Software Defined Networking and Security

Ivan Pepelnjak (ip@ipSpace.net)

ipSpace.net AG

Who is Ivan Pepelnjak (@ioshints)

- Networking engineer since 1985
- Technical director, later Chief Technology Advisor @ NIL Data Communications
- Consultant, blogger, book and webinar author @ ipSpace.net AG
- Teaching “Scalable Web Application Design” at University of Ljubljana

Cisco Champion

CISCO CCIE EMERITUS

vmware vEXPERT

Focus:

- Large-scale data centers and network virtualization
- Networking solutions for cloud computing
- Scalable application design
- Core IP routing/MPLS, IPv6, VPN

High-Level Presentation Roadmap

- What is SDN?
- What is OpenFlow?
- Security aspects of controller-based networks
- Real-life controller-based security solutions

Software Defined Networking

What is SDN?

SDN is the physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices

Open Networking Foundation

Let's call whatever we can ship today SDN

Vendor X

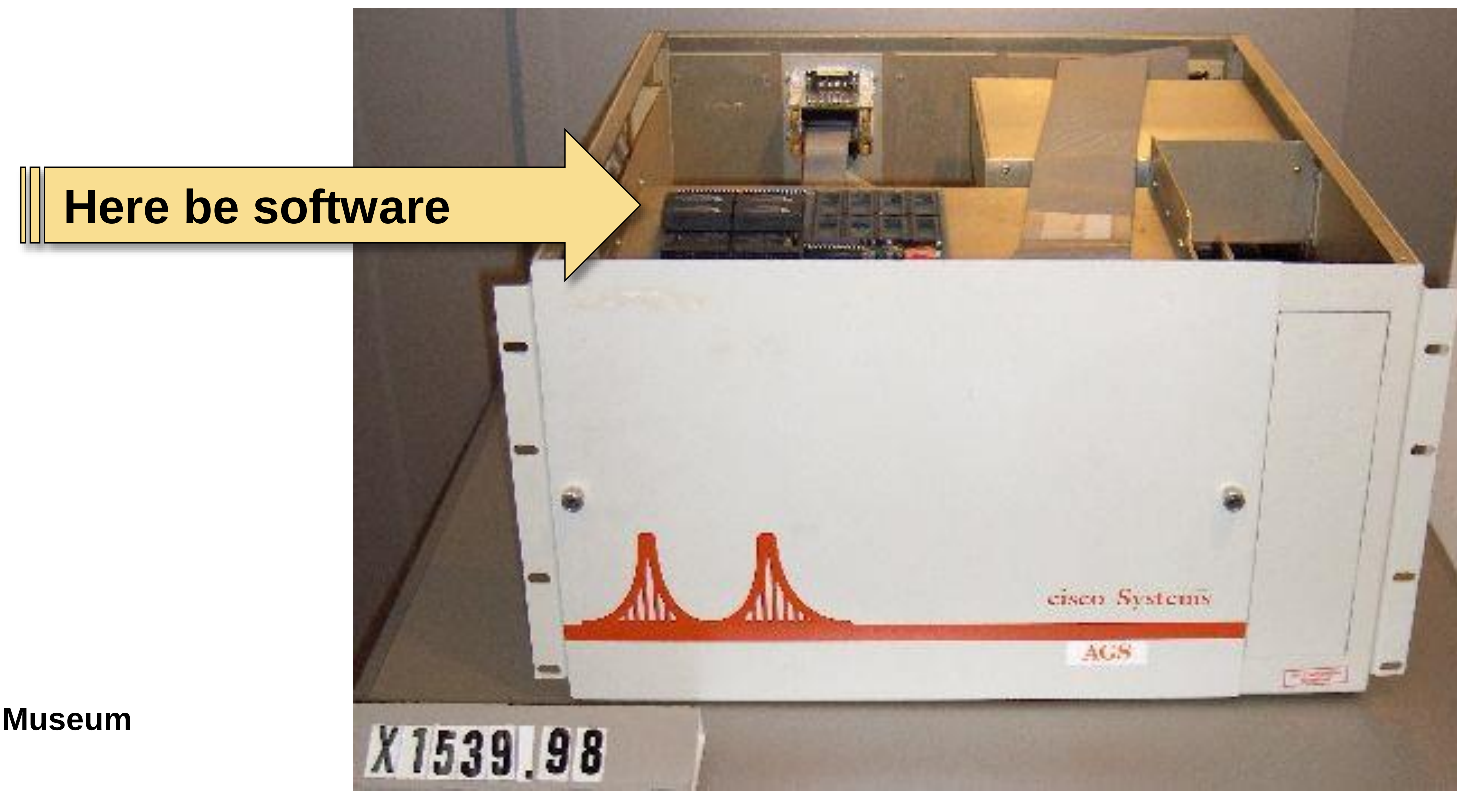
SDN is the magic buzzword that will bring us VC funding

Startup Y

Is the ONF definition too restrictive? Shall we limit SDN to their understanding of it?

Software-Defined? And What Were We Doing?

- Most networking devices have software
- Device behavior was always defined by its software
- Is it all hype … or just marketing gone bad?



Cisco AGS at Computer History Museum
Source: Evilrouters.Net

Motivations Behind SDN Movement

Very large cloud providers (ONF founders):

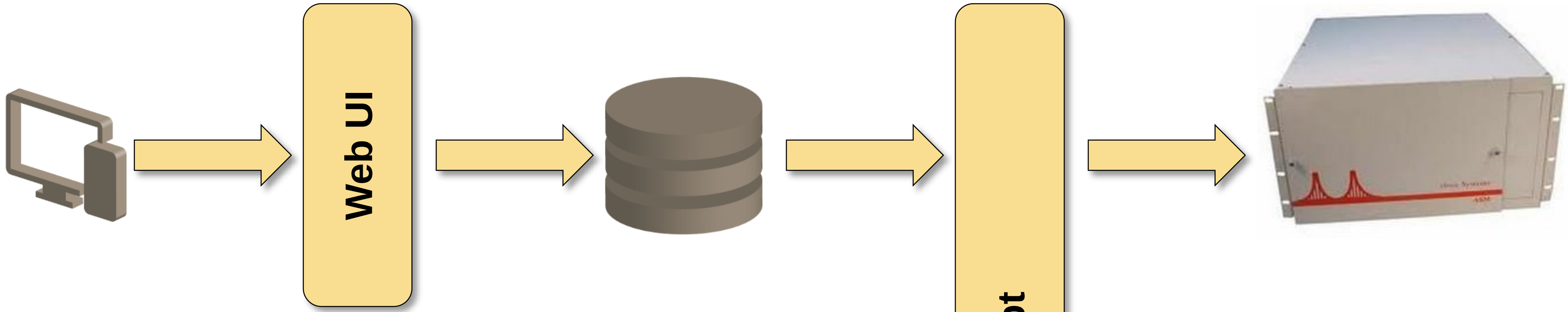
- Give me cheap hardware, I will build my software (Google)
- Implement my own features or protocols (Yahoo)
- Whitebox hardware + open-source software (Facebook)

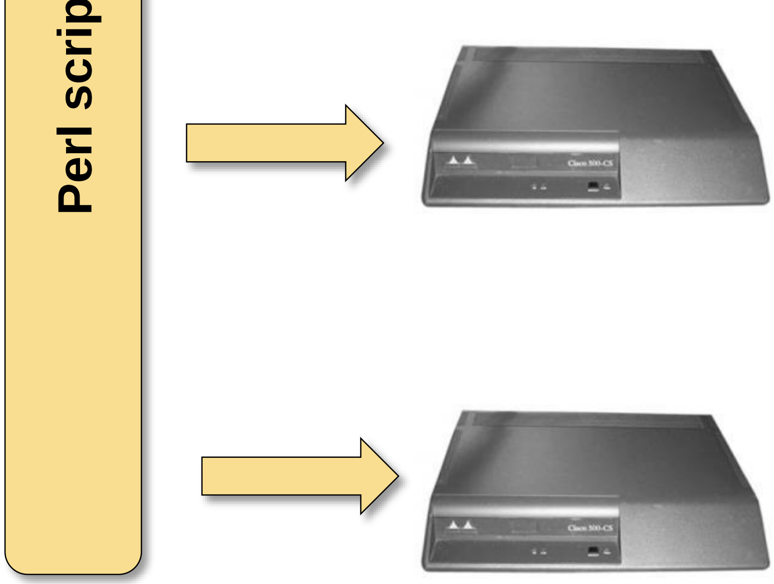
Real-life requirements

- Faster software development
- Programmable network elements
- Faster provisioning
- Centralized intelligence, visibility and policies

The second set of requirements makes more sense for most customers

Did We Have SDN in 1992?





We stopped programming the networks when they became mission critical

- Lack of programming skills
- Lack of reliable automation tools and programmatic interfaces
- Lack of (semi)standardized configuration schema
- Lack of trust

Why have we stopped doing it? What went wrong?

SDN Advantages / Perfect Use Cases

Solving hard problems that require centralized view or synchronization

Things we do well:

- Destination-only hop-by-hop L3 forwarding

Things we don't do so well:

- Large-scale provisioning or orchestration
- Synchronized distributed policies (security, QoS ...)
- Optimal traffic engineering (MPLS-TE) – the knapsack problem
- Routing of elephant flows

Things we don't do at all:

- QoS- or load-based forwarding adaptations
- L3/L4-based or source+destination-based forwarding (policy-based routing)
- Insertion of security features in the forwarding path

**More @ http://blog.ioshints.info/2011/11/openflow-enterprise-use-cases.html ,
http://networkheresy.wordpress.com/2011/11/17/is-openflowsdn-good-at-forwarding/**

Best approach: combine SDN with traditional mechanisms

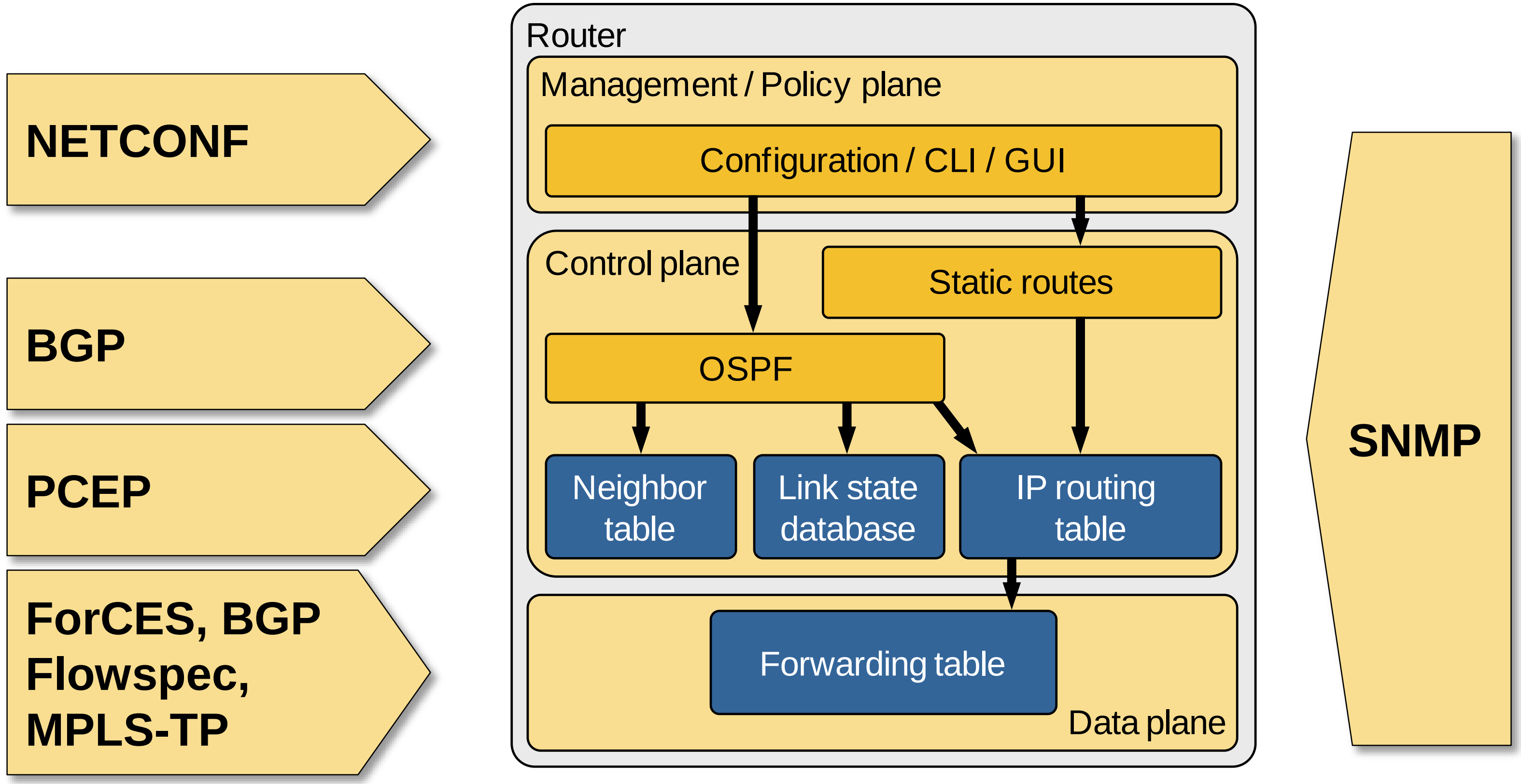
SDN Principles Revisited

- Centralized controllers
- Decisions made based on end-to-end visibility
- Automatic programming or configuration of network devices

Usual objections

- How is this different from Single-Pane-of-Glass?
- What happens when network partitions?
- Why should it work this time?

Can We Do SDN Today?



- Vendor APIs: Cisco, Juniper
- Scripting: Cisco, Juniper, Arista, Dell, F5 ...

What We Can and Cannot Do with Existing Protocols

Easy to do

- Programmatic device configuration (management plane interactions)
- IP forwarding table modifications (BGP or Flowspec)
- Simple edge policy enforcement (per-user ACLs)

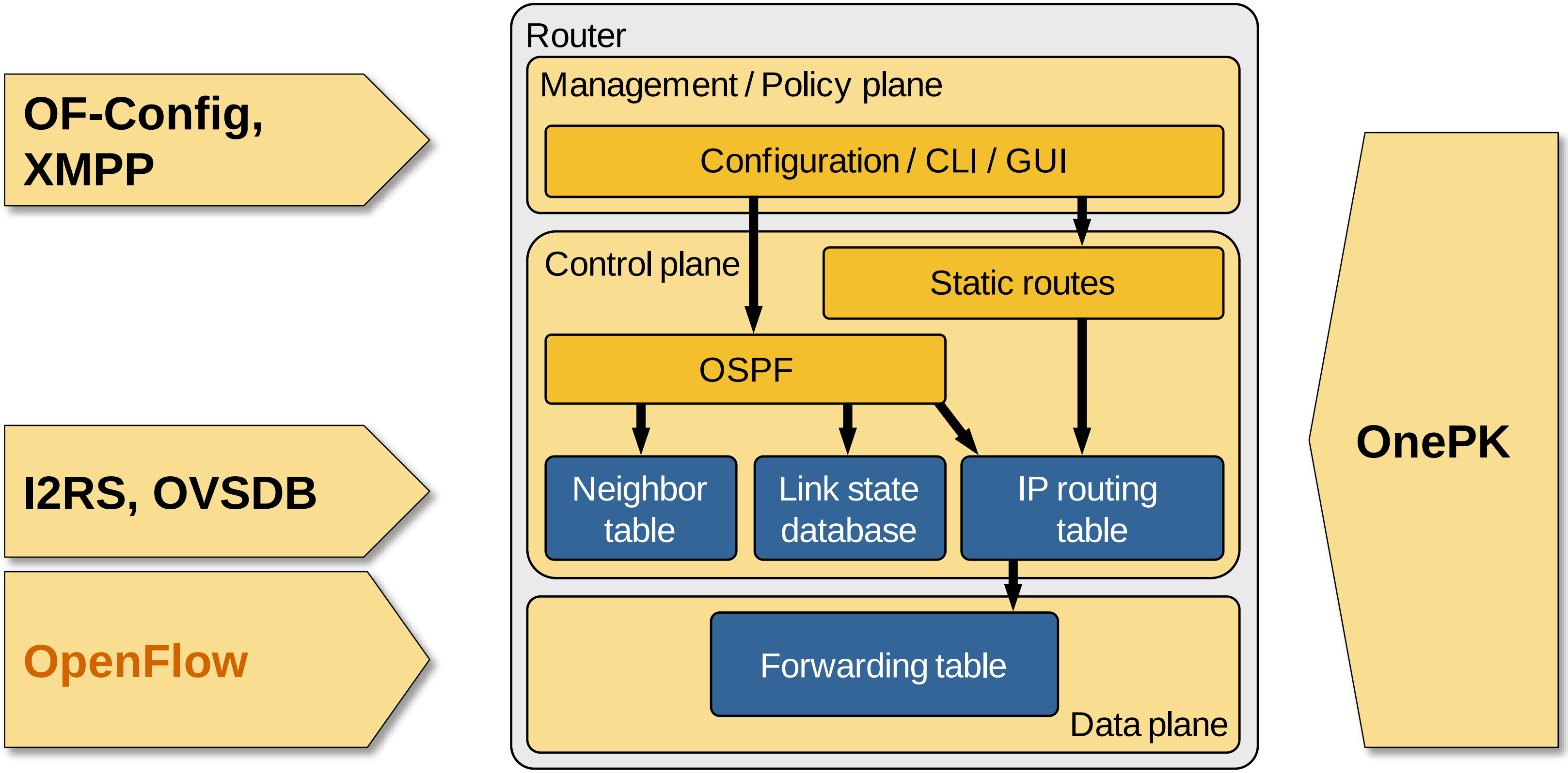
Harder to do

- Topology discovery and extraction
- Non-standard forwarding models (example: source-based IP routing)
- Multi-vendor solutions (example: no standard YANG models, vendor-specific RADIUS attributes)
- End-to-end transactional consistency

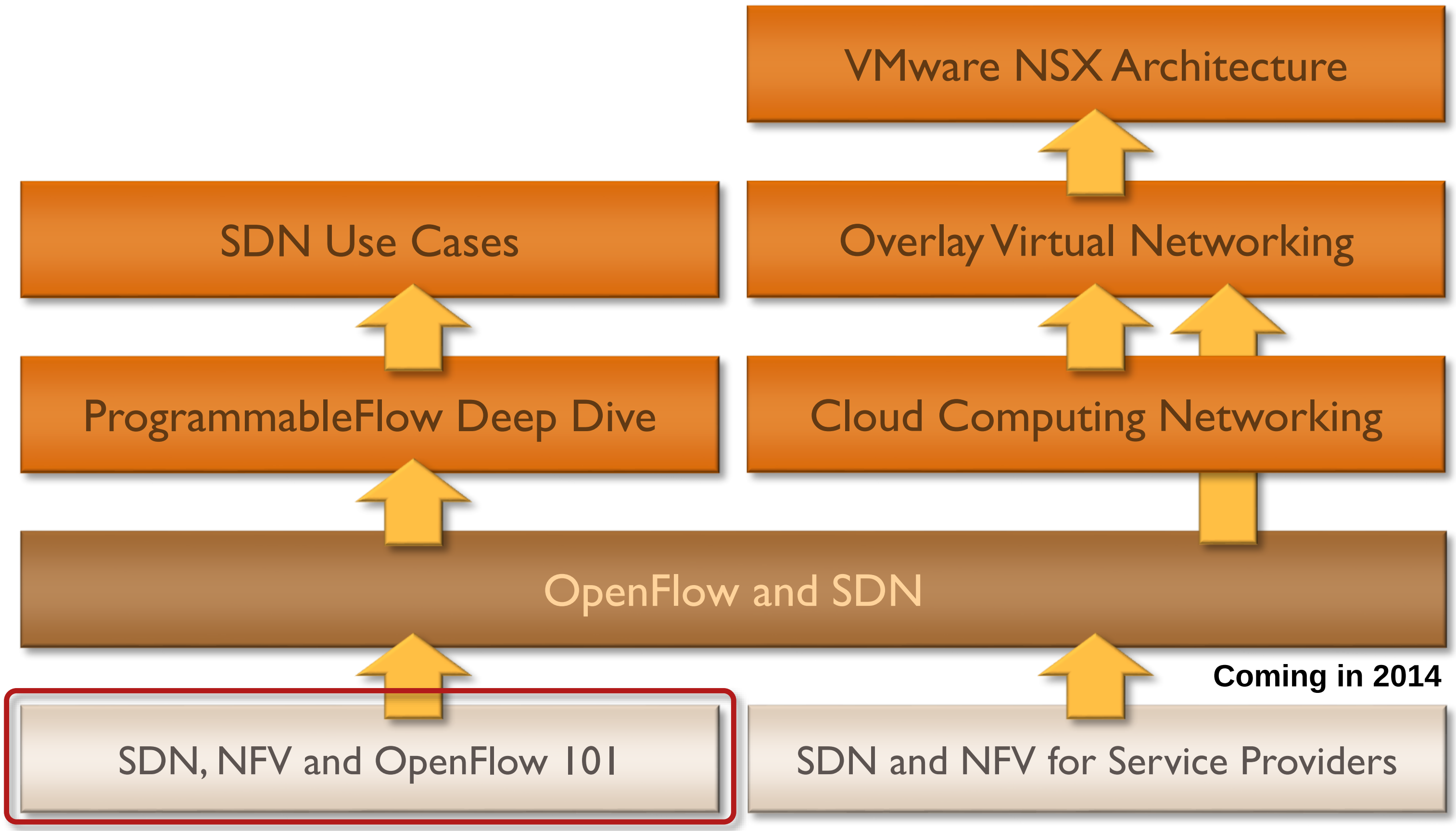
Impossible to do

- New control-plane protocols
- Modification of existing control-plane protocol behavior

Emerging Protocols



OpenFlow and SDN Webinars on ipSpace.net



Trainings

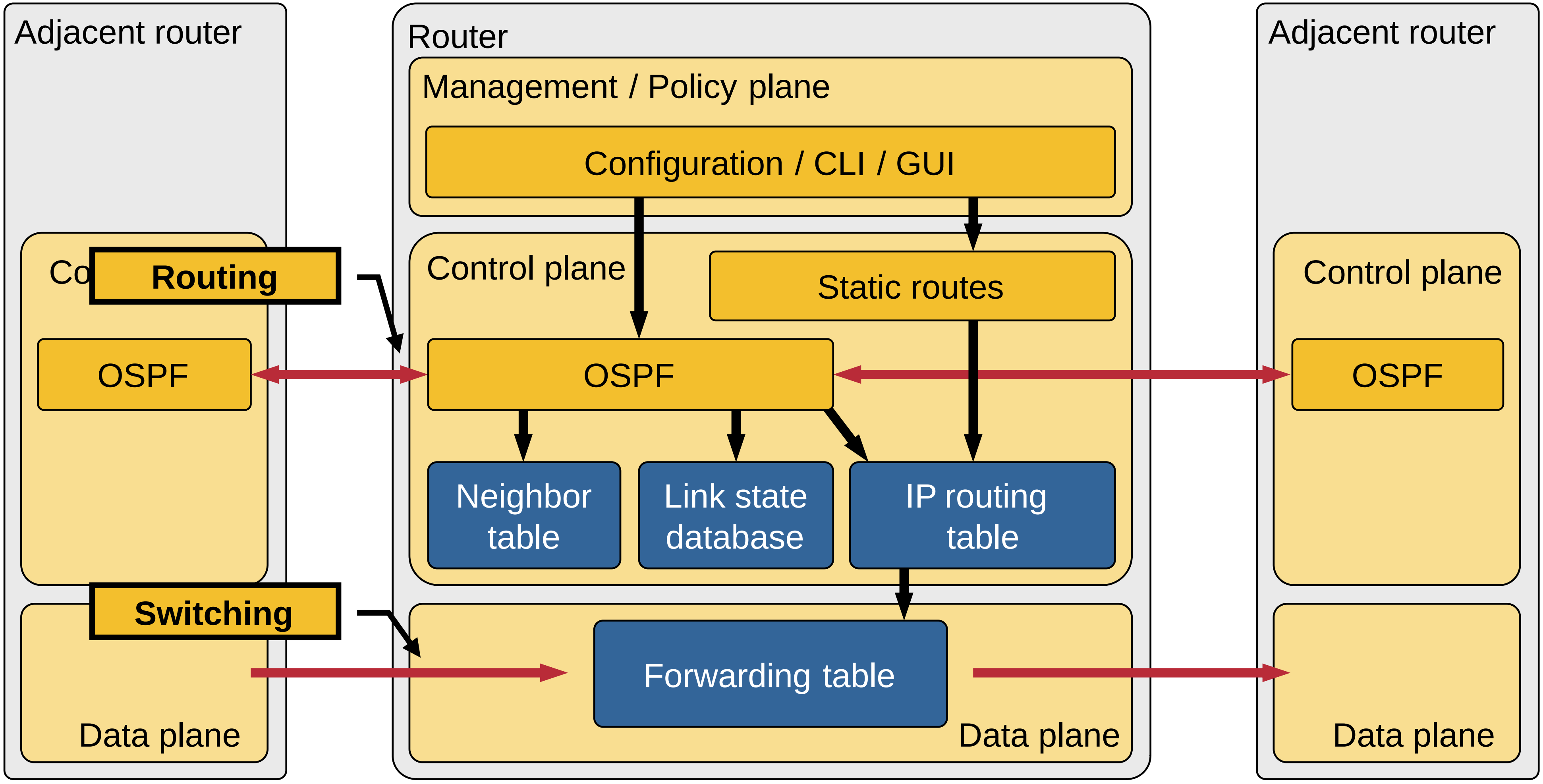
- Live sessions
- On-Site workshops
- Recordings and subscriptions

Other resources

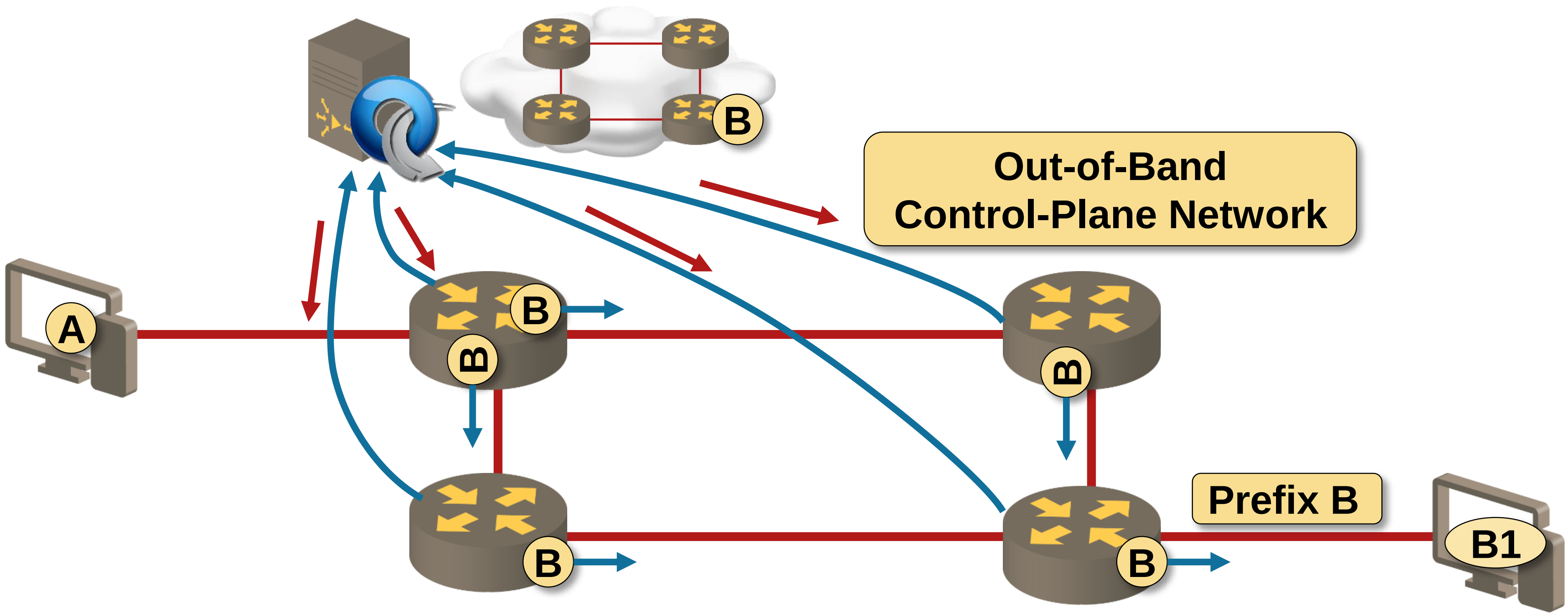
- Consulting
- Books and case studies

OpenFlow 101

Management, Control and Data Planes



OpenFlow = Control / Data Plane Separation



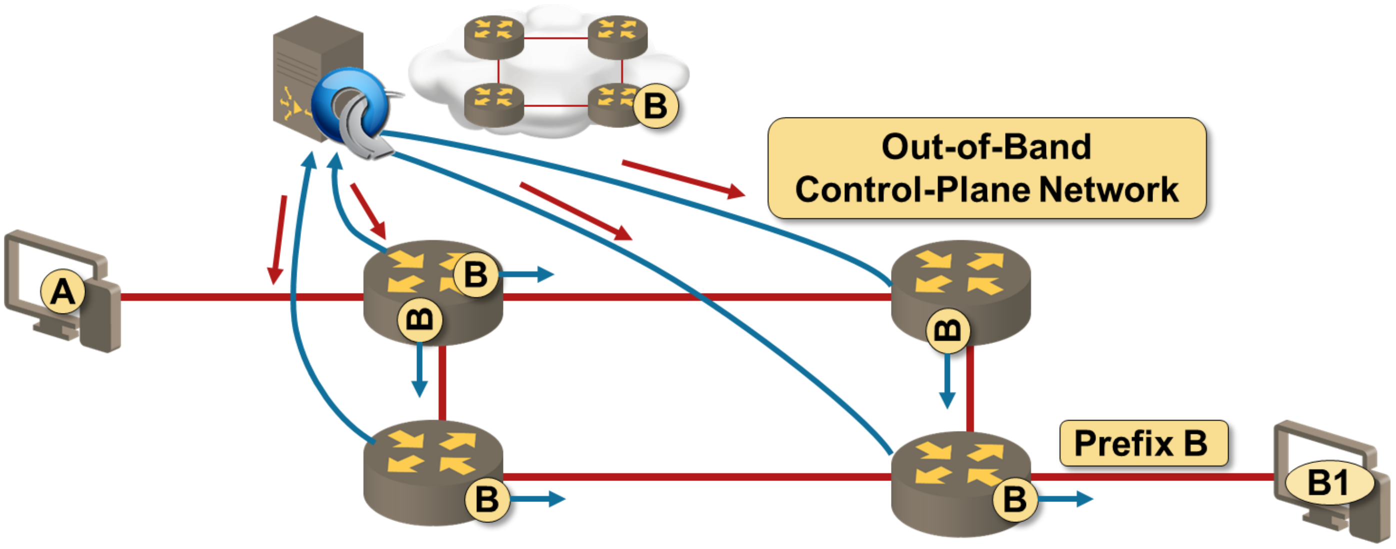
Basic principles:

- Control / Management plane in a dedicated *controller*
- Networking devices perform forwarding and maintenance functions
- IP / SSL connectivity between controller and OpenFlow switch
- OpenFlow = Forwarding table (TCAM) download protocol

Review: OpenFlow Protocol Details

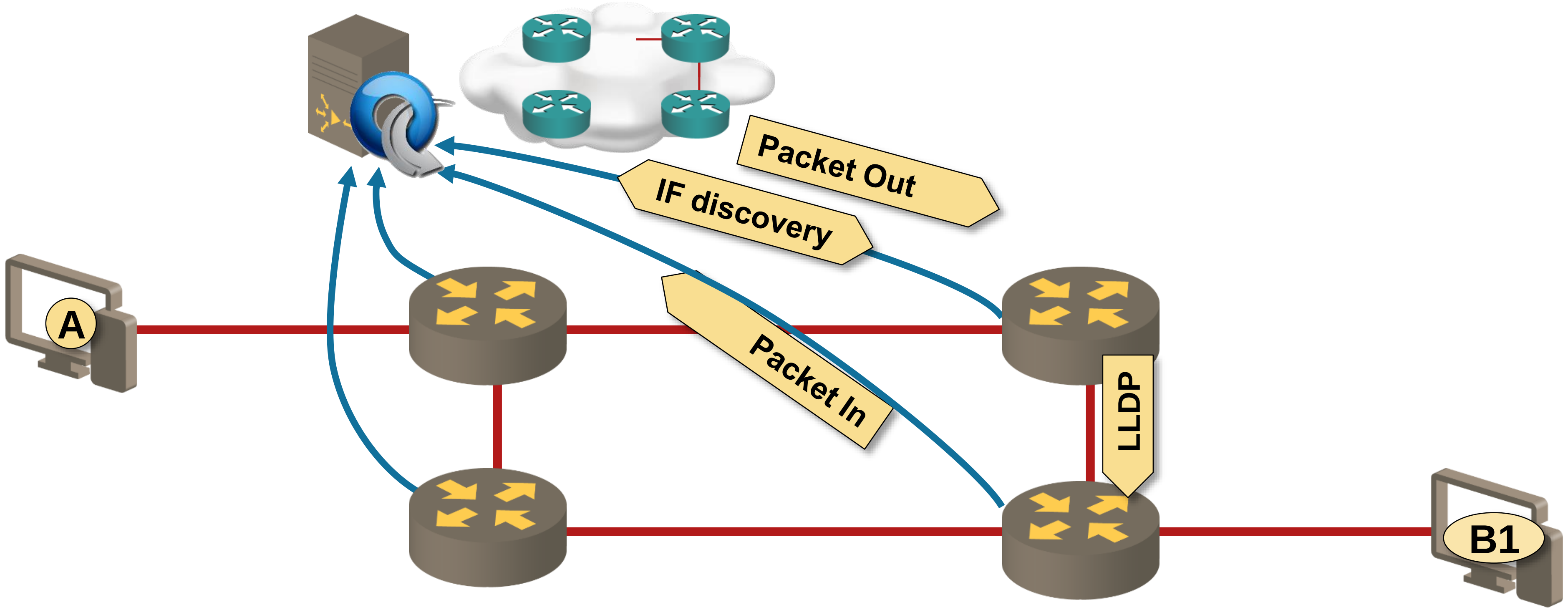
Message types:

- Configuration
- Feature requests
- Flow/Port/Table modifications ➔ install forwarding entries
- Statistics ➔ read per-forwarding-entry packet/byte counts
- Barriers (~ transactions)
- Packet In/Out ➔ control-plane protocols and packet punting



Hint: forwarding entry does not have to be a single session flow

Case Study: OpenFlow Topology Discovery



- Controller builds the network model as devices connect to it
- OpenFlow control packets used for interface
- *Packet Out* message used to send a packet through an interface
- *Packet In* message used by the switch when it receives unknown packet

OpenFlow Concepts Are not New (RFC 1925, sect 2.11)

Do you still remember ...

- Frame Relay and ATM networks
- SONET/SDH
- ForCES
- MPLS-TP

The problems are always the same:

- Forwarding state abstraction / scalability
- Distributed network resilience with centralized control plane
- Fast feedback loops
- Fast convergence (FRR, PIC)
- Linecard protocols (BFD, LACP, LLDP ...)

The important difference this time: customer pressure

OpenFlow Deployment Models

Native OpenFlow

- Works well at the edge (single set of uplinks)
- Too many complications at the core (OOB management, fast failure detection ...)

OpenFlow with vendor-specific extensions

- Link bundling
- Load balancing
- Linecard functionality (LLDP, LACP, BFD ...)
- QoS

Ships in the night

- OpenFlow in parallel with traditional forwarding
- Some ports / VLANs dedicated to OpenFlow
- Fallback from OpenFlow to *normal*
- Solves OOB management and linecard functionality

Integrated

- OpenFlow classifiers/actions become part of regular packet processing
- OpenFlow provides *ephemeral state* configuration

More @ http://blog.ioshints.info/2011/11/openflow-deployment-models.html

Shipping OpenFlow Products

Switches – Commercial

- Arista 7500/7150
- Brocade MLX/NetIron products
- Cisco Nexus 3000
- Dell N3000/N400
- Extreme BlackDiamond
- HP ProCurve
- IBM BNT G8264
- Juniper MX & EX9200 (not GA)
- NEC ProgrammableFlow switches
- Smaller vendors (Mikrotik, ODMs)

Switches – Open Source

- Open vSwitch (Xen, KVM)
- NetFPGA reference implementation
- OpenWRT
- Mininet (emulation)

Controllers – Commercial

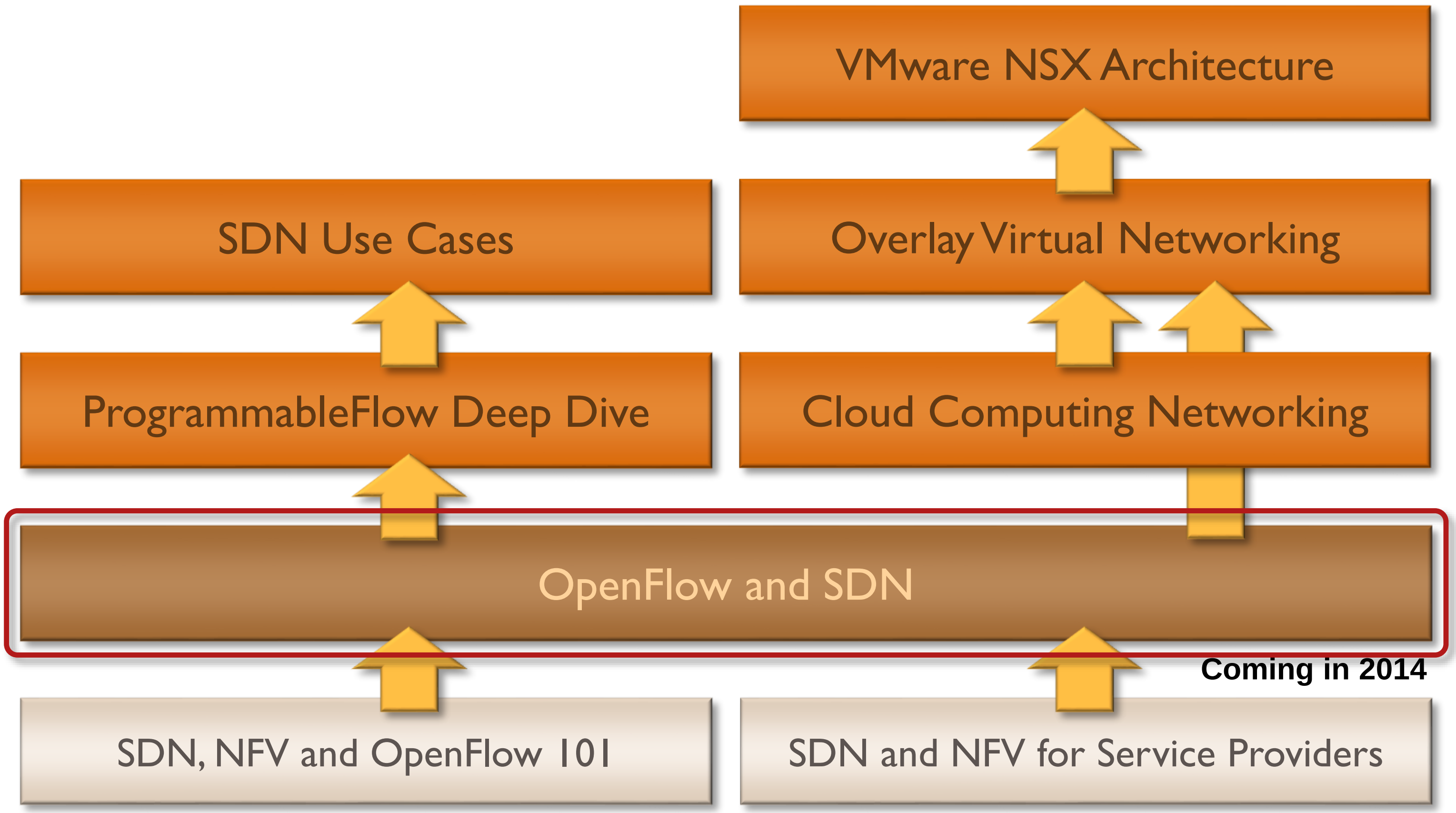
- NEC ProgrammableFlow Controller
- VMware NSX
- Big Switch Networks
- Cisco eXtensible Network Controller
- HP VAN SDN Controller

Controllers – Open Source

- Open Daylight (Java)
- NOX (C++/Python)
- Beacon (Java)
- Floodlight (Java)
- Maestro (Java)
- RouteFlow (NOX, Quagga, ...)
- NodeFlow (JavaScript)
- Trema (Ruby)

More @ http://www.sdncentral.com/shipping-sdn-products/

OpenFlow and SDN Webinars on ipSpace.net



Trainings

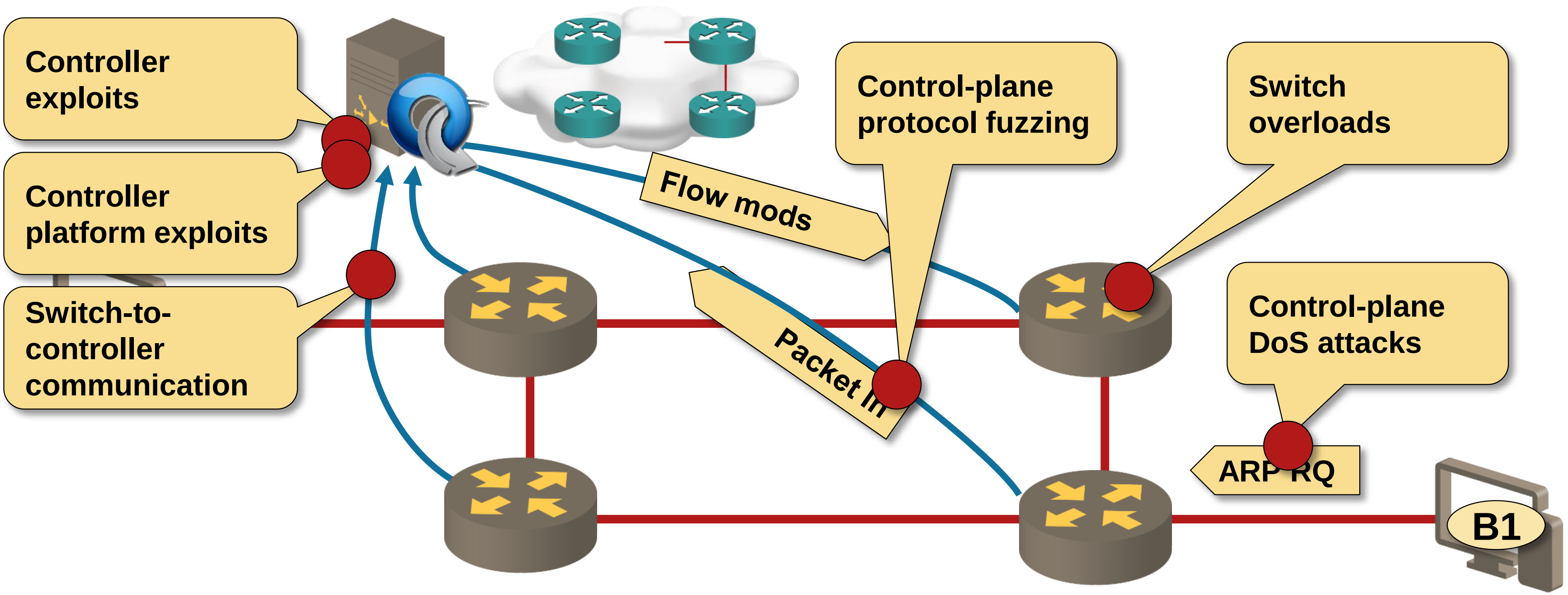
- Live sessions
- On-Site workshops
- Recordings and subscriptions

Other resources

- Consulting
- Books and case studies

SDN Security Challenges

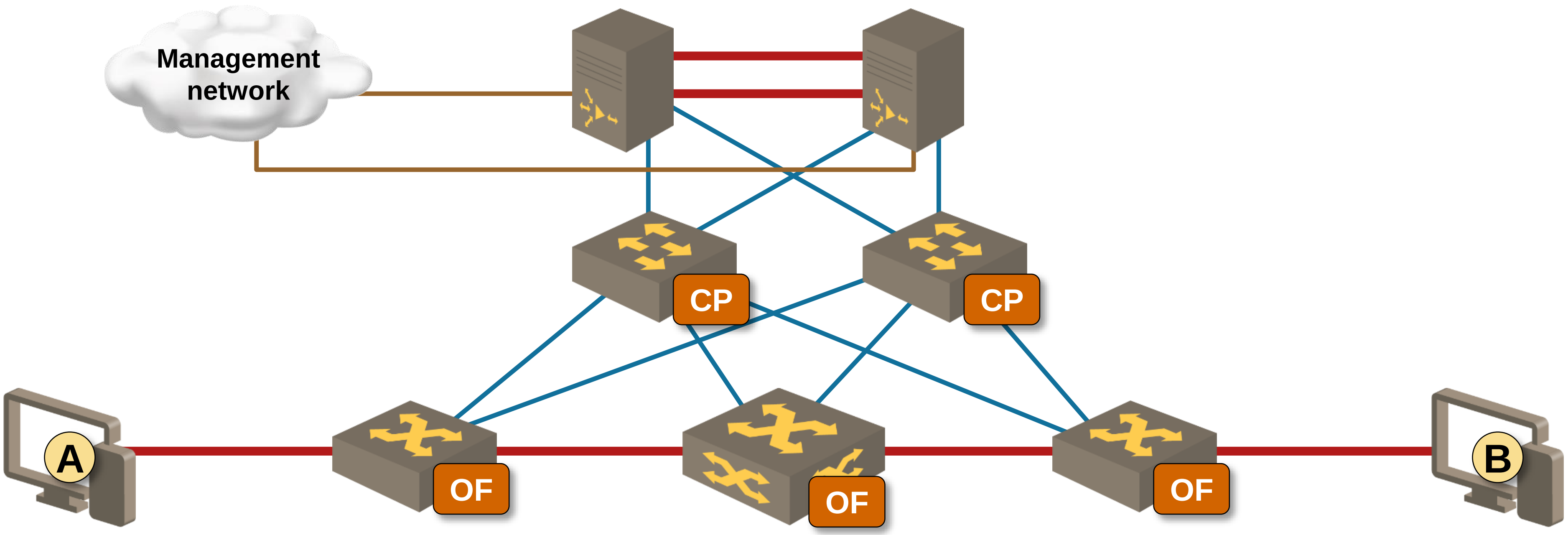
Threat Analysis



- Control-plane attacks
- Denial-of-service attack (switch and controller)
- Fuzzing attacks

SDN controller is a very lucrative target

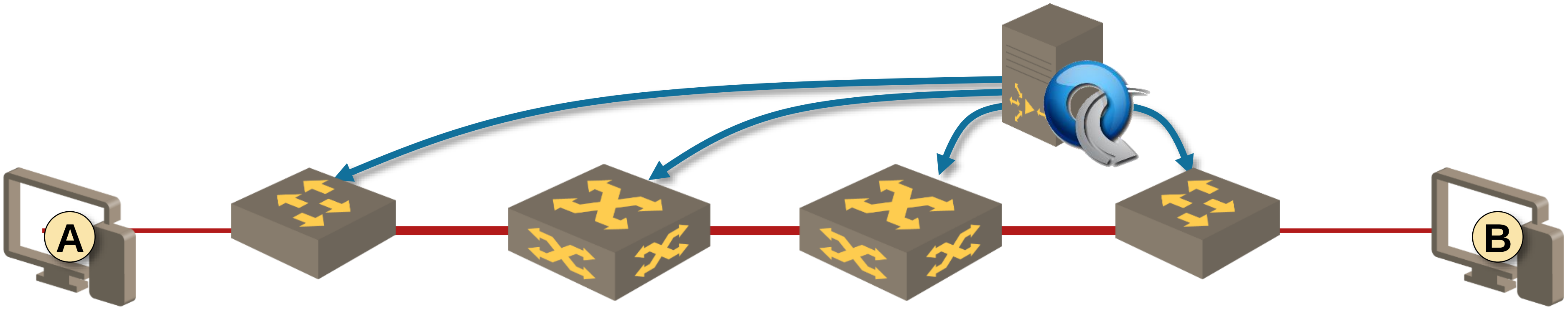
Separate Data Plane



Well-known solutions

- Encrypted switch-to-controller communications
- Separate management- or control-plane network
- Forwarding and management contexts in the switches

Proactive Flow Setups



Reactive flow setups

- Punt unknown packets to the controller
- Compute forwarding paths on demand
- Install flow entries based on actual traffic

Scalability concerns

- Flow granularity
- Packet punting rate
- Flow modification rate

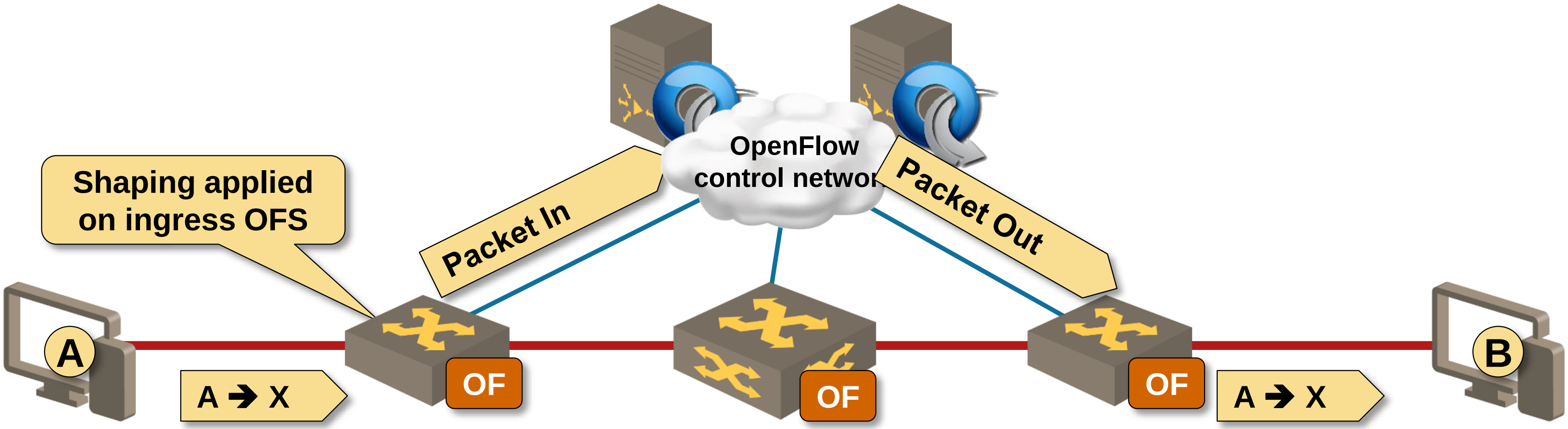
Proactive flow setups

- Discover network topology
- Discover endpoints
- Compute optimal forwarding topology
- Download flow entries

No data plane controller involvement

- Exceptions: ARP and MAC learning

Control-Plane Protection



- OpenFlow switches send all unknown packets to controller
- Unicast flooding performed through controller ➔ control plane interference
- Control-plane protection needed in ingress OpenFlow switches

Hardening an SDN Solution

Common design guidelines

- Out-of-band Control Plane
- Minimize the control-plane involvement
- Use OpenFlow solutions with coarse-grained proactive forwarding model
- Prefer solutions with distributed intelligence

Switch hardening

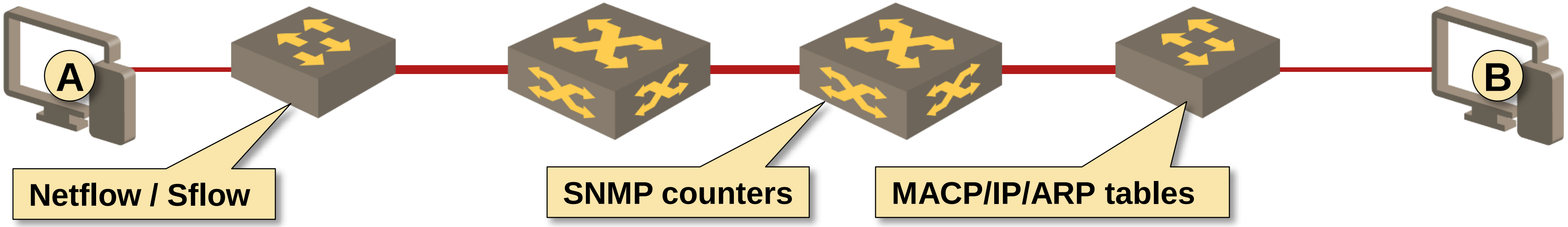
- Strict control/data plane separation
- Control-plane policing in OpenFlow networks

Controller hardening

- Most controllers run on Linux → you know what to do

Use Case: Network Monitoring and Tapping

Network Monitoring in Traditional Networks



Traffic statistics

- Too coarse (interface counters), detailed (Netflow / IPFIX) or sampled (Sflow)
- Limited visibility in multi-tenant environments

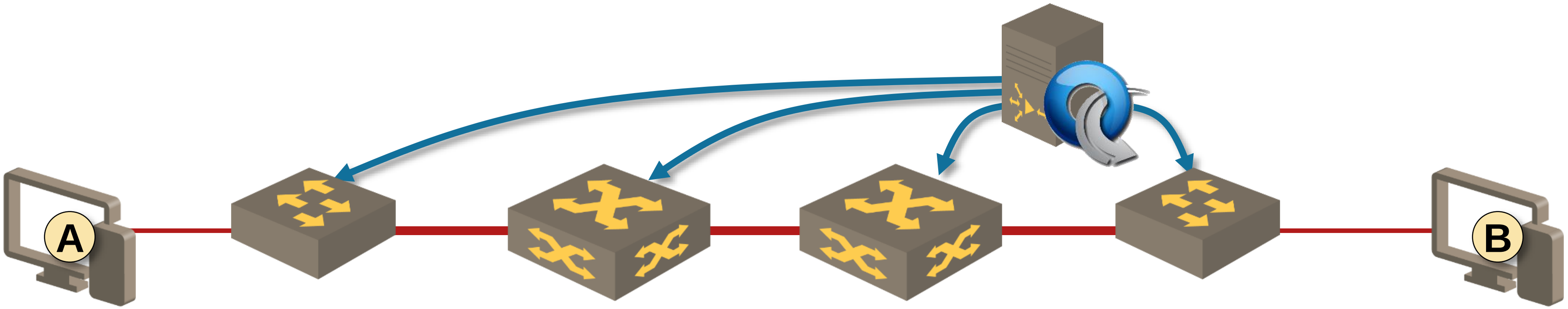
Endpoint visibility

- Available on edge network devices
- Hard to summarize into a searchable format

Forwarding information

- Information distributed across numerous devices (MAC tables, ARP tables, IP forwarding tables)
- Hard to reconstruct expected traffic path

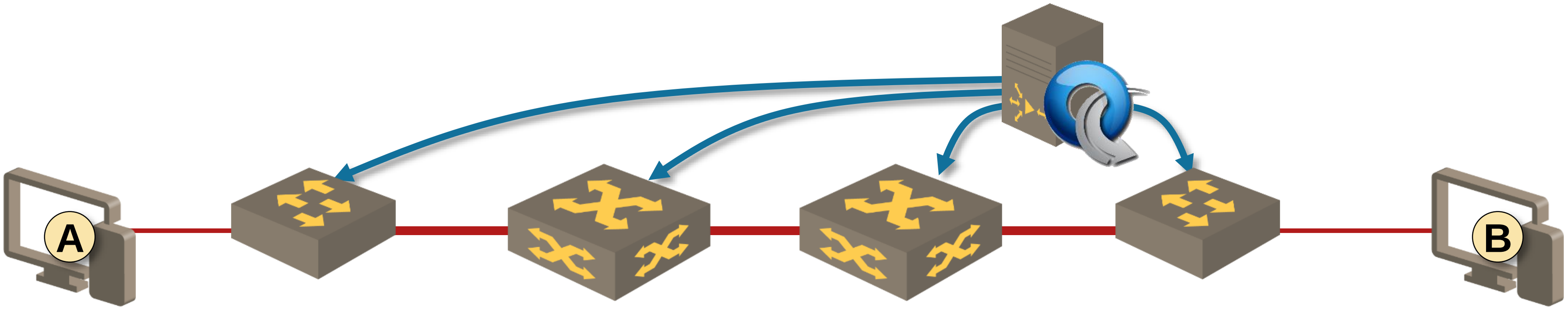
Network Monitoring in Controller-Based Networks



Controller is the authoritative source of information on

- Network configuration
- Network topology
- Forwarding paths
- Endpoints (IP prefixes or IP/MAC addresses)

Network Monitoring in OpenFlow-Based Networks



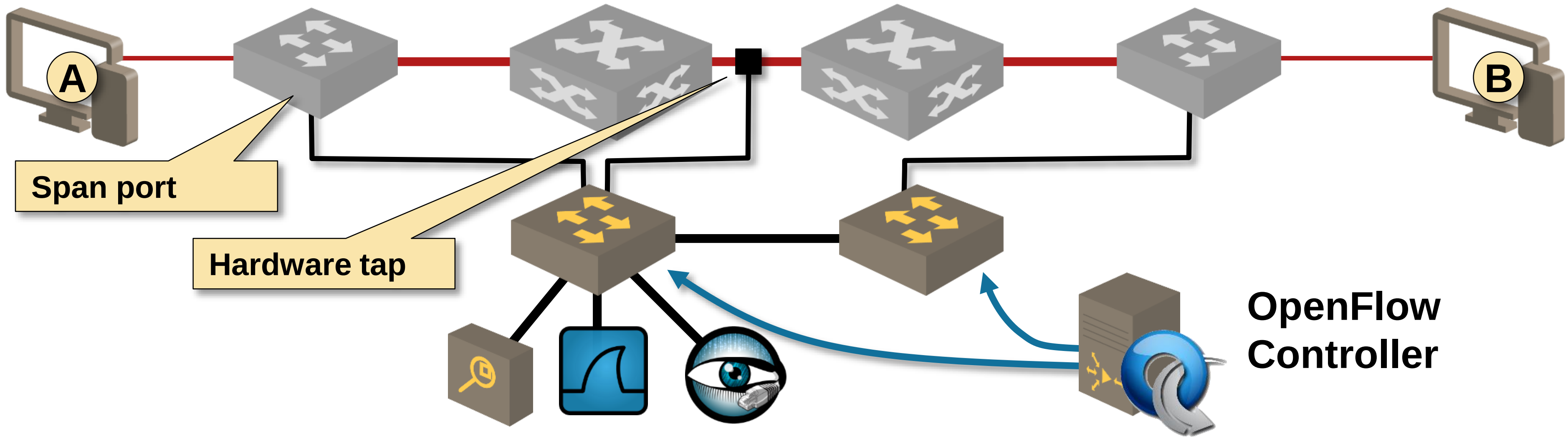
OpenFlow statistics

- Byte- and packet counters associated with every OpenFlow entry
- Controller can read flow statistics (similar to SNMP interface counters)
- Flow counters reported to OpenFlow controller every time a switch removes a flow due to idle timeout

Traffic statistics in OpenFlow controller

- Controller can collect traffic statistics at any granularity ➔ configured with flow entries downloaded to the switches
- Constraint: switch hardware or software limits

OpenFlow/SDN in Tap Aggregation Network



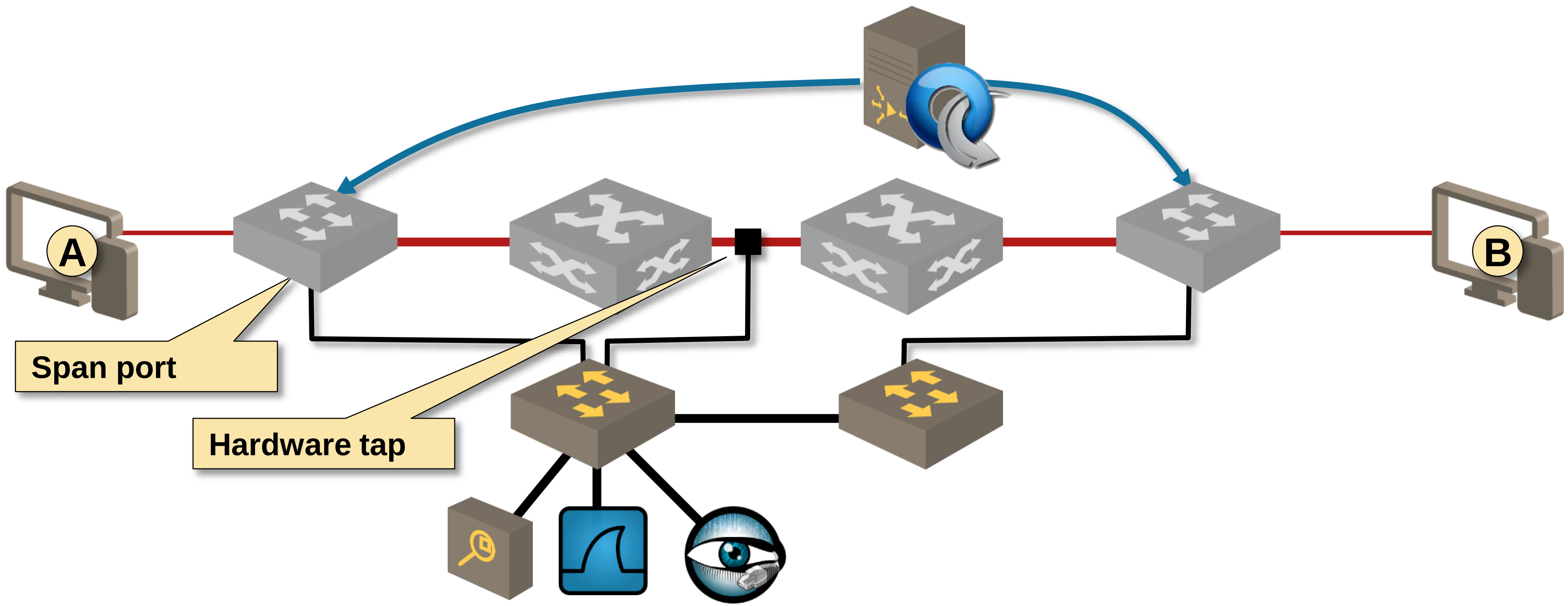
Solution Overview

- Replace dedicated tap aggregation equipment with standard OpenFlow-capable switches
- Program filtering and forwarding rules with OpenFlow

Benefits of OpenFlow

- Based on commodity switches
- Filter early in the forwarding path ➔ use capturing devices more efficiently
- N-tuple filtering
- Flow-based metering
- Simple tap- and filter changes

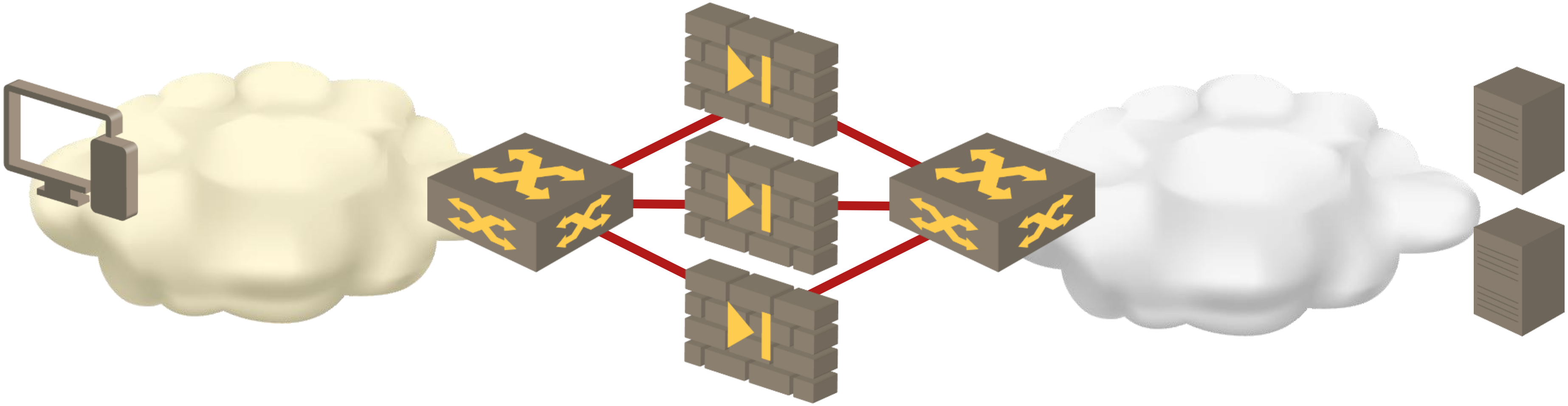
Traffic Tapping with OpenFlow Switches



- Use OpenFlow flows to mirror traffic to SPAN ports
- Higher traffic redirection granularity ➔ lower number of SPAN ports required
- Any OpenFlow controller capable of inserting individual flows could be used

Use Case: Scale-Out Network Services

Scale-Out Stateful Services Don't Scale Well



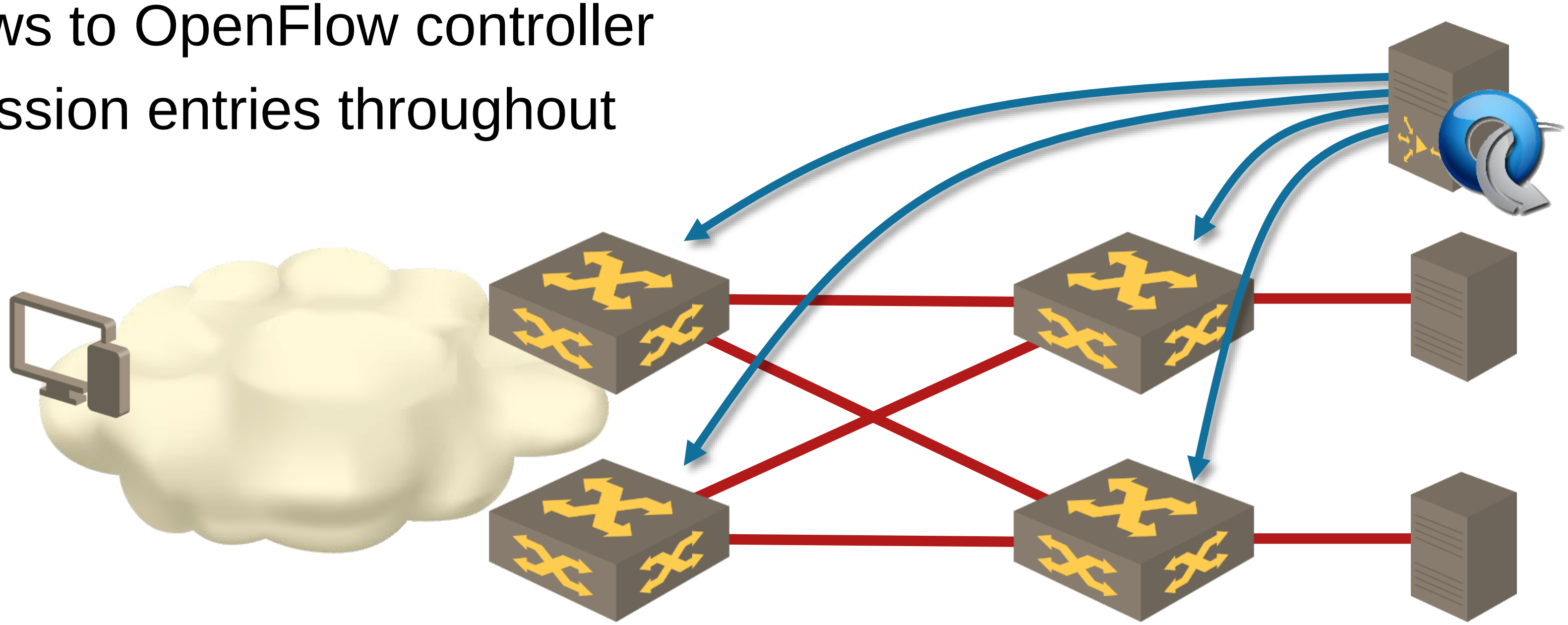
Stateful services are hard to scale out:

- Forward and reverse traffic flows might not match
- Traffic distribution might change with introduction or removal of devices
- Switches might dynamically rehash ECMP entries
- Stateful devices have to exchange state and/or user traffic

Result: scale-out performance is not linear

Simplistic OpenFlow-Based Stateful Services

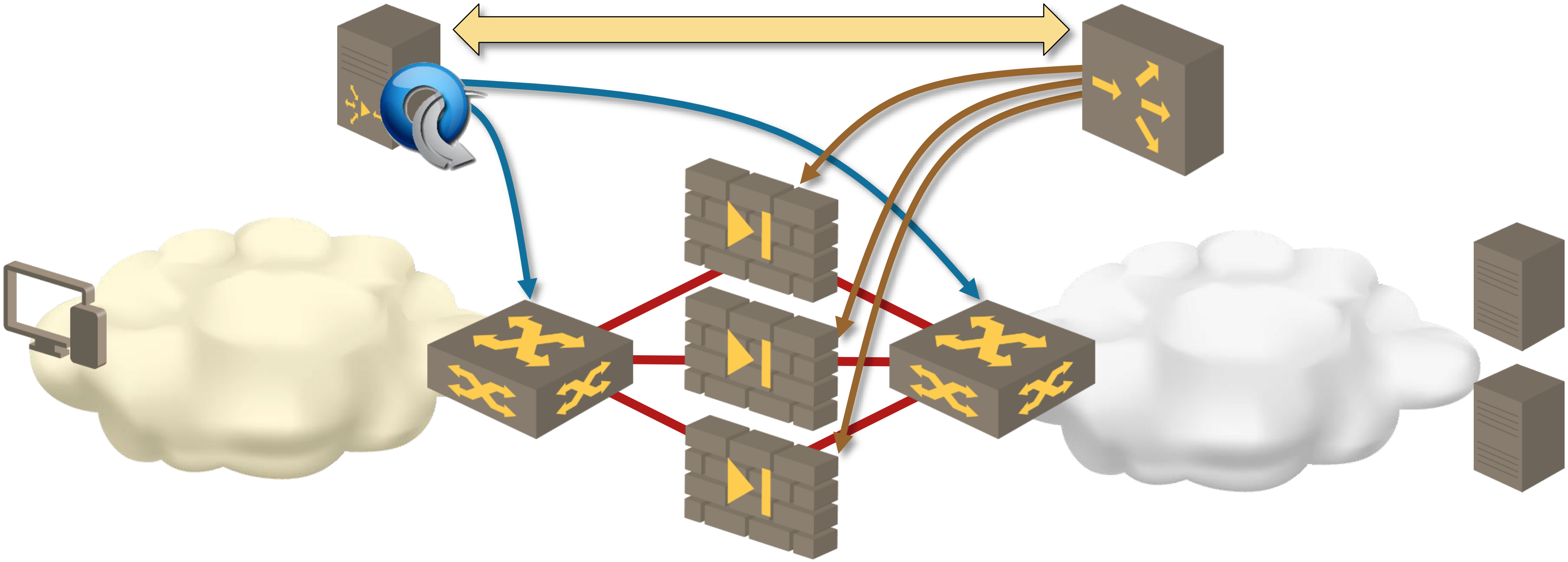
- Punt new flows to OpenFlow controller
- Install per-session entries throughout the network



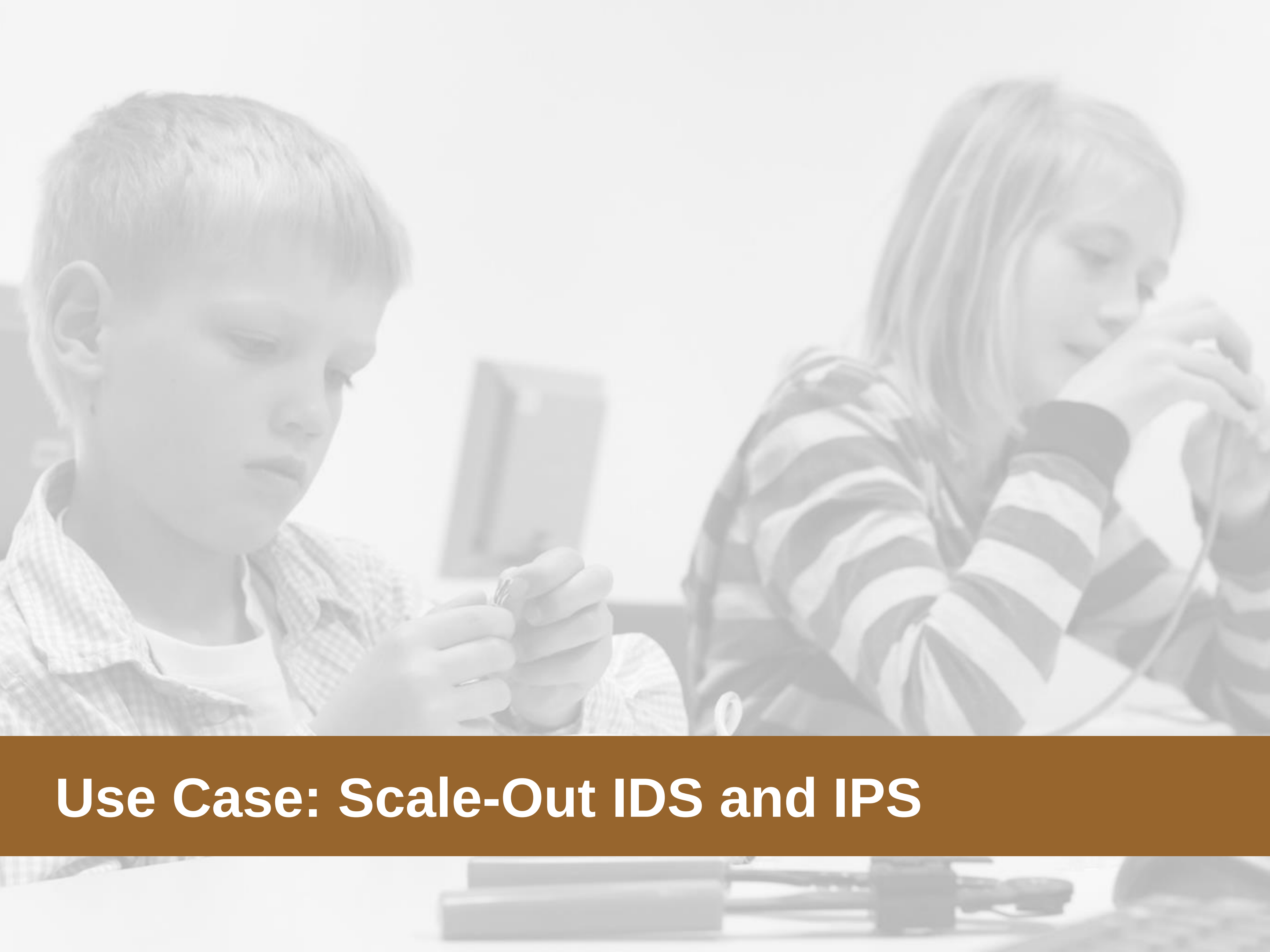
Scalability challenges

- Number of flows in hardware switches
- Control channel bandwidth
- Flow modification (installation/removal) rate – orders of magnitude too low for GE/10GE environment with reasonable traffic mix

OpenFlow-Assisted Scale-Out Services

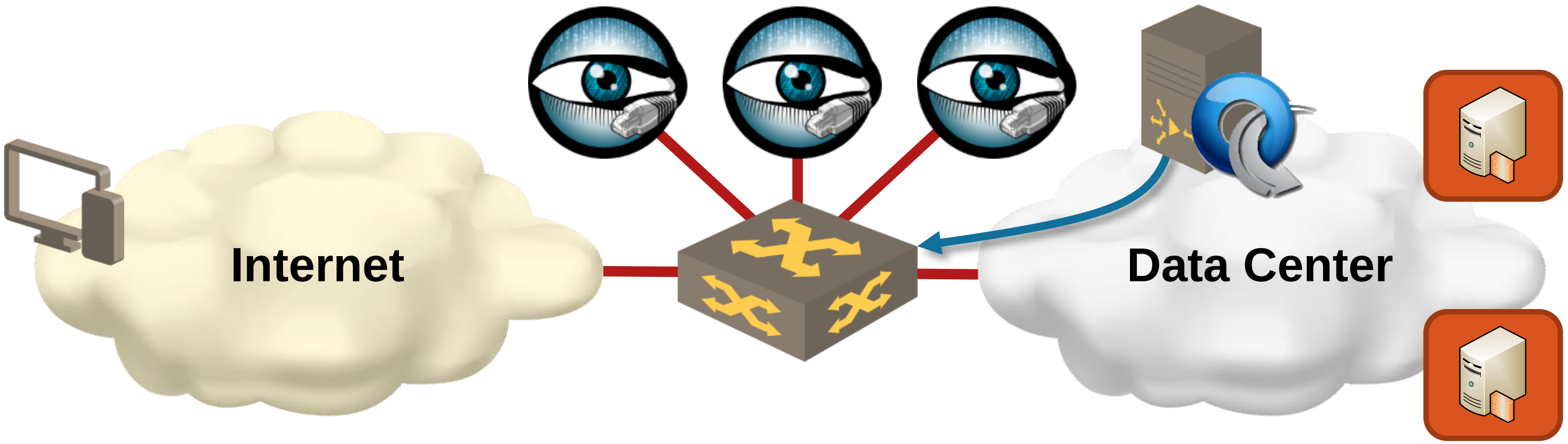


- OpenFlow switches use coarse-grained flows (based on source/destination IP address ranges)
- Load balancing appliances perform fine-grained load balancing
- Load balancing controller modifies OpenFlow flows (and traffic distribution) based on node availability and traffic load



Use Case: Scale-Out IDS and IPS

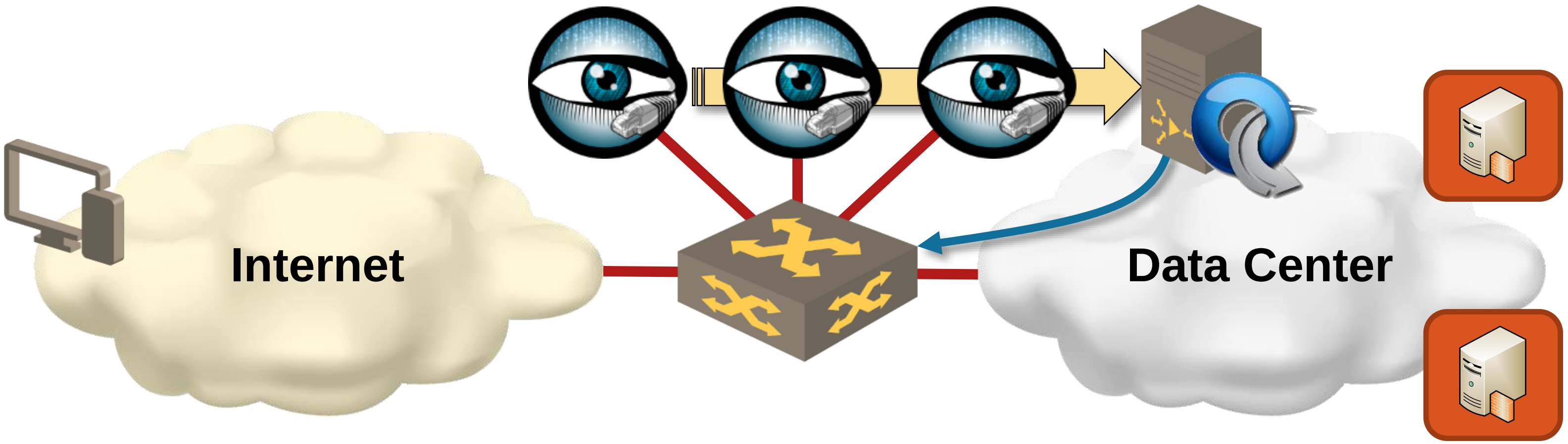
Scale Out IDS with OpenFlow Controller



OpenFlow used to distribute the load to multiple IDS appliances

- Coarse-grained flows deployed on the OpenFlow switch
- Flow granularity adjusted in real time to respond to changes in traffic
- Each appliance receives all traffic from a set of endpoints (complete session and endpoint behavior visibility)

Scale Out IPS with OpenFlow Flows



DoS detection system reports offending X-tuples

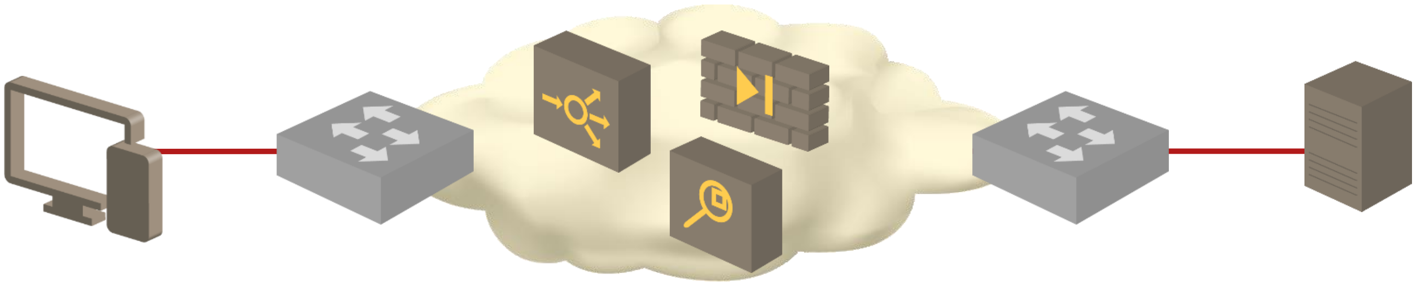
- Source IP addresses
- Targeted servers
- Applications (port numbers)

OpenFlow controller installs *drop* flows

Module for Bro IDS already available

Use Case: Service Insertion

Service Insertion 101



Service insertion principles

- Dynamically insert network services in the forwarding path
- Based on endpoints (users), applications, or both

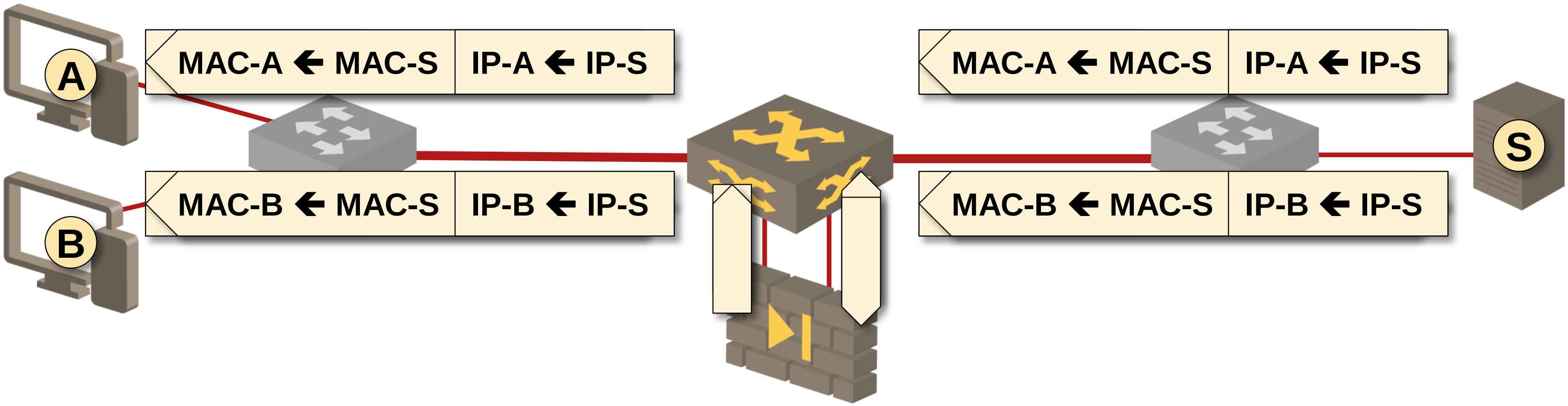
Data center use cases

- Firewalls, load balancers, IPS/IDS appliances

Enterprise and service provider use cases

- Traffic filters (firewalls or packet filters)
- Captive portals
- WAN acceleration and caching

Layer-2 Service Insertion



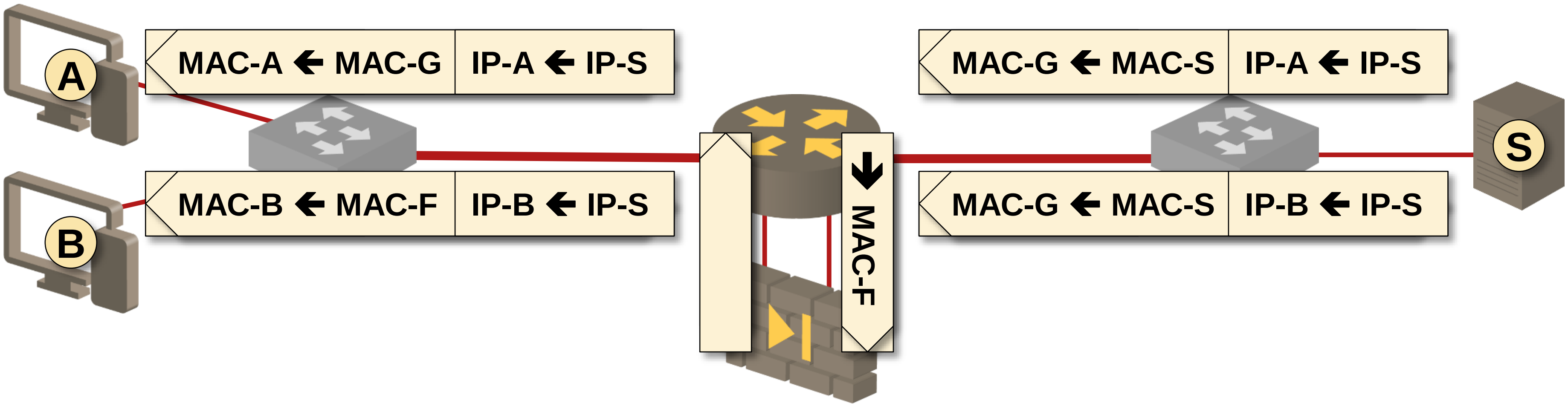
Layer-2 frames redirected to a transparent (bump-in-wire) appliance

- Based on MAC (potentially IP) headers

Typical implementation

- VLAN chaining
- Hard to implement for individual endpoints
- Impossible to implement for individual applications
- Fantastic potential for forwarding loops

Layer-3 Service Insertion



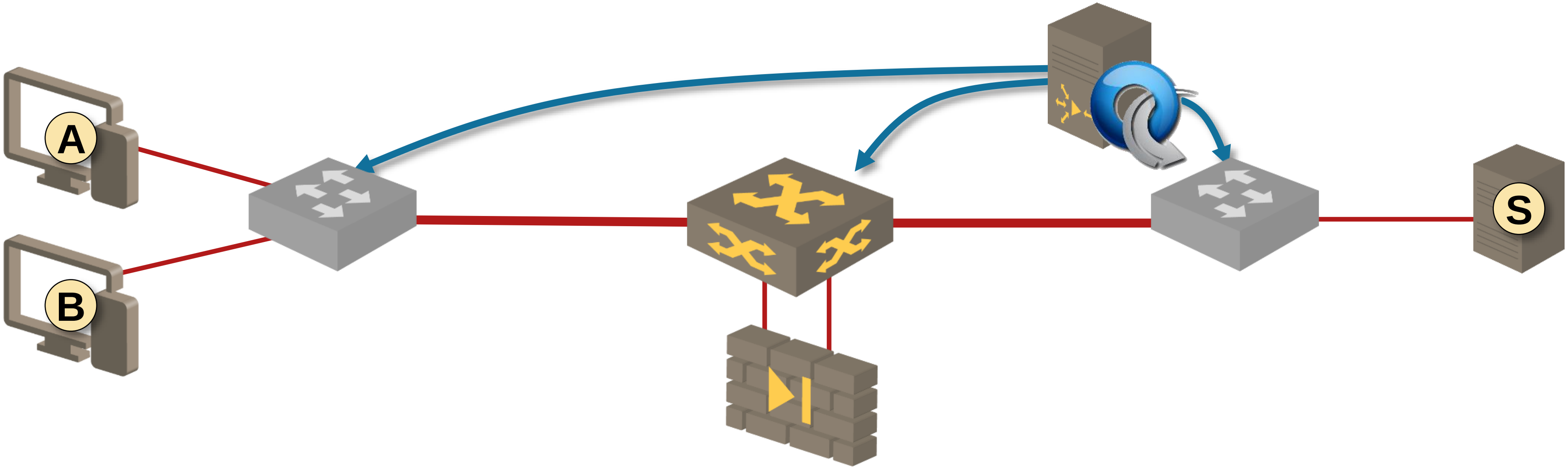
Layer-3 frames redirected to a transparent or inter-subnet appliance

- Based on IP headers
- Might require MAC header rewrite

Typical implementation

- Policy-based routing (PBR)
- MAC rewrite is automatic
- Hard to implement for appliances not close to the forwarding path

Service Insertion in OpenFlow-Based Fabrics



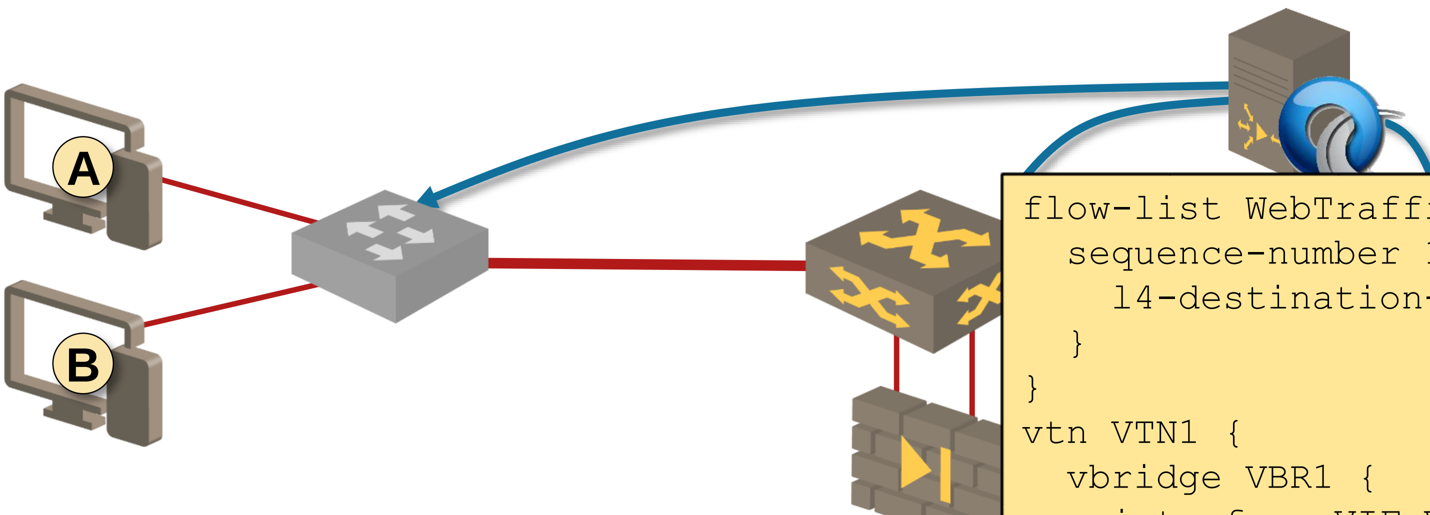
Simplistic approach

- Install redirection flow entries wherever needed
- Change flow redirection entries on every topology change

Scalable approach

- Create new forwarding paths between edge switches and appliances
- Map traffic to forwarding paths in ingress switches

ProgrammableFlow Service Insertion Example

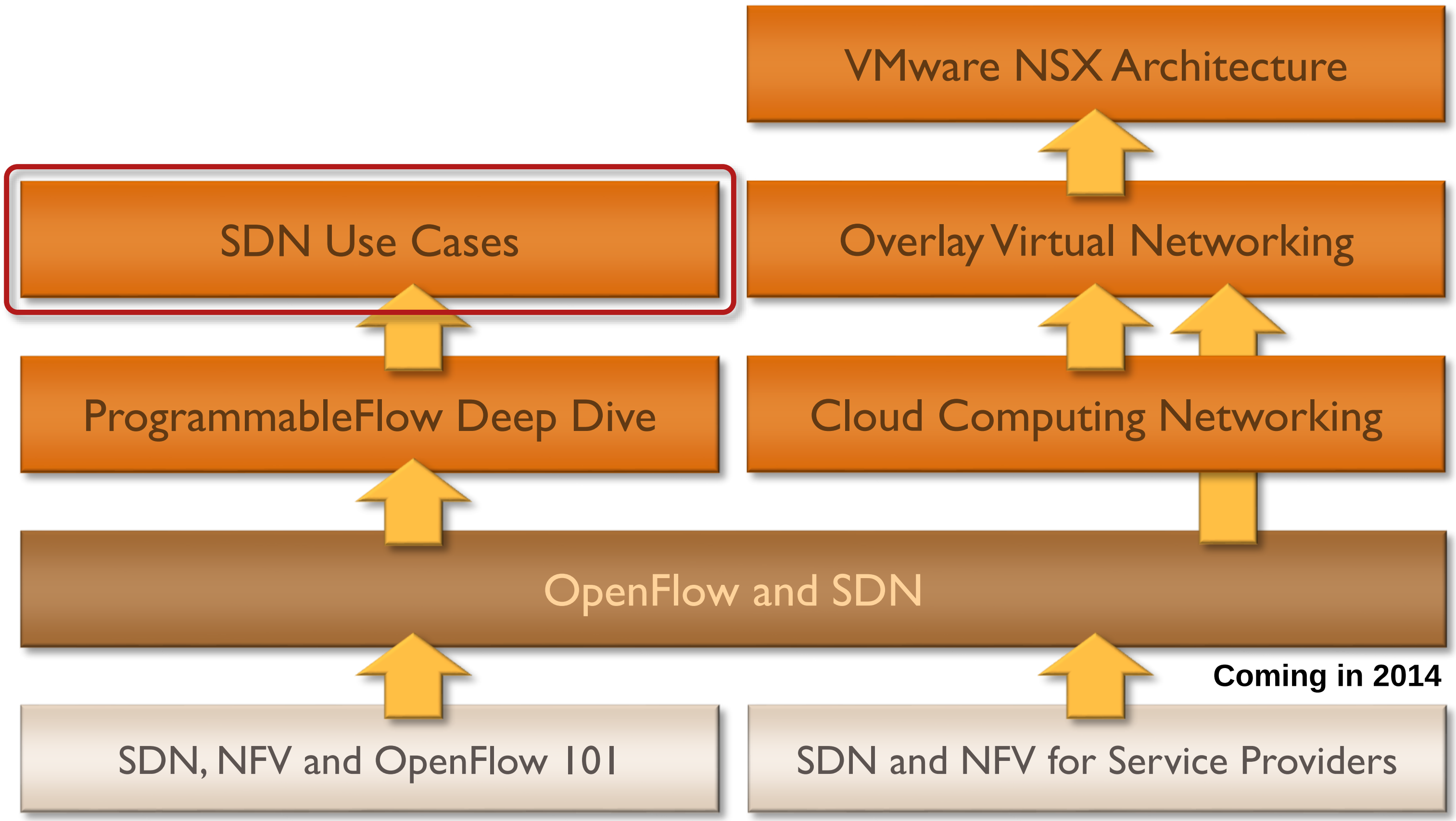



```
flow-list WebTraffic {
  sequence-number 10 {
    l4-destination-port 80
  }
}
vtn VTN1 {
  vbridge VBR1 {
    interface VIF_VEX1 {
      flow-filter in {
        sequence-number 10 {
          match flow-list WebTraffic
          action redirect
          redirect-destination ...
        }
        sequence-number 20 {
          action pass
        }
      }
    }
  }
}
```

- Create a flow list to match the traffic
- Apply a flow filter to a VTN interface
- Flow filter can include *redirect* action
- Redirect action could perform MAC rewrite

Use CLI, GUI or API to perform these operations

OpenFlow and SDN Webinars on ipSpace.net



Trainings

- Live sessions
- On-Site workshops
- Recordings and subscriptions

Other resources

- Consulting
- Books and case studies

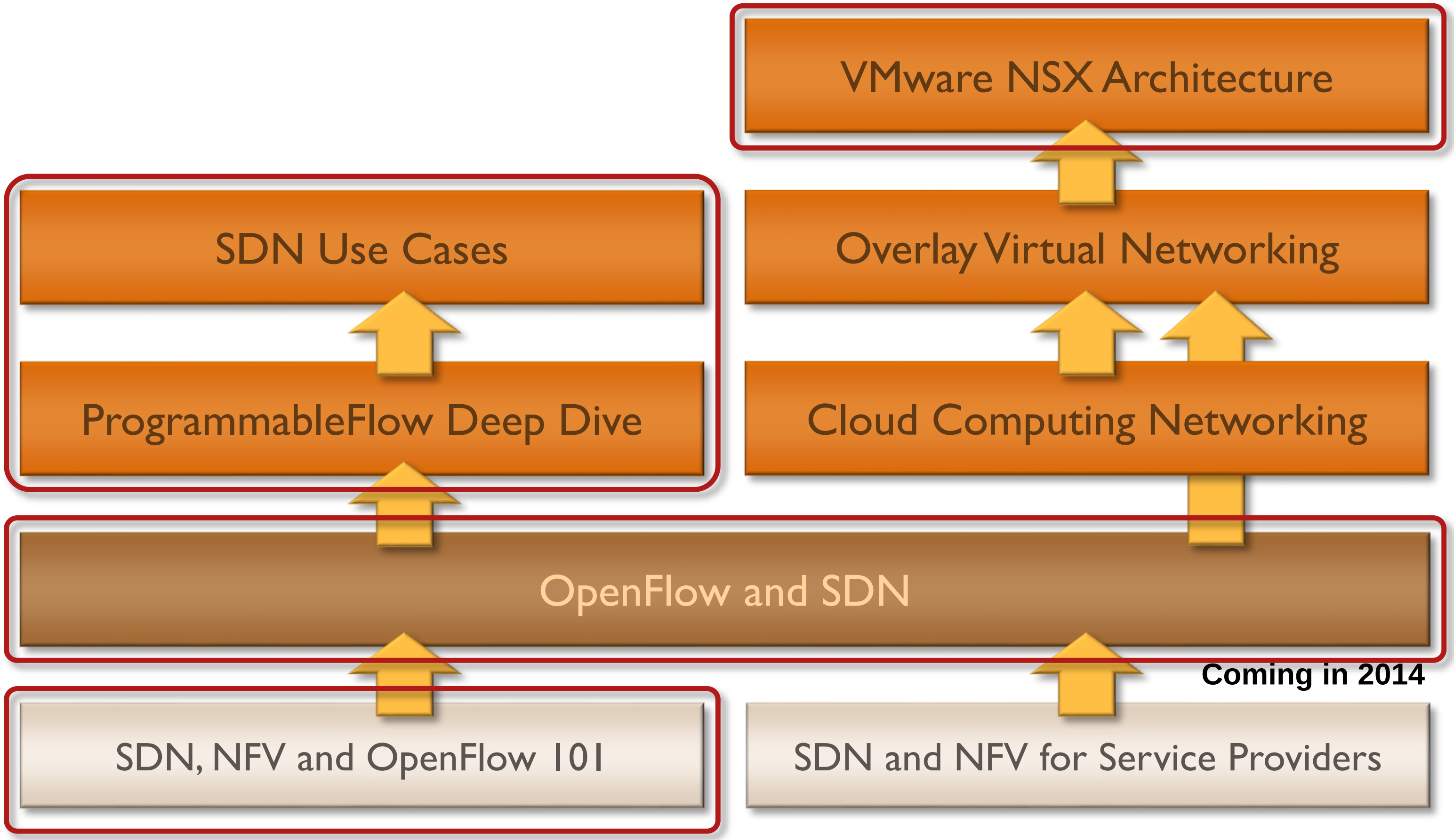
Should I Care?

Conclusions

- SDN and OpenFlow are interesting concepts
- They will significantly impact the way we do networking
- Centralized computation and management plane makes more sense than centralized control plane
- OpenFlow is just a low-level tool
- Initial SDN use cases: large data centers @ portals or cloud providers (cost cutting or virtualized networking)
- Still a very immature technology, standards are rapidly changing
- Northbound controller API is missing (but badly needed) → Creating controller vendor lock-in
- Already crossed the academic → commercial gap

If you want to get involved, NOW is a good time

OpenFlow and SDN Webinars on ipSpace.net



Trainings

- Live sessions
- On-Site workshops
- Recordings and subscriptions

Other resources

- Consulting
- Books and case studies

More information @ http://www.ipSpace.net/SDN

Questions?

Send them to ip@ipSpace.net or @ioshints