Challenges and Solutions for a Flexible High-Performant SDN Hypervisor

Nemanja Đerić
nemanja.deric@tum.de

Prof. Wolfgang Kellerer
wolfgang.kellerer@tum.de
Overview

- Introducing SDN hypervisor
- Main Hypervisor Function
- Control Plane Resource Management
  - Resource Management in HyperFLEX
  - Motivation
  - Plans & Goals
- Resolving Switch Diversity
  - Motivation
  - Problems
Introducing SDN hypervisor

- Provide SDN network as a on-demand service (NaaS)
  - Improves utilization and reduces the overall cost
- Every Tenant could use own controller and applications
Introducing SDN hypervisor

- Provide SDN network as a on-demand service (NaaS)
  - Improves utilization and reduces the overall cost
- Every Tenant could use own controller and applications
Hypervisor Functions & Shortcomings

- Three Main **Functions**: 
  - Control & Data Plane Isolation
  - Translation
  - Abstraction
Hypervisor Functions & Shortcomings

- Three Main Functions:
  - Control & Data Plane Isolation
  - Translation
  - Abstraction
Control Plane Resource Management - Motivation

- **SDN hypervisor** resources are shared between each tenant
  - e.g. CPU, RAM
Control Plane Resource Management - Motivation

- **SDN hypervisor** resources are shared between each tenant
  - e.g. CPU, RAM
- One vSDN might **over-utilize** the hypervisor resources leading to **cross-effects** between vSDNs
Control Plane Resource Management - Motivation

- **SDN hypervisor** resources are shared between each tenant
  - e.g. CPU, RAM
- One vSDN might **over-utilize** the hypervisor resources leading to **cross-effects** between vSDNs

- Admission control and resource isolation guarantees are necessary!
Control Plane Resource Management

- HyperFLEX – SDN Virtualization Tool developed at LKN [1]
- Introduces **admission control** based on the available hypervisor resources (e.g. CPU)
  - Hypervisor resources are estimated based on the offline mapping between CPU consumption and number of OF messages
HyperFLEX – SDN Virtualization Tool developed at LKN [1]

Introduces **admission control** based on the available hypervisor resources (e.g. CPU)

- Hypervisor resources are estimated based on the offline mapping between CPU consumption and number of OF messages

Online resource estimation is extended in [2]

- Machine Learning is used to learn and fit the exponential mapping between number of OF messages and CPU consumption

---

What is missing?

- **Only** Simple SDN network scenarios are considered!
  - Number of OF messages per each vSDN ✓
  - Type of OF messages ✓
  - Isolation and abstraction not considered! ❌
What is missing?

- **Only** Simple SDN network scenarios are considered!
  - Number of OF messages per each vSDN ✓
  - Type of OF messages ✓
  - Isolation and abstraction not considered! ✗

- Does control plane isolation affect resource utilization?
  - Isolation function has to process and count every transitioning packet!
What is missing?

- **Only** Simple SDN network scenarios are considered!
  - Number of OF messages per each vSDN ✓
  - Type of OF messages ✓
  - Isolation and abstraction not considered! ✗

- Does control plane isolation affect resource utilization?
  - Isolation function has to process and count every transitioning packet!

- Does complexity of abstraction of network topologies matter?
  - Tasks for *big-switch* abstraction:
    - Provide corresponding *mapping*
    - Rewrite all messages on the control plane
    - Establish routing in the *big-switch*
Resource estimation – Problems & Goals

1. Explore and study effects of complex network scenarios on hypervisor performance and resource consumption

2. Develop appropriate models for mapping between different hypervisor functions and resource consumption
Resource estimation – Problems & Goals

1. Explore and study effects of complex network scenarios on hypervisor performance and resource consumption

2. Develop appropriate models for mapping between different hypervisor functions and resource consumption

3. Extend it with online solutions as they:
   • Don’t need extensive benchmarks
   • Support live hypervisor migrations (e.g. due to failure or lack of resources)

4. Integration in HyperFLEX framework
Resolving Switch Diversity - Motivation

- **Table Size Difference [3]**

<table>
<thead>
<tr>
<th>Switch</th>
<th>Table Size [Approx]</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICA 3290</td>
<td>2000</td>
</tr>
<tr>
<td>DELL S3048-ON</td>
<td>500</td>
</tr>
<tr>
<td>PICA 3297</td>
<td>3500</td>
</tr>
</tbody>
</table>

- **Flow Mod Update Rate [4]**

Resolving Switch Diversity - Problems

1. How to gather data from switches?
   - What resources are the most influential (e.g. Flow Table Size, Update Rates)?
   - How to automate benchmarking and automation?

2. How to do embed vSDN based on the switch diversity?

3. How to add rules on run-time based on switch diversity?
References


Questions?