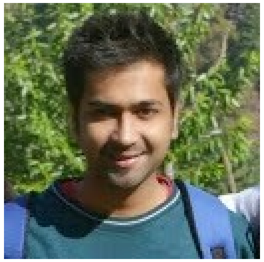
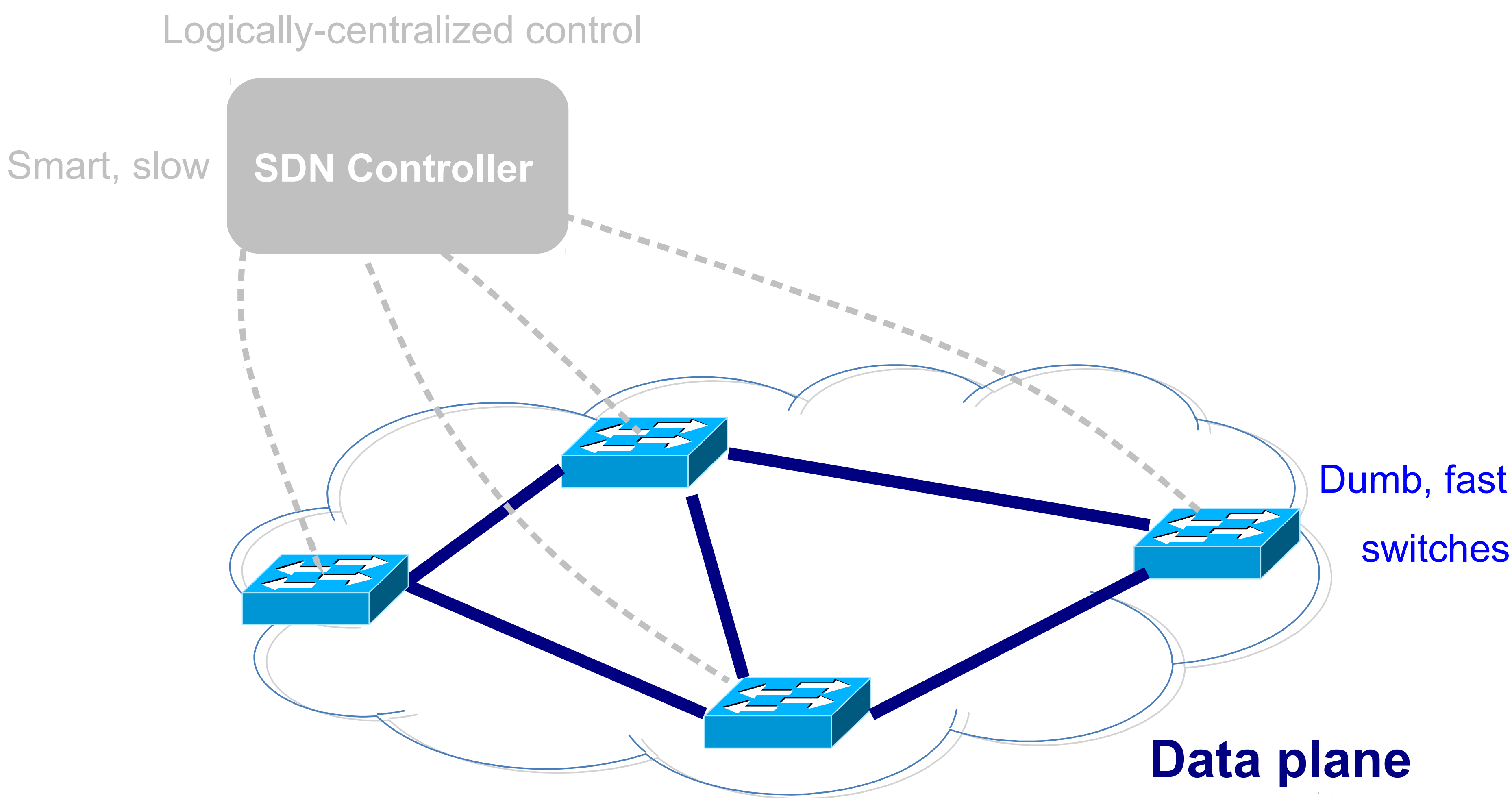
Sphinx: Detecting Security Attacks in Software-Defined Networks

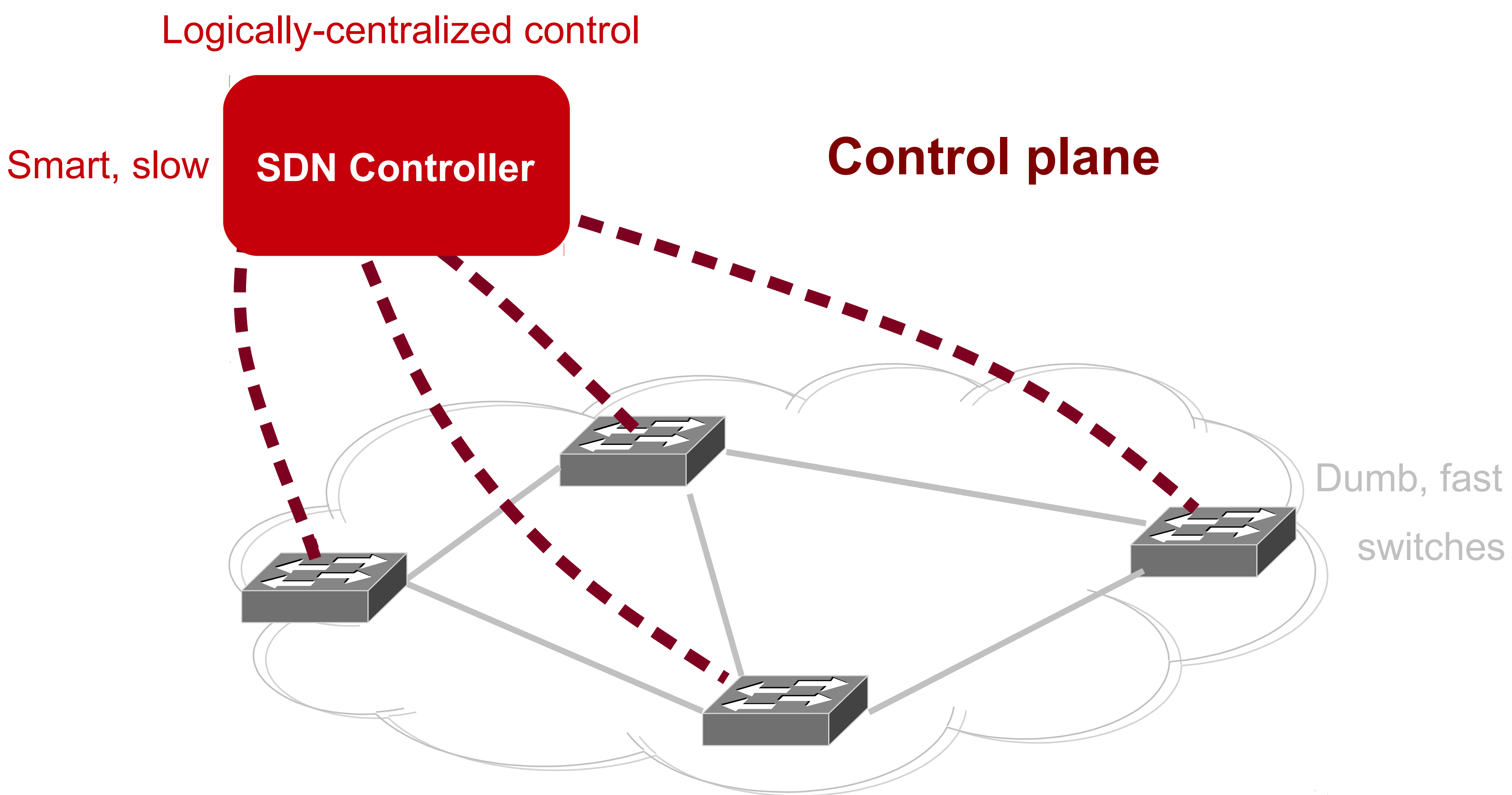
Mohan Dhawan   Rishabh Poddar   Kshiteej Mahajan   Vijay Mann

IBM Research, India

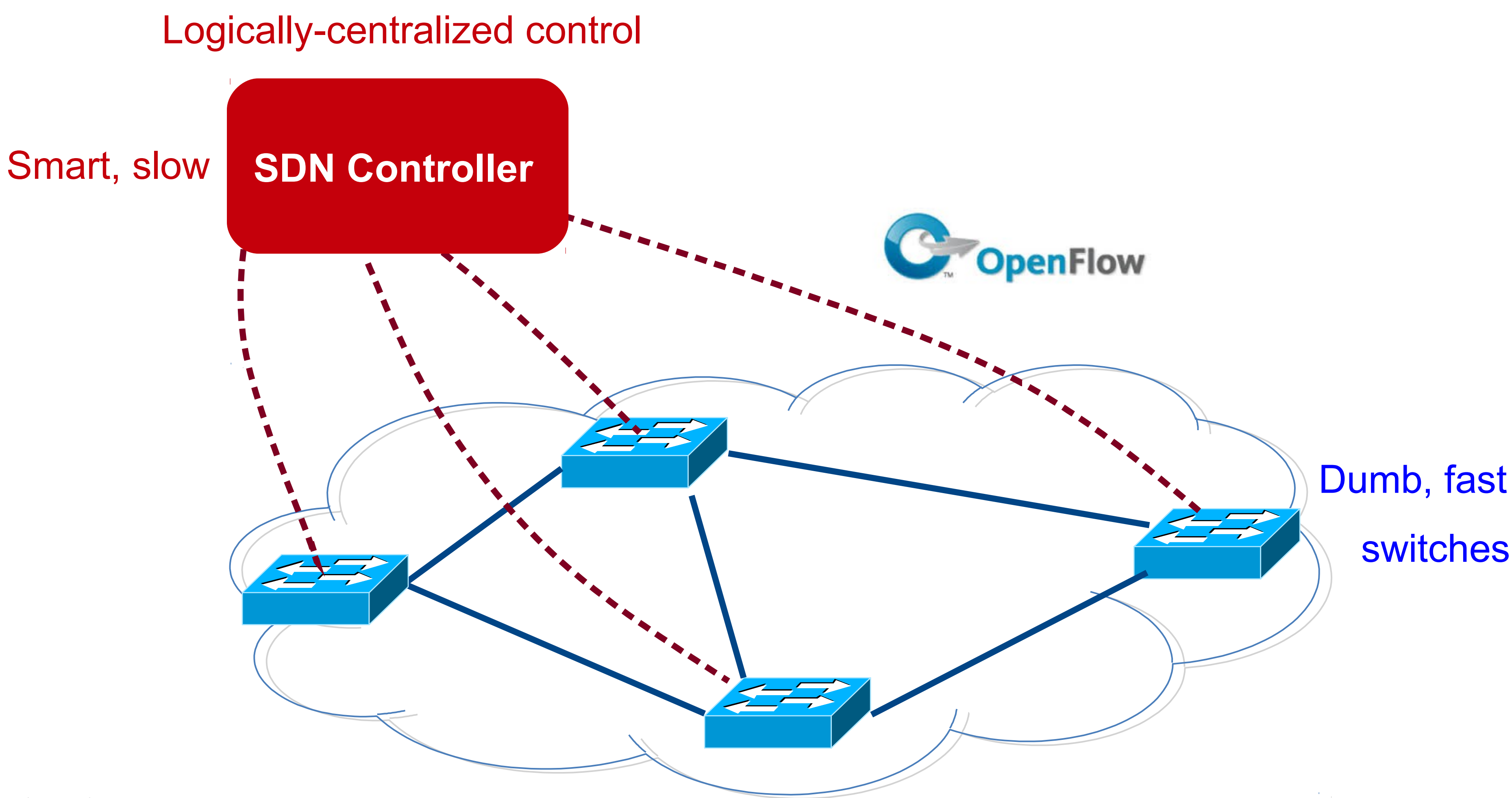
Software-Defined Network (SDN)



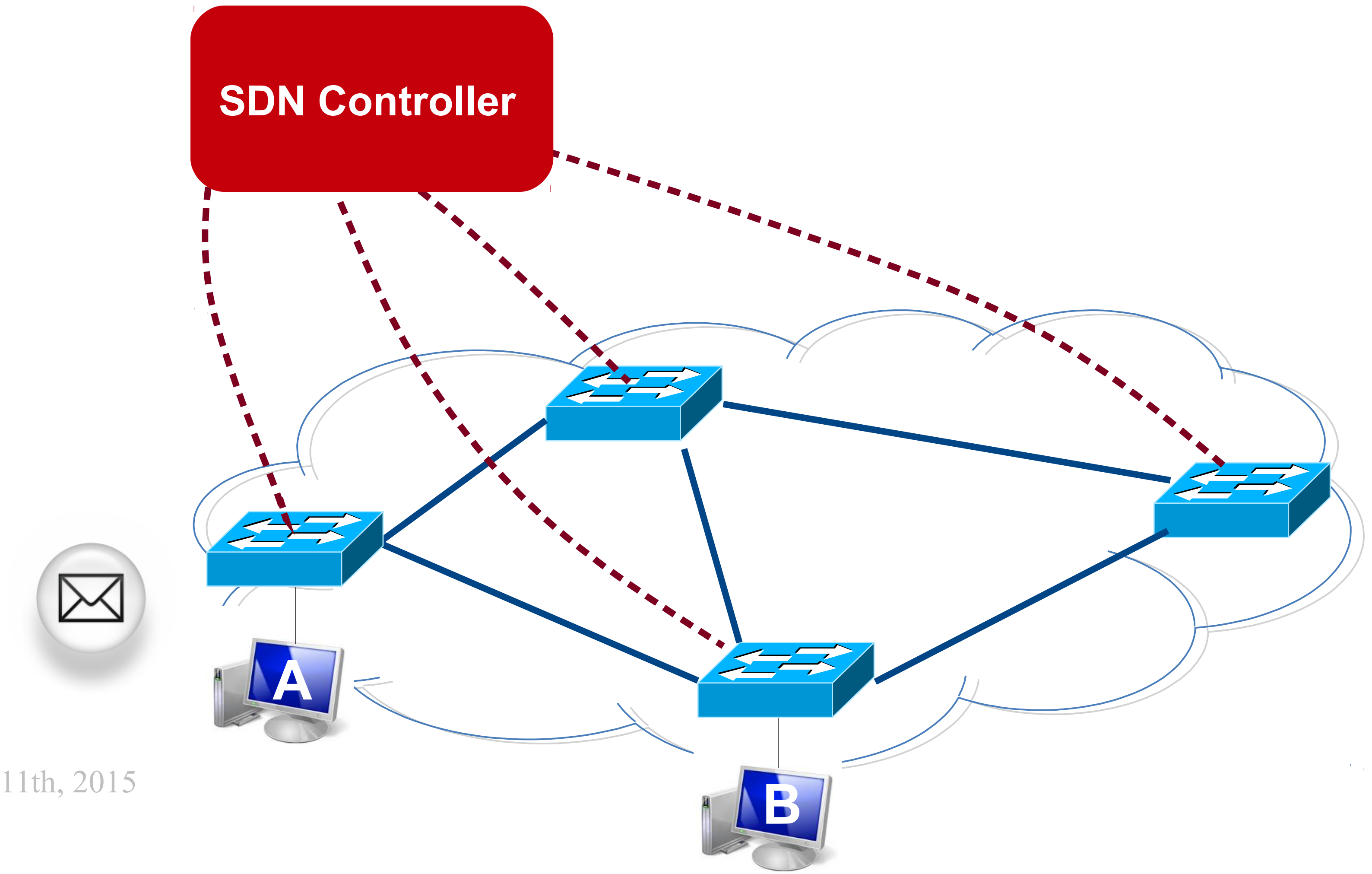
Software-Defined Network (SDN)



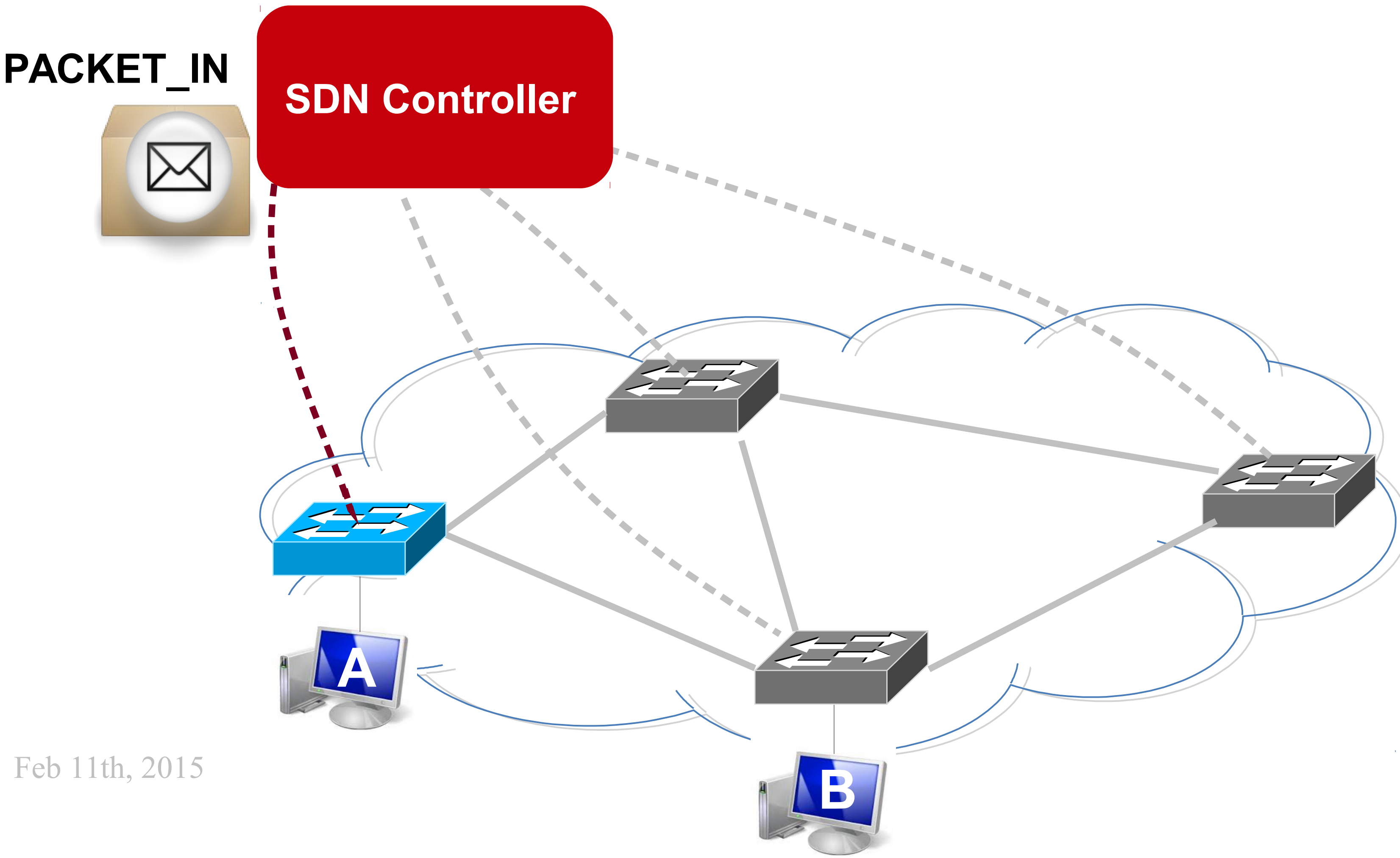
Software-Defined Network (SDN)



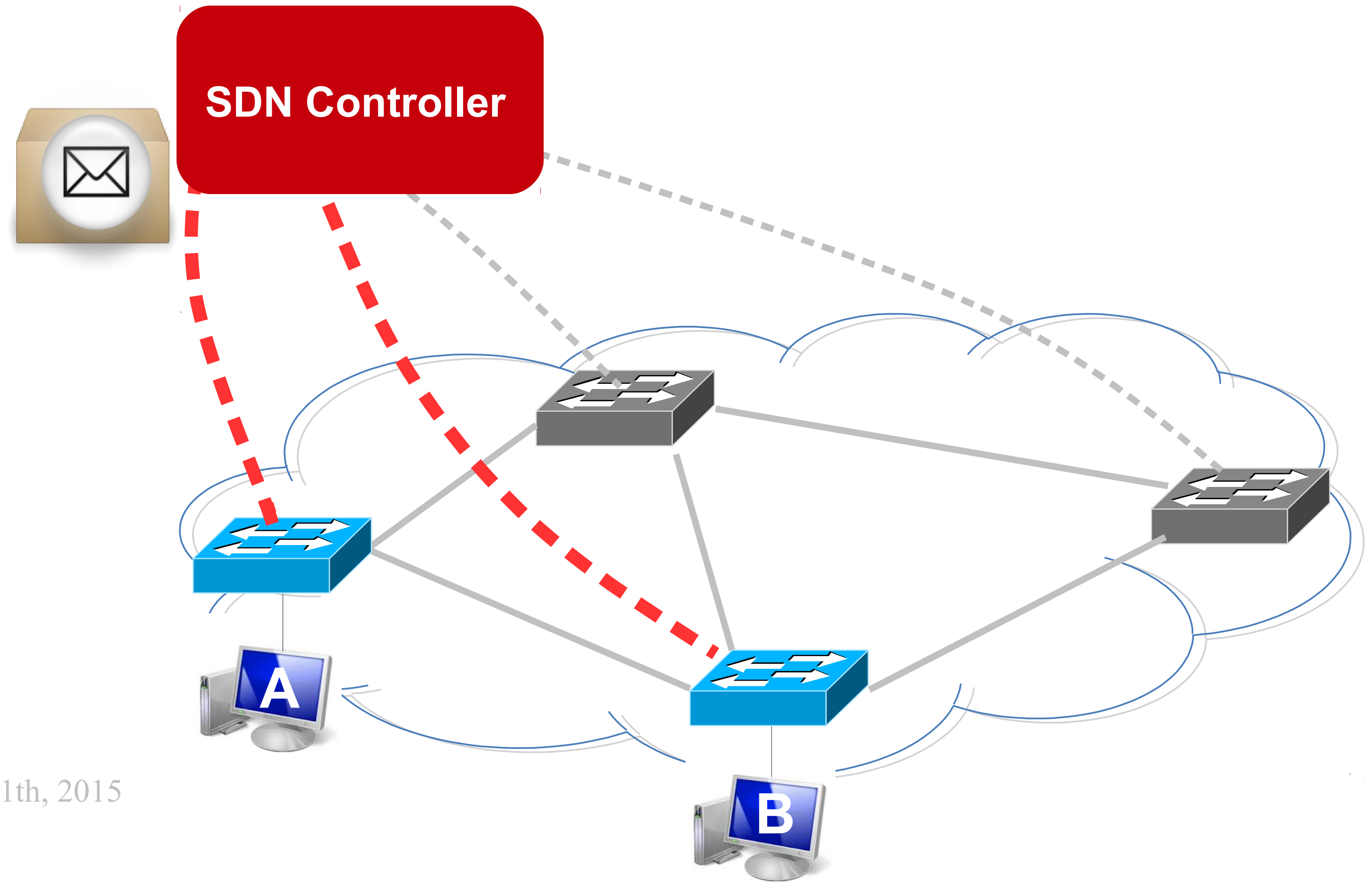
Software-Defined Network (SDN)



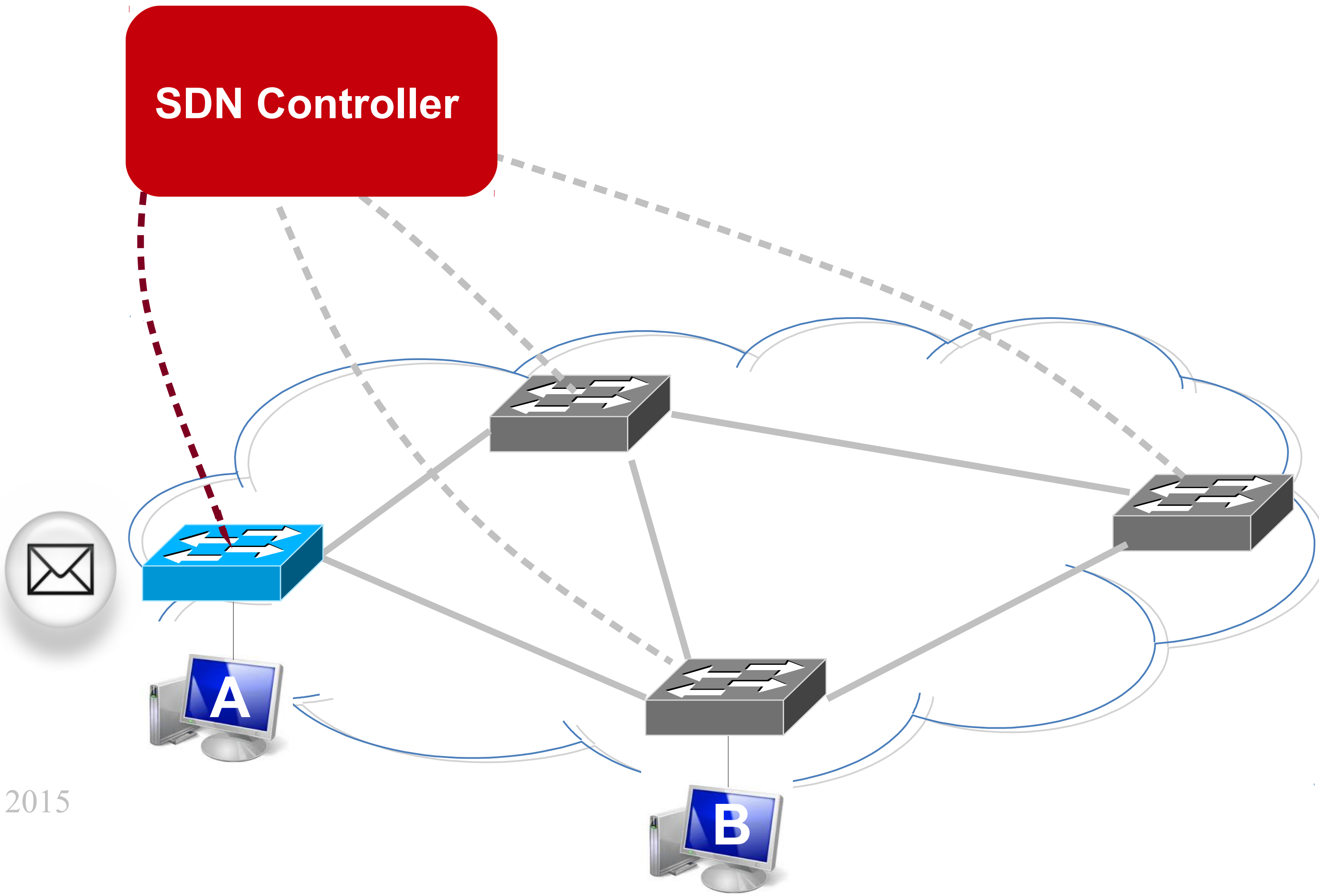
Software-Defined Network (SDN)



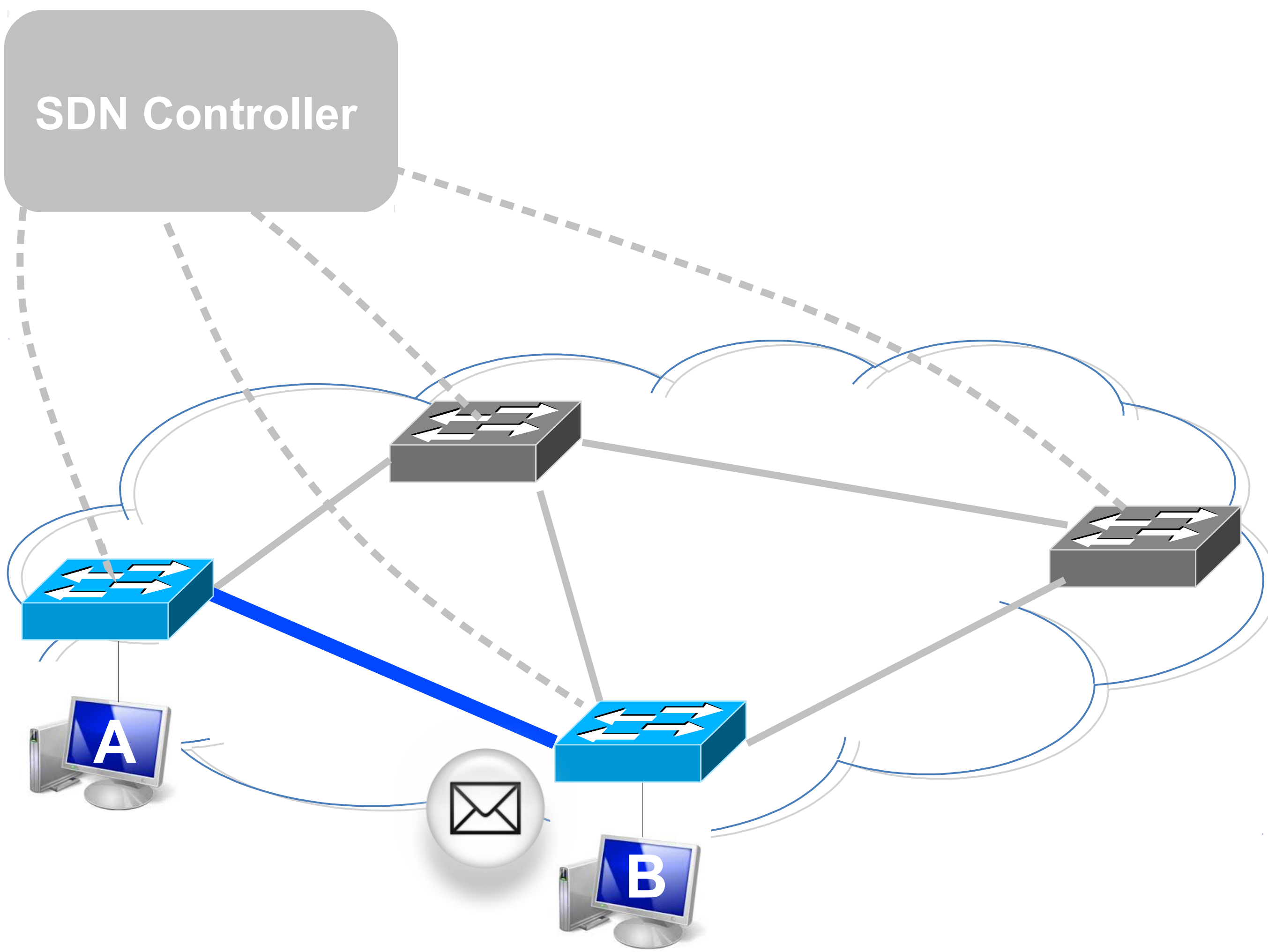
Software-Defined Network (SDN)



Software-Defined Network (SDN)



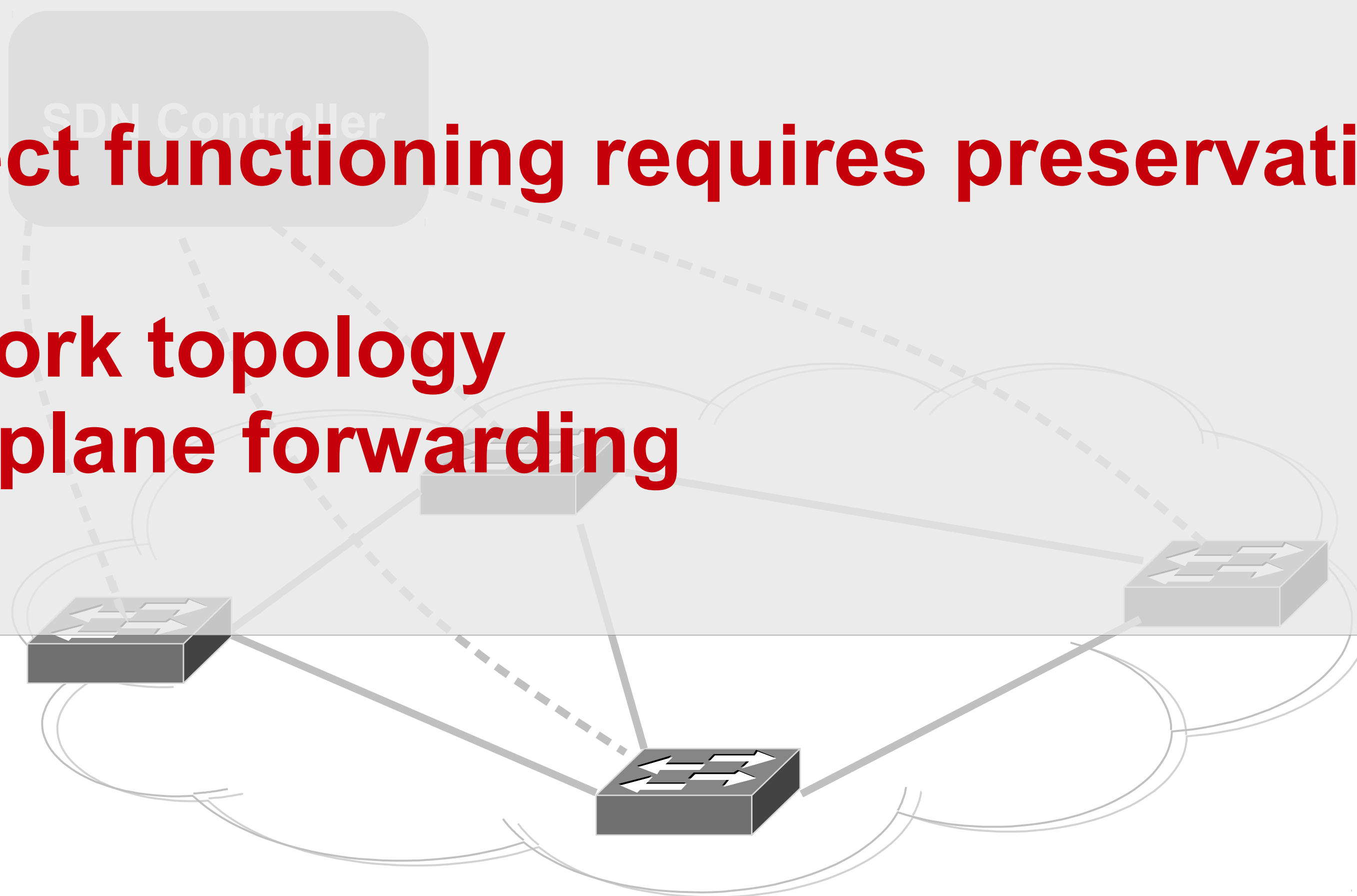
Software-Defined Network (SDN)



Software-Defined Network (SDN)

Correct functioning requires preservation of

- **Network topology**
- **Data plane forwarding**



Outline

- SDN Overview
- **Motivation**
- Sphinx
- Implementation
- Evaluation
- Conclusion

Vulnerable SDNs

- OpenFlow operational semantics
  - All unmatched packets are forwarded to the controller

Vulnerable SDNs

- OpenFlow operational semantics
  - All unmatched packets are forwarded to the controller
- Attacks afflicting traditional networks affect SDNs too
  - Traditional defenses do not work in SDNs

Vulnerable SDNs

- OpenFlow operational semantics
  - All unmatched packets are forwarded to the controller
- Attacks afflicting traditional networks affect SDNs too
  - Traditional defenses do not work in SDNs
- Attacks possible from compromised switches and end hosts
  - Soft switches on end host servers attractive targets for attackers

Several Attacks Possible

- Network topology
  - Corrupt routing table (ARP)
  - Fake topology (LLDP)
  - Multicast (IGMP)

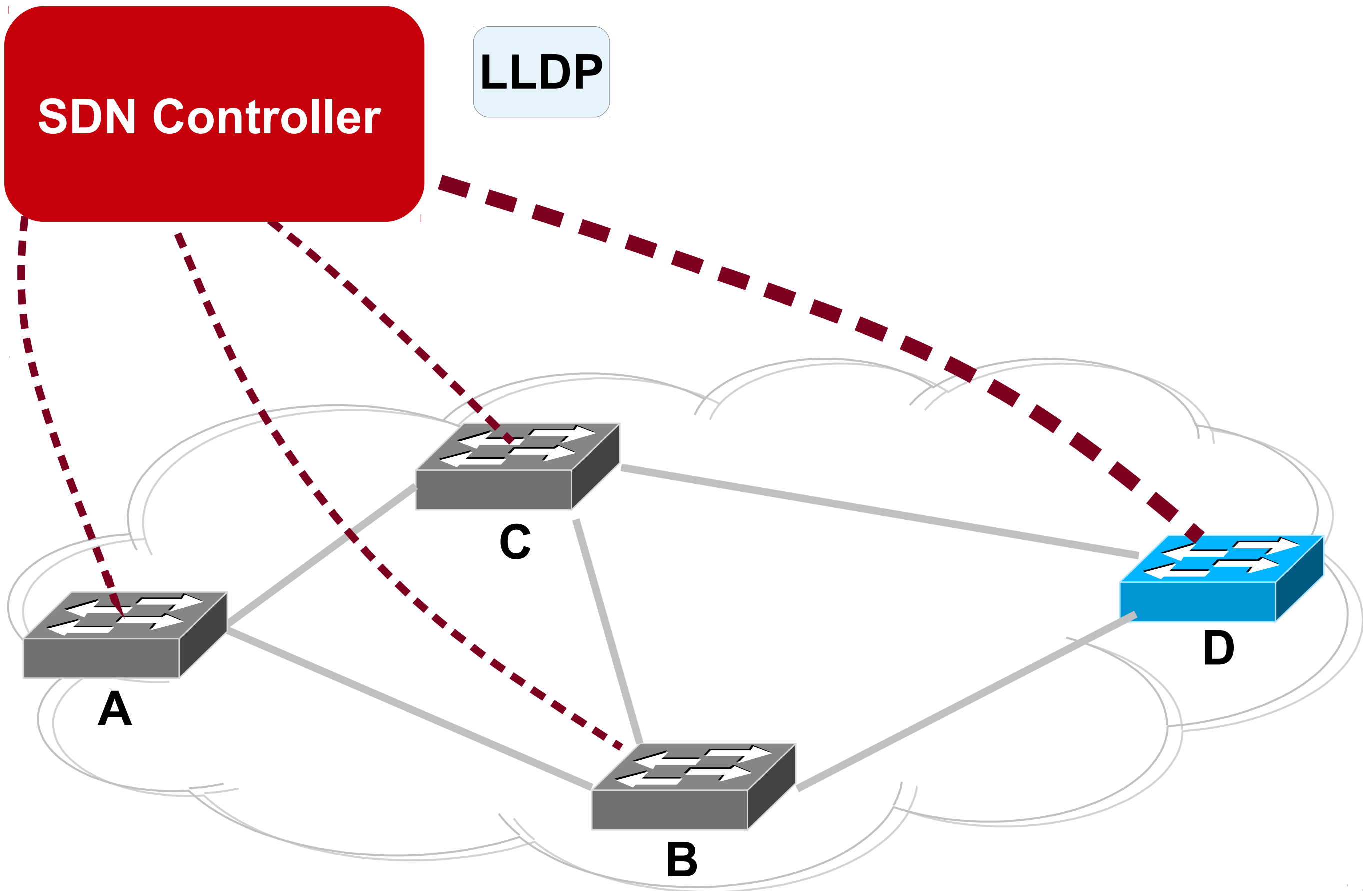
- Data plane forwarding
  - Switch TCAM exhaustion
  - Switch blackhole

Controller Vulnerability

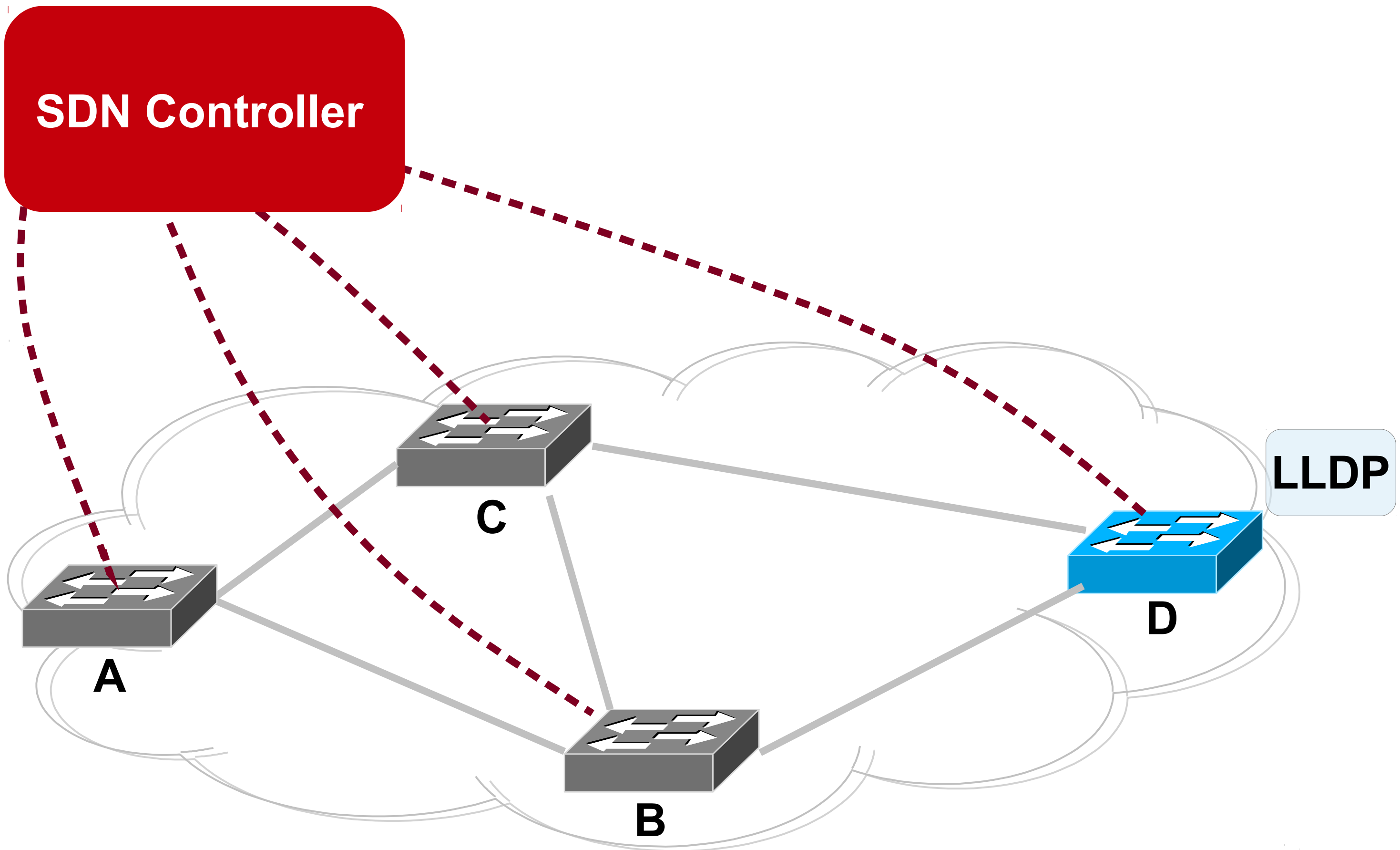
- Security analysis of four popular available SDN controllers

Attack	OpenDaylight	Floodlight	POX	Maestro
ARP poisoning	Y	Y	Y	Y
Fake topology	Y	Y	N	Y
Controller DoS	Y	N	Y	Y
Network DoS	Y	Y	Y	Y
TCAM exhaustion	N	Y	Y	Y
Switch blackhole	Y	Y	Y	Y

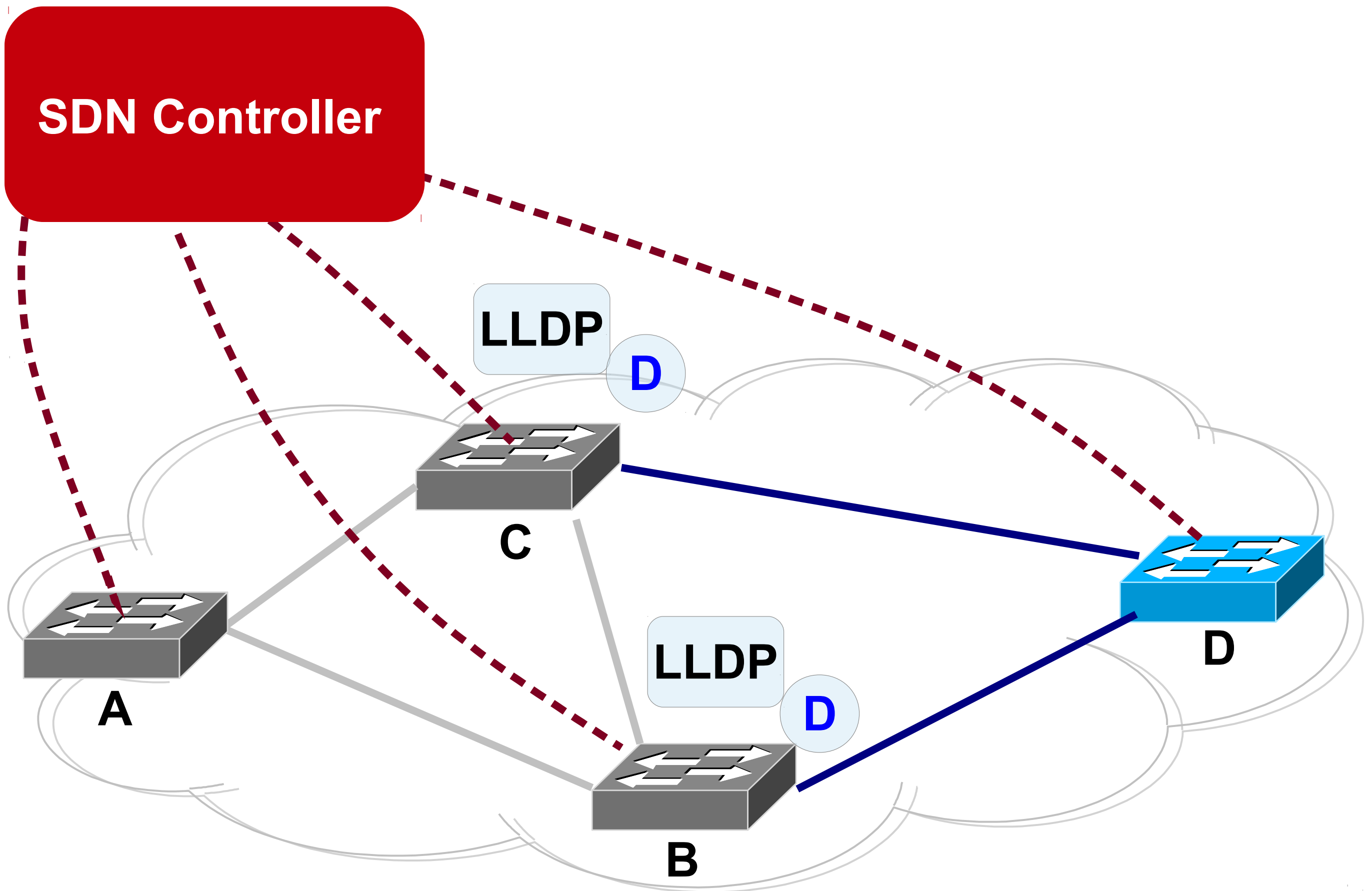
Fake Network Topology Attack



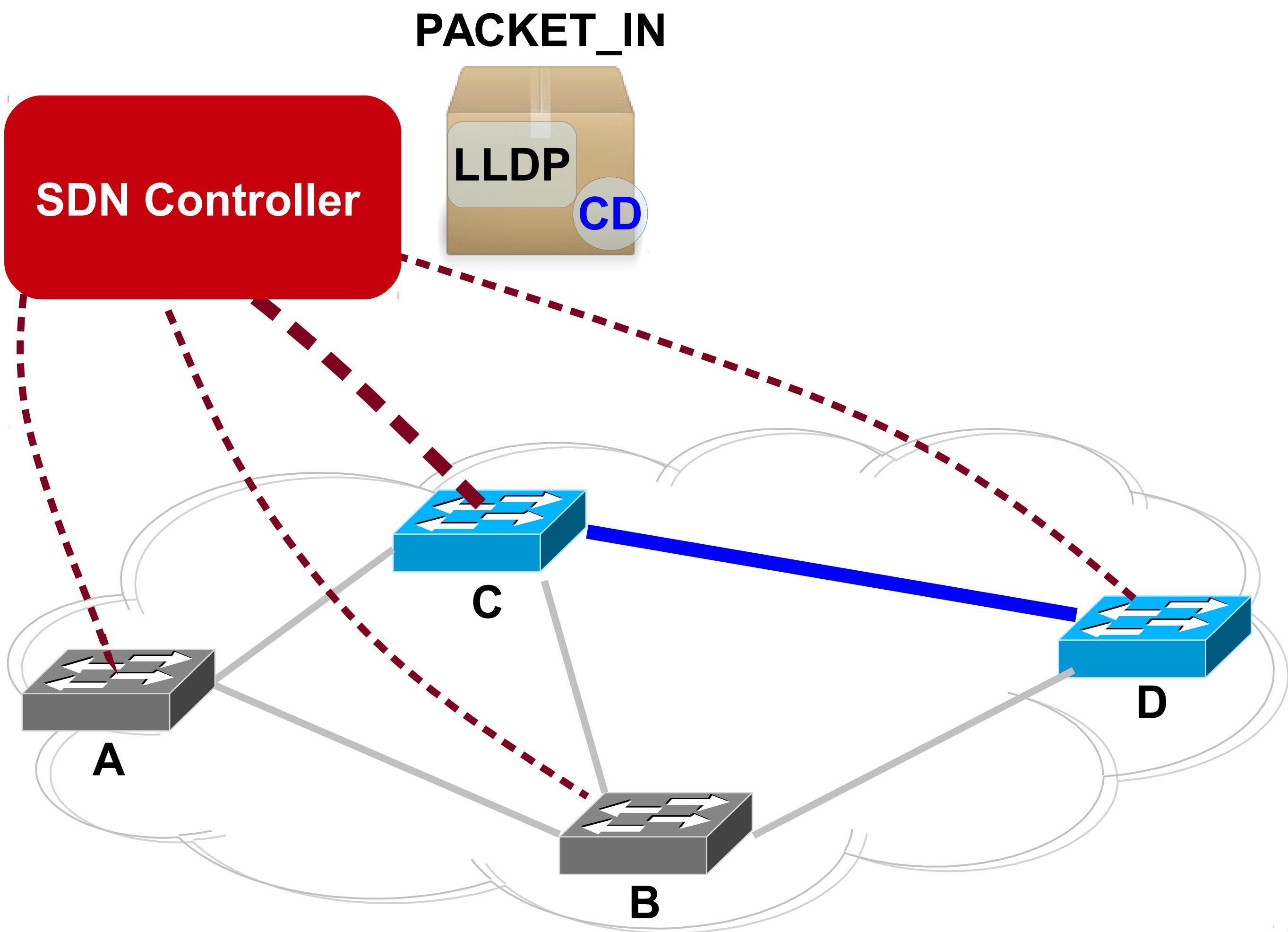
Fake Network Topology Attack



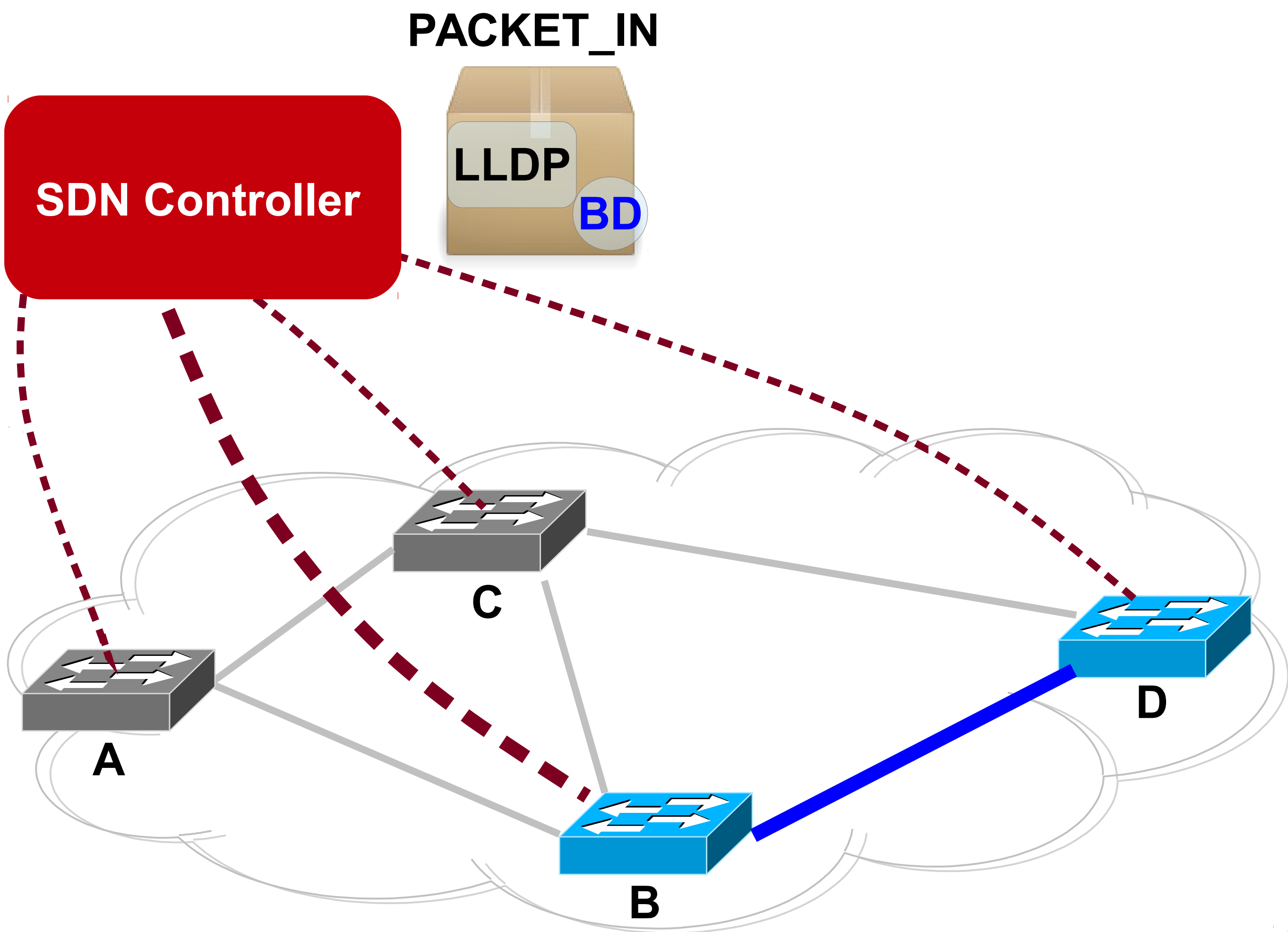
Fake Network Topology Attack



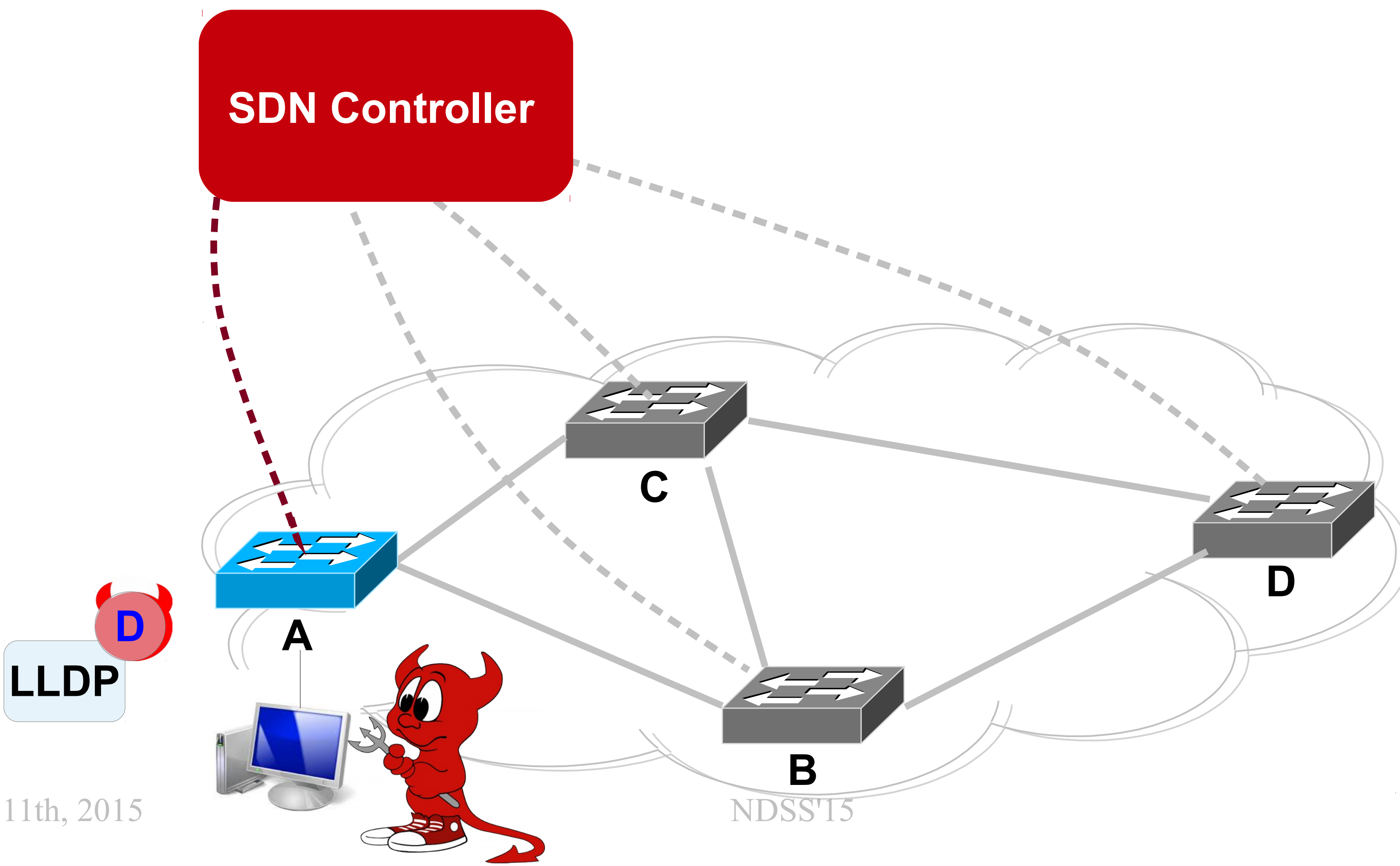
Fake Network Topology Attack



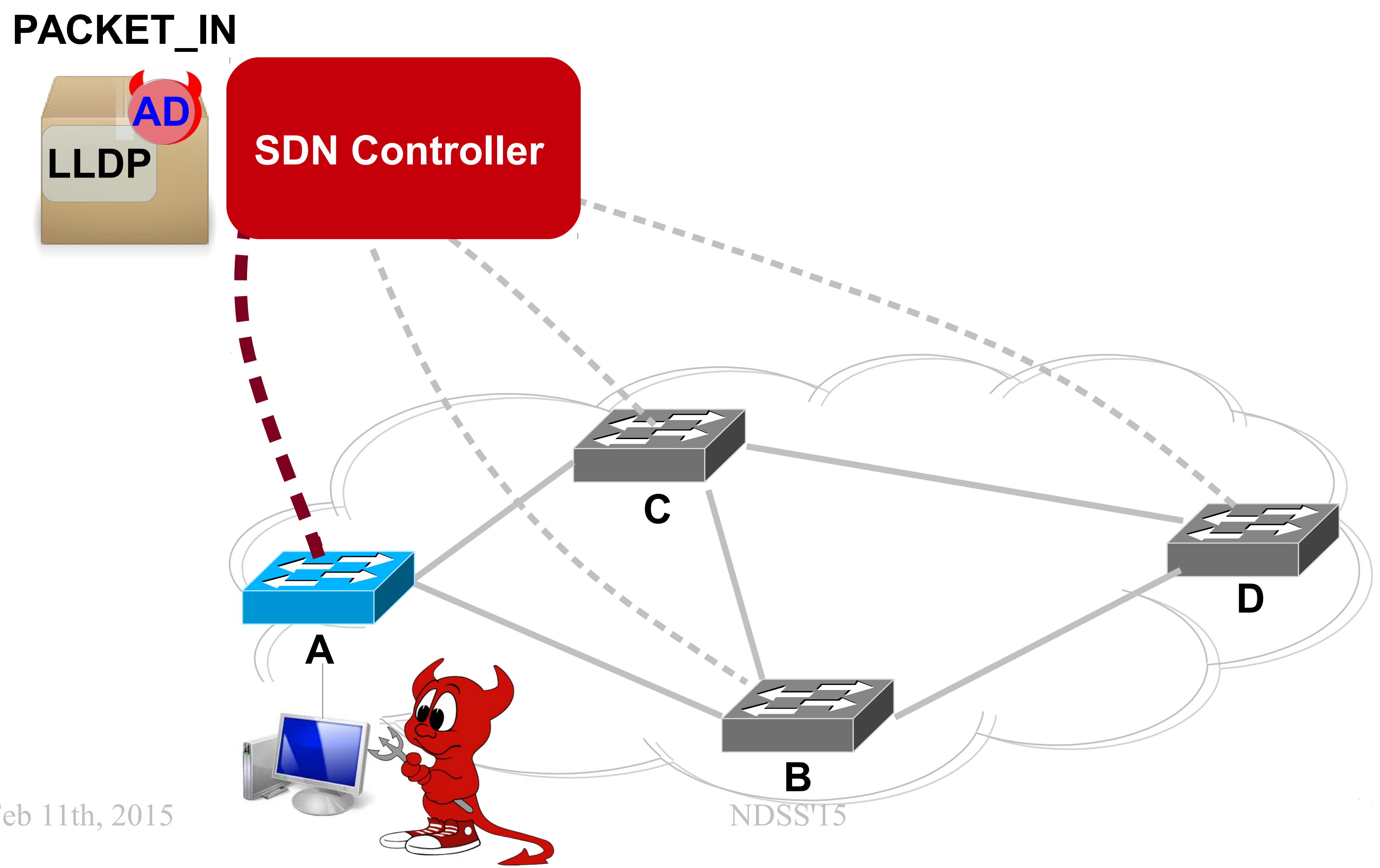
Fake Network Topology Attack



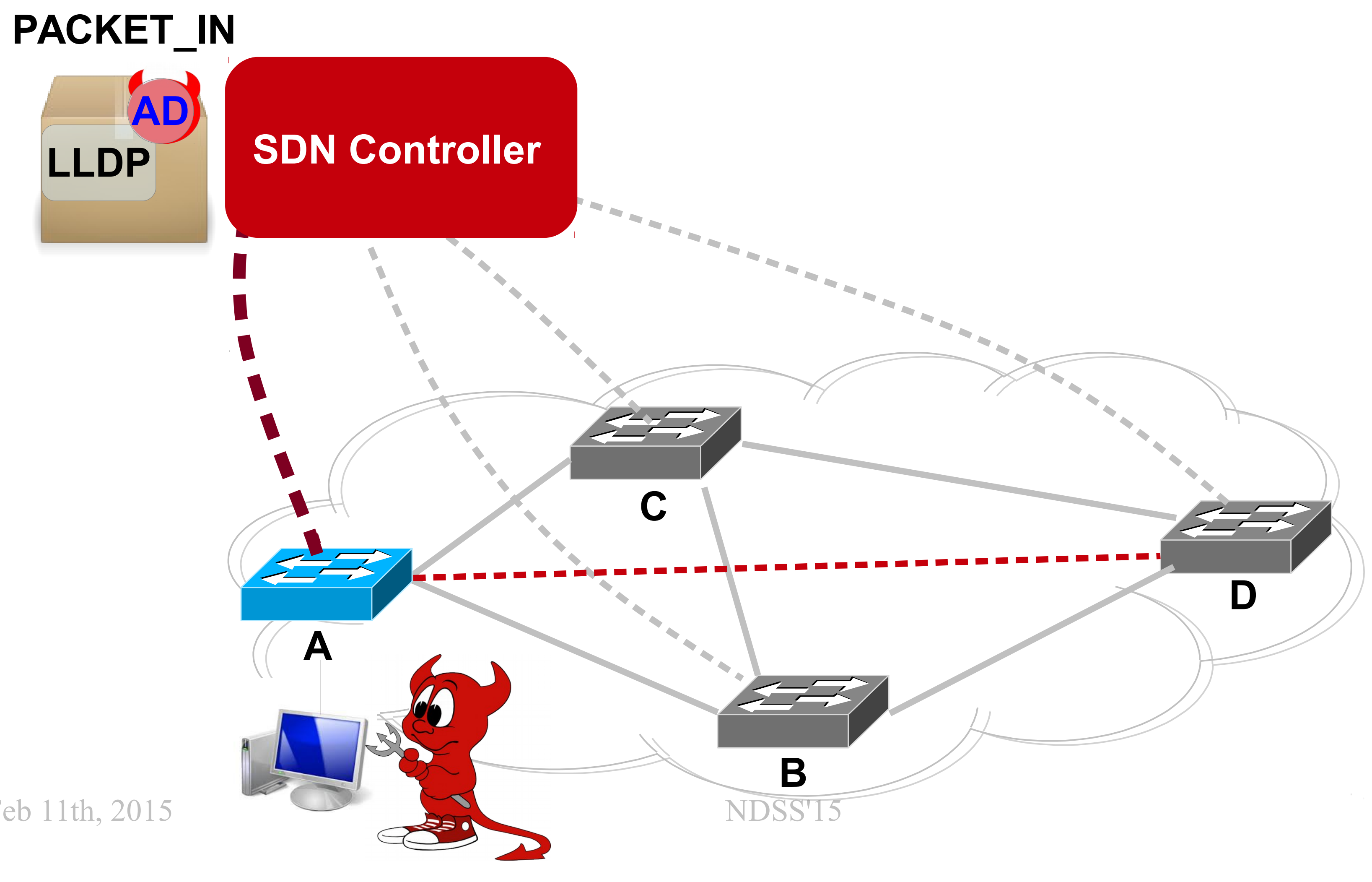
Fake Network Topology Attack



Fake Network Topology Attack

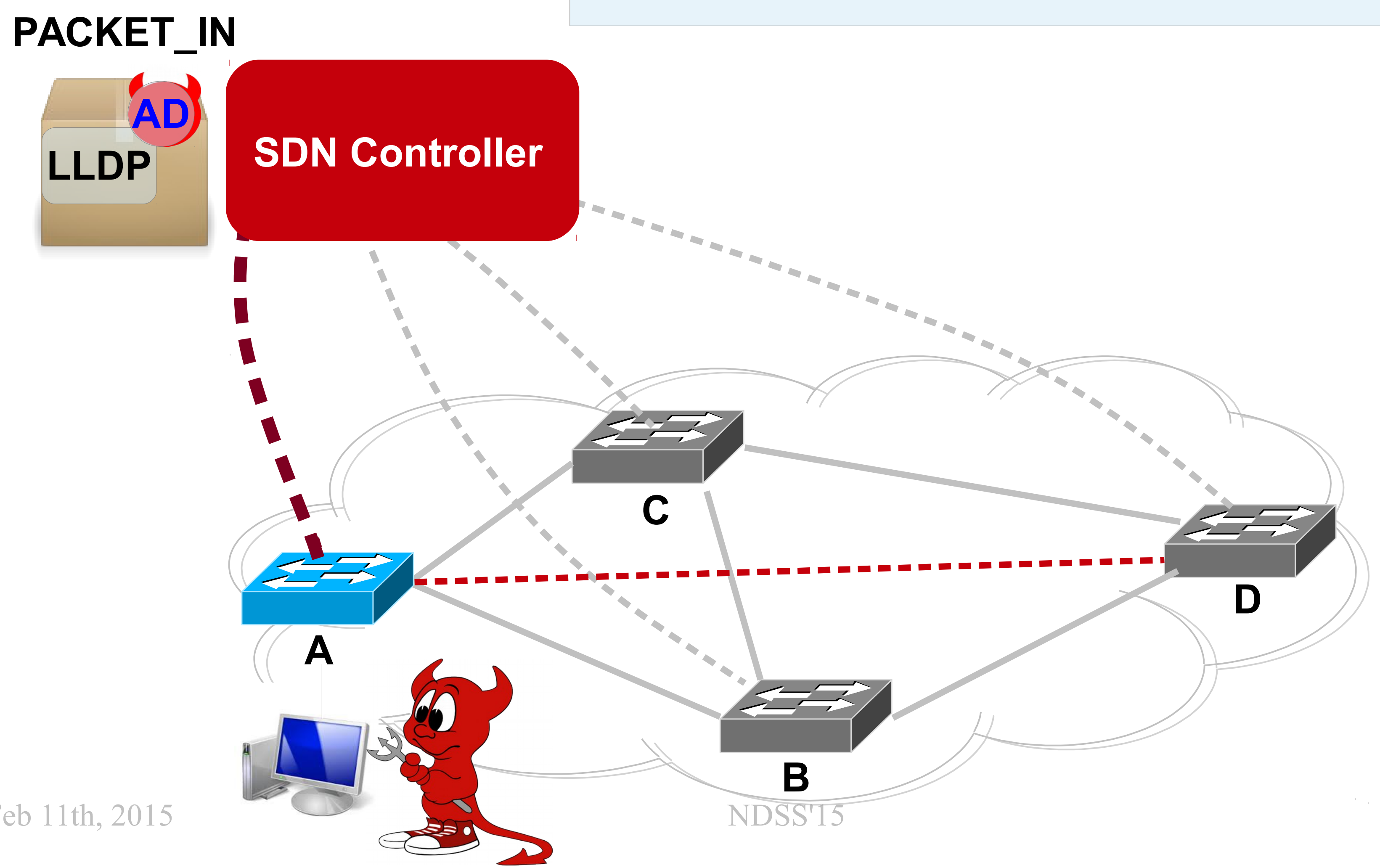


Fake Network Topology Attack



Fake Network Topology Attack

Video demo: http://goo.gl/zRG8bz

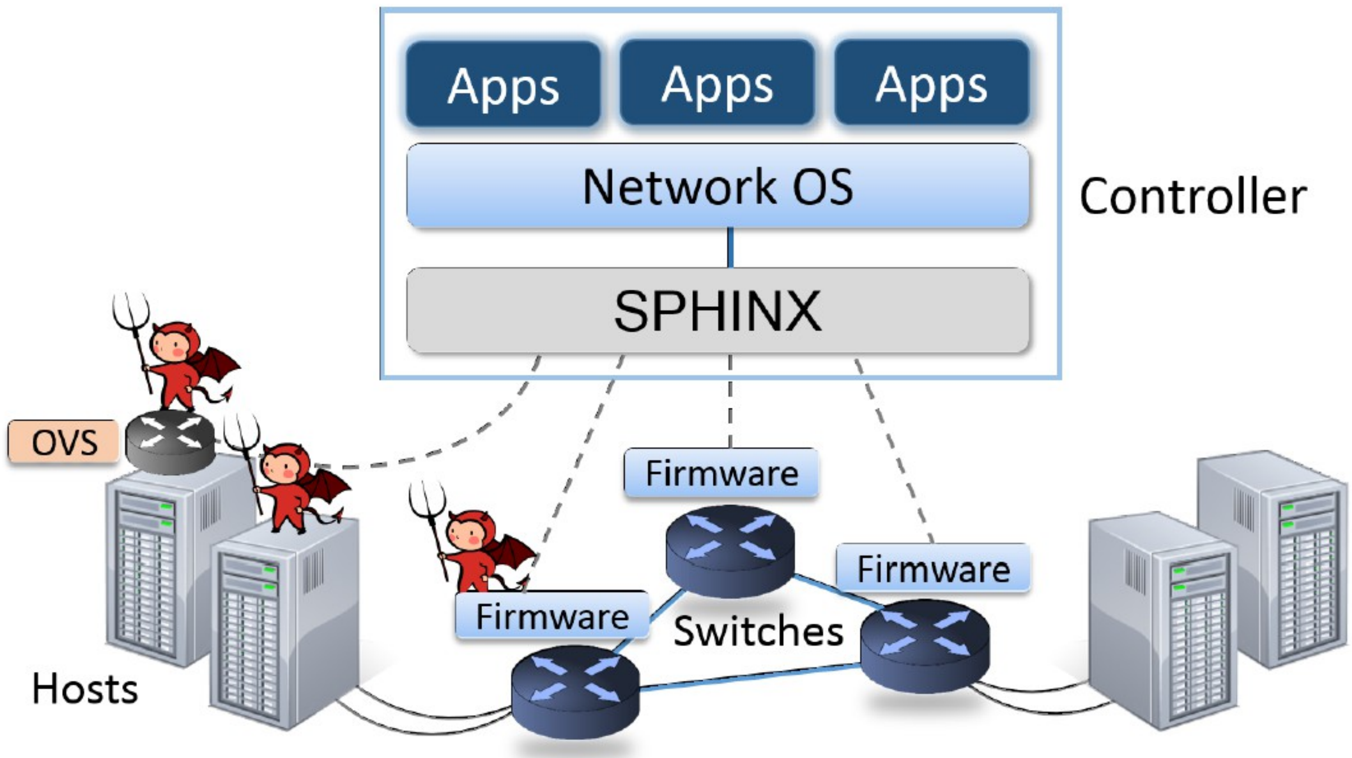


Outline

- SDN Overview
- Motivation
- **Sphinx**
- Implementation
- Evaluation
- Conclusion

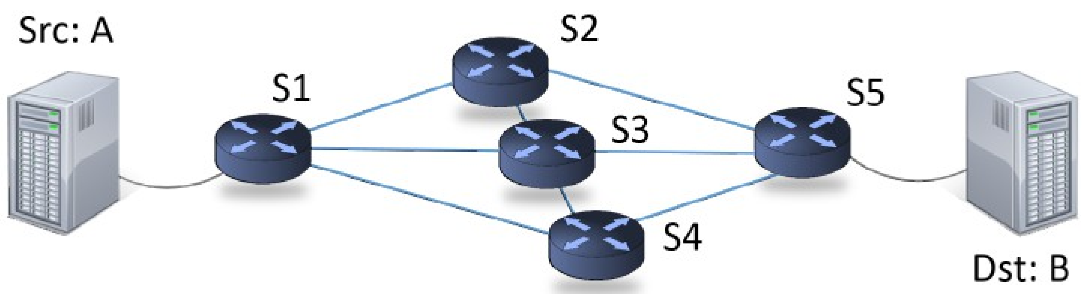
Detecting Security Threats in Real Time

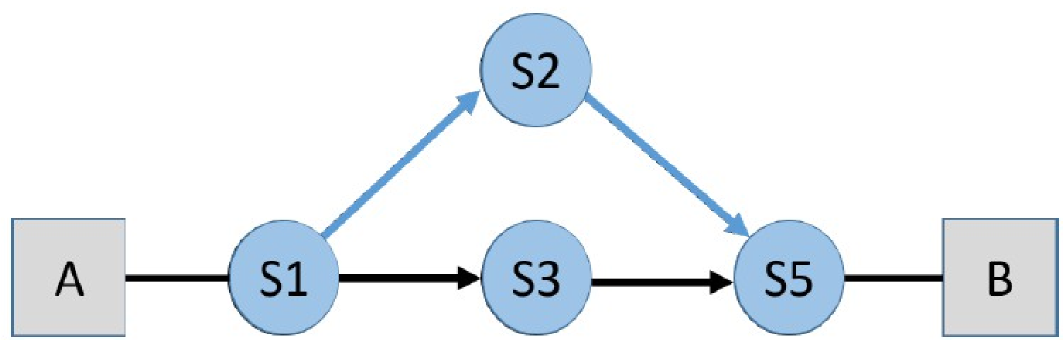
- Verify network actions using OpenFlow metadata
  - All controller communication mediated by a shim
  - Learn network behaviour and automatically generate network invariants



Key Idea: FlowGraphs

Exploit predictability and pattern in topological and data plane forwarding to detect violation

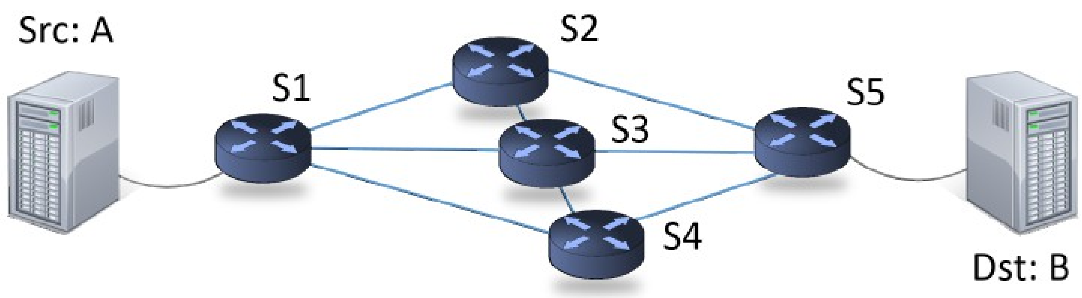


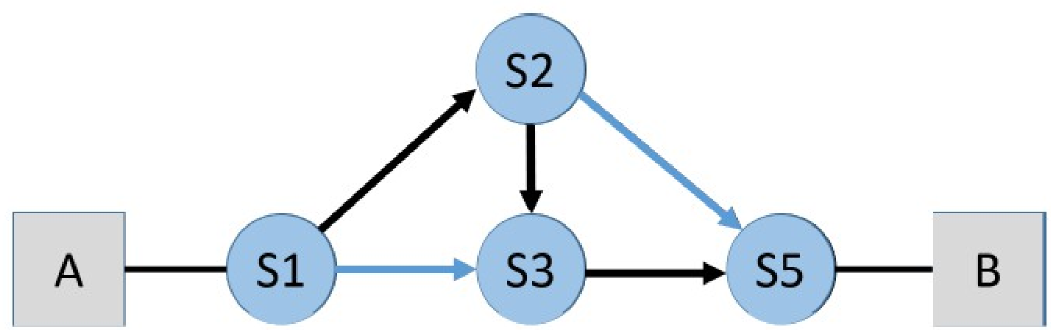


Time T1

Key Idea: FlowGraphs

Exploit predictability and pattern in topological and data plane forwarding to detect violation

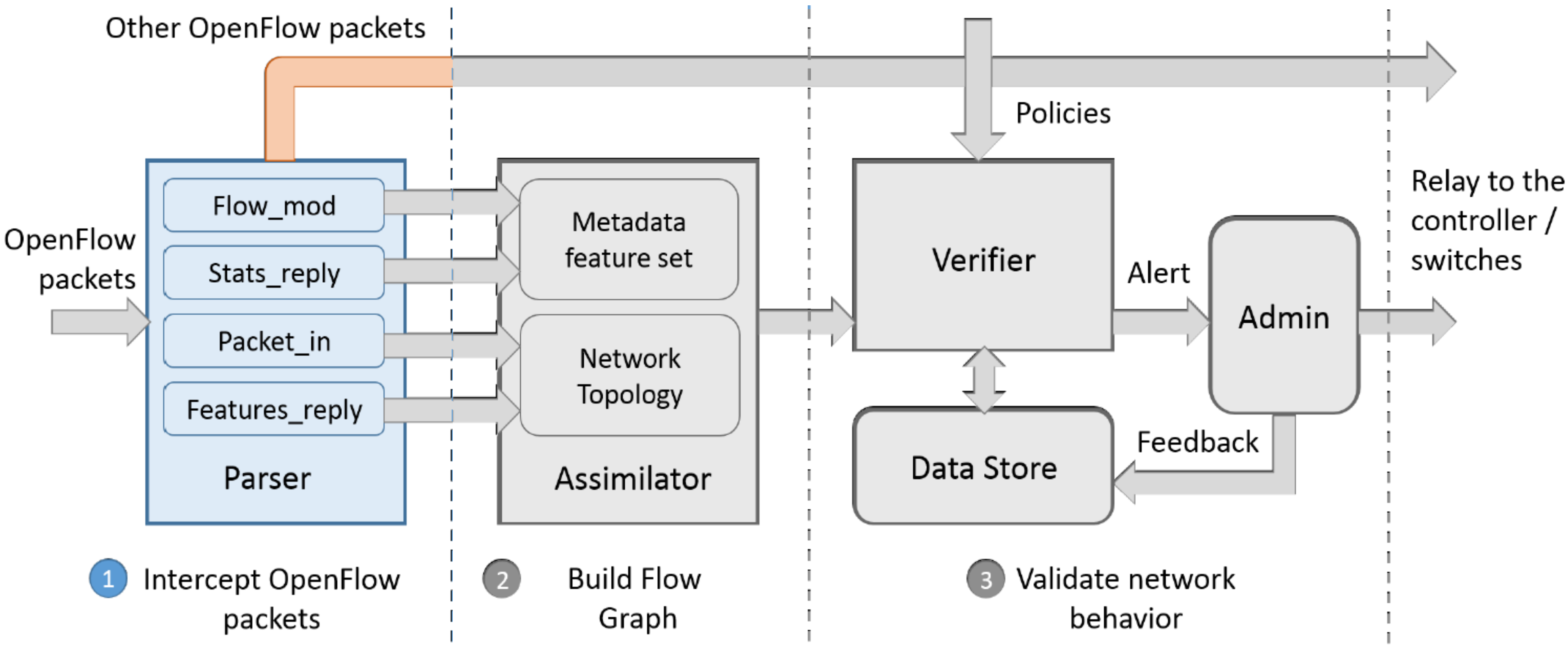




Time T2

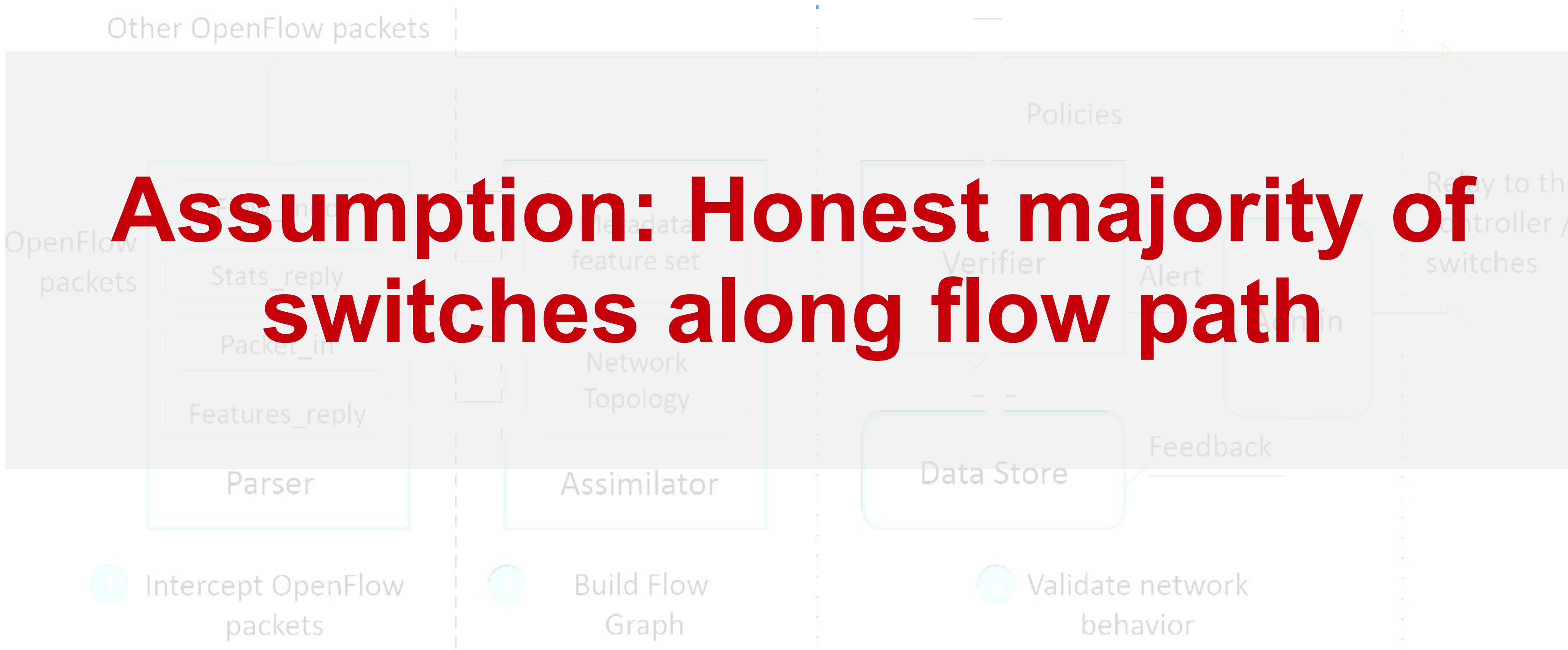
Workflow (I)

- Intercept relevant OpenFlow messages to extract topological and forwarding metadata



Workflow (I)

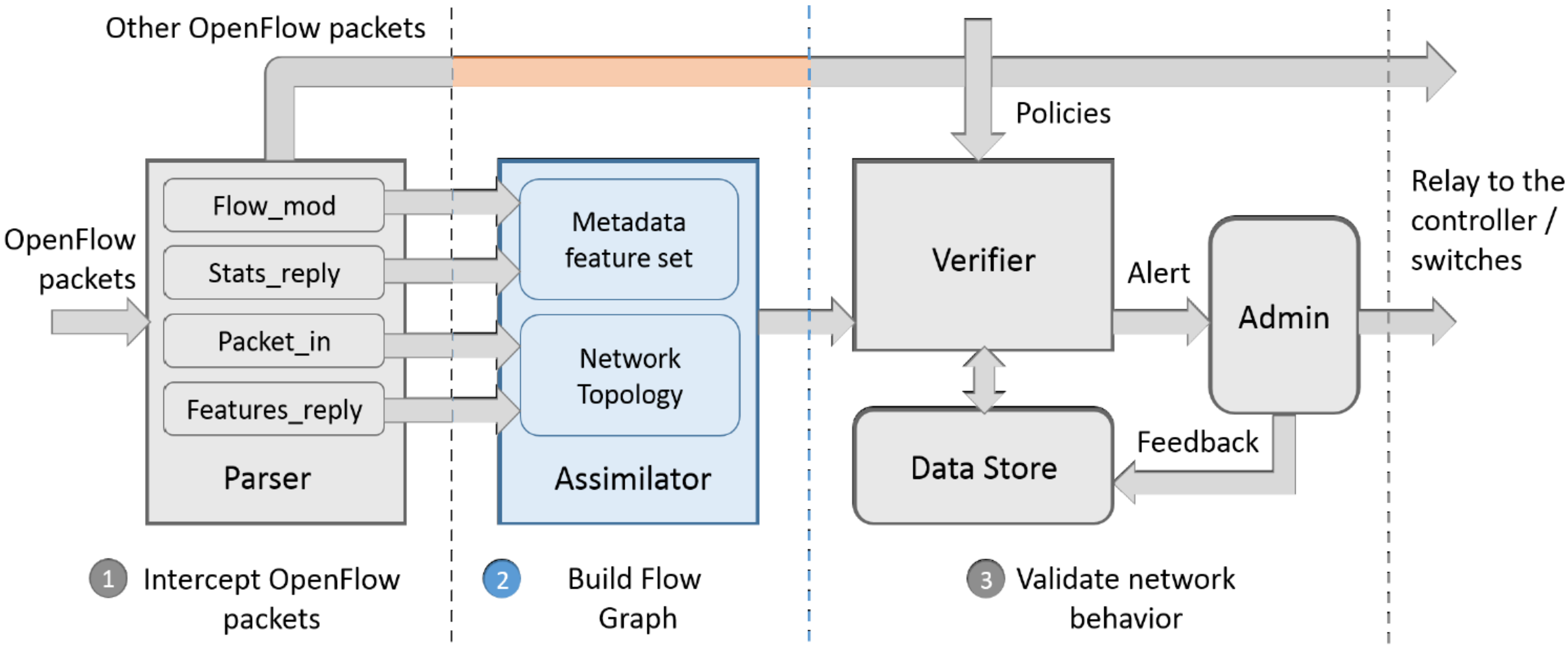
- Intercept relevant OpenFlow messages to extract topological and forwarding metadata



Assumption: Honest majority of switches along flow path

Workflow (II)

- Generate flowgraph constraints from the extracted metadata

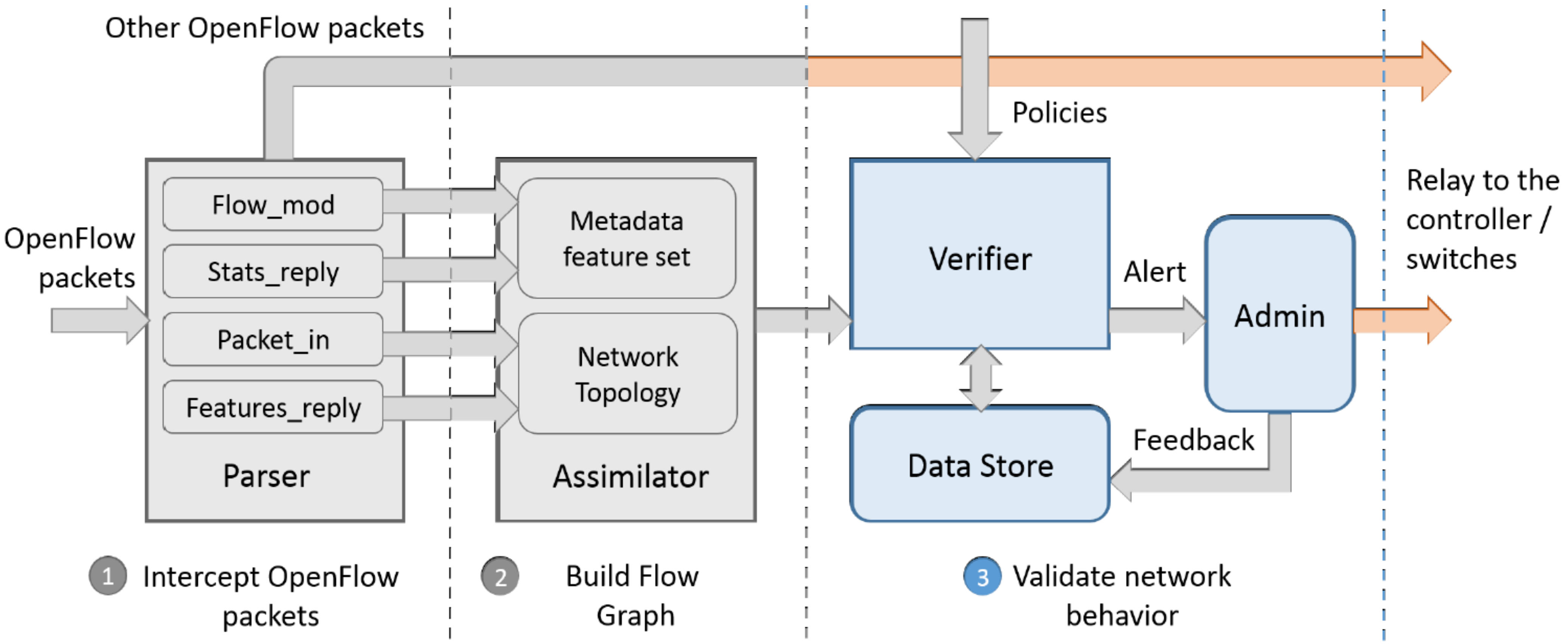


Accurate Characterization of Flows

- Maintain mapping of entities and allowed metadata
  - Hosts (Src MAC/IP/port, Dst MAC/IP/port)
  - Switches (Switch and in/out-port)
  - Flows (Flow match and statistics)
- Incrementally augment the flowgraph with such constraints

Workflow (III)

- Use custom algorithms to detect constraint violations on flowgraphs



Administrator Policies

- Specified in constraint language

Feature	Description
Subject	(SRCID, DSTID), where ∀ SRCID and DSTID ∈ {CONTROLLER | WAYPOINTID | HOSTID | $*$}
Object	{COUNTERS | THROUGHPUT | OUT-PORTS | PACKETS | BYTES | RATE | MATCH | WAYPOINT(S) | HOST(S) | LINK(S) | PORT(S) | etc.}
Operation	IN | UNIQUE | BOOL (TRUE, FALSE) | COMPARE ($\leq$, $\geq$, $=$, $\neq$) | etc.
Trigger	PACKET_IN | FLOW_MOD | PERIODIC

Administrator Policies

- Specified in constraint language

Feature	Description
Subject	(SrcID, DstID), where ∀ SrcID and DstID ∈ {Controller | WaypointID | HostID | $*$}
Object	{Counters | Throughput | Out-ports | Packets | Bytes | Rate | Match | Waypoint(s) | Host(s) | Link(s) | Port(s) | etc.}
Operation	In | Unique | Bool (True, False) | Compare ($\leq$, $\geq$, $=$, $\neq$) | etc.
Trigger	Packet_in | Flow_mod | Periodic

- Example policy to check if all flows from host H3 pass through specified waypoints S2 and S3

```
<Policy PolicyId="Waypoints">
    <Subjects><Subject value="H3, *" /></Subjects>
    <Objects>
        <Object><Waypoint value="S2" /></Object>
        <Object><Waypoint value="S3" /></Object>
    </Objects>
    <Operation value="IN" />
    <Trigger value="Periodic" />
</Policy>
```

Constraint Validation

- Topological state
  - Packet spoofing, controller DoS
  - Fast and deterministic

Constraint Validation

- Topological state
  - Packet spoofing, controller DoS
  - Fast and deterministic
- Forwarding state
  - Flow graph consistency, switch DoS, flow statistics
  - Both deterministic and probabilistic
  - Similarity Index (SI) categorizes nature of flow using statistics observed at switches along the flow path
    - Identify malicious switches along flow path

Outline

- SDN Overview
- Motivation
- Sphinx
- **Implementation**
- Evaluation
- Conclusion

Implementation

- Controller-agnostic proxy between the controller and the switches
  - Prototype compatible with OpenFlow (v1.1.0)
  - Works with OpenDaylight (v0.1.0) and Floodlight (v.0.90)
  - Written in ~2100 Java LOC
  - Uses the fast Netty I/O framework with separate queues for communication in either direction

Outline

- SDN Overview
- Motivation
- Sphinx
- Implementation
- **Evaluation**
- Conclusion

Experimental Setup

- Physical setup of three tiered datacenter topology with 14 switches
- Emulated Mininet network of up to 10K hosts
- Measure
  - Accuracy of deterministic and probabilistic verification
  - Performance impact on end user latency, throughput and policy verification

Accuracy (I)

- Attack detection times under different settings

Attack	Detection time (μs)	
	Physical testbed	$1K$ Mininet hosts
ARP poisoning	44	60
Fake topology	66	80
Controller DoS	75	900
Network DoS	75	164
TCAM exhaustion	n/a	n/a
Switch blackhole	75	900

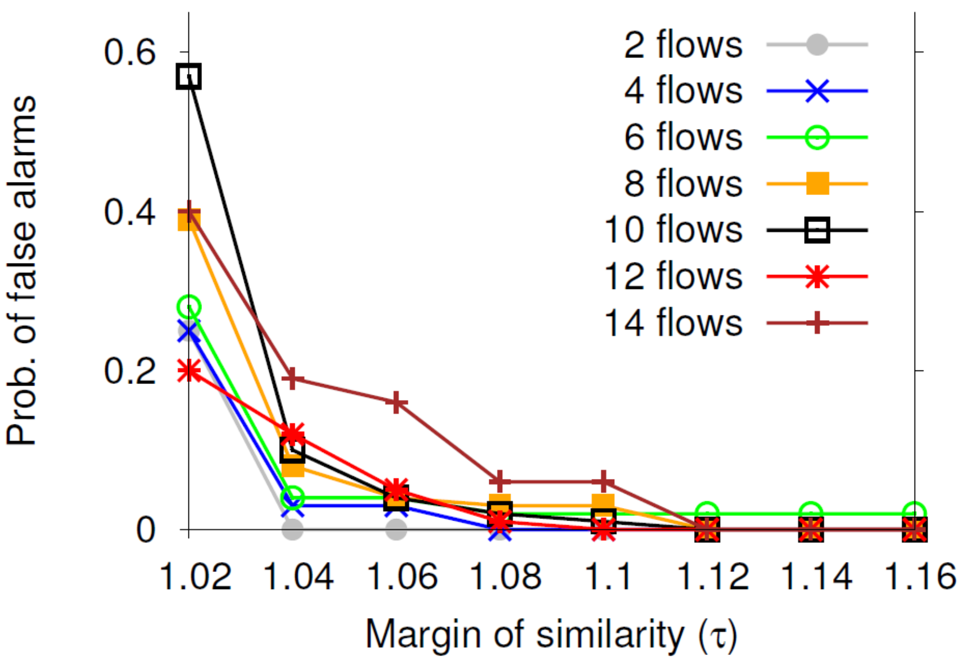
- Measure false alarms generated in three diverse benign traffic traces (14min, 65min and 2hr)
  - Execution raised no alarms

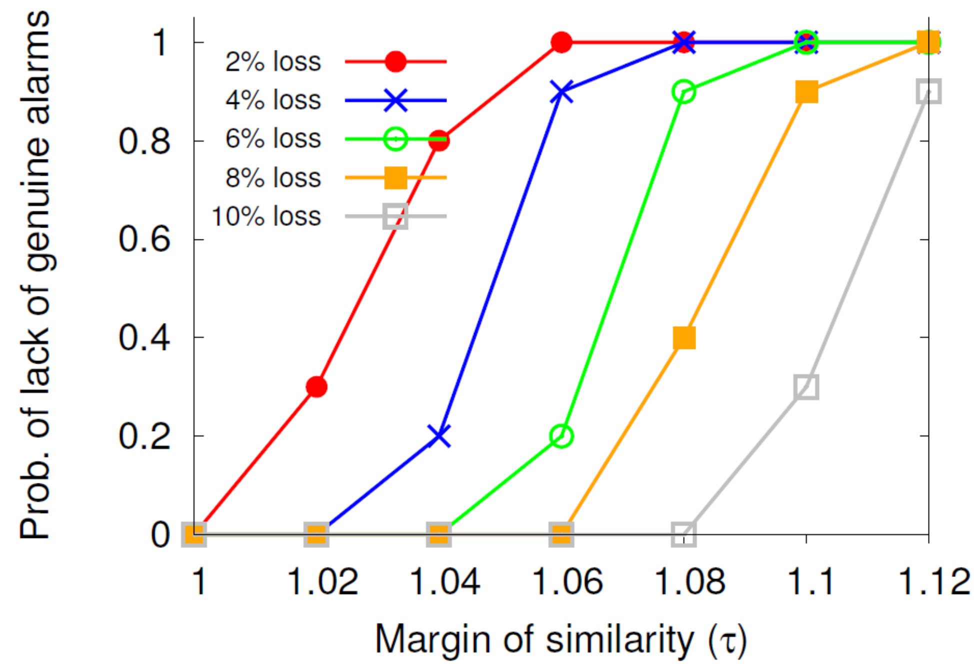
Accuracy (II)

- Probabilistic verification – probability of false alarms and lack of genuine alarms at different margins of similarity (τ)
  - τ = x implies that SI observed at each switch in the flow path must lie between SI/x and SI*x
  - τ = 1 implies that all switches along the flow path must report the same flow statistics
  - τ = 1.045 corresponds to link loss rate of 1%

Accuracy (II)

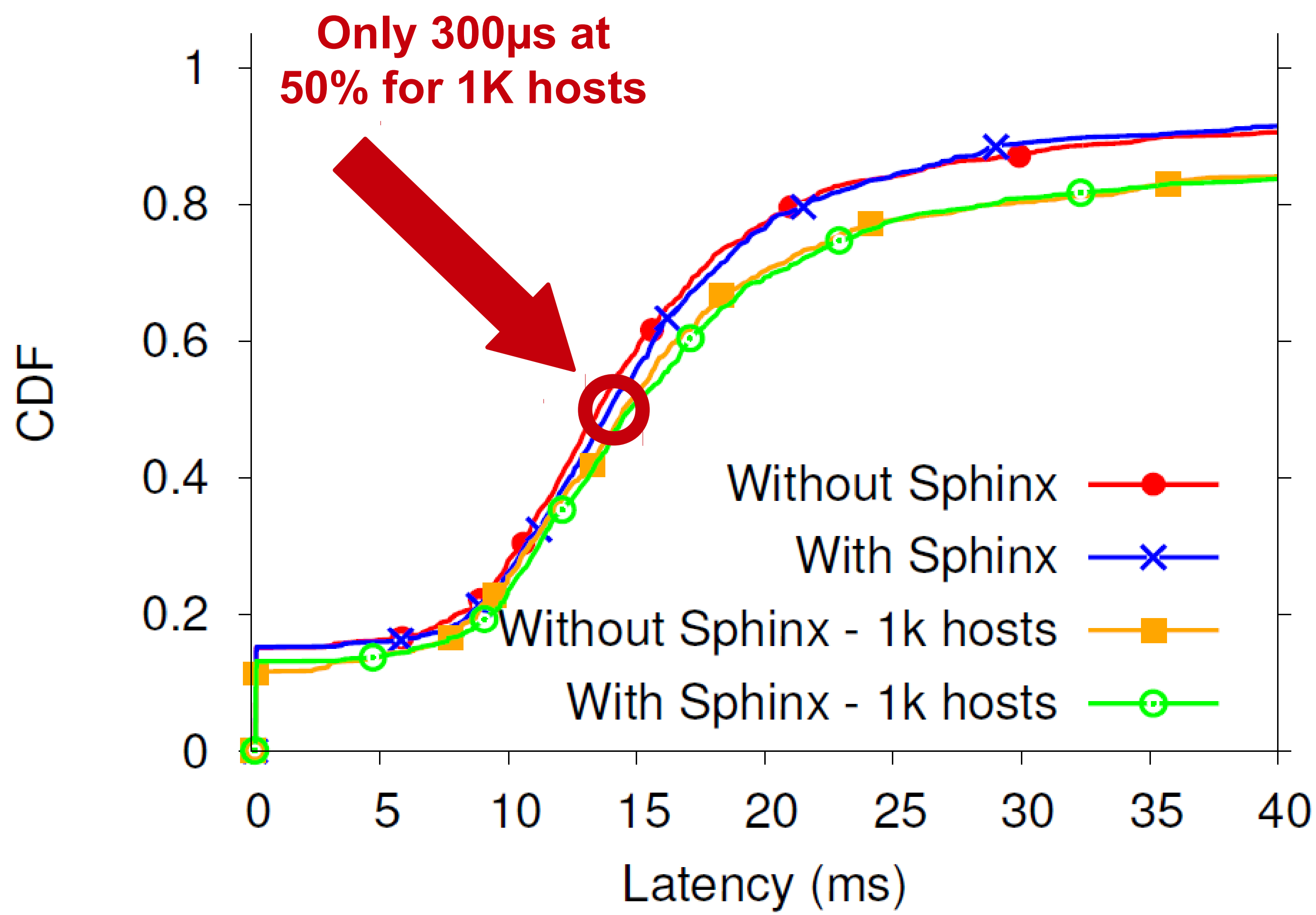
- Probabilistic verification – probability of false alarms and lack of genuine alarms at different margins of similarity (τ)





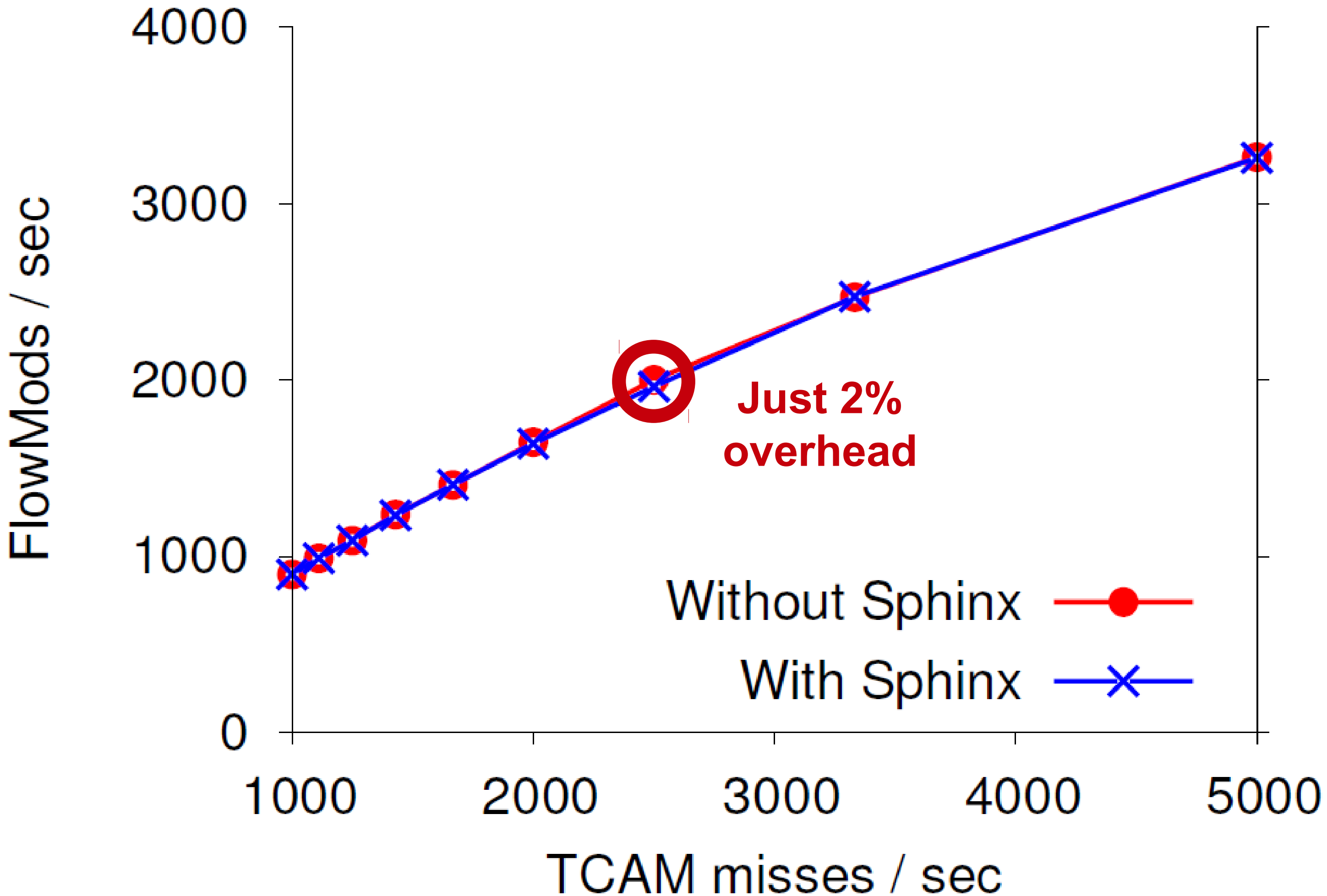
Performance (I)

- End user latency



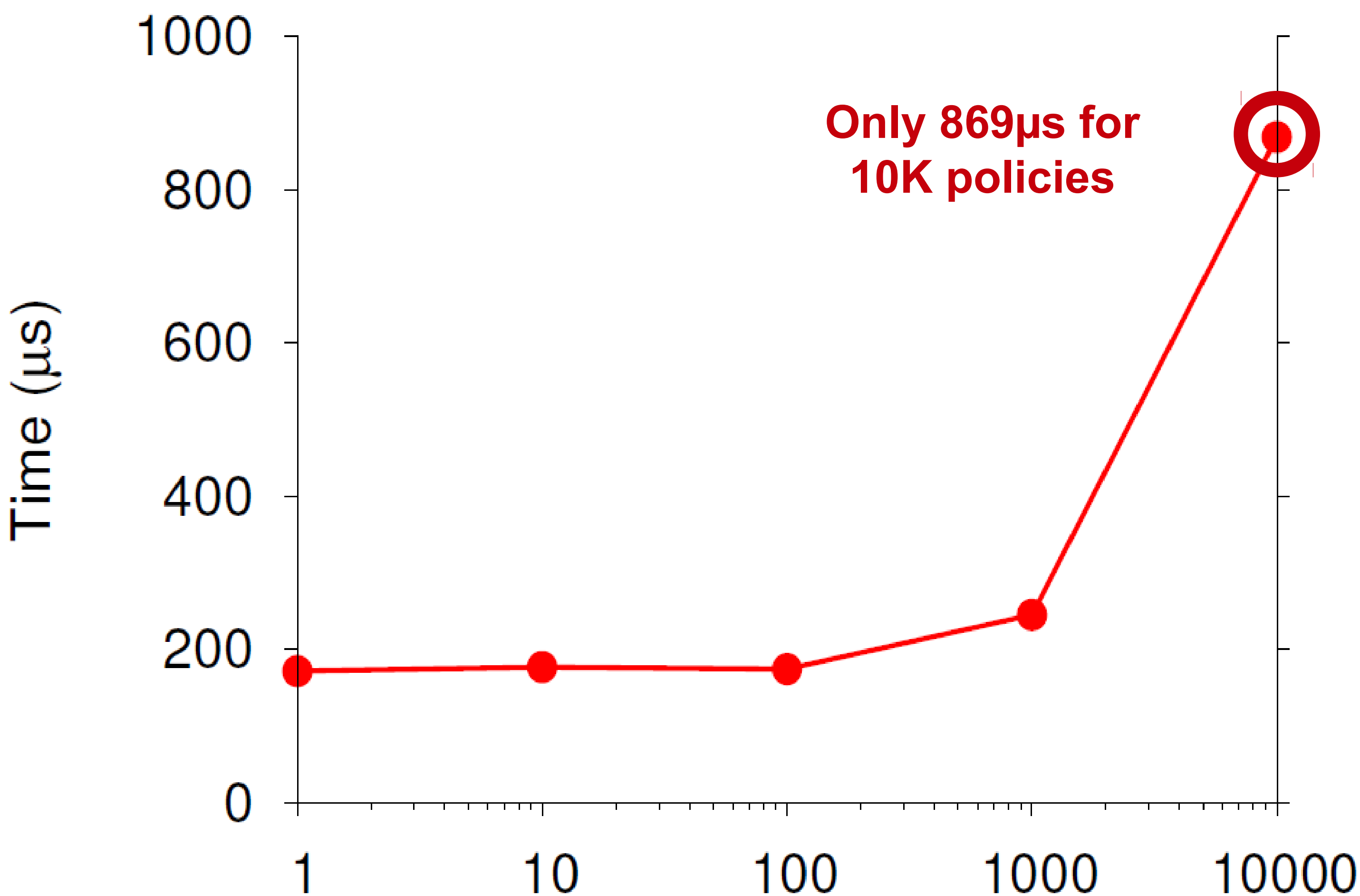
Performance (II)

- Throughput



Performance (III)

- Policy verification



Outline

- SDN Overview
- Motivation
- Sphinx
- Implementation
- Evaluation
- **Conclusion**

Conclusion

- Existing controllers are vulnerable to a wide array of attacks
- Sphinx is a controller agnostic tool that detects security threats originating within SDNs in real time
- Sphinx builds succinct metadata for each network flow and uses both deterministic and probabilistic checks to identify deviant behavior
- Our evaluation shows that Sphinx is practical and imposes minimal overheads

Thank You.

Contact: mohan.dhawan@in.ibm.com