Bringing Together a Kubernetes-Native Network Operator and P4 Programmable Testbed

Submitted by Mohammad Firas Sada (mfsada@es.net)
on behalf of the involved parties listed herein

SC26 Network Research Exhibition

1 Abstract

The convergence of programmable data planes and Kubernetes has long been hampered by an asymmetry: while compute and storage are declarative resources (Pod, PersistentVolumeClaim), guaranteed-bandwidth WAN circuits and P4-programmable hardware have remained outside the Kubernetes control plane. At SC26 we close this gap with a fully Kubernetes-native P4 testbed that treats network paths as first-class resources. The SENSE Operator (a production Kubernetes operator elevating multi-domain L2 circuit provisioning to a SensePath CRD) dynamically provisions deterministic, QoS-enabled optical paths from SCinet to a geographically distributed P4 infrastructure (SmartNICs, BlueField-2 DPUs, Tofino2 switch) integrated into the NRP Nautilus cluster. Two novelties: (1) a Kubernetes-native network resource abstraction via the operator's declarative reconcile loop, and (2) a production P4 testbed living entirely inside Kubernetes, with programmable data planes accessible via `kubectl`, from SmartNIC bitstream deployment to live packet telemetry.

2 Goals

- SENSE Operator as network resource manager: attendees run `kubectl apply -f sensepath.yaml` to instantiate a 100 Gbps L2 VLAN path from SCinet to an Alveo U55C SmartNIC in the NRP cluster, with automatic Multus NetworkAttachmentDefinition (NAD) creation binding the circuit to a pod's secondary interface.
- Kubernetes-native P4 testbed: P4-programmable devices (FPGAs, DPUs, Tofino2 switch) modeled as Kubernetes resources via Device Plugins, SR-IOV, and Multus; live deployment/reconfiguration of P4 programs on U55Cs and BlueField-2 DPUs using `kubectl exec` into operator-managed pods.
- End-to-end multi-domain programmability: SCinet (Chicago) → SENSE-provisioned path → StarLight → NRP P4 nodes (SDSC, PacWave, FABRIC); real-time telemetry via P4–DPDK–Kafka pipelines, with the operator's status mirror exposing assigned VLANs and circuit UUIDs.
- Reproducible, idempotent WAN circuit lifecycle: finalizer-guaranteed teardown (`kubectl delete sensepath`) cascades to circuit cancellation and NAD removal, eliminating manual out-of-band cleanup.
- Validation of the four-component operator pattern (declarative CRD, reconciler API bridge, status mirror, finalizer) as a template for network-as-code cyberinfrastructure.
- Security posture: unprivileged P4-DPDK virtual switches (NET_ADMIN / IPC_LOCK contexts) prove that in-network data planes can be staged without elevating the testbed to privileged access.

3 Resources

- SENSE Operator, running in production on the NRP Nautilus cluster, polling the SENSE Orchestrator REST API with OIDC auth.
- SCinet-connected Nautilus nodes (SC26 exhibit floor) running Kubernetes v1.31 + Multus CNI + SR-IOV.
- P4 programmable testbed: 32 U55C FPGAs (2×100G P4-programmable) in Nautilus; 7 BlueField-2 DPUs at PacWave peering points (SDSC, Sunnyvale, LA, Seattle, NYSERNet, StarLight, Guam) running DOCA DPL with P4-16 support; Tofino2 400G switch (Arista, SONiC PINS); Xilinx SN1000 SmartNICs and U280 FPGAs (FABRIC-CIEN mobile rack).
- ESnet SENSE Orchestrator provisioning L2 VLAN paths with deterministic QoS across SCinet, StarLight, Internet2, and FABRIC.
- Open-source tooling: ESnet SmartNIC stack (OpenNIC shell, DPDK-pktgen, automated bitstream deployment), P4-DPDK virtual switches (unprivileged, NET_ADMIN/IPC_LOCK security contexts).

4 Impact

This NRE is the first to combine a Kubernetes-native network resource orchestrator with a multisite P4 testbed fully embedded in a Kubernetes cluster. By removing the asymmetry between declarative compute/storage and imperative WAN circuit provisioning, the SENSE Operator lets scientists treat multi-domain 100 Gbps L2 paths as `kubectl`-manageable objects, exactly as they would a PersistentVolumeClaim. Any NRP user across 80+ institutions can now request a dedicated, low-jitter path to an FPGA SmartNIC or DPU, upload a P4 program, and perform real-time in-network packet processing without leaving the Kubernetes ecosystem. This lowers the barrier to advanced networking research (telemetry, congestion con-

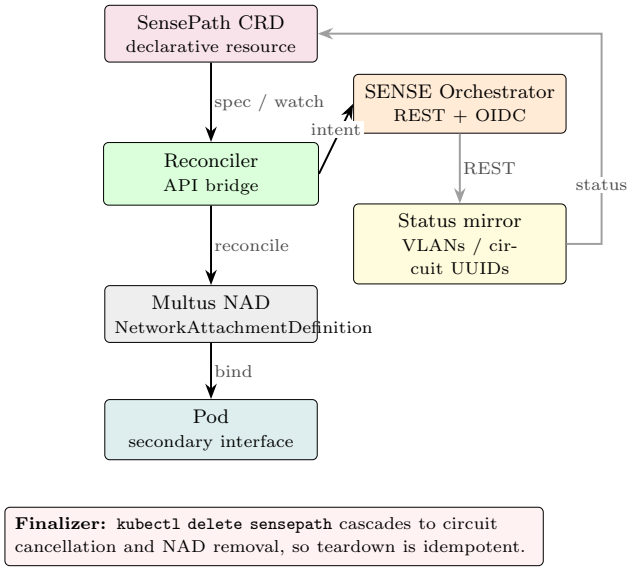


Figure 1: SENSE Operator four-component pattern (CRD, reconciler, status mirror, finalizer).

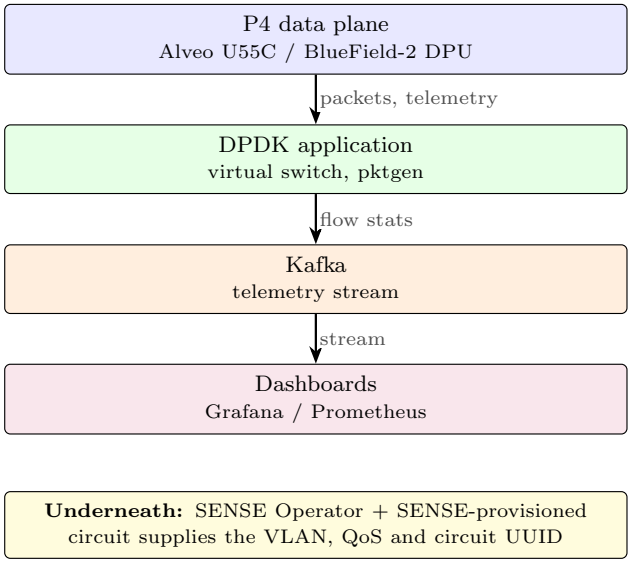


Figure 2: P4–DPDK–Kafka telemetry pipeline with SENSE-provisioned circuits.

trol, in-network machine learning, protocol offload) and makes reproducible network experiments a native part of the scientific workflow.

Beyond the demo, the four-component pattern is a reusable template: any external stateful network system (Tofino PINS, switch fabric controllers, legacy circuit services) can be bridged into Kubernetes the same way, so network-as-code becomes a design pattern rather than a one-off. Because the pattern is a stock Kubernetes operator (CRD, reconciler, status mirror, finalizer), other federations or campus teams can replicate it against their own circuit services, making network-as-code a reusable capability rather than show-floor rhetoric.

5 Interactive Experience at SC26

Live demo at the exhibit; fully replicable on NRP afterward. Attendees watch a SensePath get created, P4 firmware get deployed, and packet telemetry flow through the P4–DPDK–Kafka pipeline, all from `kubectl`.

6 Involved Parties

ESnet / LBNL: Mohammad Firas Sada (mfsada@es.net); Tom Lehman (tlehman@es.net); Xi Yang (xiyang@es.net); Justas Balcas (jbalcas@es.net); Alberto Jimenez (ajimenez@es.net); Kiran Vasu (kvasu@es.net); Chin Guok (chin@es.net); Inder Monga (inder@es.net); Dmitry Mishin (dimm@lbl.gov); **UCSD / SDSC:** John Graham (jjgraham@ucsd.edu); Frank Würthwein (fkw@ucsd.edu); Tom DeFanti (tdefanti@ucsd.edu); Tom Hutton (hutton@sdsc.edu); Mahidhar Tatineni (mahidhar@sdsc.edu); Daniel Diaz (d4diaz@ucsd.edu); Chris Hughes (crhughes@sdsc.edu); Aashay Arora (aaarora@ucsd.edu); Diego Davila (didavila@ucsd.edu); **RNP:** Marcos Felipe Schwarz (marcos.schwarz@rnp.br); **iCAIR / Northwestern:** Joe Mambretti (j-mambretti@northwestern.edu); Jim Hao Chen (jim-chen@northwestern.edu); Fei Yeh (fyeh@northwestern.edu); **Caltech:** Harvey Newman (newman@hep.caltech.edu); Preeti Bhat (preeti@caltech.edu); Andres Moya (amoya@caltech.edu); **Fermilab:** Phil Demar (demar@fnal.gov); Asif Shah (sshah@fnal.gov); Dmitry Litvintsev (litvinse@fnal.gov); **UMass:** Eduardo Bach (eduardobach@umass.edu); Rafael Coelho Lopes de Sa (rclsa@umass.edu); Jessa Westclark (jessicaw@umass.edu); **UNL:** Derek Weitzel (dweitzel@unl.edu); Huijun Zhu (huijun.zhu@unl.edu); Sam Albin (sam.albin@unl.edu); Ashton Graves (agraves10@unl.edu); **FIU:** Jeronimo Bezerra (jbezerra@fiu.edu); Julio Ibarra (julio@fiu.edu); **Internet2:** Chris Cox (ccox@internet2.edu); James Deaton (jdeaton@internet2.edu); Matt Zekauskas (matt@internet2.edu); Christian Hitchcock (chitchcock@internet2.edu); Karl Newell (knewell@internet2.edu); **Others:** Christopher Bruton (CENIC, cbruton@cenic.org); Neil McKee (inMon, neil.mckee@inmon.com); Heidi Morgan (USC ISI, hlmorgan@isi.edu); Everson Scherrer Borges (IFES, everson@ifes.edu.br); Seungmin Kim (Yonsei, ehf@yonsei.ac.kr).

References

https://github.com/esnet/esnet-smartnic-hw • https://github.com/esnet/esnet-smartnic-fw • https://github.com/esnet/open-nic-shell • https://github.com/Xilinx/open-nic-driver • https://github.com/esnet/smartnic-dpdk-docker • https://github.com/esnet/xilinx-labtools-docker • https://github.com/nrp-nautilus/esnet-smartnic • https://github.com/esnet/sense-rm • https://github.com/esnet/sense-rtmon • https://sense.es.net • https://www.gna-g.net/autogole/