Prototype of a Programmable Computing/Networking Switch for Multi-Screen Content Consumption

Namgon Kim, Jae-Yong Yoo, and JongWon Kim

Networked Computing Systems Lab., Gwangju Institute of Science and Technology (GIST)

{ngkim, jyyoo, jongwon}@nm.gist.ac.kr
Contents

• Introduction

• NetOpen Networking Service

• Design of a NetOpen Switch Node with In-network Processing Support

• Experimental Verification

• Conclusion
Introduction

- Emergence of various IP-enabled consumer devices
  - The demands of accessing video contents from heterogeneous devices with different screens and different capabilities are increasing
- Real-time content adaptation is challenging
  - Transcoding the content while maintaining good video quality
  - Maintaining appropriately short input to output latency
- We introduce a programmable switching node supporting in-network processing with balancing its use of computing/networking resources
Networking Service and SDN

• Networking Service
  – The collection of network-centric services that assist the transport of diverse flows for computing–centric services

• SDN (Software-Defined Networking)
  – Restructures network and exposes network APIs so that any software can program the network as they want

• By employing the network programmability of SDN, we attempt to fill the gap caused by existing network services
NetOpen Networking Service
Features of NetOpen Networking Service

• Extended flow-based networking:
  – An extension of flow-based networking to reinforce the balanced utilization of both networking and computing resources

• Primitive-based loosely coupled creation of services
  – *Service primitives* as linkage points between NetOpen networking services and programmable network substrates
  – Identify and then link the key features (from resource capabilities)

• Interactive service operation via intuitive UIs
A Road Map to Build SmartX Nodes

- **NetOpen Node v1.3**
  - OF + Click + GPU + Storage + 10G NIC

- **NetOpen Node v1.1**
  - OF + Click + GPU

- **NetOpen Node v1.0**
  - OF + NetFPGA

- **NetOpen Switch Nodes**

- **SmartX Node (NetOpen + MediaX)**

- **Mobile SmartX Nodes**
  - **Mobile SmartX Node v0.9 (tablet, netbook, laptop)**
  - **Mobile SmartX Node with GPU (tablet, netbook, laptop)**

- **MediaX Cloud Node**
  - Cloud-based MediaX Node v1.2
  - PC-based MediaX Node with GPU v1.1
  - PC-based MediaX Node v1.0

- **NetOpen Node v2.0**

- **PC-based MediaX Node with GPU v1.1**

- **OpenVSwitch+Cloud**

- **Wireless NetOpen Node v1.1 (wallBox)**

- **MediaX Nodes**

- **Wireless MediaX Cloud Node**

- **OF + Click + GPU + Storage + 10G NIC**

- **OF + Click**

- **OF + Click + GPU**

- **OF + NetFPGA**

- **Sep. 11, 2012**

- **Networked Computing Systems Lab. GIST**
NetOpen Switch Node (v1.2)

• A Programmable Networking Switch Node with In-network Processing Support

• Design issues
  – Independent processing module for a service functionality
  – For each flow, a customized data plane can be built by selectively combining processing modules
  – This buildup can be controlled by a logically centralized controller
In-network Processing leveraging Heterogeneous Computing

• In-network Processing
  – Additional processing on the packets before forwarding
  – Middle-boxes provide required processing functionalities, e.g., packet caching, transcoding, network coding

• Leverage Heterogeneous Computing (with CPU/GPU/…) for in-network processing
  – Powerful CPUs with multiple cores (e.g. Intel Xeon E7-2820, AMD FX-8150)
  – GPUs & GPGPUs (General Purpose Graphics Processing Units) with hundred cores (NVIDIA GTX580: 512 CUDA cores)
How to handle In-network Processing

- In-network processing for a flow is explained by a task, defined as a set of actions (provided by processing modules) mapped to a flow
  - For example, a task for a video flow can be composed of decode, resize, and forward actions with segmentation and packetization

- Segmentation: Checks the application-layer packet header of each packet and batches the payload into current segment (Assuming the sequential transport of video frames)

- Packetization: Converts a segment back into packets
We need a special purpose task dispatcher between task dispatchers and GPUs, shader, to avoid performance degradation due to multiple task dispatchers accessing GPU at the same time.
Prototype Implementation

• Hardware
  – CPU: 2 Intel Xeon quad-core CPUs (1.6Ghz) with 12GB memory
  – GPU: NVIDIA GTX590 GPU
    • 2 stream processors (each has 512 cores)
    • 3GB GDDR5 memory
  – NIC: Six 1G NICs, two 2port 1G NICs, and two WiFi NICs

• Software (Click + OpenFlow)
  – Click: A toolkit for building a software modular router by combining Click elements (i.e., a small module that has a specific functionality)
  – Each service primitive is implemented as a Click element that is controlled based on OpenFlow
Supported Primitives for In-networking Processing

- **DXT compression** ($P_{DXT_C}$) / **decompression** ($P_{DXT_D}$)
  - DXT (i.e., S3 texture compression) is a light-weight compression scheme that compresses an image frame based on 4x4-pixel blocks
  - The compression operation of each pixel block is completely independent, thus we significantly reduce the compression time by implementing DXT compression using hundreds of GPU cores

- **Rate shaping** ($P_{rshape}$)
  - Control the rate of traffic received on a network interface
  - Traffic that is less than or equal to the specified rate is sent, whereas traffic that exceeds the rate is dropped or delayed
  - Implemented using BandwidthShaper element of Click

- **YUV2RGB conversion** ($P_{Y2R}$)
  - Recent camera models produce YUV-format video content due to the small size of YUV format (i.e., half of the RGB format)
  - DXT compressor only takes RGB-format pixel blocks, we need a conversion element to support various formats of pixel blocks

- **Resize** ($P_{resize}$)
Per-flow Customized Data Plane
A Processing Element for GPU-based DXT Compression

1) Copy segment to GPU device memory
2) Launch Kernel
3) Copy DXT compressed segment from GPU device memory

Device Memory

GPU (Graphics Processing Unit)

Task Dispatcher

Pass segment

Return DXT compressed segment

Processing Element

Kernel (DXT Compression)

<4x4 RGB pixels>

Color 0
Color 1
xx xx xx xx
xx xx xx xx
xx xx xx xx
xx xx xx xx

32 bits (2x16 bits)
32 bits (16x2 bits)

< DXT1 block >

4x4 pixel block

GPU Cores (e.g., 2x 512 cores for GTX 590)

segment

DXT compressed segment

segment

DXT compressed segment

Host Memory
Experimental Verification

- **Service 1** (Shortest-path Connection)
- **Service 2** (In-network DXT compression)
- **Service 3** (Rate-shaped connection)

- **Flow-based forwarding**
- **GPU-based DXT compression**
- **Rate shaping**
Verification Results (Functionality)

Frames rates in H03 with a 800Mbps background flow injection

Frames rates in H03 with in-network DXT compression (Background: 500Mbps)

Frames rates in H03 with in-network DXT compression (Background: 800Mbps).

Frames rates in H03 with in-network DXT compression (Background traffic in-network rate-shaped from 800Mbps to 500Mbps).
Conclusion

• The NetOpen switch node
  – A programmable network substrate supporting the concept of NetOpen networking service

• Design and prototype a NetOpen switch node providing in-network processing
  – Provides service functionalities as programmable processing modules that can be enabled independently
  – For each flow, a customized data plane can be built by selectively combining service functionalities (under the logically centralized control)

• Future work
  – Enhancing the performance of in-network processing
  – Continue the next versions of NetOpen switch node
Thank you!

Send Inquiry to jongwon@gist.ac.kr

http://nm.gist.ac.kr