Active Networking On A Programmable Networking Platform

The Openet Team
Nortel Networks Technology Centre
Agenda

Challenge of “Real” Active Networks

Openet: open networking

ORE and Openet Compositions

Active Networking on Openet

Experimental Results

Summary and Future Works
Commercial Network Nodes

Forwarding Faster and Faster

Bandwidth doubles every 9 months
  Optical core: photonic replaces electronic
  New Moore’s Law

Traffic processing
  Terabit switching from Gigabit

But

Ever more use of hardware acceleration (ASIC)
  Filtering, header processing, etc
  Little flexibility to introduce new services

Static and well-defined set of protocols and funcs
  TCP, UDP, and even HTTP
  Allowing configuration rather than addition/modification
Active Networks

A User-Networking Approach

EE residing on active node
- Virtual machine for new protocol processing
- User interface for applications

Capsule or active packets running in network
- “On the fly” protocol composing by applications
- “protocol processors” with customers’ intelligence

But

Implementations mostly in host systems
- Model: Linux and Java
- Not seen in commercial network nodes
Challenge and Our Solution

Goal: Active Networks on real Internet
Active Networks requires
Open boxes
Networking programmability

Commercial Network Nodes
Lack the above two
Have diversified systems

Solution
A programmable networking platform on device
Openet !!!
Active Networks through Openet
Agenda

Challenges of “Real” Active Networks

Openet: open networking

ORE and Openet Compositions

Active Networking on Openet

Experiment and Results

Summary and Future Works
The Openet Platform

Open networking through programming
Programmability to commercial network devices
Service-enabled networking platform

Easing service creation and deployment
Value-added services across network elements
Dynamic, safe and convenient
Not degrading network performance and reliability

Standards and Partners
Active Networks, IEEE, IETF, P1520 and FAIN
Columbia U., UC Berkeley, UPenn and UToronto
MITRE, TASC, NetFuel and CSIRO
Openet Architecture

Control Console (Net Mgr)
- service initiation and policies
- network configuration
- resource administration
- repository maintenance

06/28/18
Openet: a node’s view

Application services
OpletService, Shell, Logger

User Oplets

Standard Services

Function Services

ORE

JFWD

JVM

JNI/Native Code

MEM

CPU

...  

ANTS
Firewall, DiffServ
Jcapture, HTTP, IpPacket

Control Plane

Data Plane

Filtered packets

Monitor status

New forwarding rules

Forwarding Engine

Forwarding Engine

Forwarding Engine

Forwarding Engine

Forwarding Engine
Agenda

Challenges of “Real” Active Networks
Openet: open networking
ORE and Openet Compositions
Active Networking on Openet
Experimental Results
Summary and Future Works
Openet Compositions

ORE: Oplet Runtime Environment
Service creation and deployment
Service downloading and lifecycle management

Services
Every network function is a service
Wrapped by Oplets
Open APIs

ODK: Oplet Development Kit
Ease service creation and encapsulation

Management
Manager on console and Agents on nodes
Service initiation, policy and configuration
ORE: the Openet Core

ORE

Object-oriented Runtime Environment
Services are objects
Run customized software on network nodes
Service downloading, installation, and safe execution
Neutral to heterogeneous hardware
Fully implemented using Java
Services are applications under ORE

4-tier hierarchy

Standard Services
- OpletService
- ManifestOplet

System Services
- JFWD
- JSNMP, JMIB

Function Services
- HTTP, OreServlet
- Shell, Logger

User Services
What’s an Oplet?
A self-contained downloadable unit, or service wrapper
- Encapsulates one or more service objects
- Contains service attributes, e.g., names
- Eases secure downloading and service installation
- Use other service oplets
Agenda

Challenges of “Real” Active Networks

Openet: open networking

ORE and Openet Compositions

Active Networking on Openet

Experimental Results

Summary and Future Works
Passport Routing Switch

New brand name of Accelar

L3 Routing Switch Family

High performance by separating planes

Forwarding Plane
- Distributed ASIC forwarding engines
- Wire-speed forwarding, up to 256 gbps (8600)

Control Plane
- PowerPC
- Network management

JVM available in control plane
Openet on Passport

**Active Networks Services**

**Control Plane**

**Network Services**

- ORE
- JFWD

**CPU System**

- Monitor status
- New rules

**Switching Fabric**

**Forwarding Plane**

(Wire Speed Forwarding)

- Forwarding Rules
- Forwarding Processor
- Statistics & Monitors

**Traffic Packets**

Forwarding Processor

Statistics & Monitors
ANTS (Active Node Transfer System)

Proposed by MIT
Composing and deploying new protocols
Well-packed with toolkit and applications

Service Deployment on Passport
Wrapping the ANTS code without modification
ORE ANTS service
URL: “http://www.openetlab.org/downloads/”
ORE ANTS Service

Service: “AntsNodeService”
Wrapping the MIT ANTS code

Package “com.nortelnetworks.ore.service.ants”
  AntsNodeService.java: service interface
  AntsNodeServiceImpl.java: service implementation
  AntsNodeOplet.java: Oplet
  AntsNode.mf: manifest

Service interfaces
  getNode(): connect to the ANTS EE
  getConfiguration(): set up using ANTS configuration
How AN service is deployed?

Service design and coding
Regular Java programming

Service package
Oplets development by ODK
JAR files
Uploading to downloading servers

ORE start at Passport

Service activation by ORE
Downloading, start and stop
  Startup service
  Shell service

Service execution
Agenda

Challenges of “Real” Active Networks
Openet: open networking
ORE and Openet Compositions
Active Networking on Openet
Experimental Results
Summary and Future Works
Experiment

Goals
Verification: Active Networking on commercial nodes
Performance: ANTS and regular IP applications

Testbed
Part of Intranet: active hosts (Sun Workstations)
Internal network
   Passport: active router
   non-active Linux boxes: for HTTP server and Ping use

Passport gigabit routing switch
Two types: 1100B (10gbps) and 8600 (128gbps)
JVM: JDK 1.1.7 and JDK 1.2

Software
ORE 0.3.3
ORE ANTS service, MIT ANTS 1.2 included
Ping, in Linux 2.2.14
Experiment Testbed

MIT ANTS
(Passport 1100B or 8600 Routing Switch)

Destination Host
(Sun Workstation 1)

134.177.116.108

10.120.101.50
Linux PC
(Ping use only)

10.120.101.102
Download oplets

ORE ANTS

10.120.101.201
HTTP server
(Linux PC)

Source Host
(Sun Workstation 2)

134.177.116.104

10.120.101.51
Linux PC
(Ping use only)
Device Specifications

**Accelar 1100B: 1**
PowerPC 403/66Mhz with 32 MB memory and VxWorks
Role: active router running the ORE ANTS

**Accelar 8600: 1**
PowerPC 740/266Mhz with 64 MB memory and VxWorks
Role: active router running the ORE ANTS

**Sun workstations: 2**
UltraSPARC I/167Mhz with 128 MB memory and Solaris
Role: Source and Destination hosts running MIT ANTS

**HTTP server: 1**
PII/400MHz system with 32 MB memory and Linux 2.2.14
Role: ORE service code and ORE ANTS configuration

**PCs: 2**
PII/400MHz systems with 32 MB memory and Linux 2.2.14
Role: source and destination hosts running regular Ping
ANTS Ping and Linux Ping

Aping testing with 1100B or 8600

MIT ANTS (Passport 1100B or 8600 Routing Switch)

134.177.116.108
Destination Host (Sun Workstation 1)

10.120.101.108
Linux PC (Ping use only)

10.120.101.50
Linux PC (Ping use only)

10.120.101.201
HTTP server (Linux PC)

ORE ANTS

10.120.101.102
APing

Download oplets

Router

ORE ANTS

10.120.101.51
Linux PC (Ping use only)

134.177.116.104
Source Host (Sun Workstation 2)
Data 1: Packet Received

Loss by bursty UDP and slow CPU

Packet received at Source Node
(100 packets or capsules sent)

<table>
<thead>
<tr>
<th>Interval (ms)</th>
<th>Ping</th>
<th>Aping(1100B)</th>
<th>Aping(8600)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>100</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Packets

Loss by bursty UDP and slow CPU.
Data 2: Packet Delays

Is CPU a bottleneck? Yes.

Minimal Average Delays at Source

- **Aping (1100B)**: 31 ms
- **Aping (8600)**: 11 ms
- **Ping**: 0.1 ms

Tests

Delay (ms): 0 5 10 15 20 25 30 35

Tests: 1 2 3
Data 3: Delay Distributions

Slowdown: **Java read and write**
Repeated on Linux and Sun

![Diagram showing delay distributions.]

- Source: Sun Ultra1
- Destination: Sun Ultra1
- Router: Passport
- Java I/O (4)

- Process (ms): 0, 0, 1x2, 1x2, 2x8, 2x0.5, 13, 8
- Aping(1100B) and Aping(8600) bars with percentages 83% and 38%.

06/28/18
Result 1: Capsule Time

Fast CPU speeds up more processing than transmission

<table>
<thead>
<tr>
<th>Time (ms)</th>
<th>Capsule Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Processing</td>
</tr>
<tr>
<td>Aping (1100B)</td>
<td>58%</td>
</tr>
<tr>
<td>Aping (8600)</td>
<td>27%</td>
</tr>
</tbody>
</table>

Fast CPU speeds up more processing than transmission.
Result 2: Throughput

Comparisons at Source

Ping: 10,000 pps
Aping with 1100B: 32.3 cps at maximal
Aping with 8600: 90.9 cps (triple of 1100B)

Passport: ASIC faster than CPU

ASIC processes Ping packets little
CPU processes Aping capsules fully
Watch 1: Performance Up?

Hardware approaches
Fast CPU
Network Processor

Software approaches
Fast JVM
Java network I/O

Combined S/H
Re-engineering software tightly with hardware
   Be aware of hardware-dependency!

Openet neither raises nor worsens performance
Service loading rather than scheduling and monitoring
Watch 2: is it ready?

**AN services on Commercial Node**

*Data-plane*
- Loaded in control plane
- Served with forwarding engines (along the data path)

*Control-plane*
- Loaded and served with the control plane
- Affecting the forwarding engines

**Data-plane services**

*Not* ready if they are
- Time critical
- Large traffic volume

*Yes* if they are
- Loose or elastic time requirement, like ANTS Ping

*Yes!* most control-plane services
- Configuration
- Policies
- Fault management
- Monitor
Agenda

Challenges of “Real” Active Networks
Openet: open networking
ORE and Openet Compositions
Active Networking on Openet
Experimental Results
Summary and Future Works
Openet enables
AN services onto commercial hardware
Customer services portable

Active Networking with Passport
Packet processing: forwarding engine >> CPU
Bottlenecks: CPU, and Java network I/O

AN services ready to real network
Control plane services
Some data-plane services
Other and Future Works

Actual Services
QoS: JDiffServ
Monitor: JCapture

Improving Service Deployment
Service security
Service use of resource

Openet on more hardware
Alteon
IXP
Optical core network

Performance enhance
Better use of CPU and forwarding engines
Visit us at
HTTP://www.openetlab.org

Thank You!
Backups
Standard Services: ODK

Ease customer service creation

Basic Oplet APIs

OpletService
- base service
- extended to create new service description and interfaces

ManifestOplet
- abstract oplet interface
- implemented as Oplets to encapsulate service code

Service start or stop at runtime

Relating service information, e.g., Oplet name
System Services

Access to hardware resource
Public low-level APIs

Examples
JFWD: Java Forwarding
JSNMP: SNMP v2 client
JMIB: various MIB access
System Services: JFWD

Java Forwarding

IP forwarding and routing
  Diffserv marking
  Filtering and diverting
  Forwarding priority
  Routing

Platform-independent APIs

Implemented on Passport/Accelar and Linux
Function Services

Common use utility

Public neutral APIs

Examples

HTTP: HTTP service

Shell: ORE interactive shell

IpPacket: packet construction (IP, TCP, UDP)

Logger: service runtime printout

OreServlet: Java servlet
Applications

ANTS
Active Networking on Passport 1100 and 8600

IP filtering
Dynamic priority changes on Passport 1100

JDiffserv
Diffserv forwarding and DSCP marking on Passport 8600

JSNMP and JMIB
SNMP/MIB access
Passport 1100 and 8600
Linux

JCapture: packet capture

Regatta: fault recovery
## Result: Delays and Throughput

### Bottleneck?: CPU processing

<table>
<thead>
<tr>
<th>Interval</th>
<th>First packet</th>
<th>Average</th>
<th>Throughput (pps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.2</td>
<td>0.1</td>
<td>10000</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1000</td>
<td>0.8</td>
<td>0.1</td>
<td>10000</td>
</tr>
</tbody>
</table>

### APing (1100B)

<table>
<thead>
<tr>
<th>Interval</th>
<th>First capsule</th>
<th>Average</th>
<th>Throughput (cps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3209</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>551</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>139</td>
<td>32</td>
<td>31.5</td>
</tr>
<tr>
<td>1000</td>
<td>131</td>
<td>31</td>
<td>32.3</td>
</tr>
<tr>
<td>1000 (startup)</td>
<td>760</td>
<td>53</td>
<td>19.6</td>
</tr>
</tbody>
</table>

### APing (8600)

<table>
<thead>
<tr>
<th>Interval</th>
<th>First capsule</th>
<th>Average</th>
<th>Throughput (cps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>47</td>
<td>391</td>
<td>2.55</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>11</td>
<td>90.9</td>
</tr>
<tr>
<td>100</td>
<td>12</td>
<td>11</td>
<td>90.9</td>
</tr>
<tr>
<td>1000</td>
<td>13</td>
<td>11</td>
<td>90.9</td>
</tr>
<tr>
<td>1000 (startup)</td>
<td>462</td>
<td>36</td>
<td>27.7</td>
</tr>
</tbody>
</table>