Open Programmable Architecture for Java–enabled Network Devices

A Revolution!

Tal Lavian
Technology Center
Nortel Networks
tlavian@IEEE.org
Openly Programmable devices enable new types of intelligence on the network
Agenda

- Our market is changing
- Local Computation
- Architecture
- New types of applications
- ORE - Oplet Run-time Environment
- API’s
- Summary
Our Market is Changing

- Customers demand for Openness & programmability
- IEEE P1520
- Lucent’s Programmable Networks
- Intel as driving force
Lucent in Open Programmable Networks

- Lucent’s Programmable Network Conference 9/15-16
  - Cosponsored: Sun, HP, Oracle, Novell, Compaq .......
  - Over 500 participants, enthusiastic response
- SoftSwitch as programmable network
  - 70 developers
  - Jun 99 - PR, Sept 99 - Prog Conference, October 99 - SDK, Jan 00 - APIs, Feb 00 - Dev Conference, May 00 - 3rd party Test Lab
- 7 R/E program
- $1.7B- Excel Switching acquisition (Programmable switch)
- Other products and directions in Openness and Programmability
- Marketing and PR on Lucent’s Openness
ES – Openet Center

- **Create an open development community** to deliver customer-valued solutions based on Nortel Networks’ and partners’ products and technologies
- Openet Center creates a focus to open network platforms
- Openet Center promotes Open Network Computing
- It potentially changes landscape and rules of the networking industry
Accomplishments

- JVMs Network devices
  - Switch, Router, Net-device
- ORE - Oplet Run-time Environment
- Java-enabled Device Architecture
- Java SNMP MIB API
- Implementation of Network Forwarding API
- Dynamic applications
Our market is changing
Local Computation
Architecture
Applications
ORE - Oplet Run-time Environment
API’s
Summary
Changing the Rules of the Game

- Move Turing Machine onto device
  - Add local intelligence to network devices

```java
while (true) {
    doLocalProcessingOnDevice();
}
```
Technology Concept

Web Server → Applet → Web Browser

The JVM is in the Browser

Download applications for local processing

Reversed Applet → Server
Browsers
- Introducing JVM to browsers allowed dynamic loading of Java Applets to end stations

Routers
- Introducing JVM to routers allows dynamic loading of Java Oplets to routers

This Capability WILL Change Everything
Example: Downloading Intelligence

- Dynamic loading
- Authentication
- Application

Network Device

<table>
<thead>
<tr>
<th>JVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
</tr>
<tr>
<td>HW</td>
</tr>
</tbody>
</table>

Monitor
React
Security and Stability

- secure download of Java Applications
- safe execution environment
  - insulate core router applications from dynamically loaded applications
Static-vs-Dynamic Agents

- **Static**
  - SNMP set/get mechanisms
  - Telnet, User Interfaces (cli, web, etc…)

- **Dynamic closed-loop interaction on nodes**
  - capable of dealing with new and difficult situations
  - autonomous and rational properties.
  - dynamically system monitoring & modification
  - report status and trends
Agenda

- Our market is changing
- Local Computation
- Architecture
- Applications
- ORE - Oplet Run-time Environment
- API’s
- Summary
ORE – Oplet Run-time Environment

ORE

JVM

Why ORE?
Java-enabled Device Architecture

Device HW

Device Code

C/C++ API

JNI

Java API

Device Drivers

JFWD API

ORE Service

Oplet Runtime Env

JVM

Operating System

Device HW

Download
Green Threads -vs- Native Threads

Native threads:
- provides non-interference between Java applications
- difficult thread-to-thread communication and sharing of data between threads
- creates a dependency on underlying RTOS
- multiple JVM instances consume resources

Green Threads
- single JVM must manage CPU & memory resources between concurrently running threads
Centralized, CPU-based Model

Distributed, line-card based Model

Routing software w/ COTS OS

CPU

Buffer memory

Line card (forwarding)

Line card (forwarding)

Routing software w/ router OS

Routing CPU

Buffer memory

Centralized, CPU-based Model

Distributed, line-card based Model

Control + Forwarding Functions combined

Added scalability, Flexibility, extensibility

Control separated From forwarding

Evolution of Router Architecture
Explicit Separation of Control Plane from Data Forwarding

Traditional device

Forwarding/Flow/filter Table Download

Forwarding Element

Control Element

Packet Flow

Forwarding Element

Forwarding Element

Forwarding Element
Separation of Control and Forwarding Planes

Centralized, CPU-based Router

- Control + Forwarding Functions combined
- Slow

Forwarding-Processors based Router

- Control separated From forwarding
- Wire Speed
Open Networking Architecture

Vertical Proprietary

Networking Box Level Hardware

- Proprietary Apps
- Proprietary NOS
- Custom Switch ASIC’s

Today

Open

Unified policy-based management

Application server

Control element

Forwarding element

Server Operating System

Network Services Protocol

Network Services Objects

Network OS

Connect Transport Interface

Real-time OS

Network Si

IP Telephony

Policy Server

VPN

Firewall

Copyright - Intel
Dynamic Configuration of Forwarding Rules

AN Apps

CPU

Forwarding Rules
Forwarding Processor

Forwarding Rules
Forwarding Processor

Forwarding Rules
Forwarding Processor

Forwarding Rules
Forwarding Processor

SW

HW
Real-time forwarding Stats and Monitors
Dynamic - On the Fly Configuration

Policy

Filters

Dynamic Apps

Forwarding Processor

Forwarding Processor

Packet

Filter

Packet

Packet
Packet Capture

Oplet

CPU

Forwarding Processor

JFWD to Divert or Copy

Wire Speed

Packet

Forwarding Processor

Forwarding Processor

Forwarding Processor

Forwarding Processor
Green Threads -- Present RTOS with single unified task that includes:
- Java VM (JVM)
- Java Resource Manager (JRM)
  - thread scheduling
  - manages CPU utilization
    - JVM time-slice is managed by the JRM preemptive thread scheduler
  - internal memory manager (intercepts “new”)
  - garbage collection with priority based on available memory
Multiple threads compete for resources
- memory
- CPU
- persistent storage

Denial-of-service attacks possible
- memory or CPU consumption attacks
- trusted/untrusted service interactions
Our market is changing
Local Computation
Architecture
Applications
ORE - Oplet Run-time Environment
API’s
Summary
Applications

- Active Network Management
  - Proactive Network Management
  - Diagnostic Agents
- Dynamic DiffServ Classifier
- Active Intrusion Detection
- Multicast Caching
- IP Accounting
- Application-Layer Router-Server Collaboration
- Pseudo Default Drop Capability
Download Oplet Service to the device.

Monitor MIB variables
- Might be complex conditions
- Trend analysis
- DiffServ, RMON-II, etc… MIBs

Report “events” to NMS
- drop rate, packets/second

Allow Service to take action

Download application

Adjust parameters based on direction from NMS

Active Network Management

- Extensive access to internal resources
- No more polling

Download

Complex Condition Exceeded

Monitor

Appropriate Application

router

NMS
An Open Service API Example

- SNMP API for Network Management
  - generated automatically
  - allows device-based applications to query MIB
  - device-based application -- query local MIB
  - report trends or significant events
  - initiate downloading of problem specific diagnostic code
  - take corrective action
Proactive Network Management

- Device-based Intelligence is Dynamic
  - Static Management
    - SNMP set/get mechanisms
    - Telnet, User Interfaces (cli, web, etc…)
  - Dynamic Closed-loop Management at Network Node
    - capable of dealing with new and difficult situations
    - autonomous and rational properties.
    - dynamically system monitoring & modification
    - report status and trends
  - Monitor MIB to identify poor performance and notify NMS prior to failures
  - Downloaded service can instantiate new services
Automatic trace-route from edge router where problem exists.
- Each node reached generates a report to NMS
- Trace-route code “moves” to next node in path
- Mobile agents identify router health
- Create logs for NMS
Dynamic DiffServ Classifier

- Set router filters to sample packets from edge device host ports
- Identify real-time traffic (RTP flows)
- Set filter on port to adjust DS-byte value based on policy
- Keep track of filters set
- Remove filters no longer in use
Dynamic DiffServ Classification

- Sample packets, set filters to modify DS-byte for Per-Hop-Behavior modification
Intruder is identified by Intrusion Detection software

Intruder signature is identified

Mobile agent is dispatched in direction of intruder (based on physical port of entry)

Mobile agent “chases” intruder and terminates him (shuts down link, reboot host, notify NMS)
Multicast Caching

- Reliable Multicasting
- Distribute error control throughout multicast tree
- Retransmission a local node keeps control close to lossy links
- Balances processor load away from multicast source
IP Accounting

- Project ABCD (Active Bean Counter in Device)
- Perform usage accounting at edge node
- PreCorrelate/aggregate/reduce accounting record on-site
- $1 rule for billing
- Real-time billing can be realized
- Customize billable resources
- Supports **distributed** computing applications in which network devices participate
  - router to router
  - server to router
- Supports **Intelligent Agents**
- Supports **Mobile Agents**
Application Layer Collaboration Among Routers and Servers

- Server farm load balancing
  - server state monitored; rerouting based on congestion/load
- Auctioning Applications
- Bandwidth Broker
Collaboration with Business Applications

- New paradigm of distributed applications
- Network devices collaborating with business applications
- Application aware routing
Bandwidth Broker Collaboration

- Routers Monitor RMON and DIFFSERV MIB
- Report Per-IP Address, Per Protocol statistic to resource broker
- Adjust DS-byte and Per Hop Behavior based on Bandwidth Broker directions
From downloadable Java application, we can modify the behavior of the ASICs
Agenda

- Our market is changing
- Local Computation
- Architecture
- Applications
- ORE - Oplet Run-time Environment
- API’s
- Summary
Oplet Runtime Environment – An 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
ORE – Basic Concepts

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

- Service
  - The value being added. Minimal constraints, could be anything…
  - Represented as a Java interface

- Oplet
  - 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)
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
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
Some services

- Bootstrap
  - Basic configuration
- Log
  - Centralized logging for oplets
- HTTP server
  - Simple servlet support
- Command line shell
- Administration commands
  - Manage oplets and services
An Example

- Traffic Monitor
  - RSVP MIB Group
  - MIB Engine
    - Loopback SNMP Access
    - Native MIB Variables
  - JFWD
  - HTTP Server
  - Servlet Engine
  - Log

- Native MIB
  - Loopback SNMP Access
  - Native MIB Variables
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
ORE Status

Done now
- Runs on Accelar and workstations
- First release of ORE SDK available internally

To be done
- More APIs and services (MIB, JFWD, Wrapper)
- Security (authentication)
- Oplet updates
- Persistent storage
ORE Future work

- Capabilities
  - Revocable services
- Security
  - Java 2 style permissions
- Resource limits, DoS protection
  - Probably requires support from JVM
- Jini, Oplet Directory
- Mobile Agents
- Open source
Agenda

- Our market is changing
- Local Computation
- Architecture
- Applications
- ORE - Oplet Run-time Environment
- API’s
- Summary
Initial APIs

- Console Logging API
- Generic MIB Access API
- Optimized MIB Access APIs
- Trap Interception API
- Management Authentication API
- Web-Based Management Extensions API
- CLI Extensions API
- MIB Extensions API
- Pluggable Authentication API
- Network Forwarding API
Tools

- **MIB API**
  - Monitor device Management Information Base variables
    - MIB
    - RMON and RMON-II
    - DiffServ

- **Network API (JFWD)**
  - Interface to Filters
    - set packet drop filters
    - intercept packets
    - carbon copy packets while forwarding at line-speed
MIB API Example

• API uses a MIB Map to dispatch requests to variable access routines
  • Different parts of the MIB tree can be serviced by different mechanisms
  • Two main schemes:
    • An ad hoc interface to the SNMP instrumentation layer
    • A generic SNMP loopback
An Open Service API Example

- SNMP API for Network Management
  - generated automatically
  - allows device-based applications to query MIB
  - device-based application -- query local MIB
  - report trends or significant events
  - initiate downloading of problem specific diagnostic code
  - take corrective action
Java MIB API – Proxy Mode

- Uses SNMP loopback mechanism to target a remote network element
- **API can be used to control devices that don’t have an embedded JVM**
Our market is changing
Local Computation
Architecture
New types of applications
ORE - Oplet Run-time Environment
API’s
Summary
Summary

- Programmable
  - Turing Machine on network devices
  - *dynamic* agents vs. *static* agents
  - dynamic loading

- Our market is changing

- Openness - successfully proven paradigm
  - Facilitates *innovation*
  - Domain experts - virtual development community
  - With 3rd parties we can change the networking landscape

- Application aware routing
Compare to this first flight and look where aviation is today
Why Java

- Dynamic Loading
- Reuse security mechanisms
  - byte-code verifier
  - security manager
  - classloader
- System stability
  - constrains applications to the JVM
  - Prohibits native code applications
- Extensible, portable, & distributable services
But Java is slooowwwww

- Not appropriate in the fast-path data forwarding plane
  - forwarding is done by ASICs
  - packet processing not affected
- Java applications run on the CPU
  - Packets destined for Java application are pushed into the control plane
The new concept is secure to add 3rd party code to network devices
- Digital Signature
- Administrative “Certified Optlet”
- No access out of the JVM space
- No pointers that can do harm
- Access only to the published API
- Verifier - only correct code can be loaded
- Class loader access list
- JVM has run time bounds, type, and execution checking
Old model: Not safe to add 3rd party code
  ◦ Dangerous, C/C++ Pointers
    • Can touch sensitive memory location
  ◦ Risk: Memory allocations and Free
    • Allocation without freeing (leaks)
    • Free without allocation (core dump !!!! )

Limited security in SNMP
Bandwidth x200 – start of new demand

- Intel web hosting - BIG pipes
- Last mile bandwidth x 200
- Multimedia and new applications will drive the demand.
### The P1520 Reference Model

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V interface</td>
<td>Algorithms for value-added communication services created by network operators, users, and third parties</td>
</tr>
<tr>
<td>U interface</td>
<td>Algorithms for routing and connection management, directory services etc.</td>
</tr>
<tr>
<td>L interface</td>
<td>Virtual Network Device (software representation)</td>
</tr>
<tr>
<td>CCM interface</td>
<td>Physical Elements (hardware, namespace)</td>
</tr>
</tbody>
</table>

#### Levels
- **Value Added Services Level**
- **Network Generic Services Level**
- **Virtual Network Devices Level**
- **PE Level**
CSIX Consortium

- Common switch interface for switch fabric independence
  - www.csix.org
  - Detailed interface specification between port/packet processor logic and interconnect fabric logic
  - Similar to common media interface such as Utopia, but for switch fabric interface
  - Targeted at scalable switches at higher end
  - Permits mix-and-match of silicon and software components
Open Multi-service Switching
- Common transmission and switching infrastructure
- Modular, layered architecture
- Integration at a module level through open interfaces
- Multi-vendor model with 3rd party software options

Voice Service
ATM Services
IP Services

Source: MSS