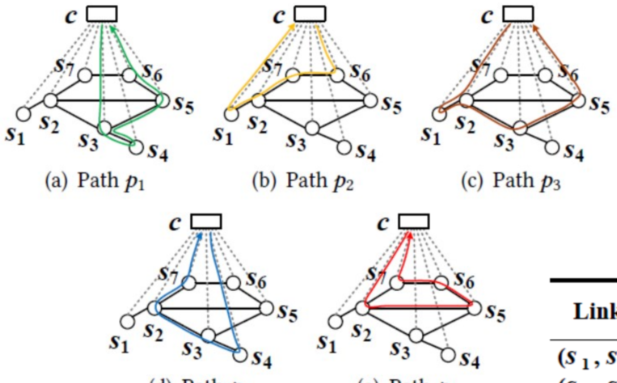
Forwarding rule cost:

$$c_{rule} = \sum_{i} \sum_{(x,y)\in E_d} \left(e^{i}_{\overrightarrow{xy}} + e^{i}_{\overrightarrow{yx}}\right) + \sum_{i} \sum_{(x,c)\in E_c} e^{i}_{\overrightarrow{xc}}$$

Overall design of *XShot*

Probing path planning: Given the network topology, it generates a probing solution consisting of probing paths and failure codes by ILP model





(a) Path p_1 (b) Path p_2 (c) Path p_3 (d) Path p_4 (e) Path p_5

Five probing paths

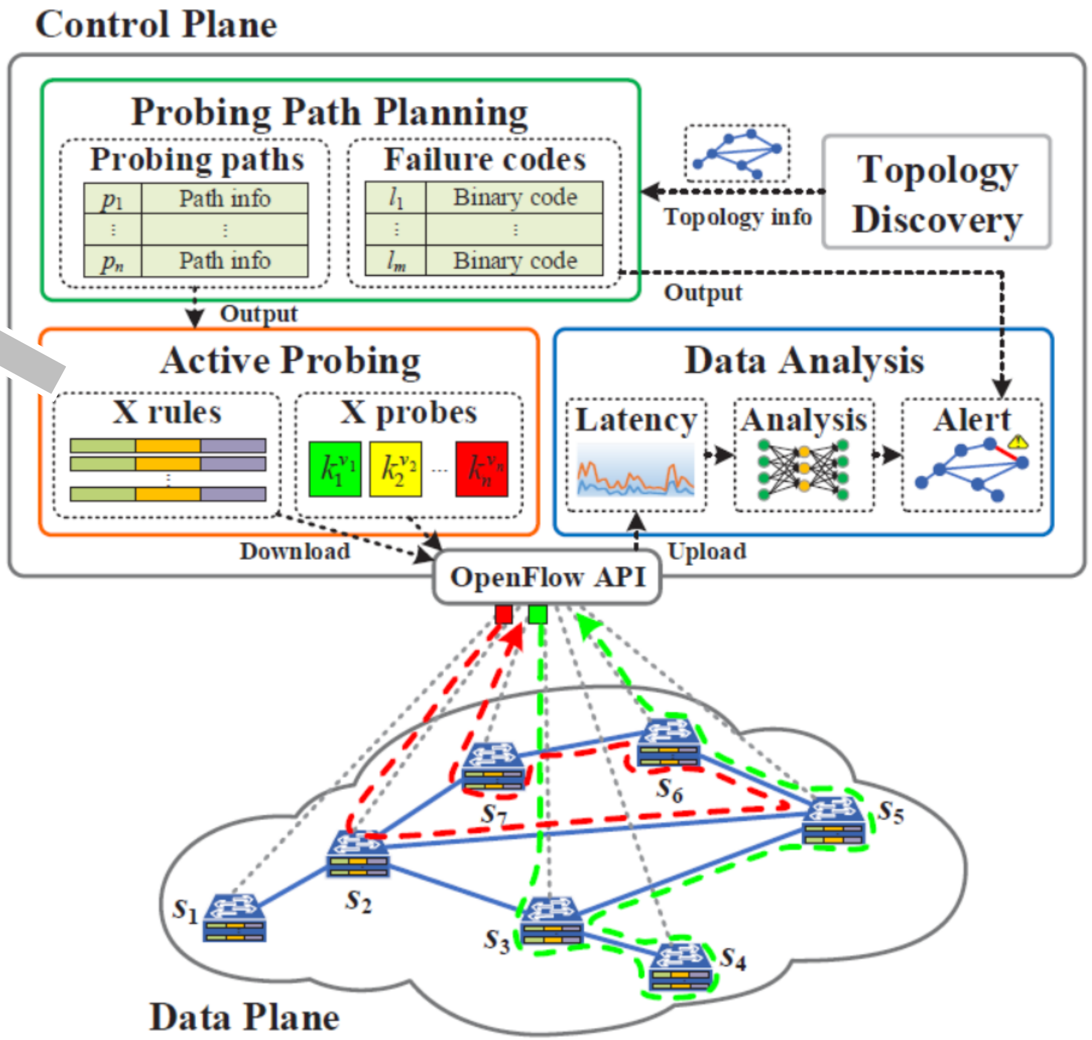
Failure codes of 15 links

Link	Binary Code					Link	Binary Code				
	p_1	p_2	p_3	p_4	p_5		p_1	p_2	p_3	p_4	p_5
(s_1, s_2)	0	1	1	0	0	(c, s_1)	0	1	0	0	0
(s_2, s_3)	0	0	1	1	0	(c, s_2)	0	0	1	0	1
(s_2, s_5)	0	0	0	0	1	(c, s_3)	1	0	0	0	0
(s_2, s_7)	0	1	0	1	0	(c, s_4)	0	0	0	1	0
(s_3, s_4)	1	0	0	1	0	(c, s_5)	0	0	1	0	0
(s_3, s_5)	1	0	1	0	0	(c, s_6)	1	1	0	0	0
(s_5, s_6)	1	0	0	0	1	(c, s_7)	0	0	0	1	1
(s_6, s_7)	0	1	0	0	1						

(f) Cross verification code for each link failure

Overall design of *XShot*

Active probing: It installs the forwarding rules on switches according to the probing paths, and sends packets along them to measure the end-to-end latency



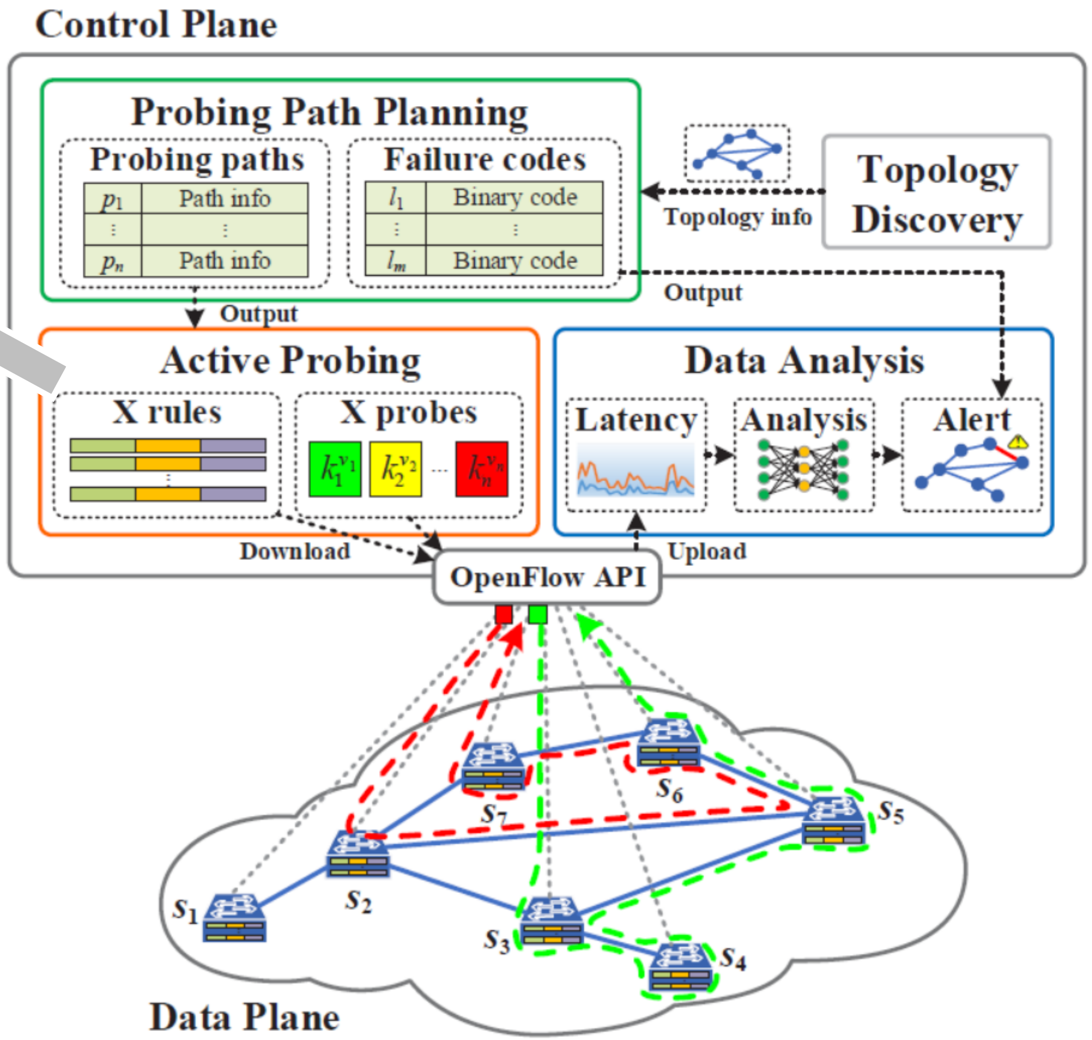
Overall design of *XShot*

Active probing: It installs the forwarding rules on switches according to the probing paths, and sends packets along them to measure the end-to-end latency

Table 2: Forwarding rules for path p_1

Switch	Forwarding Rule	
	Match Fields	***Actions***
S_3	VLAN == $vlan_1$, inport == CONTR[a]	output = $port_3^4$
	VLAN == $vlan_1$, inport == $port_3^4$	output = $port_3^5$
S_4	VLAN == $vlan_1$, inport == $port_4^3$	output = INPORT
S_5	VLAN == $vlan_1$, inport == $port_5^3$	output = $port_5^6$
S_6	VLAN == $vlan_1$, inport == $port_6^5$	output = CONTR

[a] CONTR represents CONTROLLER.



Overall design of *XShot*

Active probing: It installs the forwarding rules on switches according to the probing paths, and sends packets along them to measure the end-to-end latency

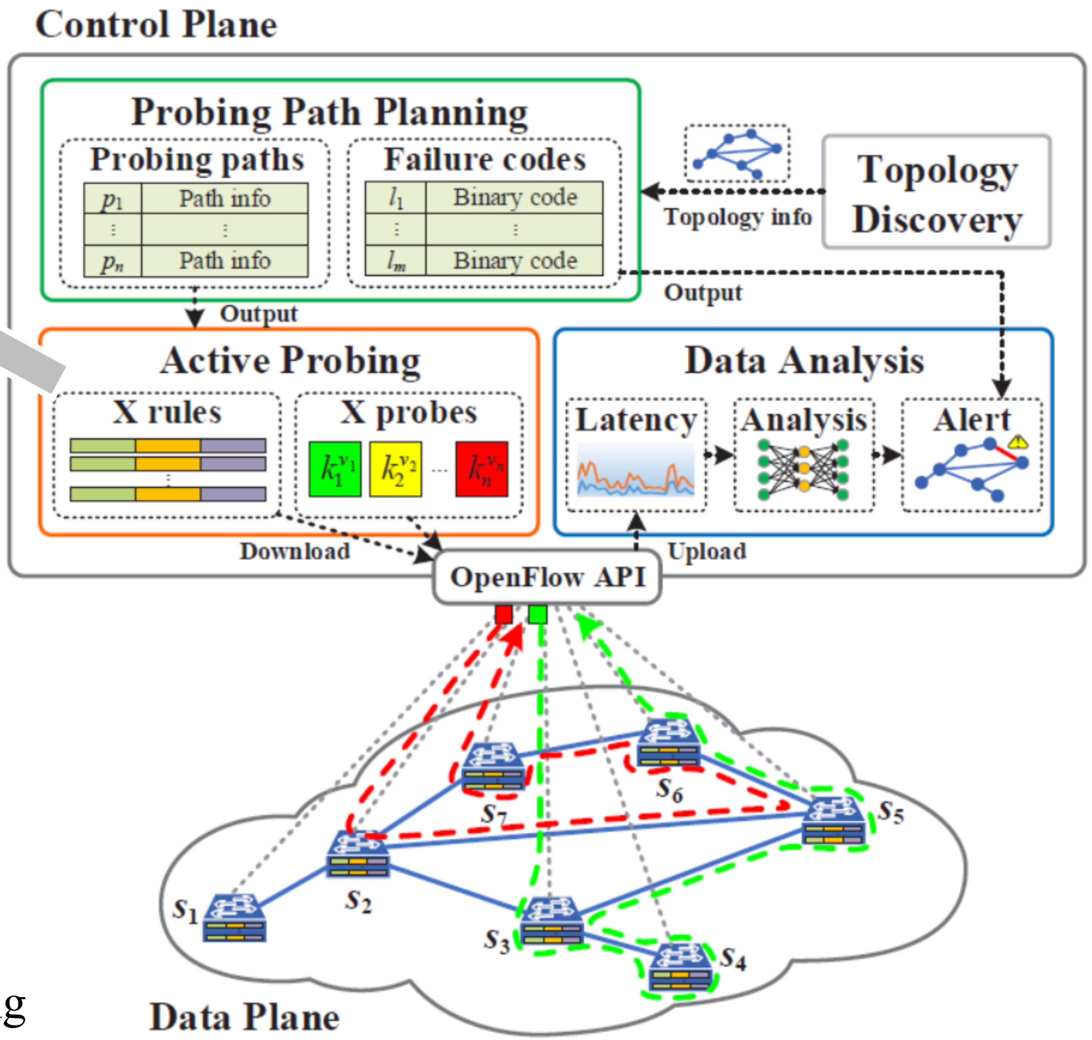
Table 2: Forwarding rules for path p_1

Switch	Forwarding Rule	
	Match Fields	***Actions***
S_3	VLAN == $vlan_1$, inport == CONTR[a]	output = $port_3^4$
	VLAN == $vlan_1$, inport == $port_3^4$	output = $port_3^5$
S_4	VLAN == $vlan_1$, inport == $port_4^3$	output = INPORT
S_5	VLAN == $vlan_1$, inport == $port_5^3$	output = $port_5^6$
S_6	VLAN == $vlan_1$, inport == $port_6^5$	output = CONTR

[a]CONTR represents CONTROLLER.

Path ID, using to distinguish the packets of different paths

Recording the sending time of the packet

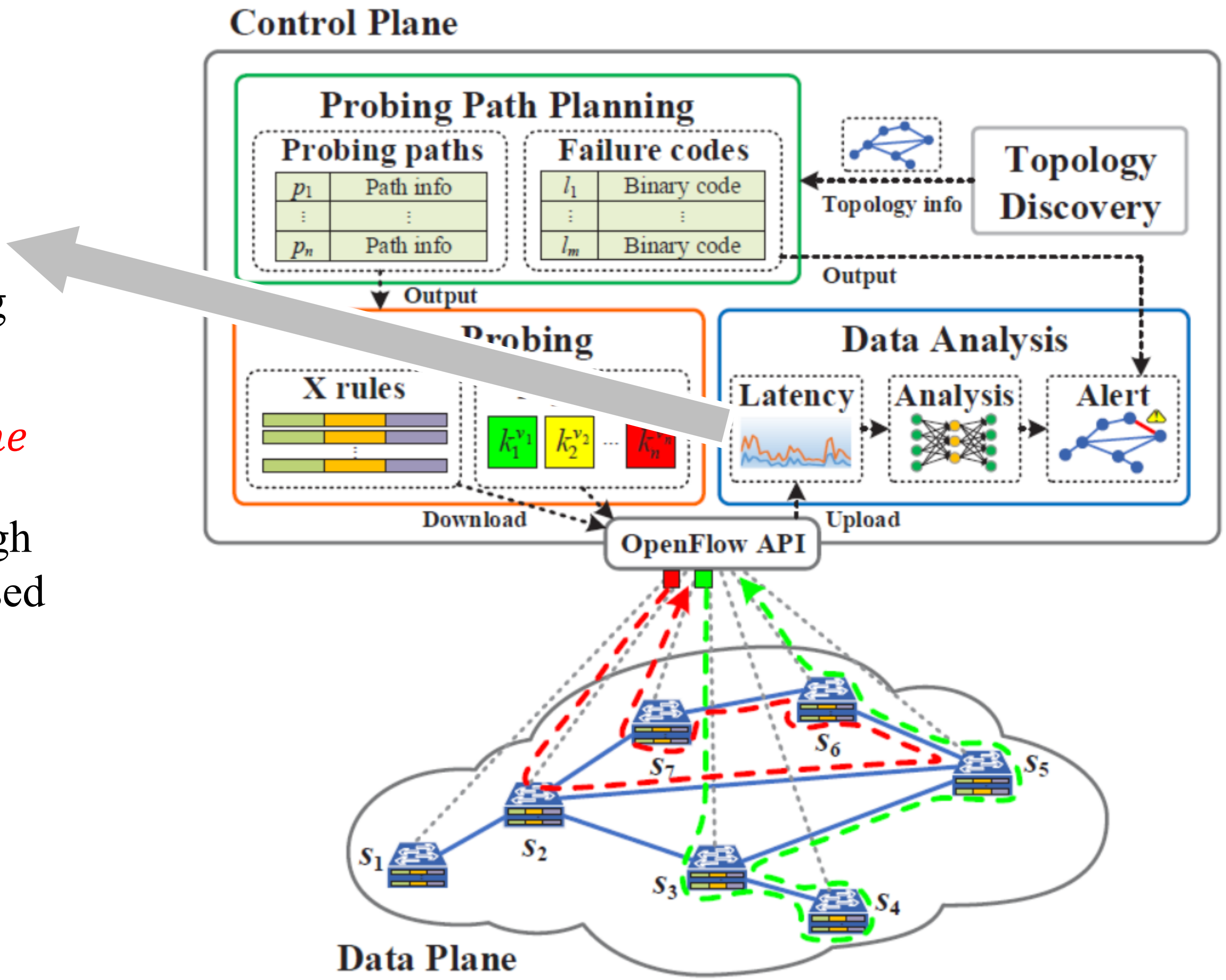


Overall design of *XShot*

Data analysis: It collects the measured latency, detects the path status using an unsupervised learning algorithm, and pinpoints the exact faulty link according to the unique binary code

$*latency = receiving\ time - sending\ time$

To detect the partial failures only causing high latency, *XShot* chooses *Donut*, an unsupervised anomaly detection algorithm based on VAE



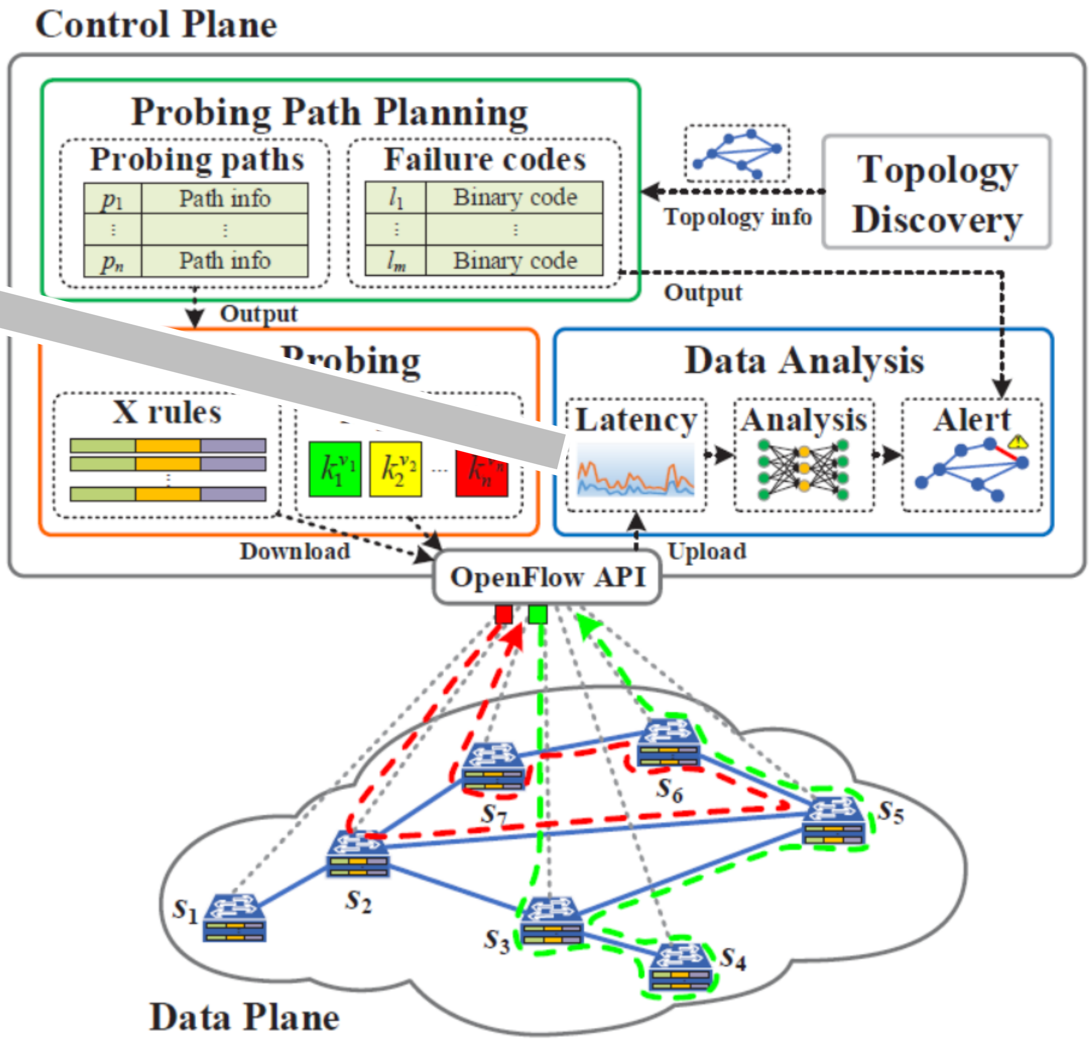
Overall design of *XShot*

Data analysis: It collects the measured latency, detects the path status using an unsupervised learning algorithm, and pinpoints the exact faulty link according to the unique binary code

$*latency = receiving\ time - sending\ time$

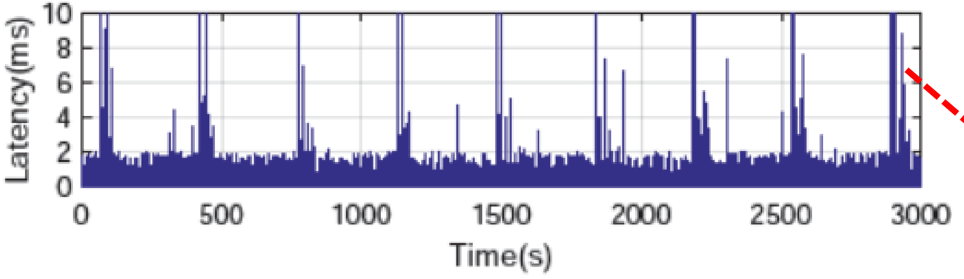
To detect the partial failures only causing high latency, *XShot* chooses *Donut*, an unsupervised anomaly detection algorithm based on VAE

Transient unexpected fluctuations exist in the measured data.



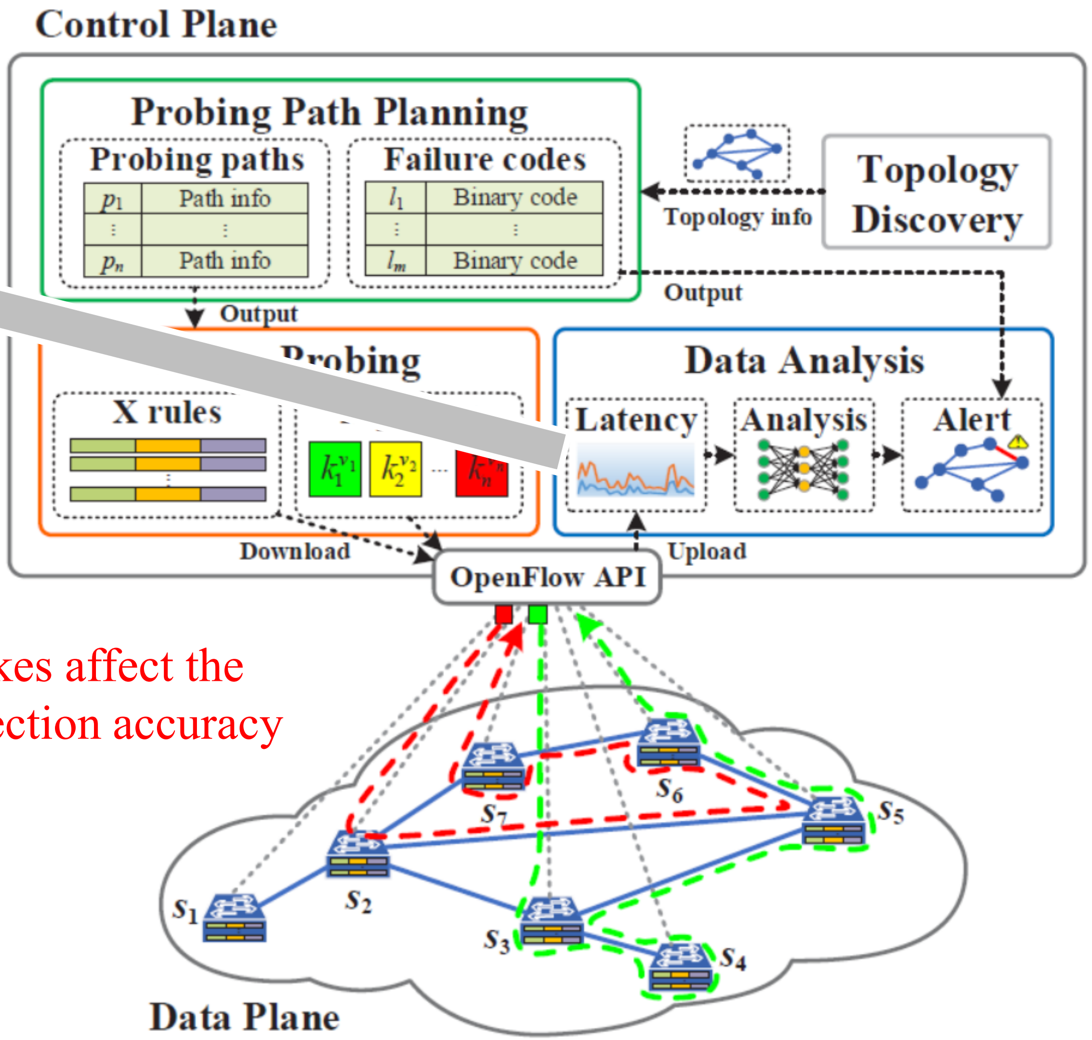
Overall design of *XShot*

Data analysis: It collects the measured latency, detects the path status using an unsupervised learning algorithm, and pinpoints the exact faulty link according to the unique binary code



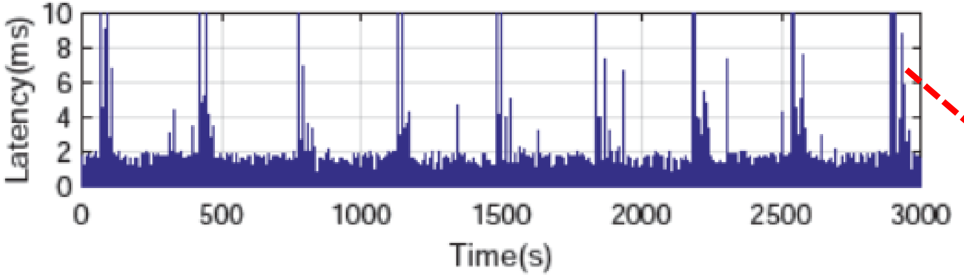
(a) Latency

Spikes affect the detection accuracy



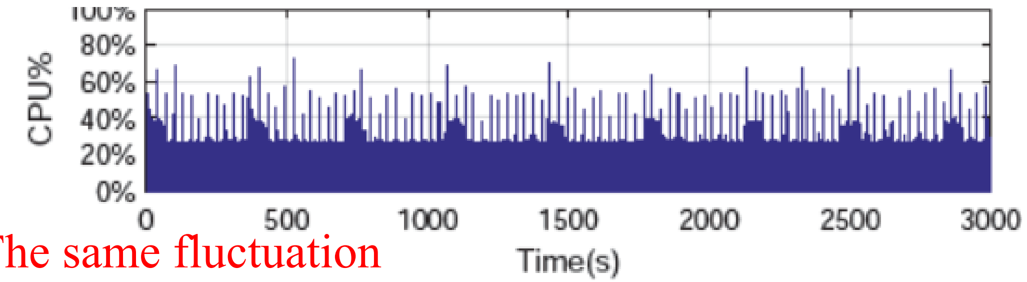
Overall design of *XShot*

Data analysis: It collects the measured latency, detects the path status using an unsupervised learning algorithm, and pinpoints the exact faulty link according to the unique binary code



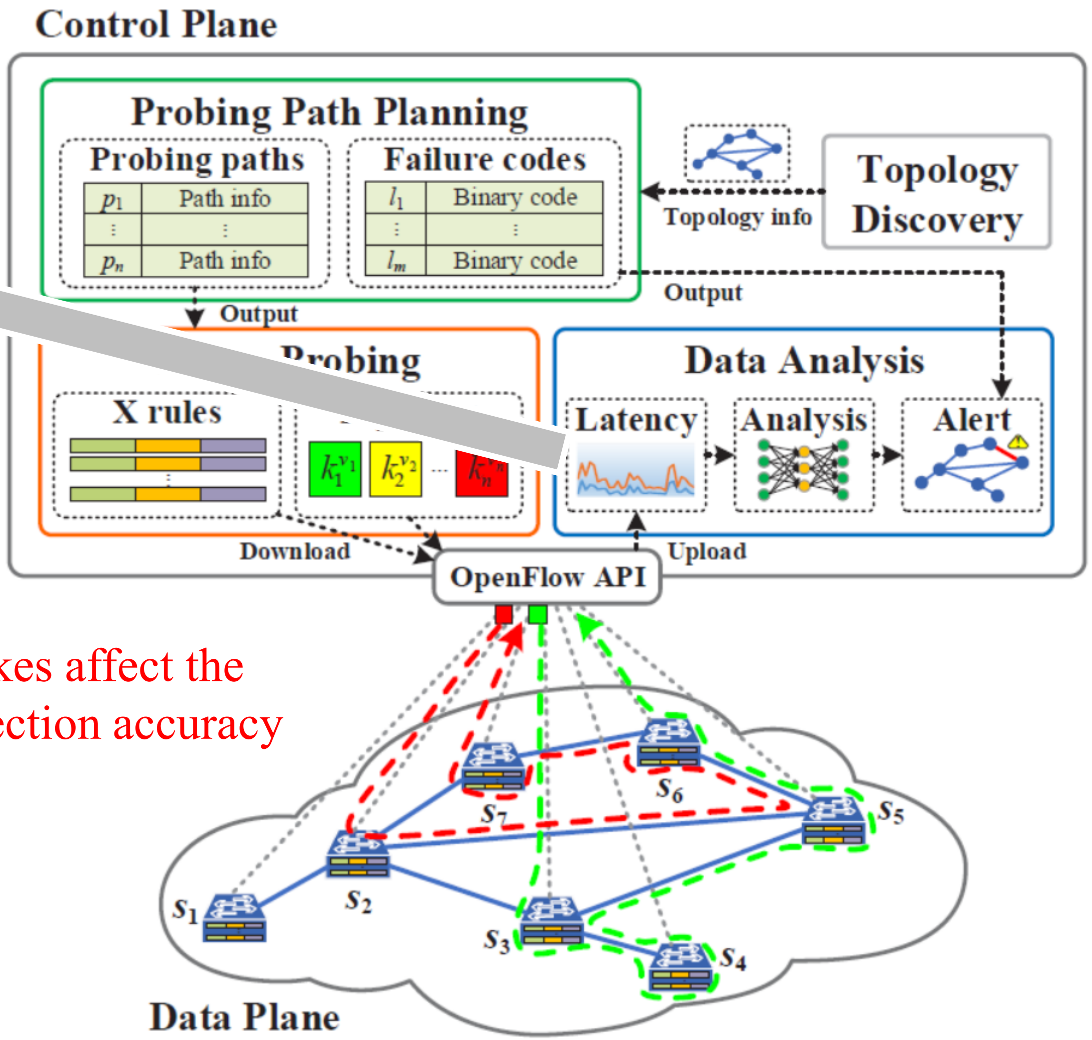
(a) Latency

Spikes affect the detection accuracy



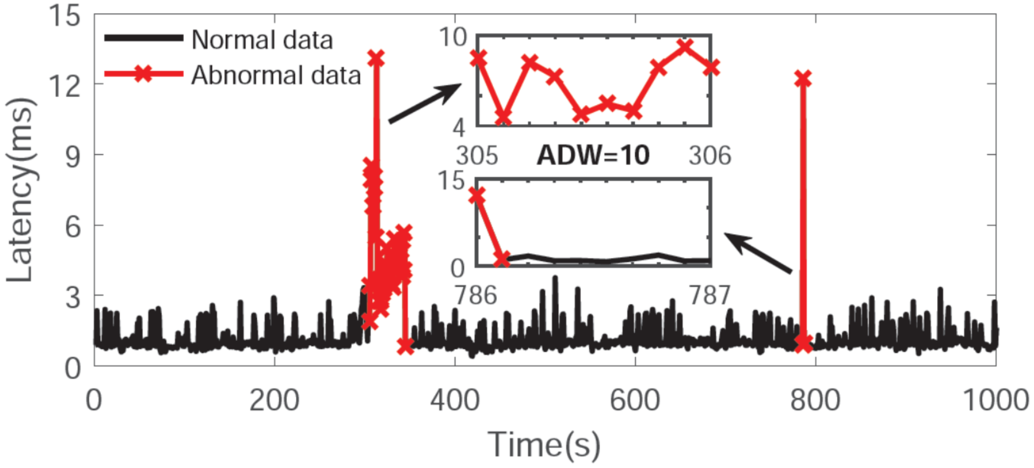
(b) CPU usage

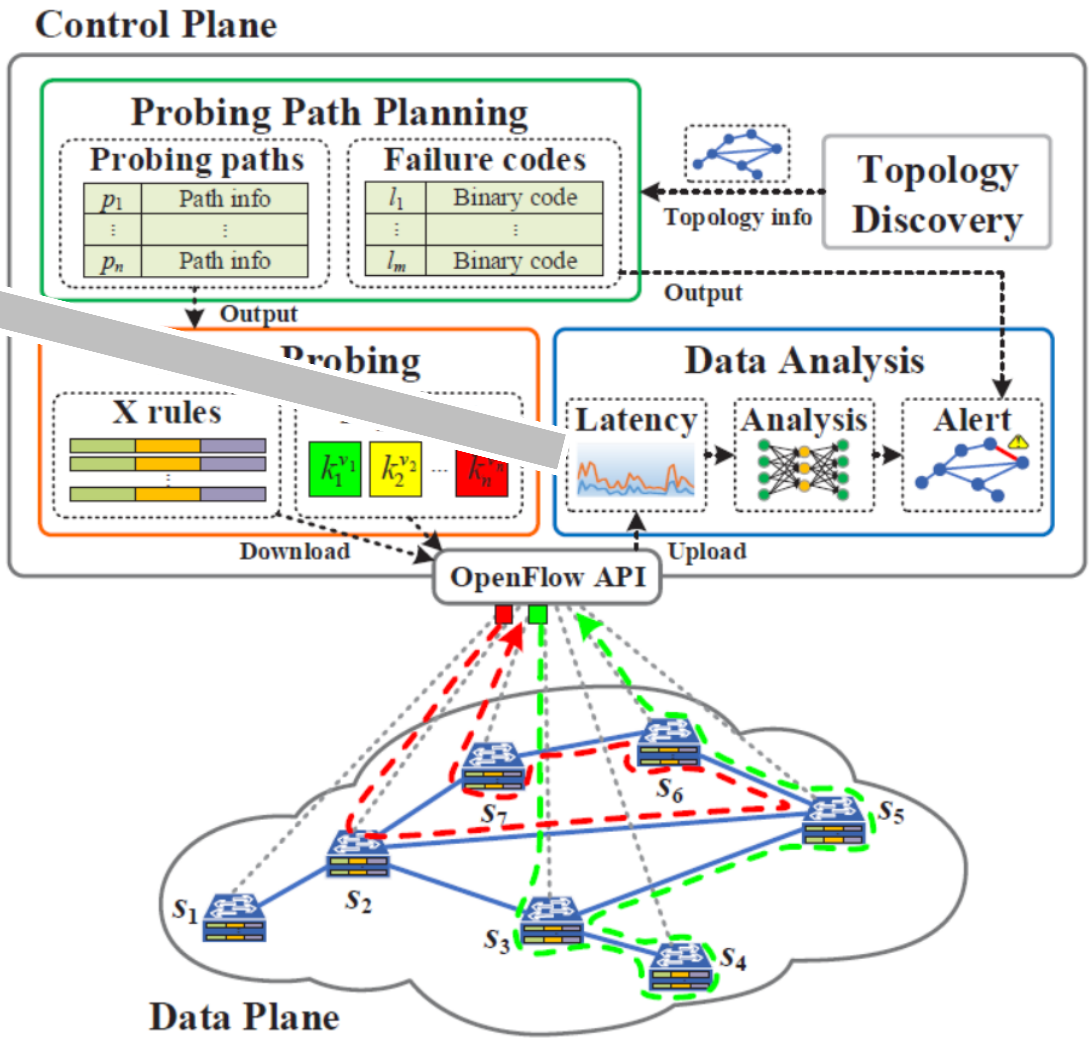
The same fluctuation frequency



Overall design of *XShot*

Data analysis: It collects the measured latency, detects the path status using an unsupervised learning algorithm, and pinpoints the exact faulty link according to the unique binary code

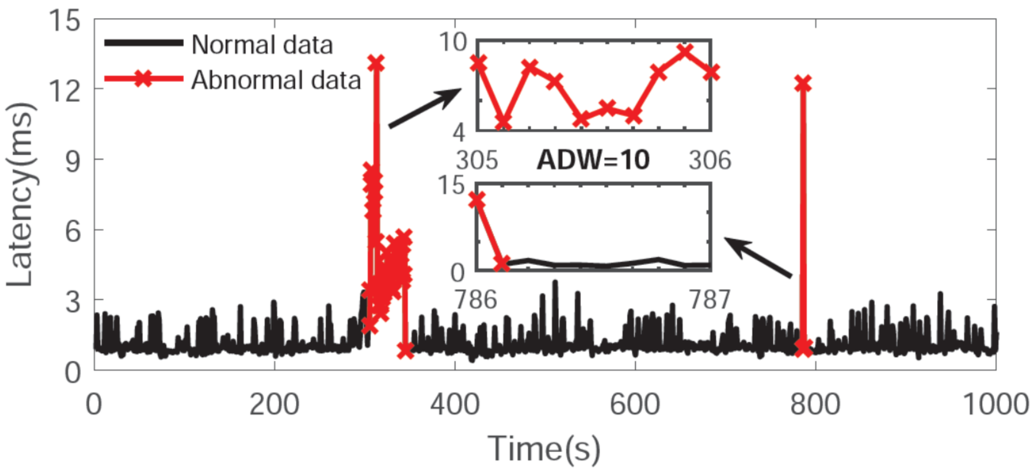




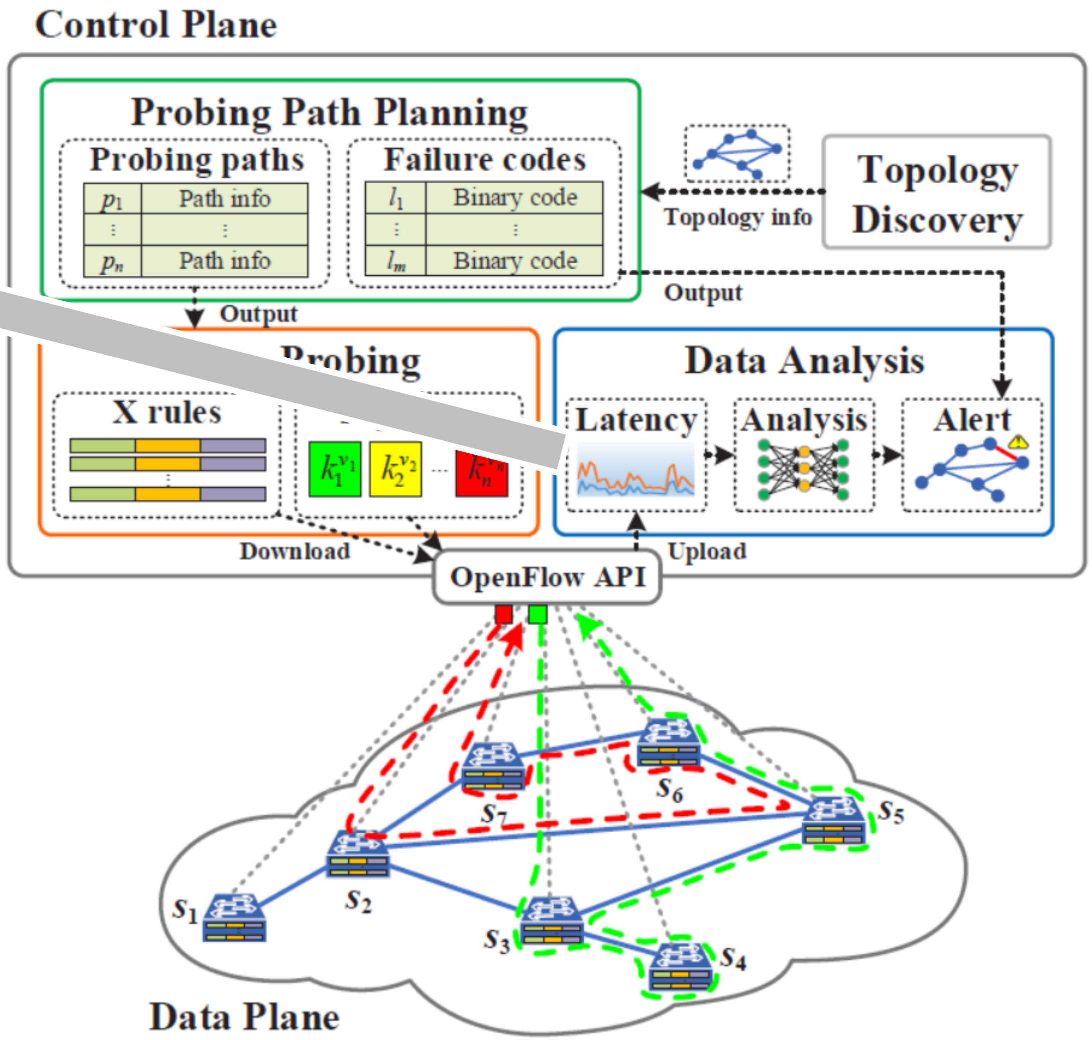
ADW-Donut: Introduce an accelerated detection window (ADW) into Donut

Overall design of *XShot*

Data analysis: It collects the measured latency, detects the path status using an unsupervised learning algorithm, and pinpoints the exact faulty link according to the unique binary code

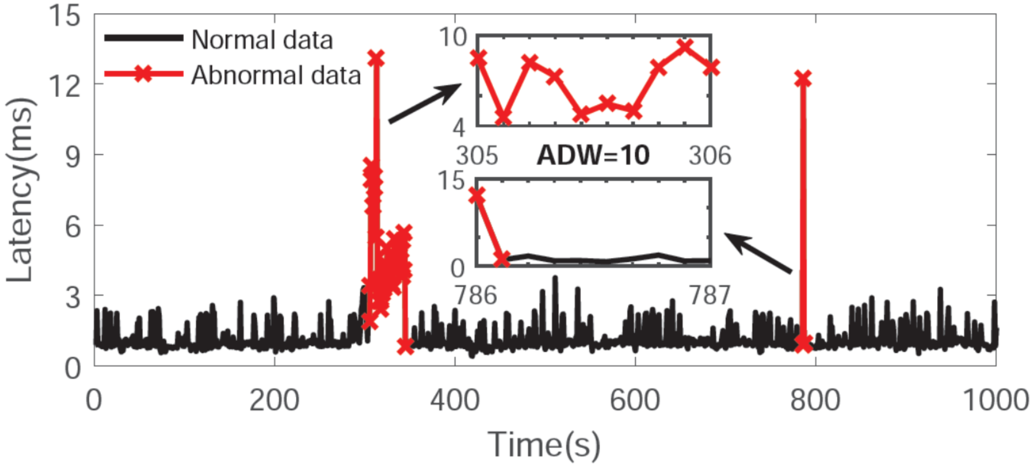


(i) Upon an anomaly, send a certain number (i.e., ADW) of additional probing packets *in a higher frequency*

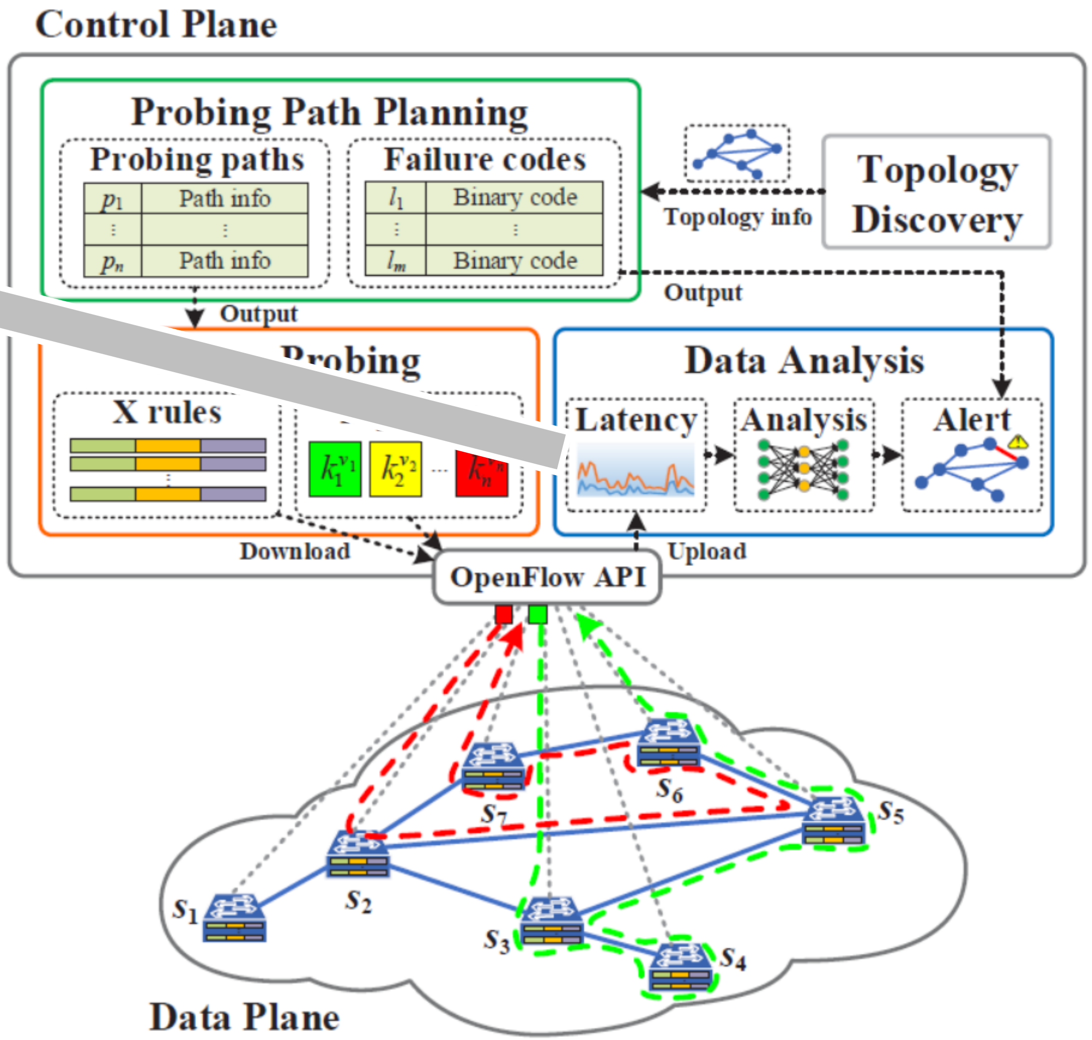


Overall design of *XShot*

Data analysis: It collects the measured latency, detects the path status using an unsupervised learning algorithm, and pinpoints the exact faulty link according to the unique binary code

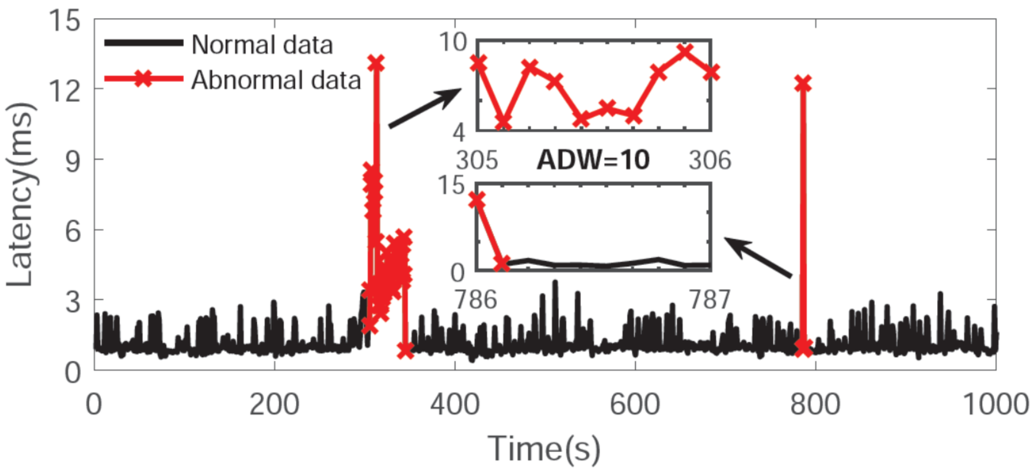


(ii) If there are *more detected anomalies* in ADW than a threshold, the detection result of Donut is *true positive*

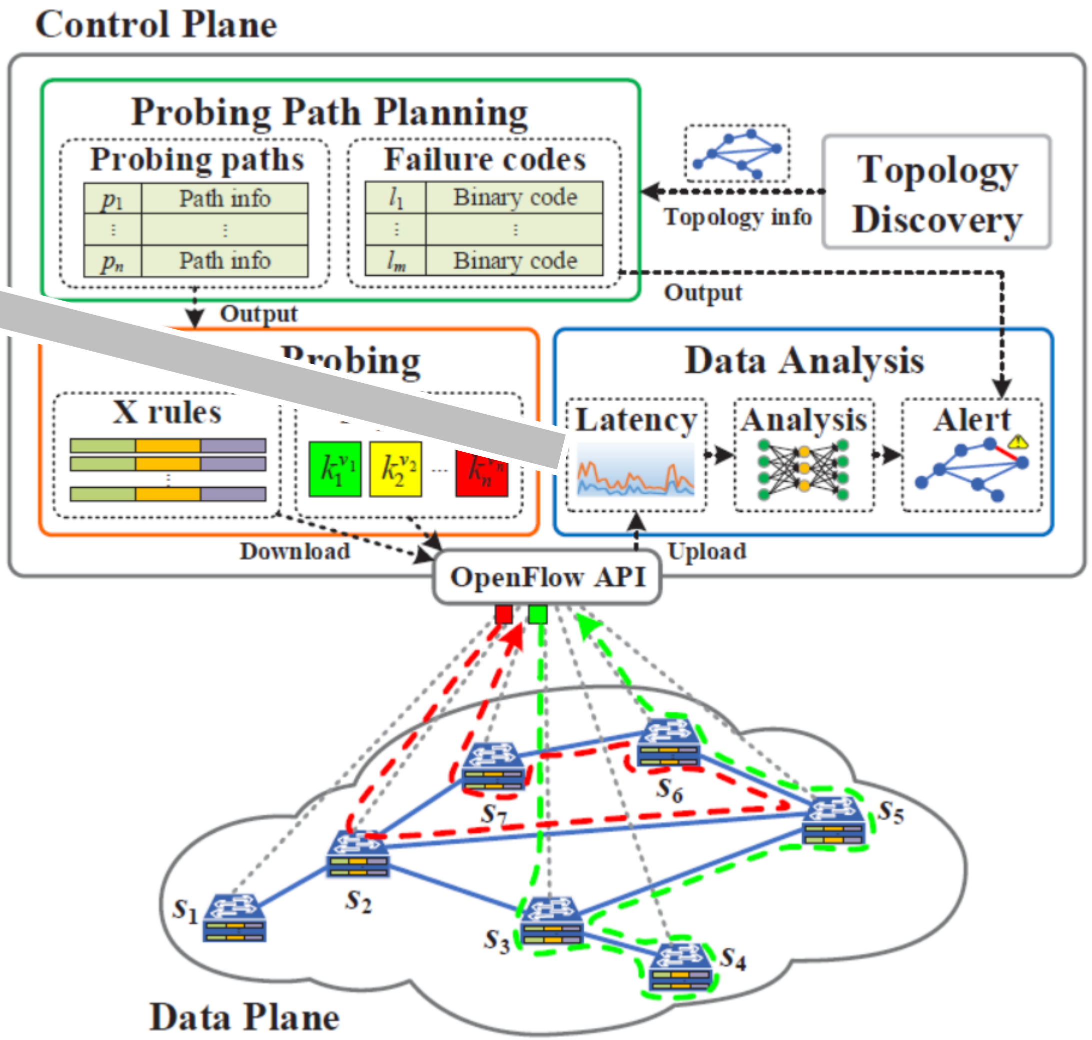


Overall design of *XShot*

Data analysis: It collects the measured latency, detects the path status using an unsupervised learning algorithm, and pinpoints the exact faulty link according to the unique binary code



(iii) Otherwise, the result is false positive and removed



Evaluation

- Set up
  - Experimental environment
    - Choose *Floodlight* as the SDN controller
    - Use *Mininet* to create an SDN network
    - Collect 63 available network topologies from the *Internet Topology Zoo*
    - Set a centralized controller on the control plane
    - The probing interval is 1 second, and ADW=10
  - Compared approaches
    - Link Layer Discovery Protocol (LLDP)
    - Logical Ring [*TON'16*]

Evaluation

- Set up
  - Metrics
    - The number of probing packets and forwarding rules
    - The failure detection precision: $precision = \frac{TP}{TP+FP}, recall = \frac{TP}{TP+FN}$
    - Controller overhead: CPU and memory usage

Evaluation

- Results
  - Number of probing packets and forwarding rules

Evaluation

- Results

In 79.37% of topologies, *XShot* averagely requires 9.63% less number of probing packets than Logical Ring.

	Renam (5,4)	MREN (6,5)	GetNet (7,8)	AI3 (10,9)	Netrail (7,10)	Heanet (7,11)	EEnet (13,13)	Abilene (11,14)	ILAN (14,15)	GRENA (16,15)	Navi… (13,17)	Sago (18,17)	GARR (16,18)	RHnet (16,18)	Nextgen (17,19)	GridNet (9,20)	FatMan (17,21)	Azrena (22,21)	BSO… (18,23)	ISTAR (23,23)	Visio… (24,23)
#pkts of XShot	3	3	4	4	4	4	4	4	5	8	5	9	5	5	5	5	5	6	5	5	7
#pkts of Ring	4	4.5	4.5	5.5	4.5	4.5	5.5	5.5	5.5	5.5	5.5	6.5	6.5	5.5	5.5	5.5	5.5	6.5	6.5	6.5	6.5
#pkts of LLDP	8	10	16	18	20	22	26	28	30	30	34	34	36	36	38	40	42	42	46	46	46
#rules of XShot	2.00	2.33	2.71	2.80	3.00	3.29	3.62	3.82	3.29	4.88	3.31	7.00	3.44	4.50	5.24	4.44	3.82	4.00	3.78	4.17	5.42
#rules of Ring	4.80	5.00	4.43	5.40	4.43	4.86	5.31	4.27	5.29	5.63	5.31	5.67	6.25	4.38	3.76	7.33	5.29	5.73	5.39	5.61	5.75
#rules of LLDP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

	IBM (18,24)	BELN… (21,24)	York… (23,24)	URAN (24,24)	AMRES (25,24)	ANS (18,25)	EasyNet (19,26)	Uni-C (25,27)	KAREN (25,28)	ARN (30,29)	Elect… (20,30)	FUNET (26,30)	Good… (17,31)	Quest (20,31)	ACOnet (23,31)	Darks… (28,31)	ARPA… (29,32)	ERNET (30,32)	Czech (32,33)	Xeex (24,34)	IINET (31,35)
#pkts of XShot	5	6	10	6	8	5	5	6	6	6	5	7	6	6	6	6	9	7	7	6	6
#pkts of Ring	5.5	5.5	5.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5
#pkts of LLDP	48	48	48	48	48	50	52	54	56	58	60	60	62	62	62	62	64	64	66	68	70
#rules of XShot	4.00	4.67	8.00	4.08	4.56	3.83	4.37	5.40	4.40	4.13	4.60	5.65	4.76	4.95	4.30	6.36	7.03	4.63	5.06	4.67	4.16
#rules of Ring	4.72	3.48	3.57	5.50	5.76	5.67	5.42	4.56	5.00	5.80	5.35	4.04	6.53	5.45	4.96	3.61	4.17	5.63	4.34	5.08	5.81
#rules of LLDP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

	Globa… (9,36)	Reuna (37,36)	Slovakia (35,37)	GEANT (27,38)	Myren (37,39)	Canerie (32,41)	Carnet (44,43)	Janet… (29,45)	SANET (43,45)	ARNES (34,46)	Lamb… (42,46)	Valley… (39,51)	RoE… (48,52)	CUDI (51,52)	ATT… (25,56)	Renater (43,56)	IIJ (37,65)	China… (42,66)	SURF… (50,68)	North… (36,76)	UUNET (49,84)
#pkts of XShot	6	7	6	6	6	7	7	7	7	6	25	8	9	7	8	9	9	14	20	15	19
#pkts of Ring	6.5	7.5	7.5	6.5	7.5	6.5	7.5	6.5	7.5	6.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5
#pkts of LLDP	72	72	74	76	78	82	86	90	90	92	92	102	104	104	112	112	130	132	136	152	168
#rules of XShot	10.44	4.92	4.34	4.67	4.62	5.16	4.50	6.21	5.93	6.09	6.74	5.87	5.63	5.20	11.08	6.02	6.92	7.45	4.56	10.17	6.27
#rules of Ring	12.00	5.84	5.54	5.85	5.76	5.19	5.86	4.62	4.63	4.82	4.29	5.10	4.92	5.61	7.64	4.67	6.24	6.26	4.88	8.00	6.37
#rules of LLDP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Evaluation

- Results

XShot and Logical Ring require roughly the same number of forwarding rules, which commonly occupy less than 0.1% of TCAM resources.

	Renam (5,4)	MREN (6,5)	GetNet (7,8)	AI3 (10,9)	Netrail (7,10)	Heanet (7,11)	EEnet (13,13)	Abilene (11,14)	ILAN (14,15)	GRENA (16,15)	Navi… (13,17)	Sago (18,17)	GARR (16,18)	RHnet (16,18)	Nextgen (17,19)	GridNet (9,20)	FatMan (17,21)	Azrena (22,21)	BSO… (18,23)	ISTAR (23,23)	Visio… (24,23)
#pkts of XShot	3	3	4	4	4	4	4	4	5	8	5	9	5	5	5	5	5	6	5	5	7
#pkts of Ring	4	4.5	4.5	5.5	4.5	4.5	5.5	5.5	5.5	5.5	5.5	6.5	6.5	5.5	5.5	5.5	5.5	6.5	6.5	6.5	6.5
#pkts of LLDP	8	10	16	18	20	22	26	28	30	30	34	34	36	36	38	40	42	42	46	46	46
#rules of XShot	2.00	2.33	2.71	2.80	3.00	3.29	3.62	3.82	3.29	4.88	3.31	7.00	3.44	4.50	5.24	4.44	3.82	4.00	3.78	4.17	5.42
#rules of Ring	4.80	5.00	4.43	5.40	4.43	4.86	5.31	4.27	5.29	5.63	5.31	5.67	6.25	4.38	3.76	7.33	5.29	5.73	5.39	5.61	5.75
#rules of LLDP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

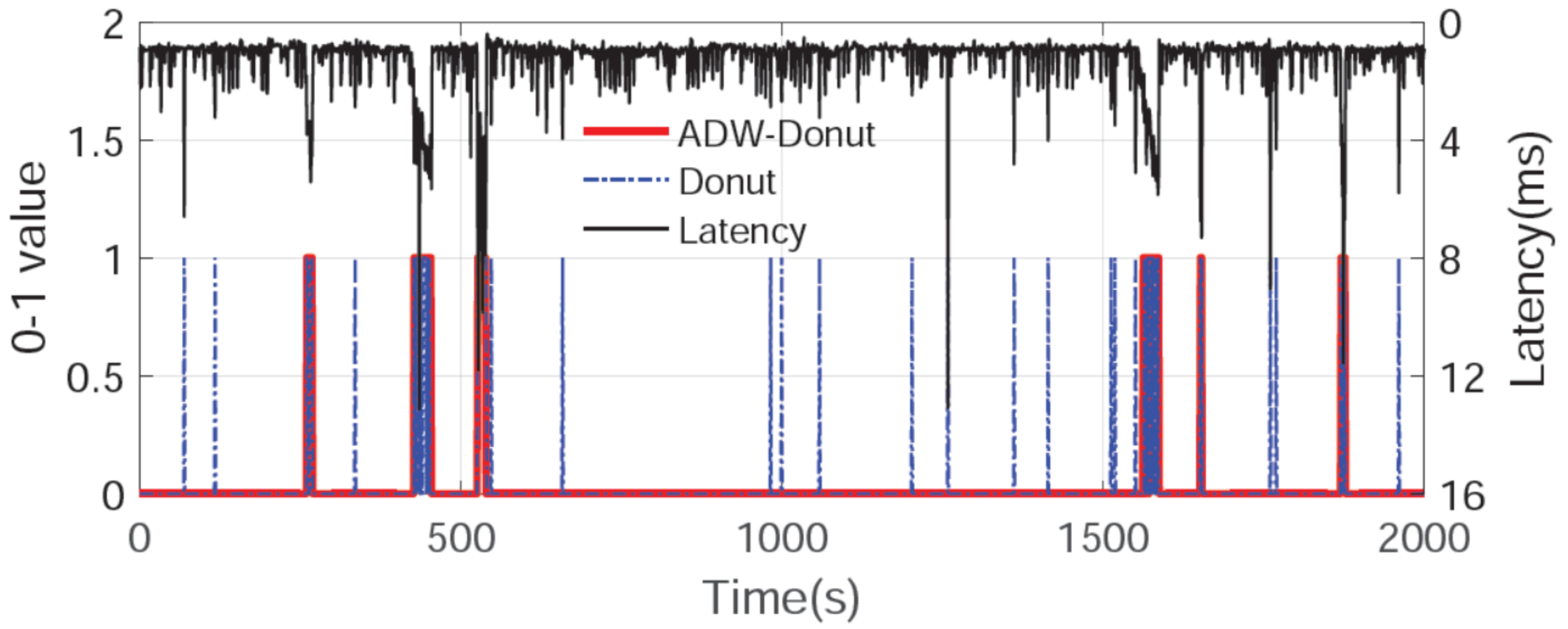
	IBM (18,24)	BELN… (21,24)	York… (23,24)	URAN (24,24)	AMRES (25,24)	ANS (18,25)	EasyNet (19,26)	Uni-C (25,27)	KAREN (25,28)	ARN (30,29)	Elect… (20,30)	FUNET (26,30)	Good… (17,31)	Quest (20,31)	ACOnet (23,31)	Darks… (28,31)	ARPA… (29,32)	ERNET (30,32)	Czech (32,33)	Xeex (24,34)	IINET (31,35)
#pkts of XShot	5	6	10	6	8	5	5	6	6	6	5	7	6	6	6	6	9	7	7	6	6
#pkts of Ring	5.5	5.5	5.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5
#pkts of LLDP	48	48	48	48	48	50	52	54	56	58	60	60	62	62	62	62	64	64	66	68	70
#rules of XShot	4.00	4.67	8.00	4.08	4.56	3.83	4.37	5.40	4.40	4.13	4.60	5.65	4.76	4.95	4.30	6.36	7.03	4.63	5.06	4.67	4.16
#rules of Ring	4.72	3.48	3.57	5.50	5.76	5.67	5.42	4.56	5.00	5.80	5.35	4.04	6.53	5.45	4.96	3.61	4.17	5.63	4.34	5.08	5.81
#rules of LLDP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

	Globa… (9,36)	Reuna (37,36)	Slovakia (35,37)	GEANT (27,38)	Myren (37,39)	Canerie (32,41)	Carnet (44,43)	Janet… (29,45)	SANET (43,45)	ARNES (34,46)	Lamb… (42,46)	Valley… (39,51)	RoE… (48,52)	CUDI (51,52)	ATT… (25,56)	Renater (43,56)	IIJ (37,65)	China… (42,66)	SURF… (50,68)	North… (36,76)	UUNET (49,84)
#pkts of XShot	6	7	6	6	6	7	7	7	7	6	25	8	9	7	8	9	9	14	20	15	19
#pkts of Ring	6.5	7.5	7.5	6.5	7.5	6.5	7.5	6.5	7.5	6.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5
#pkts of LLDP	72	72	74	76	78	82	86	90	90	92	92	102	104	104	112	112	130	132	136	152	168
#rules of XShot	10.44	4.92	4.34	4.67	4.62	5.16	4.50	6.21	5.93	6.09	6.74	5.87	5.63	5.20	11.08	6.02	6.92	7.45	4.56	10.17	6.27
#rules of Ring	12.00	5.84	5.54	5.85	5.76	5.19	5.86	4.62	4.63	4.82	4.29	5.10	4.92	5.61	7.64	4.67	6.24	6.26	4.88	8.00	6.37
#rules of LLDP	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Evaluation

Due to the fluctuations in measured latency, ADW-Donut yields less false positive results and has a better detection precision

- Results
  - Failure detection performance



Evaluation

ADW-Donut increases the precision to more than 94%, in the middle or later period of congestion, and keeps the recall more than 80%

- Results
  - Failure detection performance

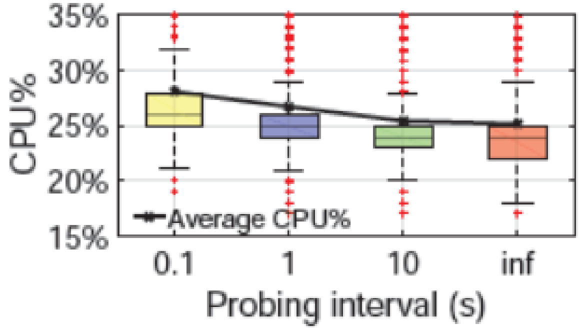
Table 3: Detection performance of Donut and ADW-Donut under different durations of congestion

	$\leq 5s$	$\leq 10s$	$\leq 20s$
Donut recall	76.87%	86.17%	87.07%
ADW-Donut recall	80.48%	87.57%	88.53%
Donut precision	75.24%	79.57%	81.56%
ADW-Donut precision	94.83%	96.28%	96.61%

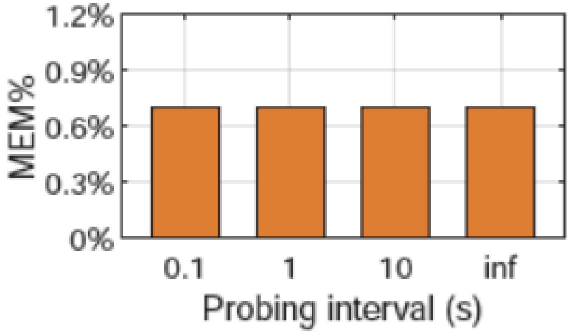
Evaluation

- Results
  - Overhead

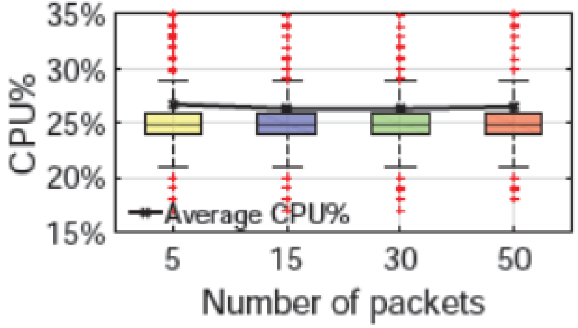
XShot increases the average CPU usage by less than 3%, compared with the XShot-not-working situation (*interval* = *inf*)



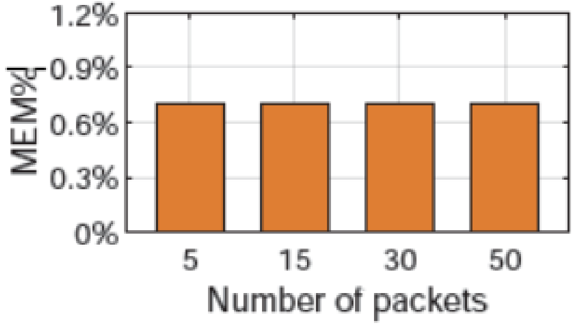
(a) CPU usage



(b) MEM usage



(a) CPU usage

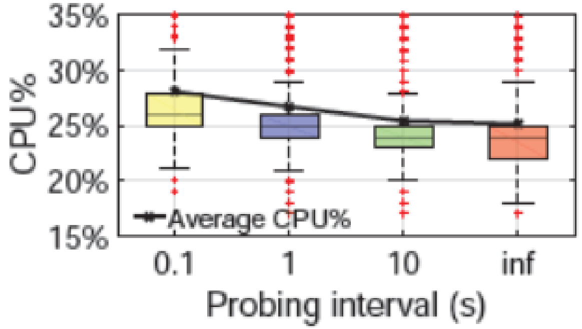


(b) MEM usage

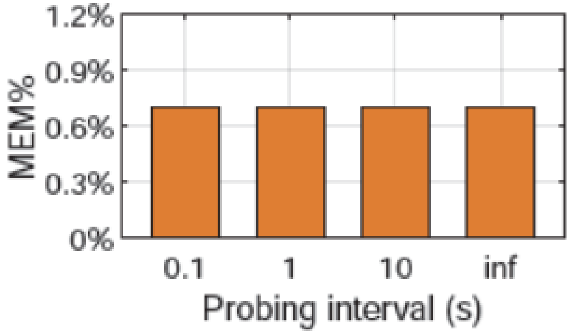
Evaluation

- Results
  - Overhead

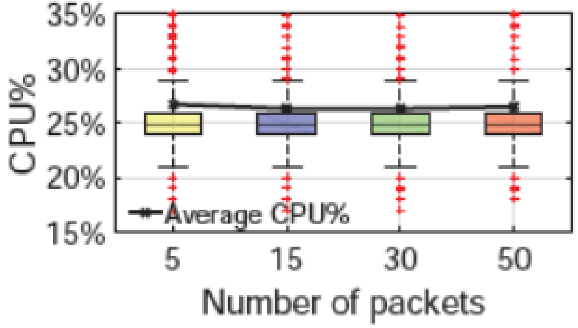
In case of changing the number of probing packets, the CPU usage has barely changes



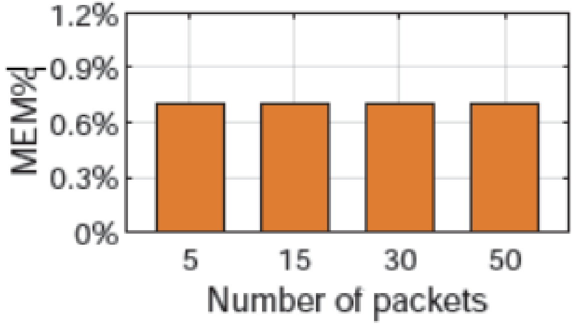
(a) CPU usage



(b) MEM usage



(a) CPU usage

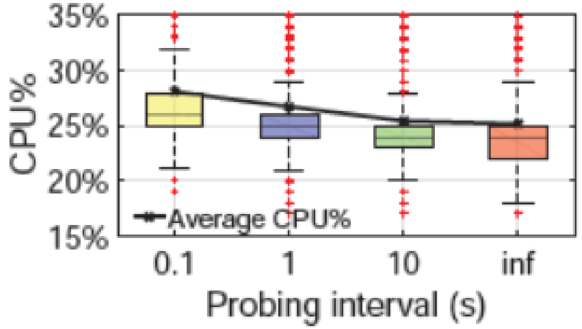


(b) MEM usage

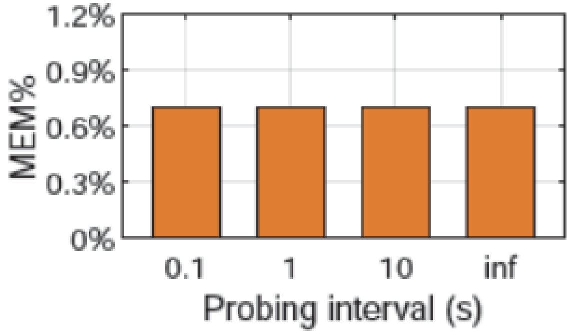
Evaluation

- Results
  - Overhead

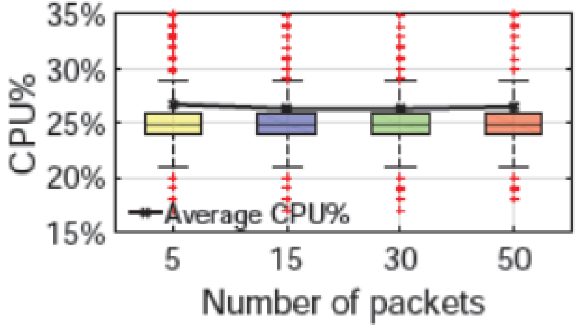
The controller consumes only around 0.7% memory, little of which is caused by *XShot*



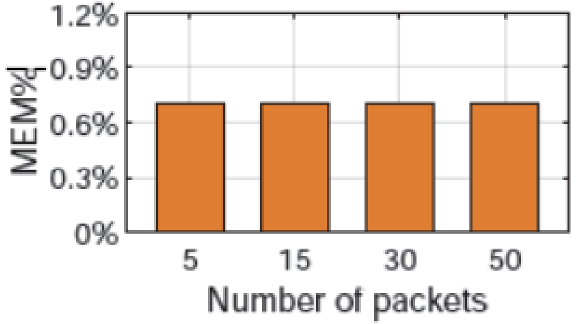
(a) CPU usage



(b) MEM usage



(a) CPU usage



(b) MEM usage

Conclusion

- *XShot* is a quick and light-weight link failure localization system in SDN
- *XShot* pinpoints the exact faulty link within just one-round shot of probing
- *XShot* reduces the number of probing packets and forwarding rules
- *XShot* identifies the partial failures, and has a detection precision of more than 94%