Integrating Active Networking and Commercial-Grade Routing Platforms

Rob Jaeger
Tal Lavian, Rob Duncan, Franco Travostino, J.K. Hollingsworth, Bobby Bhattacharjee

Networks 2000
September 13, 2000
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
Outline

• Research Objectives
• Accomplishments
• System Concepts/Components
• Open Device Architecture
• Dynamic Classification Experiment
• Status & Future
• Summary
Research Objectives

• 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

• Implement flow performance enhancement mechanisms without introducing software into data forwarding path
  — Service defined packet processing in a silicon-based forwarding engine
  — Policy-based Dynamic packet classifier
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
System Concepts/Components

• **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 (eg signatures, dependency declarations)
  — Contains services and other resources (data files, images, properties, JAR files)
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**
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
Java Forwarding API

• **Five-Tuple Filters**
  - src/dest IP
  - src/dst port numbers
  - protocol (TCP or UDP)

• **Actions**
  - copy the packet to the control plane
  - drop packet
  - set TOS field or set VLAN priority field
  - adjust priority queue
Outline

• Research Objectives
• Accomplishments
• System Concepts/Components
• **Open Device Architecture**
• Dynamic Classification Experiment
• Status & Future
• Summary
Open Device Architecture

- Device Code
- C/C++ API
- JNI
- Device Drivers
- Java API
- JFWD API
- ORE
- JVM
- Operating System
- Device HW

Download Oplet
Silicon-based Forwarding Engines

Network Services

ORE | JFWD

CPU/Operating System

Control Plane

Filtered packets
New rules

Switching Fabric

Forwarding Rules
Forwarding Processor
Statistics & Monitors

Traffic Packets

Silicon-based Forwarding Engines

Dynamic Configuration of Forwarding Rules

- Dynamic Policy
- CPU
- Forwarding Rules
- Forwarding Processor
- Forwarding Processor
- Forwarding Processor
- Forwarding Processor

SW

HW
CarbonCopy Capability
Outline

• Research Objectives
• Accomplishments
• System Concepts/Components
• Open Device Architecture
• Dynamic Classification Experiment
• Status & Future
• Summary
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
Dynamic Packet Configuration

- Policy
- Filters
- Forwarding Processor
- DSC Service
- Packet
- Filter
- Packet
Experimental Setup

Source 1
\text{tcp\_send()}

100 Mbps

Source 2
\text{tcp\_send()}

100 Mbps

\textbf{1. tcp\_recv()}

\textbf{2. tcp\_recv()}

\textbf{100 Mbps}

\textbf{1. tcp\_recv()}

\textbf{2. tcp\_recv()}

\textbf{100 Mbps}
TCP Behavior with Dynamic Priority Filters

![Graph showing TCP behavior with dynamic priority filters. The graph plots Mbps against seconds. There are two lines, one for low priority and one for high priority. The graph shows the start and end of the 2nd flow, and a change in priority at the 4th second.](image-url)
Outline

- Research Objectives
- Accomplishments
- System Concepts/Components
- Open Device Architecture
- Dynamic Classification Experiment
- Status & Future
- Summary
Status

- Runs on several Nortel routing products
- Run on workstations
- First release of ORE SDK complete
- JMIB monitor/control system through MIBs
- JFWD
- Ported ANTS Execution Environment as ORE service
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**
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
Integrating Active Networking and Commercial-Grade Routing Platforms

Rob Jaeger
Tal Lavian, Rob Duncan, Franco Travostino, J.K. Hollingsworth, Bobby Bhattacharjee

Networks 2000
September 13, 2000
Architecture

Oplet
Service

Oplet
Service

Oplet
Service

Oplet
Service

Oplet Runtime Environment

Java Virtual Machine

API Extensions
Silicon-based Forwarding Engines

![Diagram showing Silicon-based Forwarding Engines]

- Control Plane
- CPU
- Switching Fabric
- Forwarding Rules
- Forwarding Processor
- Statistics & Monitors

Wire Speed Forwarding
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