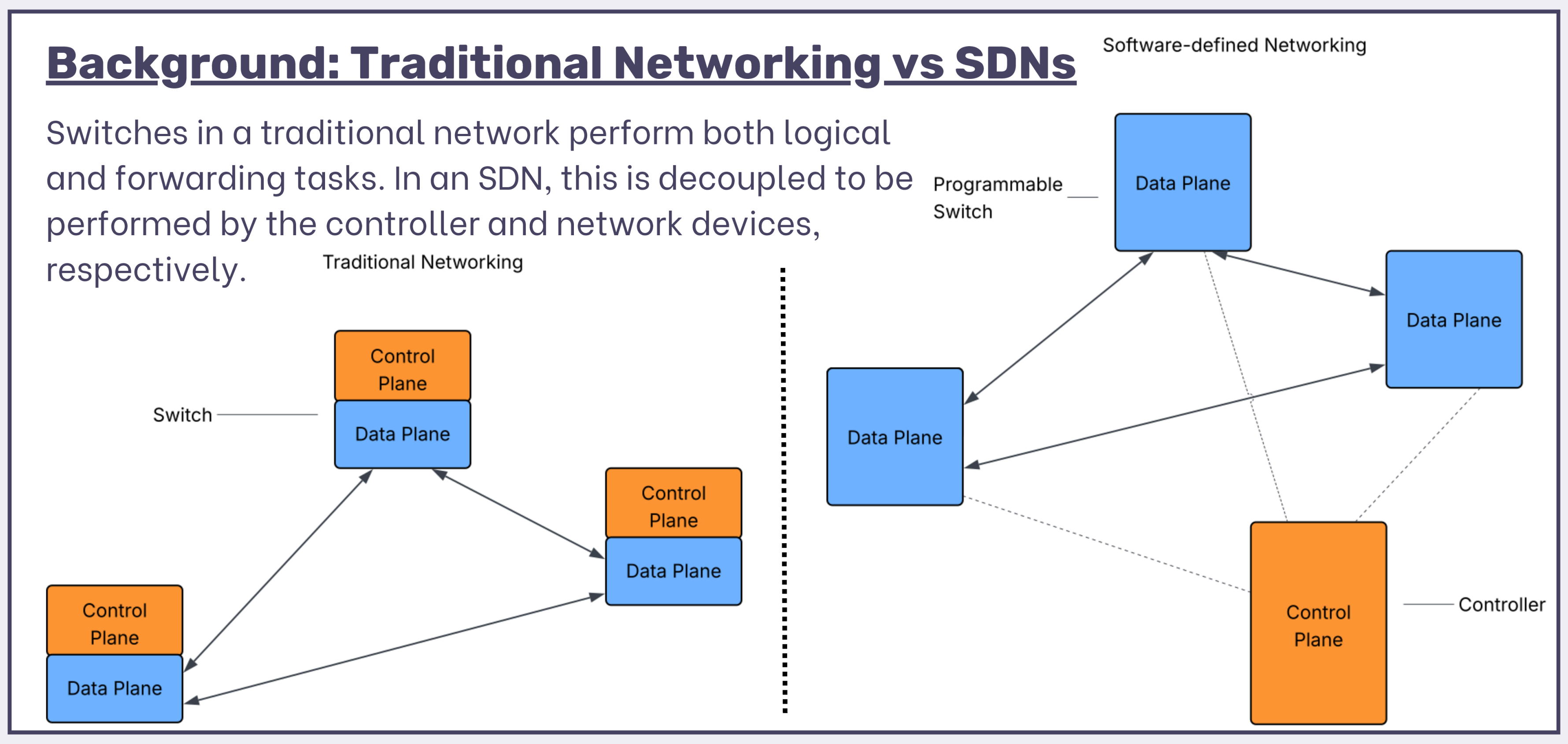
Security-Aware Data Provenance for Software-Defined Networks

Visal Dam, **Fariha Tasmin Jaigirdar**, **Kallol Krishna Karmakar**, **Adnan Anwar**
School of Information Technology, Deakin University

SDNs offer a centralized and programmable approach to network management. However, concerns in data transparency contribute to the lack of security awareness associated with network domains. While this can be addressed by data provenance, existing solutions such as [1, 2] merely collect provenance data and do not support provenance validation. Moreover, current research overlook supposedly-benign network aspects [4], such as switch authentication states, flow rules, and packet paths. Motivated by this, we propose PRISM-Prov, a security-aware provenance framework for SDNs, and evaluate our method against several attack cases. We found that our implementation imposes only 0.021 ms to 0.102 ms to average packet processing times for small to large topologies, respectively - demonstrating low performance costs.

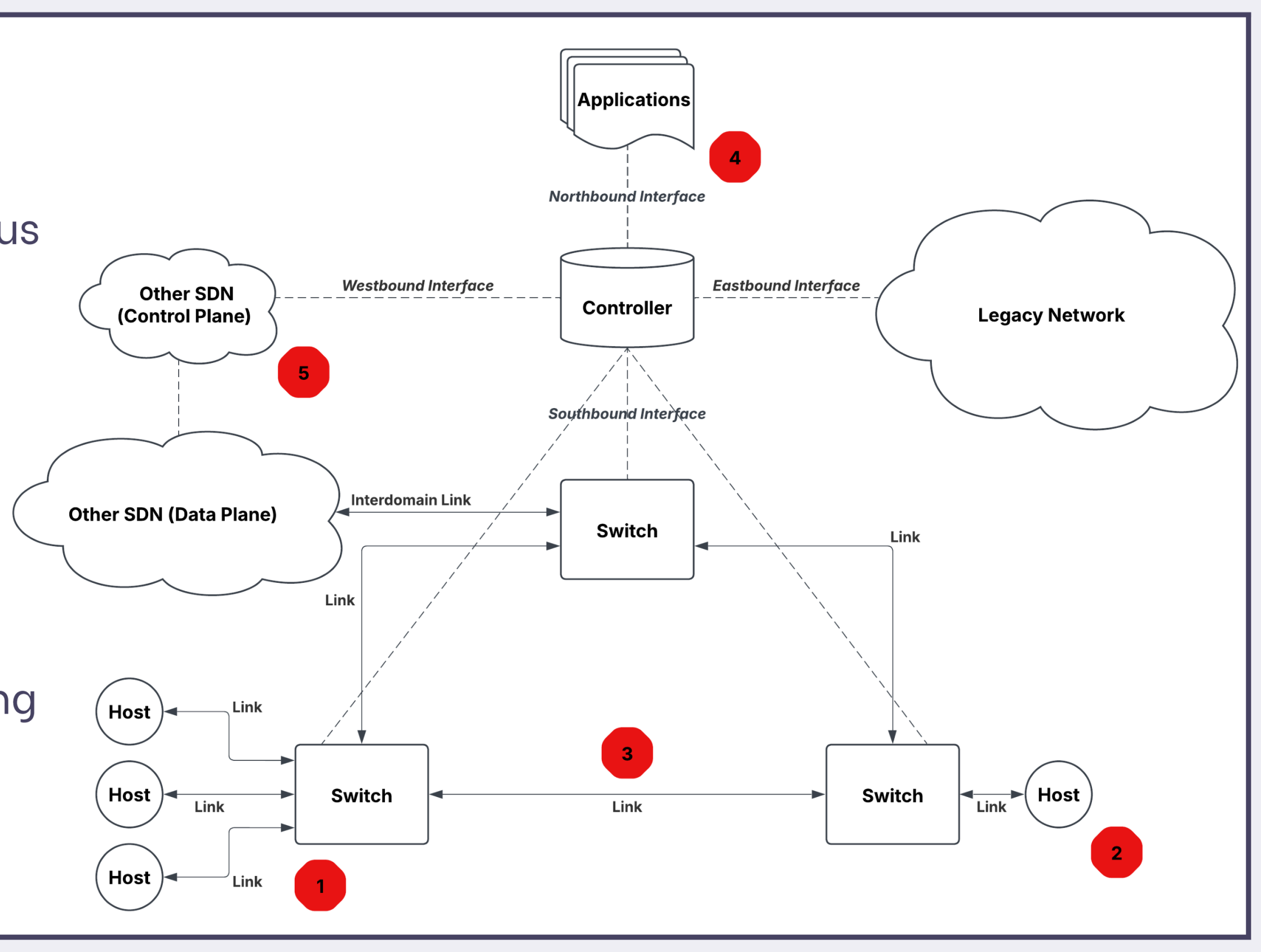
Background: Traditional Networking vs SDNs

Switches in a traditional network perform both logical and forwarding tasks. In an SDN, this is decoupled to be performed by the controller and network devices, respectively.



Threat Model

1) **Device cloning and impersonation**: lack of authentication protocols allow actors to place malicious switches and hosts.
2) **Denial of Service (DoS)**: large number of spoofed packets can overwhelm a switch.
3) **Topology Poisoning**: the controller is tricked to believing false network states, such as fabricated links and presence of invalid devices.
4) **Unauthorized Access**: lack of authorization allow attackers to perform malicious actions, such as injecting flow rules or running malware.
5) **Domain Misconfiguration**: a misconfigured SDN domain may introduce risk to another SDN domain.



Research Gaps

Forensic Tools
- Limited to single attack types.
- Does not support causality analysis.

Machine Learning
- Non-deterministic.
- Training: high costs, lack of standardized data.
- Detection only.

Provenance
- Does not support provenance validation.

Methodology

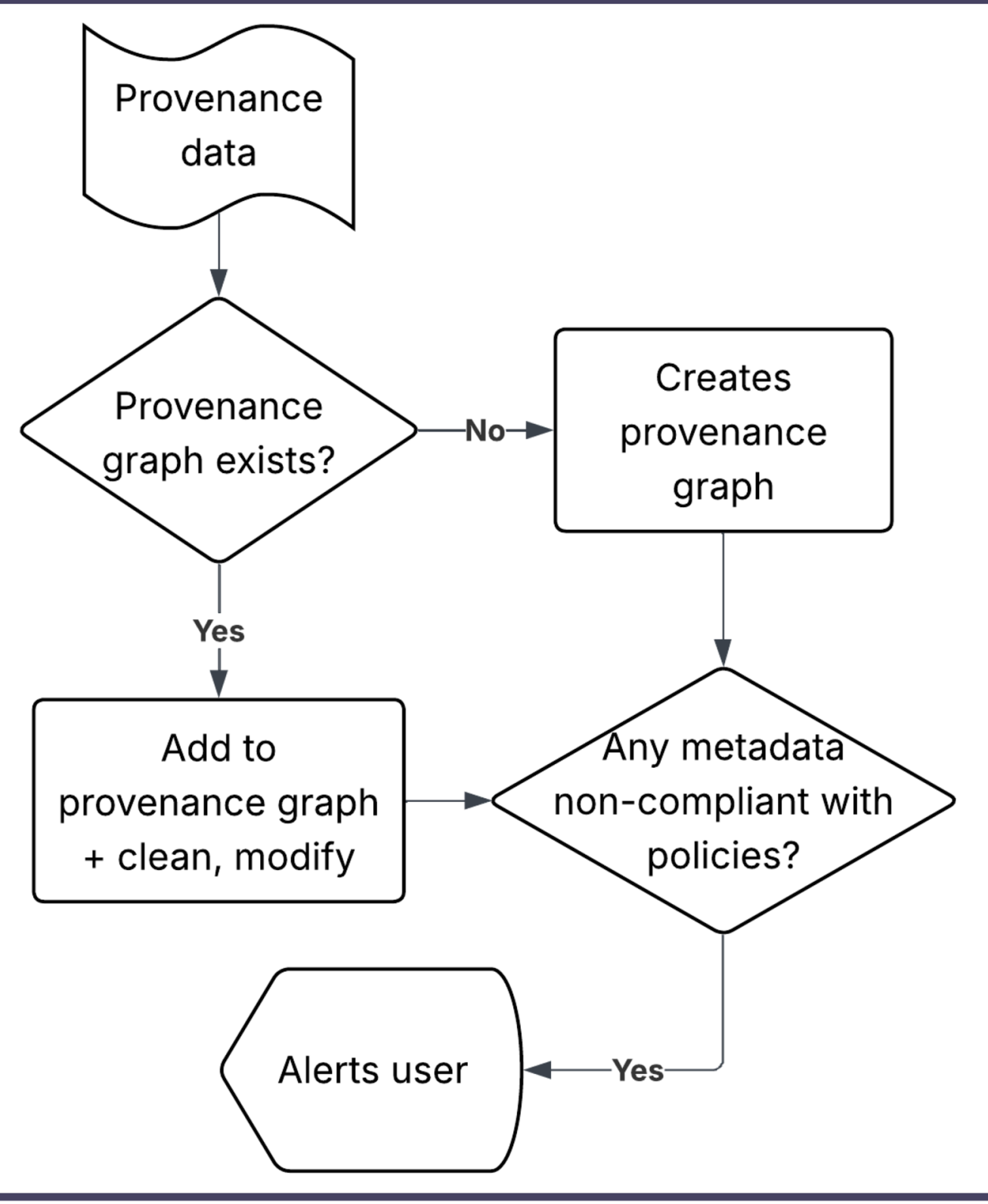
- Provenance system collects **security metadata** alongside network data.
- At each stage of data propagation, these are validated against **security policies** to check for abnormal activity.
- Different network graphs can share the same metadata, thus connecting linked events.

Entity	Security Metadata	Security Policies
Switch	Evidence of TLS, Packet In/Out rate, Power usage	min_ver_tls: 1.2, pkt_rate: < 200 pkt/s, avg_pw: < 0.5 kW/h
Packet	Packet path, Host origin	path_expected: true, from_good_host: true
Flow Rule	Creation time, Parameters	has_pkt_origin: true, permanent: false
Application	Version	min_ver_app: 2.0
Host	Host IP address, Host events	at_good_switch: true, max_policy_violations: 2

Provenance Validation

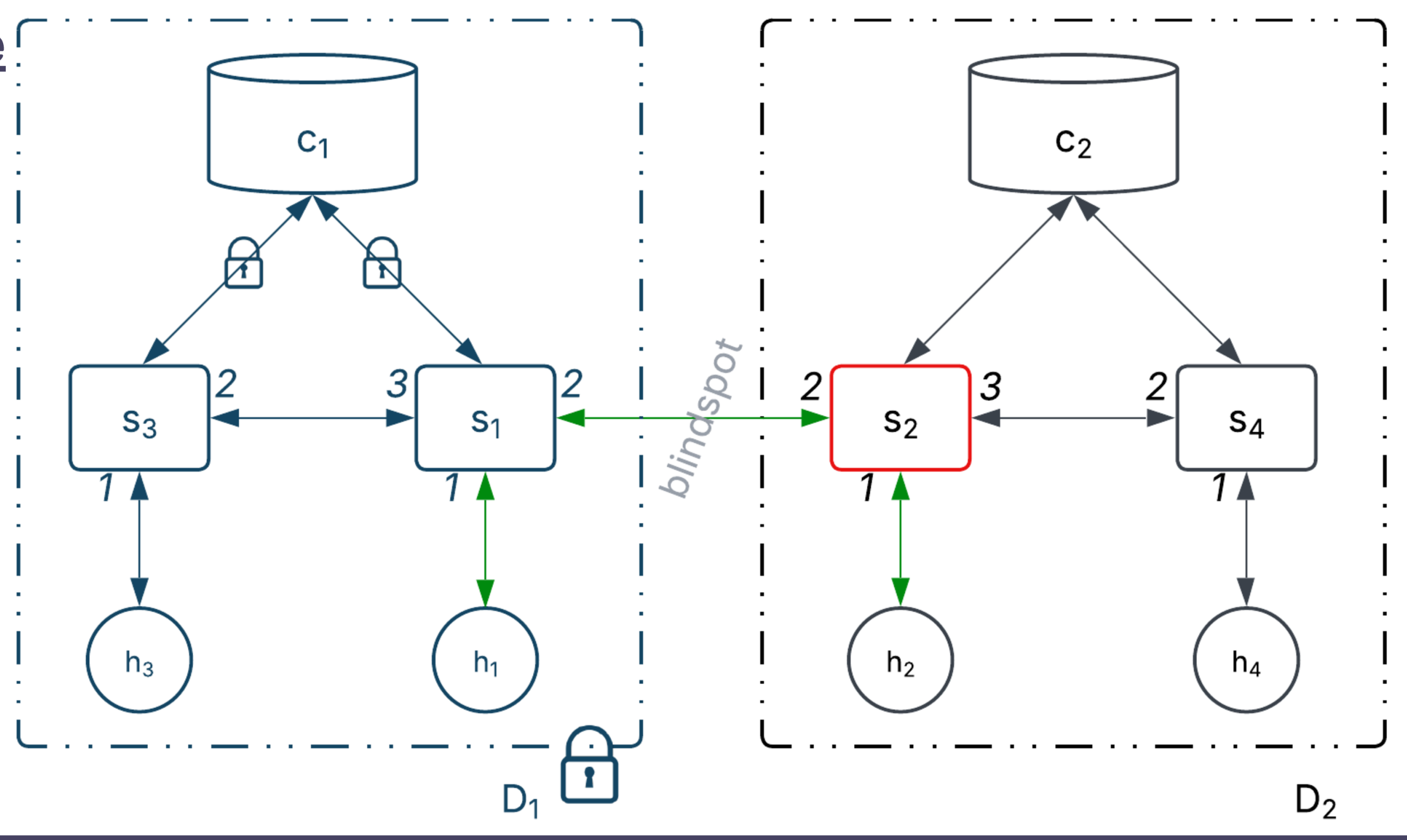
For a given provenance data, first PRISM-Prov logs it as either pertaining to a switch, packet, flow rule, or host. Next, it finds corresponding graphs to identify linked events (based on shared metadata).

Consider a packet. From its provenance graph we see its path history. If any of actual hops deviate from the expected route, the controller may have been poisoned.



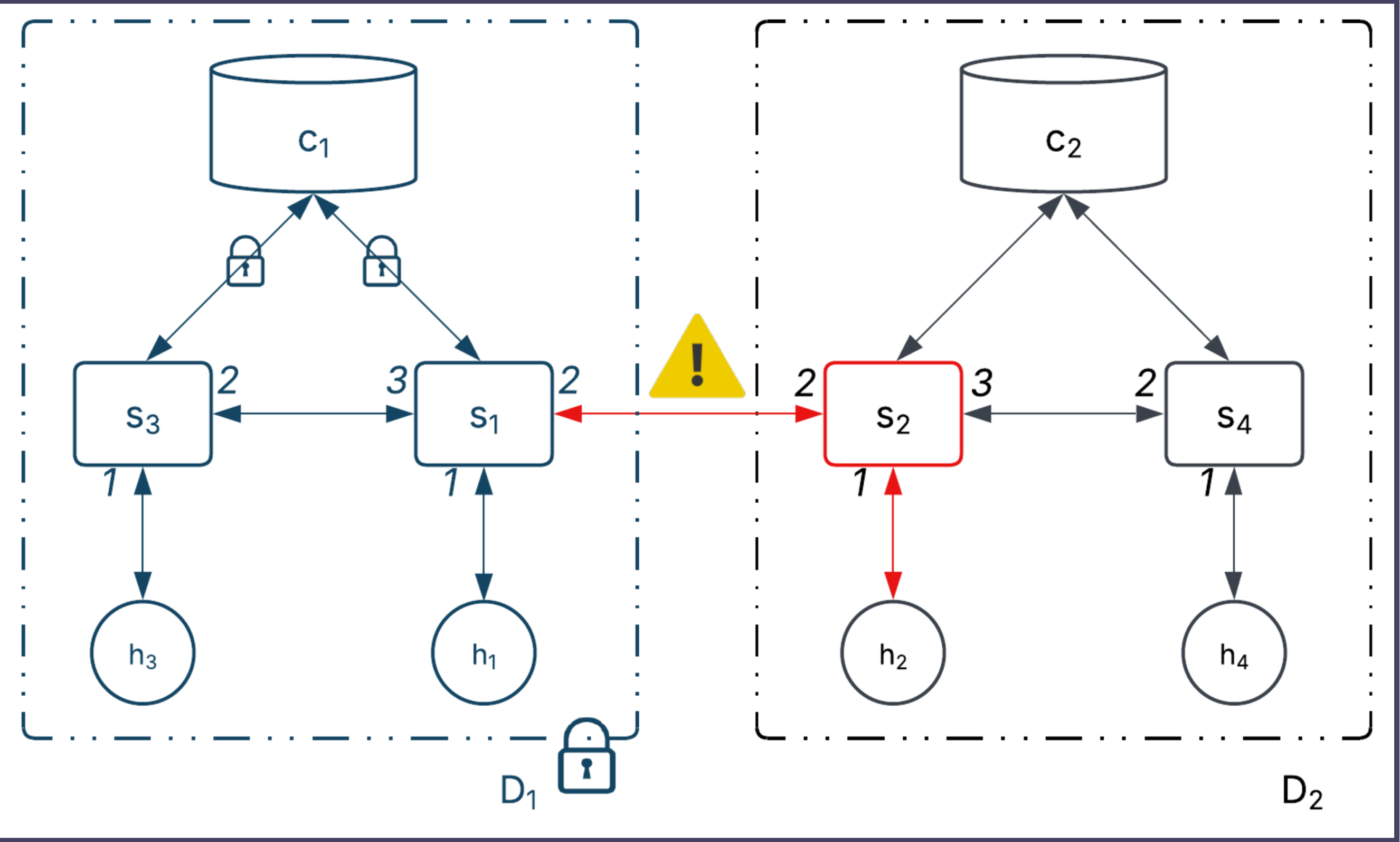
Basic Data Provenance

- Consider two domains *D1* (secured) and *D2*.
- *h2* pings *h1*. The packet goes to *s2*, then *s1*.
- *D1* does not know that *D2* is not secured via TLS. Seeing it as a normal operation, the packet is permitted to enter.
- This lack of validation exploits inter-domain blind spots.



Security-Aware Data Provenance

- There is no evidence that *s2*'s connection to *D2* was secured via TLS.
- Hence, the packet's metadata shows that it originated from a nonsecure switch.
- Thus, the inbound communication is flagged and packet is prevented from entering.



Experimental Setup

Proof-of-concept Implementation: Python, scapy, flask.
Virtualization: Kali Linux (2024), Ubuntu (ver. 24.04), Docker.
Software: Mininet (ver. 2.3.5), ONOS (ver. 2.7.0).
Topology: small (5 switch, 1 host/switch), medium (10 switches, 1 host/switch), large (10 switches, 10 hosts/switch).

Case 1: Host Migration Attack

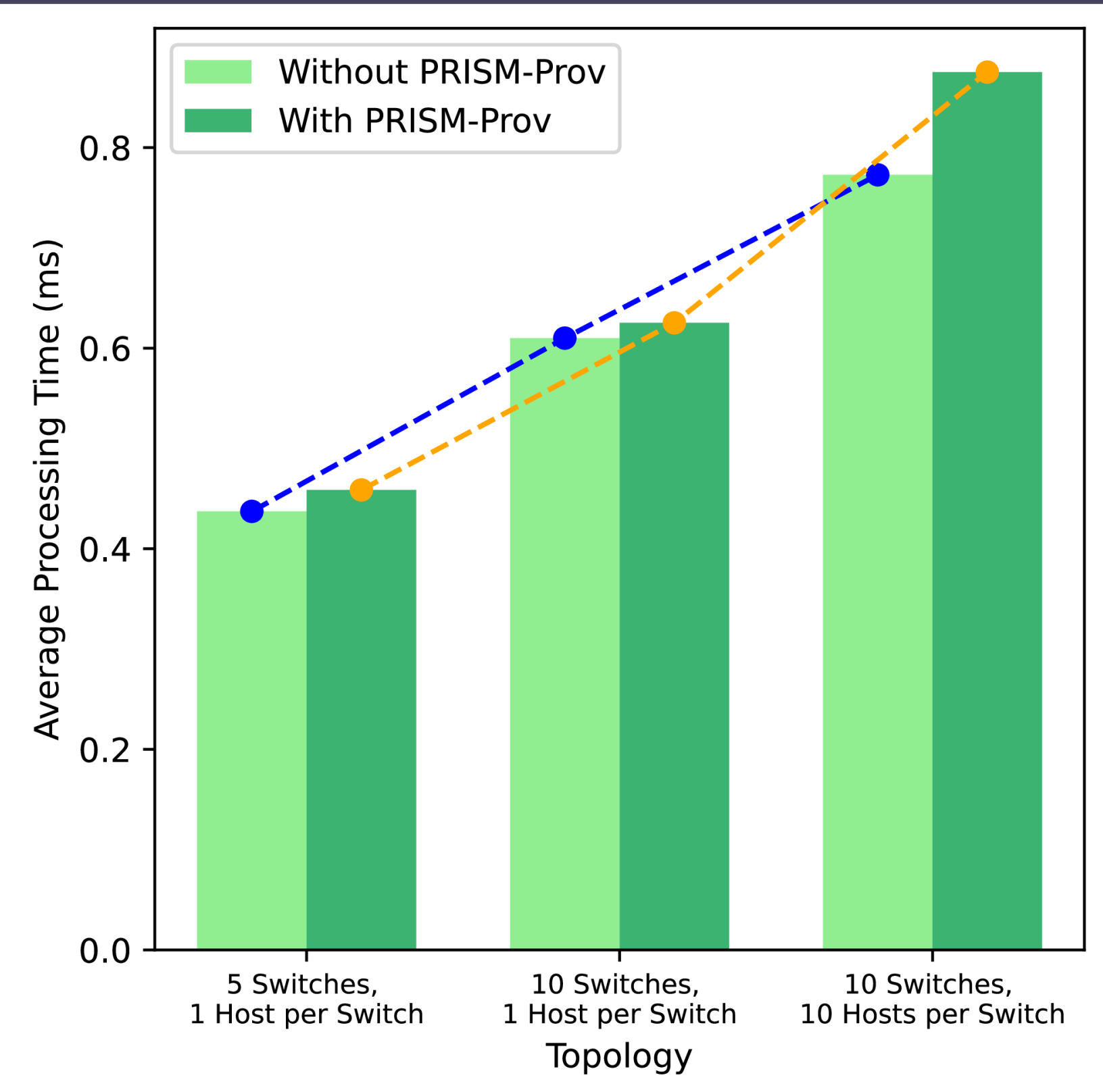
- **Attack**: a malicious host spoofs ARP messages pretending to be another.
- **Detection**: PRISM-Prov detects that a host has changed states, and now shares the same port location as another.

Case 2: Denial of Service

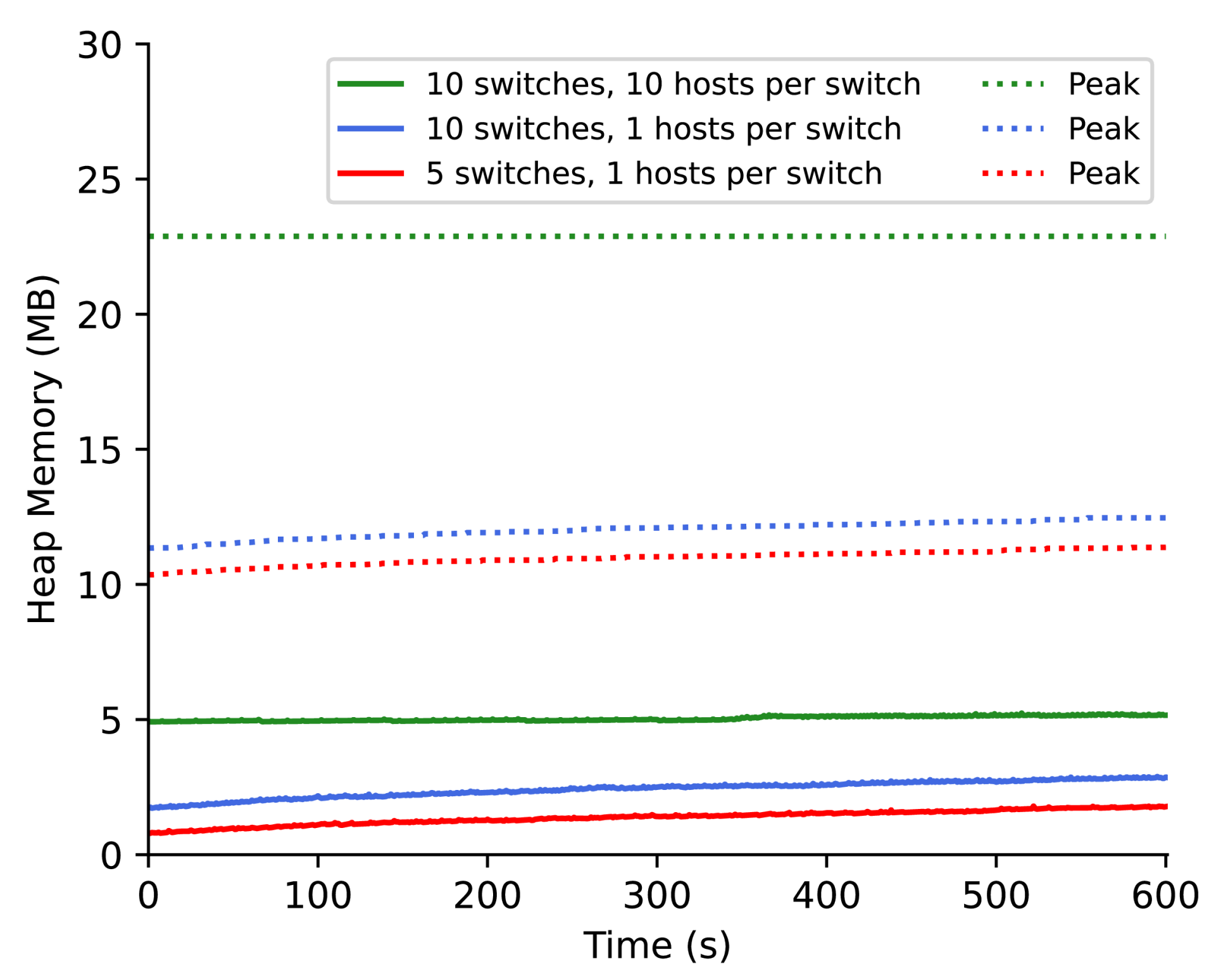
- **Attack**: a switch is flooded with unique packets, each triggering a flow rule.
- **Detection**: PRISM-Prov detects that the inbound packet rate exceeds the threshold, and alerts the network admin.

Case 3: Link Fabrication

- **Attack**: rerouting of LLDP packets fools the controller into believing false routes [4].
- **Detection**: PRISM-Prov detects a mismatch between actual packet paths and best known routes.



(a) Impact on average packet processing time (ONOS)



(b) Heap memory (tracemalloc)

Discussion

- PRISM-Prov is designed to be a **p**rogrammable, **r**eal-time, **i**nteroperable, **s**ecurity-aware, and **m**ulti-domain framework for SDNs, and is based on [3].
- Through dynamic metadata collection and step-by-step validation, attacks were detected instantly after they were executed.
- We evaluate performance costs based on impacts on the controller (fig. a) and allocated heap memory (fig. b), running for 10 minutes. We simulate TCP traffic using iperf and executed some of the attack cases.
- PRISM-Prov imposed 0.021 ms to 0.102 ms to average packet processing time for the small and large topologies - an overhead of 4.89% to 13.4% -, respectively, as measured from ONOS's *ReactivePacketProcessor*. We tracked PRISM-Prov's stable heap memory footprint using *tracemalloc*. Results confirm PRISM-Prov's real-time and low-latency performance for medium-traffic SDNs.
- Unlike previous works, PRISM-Prov is deployed outside of the network to reduce strain on the controller.
- We used logs as sources of network and security metadata. The current proof-of-concept may not be applicable to large scale SDNs with high traffic volumes. Future work should employ lower-level data access (such as in [1]) to improve efficiency and performance costs.

References

[1] B. E. Ujcich, S. Jero, R. Skowyra, A. Bates, W. H. Sanders, and H. Okhravi, "Causal analysis for Software-Defined networking attacks," in 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, Aug. 2021, pp. 3183–3200. [Online]. Available: https://www.usenix.org/system/files/sec21fall-ujcich.pdf
[2] H. Wang, G. Yang, P. Chinprutthiwong, L. Xu, Y. Zhang, and G. Gu, "Towards fine-grained network security forensics and diagnosis in the sdn era," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018,p. 3–16, https://doi.org/10.1145/3243734.3243749.
[3] F. T. Jaigirdar, C. Rudolph and C. Bain, "Prov-IoT: A Security-Aware IoT Provenance Model," 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 2020, pp. 1360-1367, doi: https://10.1109/TrustCom50675.2020.00183.
[4] M. Chen, T. Porta, T. Taylor, F. Araujo, and T. Jaeger, "Manipulating openflow link discovery packet forwarding for topology poisoning," in CCS 2024 - Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery, Inc, 12 2024, pp. 3704–3718, https://doi.org/10.1145/3658644.3690345.

Acknowledgements

- We'd like to thank **Deakin Cyber Research and Innovation Centre** for the opportunity to perform this research.
- We'd like to thank the **28th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID 2025)** for the opportunity to present this research.