Multi-Gigabit Intrusion Detection with OpenFlow and Commodity Clusters

INDIANA UNIVERSITY

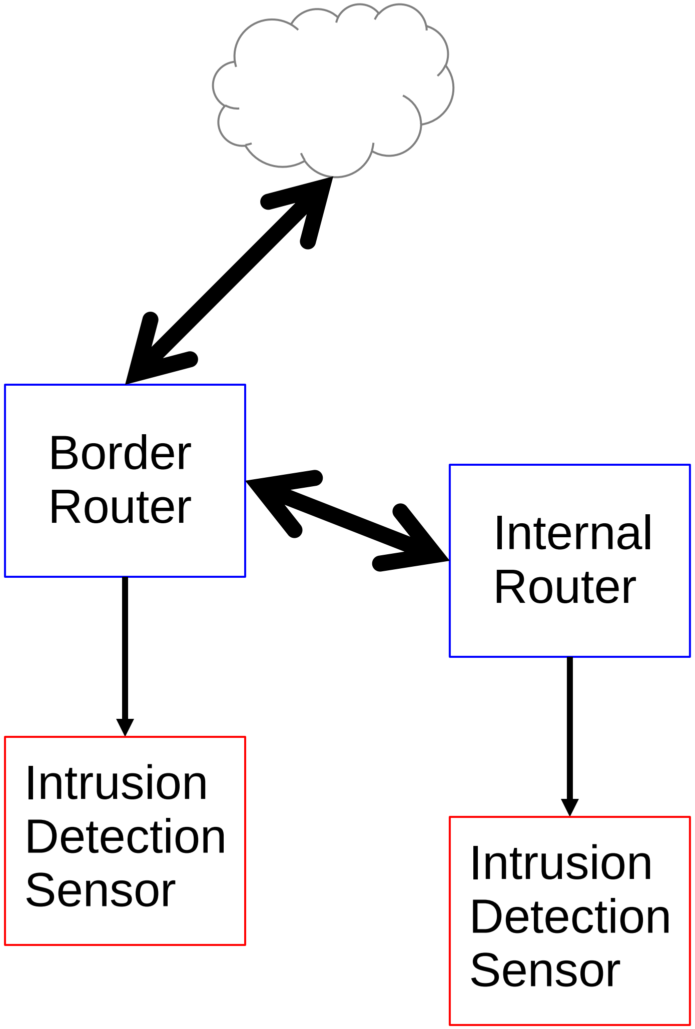
Multi-Gigabit Intrusion Detection with OpenFlow and Commodity Clusters

Keith Lehigh
University Information Security Office
Indiana University

Ali Khalfan
InCNTRE
Indiana University

May 16, 2012

INDIANA UNIVERSITY



How is Intrusion Detection done today?

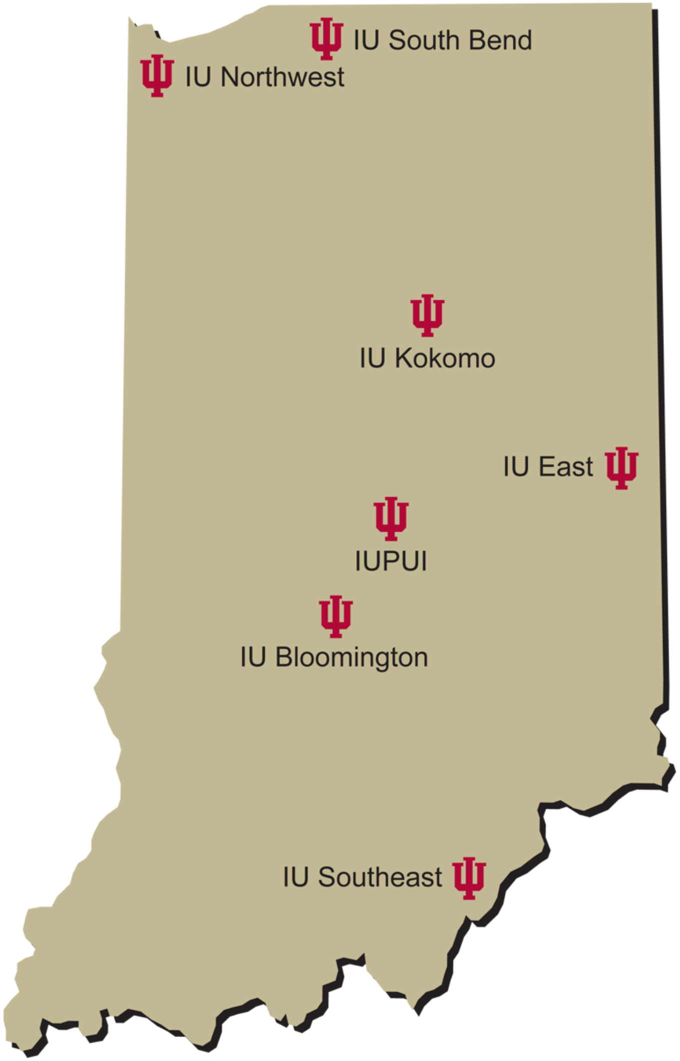
- At least a border mirror
- Mirror feed may be oversubscribed
- Often one box per router

Old IDS @ IU

- Started out as a surplus Dell desktop with 10Mb/s border feed

- Datacenter feeds / some core routers
- Prone to packet loss
  - 10Gb/s mirrors to 1Gb/s fiber
  - Media converter to 1Gb/s copper

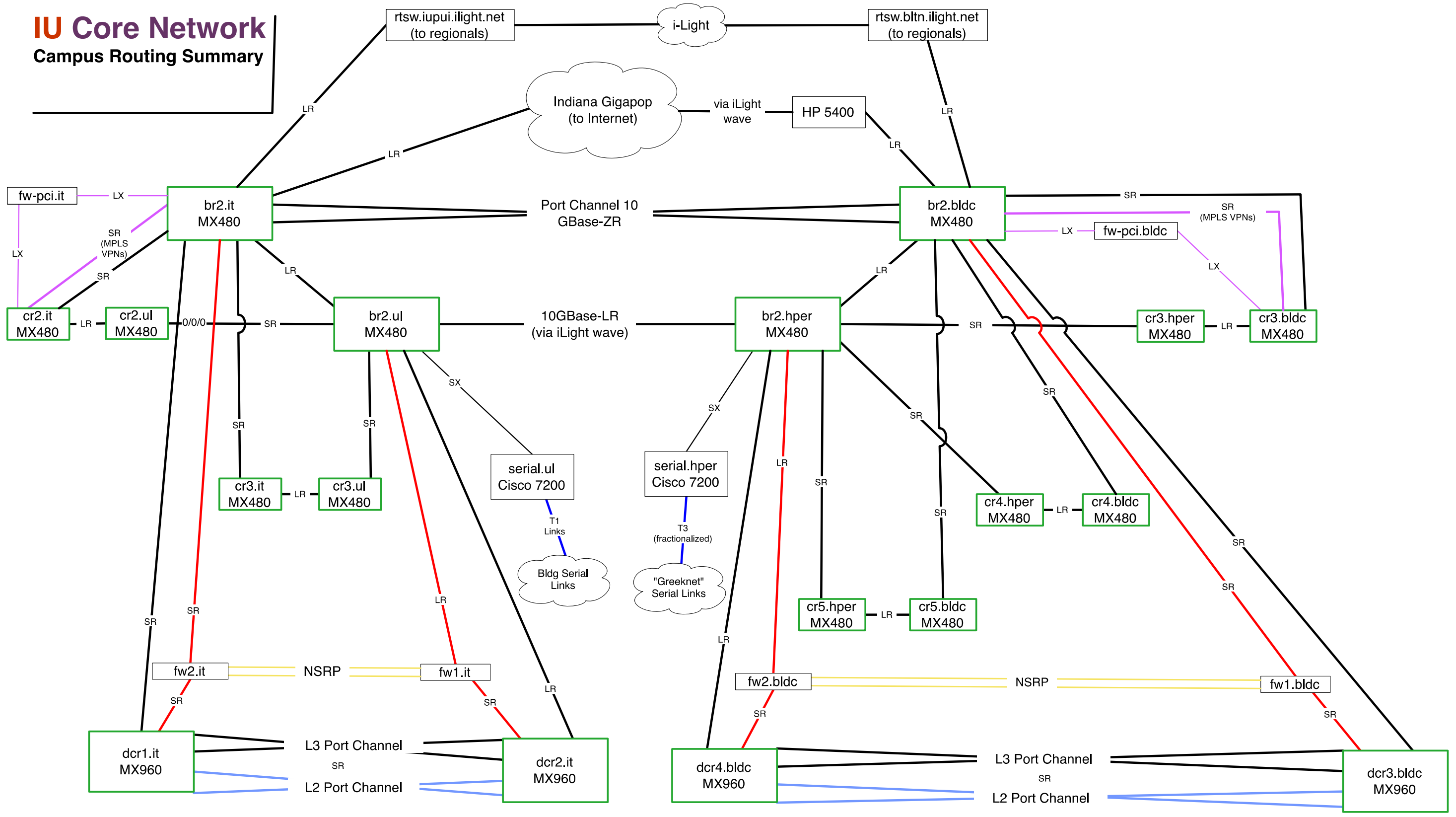
- 1:1 feeds to sensors
- Multi-core with multiple snort instances
  - BPF “load balancing”

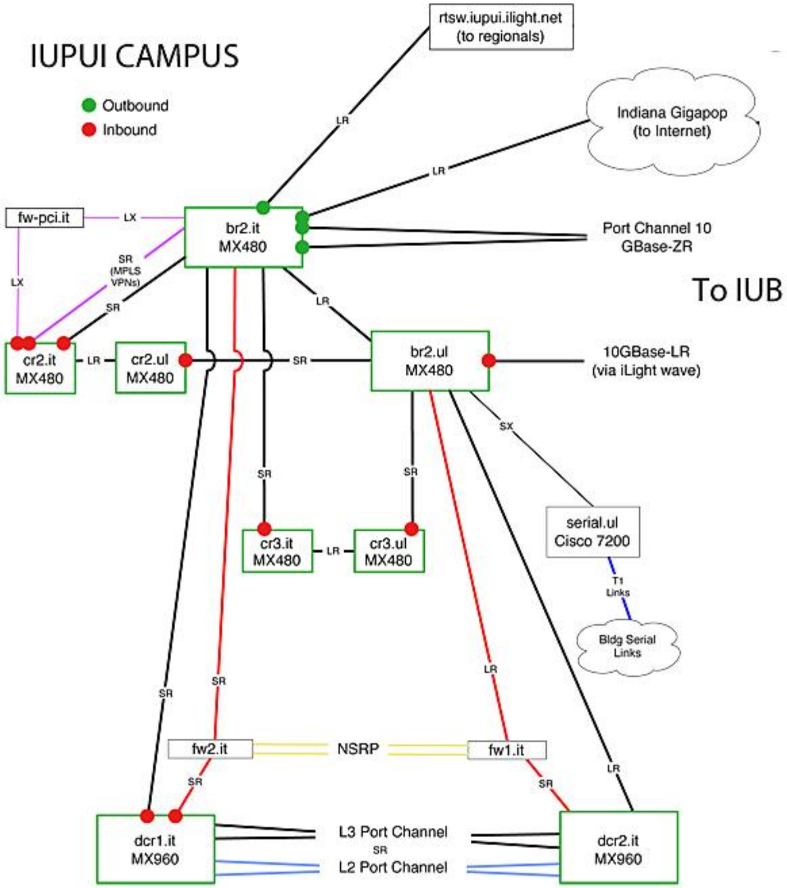


Network Master Plan

- Started in 2008
- Overhaul core network infrastructure at IUB and IUPUI
- Security funding included
- Goals of core overhaul
  - All buildings dual-homed
  - At least 10 Gb/s everywhere
- Population at IUB/IUPUI : 85,000

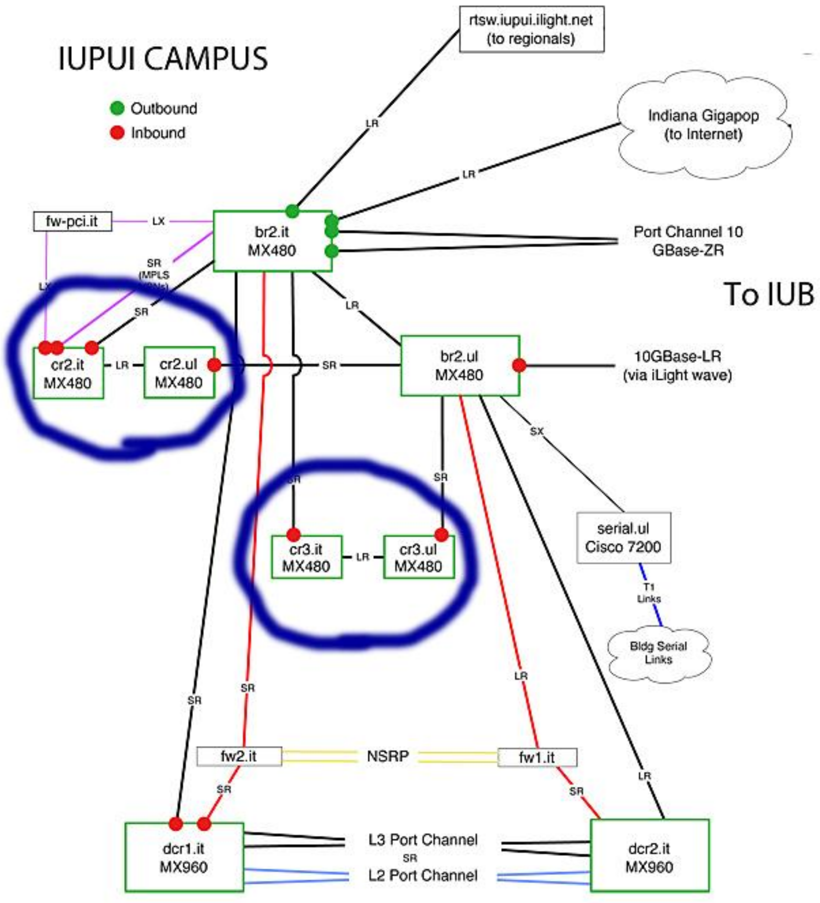
The final product





Mirrors

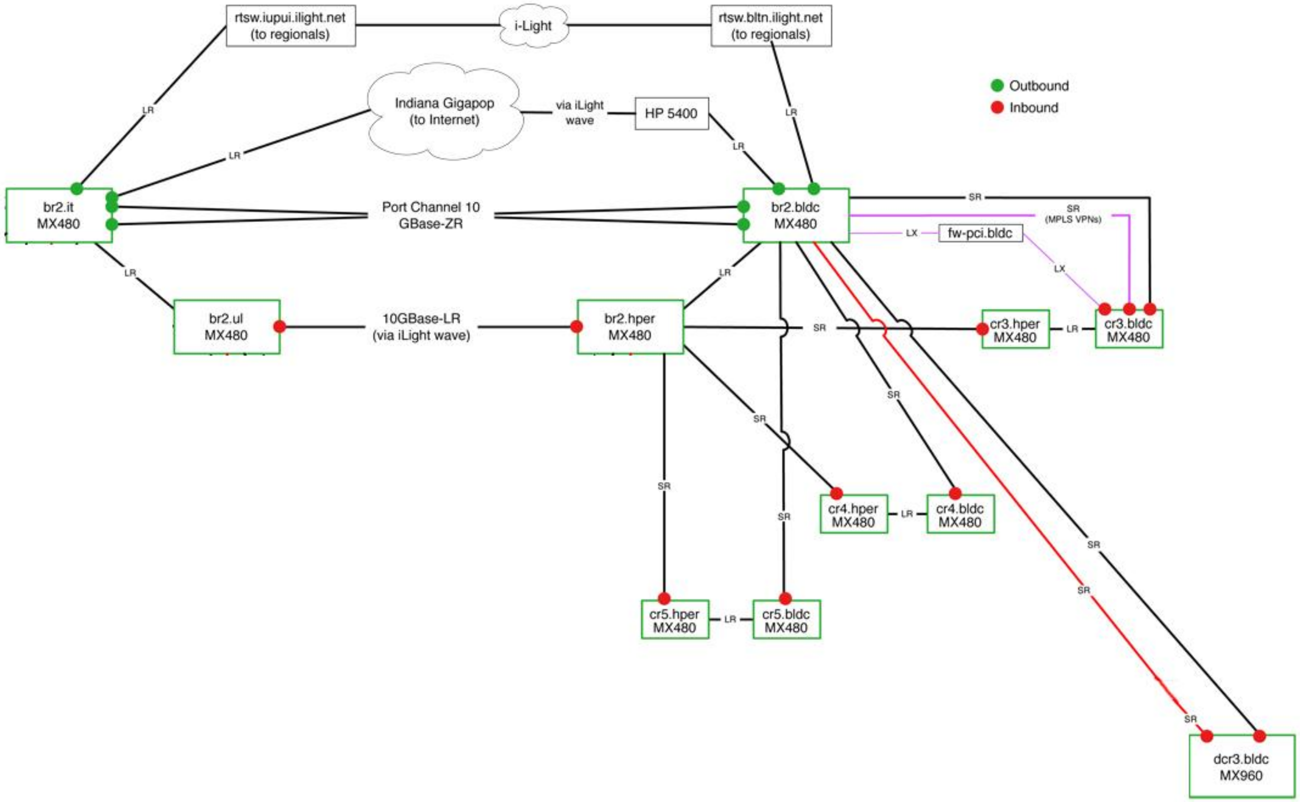
- Unidirectional mirrors
  - Copy outbound pkts at border
  - Copy inbound pkts on core routers
  - 9 @ IUB / 7 @ IUPUI
- Copied traffic sent via fiber to IDS

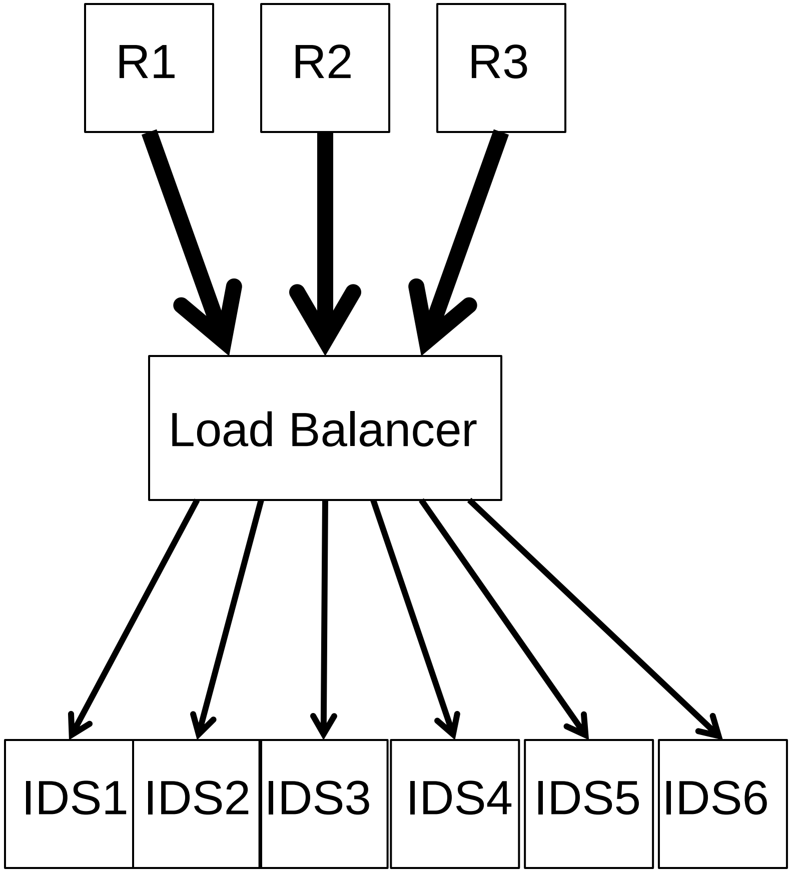


Router pairs

- Core routers are paired
  - Routers are in separate buildings
- Pairs service multiple buildings
- Traffic can route to a building via either router in a pair

Internet egress





Beyond single box IDS

- Large systems can handle multi-gigabit
  - Adding capacity?
  - multiple feeds?
- 16 feeds across two campuses
- We need a load balancer! And a cluster!

Load balancing : Build Your Own

- Software load balancing
  - 1 Gb
  - Does not scale to multiple feeds
- Surplus routers or switches
  - Lack of access to spare routers
  - Hardware warranty support

Load balancing : Commercial

- Many excellent solutions
- Even on a reasonably well funded project, still too expensive
- Limited ability to customize load balancing for issues unique to research and academic networking

OpenFlow™

Indiana Center for Network Translational Research and Education

InCNTRE

Enter OpenFlow

- InCNRTE
  - Practical applications for OpenFlow
  - Access to programming skill
  - Access to hardware for testing and development

STANFORD
UNIVERSITY

Berkeley
UNIVERSITY OF CALIFORNIA

PRINCETON
UNIVERSITY

Massachusetts
Institute of
Technology

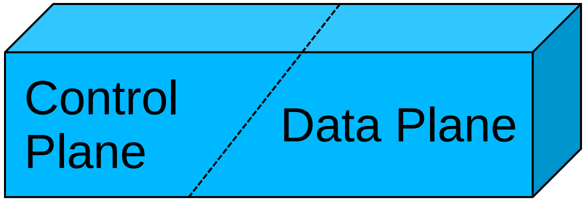
Washington
University in St.Louis

What is OpenFlow?

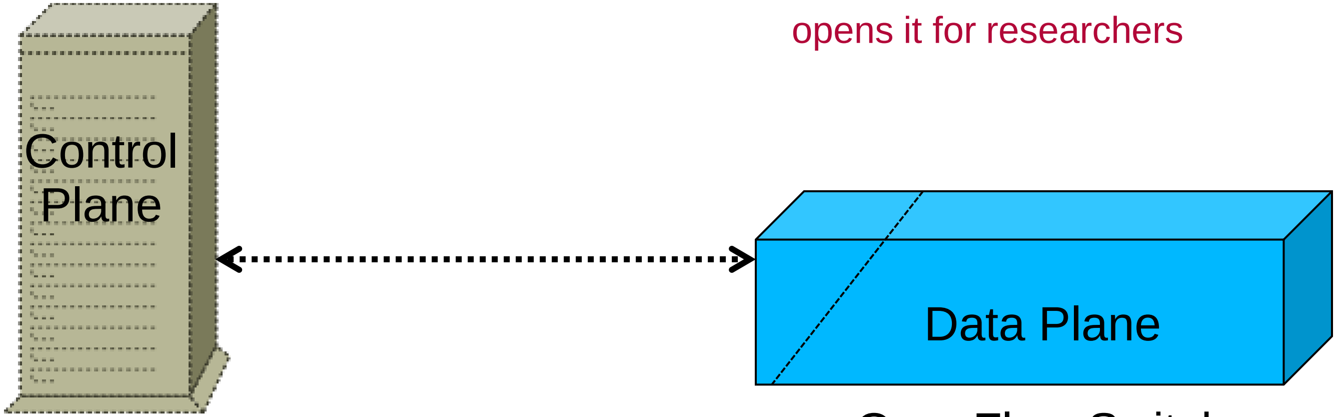
- A dominant component of Software Defined Networking
- Implemented by several vendors
- Compromise between research demands and network vendors’ requirements
- Currently deployed on several campuses

Control Plane and Data Plane

- Network devices have a control plane and a data plane
- Vendors have both controller plane and data plane locked as part of the firmware
- OpenFlow separates the control plane and opens it for researchers



Conventional L2 Switch

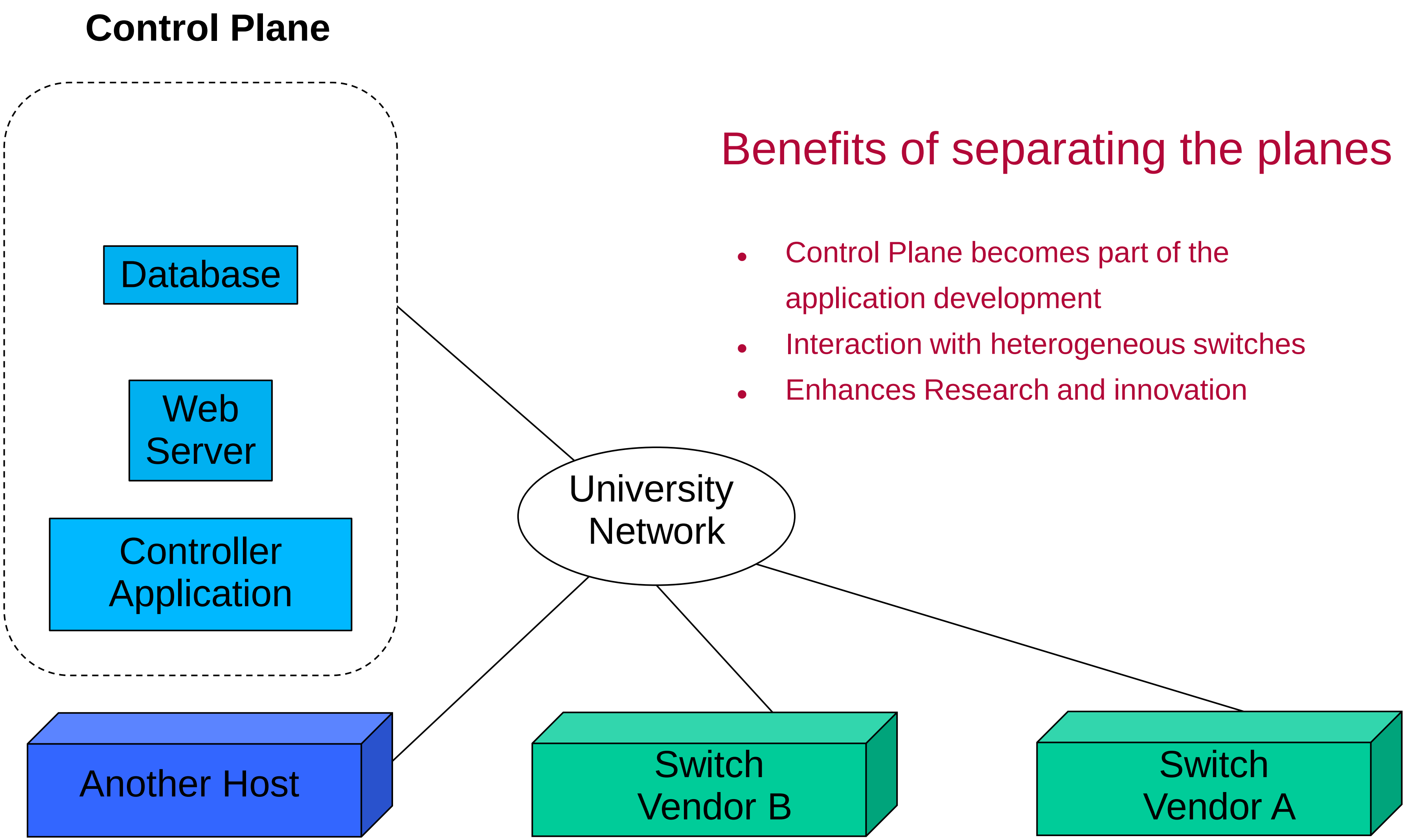


Commodity Server

OpenFlow Switch

Benefits of separating the planes

- Control Plane becomes part of the application development
- Interaction with heterogeneous switches
- Enhances Research and innovation

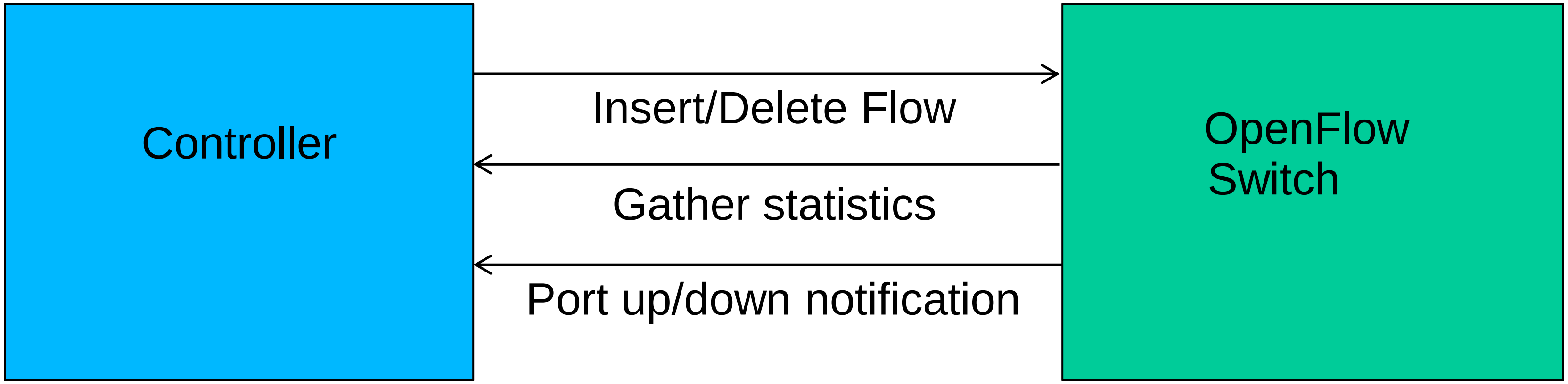


Control plane

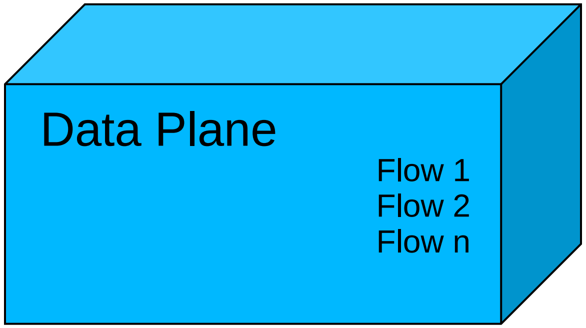
- Done by a controller using commodity hardware/software
- Controller usually implemented in high level language
  - Beacon
  - NOX
  - Floodlight

Interaction with data plane

- Insert/modify flows
- Up/down ports
- Gather statistics
- Detect switch changes



What are flows?



Data Plane

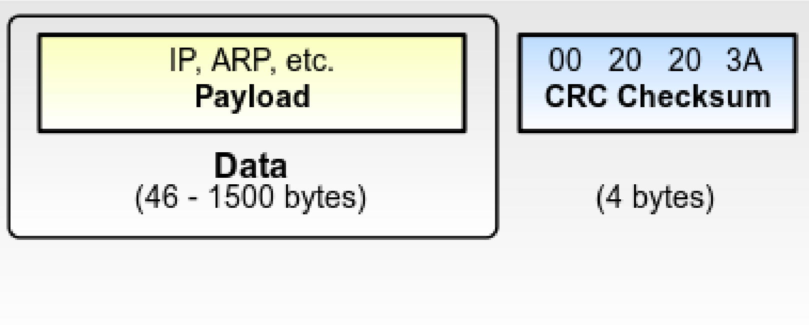
Flow 1
Flow 2
Flow n

- Headers to match against packets
- Counters for the rules
- Actions

Flow 1:

Header Fields	**Counters**	**Actions**
Nw_src =192.168.1.5, Nw_proto=tcp	Packet match: 326	Output to ports: 5,6

Headers Fields



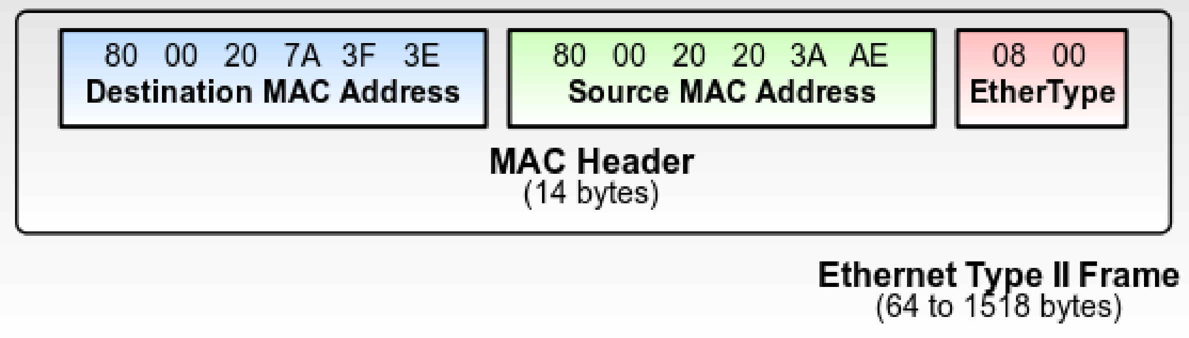
IP, ARP, etc.
Payload

Data
(46 - 1500 bytes)

00 20 20 3A
CRC Checksum

(4 bytes)

- What are header fields?
- Matches can be based on several factors related to layers 2-4 and vlan among others
- Masking is possible
- Priority



80 00 20 7A 3F 3E
Destination MAC Address

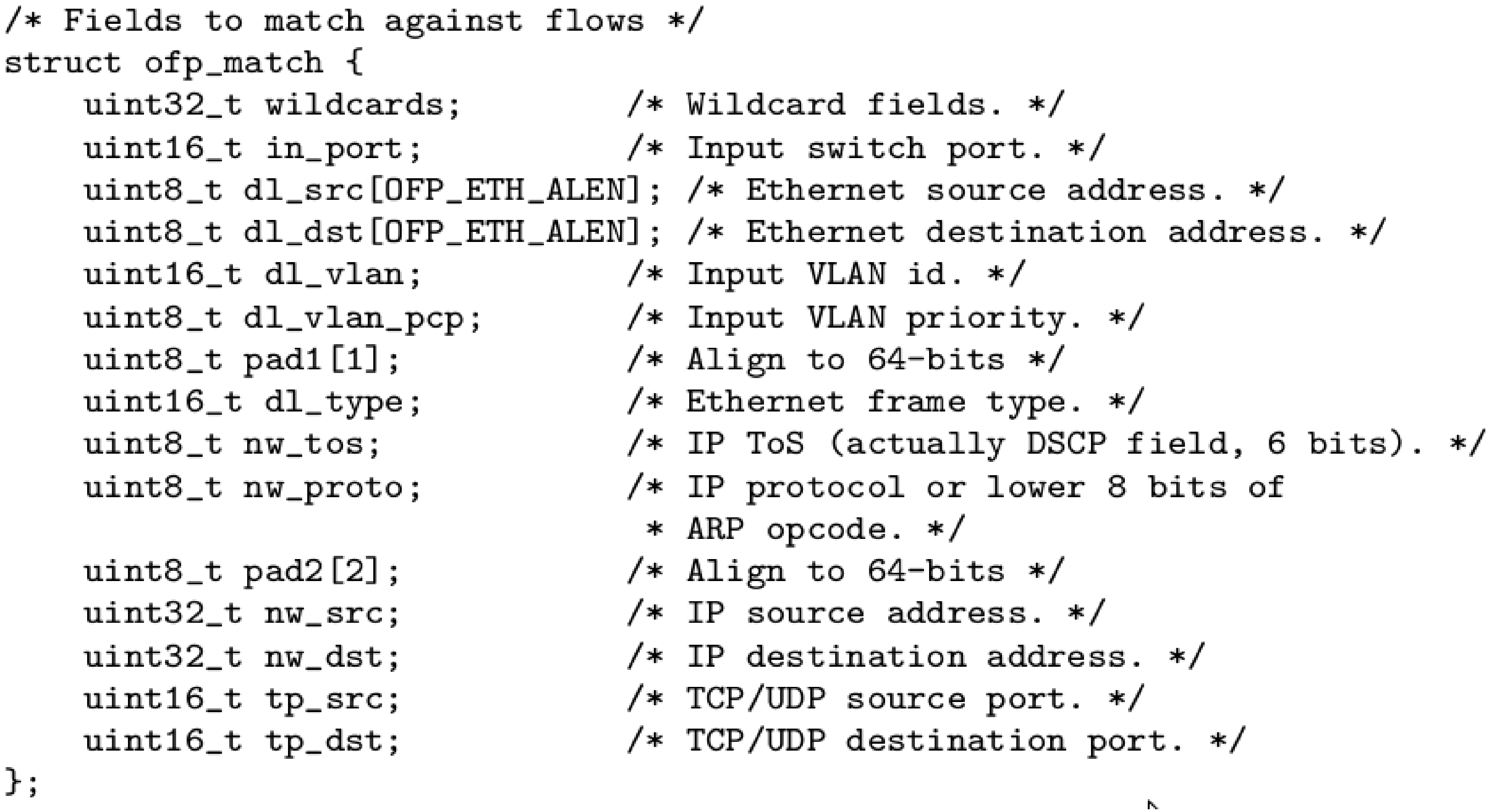
80 00 20 20 3A AE
Source MAC Address

08 00
EtherType

MAC Header
(14 bytes)

Ethernet Type II Frame
(64 to 1518 bytes)

Matches



```
/* Fields to match against flows */
struct ofp_match {
    uint32_t wildcards;        /* Wildcard fields. */
    uint16_t in_port;          /* Input switch port. */
    uint8_t dl_src[OFP_ETH_ALEN]; /* Ethernet source address. */
    uint8_t dl_dst[OFP_ETH_ALEN]; /* Ethernet destination address. */
    uint16_t dl_vlan;          /* Input VLAN id. */
    uint8_t dl_vlan_pcp;       /* Input VLAN priority. */
    uint8_t pad1[1];           /* Align to 64-bits */
    uint16_t dl_type;          /* Ethernet frame type. */
    uint8_t nw_tos;            /* IP ToS (actually DSCP field, 6 bits). */
    uint8_t nw_proto;          /* IP protocol or lower 8 bits of
                                * ARP opcode. */
    uint8_t pad2[2];           /* Align to 64-bits */
    uint32_t nw_src;           /* IP source address. */
    uint32_t nw_dst;           /* IP destination address. */
    uint16_t tp_src;           /* TCP/UDP source port. */
    uint16_t tp_dst;           /* TCP/UDP destination port. */
};
```

Actions

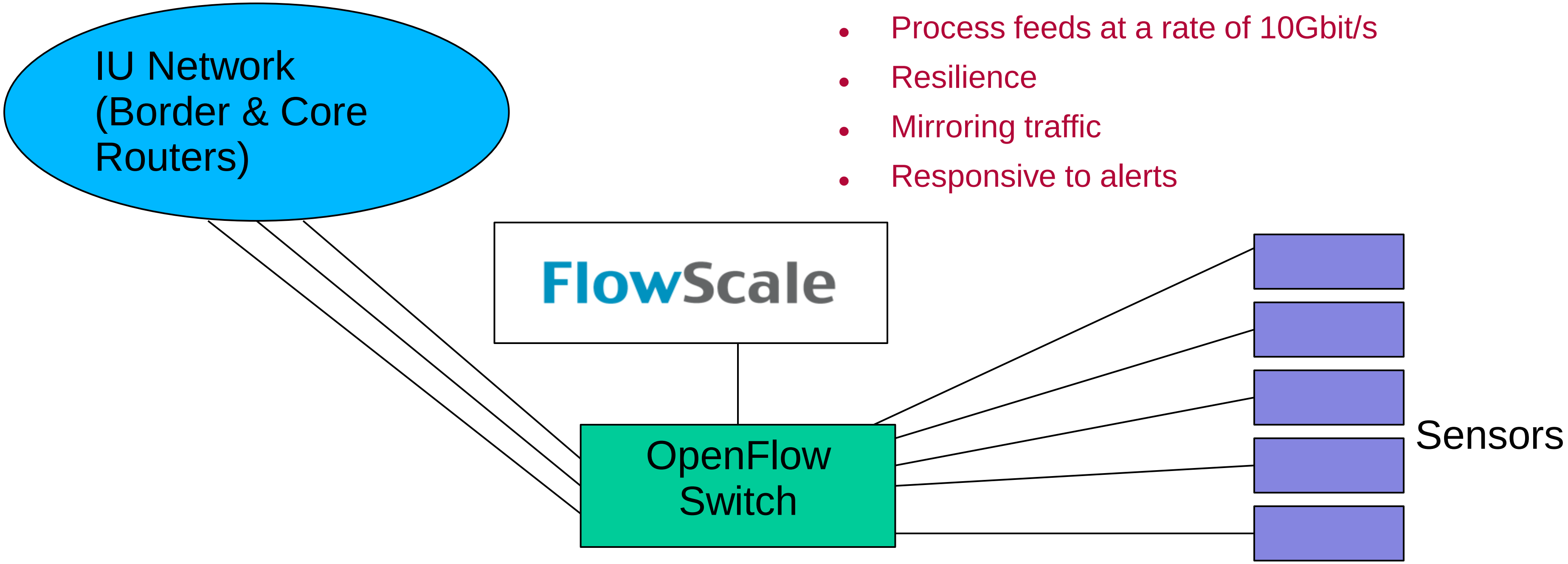
- Output
- Set/strip VLAN id
- Set data link src/dst
- Set IP src/dst
- Set network Type of Service
- Set transport src/dst
- Set 802.1q priority

Flow examples

Header Fields Nw_src =192.168.1.5, Nw_proto=tcp, Priority=100	**Counter** 326	**Actions** Output to ports: 5,6
Header Fields dl_type =0x86DD	**Counter** 45	**Actions** NONE
Header Fields dl_type =0x0800, Priority=50	**Counter** 1488	**Actions** Output to ports: 9

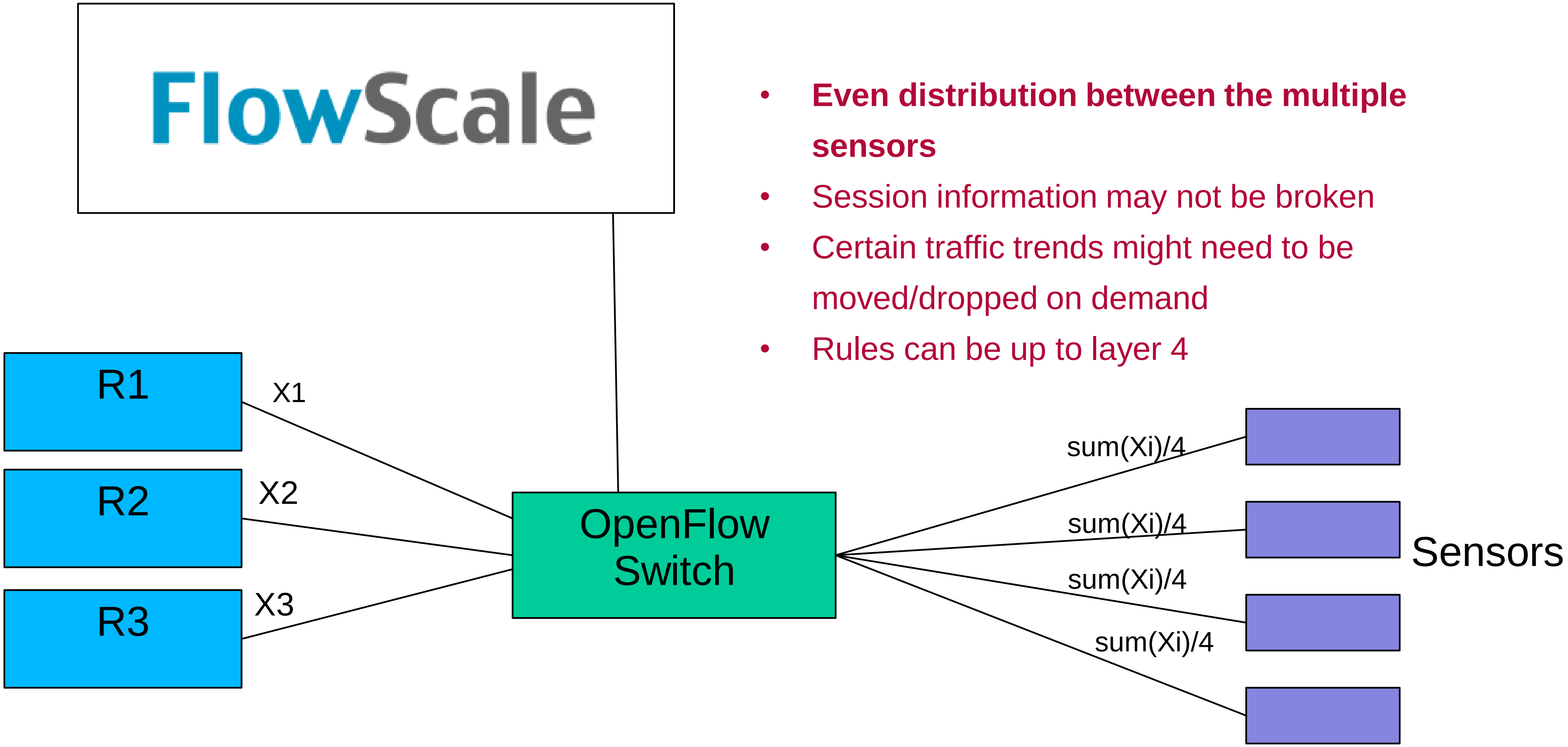
FlowScale

- Using Top of Rack switch to evenly distribute traffic (incoming and outgoing)
- Process feeds at a rate of 10Gbit/s
- Resilience
- Mirroring traffic
- Responsive to alerts



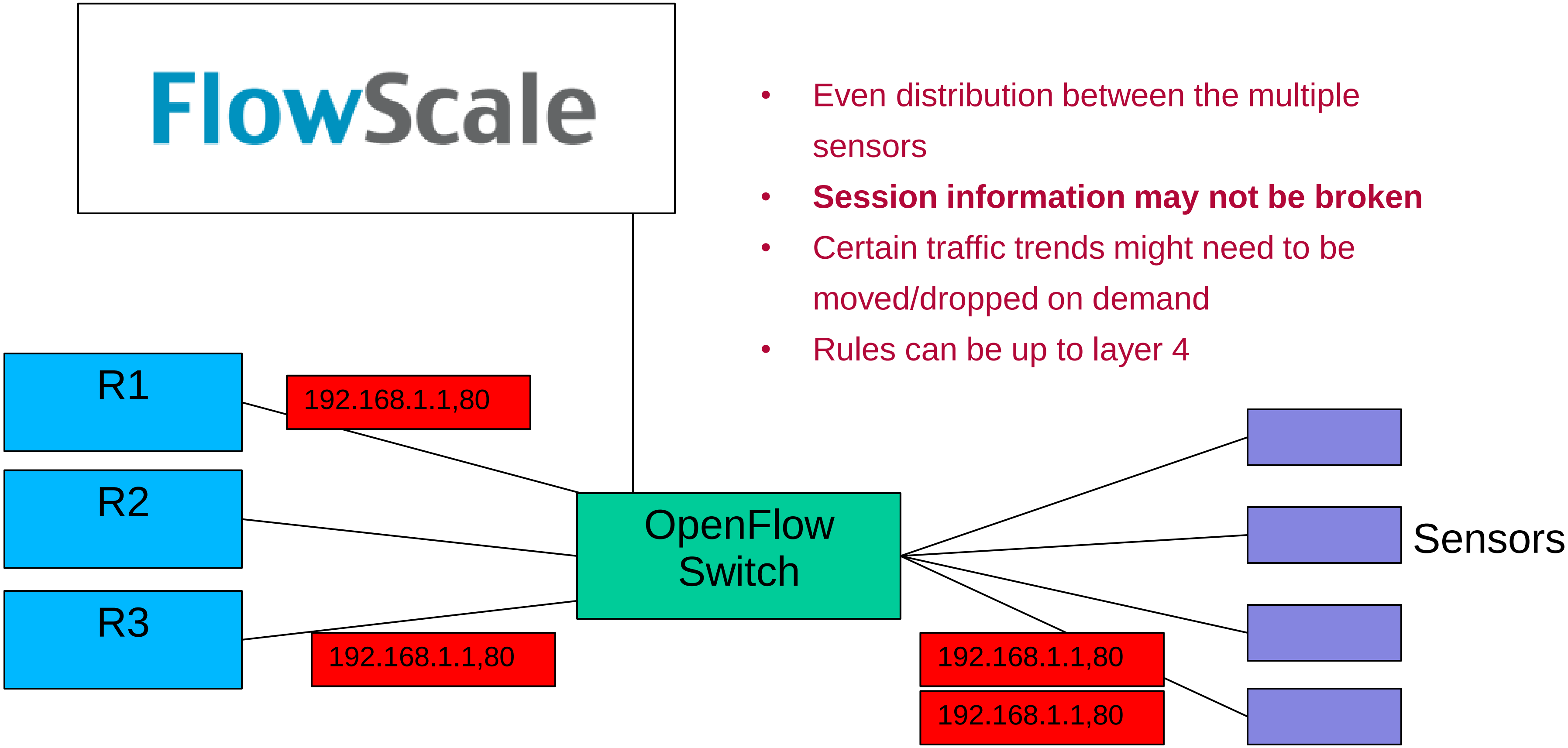
Load Distribution

- **Even distribution between the multiple sensors**
- Session information may not be broken
- Certain traffic trends might need to be moved/dropped on demand
- Rules can be up to layer 4



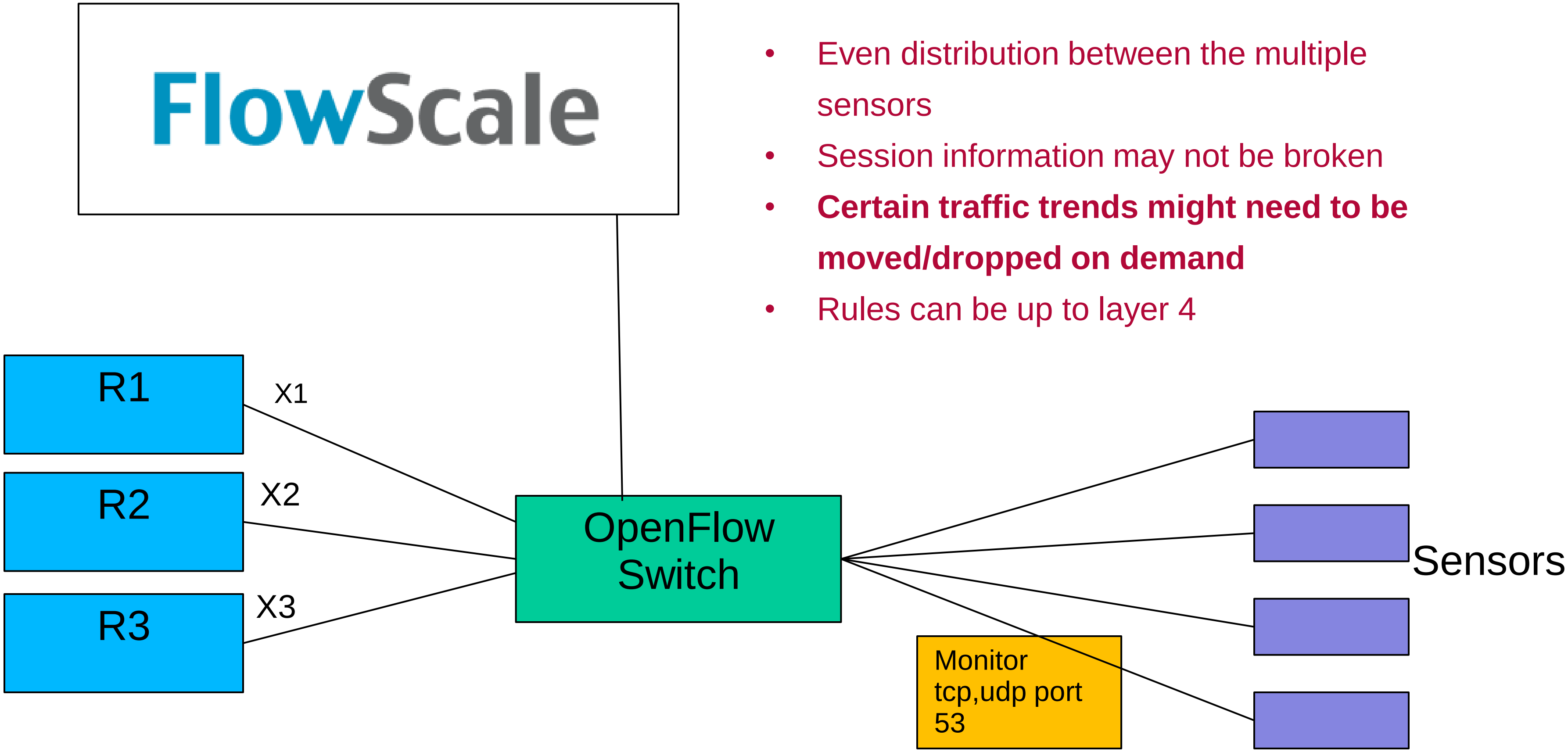
Load Distribution

- Even distribution between the multiple sensors
- **Session information may not be broken**
- Certain traffic trends might need to be moved/dropped on demand
- Rules can be up to layer 4



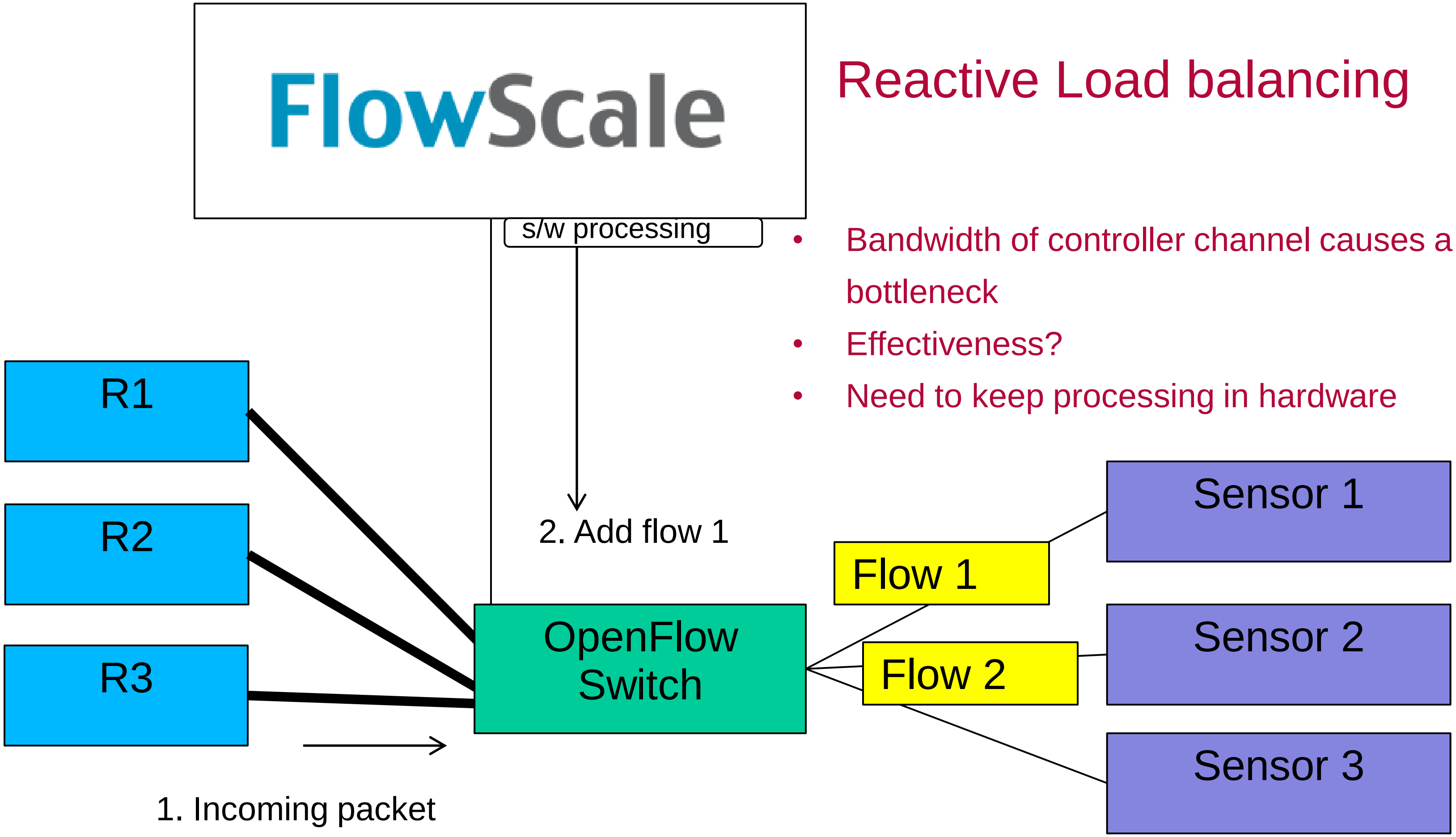
Load Distribution

- Even distribution between the multiple sensors
- Session information may not be broken
- **Certain traffic trends might need to be moved/dropped on demand**
- Rules can be up to layer 4



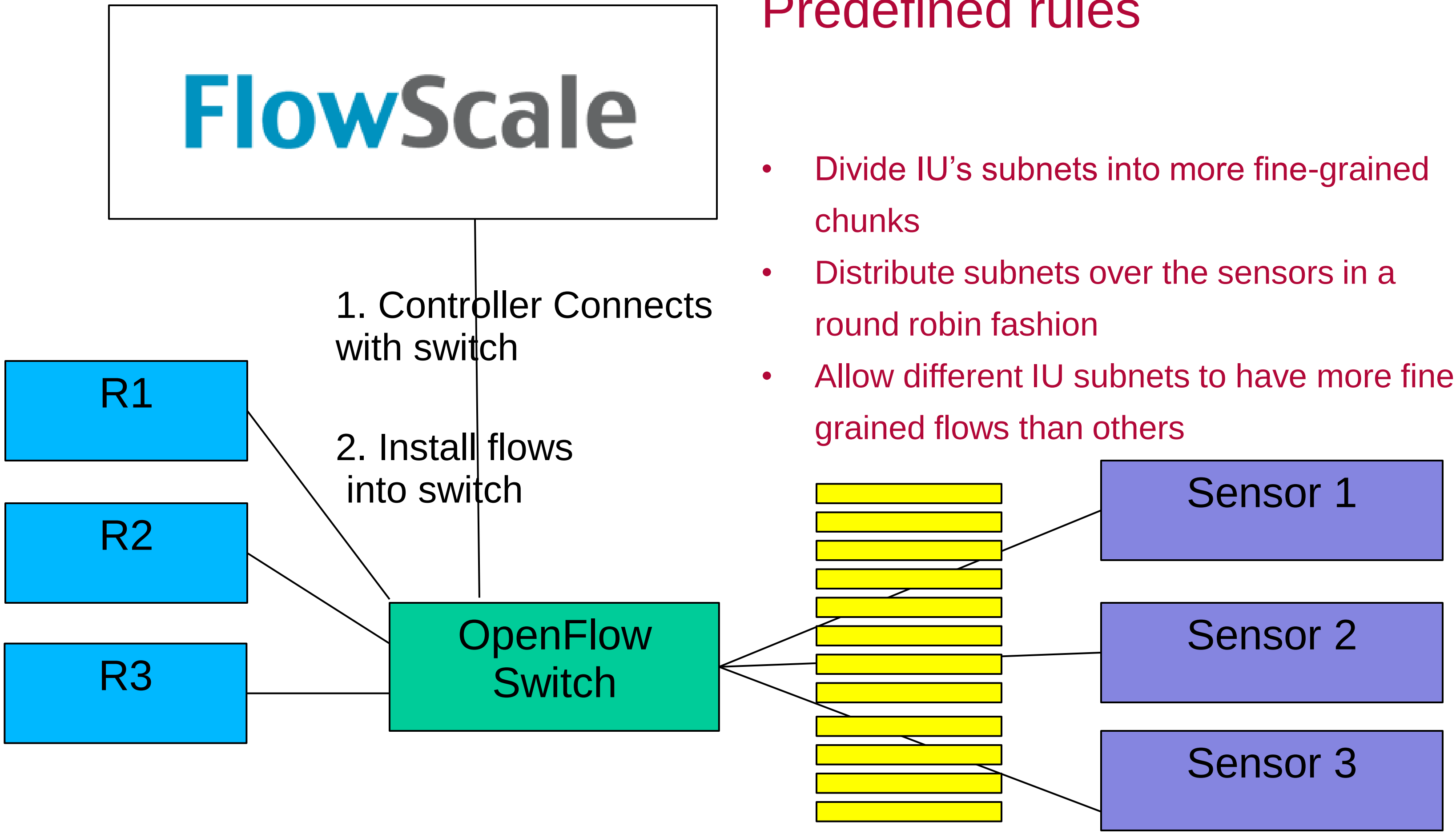
Reactive Load balancing

- Bandwidth of controller channel causes a bottleneck
- Effectiveness?
- Need to keep processing in hardware



Predefined rules

- Divide IU’s subnets into more fine-grained chunks
- Distribute subnets over the sensors in a round robin fashion
- Allow different IU subnets to have more fine-grained flows than others


FlowScale
1. Controller Connects with switch
2. Install flows into switch
R1
R2
R3
OpenFlow Switch
Sensor 1
Sensor 2
Sensor 3

Predefined rules

192.168.1.0/24

4 flows

192.168.1.0/25 (src)
192.168.1.0/25 (dst)

192.168.1.128/25 (src)
192.168.1.128/25 (dst)

172.16.0.0/16

8 flows

172.16.0.0/18 (src)
172.16.0.0/18 (dst)

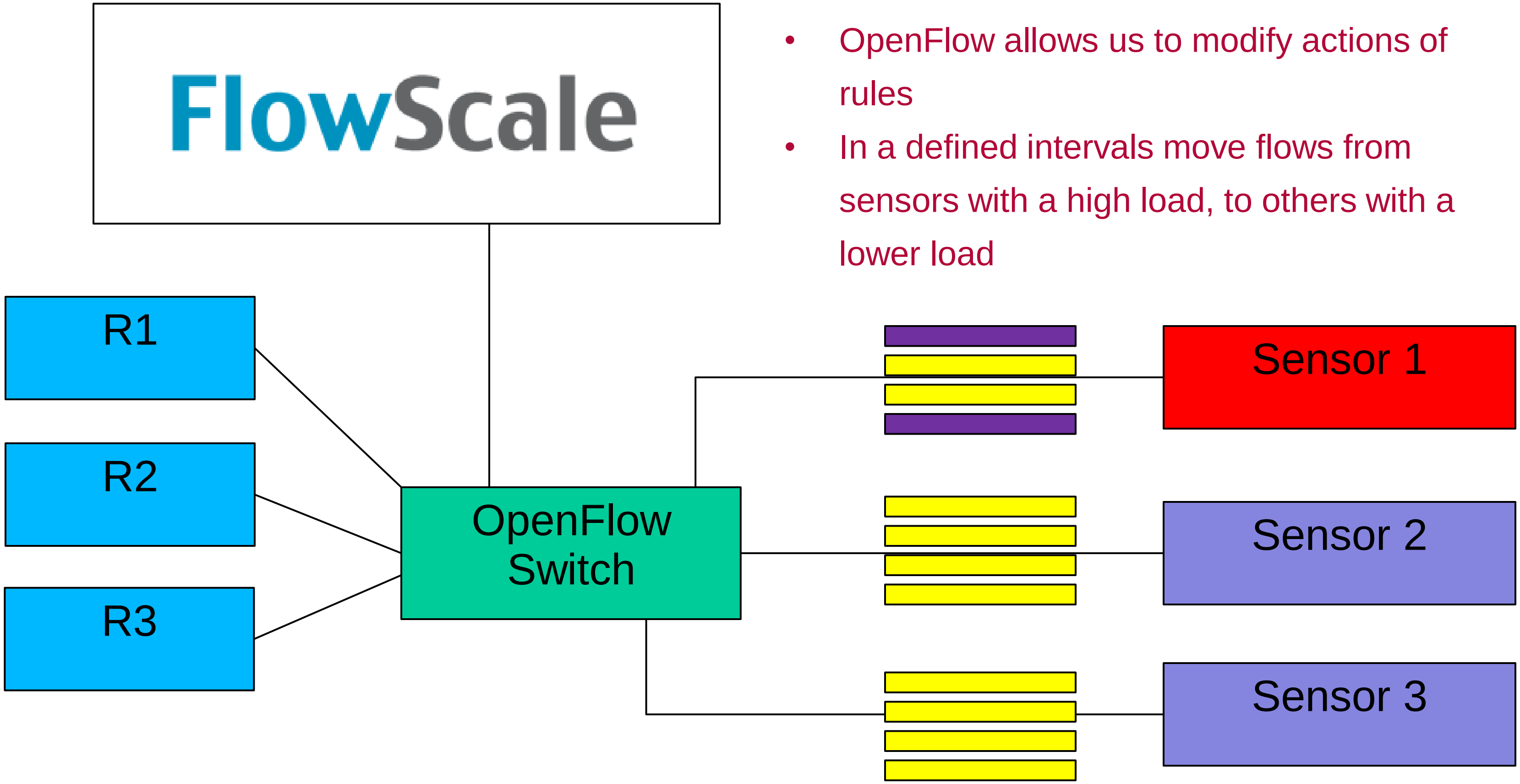
172.16.64.0/18 (src)
172.16.64.0/18 (dst)

172.16.128.0/18 (src)
172.16.128.0/18 (dst)

172.16.192.0/18 (src)
172.16.192.0/18 (dst)

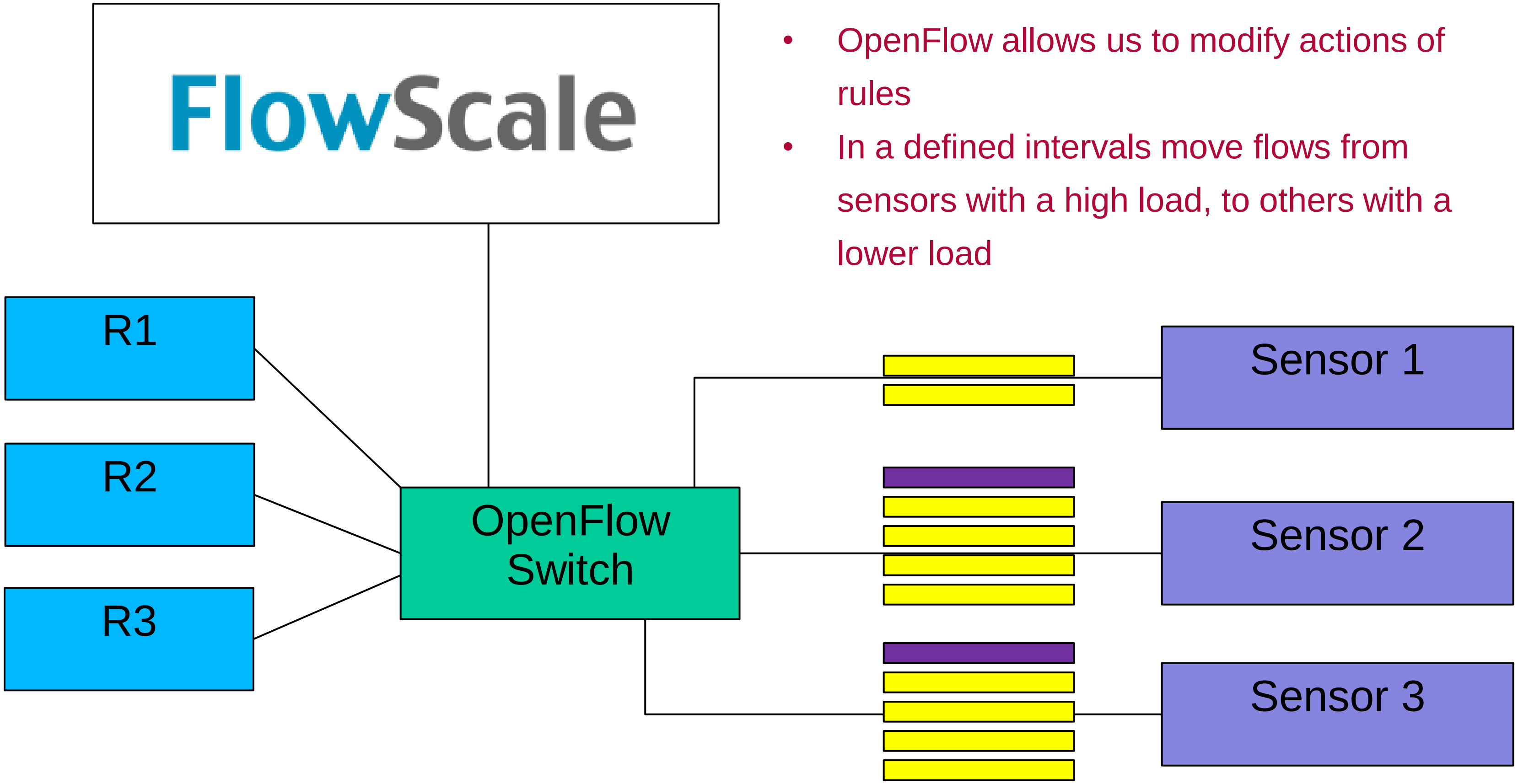
Hot swapping flows

- OpenFlow allows us to modify actions of rules
- In a defined intervals move flows from sensors with a high load, to others with a lower load



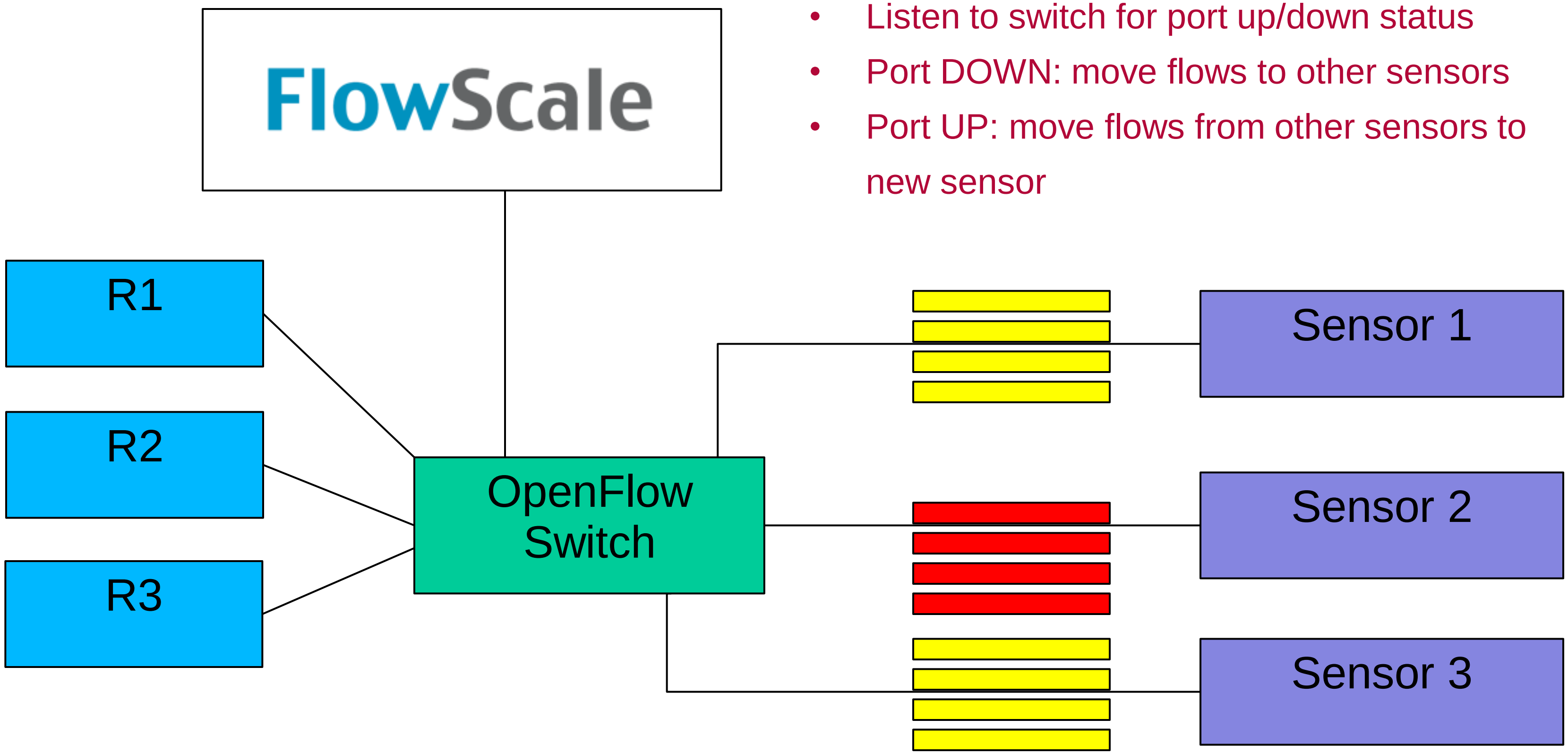
Hot swapping flows

- OpenFlow allows us to modify actions of rules
- In a defined intervals move flows from sensors with a high load, to others with a lower load



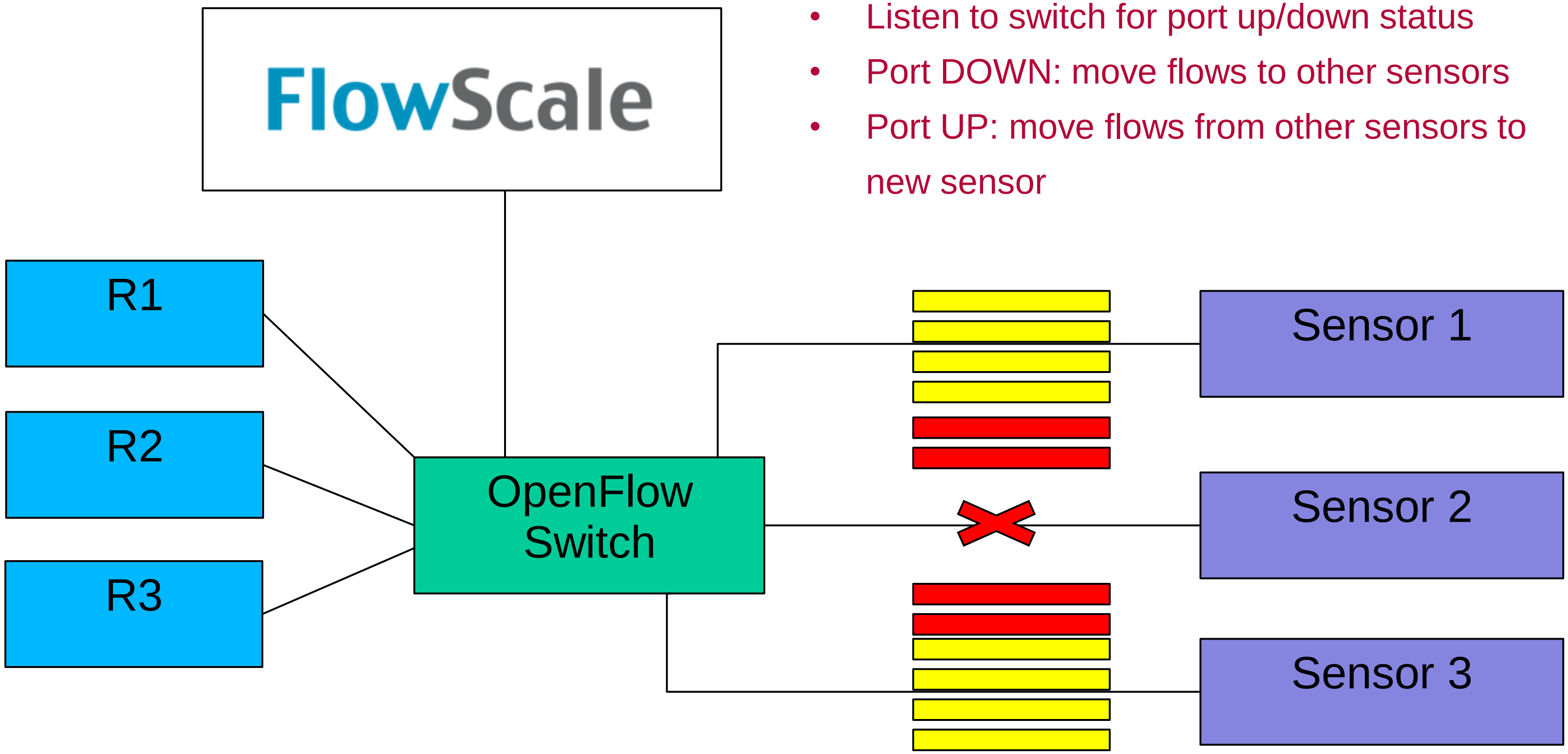
Resilience

- Listen to switch for port up/down status
- Port DOWN: move flows to other sensors
- Port UP: move flows from other sensors to new sensor



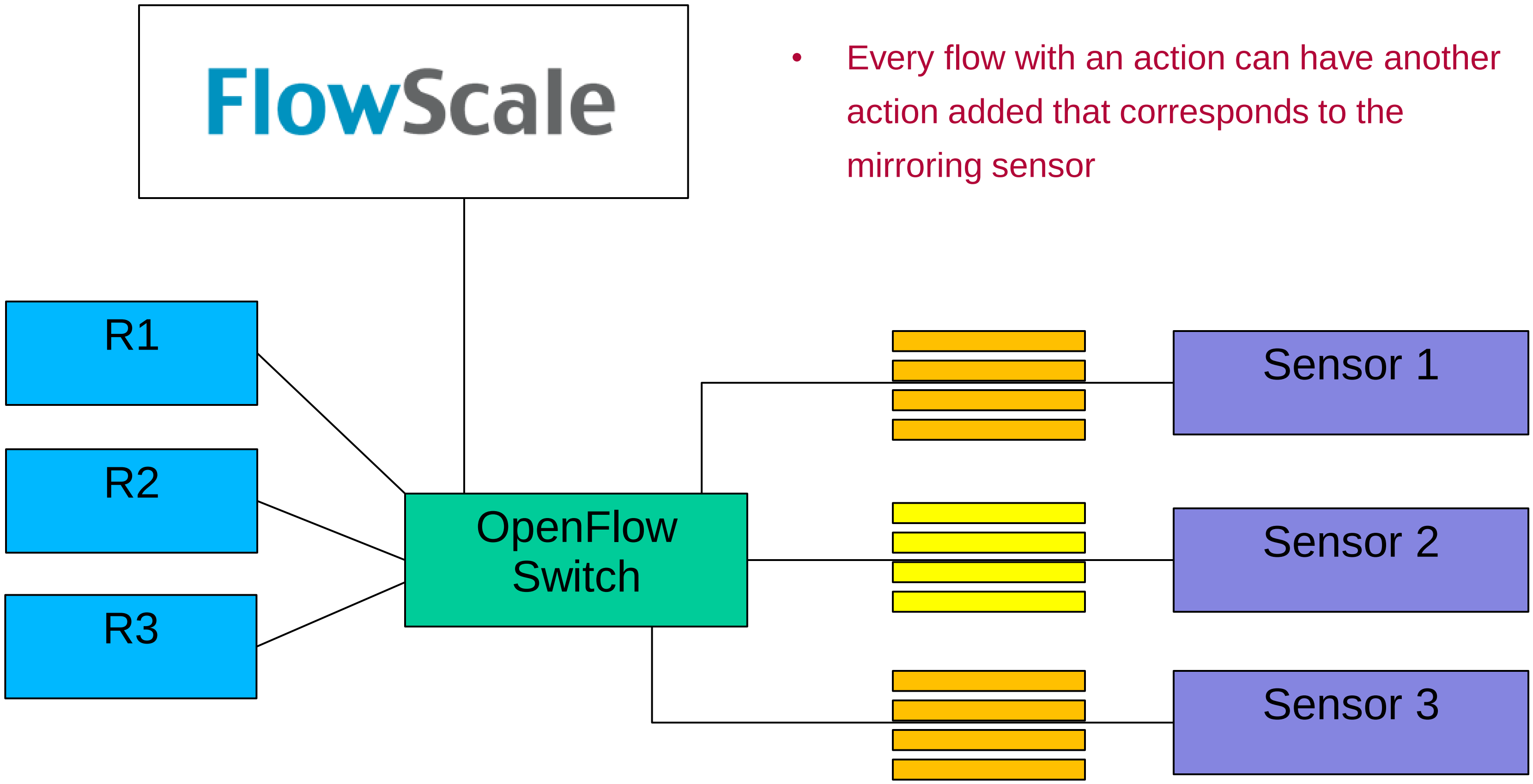
Resilience

- Listen to switch for port up/down status
- Port DOWN: move flows to other sensors
- Port UP: move flows from other sensors to new sensor



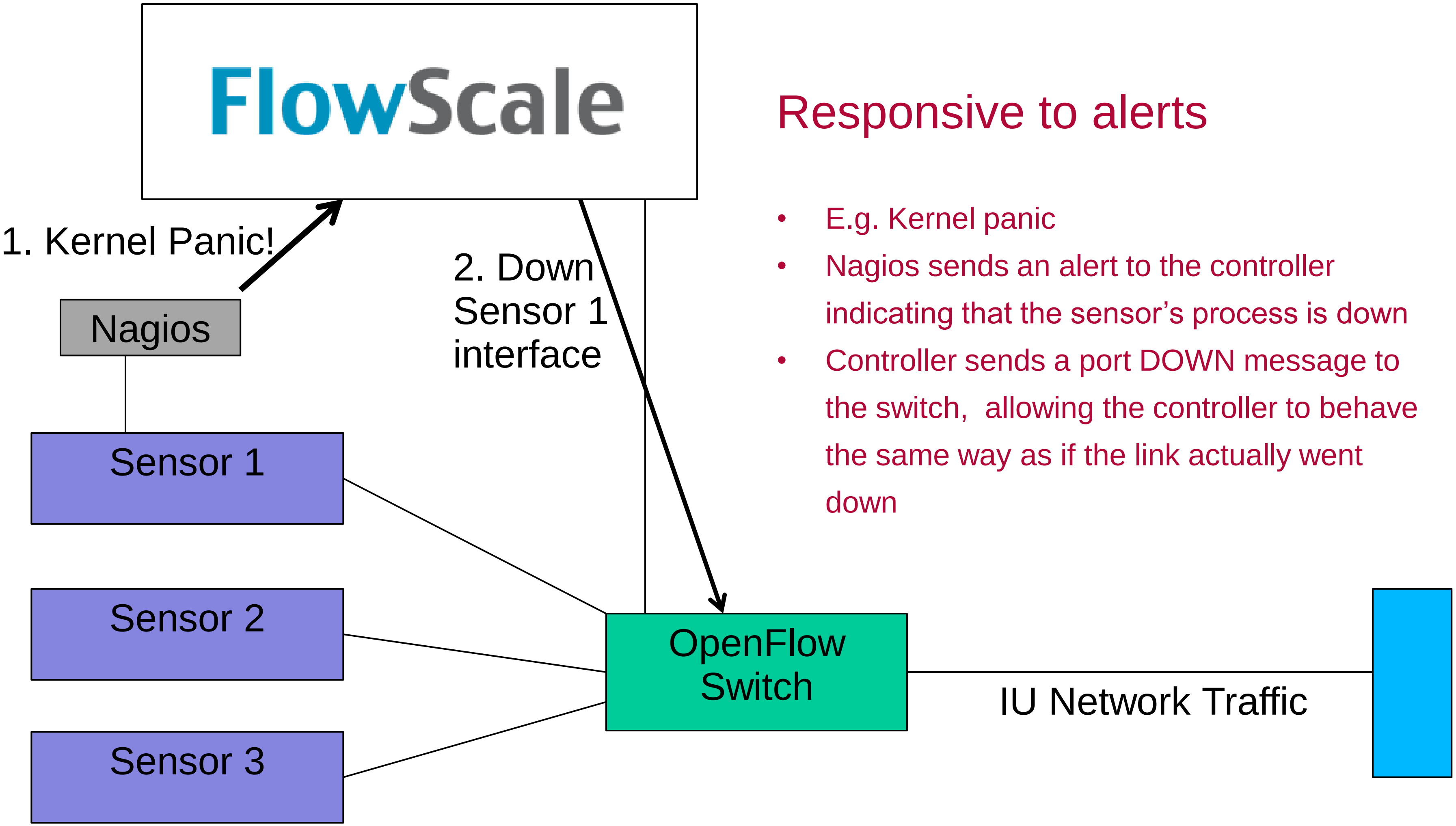
Mirroring Traffic

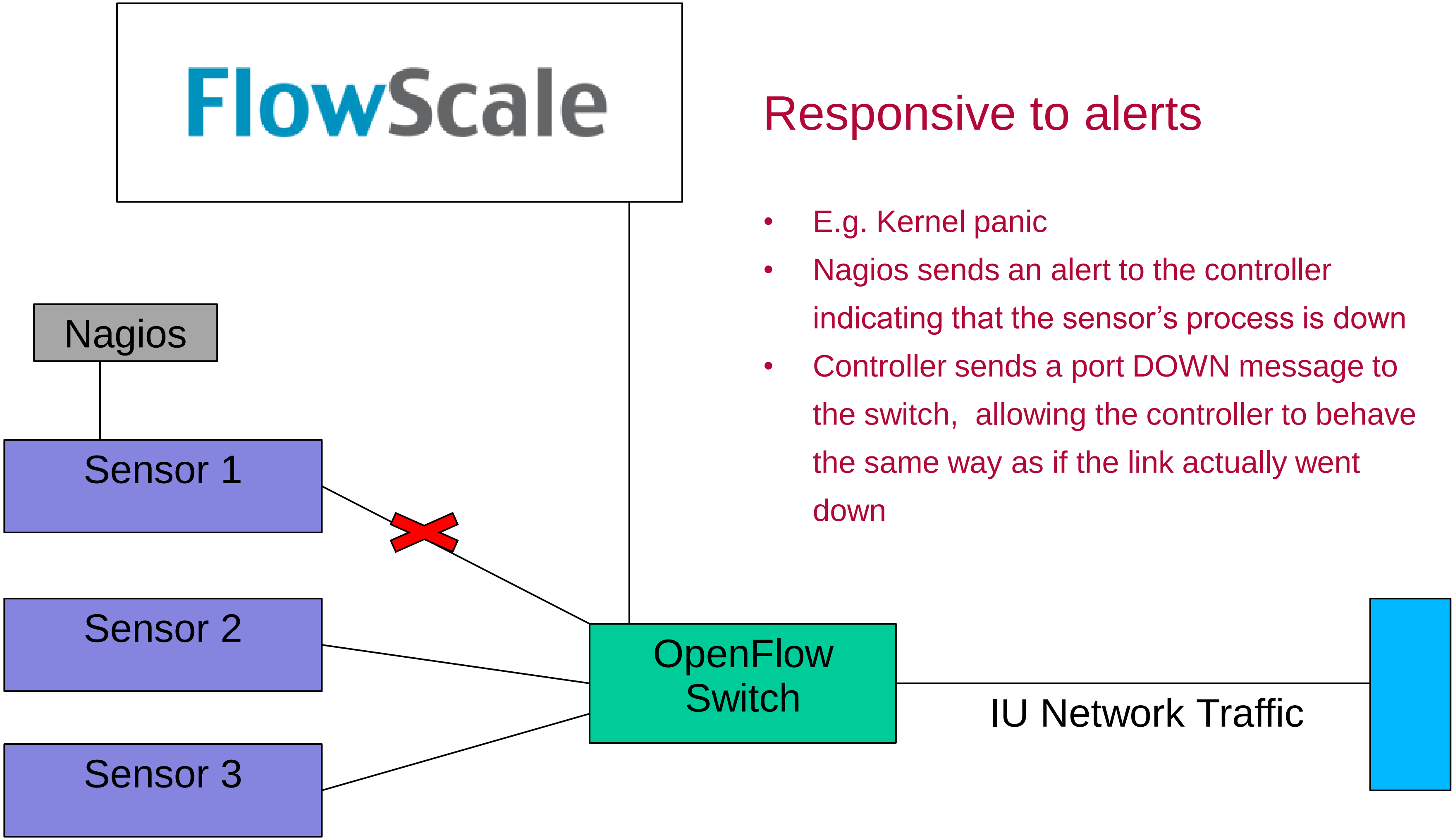
- Every flow with an action can have another action added that corresponds to the mirroring sensor



Responsive to alerts

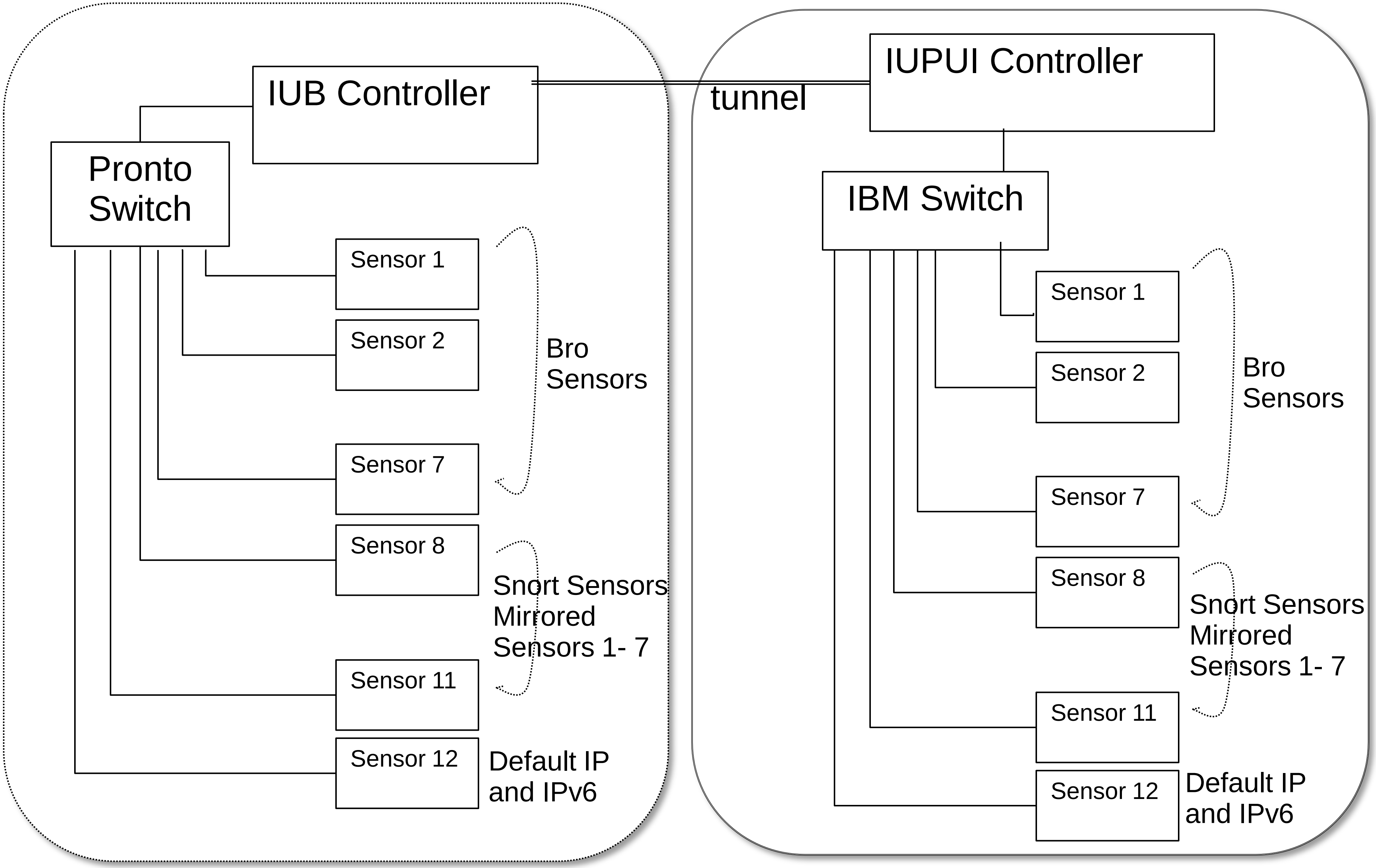
- E.g. Kernel panic
- Nagios sends an alert to the controller indicating that the sensor's process is down
- Controller sends a port DOWN message to the switch, allowing the controller to behave the same way as if the link actually went down





Responsive to alerts

- E.g. Kernel panic
- Nagios sends an alert to the controller indicating that the sensor's process is down
- Controller sends a port DOWN message to the switch, allowing the controller to behave the same way as if the link actually went down



IUB Controller

Pronto Switch

Sensor 1

Sensor 2

Sensor 7

Sensor 8

Sensor 11

Sensor 12

Bro Sensors

Snort Sensors Mirrored Sensors 1- 7

Default IP and IPv6

tunnel

IUPUI Controller

IBM Switch

Sensor 1

Sensor 2

Sensor 7

Sensor 8

Sensor 11

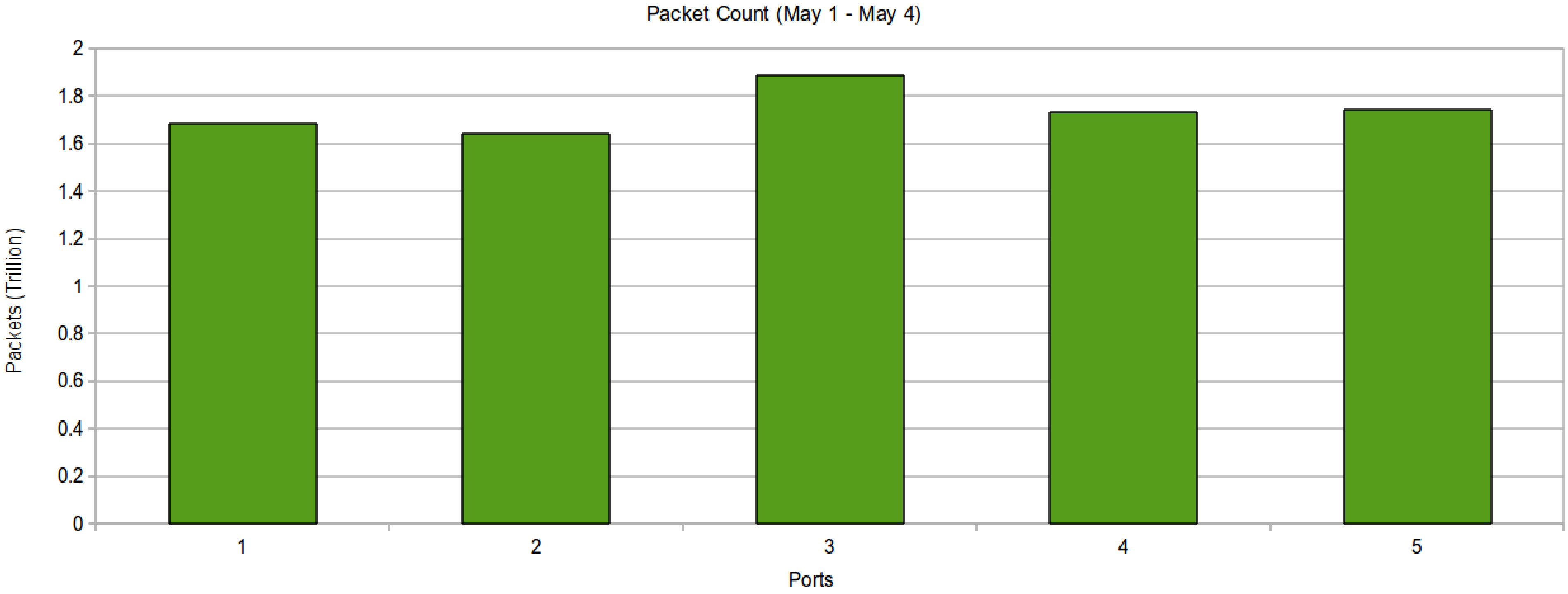
Sensor 12

Bro Sensors

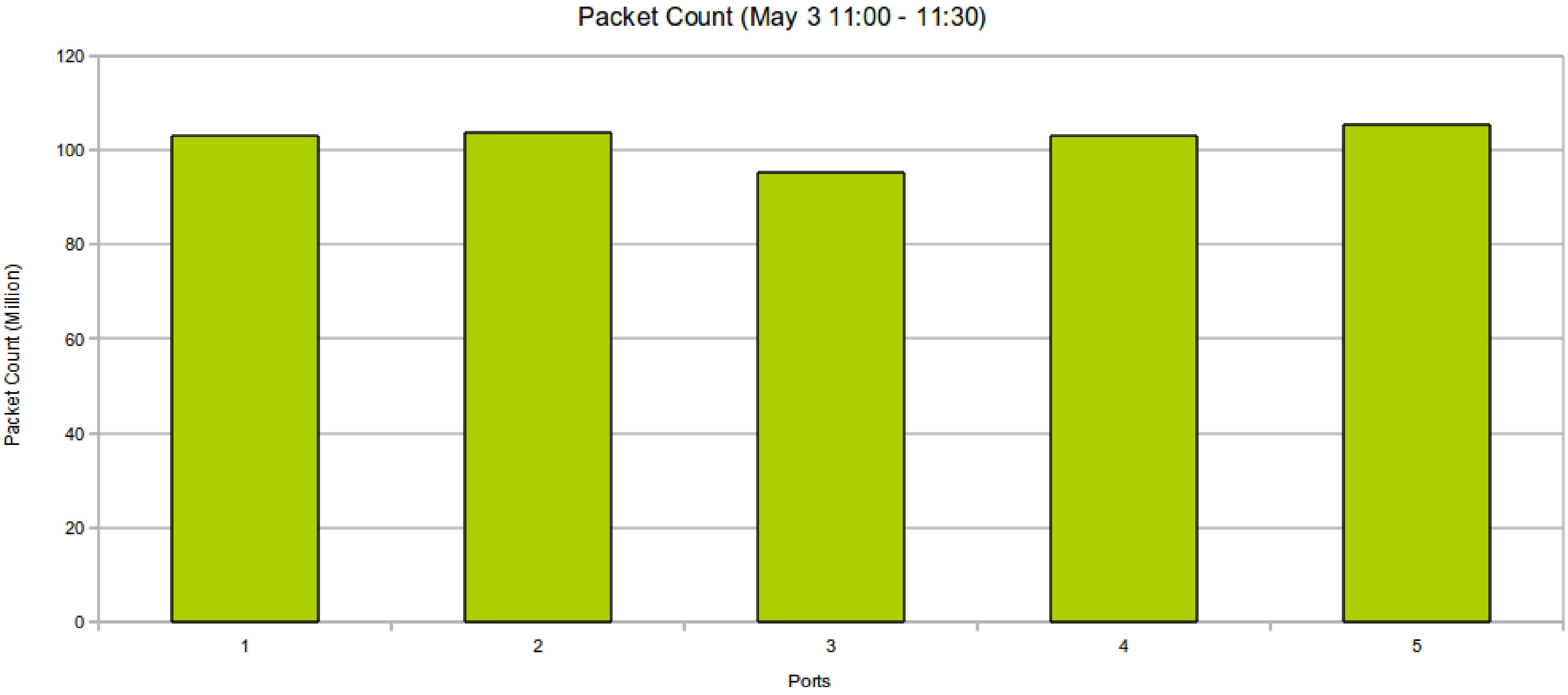
Snort Sensors Mirrored Sensors 1- 7

Default IP and IPv6

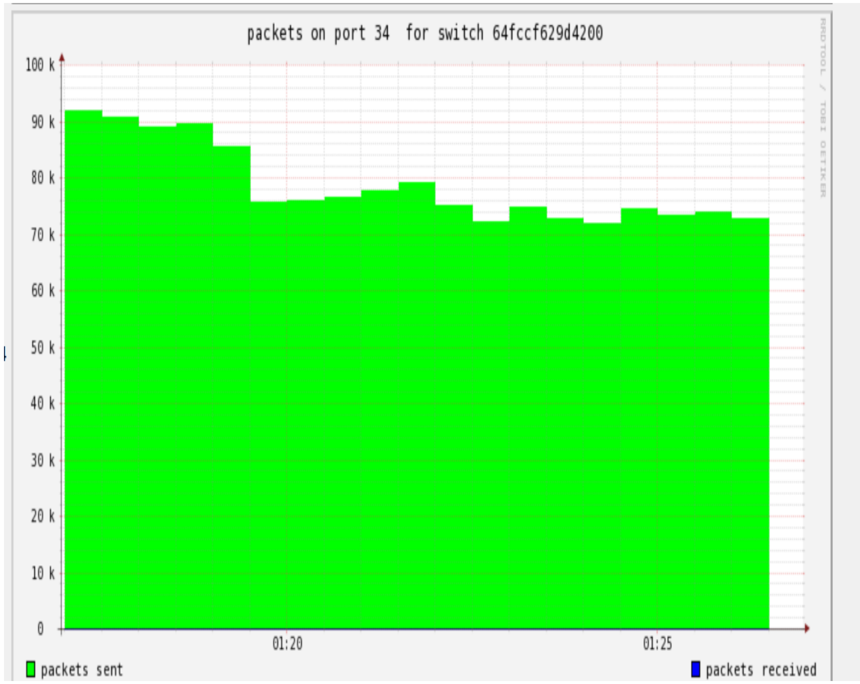
Results – Load Distribution

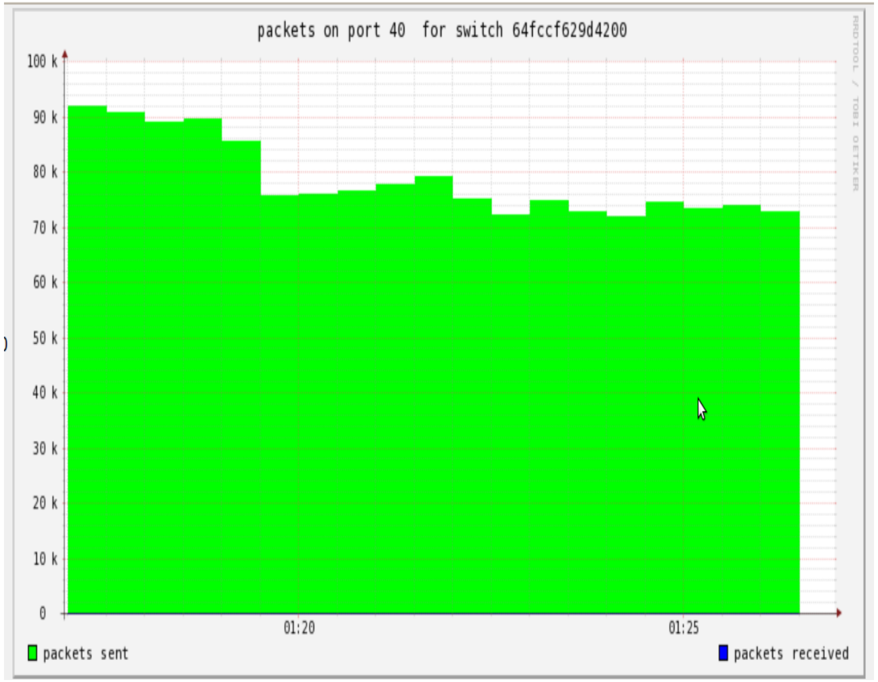


Results – Load Distribution



Results – Mirroring

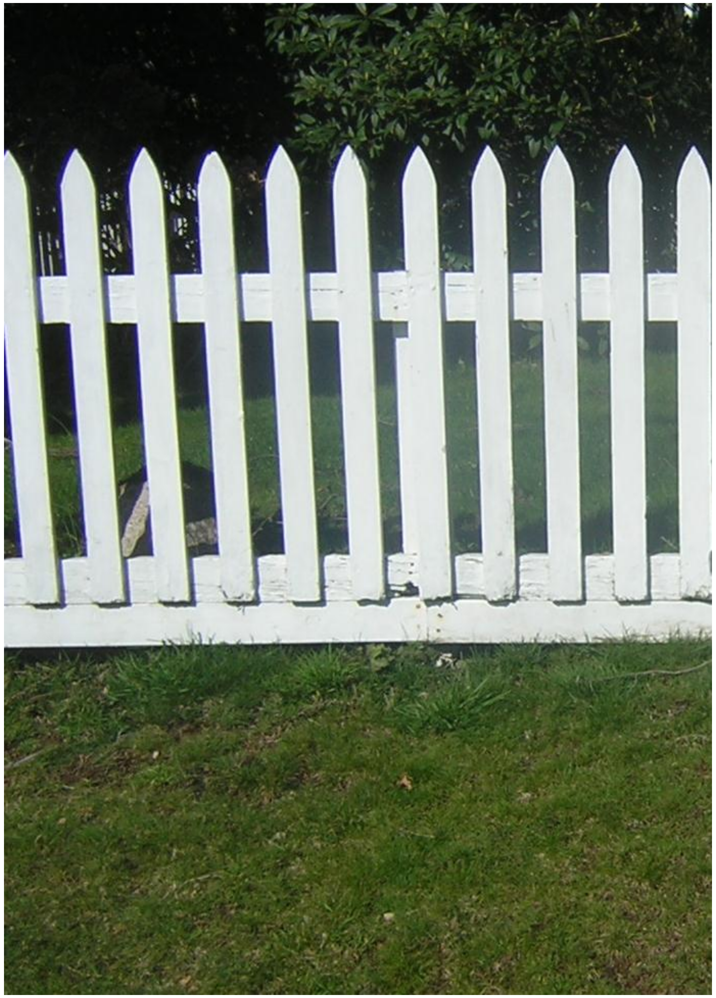




FlowScale

Summary

- ✓ Load Distribution
- ✓ Resilience
- ✓ Mirroring
- ✓ Responsive to external alerts



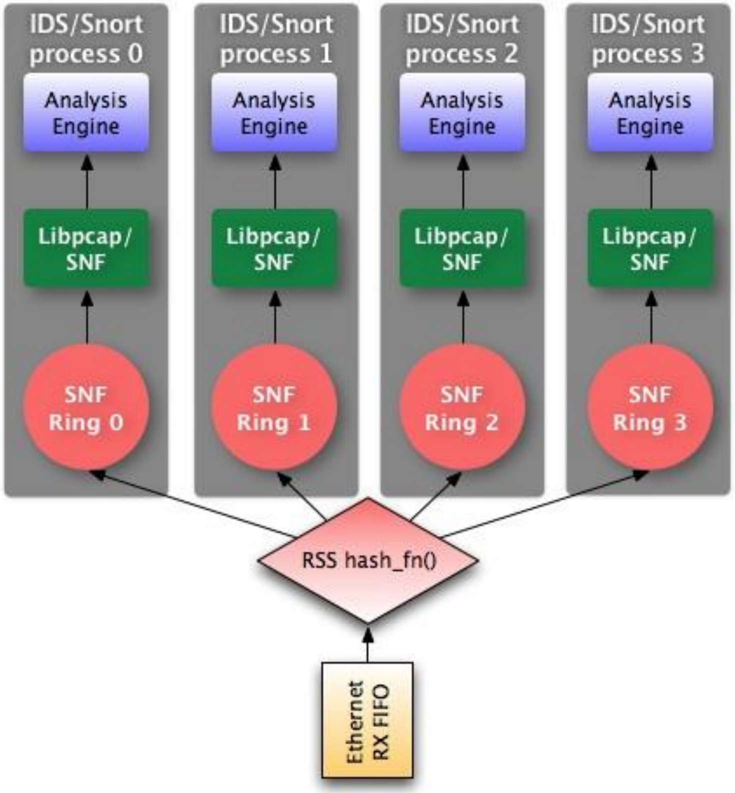
Limitations and future work

- Limitations
  - Session breaking
  - Most software is still beta
  - IPv6
- Future work
  - More fine-grained flows
  - Distribute flows based on weight of each sensor

IDS cluster hardware

- Dell R510 – manager
  - 12 core / 24 GB / 1.5 TB
- Dell R310 – OpenFlow controller
- Dell R410 (12) – workers
  - 12 core / 24 GB / 300 GB SAS
  - Myricom 10Gb NIC
  - HP Direct-Attach Cables
- FreeBSD 8
- Configuration management with Master Source
- Intra-cluster networking via private VLAN
- Load balanced traffic received via HP DAC

Another layer of load balancing – Myricom Sniffer10G

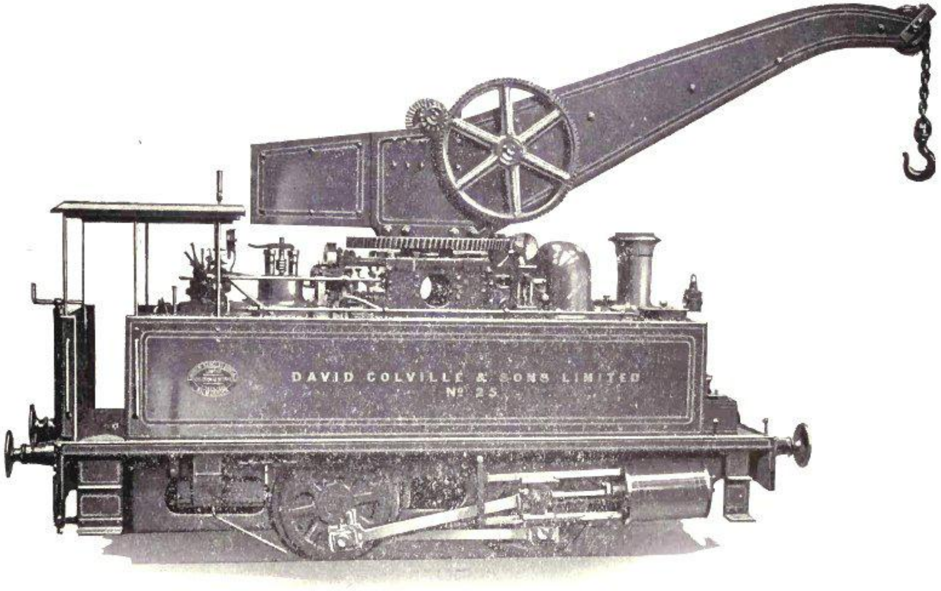


- Multiple ring buffers presented to OS
- Can perform IP-based load balancing or duplicate traffic to all rings
- Myricom provides a libpcap wrapper
- Sniffer10G controlled by environment variables

- Libpcap wrapper obscures per-ring stats
- Hard to gauge packet loss in snort
- Myricom provides tools to read packet counters and measure bandwidth at the NIC

Software stack

- Bro = Network analysis framework
  - Programmable
  - Acts like a protocol parser/logger
- Bro running on nodes 1-7
  - 10 workers per node
- Snort = packet grepper extraordinaire
- Snort running on nodes 8-11
  - 7 snort instances per node
- Node 12 monitor IPv6 traffic and catchall IPv4 traffic
- Node 12 is also our “tcpdump” host



Performance numbers

- IUB : 1.5 million pkt/sec / 3 Gb/s average
- IUB : Currently 500-750k / 1.5 Gb/s average
- Bro capture_loss
    - 3-5%
    - Short term spikes above 10%

Future cluster improvements

- FreeBSD Netmap
- Automate OS builds with NanoBSD
- Expand Bro usage
- Use Snort for heavy packet inspection
    - Think DLP

Thank you.

- Keith Lehigh: klehigh@iu.edu
- Ali Khalfan: ali.khalfan@gmail.com
- InCNTRE : incntre.iu.edu
- FlowScale : www.openflowhub.org/display/FlowScale/FlowScale+Home