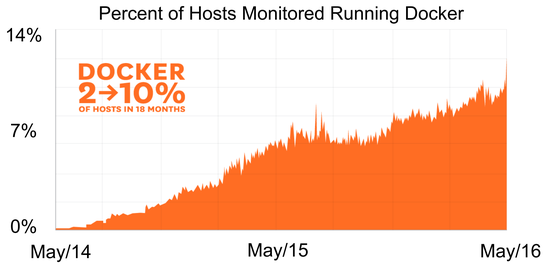
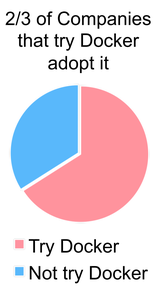
cMonitor: Monitoring the Network of Large-scale Container Deployments in Datacenters

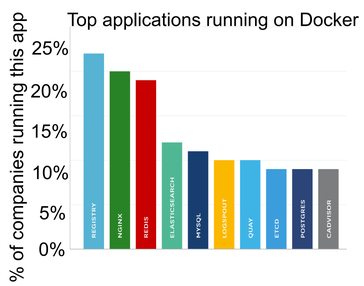
Kun Suo and Jia Rao
Department of Computer Science and Engineering at
The University of Texas at Arlington

Need and Industrial Relevance

- The adoption of containers sees explosive growth recently. 2 billion downloads by 2016.
- Advantages: rapid application deployment, portability across machines, lightweight footprint and minimal overhead, version control, simplified maintenance.







- Monitoring the network of large-scale container deployments in Datacenters has its unique challenges:
  - Containers deployment 100X denser than VM deployment
  - Highly dynamic deployment and complex inter container traffic
  - Coexistence with existing IaaS Clouds and cloud networking
  - A large number of isolated application-defined networks

Duration and Budget

- Duration:
  - 12 months
- Estimated Cost:
  - $30k Labor
  - $5k Supplies
  - $5k Others(including benefits and travel
  - $4k Overhead(10%)
- Total: $44k

Project Goals

- Our container monitoring system tries to achieve the following goals:
  - **Lightweight**: High performance data plane monitoring infrastructure
  - **Dynamic**: Real-time monitoring for fast and dynamic application deployment
  - **Transparency**: No intrusive changes to applications or the cloud infrastructure
  - **Scalability**: Scaling to monitoring the network of hundreds of containers per node
  - **Customization**: Network templates with standardized network components for various network environments

Deliverables

- **Container Monitoring System**: tracking network metrics (e.g., throughput, latency, packet loss rate) among containers
- **Data visualization**: Real-time display of various information on container network through a user-friendly graphical interface.
- **Open source**: Source code release on Github

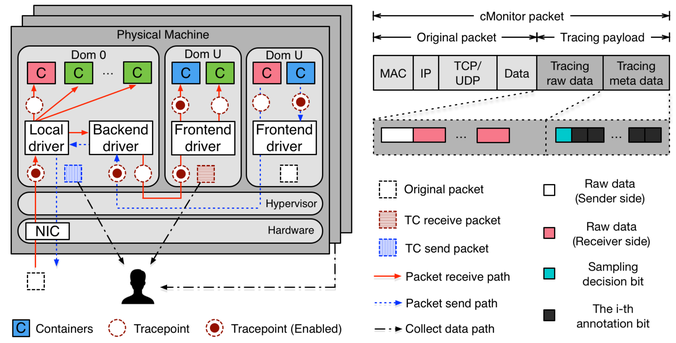
Objectives

- Develop a lightweight, real-time, application transparent container monitoring system to track large scale container networks
- The users can customize the tool based on their specific purposes
- Apply in real production container environment to monitor the container network performance and locate potential issues

Impact

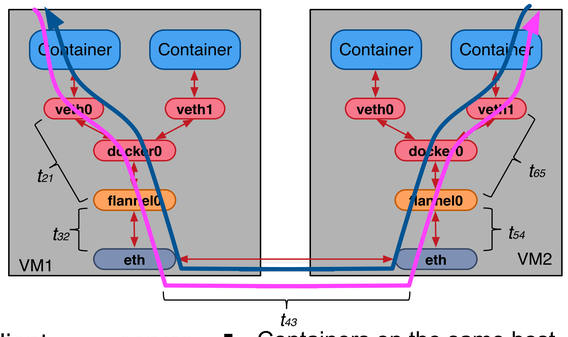
- To our best of knowledge, there are no fine-grained, non-intrusive container network monitoring tools across different layers in software-defined networks.
- With an annual 30% growth, software-defined container network will be the dominant network technology in datacenters soon.
- There lacks a comprehensive study of the performance of virtualized container network. Our proposed monitoring tool will help identify performance bugs, design deficiencies, and other possible issues.

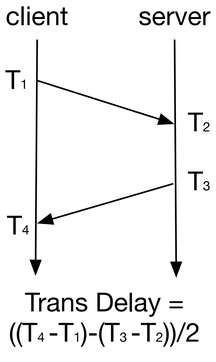
Proposed Approaches



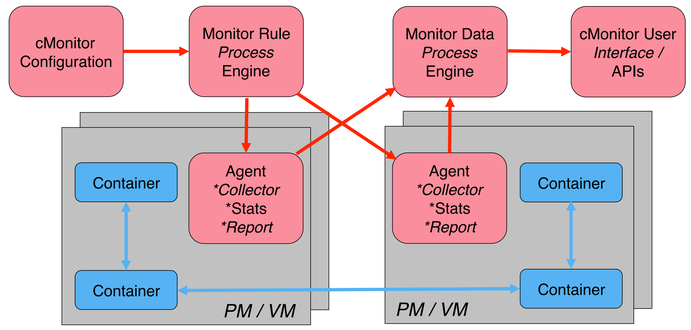
- **Basic Idea:** Embedding tracing information into network packet payload
- cMonitor places tracepoints throughout the virtualized network stack and timestamps packets at enabled tracepoints.
- The timing information is appended to the payload of the packet
- The tracing payload is removed and dumped to a kernel buffer before a packet is copied to the application buffer in user space or sent out from the physical NIC

Example: Latency Monitoring

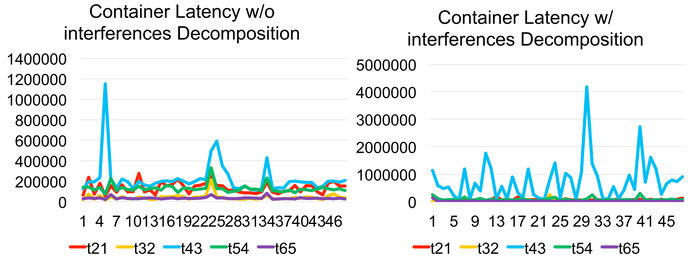




- Containers on the same host
  - Bind eBPF scripts on the network devices/path (e.g., vport, docker, flannel, Linux network stack, etc.)
- Containers on the different hosts
  - Different VMs on the same physical machines → shared TSC values
  - Different VMs on different physical machines → Round Trip Time / 2



- **Basic Idea:** Using IOvisor / eBPF scripts to dynamically trace packet processing at various places throughout the network
- Monitor Rule Process Engine
  - Receive configurations from users
  - Generate user configuration to standard rules
  - Send standard rules to cMonitor agent in each VM / PM
- Tracing agent
  - Implemented based on IOVisor (eBPF)
  - Collect network information based on received rules
  - Send collected data to data process engine periodically
- Monitor Data Process Engine
  - Receive data from all agents
  - Process the raw data and write them to local influxDB
- User Interface
  - Receive data from all agents
  - Process the raw data and write them to local influxDB



Future Work

- Extend to monitor more QoS metrics of containers : CPU utilization, Memory usage, Disk performance, etc.
- Evaluate network performance of representative workloads, including Memcached, Spark, and Hadoop.
- Explore monitoring container network under network function virtualization (NFV).

Contact Information:
Kun Suo kun.suo@mavs.uta.edu
Prof. Jia Rao jia.rao@uta.edu