Active Network Node in Silicon-Based L3 Gigabit Routing Switch

Tal Lavian 1,2 tlavian@cs.Berkeley.edu

Rob Jaeger 2,3 rfj@cs.umd.edu

1 UC Berkeley Engineering
2 Technology Center, Enterprise Solutions, Nortel Networks
3 Department of Computer Science, University of Maryland

Active Networks Workshop 1999
Outline

- Bridge between research and marketplace
- Implementation of Commercial Grade Active Networks node on Silicon-Based Gigabit L3 Routing Switch
- Demo1 - ANTS on Accelar
- Demo2 - Dynamic Filtering & Configuration
- Demo3 - Packets Capture
- Future: Active Networks Routing Protocols
Bridge between research and marketplace

- Bridges the gulf between theory and practice
- Active Networks - industry participation
- Publish the Accelar JVM and development environment
- Publish Linux simulator, tools, and docs
- Scaling up Active Networks Routing Protocol to commercial networks
  - Heterogeneous Topology - AN/Non AN
Accomplishments

- JVM on a silicon-based L3 Routing Switch
- ORE - Oplet Run-time Environment
- Java-enabled Device Architecture
- Active Networks apps that dynamically control and modify Silicon-Based Forwarding
- Packet Interception
- Implementation of Network Forwarding API
Accelar - Industrial Strength Active Networks Implementation

- Up to 96 Gigabit ports
  - (or 384 10/100Mbs + combinations)
- 50Gbps L3 Switching capacity
  - Scaling up to 256Gbps
- Natural Migration - 10Gbs
  - WAN OC-192
- Wire speed - low latency
- High availability
- LAN/MAN/WAN
  - 5km multi mode, 50km single mode
Separation of Control and Forwarding Planes

Centralized, CPU-based Router

Control + Forwarding Functions combined

Forwarding-Processors based Router

Control separated From forwarding

Routing SW

CPU

Slow

Wire

Speed
Active Networks - Node Architecture

Device HW

Device Code

C/C++ API

JNI

Device Drivers

JFWD API

ORE Service

Oplet Runtime Env

JVM

Operating System

Device HW

AN Packet Interception

Oplet
ORE - Oplet Run-time Environment
ORE - Oplet Run-time Environment

ORE

Service A

Service B

Service C

Oplet 1

Oplet 2

JVM

Why ORE?
ORE Protection

- ORE uses JVM mechanisms to:
  - protect itself from the Oplets
  - protect Oplets, one from another
- Mechanisms include features of the Java
  - type safety, access control, ByteCode verification
  - built-in sandbox security manager support
  - signed code
  - strong cryptography infrastructure
• Java facilities are buttressed by ORE control over the allocation of as many of the system resources as possible
• Extra JVM support is necessary to protection against misbehavior by Oplets
  • Accounting of memory and CPU consumption
  • Promising possibility for memory accounting:
    • the ability to partition the object heap to enforce limits on the memory usage by an Oplet
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

Laptop 1

Laptop 2

Java-enabled Routing Switch

ORE Services
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 Configuration of Forwarding Rules

AN Apps

CPU

Forwarding Rules
Forwarding Processor

SW

HW
Real-time forwarding Stats and Monitors

AN Apps

CPU

Forwarding Rules
Forwarding Processor
Statistics &Monitors

Forwarding Rules
Forwarding Processor
Statistics &Monitors

Forwarding Rules
Forwarding Processor
Statistics &Monitors

SW

HW
Dynamic - On the Fly Configuration
Dynamic - On the Fly Configuration

- From downloadable Java application, we can dynamically modify the behavior of the Forwarding Processors (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 TypeID
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
- Enabler for Active Network routing protocols
Scaling up Active Networks Routing Protocol to commercial networks
Scaling up Active Networks Routing Protocol to commercial networks

- Overcome the need to predefine the next hop
- No need to know AN topology a head of time
- Divert/CarbonCopy specific packets to control plane (e.g. packets on ANEP port)
- Wire speed of all other packets
- End to end forwarding
- Future: Active Networks Routing Protocols
Mixed Topology of AN system

NO need to know the AN topology ahead of time

- AN Node
- Non AN Node
Virtual Topology of AN system

NO need to know the AN topology ahead of time
Summary

- Bridge between research and marketplace
- Implementation of commercial grade Active Networks node on Silicon-Based Gigabit L3 Routing Switch
- ORE - Oplet Run-time Environment
- Demo 1 - ANTS on Accelar
- Demo 2 - Dynamic Filtering & Configuration
- Demo 3 - Packets Capture to control plane
- Future: Enables Active Networks Routing Protocols