Integrating Active Networking and Commercial-Grade Routing Platforms

The University of Maryland

Rob Jaeger
.rfj@cs.umd.edu

J.K. Hollingsworth
Bobby Bhattacharjee
The Network Paradigm Spectrum

Traditional Networks
- end-to-end connectivity
- well defined protocols
- increasingly perform forwarding in hardware

Active Networks
- on-the-fly service introduction
- per-flow granularity possible
- inject software in data path
Objectives

• Implement flow performance enhancement mechanisms \textbf{without} introducing software into data forwarding path
  
  — Service defined packet processing in a silicon-based forwarding engine
  — Policy-based \textbf{Dynamic} packet classifier

• Create OPEN platform for introduction of new services
  
  — Specify OPEN interfaces for Java applications to control a generic, platform-neutral forwarding plane
  — Enable downloading of services to network node
  — Allow object sharing and inter-service communication
Accomplishments

— JVM on a Silicon-Based Routing Switch

— ORE - Oplet Run-time Environment
  - Java-enabled platform for secure downloading and safe execution of services
  - Ensures required services are installed for a downloaded Oplet

— Java SNMP API (proxy mode for non Java devices)

— Implementation of Network Forwarding API (JFWD)

— RESULT: **Dynamic** Classification in Silicon-Based forwarding engine on a Gigabit Routing Switch
Oplet Runtime Environment Overview

• A platform to dynamically deploy services on network elements

• Desirable properties
  — Portable to many different devices
  — Secure, reliable
  — Low impact on device performance
  — Open
  — Provide a framework to structure code
    — Reusable, maintainable, robust

• Implemented in Java
Basic Concepts

- **Oplet Runtime Environment (ORE)**
  - A kernel that manages the life cycle of oplets and services
  - Provides a registry of services

- **Services**
  - The value being added. Minimal constraints
  - Represented as a Java interface

- **Oplets**
  - The unit of deployment: a JAR file
  - Contains meta-data (e.g., signatures, dependency declarations)
  - Contains services and other resources (data files, images, properties, JAR files)
Architecture

Oplet Runtime Environment

Java Virtual Machine

API Extensions
Oplet Lifecycle

- **Install**
  - Loaded from URL

- **Start**
  - Services that are depended on must already be started

- **Stop**
  - Any oplets that depend on this oplet’s services will be stopped
  - Code and data can be unloaded from ORE

- **Uninstall**
Dependencies

- A service S can use facilities provided by another service T
- This means that the oplet containing S has a dependency on service T
- Before an oplet can be started, all of its dependent services must have been started
- ORE manages dependencies and lifecycle of oplets and services
Some services

- Bootstrap (ORE start time) - basic configuration
- Log - Centralized logging for oplets
- HTTP server
  - Simple servlet support
- Command line shell -
  - service depends on shell to register commands
- Administration commands -
  - Manage oplets and services
- Access to router resource including hardware instrumentation via JMIB
Security Issues

- **Sandbox**
  - Each oplet provides a Java name space and applet-like sandbox

- **Signed oplets**
  - Oplets can be signed for assigning trust

- **Denial of service**
  - Vulnerable to DoS (memory, cycle, bandwidth, persistent storage, monitors) like all Java applications
  - Resource management is a problem
ORE Status

• **Done now**
  - Runs on several Nortel routing products
  - Run on workstations
  - First release of ORE SDK complete
  - JMIB monitor/control system through MIBs
  - JFWD
Future ORE work

• **Capabilities**
  — Revocable services

• **Security**
  — Java 2 style permissions to perform operations

• **Resource limits, DoS protection**
  — Probably requires support from JVM

• **Jini, Oplet Directory - locate and load services**

• **Agents/Services**

• **Open source**
Open Device Architecture

Device Hardware

- Device Code
  - C/C++ API
  - JNI

- Device Drivers
  - Java API

- Open Device Environment (ORE)
  - JVM
  - Operating System

Download Oplet

APIs:
- Java API
- C/C++ API

Service:
- ORE Service
Silicon-based Forwarding Engines

Switching Fabric

Control Plane

CPU

Wire Speed Forwarding

Forwarding Rules
Forwarding Processor
Statistics & Monitors

Forwarding Rules
Forwarding Processor
Statistics & Monitors

Forwarding Rules
Forwarding Processor
Statistics & Monitors

...
Dynamic Configuration of Forwarding Rules

Dynamic Policy

CPU

Forwarding Rules

Forwarding Processor

Forwarding Processor

Forwarding Processor

Forwarding Processor

SW

HW
CarbonCopy Capability
Dynamic Packet Configuration
Dynamic Classification

- Identify real-time flows (e.g. packet signature/flowId)
  1. Use CarbonCopy filters to deliver multimedia control protocols to control plane
     - e.g. SIP, H.323, RTSP
     - Determine dynamically assigned ports from control msgs
  2. Use CarbonCopy filters to sample a number of packets from the physical port and identify RTP packets/signature

- Set a packet processing filter for packet signature to:
  - adjust DS-byte  OR
  - adjust priority queue
JFWD 5-tuple Filtering

- copy the packet to the control plane
- don't forward the packet
- set TOS field
- set VLAN priority
- adjust priority queue
ANTS on Gigabit Router
Demo - 1
ANTS Demo Configuration

- RoutingSwitch loads boot image from TFTP server
- RoutingSwitch dynamically loads Oplets from the Class Server
  - Laptop 1 originates the ping
- Router gets Ping code from Laptop 1.
- Router “evaluates” ping
- Ping forwarded to Laptop2
- Laptop 2 requests code
- Laptop 2 perform ping reply

1. Class Server
2. TFTP Server
ANTS Demo

Laptop 1

AN Ping

AN Ping

ORE Services

Java-enabled Routing Switch

AN Ping

AN Ping

Laptop 2

Demo 1
ANTS Demo
ANTS Demo

- Java application running on the router
- ORE facilitate downloading services
- Interoperable with ANTS Distribution
- Minimum changes to make it conform to ORE service specification
Dynamic Filtering & Configuring
Demo - 2
Dynamic - On the Fly Configuration

- Policy
- Filters

AN Apps

Forwarding Processor

Packet → Filter → Packet

Packet
Dynamic - On the Fly Configuration

- From downloadable Java application, we can modify the behavior of the ASICs
Active Networks Packets Interception
Demo 3 -
Active Networks Packet Capture

AN Apps

CPU

Forwarding Processor

Forwarding Processor

Forwarding Processor

Forwarding Processor

JFWD to Divert or Copy

Wire Speed

Packet
Packet Divert

- Active Network topology is unknown
- ANEP packets NOT addressed to this node are delivered to the control plane for processing
- ANEP daemon receives packets and delivers them to the appropriate EE based on TipoID
Active Networks Packet Capture

• Be able to get the packets from the forwarding plane to the control plane

• Process Active Networks packets in the control plane
Experimental Setup

Source 1
    tcp_send()

Accelear 1100B
Routing
Switch

Source 2
    tcp_send()

Destination
1. tcp_recv()
2. tcp_recv()
Summary

• Developed the ORE for downloading and safely running services onto network devices

• **Without** introducing software into data path we performed **Dynamic Classification of flows in a Silicon-Based Gigabit Routing Switch**
  — Introduced a new service to a Gigabit Routing Switch
  — Identified real-time flows
  — Performed policy-based flow behavior classification
  — Adjusted DS-byte value
  — Showed that flow performance can be improved

For more info email: rfj@cs.umd.edu