SHARKFEST '12

Wireshark Developer and User Conference

Microsoft's Demon

Datacenter Scale Distributed Ethernet Monitoring Appliance

Rich Groves
Principal Architect
Microsoft GNS

Bill Benetti
Senior Service Engineer
Microsoft MSIT

Before We Begin

- We are Network Engineers.
- This isn’t a Microsoft product.
- We are here to share methods and knowledge.
- Hopefully we can all foster evolution in the industry.

Microsoft is a great place to work!

- We need experts like you.
- We have larger than life problems to solve.
- Networking is important and well funded.
- Washington is beautiful.

The Microsoft Demon Technical Team

- Rich Groves
- Bill Benetti
- Dylan Greene
- Justin Scott
- Ken Hollis
- Tanya Ollick
- Eric Chou

About Rich Groves

- Microsoft's Global Network Services

NOUS – Network of Unusual Scale

- Microsoft IT

EOUS – Enterprise of Unusual Scale

- Time Warner Cable

- Endace

Made cards, systems, software for "Snifferguys"

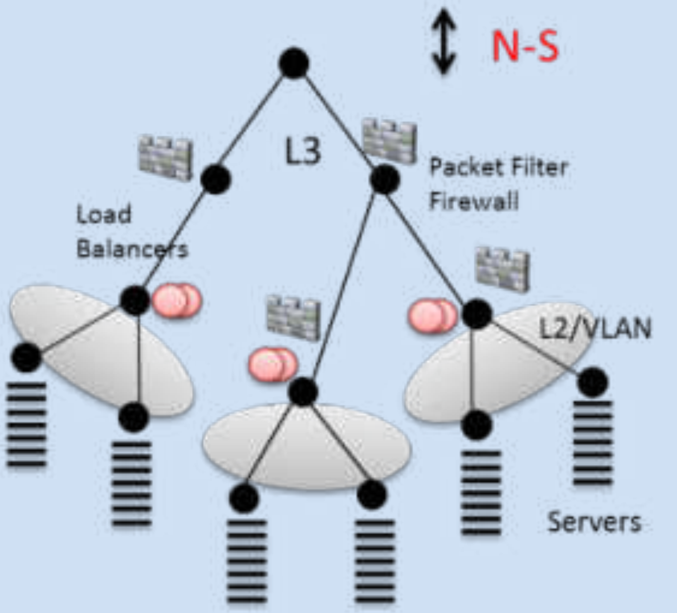
- AOL

"Snifferguy"

- MCI

Artist's Approximation

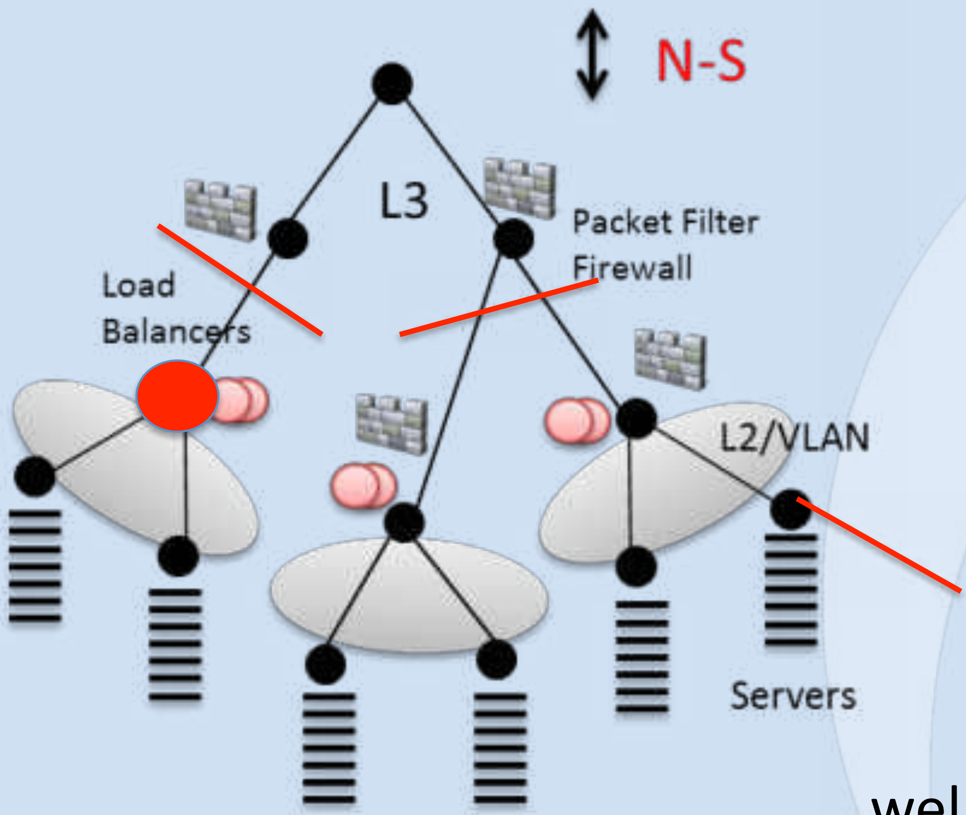
The Traditional Network



Hierarchical Tree Structure – Optimized for N-S traffic

- hierarchical tree optimized for north/ south traffic
- firewalls, load balancers, and WAN optimizers
- not much cross datacenter traffic
- lots of traffic localized in the top of rack

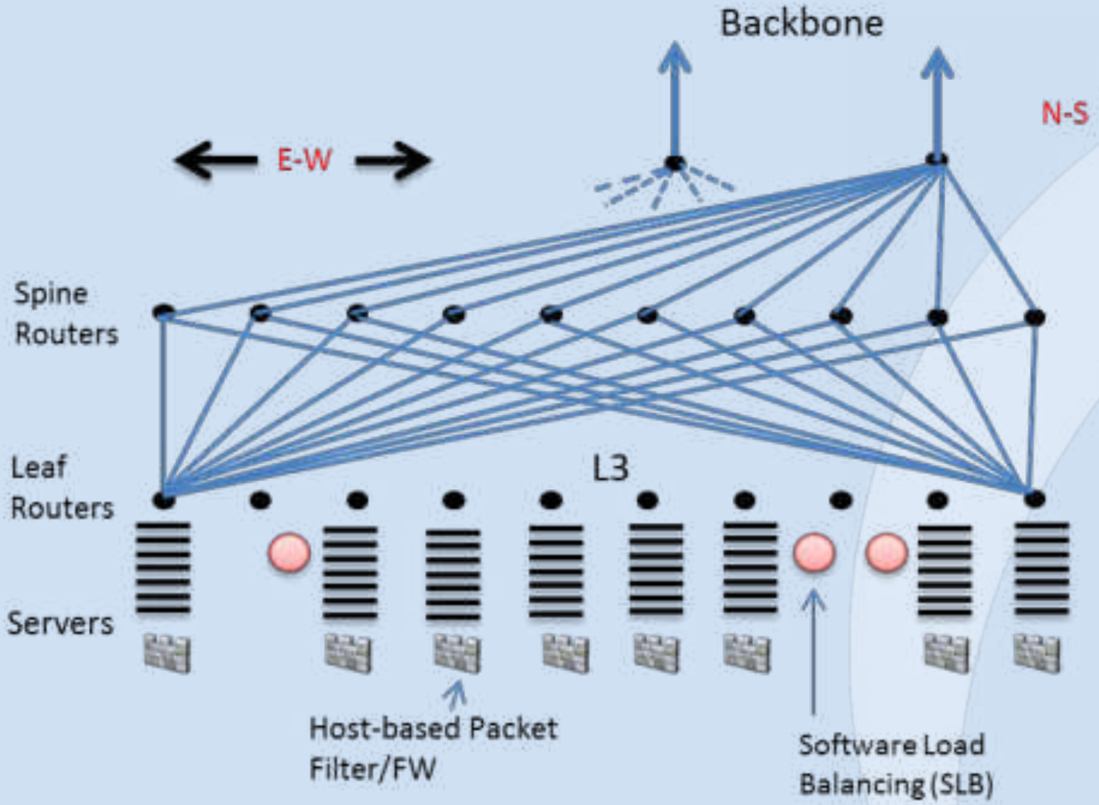
Analyzing the Traditional Network



- insert taps within the aggregation
- port mirror at the top of rack
- capture packets at the load balancer

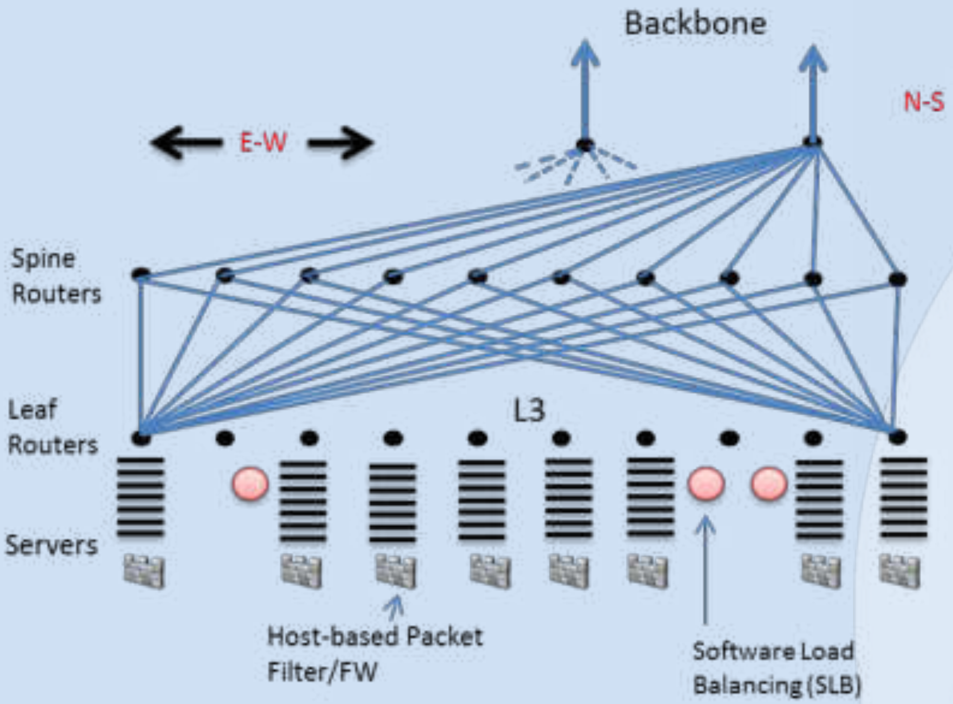
well understood but costly at scale

The Cloud Datacenter



- tuned for massive cross data center traffic
- appliances removed for software equivalents

Can you tap this cost effectively?



- 8,16,and 32x10g uplinks

- Tapping 32x10g ports requires 64 ports to aggregate.

(Who can afford buying current systems for that?)

- ERSpan could be used, but it impacts production traffic.

- Even port mirrors are a difficult task at this scale.

Many attempts at making this work

- Capturenet
  - complex to manage
  - purpose built aggregation devices were far too expensive at scale
  - resulted in lots of gear gathering dust

- PMA - “Passive Measurement Architecture”
  - failed due to boring name
  - rebranded as PUMA by outside marketing consultant (Rich’s eldest daughter)

- PUMA
  - lower cost than Capturenet
  - extremely feature rich
  - too costly at scale

- Pretty Pink PUMA
  - attempt at rebranding by Rich’s youngest daughter
  - rejected by the team

Solution 1: Off the Shelf

- used 100% purpose built aggregation gear
- supported many higher end features (timestamping,slicing,etc)
- price per port is far too high
- not dense enough (doesn't even terminate one tap strip)
- high cost made tool purchases impossible
- no point without tools

Solution 2: Cascading Port Mirrors

How

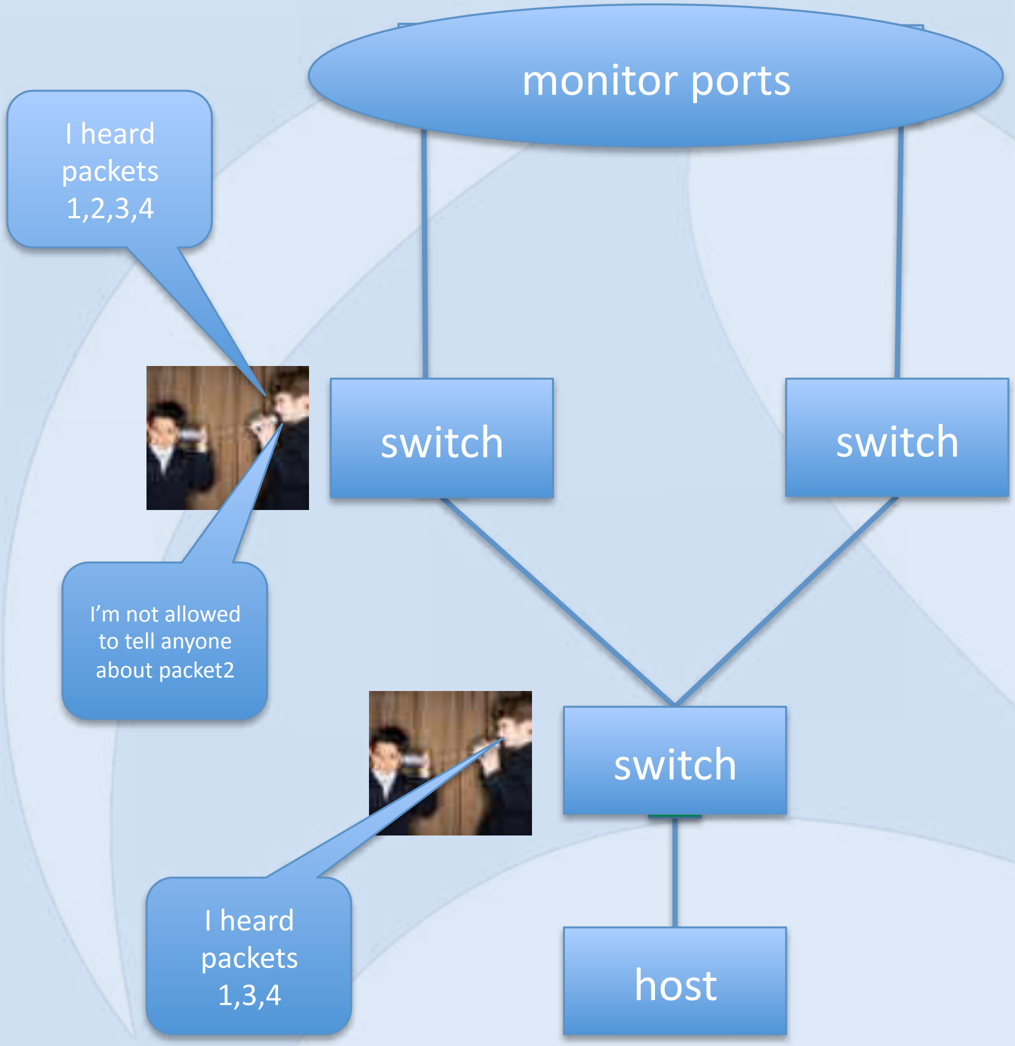
- mirror all attached monitor ports to next layer
- pre-filter by only mirroring interfaces you wish to see

The Upside

- cost effective
- uses familiar equipment
- can be done using standard CLI commands in a config

The Downside

- control traffic removed by some switches
- assumes you know where to find the data
- lack of granular control
- uses different pathways in the switch
- quantity of port mirror targets is limited



Solution 3: Making a Big Hub

How

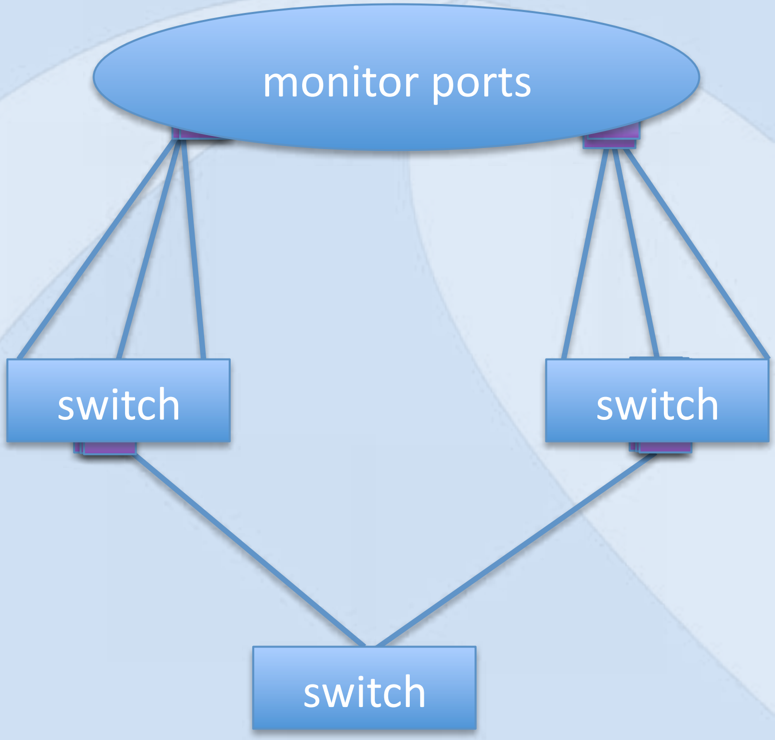
- turn off learning
- flood on all ports
- unique outer VLAN tag per port using QinQ
- pre-filter based on ingress port through VLAN pruning

Upside

- cost effective

Downside

- Control traffic is still intercepted by the switch.
- Performance is non-deterministic.
- Some switches need SDK scripts to make this work.
- Data quality suffers.



The End

- Well not really, but it felt like it.

Core Aggregator Functions

- terminates links
- 5-tuple pre-filters
- duplication
- forwarding without modification
- low latency
- zero loss
- time stamps
- frame slicing

Let’s solve 80 percent of the problem:

- terminates links
- 5-tuple pre-filters
- duplication
- forwarding without modification
- low latency
- zero loss

do-able in merchant silicon switch chips

costly due to lack of demand outside of the aggregator space

Reversing the Aggregator

The Basic Logical Components

- terminate links of all types and a lot of them
- low latency and lossless
- N:1, 1:N duplication
- some level of filtering
- control plane for driving the device

What do these platforms have in common?

Can you spot the commercial aggregator ?

[redacted]

[redacted]

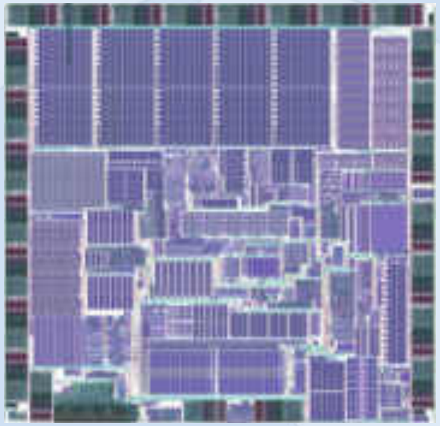
Introducing Merchant Silicon Switches

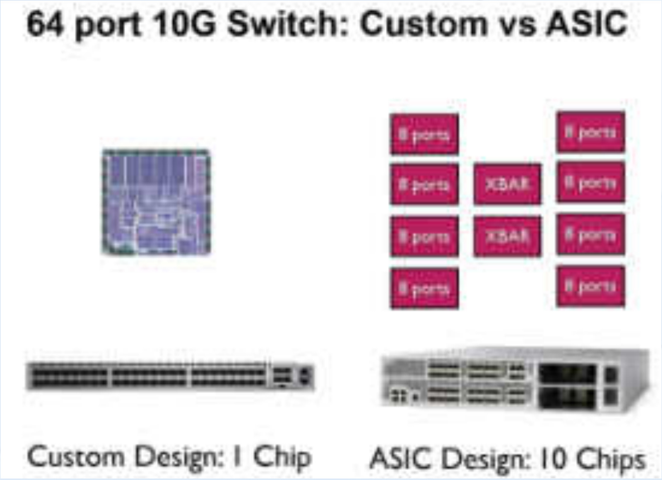
Advantages of merchant silicon chips:

- more ports per chip (64x10g currently)
- much lower latency

(due to fewer chip crossings)

- consume less power
- more reliable than traditional ASIC based multi-chip designs





Merchant Silicon Evolution

Year	2007	2011	2013	2015
10G on single chip	**24**	**64**	**128**	**256**
Silicon Technology	130nm	65nm	40nm	28nm

Interface speed evolution: 40G, 100G, 400G(?), 1Tbps

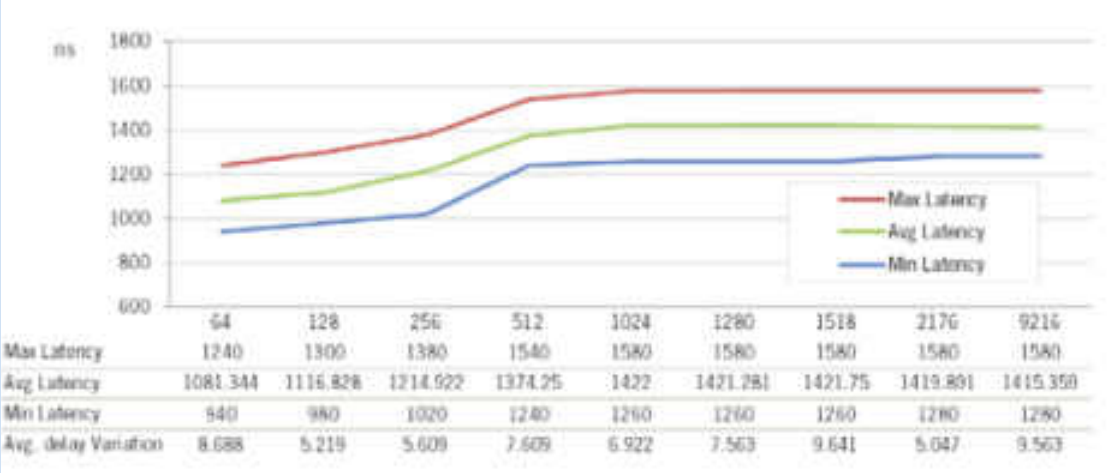
This is a single chip. Amazingly dense switches are created using multiple chips.

Reversing the Aggregator

The Basic Logical Components

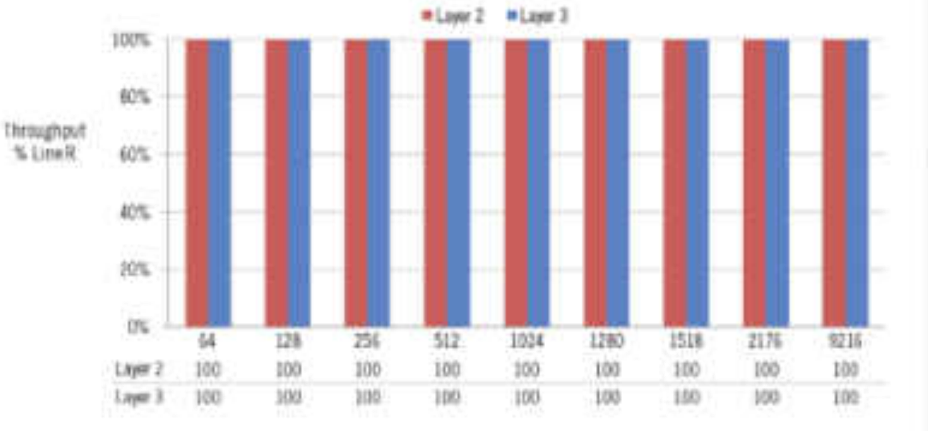
- terminate links of all types
- low latency and lossless
- N:1, 1:N duplication
- some level of filtering
- control plane for driving the device

Port to Port Characteristics of Merchant Silicon



	64	128	256	512	1024	1280	1518	2176	9216
Max Latency	1240	1300	1380	1540	1580	1580	1580	1580	1580
Avg Latency	1081.344	1116.828	1214.922	1374.25	1422	1421.281	1421.75	1419.891	1415.359
Min Latency	940	980	1020	1240	1260	1260	1260	1280	1280
Avg. delay Variation	8.688	5.219	5.609	7.609	6.922	7.563	9.641	5.047	9.563

Latency port to port (within the chip)



	64	128	256	512	1024	1280	1518	2176	9216
Layer 2	100	100	100	100	100	100	100	100	100
Layer 3	100	100	100	100	100	100	100	100	100

Loss within the aggregator isn’t acceptable.

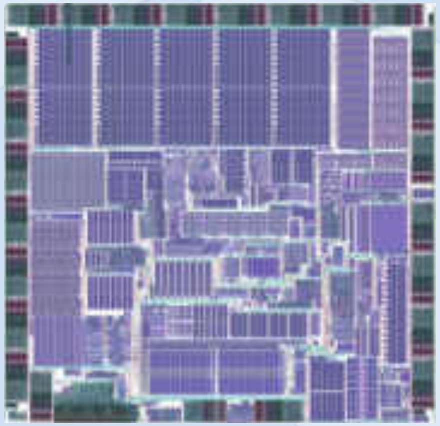
Such deterministic behavior makes a single chip system ideal as an aggregator.

Reversing the Aggregator

The Basic Logical Components

- terminate links of all types
- low latency and lossless
- N:1, 1:N duplication
- some level of filtering
- control plane for driving the device

Duplication and Filtering



Duplication

- line rate duplication in hardware to all ports
- facilitates 1:N, N:1, N:N duplication and aggregation

Filtering

- line rate L2/L3/L4 filtering on all ports
- thousands of filters depending on the chip type

Reversing the Aggregator

The Basic Logical Components

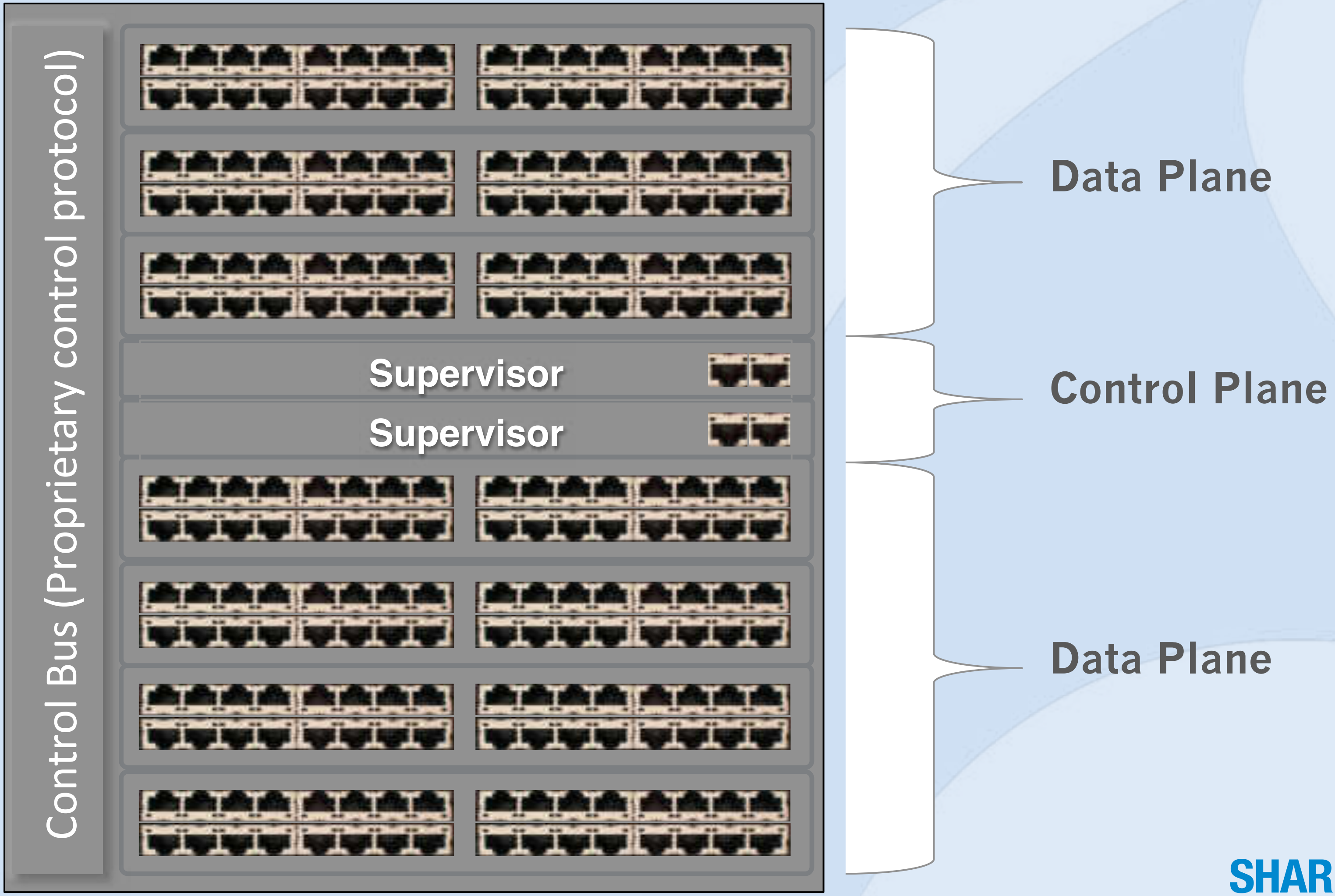
- terminate links of all types
- low latency and lossless
- N:1, 1:N duplication
- some level of filtering
- control plane for driving the device

Openflow as a Control Plane

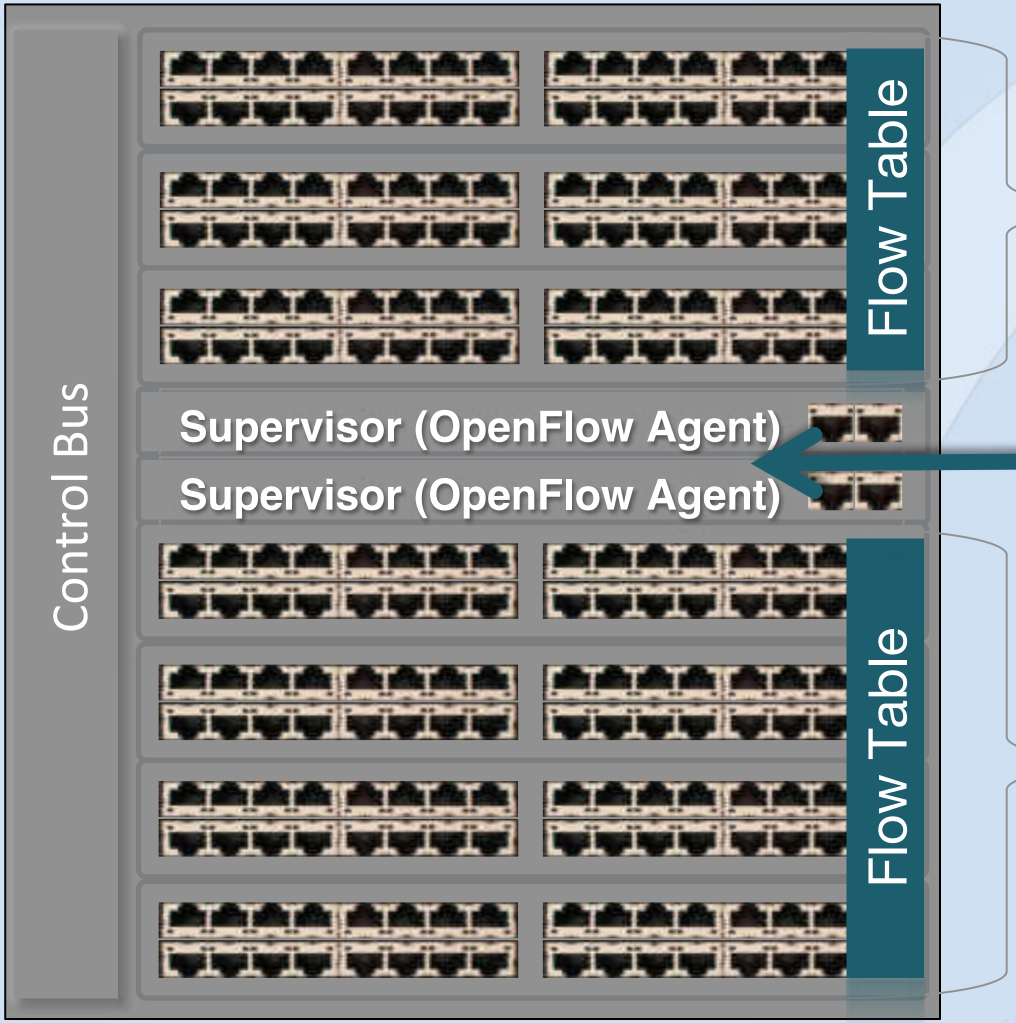
What is Openflow?

- remote API for control
- allows an external controller to manage L2/L3 forwarding and some header manipulation
- runs as an agent on the switch
- developed at Stanford 2007-2010
- now managed by the Open Networking Foundation

Common Network Device



Controller Programs Switch's "Flow Tables"

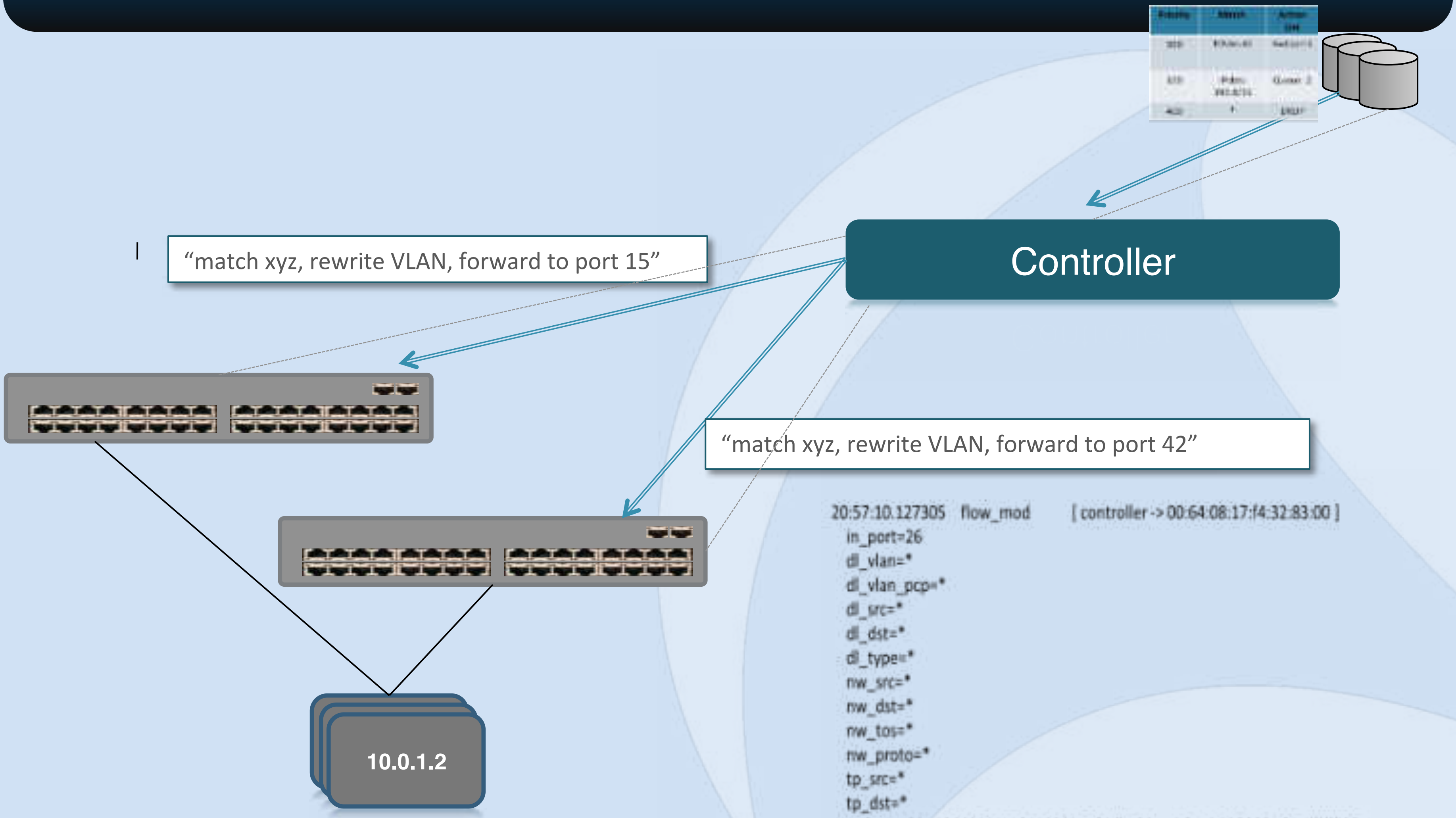


Priority	Match	Action List
300	TCP.dst=80	Fwd:port 5
100	IP.dst= 192.8/16	Queue: 2
400	*	DROP

OpenFlow Controller

Priority	Match	Action List
500	TCP.dst=22	TTL--, Fwd:port 3
200	IP.dst= 128.8/16	Queue: 4
100	*	DROP

Proactive Flow Entry Creation



“match xyz, rewrite VLAN, forward to port 15”

“match xyz, rewrite VLAN, forward to port 42”

```
20:57:10.127305   flow_mod     [ controller -> 00:64:08:17:f4:32:83:00 ]
  in_port=26
  dl_vlan=*
  dl_vlan_pcp=*
  dl_src=*
  dl_dst=*
  dl_type=*
  nw_src=*
  nw_dst=*
  nw_tos=*
  nw_proto=*
  tp_src=*
  tp_dst=*
  ADD: cookie:18446744072143941255 idle:0 hard:0 pri:32768 buf:0xffffffff flg:
```

Openflow 1.0 Match Primitives (Demon Related)

Match Types

- ingress port
- src/dst MAC
- src/dst IP
- ethertype
- protocol
- src/dst port
- TOS
- VLAN ID
- VLAN Priority

Action Types

- mod VLAN ID
- drop
- output
- controller

Flow Table Entries == “if,then,else”

if “ingress port=24 and ethertype=2048(IP) and dest IP=10.1.1.1”
then “dest mac=00:11:22:33:44:55 and output=port1”

if “ethertype=2054(ARP) and src IP=10.1.1.1”
then “output=port2,port3,port4,port5,port6,port7,port8,port9,port10”

if “ethertype=2048(IP) and protocol=1(ICMP)”
    then “controller”

Openflow 1.0 Limitations

- lack of QinQ support
- lack of basic IPv6 support
  - no deep IPv6 match support
  - can redirect based on protocol number (ether-type)
- no layer 4 support beyond port number
  - cannot match on TCP flags or payloads

Multi-Tenant Distributed Ethernet Monitoring Appliance

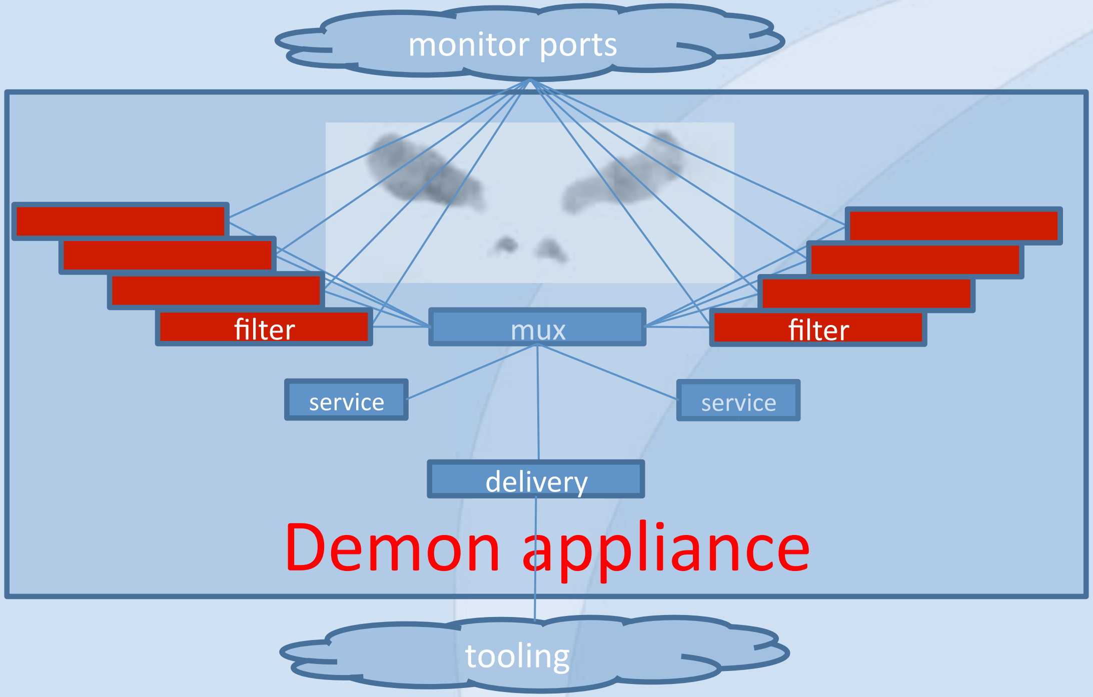
Enabling Packet Capture and Analysis at Datacenter Scale

4.8 Tbps of filtering capacity
find the needle in the haystack

Industry Standard CLI

sFlow

save valuable router resources
using the
Demon packet sampling offload



more than 20X cheaper
than “off the shelf” solutions

self serve
using a RESTful API

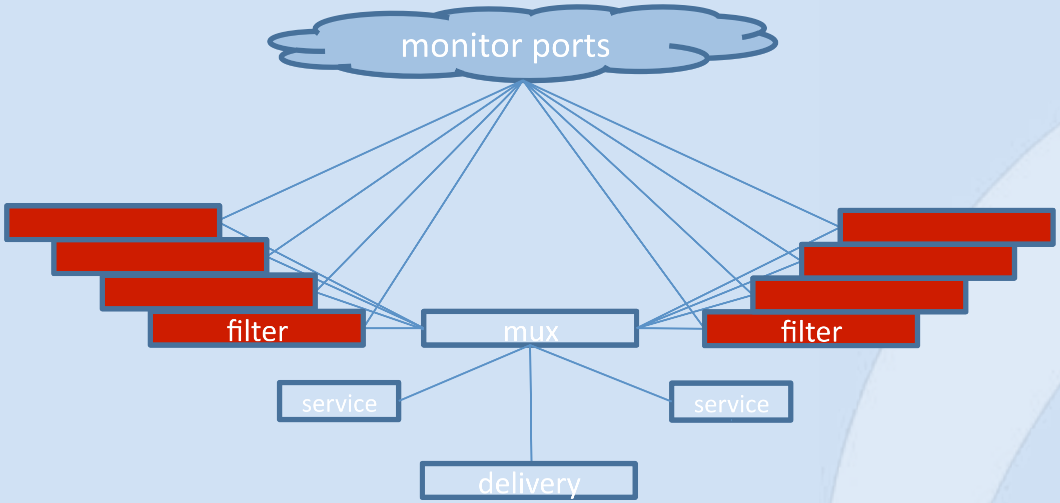
OpenFlow

leveraging Openflow
for
modular scale and granular control

filter and deliver to any
“Demonized” datacenter even
to hopboxes and Azure

based on low-cost
merchant silicon

Filter Layer



terminates inputs from 1,10,40g ports

initially drops all traffic inbound

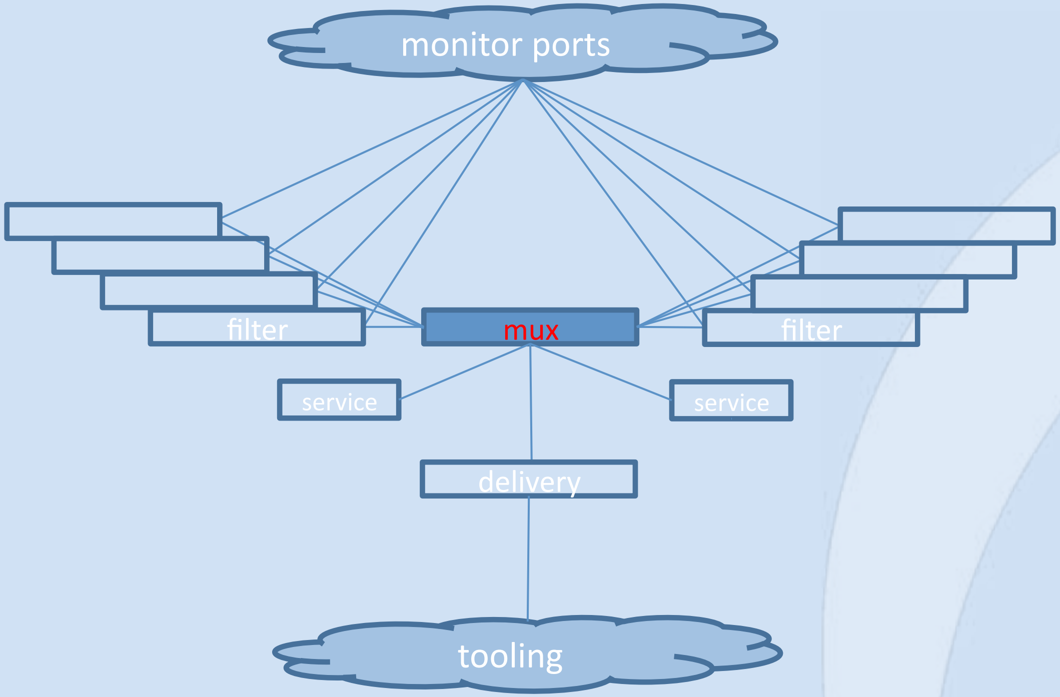
approximately 1000 L3/L4 Flows per switch

performs longest match filters

high rate sFlow sampling with no ”production impact”

- filter switches have 60 filter interfaces facing monitor ports
- filter interfaces allow only inbound traffic through the use of high priority flow entries
- 4x10g infrastructure interfaces are used as egress toward the mux

Mux Layer



terminates 4x10g infrastructure ports from each filter switch

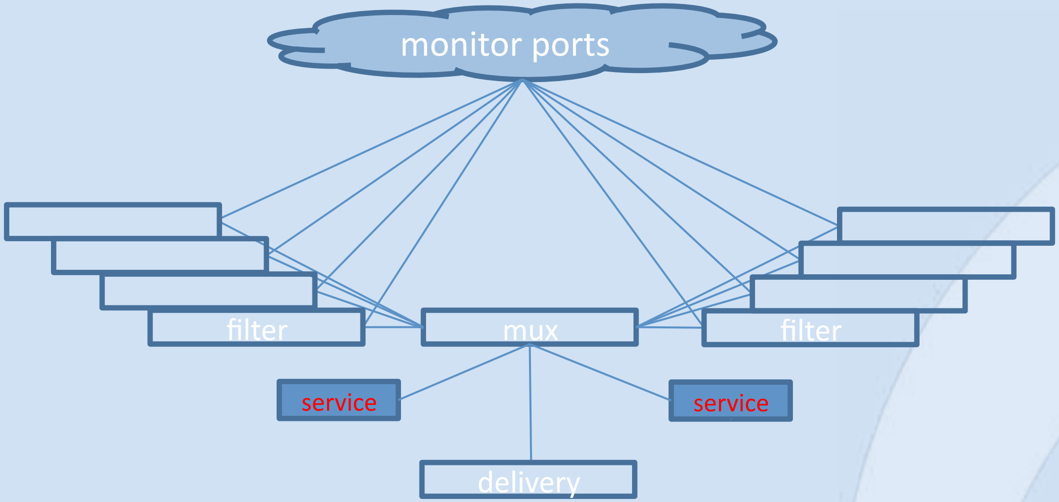
performs shortest match filters

provides both service node and delivery connectivity

duplicates flows downstream if needed

- introduces pre-service and post-service ports
- used to aggregate all filter switches
- directs traffic to either service node or delivery interfaces

Services Nodes



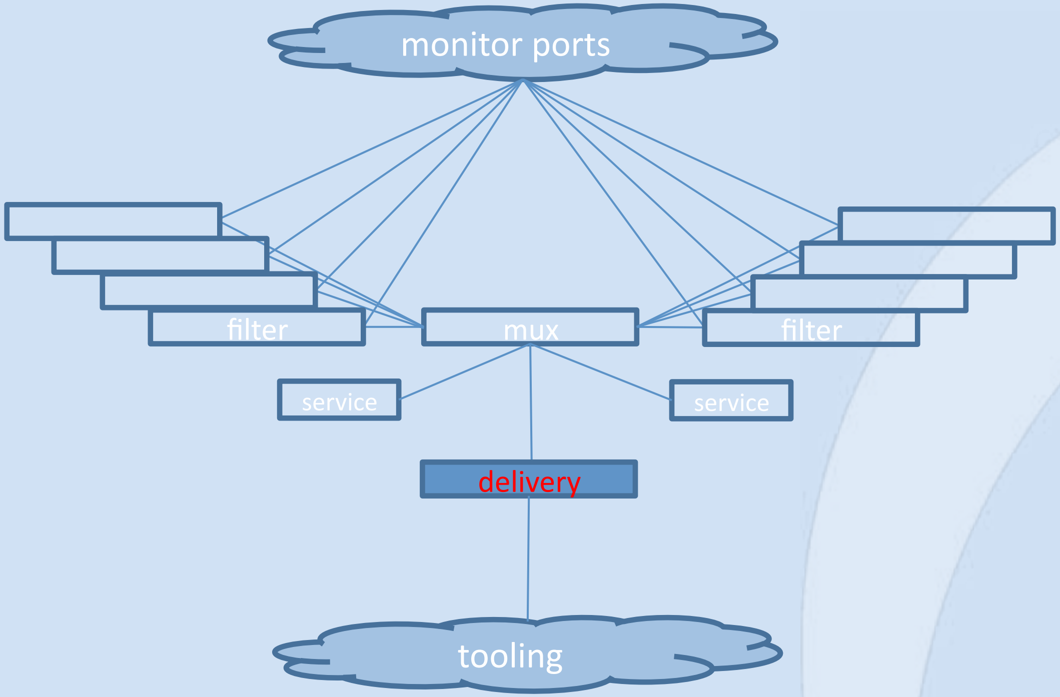
leverage higher end features on a smaller set of ports

possible uses:

- deeper filtering
- time stamping
- frame slicing
- encapsulation removal for tunnel inspection
- configurable logging
- higher resolution sampling
- encryption removal
- payload removal for compliance
- encapsulation of output for location independence

- connected to mux switch through pre-service and post-service ports
- performs optional functions that Openflow and merchant silicon cannot currently provide

Delivery Layer



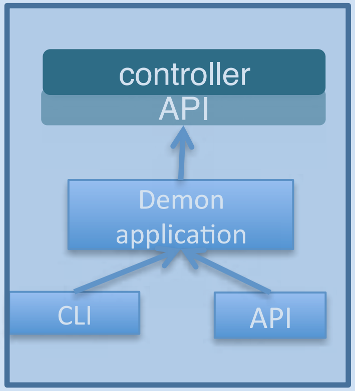
1:N and N:1 duplication

data delivery to tools

further filtering if needed

- introduces delivery interfaces which connect tools to Demon
- can optionally fold into mux switch depending on tool quantity and location

Advanced Controller Actions



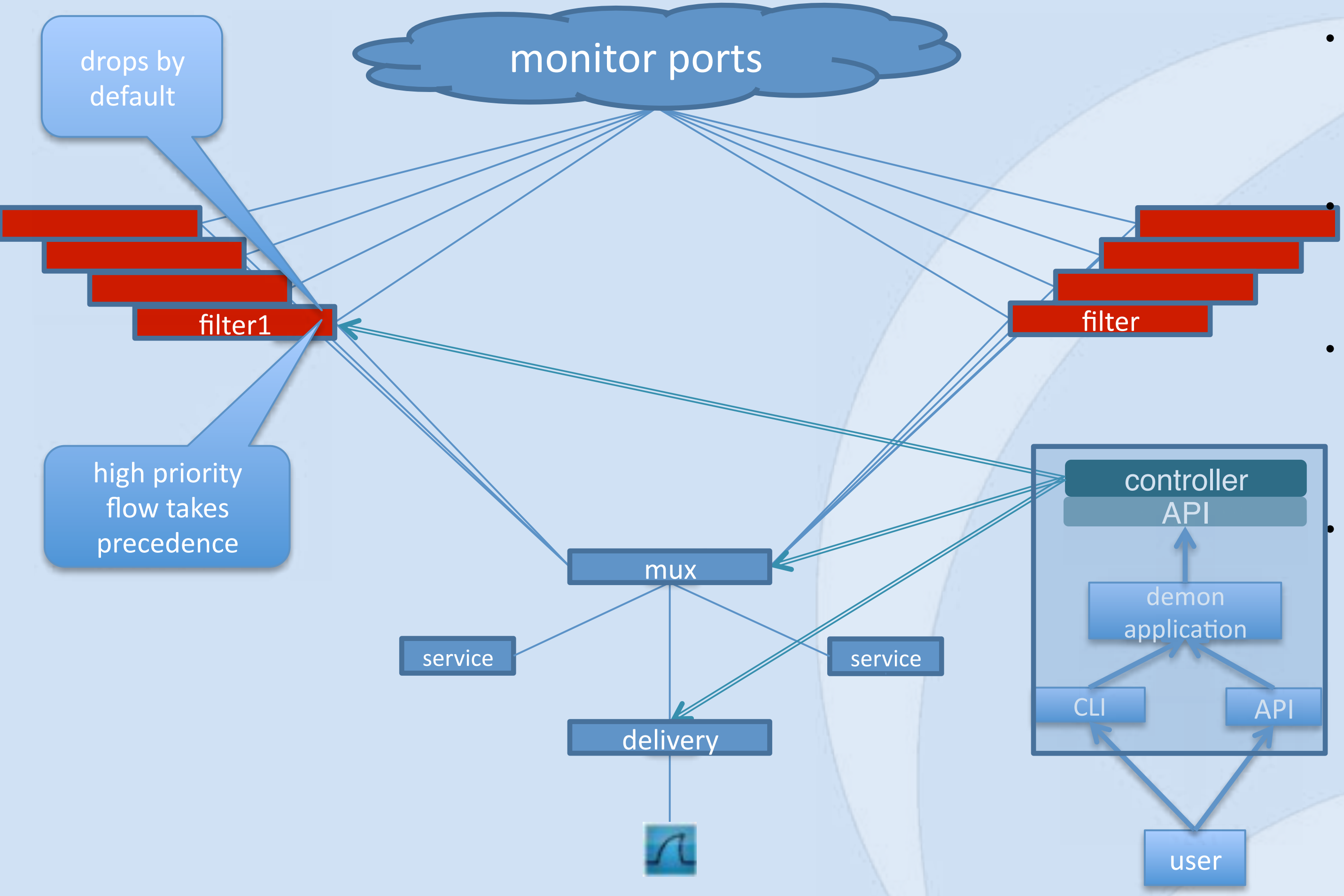
receives packets and octets of all flows created

above used as rough trigger for automated packet captures

duplicate LLDP, CDP, and ARP traffic to the controller at low priority to collect topology information

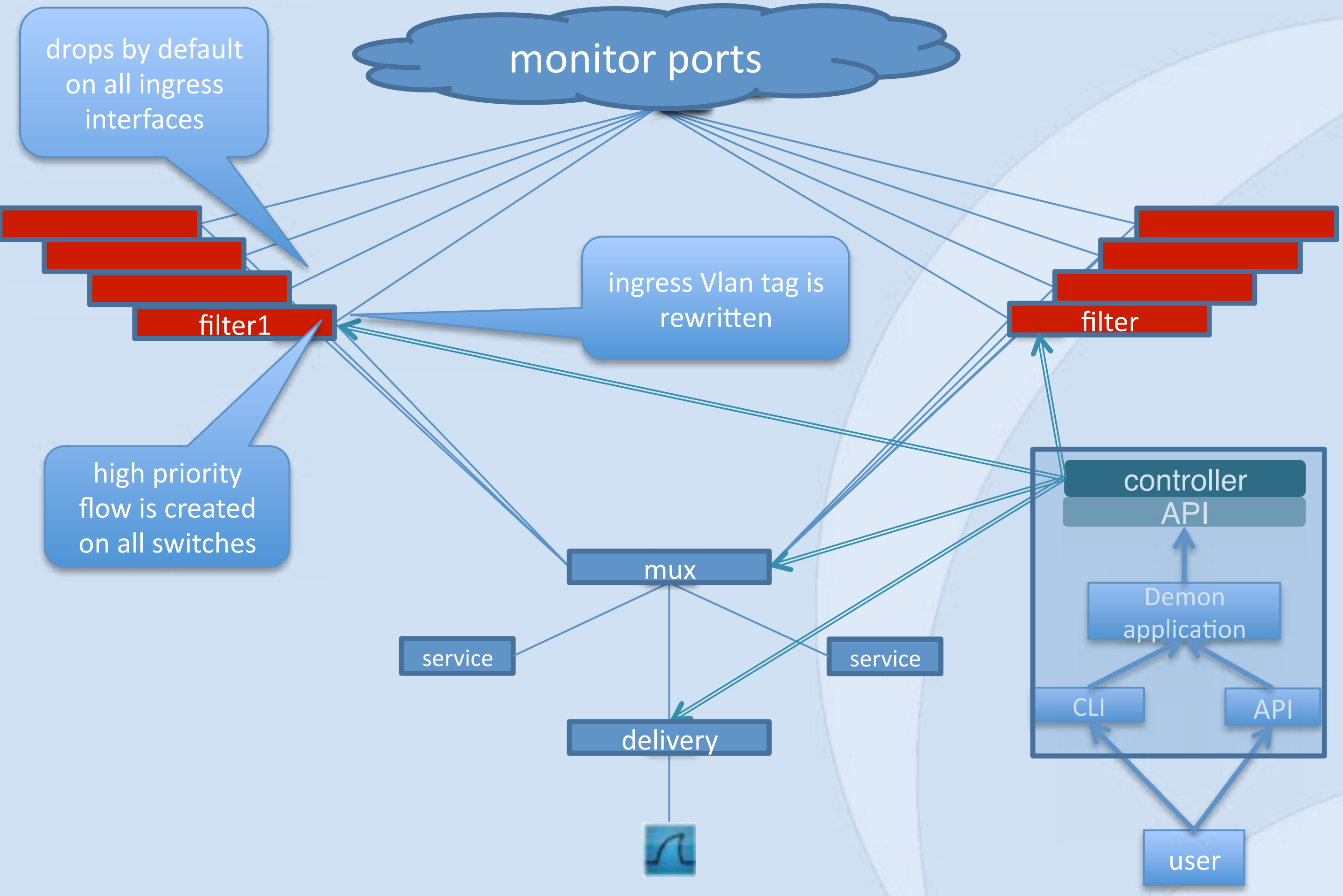
source “Tracer” documentation packets to describe the trace

Location Aware Demon Policy



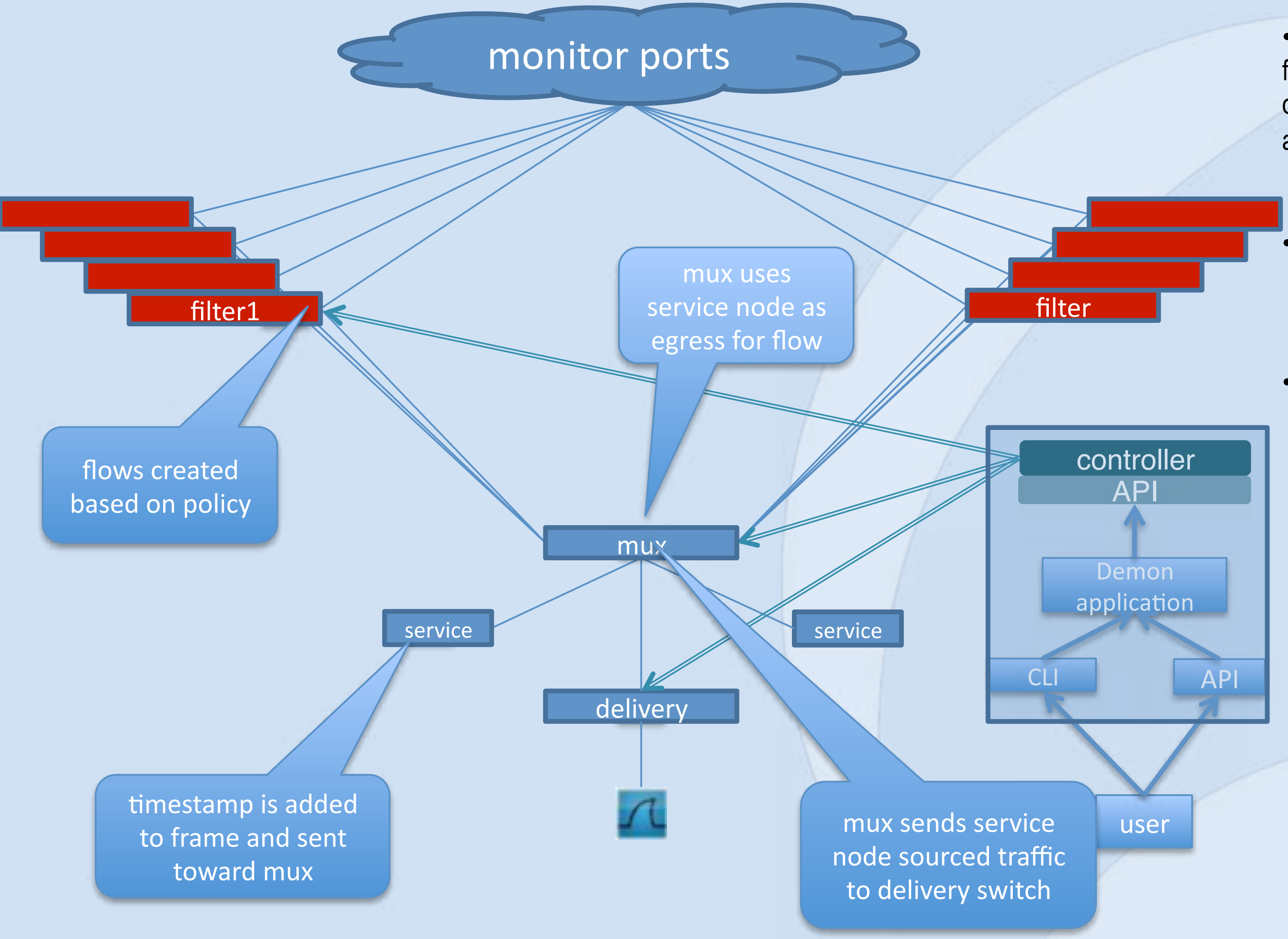
- policy created using CLI or API
  “forward all traffic matching tcp dest 80 on port1 of filter1 to port 1 of delivery1”
- Demon app creates flows though controller API
- controller pushes a flow entry to filter1,mux,and delivery to output using available downstream links
- traffic gets to the wireshark system

Location Independent Demon Policy


drops by default on all ingress interfaces
monitor ports
filter1
ingress Vlan tag is rewritten
filter
high priority flow is created on all switches
controller
API
Demon application
CLI
API
mux
service
service
delivery
user

- policy created using CLI or API
- if TCP dst port 80 on any ingress port on any filter switch then add location meta-data and deliver to delivery1
- Ingress VLAN tag is rewritten to add substrate locale info and uniqueness to duplicate packets.
- Traffic gets to Wireshark.

Inserting a Service Node

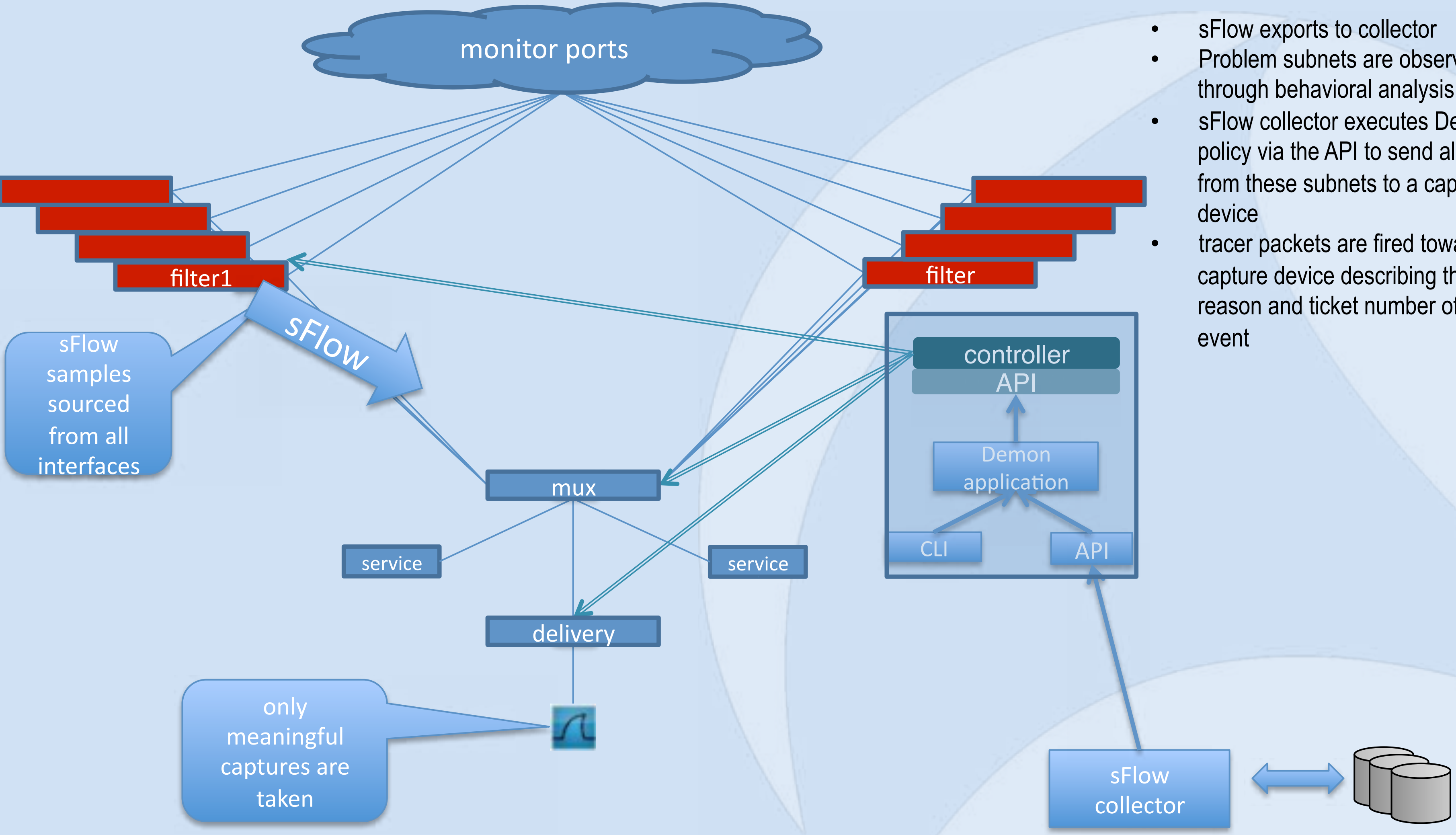

monitor ports
filter1
filter
mux uses service node as egress for flow
flows created based on policy
mux
service
service
delivery
controller
API
Demon application
CLI
API
user
timestamp is added to frame and sent toward mux
mux sends service node sourced traffic to delivery switch

- policy created using CLI or API forward all traffic matching tcp dest 80 on port1 of filter1 to port 1 of delivery1 and use service node "timestamping""

- flows created per policy on the filter and mux to use the service node as egress

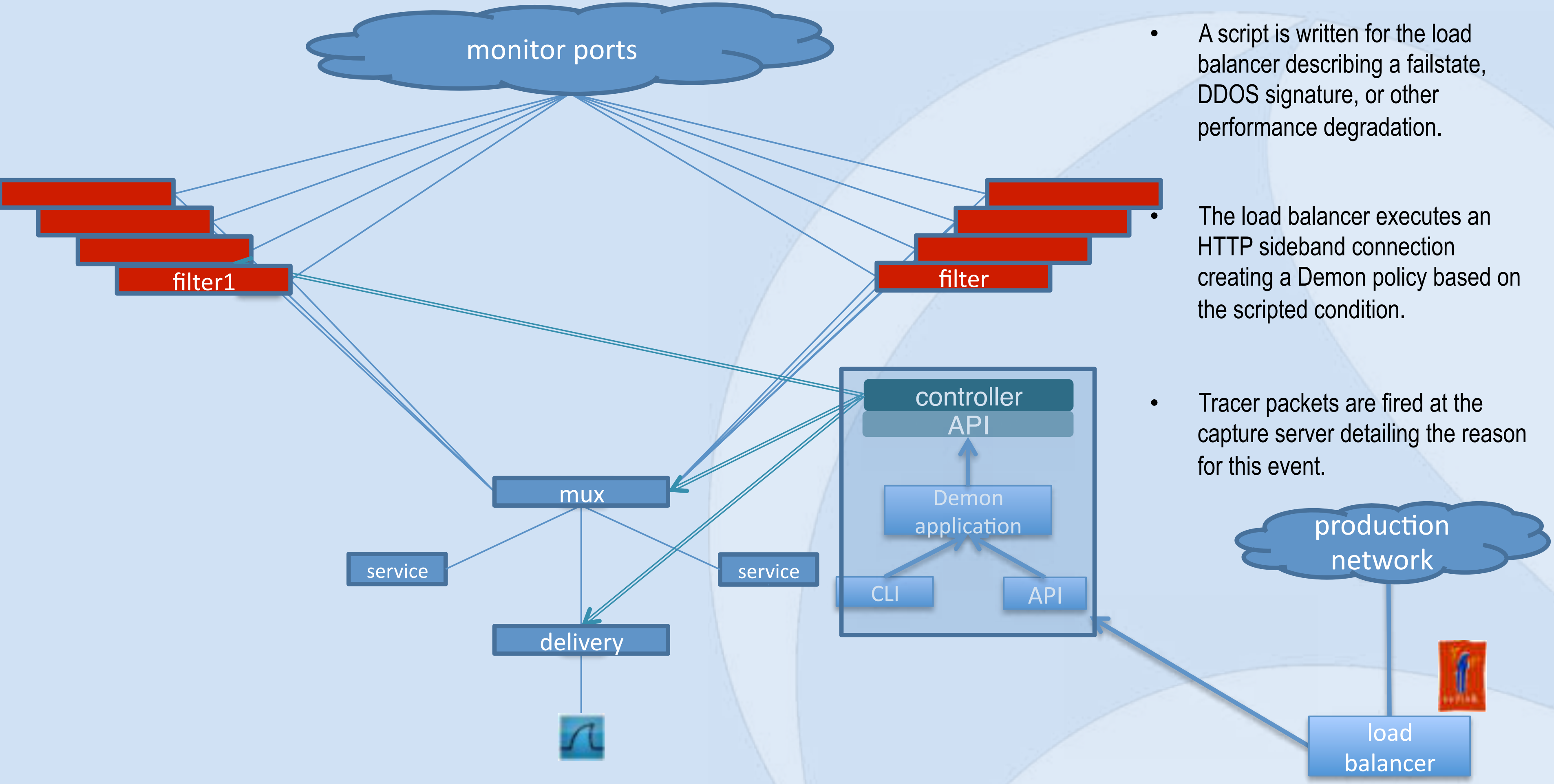
- traffic gets to Wireshark

Advanced Use Case 1: Closed Loop Data Collection



- sFlow exports to collector
- Problem subnets are observed through behavioral analysis.
- sFlow collector executes Demon policy via the API to send all traffic from these subnets to a capture device
- tracer packets are fired toward the capture device describing the reason and ticket number of the event

Advanced Use Case 2: Infrastructure Cries for Help



- A script is written for the load balancer describing a failstate, DDOS signature, or other performance degradation.
- The load balancer executes an HTTP sideband connection creating a Demon policy based on the scripted condition.
- Tracer packets are fired at the capture server detailing the reason for this event.

Summary

- The use of single chip merchant silicon switches and Openflow can be an adequate replacement for basic tap/mirror aggregation at a fraction of the cost.
- An open API allows for the use of different tools for different tasks.
- Use of an Openflow controller enables new functionality that the industry has never had in a commercial solution.

Thanks

- Q&A
- Thanks for attending!