National Institute of Informatics (NII) - Tokyo, Japan

# Early-stage anomaly detection and mitigation for large-scale networks

Thien Xuan Phan

University of Information Technology – Vietnam
National University – HoChiMinh City
thienpx@uit.edu.vn
thien@nii.ac.jp

March 3rd, 2015

# Contents

- Motivation: anomaly detection
- Problem statement
- Advantages vs Disadvantages of SDN in anomaly detection
- Proposal approach: Anomaly detection with SDN
  - System Architecture
  - Anomaly detection method
- Analysis: scalability issue of SDN in anomaly detection
- Plan for next steps
- Q&A

# Motivation: anomaly detection

- Increasing number of anomalies such as misconfiguration and remote attacks.
- These Internet traffic anomalies cause a serious problem for the users and Internet service operators:
  - Affect directly the availability of network services
  - Prevent legitimate users from accessing the networks resources.
- Existing anomaly detection approaches: based on the conventional network architecture, demand heavy processing to extract feature information for traffic analysis.

  -> delay time in detection, inflexible in reaction

# Motivation: anomaly detection (cont.)

- Existing anomaly detection approaches:
  - Based on the conventional network architecture
  - Demand heavy processing to extract feature information for traffic analysis.
    - Delay time in detection.
    - Inflexibility and latency in reaction.
    - Even more challenged in large scales networks since a large number of switches/routers need to be investigated.

# Problem Statement

- Focuses on the problem of anomaly detection and protection.
- Architectural solution for anomaly detection on large-scale inter-networks in an early state and flexible reaction in case of network attacks.
- Detection and mitigation method against DDoS based on the proposed architecture as case study in our solution.

# Solution Requirements and Challenges

- Solution requirements
  - Detect anomaly traffic in an early-stage
  - Quickly react to mitigate the possible attack
  - Be applicable for large-scale network including a number of distributed networks
- Challenges:
  - Similarity between abnormal traffic and normal traffic
  - Early-stage detection is challenging since detection is based on analysis of receiving traffic, and retrieving data for analysis take an certain amount of time
  - Challenges in implementation the architecture, deployment and experiment in large-scale network

# Advantages/disadvantages of SDN in anomaly detection

- Advantages:
  - Fast query of flow-level statistics by SDN's Soundbound APIs (OpenFlow)
  - Easily and quickly dropping desired packets/flows by system-defined actions -> mitigation becomes faster and simpler
  - Detecting system will be flexible with SDN's programmability (easier for deploying, flexible in controlling network behavior for detection/mitigation)
- Disadvantages:
  - Only flow-level statistics supported, packet-level statistics not well supported (event-based v.s flow statistics v.s packetIn messages)
  - Limited kind of anomalies may be benefited by applying SDN
  - Scalability issue: limited number of flow entries allowed in SDN physical switches (while detecting system needs to monitor a large number of networking flows)
  - SDN may not benefit much for anomaly detection, not sure SDN-based detector will perform better in detection rate/accuracy

# Proposed approach: anomaly detection with SDN

- Architecture includes a data plane, a control plan and a detector
  - Data plane: for packet forwarding
  - Control plane: controller, operating network devices (switch)
  - Detector: for anomaly detecting
- Detector - controler communication: through Northbound interface
- Controller – switches communication: through Soundbound interface (e.g. OpenFlow)

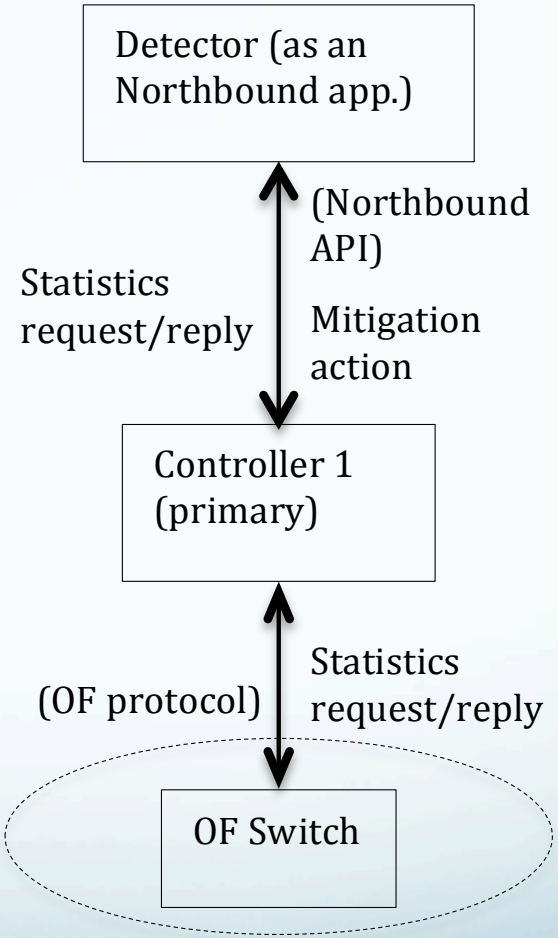



Fig. 1: SDN Anomaly Detecting architecture

# Proposed architecture: anomaly detection with SDN (cont.)

- Detector monitor traffic volume traversing through network and detect anomaly traffic
- Controller queries statistical information of flow-based traffic volume from switches
- Replies from switches received at controller are delegated to Detector through Northbound API
- Detector calculates traffic volume changes in observing traffic flows to find out anomalous flows
- If network attack detected, Detector suggests mitigation actions to controller to block attacking traffics

# Anomaly detecting method

- Inherit flow-based algorithm (ASTUTE [1]) since this kind of algorithm exploit well SDN advantage in monitoring statistics of traffic flows
- 2 main phases:
  - (1) Query statistics from switches
  - (2) Calculate traffic volume changes in flows to find out anomaly
- Algorithm:
  - Once a first packet of a flow arriving a switch -> switch forwards it to controller -> controller add a Flow Entry in which Match Field including 5 tuples {scr IP, src Port, dst IP, dst Port, Proto}
  - Detector creates a Monitoring Table to record traffic volume changes in flows, including fields: {5-tuples, packet count, byte count}

[1] F. Silveira, C. Diot, N. Taft, R. Govinda, "ASTUTE: detecting different classes of traffic anomalies", Proceedings of ACM SIGCOMM 2010 conference, Sep. 2010, India.

# Anomaly detecting method (cont.)

- Algorithm (cont.): For every time interval 5 minutes or {10 min, 15 min, 30 min,...}, repeat 1000 times:
  - 1. Controller sends an Individual Flow Statistics Request to switch
  - 2. Individual Statistics Reply from switch include a list of flow statistics of all flow entries existing in its Flow Table -> controller delegate it to Detector
  - 3. For each statistics in the list (correspond to a flow) -> Detector save information (as an item) to Monitoring Table (MT)
  - 4. For each item in MT, Detector calculate traffic volume change in that flow (ASTUTE-inherited):
    - ✓ 4.1. Substracting packet_count of this query to packet_count of previous query (volume change is called $\delta$f,i)

# Anomaly detecting method (cont.)

- ✓ 4.2. Assume F: number of observing flows, compute sample mean $\delta i$, sample standard deviation $\sigma i$ of volume changes -> computer the K' (Astute assessment value, AAV):

$$\hat{\delta}_i = \sum_{f=1}^{F} \frac{\delta_{f,i}}{F} \quad \therefore \quad \hat{\sigma}_i = \left[ \sum_{f=1}^{F} \frac{(\delta_{f,i} - \hat{\delta}_i)^2}{F-1} \right]^{\frac{1}{2}} \qquad K' = \frac{\delta_i}{\hat{\sigma}_i} \sqrt{F}.$$

- ✓ 4.3. Check if |K'| larger than K(p) -> mark observed flow as anomaly. Threshold K(p): examined through experiment, possible values: 3, 6, 9
- ✓ 4.4. Detector saves the detected anomalous flows.

# Analysis: flow entry limit of SDN and its affect in anomaly detection

- Scalability issue
  - SDN physical switch: maximum number of flow entries (max-NF): just about thousands (e.g. 1000 entries for Pica8 10G switch)
  - Our analysis based on real network traffic data (tcpdump file recorded in backbone network in 15 minutes): number of flows can reach the value about ~1.5 million
- Our approach: decrease max-NF by Timeout and sampling
  - Exploit Flow Expiry mechanism (of SDN), use 'Timeout' field in each flow entry (less timeout may leads to less NF)
  - Use sampling technique to decrease NF needed to be monitored for anomaly detection

# Analysis: scalability issue of SDN in anomaly detection (cont.)

- Evaluation: ratio of expired flows vs Timeout (Tau) and Sampling rate (switch actions simulated on Python program)
  - Tau = {0.5, 1, ..., 15}
  - Sampling rate = {0.1, 0.01, 0.001}

| Tau (minute) | Sampling rate | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P=0.1 | | | | P=0.01 | | | | P=0.001 | | | |
| | #Existing flows | #Ignored flows | #Expired flows | Ratio expired flows (%) | #Existing flows | #Ignored flows | #Expired flows | Ratio expired flows | #Existing flows | #Ignored flows | #Expired flows | Ratio expired flows |
| 0.5 | 355535 | 1531098 | 43632 | 10.93 | 57989 | 1680260 | 7670 | 11.68 | 8744 | 1694955 | 1069 | 10.89 |
| 1 | 362866 | 1530793 | 27326 | 7.00 | 60687 | 1679931 | 4692 | 7.18 | 9114 | 1694946 | 624 | 6.41 |
| 2 | 371583 | 1529447 | 14146 | 3.67 | 63002 | 1679844 | 2020 | 3.11 | 9446 | 1694937 | 272 | 2.8 |
| 3 | 374853 | 1528318 | 9632 | 2.51 | 63384 | 1679721 | 1382 | 1.98 | 9535 | 1694958 | 165 | 1.7 |
| 4 | 375615 | 1528558 | 7426 | 1.94 | 63791 | 1679726 | 1032 | 1.59 | 9660 | 1694890 | 122 | 1.25 |
| 5 | 378180 | 1527850 | 4660 | 1.22 | 64425 | 1679730 | 592 | 0.91 | 9634 | 1694971 | 98 | 1.01 |
| 6 | 379653 | 1527288 | 3058 | 0.8 | 64496 | 1679675 | 385 | 0.59 | 9606 | 1694938 | 38 | 0.39 |
| 7 | 380927 | 1526598 | 2157 | 0.56 | 64280 | 1679758 | 303 | 0.47 | 9737 | 1694887 | 29 | 0.3 |
| 8 | 381368 | 1526584 | 1429 | 0.37 | 64918 | 1679394 | 184 | 0.28 | 9702 | 1694906 | 23 | 0.24 |
| 9 | 380912 | 1527206 | 1031 | 0.27 | 64670 | 1679641 | 132 | 0.2 | 9689 | 1694878 | 16 | 0.16 |
| 10 | 382680 | 1526348 | 608 | 0.16 | 64730 | 1679559 | 80 | 0.12 | 9717 | 1694927 | 5 | 0.05 |
| 11 | 381316 | 1527070 | 360 | 0.09 | 64993 | 1679583 | 40 | 0.06 | 9691 | 1694916 | 10 | 0.1 |
| 12 | 381816 | 1526539 | 189 | 0.05 | 64840 | 1679474 | 22 | 0.03 | 9796 | 1694846 | 6 | 0.06 |
| 13 | 381775 | 1526886 | 84 | 0.02 | 64489 | 1679747 | 15 | 0.02 | 9919 | 1694894 | 0 | 0 |
| 14 | 381415 | 1527384 | 15 | 0 | 65030 | 1679541 | 1 | 0 | 9659 | 1694843 | 0 | 0 |
| 15 | 382296 | 1527096 | 0 | 0 | 64863 | 1679495 | 0 | 0 | 9712 | 1694929 | 0 | 0 |

# Analysis: scalability (cont.)

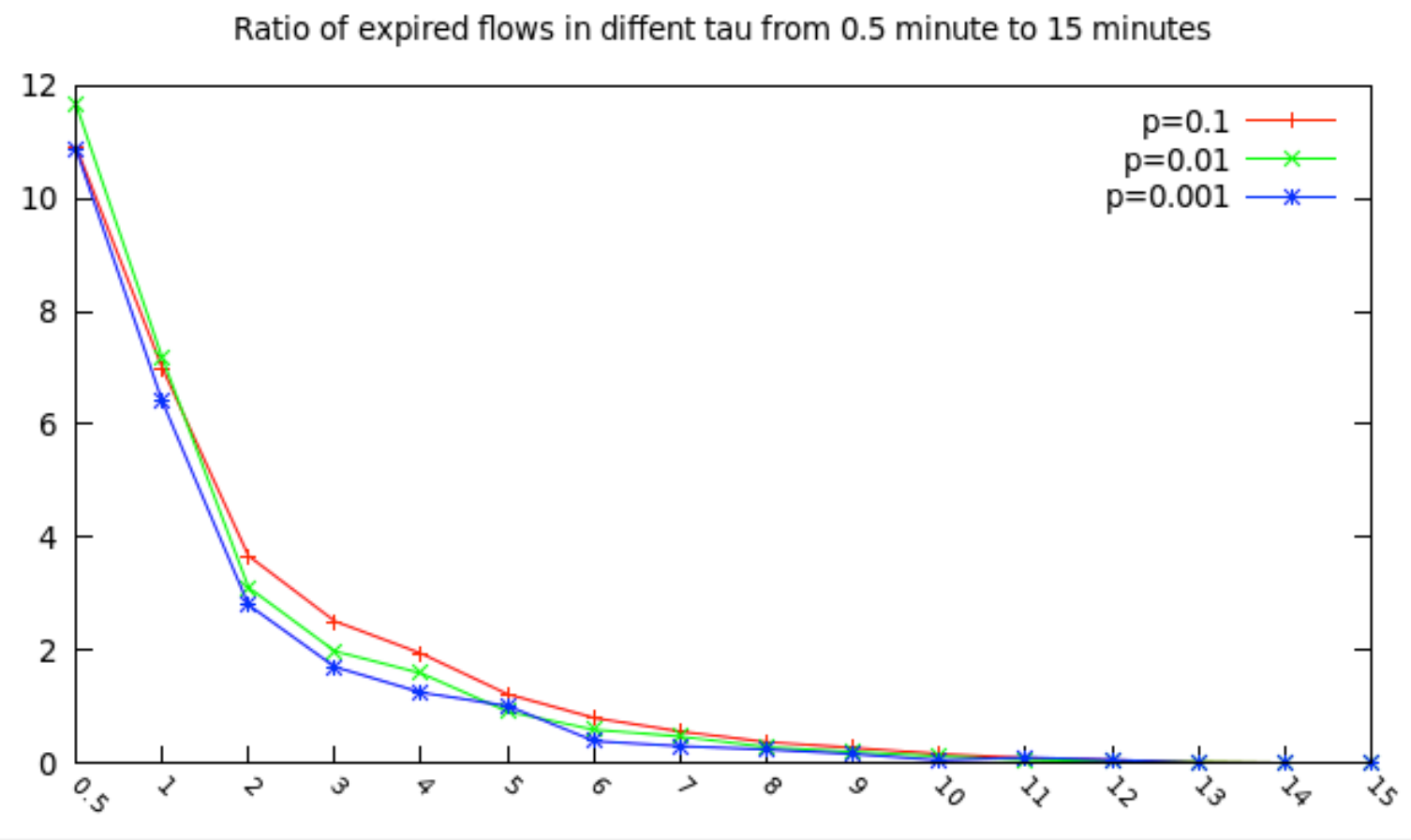



Fig. 1: Graph showing the ratios of expired flows in Tau values ranging from 0.5-15 and sampling rate p in $\{0.1, 0.01, 0.001\}$ (flows that are not chosen by sampling are ignored, y-axis value is in %)

# Analysis: flow entry limit of SDN

- Conclusion: current physical SDN switch not support well for monitoring many flows (especially in large networks)
- Approach: use virtual switch + extension
  - Open vSwitch
  - Add app modules (AppTable + SwApp)

# Flow entry limit of SDN - Approach

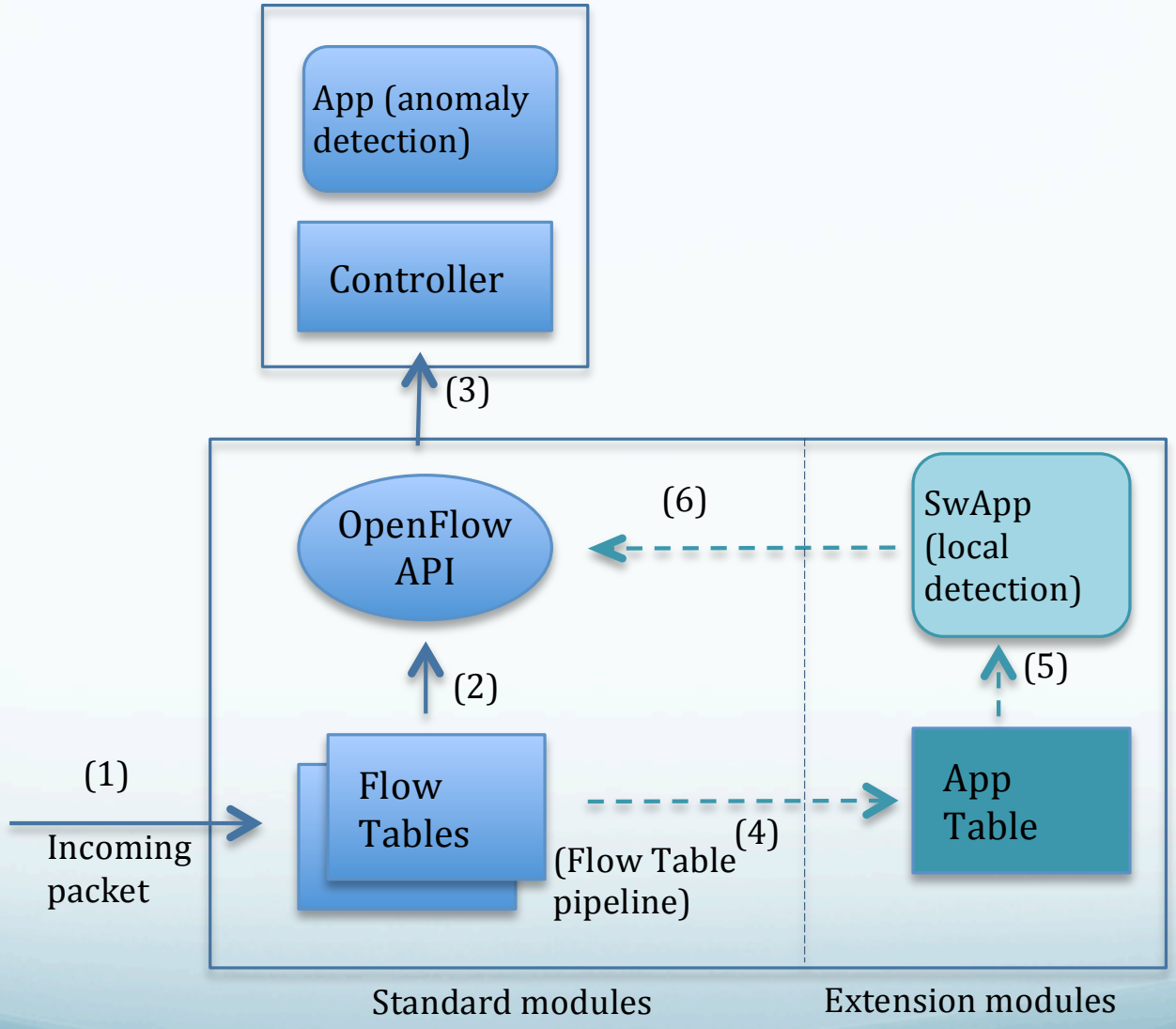
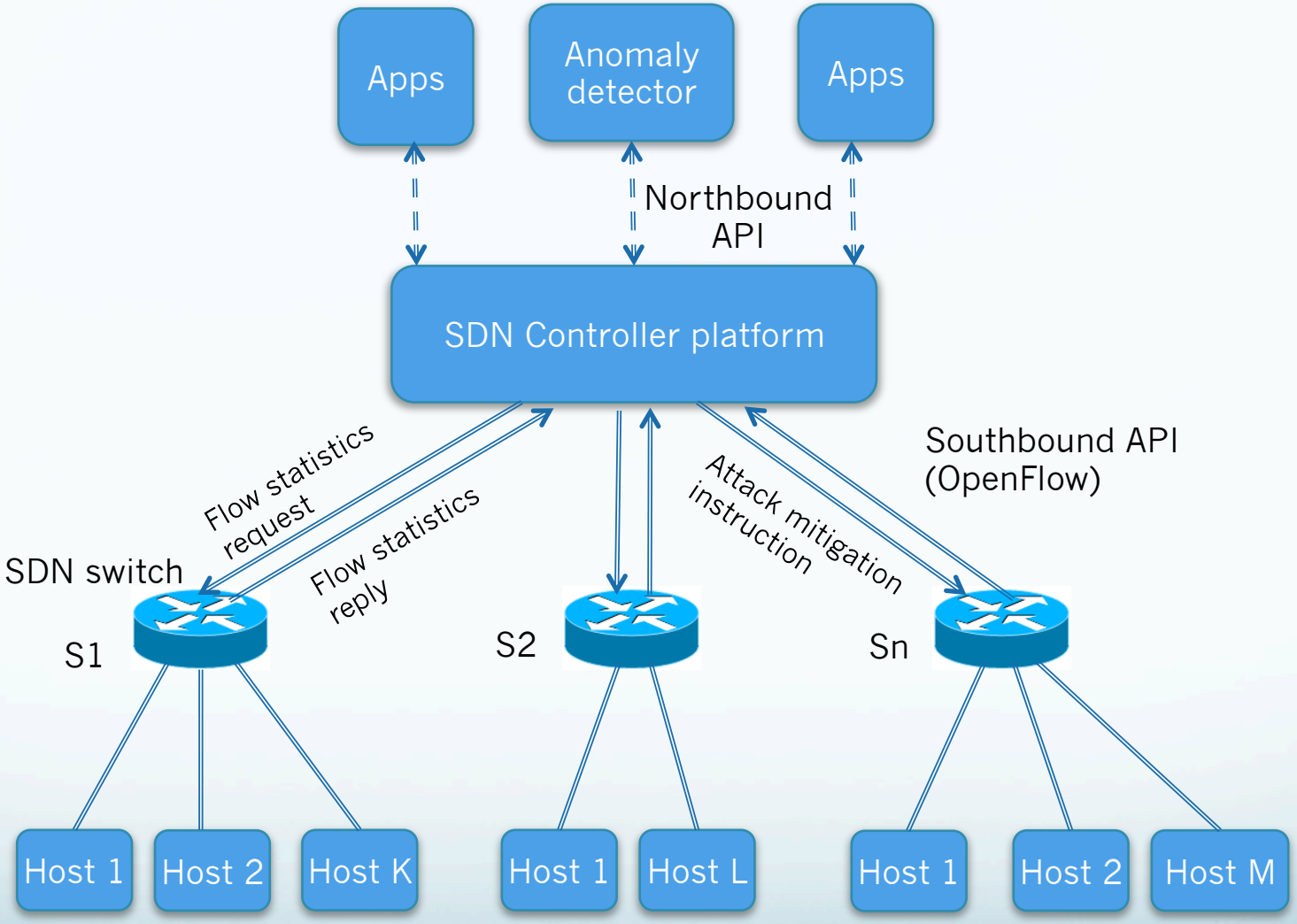



*Fig. 1: Architectural design of SDN switch extension*

# Plan for next steps

- Implementation (proposed system and detection method)
- Implement the switch extension (Open vSwitch)
- Physical deployment: set up a real SDN network (Pica8 switches)
- Simulate network attacks in a variety of scenarios for experiments
- Extend the solution:
  - Other anomaly detection method (?)
  - Extend the architecture for large-scale networks (?)
- Evaluation metrics (expected):
  - Detection time (delay time the system raise alert since the first packet of the attack arrive the network)
  - Accuracy: Detection Rate (DR), False Alarm rate (FA)
  - Effectiveness of mitigation: time for react/recover network services after attack is detected.




Anomaly detection and mitigation architecture

Thank you for your attention!

# Q&A