High-Performance SFC in NFV

David Dolson (Sandvine)

IETF95 - Buenos Aires

Sandvine Experience

- Network appliances using commodity Intel processors for ~14 years
- Achieving horizontal scale with load-balancing and a form of service chaining for >10 years
- Demonstrated >1Tbps in 10 RU (rack units) of commodity hardware (a Dell blade server)

Assumptions

- Perspective of a transparent middle-box
- Intersecting multiple network links
  - Asymmetry
- Horizontal scale
- Focus on data plane

Goals

- Minimize latency
- Efficiency (Gbps/W, Gbps/RU)
- COTS hardware

How to make a fast VNF Component

Software Can Go Fast Enough

- For simple tasks, software can keep up with interface rates
  - Design threads to run independently
  - Lock application threads to physical cores
  - Connect threads to physical hardware
  - Use zero-copy packet forwarding

Independent Processes/Threads

- Utilize multi-core processors by having independent (share nothing) threads
- Slice the network data so that different threads work on independent bundles of traffic
  - Hashing end-point IP address is one way
- “Thread” may mean light-weight thread, process or VM

Socket and Core Affinity

- Give affinity of application threads to physical cores (taskset)
  - Dedicate physical core(s) to VM
  - Within VM, lock packet thread(s) to virtual core(s)
- On multi-CPU hosts, run fast-path code on CPUs with fastest access to interfaces
  - Not all CPU sockets have equivalent access to interfaces or memory. You might choose not to use some CPUs for fast path.

PF Passthrough, SR-IOV and Zero-Copy

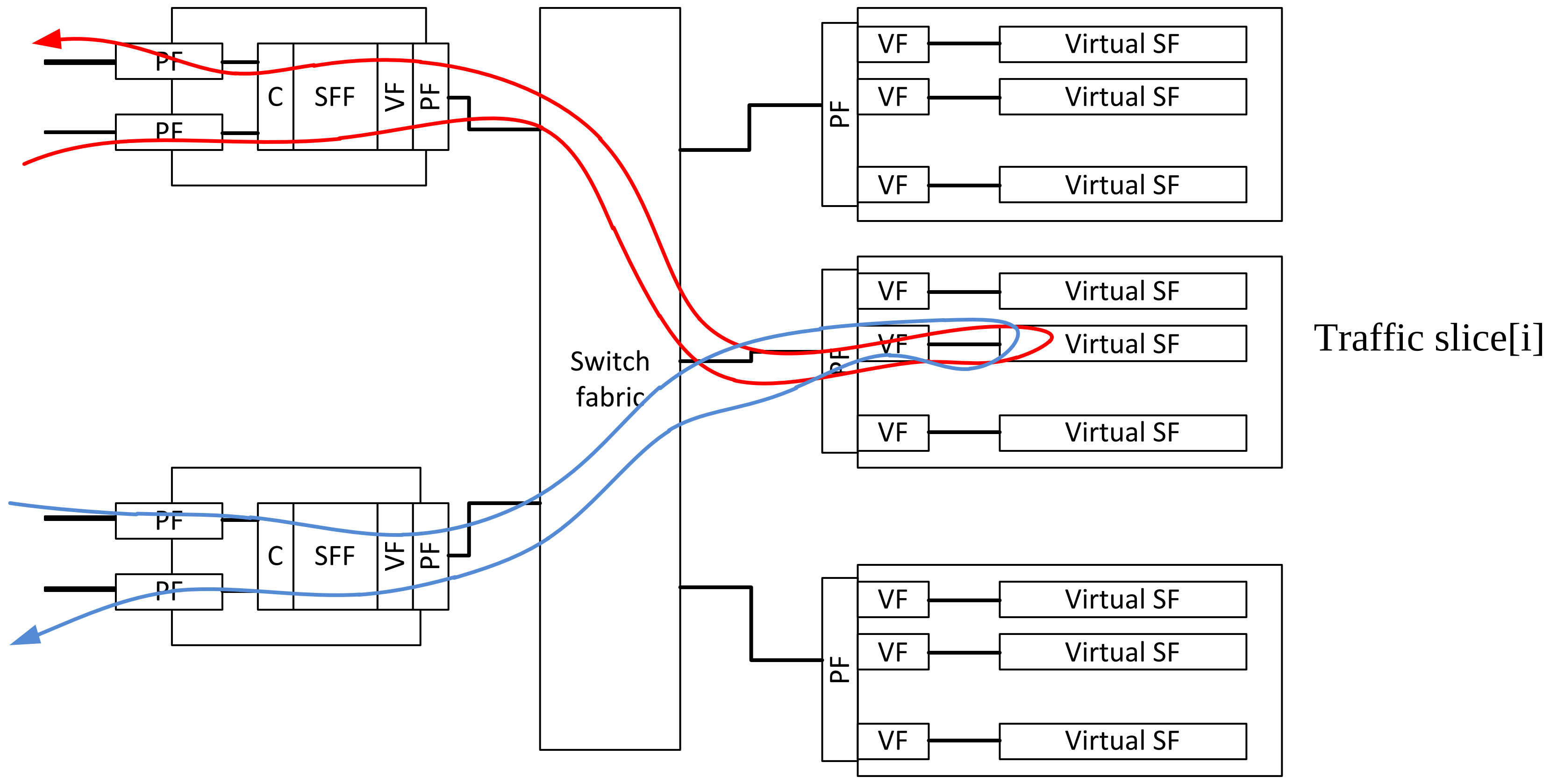
- Use PF Passthrough for promiscuous interfaces, otherwise SR-IOV for end-points
  - Avoid software touch of packets between the physical interface hardware and the application thread
- SR-IOV Allows multiple fast-path threads to share a physical interface, each at a unique MAC address
  - Create one (or more) virtual interface per fast-path thread

How to Scale

Asymmetrical Traffic

- Some (transparent) Service Functions require seeing a complete picture of a traffic “flow”
  - E.g., both directions of a TCP flow
  - E.g., all of an internet subscriber’s traffic
- But flows typically use multiple links, especially for up vs. down traffic

Consistent Multi-link Slicing



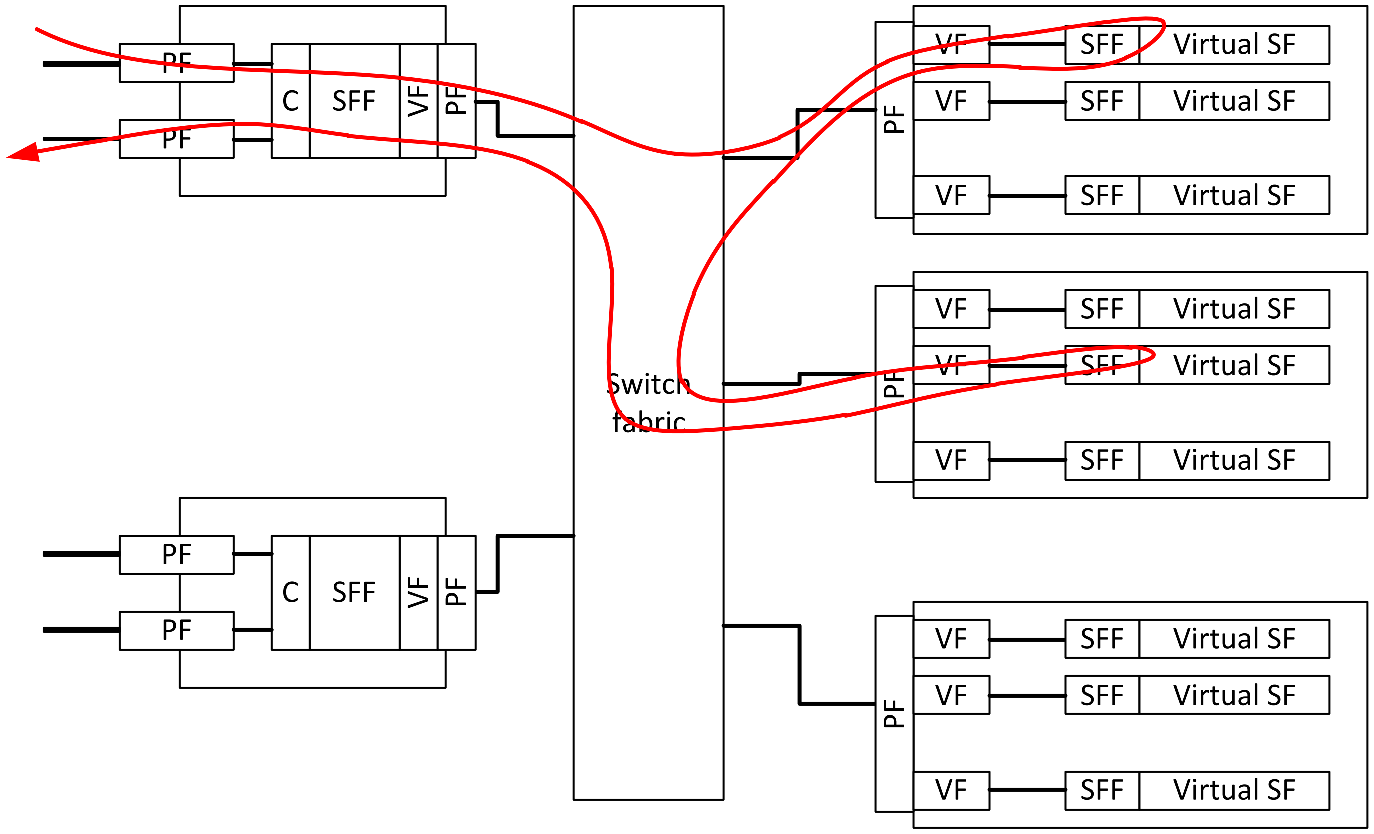
East-West Bottleneck

- As packets visit multiple virtual machines, interface bandwidth is consumed
- Additional overhead due to encapsulation
- So a two-touch solution may be half the performance (or twice the gear) of a one-touch solution
  - (Based on network bottleneck vs. CPU bottleneck)

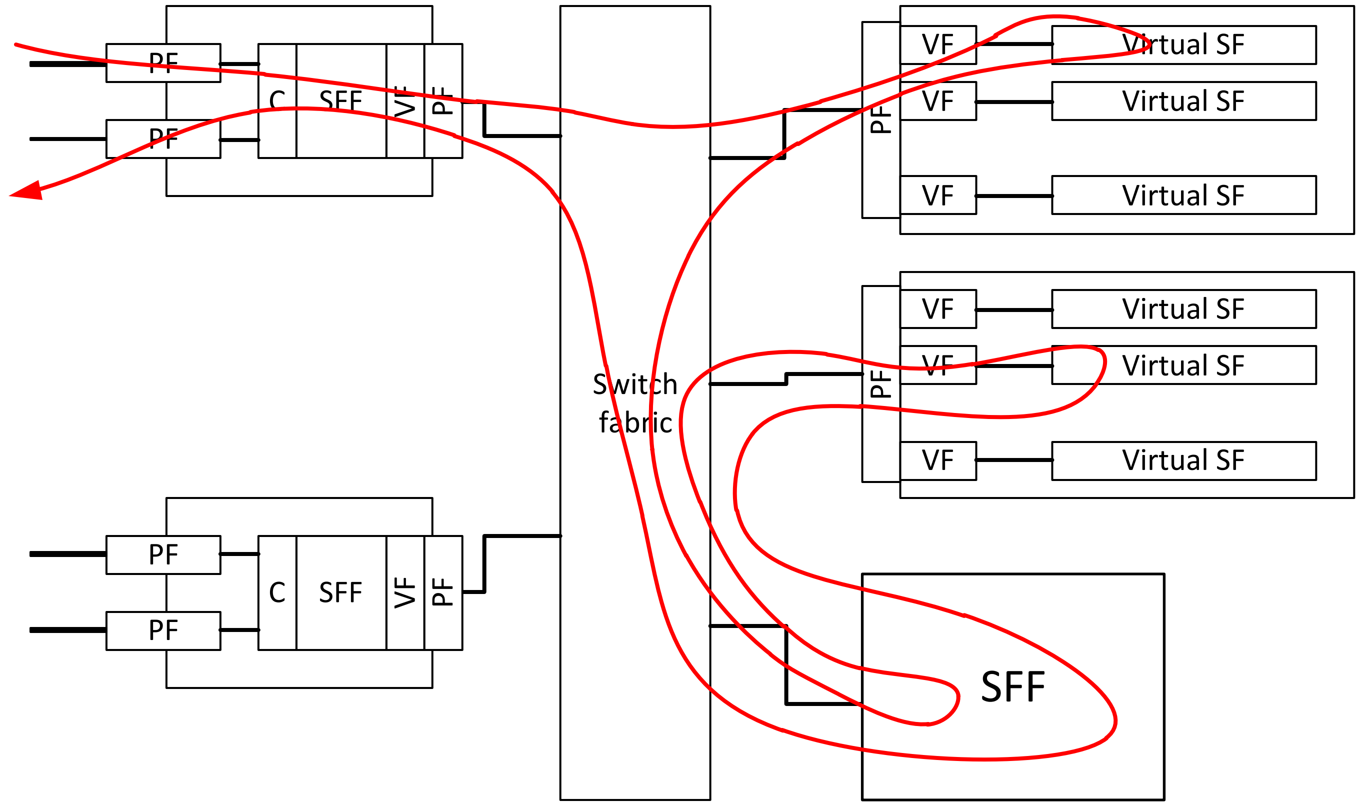
Putting SFF tables in the SF thread

- In the data plane, SFF is a simple software operation to compute the next hop by table lookup.
- If SFF is a separate process,
  - Dedicate a core per 10GE interface
  - Adds latency
  - Consumes east-west budget

SFF Co-located with SF



External SFF – Bandwidth bottleneck!



Minimize Encapsulation Overhead

- Encapsulation inflates packet sizes, reducing the productive bandwidth of switch fabric and NICs
- So we would choose MAC/NSH encapsulation when devices are on the same layer-2 segment, and use an IP encapsulation otherwise
- We would choose NSH MD type-2 when metadata is not being used

Thoughts on Performance Qualification

- Q: “How many VMs do I need?”
- A: “It depends!”

- Need to understand too much:
  - VM software design
  - Host technology (NUMA architecture)
  - NIC offload technology
  - Assignment of threads to cores
  - Path of each packet in the use-cases

Summary

- Zero-copy packet code
- PCI Pass-through and SR-IOV
- Slice your network traffic
- Minimize East-West traffic and touches per packet
- SFF within each SF thread
- Minimize packet overhead

References

- Sandvine blog post on >1Tbps in a blade server: http://www.internetphenomena.com/2015/10/breaking-the-virtual-terabit-barrier/
- Some of this is discussed in draft-dolson-sfc-nfv-patterns