Virtual Filtering Platform

A retrospective on 8 years of shipping Host SDN in the Public Cloud

Daniel Firestone

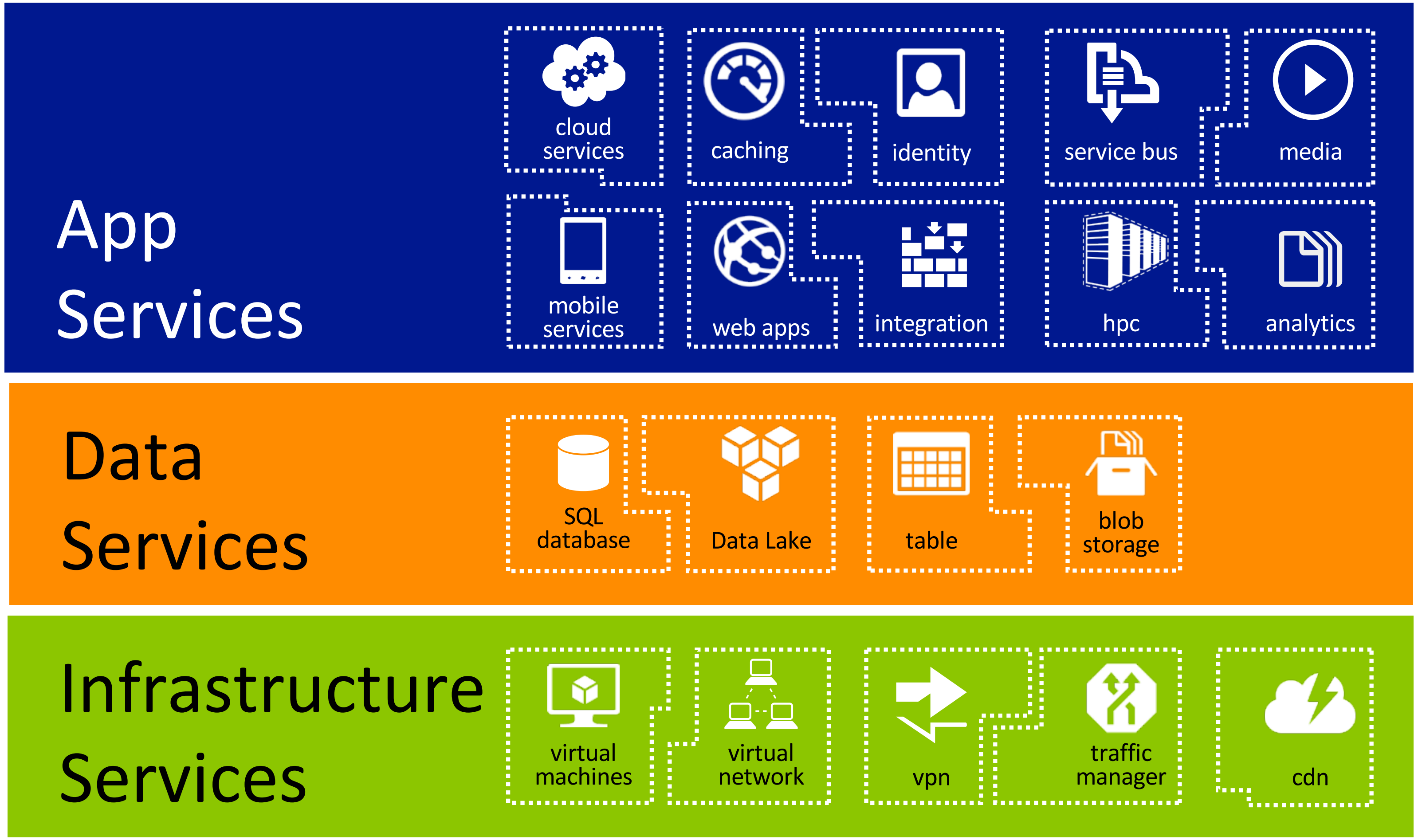
Tech Lead and Manager, Azure Networking Host SDN Team

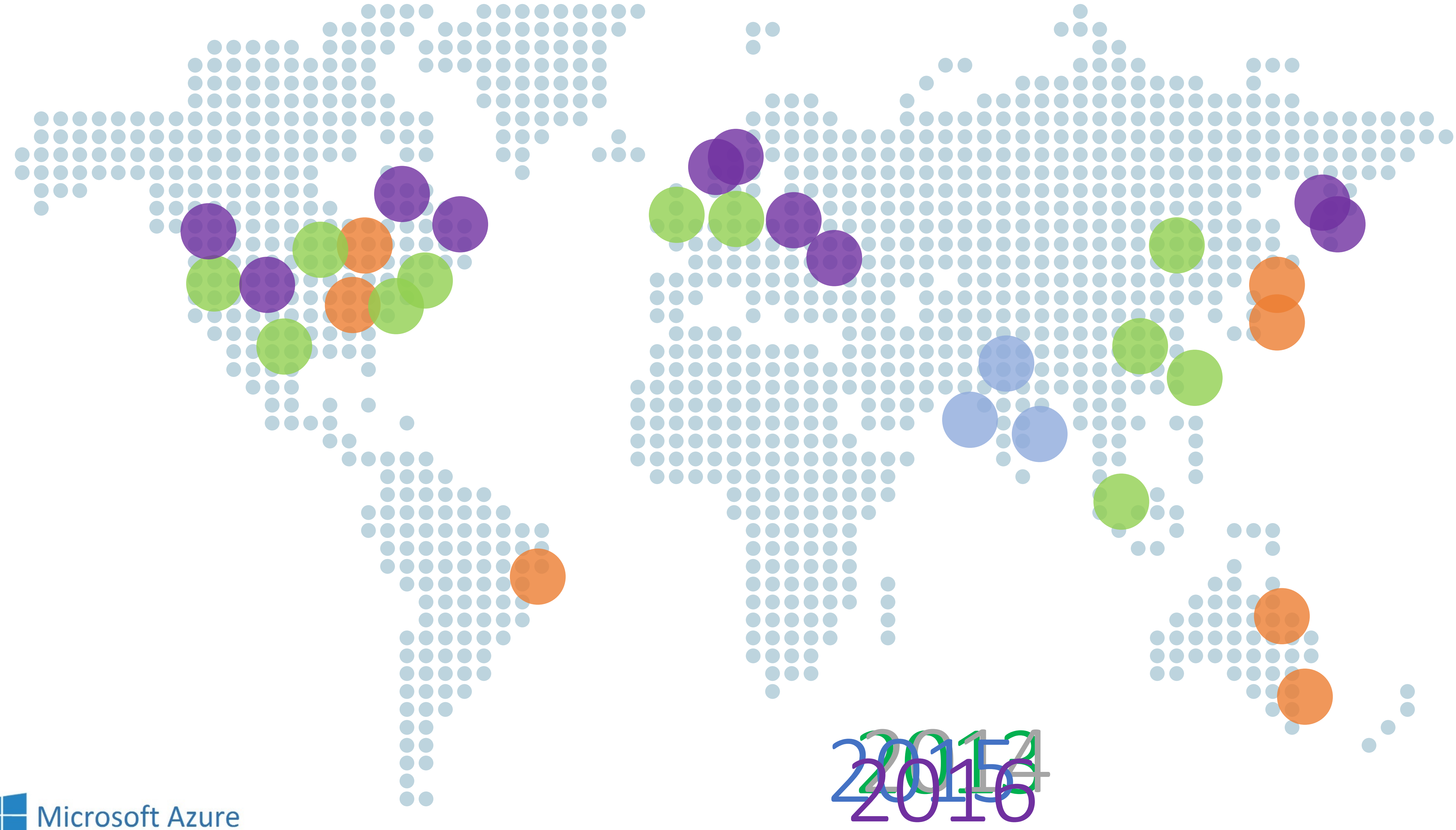
Microsoft Azure

Overview

- Azure and Scale
- Why Virtual Switches for SDN?
- Early implementations of Azure Host SDN
- Azure Host SDN Platform Goals
- VFP – Our platform for host SDN
- VFPv2 – Addressing Challenges of Scale
- Hardware Offloads
- Conclusion and Future

Microsoft Azure





Microsoft Azure

	2010	2017
Compute Instances	100K	Millions
Azure Storage	10's of PB	Exabytes
Datacenter Network	10's of Tbps	Pbps

Microsoft Azure

Azure Scale & Momentum

>85%
Fortune 500 using Microsoft Cloud

>120,000
New Azure customers a month

>9 MILLION
Azure Active Directory Orgs

>18 BILLION
Azure Active Directory authentications/week

> 3 TRILLION
Azure Event Hubs events/week

>60 TRILLION
Azure storage objects

>900 TRILLION
requests/day

1 out of 3 Azure VMs
are Linux VMs

>110 BILLION
Azure DB requests/day

Microsoft Azure

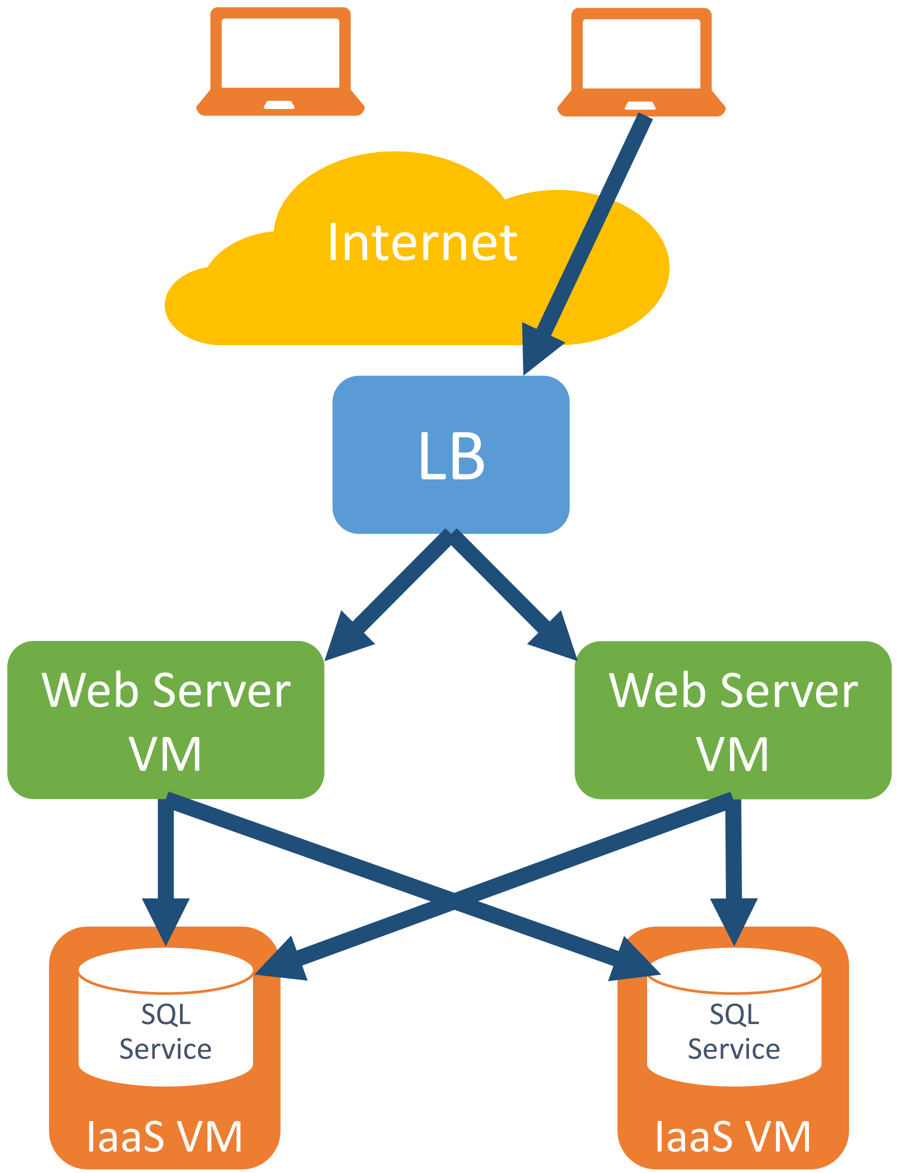
Why do we need Virtual Switch SDN Policy?

Policy application at the host is more scalable!

Example #1: LB (From Ananta, SIGCOMM '13)

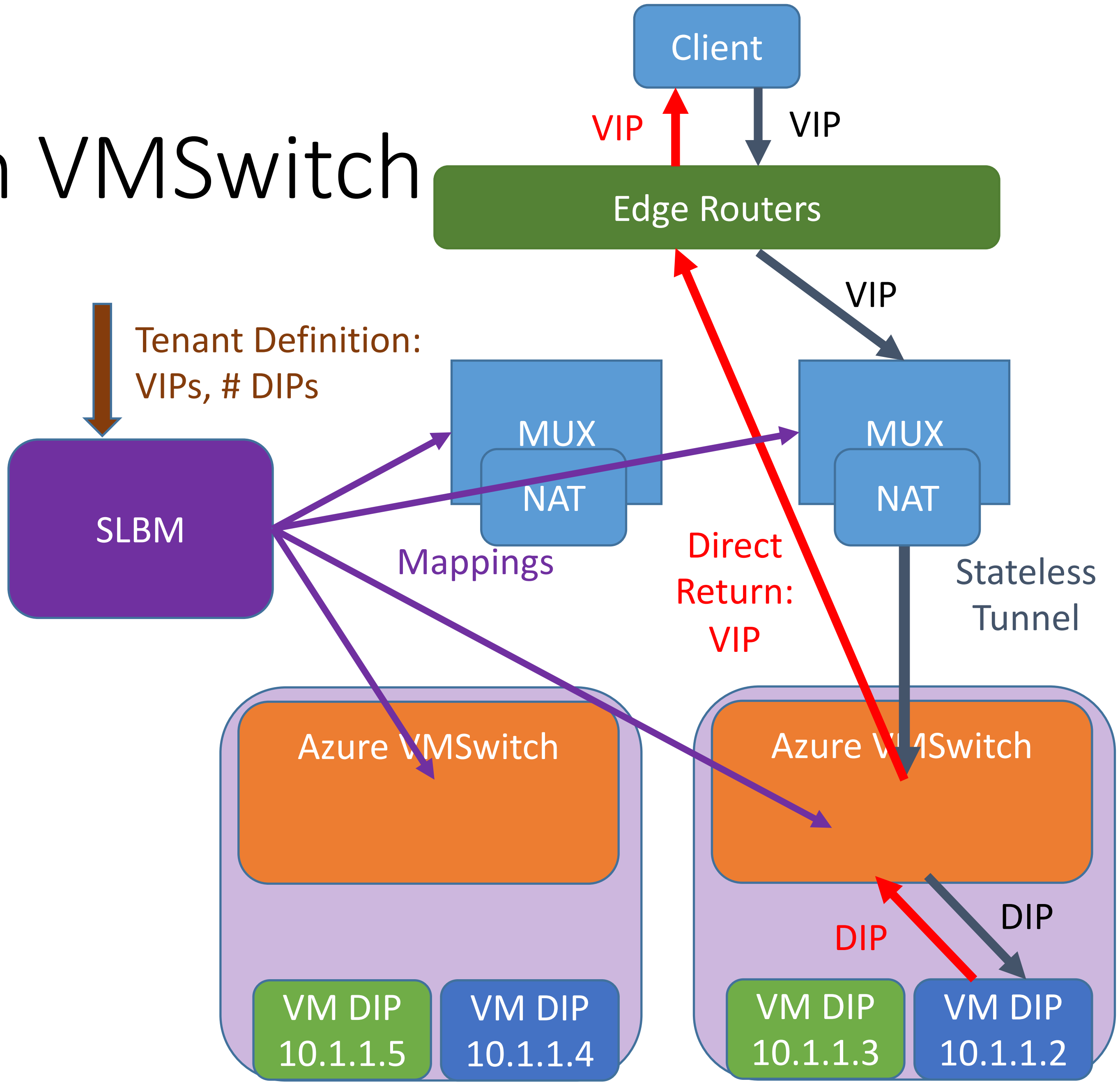
- All infrastructure runs behind an LB to enable high availability and application scale
- How do we make application load balancing scale to the cloud?
- Challenges:
  - How do you load balance the load balancers?
  - Hardware LBs are expensive, and cannot support the rapid creation/deletion of LB endpoints required in the cloud
  - Support 10s of Gbps per cluster
  - Need a simple provisioning model

Microsoft Azure



“SDN” Approach: Software LB with NAT in VMSwitch

- Goal of an LB: Map a Virtual IP (VIP) to a Dynamic IP (DIP) set of a cloud service
- Two steps: Load Balance (select a DIP) and NAT (translate VIP->DIP and ports)
- Pushing the NAT to the vswitch makes the MUXes stateless (ECMP) and enables direct return
- Single controller abstracts out LB/vswitch interactions

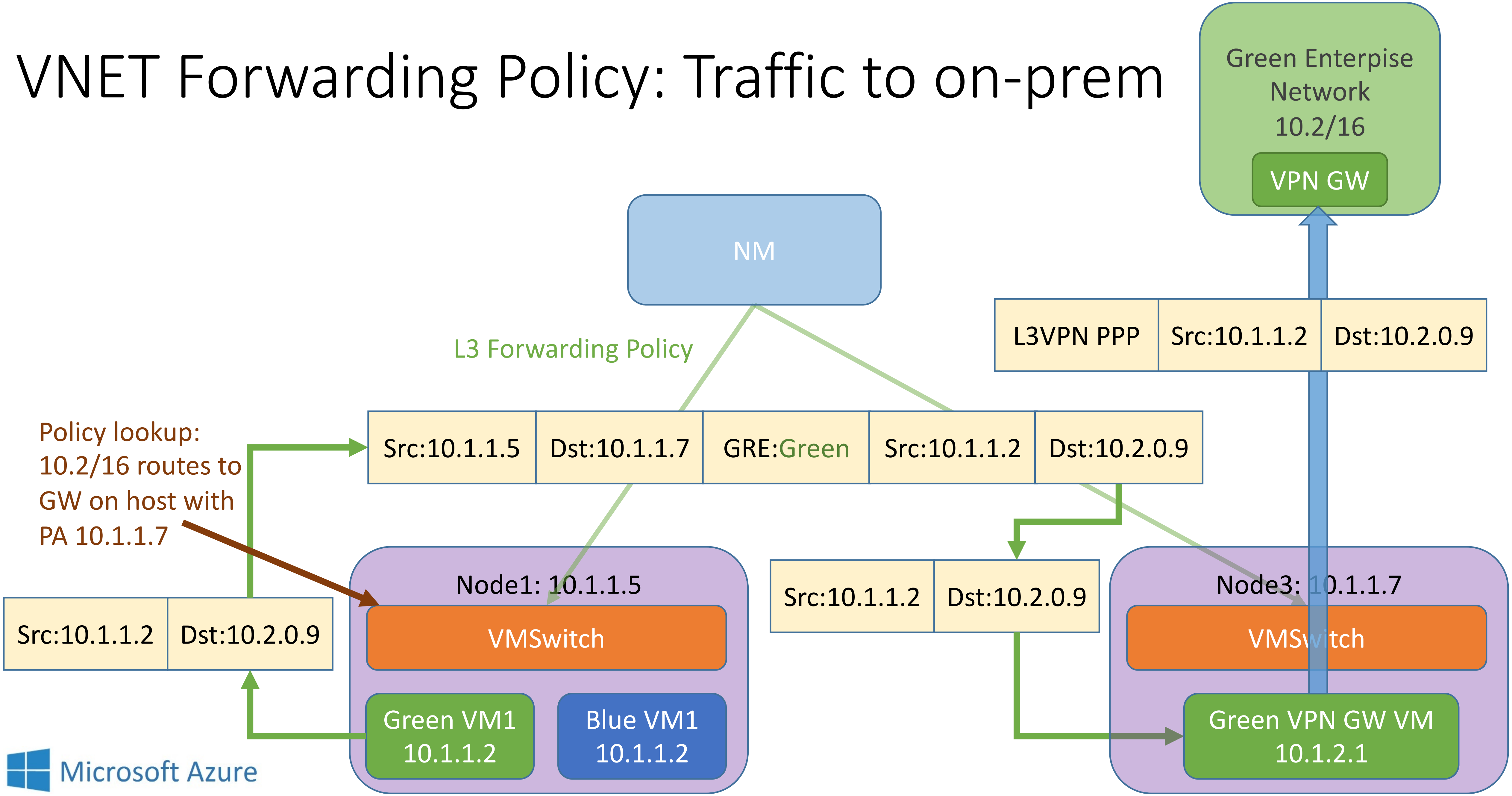


Microsoft Azure

Example #2: Vnet

- Ideas from VL2 (SIGCOMM '09)
- Goal is to map Customer Addresses (e.g. BYO IP space) to Provider Addresses (real 10/8 addresses on the physical network)
- This requires a translation of *every* packet on the network – no hardware device on our network is scalable enough to handle this load along with all of the relevant policy
- Enables companies to create their own virtual network in the cloud, defining their own topologies, security groups, middleboxes and more

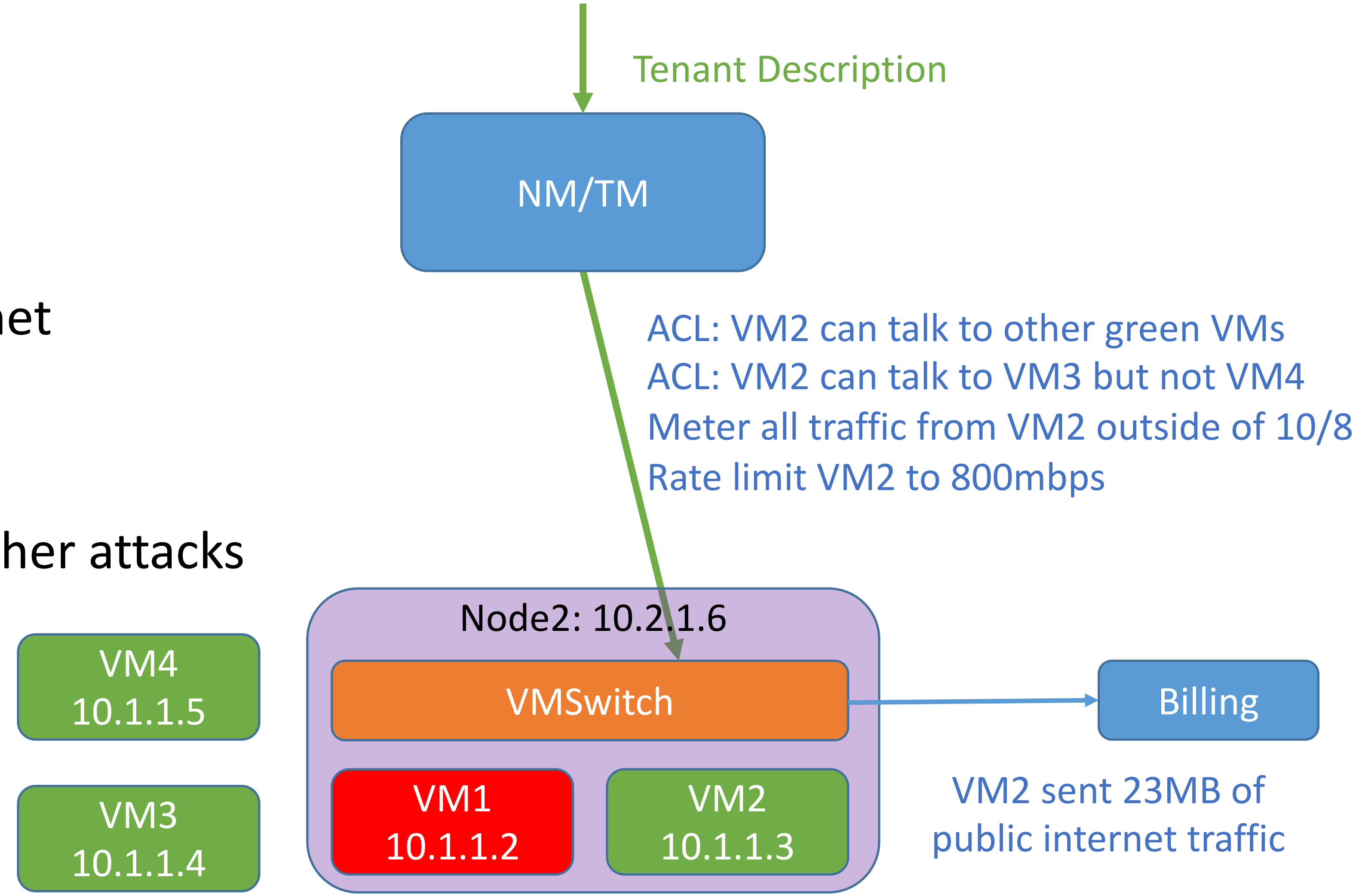
VNET Forwarding Policy: Traffic to on-prem



Microsoft Azure

Even More VSwitch…

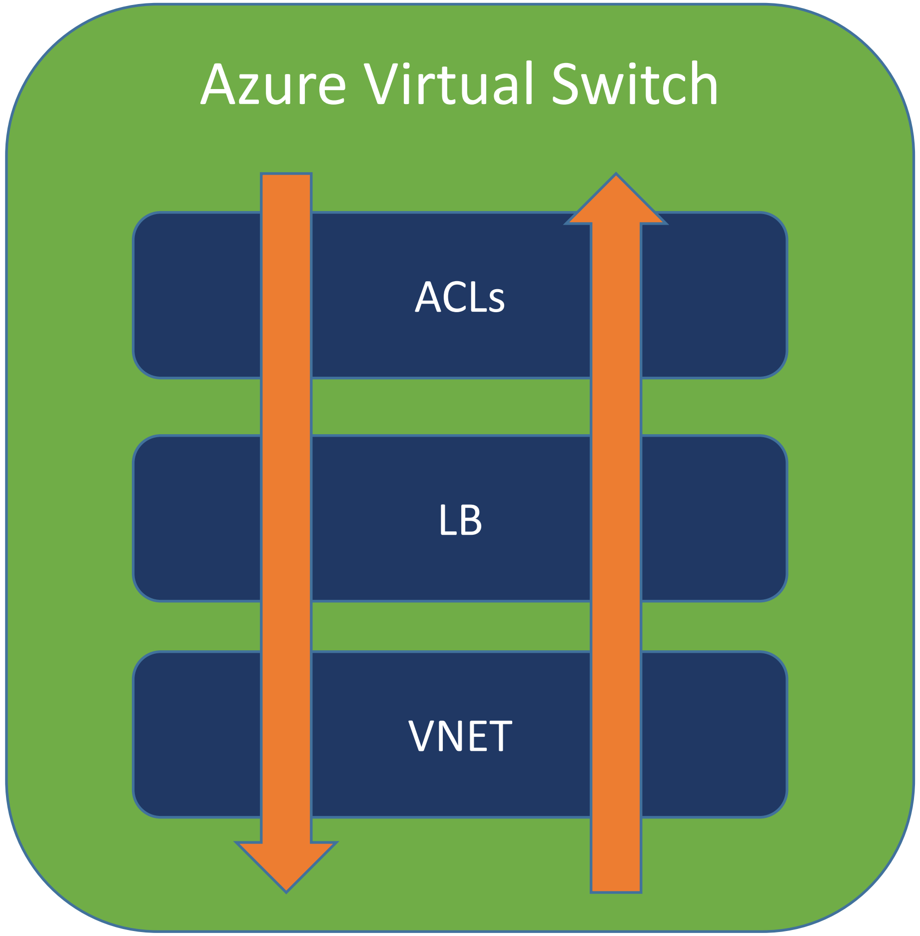
- 5-tuple ACLs
  - Infrastructure Protection
  - User-defined Protection
- Billing
  - Metering traffic to internet
- Rate limiting
- Security Guards
  - Spoof, ARP, DHCP, and other attacks
- More in development all the time…



Microsoft Azure

Early Approach to Azure Vswitch (2009-2011): Stacked SDN drivers per app

- Each SDN application is a driver module hard compiled into the vswitch, handling packets on its own
- Changes to SDN policy require kernel space changes, and an OS update
- Was revolutionary for us in shipping LB and VNET and Host SDN – but not easy to add new SDN Apps
- After a couple of years we decided we needed a more flexible Host SDN platform



Overview

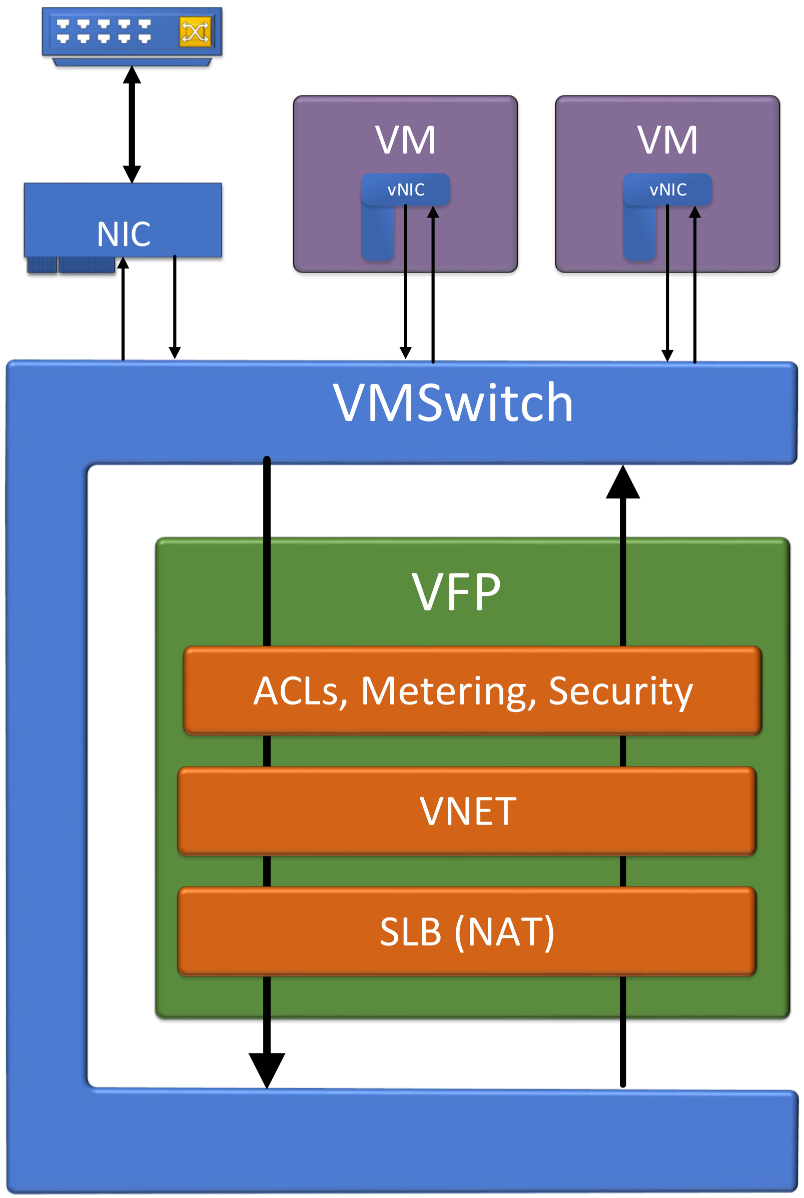
- Azure and Scale
- Why Virtual Switches for SDN?
- Early implementations of Azure Host SDN
- Azure Host SDN Platform Goals
- VFP – Our platform for host SDN
- VFPv2 – Addressing Challenges of Scale
- Hardware Offloads
- Conclusion and Future

Original Goals for Azure Host SDN Platform

- **Goal 1:** Provide a programming model allowing for multiple simultaneous, independent network controllers to program network applications, minimizing cross-controller dependencies
- **Goal 2:** Provide a MAT programming model capable of using connections as a base primitive, rather than just packets – stateful rules as first class objects
- **Goal 3:** Provide a programming model that allows controllers to define their own policy and actions, rather than implementing fixed sets of network policies for predefined scenarios

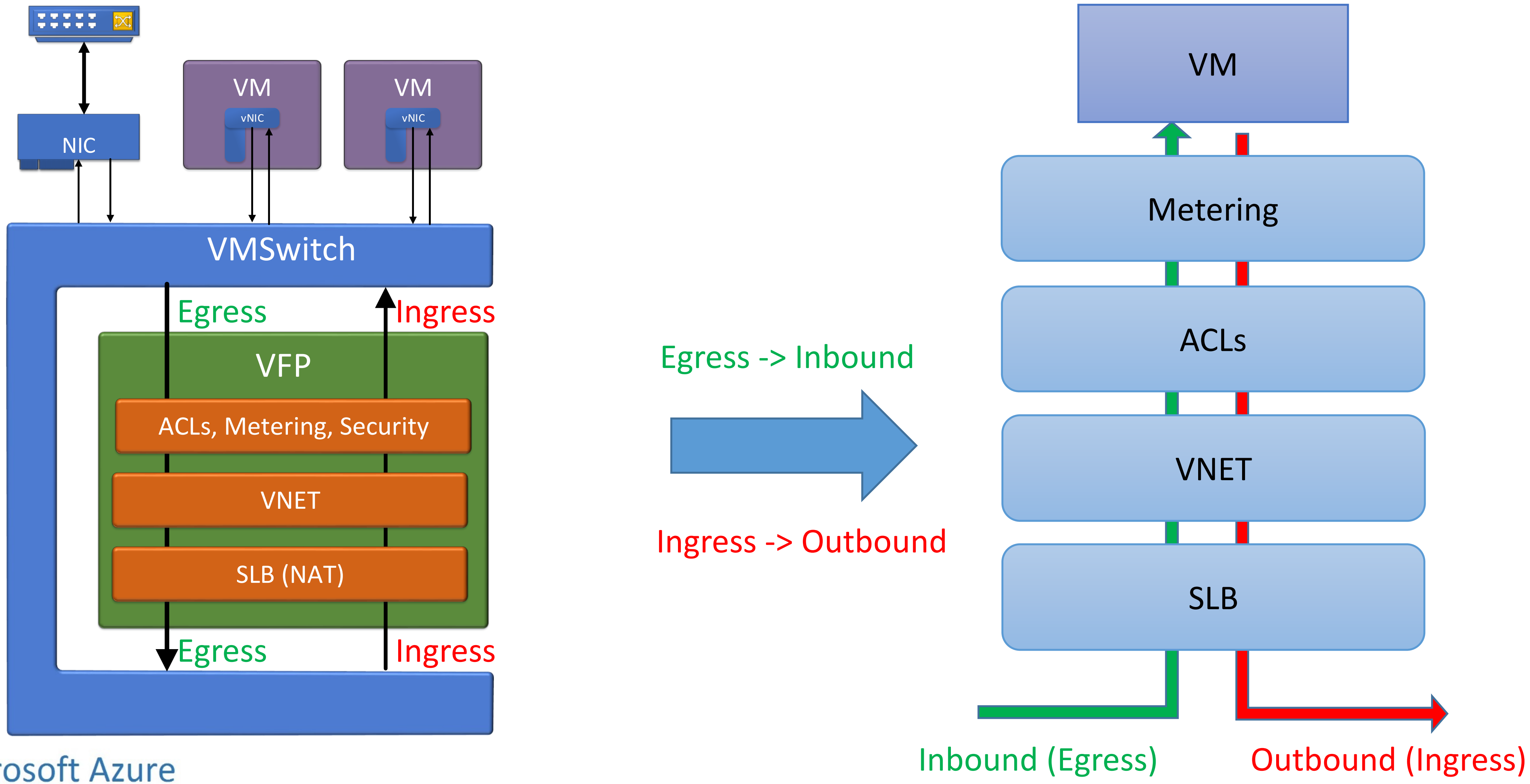
What is VFP?

Virtual Filtering Platform (VFP)
Azure's SDN Dataplane



- Plugin module for WS2012+ VMSwitch
- Provides core SDN functionality for Azure networking services, including:
  - Address Virtualization for VNET
  - VIP -> DIP Translation for SLB
  - ACLs, Metering, and Security Guards
- Uses programmable rule/flow tables to perform per-packet actions
- Programmed by multiple Azure SDN controllers, supports all dataplane policy at line rate with offloads

VFP Translates L2 extensibility (ingress/egress to switch) to L3 extensibility (inbound/outbound to VM)



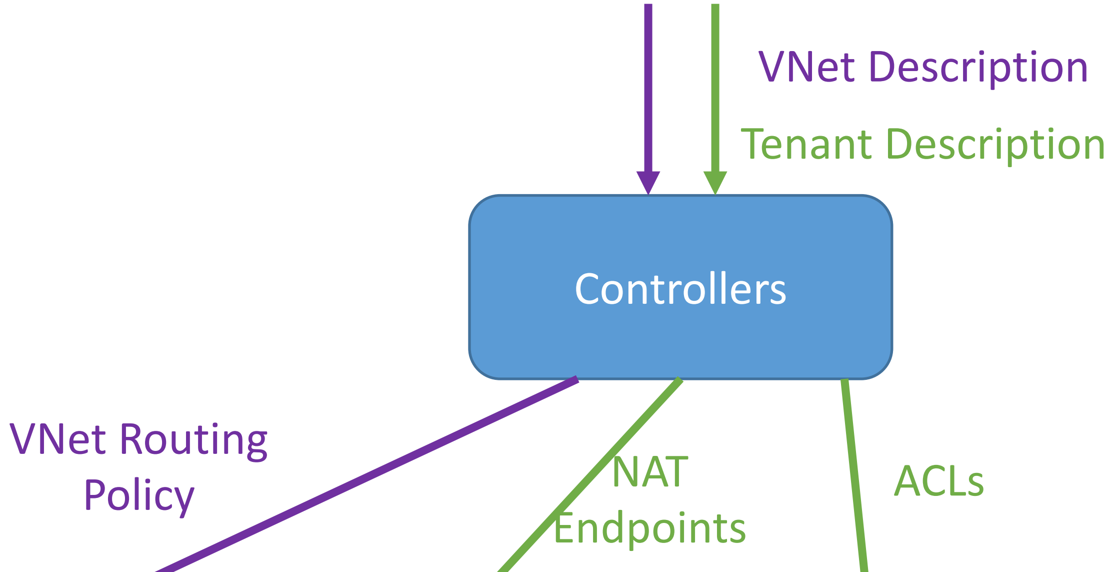
Microsoft Azure

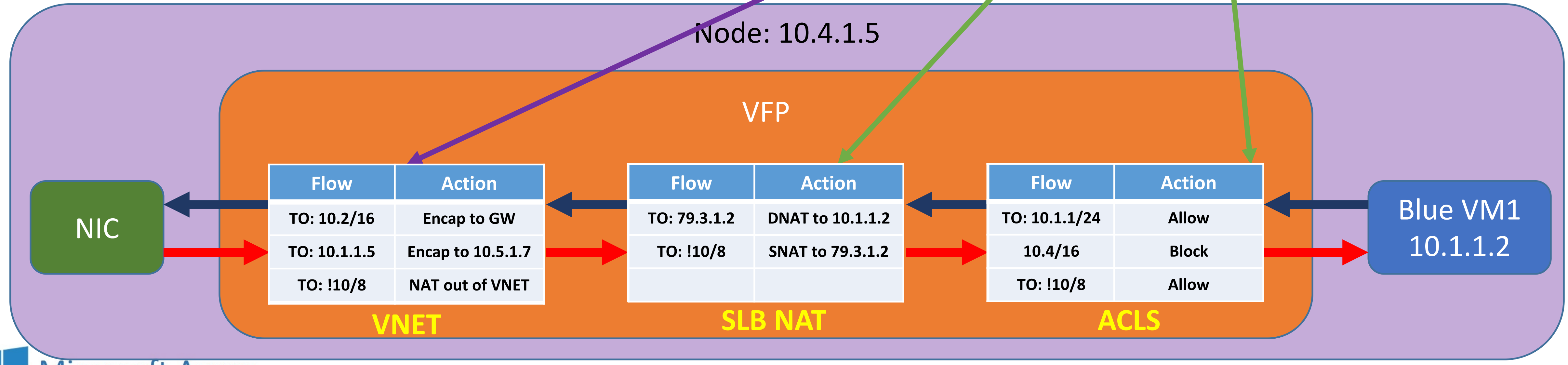
Goal: All Policy is in the Controller - VFP is a Fast, Flexible Implementation of Policy

- To enable agility, allow controllers to specify exactly what they want to do at the flow/packet level, so they can implement new SDN scenarios without dataplane driver changes
- VFP focuses on integrating multi-controller policies and scaling the host dataplane – perf and offloads without sacrificing flexibility
- 3 Key Primitives we expose to controllers:
  - **Layers** – independent flow tables per controller to order the pipeline
  - **Rule Matches** – define which packets match which rule
  - **Rule Actions** – what to do with a packet for a given rule

Key Primitive: Match Action Tables

- VFP exposes a typed Match-Action-Table API to the agents/controllers
- One table (“**Layer**”) per policy
- Inspired by OpenFlow and other MAT designs, but designed for multi-controller, stateful, scalable host SDN applications

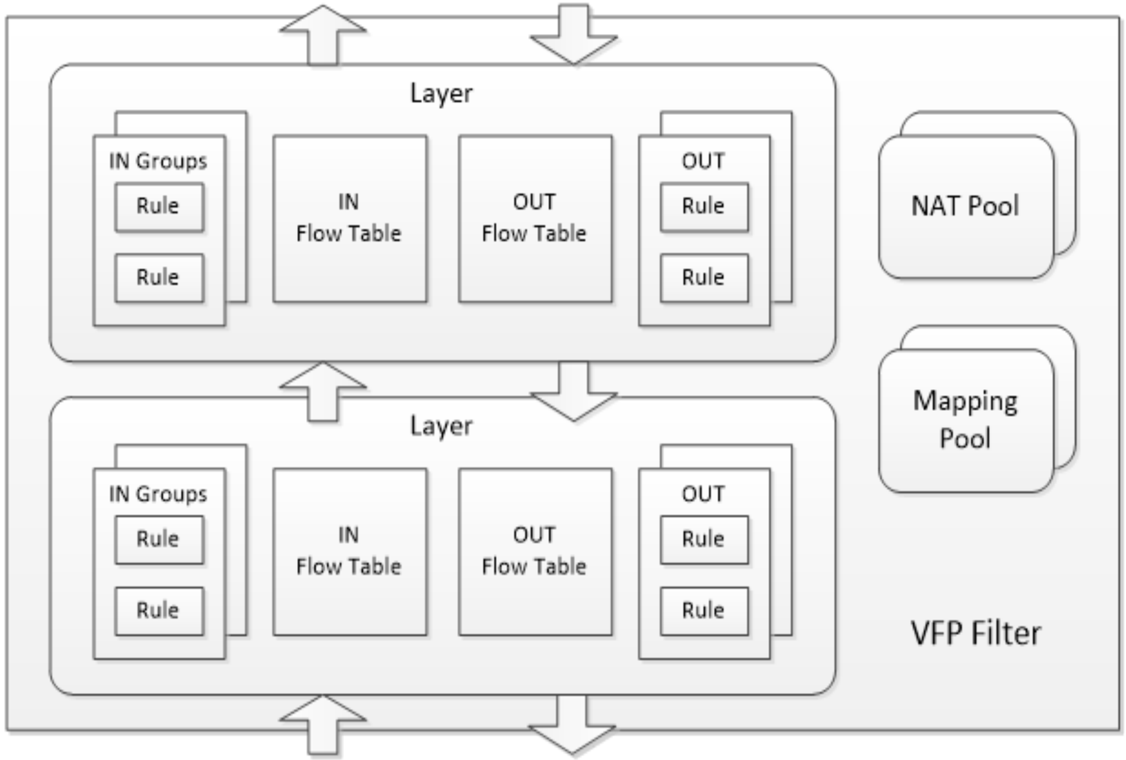




Microsoft Azure

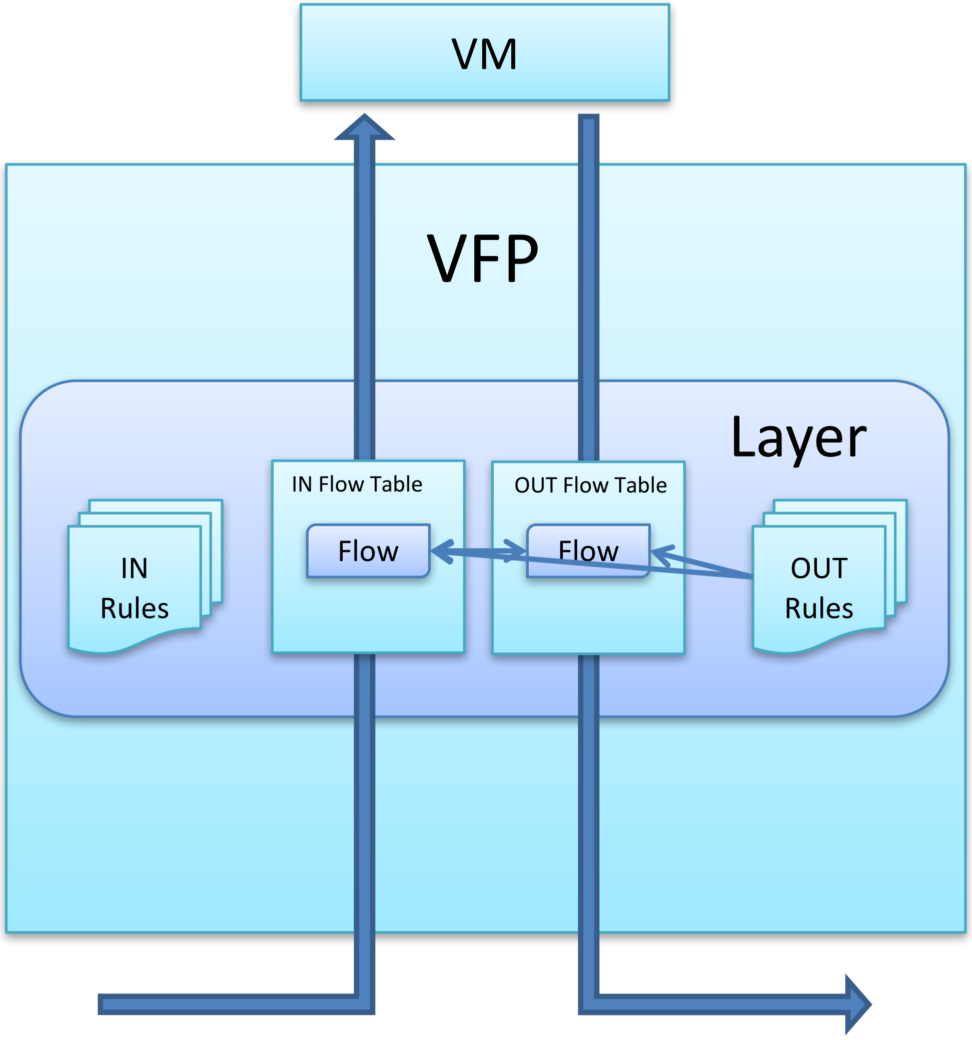
Layers

- A VFP layer is not a built-in function – it is a generic set of rule/flow tables
- Any layer can be created at any time – it is only an “LB layer” or a “VNET layer” based on what rules are plumbed into it
- Resources like NAT pools or PA->CA mapping pools are available to any layer to implement special functionality (e.g. SLB or VNET)

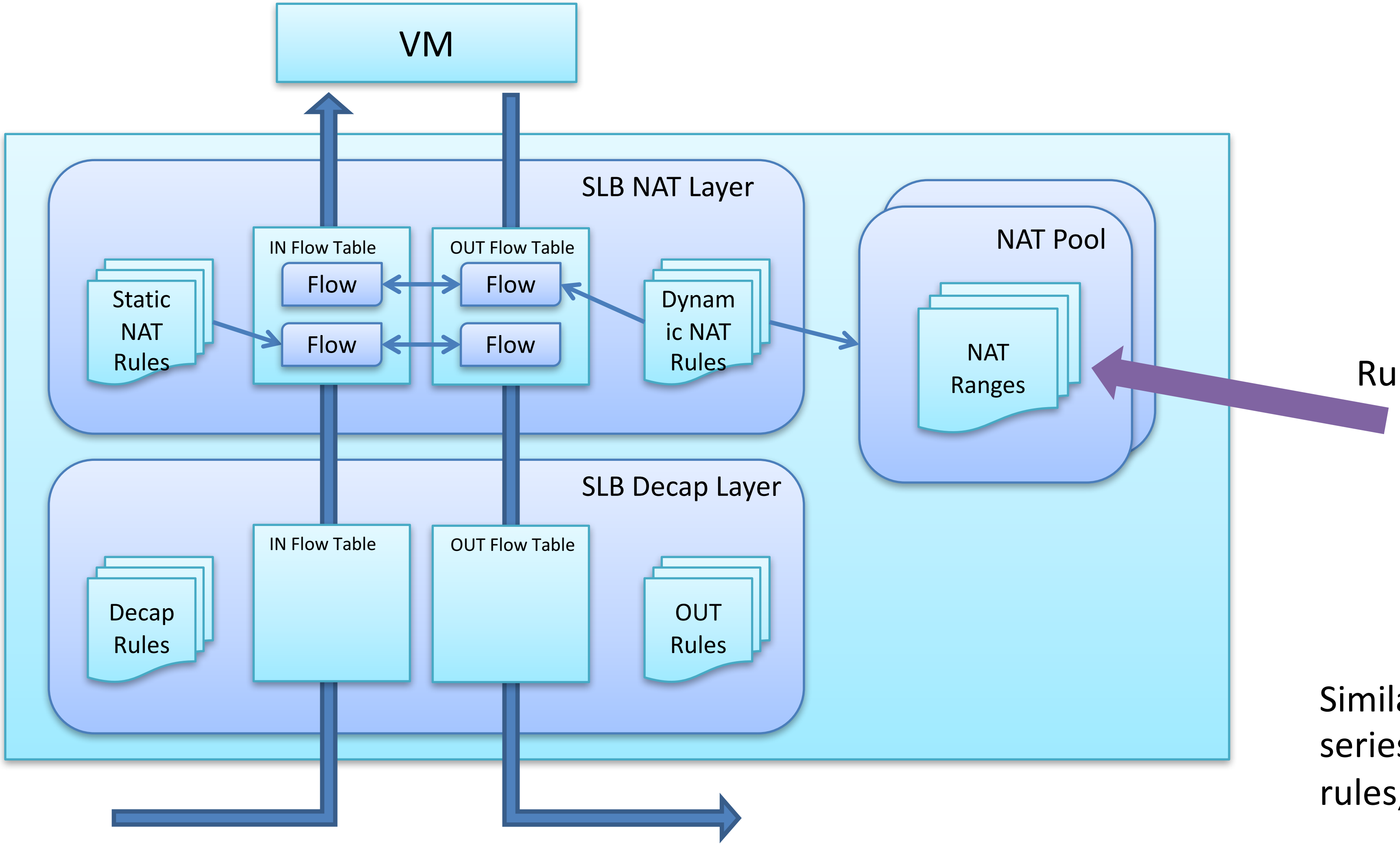


Everything is Stateful

- The core primitive of most policy is a (TCP, UDP, …) connection – translates to a two-way flow
- 5-tuple ACLs, VIP-DIP SLB NAT, dynamic outbound SNAT, and more
- Stateful rules make it easy to reason about asymmetric policy – rules apply to whichever side started the flow, and the reverse happens automatically for the other direction
- Flow state managed by TCP connection tracker



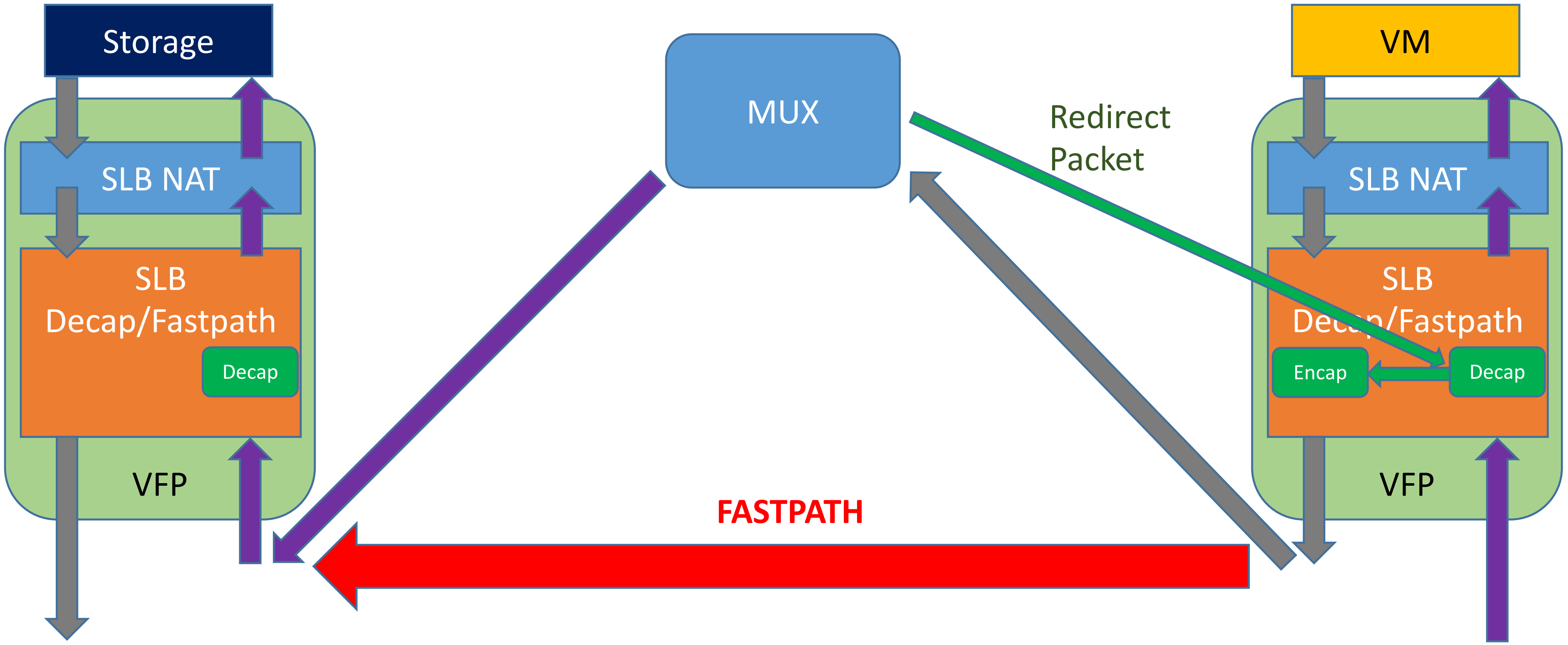
Example: Software LB Support



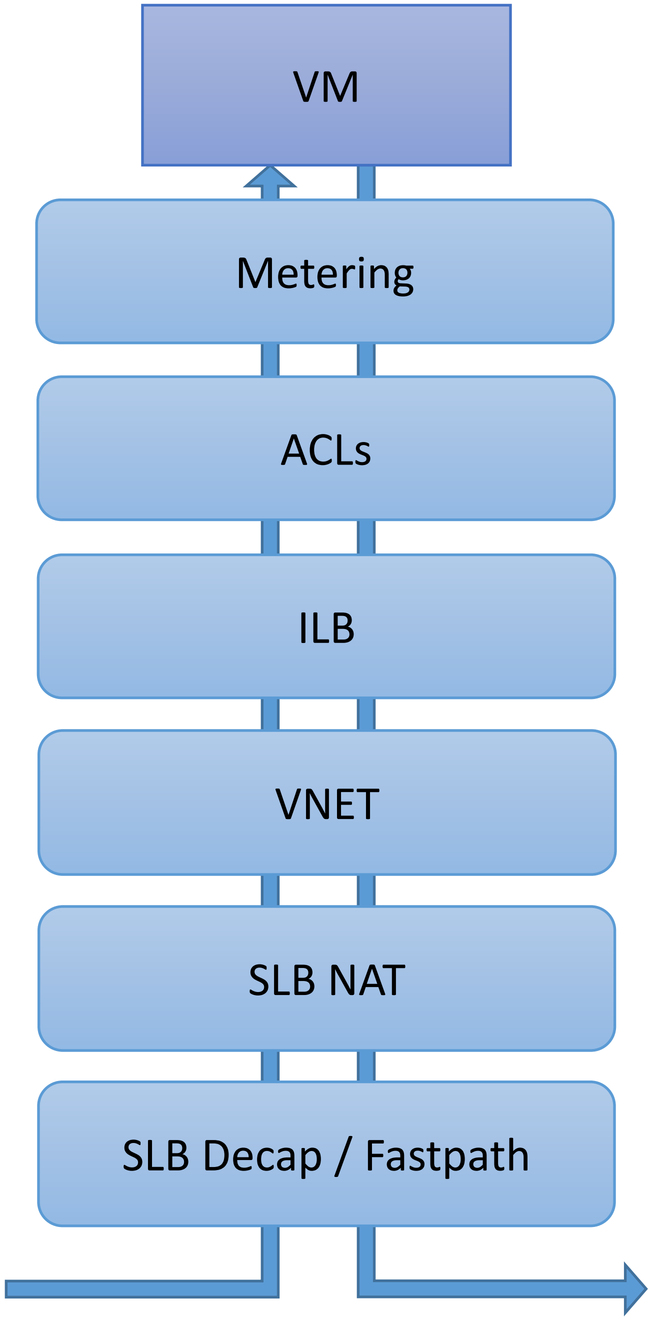
Rules can reference **Resources**, like dynamic NAT pools or PA-CA mapping tables

Similarly, VNET can be expressed as a series of (encap, decap, rewrite, etc) rules, rather than fixed policy

Cool Uses of Stateful Flows – LB Fastpath



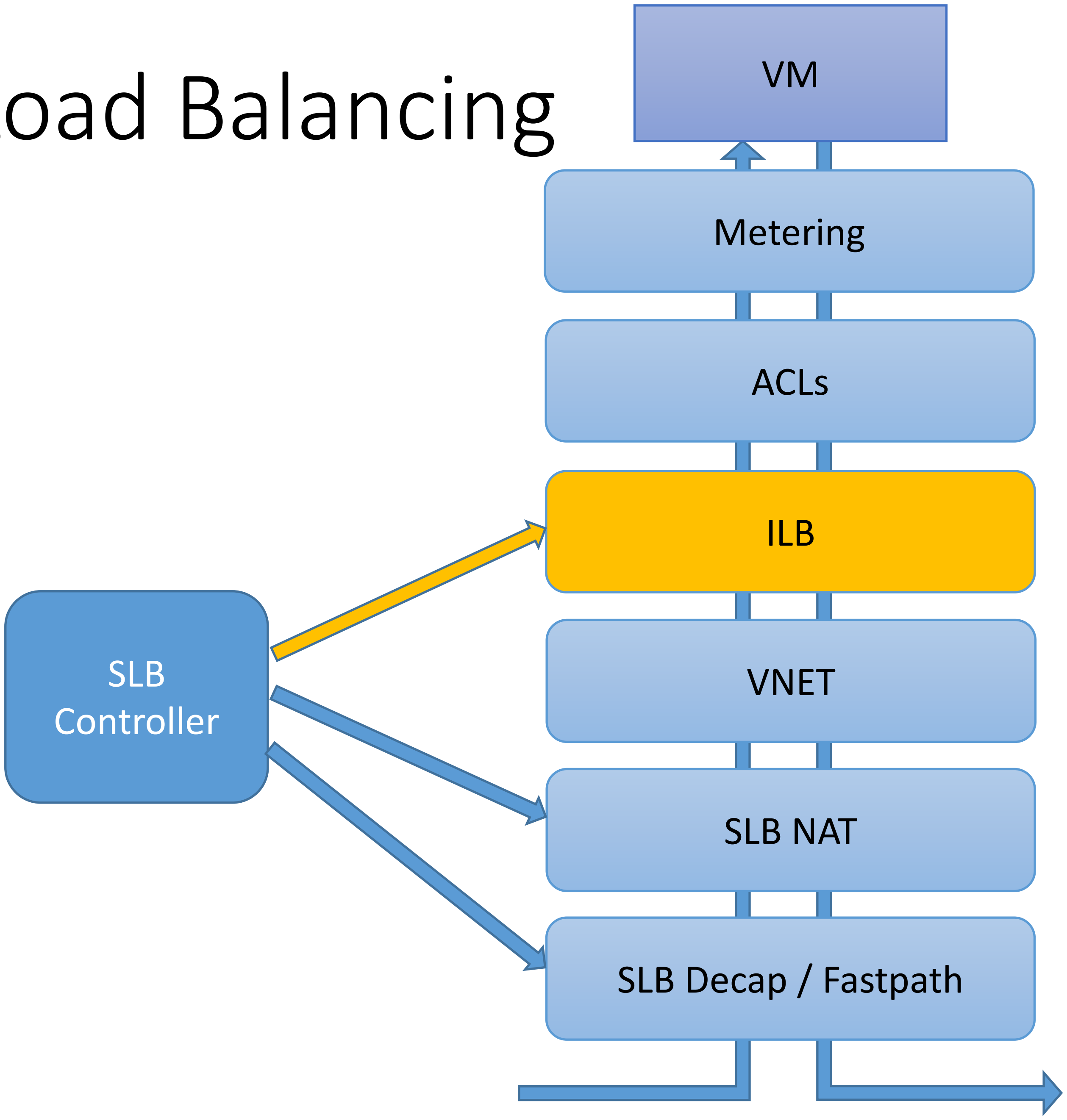
Example VFP Layers:
Support for LB, VNET, Security Groups, and Billing



Successfully deployed across Azure in 2012

Agility Example: Internal Load Balancing

- LB team wanted to offer CA-space LB in addition to PA-space LB
- All they had to do was create a new layer – added new policy by specifying CA-space rule matches for NAT rules
- **No new work in VFP, because we picked the right primitives**

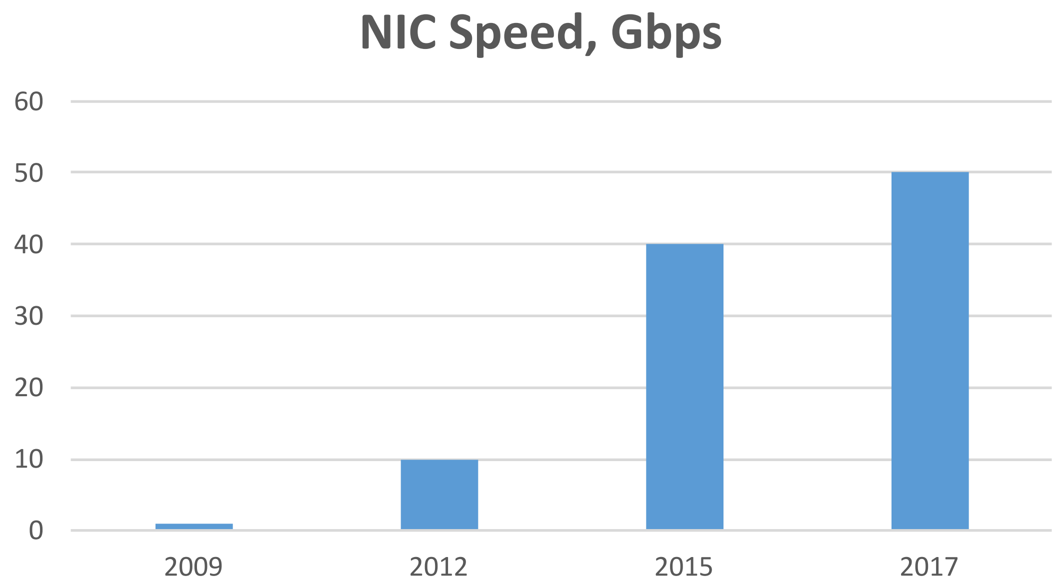


Overview

- Azure and Scale
- Why Virtual Switches for SDN?
- Early implementations of Azure Host SDN
- Azure Host SDN Platform Goals
- VFP – Our platform for host SDN
- VFPv2 – Addressing Challenges of Scale
- Hardware Offloads
- Conclusion and Future

Scaling Up SDN: NIC Speeds in Azure

- 2009: 1Gbps
- 2012: 10Gbps
- 2015: 40Gbps
- 2017: 50Gbps
- Soon: 100Gbps?



We got a 50x improvement in network throughput, but not a 50x improvement in CPU power!

Microsoft Azure

New Goals for VFPv2 (2013-2014)

- **Goal 4:** Provide a serviceability model allowing for frequent deployments and updates without requiring reboots or interrupting VM connectivity for stateful flows, and strong service monitoring

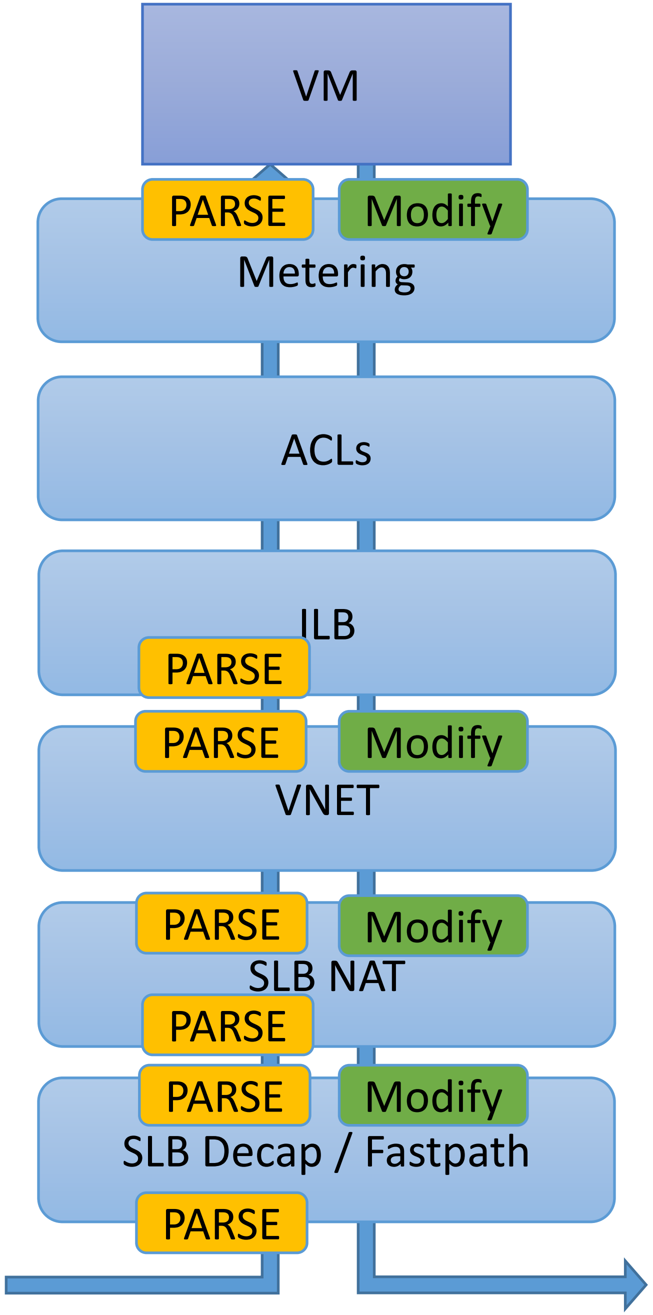
- **Goal 5:** Provide very high packet rates, even with a large number of tables and rules, via extensive caching

- **Goal 6:** Implement an efficient mechanism to offload flow policy to programmable NICs, without assuming complex rule processing

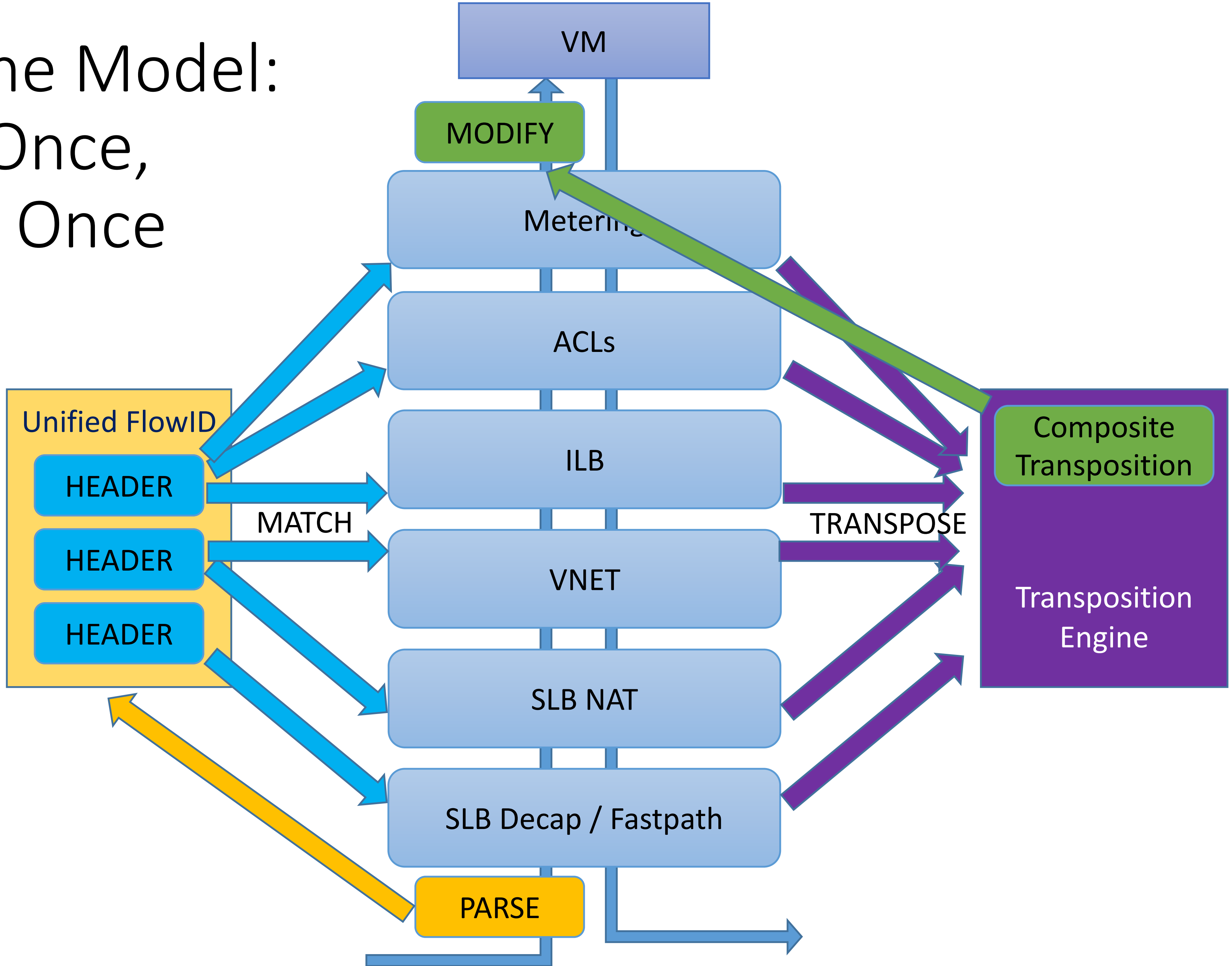
VFPv1 Layers - Challenges

- Holdover from original vswitch design – every layer independently handles, parses, and modifies packets
- Most of our layers want to be stateful – but this means independent connection tracking and flow state at each layer
- As host SDN became easy to program and widely used, people wanted to add new layers all the time
- Couldn't keep adding layers and scaling up

We need a better primitive for actions!



ASIC Pipeline Model: Parse Once, Modify Once



Shipped in 2014

Microsoft Azure

Header Transposition - Actions

Headers

Header	Parameters
Outer Ethernet	Source MAC, Dest MAC
Outer IP	Source IP, Dest IP
Encap	Encap Type, GRE Key / VXLAN VNI
Inner Ethernet	Source MAC, Dest MAC
Inner IP	Source IP, Dest IP
TCP/UDP	Source Port, Dest Port (note: does not support Push/Pop)

Header Actions

Action	Notes
Pop	Remove this header. No params supported.
Push	Push this header onto the packet. All params must be specified.
Modify	Modify this header. All params are optional, but at least one must be specified.
Ignore	Leave this header as is. No params supported.
Not Present	This header is not expected to be present (based on the match conditions). No params supported.

Header Transposition – Example Actions

Header	NAT	Encap	Decap	Encap+NAT	Decap+NAT
Outer Ethernet	Ignore	Push (SMAC,DMAC)	Pop	Push (SMAC,DMAC)	Pop
Outer IP	Modify (SIP,DIP)	Push (SIP,DIP)	Pop	Push (SIP,DIP)	Pop
GRE / VxLAN	Not Present	Push (Key)	Pop	Push (Key)	Pop
Inner Ethernet	Not Present	Modify (DMAC)	Ignore	Modify (DMAC)	Ignore
Inner IP	Not Present	Ignore	Ignore	Modify (SIP,DIP)	Modify (SIP,DIP)
TCP/UDP	Modify (SPt,DPt)	Ignore	Ignore	Modify (SPt,DPt)	Modify (SPort,DPt)

Allows rules to express more complex actions across headers

Unified Parsing and Matching

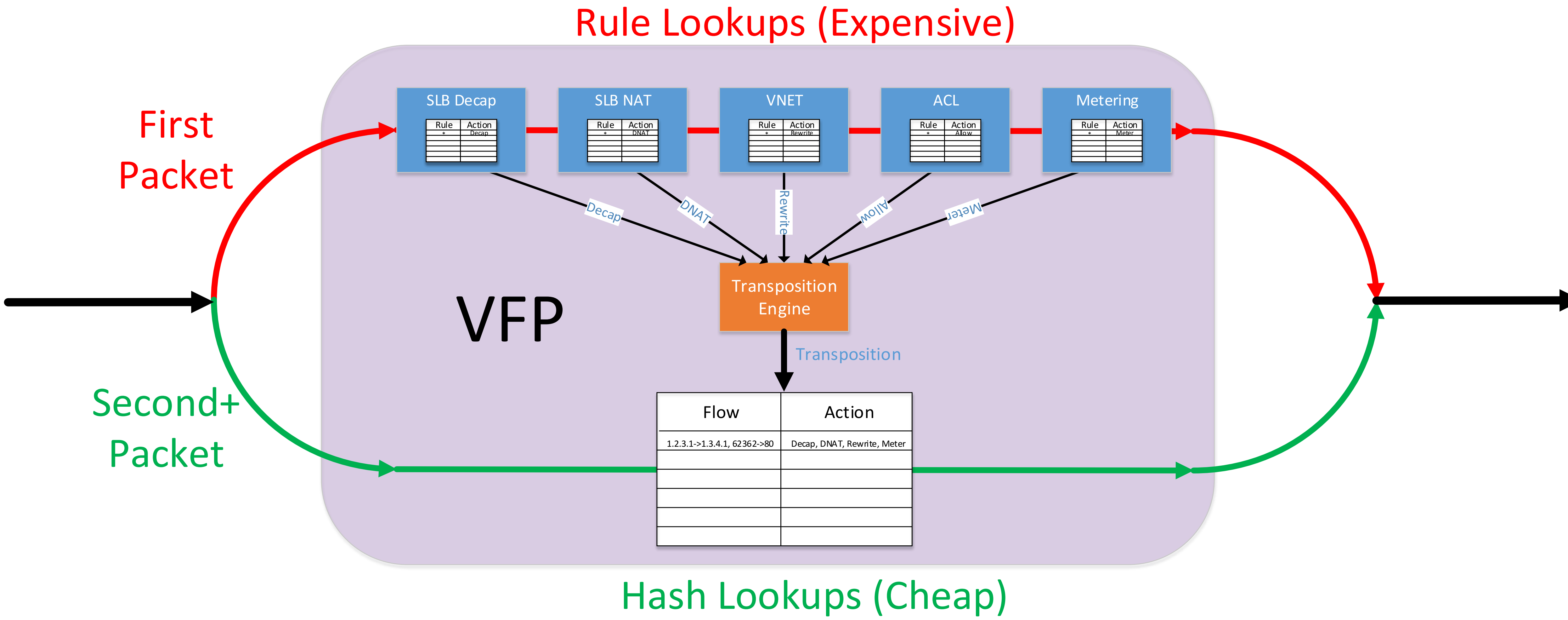
Condition	Notes
Source VPort	N/A
(Outer) Source MAC Address	N/A
(Outer) Destination MAC Address	N/A
(Outer) Source IP Address	IPv4 or IPv6
(Outer) Destination IP Address	IPv4 or IPv6
(Outer) IP Protocol	N/A
Source Port	Applies if Protocol == TCP or UDP
Destination Port	Applies if Protocol == TCP or UDP
ICMP Type	Applies if Protocol == ICMP (v4 or v6)
Destination Vport	N/A
GRE Key / VxLAN VNI (Tenant ID)	Applies if Outer Protocol == GRE / VxLAN
(Inner) Source MAC Address	N/A
(Inner) Destination MAC Address	N/A
(Inner) Source IP Address	IPv4 or IPv6
(Inner) Destination IP Address	IPv4 or IPv6
(Inner) IP Protocol	N/A

Microsoft Azure

Header Transpositions Complete our Generic Southbound API Capability Story

- In order to enable agility, we want controllers to be able to define new types of policy dynamically without needing to change VFP.
- We already provide flexibility in:
  - **Layers**: Controllers can define new layers dynamically for their own policy without interfering with other controllers' layers
  - **Rules:** Controllers can define which rules match which packets via a consistent 5-tuple match API, nothing specific to special policies
- **Header transpositions** provide the key third primitive: Ability to specify what exactly a rule does once it is matched
- All built in rules define HTs, but controllers can define their own rules by creating new ones out of HTs on the fly

Unified Flow Tables – A Fastpath Through VFP



Microsoft Azure

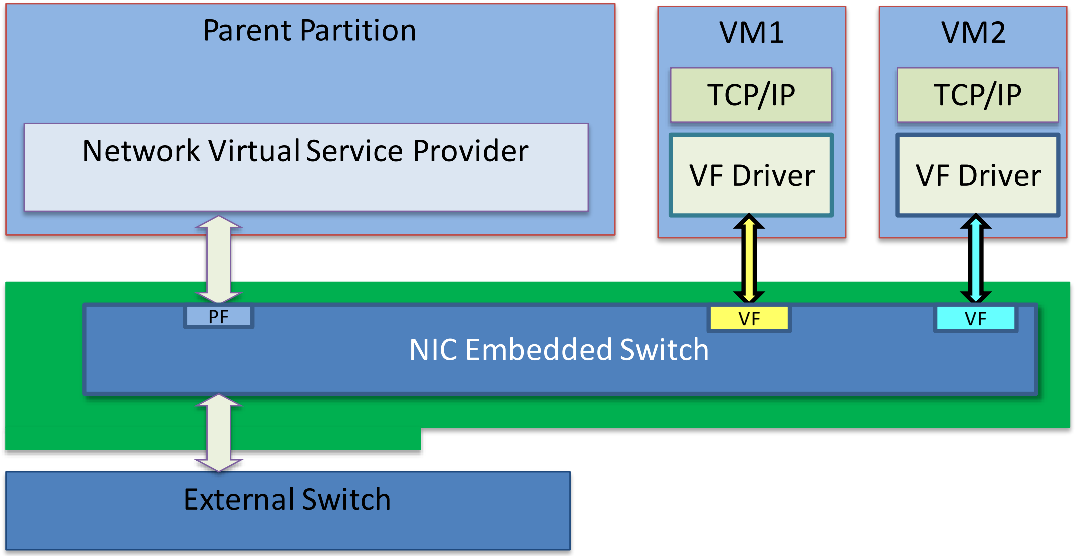
Unified Flow Tables

- Single hash lookup for each packet after flow is created
- Leaves room for new layers w/o perf impact (e.g. ILB, etc)
- Single flow table per VM can be sized with VM size
- All VFP actions can be expressed as header transpositions – e.g. encap/decap/l3 rewrite/l4 NAT
- Any set of header transpositions can be composed and expressed as one transposition
- Unified Flow Table: One match (per entire flowid, inner and outer) and one action (header transposition) per flow

Overview

- Azure and Scale
- Why Virtual Switches for SDN?
- Early implementations of Azure Host SDN
- Azure Host SDN Platform Goals
- VFP – Our platform for host SDN
- VFPv2 – Addressing Challenges of Scale
- Hardware Offloads
- Conclusion and Future

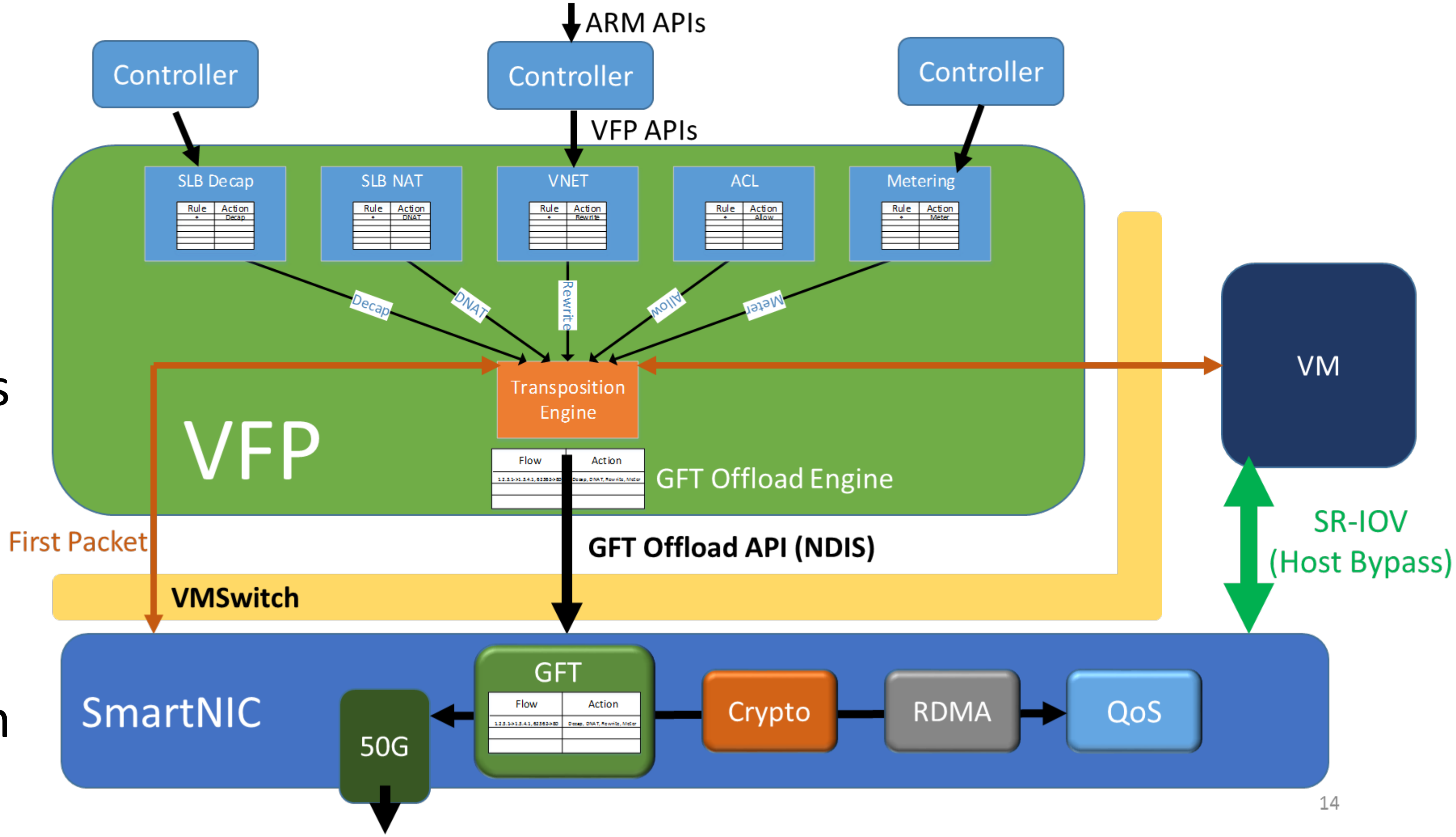
Single Root IO Virtualization (SR-IOV): Native Performance for Virtualized Workloads



But where is the SDN Policy?

2016: Accelerating VFP with FPGA SmartNICs

- Goal: Offload a cache of our internal (unified) flow table to the NIC
- Package Header Transpositions and Unified Flow IDs into hardware API
- Allows us to enable SR-IOV, applying virtualization policy in hardware and bypassing the host completely



Future of Host SDN: New Hardware/Software co-design models, programmable acceleration for transports, QoS, crypto, and more!

Results -
Azure Accelerated Networking: Fastest Cloud Network!

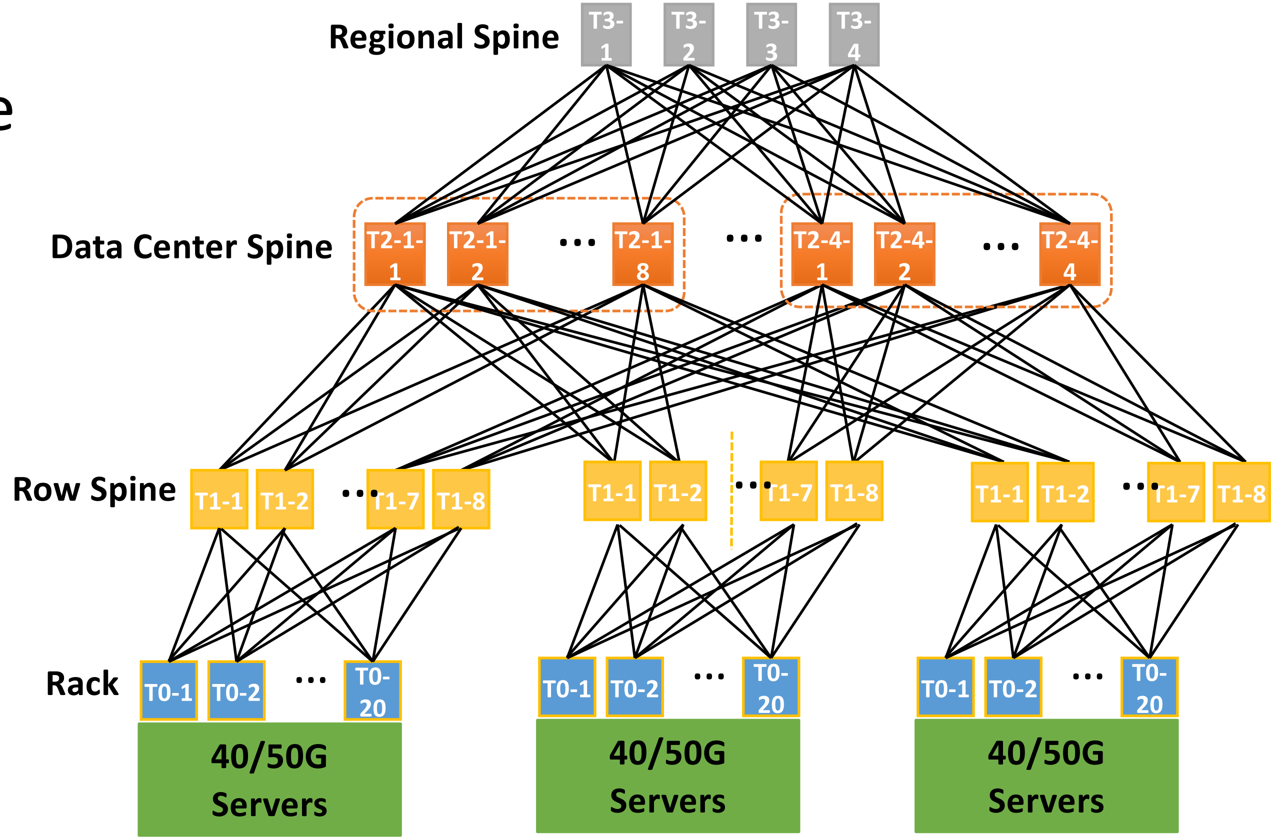
- Highest bandwidth VMs of any cloud
  - DS15v2 & D15v2 VMs get up to 25Gbps

- Consistent low latency network performance
  - Provides SR-IOV to the VM
  - 10x latency improvement
  - Increased packets per second (PPS)
  - Reduced jitter means more consistency in workloads

- Enables workloads requiring native performance to run in cloud VMs
  - >2x improvement for many DB and OLTP applications

Host Networking makes Physical Network Fast and Scalable

- Massive, distributed 40/100GbE network built on commodity hardware
  - No Hardware per tenant ACLs
  - No Hardware NAT
  - No Hardware VPN / overlay
  - No Vendor-specific control, management or data plane
- All policy is in software on hosts – and everything's a VM!
- Network services deployed like all other services
- VFP, battle tested in the cloud, is now available in Microsoft Azure Stack for private cloud as well!



Thanks!

- VFP Developers
  - Yue Zuo, Harish Kumar Chandrappa, Praveen Balasubramanian, Vikas Bhardwaj, Somesh Chaturmohta, Milan Dasgupta, Mahmoud Elhaddad, Luis Hernandez, Nathan Hu, Alan Jowett, Hadi Katebi, Fengfen Liu, Keith Mange, Randy Miller, Claire Mitchell, Sambhrama Mundkur, Chidambaram Muthu, Gaurav Poothia, Madhan Sivakumar, Ethan Song, Khoa To, Kelvin Zou, and Qasim Zuhair
- Design Influence
  - Alireza Dabagh, Deepak Bansal, Pankaj Garg, Changhoon Kim, Hemant Kumar, Parveen Patel, Parag Sharma, Nisheeth Srivastava, Venkat Thiruvengadam, Narasimhan Venkataramaiah, Haiyong Wang
- Dave Maltz, Mark Russinovich, and Albert Greenberg for years of support

Want to come build the next generation of scalable Host SDN? We're hiring!

fstone@microsoft.com