Deeply Programmable Network (DPN) through Advanced Network Virtualization

Aki Nakao
University of Tokyo
nakao@iii.u-tokyo.ac.jp
2012/8/24
Network Virtualization Symposium 2012
Deeply Programmable Network

- Network that flexibly adapts to demands by means of not only **programmability in control plane** for network operations and management, but also **deeper programmability for data-plane** for processing data and dealing with new protocols (non-IP), etc.
- Extension to **SDN (Software Defined Network)**
Deep Programmability

- **Control-Plane Programmability**
  - Route Control
  - Access Control
  - Network Management

- **Data-Plane Programmability**
  - Packet Data Processing
    - Cache
    - Transcode
    - DPI
  - Handling New Protocols
    - IPvN (N>6)
    - New Layer2
    - Named Data Network (NDN)
    - Content Centric Network (CCN)

Possible with OpenFlow

Hard with OpenFlow +Processors

Deeply Programmable Network

Deeper Programmability
How deep programmability do we want?

Several questions to ask:

- Control plane programmability only?
- Data plane too (cache, transcode, DPI)?
- OpenFlow in A Slice
- Can we program “OpenFlow”, “OpenFlow++”, or “OpenXXX” data-plane APIs? Yes! we can do all!
- Can we change L2 protocol?
VNode Project

- VNode Infrastructure Enables Deeply Programmable Network
  (Project Leader: Aki Nakao)
- 2008-2010 Collaborative Research (NICT/Utoko/NTT/NEC/Hitachi/Fujitsu)
- 2011-2014 Collaborative Research (Utoko/NTT/NEC/Hitachi/Fujitsu/KDDI)
VNode Infrastructure (extended to US!)

- 7 VNodes, 2 Network Connectors, 11 Access Gateways
- Deep Programmability for Experimenting with Arbitrary Protocols (Non-IP)
- Slice-Around-The-World Project (A VNode in U of Utah on ION/StarLight!)

The University of Tokyo Confidential
Deep Programmability for Backbone Network

- Create slices for running **arbitrary protocols**
- Programmability for **both control plane and data plane**
- Mainly focused on **wired / fixed network virtualization**

**VNode Infrastructure**
Deep Programmability for Network Edge

- “Tangible” small-form-factor (1U) VNode
- Deeply programmable, even at L2, yet high performance
- Fixed-mobile converged slicing

Network Edge Slicing

Access Point

Lightweight Slicing

GRE Tunnel

Small-Factor VNode

GRE Tunnel
FLARE
Open Deeply Programmable Switch

Challenges:
FLARE Architecture

FLARE Switch

Controller

Slice N
Slice 2
Slice 1
HyperVisor
(Packet Slicer + Node Manager)

Control Plane
Data Plane
Virtual Ports

Physical Ports

Node Manager
Packet Slicer
Solar Wind Version

SLIVER = Virtual Machine Environment for Programming (C/D Planes)

Create/Remove/Access Sliver

FLARE Controller

Node Manager

Developer Programs

Slicer Controller

Slicer

C-Plane

D-Plane

Virtual Network Interfaces

Physical Network Interfaces
Per-Slice Switching

Slicer Slice

- Slicer Controller
- Slicer
  - SID1
  - Pack
  - Packet

Slice 1

- C-Plane
- D-Plane
  - SID 1
  - Pack
  - Packet

Slice 2

- C-Plane
- D-Plane

Slice N

- C-Plane
- D-Plane

Physical Network Interfaces
Three Primitives in FLARE

• Extract SliceID (Slicer)
• Demux/Mux to/from Slices (Redirector)
• Execute Programs on Slices (Programmer)
Extracting Slice ID

General Slicing
(Supported in DarkFlow Version)

Packet

Arbitrary Offset, Arbitrary Length

Multiple Parts

SID

SID

Packet

Internal Packet Format
Light Weight Network Virtualization
(Application of FLARE to NV)

- Tunneling (Good for Fixed Networks)
- VLAN (Limited to Ethernet/ 12bits)
- MPLS (Originally Traffic Engineering)

Essentially just need globally unique Slice ID
Trailer Slicing

Globally Unique SID Embedded in Trailer
Slicer-slice identifies Slice ID(SID)
Network Virtualization
Through Slice ID Embedded in Packet Trailers

- SID visible at all the layers (if layered architecture is adopted at all)
- VLAN Stitching may not be necessary
FLARE (SolarWind Version) Implementation

- Hybrid of many-core processors + x86 processors
- LXC Resource Container Running on Data/Control Planes
- 1U Form Factor
- Trailer Slicing (Embedding Slice ID in Packet Trailers)
- 4 x 10Gbps (20 Gbps Non-Blocking Switching)
- Up to 16 Data Plane Slices
- Control Plane & Data Plane Linux Programmability (Multi-Threaded Click)
- OpenFlow In A Slice (Implemented)
FLARE Controller

SolarWind rev6431

Welcome to your new Solarwind website!
Multi-Protocol/Control Coexistence

FLARE Switch

Controller

OpenFlow

New L2 Switch

Ether Switch

Node Manager

Packet Slicer

HyperVisor

(Packet Slicer + Node Manager)

Physical Ports

Slice 1

Slice 2

Slice N
# OpenFlow vs. FLARE Differences

<table>
<thead>
<tr>
<th></th>
<th>OpenFlow</th>
<th>FLARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary Frame</td>
<td>X (Internet Protocol)</td>
<td>O (Non-Internet Protocol)</td>
</tr>
<tr>
<td>Control Plane</td>
<td>O</td>
<td>Multiple APIs</td>
</tr>
<tr>
<td>Programmability</td>
<td></td>
<td>OpenFlow++ Multiple OpenFlow SWs</td>
</tr>
<tr>
<td>Data Plane</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>Programmability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Packet Process)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Plane Programming</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>(New Protocol)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### OpenFlow vs. FLARE Differences

<table>
<thead>
<tr>
<th></th>
<th>OpenFlow</th>
<th>FLARE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Programming Model</strong></td>
<td>Flow Pattern Match Action</td>
<td>Linux Container (LXC) Multi-Threaded Click</td>
</tr>
<tr>
<td><strong>Processing Environment</strong></td>
<td>Not Integrated</td>
<td>Integrated</td>
</tr>
<tr>
<td><strong>Slicing Programmability</strong></td>
<td><strong>X</strong> (Fixed to Flow Matching)</td>
<td><strong>O</strong> (fully supported in DarkFlow version)</td>
</tr>
</tbody>
</table>
How deep programmability do we want?

Several questions to ask:

- Control plane programmability only?
- Data plane too (cache, transcode, DPI)?
- Can we program “OpenFlow”, “OpenFlow++”, or “OpenXXX” dataplane APIs?
- Can we change L2 protocol?
A Case in Data Center Network

- Limitation in MAC address space
  - Conflict of MAC addresses in VM migration
- Limitation in VID (802.1Q) space
  - The number of tenants increases in IaaS

Data Center Network depends heavily on L2 leading to solutions such as EUI-64 and VXLAN
Mac Address Extension

EUI-64 (64bit Extended Unique Identifier)

http://en.wikipedia.org/wiki/MAC_address

The University of Tokyo Confidential
Why not just extend MAC ID space through deep programmability enabled by FLARE Switch?
Extended MAC (96bit) Switching

FLARE Switch

Slice 3
Slice 2
Slice 1
HyperVisor

Extended MAC Switching
(Click Implementation)

End System 1
Extended MAC Address
(96-bit MAC)

End System 2
Extended MAC Address
(96-bit MAC)
Performance Check

2 Port 10Gbps Switching Performance (Gbps)
MTU=1500 bytes

![Graph showing Gbps performance for 1 Core, 2 Cores, 3 Cores, and 4 Cores with two data sets: MAC(48bits) and MAC(96bits).]
InterCloud DPN
(VM Migration With Extended MAC)

UTokyo(Tokyo)  Starbed(Hokuriku)

Guest VM  Guest VM  Guest VM  Guest VM
Extended    Extended    Extended    Extended

WAN Migration

FLARE  VNode  FLARE

Extended DMAC  Extended SMAC  Type  IP Datagram
Two Different Kinds of Switching via Tailer Slicing

FLARE Switch

Slice 3
Slice 2
Slice 1

HyperVisor

End System 1
Extended MAC Address (96-bit MAC)

MAC Address
mux/demux

Slice 1

ExDMAC ExSMAC T T IP

Slice 2

DMAC SMAC T T IP

End System 2
Extended MAC Address (96-bit MAC)

MAC Address
mux/demux

Slice 2

ExDMAC ExSMAC T T IP

Etherneet Switching
Extended MAC Switching

34
Examples of Deep Programmability

- Cache Oriented Network Architecture (UTokyo)
- OpenFlow In A Slice (UTokyo)
- Content Oriented Network (Here and There)
- Crazy examples (Non-IP, Extended Mac)
- Artificial Delay for Traffic Control (UTokyo)
Conclusion

- SDN can be extended to Deeply Programmable Network
- We might recall what we initially aimed at, namely, by “Clean Slate” re-design of the Internet
- VNode/FLARE projects now put forth edge area network virtualization and deep programmability