FlowOps: Open Access Network Management and Operation

MS Thesis Defense
May 10, 2013
Matt Strum
mstrum@flux.utah.edu
Motivation

NETWORK MANAGEMENT IS HARD

Services  Security
Configuration

Faults

Performance
Monitoring

Switch
A Inc.

Switch
B Inc.

Switch
C Inc.
Motivation

OPEN ACCESS NETWORKS

Closed Access Network

- Network Operator
- Internet
- TV
- Phone

Vs.

Open Access Network

- Network Operator
- Service Provider
- Internet
- TV
- Phone
AUTOMATION AND INTEGRATION

Motivation
Current Solutions

• Still...
  – Vendor oriented
  – Task focused

• Examples
  – Junos Space Network Management
  – HP Automated Network Management
  – Cisco Prime
Method

AUTOMATION THROUGH RULES

Configuration Management
- Affected services
- Fixed configuration

Fault Management
- Fix
- Error!

Network
Thesis Statement

An automated network management and operations framework built on a production rule system can capture the dependencies and relationships of both the network infrastructure and the role players in open access network environments.
Contributions

• **Layered model** defining entities and behaviors
  – **Knowledge Store** holding the entities
  – **Rules Engine** to enforce behavior
• **Driver Engine** for network device support
• **Prototype** demonstrating utility to all actors
Work Built On

• KnowOps [Chen et al. Hot-ICE11] – Unification of:
  – PACMAN [Chen et al. ACM 5th 2009]
    • Network management workflow tasks
  – COOLAID [Chen et al. ACM 6th 2010]
    • Declarative language to capture knowledge from domain experts and documents
  – DÉCOR [Chen et al. SIGCOMM 2010]
    • Database-oriented network management system
1. Introduction
2. Architecture
3. Implementation
4. Evaluation
5. Related Work
6. Conclusion
SYSTEM

Architecture

Network Operator

Service Providers

End Users

API

Abstraction Layers

Knowledge Store / Rules Engine

Driver Engine

ACME Driver

OpenFlow Driver

ACME Switch

ACME Switch

OpenFlow Switch

OpenFlow Switch

Infrastructure
Architecture

VIEWS

Customer X

Customer Y

ISP A

Network Operator

Provided Service

Provided Service
Alert!

Infrastructure
Architecture

**DRIVER ENGINE**

- Service Path
- Path Hop Configuration
- Driver Engine
- Driver
- Switch
- API
- Knowledge Store / Rules Engine
- Abstraction Layers
- Network Operator
Architecture

WORKFLOW

End User
1. Browse services
2. Order service

Network Operator
3. Receive order
4. Reserve slice of open access network
5. Reserve slice of own network
6. Configure all

Service Provider
7. Notify ready
8. Reservation ID
9. Use service

List of services
Resources needed
Reserve
Reservation ID
Resources needed
Reserve
Reservation ID
Configure
Reservation ID
Configure
Architecture

Allocation

- Network Operator
- Service Providers
- End Users

API

Abstraction Layers

Knowledge Store / Rules Engine

Driver Engine

ACME Driver

OpenFlow Driver

ACME Switch

OpenFlow Switch

Infrastructure
Architecture

ALLOCATION
**Architecture**

**ALLOCATION**

**Service Provider**
- Allocate Ethernet service
- Insert Allocation Fact(s)

**API**

**Service Layer**
- Ethernet service
  - Bob:Eth
  - ServiceProvider:Eth
  - If: Ethernet service requested
    - Then: Allocate backbone VLAN

**Network Operator Layer**
- Backbone VLAN Hop
  - Bob:Eth:1

**Infrastructure Layer**
- If: No conflicts exist
  - Then: Allocation is valid
ALERTS

Architecture

Users

Network Operator

Knowledge Store

Business Layer

Service Layer
- Provided Service Alert
  - If: Provided service interrupted
  - Then: Alert actors

Network Operator Layer
- Backbone VLAN Alert
- Hop Alert
- Port Alert
  - If: Backbone VLAN broken
  - Then: Fix if possible, alert results

Infrastructure Layer
- Port Down Alert
  - If: Port Down Alert thrown
  - Then: Throw Alerts on affected services
- Port Down

Driver Engine
- ACME Driver

SNMP Trap: Port down

ACME Switch

Insert Event: Port Down
3. **Implementation**
Knowledge Store

• Service layer services
  – LAN, VLAN
  – E-LINE, E-LAN, E-TREE

• Network operator layer services
  – VLAN

• API for allocation, views, etc.

• Uses Drools rule engine
rule "Port down detected"
when
  portStatus: PortStatus(!up)
  port: Port(id == portStatus.id)
  services: collect(Service(ports.contain(port))
then
  for(Service service : services) {
    insert(new ServiceAlert(service, portStatus))
  }
  modify(port) { up = false }
  retract(portStatus)
end
Driver Engine

• Commands
  – AddVlan, RemoveVlan

• Simulated driver
  – Acts as no-ops
  – Hosts API to simulate faults

• OpenFlow driver
1. Introduction
2. Architecture
3. Implementation
4. Evaluation
5. Related Work
6. Conclusion
Environment

Service provider configuration

Network operator configuration
Emulating the Network
Configuration

Alerts
Configurations

- Used Mininet environment
- Services between various actors
- All combinations of service types
Example Service Definition

<resource_specification>
  <link type="ethernet">
    <endpoint id="ServiceProvider:Port" />
    <endpoint id="Bob:Port" />
  </link>
</resource_specification>
Fault Management

- Used simulated driver
- Brought port down
Fault Management

• Allocate path
• Bring non-critical port down
• Path automatically fixed
  – Notify actors about action
Fault Management

- Allocate path
- Bring critical port down
- Path unfixable
  - Notify actors about state
Evaluation

• FlowOps enables...
  – Actor to infrastructure automation and integration in an open access network environment
1. Introduction
2. Architecture
3. Implementation
4. Evaluation
5. Related Work
6. Conclusion
Related Work

• PRESTO [Enck et al. IEEE 2009]
  – Configuration with transformation from high-level templates

• 4D [Greenberg et al. ACM SIGCOMM 2005]
  – Configuration through layers

• Network federation [Hayashi et al. AICT 2011]
  – Standardize services at the edges

• ChoiceNet [Rouskas et al.]
  – Expose several service layers
Related Work

• Generic Root Cause Analysis (G-RCA) [Yan et al. ACM 2010]
  – Analyze network events using rules

• Network-wide Information Correlation and Exploration (NICE) [Mahimkar et al. CoNEXT 2008]
  – Troubleshoot chronic network issues through analyzing statistical correlations
6. Conclusion
Contributions

• **Layered model** defining entities and behaviors
  – **Knowledge Store** holding the entities
  – **Rules Engine** to enforce behavior
• **Driver Engine** for network device support
• **Prototype** demonstrating utility to all actors
Conclusions

• Layered model simplifies logic
• Rule engine enables automation of...
  – Dependencies in our model
  – Network management tasks
  – Actor to infrastructure integration in an Open Access Network environment
Ongoing Work

• Enhancing prototype implementation
• Moving to test in lab environment
Future Work

• Major
  – More drivers, including traditional switches
  – Create sample services

• Expand network...
  – accounting, performance, and security

• Support more services at service and network operator layers
Acknowledgements

• EntryPoint LLC
  – Jeff Christensen
  – Robert Peterson
Thank you!

ANY QUESTIONS?
Background

OPEN ACCESS NETWORKS

• Public
  • Utah’s Utopia
  • Stockholm’s Stokab
  • Amsterdam’s CityNet

• Private
  • Reggefiber
  • Quadracom
  • MBC’s network
Background

RETE ALGORITHM

Fact A

Exists

Fact B

id == targetID

Rule 1 Triggered

Agenda
**Background**

**DROOLS ENTRY-POINTS**

- **Knowledge Session**
  - **Default Store**
    - Fact A
  - **“X” Store**
    - Fact A

- **Rules**
  - rule “Rule 1”
  - when
  - factA : FactA()
  - factA : FactA() from entry-point “X”
**DROOLS CHANNELS**

Background

**Rule** ...

```
channel[“Y”].send(object);
```

...
**DROOLS SALIENCE**

**Background**

Simultaneously triggered rules

- Triggered Rule 1
  - Salience 3
- Triggered Rule 2
  - Salience 9
- Triggered Rule 3
  - Salience 6

Ordered agenda

- Triggered Rule 2
  - Salience 9
- Triggered Rule 3
  - Salience 6
- Triggered Rule 1
  - Salience 3
Motivation

SERVICES
Motivation

SERVICES
Motivation

ACTORS