



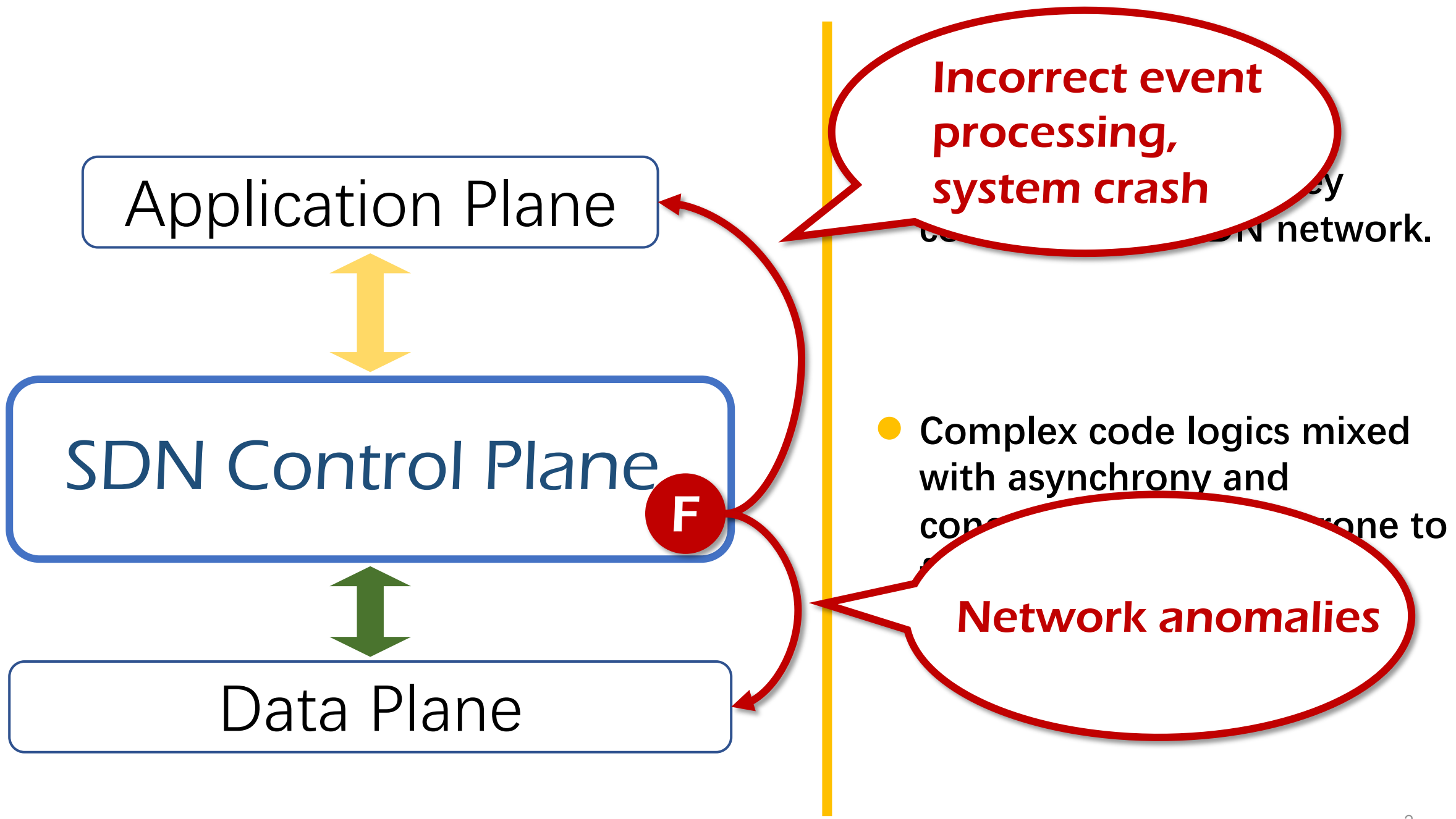
IFIP/IEEE International Symposium on  
Integrated Network Management



# Thinking inside the Box: Differential Fault Localization for SDN Control Plane

**Xing Li**<sup>1</sup>, Yinbo Yu<sup>2</sup>, Kai Bu<sup>1</sup>, Yan Chen<sup>1, 3</sup>, Jianfeng Yang<sup>2</sup>, Ruijie Quan<sup>1</sup>

<sup>1</sup>Zhejiang University, <sup>2</sup>Wuhan University, <sup>3</sup>Northwestern University



Fault Diagnosis in

SDN Control Plane

# Fault Diagnosis in SDN Control Plane

## Controller Troubleshooting (CT):

STS [SIGCOMM'14], JURY [DSN'16], CONGUARD [USENIX'17]

- **Cannot** exhaust all faults in the test environment
- **Cannot** point out the root causes of the faults

## Program Analysis (PA):

NICE [NSDI '12], Nelson et al. [FM' 15], DiffProv [SIGCOMM'16]

- **Only** focus on several specific issues
- **Cannot** adapt to frequently changing SDN controller software

# Fault Diagnosis in SDN Control Plane

What is the **ideal** SDN control plane fault diagnosis mechanism?

- **Can** cover most kinds of faults in SDN control plane
- **Can** provide root causes of the faults
- **Can** adapt to dynamic changes of SDN software
- **Only** introduce low runtime overhead

# Faults in

# SDN Control Plane

All 298 confirmed bugs of ODL from 2014 to October 16, 2017.

Bug Category	Count
Logic/design Flaw	15
Coding Mistake	15
Performance Anomaly	15

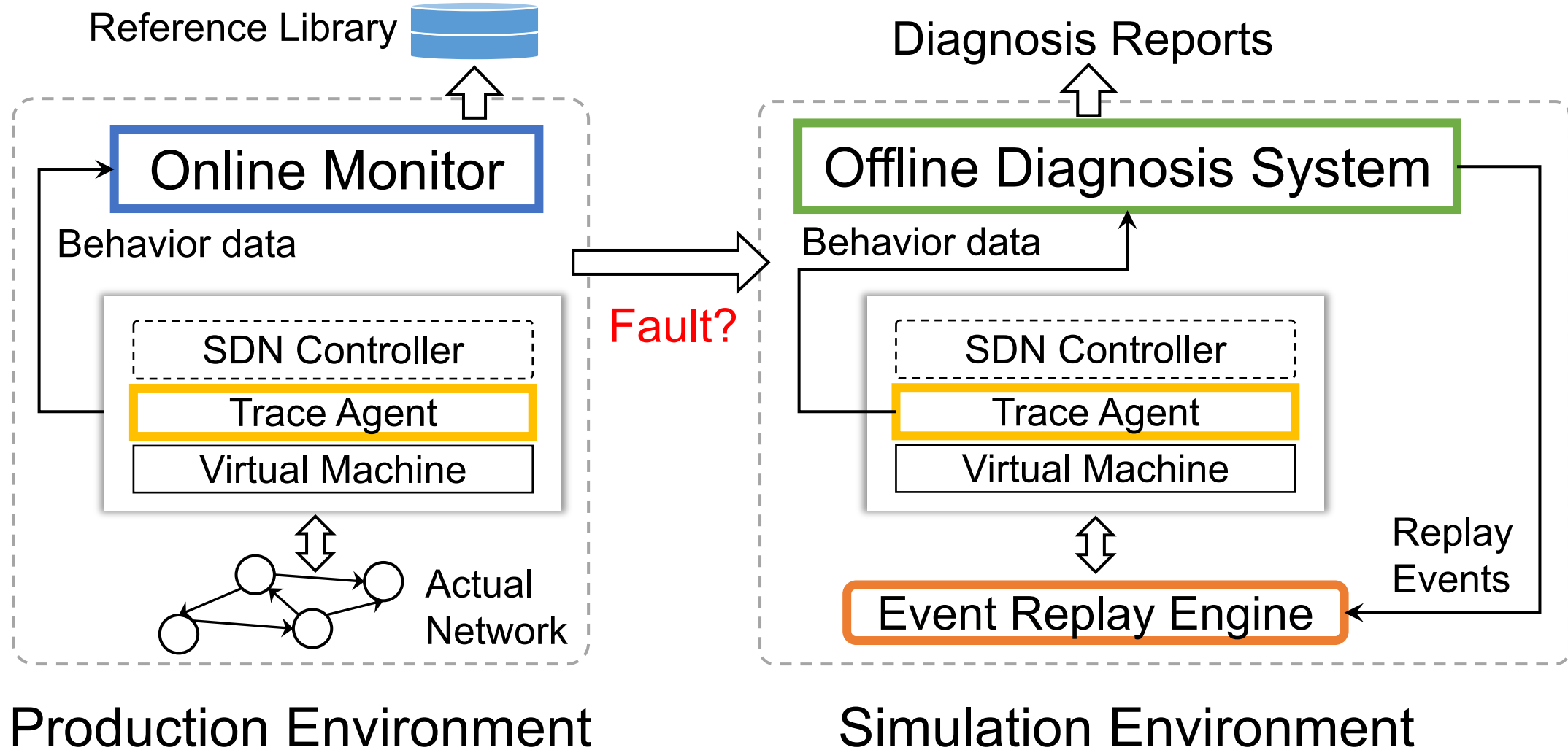
## Core idea

- Modeling system behaviors by their internal execution paths
- Inferring the fault cause from the differences between the normal system behavior and the faulty one

**Key property:**

**Incorrect internal executions**

# FALCON: Overview



# FALCON: Overview

Reference Library

High level steps:

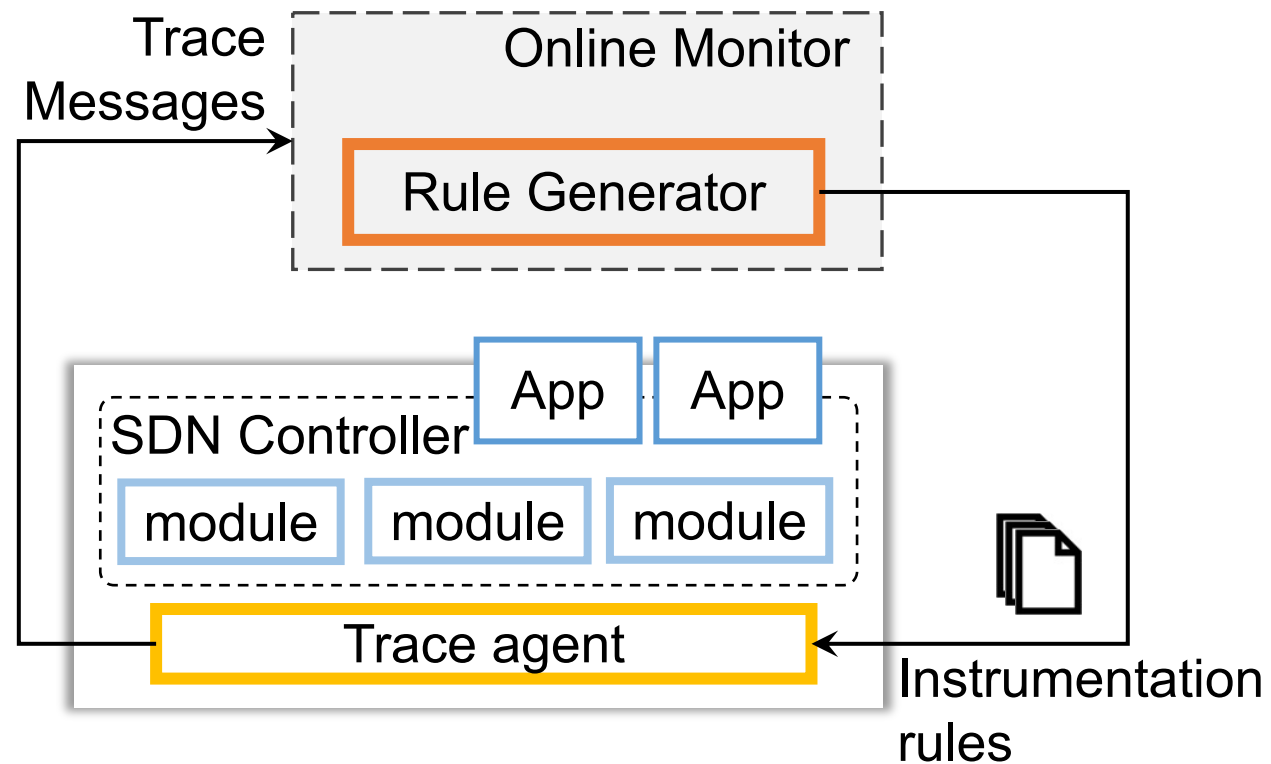
1. Track system behaviors
2. Model system behaviors
3. Perform fault localization

Production Environment

Simulation Environment

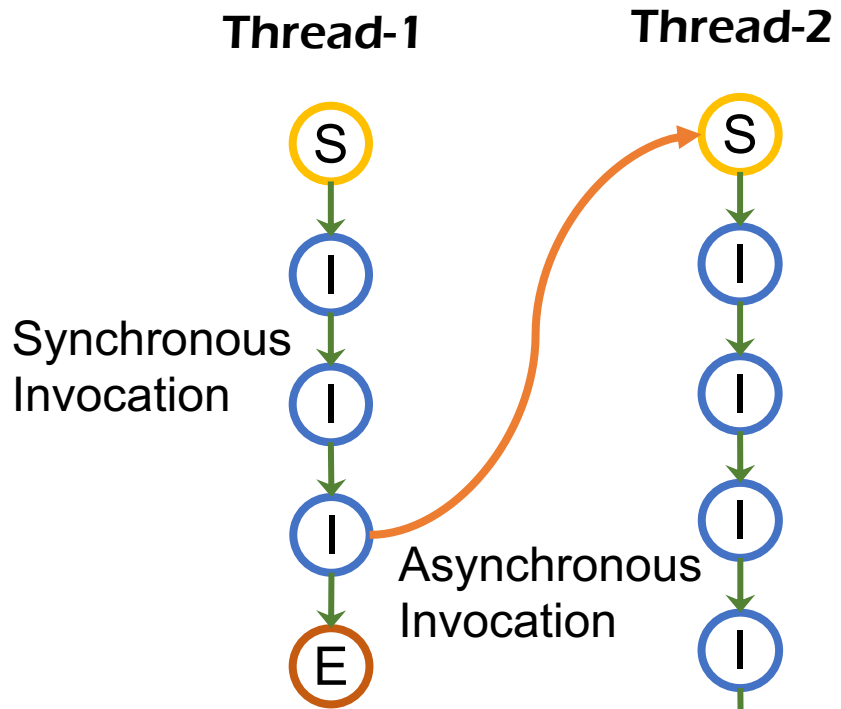


# FALCON: System Behavior Tracing



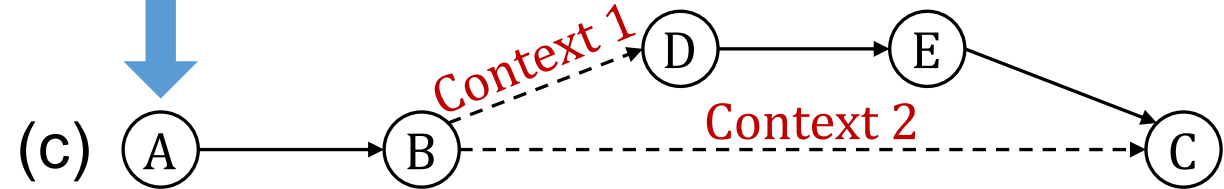
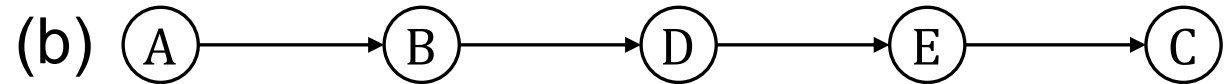
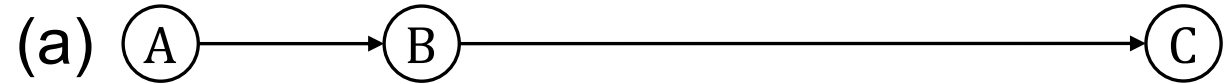
- Rule-based instrumentation mechanism
- Based on bytecode instrumentation
- Trace at module-level to reduce the overhead

# FALCON: System Behavior Modeling

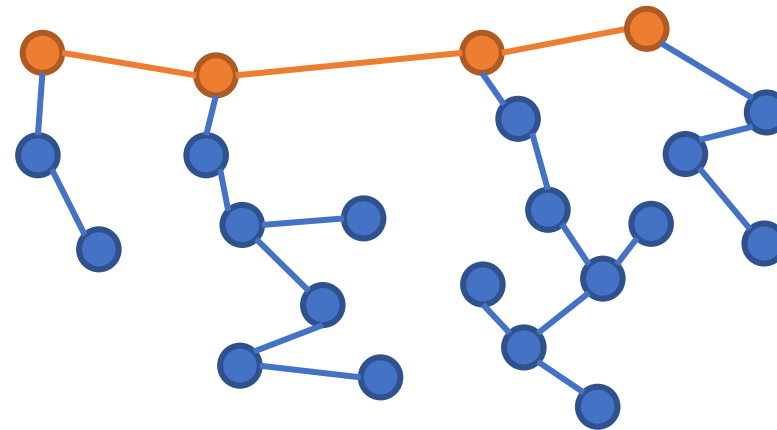


- S** : Starting Node
- I** : Intermediate Node
- E** : Ending Node

Trace graph



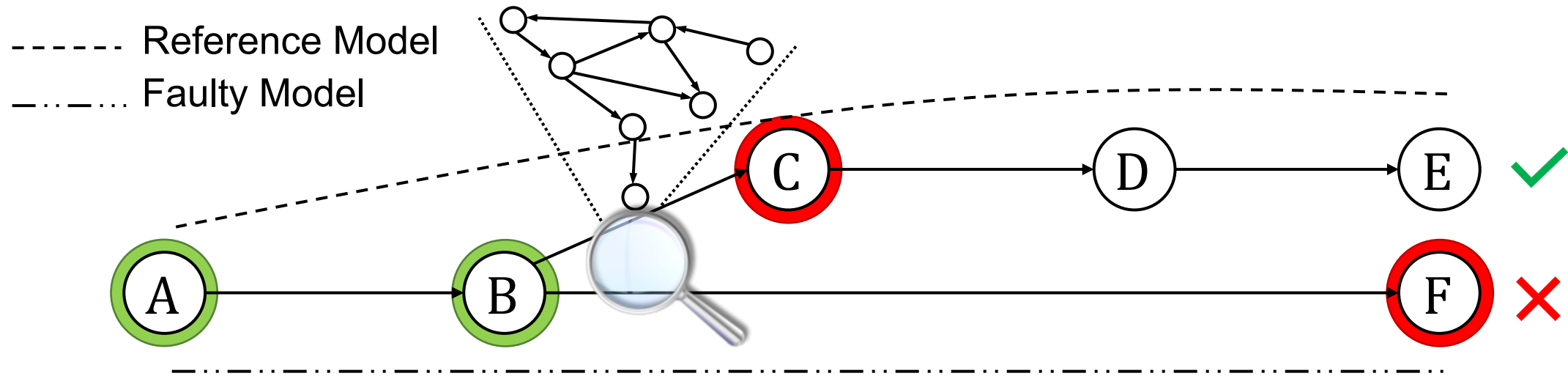
Context-aware model (CAM)



Augmented CAM

# FALCON: System Behavior Fault Localization

1. Locate the faulty models and their references
2. Conduct differential checking
3. Perform static analysis  $\longrightarrow$  Key contexts



# FALCON: Implementation

~ 10,000 LOC in Java for OpenDaylight Controller

- Trace Agent
  - ASM
  - LMAX Disruptor
  - Chronicle Queue
- Event Replay Engine
  - Extended STS simulator

# FALCON: Evaluation - Effectiveness

## Case Studies

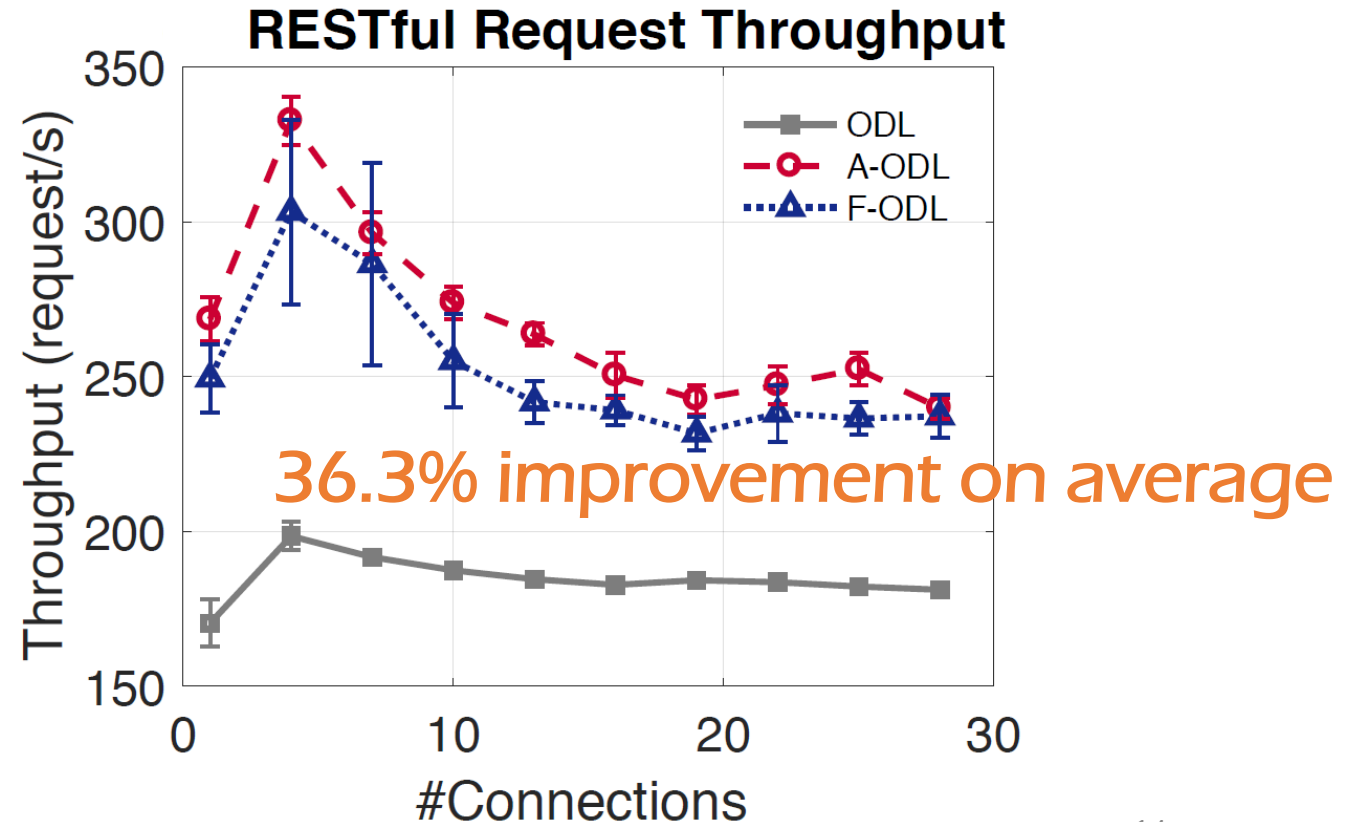
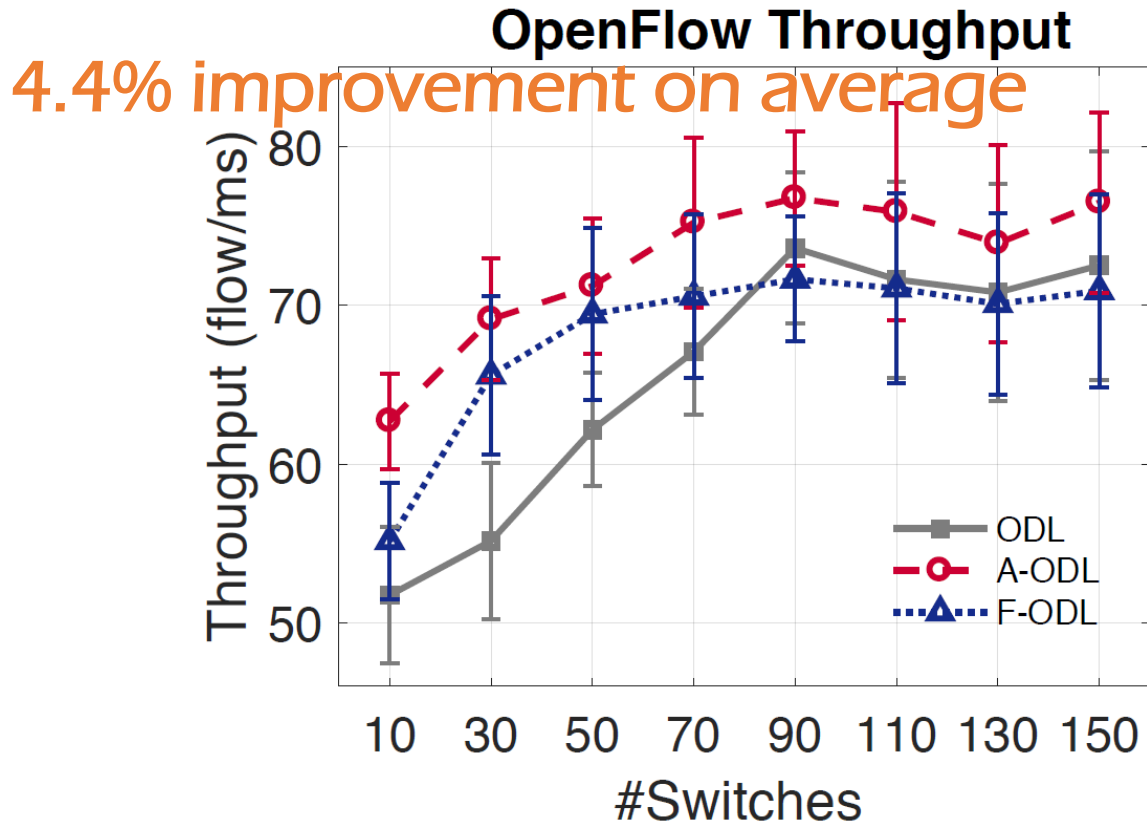
### 8 real-world faults from ODL Bugzilla

Bug ID	Symptom	Root cause	Category	Diagnose
5033	Unexpected response	race condition	logic flaw	Yes
5816	Unexpected response	constant misconfiguration	logic flaw	Yes
8157	Error in log message	defective user deletion	logic flaw	Yes
3345	Unreachability	incomplete topology update	design flaw	Indirectly
6053	NPE in log message	incomplete JSON parsing	design flaw	Yes
7933	NPE in log message	incomplete YANG support	design flaw	Yes
8939	Error in log message	interface migration	coding mistake	Indirectly
8988	NPE in log message	method misuse	coding mistake	Indirectly

# FALCON: Evaluation - Performance

## Throughput

- A-ODL: OpenDaylight with ASM
- F-ODL: OpenDaylight with FALCON



# FALCON: Conclusion

- **FALCON: the first FAult Localization tool for SDN CONtrol plane**
  - A rule-based dynamic tracing mechanism
  - A context-aware modeling mechanism
  - A differential fault localization mechanism
- **Strong fault diagnosis capability with low overhead**

# Thanks

[xing\\_li@zju.edu.cn](mailto:xing_li@zju.edu.cn)



Back up pages

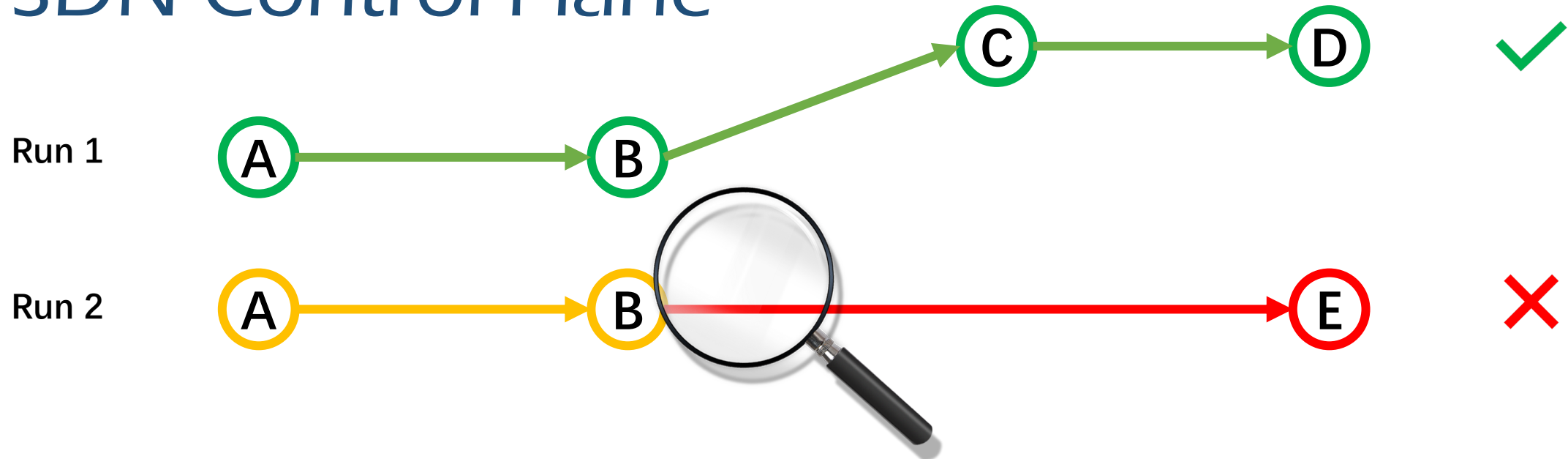
# Fault Diagnosis in SDN Control Plane

What is the **ideal** SDN control plane fault diagnosis mechanism?

- **Challenge 1:** High false fault rate in SDN control plane
- **Challenge 2:** Accuracy of fault localization
- **Challenge 3:** Flexibility changes of SDN software
- **Challenge 4:** Lightweight overhead

## Record and model internal system behaviors

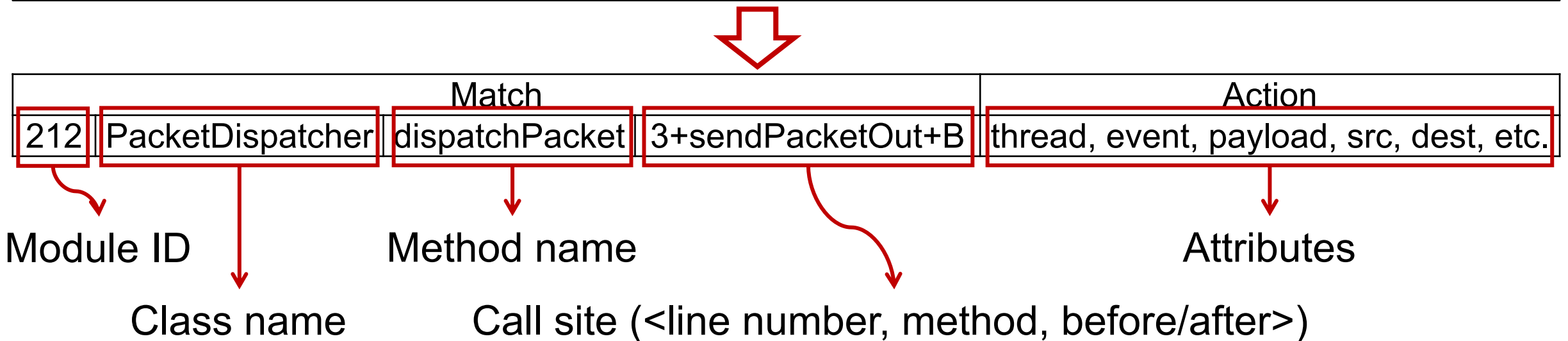

# Thinking inside the Box: Differential Fault Localization for SDN Control Plane



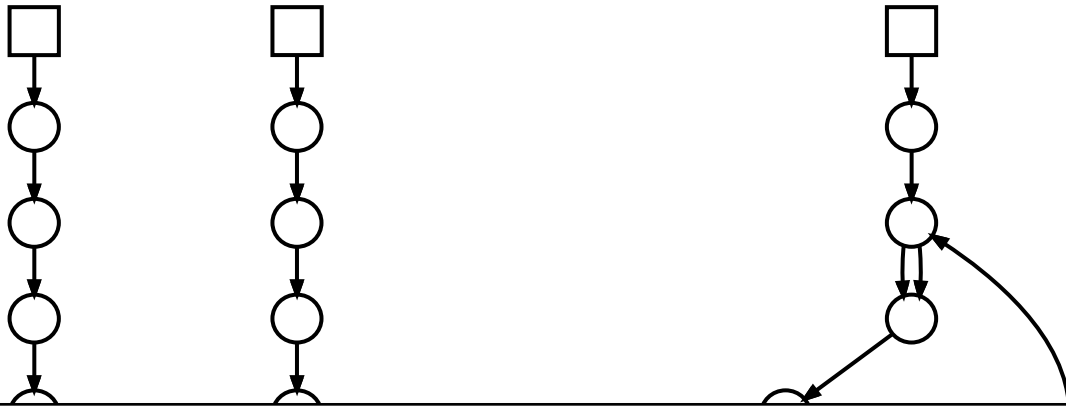
# FALCON: *dynamic system behavior tracing*

```
public class PacketDispatcher {
    public void dispatchPacket(payload, ingress, srcMac, destMac) {
0: NodeConnectorRef src = inventoryReader.getControllerSwitchConnectors(srcMac);
1: NodeConnectorRef dest = inventoryReader.getNodeConnector(destMac);
2: if (src != null)
3:     if (dest != null) sendPacketOut(payload, src, dest);
4:     else floodPacket(nodeId, payload, ingress, src);
    } }

```



# FALCON: context-aware system behavior modeling



3 trace graphs from the same input event

```
NodeConnectorRef src = inventoryReader.getControllerSwitchConnectors(...);
NodeConnectorRef dest = inventoryReader.getNodeConnector(...);
if (srcConnectorRef != null) {
    if (destNodeConnector != null) {
        sendPacketOut(payload, srcConnectorRef, destNodeConnector);
    } else {
        floodPacket(nodeId, payload, ingress, srcConnectorRef);
    }
} else {.....}
```

**Conditional branches**

# FALCON: context-aware system behavior modeling

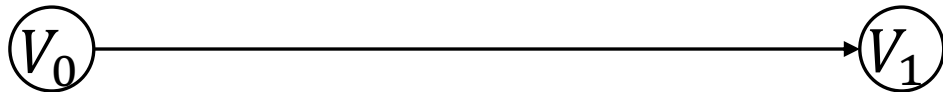
Model the Trace graph



Merge heterogeneous models

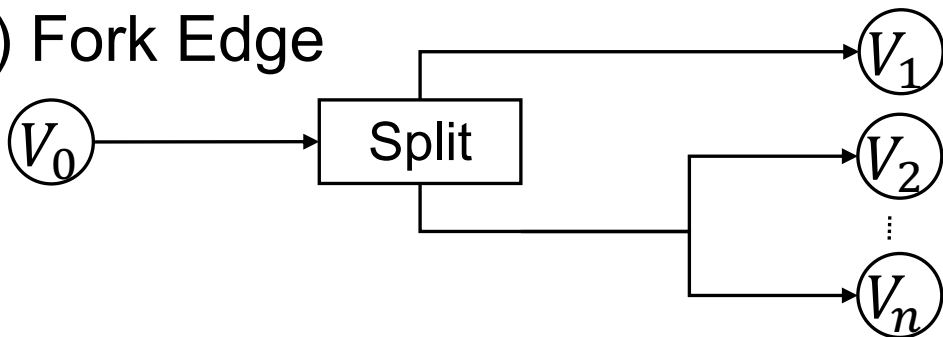
---

(a) Concrete Edge



Synchronous invocations

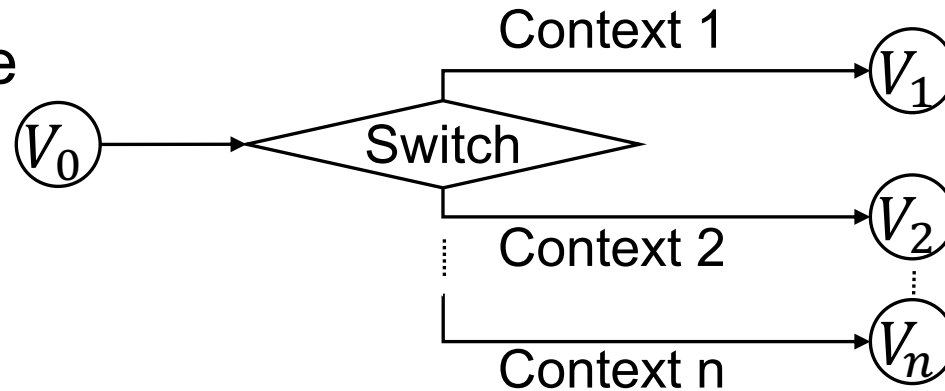
(b) Fork Edge



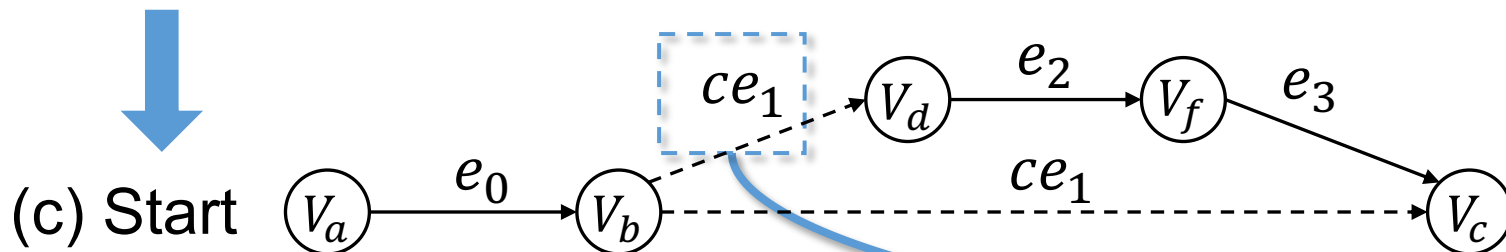
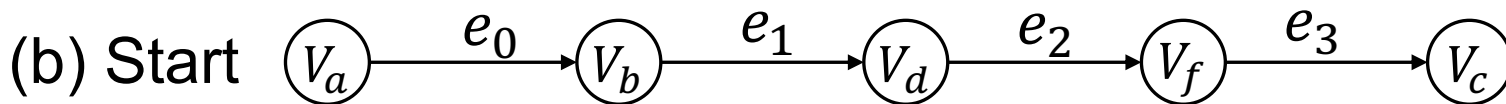
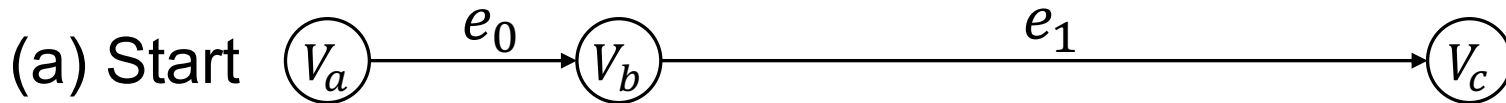
Asynchronous invocations

# FALCON: context-aware system behavior modeling

(c) Contextual Edge



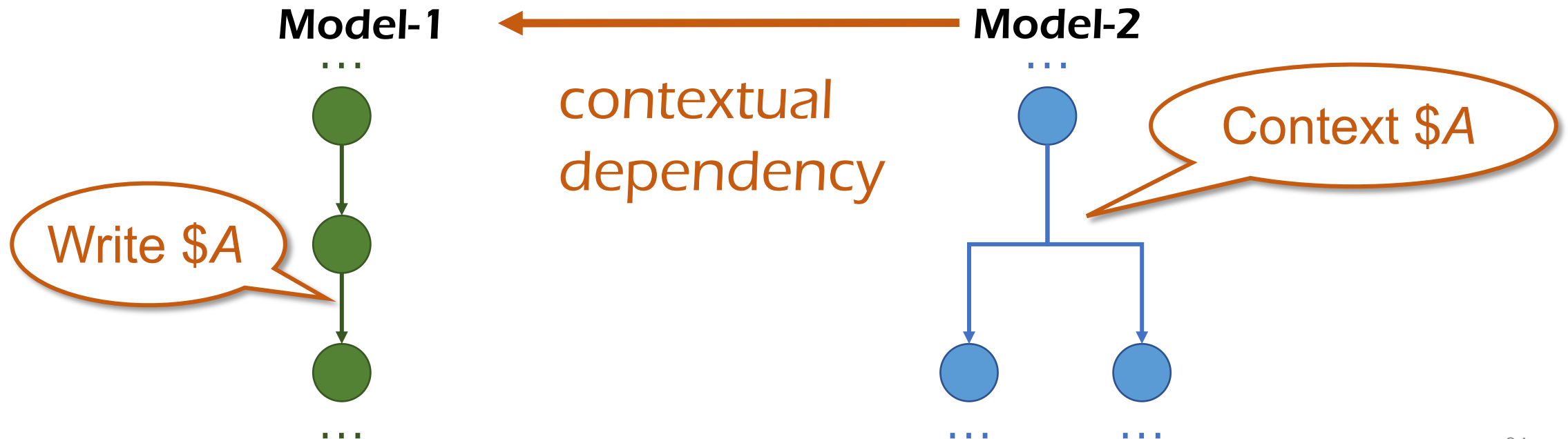
Conditional branch



**Context:**  
Mine conditions on  
corresponding control  
flow graphs

# FALCON: context-aware system behavior modeling

## Model augmentation with Model Dependency





# FALCON: differential fault systemization behavior modeling

1. Input the symptom
2. Perform fault localization
3. Get the result report

# FALCON: differential fault localization

## 1. Input the symptom

'time': ('timestamp' | null)

'type': ('REST' | 'log' | 'flow' | 'rule')

'request': ('method' & 'url' & 'payload' & 'response content' & 'response status')

'log': ('status' & 'content')

'flow': ('messageType' & 'switchID' & 'OFVersion' & 'content')

'rule': ('switchID' & 'ruleID' & 'match' & 'action')

# FALCON: differential fault localization

## 2. Perform the fault localization

2-1 Locate the faulty models and their references

**Explicit** symptom vs. **Implicit** symptom



Error log messages;  
Code exceptions

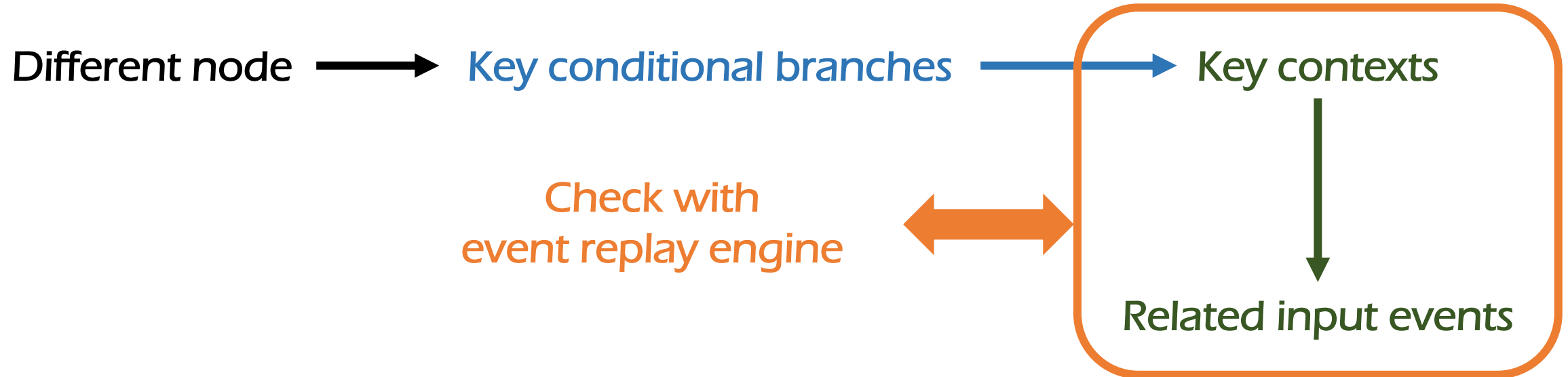


Network problems;  
Unexpected NBI responses

# FALCON: differential fault localization

## 2. Perform fault localization

### 2-3 Conduct static analysis



# FALCON: differential fault localization

## 3. Get the result report

Faulty models and references

Key contexts

 Fault diagnosis report

Locations in code

Related input events

# FALCON: Evaluation - Performance

## Latency

- A-ODL: OpenDaylight with ASM
- F-ODL: OpenDaylight with FALCON

