# OpenFlow MPLS and the Open Source Label Switched Router

James Kempf*, Scott Whyte†, Jonathan Ellithorpe‡, Peyman Kazemian‡,

Mart Haitjema**, Neda Beheshti*, Stephan Stuart†, Howard Green*

* ERICSSON

† Google

** Washington University in St. Louis

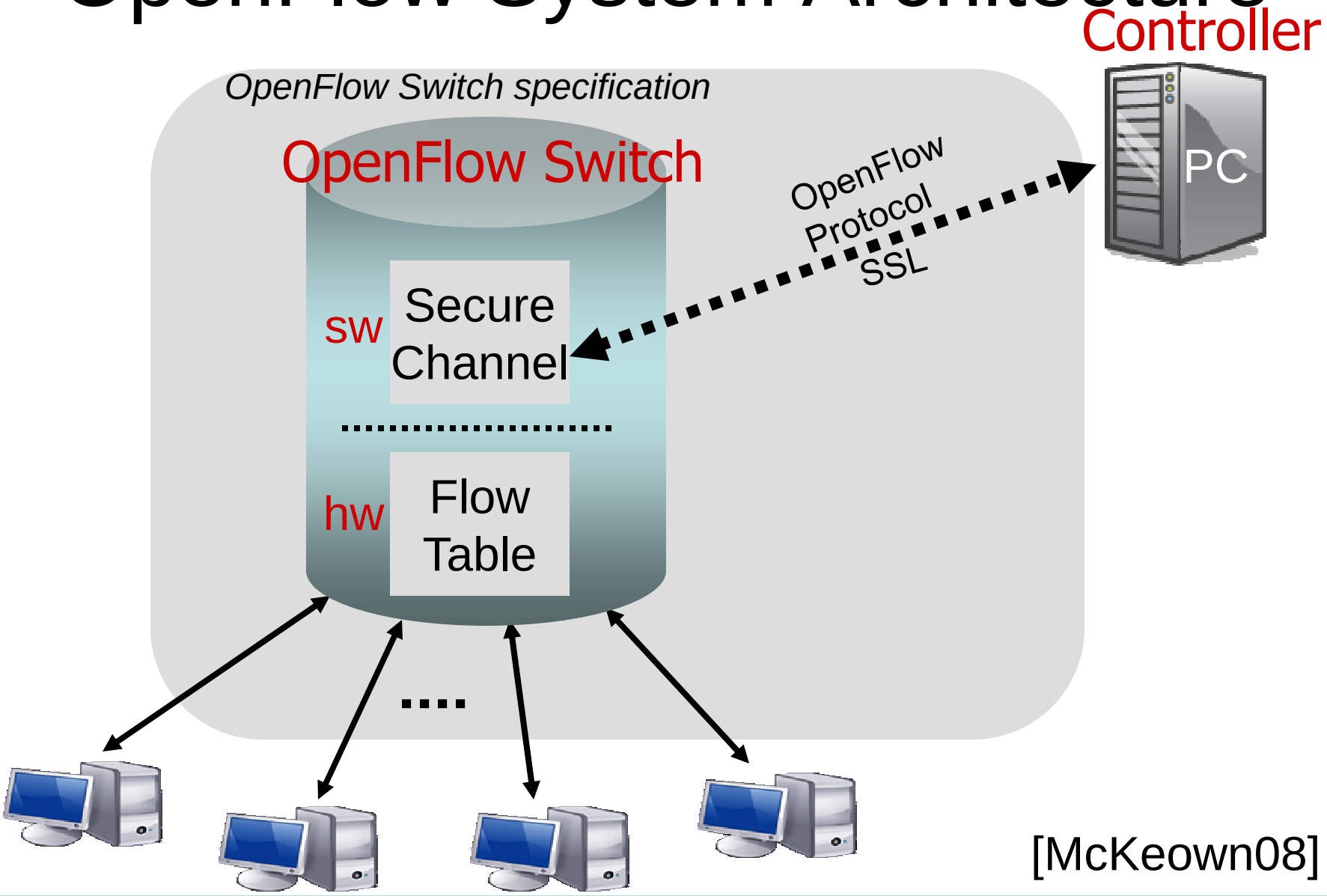
‡ STANFORD UNIVERSITY

# Outline

- OpenFlow Architecture
- OpenFlow MPLS Design
- NetFPGA Implementation
- The Open Source LSR
- Summary

# OpenFlow Architecture

# OpenFlow System Architecture




[McKeown08]

**Note: In the Open Source LSR, the controller and switch are in the same chassis!**

# OpenFlow Switch Architecture

OpenFlow Flow Table Entry


- **Rule:**
  - **Defines the flow**
  - **Currently a 10-tuple:**

[OFSpec]

| In Port | VLAN ID | Ethernet | | | IP | | | Tranport | |
|---|---|---|---|---|---|---|---|---|---|
| | | SA | DA | Type | SA | DA | Proto | Src | Dst |

- **Action:**
  - **Specifies what action the switch should take**
  - **Very simple actions such as:**
    - **Forward out port, send to controller, drop packet**
    - **Rewrite certain header fields (with fixed values)**
- **Stats:**
  - **Packet & Byte counters**

# OpenFlow MPLS Design

# OpenFlow MPLS Switch Architecture Changes

**Matching**

- Extend 10-tuple to include top 2 labels in label stack
- Most usage scenarios covered by 2 labels
- Label is 20-bit label (not exp, ttl, stack)

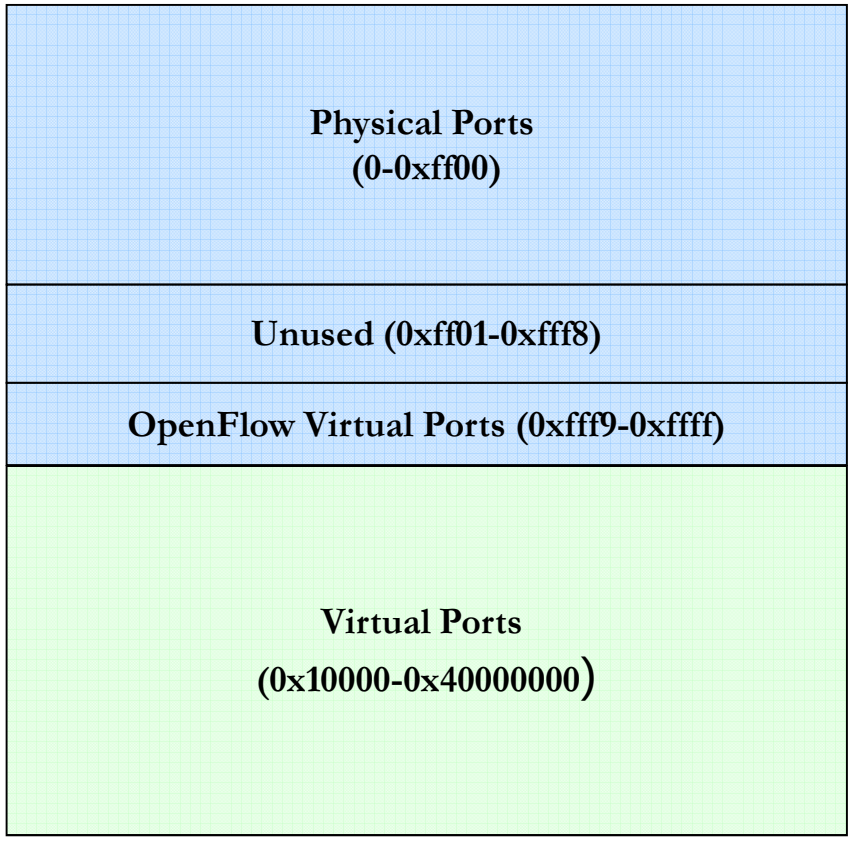
| In Port | VLAN ID | Ethernet | | | MPLS | | IP | | | Tranport | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SA | DA | Type | Label 2 | Label 1 | SA | DA | Proto | Src | Dst |

**Actions**

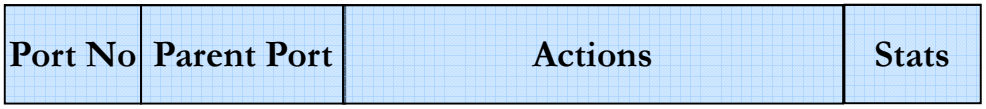
- Send packet to virtual port to perform complex protocol specific actions (i.e. MPLS pop, push, swap)
- Drop TTL expired packets & add mpls_ttl_exp stat

# Virtual Port Abstraction

- Virtual Ports
  - Extend notion of virtual ports
  - Virtual ports handle complex protocol/port specific actions
- Virtual Port Table
  - Similar to flow table
    - Virtual port number instead of 10-tuple
    - List of actions to perform
  - Parent port may be
    - physical port
    - another virtual port




Virtual Port Table




Virtual Port Table Entry

# Virtual Port Actions

**Virtual Port Table Actions:**

- *push_mpls* action
  - Push a 32-bit MPLS label onto top of label stack
  - Copies TTL and Exp bits from IP header if required
  - Parameters:
    - 20-bit label to push
    - 8-bit ttl value (optional)
    - 3-bit exp value (optional)
    - Flags (to indicate special behavior)

- *pop_mpls* action
  - Pop the top 32-bit MPLS label off the label stack
  - Copies TTL and Exp bits to IP header if required
  - Parameters
    - Eth_type (if popping last label)
    - Flags (to indicate special behavior)

# OpenFlow MPLS Changes

**Changes to Flow Table:**

- Output action
  - Port number now 32-bits
- Set_mpls_label action
  - Rewrite 20-bit label on top label
  - Similar to existing set_vlan_id action
- Set_mpls_exp action
  - Rewrite 3-bit exp field on top label
  - Similar to existing set_vlan_pcp action

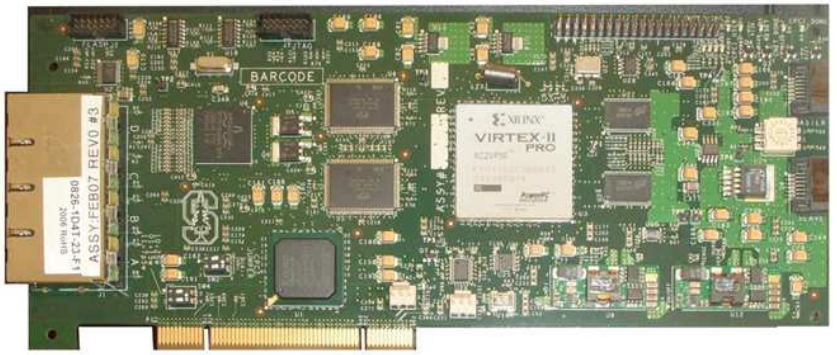
**New OpenFLow Messages**

- *vport_mod* message
  - Add or remove a virtual port table entry
  - Parameters:
- Virtual port number
    - Parent port number
    - Array of virtual port actions
- *vport_table_stats* message
  - Stats for virtual port table
  - Includes
    - *max_vports*
    - *active_vports*
    - *lookup_count*
    - *port_match_count*
    - *chain_match_count*
- port_stats message
  - Now also used for virtual ports
  - Only *tx_bytes* and *tx_packets* fields are used

# NetFPGA Implementation

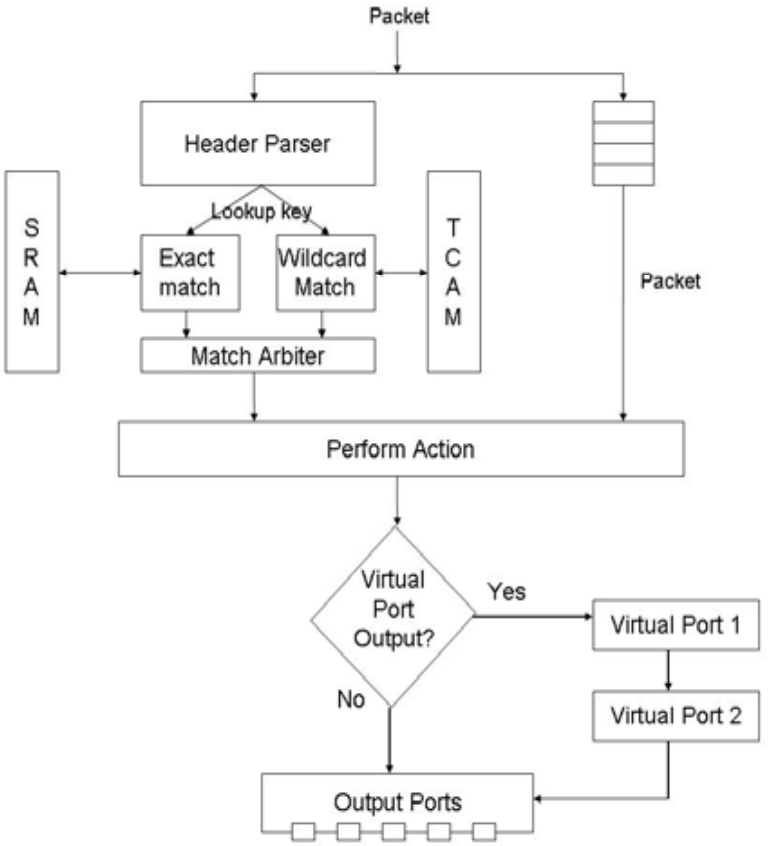
# NetFPGA Hardware

- 4 1G Ethernet ports
- Standard PCI form factor
- Xilinx Virtex-II Pro 50 FPGA
  - 53K logic cells
  - 4M block RAM
  - 700K distributed RAM
  - 2 Power PC cores
- Open source software



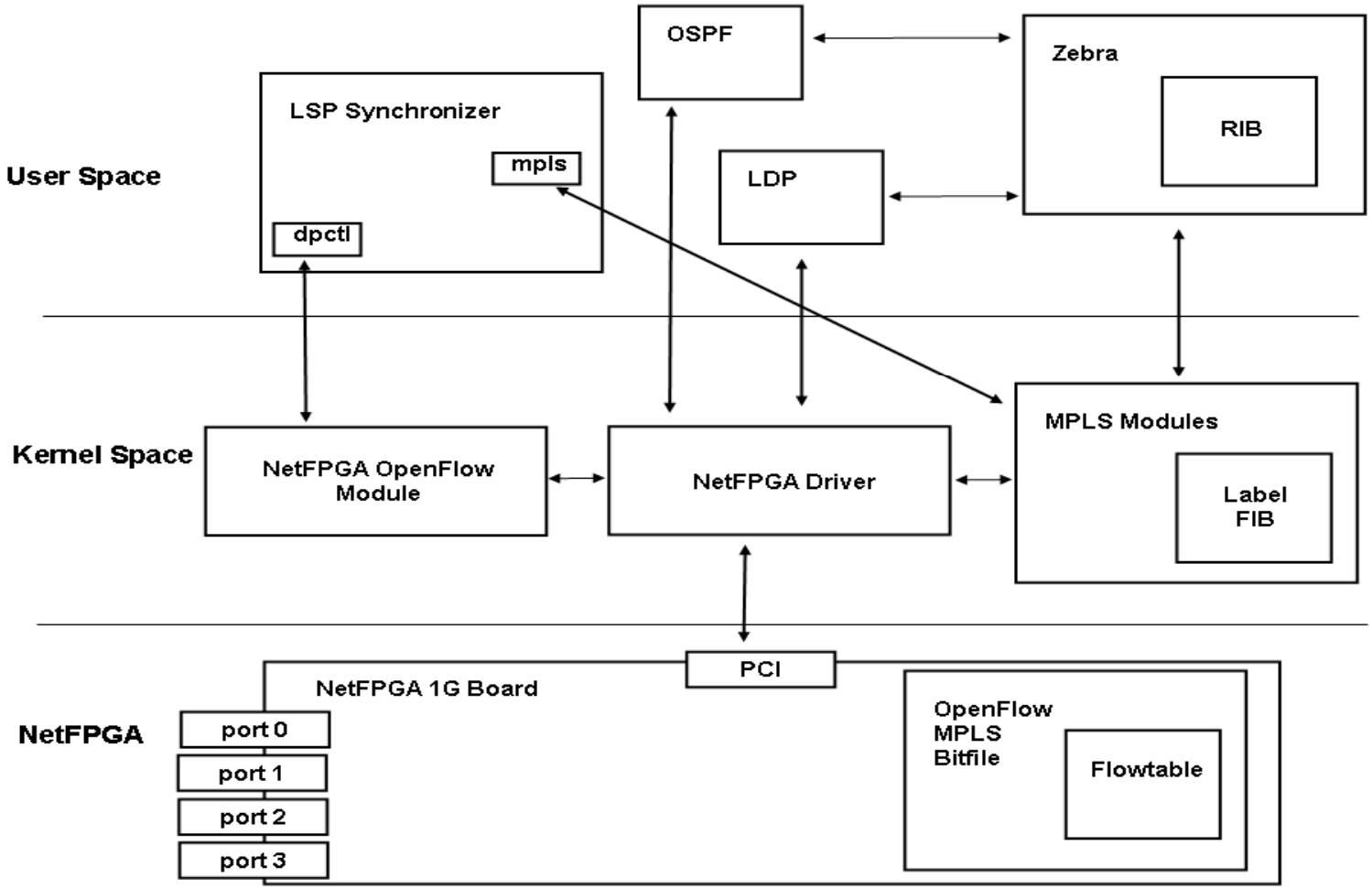
# OpenFlow MPLS NetFPGA Implementation

- Maximum two virtual port cascades.
  - Virtual ports must be of the same type
- 64 “push” virtual ports and 64 “pop” virtual ports
- Aggregated statistics on virtual port cascades
  - No per virtual port statistics

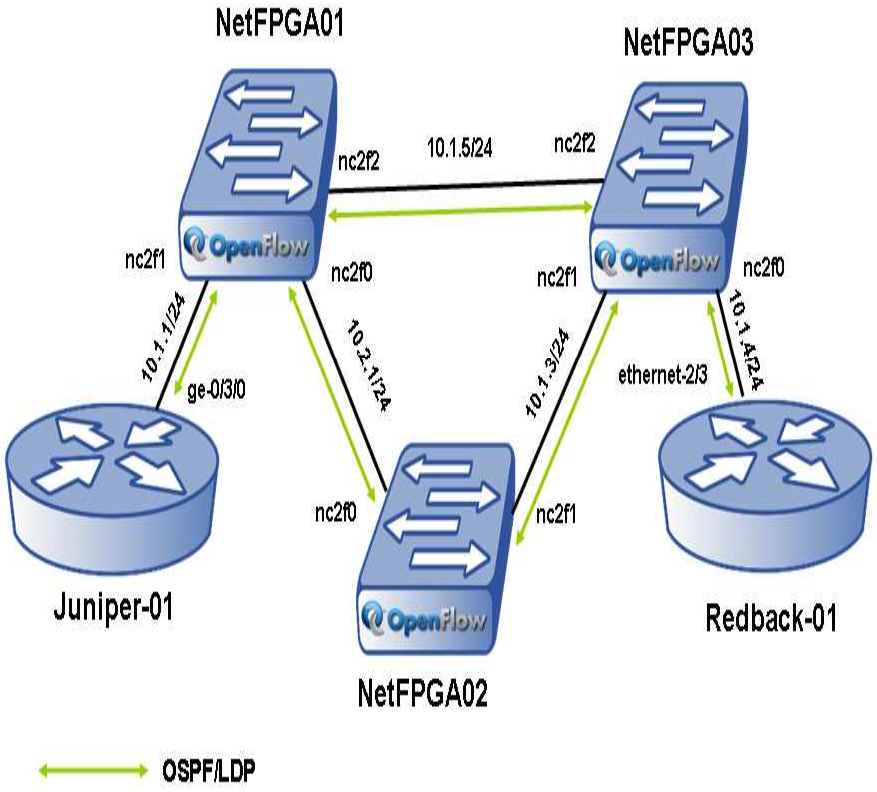


# The Open Source LSR

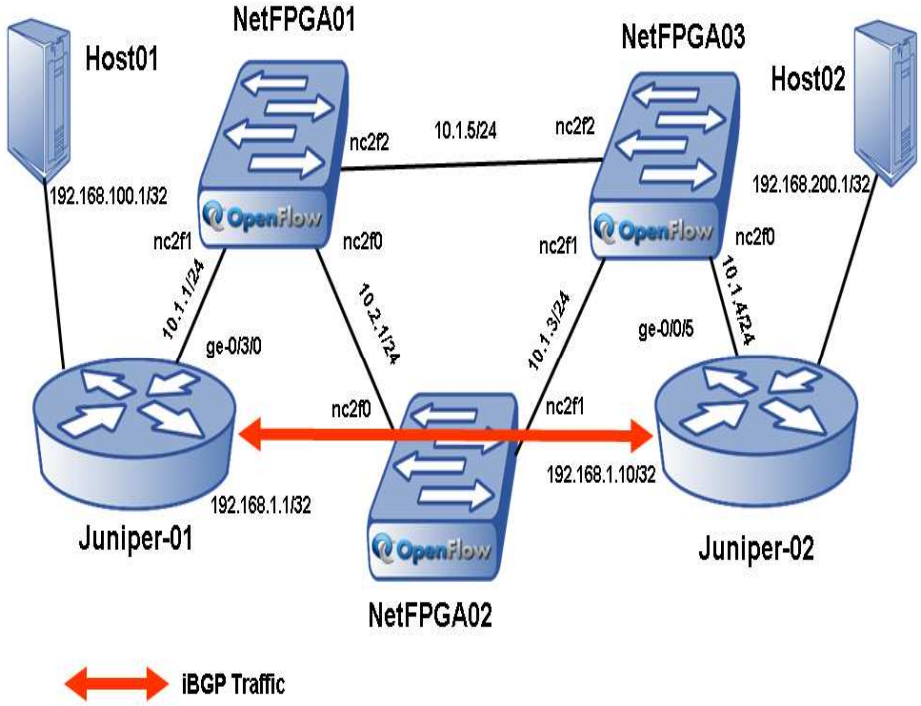
# Open Source LSR Block Diagram



# Interoperability Tests

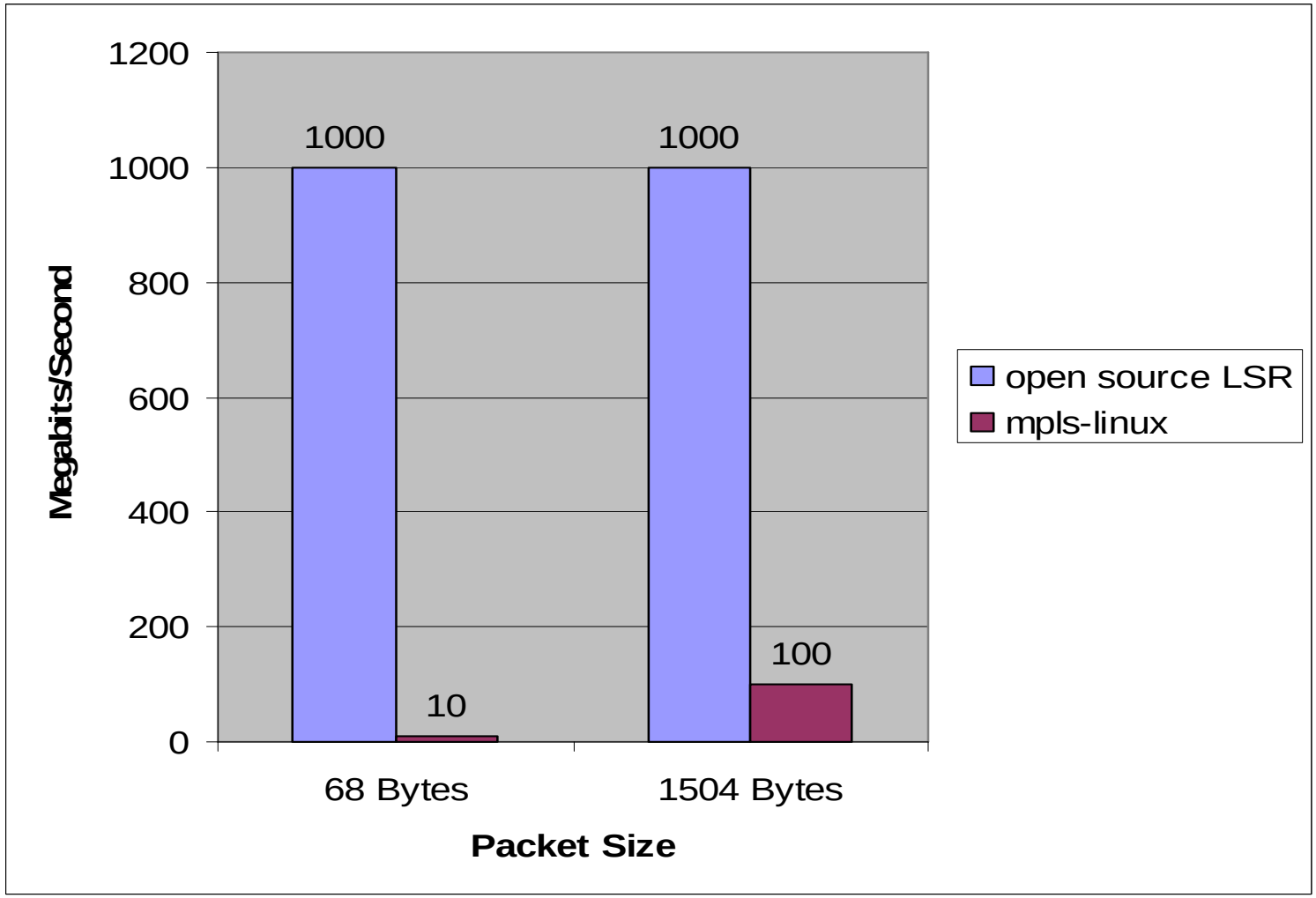



Basic Interoperability Network




Core Interoperability Network

# Comparative Forwarding Performance



# Summary

# Summary

- Extension of OpenFlow 1.0 switch model to include MPLS
  - Two labels
  - Virtual port abstraction for MPLS header operations
- Software and hardware implementation
  - OpenVSwitch and Stanford Reference Switch software implementations
  - NetFPGA hardware implementation
- Open source Label Switched Router (LSR) implementation
  - OpenFlow controller runs on the same chassis as switch
  - Software available at http://code.google.com/p/opensource-lsr
- Test results
  - Superior performance to a software implementation
  - Interoperates with commercial IP/MPLS routers