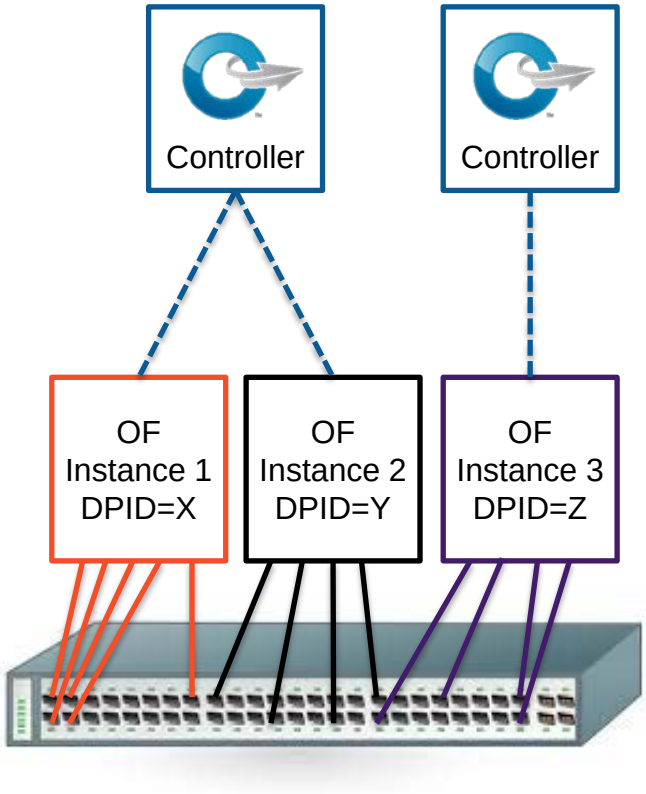
CLEMSON UNIVERSITY

OpenFlow Port Proxy Design for use in GTS

Ryan Izard
rizard@g.clemson.edu

Background

- Network switches have physical port numbers
  - e.g. 1, 2, …, 47, 48
- OpenFlow (OF) instances on a physical OF switch partition the switch into many logical OF switches
  - OF instance exposed to controller as unique data path ID
  - Physical ports set/configured for each OF instance



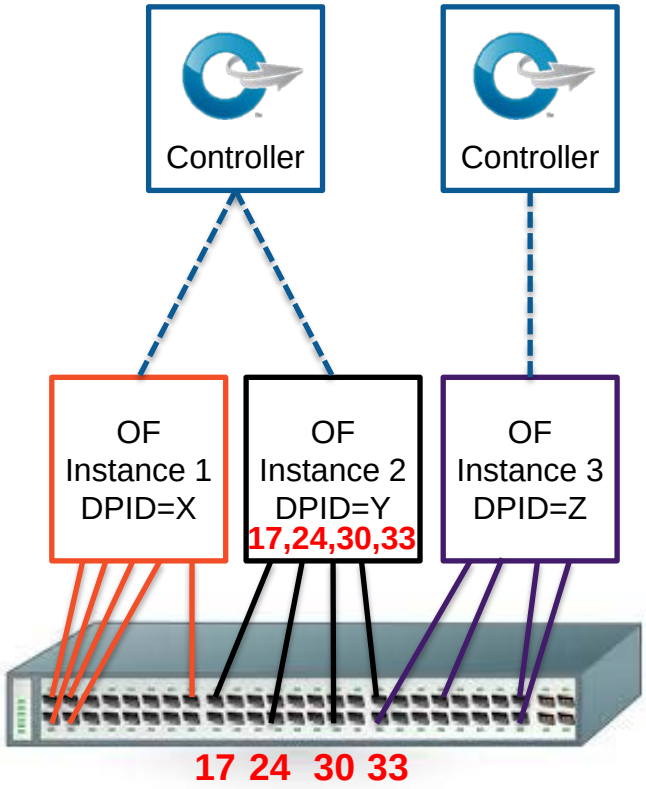
CLEMSON UNIVERSITY

Problem

How should the physical switch *expose* the port numbers *to the controller* for each OF instance?

The Simple Approach

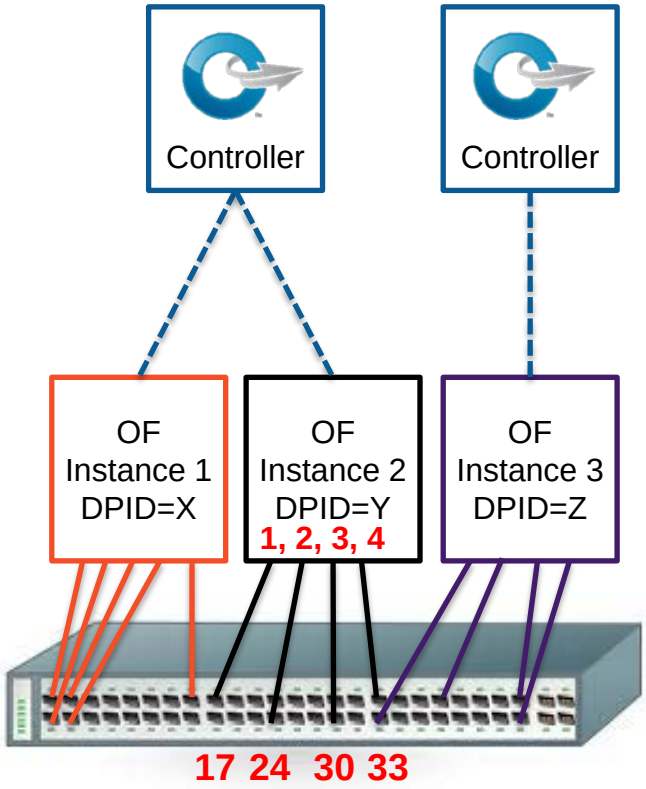
- Make each OF instance appear as the actual, physical partition of the physical switch
  - Port numbers exposed to controller as physical ports themselves
  - e.g. 17, 24, 30, 33



The More Conscientious Approach

- Make each OF instance appear as an independent switch
  - Port numbers exposed to controller as *constant sequence of ports*
  - e.g. 1, 2, 3, 4
  - Physical ports mapped to virtual ports

phys		virt
17	<-->	1
24	<-->	2
30	<-->	3
33	<-->	4



CLEMSON UNIVERSITY

Why Mapping the Ports is Better

Use Physical Port IDs

- Controller “sees” real network configuration (useful for admins?)
- Inconsistent port numbers for switch users
- Requires user apps handle/learn port numbers

Use Virtual Port IDs

- User controller “sees” same switch representation for each experiment
- No need for user to handle variable port numbers
- More portable user experiments/code

CLEMSON UNIVERSITY

How to Implement Port Mapping?

- Have the switch vendor do it
  - Some switch vendors use constant, sequential port numbers for OF instances already (yay!)
  - …but others do not (aww)
  - Vendors not likely to willingly expose their OF SDK
  - Requires custom solution possibly per vendor, per model, and per firmware version
  - *Not practical long-term*

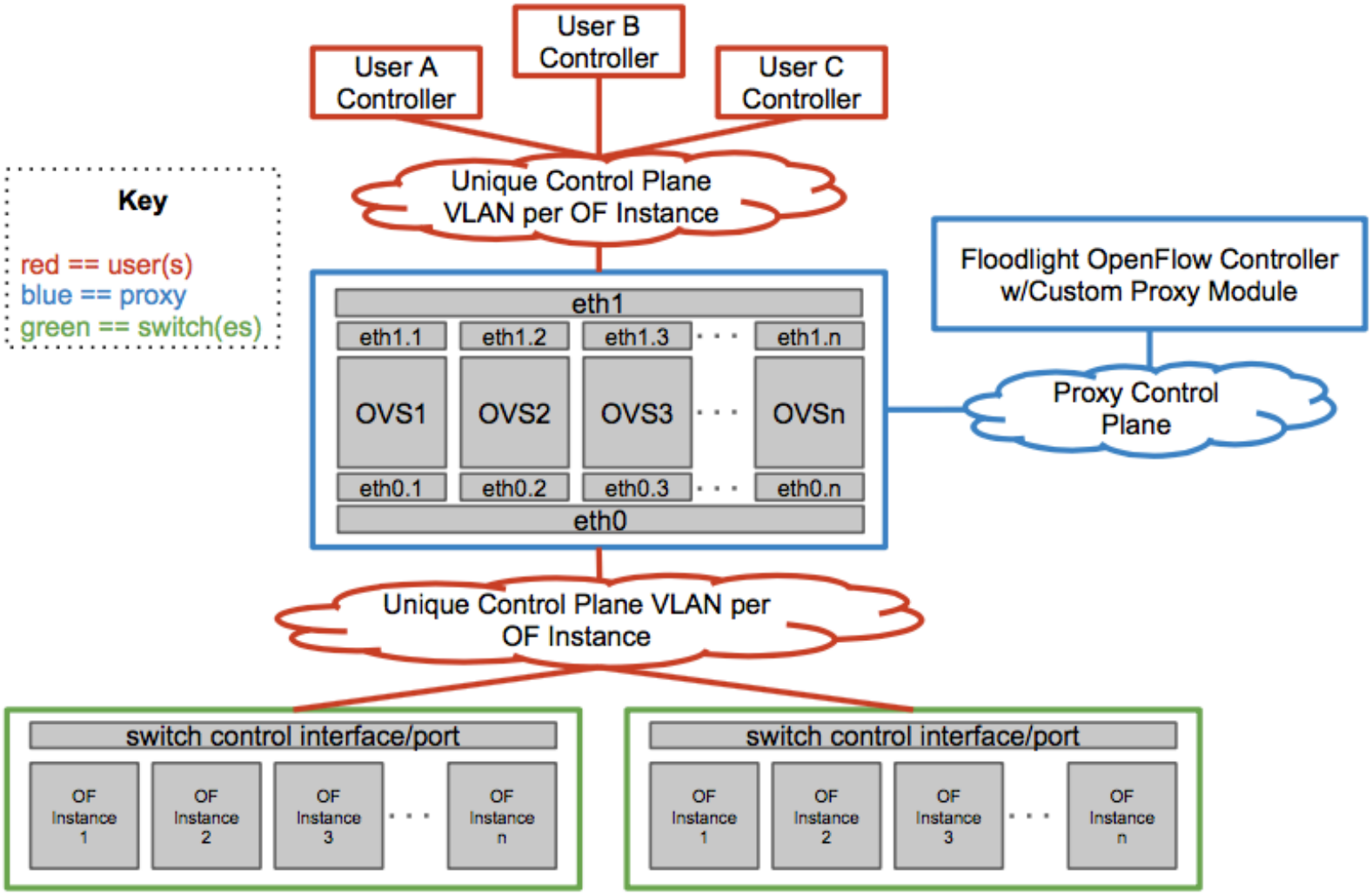
CLEMSON UNIVERSITY

How to Implement Port Mapping?

- Do it ourselves
  1. Idea #1: Proxy talks to switch as a user controller; proxy talks to user controller as a switch
     - Perform mapping within
     - Hmm, sounds a lot like FlowVisor (minus port mapping)
     - FlowVisor only designed for OF 1.0; we need OF 1.3
  2. Idea #2: Use a transparent proxy to perform the mapping on the live switch-to-controller connection
     - Need to intercept each control packet and swap the ports
     - *Sounds like a good use-case for OpenFlow itself!*

CLEMSON UNIVERSITY

Proxy Design



CLEMSON UNIVERSITY

Proxy Design: Based on OpenFlow

- OF can manipulate other OF control packets
  - They're just TCP packets that happen to contain OF data
  - Where can we perform the rewrite to map the ports?
    - Send to application
    - Send to the controller
    - In the switch itself with experimenter match/actions
- Flexible rewrite location, *but chose controller for simplicity*
- Not limited to mapping port numbers
  - Can map *any* OpenFlow type
    - DPID, switch attributes, port MAC addresses, etc.
  - Can map like-features between OpenFlow versions

CLEMSON UNIVERSITY

Proxy Design: What Needs Mapping?

Flow mods aren't the only OpenFlow messages that contain switch port numbers:

- Flow mods
- Group mods
- Port configuration
- Queue configuration
- Flow removed
- Statistics requests/replies
- Multipart requests/replies
- Features reply
- Packet-in/out

CLEMSON UNIVERSITY

Proxy Design: Floodlight Controller

- Java-based OpenFlow controller
- Implement proxy as a Java module
- OpenFlow 1.0 - 1.4 support
- OpenFlowJ-Loxi library
  - Facilitates high-level coding
  - Relatively simple to implement an OF version-agnostic solution

CLEMSON UNIVERSITY

Proxy Design: GTS Integration

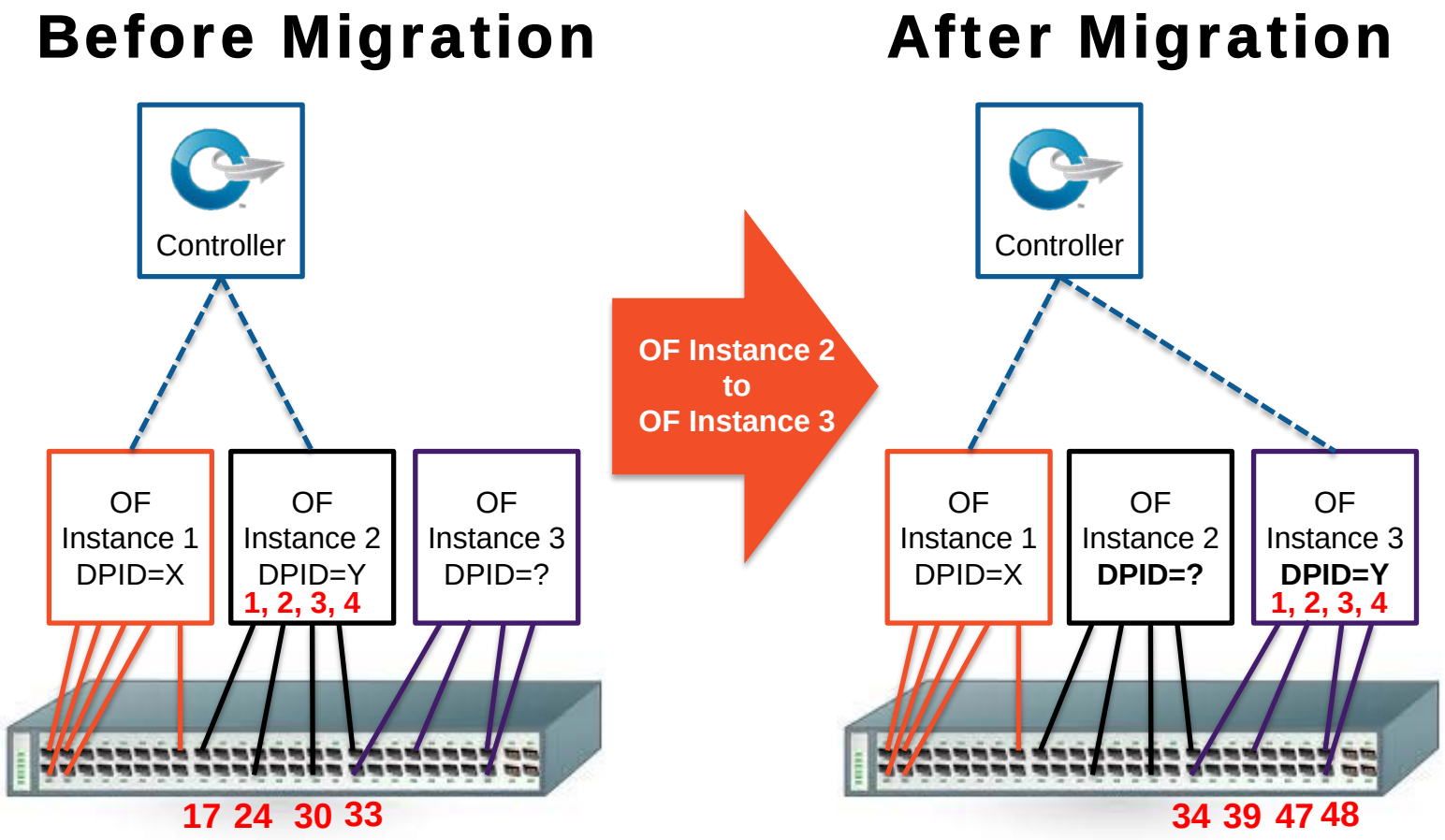
- One proxy per site/RCA
- RCA communicates with proxy
  - JSON over secure REST API
  - Informs of new connections, the port mappings, and other mappings to perform (such as DPID if desired)
- Differentiate user control connections by their VLAN IDs
  - Necessary b/c users can choose their own IP subnets

CLEMSON UNIVERSITY

Proxy Design: Migration Support

- Need to maintain virtual ports when moving a user's switch from one OF instance to another
  1. RCA issues proxy "snapshot" command to capture user flowspace for a particular switch; proxy provides flow information
  2. GTS determines new location for user's switch
  3. RCA at new location notifies proxy of new connection and of the flows to preinstall
  4. Proxy inserts flows w/new physical port mapping on the switch prior to allowing user controller to handshake
  5. User's controller reconnects; flows are logically equivalent (minus those with timeouts)
- Note GTS/RCA is responsible for maintaining a consistent port mapping with respect to the topology

Migration Example



CLEMSON UNIVERSITY

Demonstration