“Extended OpenFlow protocol for circuit switching domain and optical network virtualization”

Siamak Azodolmolky, Reza Nejabati, Dimitra Simeonidou

{sazod, rnejab, dsimeo}@essex.ac.uk

OFELIA-CHANGE summer school

7-11 November 2011

University of Essex

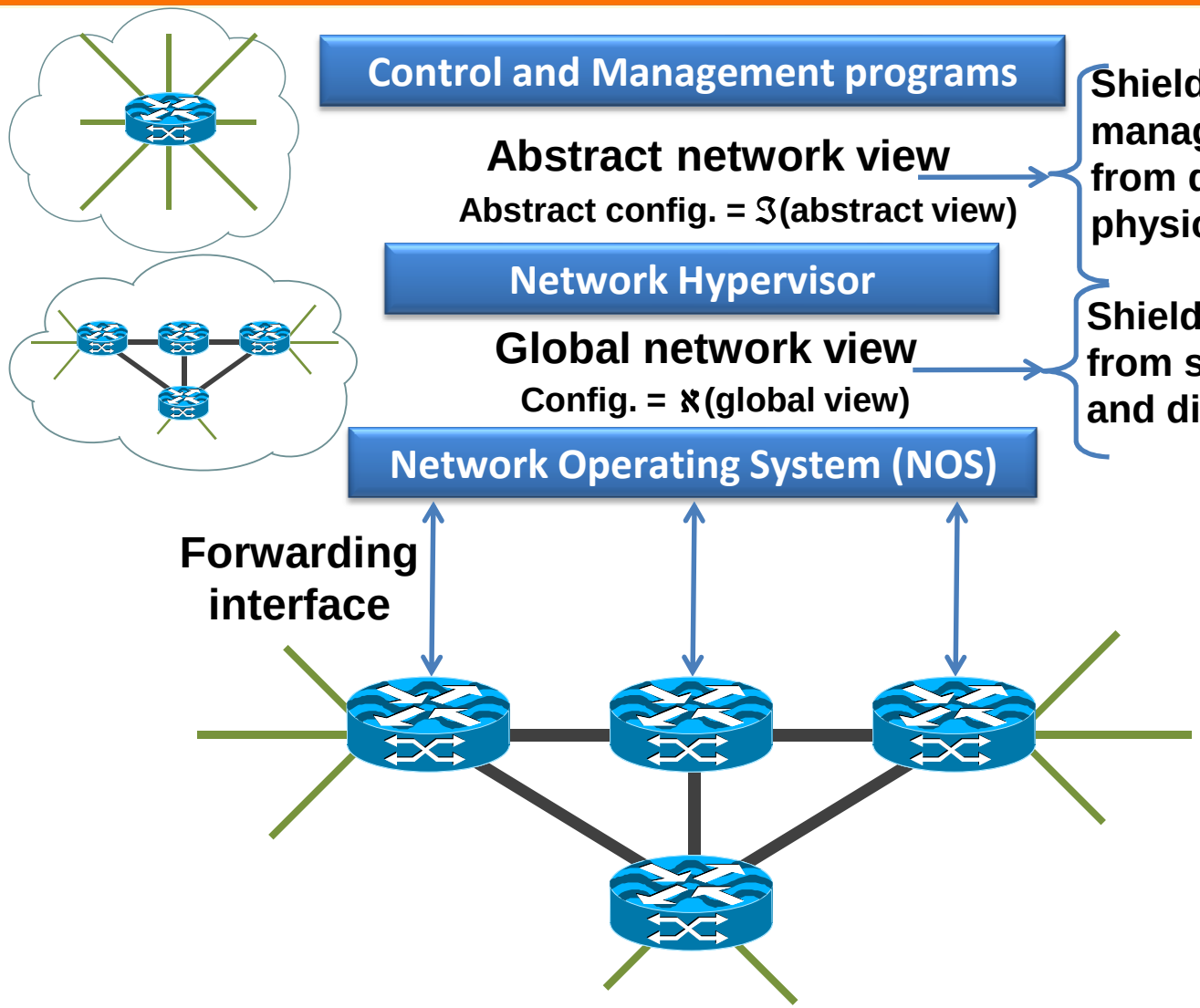
Talk layout

- Software Defined Networking: A review
- Unified Control and Management
  - An OpenFlow approach
- Extended OpenFlow to support circuit switching
- Optical network considerations
- GMPLS control plane
- Interworking of OpenFlow and GMPLS
- OFELIA approach
- Conclusions

Software Defined Networking: A review

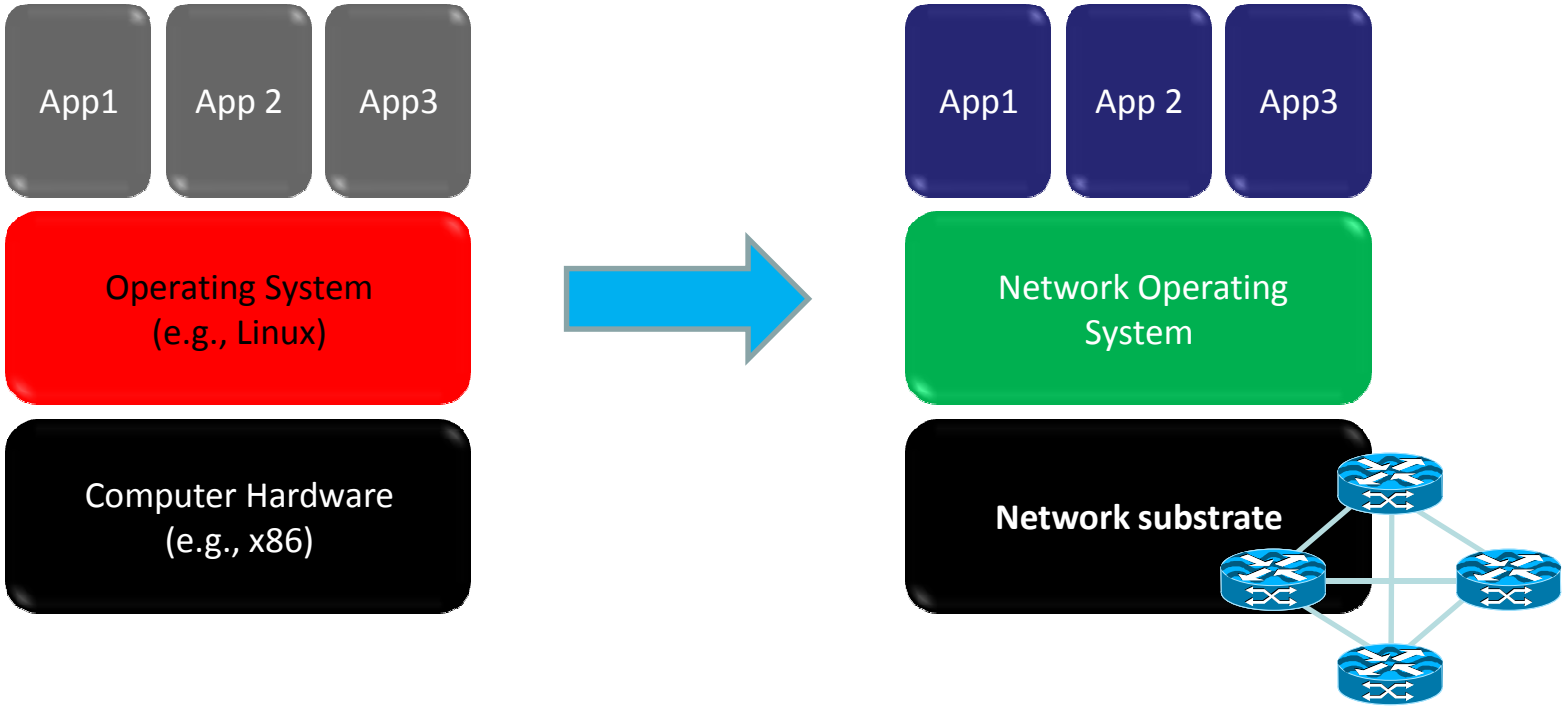
- Control and management mechanisms are not advancing with the same peace compared with the evolution of networking technologies.
- The ability to **master complexity** is not the same as the ability to **extract simplicity.**
- Software defined networking (SDN) is an approach to networking, which enables **applications** (control and management programs) to **manipulate the control software of networking equipments**.

Software Defined Networking framework



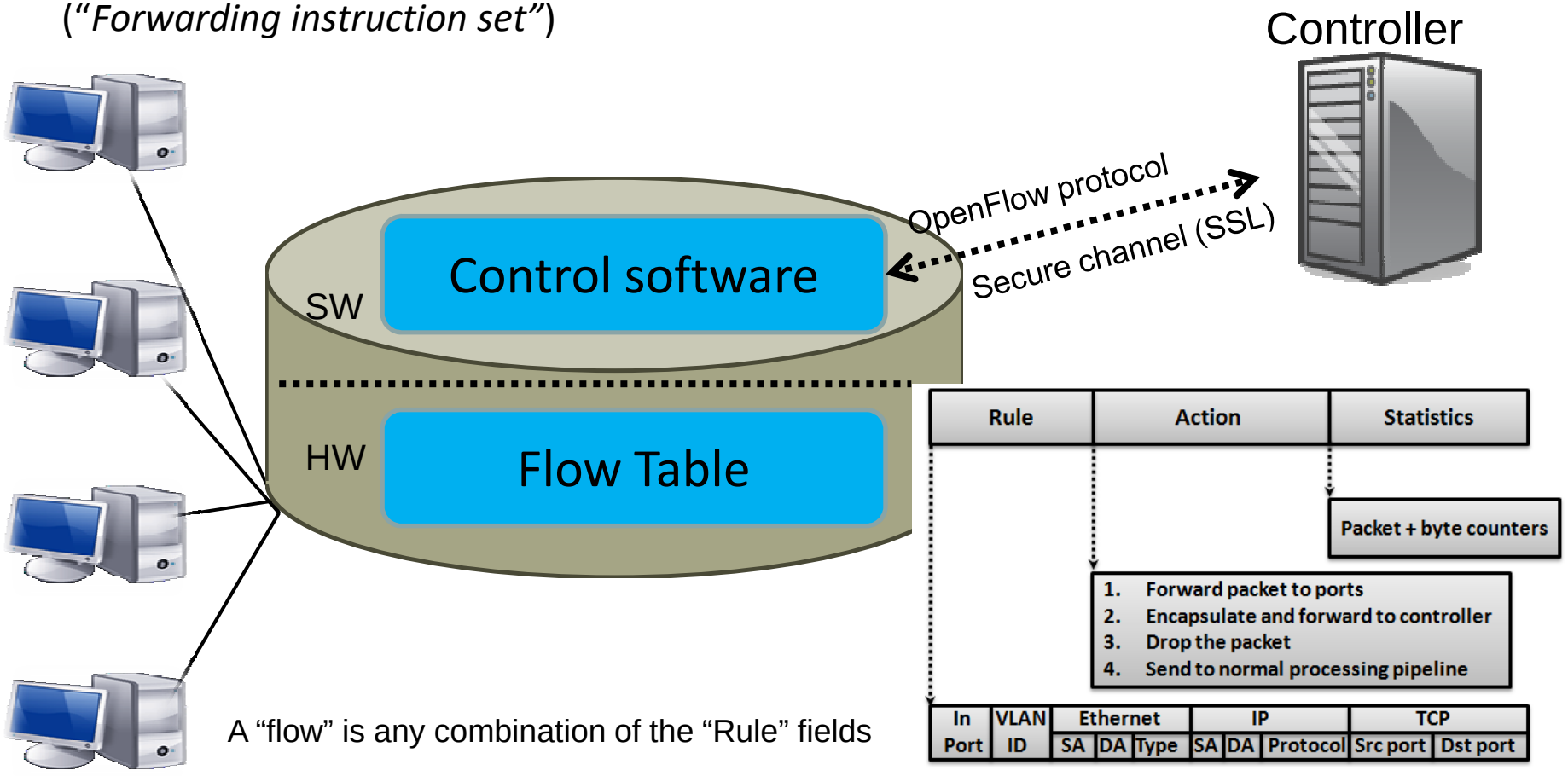
Innovation pattern

- **Computing industry infrastructure**:
  - Hardware substrate below and **programmability**, **strong isolation model**, and **competition above → Faster innovation**



OpenFlow: An SDN implementation

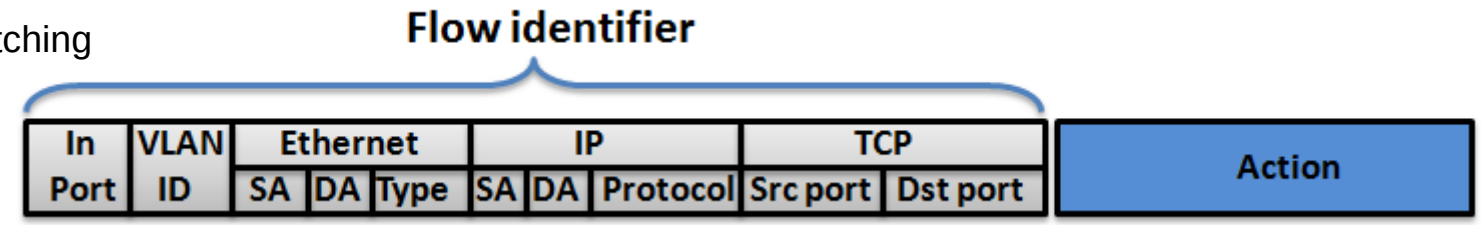
- Decouple **control** from **data** path
- Cache **control decisions** in data path (flow table) using small set of primitives (“*Forwarding instruction set*”)



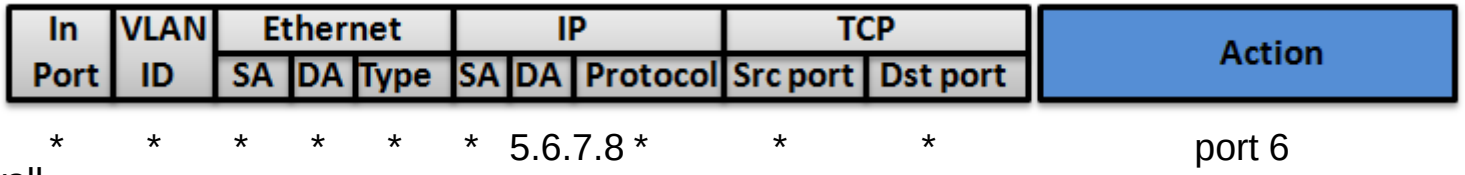
A “flow” is any combination of the “Rule” fields

Examples

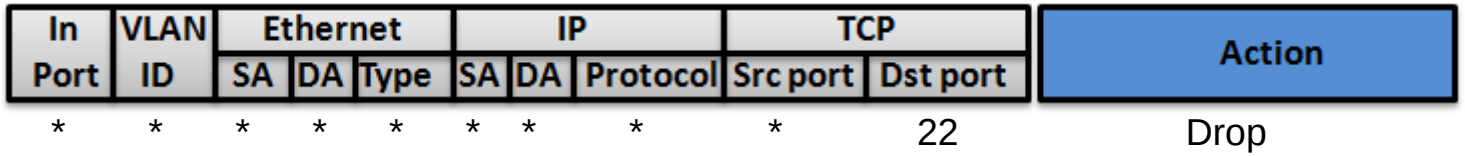
Switching



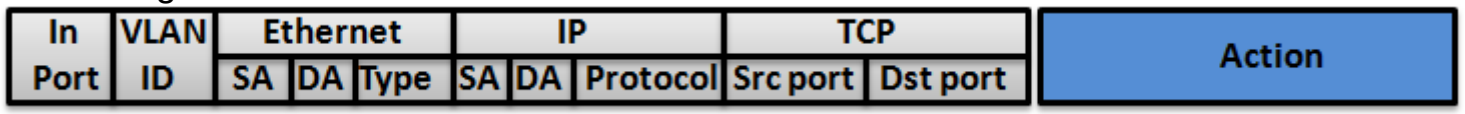
Routing



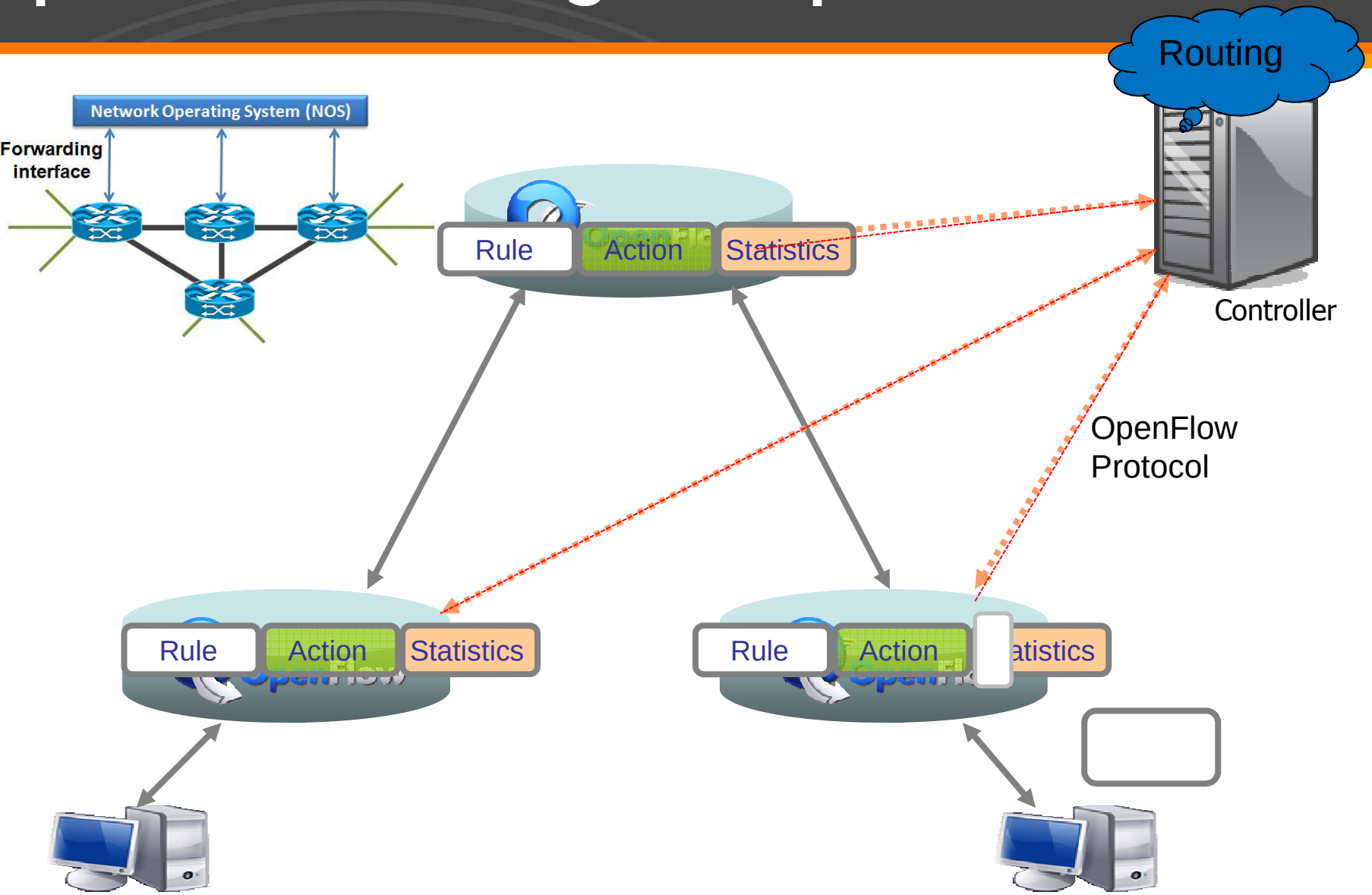
Firewall



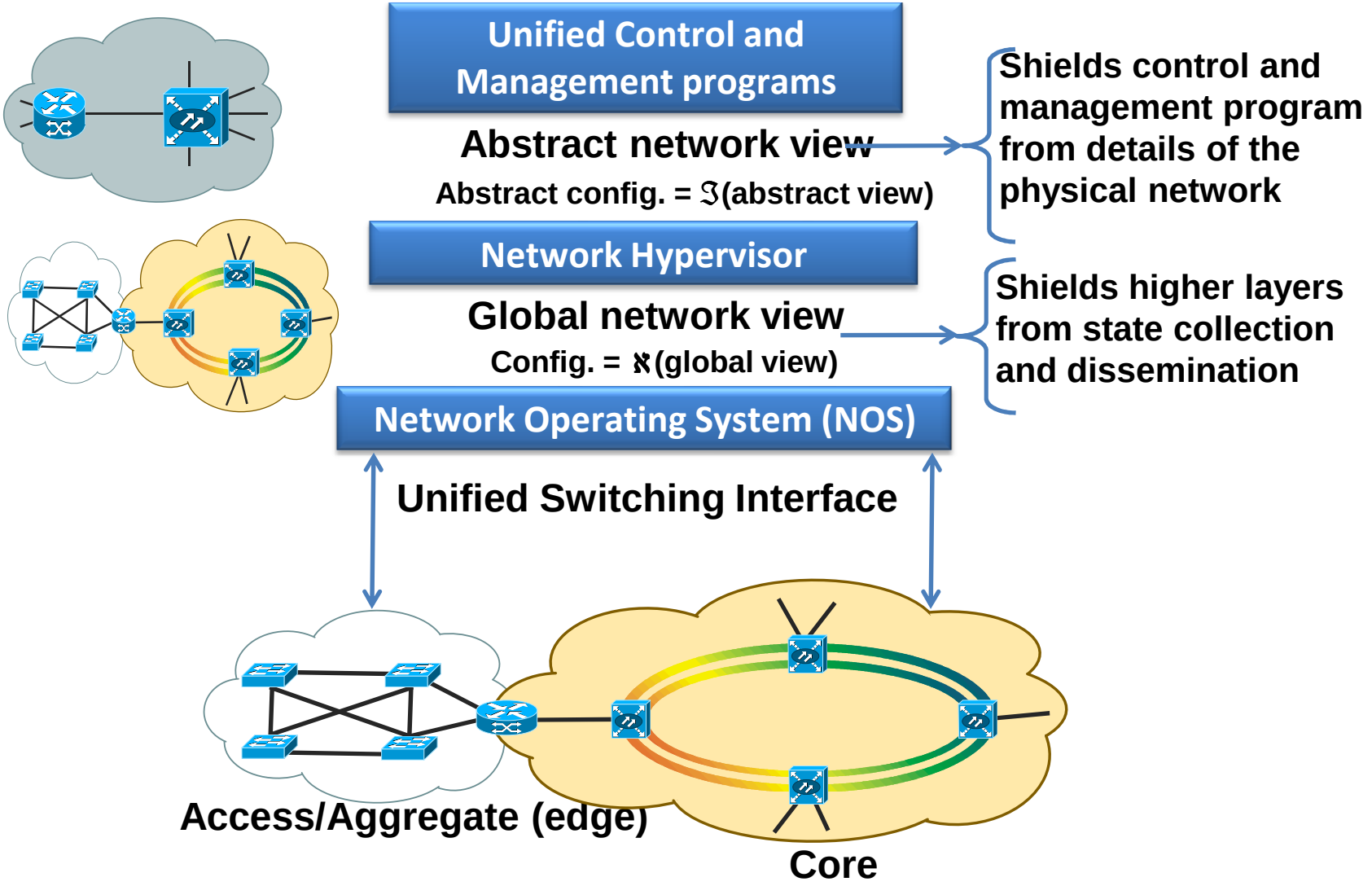
VLAN switching



OpenFlow Routing Example



Unified Control & Management



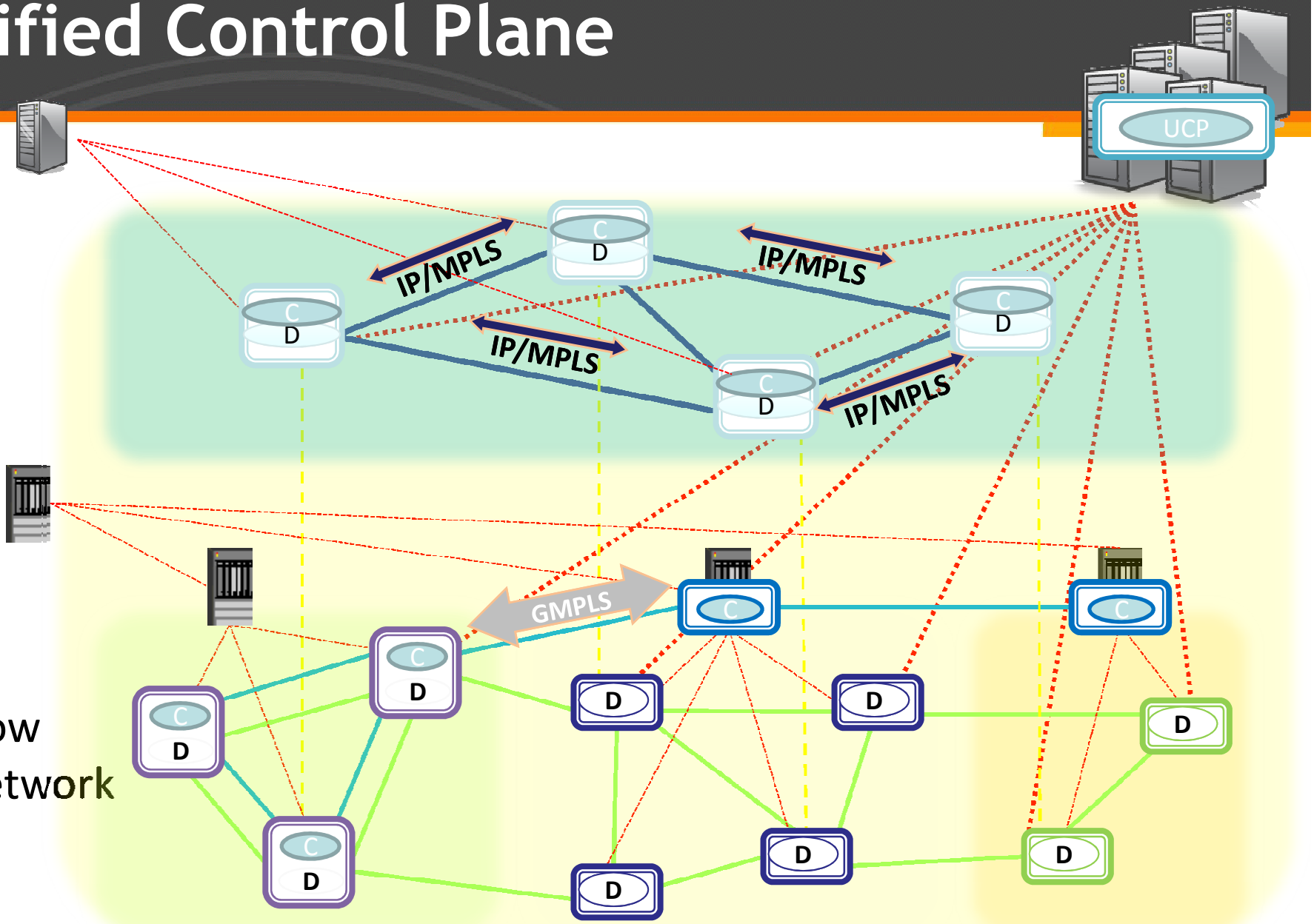
Talk layout

- Software Defined Networking: A review
- Unified Control and Management
  - An OpenFlow approach
- Extended OpenFlow to support circuit switching
- Optical network considerations
- GMPLS control plane
- Interworking of OpenFlow and GMPLS
- OFELIA approach
- Conclusions

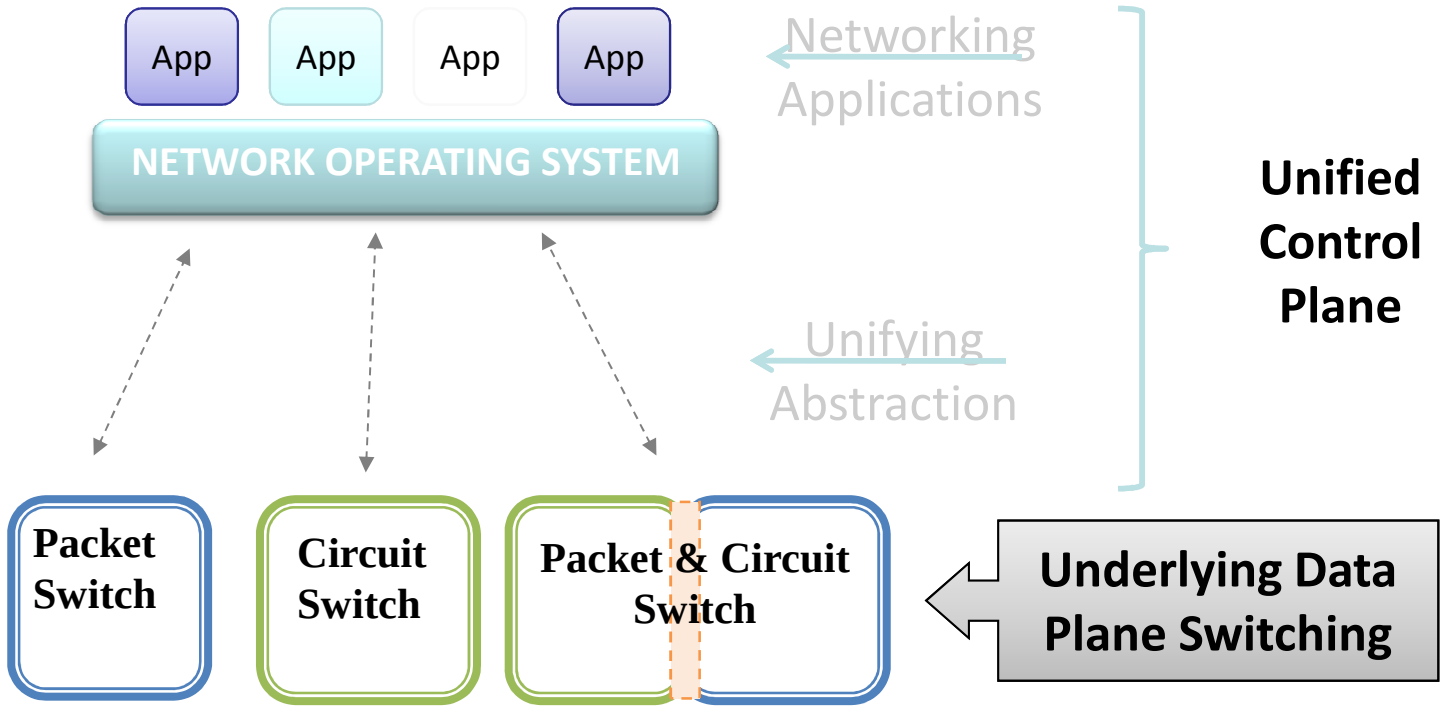
Convergence

- Packet and optical circuit switching convergence
- Motivations:
  - IP and Transport networks are separate networks that are controlled and managed independently leading to duplication of functions and resources in multiple layers (leading to high CAPEX and OPEX)
  - IP and Transport networks do not dynamically interact (no or limited cross-layer optimization opportunities)
  - IP and Transport networks have different architectures that makes integrated operation and convergence challenging

Unified Control Plane



Unified control plane: An OpenFlow approach



OpenFlow and circuit switching

Packet Flows

In Port	VLAN ID	Ethernet			IP			TCP		Action
		SA	DA	Type	SA	DA	Protocol	Src port	Dst port	

Circuit flows

In Port	In Lambda	VCG	Starting Time-Slot	Signal Type	⇔	Out Port	Out Lambda	VCG	Starting Time-Slot	Signal Type

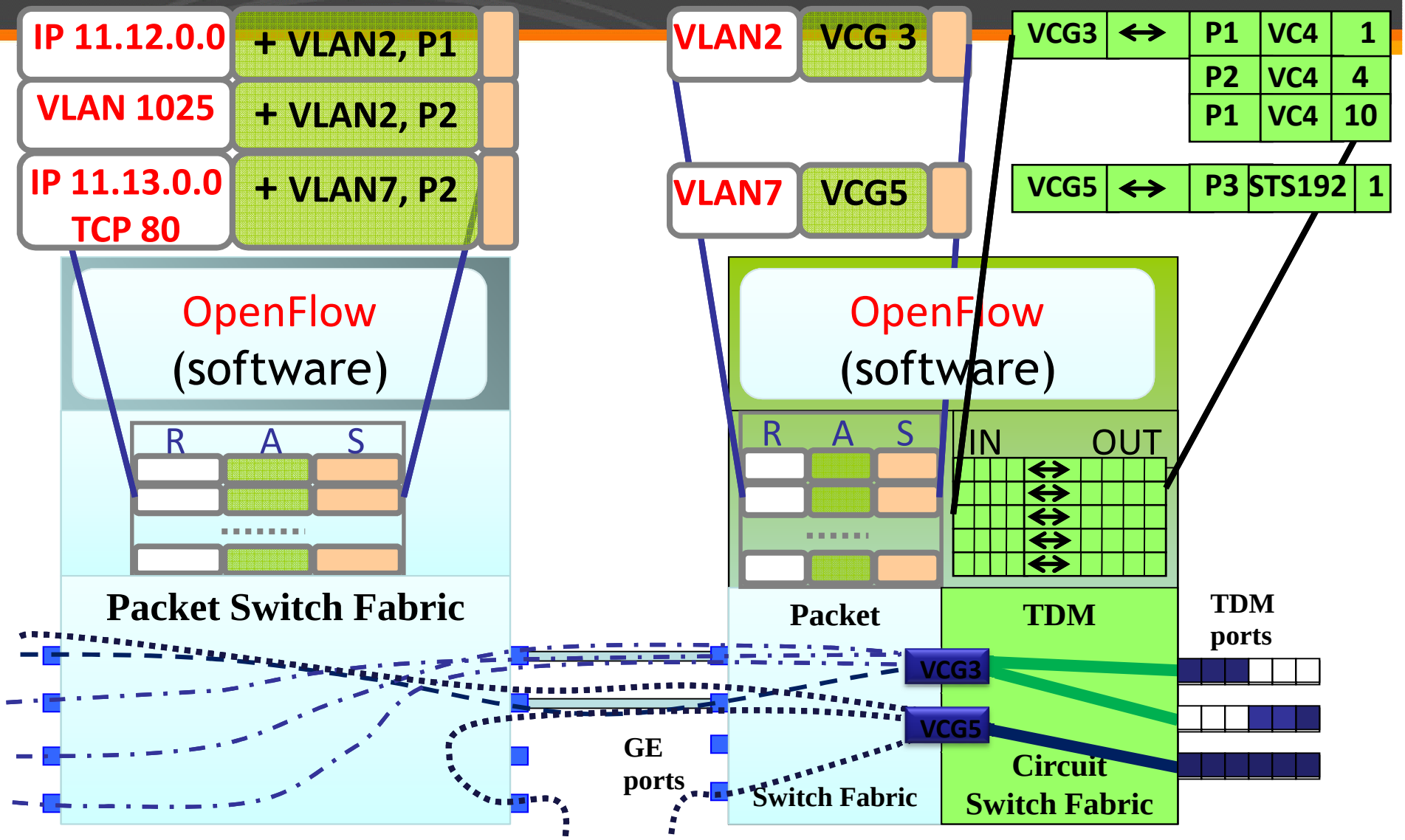
Exploit the cross-connect table in circuit switches

Cross-connect table (could) exists in circuit switch hardware

We can establish API to that table via Extended OpenFlow

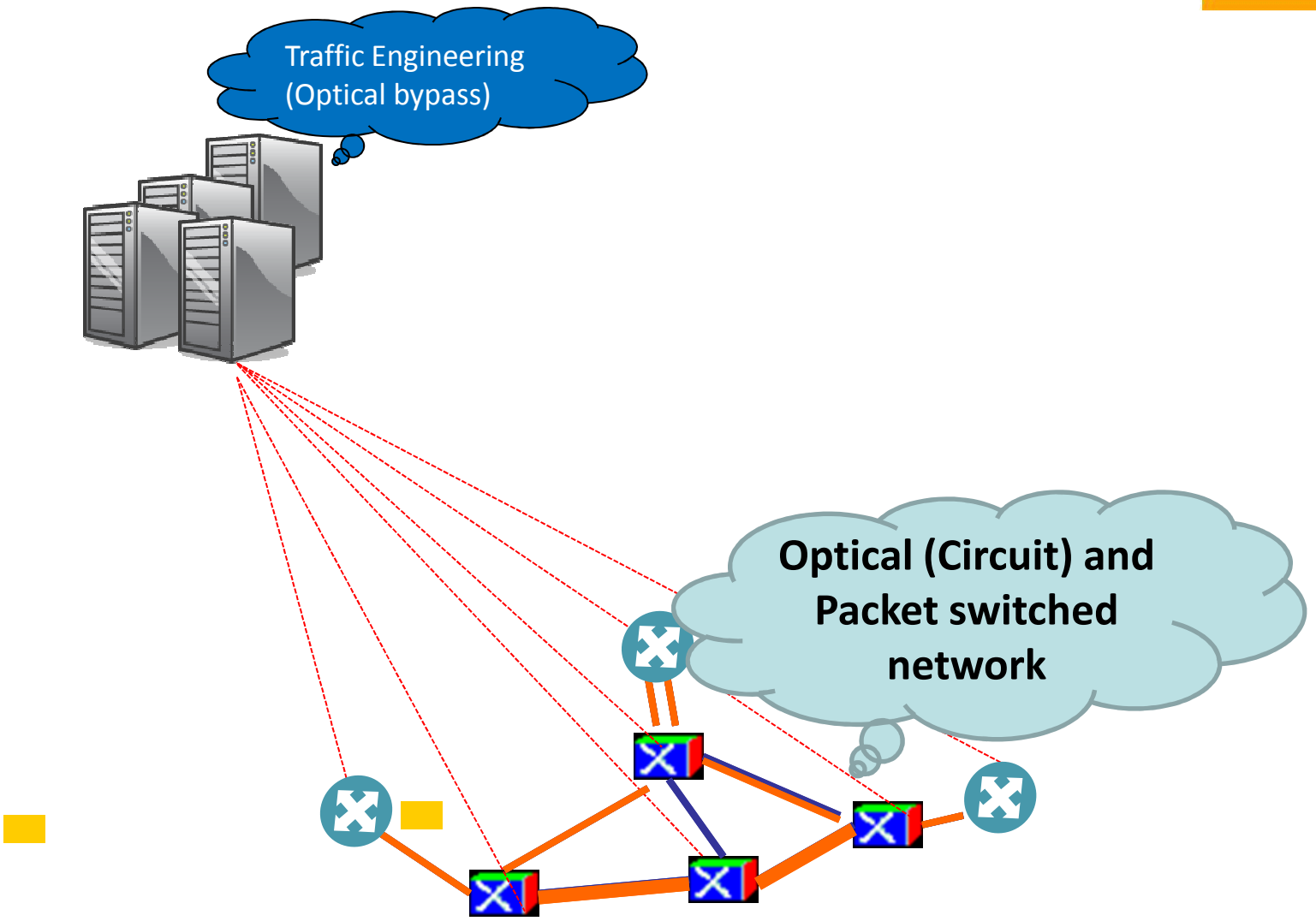
The Flow Abstraction presents a unifying abstraction which removes the distinction between underlying packet and circuit switched networks and regarding both as flows in a flow-switched networks

An example

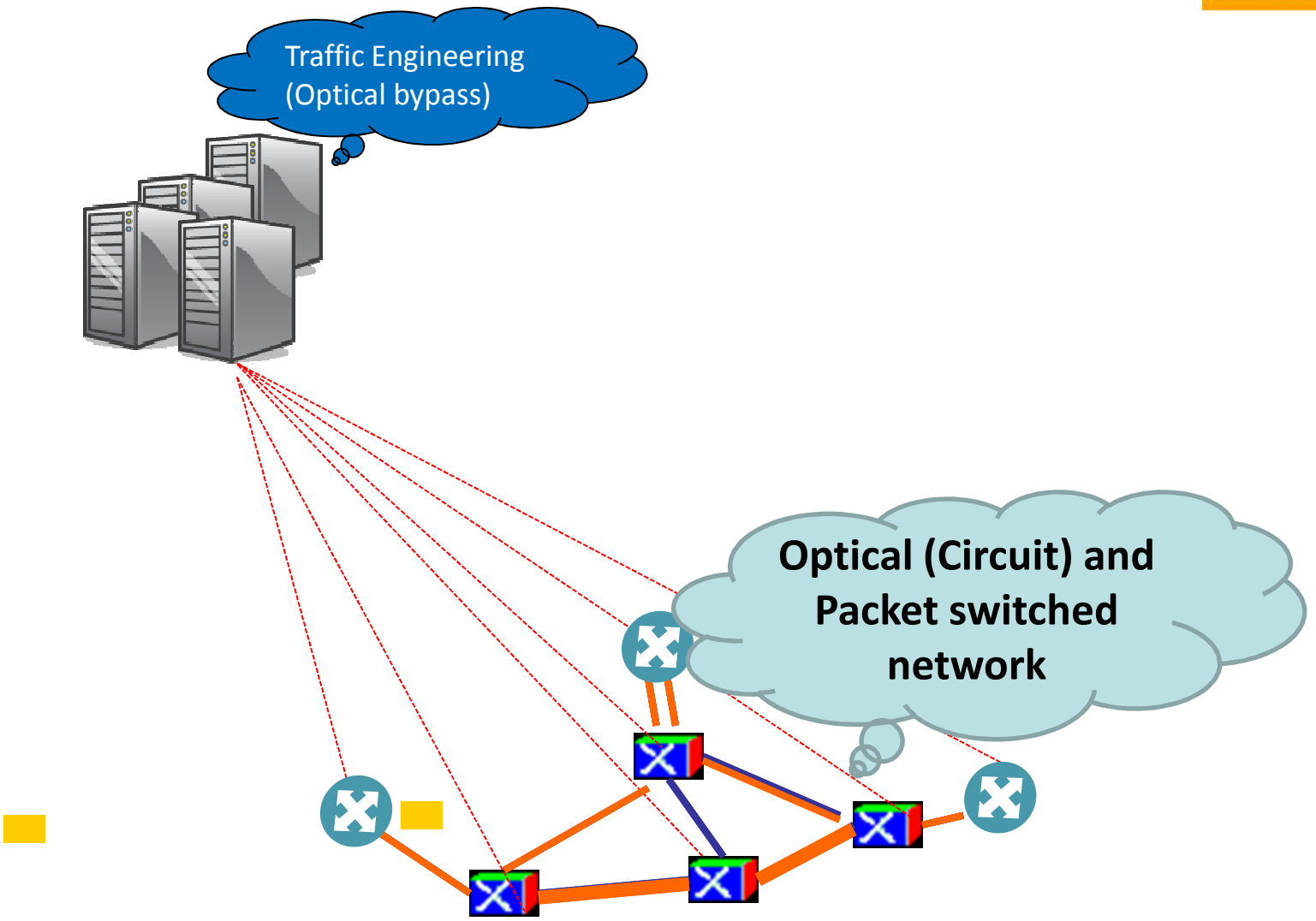


Source:Saurav Das, Guru Parulkar, & Nick McKeown, "Unifying Packet & Circuit Networks with OpenFlow", 3 Feb. 2010

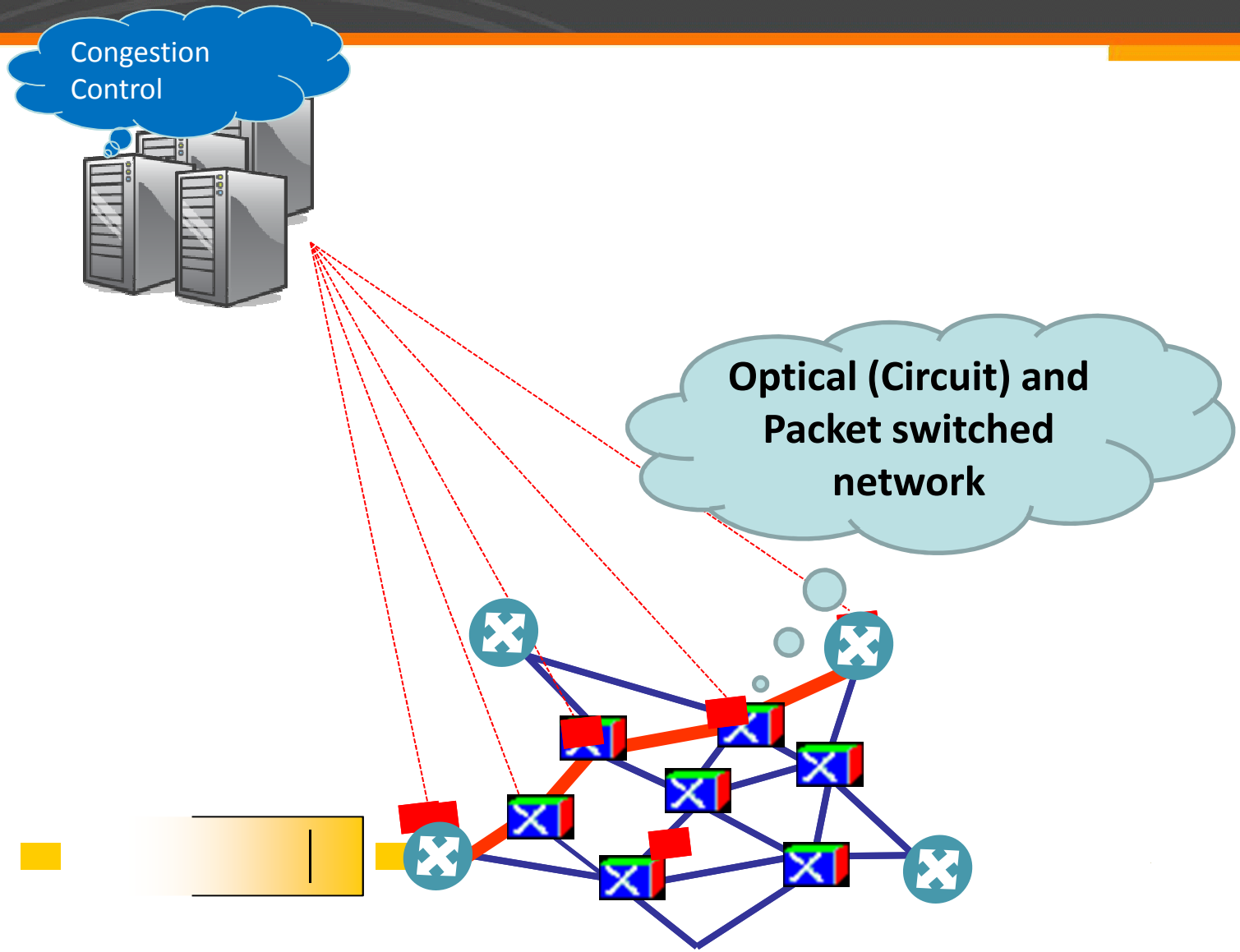
Sample application: Traffic Engineering



Sample application: Traffic Engineering



Sample application: Congestion control



Efforts so far…

- S. Das, G. Parulkar, N. McKeown, “Unifying Packet and Circuit Switched Networks,” IP Networking Workshop, **Globecom 2009**.
- V. Gulda, S. Das, A. Shastri, G. Paulkar, N. McKeown, L. Kazovsky, “Experimental Demonstration of OpenFlow Control of Packet and Circuit Switches,” **OFC 2010**.
- S. Das, G. Parulkar, N. McKeown, “Packet and Circuit Network Convergence with OpenFlow,” **OFC 2010**.
- S. Das, “Extensions to the OpenFlow Protocol in Support of Circuit switching,” OpenFlow protocol Spec. (v1.0), Addendum 0.3, June **2010**.
- S. Das, Y. Yiakoumis, G. Parulkar, N. McKeown, “Application-Aware Aggregation and Traffic Engineering in a Converged Packet-Circuit Network,” **OFC 2011**.
- S. Azodolmolky, R. Nejabati, E. Escalona, R. Jayakumar, N. Efstathiou, D. Simeonidou, “Integrated OpenFlow-GMPLS Control Plane: An Overlay Model for Software Defined Packet over Optical Networks,” **ECOC 2011**.

Summary

- An SDN based (e.g. OpenFlow) approach is a *candidate* to materialize a unified control and management framework for converged packet and (optical) circuit switched domains.

Talk layout

- Software Defined Networking: A review
- Unified Control and Management
  - An OpenFlow approach
- Extended OpenFlow to support circuit switching
- Optical network considerations
- GMPLS control plane
- Interworking of OpenFlow and GMPLS
- OFELIA approach
- Conclusions

Existing proposed extension

- S. Das, “Extensions to the OpenFlow Protocol in Support of Circuit switching,”
  - OpenFlow protocol Specification (v1.0),
  - Addendum 0.3, **June 2010**
  - **http://www.openflow.org/wk/index.php/PAC.C**

Definitions and assumptions (1/2)

- An **OpenFlow circuit switch** or **hybrid switch** consists of a **cross-connect table** (see Fig.1), which caches the information about the existing circuit flows (or cross-connects made in the switch), and a **secure channel to an external controller**, which manages the switch over the secure channel using the OpenFlow protocol.
- Unlike an OpenFlow packet switch, the **cross-connect table** is not used to lookup flows, as circuits ports typically have no visibility into packets (the payload).
- The controller is responsible for **provisioning and removing connections** in an OpenFlow circuit switch via the **extended OpenFlow protocol**.

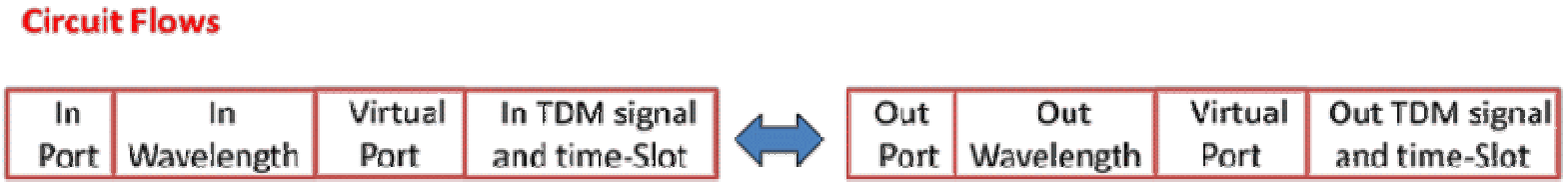


Fig. 1: Circuit flow table (cross-connect table) entry

Definitions and assumptions (2/2)

- Some circuit switches include **packet interfaces** and can additionally switch packets electronically. These switches should maintain **separate** packet and circuit flow tables. The extended OpenFlow protocol defines the mapping of **packet flows** to **circuit flows** (and back) via **virtual ports**.

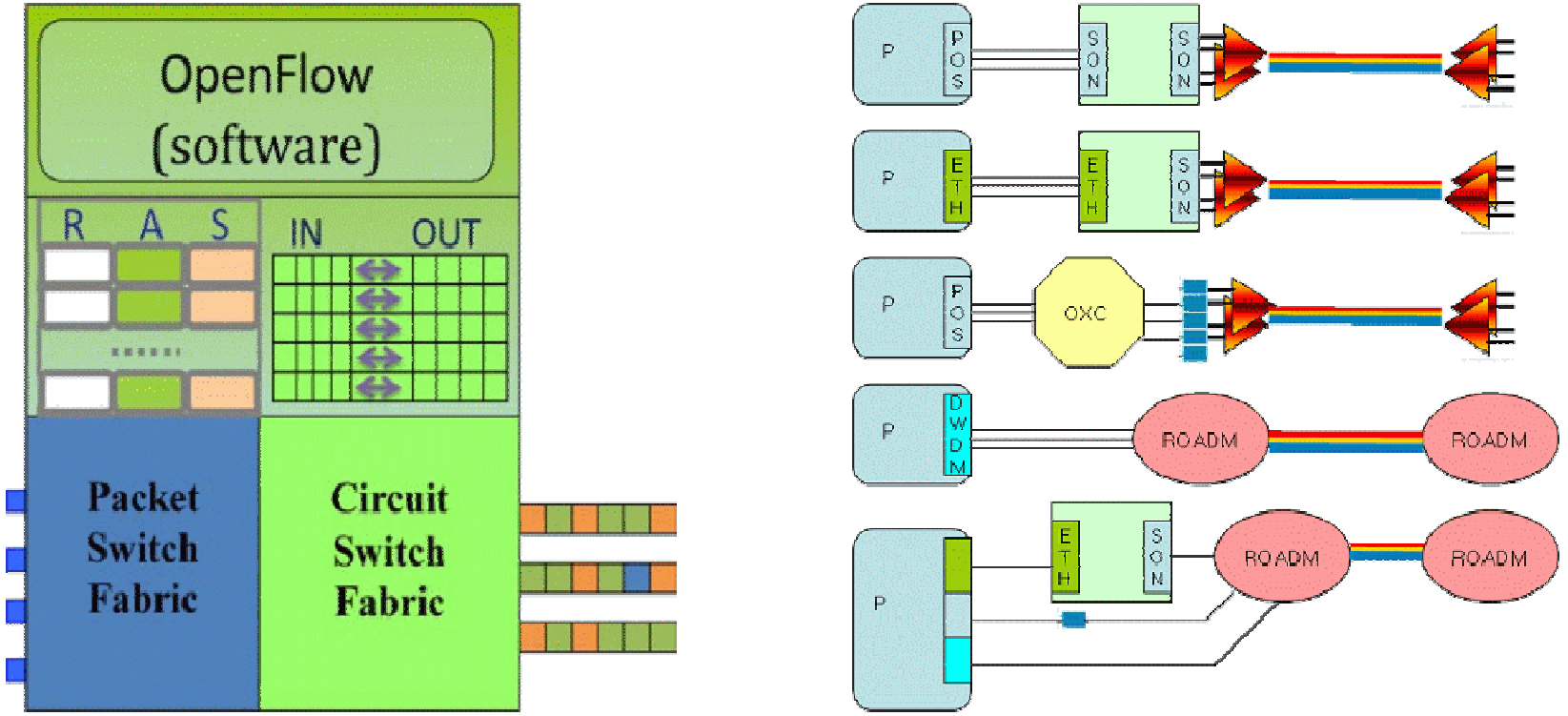


Fig 2(a) An OpenFlow switch with packet and circuit hardware flow tables; 2(b) Various ways to interconnect packet and circuit switches

OpenFlow extensions (draft v0.3)

- Extensions to OpenFlow 1.0 (openflow.h):
  - Switch features
  - Port structure
  - Cross-connect structure
  - Circuit flow add/modify/delete
  - Circuit flow action types
  - Error messages
  - Port status

1) Switch features

- The following capabilities are defined for circuit switches:

```
OFPC_CTG_CONCAT     = 1 << 31, /* Contiguous concat on all TDM ports. */
OFPC_VIR_CONCAT     = 1 << 30, /* Virtual concat on all TDM ports. */
OFPC_LCAS           = 1 << 29, /* Link Capacity Adjustment Scheme */
OFPC_POS            = 1 << 28, /* Packet over Sonet adaptation */
OFPC_GFP            = 1 << 27, /* Generic Framing Procedure */
OFPC_10G_WAN        = 1 << 26  /* native transport of 10G_WAN_PHY
                                  on OC-192 */
```

2) Port structure (1/5)

- The only change made from the OpenFlow packet switch specification is that the number of physical ports is limited to **0xfa00** instead of **0xff00** and the numbers in between is used to specify **internal** and **virtual ports**.
  - These **internal port** numbers can be used to specify "mapper" ports that map Ethernet packets to TDM time-slots.
  - The **virtual ports** can be used to define Virtual Concatenation Group (VCG) numbers used for VCAT technology in TDM switches.
  - **Virtual concatenation (VCAT)** is an inverse multiplexing technique creating a large capacity payload container distributed over multiple smaller capacity TDM signals. Virtual concatenation has been defined for SONET/SDH, OTN and PDH path signals.

```
/* Maximum number of physical switch ports. */
/* Switch internal ports - 0xfa00 to 0xfaff */
/* Switch virtual circuit ports - 0xfb00 to 0xfeff */
/* Other virtual ports - 0xff00 to 0xffff */
```

2) Port structure (2/5)

- The “features” bitmap has been modified to include line-rates in transport networks.
- Notes: This specification:
  - **does not** currently fully support OTN (Optical Transport Network).
  - **does not** currently support MPLS label switching or TDM switching based on OTN frame formats.
  - **does not** support TDM signals smaller than STS-1

```
/* The following have been added for WAN interfaces */
OFPPF_X           = 1 <<20, /* Dont care applicable to fiber ports */
OFPPF_OC1         = 1 <<21, /* 51.84 Mbps OC-1/STM-0 */
OFPPF_OC3         = 1 <<22, /* 155.52 Mbps OC-3/STM-1 */
OFPPF_OC12        = 1 <<23, /* 622.08 Mbps OC-12/STM-4 */
OFPPF_OC48        = 1 <<24, /* 2.48832 Gbps OC-48/STM-16 */
OFPPF_OC192       = 1 <<25, /* 9.95328 Gbps OC-192/STM-64 */
OFPPF_OC768       = 1 <<26, /* 39.81312 Gbps OC-768/STM-256 */
OFPPF_100GB       = 1 <<27, /* 100 Gbps */
OFPPF_10GB_WAN    = 1 <<28, /* 10 Gbps Ethernet WAN PHY (9.95328 Gbps) */
OFPPF_OTU1        = 1 <<29, /* OTN OTU-1 2.666 Gbps */
OFPPF_OTU2        = 1 <<30, /* OTN OTU-2 10.709 Gbps */
OFPPF_OTU3        = 1 <<31  /* OTN OTU-3 42.836 Gbps */
```

2) Port structure (3/5)

- The **peer_port_no** and **peer_datapath_id** fields report to the controller the discovered peer's datapathid (i.e., circuit switch id) and the port to which this port connect to.
  - Currently the specification assumes that the circuit switch is running an instance of a **neighbor discovery protocol** as is common using for example the SONET/SDH header bytes.
  - After such discovery **at power-up**, the switch reports this information to the controller in the "switch_features_reply".
  - The controller can then develop the full circuit topology.

2) Port structure (4/5)

- Meaning of "**bandwidth**" field:
  - For a switching type of OFPST_WAVE, bandwidth has the following meaning:
    - The lower 10 bits of the 64 bit uint64_t will be used for flags with special meaning. The upper 54 bits will be used to designate ITU-T grid frequencies supported by the switch port.

```
enum ofp_port_lam_bw {
    OFPCBL_X        = 1 << 0,/* 1 if fiber switch, 0 if wavelength switch */
    OFPCBL_100_50   = 1 << 1,/* 1 if 100GHz channel spacing, 0 if 50GHz */
    OFPCBL_C_L      = 1 << 2,/* 1 if using C-band frequencies, 0 if L-band */
    OFPCBL_OSC      = 1 << 3,/* 1 if supporting OSC at 1510nm, 0 if not */
    OFPCBL_TLS      = 1 << 4,/* 1 if using a TLS, 0 if not */
    OFPCBL_FLAG     = 1 << 5 /* 1 if reporting wave-support, 0 if reporting used waves
};
```

2) Port structure (5/5)

- Meaning of "**bandwidth**" field:
  - For a switching type of OFPST_T_SONET or OFPST_T_SDH, **bandwidth1** and **bandwidth2** fields should be used together to identify the available starting time slots on the optical carrier.
  - This then depends on the line-rate of the optical carrier and the minimum switching granularity of the switch.
  - This specification has currently **only** provided the bit interpretation for STS-1 signal.

3) Cross-connect structure

- The logical equivalent of the ofp_match structure from the packet switching specification is the **ofp_connect** structure in the circuit switching addendum.
- It is used to describe the **circuit flow** much like the match structure is used to describe the packet flow.
- When describing a cross-connection, the **ofp_connect** is used within **ofp_cflow_mod** message.
- The **wildcards** field simply identifies which fields in the **ofp_connect** structure should be ignored when looking to cross-connect an incoming port to an out-going port.
  - 0x003C → regular in and output ports
  - 0x0033 → TDM to TDM cross-connection
  - 0x00FF → Lambda cross connection

```
/* Circuit flow wildcards */
enum ofp_connect_wildcards {
    OFPCW_IN_PORT    = 1 << 0,
    OFPCW_OUT_PORT   = 1 << 1,
    OFPCW_IN_TPORT   = 1 << 2,
    OFPCW_OUT_TPORT  = 1 << 3,
    OFPCW_IN_WPORT   = 1 << 4,
    OFPCW_OUT_WPORT  = 1 << 5
};
```

4) Circuit flow add/modify/delete

- Modification to the flow table (cross-connect table) in a circuit switch are accomplished via the **OFPT_CFLOW_MOD** message.
  - Controller → datapath
- OFPFC_MODIFY_STRICT and OFPFC_DELETE _STRICT are used to modify and terminate existing connections.
- OFPFC_DROP command is used to signify termination of a circuit flow and subsequent extraction of packet flows from it.

5) Circuit flow action types

- The defined action types are:
  - OFPAT_CKT_OUTPUT = 0xfffd
  - OFPAT_CKT_INPUT = 0xfffe
- These two actions are defined and used in the context of **inserting** and **extracting** packet flows from circuit flows.
- The following operations are defined:
  - Adapting packet flows to circuit flows
    - OFPCAT_NONE, OFPCAT_POS (Packet over SONET/SDH), OFPCAT_GFP (Generic Framing Procedure), and OFPCAT_10G_WAN (10G Ethernet WAN PHY framing for OC-192 of SDH STM-64) are defined as "**adaptation**" types.
    - As a **special case** VCG operations(creating, adding members to VCG, modifying and deleting VCG) are explained in the proposed extension.
  - Extracting packet flows from circuit flows
    - This is done by defining a cross-connect to a drop port

6) Error messages

- A new error message is defined with associated error codes for reporting problems with circuit flow setup.
  - OFPET_CFLOW_MOD_FAILED = 0xffff /* Problem modifying circuit flow entry */

```
/* ofp_error_msg 'code' values for OFPET_CFLOW_MOD_FAILED.  'data' contains
 * at least the first 64 bytes of the failed request. */
enum ofp_cflow_mod_failed_code {
  OFPCFMFC_VCG_DEF,     /* Creating or modification of VCG failed */
  OFPCFMFC_OVERLAP,     /* Attempted to add overlapping flow with currently
                           used time-slots */
  OFPCFMFC_MISMATCH     /* Mistmatched tsignals in ofp_connect struct */
};
```

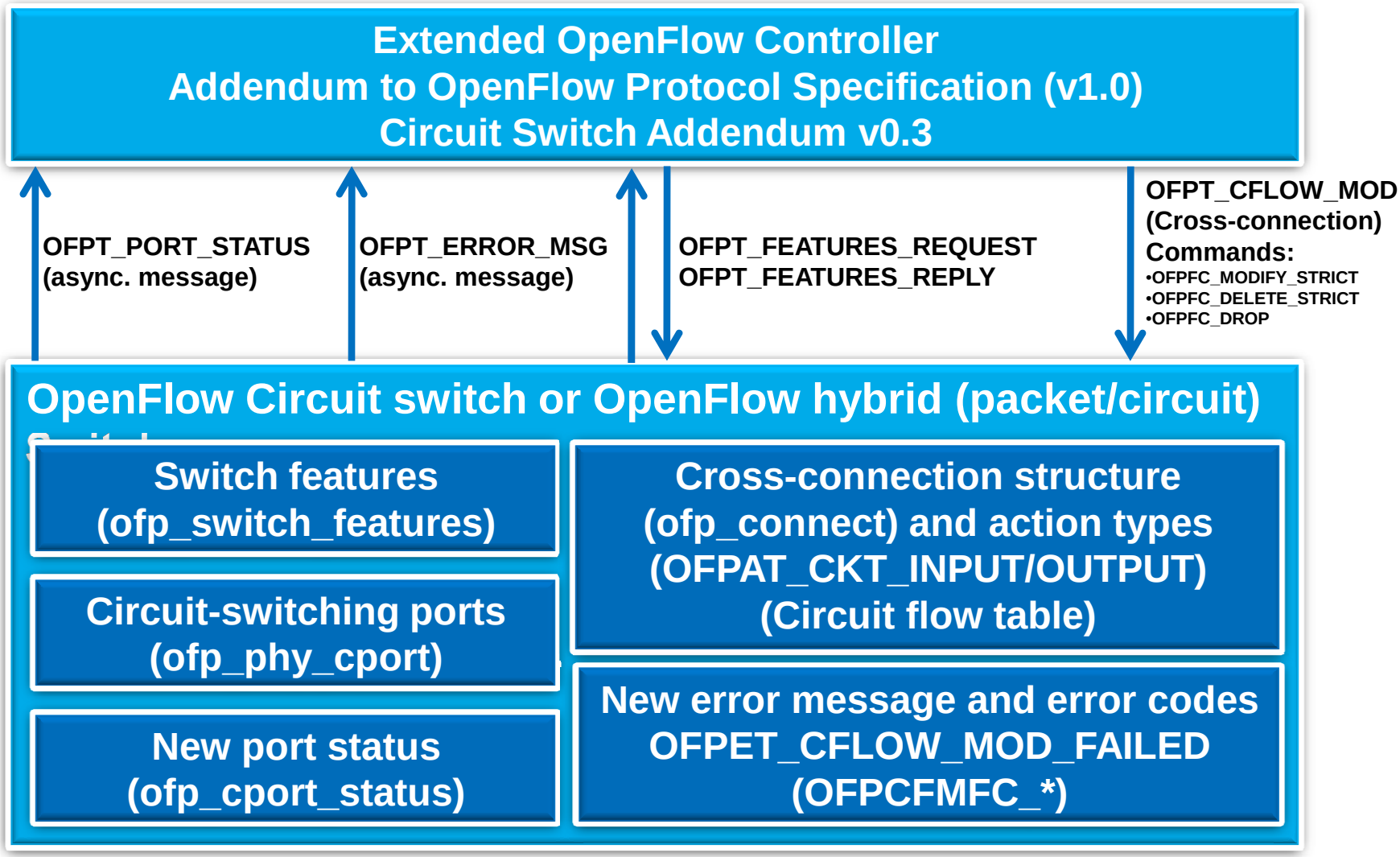
7) Port status

- Switch can report status changes of circuit port using the “ofp_cport_status” structure. The “reason” field can take any of the following values:
  - OFPPR_BW_MODIFY = 255; /* bandwidth usage has changed */
  - OFPPR_BW_DOWN = 254; /* bandwidth time-slots have become un-usable */
- OFPPS_LINK_DOWN = 1<<0; is added as a physical port status.

Summary of extensions (1/2)

1. Extension of **switch features** (ofp_switch_features) to include circuit ports.
2. Definition of **circuit-switching ports** (ofp_phy_cport).
3. Definition of structure **ofp_connect** for specifying the cross connection in circuit switches.
4. New **OpenFlow messages**: **OFPT_CFLOW_MOD** to create virtual ports and to specify cross-connection and **OFPT_CPORT_STATUS** (with new reasons ofp_port_reason), and **error messages (OFPT_ERROR_MSG)**.
5. Two new **action types**: **OFPAT_CKT_OUTPUT** and **OFPAT_CKT_INPUT** to account for adapting packet flows to circuit flows and extracting packet flows from circuit flows.
6. A **mechanism** for the switch to report to the controller that some features of the circuit port has changed (support for network recovery) (OFPT_CPORT_STATUS).

Summary of extensions (2/2)

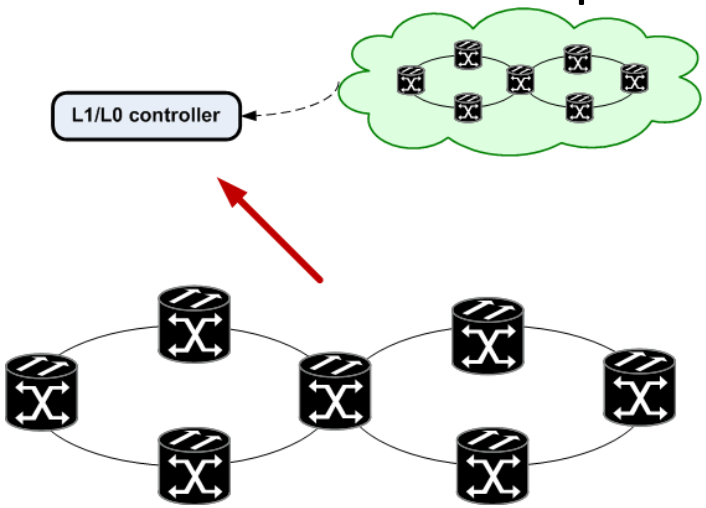


Talk layout

- Software Defined Networking: A review
- Unified Control and Management
  - An OpenFlow approach
- Extended OpenFlow to support circuit switching
- Optical network considerations
- GMPLS control plane
- Interworking of OpenFlow and GMPLS
- OFELIA approach
- Conclusions

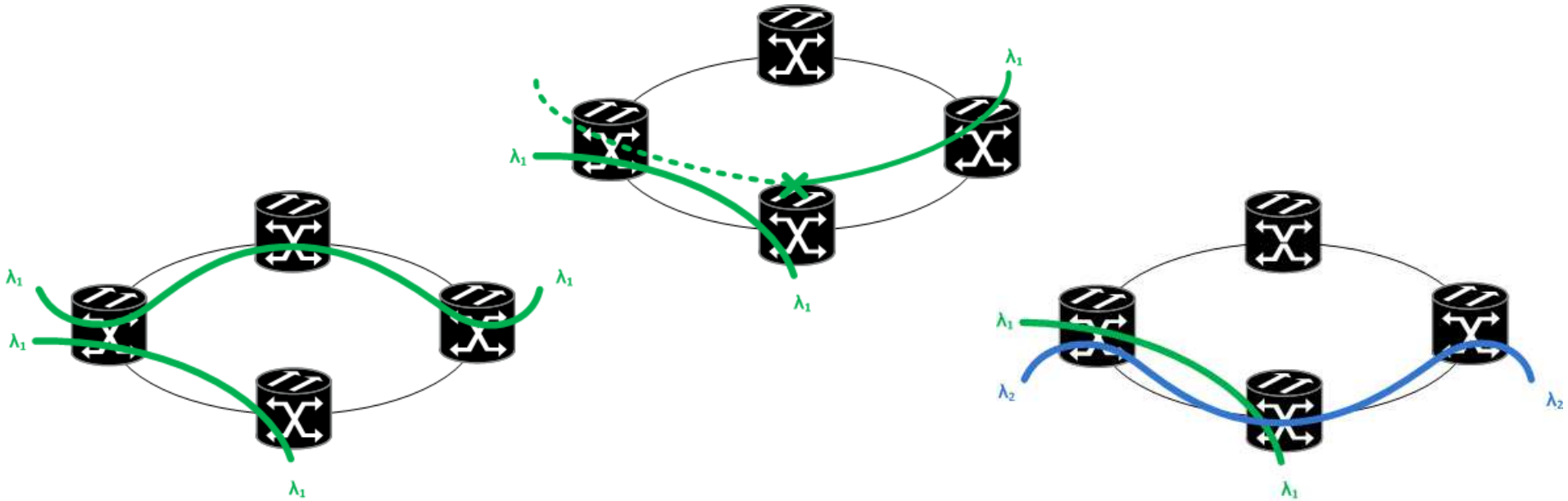
Topology discovery

- TE topology discovery and advertisements
  - Physical adjacency discovery mechanisms or manual provisioning
  - The lightpath through the network needs to be established prior to the data transmission
  - A scalable mechanism for retrieving topology information and network capabilities should be developed



Constraints in optical networks

- In optical networks a lightpath (route and wavelength) is an end-to-end tunnel.
  - Wavelength continuity must be guaranteed
  - Ingress and egress with fixed ports,
  - Ingress and egress with tunable ports,
  - More complex path computation for the systems with lamda conversions



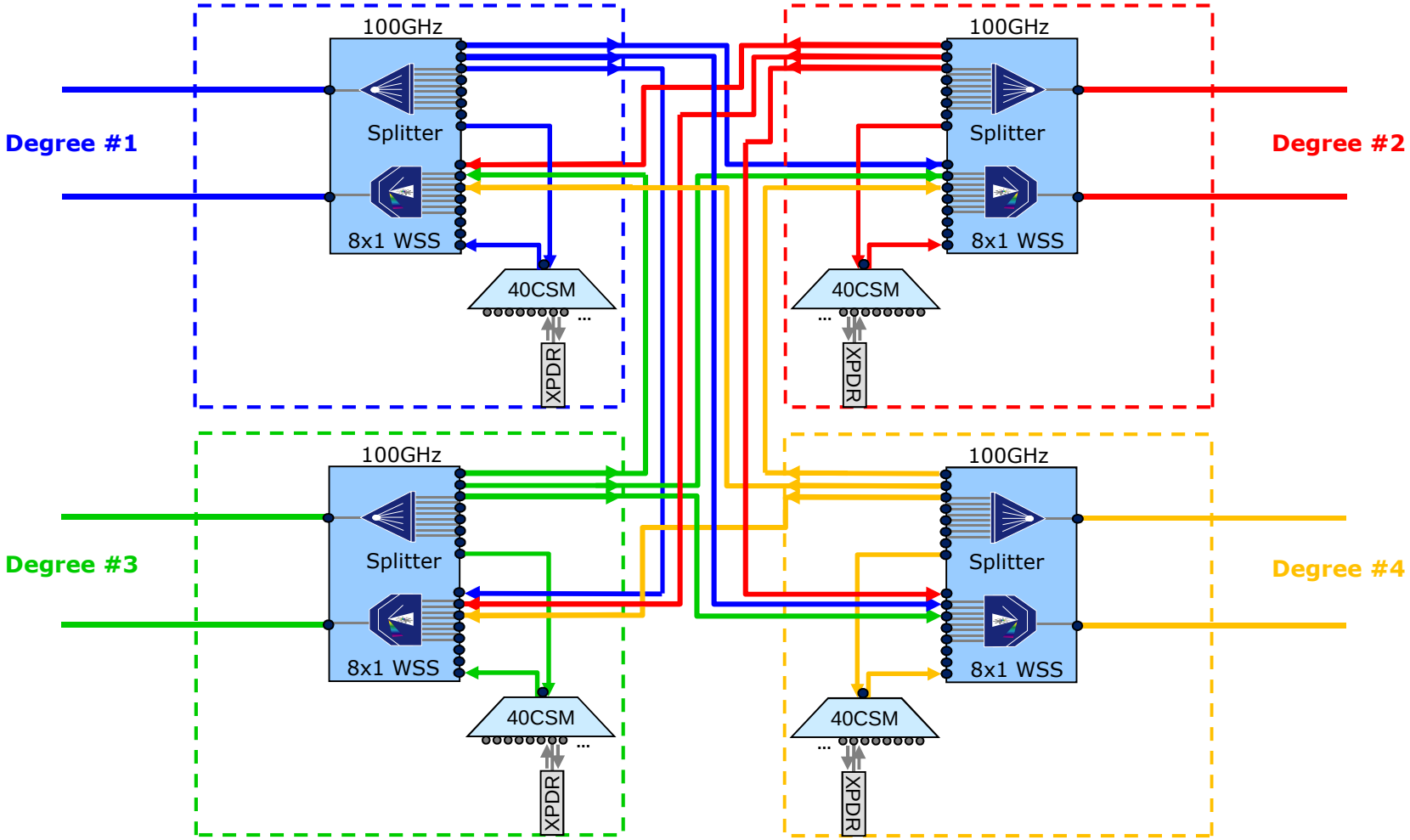
Switching constraints

- Switching constraints
  - map of internal connections and dependancies in digested form,
  - updated per each operation in transport plane,
  - used (together with topology graph) as an input for path computation,

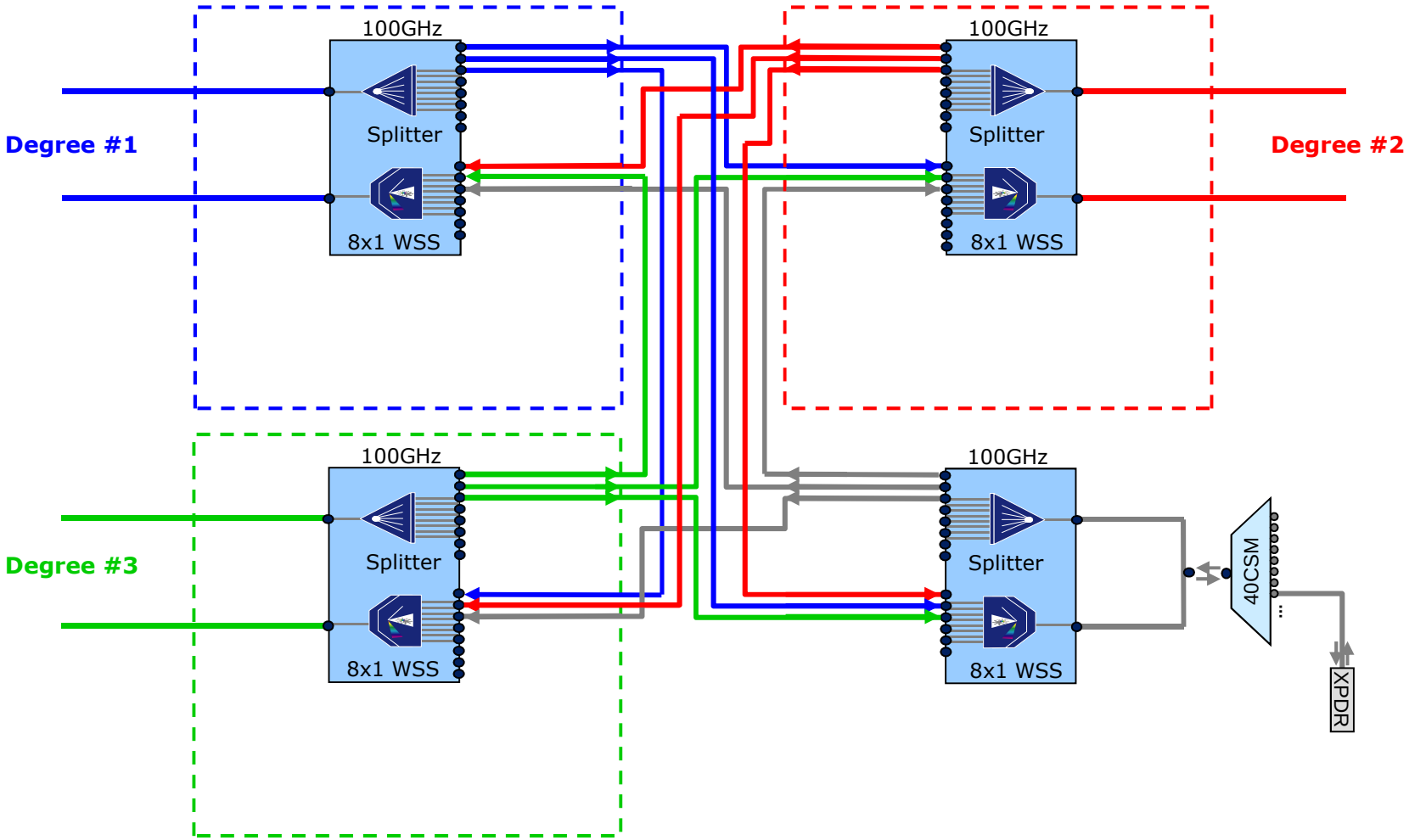
- Variety of different hardware configurations
  - FOADM (Fixed Optical Add Drop Multiplexer),
  - ROADM (directed, directionless, directionless and colorless),
  - support for 40 and 80 channel grids (interleved and non-interleved),
  - regenerators,
  - Many more…

Directed add-drop ROADM

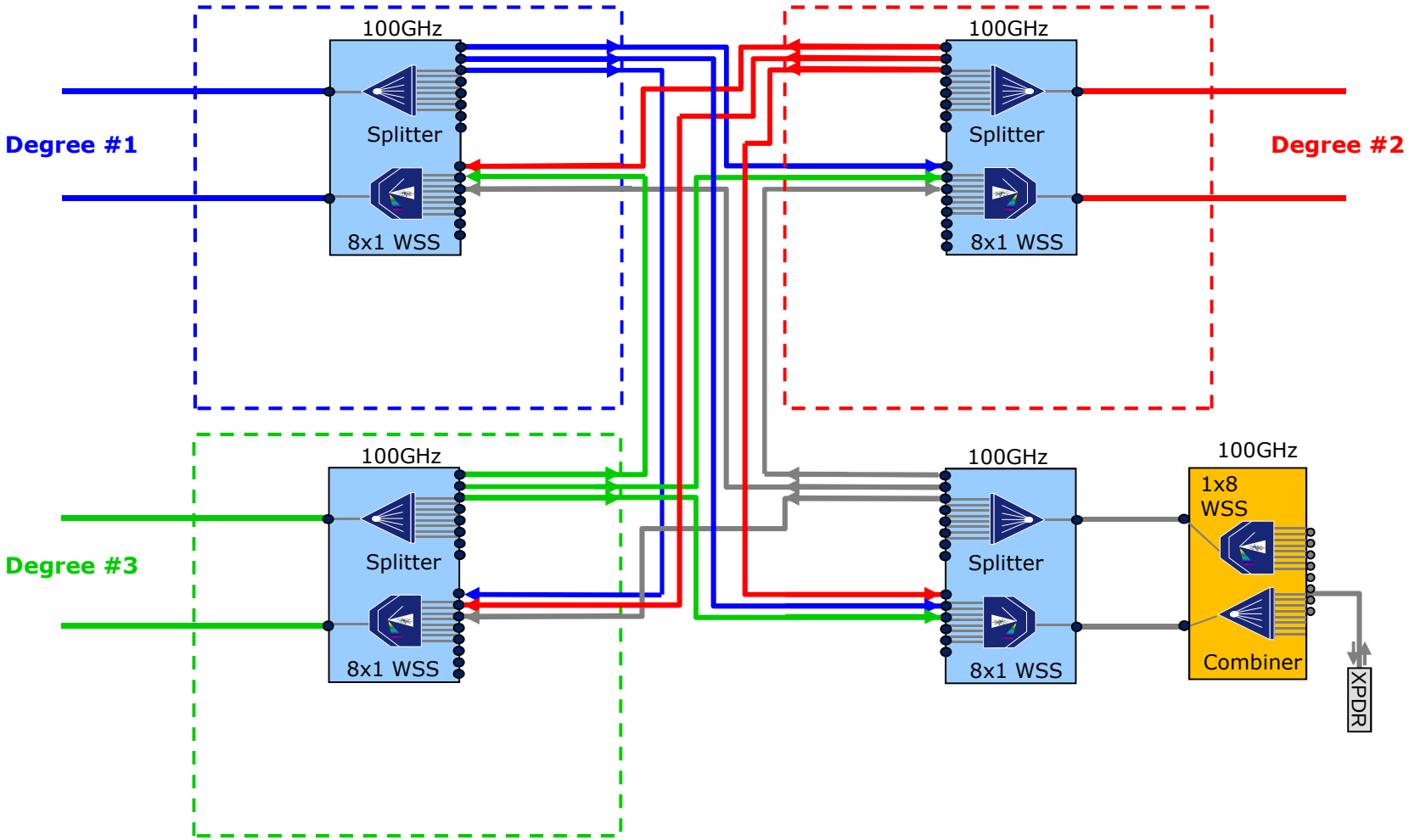
Reconfigurable Add Drop Multiplexer



Directionless ROADM



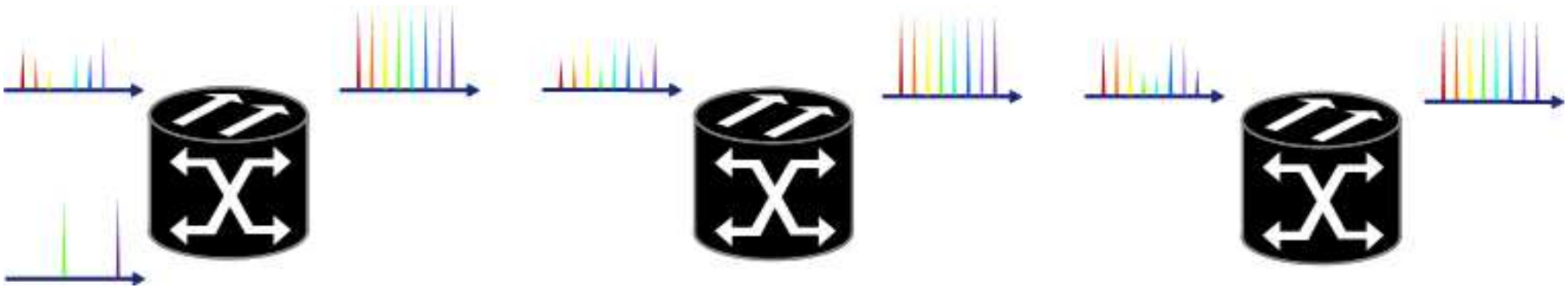
Directionless and colorless ROADM



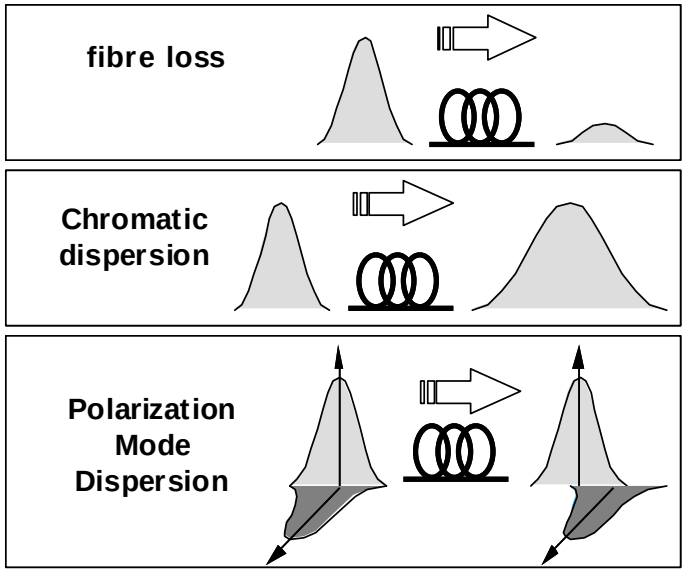
Power equalization

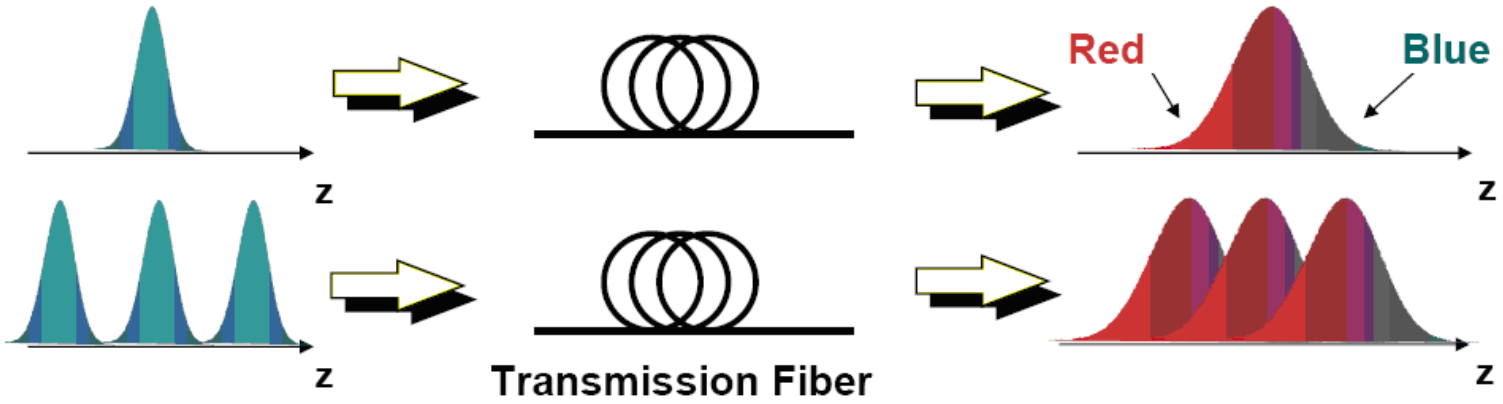
- Important part of service provisioning and maintanance
  - allow to measure and adjust individual wavelength power values,
  - maintain the wave-lengths within an adjustment band,
  - adding a new lambda results in correction in the whole spectrum,
  - power equalization process is bi-directional,
  - control plane automates the process by means of RSVP-TE
  - propietary extensions,

- → Changes in the controller and adaptations in OpenFlow protocol required



Physical layer impairments





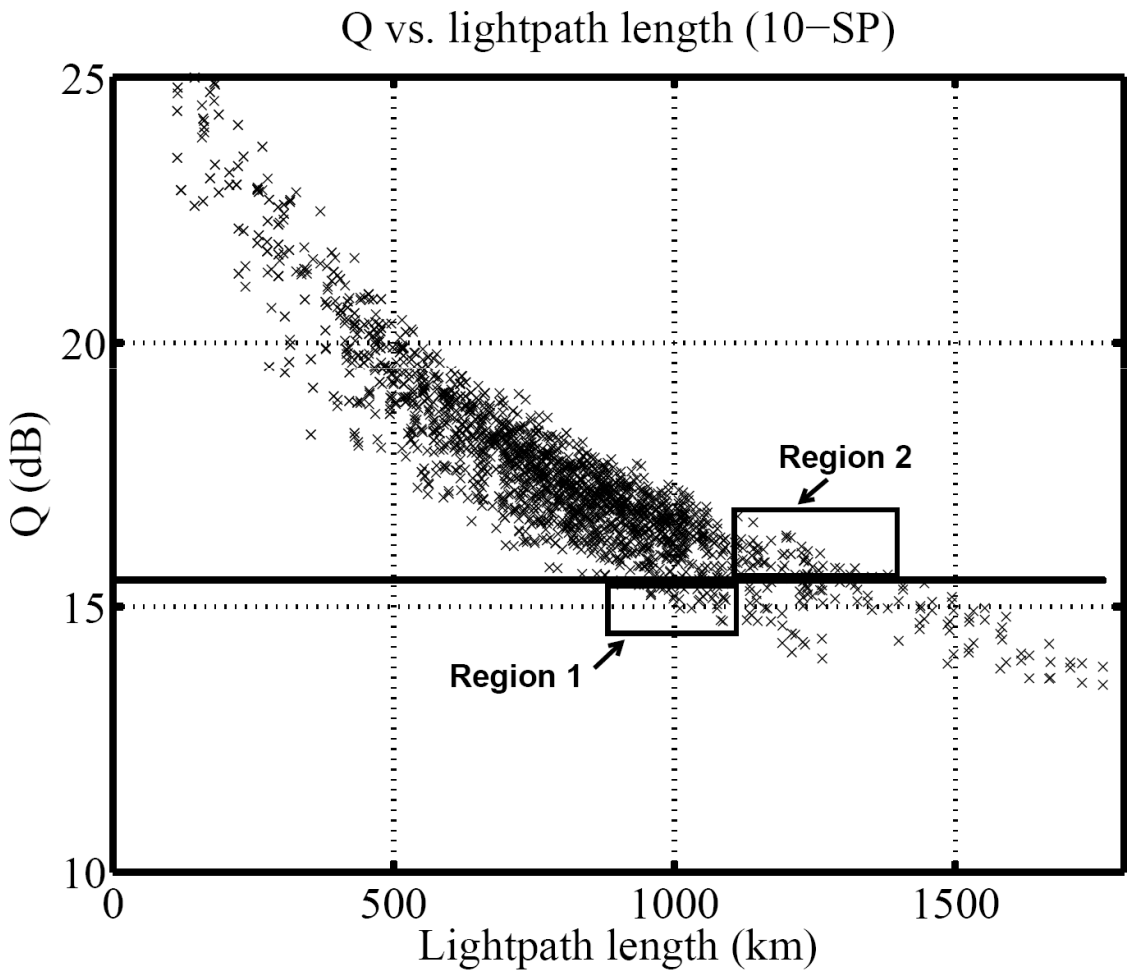
Quality of Transmission

- To estimate the Quality of Transmission (QoT) of a signal, a metric called “Q-factor” is estimated using a physical layer performance evaluator.
- The Q-factor is directly related to the Bit-Error Rate (BER) of transmission system via the following relationship:
  - (on-off keying intensity modulated signals)

$$Q = \frac{P_1 - P_0}{\sigma_1 + \sigma_0} \qquad BER = \frac{1}{2} erfc\left(\frac{Q}{\sqrt{2}}\right)$$

Impairment aware RWA

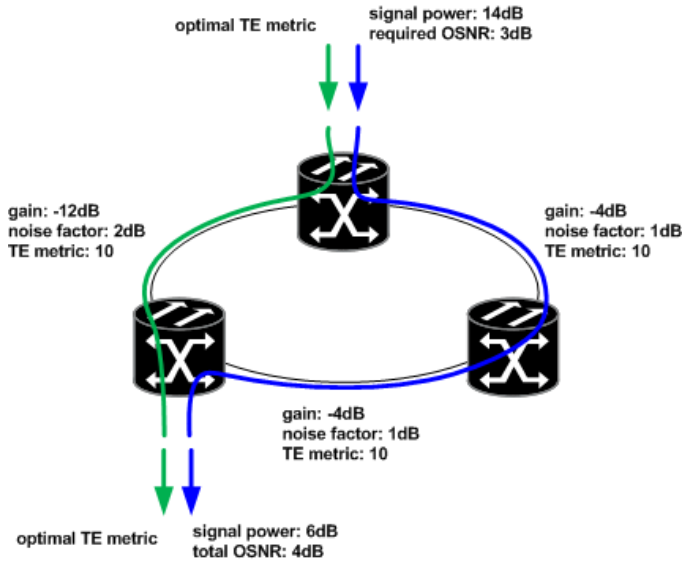
- Shortest path vs. physical layer impairment aware routing



Optical impairments

- Path computation with optical impairments
  - the objective is to compute an optimal path satisfying given set of constraints,
  - path computation algorithms enhancements
  - physical attributes depend on the direction within the optical link,
  - transmission parameters depend on wavelength

- Changes in the controller and adaptations in OF protocol required



Summary

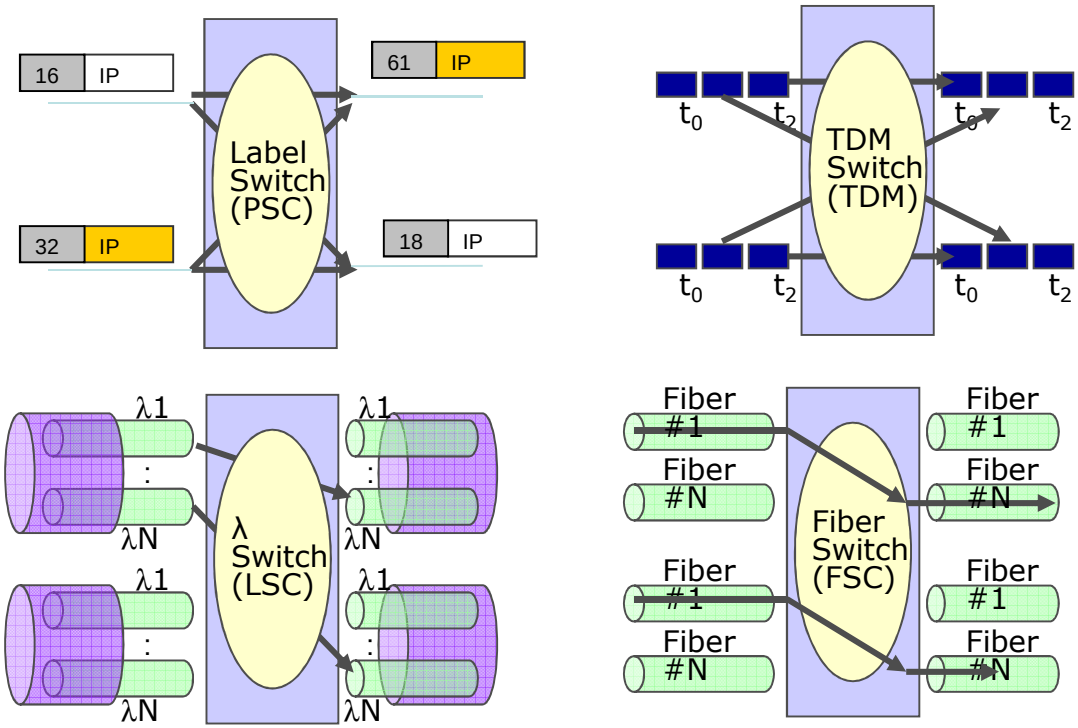
- Topology discovery
- Node structure and constraints
- Physical layer impairments
- Power equalization
- Impairment aware routing and wavelength assignment (IA-RWA)

Talk layout

- Software Defined Networking: A review
- Unified Control and Management
  - An OpenFlow approach
- Extended OpenFlow to support circuit switching
- Optical network considerations
- GMPLS control plane
- Interworking of OpenFlow and GMPLS
- OFELIA approach
- Conclusions

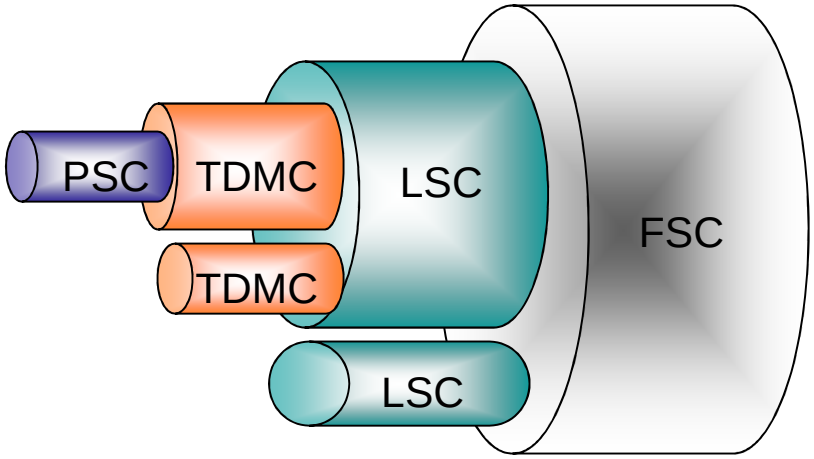
GMPLS

- Generalized Multi-protocol Label Switching
- Controls both packet-switching and non-packet-switching networks.
- Creates data-plane path called label-switched path (LSP) by instructing to program cross-connect at each hop.



Control interfaces

- Extend the MPLS to support more interfaces other than packet switch
  - Packet Switch Capable (PSC)
    - Router/ATM Switch/Frame Reply Switch
  - Time Division Multiplexing Capable (TDMC)
    - SONET/SDH ADM/Digital Cross connects
  - Lambda Switch Capable (LSC)
    - All Optical ADM or Optical Cross connects (OXC)
  - Fiber-Switch Capable (FSC)
- LSPs of different interfaces can be nested inside another

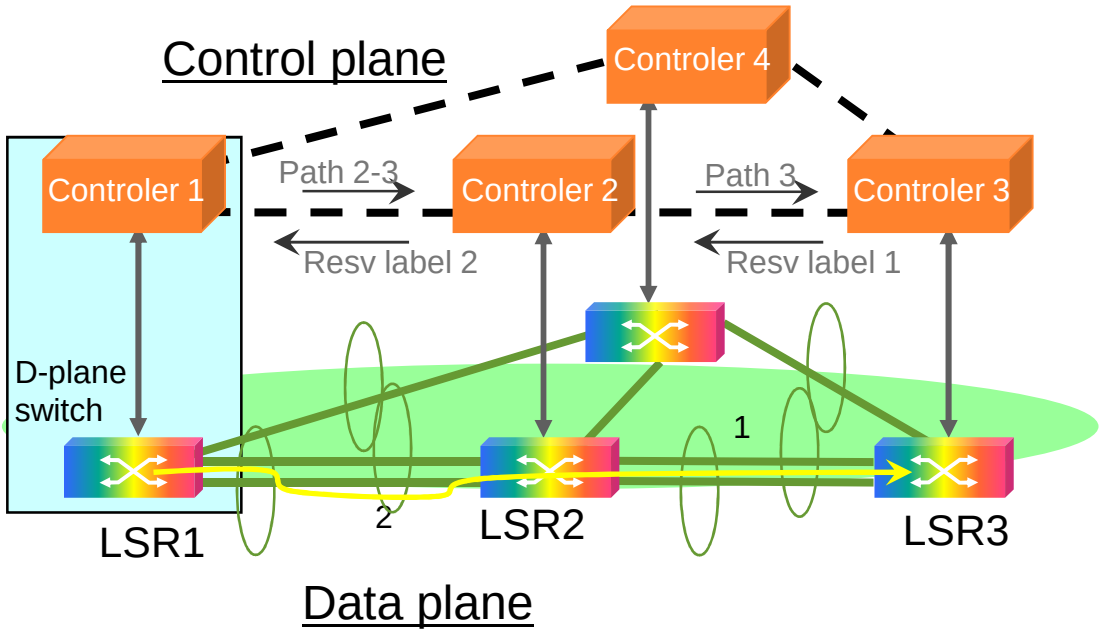


Challenges

- Routing challenges
  - Limited number of labels
  - Very large number of links
    - Link identification will be a big problem
    - Scalability of the Link state protocol
    - Port connection detection
- Signaling challenges
  - Long label setup time
  - Bi-directional LSPs setup
- Management challenges
  - Failure detection
  - Failure protection and restoration

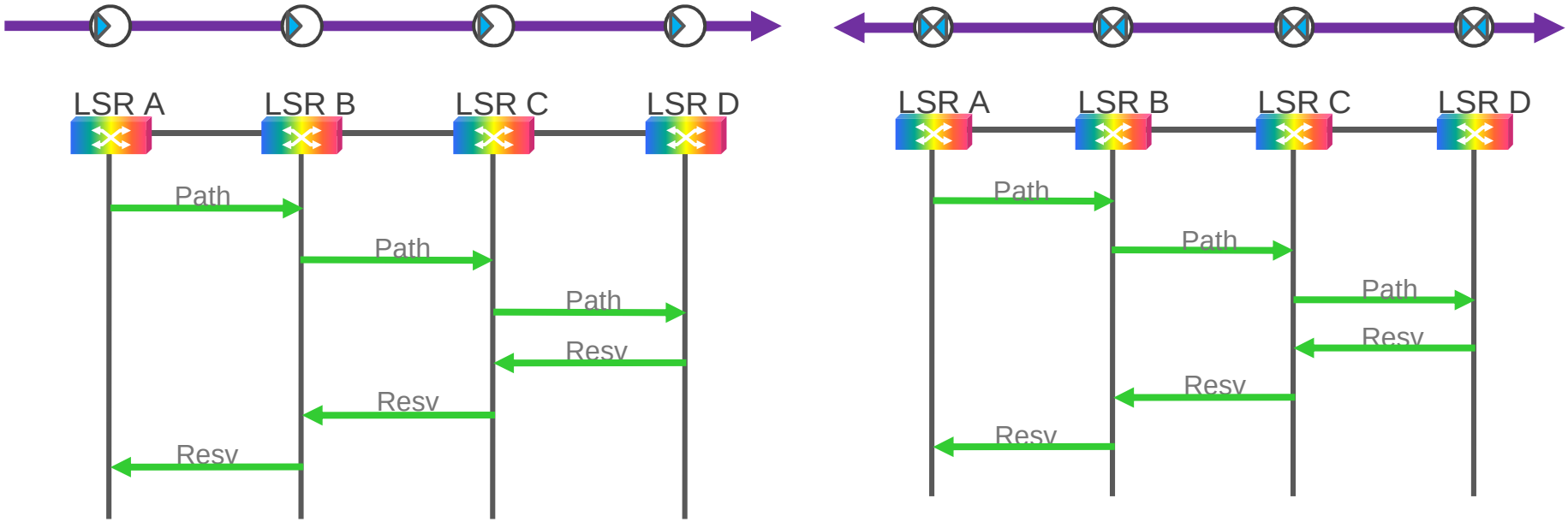
GMPLS for circuit switching networks

- Control-plane is (usually) a separate IP network (DCN).
- Signaling protocol (e.g., RSVP-TE) runs in control-plane to establish LSP in data-plane.



Signaling protocol: RSVP-TE

- **RSVP-TE** is the signaling protocol for GMPLS.
  - Path msg. from ingress to egress
  - Resv msg. from egress to ingress
- **Bi-directional LSPs**: Both forward and backward LSPs are created in data plane by a single signaling exchange.



When is it safe to send data?

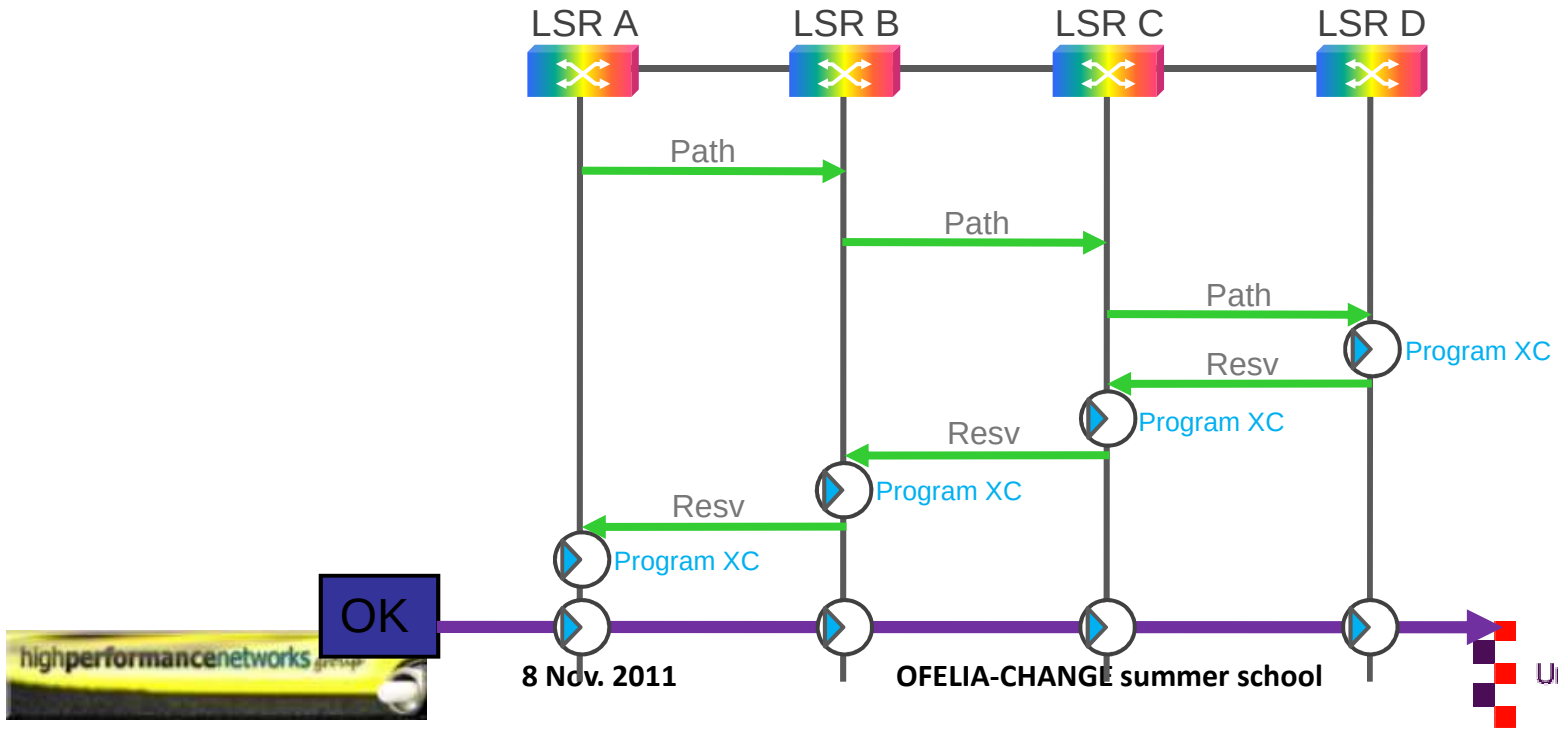
- Cross-connect in data plane is set up under control of the signaling protocol.
- Cannot send data before all cross-connects are correctly set up.
- Is it OK to start sending data when the control plane completes its message exchange?

What does it mean to be "*safe to send data*"?

- Data that is sent should be delivered to the LSP egress
  - Reliable data delivery
- Data that is sent **must not** be delivered to the wrong egress
  - Security and confidentiality
- In optical networks there are **considerable safety concerns** about turning on lasers to **misconnected fibers**.

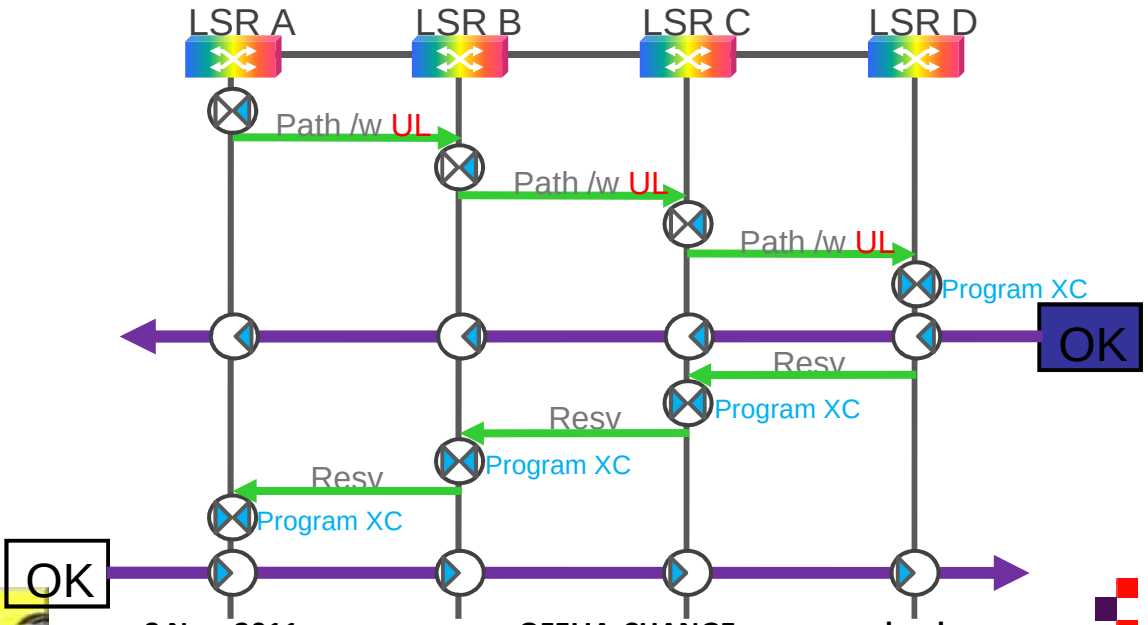
Fundamental message processing rule

- A **node** should program cross-connect before sending the **Resv** message to its upstream neighbor.
  - "The node then sends the new LABEL object as part of the Resv message to the previous hop. The node SHOULD be prepared to forward packets carrying the assigned label prior to sending the Resv message." [***RFC 3209, Section 4.1.1.1***]
- Thus the ingress LSR can safely start to send data when it receives the Resv message (and programs its own cross-connect).



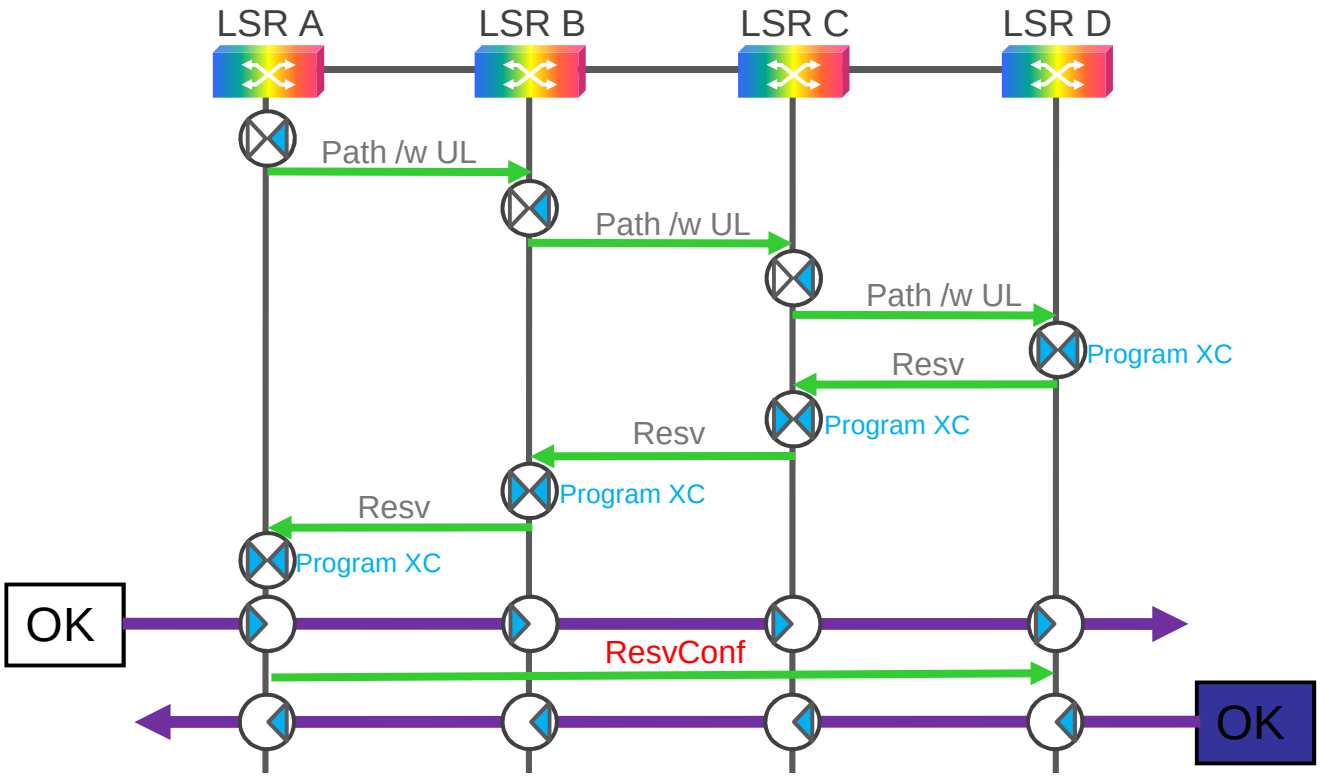
Notes for bi-directional LSPs (1/2)

- An LSR should program the cross-connect for the backward data path before it propagates the Path msg. with Upstream-label object
  - "… An intermediate node must also allocate a label on the outgoing interface and establish internal data paths before filling in an outgoing upstream label and propagating the Path message. …" [**RFC3473, Section 3.1**]
- Thus the egress LSR can safely start to send data when it receives the Path message (and programs its own cross-connect).
- As before, the ingress LSR can safely start to send data when it receives the Resv message.



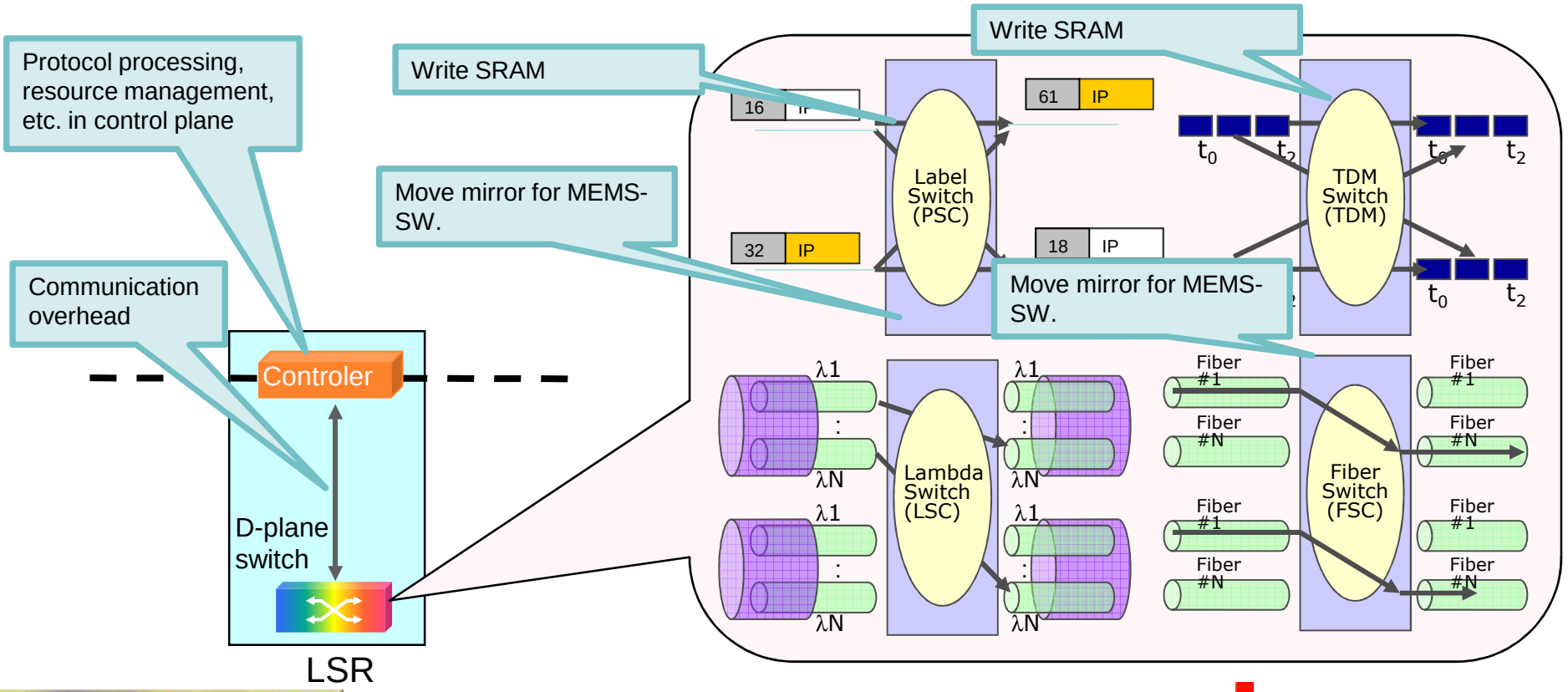
Notes for bi-directional LSPs (2/2)

- The ingress LSR MAY send a **ResvConf** msg. after it receives the Resv msg.
- The egress LSR could use this to learn that the ingress LSR is ready to receive data.
  - Some implementations don't support ResvConf
  - ResvConf is not reliably delivered



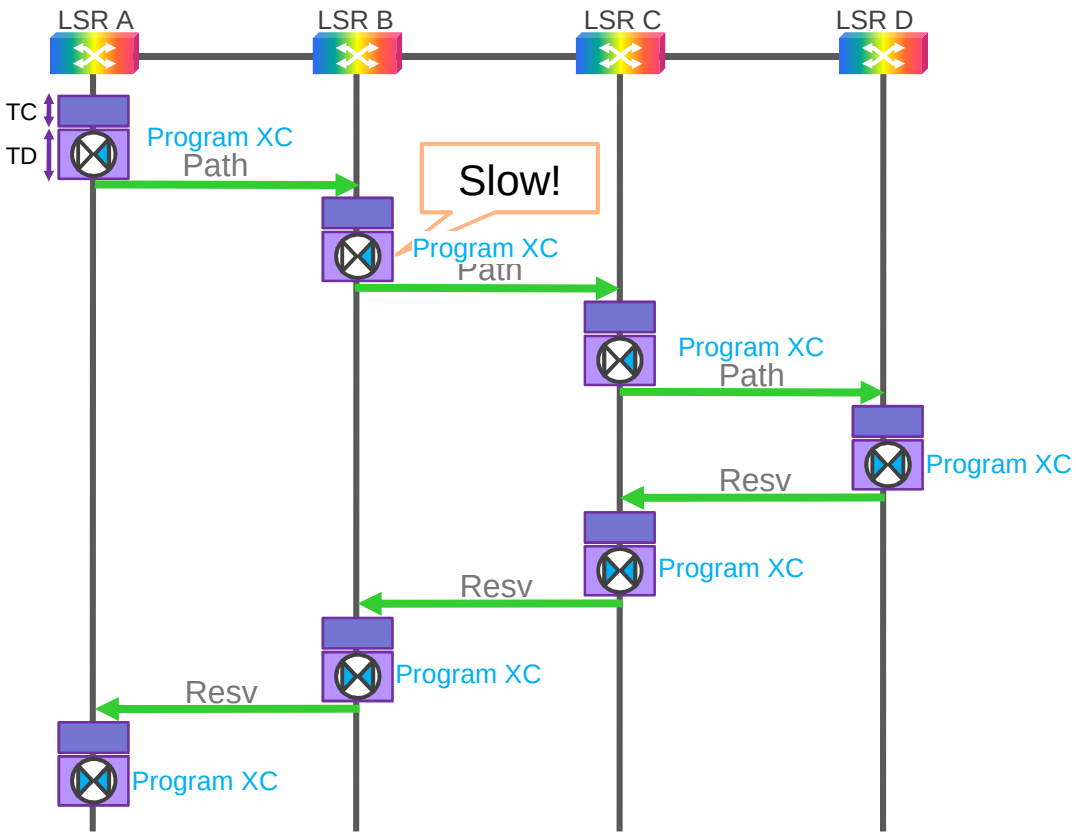
Implementation consideration

- After the label is decided in the **control plane**, the **LSR** programs cross-connects in data plane.
- The control plane processors can be physically distinct from the data plane cross-connects.
  - There is **a communication protocol** operating between the **control plane processor** and the **data plane** switch.
  - The successful completion of control plane signaling **cannot** necessarily be taken as evidence of correct data plane programming.
- How long it takes to finish programming cross-connects depends on the implementation and the data plane switch type: packet, TDM, lambda, and fiber.



Performance/operation implications

- **Control plane (TC)**
  - Protocol processing
  - Resource management
- **Data plane (TD)**
  - Communication overhead b/w controller and switch hardware
  - Programming cross-connects in switch hardware
- Some switch types or implementations can take too long to program cross-connects (TD too big)
  - Results in slow LSP setup



Summary

- Topology discovery and path calculation should be properly implemented in the extended OpenFlow controller.
- Optical network constraints (e.g., wavelength continuity, node constraints, switching constraints, physical impairments, power equalization) should be properly addressed. If these are left to the user application on top of the OpenFlow controller, then the **provisioning of a feasible lightpath will become almost impossible**.
- "**When is it safe to send data?**" and the "**Fundamental message processing rule**" should be properly respected/implemented.
- Many of the required functionalities have been already implemented in GMPLS control plane.

Talk layout

- Software Defined Networking: A review
- Unified Control and Management
  - An OpenFlow approach
- Extended OpenFlow to support circuit switching
- Optical network considerations
- GMPLS control plane
- Interworking of OpenFlow and GMPLS
- OFELIA approach
- Conclusions

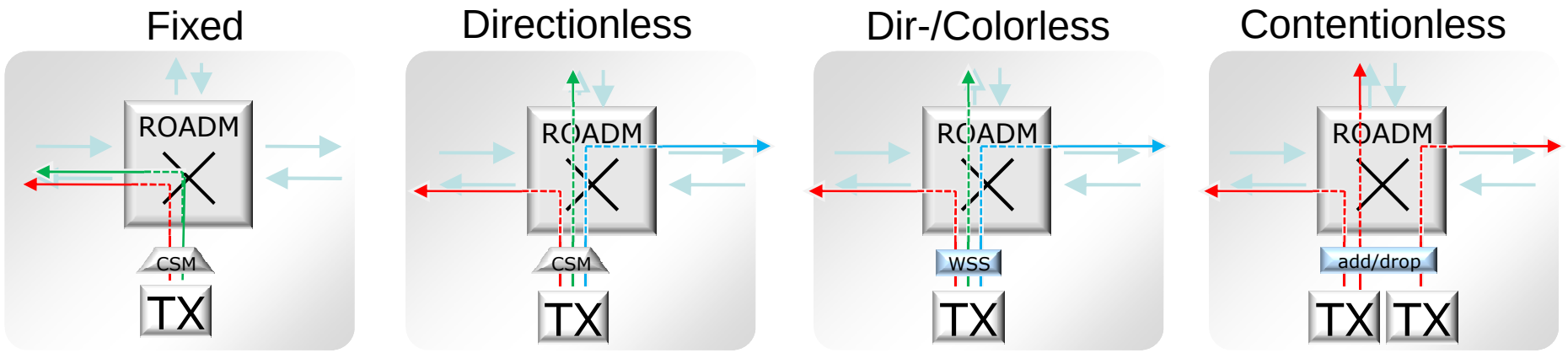
Requirements

- "**When is it safe to send data?**" and the "**Fundamental message processing rule**" should be properly respected/implemented.
- Required Optical Functionality
  - Modular optical nodes with switching constraints (colored/colorless, directed/undirected, blocking/contentionless add/drop)
  - Wavelength continuity → Routing and Wavelength Assignment
  - Physical impairments → Impairment aware path calculation (IA-RWA)
  - Optical power equalization

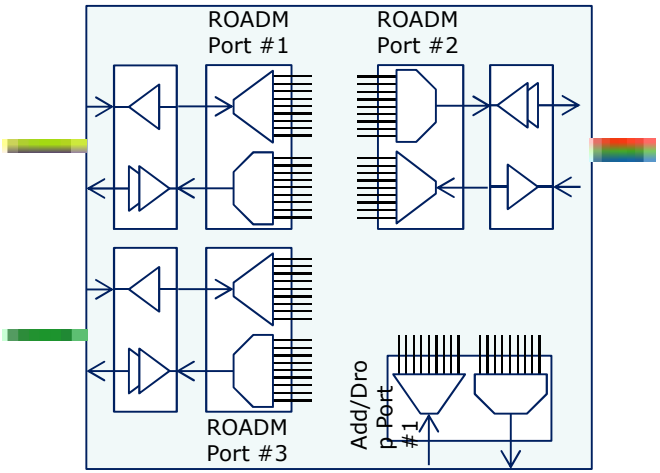
Which optical layer functions should be exposed to OpenFlow?
Optical power leveling, internal modular node structure, …

Many of the required functionalities are already implemented in GMPLS.
→ Should these functions be re-implemented or should a GMPLS interworking reuse these functions?

Optical Node & Switching Constraints



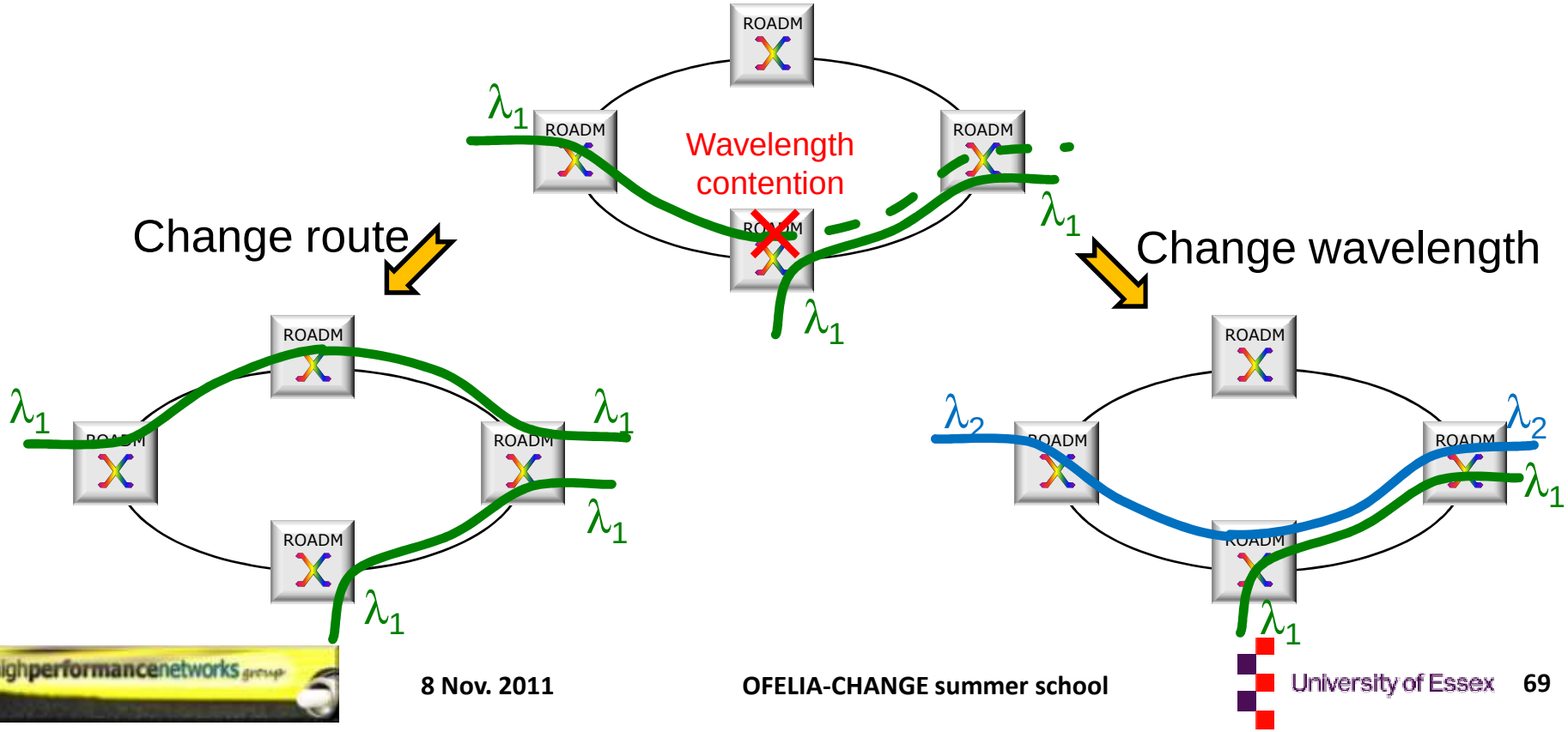
- Variety of different hardware configurations
  - FOADM (Fixed Optical Add/Drop Multiplexer), Regenerators,
  - ROADM (reconfigurable OADM): fixed, directionless, colorless, …
  - support for 40 and 80 channel grids (interleaved and non-interleaved), …
- Switching constraints
  - map of internal connections and dependencies in digested form,
  - updated per each operation in transport plane,
  - used (together with topology graph) as an input for path computation



Variety of network element with varying degree of flexibility and switching constraints must be supported by OpenFlow Controller & Protocol

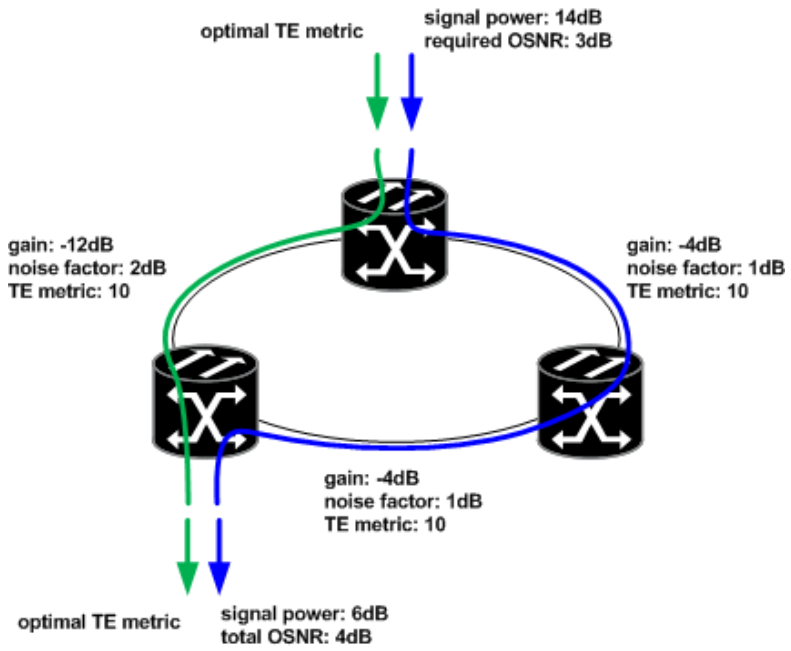
Routing and wavelength assignment

- Lightpath continuity must be guaranteed
  - ingress and egress with fixed ports,
  - ingress and egress with tunable ports,
  - more complex path computation for the systems with lamda conversions



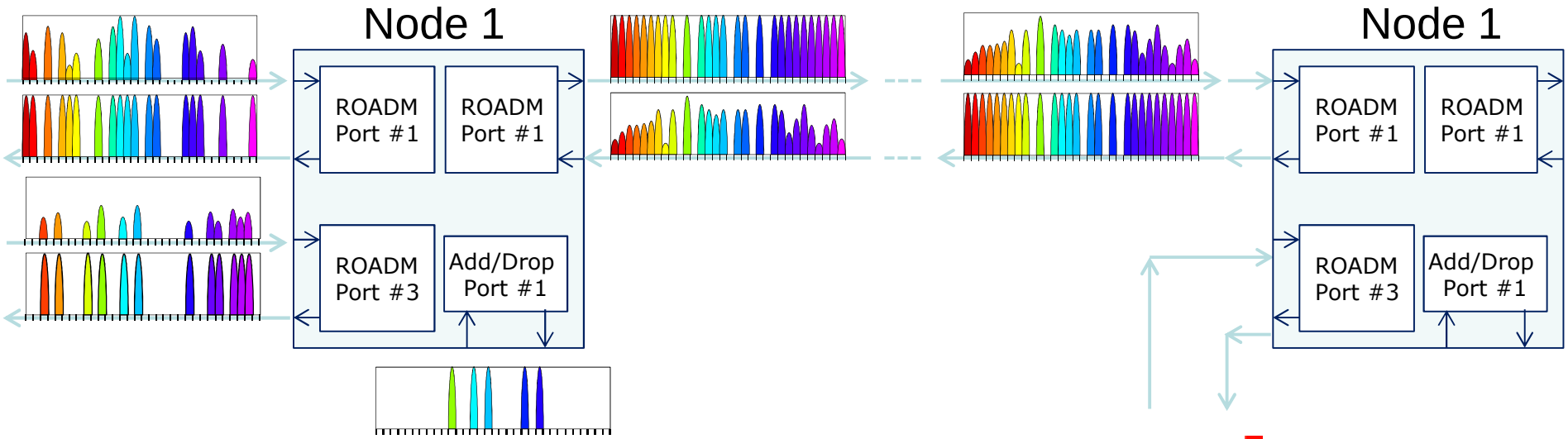
Impairment aware RWA

- Successful optical path setup depends on a variety of linear and non-linear optical performance parameters
- Parameters depend on wavelength and fiber allocation

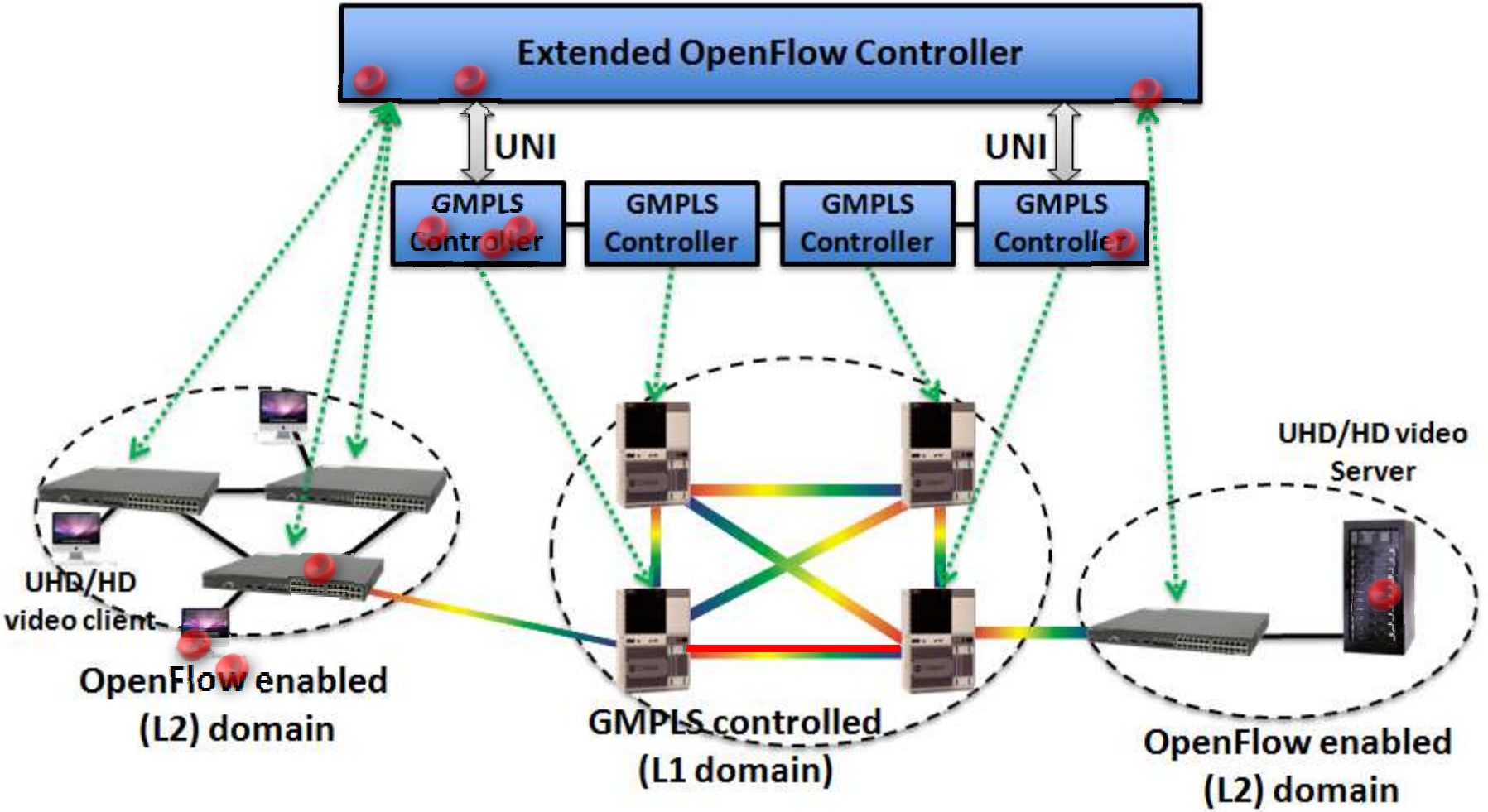


Optical Per-Channel Power Balancing

- Required for meshed optical networks with lightpath add/drop
- Bi-directional process, measures and adjust individual wavelength power values
- Triggered for each new lightpath to adjust the per-channel optical power in the whole spectrum on every passed ROADM / WSS
- For OpenFlow lambda switching, a hop-by-hop power balancing process must be synchronized with the vertical OpenFlow communication between the OpenFlow controller and the ROADMs along the lightpath



OpenFlow-GMPLS interworking



Talk layout

- Software Defined Networking: A review
- Unified Control and Management
  - An OpenFlow approach
- Extended OpenFlow to support circuit switching
- Optical network considerations
- GMPLS control plane
- Interworking of OpenFlow and GMPLS
- OFELIA approach
- Conclusions

OFELIA

OpenFlow in Europe – Linking Infrastructure and Applications

- The goal OFELIA project is to create **a unique experimental facility** that allows researchers to not only **experiment on a test network** but to **control the network itself** precisely and dynamically.
  - The OFELIA facility is based on **OpenFlow** that allows virtualizing and controlling the network environment through secure and standardized interfaces.
  - OFELIA belongs to the **second wave** of **FIRE** projects under FP7
    - FIRE: "Experimentally validating highly innovative and revolutionary ideas"

- EC contribution:€ 4,450,000
- Project start date: 1 October 2010
- Duration: 36 months (3 years)

European Center for Information and Communication Technologies EICT GmbH

i2cat FUNDACIÓ

NEC

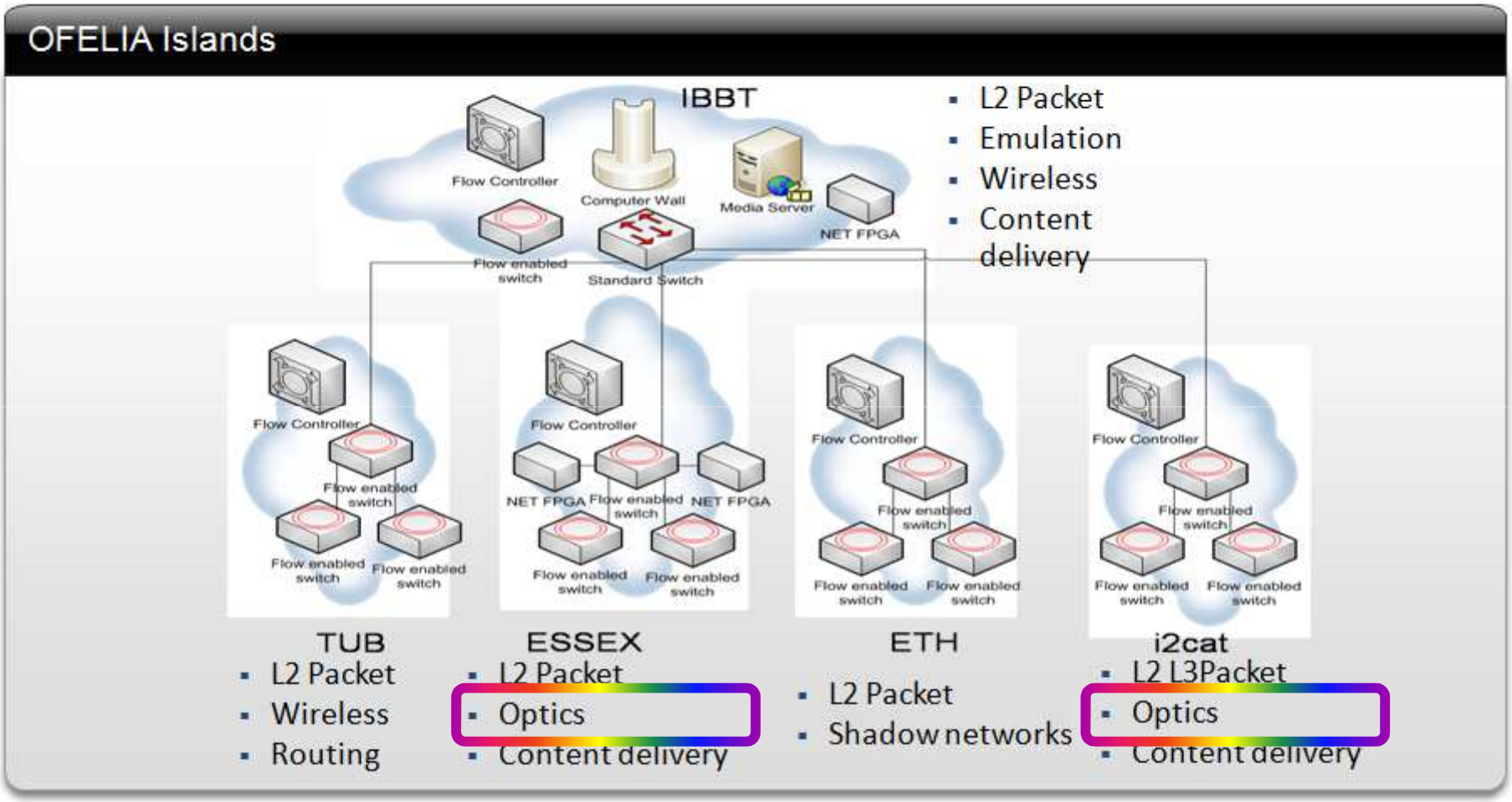
University of Essex

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

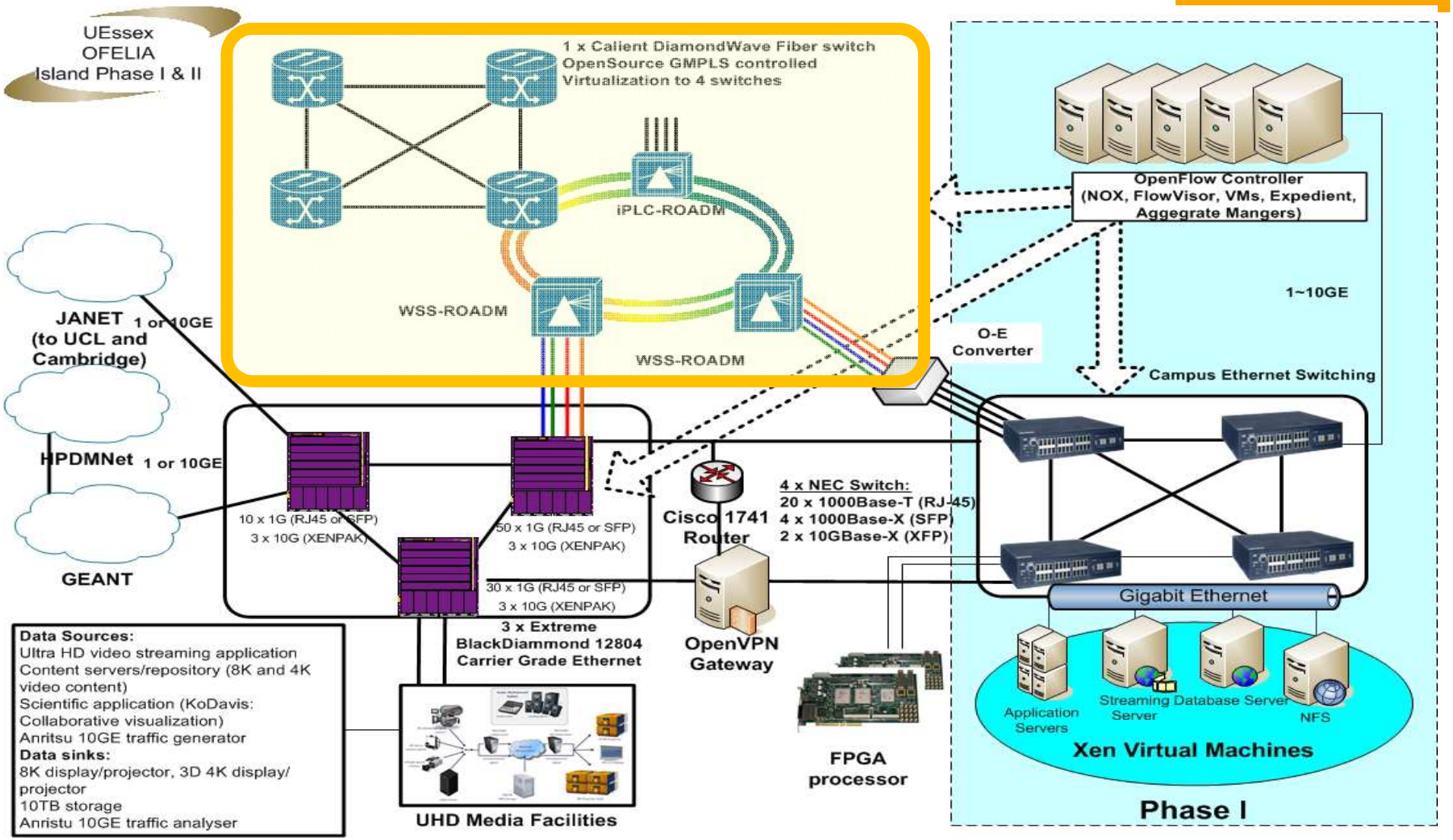
ibbt connect.innovate.create

ADVA Optical Networking

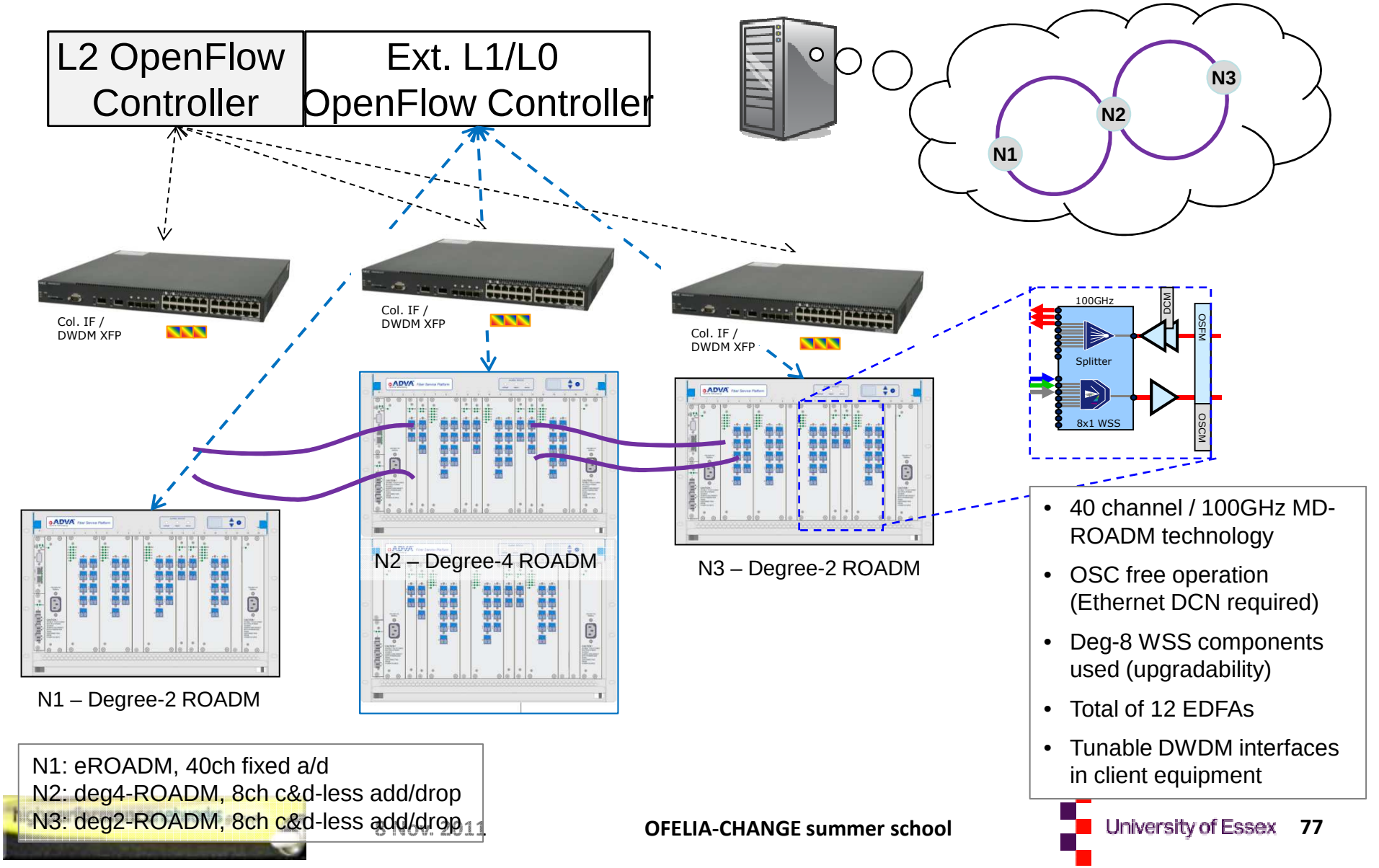
OFELIA islands



UEssex Island with Optical Network



OFELIA UEssex/ADVA Optical Testbed



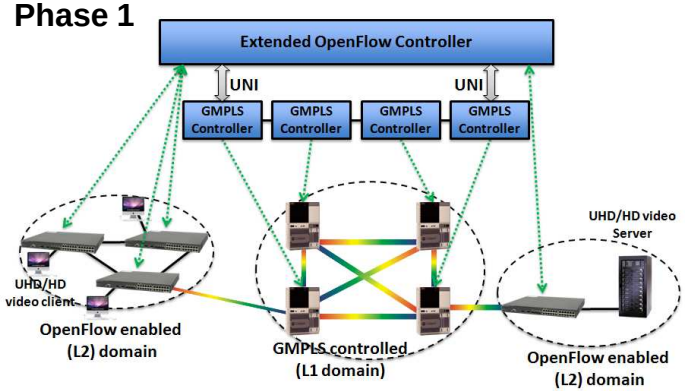
- 40 channel / 100GHz MD-ROADM technology
- OSC free operation (Ethernet DCN required)
- Deg-8 WSS components used (upgradability)
- Total of 12 EDFAs
- Tunable DWDM interfaces in client equipment

N1: eROADM, 40ch fixed a/d
N2: deg4-ROADM, 8ch c&d-less add/drop
N3: deg2-ROADM, 8ch c&d-less add/drop

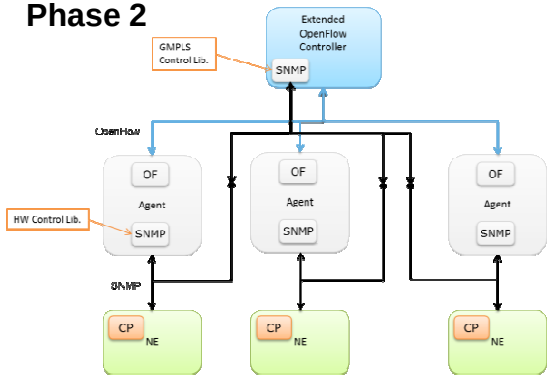
OFELIA Solution Approaches

- OFELIA Phase 1
  - Interworking with GMPLS Control Plane via UNI
- OFELIA Phase 2
  - OpenFlow controlled optical layer with loose / explicit path setup
  - OpenFlow agent in each optical NE
  - Reuse of GMPLS Control Plane functionality path setup & power leveling
- OFELIA Phase 3
  - Exposing of optical constraints to OpenFlow controller to support advance software defined optical networking (e.g. RWA, impairment aware path calculation, …)

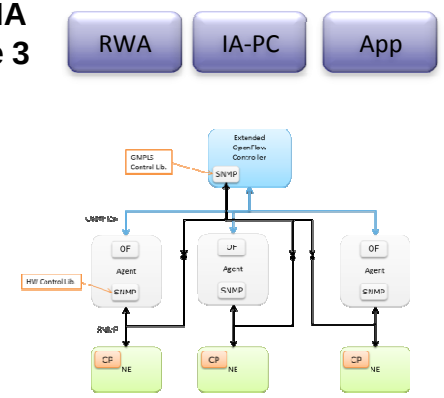
OFELIA Phase 1



OFELIA Phase 2

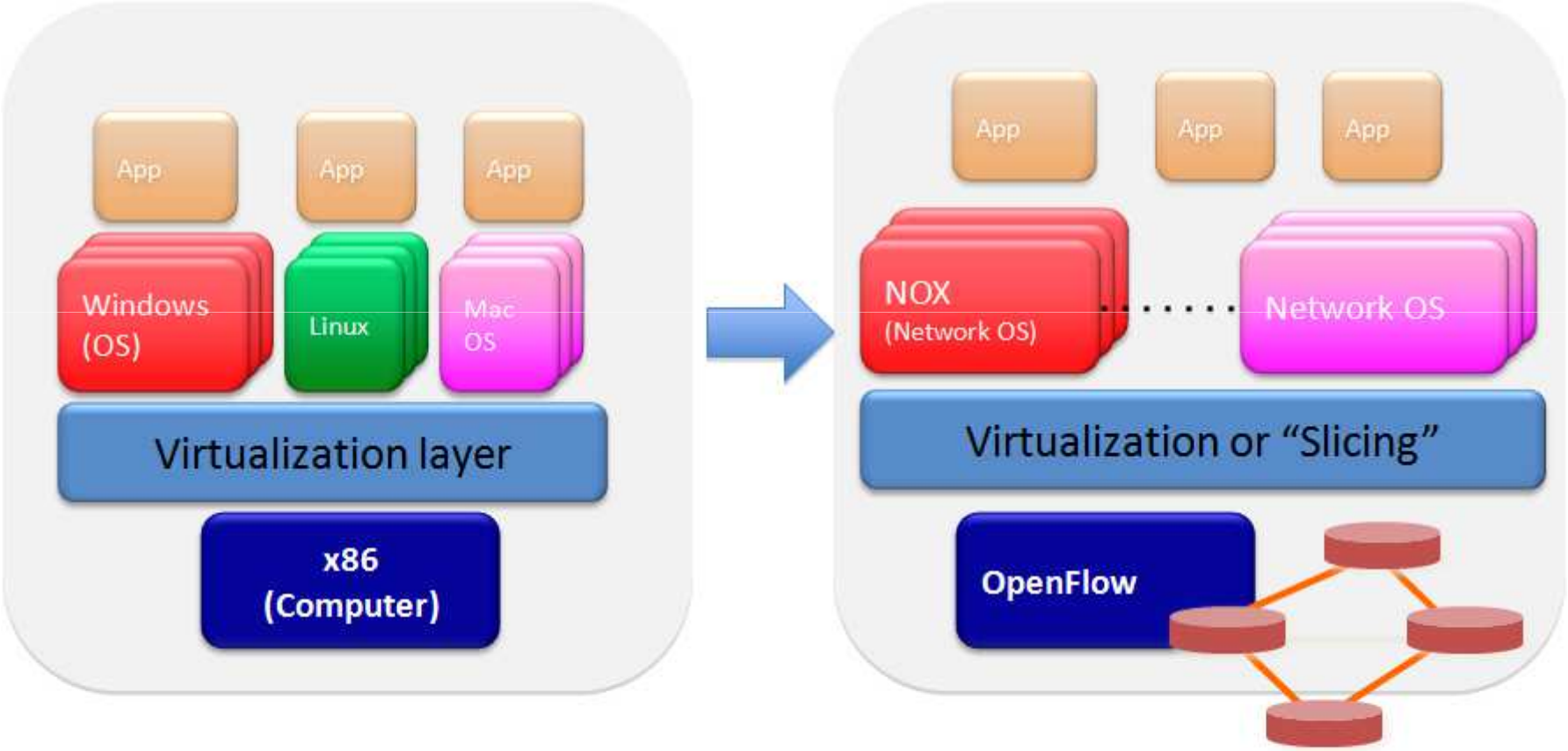


OFELIA Phase 3

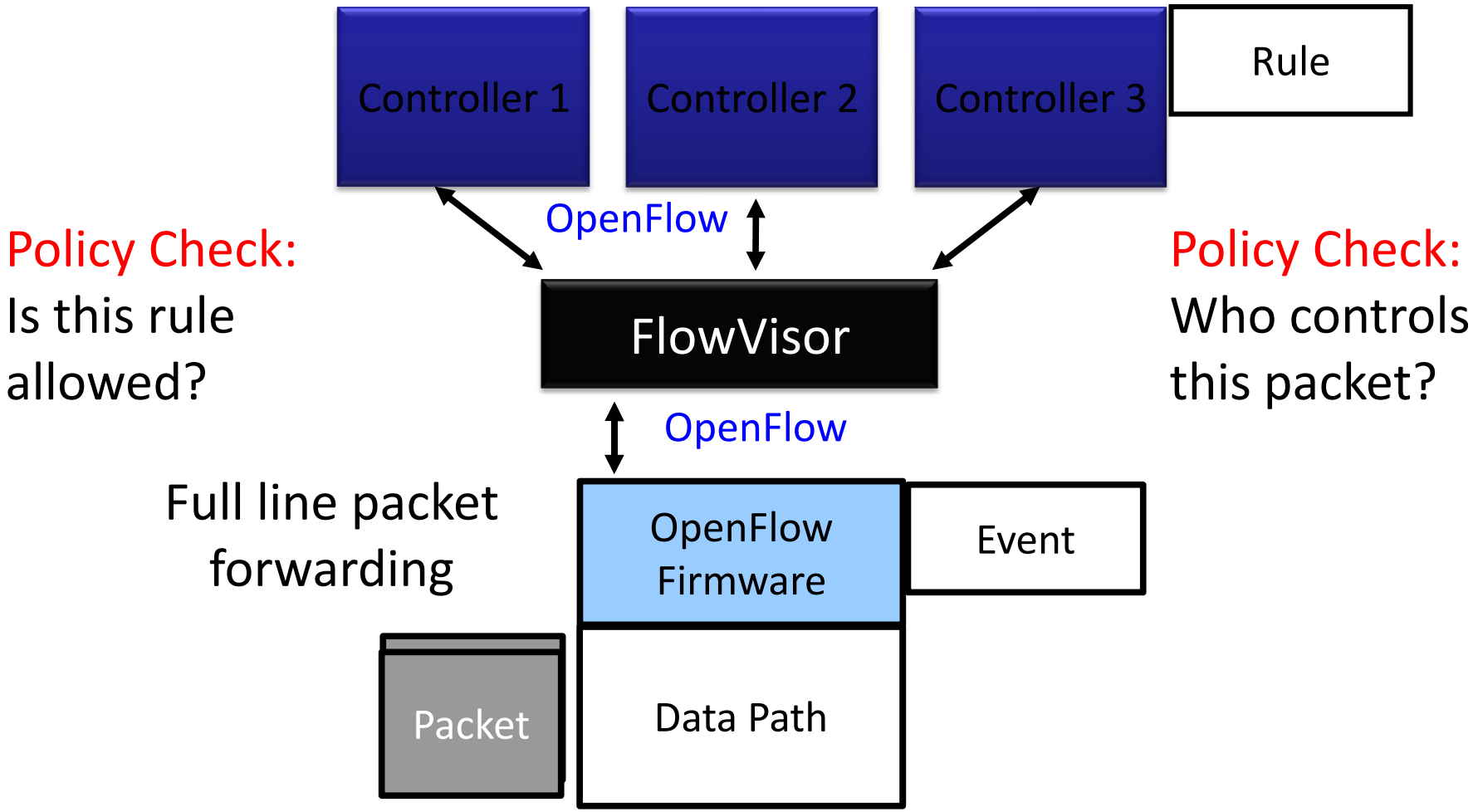


OpenFlow virtualization

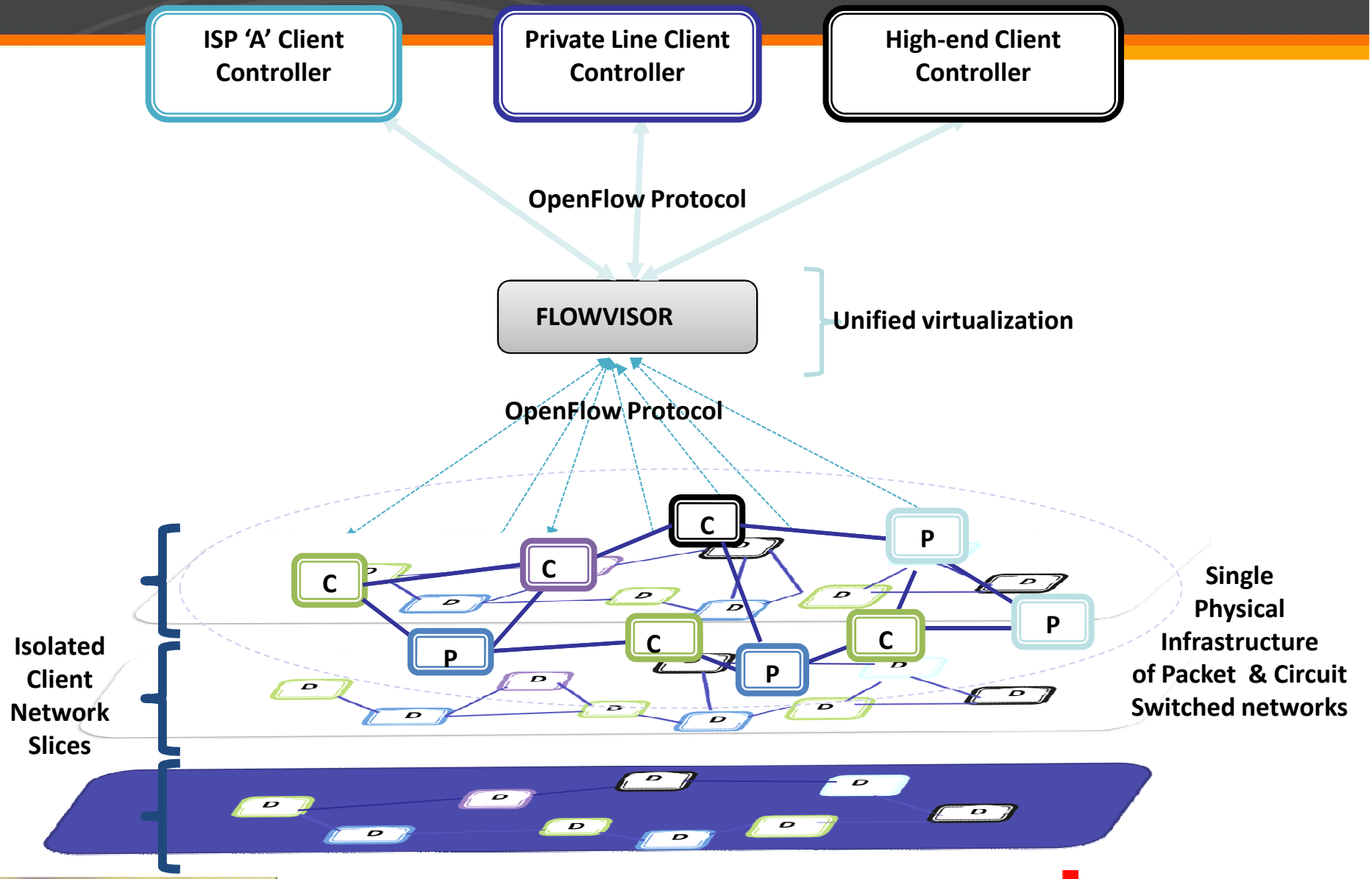
- Another inspiration from “Computing industry”



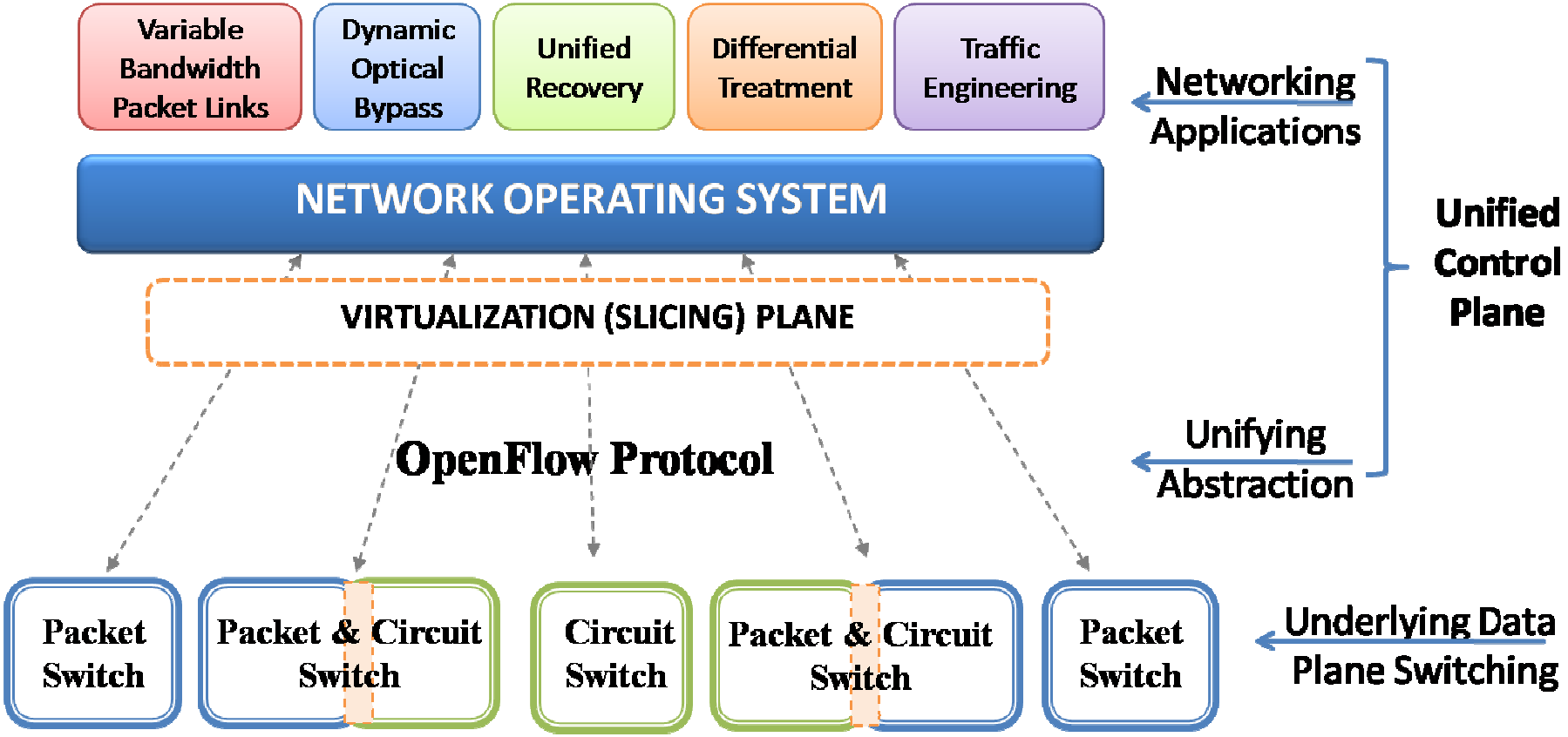
FlowVisor: OpenFlow Virtualizer



Optical Network virtualization: An OpenFlow approach

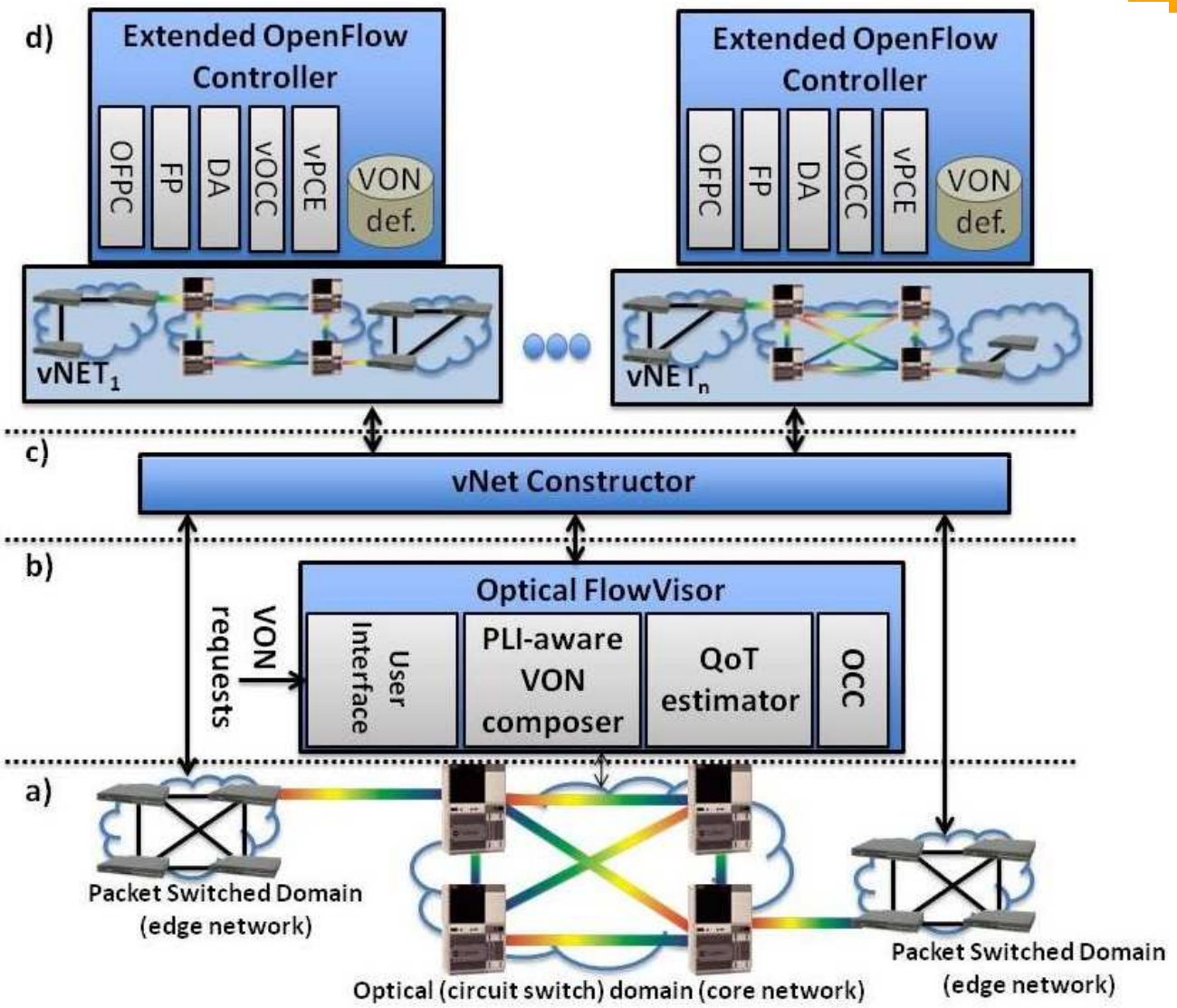


Unified network virtualization



Source:http://www.openflow.org/wk/index.php/PAC.C

Optical FlowVisor - Architecture



Conclusions

- OpenFlow as an early implementation of Software defined networking is extended to demonstrate the feasibility of shaping a unified control and management framework.
- There are special considerations for circuit switched networks and in particular optical circuit domain.
- Many of the functionalities are available in GMPLS, so the extended OpenFlow can utilize these functionalities either through interworking or full integration
- Three phase approach will be used in OFELIA to extend OpenFlow to support optical circuit switching.
- The optical FlowVisor can be used as a unified network virtualization mechanism

Thank you!

- Q & A