UMON: Flexible and Fine Grained Traffic Monitoring in Open vSwitch

An Wang, **Yang Guo**,

Fang Hao, T.V. Lakshman and Songqing Chen

NIST

Outline

- **Introduction**
- **UMON design and implementation**
- **Evaluation**
- **Summary**
- **Credit:**
  - An Wang, Songqing Chen from GMU
  - Fang Hao, T.V. Lakshman from Bell Labs

Introduction

- **Fine-grained network traffic monitoring is important for effective network management**
  - Traffic engineering, anomaly detection, network diagnosis, traffic matrix estimation, DDoS detection and mitigation, etc.
- **Scalability has been the main challenge**
  - High switching speed
  - Large number of flows
  - *Solution*: sampling, probabilistic based measurement, hardware enhanced measurement solutions, etc.
- **Open vSwitch (OVS) is a popular software switch widely employed by SDN**
  - Developed by *Nicira* as an edge switches for Data center SDN solution
  - slower switching speed, smaller #flows, access to more CPU and memory resources
  - Similar monitoring tools as hardware switches: *Netflow, sFlow, SPAN, RSPAN, flow entry counts*

Introduction

- **Recent push to use flow entry counts for traffic monitoring**

- **Challenges in flow entry counts monitoring**
  - TCAM space is limited in hardware switches
  - header fields of interest for packet forwarding may not overlap with those of interest for monitoring
  - Interaction between forwarding and monitoring is not trivial
  - May force SDN to work in reactive mode: *constant controller involvement*

- **Our Idea: leverage software switch to provide *user-defined traffic monitoring***

Introduction

- **Why software switch?**
  - Slower switching speed
  - Access to more resources (both CPU and memory)
  - Sitting at the edge
  - Open source

- **What UMON likes to achieve?**
  - Monitor arbitrary fields
  - Sub-flow monitoring, e.g., monitor micro/sub-flows of a mega-flow, without constant controller involvement
  - Allow to push other management functions, such as anomaly detection, to the switches

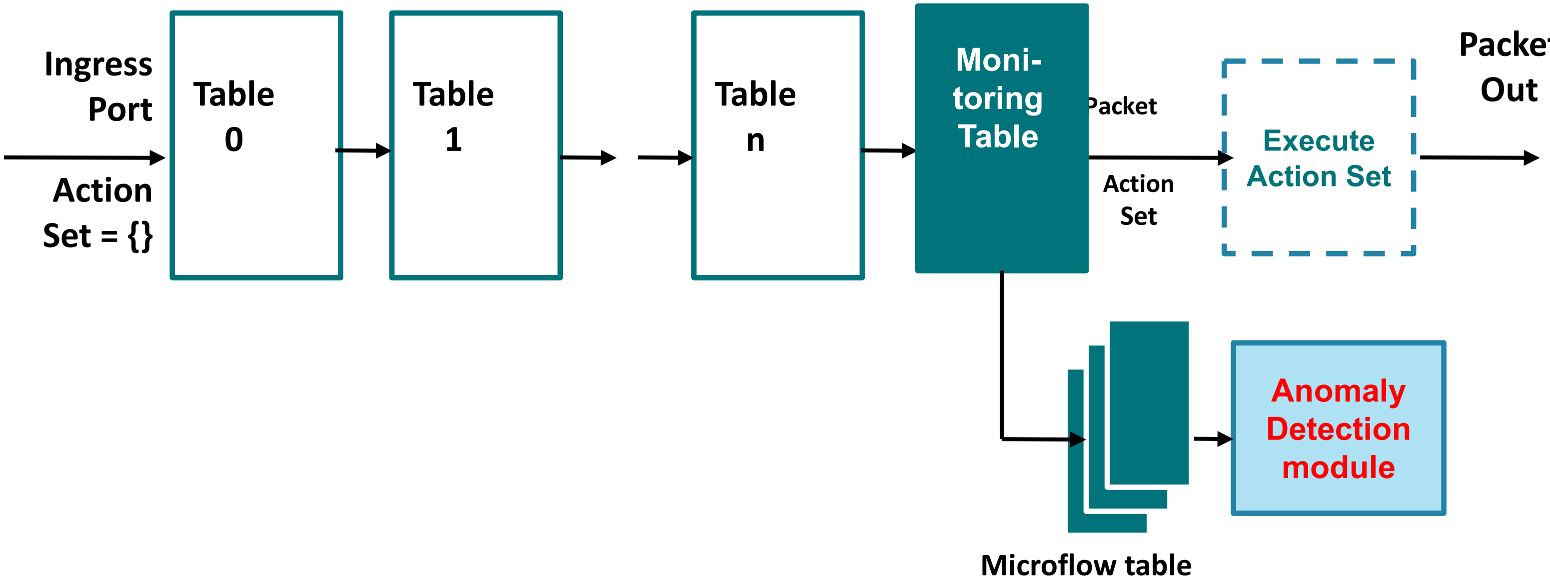
UMON: Design and Implementation

- **How to instrument the software switch to support UMON?**
  - Decoupling monitoring from forwarding
  - Monitoring does not interfere with forwarding

- **Design must integrate well with the OVS architecture**
  - Two-tiered forwarding architecture
    - User-level: full blown pipelined routing
    - Kernel-level: flow entry caching

UMON: Design and Implementation

- **User level decoupling**
  - a separate monitoring flow table, where the monitoring rules are stored



UMON: Design and Implementation

- **Kennel level decoupling**
  - Kernel rule does not support priority
  - For a packet, at most one rule matches the header
  - Adding a monitoring table in kernel is ‘heavy’
  - carefully designed kernel flow rules that satisfy the monitoring requirements
    - *Kernel rule must be ‘finer’ than the monitoring rule*

- **Let $(\boldsymbol{r_f}, \boldsymbol{m_f})$ be the generated kernel flow rule and its mask; $(\boldsymbol{r_i}, \boldsymbol{m_i})$, $\boldsymbol{i} \in \boldsymbol{I}$, be the monitoring rule set in the monitoring table**

- $\boldsymbol{m_f^*} \triangleq \boldsymbol{m_f} \mid \left(\mid_{i \in I_f} \boldsymbol{m_i} \right)$,

  where

  $\boldsymbol{I_f} \triangleq \{\boldsymbol{i} \mid \boldsymbol{r_f} \,\&\, \boldsymbol{m_{fi}} = \boldsymbol{r_i} \,\&\, \boldsymbol{m_{fi}}, \boldsymbol{i} \in \boldsymbol{I}\}$, $\boldsymbol{m_{fi}} \triangleq \boldsymbol{m_f} \,\&\, \boldsymbol{m_i}$.

UMON: Design and Implementation

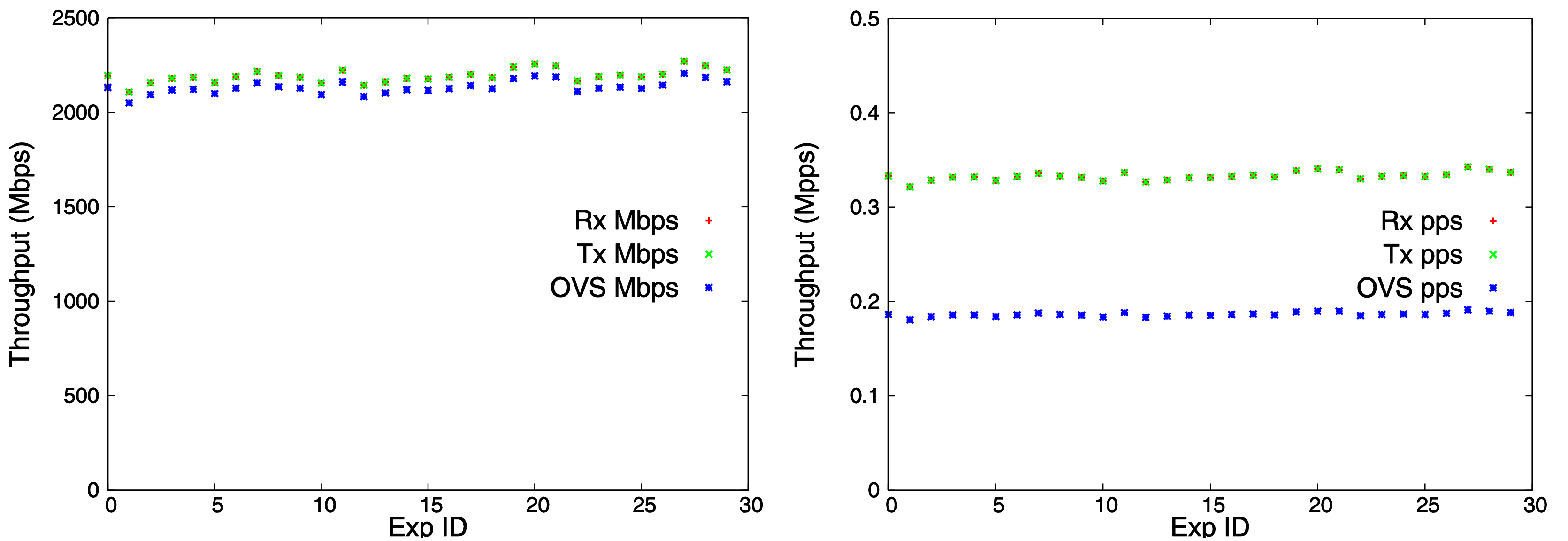
- **Traffic monitoring of non-routing fields**
  - New monitoring actions to collect stats of non-routing fields
  - E.g. *SYN Monitoring Action*, *ACK Monitoring Action*, etc.
- **Sub-flow monitoring**
  - Sub-flows are the fine-grained flows that belong to a mega-flow as defined by the monitoring rule
  - Sub-flow is defined by sub-flow mask s_i
  - generate proper kernel flow rules
- **Monitoring rule insertion/deletion**
  - When removing a monitoring rule => ‘lazy’ approach
  - When a monitoring rule is added => ‘complex’
    - make sure the kernel rule’s granularity is still fine
    - If not, purge the rules. Proper rule will be added when next packet arrives

Evaluation

- **Setting:**
  - Open vSwitch (version 2.3)
  - A standalone machine with 2.67GHz CPU (12 cores), 64G memory, and an Intel NIC of two 10G ports
  - One server, one client
  - Compare performance of UMON, default OVS, and micro-flow enabled OVS

Evaluation

- **UMON overhead evaluation**
  - DECONF trace with 272 hosts and 4432 micro-flows
  - Monitor 150 hosts with micro-flow monitoring on
  - Transmit at 2.2 Gbps



'Gap' is due to Generic Receive Offload option (GRO) at NIC

Evaluation

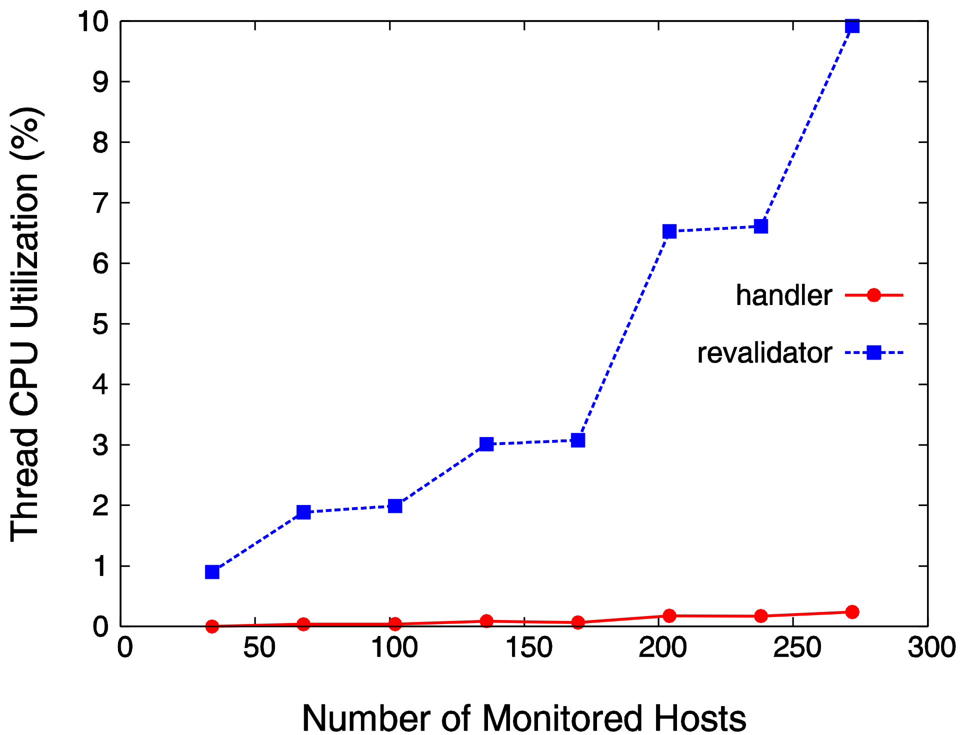
- **UMON overhead evaluation**

	Handler	Revalidator	FlowTableSize	MissPktRate
OVS	0.0%	0.60%	295	0
Microflow OVS	0.15%	6.8%	4381	30
UMON	0.21%	9.9%	4301	26

- **CPU utilizations are low for all three types of vSwitches**
- **Revalidator threads consume much more CPU resources than the handler threads due to large flow table size and monitoring activity**

Evaluation

- **Effect of monitoring rules**



Tradeoff between #monitoring-rules, kernel flow table size, and CPU utilization is possible

Conclusions and Future Work

- **UMON: decouples monitoring from forwarding, and offers flexible and fine-grained monitoring in OVS**
- **Design and implement UMON**
- **Evaluate the prototype**

- ***Design and specify OpenFlow interface for UMON***
- ***Distributed UMON monitoring network for DDoS detection***

Backup slides

Evaluation

- **Effect of monitoring rules**