Introduction to NetFPGA and OpenFlow for Network Virtualization

Feb. 22, 2010

Sunwoong Choi
Kookmin University
(schoi@kookmin.ac.kr)

Contents

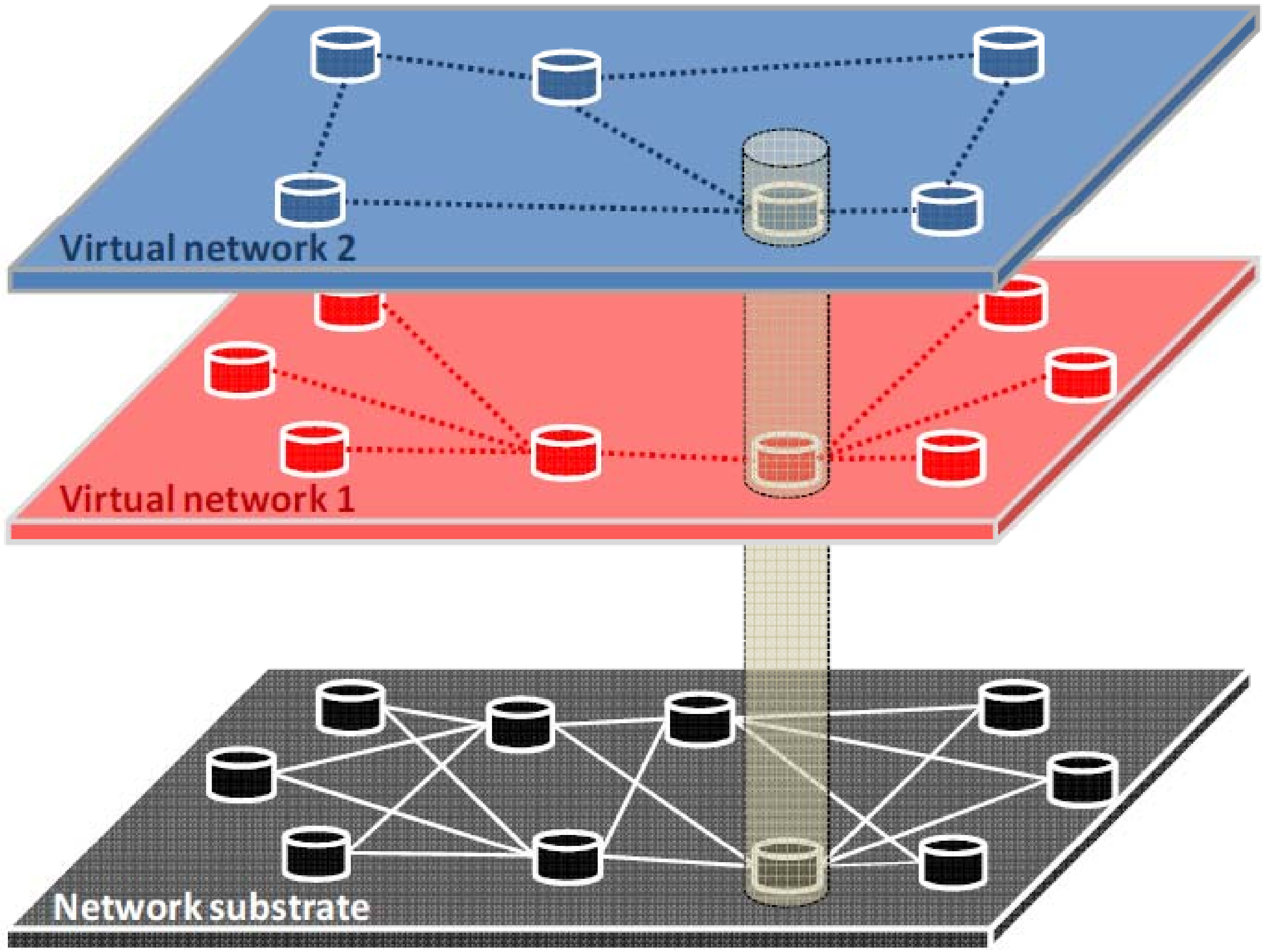
Why Network Virtualization

- Today's Internet is facing many challenges.
  - Internet's increasing ubiquity and centrality has brought with it a number of challenges.
- But, Internet is so successful.
  - It is hard to change its architecture.
  - This has led to an increasing number of ad hoc workarounds.
- Impasse of the current Internet
  - Evaluation issue
    - Traditional testbeds have limitations.
  - Deployment issue
    - It is needed that an agreement of ISPs.

Overcoming the Impasse

- Three separate requirements
  - Easy experiment with new architectures on live traffic
  - A plausible deployment path for putting validated architectural ideas into practice.
  - Comprehensive solutions, addressing the broad range of current architectural problems

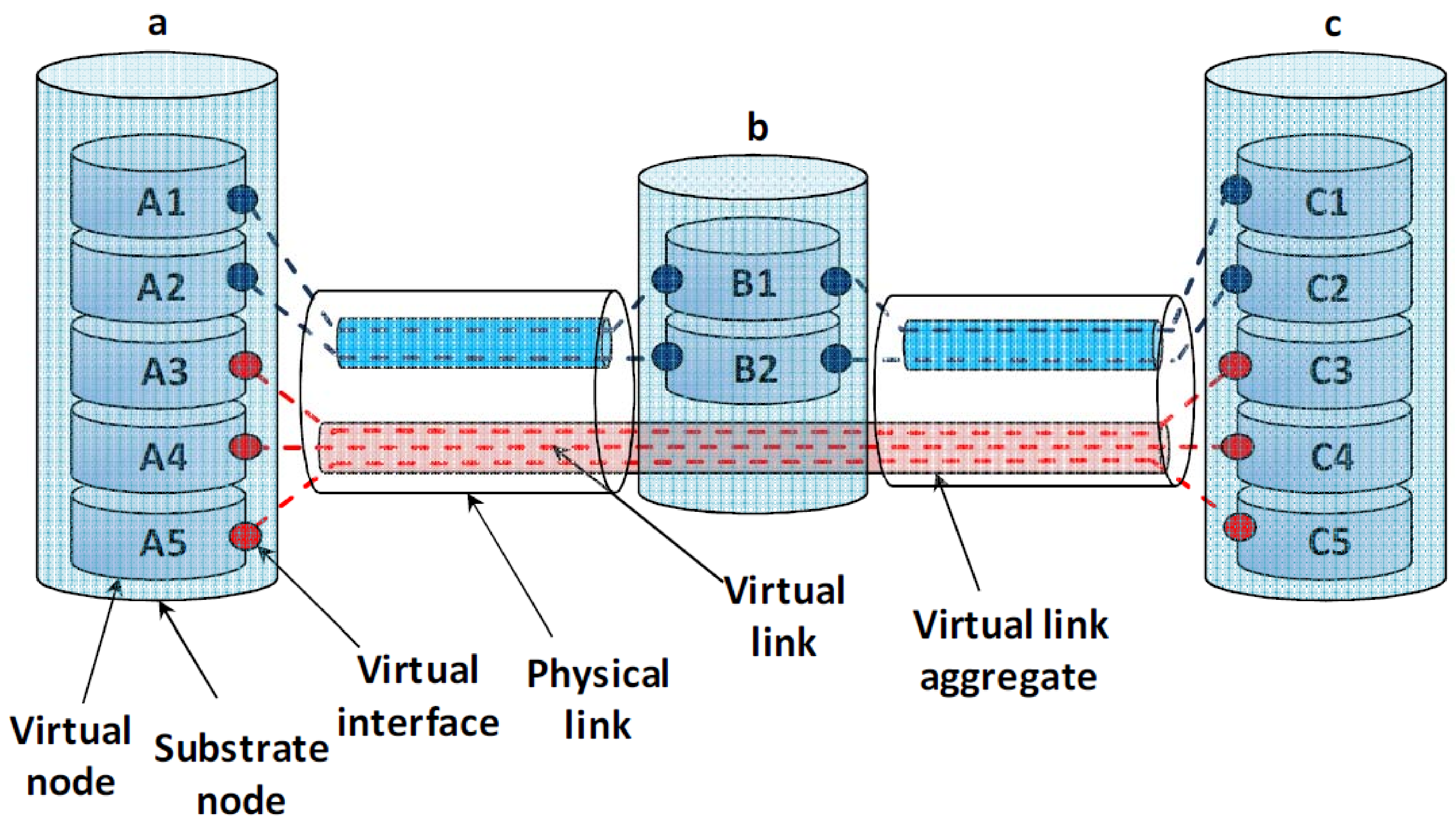
Network Virtualization Model



Substrate Node

- At the substrate level, the substrate node is network equipment capable of supporting virtual nodes by means of any virtualization technology.
- A single substrate node typically contains a number of virtual nodes.

Link Virtualization



Benefits of NV

- Network virtualization
  - provides a platform upon which novel network architectures can be built, experimented and evaluated, freed from legacy technology constraints
  - enables the coexistence of multiple networking approaches
  - provides a clean separation of services and infrastructure and facilitate of network resources among multiple providers and customers

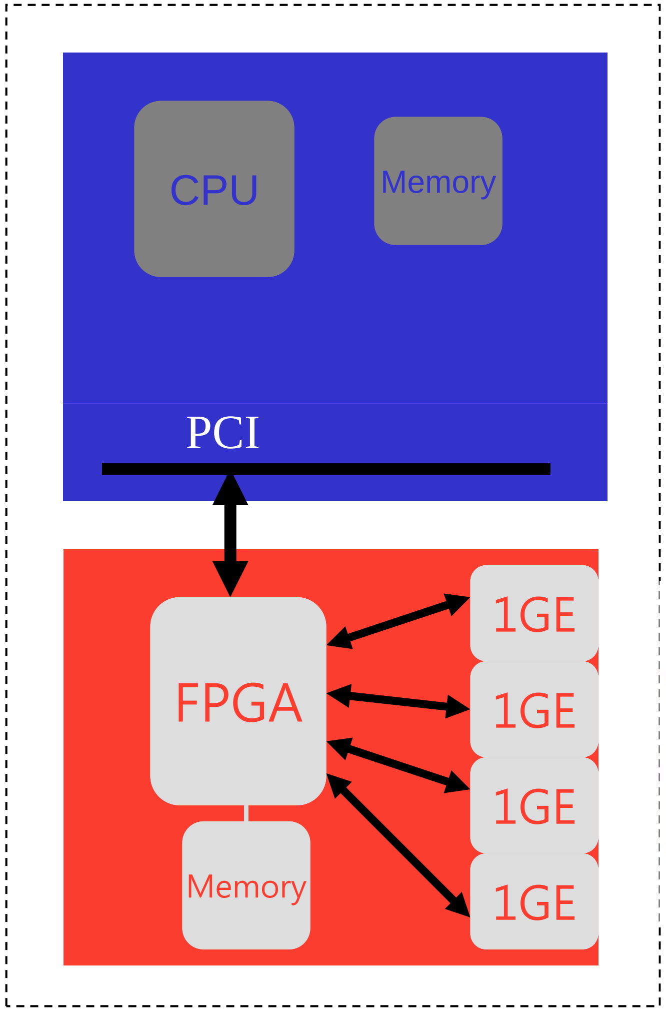
NetFPGA

- The NetFPGA is a low-cost open platform and available to developers worldwide.
- The NetFPGA is a PCI card that contains a large Xilinx FPGA and 4 Gigabit Ethernet ports.
- The NetFPGA platform enables researchers and instructors to build working prototypes of high-speed, hardware-accelerated networking systems.

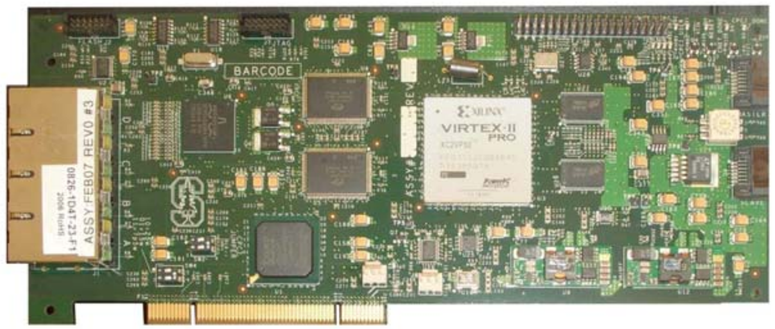
What is the NetFPGA?

Networking Software running on a standard PC

A hardware accelerator built with Field Programmable Gate Array driving Gigabit network links



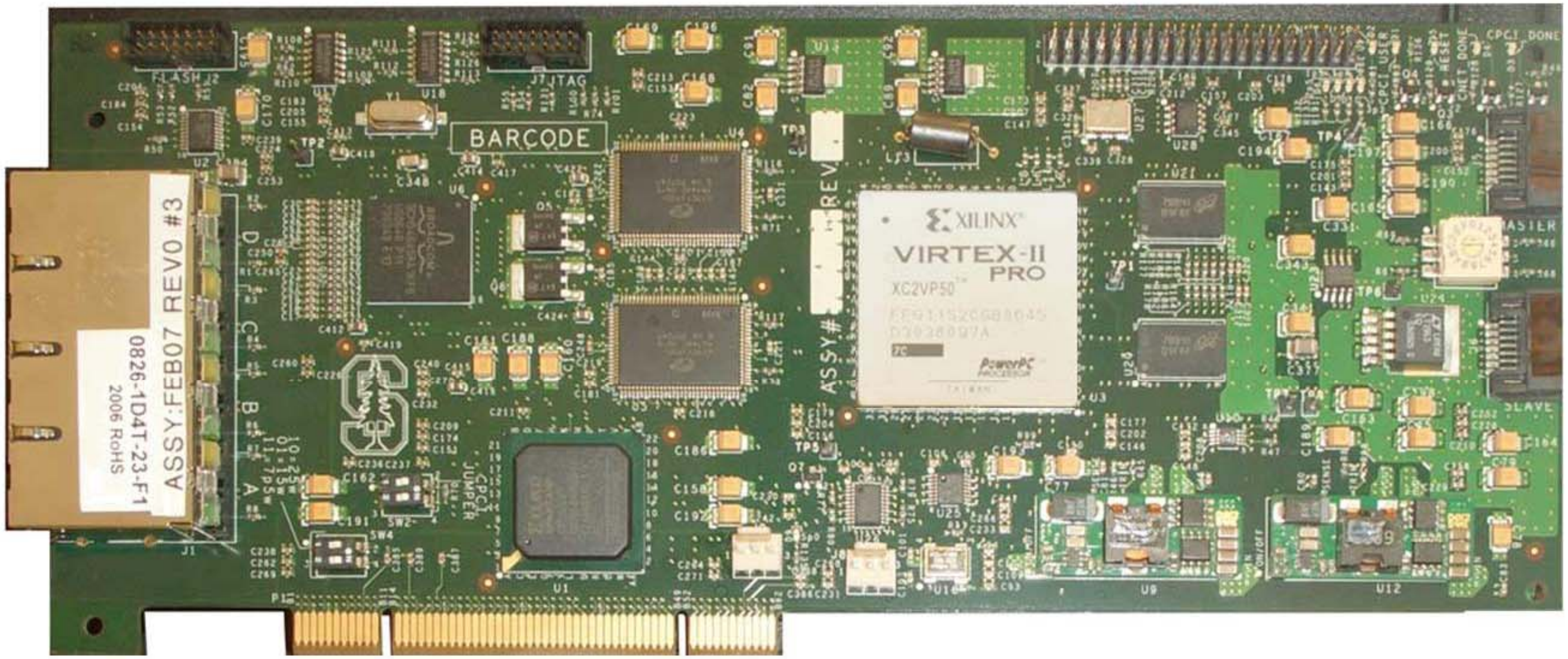
PC with NetFPGA


NetFPGA Board

Who, How, Why

- Who uses the NetFPGA?
  - Teachers
  - Students
  - Researchers
- How do they use the NetFPGA?
  - To run the Router Kit
  - To build modular reference designs
    - IPv4 router
    - 4-port NIC
    - Ethernet switch, …
- Why do they use the NetFPGA?
  - To measure performance of Internet systems
  - To prototype new networking systems

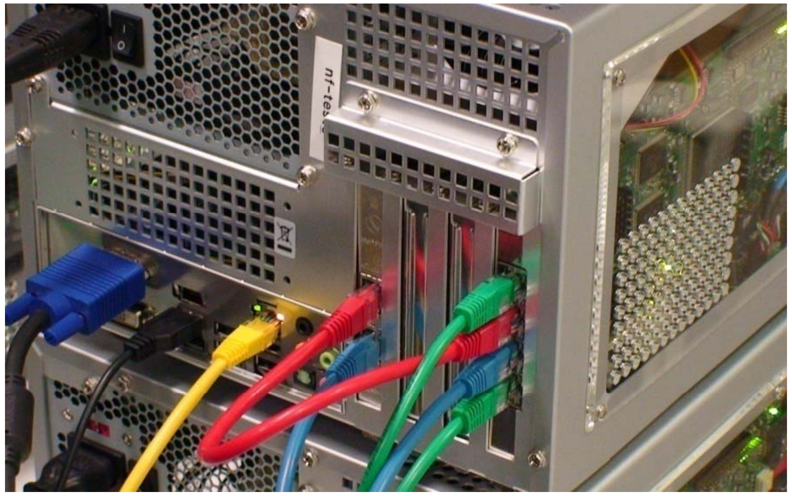
NetFPGA's Hardware Components



- Xilinx Virtex-2 Pro FPGA for User Logic (53,136 Logic Cells)
- Xilinx Spartan FPGA for PCI Host Interface
- Cypress: 2 * 2.25 MB ZBT SRAM
- Micron: 64MB DDR2 DRAM
- Broadcom: PHY for 4 Gigabit Ethernet ports
- Two SATA connectors

NetFPGA System Components

- NetFPGA
  - http://www.digilentinc.com
  - $1199, $599(Academic)
- Motherboard
  - Standard AMD or Intel-based x86 computer with PCI and PCI-express slots
- Processor
  - Dual or Quad-Core CPU
- Operating System
  - Linux CentOS 5.x



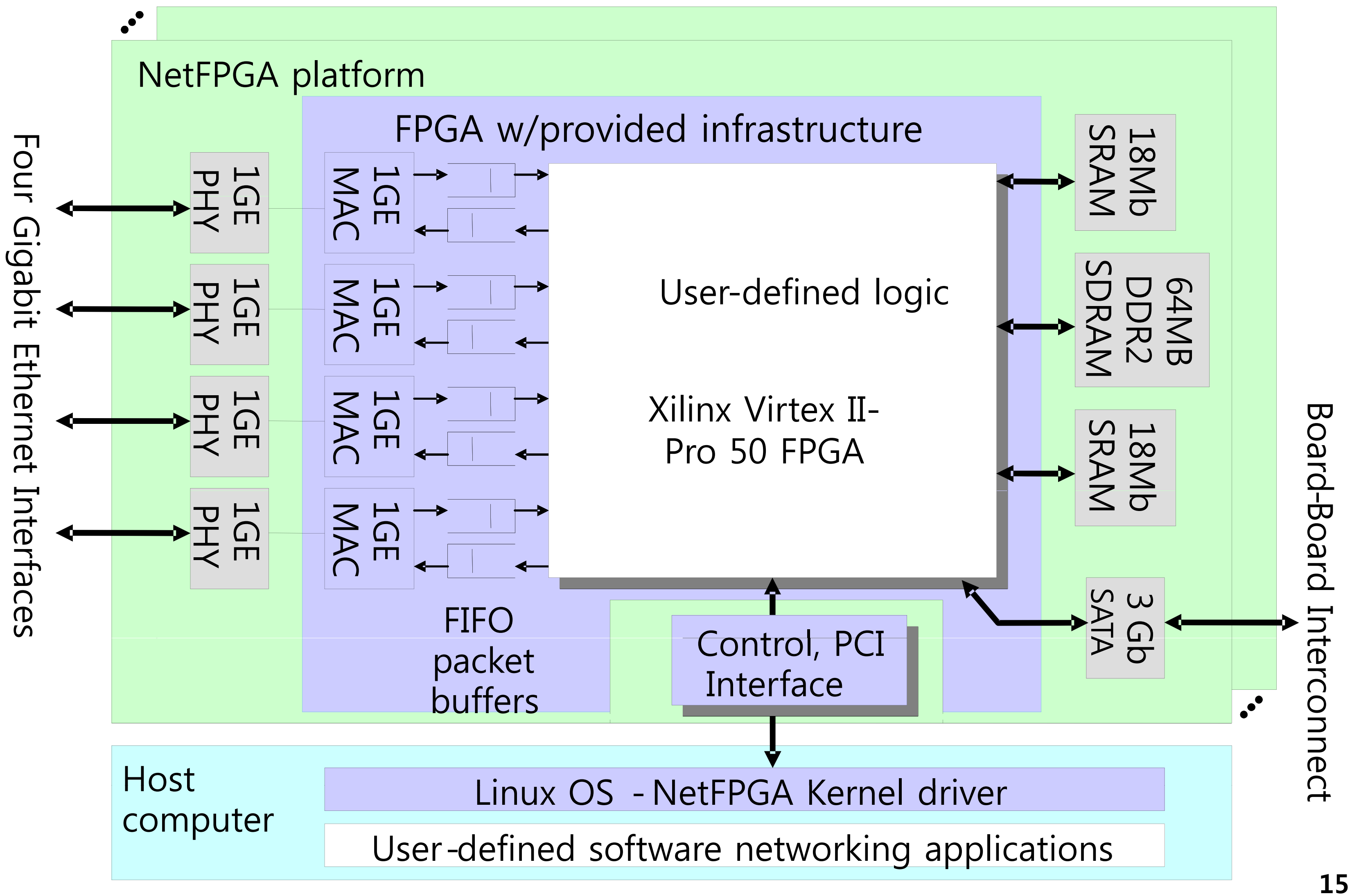
NetFPGA System

- Pre-built systems available
  - http://www.accenttechnologyinc.com/
  - $1421 ~ $1849

Cube system

Rackmount Server

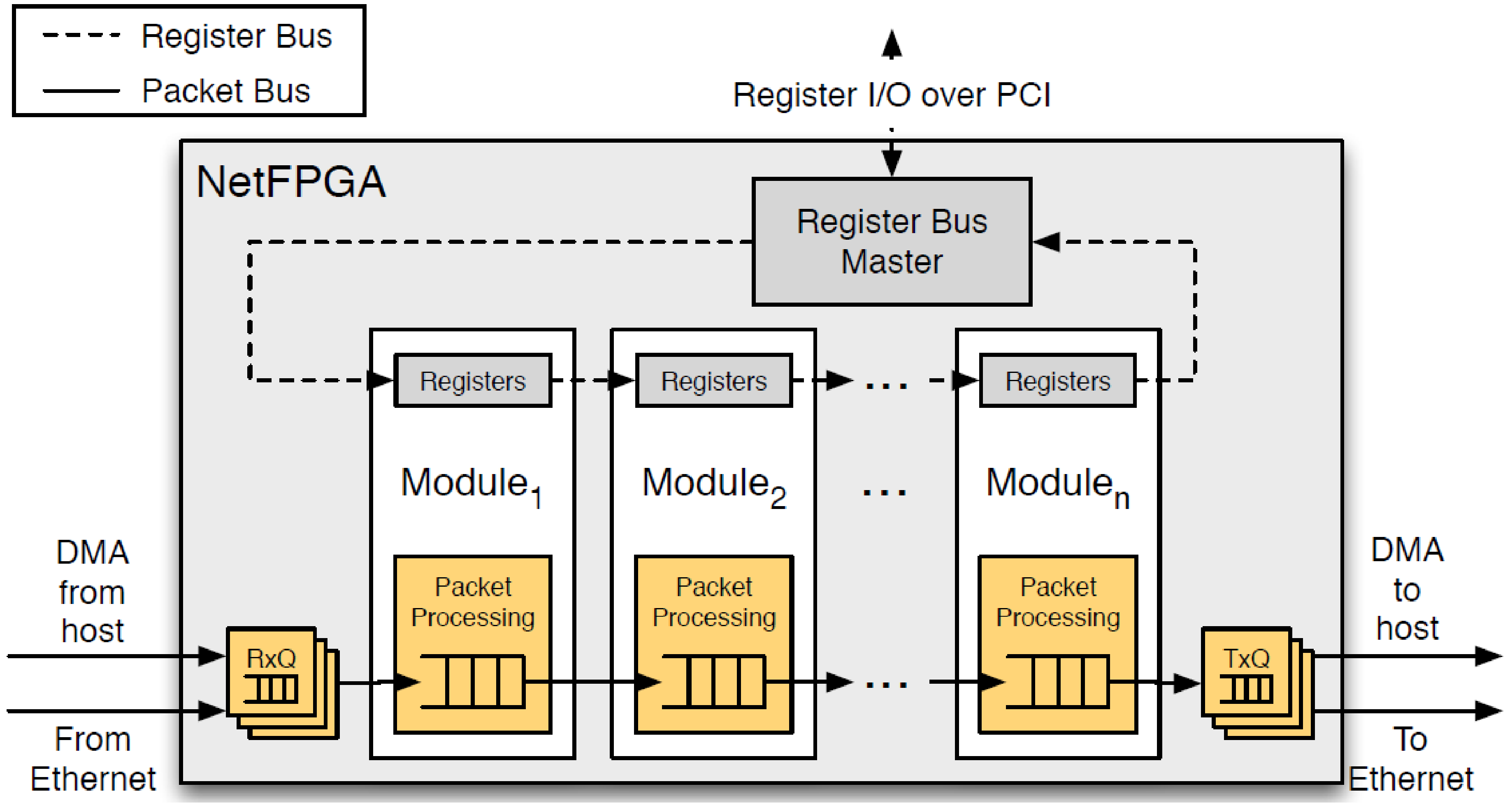
NetFPGA Block Diagram



Entry and Exit Points

- Three types of I/O ports
  - Ethernet Rx/Tx queues
    - send and receive packets via GigE ports
  - CPU DMA Rx/Tx queues
    - transfer packets via DMA between NetFPGA and the host CPU
  - Multi-gigabit serial Rx/Tx queues (to be added)
    - allow transferring over two 2.5 Gbps serial links
    - allow extending the NetFPGA by, for example, connecting it to another NetFPGA to implement an 8-port switch or a ring of NetFPGAs

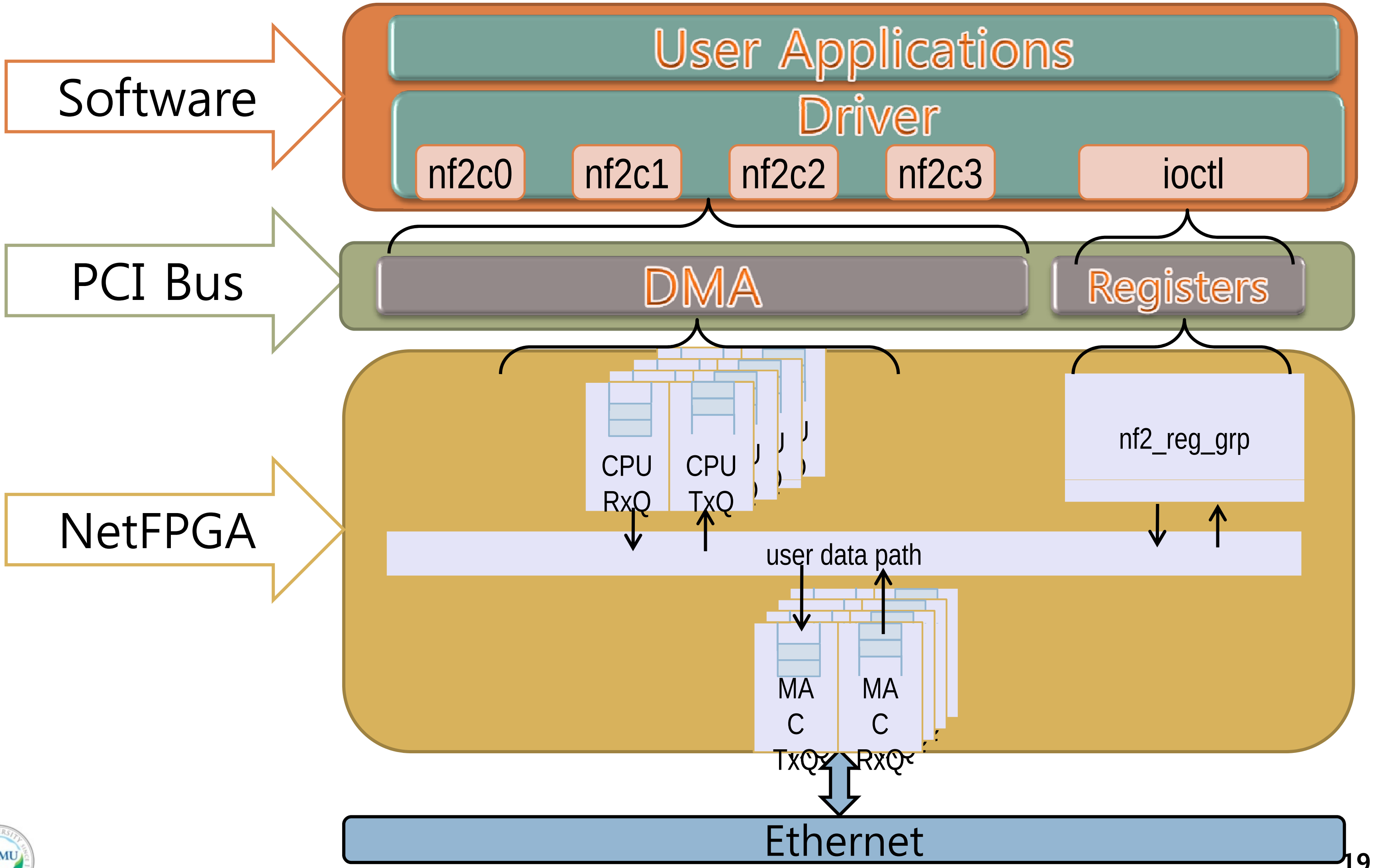
NetFPGA Pipeline



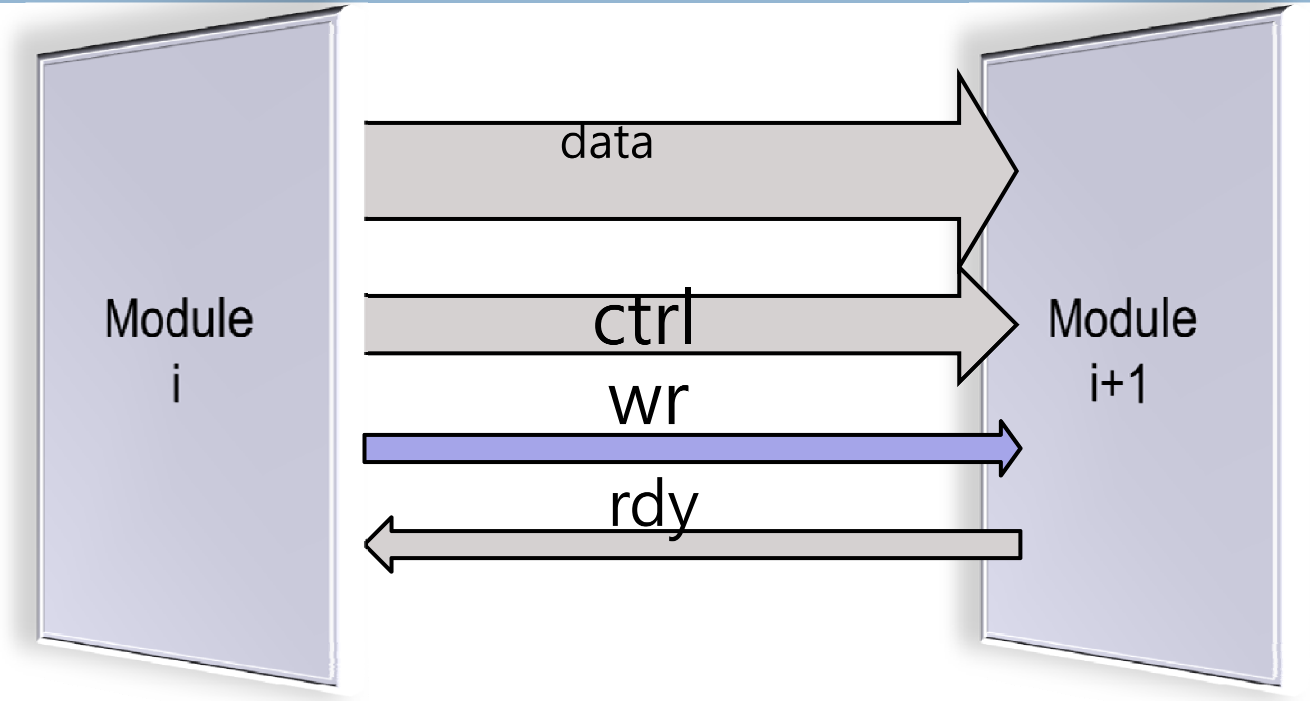
NetFPGA Pipeline

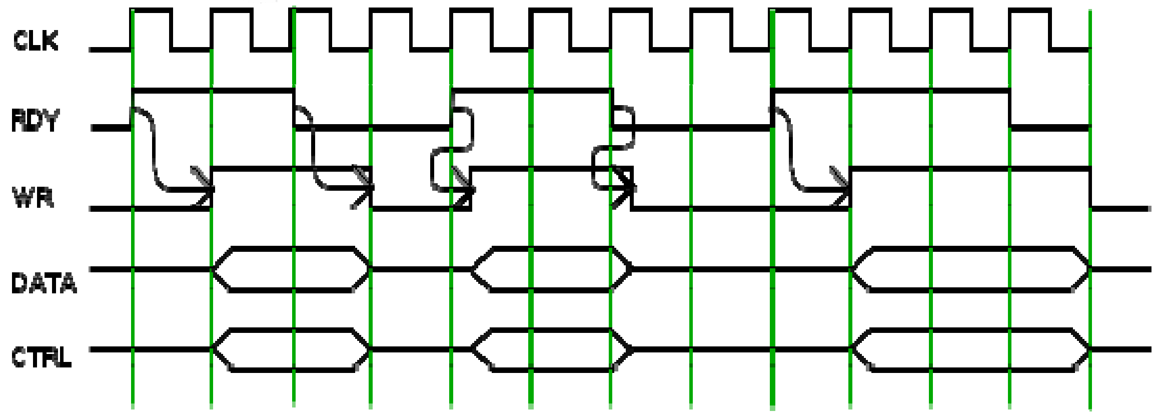
- NetFPGA modules are connected as a sequence of stages in a pipeline.
- Stages are interconnected using two point-to-point buses: the packet bus and the register bus.
  - The packet bus transfers packets from one stage to the next using a synchronous FIFO packet-based push interface, over a 64-bit wide bus running at 125MHz (an aggregate rate of 8Gbps).
  - The register bus provides another channel of communication that does not consume packet bus bandwidth.

Full System Components



Packet Bus





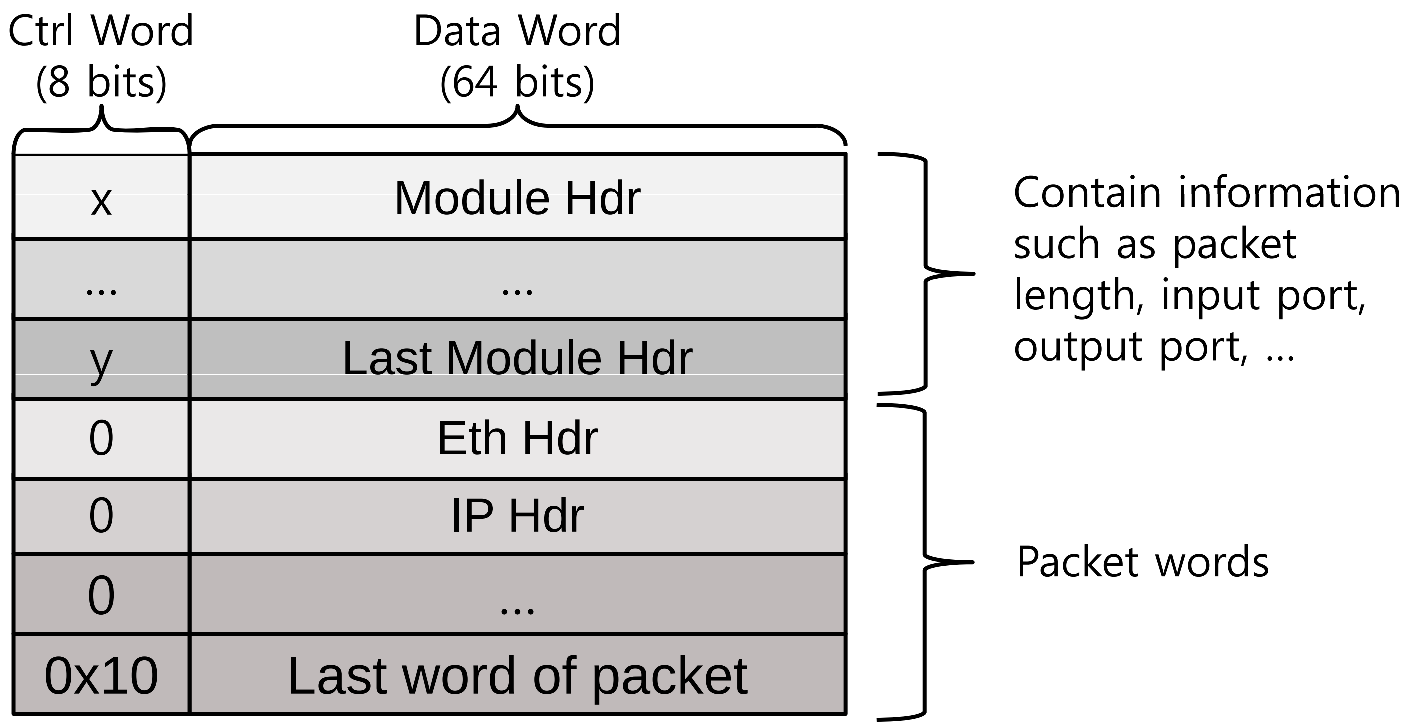
Packet Bus

- Packet bus consists of a ctrl bus and a data bus along with a write signal and a ready signal.
  - Write signal indicates that the buses are carrying valid information.
    - Stage i sets the ctrl and data buses, and asserts the write signal to write a word to Stage (i+1).
  - Ready signal from Stage (i+1) to Stage i provides flow control using backpressure.
    - Stage (i+1) asserts the ready signal indicating it can accept at least two more words of data, and deasserts it when it can accept only one more word or less.

Packet Format on the Packet Bus

Using "Module Headers"

- uniquely identified by ctrl word from other module headers

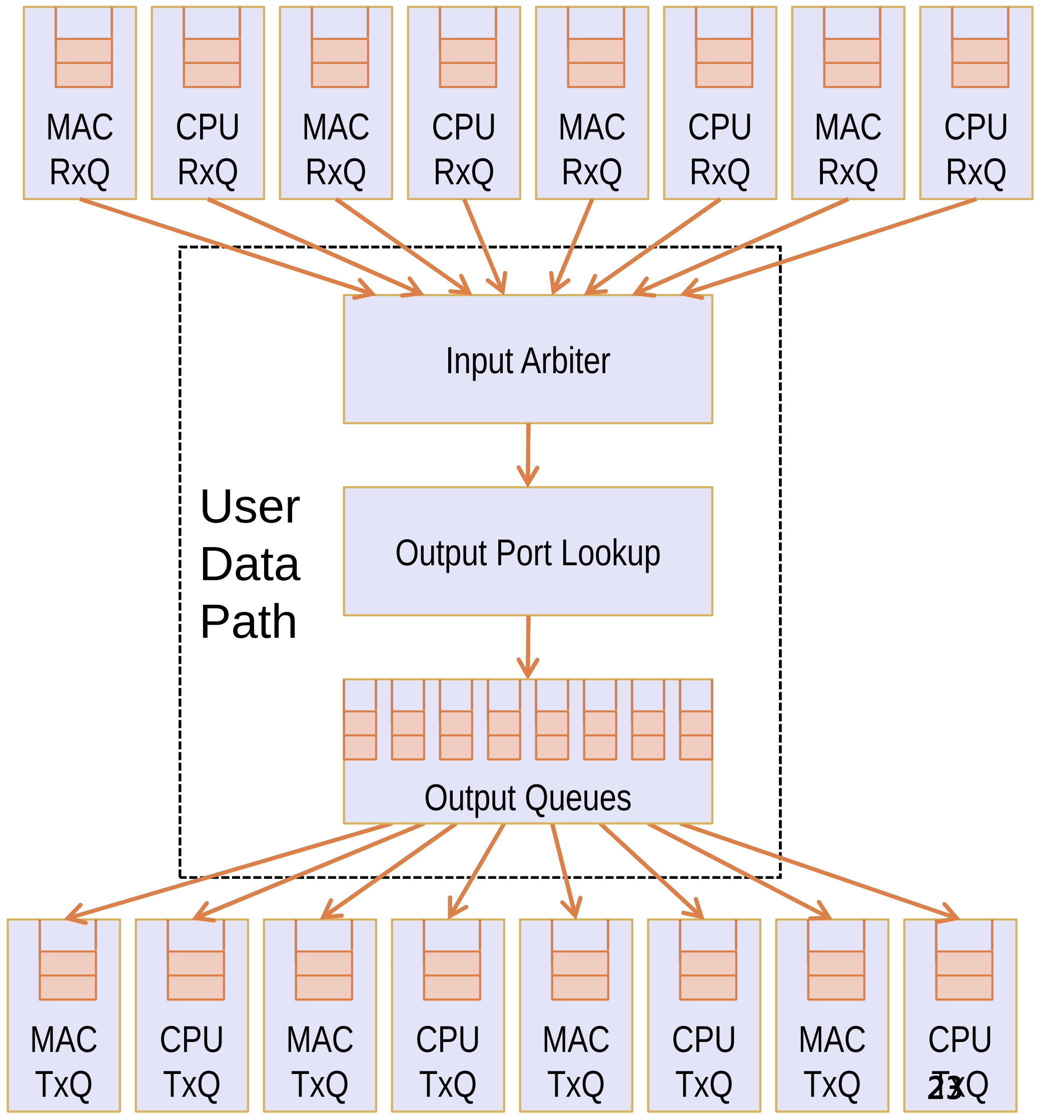


Ctrl Word (8 bits)	Data Word (64 bits)
x	Module Hdr
…	…
y	Last Module Hdr
0	Eth Hdr
0	IP Hdr
0	…
0x10	Last word of packet

Contain information such as packet length, input port, output port, …
Packet words

Reference Router Pipeline

- Five stages
  - Input
  - Input arbitration
  - Routing decision and packet modification
  - Output queuing
  - Output
- Packet-based module interface
- Pluggable design

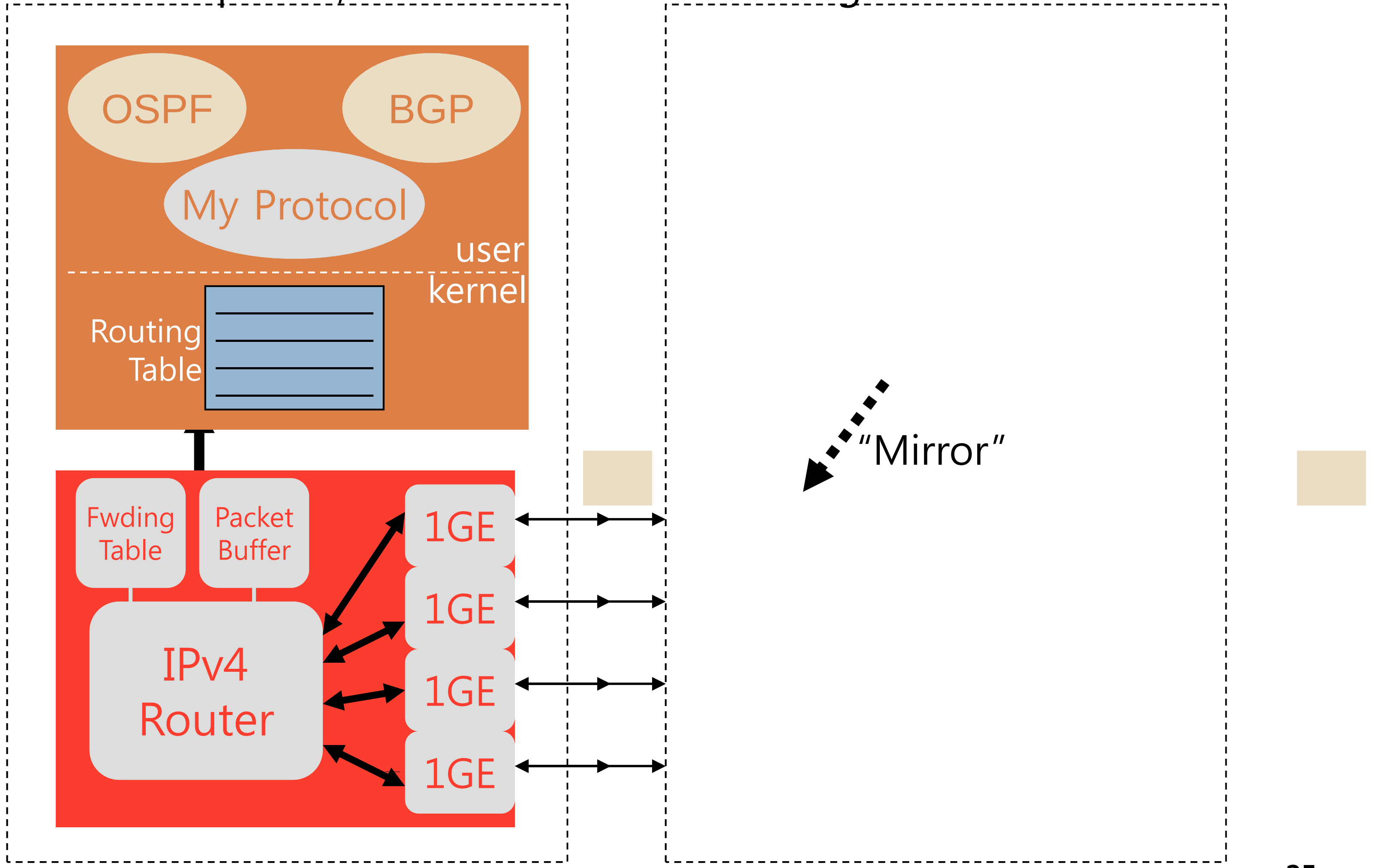


How to Use NetFPGA

- Three ways to create new systems using NetFPGAs
  - Less experienced users will reuse entire pre-built projects on the NetFPGA card.
  - Some others will modify pre-built projects and add new functionality to them by inserting new modules between the available modules.
  - Others will design completely new projects without using any pre-built modules.

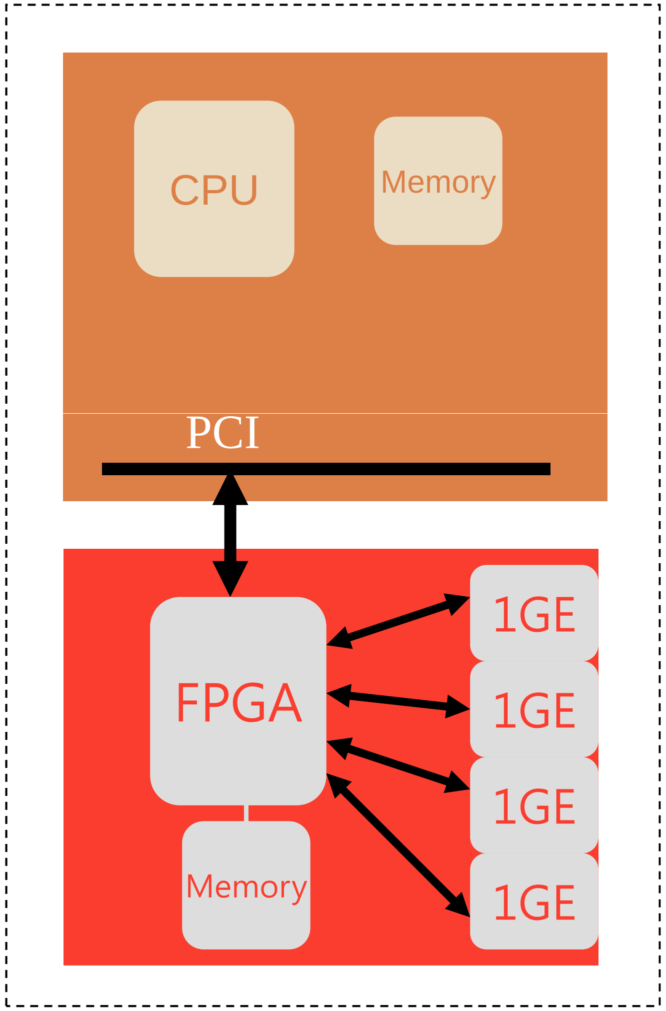
Running the Router Kit

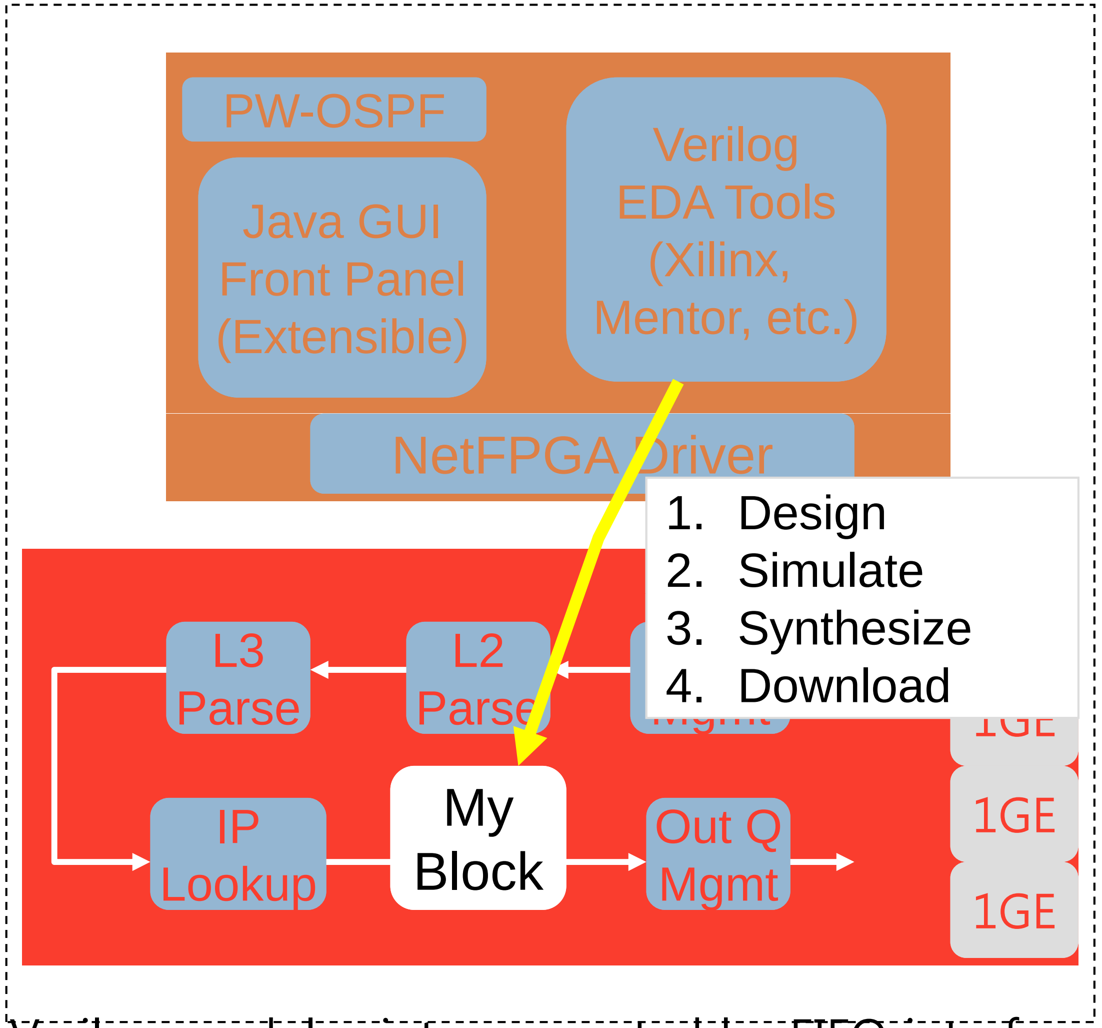
User-space development, 4x1GE line-rate forwarding



Usage #2

Enhancing Modular Reference Designs





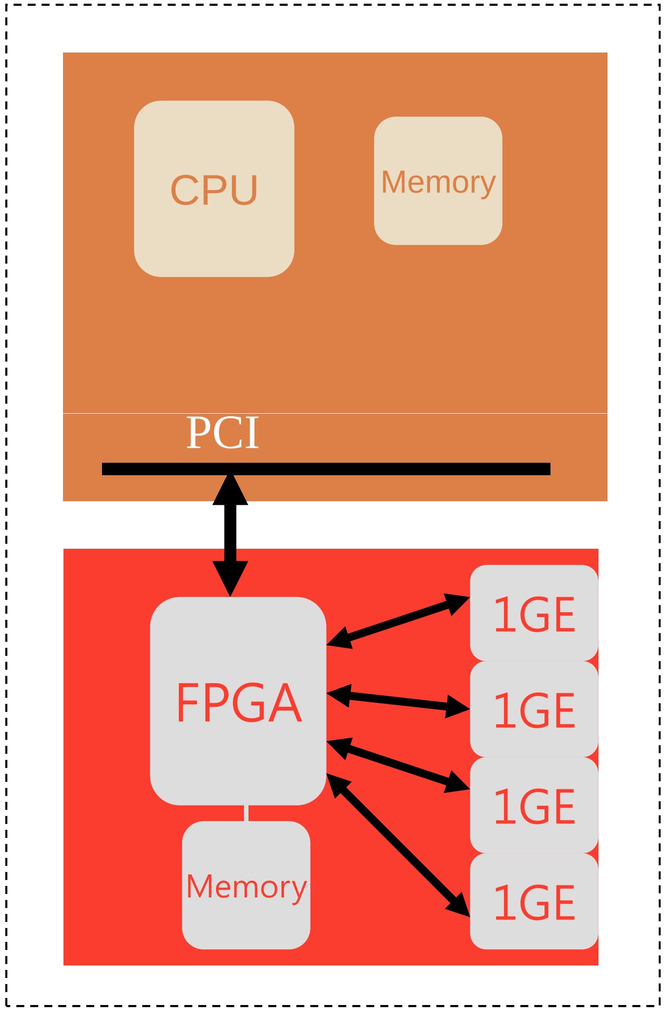
PW-OSPF
Java GUI Front Panel (Extensible)
Verilog EDA Tools (Xilinx, Mentor, etc.)
NetFPGA Driver
L3 Parse
L2 Parse
IP Lookup
My Block
Out Q Mgmt
1GE
1GE
1GE

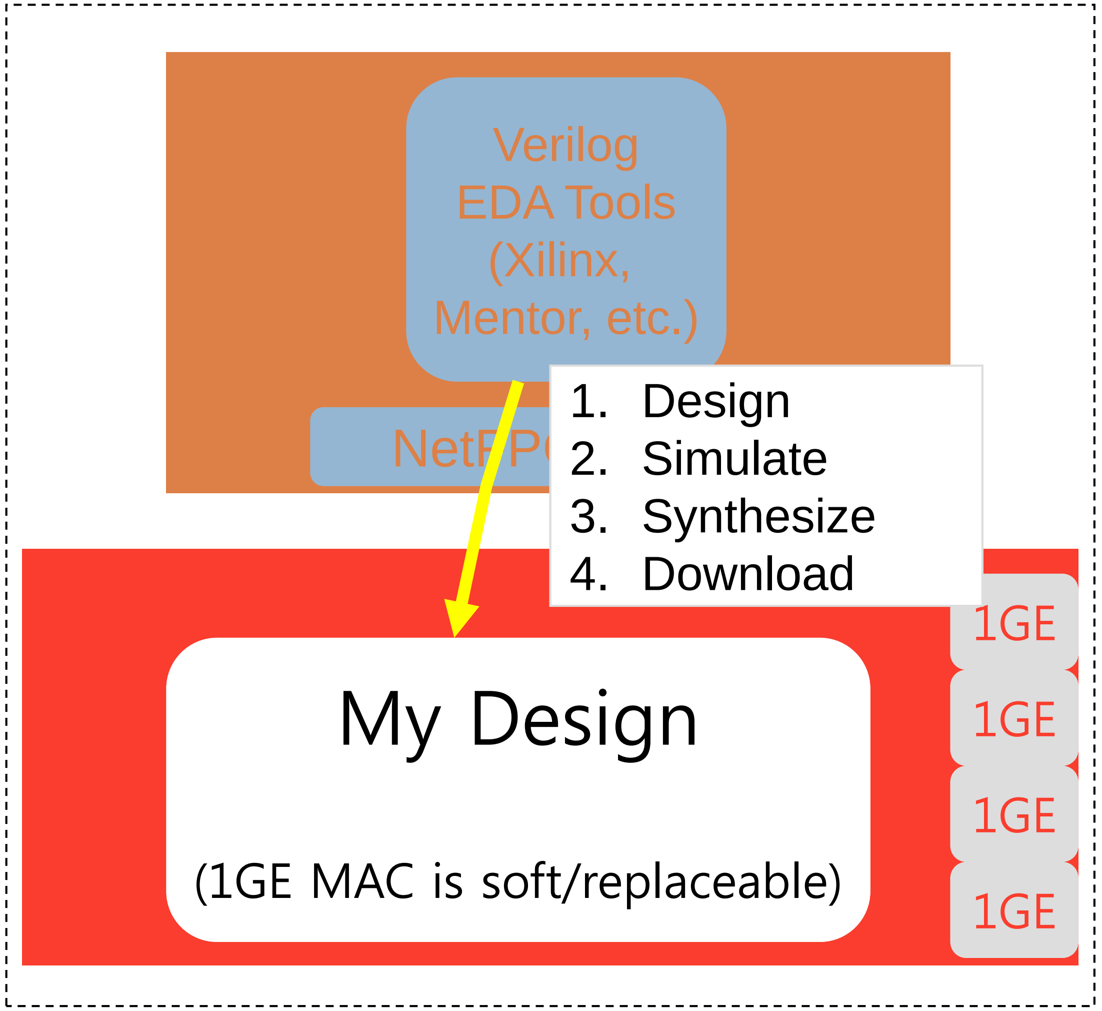
1. Design
2. Simulate
3. Synthesize
4. Download

Verilog modules interconnected by FIFO interfaces

Usage #3

Creating new systems



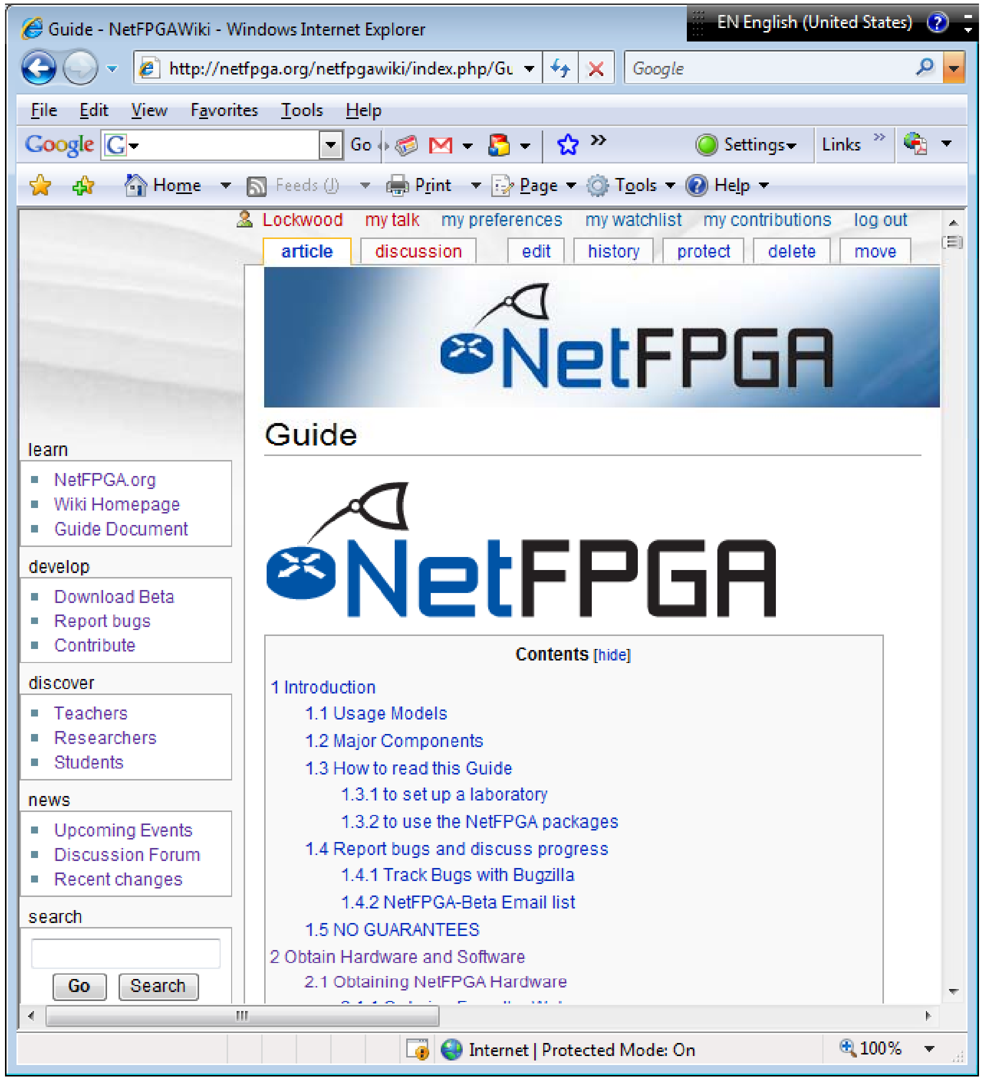


Projects that run on the NetFPGA

- http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/ProjectTable
  - IPv4 Reference Router
  - Quad-Port Gigabit NIC
  - Ethernet Switch
  - Buffer Monitoring System
  - Hardware-Accelerated Linux Router
  - DRAM-Router
  - Packet Generator
  - OpenFlow Switch
  - AirFPGA
  - RCP Router
  - …

Visit http://NetFPGA.org

- Obtain hardware, software, & gateware
- Install software, CAD tools, & simulation models
- Verify installation using regression self-tests
- Walk through the reference designs
- Learn about contributed packages


Guide - NetFPGAWiki - Windows Internet Explorer
http://netfpga.org/netfpgawiki/index.php/Gu
File Edit View Favorites Tools Help
Lockwood my talk my preferences my watchlist my contributions log out
article discussion edit history protect delete move
learn
NetFPGA.org
Wiki Homepage
Guide Document
develop
Download Beta
Report bugs
Contribute
discover
Teachers
Researchers
Students
news
Upcoming Events
Discussion Forum
Recent changes
search
Go Search
Internet | Protected Mode: On
100%

Guide

Contents [hide]

OpenFlow

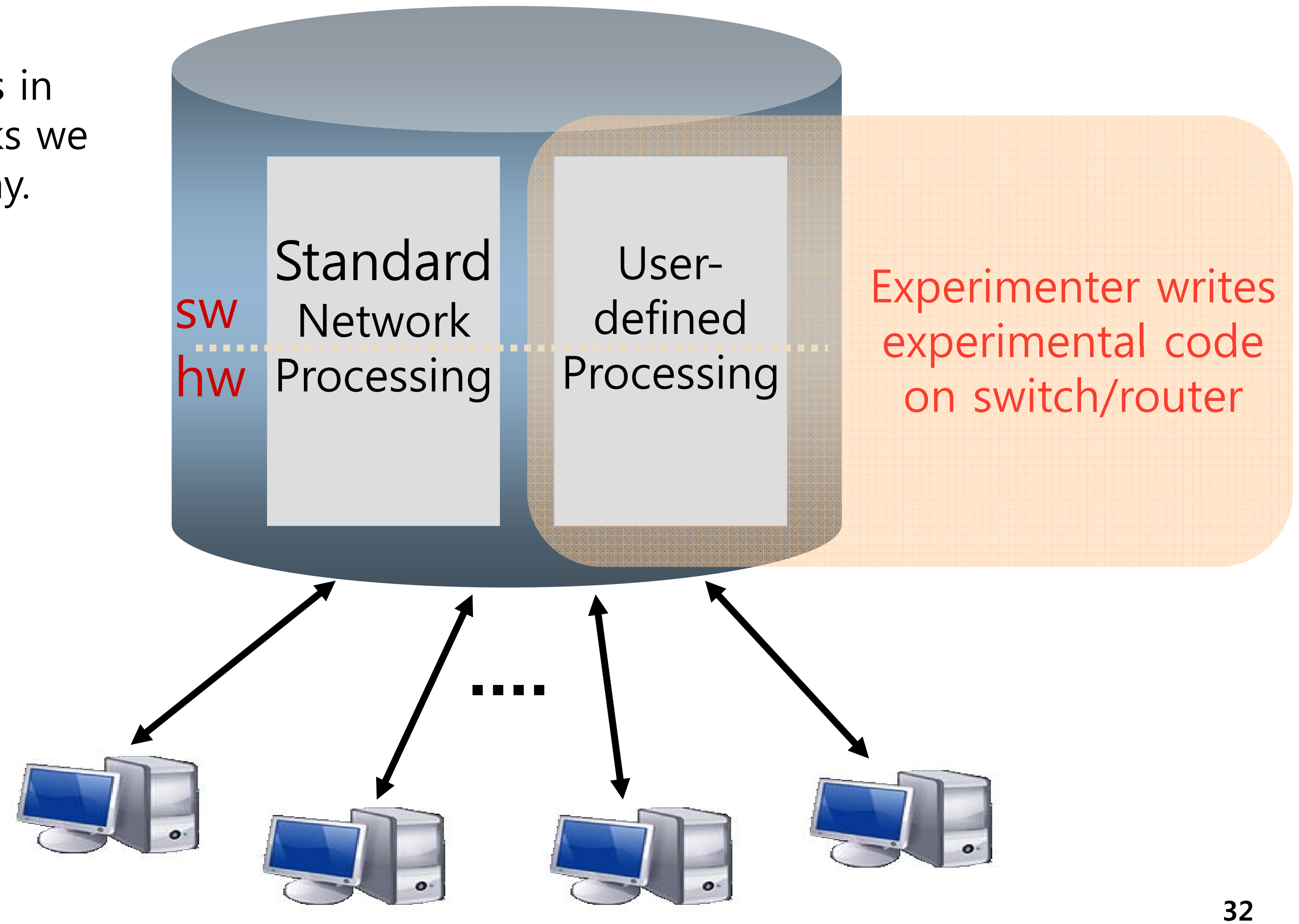
- Basics
  - An Ethernet switch (e.g. 128-ports of 1GE)
  - An open protocol to remotely add/remove flow entries
  - Allow researchers to run experiments in their network without requiring vendors to expose internal workings
  - Being implemented by major switch vendors
  - http://www.openflowswitch.org/

Motivation

- Experiments we'd like to do
  - Mobility management
  - Network-wide energy management
  - New naming/addressing schemes
  - Network access control
- Problem with our network
  - Paths are fixed (by the network)
  - IP-only
  - Addresses dictated by DNS, DHCP, etc
  - No means to add our own processing

Experimenter's Dream (Vendor's Nightmare)

Experiments in the networks we use everyday.



sw
hw
Standard Network Processing
User-defined Processing
Experimenter writes experimental code on switch/router
....

No obvious way

- Commercial vendor won't open software and hardware development environment
  - Complexity of support
  - Market protection and barrier to entry

- Hard to build my own
  - Prototypes are flakey
  - Software only: Too slow
  - Hardware/software: Fanout too small (need >100 ports for wiring closet)

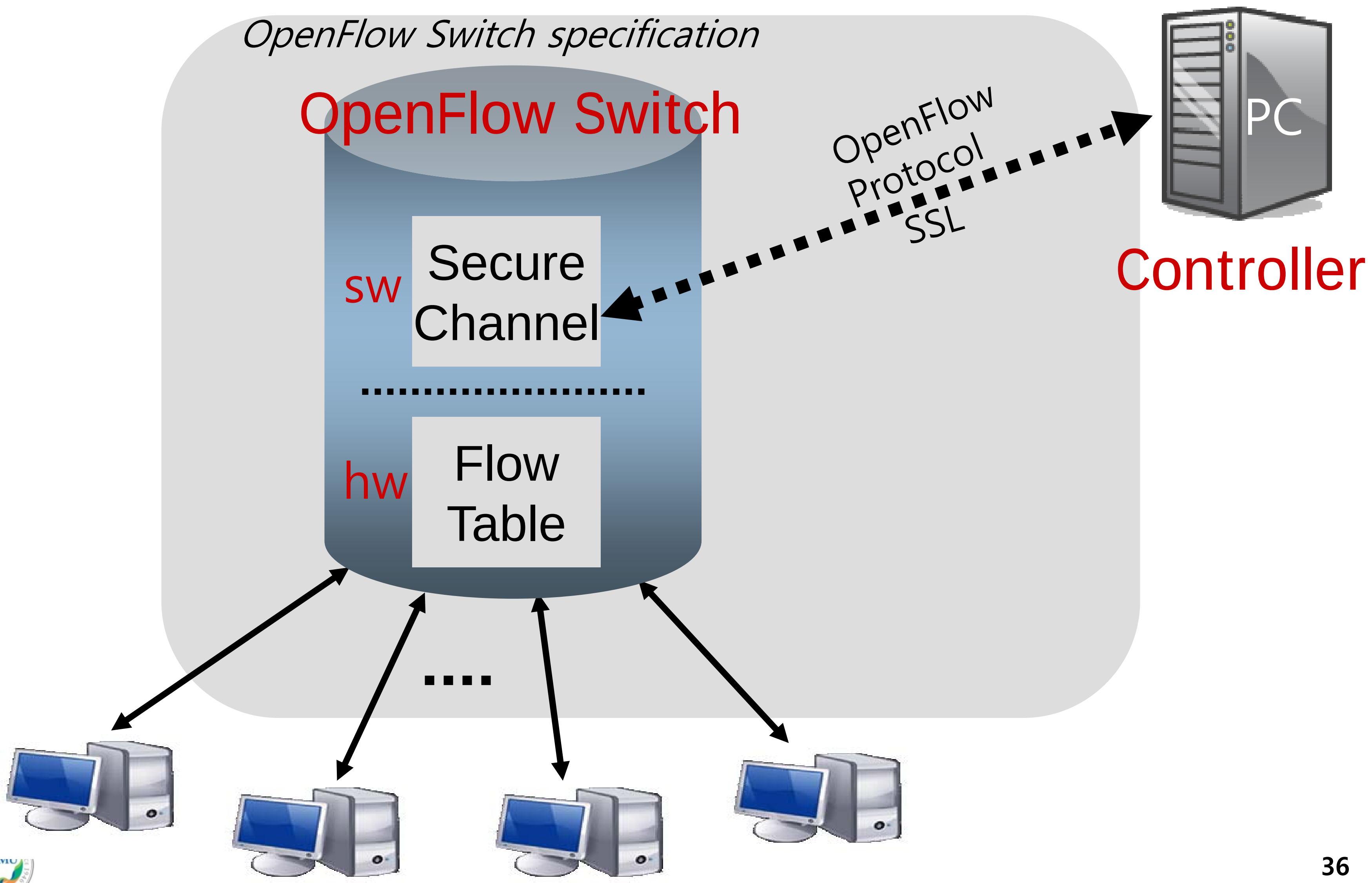
Furthermore, we want…

- Isolation: Regular production traffic untouched
- Virtualized and programmable: Different flows processed in different ways
- Equipment we can trust in our wiring closet
- Open development environment for all researchers (e.g. Linux, Verilog, etc).
- Flexible definitions of a flow
  - Individual application traffic
  - Aggregated flows
  - Alternatives to IP running side-by-side
  - …

Need for Programmable Networks

- Amenable to high-performance and low-cost implementations.
- Capable of supporting a broad range of research.
- Assured to isolate experimental traffic from production traffic.
- Consistent with vendors' need for closed platforms.

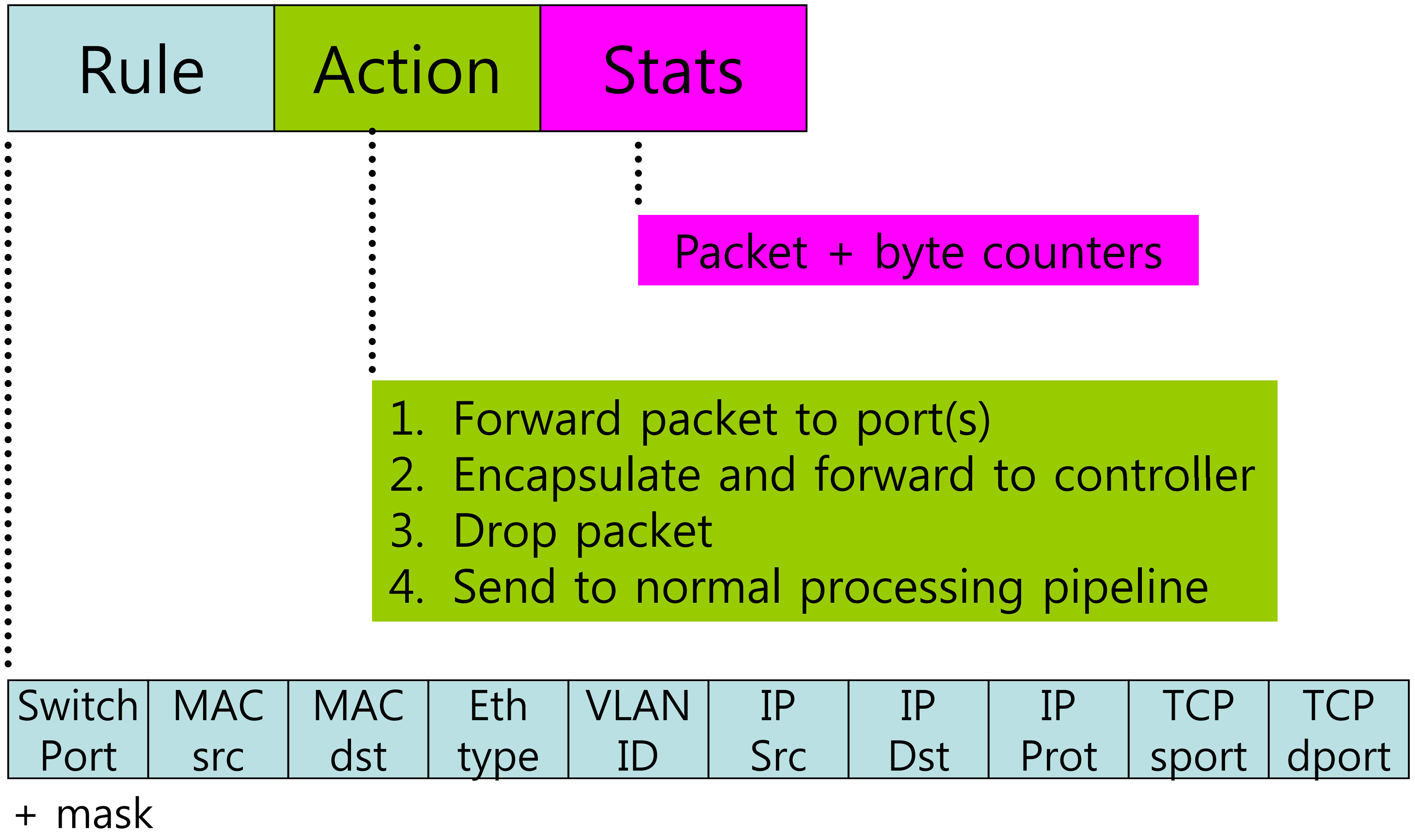
OpenFlow Switching



Two Types OpenFlow Switches

- Type 0
  - The minimum requirements for any conforming OpenFlow Switch.
- Type 1
  - Superset of Type 0, and remain to be defined.

Flow Table Entry (Type 0)

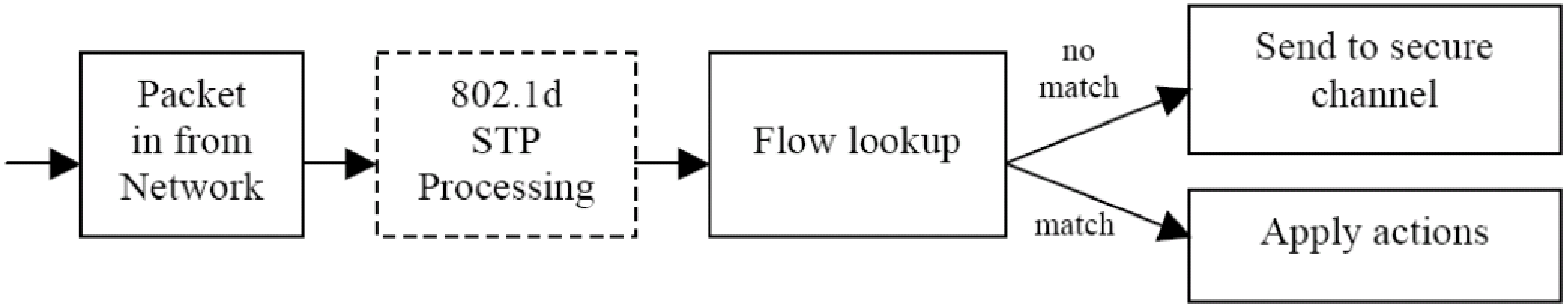


1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

Dataflow

- On receipt of a packet, an OpenFlow Switch performs the functions.
  - Rules specifying an ingress port are matched against the physical port that received the packet.
  - The Ethernet headers are used for all packets.
  - If the packet is a VLAN (Ethernet type 0x8100), the VLAN ID is used in the lookup.
  - For IP packets (Ethernet type equal to 0x0800), the lookup fields also include those in the IP header.
  - For IP packets that are TCP or UDP (IP protocol is equal to 6 or 17), the lookup includes the transport ports.



OpenFlow “Type 1”

- Definition in progress
- Additional actions
  - Rewrite headers
  - Map to queue/class
  - Encrypt
- More flexible header
  - Allow arbitrary matching of first few bytes
- Support multiple controllers
  - Load-balancing and reliability

Secure Channel

- SSL Connection, site-specific key
- Controller discovery protocol
- Encapsulate packets for controller
- Send link/port state to controller

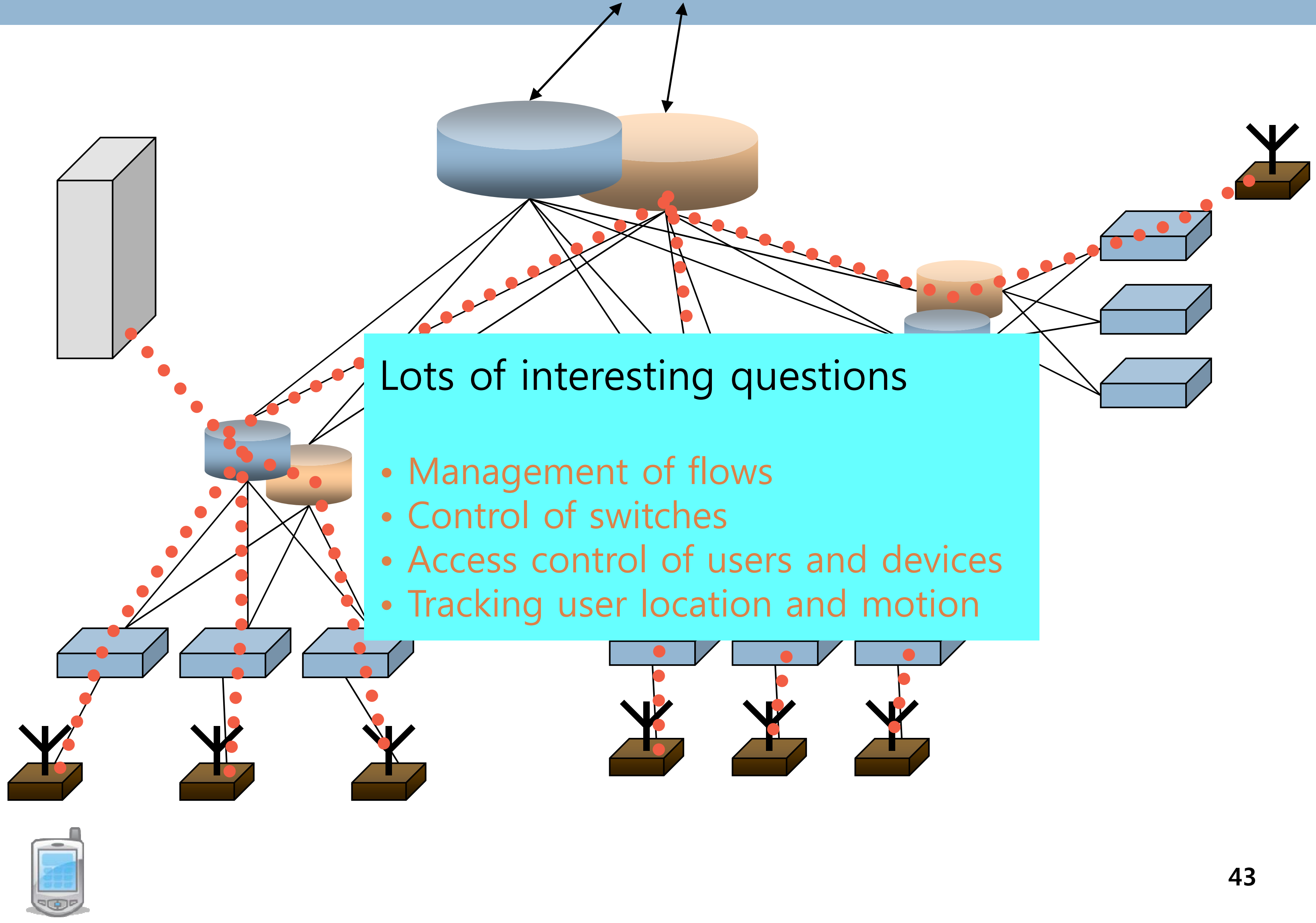
OpenFlow Usage Models

- Experiments at the flow level
  - User-defined routing protocols
  - Admission control
  - Network access control
  - Network management
  - Energy management
  - VOIP mobility and handoff
  - …

  } • Experiment-specific controllers
  • Static or dynamic flow-entries
- Experiments at the packet level
  - Slow: Controller handles packet processing
  - Fast: Redirect flows through programmable hardware
  - Modified routers, firewalls, NAT, congestion control…
- Alternatives to IP
  - Flow-table is Layer-2 based
  - e.g. new naming and addressing schemes
  - …

Example Experiment at the flow level

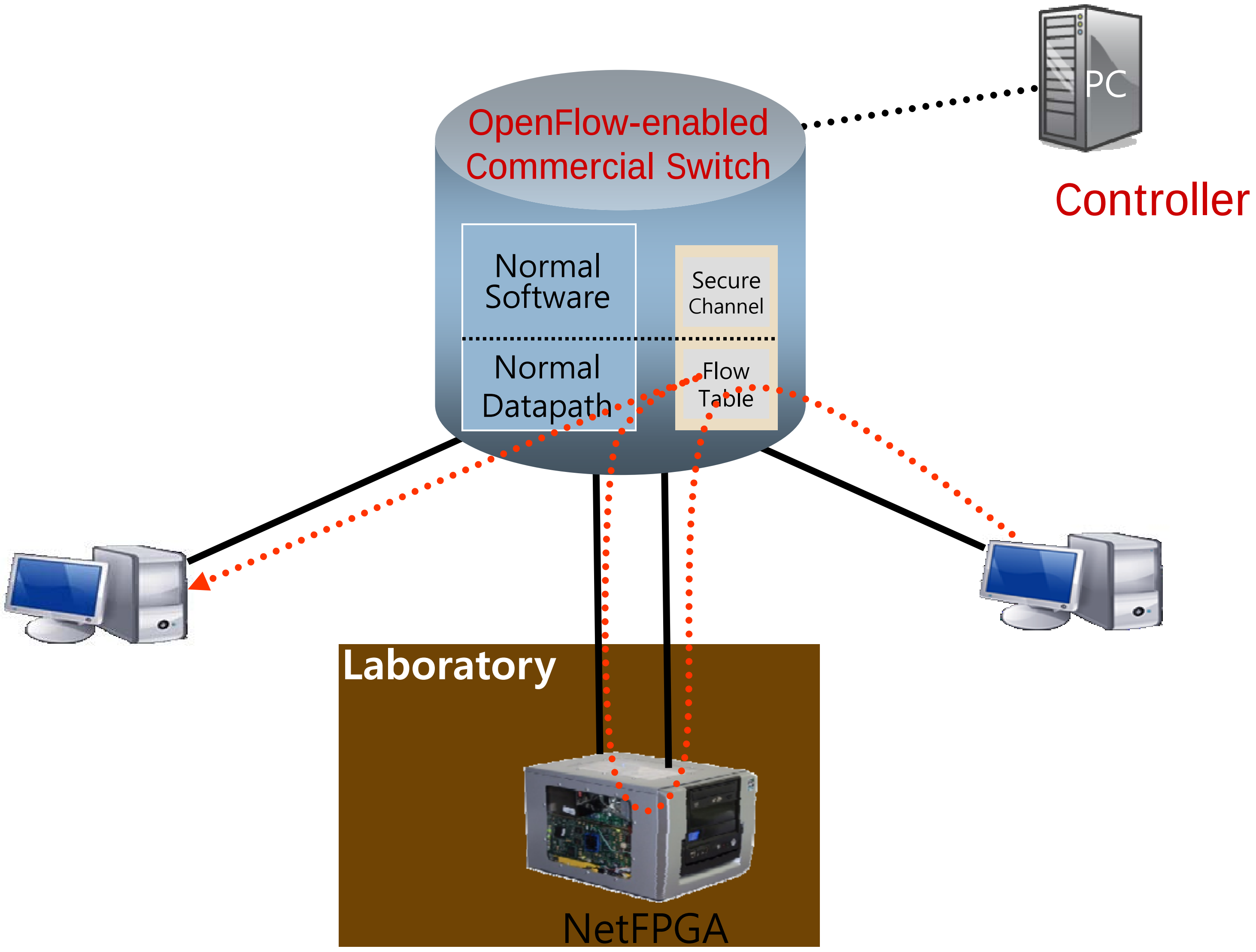
Mobility



Lots of interesting questions

- Management of flows
- Control of switches
- Access control of users and devices
- Tracking user location and motion

Experiments at the packet level



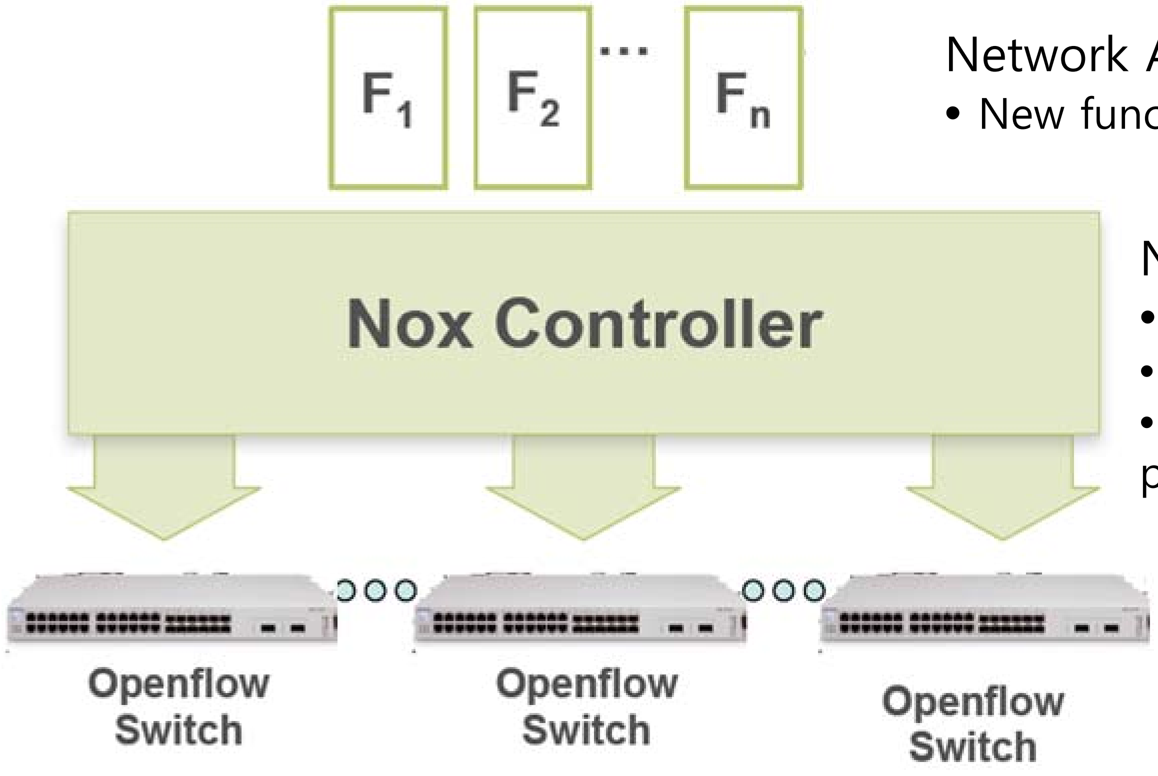
OpenFlow Switch Specification

- OpenFlow Switch Specification, Version 1.0.0 (December 31, 2009)
  - The standards document that describes the protocol that is used between an OpenFlow Switch and the OpenFlow Controller.
  - Cover the components and the basic functions of the switch, and the OpenFlow protocol to manage an OpenFlow switch from a remote controller.

NOX Controller

- NOX is an open platform for developing management functions
  - NOX enables the following:
    - NOX provides sophisticated network functionality (management, visibility, monitoring, access controls, etc.) on extremely cheap switches.
    - Developers can add their own control software and, unlike standard *nix based router environments, NOX provides an interface for managing off the shelf hardware switches at line speeds.
    - NOX provides a central programming model for an entire network – one program can control the forwarding decisions on all switches on the network. This makes program development much easier than in the standard distributed fashion.
  - http://noxrepo.org

NOX Components



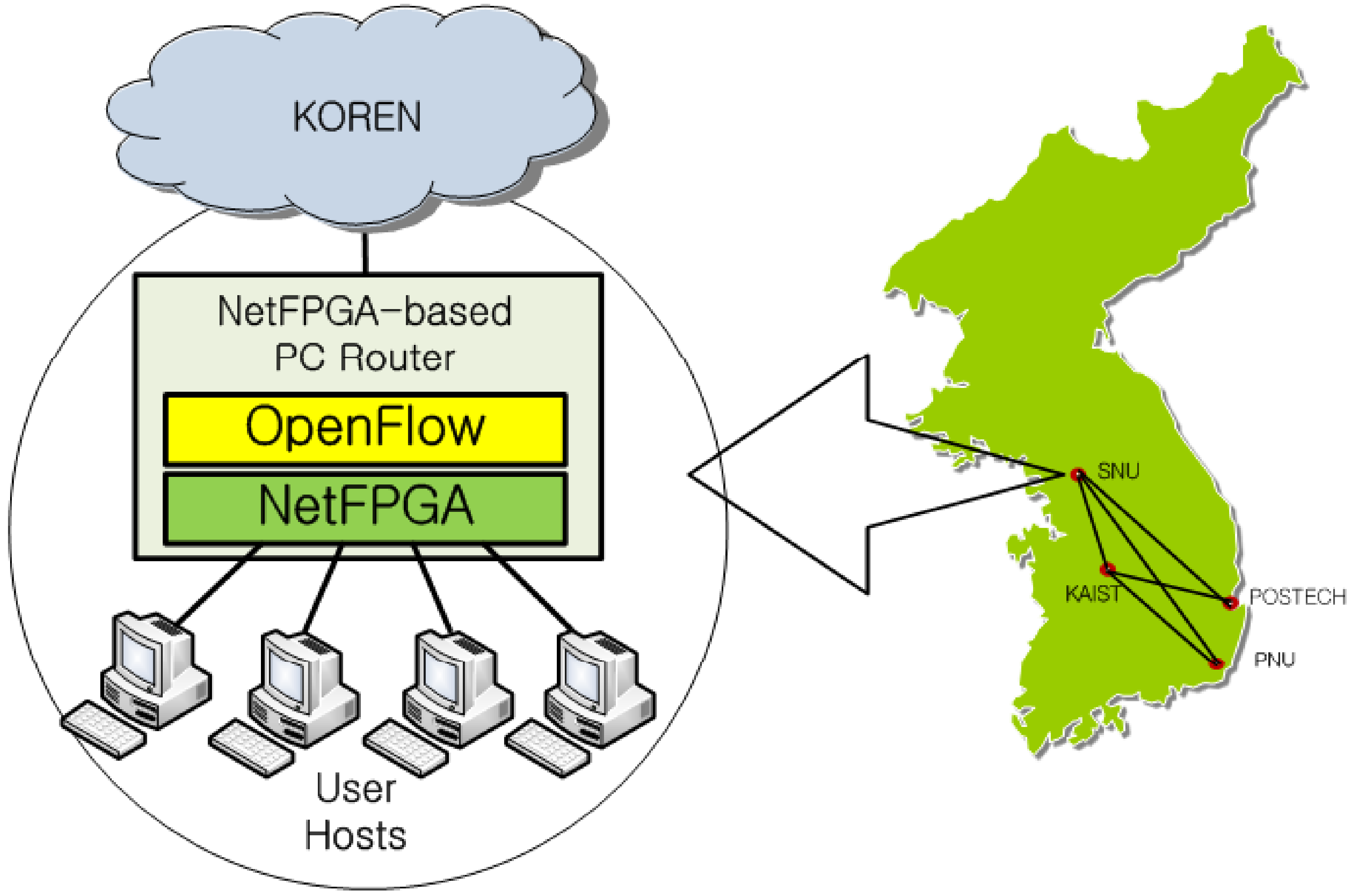
Network Application Services
- New functions as software services

Nox Controller
- Open platform for network control
- Provides system-wide abstractions
- Turn networking into a software problem

KOREN

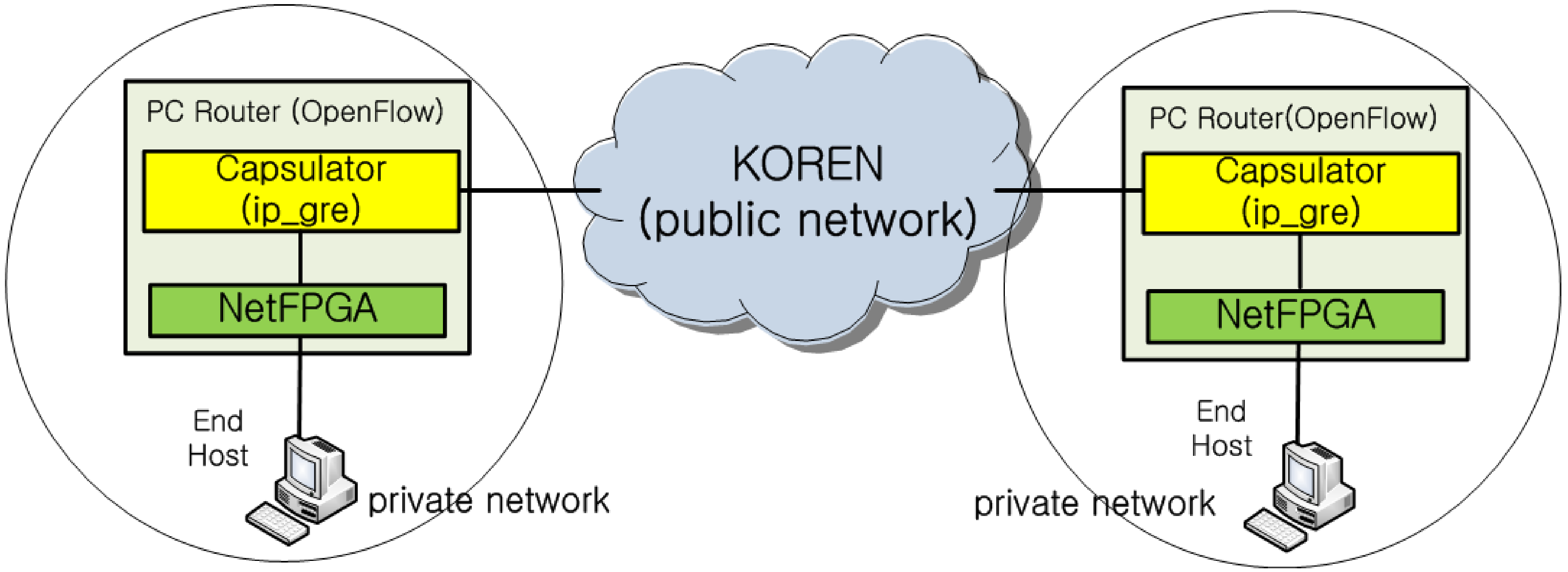
- KOREN (Korea Advanced Research Network)
  - a non-profit testbed network infrastructure established for facilitating research and development and international joint research cooperation.
  - provides quality broadband network testbed for domestic and international research activities to the industry, academia, and research institutions, enabling testing of future network technologies and supporting R&D on advanced applications.

Overlay Network over KOREN

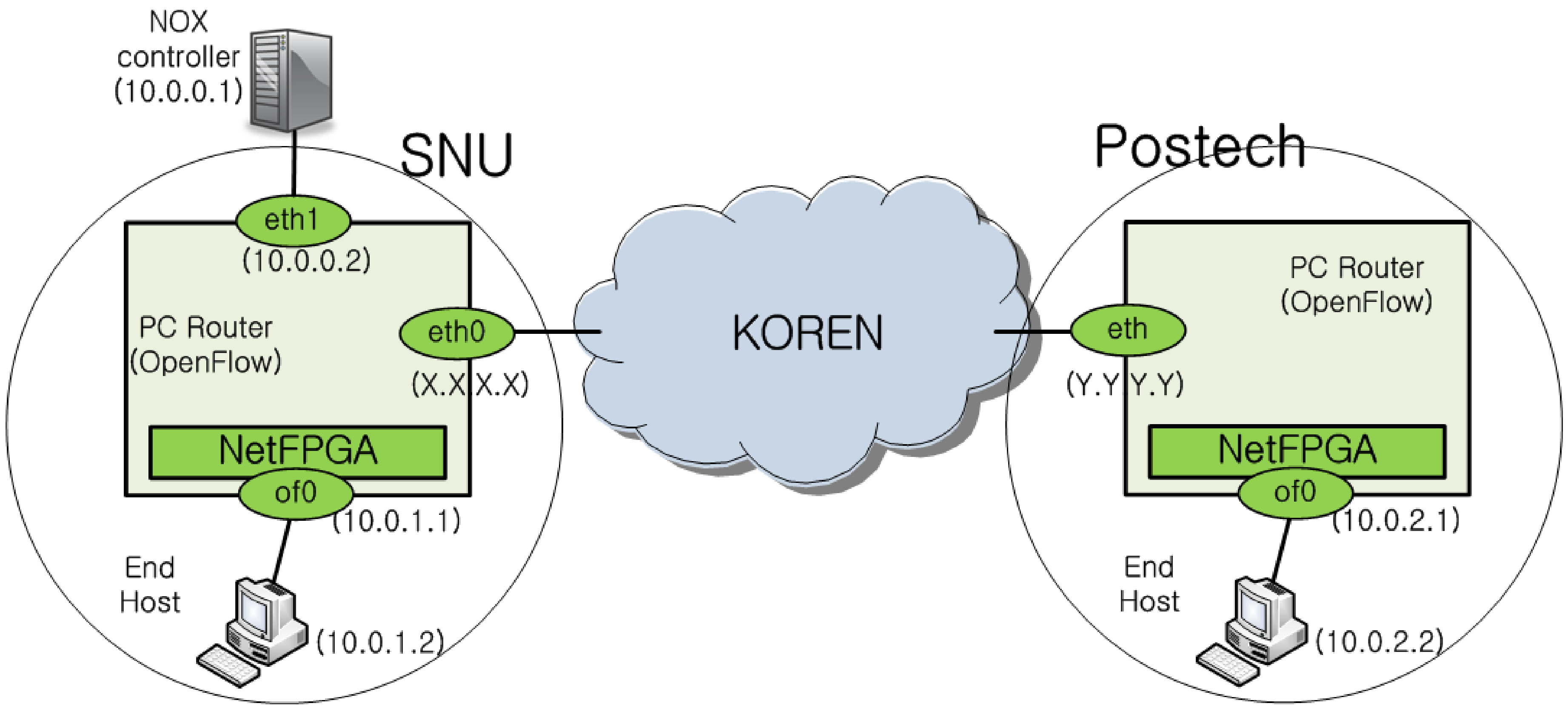


GRE Tunneling

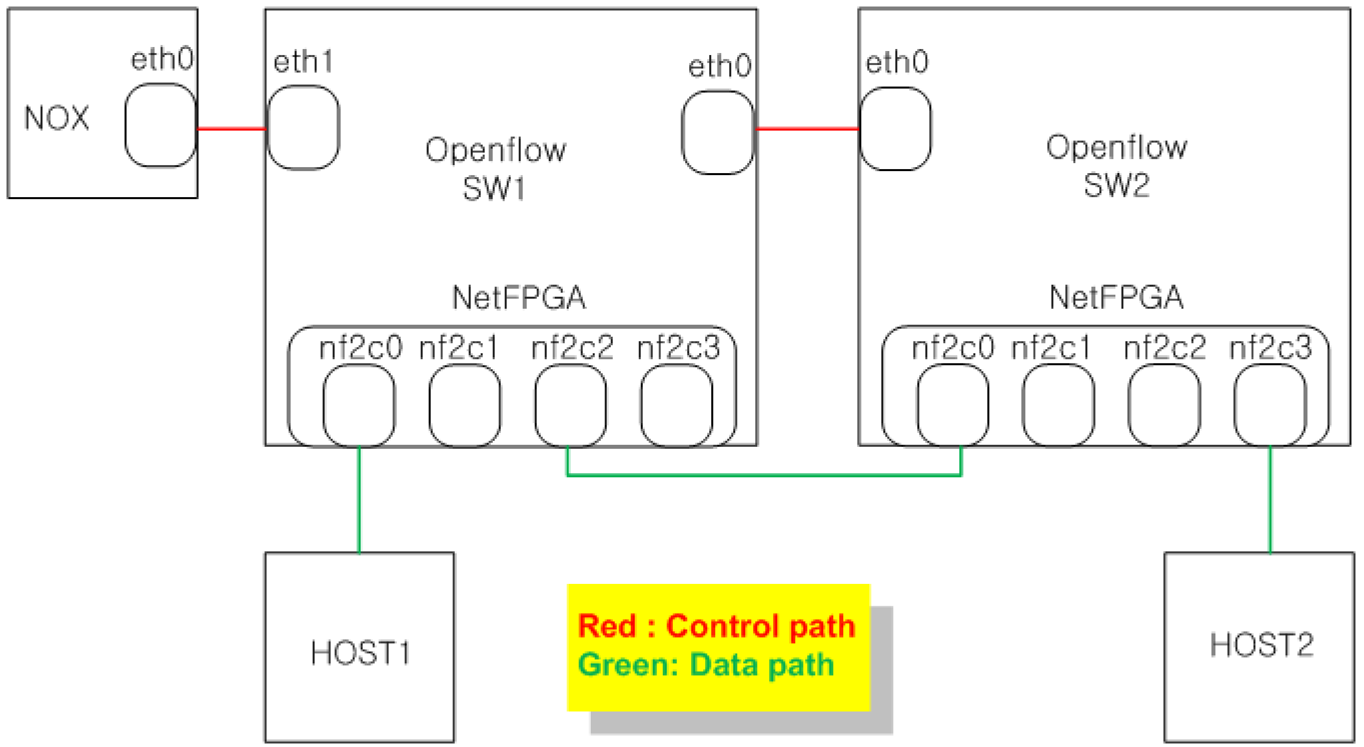
- For tunneling, *ip_gre* module in Linux kernel is used.



SNU-Postech Link

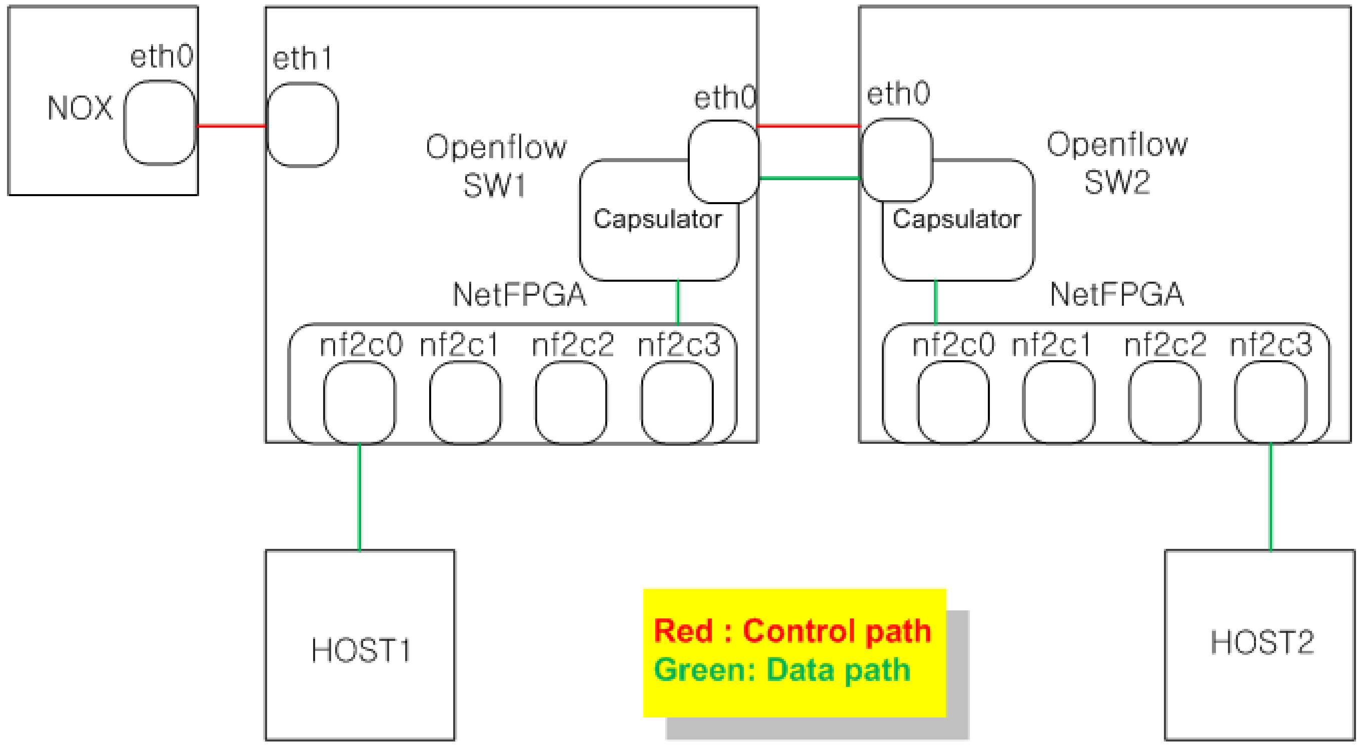


Performance w/o GRE (in Lab.)



- Throughput = 905Mbps, packet loss rate = 0.00088%
  - Using iperf
    - UDP traffic, packet size = 1472bytes, run time = 1200seconds

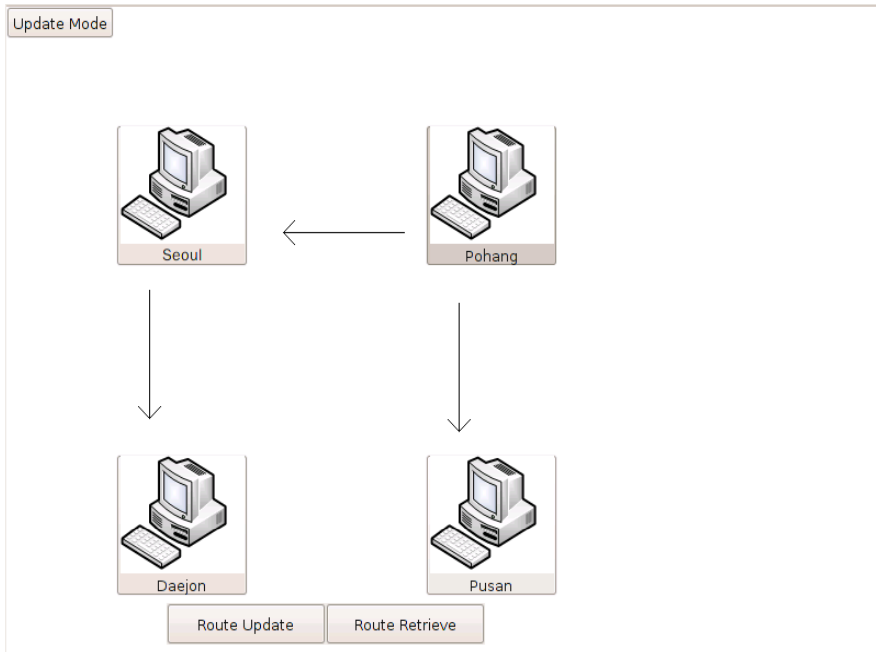
Performance w/ GRE (in Lab.)



- Throughput = 172Mbps, packet loss rate = 0.021%
  - Using iperf
    - UDP traffic, packet size = 1472bytes, run time = 1200seconds

What we did

- Performance in overlay network over KOREN
  - around 90Mbps
  - because of the link capacity in campus network
- Dynamic routing path controller



References

- Jorge Carapinha, Javier Jimenez, "**Network Virtualization – a View from the Bottom**," SIGCOMM VISA Workshop, August 2009.
- http://NetFPGA.org/
- Jad Naous, Glen Gibb, Sara Bolouki, and Nick McKeown, "**NetFPGA: Reusable Router Architecture for Experimental Research**," SIGCOMM PRESTO Workshop, August 2008.
- http://www.openflowswitch.org/
- Nick McKeown, Tom Anderson, HariBalakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner, "**OpenFlow: enabling innovation in campus networks**," SIGCOMM Computer Communication Review, March 2008.
- http://noxrepo.org
- Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, Scott Shenker, "**NOX: Towards an Operating System for Networks,"** SIGCOMM Computer Communication Review, July 2008.
- http://www.koren.kr

THE
END

Thank You