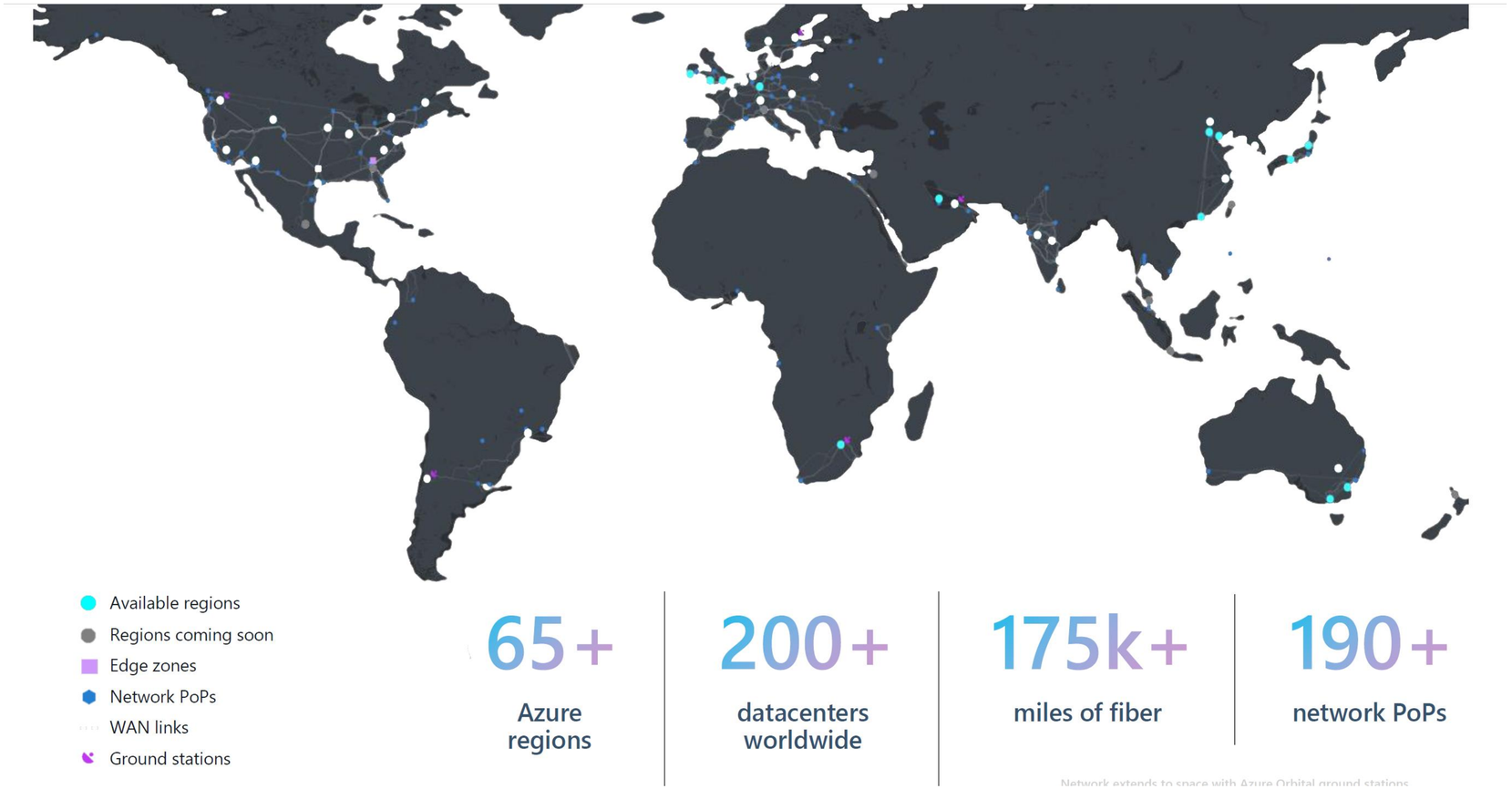
Offloading Cloud Network Services at Production Scale with SONiC DASH SmartSwitch

Shaofeng Wu, Zhixiong Niu, Riff Jiang, Lawrence Lee, Junhua Zhai, Ze Gan,
Vasundhara Volam, Prabhat Aravind, Prince Sunny, Prince George, Qi Luo, Evan Langlais,
Soumya Tiwari, Venkat Satish Katta, Weixi Chen, Rishiraj Hazarika, Sachin Jain, Deven Jagasia,
Michal Zygmunt, Avijit Gupta, Neeraj Motwani, Pranjal Shrivastava, Qiang Su, Anil Reddy Pannala,
Kristina Moore, James Grantham, Anupam Pandey, Xin Liu, Guohan Lu, Gerald De Grace,
Rishabh Tewari, Lihua Yuan, Erica Lan, Deepak Bansal, Dave Maltz, Yongqiang Xiong, Hong Xu

香港中文大學
The Chinese University of Hong Kong



Cloud Infrastructure Spending Surges to $129 bn in Q1 2026 as Growth Hits 35% and AI Reshapes Market Dynamics

AWS Vs. Google Cloud Vs. Microsoft Azure Q1 Earnings Face-Off

Cloud network services in Azure

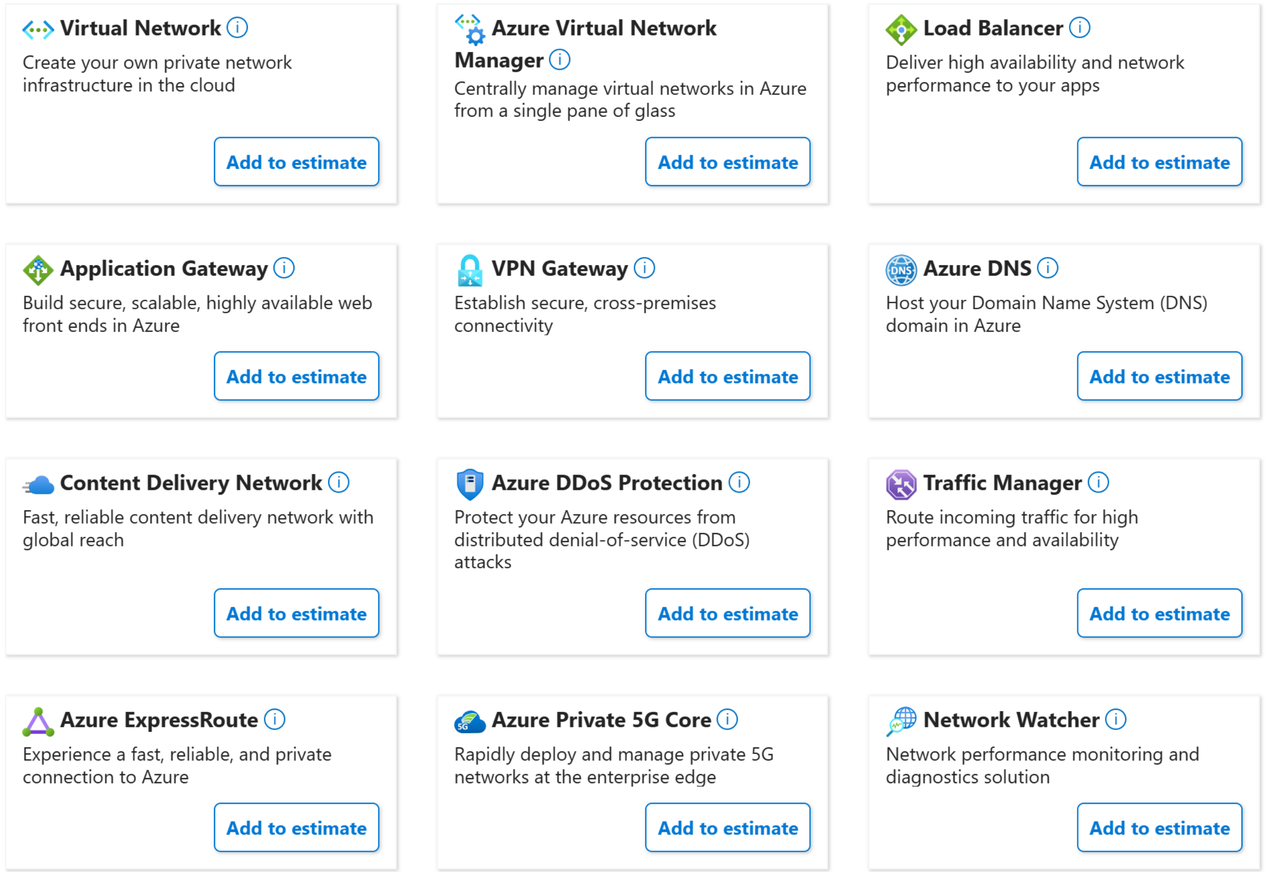
Hyper-scale clouds deliver infrastructure as a service (IaaS)

Cloud network services in Azure

Hyper-scale clouds deliver infrastructure as a service (IaaS)

Examples of general cloud network services:

- Stateful firewalls
- Stateful load balancers
- Azure Private Link
- Azure virtual networks …



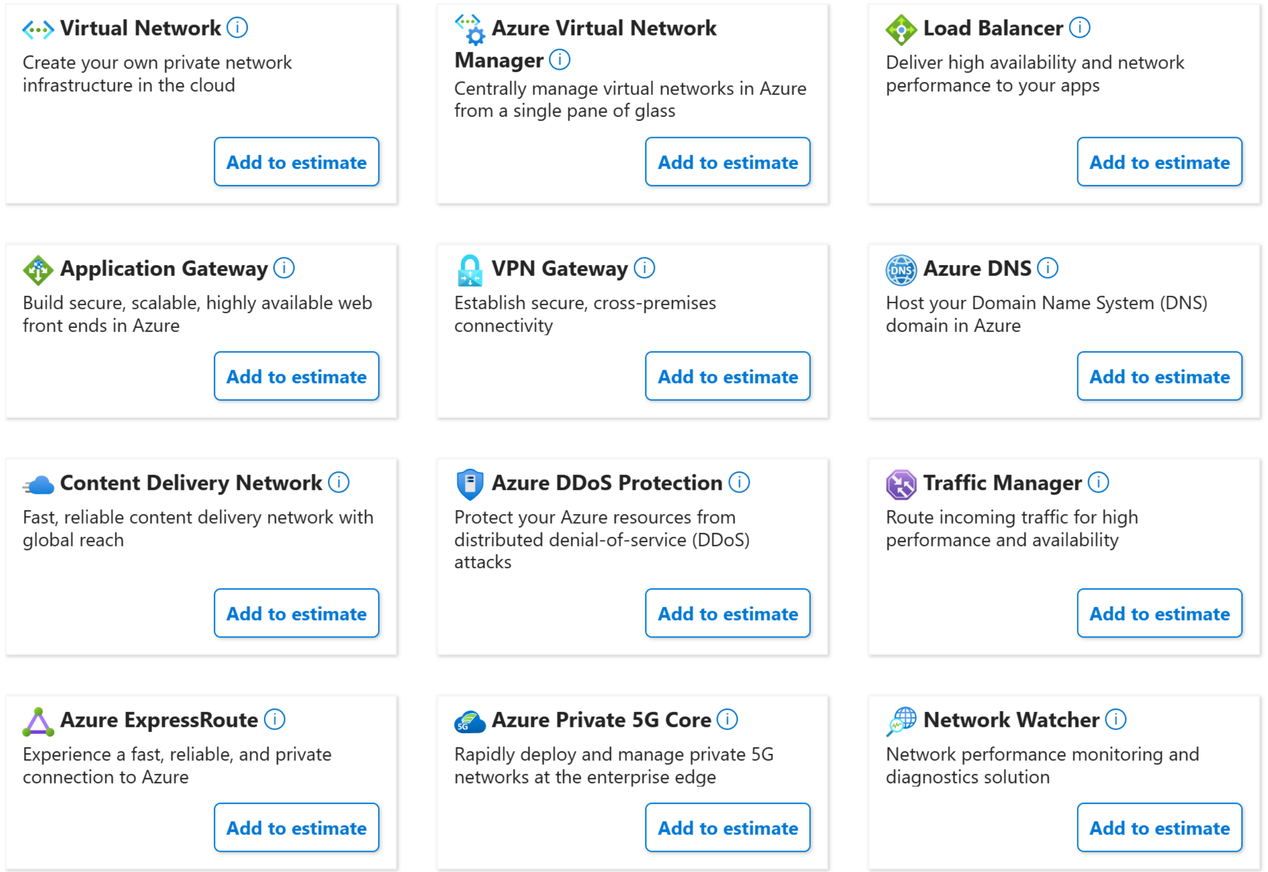
Cloud network services in Azure

Hyper-scale clouds deliver infrastructure as a service (IaaS)

Examples of general cloud network services:

- Stateful firewalls
- Stateful load balancers
- Azure Private Link
- Azure virtual networks …

Connection-oriented (a.k.a, stateful)



Cloud network services as the performance and cost bottleneck

Cloud network services as the performance and cost bottleneck

Performance

CPU is inefficient for cloud network services, especially for heavy middleboxes

< 300K CPS with a single-rule stateful firewall[1]

Cloud network services as the performance and cost bottleneck

Performance

CPU is inefficient for cloud network services, especially for heavy middleboxes

< 300K CPS with a single-rule stateful firewall[1]

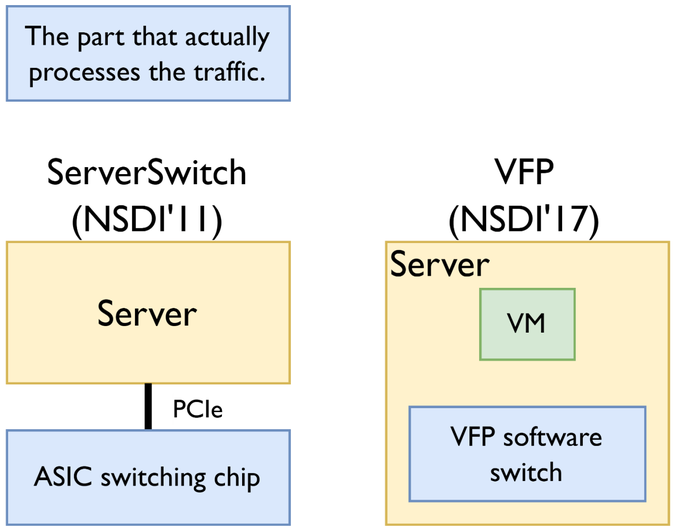
Cost

VM revenue = customer price – hosting cost[2], pack more VMs per server -> higher revenue

< Azure D2s v5: 2vCPUs, $0.096/hour[3]

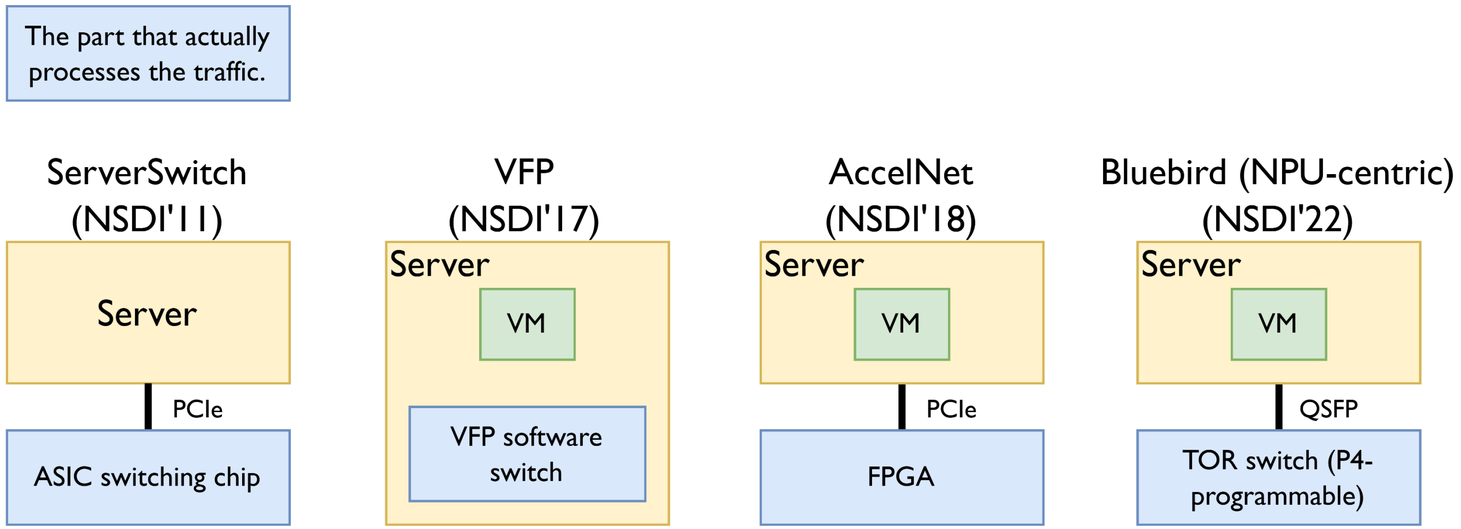
[1] Deepak Bansal, et al. Disaggregating stateful network functions. InProc. USENIX NSDI, 2023.
[2] Daniel Firestone, et al.. Azure Accelerated Networking: SmartNICs in the public cloud. In Proc. USENIX NSDI, 2018.
[3] Pricing Calculator | Microsoft Azure, 2025.

Evolution of Cloud Network Service Offload in Azure



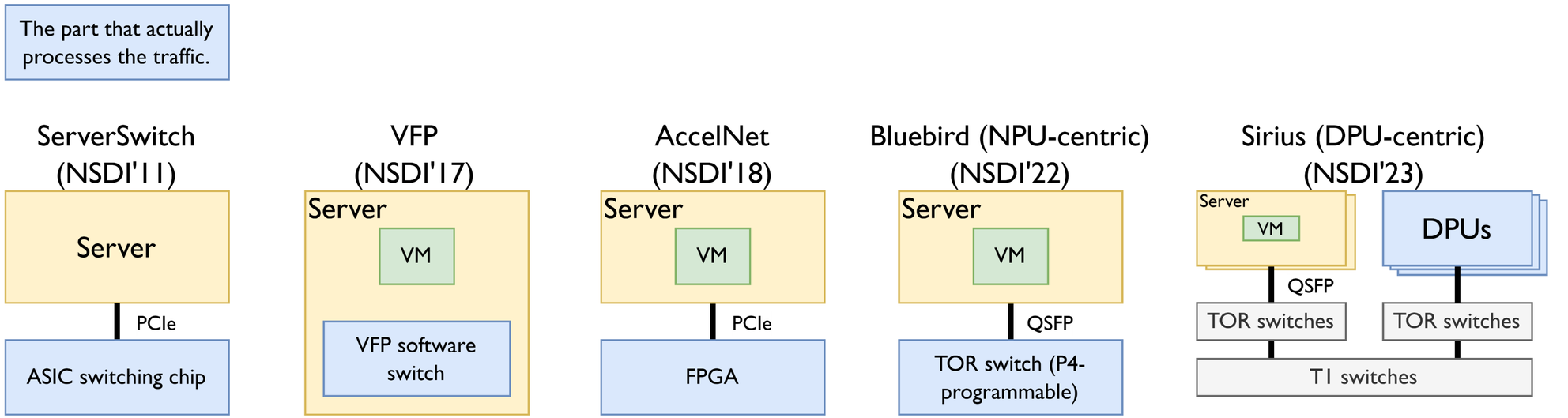
Throughput	N/A	<40Gbps/core
CPS		O(100K)/core, expensive to scale-up
# Connections		>100M
Cost		High, consumes server cores

Evolution of Cloud Network Service Offload in Azure



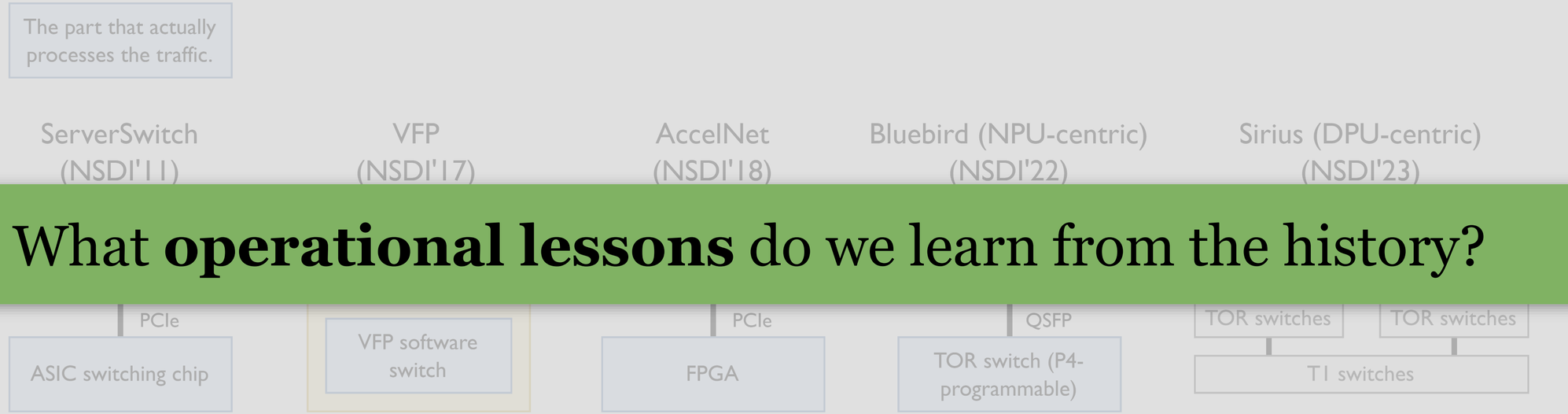
Throughput	N/A	<40Gbps/core	Up to 200Gbps	6.4-12.8Tbps
CPS		O(100K)/core, expensive to scale-up		O(1M), hard to scale-up
# Connections		>100M	O(10M)	O(1M)
Cost		High, consumes server cores	High, per server and high CAPEX	Low

Evolution of Cloud Network Service Offload in Azure



Throughput	N/A	<40Gbps/core	Up to 200Gbps	6.4-12.8Tbps	**O(1Tbps)**
CPS		O(100K)/core, expensive to scale-up		O(1M), hard to scale-up	**O(10M)**
# Connections		>100M	O(10M)	O(1M)	**O(100M)**
Cost		High, consumes server cores	High, per server and high CAPEX	Low	Medium, shared pool

Evolution of Cloud Network Service Offload in Azure



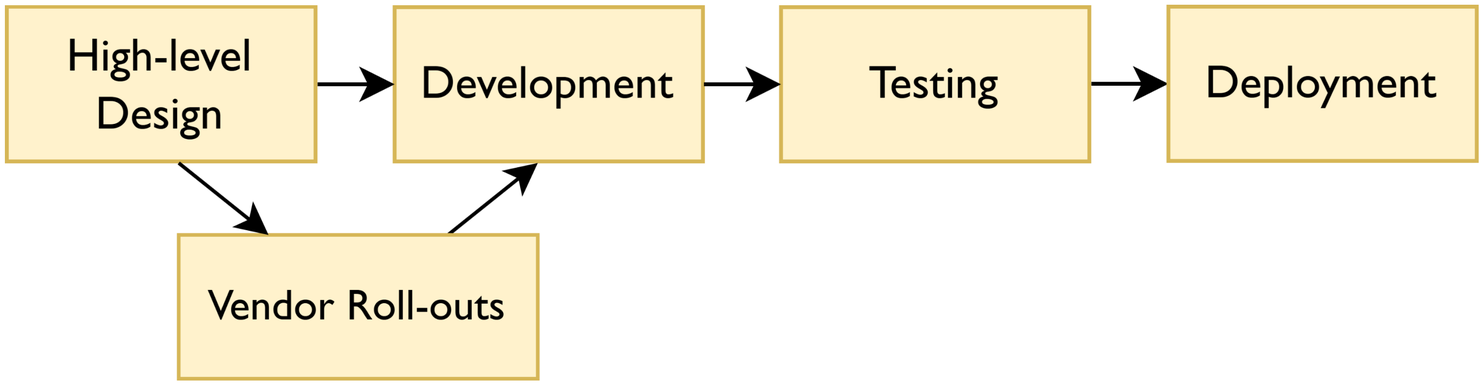
What operational lessons do we learn from the history?

Throughput	N/A	<40Gbps/core	Up to 200Gbps	6.4-12.8Tbps	**O(1Tbps)**
CPS		O(100K)/core, expensive to scale-up		O(1M), hard to scale-up	**O(10M)**
# Connections		>100M	O(10M)	O(1M)	**O(100M)**
Cost		High, consumes server cores	High, per server and high CAPEX	Low	Medium, shared pool

Lesson beyond prototyped perf.
Software development model is important

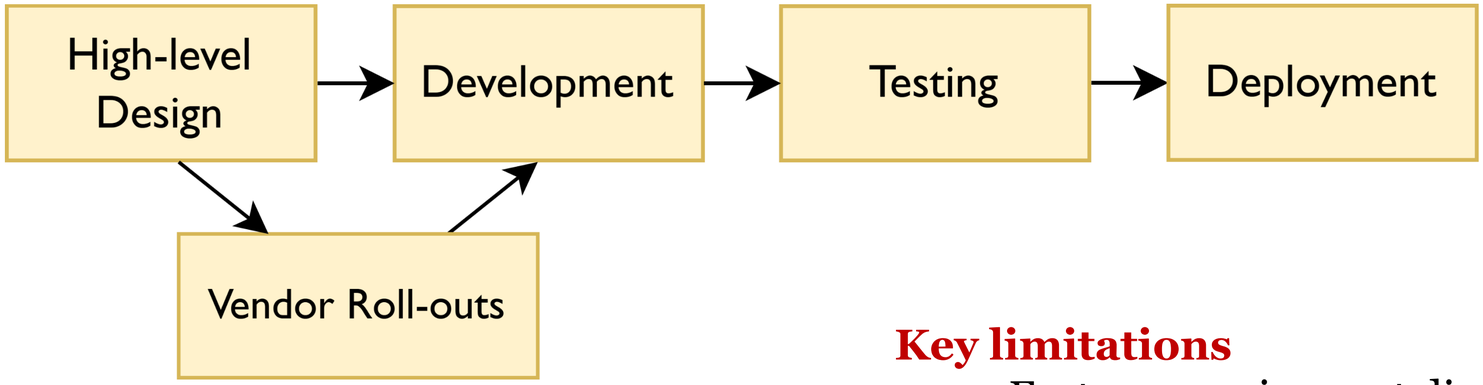
Lesson beyond prototyped perf.
Software development model is important

We (and all clouds) follow a non cloud-centric development model for service offload



Lesson beyond prototyped perf.
Software development model is important

We (and all clouds) follow a non cloud-centric development model for service offload



Key limitations

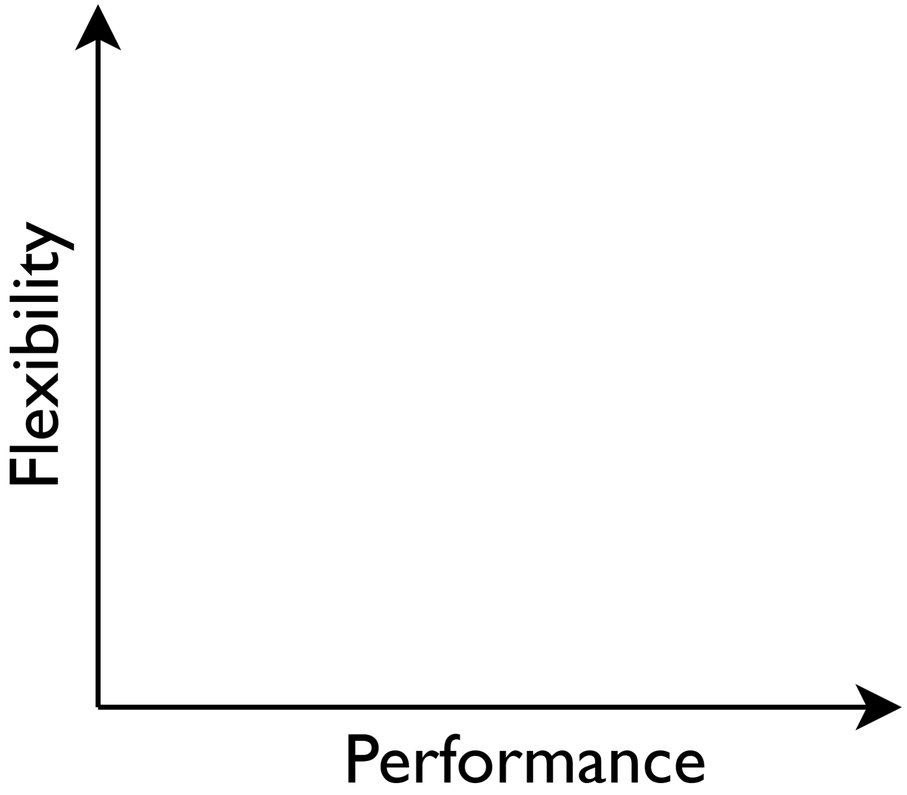
- Feature-requirement divergence
- Slow roll-outs (6–12 months/release)
- Vendor lock-in & supply chain risk[1,2]
- Black-boxed behavior

[1] Manikandan Arumugam, Deepak Bansal, Navdeep Bha-tia, James Boerner, Simon Capper, Changhoon Kim,Sarah McClure, Neeraj Motwani, Ranga Narasimhan,Urvish Panchal, Tommaso Pimpo, Ariff Premji, Pran-jal Shrivastava, and Rishabh Tewari. Bluebird: High-performance SDN for bare-metal cloud services. In Proc. USENIX NSDI, 2022.
[2] Intel® Tofino Products, PCN 827577-00, Product Discontinuance, Tofino End of Life, 2024.

Lesson beyond prototyped perf.
How we model services affects performance

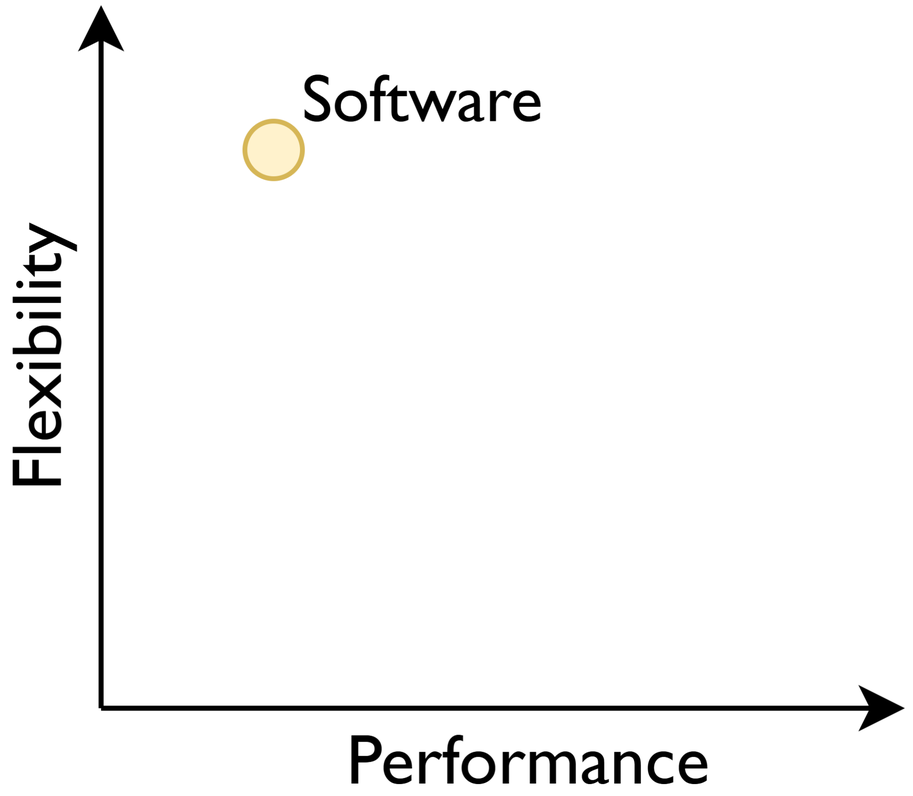
Lesson beyond prototyped perf.
How we model services affects performance

Trade-off between flexibility & performance



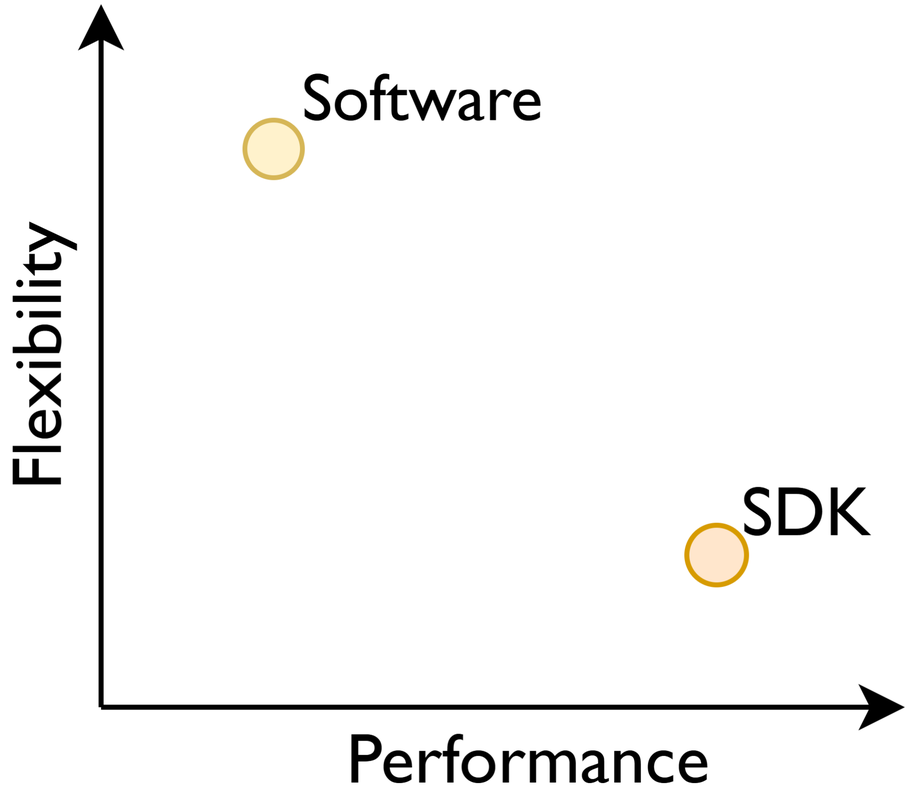
Lesson beyond prototyped perf.
How we model services affects performance

Trade-off between flexibility & performance
- Software-based: VFP, …
  - flexible service model, software-native performance



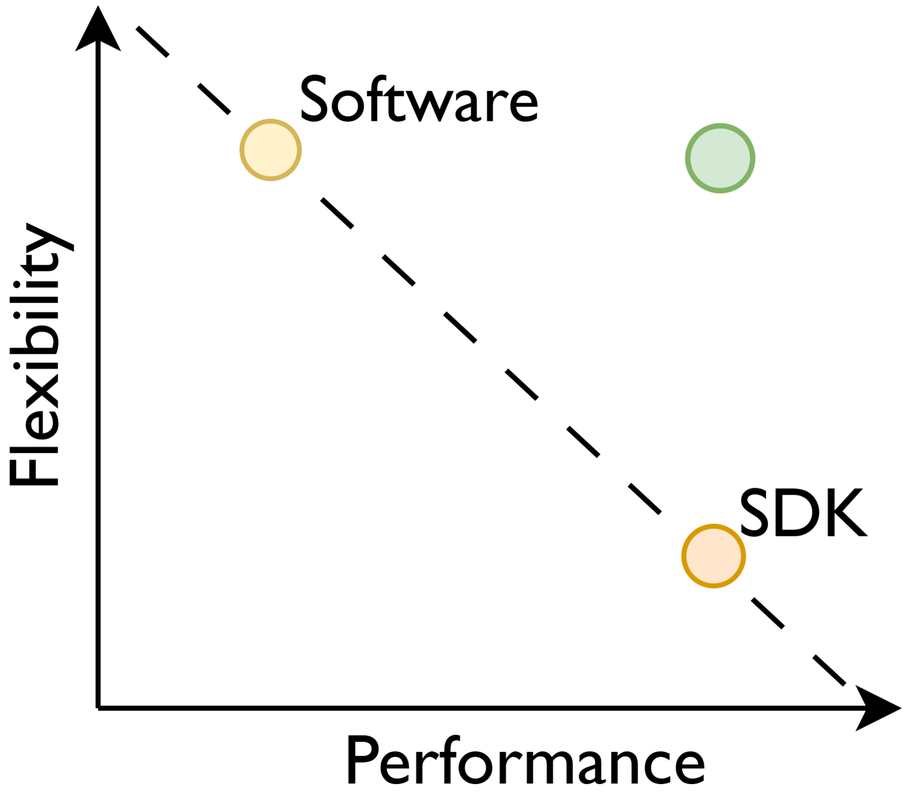
Lesson beyond prototyped perf.
How we model services affects performance

Trade-off between flexibility & performance
- Software-based: VFP, …
  - flexible service model, software-native performance
- Vendor SDKs
  - proprietary model, hardware-native performance



Lesson beyond prototyped perf.
How we model services affects performance

Trade-off between flexibility & performance
- Software-based: VFP, …
  - flexible service model, software-native performance
- Vendor SDKs
  - proprietary model, hardware-native performance

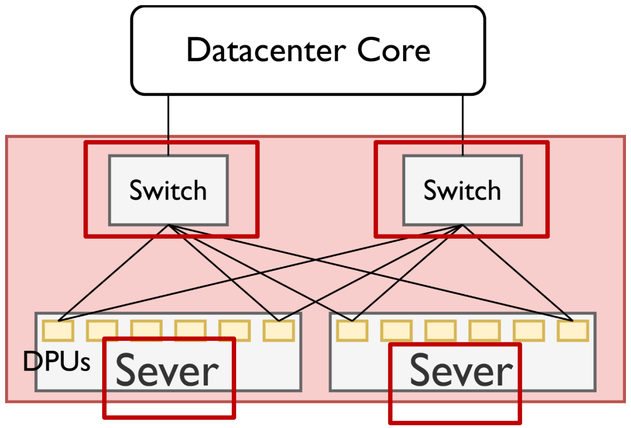


Lesson beyond prototyped perf.
Physical footprint matters

Simple question: Do things fit?
A bigger problem of efficiency, deployability & scalability

Lesson beyond prototyped perf.
Physical footprint matters

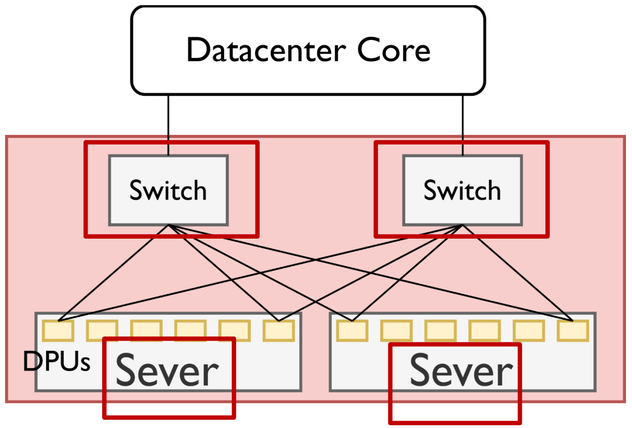
Simple question: Do things fit?
A bigger problem of efficiency, deployability & scalability

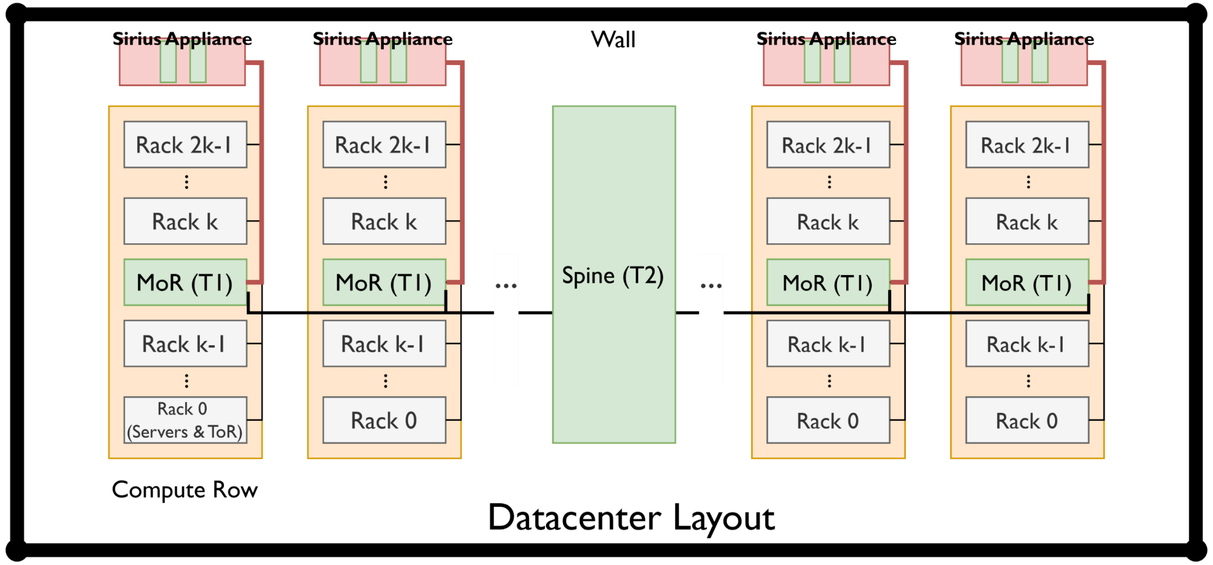


Excessive redundant equipment, deployed as
a **48U purpose-built appliance chassis**

Lesson beyond prototyped perf.
Physical footprint matters

Simple question: Do things fit?
A bigger problem of efficiency, deployability & scalability





Excessive redundant equipment, deployed as a **48U purpose-built appliance chassis**

Forcefully surpassing pre-designed floor space and cooling cap
Negative impact on circulation, safety, maintenance, decommission, …

Loss: ~$100M/annum

Design goals from the lessons

Design goals from the lessons

Vendor-neutrality

Leveraging multi-sourced hardware solution for same services

Design goals from the lessons

Vendor-neutrality	Leveraging multi-sourced hardware solution for same services
Service modelling	Specific enough for performance optimizations, general enough for cloud network services in production

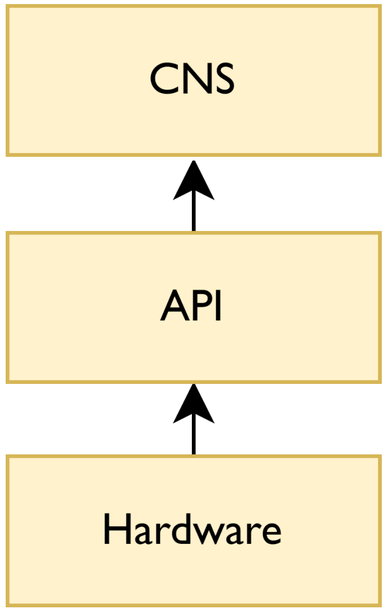
Design goals from the lessons

Vendor-neutrality	Leveraging multi-sourced hardware solution for same services
Service modelling	Specific enough for performance optimizations, general enough for cloud network services in production
Deployability	Highly scalable in real data centers, reduced footprints, high efficiency

Design goals from the lessons

Vendor-neutrality	Leveraging multi-sourced hardware solution for same services
Service modelling	Specific enough for performance optimizations, general enough for cloud network services in production
Deployability	Highly scalable in real data centers, reduced footprints, high efficiency
Performance	Performant stateful packet processing and CPS (same as or higher than previous gen under the above contexts, enabling heavy workloads)

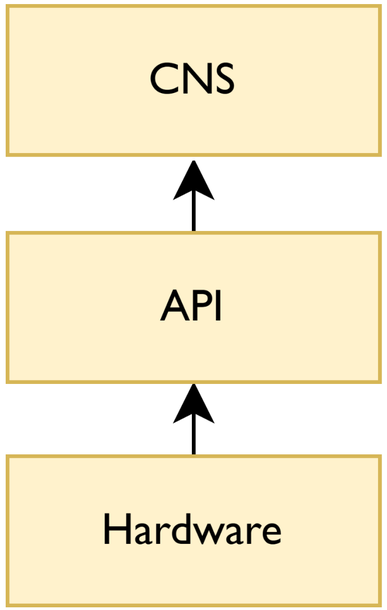
Changing the methodology of development



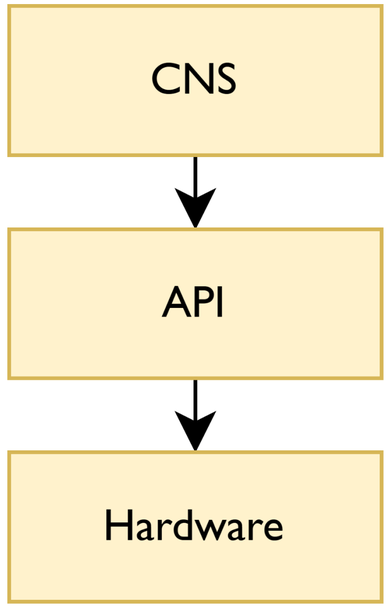
Non cloud-centric

Changing the methodology of development

Key idea: cloud requirements should determine/guide hardware features



Non cloud-centric

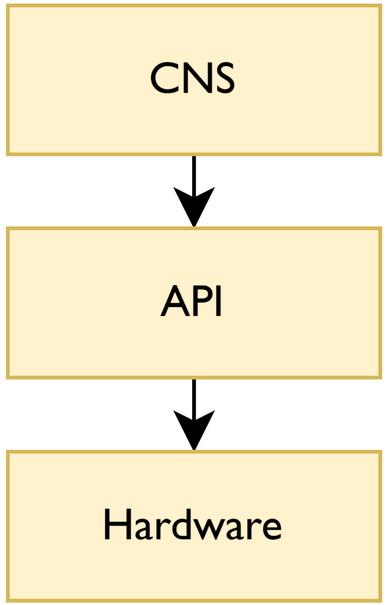


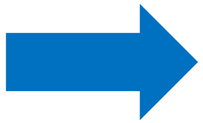
Ideal: Service-oriented & community-driven

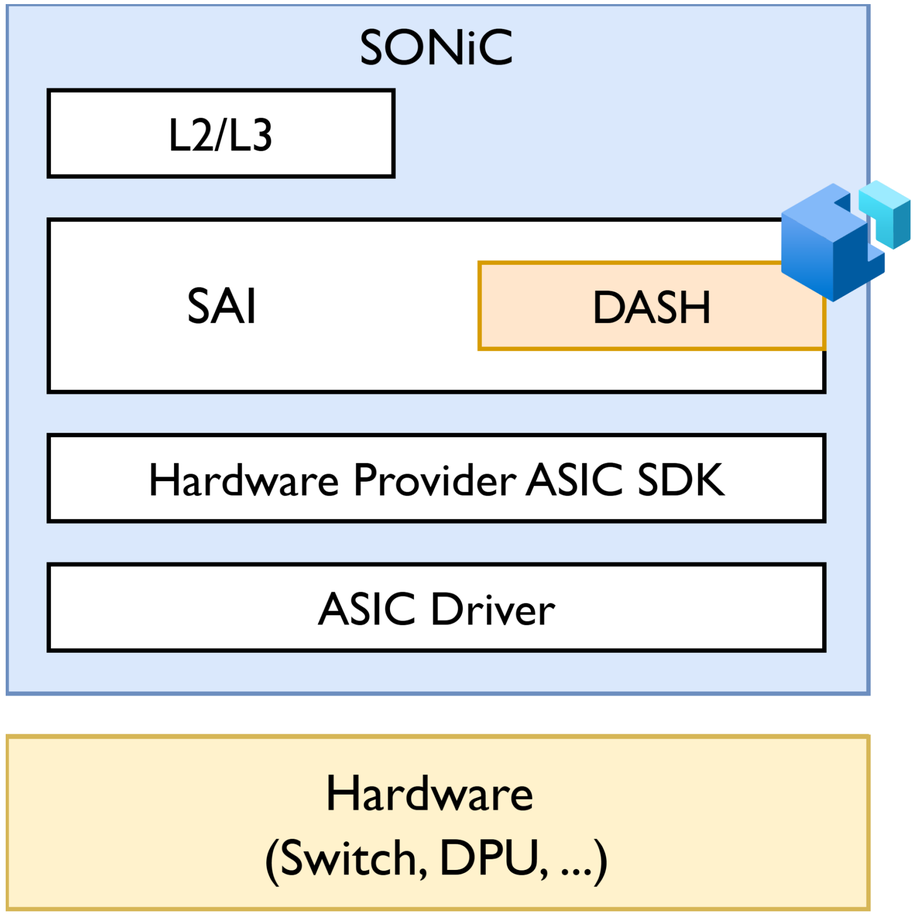
SONiC-DASH Initiative

Extending Switch Abstraction Interface (SAI) for SDN use cases with **Disaggregated APIs for SONiC Hosts** (DASH)

- Vendor-neutral
- Re-use mature code
- Compatible with SONiC deployments and ecosystem





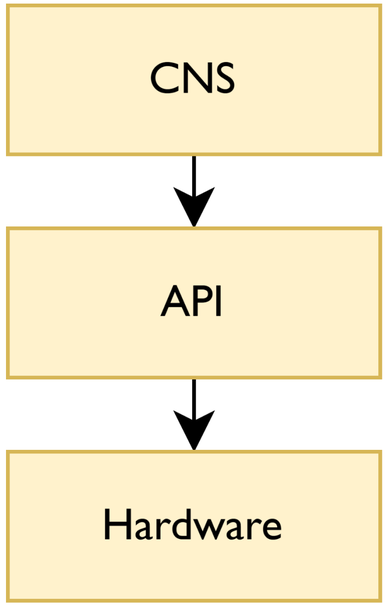


Ideal: Service-oriented & community-driven

SONiC-DASH Initiative

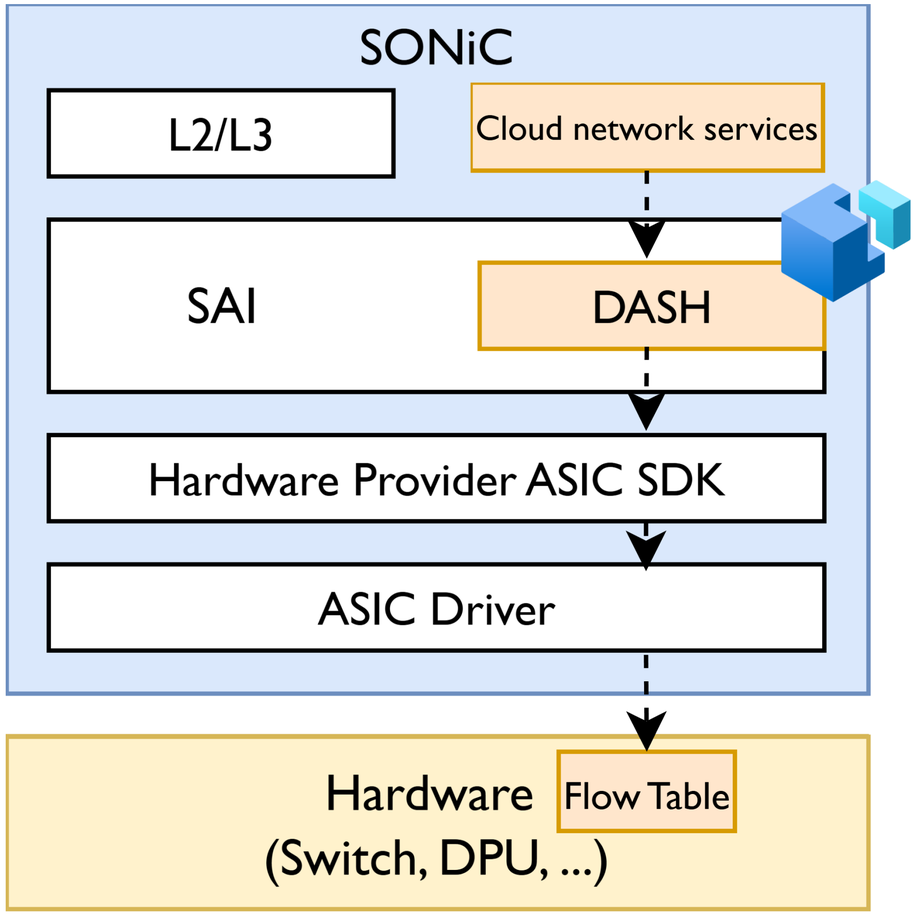
Extending Switch Abstraction Interface (SAI) for SDN use cases with **Disaggregated APIs for SONiC Hosts** (DASH)

- Vendor-neutral
- Re-use mature code
- Compatible with SONiC deployments and ecosystem



Ideal: Service-oriented & community-driven

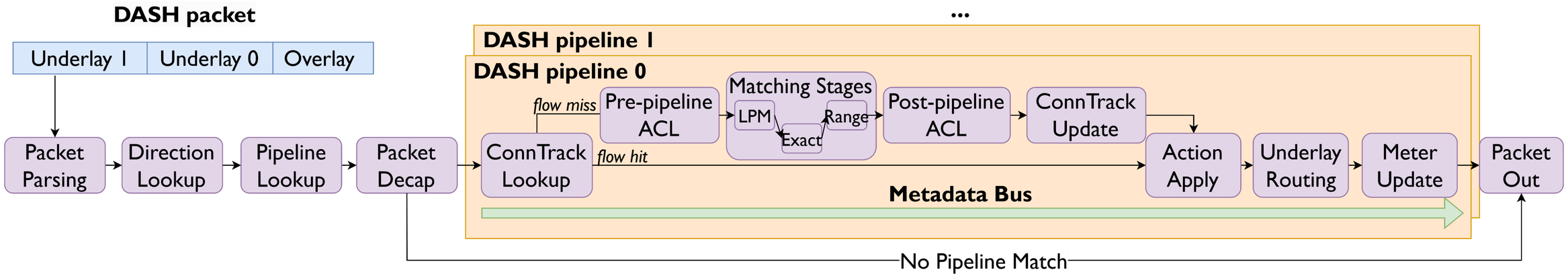
Result: state-of-the-art performance delivered at higher density & with vendor-neutrality



Modeling services with DASH pipeline

A **universal template MAT pipeline** to model L4 cloud network service cases

- Cloud network services share the same set of “functionality modules”



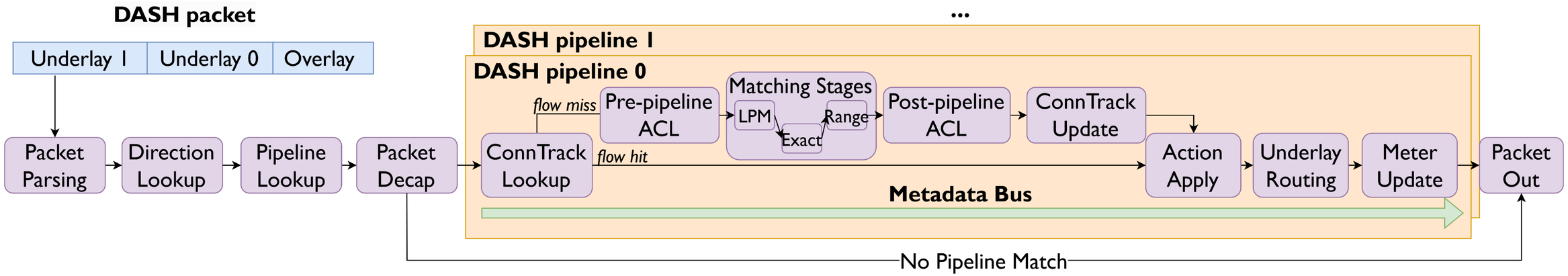
Modeling services with DASH pipeline

A **universal template MAT pipeline** to model L4 cloud network service cases

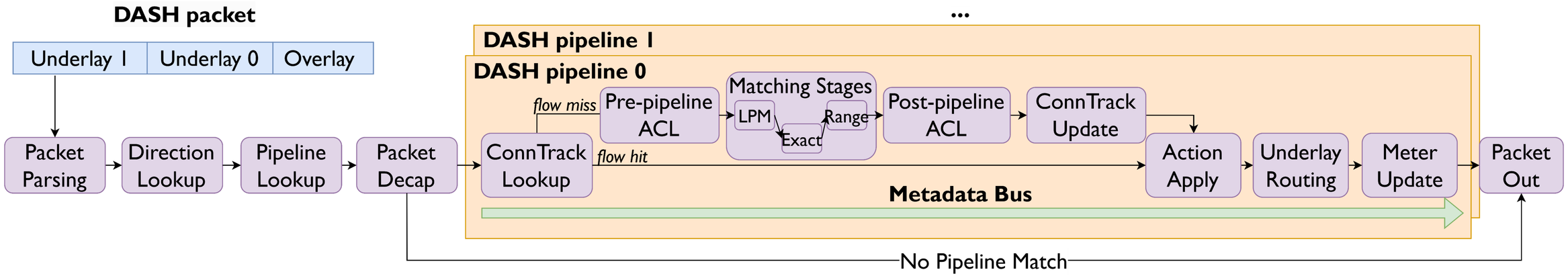
- Cloud network services share the same set of “functionality modules”
- Reflect the exact set of features required by major public clouds
- A clear goal for different hardware to target at and optimize against

Basic unit: stages

- A logical configurable block with certain SDN functionality
- Programming a service = applying specific configurations on the template pipeline



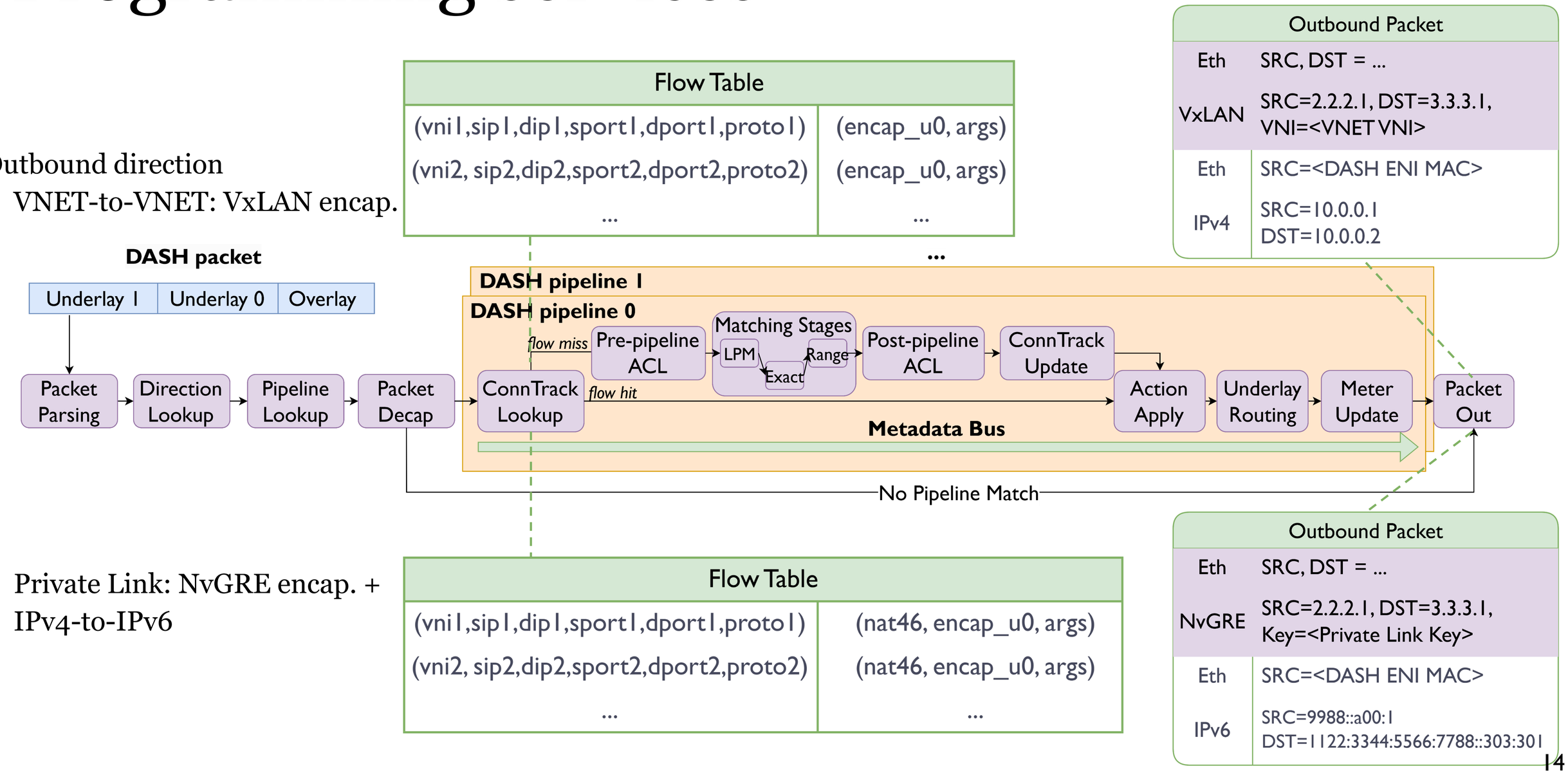
Programming services



Programming services

Outbound direction

VNET-to-VNET: VxLAN encap.

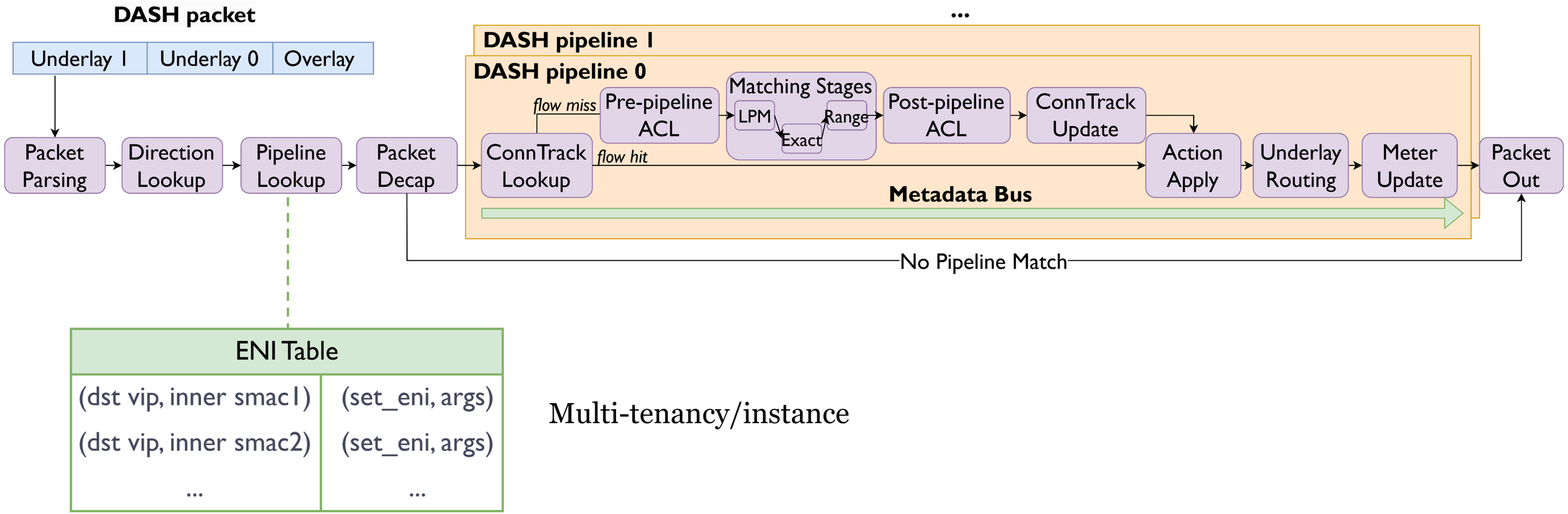


Private Link: NvGRE encap. + IPv4-to-IPv6

Flow Table	
(vni1,sip1,dip1,sport1,dport1,proto1)	(nat46, encap_u0, args)
(vni2, sip2,dip2,sport2,dport2,proto2)	(nat46, encap_u0, args)
...	...

Outbound Packet	
Eth	SRC, DST = ...
NvGRE	SRC=2.2.2.1, DST=3.3.3.1, Key=<Private Link Key>
Eth	SRC=<DASH ENI MAC>
IPv6	SRC=9988::a00:1 DST=1122:3344:5566:7788::303:301

Programming services

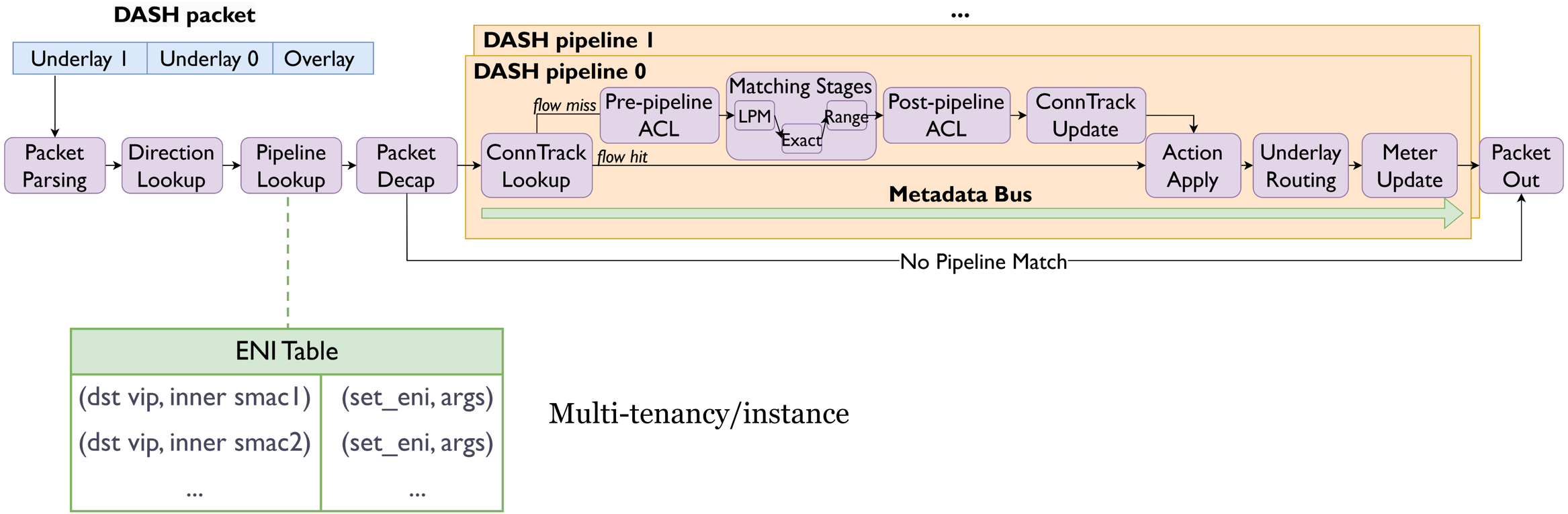


Multi-tenancy/instance

Programming services

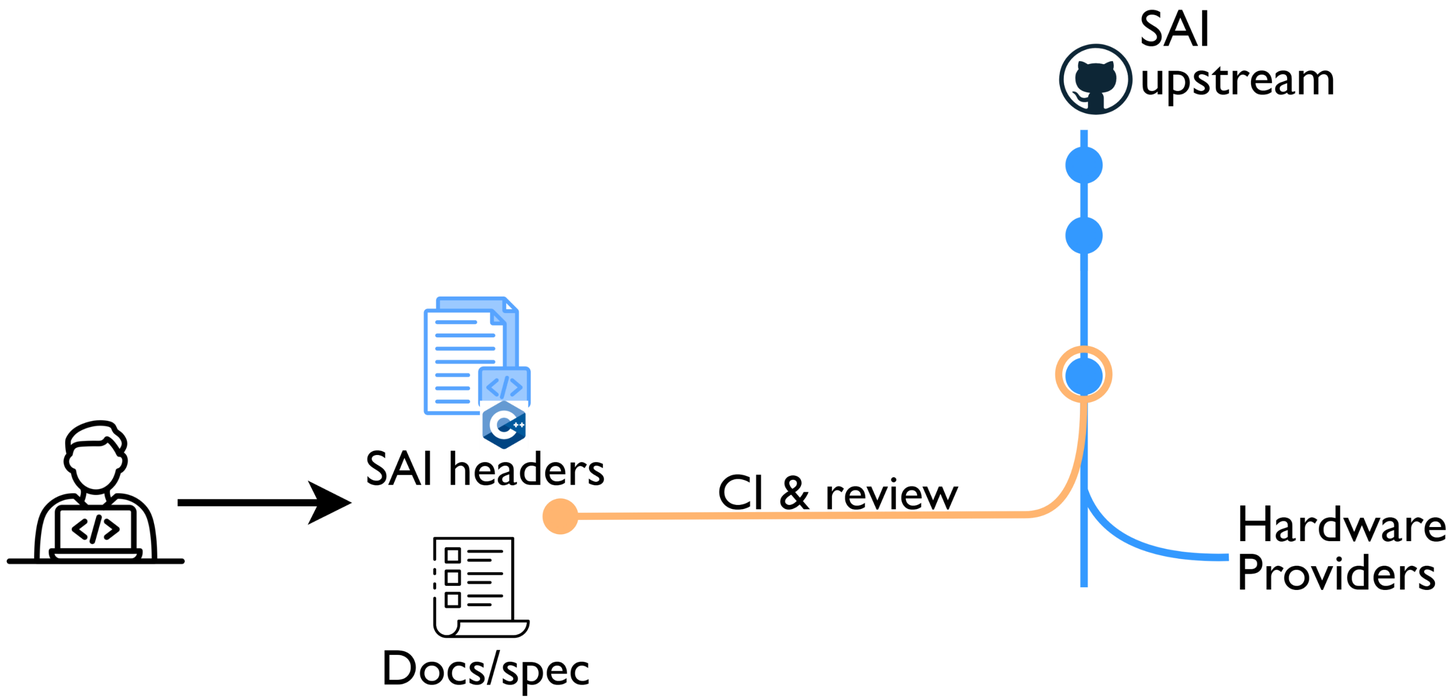
Service scenarios

VNET-to-VNET, VNET peering, private link, express route, stateful load balancer, NAT gateway, stateful ACL, metering, …

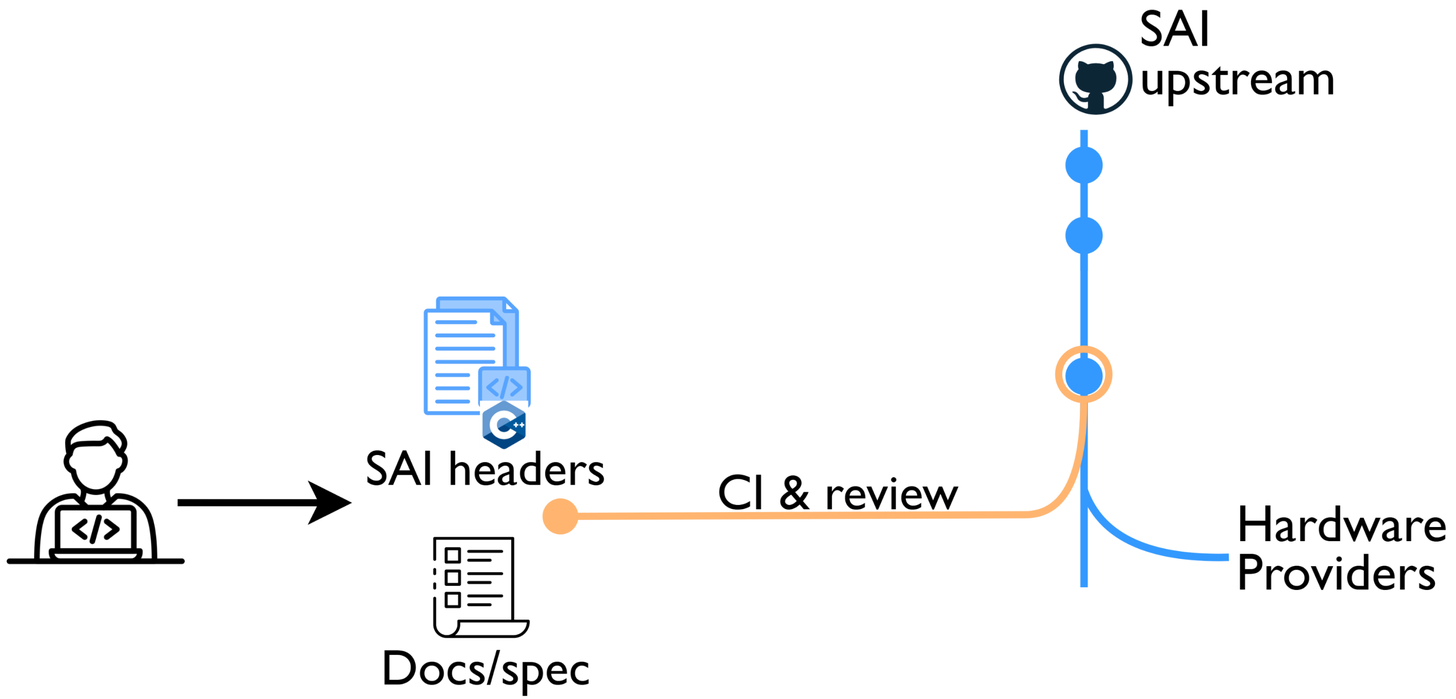


Multi-tenancy/instance

Making DASH pipeline a Behavior Model

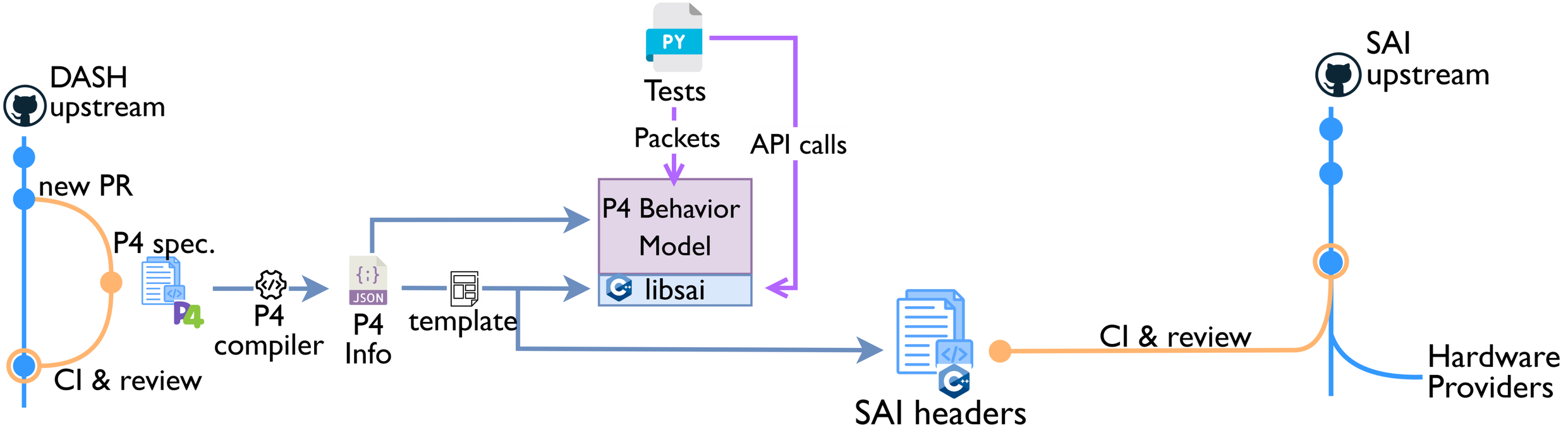


Making DASH pipeline a Behavior Model



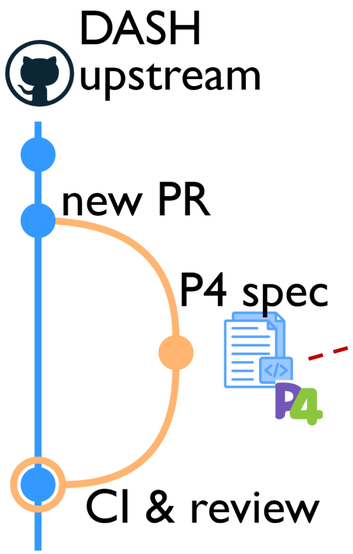
Vendor-neutrality	Achieve with unified SAI interfaces
Ambiguity	Specs written in English are inheritably ambiguous
Automated integration	APIs details handcrafted by developer

Making DASH pipeline a Behavior Model



Key idea: using P4 to model the behavior of services

P4 as the Behavior Model



Modelling service behaviour in **P4**

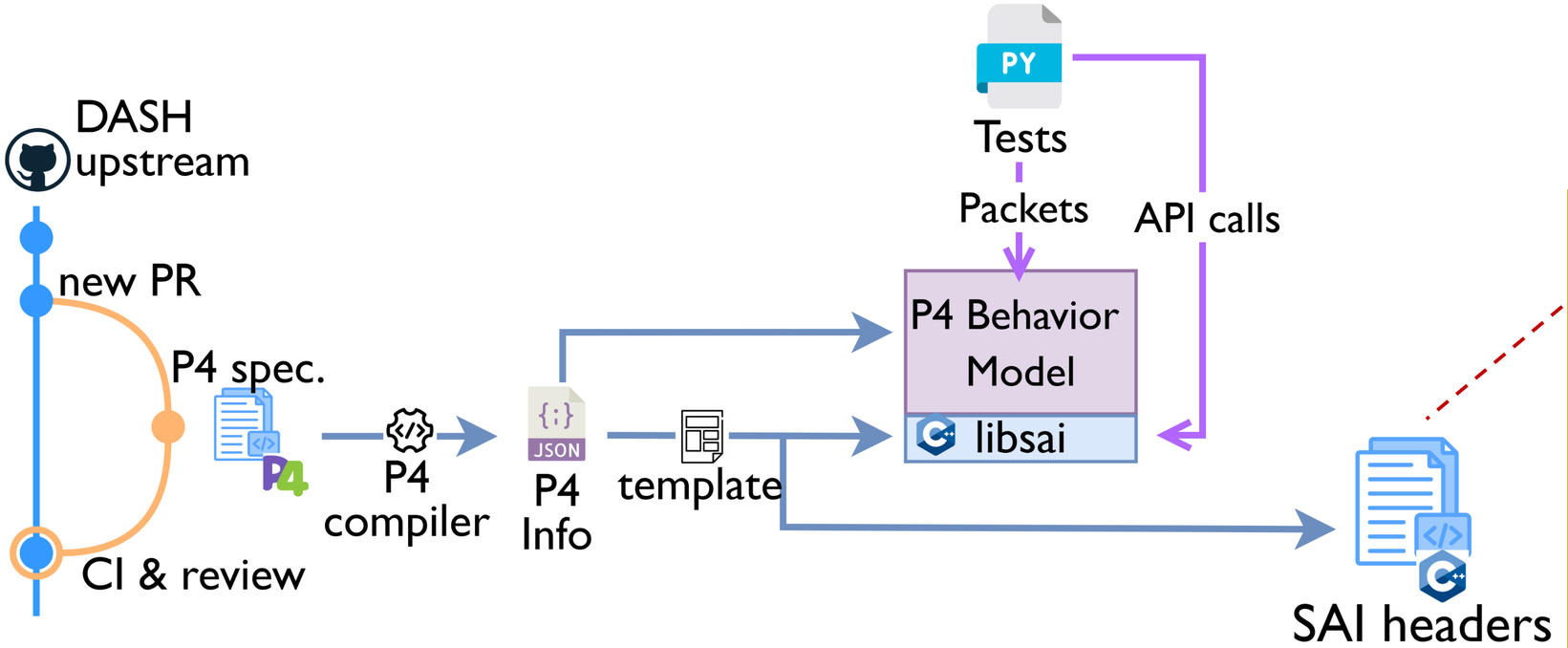
- verifiable, testable & unambiguous ground truth
- realistic for most hardware platforms
- executable & automating downstream tasks

conntrack_lookup.p4

```
control conntrack_lookup_stage(inout headers_t hdr, inout metadata_t meta){
    action set_flow_table_attr(dash_flow_enabled_key_t dash_flow_enabled_key, ...){
        meta.flow_table.flow_enabled_key = dash_flow_enabled_key;
        // Set other flow table attributes
        }
    @SaiTable[name="flow_table", api="dash_flow", order=0, isobject= "true"]
    table flow_table{
        key = {meta.flow_table.id:exact;}
        actions = {set_flow_table_attr;}
    }

    action set_flow_entry_attr(...){...}
    @SaiTable[name="flow", api="dash_flow", order=1, enable_bulk_get_api="true", \
    enable_bulk_get_server="true"] // Directives for SAI API generation.
    table flow_entry{
        key = {
            meta.flow_key.eni_mac:exact;
            meta.flow_key.vnet_id:exact;
            meta.flow_key.src_ip:exact;
            meta.flow_key.dst_ip:exact;
            meta.flow_key.src_port:exact;
            meta.flow_key.dst_port:exact;
            meta.flow_key.ip_proto:exact;
        }
        actions = {
            set_flow_entry_attr; // Action for publishing metadata for matched flow.
            @defaultonly flow_miss;
        }
        const default_action = flow_miss;
    }
    apply{
        flow_table.apply();
        if (meta.flow_table.flow_enabled_key & dash_flow_enabled_key_t.ENI_MAC != 0)
        {
           hdr.flow_key.eni_mac = meta.eni_addr; // Mask valid flow key fields.
        }
        ... // Other flow key fields are set similarly.
        flow_entry.apply();
    }
}
```

Community-Driven Development



saiexperimentaldashflow.h

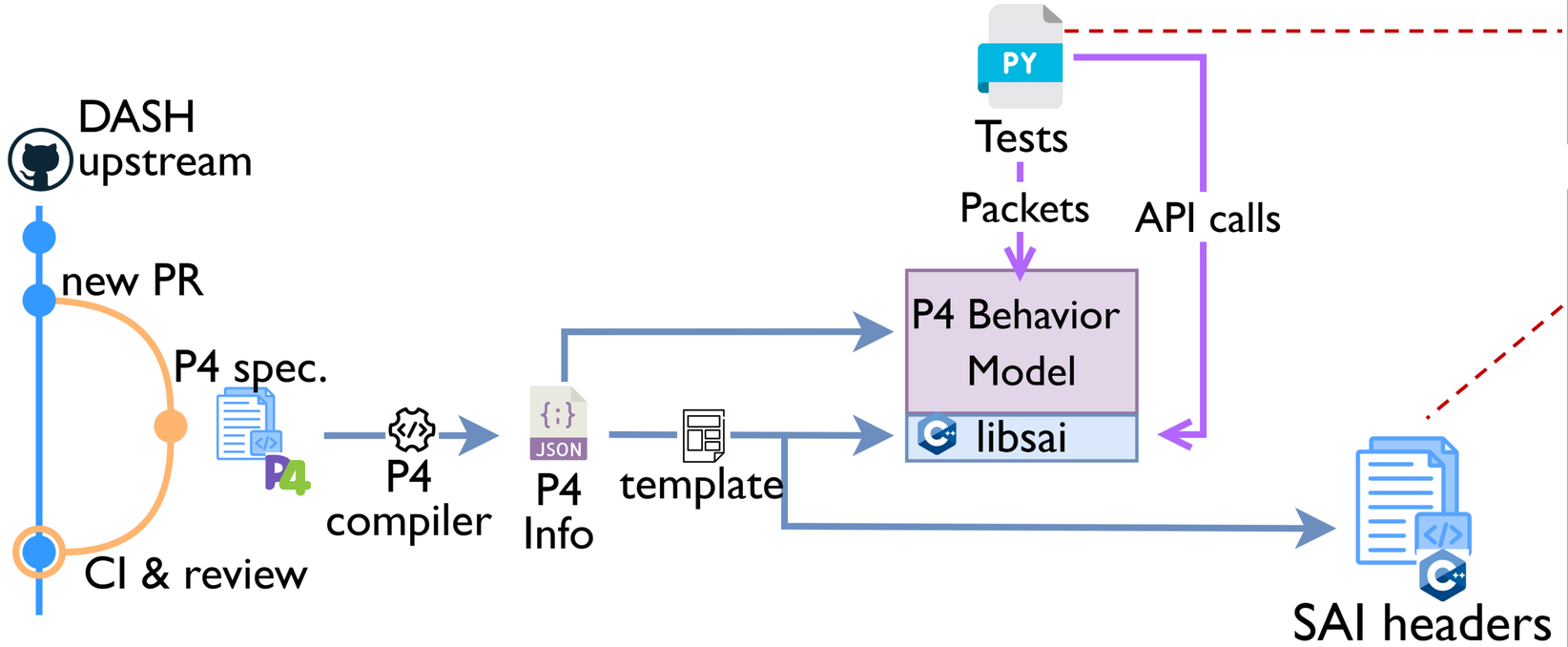
```
typedef struct _sai_flow_entry_t
{
    sai_object_id_t switch_id;
    sai_mac_t eni_mac;
    sai_uint16_t vnet_id;
    sai_uint8_t ip_proto;
    sai_ip_address_t src_ip;
    sai_ip_address_t dst_ip;
    sai_uint16_t src_port;
    sai_uint16_t dst_port;
} sai_flow_entry_t;

/**
 * @brief Create flow entry
 * @param[in] flow_entry Entry
 * @param[in] attr_count Number of attributes
 * @param[in] attr_list Array of attributes
 * @return #SAI_STATUS_SUCCESS on success Failure
 *         status code on error
 */
typedef sai_status_t (*sai_create_flow_entry_fn)(
        _In_ const sai_flow_entry_t *flow_entry,
        _In_ uint32_t attr_count,
        _In_ const sai_attribute_t *attr_list);
// Other APIs for flow entry...
typedef sai_status_t (*sai_create_flow_table_fn)(
        _Out_ sai_object_id_t *flow_table_id,
        _In_ sai_object_id_t switch_id,
        _In_ uint32_t attr_count,
        _In_ const sai_attribute_t *attr_list);
// Other APIs for flow table...
```

DASH BM -> SAI headers

- Leveraging mature toolchain of P4
- Auto-generating DASH headers conforming to SAI standards with templates

Community-Driven Development



saidashflow.py

```
sai_thrift_create_flow_table()
sai_thrift_create_flow_entry()
...
send_packet()
verify_packet()
```

saiexperimentaldashflow.h

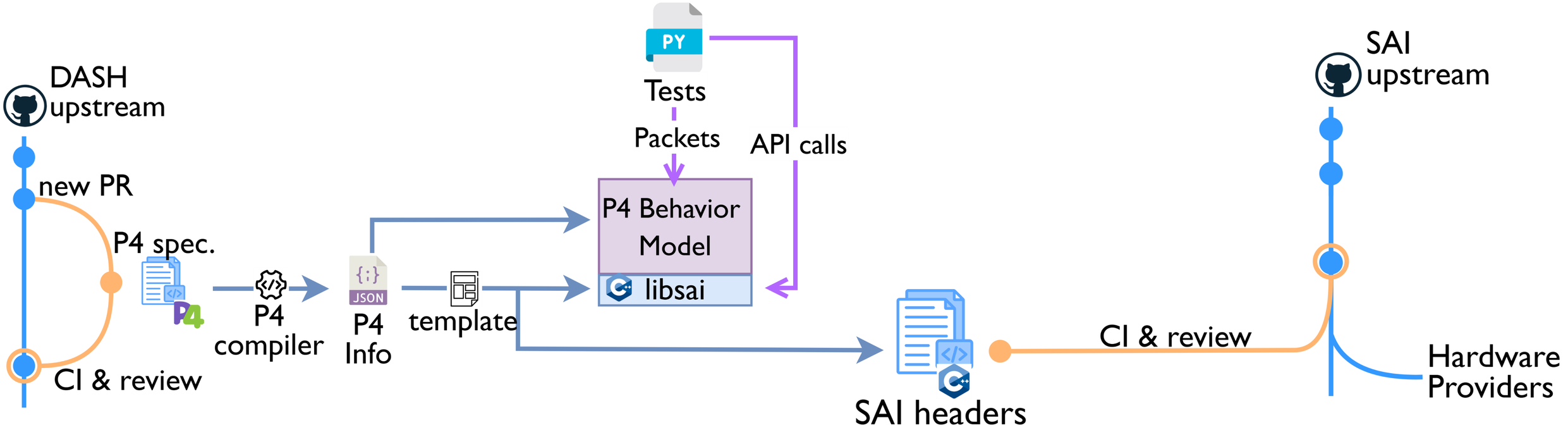
```
typedef struct _sai_flow_entry_t
{
    sai_object_id_t switch_id;
    sai_mac_t eni_mac;
    sai_uint16_t vnet_id;
    sai_uint8_t ip_proto;
    sai_ip_address_t src_ip;
    sai_ip_address_t dst_ip;
    sai_uint16_t src_port;
    sai_uint16_t dst_port;
} sai_flow_entry_t;

/**
 * @brief Create flow entry
 * @param[in] flow_entry Entry
 * @param[in] attr_count Number of attributes
 * @param[in] attr_list Array of attributes
 * @return #SAI_STATUS_SUCCESS on success Failure
 *         status code on error
 */
typedef sai_status_t (*sai_create_flow_entry_fn)(
        _In_ const sai_flow_entry_t *flow_entry,
        _In_ uint32_t attr_count,
        _In_ const sai_attribute_t *attr_list);
// Other APIs for flow entry...
typedef sai_status_t (*sai_create_flow_table_fn)(
        _Out_ sai_object_id_t *flow_table_id,
        _In_ sai_object_id_t switch_id,
        _In_ uint32_t attr_count,
        _In_ const sai_attribute_t *attr_list);
// Other APIs for flow table...
```

DASH BM -> SAI headers
-> shift-left testing

- Leveraging mature toolchain of P4
- Auto-generating DASH headers conforming to SAI standards with templates
- Executable on reference software switch

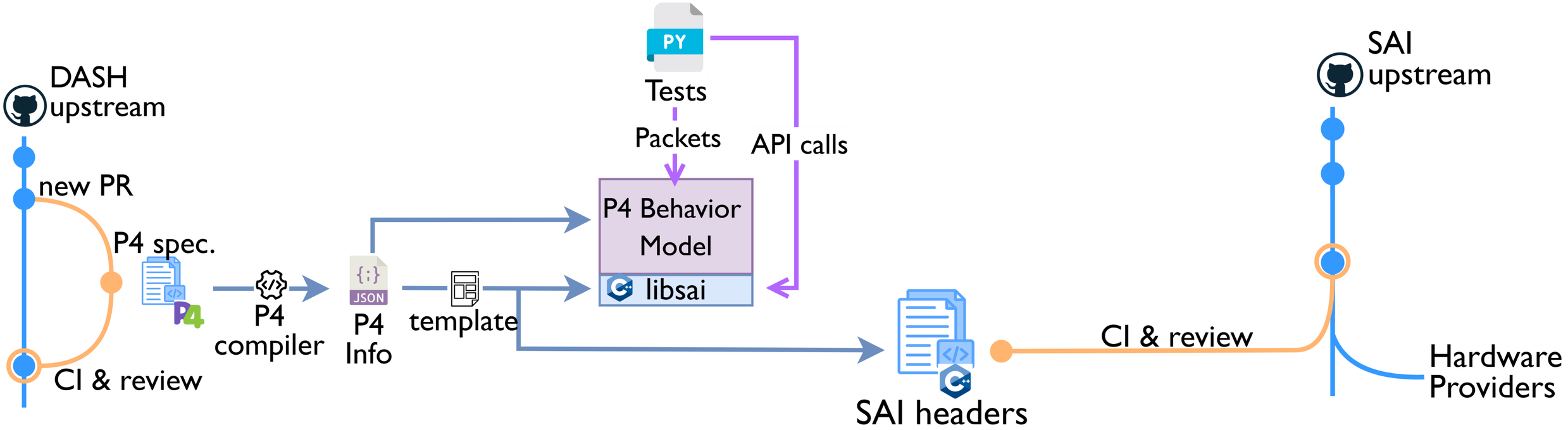
Community-Driven Development



Hardware technology providers implement and optimize DASH data plane & APIs on their products

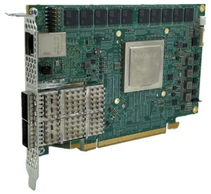
- DASH BM as the ground truth
- Shift-left test cases can be directly used for testing real devices

Community-Driven Development



Hardware technology providers implement and optimize DASH data plane & APIs on their products

- DASH BM as the ground truth
- Shift-left test cases can be directly used for testing real devices

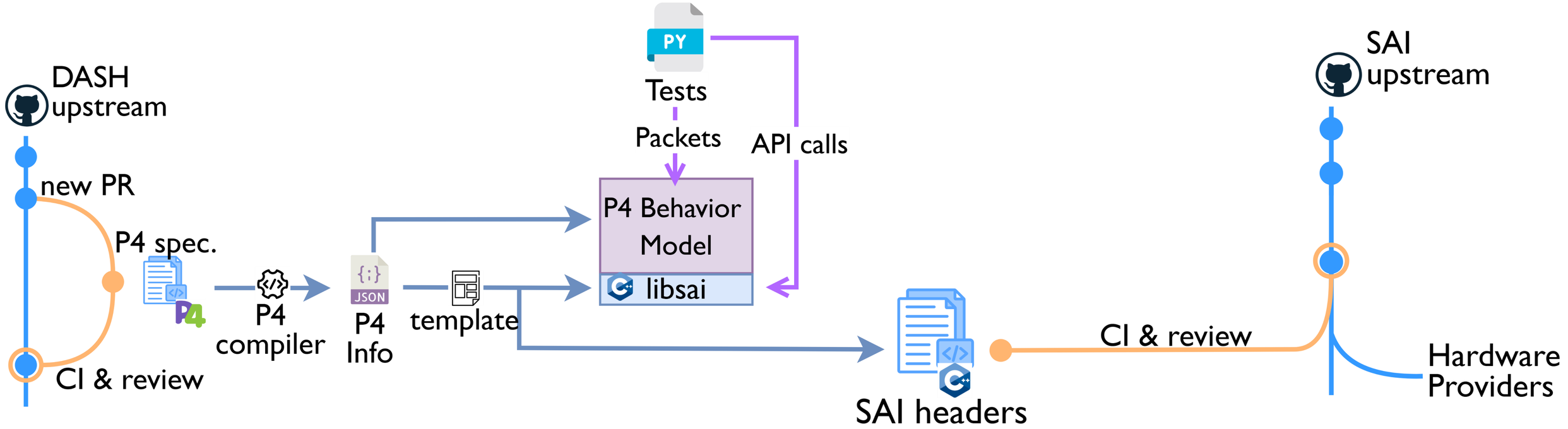


...

Multi-sourced, best-of-the-class options are available with

- conformance guarantees on any DASH-capable device
- vendor-specific competitive edge
- cloud-oriented features

Community-Driven Development

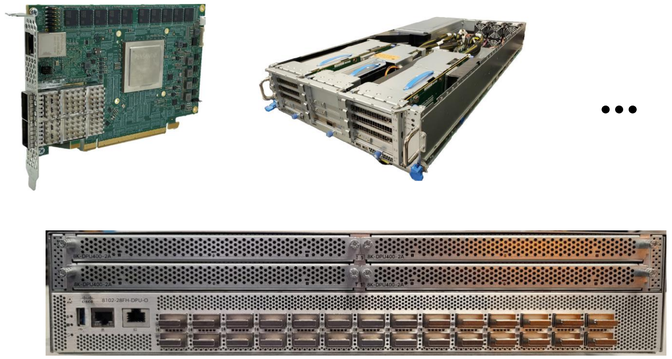


Hardware technology providers implement and optimize DASH data plane & APIs on their products

- DASH BM as the ground truth
- Shift-left test cases can be directly used for testing real devices

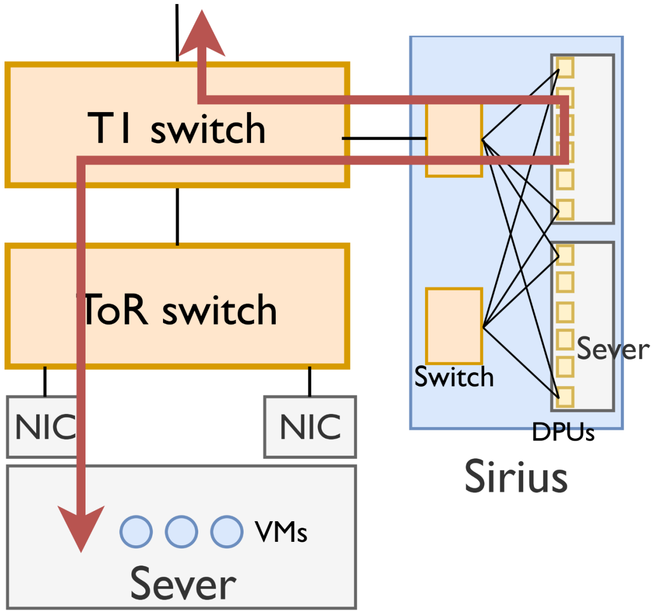
Multi-sourced, best-of-the-class options are available with

- conformance guarantees on any DASH-capable device
- vendor-specific competitive edge
- cloud-oriented features



DASH-capable SmartSwitch

Key motivation for a SmartSwitch architecture

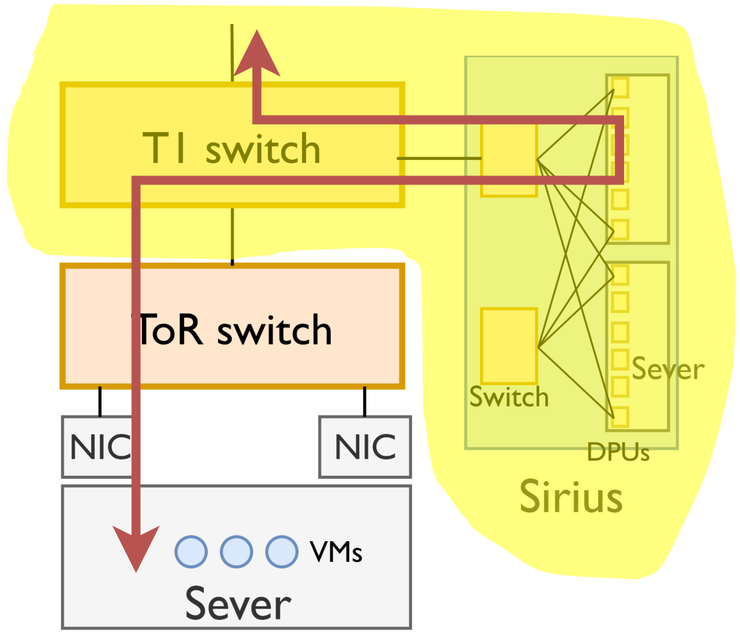


High performance, flexible scale-up, resource pooling, high availability.

High rack space usage and power footprint
Customized build-out, onboarding, and deployment
Detours

Key motivation for a SmartSwitch architecture

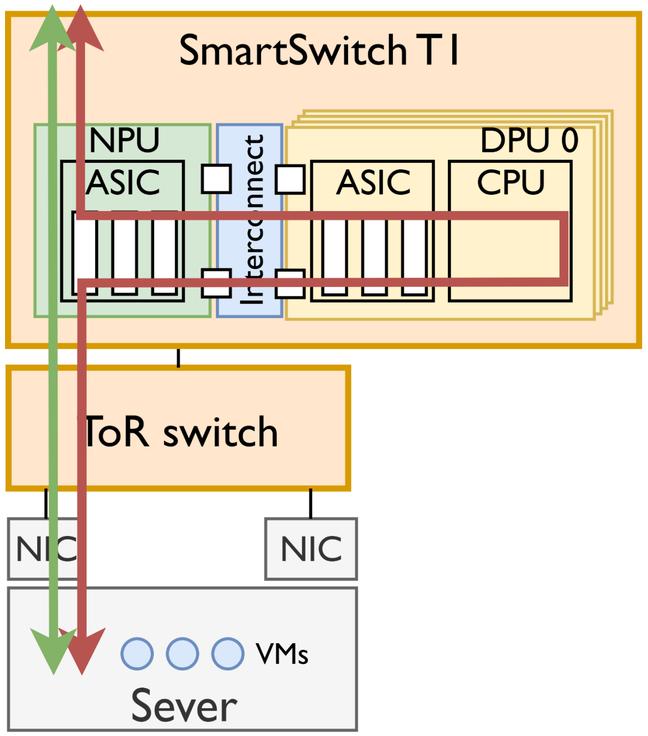
Converging the role of switches and DPU appliance.



High performance, flexible scale-up, resource pooling, high availability.

High rack space usage and power footprint
Customized build-out, onboarding, and deployment
Detours

Key motivation for a SmartSwitch architecture

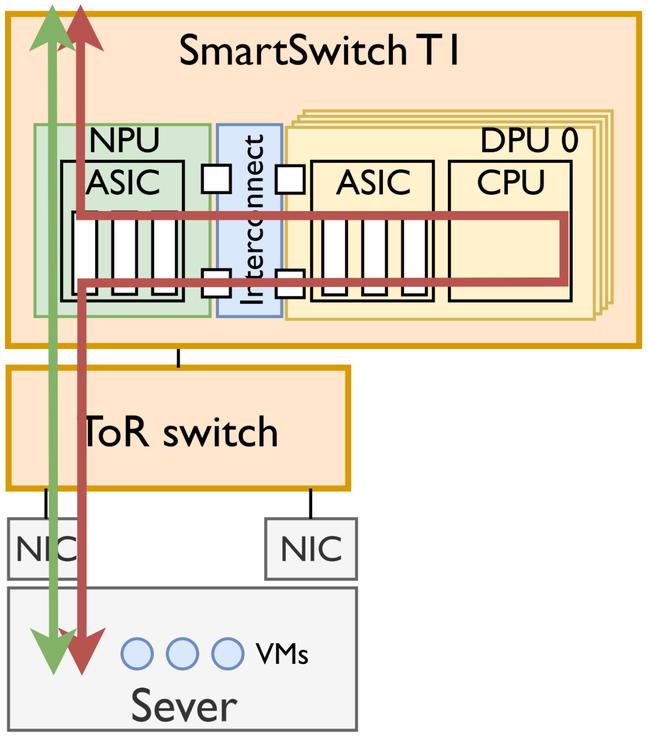


Converging the role of switches and DPU appliance.

Uni-box SmartSwitch
Performance scalability (similar to DPU-centric model)
High performance, flexible scale-up, resource pooling, high availability.

High rack space usage and power footprint
Customized build-out, onboarding, and deployment
Detours

Key motivation for a SmartSwitch architecture

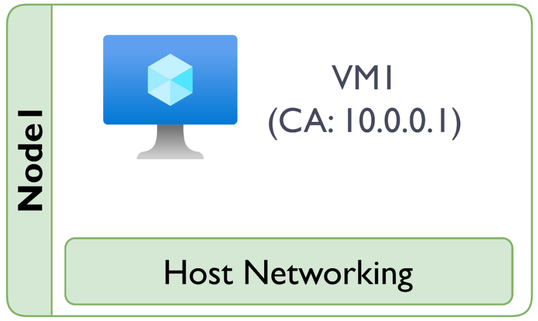


Converging the role of switches and DPU appliance.

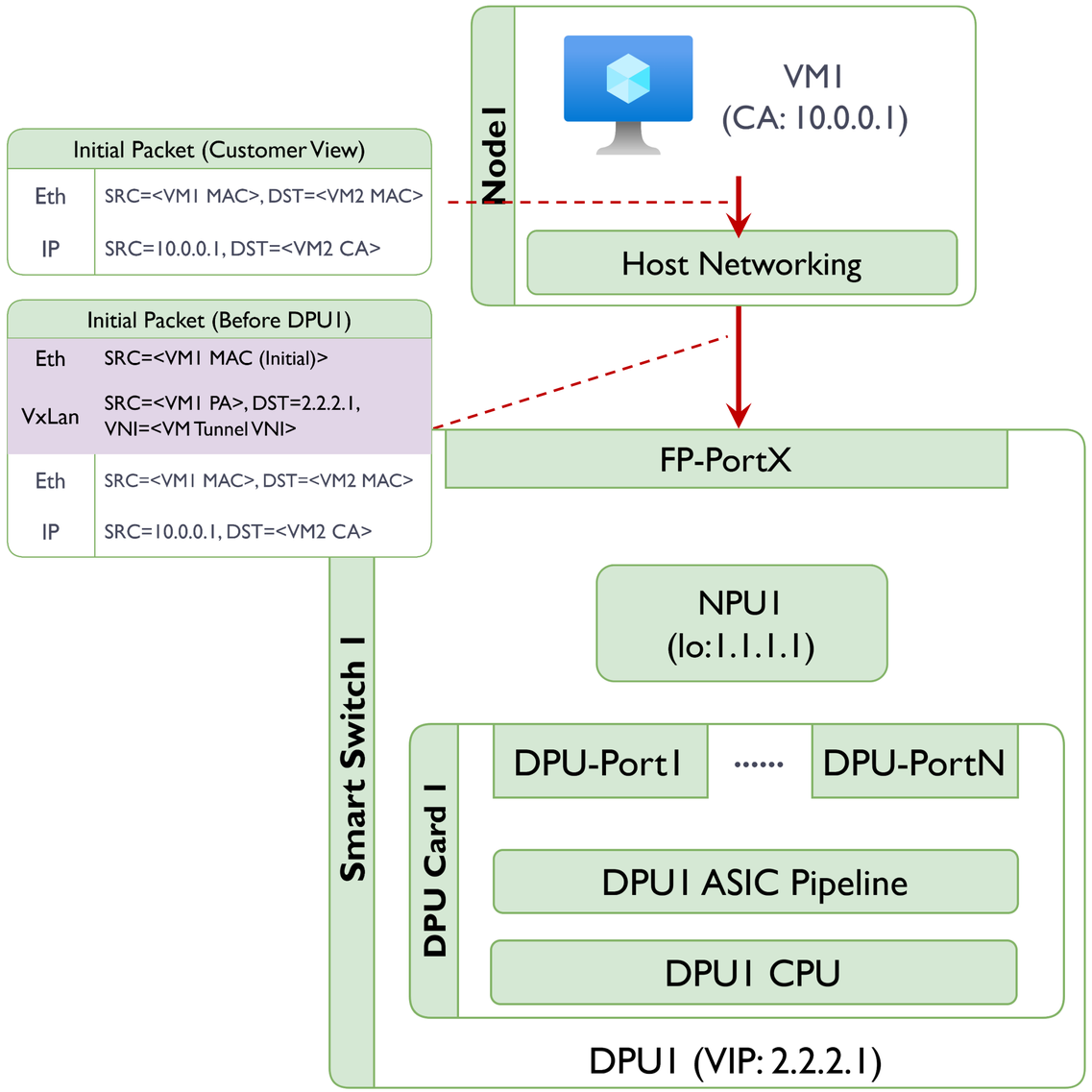
Uni-box SmartSwitch
Performance scalability (similar to DPU-centric model)
High performance, flexible scale-up, resource pooling, high availability

Physical deployability (similar to NPU-centric model)
~~High rack space usage and power footprint~~
~~Customized build-out, onboarding, and deployment~~
~~Detours~~

SmartSwitch Offload Datapath



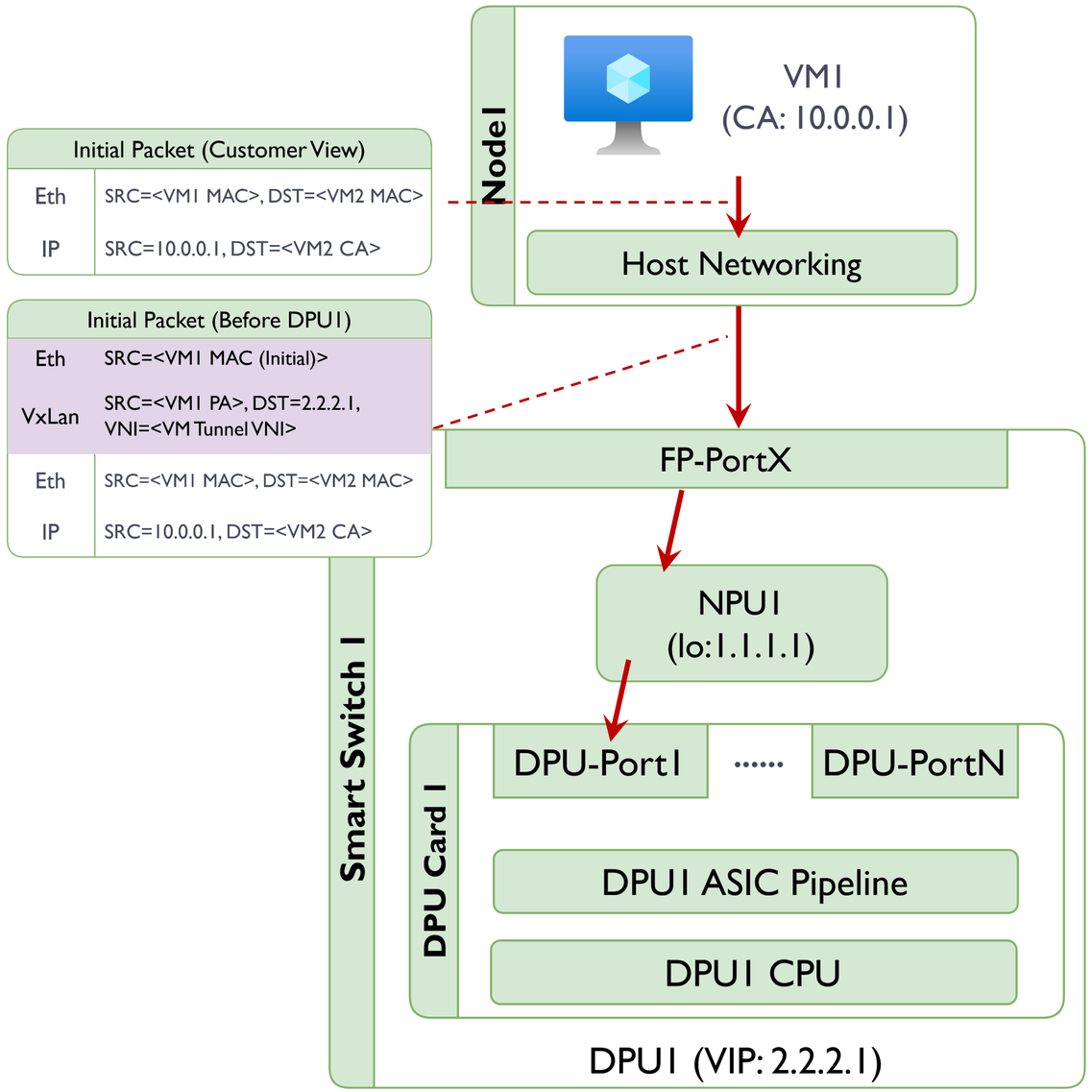
SmartSwitch Offload Datapath



Reachability Host-SS tunnel

- Stateless encap with reserved VNI and SS VIP
- Minimal overhead for hosts

SmartSwitch Offload Datapath



Reachability

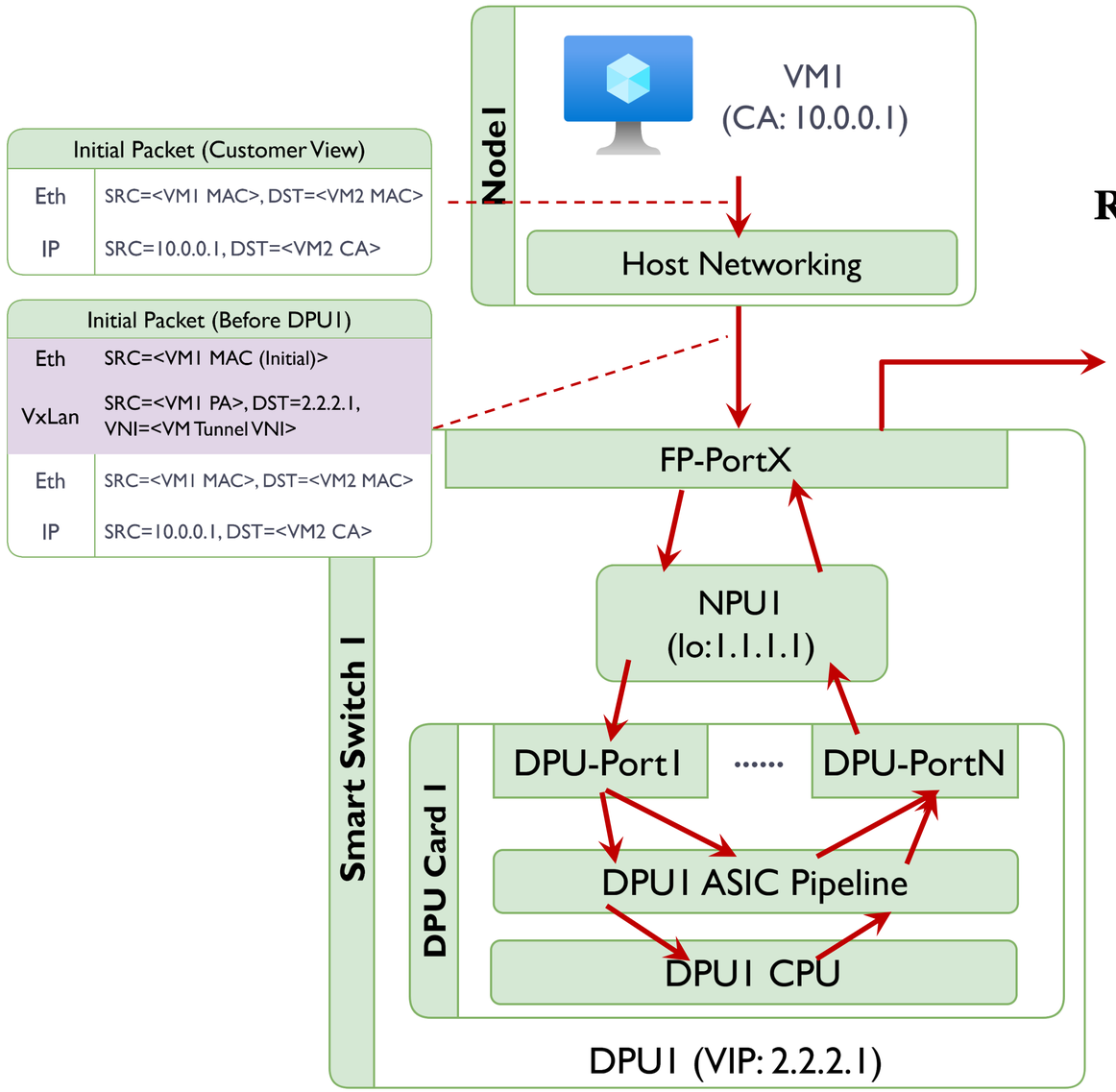
Host-SS tunnel

- Stateless encap with reserved VNI and SS VIP
- Minimal overhead for hosts

Front-panel NPU

- VIP announced via BGP and shared per SS/per row
- ENI-based forwarding (ACL rules <VIP, ENI MAC>)

SmartSwitch Offload Datapath



Reachability

Host-SS tunnel
- Stateless encap with reserved VNI and SS VIP
- Minimal overhead for hosts

Front-panel NPU
- VIP announced via BGP and shared per SS/per row
- ENI-based forwarding (ACL rules <VIP, ENI MAC>)

DPU for performant stateful packet processing
- Including heavy use cases (middleboxes)

Other aspects

- High availability & state consistency algorithms
  - https://github.com/sonic-net/SONiC/blob/master/doc/smart-switch/high-availability/smart-switch-ha-hld.md
- SONiC integrations for DASH & SmartSwitch
  - https://github.com/sonic-net/SONiC/blob/master/doc/dash/dash-sonic-hld.md
  - https://github.com/sonic-net/SONiC/tree/master/doc/smart-switch
- Performance optimizations on specific SmartSwitch architecture
  - NPU-DPU co-processing
- …

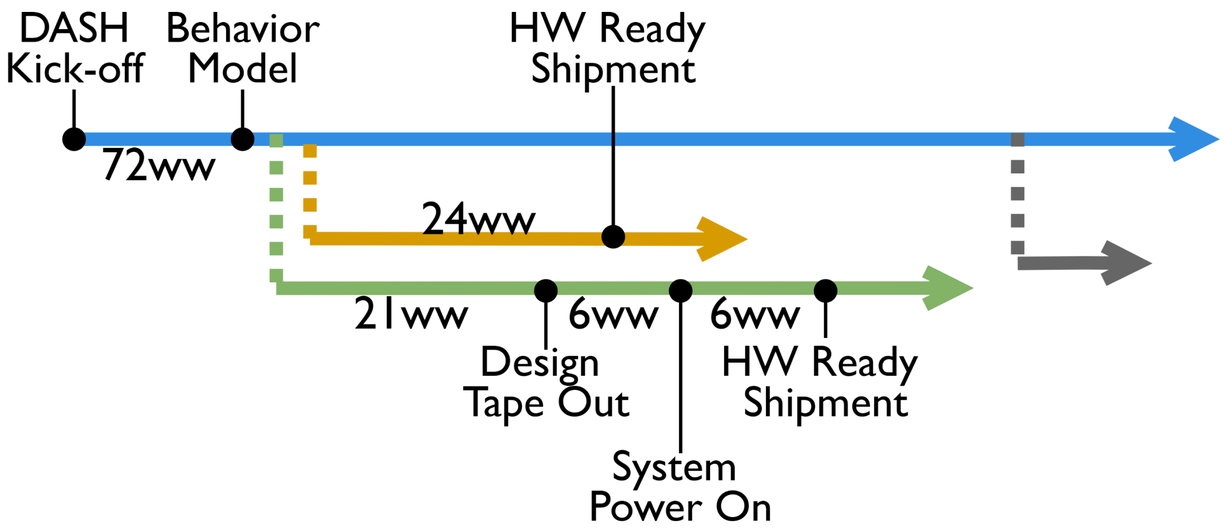
Operating the development model in Azure

Operating the development model in Azure

DASH driven by **community contributions**

- Collective efforts by hundreds of community members
- Supported on >10 platforms and is organically growing

Operating the development model in Azure



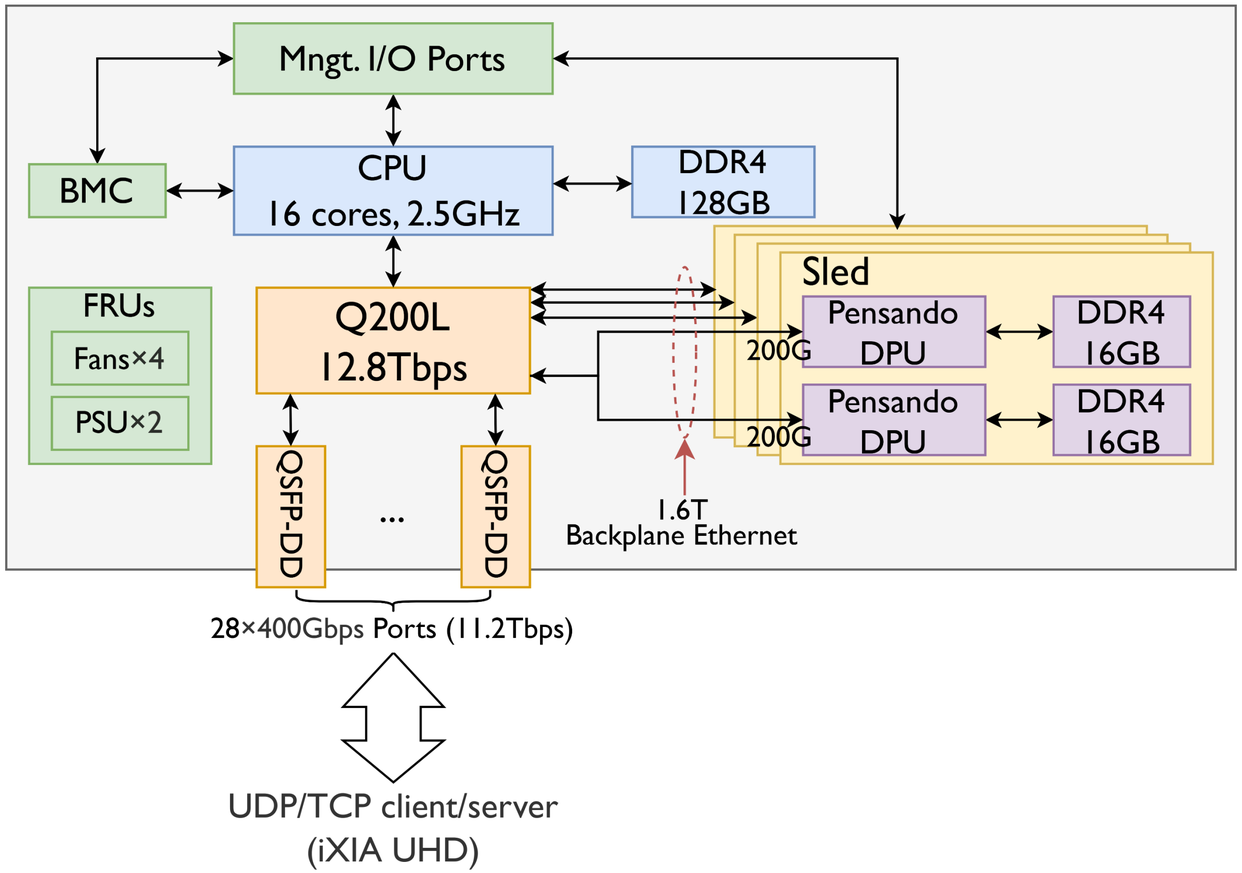
DASH driven by **community contributions**

- Collective efforts by hundreds of community members
- Supported on >10 platforms and is organically growing

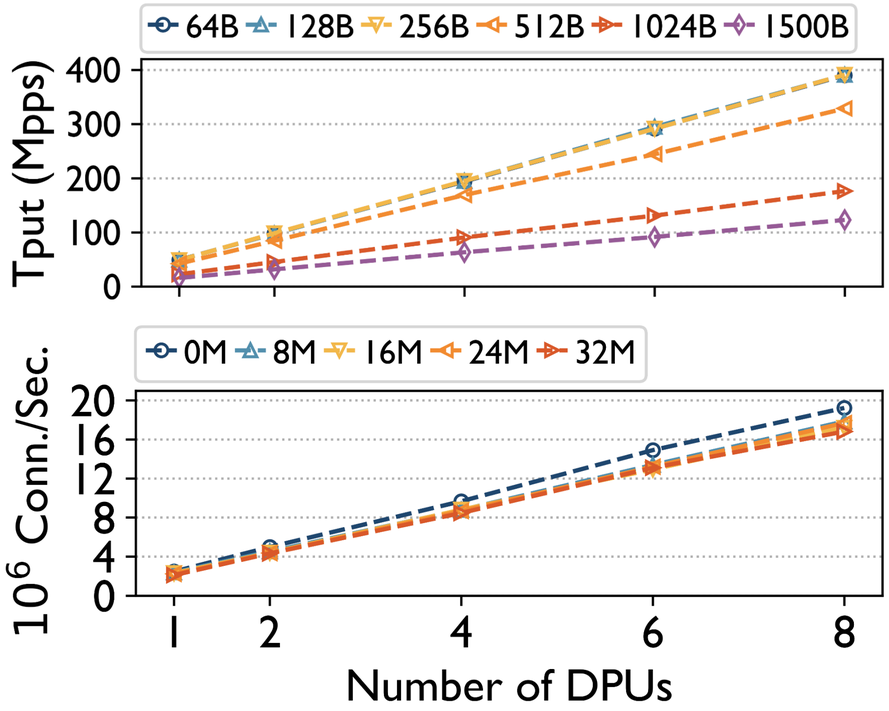
SmartSwitches in Azure

- Implemented by two vendors and deployed at scale
- 100s of work-weeks saved

Evaluation


Mngt. I/O Ports
BMC
CPU
16 cores, 2.5GHz
DDR4
128GB
FRUs
Fans×4
PSU×2
Q200L
12.8Tbps
Sled
Pensando
DPU
DDR4
16GB
Pensando
DPU
DDR4
16GB
200G
200G
QSFP-DD
...
QSFP-DD
1.6T
Backplane Ethernet
28×400Gbps Ports (11.2Tbps)
UDP/TCP client/server
(iXIA UHD)

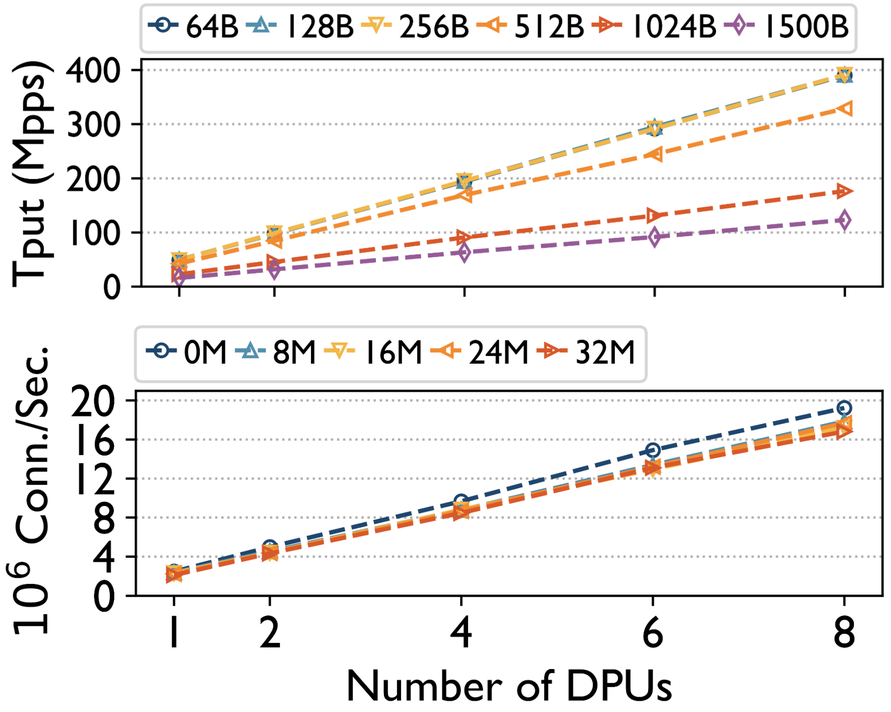
Evaluation

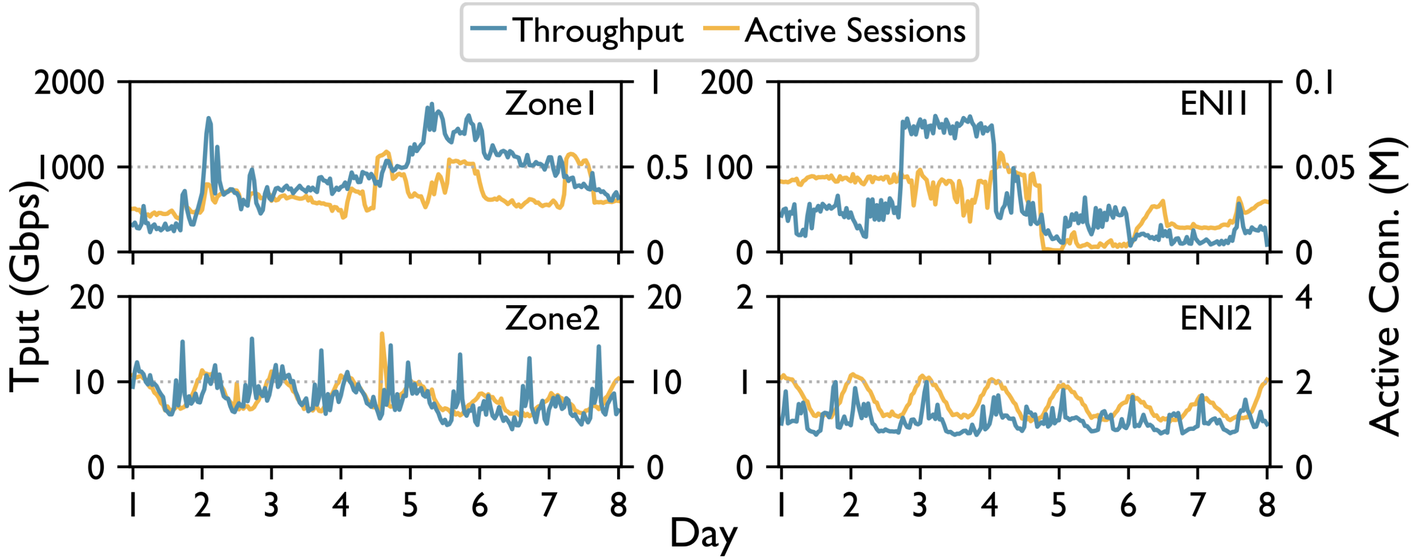


* 32 ENIs + 32 VNETs

- Tput: scaling to **391.02Mpps** (for small packets) & **1.53Tbps** (for large packets) at **256M** connections
- CPS: scaling to **19.22M CPS**

Evaluation





* 32 ENIs + 32 VNETs

- Tput: scaling to **391.02Mpps** (for small packets) & **1.53Tbps** (for large packets) at **256M** connections
- CPS: scaling to **19.22M CPS**

Serving traffic with different patterns (including heavy use cases) in production, with reasonable capacity for organic growth

Footprint & efficiency benefits

Metric	Sirius	SmartSwitch
Power (W)	1406.65	**1040.83**
Space (RU)	4	**2**
Power Efficiency (Mpps/kW)	207.11	**374.76**
Space Efficiency (Mpps/RU)	72.83	**195.03**
Cost Efficiency[1] (Mpps/K$)	2.80	**3.73**
Savings (# cores)	530	**709**

1. Calculated with list price. Product models may not exactly match the ones used by Sirius and SmartSwitch, but we use same model to fairly calculate and compare their cost efficiency.

2RU SmartSwitch

- +1RU, +0 wiring, +449W over T1 switch
- **1.81×**, **2.68×**, and **1.33×** power-, space- and cost-efficiency (calculated with list price) compared to DPU-centric Sirius

State-of-the-art performance delivered at higher density & w/ less $.

Operational Experiences

- Reduce hard downtime with rolling updates.
- Use To for bulky flow synchronization.
  - FHA[1]: bulky flow state sync for high availability on T1 SmartSwitches
- Fine-grained debugging needed for stateful workloads.

[1] Ying Chu, Ziyuan Liu, Riff Jiang, Ze Gan, Junhua Zhai, Guohan Lu, Zhixiong Niu, and Yongqiang Xiong. FHA: Flow-level High Availability on Programmable Network Hardware for Cloud Provider. . In Proc. ACM APsys, 2024.

Summary

Operational lessons from evolutions of Azure's CNS offload solutions

Novel techniques & paradigms

- Vendor-neutral DASH
- Pipeline programming model
- Uni-box SmartSwitch architecture
- Community-driven development

State-of-the-art performance delivered at **higher density** and with **less $**
Saved work-weeks & **multi-sourcing**

SONiC DASH SmartSwitch under **testing & deployment at scale** in Azure

- Lessons on firmware updates, state sync, and debugging

https://github.com/sonic-net/dash

https://sonicfoundation.dev

Thank you.
Q&A

https://github.com/sonic-net/dash

https://sonicfoundation.dev