# SDN Features, Packet Transforms and Scale

## First Target Scenario: Highly Optimized Path, Dedicated Appliance, Little Processing or Encap to SDN Appliance and Policies on SDN Appliance

Why do we need this scenario? There is a huge cost associated with establishing the first connection (and the CPS that can be established).

* **High CPS / Flow SKU for NVA.**

Diagram

Description automatically generated

## Scale per DPU

\*\* Below are the expected numbers per Data Processing Unit (DPU); this applies to both IPV4 and IPV6 underlay and overlay.

\*\* IPV6 numbers will be lower.

|  |  |
| --- | --- |
| **Flow Scale** | * 1+ million per v-port * 50 million per DPU   + single encap IPv4 overlay and IPV6 underlay.   + single encap IPv6 overlay and IPV6 underlay. (This can be lower)   + single encap IPV4   + Encap <IPv6> <IPV4>   \*\* These are complex flows, details are below. |
| **CPS** | 4 million+ (max) 4M TCP CPS – ex: 4M SYN in both directions |
| **Routes** | 100k per v-port (max) |
| **ACLs** | 100k IP-Prefixes, 10k Src/Dst ports per v-port (max) |
| **NAT** |  |
| **V-Port** | 10k (max) |
| **Mappings** | 10 million total mapping per DPU |
|  |  |

## Milestone and Scoping

|  |  |  |  |
| --- | --- | --- | --- |
| S. No | Feature | Timeline | Perf |
|  | * VNET – VNET * Route support * LPM support * ACL support |  | CPS  Flow  PPS  Rule scale |
|  | * ILB inbound * Vip Inbound |  |  |
|  | * Private Link outbound (transposition) encap and change packet IPv4 to IPv6 (4 bits embedded) |  |  |
|  | * L3/L4 source NAT (correlated w/#2) outbound perspective (Cx address) to Internet; changing Cx source IP to Public IP (1:1 mapping) |  |  |
|  | * Private Link Service PLS (dest side of Private Link) IPv6 to IPv4; DNAT’ing |  |  |
|  | * Flow replication; supporting High Availability (HA); flow efficiently replicates to secondary card; Active/Passive (depending upon ENI policy) or can even have Active/Active; **OR** provision the same ENI over multiple devices w/o multiple SDN appliances – Primaries for a certain set of VMS can be on both | Not a must have for Private Preview |  |

## Virtual Port, Flows and Packet direction

SDN appliance in a multi-tenant network appliance (meaning 1 SDN appliance will have multiple cards, 1 card will have multiple machines or bare-metal servers), which supports Virtual Ports which can map to policy buckets corresponding to customer workloads, example: Virtual Machines, BM servers.

* **SDN controller** will create these virtual ports on SDN appliance and associate corresponding SDN policies like – Route, ACL, NAT etc. to these virtual ports. In other words, our software will communicate with the cards, hold card inventory and SDN placement, call API’s that are exposed through the card create policies, setup ENI, routes, ACLs, NAT, and different rules.
* Each **Virtual port** will be created with an **ENI identifier** like – Mac address, VNI or more.
  + Virtual port will also have attributes like – **Flow time-out**, **QOS, port properties related to the port.**
  + Virtual port is the container which holds all policies.



* On receiving a packet from the wire, the **SDN appliance** will determine the matching ENI, Packet direction and packet processing strategy based on *Encap Transformation and Rules Evaluation*.
  + On receiving a packet, the SDN appliance will perform a lookup based upon the inner source mac (VXLAN encap packet), if a **matching ENI is found**, and corresponding rule / flow processing will start.
  + Once the ENI is matched, the packet is first matched with flow table to see if an existing flow already matches this. If a flow match is found, a corresponding match action is executed without going into rule processing. Flow match direction is identified based on source and destination mac.
  + If **no flow match is found**, the ENI rule processing pipeline will execute.
    - **Inbound rule** processing pipeline is executed if destination mac in the packet matches the ENI mac. Once rule pipeline is executed corresponding flows are created.
    - **Outbound rule** processing pipeline is executed if source mac in the packet matches the ENI mac.
      * Once outbound rule processing is complete and final transforms are identified, the corresponding flow is created in the flow table.
      * Depending on implementation of flow table, a corresponding inbound flow may also be inserted to enable response packets to match the flow and bypass the rule processing pipeline.

**Example**: VM with IP 10.0.0.1 sends a packet to 8.8.8.8, VM Inbound ACL blocks all internet, VM outbound ACL allows 8.8.8.8 – Response packet from 8.8.8.8 must be allowed without opening any inbound ACL due to the flow match.



## Packet processing Pipeline

## ACL

* The ACL pipeline has 3 levels; an ACL decision is based on the most restrictive match across all 3 levels. The 1st layer (contains default rules) is controlled by Azure/MSFT. The 2nd and 3rd layers are Customer controlled.
* If an ACL rule with bit exit ACL pipeline on hit is matched, the ACL pipeline is abandoned.
* Expected ACL scale – max 100k prefixes, max 10k ports
* ACL table entry count = 1000 per table. (NOTE: Each table entry can have comma separated prefix list.)
* Action Definitions:
  + Block (terminate)
    - If ‘terminate’ is not used here, the last line is the most important in ACL Level1
  + Soft Block (general block, with specific permits, non-terminating, proceed to next group) - or think of this as a Block, and then a ‘no’ for ‘termination’
  + Allow (non-terminate, proceed to next)
  + Default action = Deny
* ACL Group: evaluate rules based on Priority (within an ACL Group); Terminate vs non-Terminate pertains to the Pipeline

**ACL\_LEVEL1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Source | Destination | Source Port | Destination Port | Protocol | Action | Priority | Exit ACL pipeline on hit?  (Is Terminating) |
| 10.0.0.0/24, 20.0.0.0/24,  30.0.0.0/24 | 10.0.0.10/32  10.0.0.11/32  10.0.0.12/32  10.0.0.13/32  10.0.0.14/32  30.0.0.0/24 | \* | \* | TCP | Allow | 0 | No |
| 10.0.0.0/24, 20.0.0.0/24,  30.0.0.0/24 | 10.0.0.200/32 | \* | \* | TCP | Allow | 1 | No |
| 10.0.0.0/24, 20.0.0.0/24,  30.0.0.0/24 | 10.0.0.201/32 | \* | \* | TCP | Block | 2 | Yes |
| 10.0.0.0/24, 20.0.0.0/24,  30.0.0.0/24 | 10.0.0.202/32 | \* | \* | TCP | Allow | 3 | Yes |
| 10.0.0.0/24, 20.0.0.0/24,  30.0.0.0/24 | 10.0.0.203/32 | \* | \* | TCP | Allow | 4 | No |
| \* | 8.8.8.8/32 | \* | \* | \* | Block | 5 | Yes |
| \* | 8.8.8.8/32 | \* | \* | \* | Allow | 6 | Yes |
| \* | 9.9.9.9/32 | \* | \* | \* | Allow | 7 | Yes |
| \* | \* | \* | \* | \* | Block | 8 | No |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**ACL\_LEVEL2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Source | Destination | Source Port | Destination Port | Protocol | Action | Priority | Exit ACL pipeline on hit?  (Is Terminating) |
| 10.0.0.0/24 | \* | \* | \* | TCP | Allow | 1 | No |
| 10.0.0.0/24 | 10.0.0.202/32 | \* | \* | TCP | Block | 1 | Yes |
| 10.0.0.0/24 | 10.0.0.203/32 | \* | \* | TCP | Block | 1 | Yes |
| \* | 8.8.8.8/32 | \* | \* | \* | Allow | 2 | No |
| \* | 9.9.9.9/32 | \* | \* | \* | Block | 2 | Yes |
| \* | **1.1.1.2/32** | \* | \* | \* | Allow | 30 | No |
| \* | \* | \* | \* | \* | Block | 3 | No |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Order of evaluation / priority of evaluation**

* ACL\_LEVEL1 -> ACL\_LEVEL2

**Test Scenarios and expected results**

* For simplicity below table only has IP conditions, but the same combinations exist for ports also.
* ACL rules are direction aware, below example is assuming a VM with source IP = 10.0.0.100 which is trying to send packets to various destinations and has above ACL rules on its v-port.

**Outbound Traffic example evaluation and outcome**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Source IP** | **Destination IP** | **Decision of ACL\_LEVEL1** | **Decision of ACL\_LEVEL2** | **Outcome** |
| 10.0.0.100 | 10.0.0.200 | Allow (Terminating = false) | Allow (Terminating = false) | Allow |
| 100.0.0.100 | 100.0.0.201 | Block (Terminating = True) | Not evaluated or Ignored | Block |
| 100.0.0.100 | 100.0.0.202 | Allow (Terminating = True) | Not evaluated or Ignored | Allow |
| 100.0.0.100 | 100.0.0.203 | Allow (Terminating = false) | Block (Terminating = True) | Block |
| 100.0.0.100 | **8.8.8.8** | Block (Terminating = True) | Not evaluated or Ignored | Block |
| 100.0.0.100 | **1.1.1.1** | Block (Terminating = false) | Block (Terminating = false) | Block |
| 100.0.0.100 | **1.1.1.2** | Block (Terminating = false) | Allow (Terminating = false) | Allow |

## Routes and Route-Action

* Routes are usually LPM based Outbound
* Each route entry will have a prefix, and separate action entry
* The lookup table is per ENI, but could be Global, or multiple Global lookup tables per ENIs
* Outer Encap IPv4 using permits routing between servers within a Region; across the Region we use IPv6

*Why would we want to use these*?

* Example: to block prefixes to internal DataCenter IP addresses, but Customer uses prefixes inside of their own VNET
* Example: Lookup between CA (inside Cx own VNET) and PA (Provider Address) using lookup table (overwrite destination IP and MAC before encap)
* Example: Customer sends IPv4, we encap with IPv6
* Example: ExpressRoute with 2 different PAs specified (load balancing across multiple PAs) using 5 tuples of packet to choose 1st PA or 2nd PA

|  |  |
| --- | --- |
| Route Type | Example |
| Encap\_with\_lookup\_V4\_underlay | * Encap action is executed based on lookup into the mapping table. * V4 underlay is used |
| Encap\_with\_lookup\_V6\_underlay | * Encap action is executed based on lookup into the mapping table. * V6 underlay is used |
| Encap\_with\_Provided\_data (PA) | * Encap action is executed based on provided data. * Multiple PA can be provided. |
| Outbound NAT (SNAT)\_L3 | * L3 NAT action is executed on source IP, based on provided data. |
| Outbound NAT (SNAT)\_L4 | * L4 NAT action is executed on source IP, source port based on provided data. |
| Null | * Blocks the traffic |
| Private Link | - |
|  |  |

**Mapping Table for a v-port**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customer Address | Physical Address – V4 | Physical Address – V6 | Mac-Address for D-Mac Rewrite | VNI to Use |
| 10.0.0.1 | 100.0.0.1 | 3ffe::1 | E4-A7-A0-99-0E-17 | 10001 |
| 10.0.0.2 | 100.0.0.2 | 3ffe::2 | E4-A7-A0-99-0E-18 | 10001 |
| 10.0.0.3 | 100.0.0.3 | 3ffe::3 | E4-A7-A0-99-0E-19 | 20001 |
| 10.0.0.4 | 100.0.0.4 | 3ffe::3 | E4-A7-A0-99-0E-20 | 10001 |

**Route Table for a v-port**

* LPM decides which route is matched.
* Once the route is matched, a corresponding action is executed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Route** | **Action** | **Route Type** | **Route Id** |
| 10.0.0.0/24, 20.0.0.0/24, 30.0.0.0/24, 10.0.0.0/8, …… more prefixes (up-to 20k) | Encap Type: VXLAN  Action – lookup mapping table for exact destination, VNI and D-Mac re-write info. | Encap\_with\_lookup\_V4\_underlay | 1 |
| 10.0.0.100/32 | Encap Type: VXLAN  Action: Encap\_with\_Provided\_data   * Encap with source PA = 100.0.0.1 * Encap with destination PA = 23.0.0.1 * **Re-write D-Mac** to E4-A7-A0-99-0E-28 * Use VNI = 90000 | Encap\_with\_Provided\_data | 2 |
| 10.0.0.101/32 | Encap Type: VXLAN  Action: Encap\_with\_Provided\_data   * Encap with source PA = 100.0.0.1 * Encap with destination PA = 23.0.0.10, 23.0.0.11, 23.0.0.13, 23.0.0.14 * Re-write D-Mac to E4-A7-A0-99-0E-29 * Use VNI = 90000 | Encap\_with\_Provided\_data\_ECMP | 3 |
| 8.8.8.8/32 | L3 NAT  **Action**:  Transpose source IP to provided NAT IP, keep all ports same.  NAT IP: 10.0.0.1 -> 15.0.0.1 | Outbound NAT (SNAT)\_L3 | 4 |
| 9.9.9.9.9/32 | L4 NAT  **Action**:  Transpose source IP and source port re-write ports from configured port pool. | Outbound NAT (SNAT)\_L4 | 5 |
| 0.0.0.0/32 | **NULL** | Null | 6 |
| **23.0.0.1/32** | **Service endpoint** | **ST** | 7 |
| **23.0.0.2/32** | **Private Link - TBD** | **Private Link - TBD** | 8 |

**Route example - Outbound packets**

|  |  |  |  |
| --- | --- | --- | --- |
| **Original Packet** | **Matched route** | **Transform** | **Route Type** |
| **10.0.0.1 -> 10.0.0.2**  **SMAC1-> DMAC\_FAKE**  **Outer:**  SRC: [Physical IP of host]  DST: [Physical IP of SDN Appliance]  **VXLAN**   * VNI: custom   **Inner Mac:**  SRC – SMAC1  DST - **DMAC\_FAKE**  **Inner IP:**  [10.0.0.1] -> [10.0.0.2] | **Route Id = 1** | **Outer:**  SRC: [SDN Appliance IP]  DST: [100.0.0.2] # Came from mapping table lookup  **VXLAN**   * VNI: 10001   **Inner Mac:**  SRC – SMAC1  DST - E4-A7-A0-99-0E-18  **Inner IP:**  [10.0.0.1] -> [10.0.0.2] | Encap\_with\_lookup\_V4\_underlay |
| **10.0.0.1 -> 10.0.0.100**  **SMAC1-> DMAC\_FAKE**  **Outer:**  SRC: [Physical IP of host]  DST: [Physical IP of SDN Appliance]  **VXLAN**   * VNI: custom   **Inner Mac:**  SRC – SMAC1  DST - **DMAC\_FAKE**  **Inner IP:**  [10.0.0.1] -> [10.0.0.2] | **Route Id = 2** | **Outer:**  SRC: [SDN Appliance IP]  DST: [23.0.0.1] # Came from mapping table lookup  **VXLAN**   * VNI: 90000   **Inner Mac:**  SRC – SMAC1  DST - E4-A7-A0-99-0E-28  **Inner IP:**  [10.0.0.1] -> [10.0.0.100] | Encap\_with\_Provided\_data |
| **10.0.0.1 -> 10.0.0.101**  **SMAC1-> DMAC\_FAKE**  **Outer:**  SRC: [Physical IP of host]  DST: [Physical IP of SDN Appliance]  **VXLAN**   * VNI: custom   **Inner Mac:**  SRC – SMAC1  DST - **DMAC\_FAKE**  **Inner IP:**  [10.0.0.1] -> [10.0.0.2] | **Route Id = 3** | **Outer:**  SRC: [SDN Appliance IP]  DST: ECMP on [23.0.0.10, 23.0.0.11, 23.0.0.13, 23.0.0.14] # Came from mapping table lookup  **VXLAN**   * VNI: 90000   **Inner Mac:**  SRC – SMAC1  DST - E4-A7-A0-99-0E-29  **Inner IP:**  [10.0.0.1] -> [10.0.0.100] | Encap\_with\_Provided\_data\_ECMP |
| **10.0.0.1 ->** 8.8.8.8  **SMAC1-> DMAC\_FAKE**  **Outer:**  SRC: [Physical IP of host]  DST: [Physical IP of SDN Appliance]  **VXLAN**   * VNI: custom   **Inner Mac:**  SRC – SMAC1  DST - **DMAC\_FAKE**  **Inner IP:**  [10.0.0.1] -> [8.8.8.8] | **Route Id = 4** |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Packet Flow

### Inbound

**Fast Path – Flow Match**

**Slow Path – No flow match**

### Outbound

**Fast path – flow match**

**Slow Path (policy evaluation) – No flow match**

## Packet Transform Examples

### VNET to VNET Traffic

**V-Port**

* **Physical address = 100.0.0.2**
* **V-Port Mac = V-PORT\_MAC**

**VNET Definition:**

* + 10.0.0.0/24
  + 20.0.0.0/24

**VNET Mapping Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **V4 underlay** | **V6 underlay** | **Mac-Address** | **Mapping Action** | **VNI** |
| **10.0.0.1** | 100.0.0.1 | 3ffe :: 1 | **Mac1** | VXLAN\_ENCAP\_WITH\_DMAC\_DE-WRITE | 100 |
| **10.0.0.2** | 100.0.0.2 | 3ffe :: 2 | **Mac2** | VXLAN\_ENCAP\_WITH\_DMAC\_DE-WRITE | 200 |
| **10.0.0.3** | 100.0.0.3 | 3ffe :: 3 | **Mac3** | VXLAN\_ENCAP\_WITH\_DMAC\_DE-WRITE | 300 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Packet Transforms**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SRC -> DST** | **Out-ACL1** | **Out-ACL2** | **Out-ACL3** | **Routing** | **Final** |
|  | Block 10.0.0.10  Allow \* | Block 10.0.0.11  Allow \* | Allow \* | 10.0.0.0/24 - Route Action = VNET  20.0.0.0/24 - Route Action = VNET |  |
| **10.0.0.1 -> 10.0.0.10**  **SMAC1-> DMAC\_FAKE** | **Block** |  |  |  | **Blocked** |
| **10.0.0.1 -> 10.0.0.11**  **SMAC1-> DMAC\_FAKE** | **Allow** | **Block** |  |  | **Blocked** |
| **10.0.0.1 -> 10.0.0.2**  **SMAC1-> DMAC\_FAKE**  **Outer:**  SRC: [Physical IP of host]  DST: [Physical IP of SDN Appliance]  **VXLAN**   * VNI: custom   **Inner Mac:**  SRC – SMAC1  DST - **DMAC\_FAKE**  **Inner IP:**  [10.0.0.1] -> [10.0.0.2] | **Allow** | **Allow** | **Allow** | * Matched LPM route 10.0.0.0/24. * Execute action VNET – which will lookup in mapping table and take mapping action. | **Highlighted the changes in packet.**  **Outer:**  SRC: [100.0.0.2]  DST: [100.0.0.1]  **VXLAN**   * VNI: 200   **Inner Mac:**  SRC – SMAC1  DST - Mac1  **Inner IP:**  [10.0.0.1] ->  [10.0.0.2] |
| **10.0.0.1 -> 10.0.0.3**  **SMAC1-> DMAC\_FAKE** |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

### VNET to Internet - TBD

### VNET to Service Endpoints - TBD

### VNET to Private Link - TBD.

## Metering

* Metering will be based on per flow stats, metering engine will consume per flow stats of bytes-in and bytes-out.

## VNET Encryption

## Telemetry

## BGP

## Watchdogs

## Servicing

## Debugging

Counters per rule to trace an increment per layer, ACL hits, Packet Captures, Bandwidth Metering for Routing Rules to count bytes (each flow associated with a bandwidth counter when an LPM is hit – many flows *may* share the same counters).

## Flow Replication

## Unit Testing and development

* Need ability to run rule processing behavior on dev box / as part of merge validation.

## Internal Partner dependencies

* SLB VXLAN support
* Reduced tuple support on host.

## Packet transforms

### VNET



### Internal Load balancer



### Private Link



### Private Link Service



### Service Tunneling



### Inbound from LB



### Outbound NAT – L4

(L3 works in same way except port re-write)

