June 22, 2018

# Techniques for Monitoring and Measuring Virtualized Networks

Aman Shaikh & Vijay Gopalakrishnan

AT&T Labs - Research



AT&T

# Outline

## Introduction and Background

- Service-provider and datacenter networks
- Introduction to Network Function Virtualization (NFV)
  - Software Defined Networking (SDN)

## Challenges

- General challenges in network measurements
- Challenges specific to NFV

## Measurement and monitoring

- What and how of data collection
- Data collection and tools in today's cloud
- Measurement applications: trouble-shooting, performance improvement, verification
- Leveraging SDN and end-hosts for measurements

## Summary

- Additional Resources

AT&T

# Introduction and Background

Why the move towards SDN and NFV

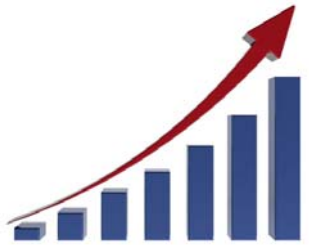


AT&T

# Network Traffic Trends*

## Unprecedented growth in data traffic

- Ten-fold increase in traffic volume b/w 2014 and 2019



## Large number of diverse devices

- Devices becoming mobile
- 11.5 billion devices, including 3B IoT devices by 2019


## Multitude of Demanding applications

- ¾ of the data traffic will be video by 2019

## Growth of Cloud Computing

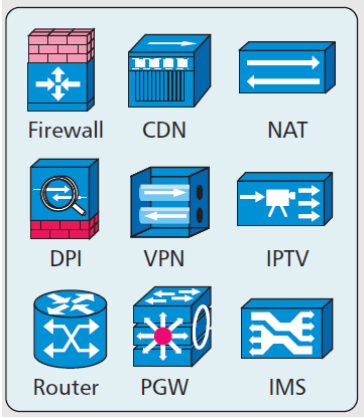
- Variable demand, transient applications

*Source: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014-2019

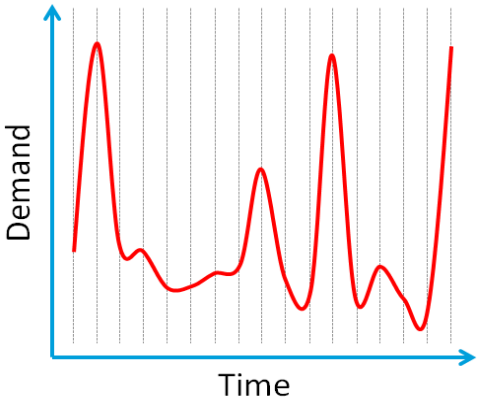
AT&T

# Traditional Networking ≠ Technology trends

## Introducing, modifying, or scaling networking services is difficult




Proprietary, fixed function, and hardware nature of network appliances

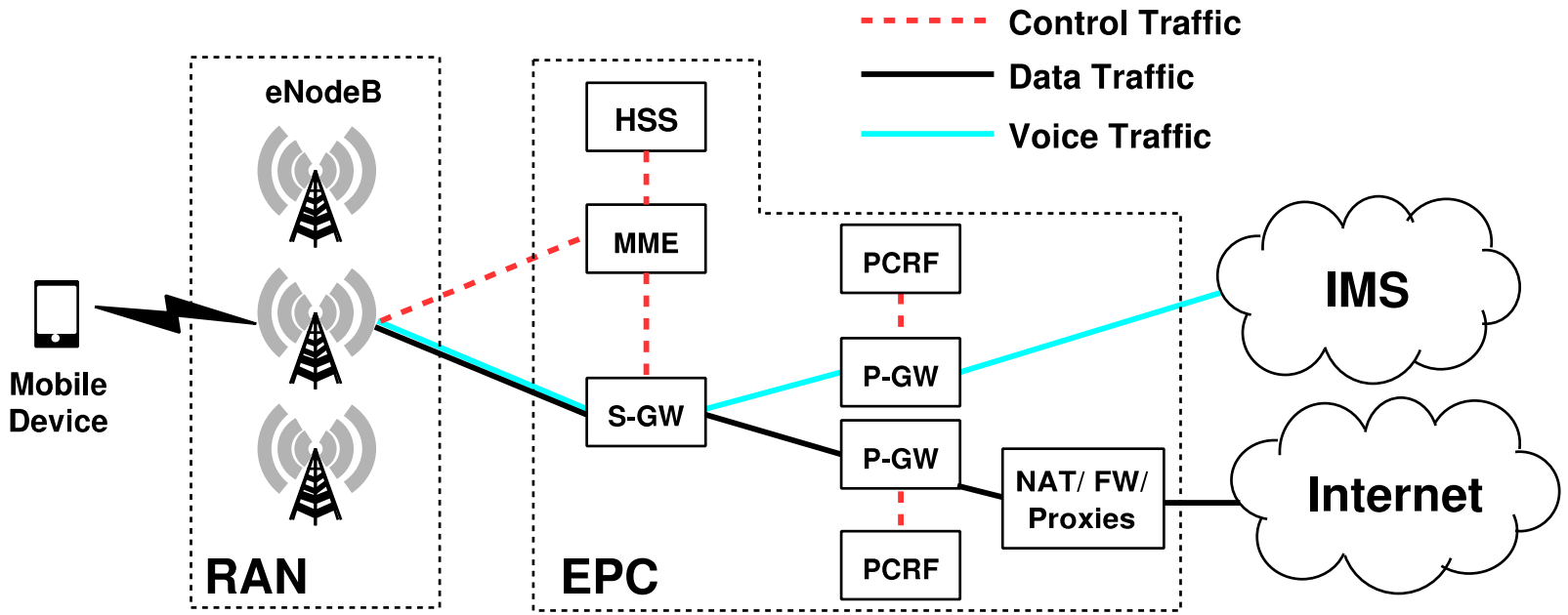



High spatial and temporal variability in “demand”, especially closer to the edge

The cost of reserving the space and energy for a variety of network equipment

The cost and lack of skilled professionals to integrate and maintain network services

# Concrete Example: Cellular Network Architecture



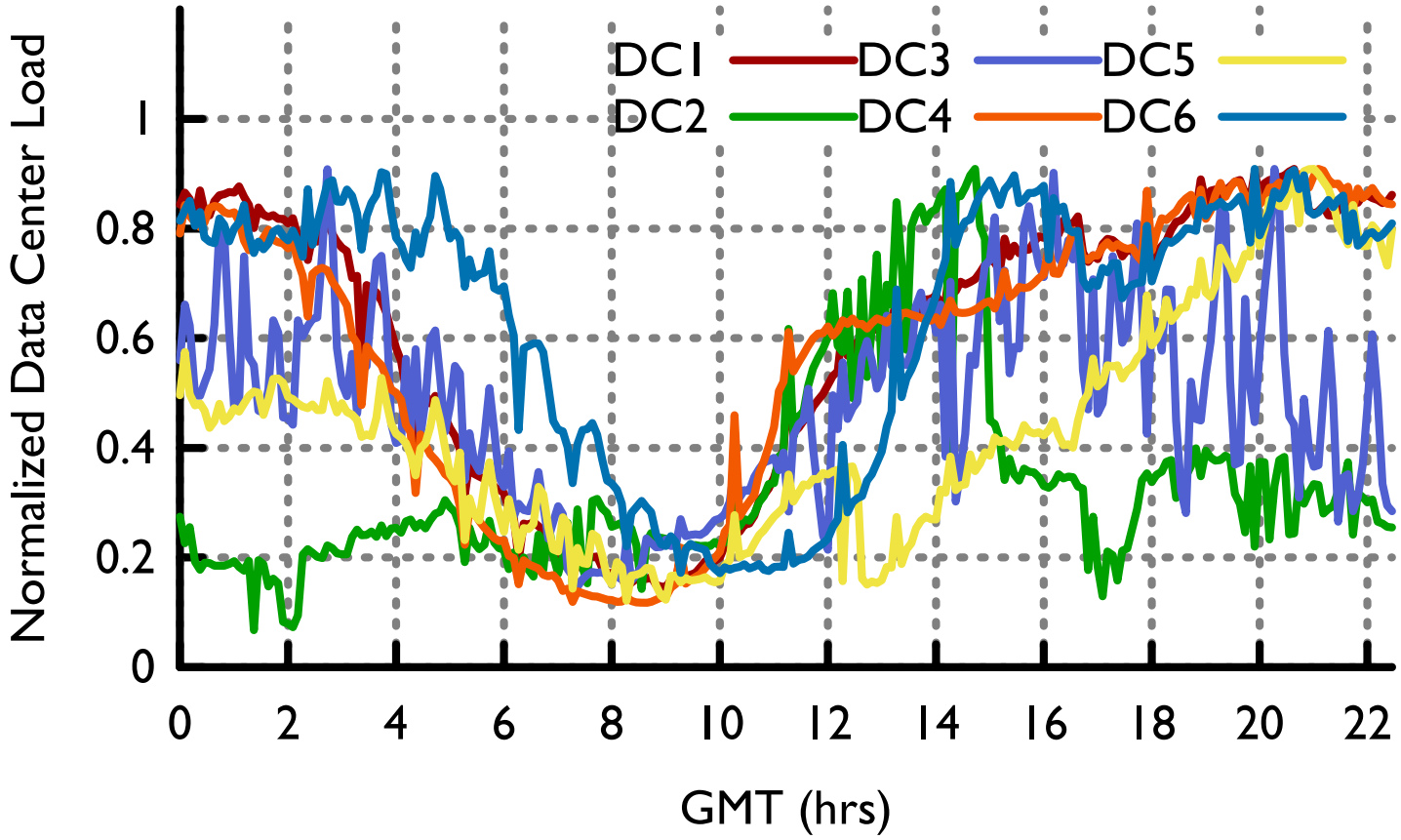

- Hardware appliances supporting specific cellular network functions
  - specialized, expensive

- Statically provisioned at a few central locations
  - hard to predict load at edge, cannot repurpose

# Problems with Traditional deployments

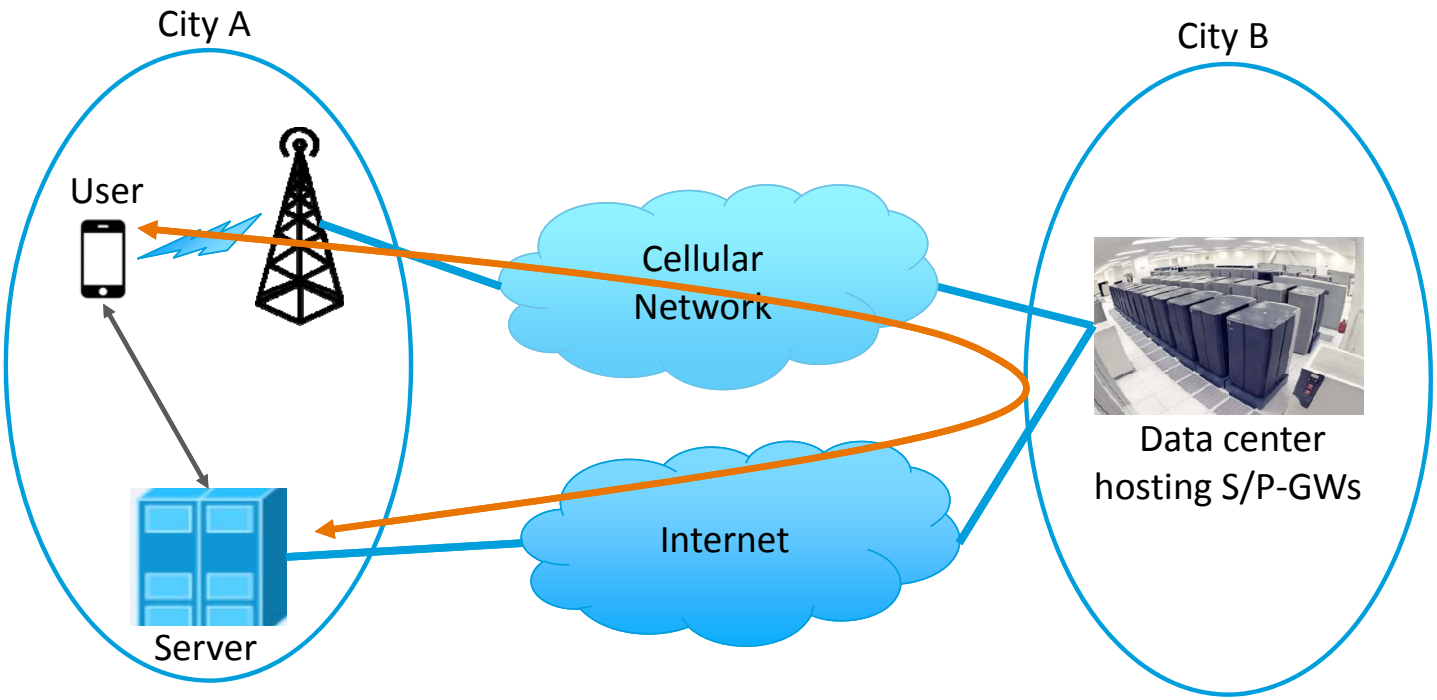
| | |
|---|---|
| Inefficient load distribution | Path Inflation |
| Performance Impacts | Limited Customization |

# Problems with Traditional deployments

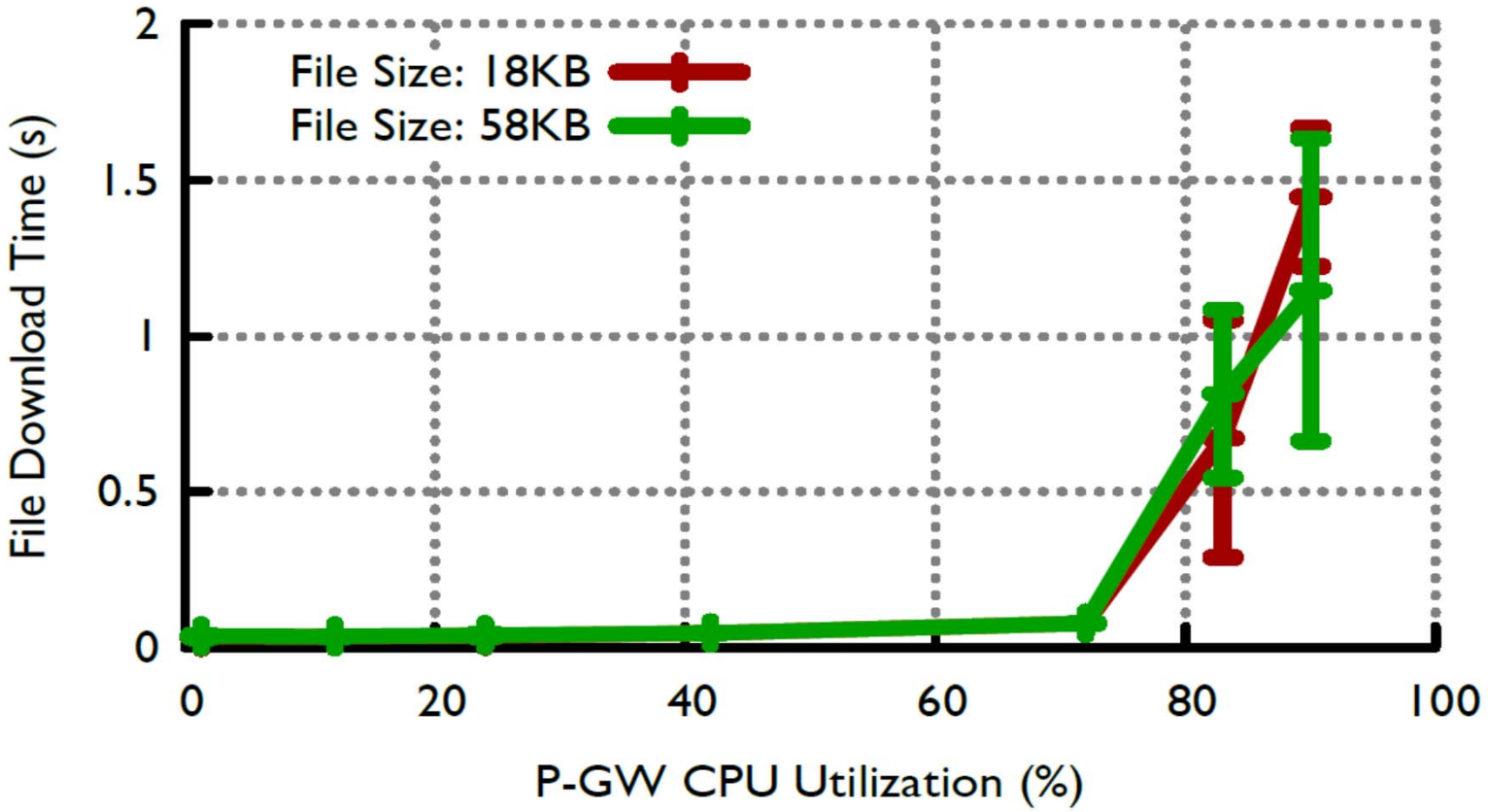



AT&T

# Problems with Traditional deployments



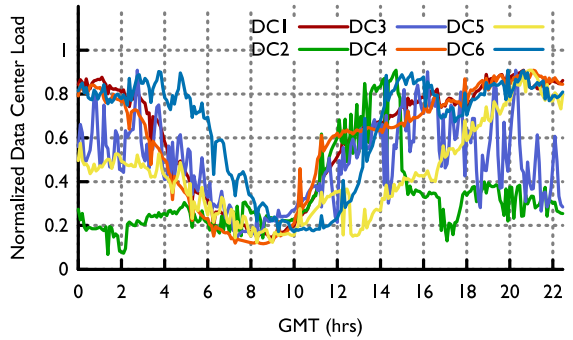
# Problems with Traditional deployments



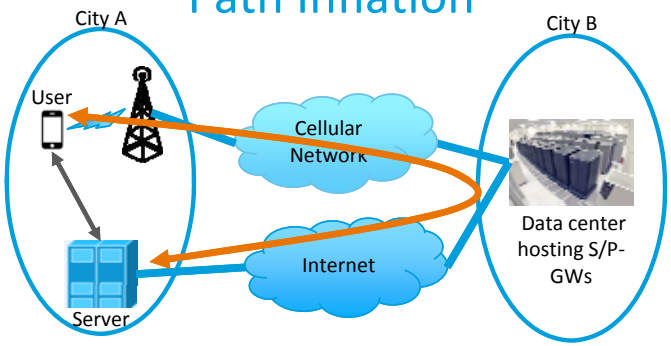

AT&T

# Problems with Traditional deployments

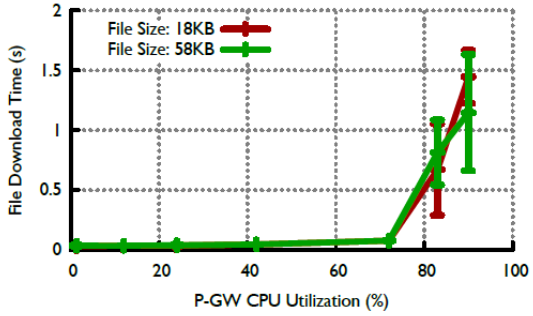
## Inefficient load distribution




## Path Inflation




## Performance Impacts




## Limited Customization

**Diverse Devices**


Smartphone


Fleet tracking


Smart meter

**Diverse applications**

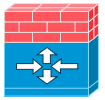
GAMING

**Diverse NFs**

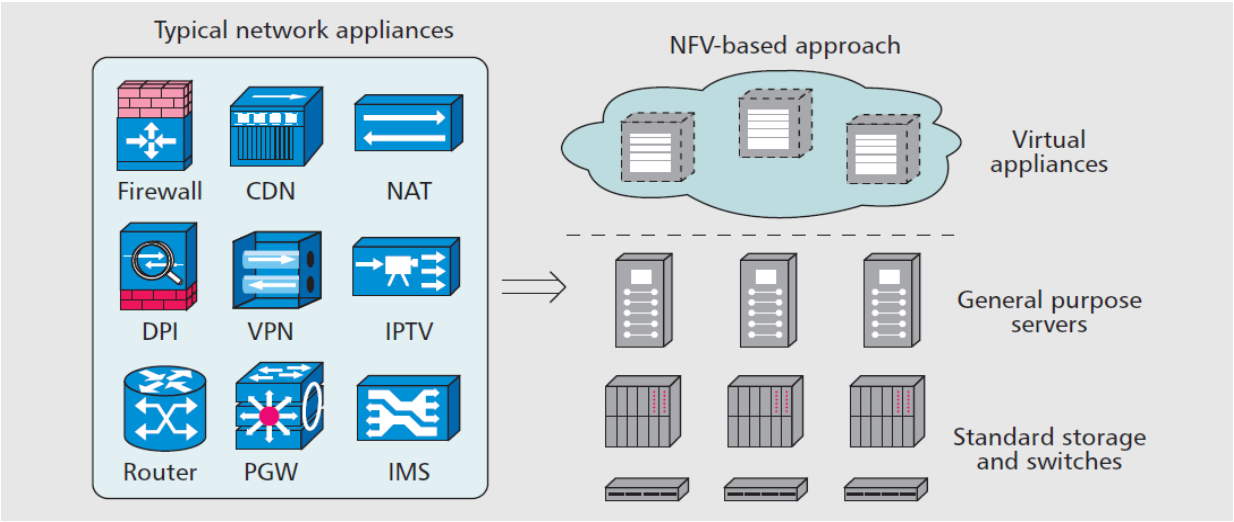
Transcoder

IDS

Firewall

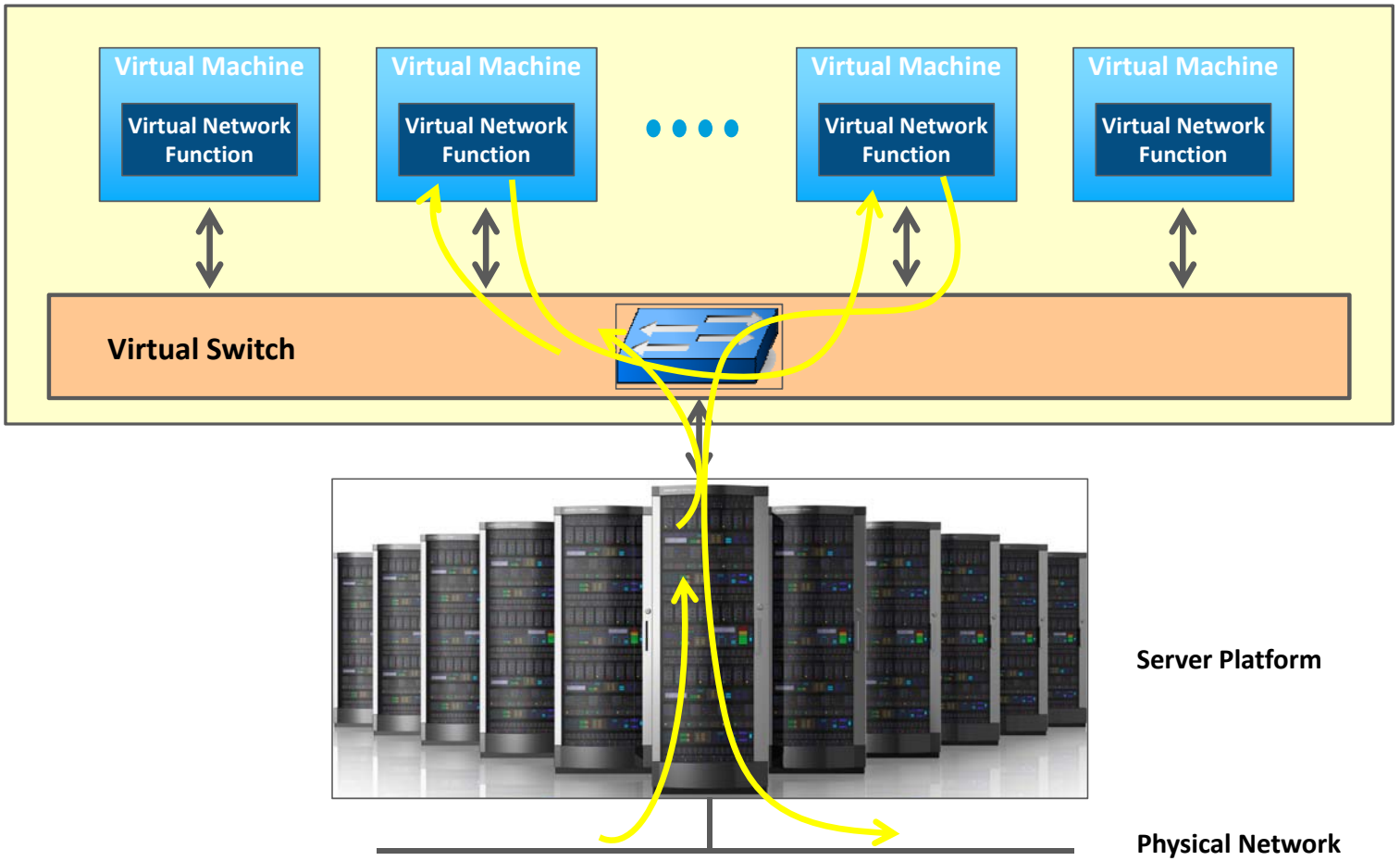


# Moving towards SDN and NFV




- Take advantage of advances in Cloud technology
- Network elements are applications running on a common hardware platform
- Reduce capital and energy expenses by consolidating network elements
- Software as a driver to change the innovation cycle
- Reduce time to market customer targeted and tailored services

# How does it work?



# Key NFV technology

Strengths and Weaknesses

# NFV Design Principles

## Separation of software from hardware

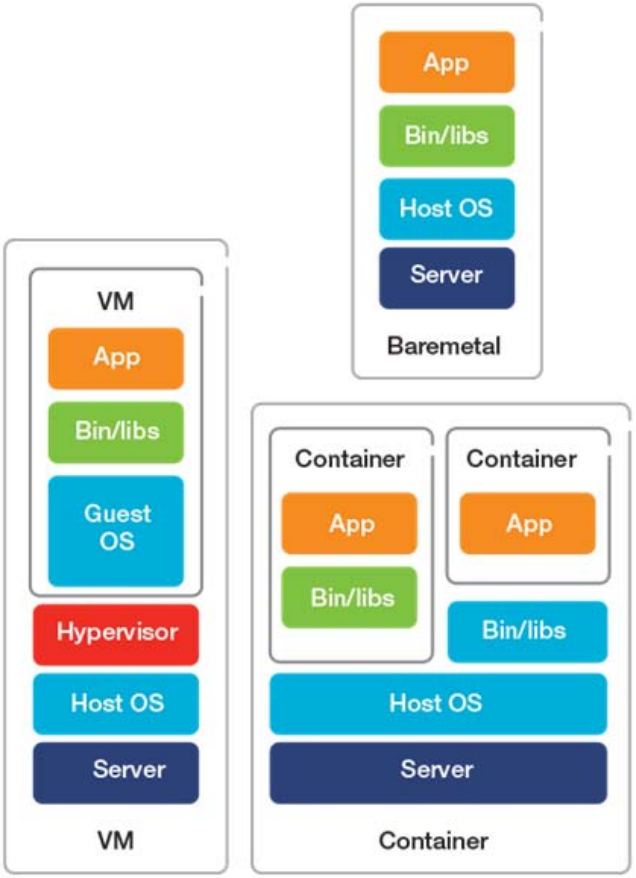
- Enable software to evolve independent from hardware, and vice versa

## Flexible deployment of network functions

- Deploy automatically network-function software on a pool of hardware resources
- Run different functions at different times in different data centers

## Dynamic service provisioning

- Scale the NFV performance dynamically and on a grow-as-you-need basis
- Fine granularity control based on the current network conditions



# Technical Requirements

## Performance

- Keep the degradation as small as possible
- Understand the maximum achievable performance of the underlying programmable hardware

## Manageability

- Different from data center networking, where the hardware resources are almost equivalent
- Support sharing spare resources and elastic provisioning of network services effectively

## Reliability and Stability

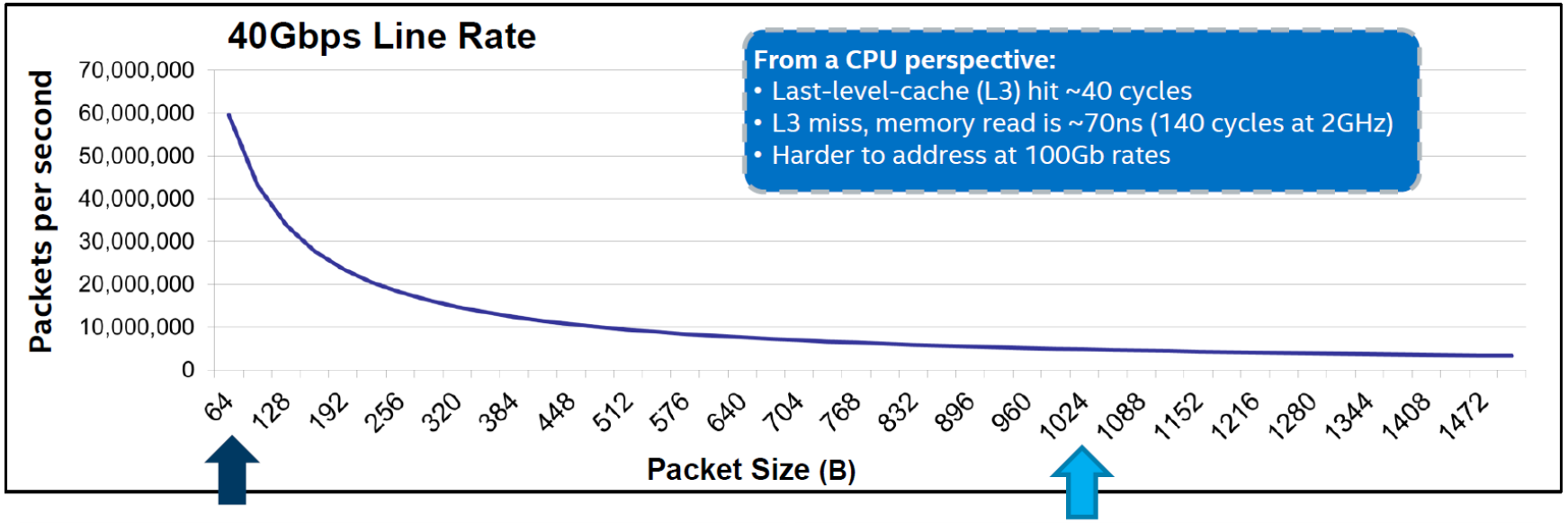
- Minimize service reliability and service level agreement impacts when evolving to NFV
- Ensure service stability when reconfiguring or relocating software-based virtual appliances

## Security

- Share underlying networking and storage, run in other's data center, outsource to 3rd party
- Introduction of new elements, such as orchestrators and hypervisors

AT&T

# Problem with Software-based Packet Processing




| Packet Size | 64 bytes |
|---|---|
| 40G Packets/second | 59.5 Million each way |
| Packet arrival rate | 16.8 ns |
| 2 GHz Clock cycles | 33 cycles |

Network Infrastructure Packet Sizes

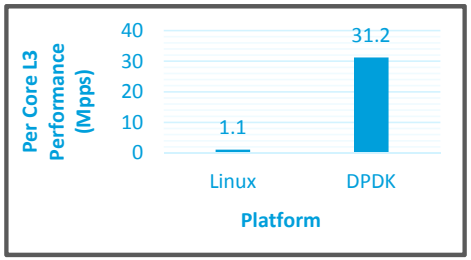
| Packet Size | 1024 bytes |
|---|---|
| 40G Packets/second | 4.8 Million each way |
| Packet arrival rate | 208.8 ns |
| 2 GHz Clock cycles | 417 cycles |

Typical Server Packet Sizes

AT&T

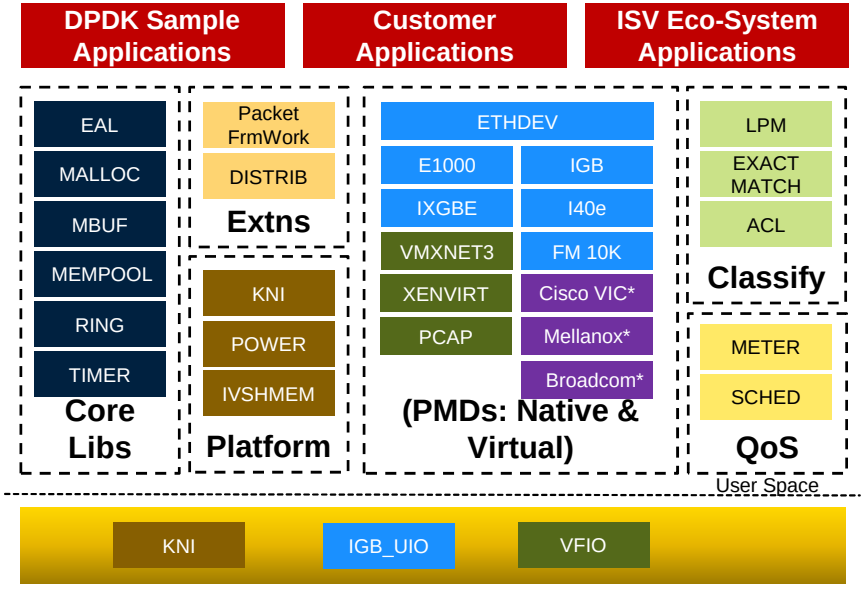
# Enter DPDK (Data Plane Development Kit)

## User-space software libraries for accelerating packet processing workloads on Commercial Off The Shelf (COTS) hardware platform




## High-performance, community-driven solution

- Delivers 25X performance jump over Linux
- Open Source, BSD License; Vibrant community support and adoption;
- Comprehensive NFV and Intel architecture support
- User space application development; multiple models
- Over two-dozen pre-built sample applications




## Host of other techniques to improve performance

- Software prefetch, Core-thread affinity, vector instructions, Algorithmic optimizations, Hardware offload, Bulk Functions



AT&T

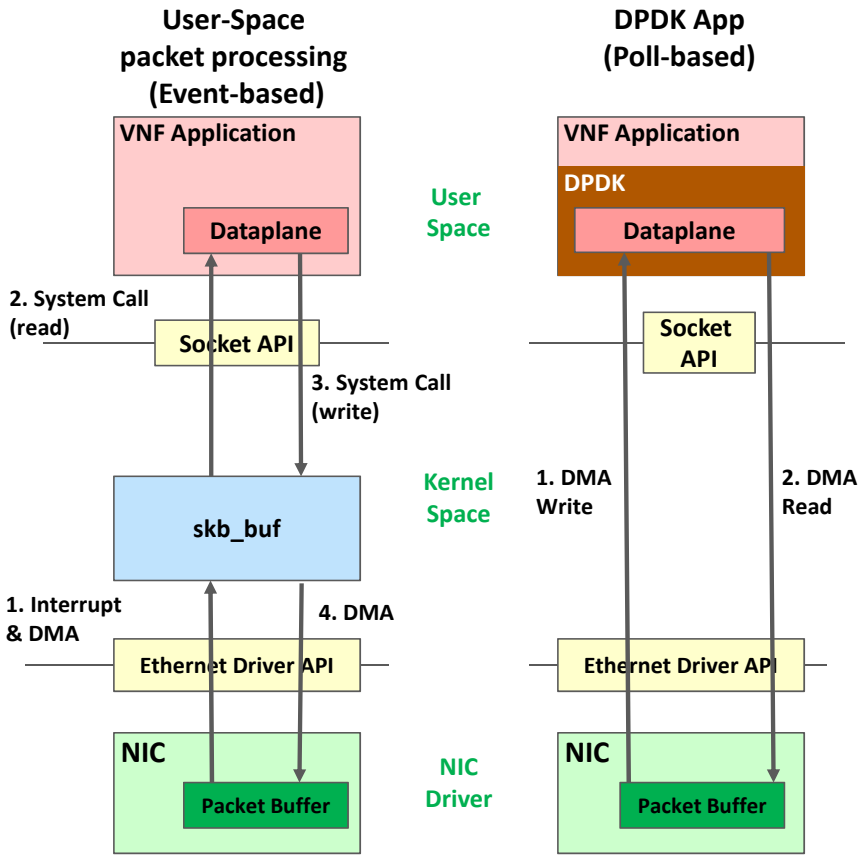
# How does DPDK work?

## Driver constantly polls the NIC for packet ➔ core is pegged at 100% use

- Takes advantage of multiple cores on modern CPU
- Not all cores used for computation; Dedicate a (few) core(s) for packet processing
- Gets around limitations with interrupt driven packet processing (used by traditional NIC drivers)

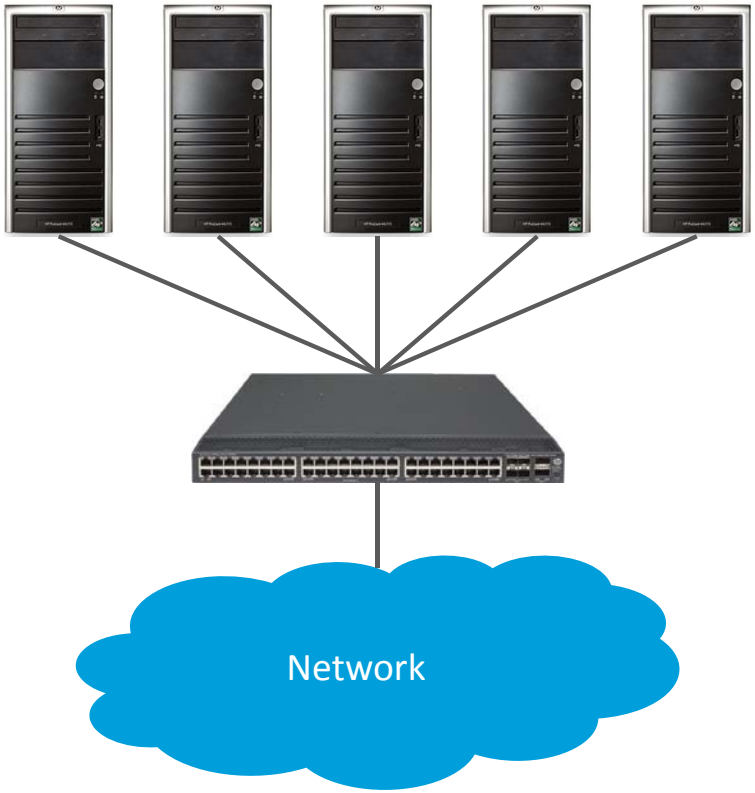
## Provides memory management libraries for efficient packet copy

- Allocate large chunk of memory in application mapped to DPDK
- Raw packets copied directly into application memory

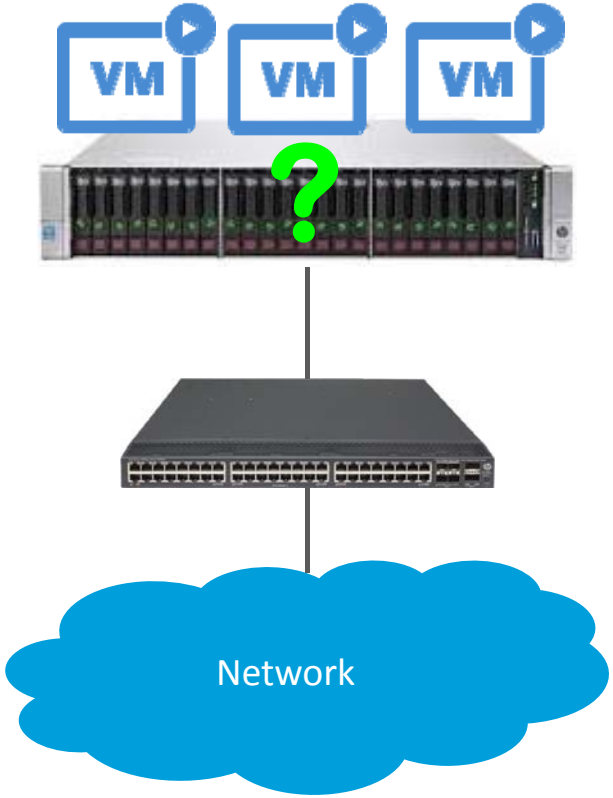


# Need for Interface Sharing

**Physical Servers**




**Virtualized Servers**



# Software-Based Sharing

## Utilizes emulation techniques to provide a logical I/O hardware device to the VM

- Interposes itself between the driver running in the guest OS and the underlying hardware (via emulation or split-driver)

## Features

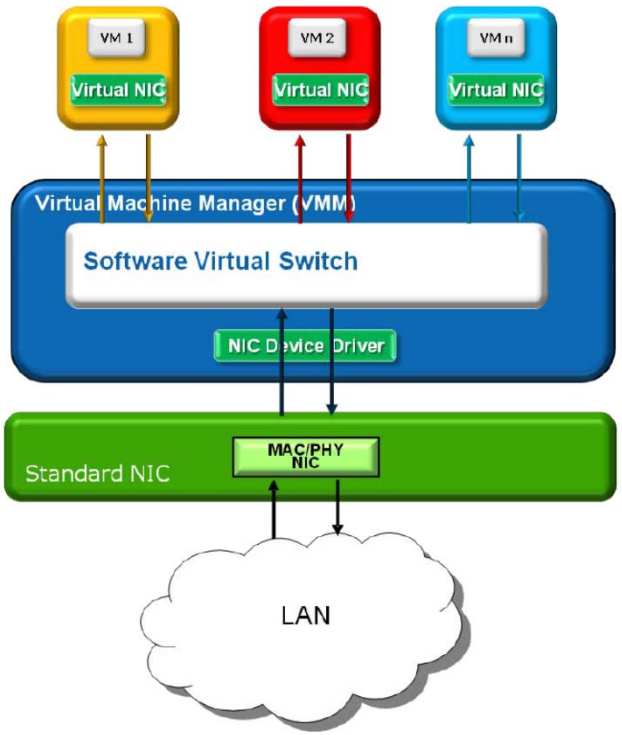
- Parses the I/O commands
- Translates guest addresses into host physical addresses
- Ensures that all referenced pages are present in memory
- Maps multiple I/O requests from VMs into a single I/O stream for underlying hardware

## Examples

- Open VSwitch (Layer-2), Juniper Contrail (Layer-3)

## Drawback

- Poor performance




* Source: Intel White Paper on SR-IOV



AT&T

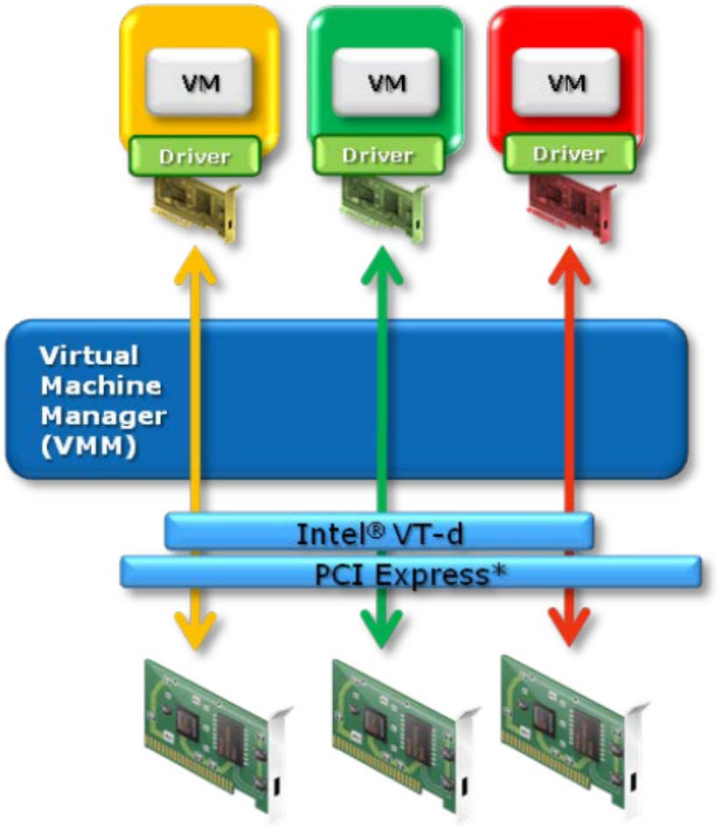
# Direct Assignment

## Bypasses the VM I/O emulation layer to write directly to the memory space of a virtual machine

- Akin to servers' ability to safely DMA data to directly to/from host memory
- Uses enhancements like Intel VT-d
- Results in throughput improvement for the VMs

## Drawback

- Limited scalability; a physical device can only be assigned to one VM



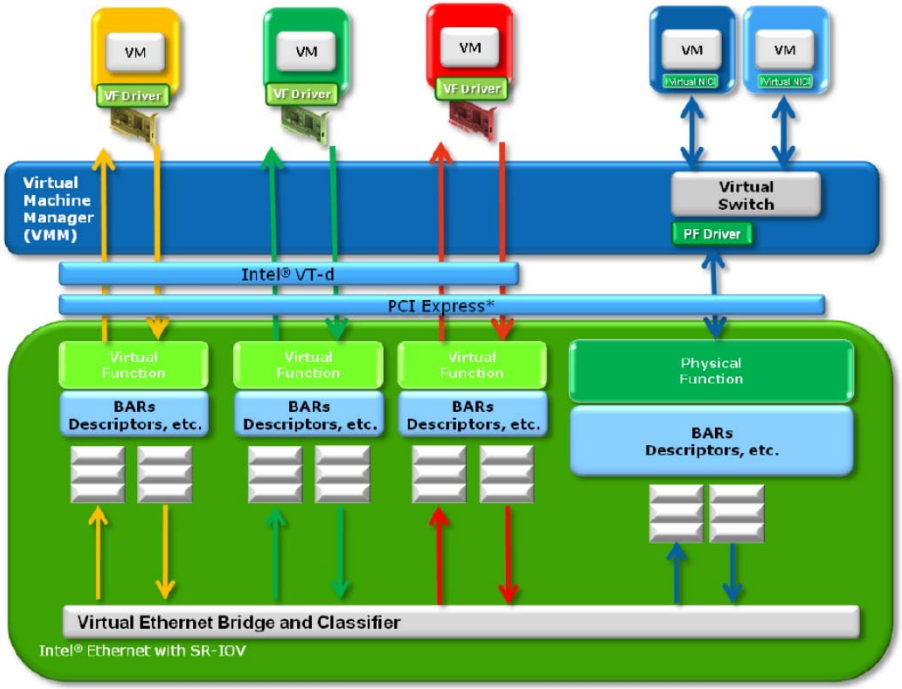

* Source: Intel White Paper on SR-IOV

# Single Root I/O Virtualization (SR-IOV)

PCI-SIG Standard to find middle ground between direct attached & software-based

Standardizes a way to bypass the hypervisor involvement in data movement

- SR-IOV capable device appears as multiple devices, Physical or Virtual Functions (PF / VF)
- Provides independent memory space, interrupts, and DMA streams for each virtual machine
- SR-IOV assigns one or more VF to a virtual machine
- SR-IOV enables hypervisor bypass by providing the ability for VMs to attach to a VF

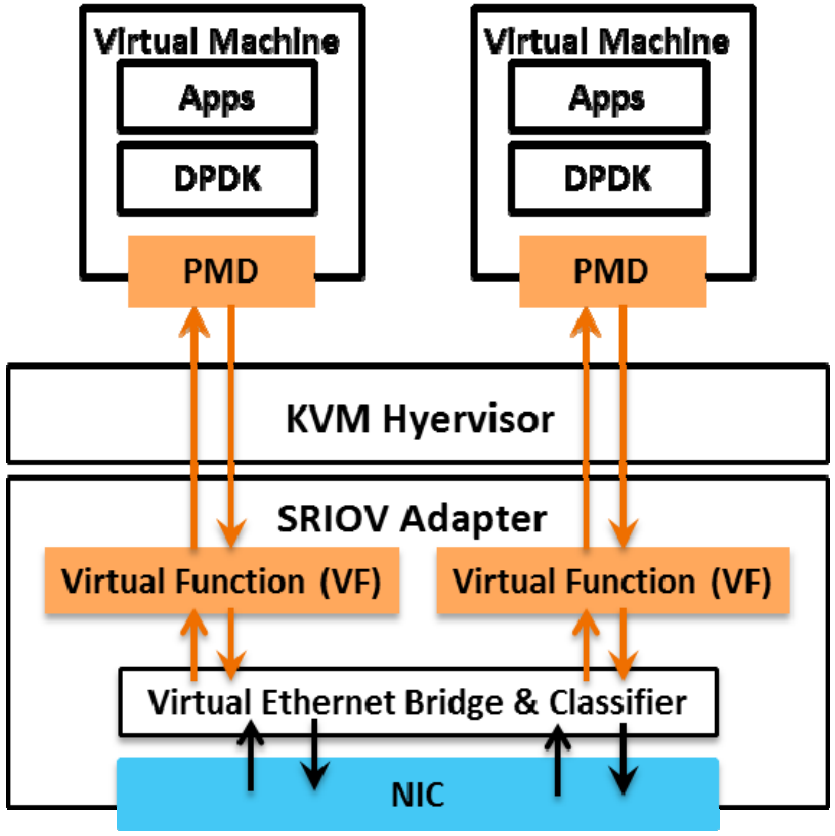



* Source: Intel White Paper on SR-IOV



AT&T

# Using SR-IOV and DPDK




Existing VNF implementations rely on this approach for performance!

# Pros and Cons of these approaches

## Software-based sharing

- ✔ High level of flexibility and control, easy extensibility
- ✔ Preferred approach with virtualization because of recovery and migration
- ✔ Recent efforts on getting better performance
- ✖ Supports only subset of functions provided by the network card (NIC)
- ✖ Significant CPU overhead

## SR-IOV

- ✔ Provides good network performance and sharing of network interface
- ✖ Requires support in the BIOS as well as in the operating system/hypervisor
- ✖ Orchestration limitations; no easy way to migrate VMs
- ✖ Networking restrictions in a cloud environment (because of hypervisor bypass)



AT&T

# NFV platforms/building blocks

## Vector Packet Processor (VPP)- Originally from Cisco, but currently open source (fd.io)

- DPDK based data plane, features implemented as graph, Large code base with lots of features

## DANOS project – Started by AT&T, but now an open source project (getting set up)

- Provides an architecture and code to build disaggregated network functions

## Other efforts: ClickOS (NSDI 2014), NFF-Go (Intel), etc…

## Control plane: Quagga, Free Range Routing (FRR), BIRD, GoBGP, XORP,

## Cellular: OpenAirInterface, OpenEPC, etc…

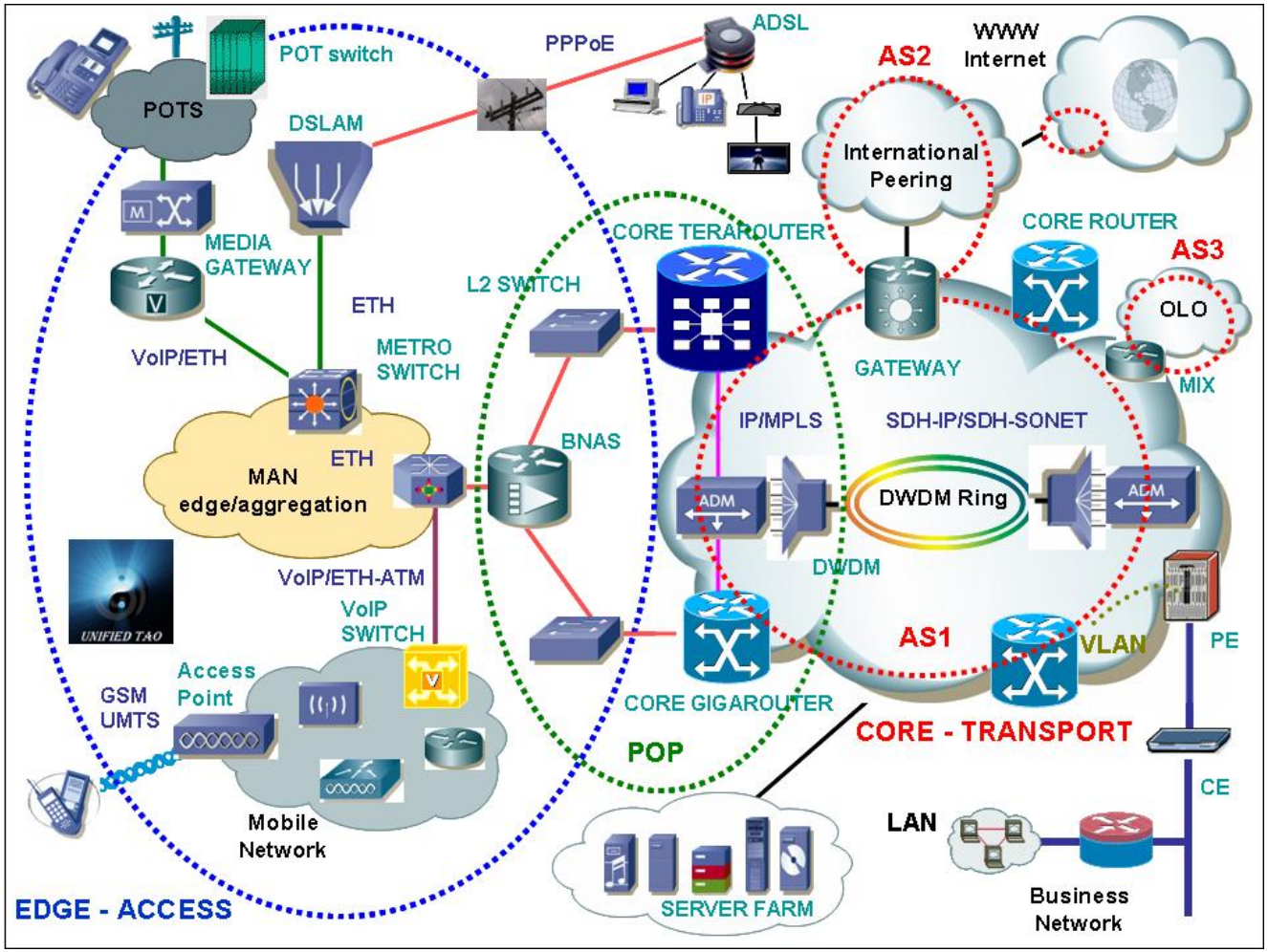


AT&T

# Challenges

With networks, and specifically virtualized networks

# Troubleshoot  the problem with this network!

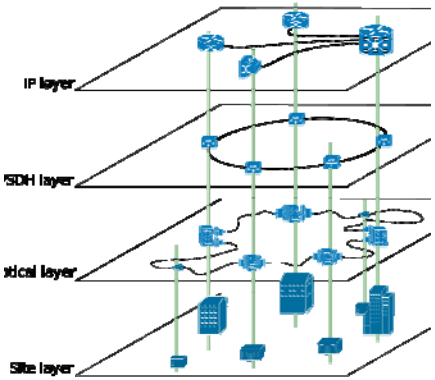


# Traditional Challenges in Monitoring and Measuring IP Networks

## Networks are large, complex, diverse and change constantly →Not trivial to monitor

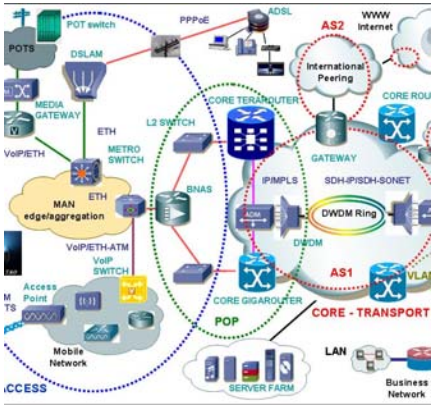
### Scale and size of networks

Hundreds of locations, hundreds of thousands of elements, millions to billions of devices spanning the globe




### Complexity of networks

Networks are built on layers with each layer having a its own quirks; and it is hard to reason about cross-layer impacts




### Diversity of Elements

Most networks have diverse sets of devices, each differing in terms of their function and implementation


### Networks change

Technology evolution, traffic demands and day-to-day updates cause networks to changing constantly

# New Challenges due to NFV and SDN

## Network functions are implemented in a different way

- Composition and disaggregation of functions
- Challenge: changes how a function is monitored

## Additional components come into play

- Virtualization layer (e.g., hypervisor) on servers and for network, cloud management systems, …
- Challenge: need to monitor these additional components

## More ways in which network functions can interact

- Network functions running on same server affect one another's performance
- Challenge: makes it harder to understand behavior of co-hosted NFs

## Nascent technology

- Move to NFV and SDN started recently; IP networks have been around for about 50 years
- Challenge: absence or immature monitoring and management capabilities

AT&T

# Fallacies Recast to NFV*

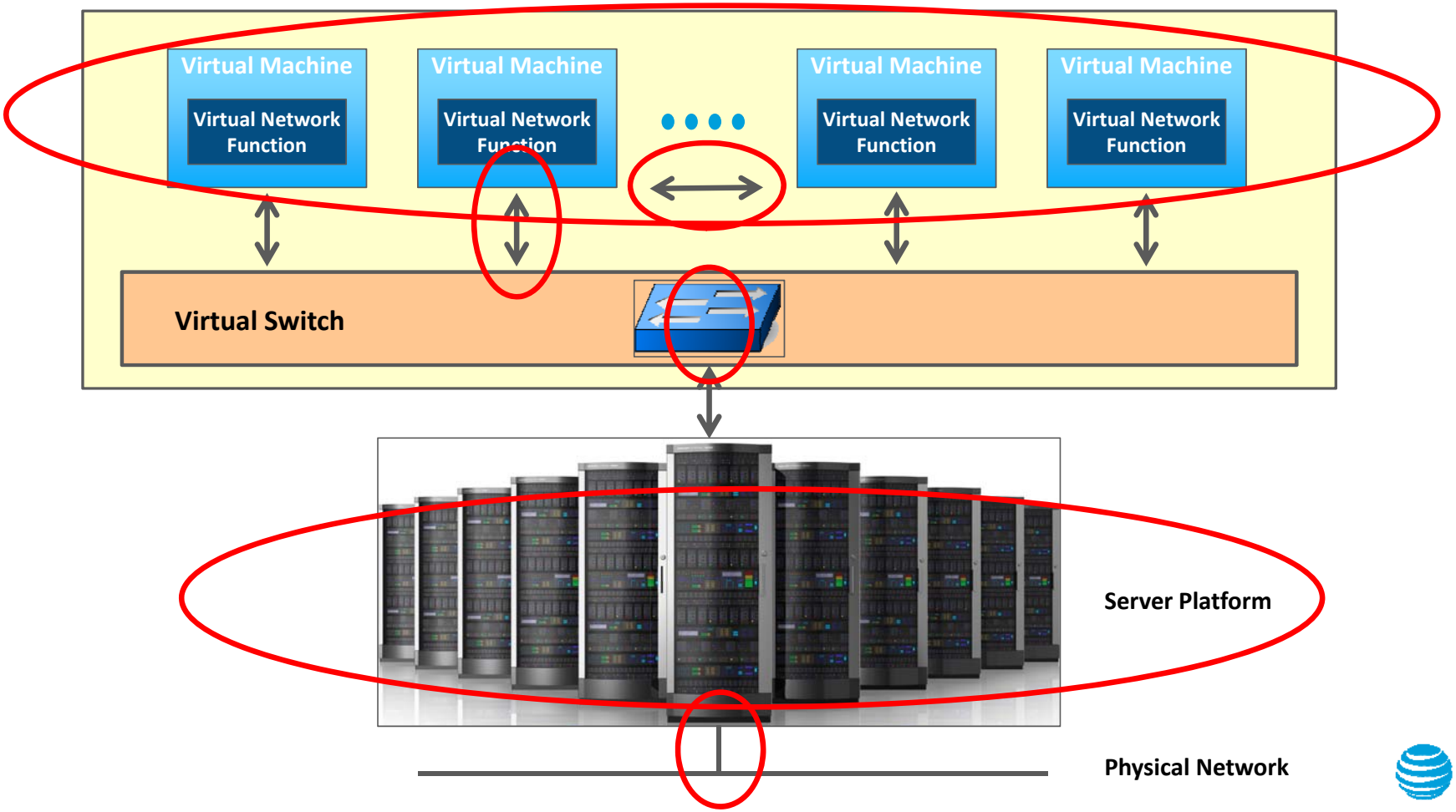
| | |
|---|---|
| The network is reliable. | -> VNF Designs assume the NFVI is NOT reliable |
| Latency is zero. | -> Predictable Performance Matters |
| Bandwidth is infinite. | -> Bandwidth Bottlenecks Occur |
| The network is secure. | -> Security by Design is Needed |
| Topology doesn't change. | -> Change is Continuous |
| There is one administrator. | -> Independent Administrations Exist |
| Transport cost is zero. | -> Cost is Complicated |
| The network is homogeneous. | -> NFVI Heterogeneity is normal |

* Source: Steven Wright @ Dell NFV Summit 2015



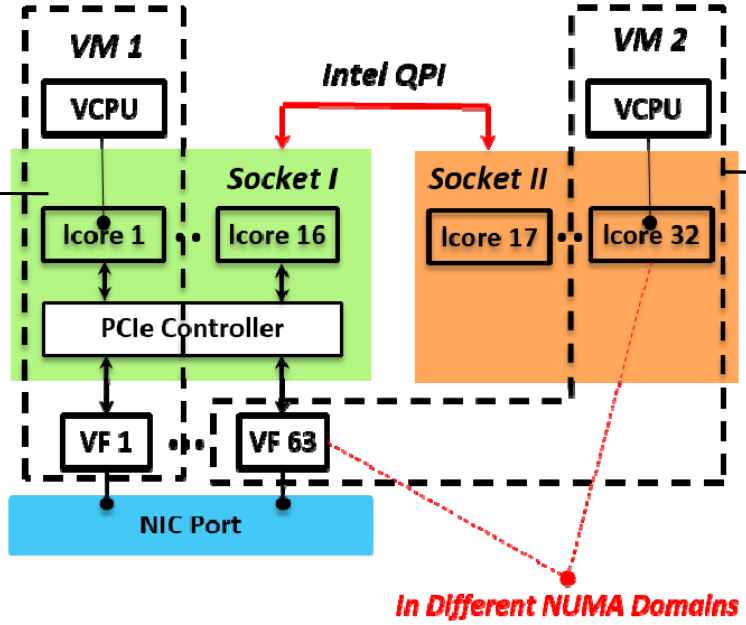
AT&T

# Performance Bottlenecks



# Architectural Factor Impacting Performance

- 32 lcores in total.
- 16 lcores are local (in the same NUMA domain as the NIC port)




Strategies:

- ***NUMA-Aware:*** assign a local lcore (1-16) to a VM.
- ***Random:*** randomly assign a lcore to a VM.
- ***Worst Case:*** assign a remote lcore (17-32) to a VM

# Summary of Performance Issues

## Packet Processing at line rate

- Architecture factors affecting packet I/O at line rate
- Memory allocation and packet copy
- Sharing the network interface card across VMs

## Virtual Switching between virtual network functions (VNFs) at line rate

- Virtual switch implementation and features

## VNF implementation and Inter-VNF communication

- No hardware acceleration, Protocol details, Packet copy overhead

## Running multiple VNFs on a single physical server

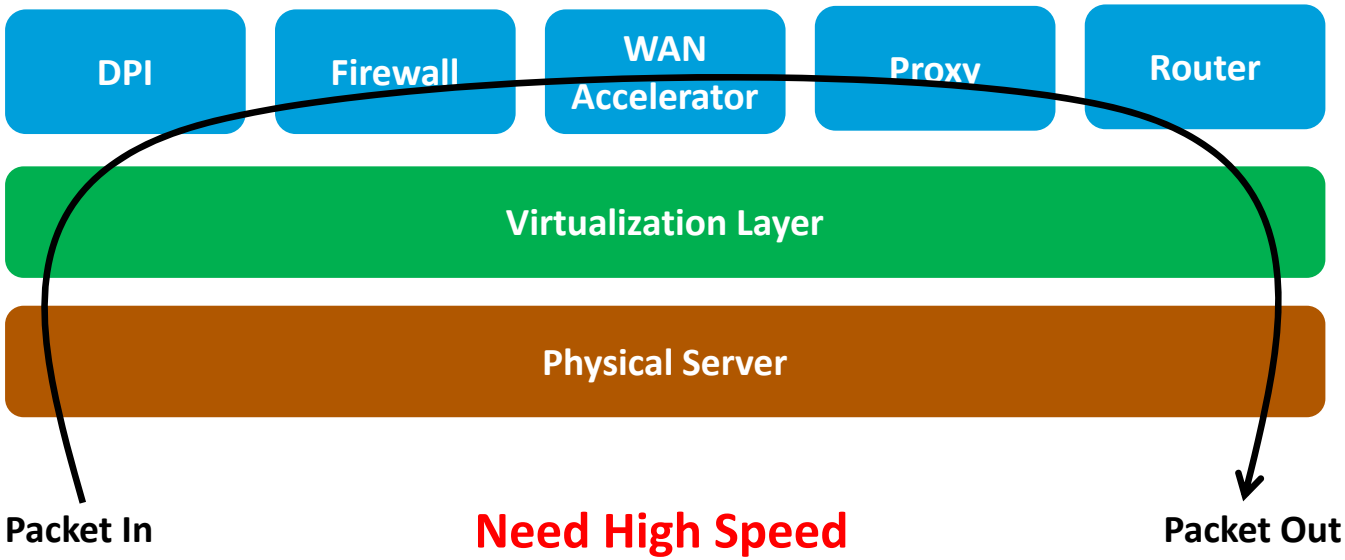
- Hardware architecture, Resource sharing, scheduling



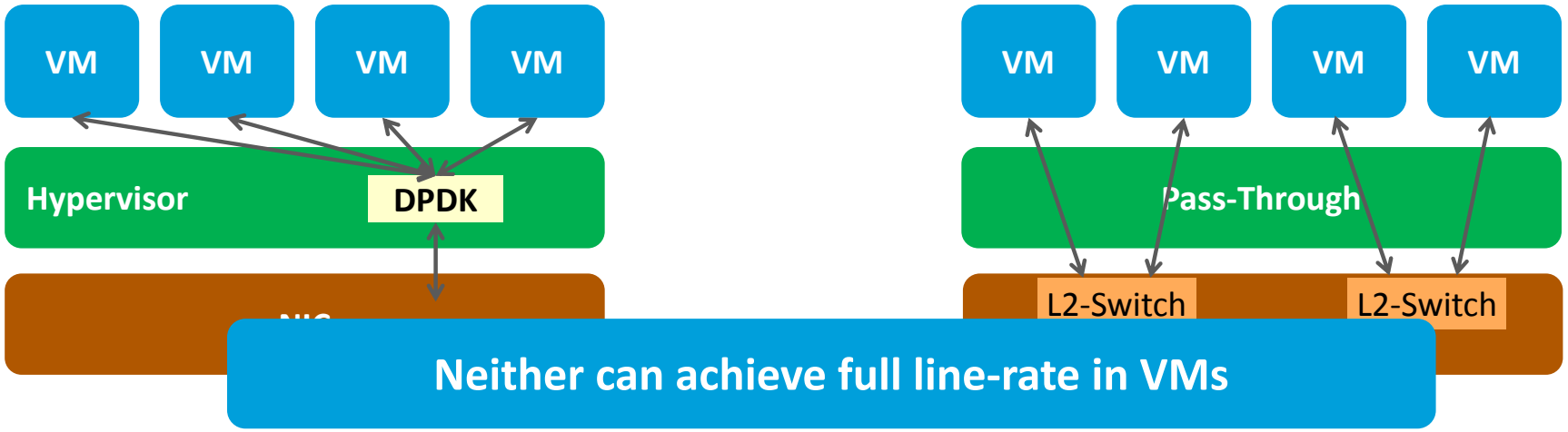
AT&T

# Chained Functionality

## Functions are often sequential



# Architectural Variations Available




- Flexible(dynamic) Configuration
- Control over packet switching
- Control over load balancing
- Has more overhead

- Max 63 virtual functions (tx/rx)
- Static configuration
- Inter-VM switch is limited per port
- No control over packet switching
- No control over load balancing

AT&T

# Enter NetVM…

NetVM (with DPDK) runs in hypervisor User Space

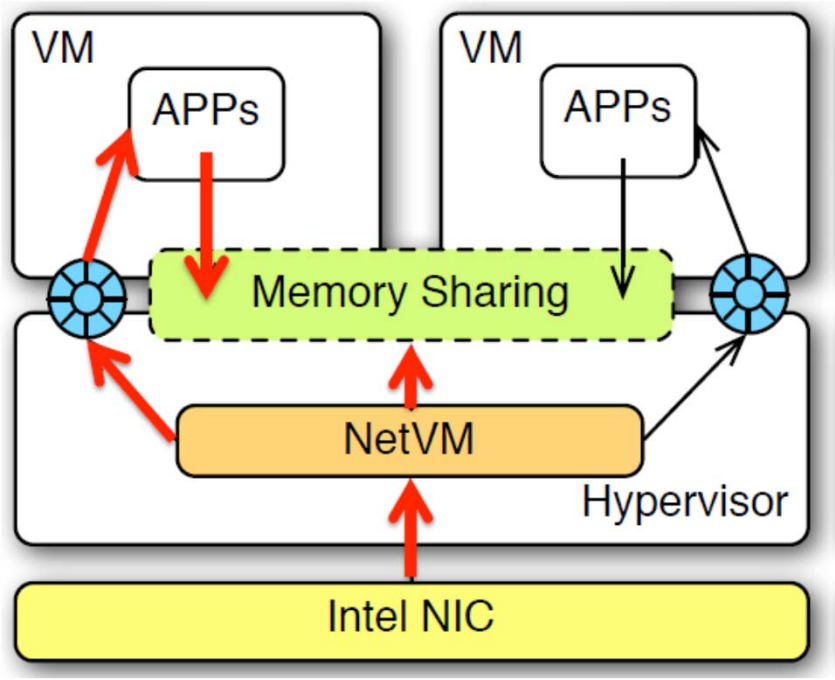
Each VM has its own ring to receive/transmit a packet descriptor

Packet directly DMAs into huge page memory

- Applications in VM receive references (location) via shared ring buffers
- Applications decide an action: chain to another VM, send out, discard

Applications in VMs pass packet references to other VM

Lockless design with core-queue matching (horizontal cores) and data structure separation (vertical cores)




**NetVM can achieve full line-rate in chained VMs**

AT&T

# Virtual Switching Limitations

## Initial virtual switching implementations too slow

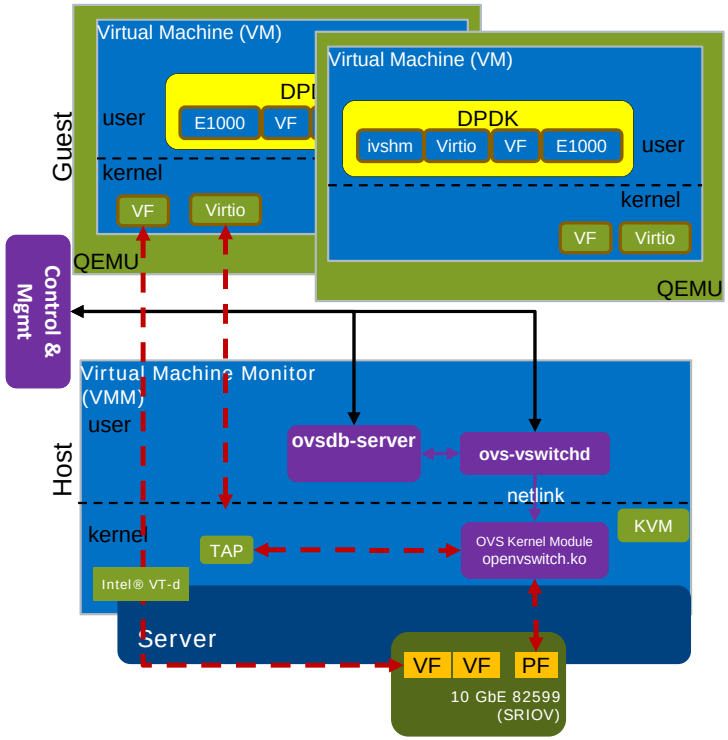
## Poor performance for VM Linux (kernel) virtio loopback ~700kpps

- Involves a packet copy from the Tap Device to the virtio frontend in the Guest, via the backend Vhost module in the host kernel
- Virtio for Linux needs to cross kernel boundary to access user space

## Open vSwtich also slow

- OVS kernel code responsible for packet I/O, tunneling, etc.
- Packet need to cross kernel boundary

## Leads to slow VM-VM access



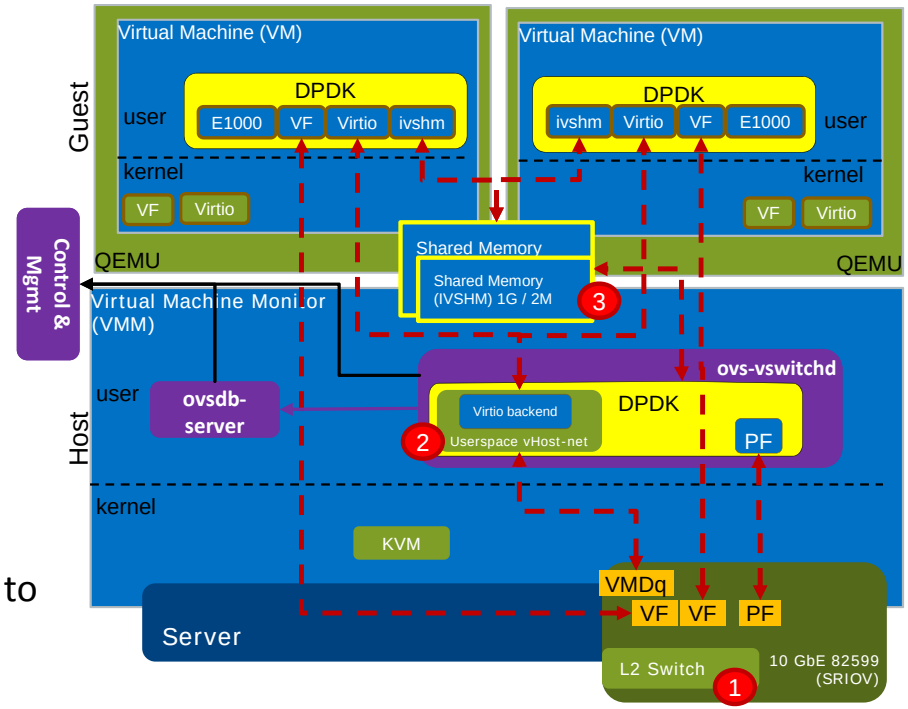
# Accelerating Open vSwitch with DPDK

## Open vSwtich

- OVS kernel code responsible for packet I/O, tunneling, etc.

## DPDK addition

- DPDK PMD takes on packet I/O
  - Some loss of functionality e.g. Tunnels
- DPDK memory allocation with Hugepages (1G) reduces TLB thrashing Virtio
- VM communications via IVSHM model
- Multi-threaded OVS
  - DPDK PMD with RSS (Receive Side Scaling) can distribute packets to cores
  - Flow classification helped by DPDK exact match model



# Other Notable Soft-Switch Efforts

Routebricks – Using software and servers for routing and forwarding

- Basis for much of the enhancements from Intel on packet processing

Netmap - Research efforts from Univ. of Piza for packet processing (similar to DPDK)

- Vale is a software switch implementation based on Netmap

mSwitch – Modular software switch with good performance (Best Paper SOSR'15)

BESS – Modular packet processing pipeline graph, with each node performing a function

Click Modular Router, Juniper Contrail,…

AT&T

# Measurement and Monitoring

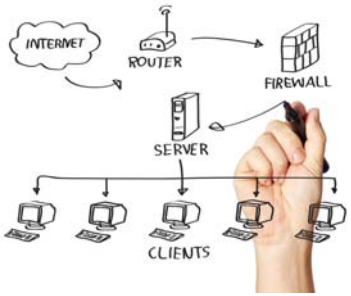
What type of data can be collected?

# Why Measure?

Troubleshooting


Performance

Verification

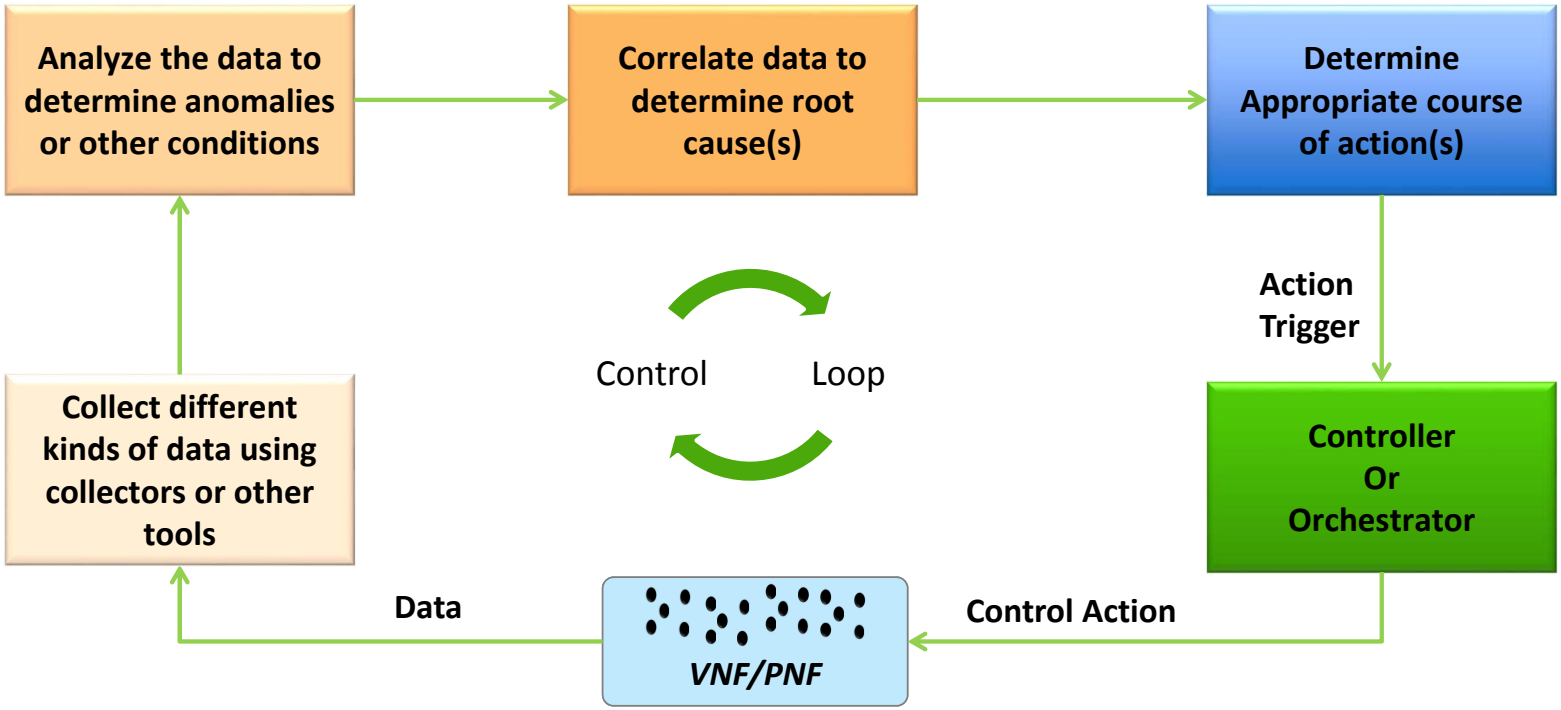



Design and Planning

Maintenance and Upgrades

Security

# Typical Operational Process



# Measurement Idealism

## Measure everything … with zero overhead

- Input to the network:
  - Traffic, commands, (surrounding) environment
- Response from the network:
  - Change in the internal state resource consumption
- Output:
  - Traffic, answers, changes to externally visible state of the network

## No such thing as “zero overhead” measurements

- Corollary: measure as much as possible with as little overhead as possible

AT&T

# NetSight: Achieving Measurement Idealism for Network Traffic*

## Goal: capture packet history for *every* packet at *every* hop

- Packet history = path taken by the packet + header modifications + switch state encountered

## NetSight

- Captures packet histories
  - Every switch sends a *postcard* to NetSight Infrastructure
  - Postcard contains packet header, switch ID, output port(s) and version number for switch forwarding state
- Provides a query language for applications to concisely and flexibly retrieve packet histories of interest

## Paper shows feasibility and practicality of NetSight on Stanford backbone network

- Average path-length: 5 hops, average packet-size: 1031 bytes
- Overhead of NetSight: 3% extra traffic

## AT&T's network:

- Carries more than 150 petabytes on an average business day as of December 2016**
- Overhead of NetSight: 4.5 Petabytes/day ☹

***I know what your packets did last hop: Using Packet Histories to Troubleshoot Networks**, USENIX NSDI 2014

** **http://about.att.com/content/csr/home/issue-brief-builder/people/network-reliability.html**



AT&T

# Active v/s Passive Network Monitoring

## Passive Monitoring

- Collect different types of data and infer what is going on
- Let network function provide information “on their own”
- Examples: SNMP, Syslog, IPFIX/NetFlow, sFlow, control-plane monitors

## Active Monitoring

- Inject packets into network for desired measurements
- Examples: Ping, Traceroute, iPerf

# SNMP (Simple Network Management Protocol)

## Purpose

- Collect information from network functions about traffic and state
- Get notification of events

## How

- Information organization
  - Hierarchical and extensible structures called Management Information Base (MIB)
- SNMP Manager …
  - queries network function for specific information periodically
  - receives traps for desired events

## Limitations

- Periodic query of information means sampling
- Data can get lost
- Have to go through IETF to standardize new information

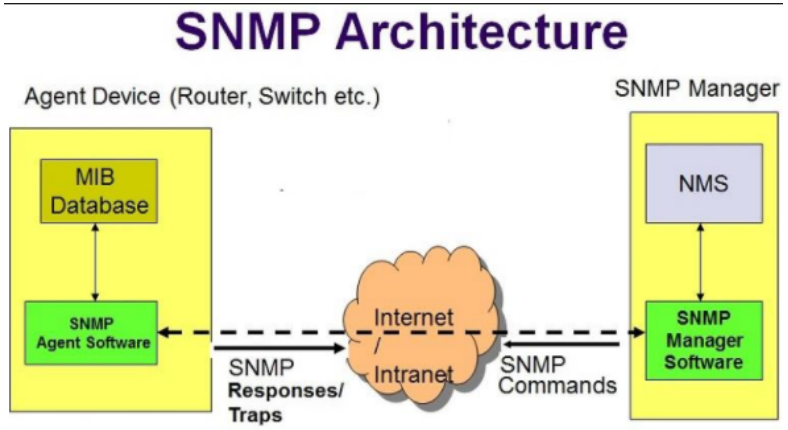



Image source: http://aalvarez.me/blog/posts/understanding-snmp-and-net-snmp.html

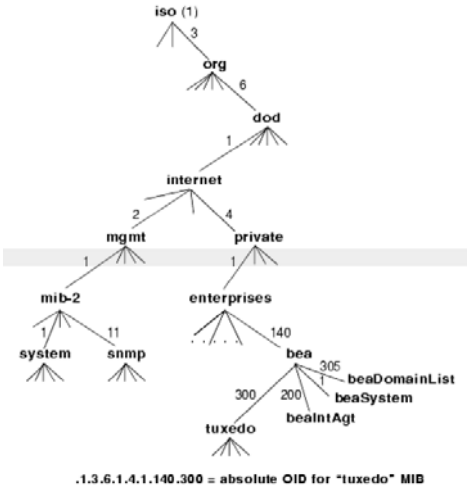



Image source: https://docs.oracle.com/cd/E13203_01/tuxedo/tux81/snmpmref/1tmib.htm

# Syslog

## Purpose

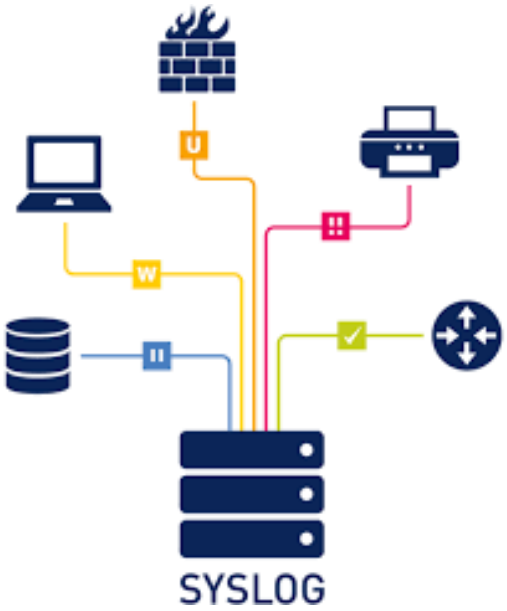
- Allow network functions to convey errors, warnings, information via {key, value} pairs or textual messages

## How

- Content conveyed between syslog applications: generators, relays and collectors
- Generators, relays and collectors can be on same or different network functions

## Limitations

- Lack of standardization in how content is formatted
- Content is transported in an unreliable manner




AT&T

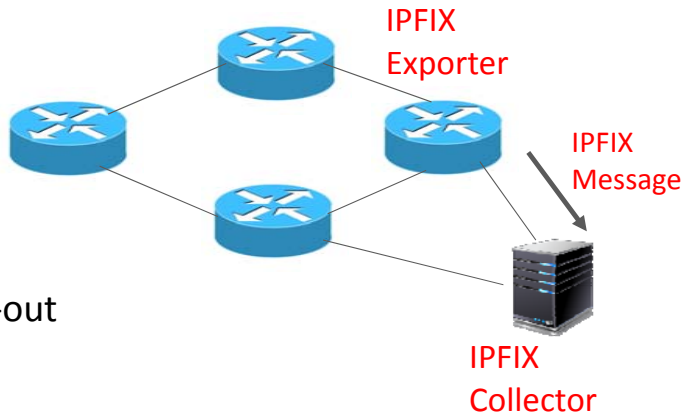
# Netflow / IPFIX (IP Flow Information Export)

## Purpose

- Gather information about amount of traffic on a per-flow basis
  - Flow = source address, destination address, source port, destination port, transport protocol

## How

- Each router …
  - samples 1 out of N packet and updates count for the flow
  - sends flow record to one or more collectors upon flow completion or time-out
- IETF-standardized protocol based on Cisco-proprietary Netflow




## Limitations

- Sampling leads to information loss, especially for flows with low traffic volume
- Does not provide any information about the traffic content

# sFlow (Sampled Flow)

## Purpose

- Gather information about individual packets within traffic

## How

- Routers sample and truncate packets before sending them to a collector
  - Packets to be sampled can be specified through configuration
- Routers also send number of packets sent/received
- Information is sent over UDP

## Limitations

- Sampling leads to information loss

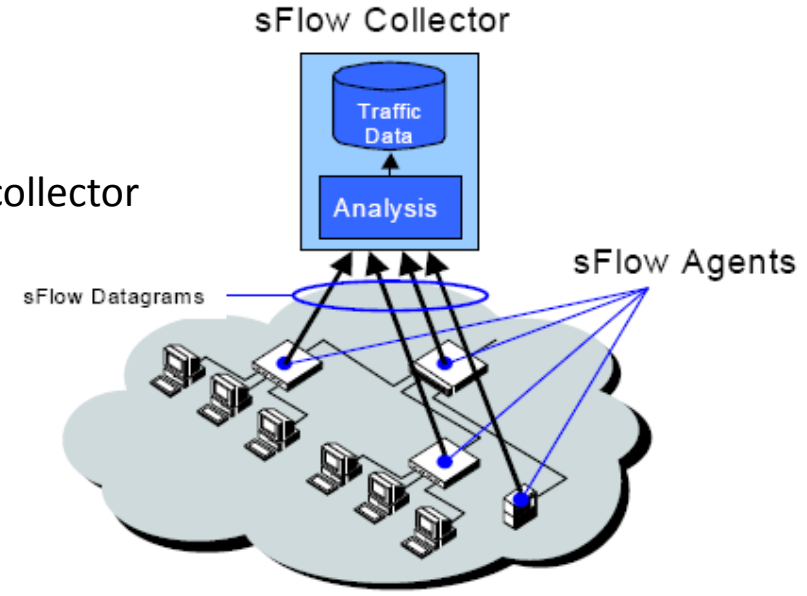



Image source: https://kb.juniper.net/InfoCenter/index?page=content&id=KB14855

# FlowRadar: IPFIX/Netflow for Data-centers*

## Problem with running IPFIX/Netflow in data-centers

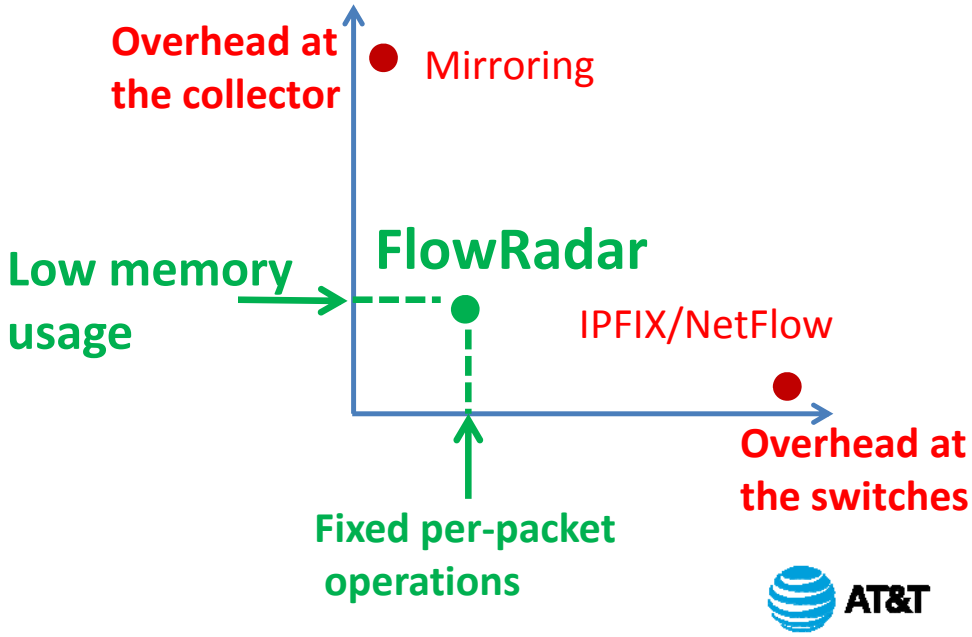
- Running IPFIX exporter on switches place is enormously expensive

## Goal of FlowRadar: report counters for all flows at fine-grained time granularity

- Mirroring packets from switches to a collector puts enormous burden on network and collector

## FlowRadar Key Idea:

- Division of labor between switches and collector




* **FlowRadar: A Better NetFlow for Data Centers**, USENIX NSDI 2016

# FlowRadar Details

## How FlowRadar performs division of labor

- At every switch
  - Encode per-flow counter in an efficient data-structure
    - Use of Invertible Bloom Lookup Table
      - Easy to implement in merchant silicon
      - Fixed operations in hardware
      - Small memory requirement: 2.36 MB for 100,000 flows
  - Export encoded flow-set every 10 ms to collector
- At the collector
  - Decode flow-sets to determine per-flow counters
    - Leverage information from multiple switches

## Prototype implementation in P4 simulator

- Two use-cases on top of FlowRadar
  - Transient loop/blackhole detection
  - Per-flow loss map



AT&T

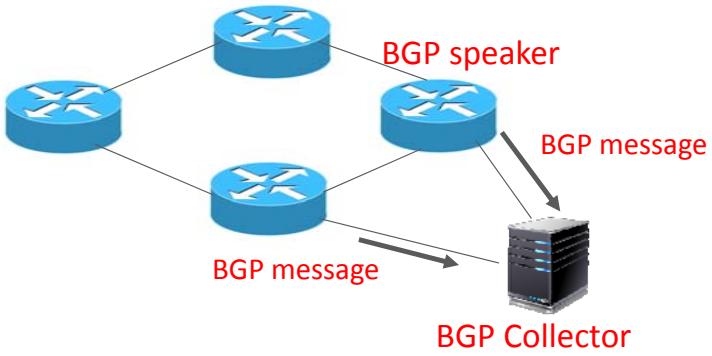
# Route Monitoring

## Purpose

- Determine how packets are routed in (traditional) networks

## How

- Collect routing messages from routers by establishing a peering session
- Use BGP-LS (link-state) and BMP (BGP Monitoring Protocol)
  - BGP-LS: inject topology changes conveyed by IGP into BGP
  - BMP: BGP info is conveyed to a collector




## Benefits

- Allows tracking of control-plane state in (near) real-time
- Provides visibility into how routers are forwarding traffic

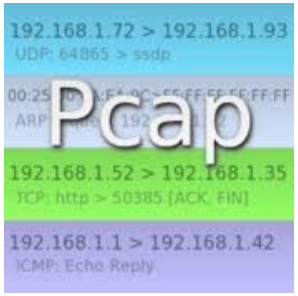
## Limitations

- Places extra load on routers

# Programmable Packet Capture

## Purpose

- Capture packets traversing a network function matching specified criteria




## How

- Capture packets arriving or leaving an interface in the kernel
  - Several libraries allow packet capture
    - libpcap (Unix), Berkeley Packet Filter (BPF), Data Link Provider Interface (DLPI), SOCKET_PACKET sockets (Linux only)
  - Several tools exist for processing and exploring packet captures (real-time or off-line)
    - tcpdump, wireshark, …

## Limitations

- Requires privileged (root) access - privacy and security concerns
- High overhead (especially on high-bandwidth links)

AT&T

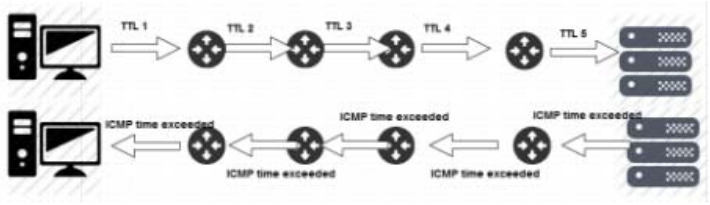
# Active Measurement: Ping, Traceroute, iPerf

## Purpose

- Determine reachability, path and bandwidth/latency to a destination

## How

- Ping
  - Sender sends ICMP echo request packets; destination responds with an echo reply packet
- Traceroute: Exploits TTL expiry mechanism to determine path
  - IP/MPLS packets have a TTL (Time To Live) which is decremented by every router along the path
  - If TTL reaches zero, the router drops packet and sends ICMP TTL Exceeded message back to sender
  - Sender sends first packet with TTL=1 which elicits response from first router along the path
    - Continues sending packets with increasing TTL eliciting responses from router along the path




- iPerf: tool for active measurements of the maximum achievable bandwidth on IP networks
  - Reports the bandwidth, loss and other parameters

Image source: https://www.slashroot.in/how-does-traceroute-work-and-examples-using-traceroute-command

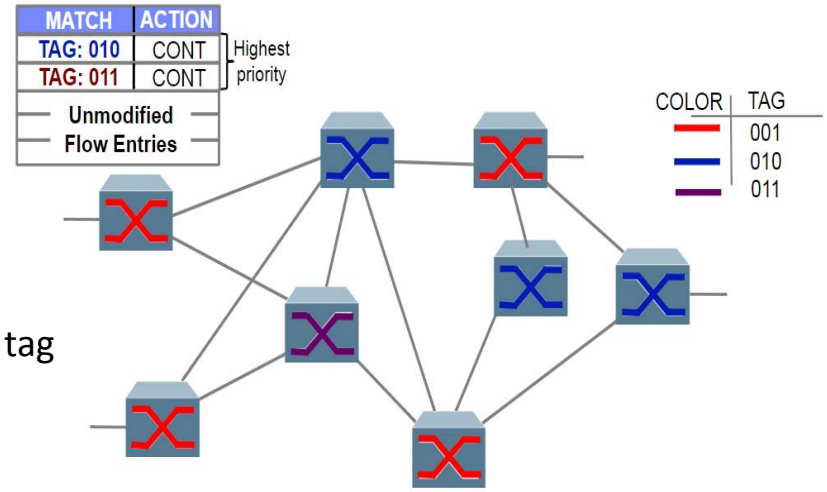
## Limitation

- Places overhead on the network
- iPerf requires a process running on the server

# SDN Traceroute*

## Main idea

- Inject a probe packet at ingress switch
- Program SDN switches so that a probe packet goes alternatively between controller and switches
- Allows controller to record the path taken by the probe packet




## How SDN controller programs switches

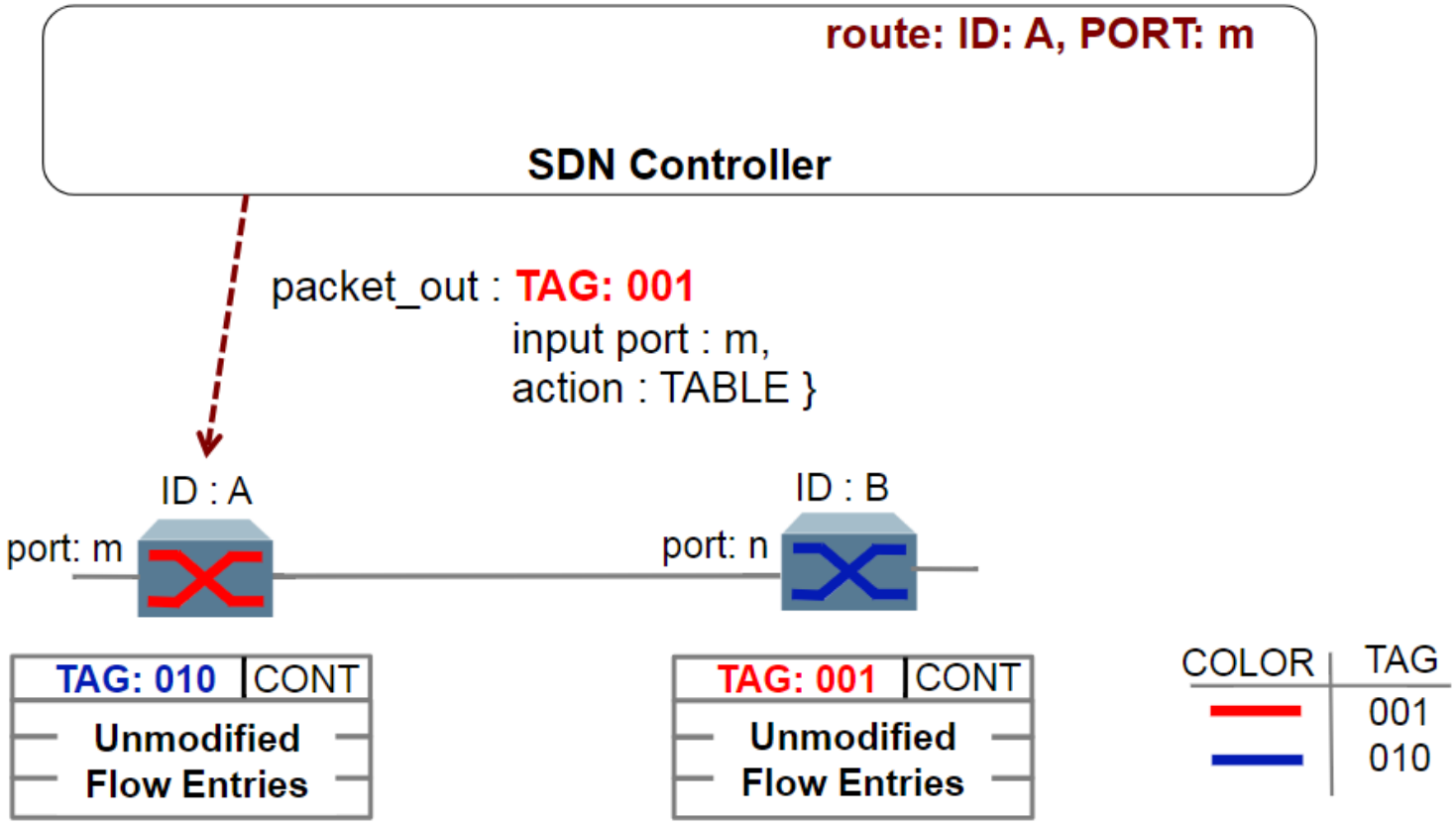
- Assign tags to switches where no two adjacent switches share the same tag
  - Vertex coloring problem
- Adds highest priority rules to every switch
  - If a packet comes in with any tag other than switch's own tag, send to controller
  - If a packet comes switch's own tag, forward along the normal next-hop

*** SDN Traceroute: Tracing SDN Forwarding without Changing Network Behavior**, ACM Sigcomm HotSDN 2014



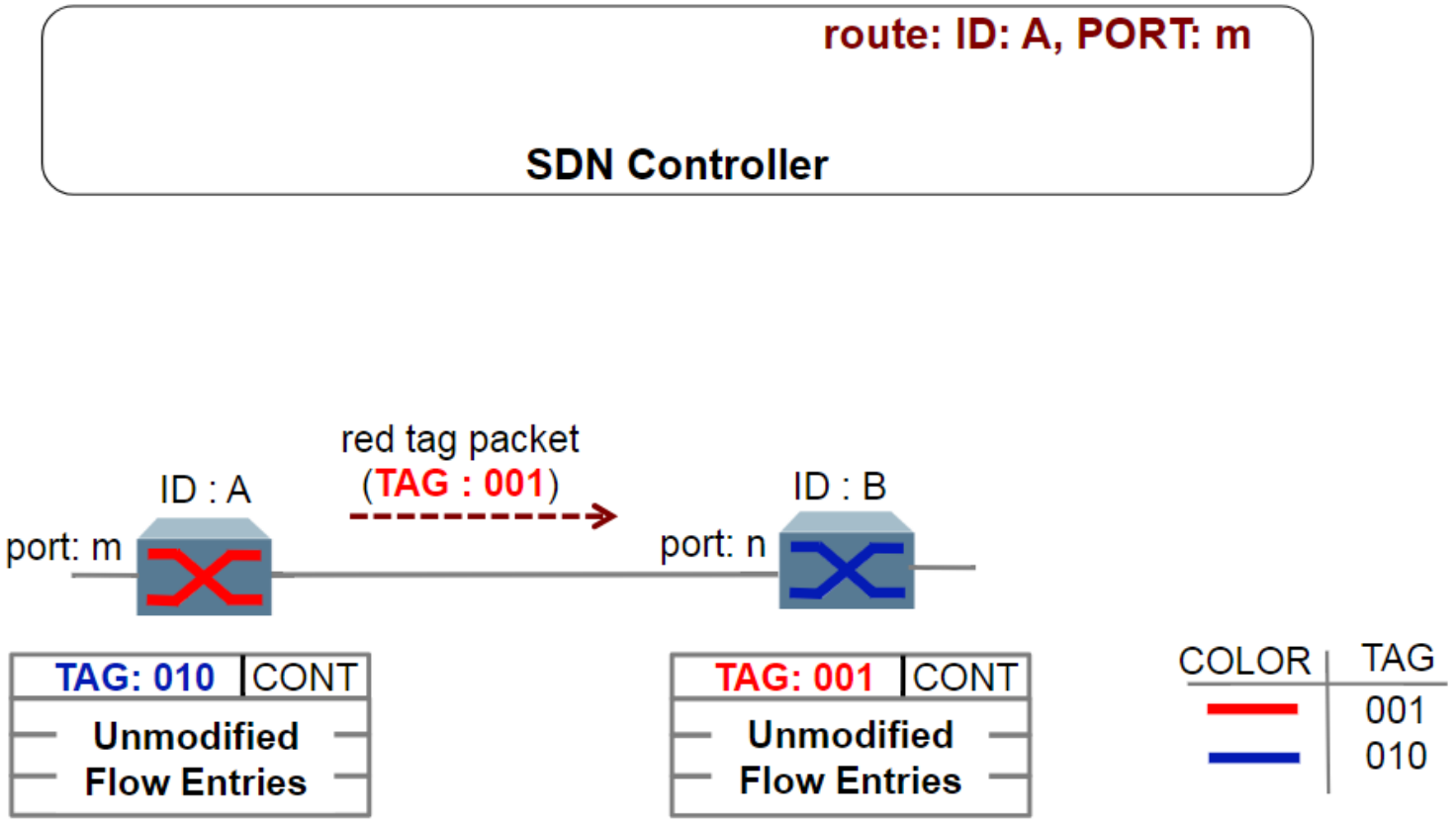
AT&T

# SDN Traceroute Data-plane: Controller Inserts Probe Packet into Network



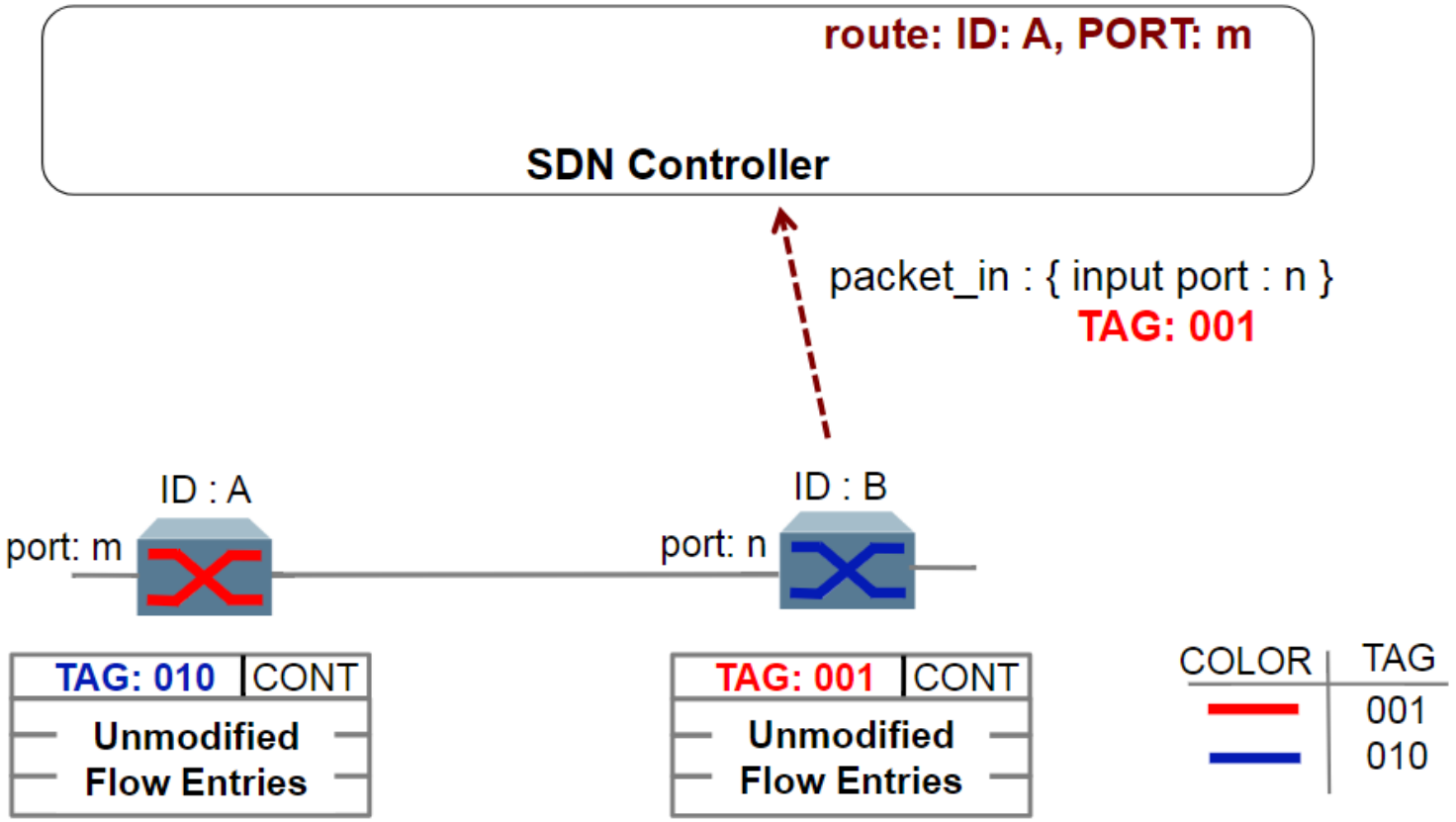

**SDN Traceroute: Tracing SDN Forwarding without Changing Network Behavior**, ACM Sigcomm HotSDN 2014

# SDN Traceroute Data-plane: First Switch Forwards Packet to Second Switch



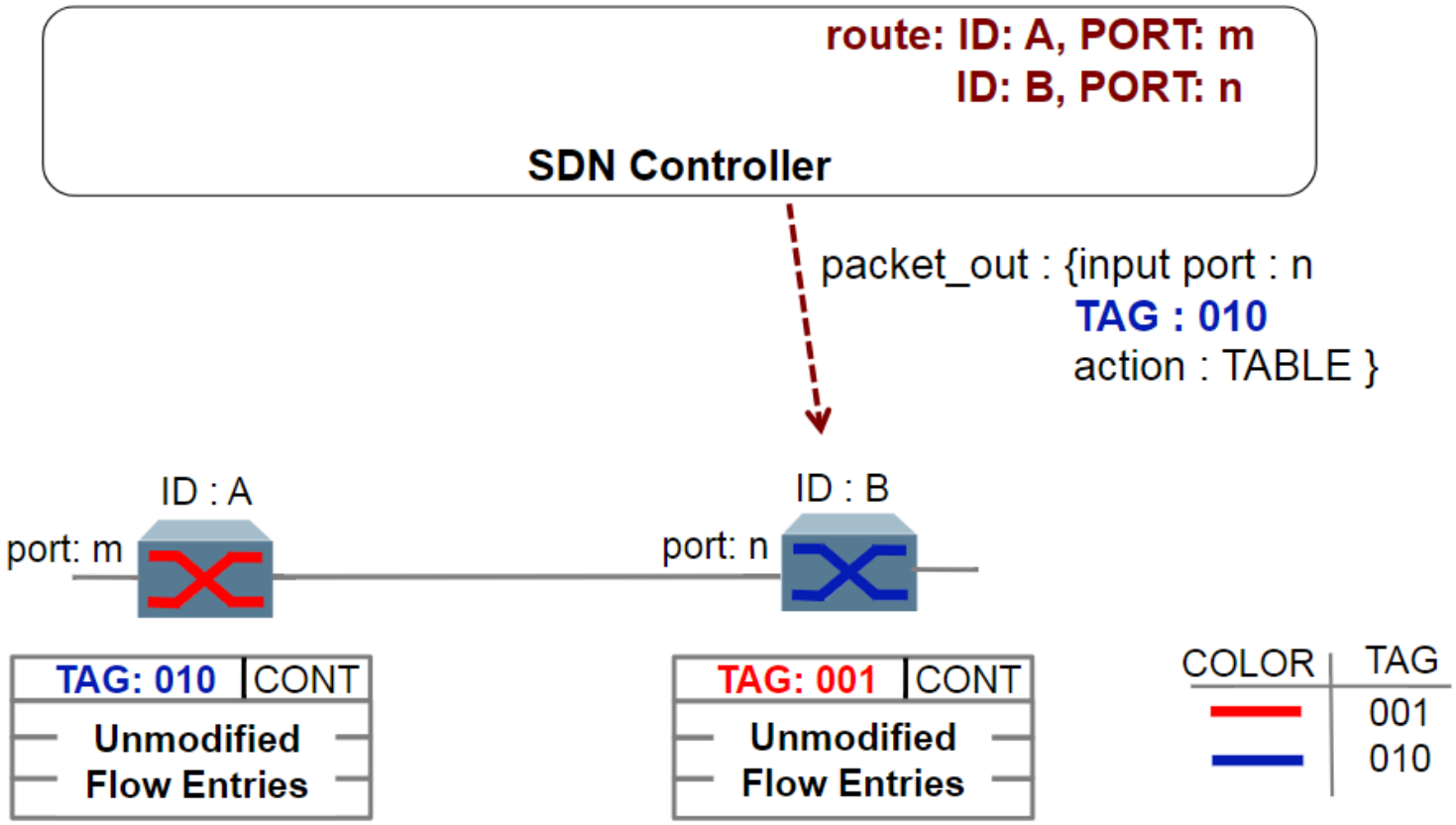

**SDN Traceroute: Tracing SDN Forwarding without Changing Network Behavior**, ACM Sigcomm HotSDN 2014

# SDN Traceroute Data-plane: Second Switch Sends Packet to Controller




**SDN Traceroute: Tracing SDN Forwarding without Changing Network Behavior**, ACM Sigcomm HotSDN 2014

# SDN Traceroute Data-plane: Controller Returns Packet to Second Switch

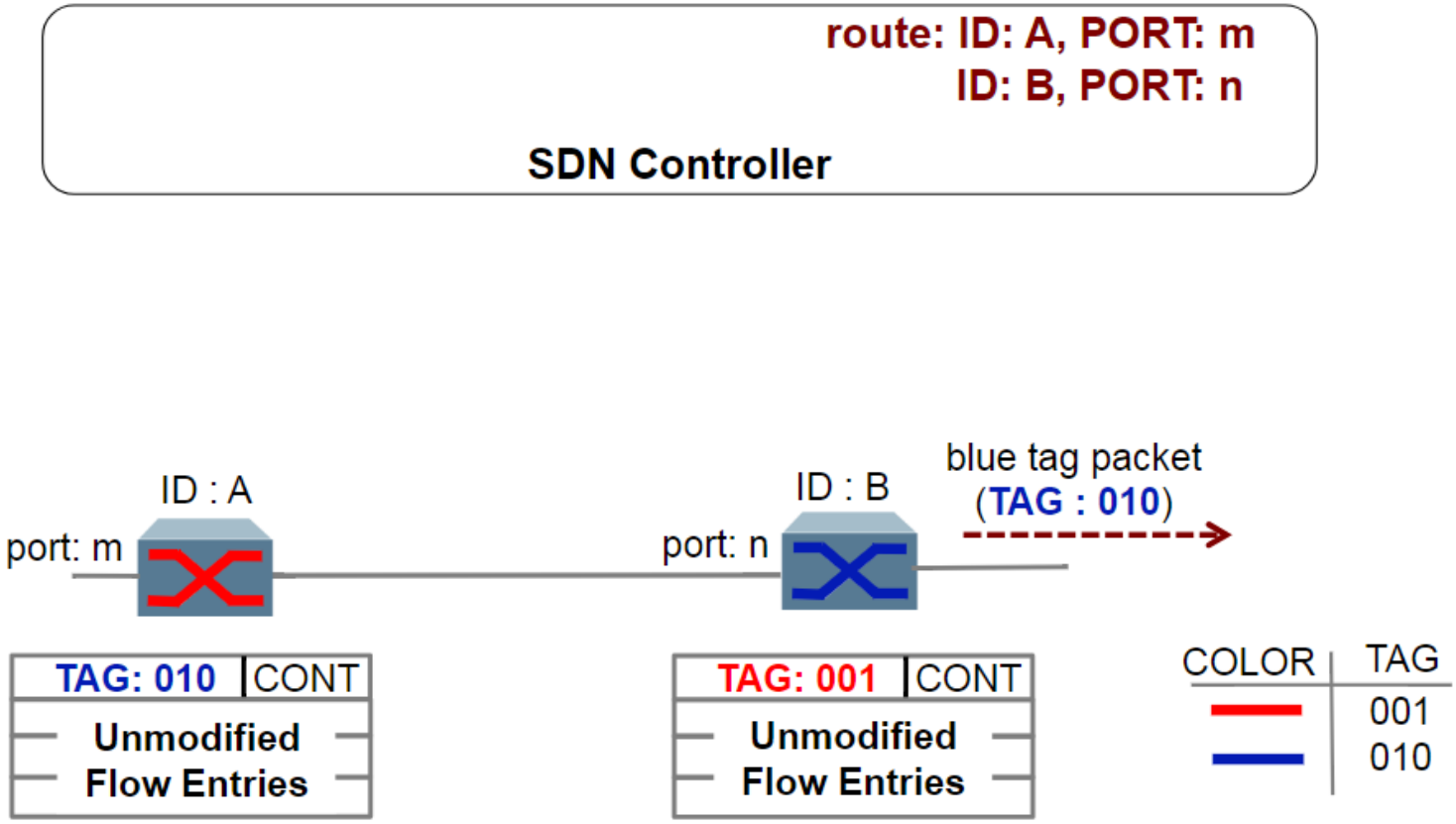





**SDN Traceroute: Tracing SDN Forwarding without Changing Network Behavior**, ACM Sigcomm HotSDN 2014

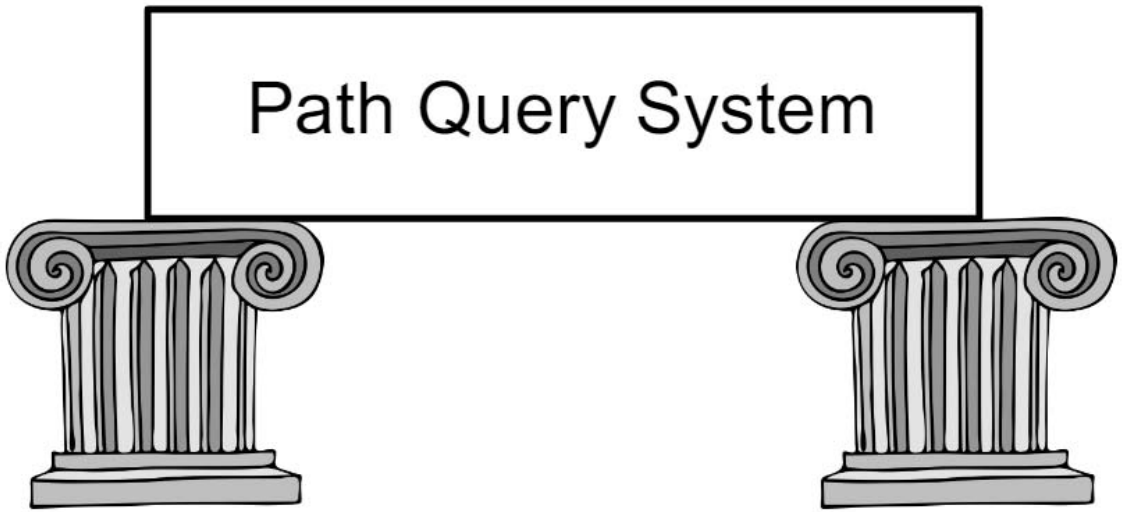
AT&T

# SDN Traceroute Data-plane: Second Switch Forwards Packet to Third Switch




**SDN Traceroute: Tracing SDN Forwarding without Changing Network Behavior**, ACM Sigcomm HotSDN 2014

# Measurements Along a Path in SDN*




**Declarative Query Specification**

Independent of Forwarding
Independent of Other Measurements
Independent of Hardware Details

**Query-Driven Measurement**

Accurate Answers
Pay Exactly For What You Query
Commodity ("Match-Action") Hardware

**Path Query Language**

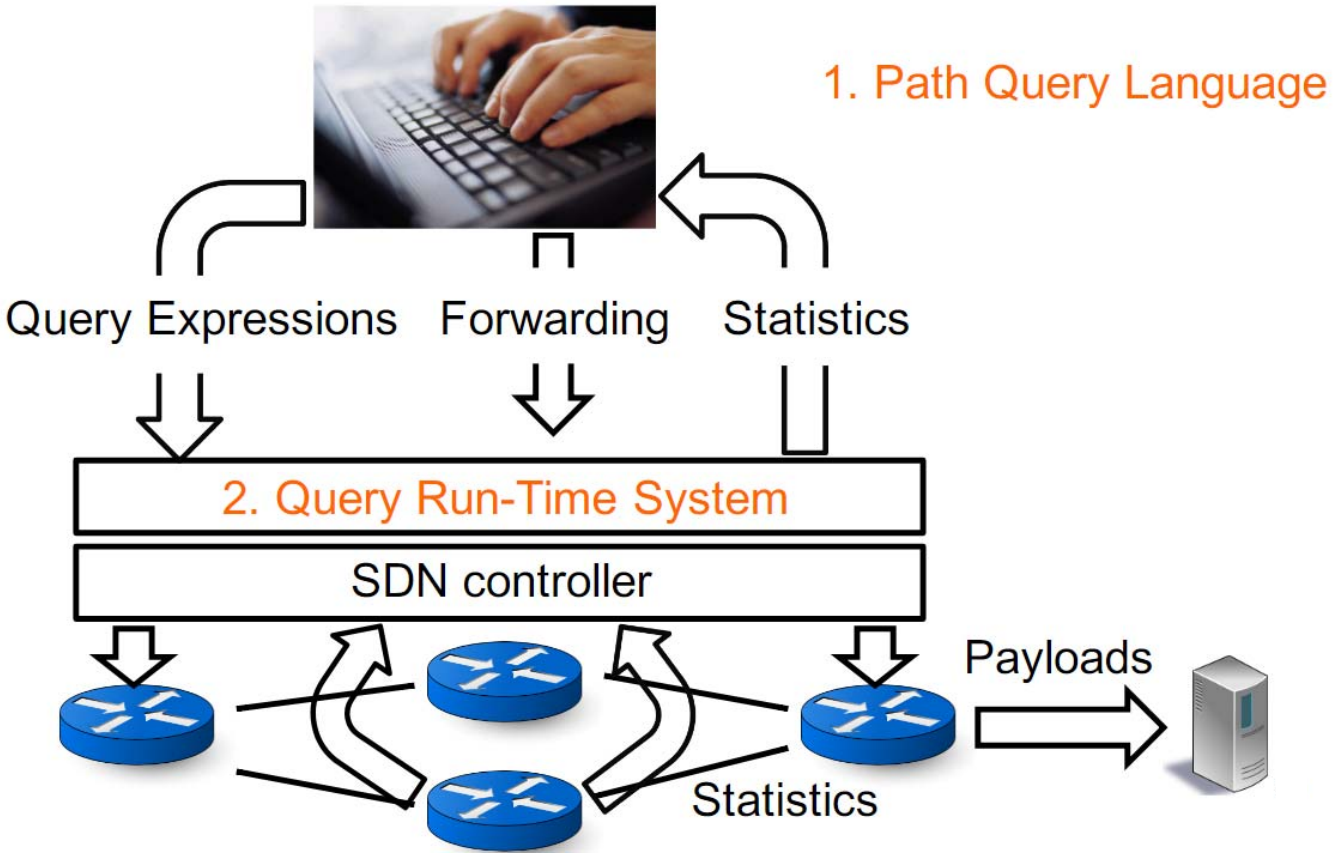
**Query Run-Time System**

***Compiling Path Queries**, USENIX NSDI 2016



AT&T

# How the Path Querying System Works




**Compiling Path Queries**, USENIX NSDI 2016



AT&T

# Software Profiling

## Purpose

- Analyze performance and resource usage of a running program


## How

- Instrument the source-code or binary of a program using a profiler tool
- Determine memory usage, time complexity, usage of particular instructions, or frequency and duration of function calls
- Example: OProfile, an open source statistical profiler for Linux systems

## Limitation

- Provides details of program run-time and resource usage, but overhead can skew its performance



AT&T

# Software Tracing

## Purpose

- Record details of program execution for debugging and trouble-shooting

## How

- Trace execution of a program to identify source of problems
  - Write low-level details and messages while the program executes, geared towards developers
- Example: LTTng (Linux Trace Toolkit next generation), a tracing framework for applications
  - For applications written in C/C++, Java, Python, …

## Limitation

- Can slow execution of the program

AT&T

# Measurement and Monitoring

Data Collection and Tools in Today's Cloud

# A Study of Issues in Cloud Systems*

## Issues opened by users for six cloud-based systems:

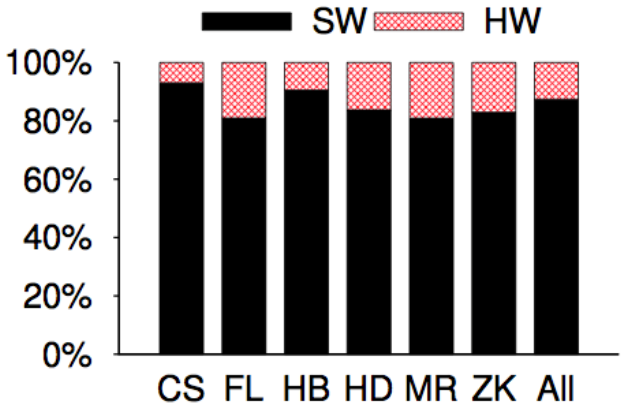
- Hadoop MapReduce, Hadoop File System (HDFS), HBase, Cassandra, ZooKeeper and Flume
- Issues submitted over three years

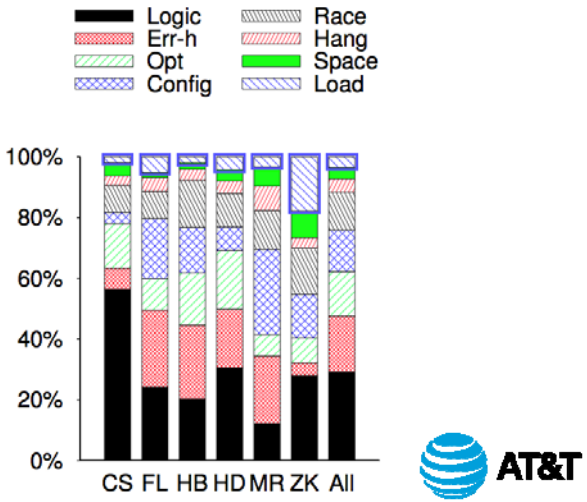
## Classification of issues across various dimensions

- Examples:
  - Bug scope (single machine, multiple machines, entire cluster),
  - Hardware (core/processor, disk, memory, network, node)
  - Software (logic, error handling, optimization, config, race, hang, space, load)

## Cloud Bug Study Database:

- http://ucare.cs.uchicago.edu/projects/cbs/




* **What Bugs Live in the Cloud? A Study of 3000+ Issues in Cloud Systems**, ACM SoCC 2014

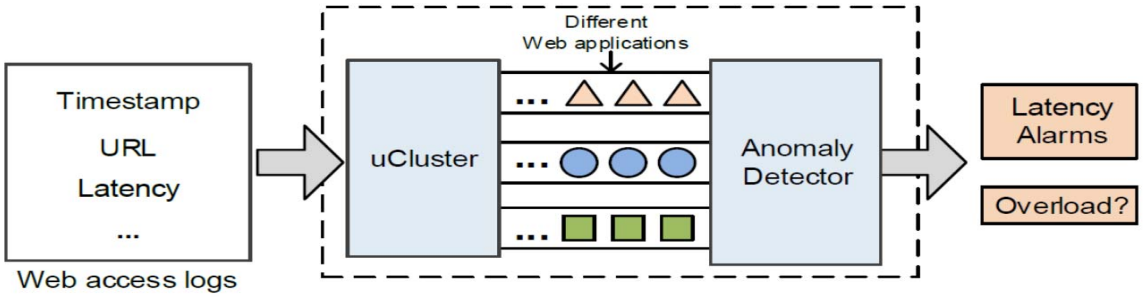
# Amazon CloudWatch* and CloudWatch+**

## CloudWatch monitors EC2 resources and applications in real-time

- Allows users to (a) collect and track metrics, (b) define policies on the metrics
- Several metrics are available (e.g., CPU, Disk, network, etc.)
  - Metrics are organized into namespaces (e.g., name-space: AWS/EC2 for above metrics)
  - Metrics have dimension which can be used to filter statistics collection based on name/value pairs

## CloudWatch+: Application-aware web latency monitor for cloud tenants

- Two components:
  - uCluster: automatically classifies web requests into applications based on the URL clustering
  - Anomaly Detector: identifies time-periods with anomalous latency based on a threshold




* Guide: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/acw-ug.pdf#cloudwatch_architecture

** **Application-Aware Latency Monitoring for Cloud Tenants via CloudWatch+**, IEEE CNSM 2014

# Network Performance in Amazon EC2 [Infocom 2014]


## Conducted analysis of networking performance for VMs in Amazon EC2

- Metrics: throughput, delay and packet loss
- Transport protocols: TCP and UDP

## Findings

- Performance of some VMs is worse than other VMs
- Variation and degradation is due to the role played by CPU scheduler for handing computation tasks and networking tasks (e.g., copying packets between physical NICs and VMs)

## How to mitigate the issue?

- CPUs are divided into two pools:
  - Pool for network-intensive VMs
  - Pool for compute-intensive VMs

# Monitoring in Microsoft Azure*


## Allows users to collect various metrics, activity logs and diagnostic logs

- Deep application monitoring (application insights) and Deep infrastructure monitoring (log analytics, management solutions, network monitoring, service map)
- Core monitoring (azure monitor, advisor, service health and activity log)
- Shared capabilities (alerts, dashboards, metrics exporter)

## NetPoirot: Fault-localization based on TCP statistics collected at end-hosts (Sigcomm 2016)

- Fault localization happens at the granularity of client, server or network
  - Fine-grained localization and root-cause analysis performed by administrators
- Use of machine learning algorithm for classifying observed TCP stats into fault localization classes
  - Supervised learning on 132 numeric values collected every 30 seconds for each TCP connection
- Metric collection is done by NetPoirot agents running on VMs (or hypervisors)
  - Prototype implementation for Windows and is deployed in Microsoft Azure production cloud

*Guide: https://docs.microsoft.com/en-us/azure/opbuildpdf/monitoring-and-diagnostics/toc.pdf?branch=live

AT&T

# Monitoring in Microsoft Azure (Cont'd)


## Pingmesh: active probing between pairs of servers in a data-center [Sigcomm 2015]

- Forms multiple levels of complete graphs, and then performs pings within these graphs
- Helps define and track service level agreements (SLAs), determine if issue is related to network
- Allows determining issues like network latency, packet drop rate, silent packet drops, etc

## EverFlow: allows tracing of specific packets; implements packet filter on top of "match and mirror" functionality of commodity switches [Sigcomm 2015]

- Match and Mirror important packets
  - TCP packets with SYN, FIN and RST flags, desired packet with a debug bit in header, packets belonging to control-plane protocols such as BGP
- Allows insertion of probe packet at any switch and tracing of the packet through the network
- Scalable trace analyzers and switch-based reshufflers to balance load, and maintain flow affinity with analyzer

# Measurement and Monitoring in Google Cloud

## Stackdriver Debugger

- Allows inspection of application-state without stopping or slowing down the running app or without adding any logging information

## Stackdriver Monitoring

- Collects metrics, events and metadata from Google cloud platform, AWS, hosted uptime probes, application instrumentation

## Stackdriver Logging

- Allows you to store, search, analyze, monitor and alert on log data and events from Google cloud platform and AWS

## Stackdriver Error Reporting

- Aggregates and displays errors produced by running services

## Stackdriver Trace

- Distributed tracing system that collects latency data from apps

## Stackdriver Profiler

- Continuously gathers CPU usage and memory-allocation information from the service and attributes usage to the application's source-code

# OpenStack Monitoring


## Ceilometer Framework for monitoring and metering OpenStack cloud*

- Distributed system to collect data used for metering, monitoring and alerting
- Agent-based architecture with three kinds of agents
  - Compute agent: runs on compute nodes and collects resource utilization statistics
  - Central agent: runs on a central management node and collects resource utilization statistics for resources not tied to instances or compute nodes
  - Notification agent: runs on a central management server(s) and consumes messages from the message queue(s) to build event and metering data

## OpenStack is complex; has bugs that can cause instability and failures (SoCC 2013)

- Analysis of OpenStack by systematically injecting faults while executing request
- Compare fault-free execution with “faulty” execution; compare predefined state and behavior specification
- Twenty-three bugs identified in OpenStack; spread across into seven categories

*Guide: https://docs.openstack.org/ceilometer/queens/admin/telemetry-system-architecture.html

# Debugging OpenStack: Hansel [CoNext 2015]


## System that leverage non-intrusive network monitoring to expedite root-cause analysis of faults manifesting in OpenStack operation

- Input: sequence of REST and RPC calls issued during Openstack operations
  - REST calls are used for inter-service communication (e.g., Nova to Neutron)
  - RPI calls are used for intra-service communication (e.g., Nova controller to an agent running on a server)
  - Calls can be logged out-of-order (temporaly speaking), and calls for multiple tasks (e.g., VM create) can be interleaved
- Output: Identify temporal sequence of call and determine sequence of such calls for every task
  - In the event of a fault, determine the task that resulted in the fault by back-tracing the sequence of calls from fault to the task
- Prototype built for Openstack Junos:
  - Network monitoring agents atop Bro in the form of a protocol parser for RabbitMQ messaging protocol
    - 60 lines in C++
  - Central analyzer that receives events from the monitoring agents and performs root-cause analysis
    - ~800 lines in Python



AT&T

# Debugging OpenStack: Gretel [CoNext 2016]


## Leverage non-intrusive network monitoring to expedite root-cause analysis of faults manifesting in OpenStack operations

- Determines fingerprint of every OpenStack task *a priori* by leveraging integration test
- Input: sequence of REST and RPC calls issued during Openstack operations
  - REST calls are used for inter-service communication (e.g., Nova to Neutron)
  - RPI calls are used for intra-service communication (e.g., Nova controller to an agent running on a server)
  - Calls can be logged out-of-order (temporally speaking), and calls for multiple tasks (e.g., VM create) can be interleaved
- Output: Identifies root-cause by matching actual sequence of REST and RPI calls with the fingerprints
  - Determines the task responsible for the problem
- Prototype built for Openstack Liberty
  - Network monitoring agents atop Bro in the form of a protocol parser for RabbitMQ messaging protocol
    - 60 LOC in C++
  - Central analyzer that receives events from the monitoring agents and performs root-cause analysis
    - ~1600 LOC in Python

# Open vSwitch (OVS)

## Software switch that runs in hypervisors to connect VMs

- Supports Netflow, sFlow, allows tracing of a packet as it flows through OVS
- Commercial tool support include Open v_Monitor (OVM) and ZenPack

## Umon: decouples monitoring from forwarding; offers flexible and fine-grained traffic stats [CoNext 2015]

- Deals with monitoring in both user-space and kernel-space models of OVS
  - Implements a monitoring flow table to separate monitoring rules from forwarding rules
  - Algorithm to modify flow rules in the kernel flow table to satisfy both forwarding and monitoring requirements
- Implementation to OVS required modifying two threads:
  - *Handler* thread that implements packet processing logic in the user-space
  - *Revalidator* thread that manages the kernel flow table and retrieves flow stats from the kernel
- Supports rules that allow monitoring on non-routing fields

# Measurement and Monitoring

Troubleshooting

# Virtual Network Diagnosis as a Service [SoCC 2013]

## Framework to allow a cloud provider to offer virtual network diagnosis to its tenants

- Exposes interfaces for configuring diagnosis and querying traffic traces to cloud tenants for troubleshooting their virtual networks
  - Tenants specify a set of flows to monitor.
  - VNF controls the appropriate software switches to collect flow traces and distributes traffic traces of different tenants
- Scales to many tenants by carefully choosing how and where data collection and data aggregation happens
  - Data collection performed on hypervisor virtual switches without impacting existing user traffic
  - Queries executed quickly on distributed "table" servers



AT&T

# Minimal Causal Sequences [Sigcomm 2014]

## Tool and methodology to determine minimal causal sequence of events (captured in a log) that triggers observed bug in an SDN controller

- Rationale: Shorter sequence of events are easier to understand (for a human) than longer ones
- Methodology (input: trace of events that triggered a bug)
  - Replay events in a QA testbed
  - Apply delta debugging (prior work) to inputs
  - Deal with asynchrony by interposing on messages
  - Deal with divergence by inferring absent events
  - Deal with non-determinism by replaying multiple times
- Evaluation
  - Five OpenFlow controllers
    - POX, NOX, Floodlight, Frenetic, ONOS
  - Showed substantial reduction in trace sequence for 17 case-studies comprising of discovered, known and synthetic bugs



AT&T

# OFf [CoNext 2014]

## Debugger for SDN controller applications

- Runs on top of fs-sdn simulator
- Supports controller applications written for POX, OpenDayLight, Ryu, Trema and Floodlight
- Features:
  - Debugging features such as stepping, break-points and watch variables
  - Features for network visibility such as packet tracing, packet replay, alerting and visualization



AT&T

# Measurement and Monitoring

Performance Improvement

# Cut Payload [NSDI 2014]

## Addresses TCP performance problems encountered in data-centers:

- Incast, outcast, out-of-order packets, long-query completion times

## Three key issues with TCP in large-scale data-centers:

- Self-clocking stall
- Inaccurate packet loss notification
- Slow packet loss detection

## Cut payload (CP) proposes two things to solve these problems:

- Drop a packet's payload at an overloaded switch while letting the header go through
- Use a SACK-like precise ACK (called *PACK*) to accurately inform senders about lost packets

## Simulation shows efficacy of CP

- CP works well with existing congestion control protocols in DCN

# ALMA [CNSM 2014]

## ALMA: Application-aware Live Migration Architecture

- Takes into account application-level information, in addition to traditional system level metrics, to determine the best time to perform a VM migration
- Empirical evaluation
  - Three applications: OpenModeller, BRAMS and TPC-H
  - Showed substantial reduction in network bandwidth and live migration time



AT&T

# OpenSample [ICDCS 2014]

## Low-latency, sampling-based measurement platform for SDN

- At every switch sampled packets are collected from incoming ports using sFlow
  - Sampled packets are sent to OpenSample controller
- Every 100 ms, the controller estimates:
  - Utilization of every switch port
  - Set of elephant flows along with their estimated bandwidth and current paths
- A traffic engineering application re-routes elephant flows to relieve congested links if needed
- Prototype implementation for FloodLight OpenFlow Controller
- Evaluation using a four-switch physical test-bed and simulation on Mininet-HiFi



AT&T

# PerfSight [IMC 2015]

## System for accurate diagnosis of performance problems with VNFs

- Models a chain of VNFs as a pipeline of elements (including NFs, hypervisors, host OS etc.) operating on packets
  - Each element modeled as moving a packet from its input to its output with possible code-paths where the packet can be dropped
  - Instruments the relevant code to keep track of number of incoming, outgoing and dropped packets/bytes for every element
    - Wherever possible, takes advantage of existing measurement primitives
- System consists of three components:
  - Agents running on physical servers which collect metrics from various packet-processing elements
  - Controller which receives data from agents based on user queries
  - Applications which perform diagnosis using the data from the controller
- Prototype implementation for Linux kernel version 3.20, OVS and QEMU



AT&T

# Mozart [SOSR 2016]

## System which allows hosts and switches in a data-center to coordinate on flow measurements when certain events happen

- Example: when an end-host detects losses for a flow, it asks switches to collect more information about the flow to detect where the loss is occurring
- Two parts of the system:
  - Coordination algorithms
  - Placement algorithms to maximize tasks that can be run

# Measurement and Monitoring

Verification

# Anteater [Sigcomm 2011]

## Verifies that data-plane satisfies certain invariants

- Example invariants: loop-free forwarding, absence of black-holes, …
- Collects forwarding tables from network functions
  - Represents forwarding table snapshots and invariants as a SAT forumale
    - Solves the SAT to verify invariants
- Implementation as 3,500 lines of C++, Ruby and sed/awk scripts
  - Uses Boolector to solve SAT queries

# Header Space Analysis [NSDI 2012]

## Verifies data-plane satisfies properties

- Example properties: reachability, loop-free paths, isolation, …
- Methodology:
  - Model packet header as a point in $\{0, 1\}^L$ space -- the header space
  - Model each networking box as a transformer of the header space
  - Check for properties
- Prototype in Python

# VeriFlow [HotSDN 2012]

## Verifies network-wide invariants in real-time in SDN

- Checks network-wide invariants in real-time before SDN controller installs rules in switches
- Acts as a shim layer between SDN controller and network devices to check invariants
- Two key ideas:
  - Divides network into equivalence classes of traffic
    - Each class represents a distinct forwarding behavior within the network
  - Model the forwarding behavior of each class using a forwarding graph
- Prototype implementation as a proxy that sits between an OpenFlow controller and OpenFlow switches



AT&T

# Monocle [CoNext 2015]

An active probing tool that allows operators to verify that the data-plane corresponds to the view that an SDN controller installs via the control-plane

- Probes are generated through a SAT formulation to target specific rules in the switch forwarding tables
- Quickly adapts to changes to the rules

# Measurement and Monitoring

## Leveraging SDN/OpenFlow Switches for Measurements

# OpenTM: Traffic Matrix Estimation in an OpenFlow Network [PAM 2010]

## Definition of Traffic Matrix (TM)

- Amount of traffic flowing between every Origin-Destination (OD) pair in a network
  - OD pair = source-destination IP pair, router-pair, …

## Usage of traffic matrix

- Trouble-shooting: anomaly detection
- Performance improvement: load-balancing,
- Design and planning: capacity planning

## No direct way of measuring traffic matrix in traditional IP networks

- Several methods to estimate TM based on measurements available from routers
  - Link utilization
  - Flow information



AT&T

# How OpenTM Measures Traffic Matrix

## OpenTM leverages OpenFlow switches to measure TM

- OpenFlow switches keep track of number of bytes and packets directed using every flow rule
- OpenFlow controller allows determination of flow paths

## OpenTM methodology

- Determine TM periodically
  - Interval is kept fixed
- Every interval, determine …
  - all active flows in the network
  - path(s) of every flow
  - traffic flowing (number of bytes and packets) through a least loaded switch along the path of every flow
    - Use an adaptive algorithm to determine least-loaded switch for every flow
  - traffic flowing between every switch pair

## Prototype implementation

- C++ application for the NOX OpenFlow controller



AT&T

# Other Measurement Platforms that use SDN/OpenFlow Switches

FlowSense [PAM 2013]

- Estimate link utilization in a complete passive way

OpenNetMon [NOMS 2014]

- Monitor per-flow metrics such as throughput, delay and jitter

iSTAMP [Infocom 2014]

- Measure fine-grained traffic flow metrics

SLAM [PAM 2015]

- Determine latency distribution across every path

DREAM [Sigcomm 2014]

- Perform concurrent measurement tasks using TCAM counters on switches
  - Supported tasks: heavy hitter detection, hierarchical heavy hitter detection and change detection

SCREAM [CoNext 2015]

- Perform concurrent measurement tasks using sketch-based measurement tasks (in SRAM) of switches
  - Supported tasks: heavy hitter detection, hierarchical heavy hitter detection and super source/destination

OFRewind [Usenix ATC 2011]

- Infrastructure for recording and replaying control and data-plane traffic

# Measurement and Monitoring

## Leveraging End-Hosts in Data-centers for Measurements

# HONE [Journal of Network and Systems Management 2015]

## Platform for joint HOst-NEtwork (HONE) traffic management in SDN

- System components:
  - A controller that provides programming framework (for users) and run-time system which coordinates with hosts and switches to perform measurements, and merges data for final results
  - Host agents that perform measurement tasks and stream data back to aggregators or controllers
  - Network controller which programs switches and collects required data
- Prototype implementation in C, Python and as a custom module in FloodLight SDN controller
- Github repository at https://github.com/pupeng/hone



AT&T

# Other Proposals that use End-hosts

## Trumpet [Sigcomm 2016]

- Monitor every packet passing through hypervisor at every end-host for detecting events of interest
  - An event definition language for users to specify events of interest
- Describes three use-cases:
  - Pacing traffic
  - Automatically identifying flows responsible for transient congestion
  - Detecting services whose combined volume exceeds a threshold

## Felix [SOSR 2016]

- System for using end-hosts instead of switches for measurement tasks in an SDN
  - Measurements tasks are specified for end-hosts using a declarative query language
  - Measurement data collected at end-hosts is then aggregated into a final value at a centralized controller

# Causeway [Middleware 2005]

## Allows passing of meta-data between threads of a multi-threaded application during run-time

- The meta-data is passed during so-called transfer points or channels, such as sockets, pipes, context-switches between user and kernel-space and shared memory
  - For system-visible channels (e.g., sockets, pipes), Causeway requires instrumentation of kernel or system libraries
  - For system-opaque channels (e.g., shared memory), Causeway requires instrumentation of the application
- Applications written on top of Causeway (called meta-applications) use Causeway to pass meta-data around to fulfill their objectives
  - An example application which implements priority-based scheduling of request processing in a three-tiered web application called TPC-W
- Implementation in FreeBSD 5.1

# X-Trace [NSDI 2007]

## A framework that attaches meta-data to packets as they flow up and down protocol-stack and across the network

- Meta-data includes a unique task-id for packets generated in response to each task
  - For example, user issuing a request to access a web-page
- Meta-data reports sent to a centralized location for constructing task trees, which in turn can be used for network management tasks
- Requires modification to protocol software
- Project home-page: http://www.x-trace.net/wiki/doku.php



AT&T

# ProgMe [Sigcomm 2007]

## A programmable measurement architecture based on the concept of a *flowset*

- Flowset: set of arbitrary flows defined by application requirements and/or traffic conditions
- Consists of two components:
  - Flowset Composition Language (FCL) which allows applications/users to specify flowsets
  - Flowset-based Query Answering Engine (FQAE) that answers user queries about flowsets
- ProgME extension called adaptive multi-resolution tiling (MRT) algorithm that can iteratively re-program itself to identify heavy hitters



AT&T

# Frenetic [Presto Workshop 2010]

## A High-Level Language for managing OpenFlow Networks

- Declarative language that allows users to specify forwarding and monitoring rules for a network of OpenFlow switches
  - Contains high-level packet-processing operators inspired by FRP (Functional Reactive Programming)
    - FRP: a model in which programs manipulate streams of values
  - Example: determine amount of web-traffic arriving on one of the ports of a two-port switch

```
def monitor_sf():
    return (Filter(inport_p(2) & srcport_p(80)) |o| GroupByTime(30) |o| SumSizes()
def monitor():
    stats = Apply(Packets(), monitor_sf())
            print_stream(stats)
```

- Prototype implementation in Python that runs on top of NOX OpenFlow controller
- Frenetic web-site: http://frenetic-lang.org/



AT&T

# Summary

# The shift to software is a BIG deal

## Network operators are moving towards software-based networks

- NFV, SDN and Cloud are the underpinnings of the new network

## Opens up new challenges and opportunities

- Many of the traditional data collection and processing still applicable, but
- Advanced ideas can actually be tried out because of use of software!

## Lots of focus has been on (virtual) switch layer and at controller

- Not as much focus on what can be done with network functions themselves
- Lots of opportunity!!!
- But need to balance measurement with performance impact



AT&T