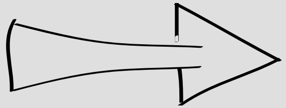
OpenNFP

P4-based VNF and Micro-VNF chaining for servers with SmartNICs

David George

david.george@netronome.com



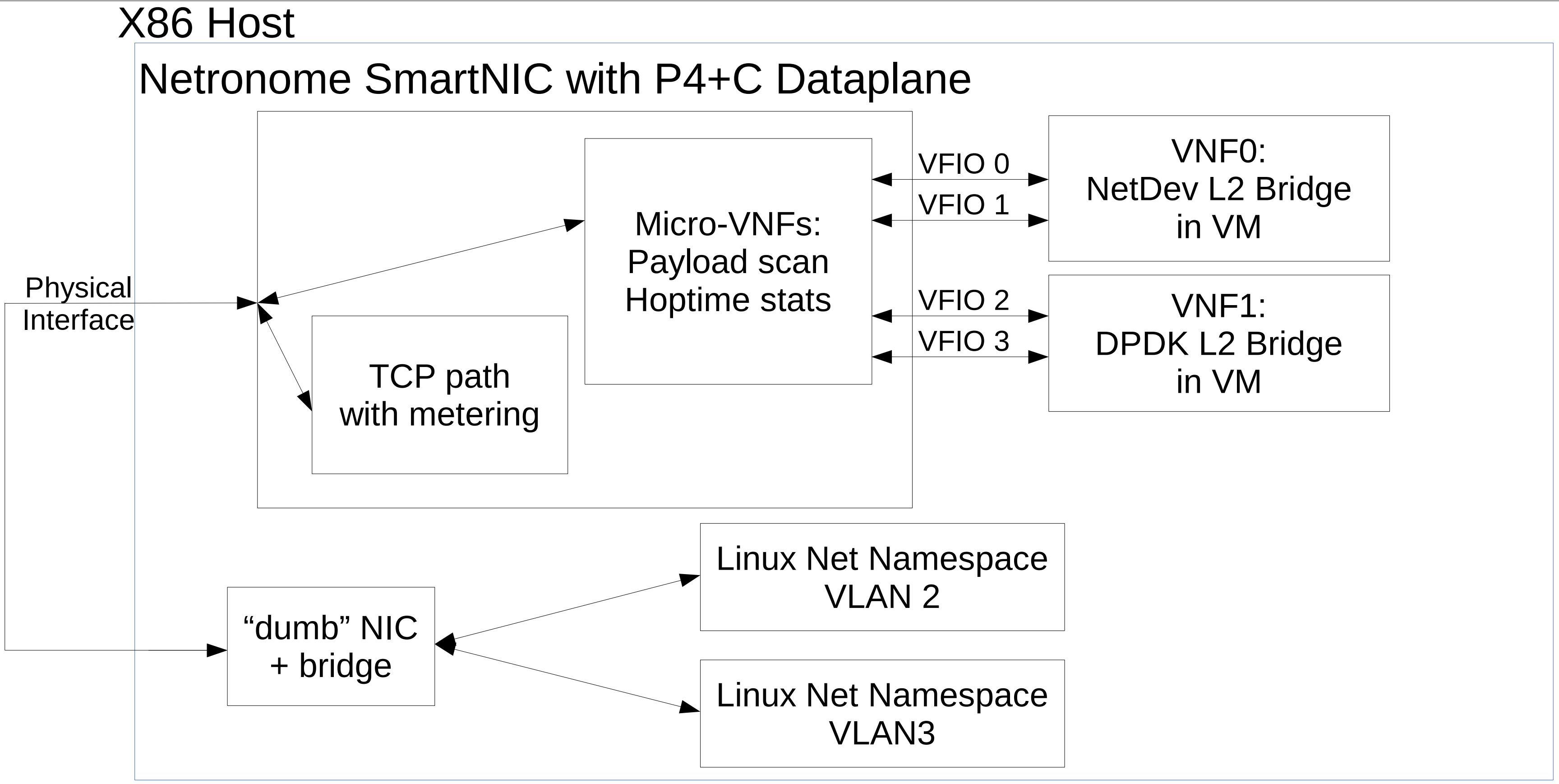
Session Agenda

- Introduction
- Design overview
- A look under the hood
  - P4, SandboxC and rules
- Run-through
  - Demo of key features
- Conclusions + questions
- (time permitting) A quick overview of SDK6 P4 runtime

Introduction

- VNF – Virtualized Network Function
  - Network function hosted independent of physical hardware
    - examples: Firewall, intrusion detection
  - Chaining a sequence of VNF
- Micro-VNF?
  - Light-weight VNF
  - Possible to integrate close to dataplane
    - e.g in kernel, on smartNIC
  - examples: telemetry, statistics
- Goal to illustrate how this can be achieved:
  - SmartNIC P4 dataplane using Netronome SDK 6
  - Micro-VNFs in sandbox C
  - Simple VNFs with libvirt + VFIO
  - Other 'neat' P4 stuff
- Why this example?
  - Illustrates how simple it can be to solve challenging problems with P4 + SmartNICs:
    - Marshalling data into VNFs
    - Gathering telemetry info + processing statistics

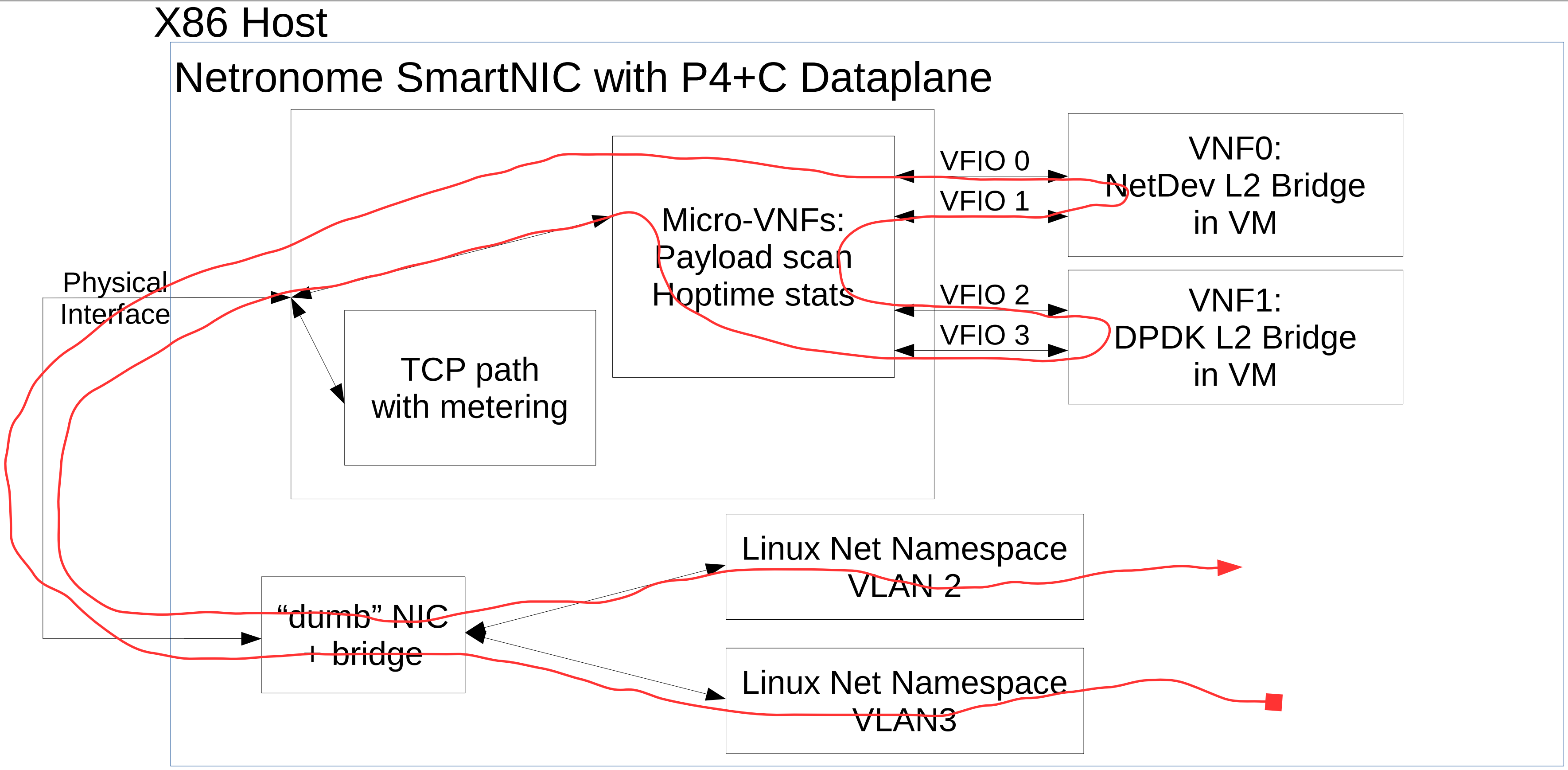
Design/Demo Overview



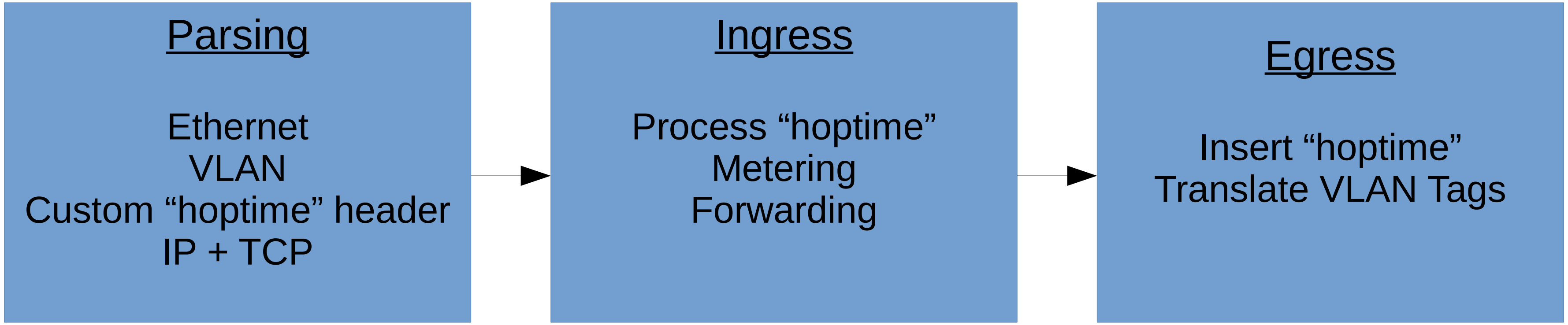
Design/Demo Overview (cont.)

- P4 Dataplane
  - Run-of-the-mill parse + match + action + forwarding
- VNFs: L2 Forwarding “cooked two ways” DPDK and NetDev
  - Simple way to illustrate use
  - BUT complicated routing:
    - Must be unique per port direction!
- MicroVNFs:
  - VF “Hoptime” statistics
    - Min, max, avg reported between VF send + receive
    - Achieved using custom tunnel
  - Simple payload scanner: search for a compile time token
- Extra points of interest
  - Meters
- Traffic generation:
  - VLAN IDs used for routing between VFs
  - Linux Network Namespaces to isolate address spaces (all on one host)

Traffic Generation



Coarse look at P4 design



A look under the hood + demo:

Conclusions

- Illustrated simple solution for VNF chaining using P4
- Showed the potential for implementing Micro-VNFs in C sandbox
  - Support for going beyond the P4 parsed headers in C
  - Measurement of VNF processing time
- Showed how flexibly P4 meters can be used

[redacted]

Github Resources

- https://github.com/open-nfpsw/p4_vnf_uvnf_demo
- Includes:
  - P4 , sandbox C & P4 configs
    - Building & loading using CLI or PS IDE
  - Host scripts

[redacted]

Questions?

SDK P4 Runtime Environment

- Typically programmer studio drives the configuration
  - Hitting “debug” programs firmware, load rules, configures meters etc.
- The RTE ships with python tools and thrift interface for interacting with P4 Runtime Environment Server via Thrift RPC
  - Allows command line driven loading configuration
  - Possible to build a stand-alone application for interacting with design
    - Example: L2 bridge receiving P4 digests and writing table updates

SDK P4 Runtime Environment (cont)

- in RTE installation you will find the following:
  - thrift/sdk6_rte.thrift
    - Thrift interface file, RPC stubs generated from this and can be used with python, C++...
  - thrift/client/RTEInterface.py
    - A python module that provides an abstraction for the Thrift interface
  - thrift/client/sdk6_rte_cli.py
    - A swiss army knife command line tool
  - thrift/client/digest_listener
    - Example tool for dumping digest events