Implementation of a Programmable Service Composition Network using NetFPGA-based OpenFlow Switches

Choi Yun Chul

2010.06.14

CNU

Contents

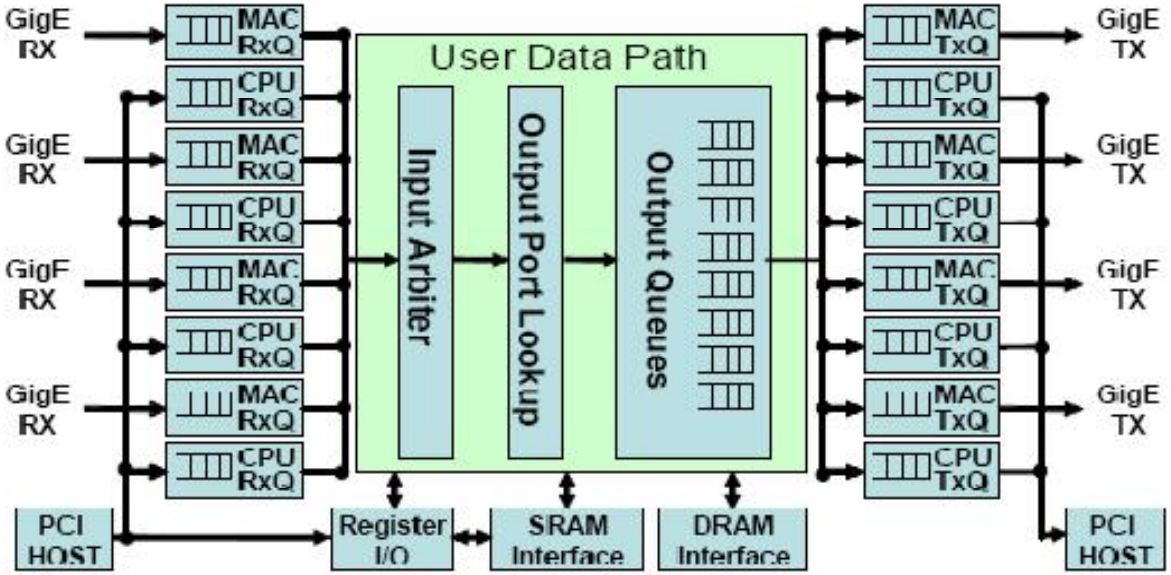
- Introduction
- Related work
  - NetFPGA Platform
  - OpenFlow Switch
  - Service Composition
- NetFPGA/OpenFlow-Based Programmable Network
  - Overall Architecture
  - GUI for Network and Flow Visualization
  - QoS Routing Control Framwork
  - NetFPGA-based hardware-accerelated Capsulator
- Experimentation
- Conclusion

Introduction

- FiRST(Future Internet Research for Sustainable Testbed)
  - NetFPGA-based OpenFlow switches are deployed
  - New service operation and control framework on dynamic virtualized slices is applied
- NetFPGA-based Capsulator module
  - One local testbed should be encapsulated at the sending OpenFlow switch and tunneled to the destination OpenFlow switch
- Programmable service composition
  - QoS routing algorithm
    - Calculate optimal routing path using service requirements table and network monitoring components
    - OpenFlow controller whenever topology or network status changes

Related work

- NetFPGA Platform
  - Handle packet processing at line rate without CPU participation
  - Designed in a modular fashion to allow users to modify or reconfigure modules to implement other useful devices

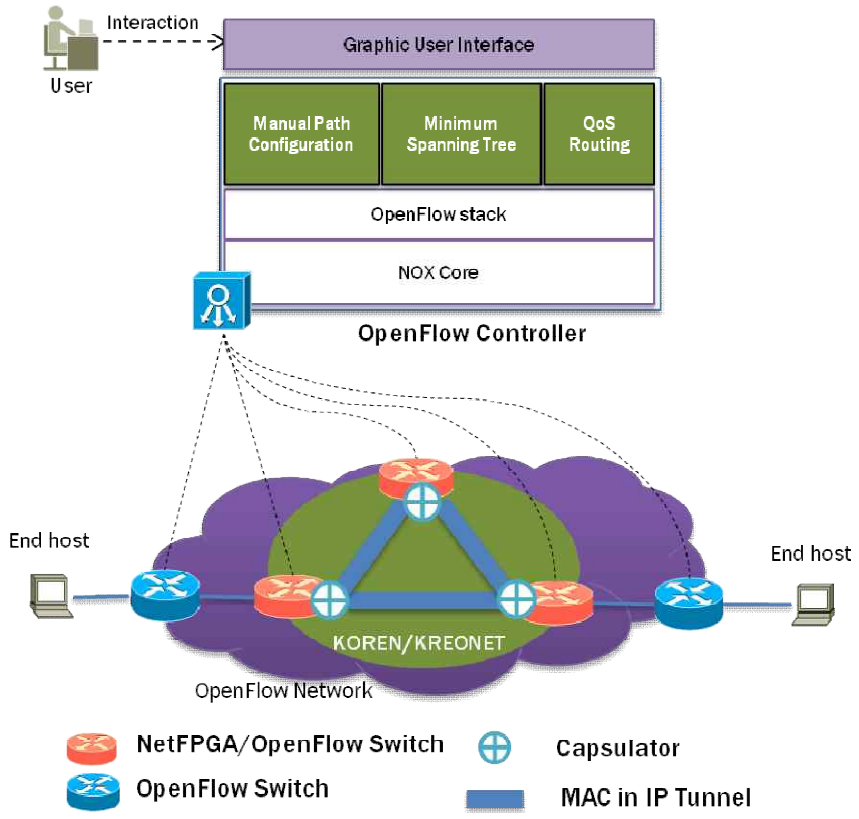


Related work

- OpenFlow Switch
  - Open protocol to program the flow table in various switches and routers
  - OpenFlow controller can add or delete forwarding entries
    - packets are forwarded according to the centralized controller's decision
  - Classifies packets into flows based on 10-tuple
    - Find the appropriate actions associated with the flow
  - Intelligence of network traffic control
    - Datapath element that forwards between ports as ordered by the controller
    - Multiple independent experiments run on different sets of flows
- Service Composition
  - Efficient provisioning and improved reusability of components in building applications

Programmable Network

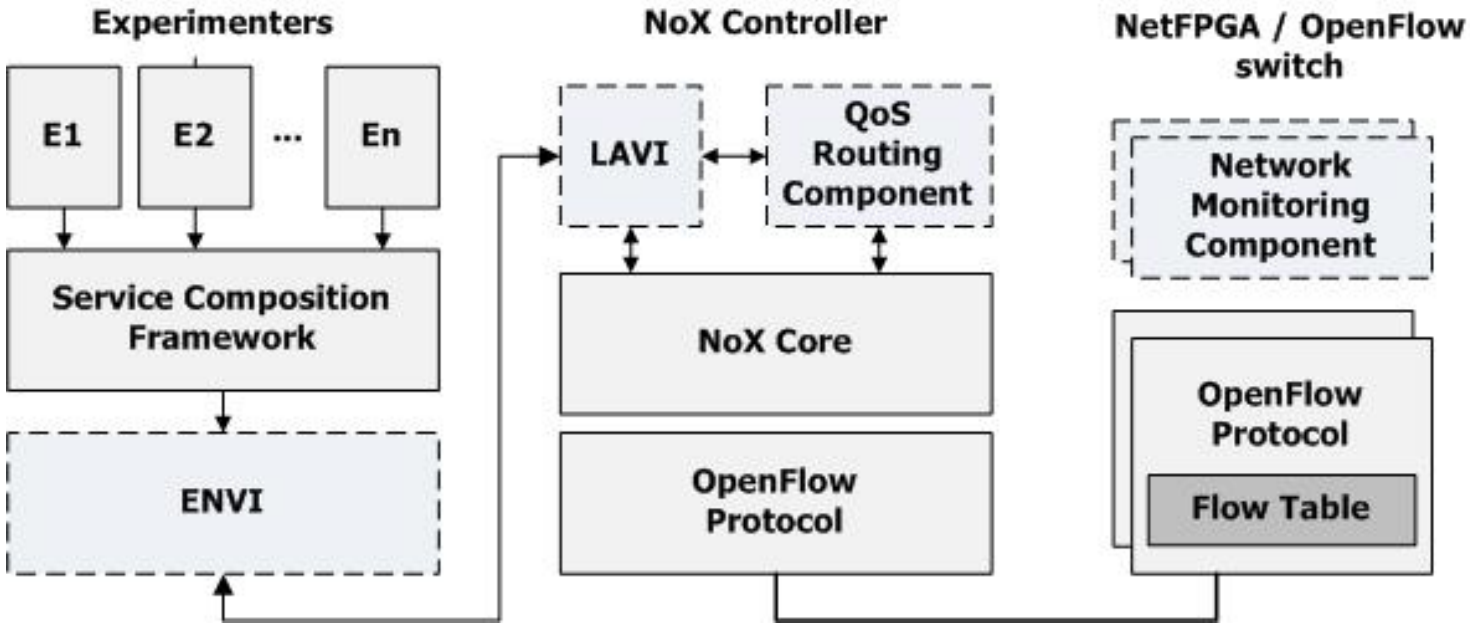
- Overall Architecture
  - Network substrate consists of interconnections between NetFPGA/OpenFlow switches



Programmable Network

- GUI for Network and Flow Visualization
  - Implemented GUI on the OpenFlow controller by extending ENVI

- QoS Routing Control Framwork



Programmable Network

- Experimental environment that guarantees QoS for experimenters
  - Performs service composition experiment on OpenFlow testbed
  - Request a slice with service requirement
  - The slice can be consist of nodes that have an ability to perform service composition
  - The slices can work independently each other

- Proposed QoS routing control framework
  - ENVI
    - Developed to visualize OpenFlow network situation
    - The experimenter is able to check the status of link connection on its own slice
  - LAVI
    - One of NOX components plays a role as a backend server of ENVI
    - ENVI sends service requirements to LAVI and then LAVI delivers this information to QoS routing component

Programmable Network

- Network monitoring component
  - Observes network conditions : available bandwidth,link delay,packet loss,queue status
  - Sends to QoS routing component periodically
- QoS routing component
  - Finds routing path to satisfy service requirement of all slices under time-varying network condition
  - Updates flow table of corresponding OpenFlow switch by QoS routing information

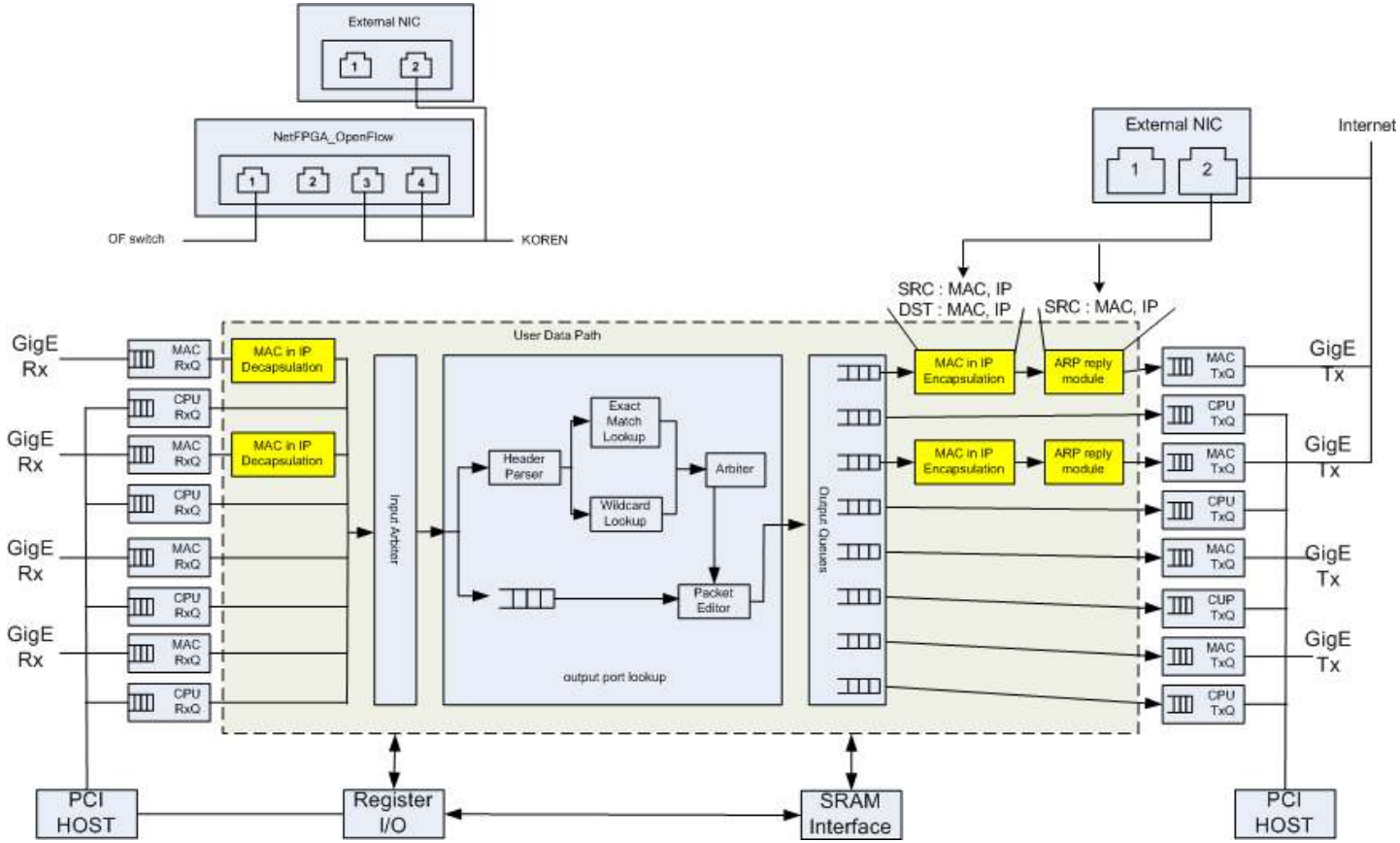
❖ NetFPGA-based hardware-accerelated Capsulator

CTRL	user_data_path							
	Bits 63-56	Bits 55-48	Bits 47-40	Bits 39-32	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0
0x00	mac_dst 48						mac_src_hi 16	
0x00	mac_src_lo 32				mac_ethertype 16		ip_ver 4 ip_header_ler	ip_ToS 8
0x00	ip_total_length 16		ip_id 16		ip_flags 3 ip_flag_offset 13		ip_TTL	ip_prot 8
0x00	ip_header_checksum 16		ip_src 32				ip_dst_hi 16	
0x00	ip_dst_lo 16		udp_src 16		udp_dst 16		udp_length 16	
0x00	udp_checksum 16		capsulator_reserved 48					

Programmable Network

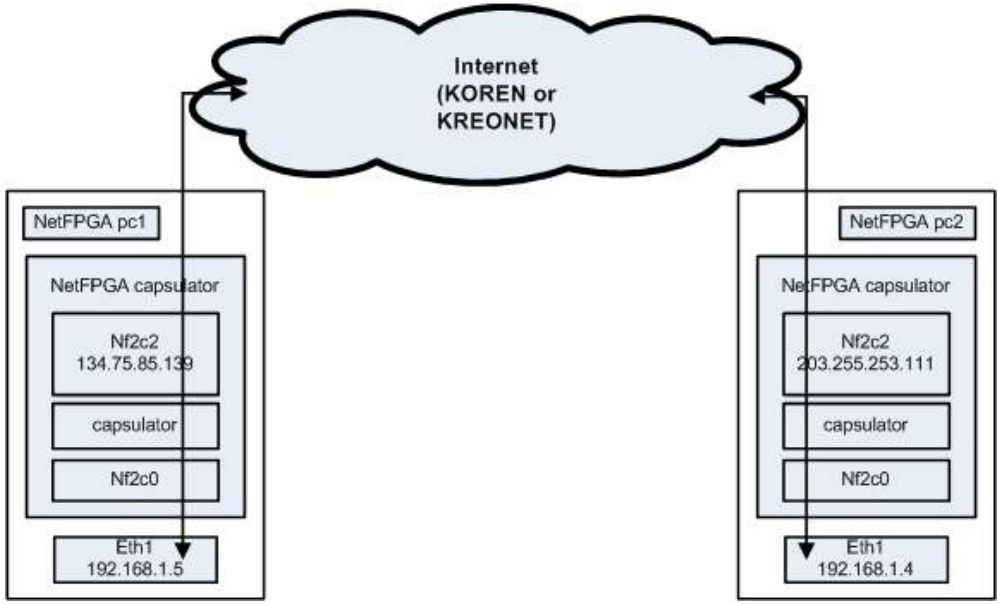
- IP encapsulation header format
  - Source (destination) IP address is set as transmitting (receiving) capsulator's IP address
  - Get a MAC address of gateway which is connected to OpenFlow switch
    - Obtain gateway MAC address by looking up ARP table of kernel-level OS
    - Fill the register of NetFPGA-based OpenFlow switch using obtained gateway MAC address and user-input network information
    - Download OpenFlow_capsulator bit file and load OpenFlow kernel module using insmode command
    - Run script for OpenFlow switch operation
- ARP operation
  - Using register interface, enter MAC and IP address of Capsulator for ARP reply.
  - After monitoring an entered frame at the Capsulator's port, check if it is an ARP request or not by comparing MAC frame type and destination. If it is an ARP request, activate arp flag and record MAC and IP address.
  - If the arp flag is set to 1, ARP reply frame is made using an entered MAC and IP information and transmitted to the Capsulator port.

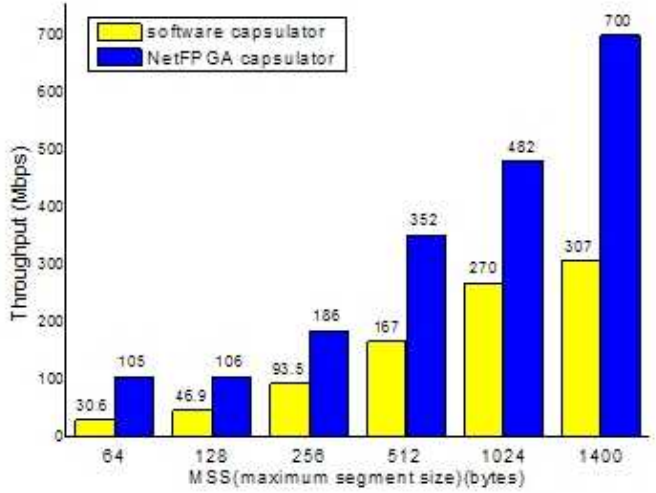
Programmable Network



Experimentation

- Tested NetFPGA-based hardware Capsulator
  - Using iperf [14] program for various TCP MSS





Conclusion

- Introduced flow-based Openflow protocol and Future Internet testbed using programmable NetFPGA system
- Implemented hardware Capsulator for MAC-in-IP tunneling to interconnect separate local Openflow testbeds
- Deployed programmable service composition network and applied QoS routing based on user requirements and network status
- We'll deploy MediaX Library for developing media-oriented component service using this testbed
  - Capture/Display library for heterogeneous input/output device, communication library for flexible media contents and component service module for compressing, decompressing, mixing, tiling and display services will be developed