SDN Management Layer: Design Requirements & Future Direction

Yuefeng Wang
Joint work with Ibrahim Matta

Computer Science Department
Boston University

Oct 21, 2014
Outline

- Computer Network Management
- Management Architecture for Software-Defined Networks (SDN)
- SDN Management Layer: Design Requirements and Open Issues
- Our Solution to Better Network Management

This talk is about what should be provided
Not about how to provide these requirements
Network Management

- Goal:
  - Monitor and control the network

- Example functions:
  - Routing
  - Measurement and monitoring
  - Security (e.g. access control, attack detection)
  - Traffic engineering (e.g. load balancing)
  - Network virtualization
  - Virtual Machine (VM) migration
  - Virtual network migration
  - ...
A switch (or router) consists of two parts:

- Data plane: forward, filter, buffer, rate-limit, count packets, *etc.*
- Control plane: track topology changes, compute routes, install forwarding rules, *etc.*
Traditional Networks

- Control plane and data plane are tightly coupled in each device
  - A highly decentralized architecture
  - The network is more resilient

- Problems: complex and hard to manage
  - Vendor-specific switch configurations
  - Network configuration is hard to debug due to decentralization
  - A single misconfiguration may lead to severe consequences, e.g., routing loop or black hole
Software-Defined Networks (SDN)

- Enable programmable networks and support flexible network management
- Separation of control plane and data plane
- OpenFlow (McKeown et al 2008) enables SDN
Management Architecture for SDN

Northbound API

Southbound API
Key Idea of SDN

- Policy-based network management
  - Network management is expressed in terms of high-level policies instead of switch configurations that are low-level and vendor-specific
  - Advantages: simplify network management, and enable the provisioning of different services to different users, etc.

- Management layer is the core
  - Map high-level network policies specified by management applications to low-level switch configurations
Design Requirements of Management Layer

- A Global Network View and General API
- Distributed Controllers
- Network Virtualization
- User-level Interface Support
- Network Orchestration
Design Requirements of Management Layer

- A Global Network View and General API
- Distributed Controllers
- Network Virtualization
- User-level Interface Support
- Network Orchestration
A Global Network View and General API

- Goal: simplify the programming of management applications
- Example: NOX (Gude et al. 2008)
  - Network view: includes switch-level topology, and location of users, hosts and services
  - NOX API: management applications register handlers for predefined events
Open Issues

- Problems with NOX
  - Switch-level programming abstraction, which makes programming hard and error-prone
  - Flow rules are not easy to compose, e.g., union or negation are not supported

- Many high-level languages (e.g., FML, Procera, Frenetic and Maple) have been proposed to improve NOX, but
  - Some of them still have to deal with low-level details (per-link or per-switch configurations)
  - **Fragmented efforts and no standard SDN management API**
Design Requirements of Management Layer

- A Global Network View and General API
- Distributed Controllers
- Network Virtualization
- User-level Interface Support
- Network Orchestration
Distributed Controllers

- Goal: deal with scalability and reliability issues
- Example: Onix (Koponen et al. 2010)
  - Network Information Base (NIB) is replicated and distributed over Onix instances through built-in network state distribution primitives
Open Issues

- Concept of scope (our contribution)
  - A network management layer manages a network over a certain scope, which includes physical components (switches) and logical components (processes)
  - Scoping means management layer explicitly defines the subscope for a given policy and dynamically creates new subscopes for new controllers
  - Scoping is important to support policy-based management

- Scoping is not well supported by Onix and other existing SDN management layers
  - Scope is flat, i.e., only one level, and subscope of controllers and policies is restricted to physically close components
Design Requirements of Management Layer

- A Global Network View and General API
- Distributed Controllers
- Network Virtualization
- User-level Interface Support
- Network Orchestration
Network Virtualization

- Goal: improve resource utilization and build isolated testbeds

- Example: FlowVisor (Sherwood et al. 2009)
  - **Slice**: an instance of virtual network controlled by user controller
  - **FlowSpace**: a collection of flows identified by packet headers
Open Issues

- Problems with FlowVisor
  - Scope is flat: virtual network topologies are restricted by the physical topology, where neighboring virtual nodes have to be mapped onto neighboring physical nodes
  - Scope is global: virtual networks do not have a separate virtual flowspace, i.e. no virtual (private) address space

- Many management layers (e.g., ADVisor, FlowN) are proposed to overcome drawbacks above, but
  - Virtualization is provided only on top of networks under a single administrative domain. Virtualization across multiple domains is not supported
Design Requirements of Management Layer

- A Global Network View and General API
- Distributed Controllers
- Network Virtualization
- User-level Interface Support
- Network Orchestration
User-level Interface Support

- Goal: beyond administrative-level interface, enable users to achieve better user application performance
- Example: PANE (Ferguson et al. 2013)
  - A share tree specifies users’ privileges on certain flows
  - Users can ask for certain actions on the network traffic
Open Issues

- Interface with QoS (Quality of Service) support
  - QoS aspects include throughput, loss rate, delay, etc.
  - Provides predictable network service which is crucial for application performance

- PANE provides limited QoS support, where only bandwidth reservation is supported, and no loss rate or delay guarantees

- QoS is not considered as a requirement by most SDN management layers
Design Requirements of Management Layer

- A Global Network View and General API
- Distributed Controllers
- Network Virtualization
- User-level Interface Support
- Network Orchestration
Network Orchestration

- Goal: resolve conflict requests from different applications
- Example: PANE (Ferguson et al. 2013)
  - Hierarchical Flow Tables (HFTs)
    - **Policy tree**: each node stores one or more policy atoms, which is a pair of matching rule (flow) and corresponding action
    - **Conflict-resolution operators**: resolve conflicts between two policy atoms in the policy tree, and semantics can be extended
Open Issues

- Many management layers (such as PANE) provide network orchestration
  - Only focus on how to resolve conflicts between different requests sent to the management layer
- What is not well studied is:
  - How to harmonize more complex policies in the same scope
  - How to compose different policies over different scopes at different levels
Design Requirements of Management Layer

- A Global Network View and General API
- Distributed Controllers
- Network Virtualization
- User-level Interface Support
- Network Orchestration
Main Open Issues in Summary

- QoS Support
  - Tied to TCP/IP architecture, which only provides “best-effort” delivery service

- Scoping Support
  - Policy-based management through scoping is not well supported
  - How to compose policies over different scopes is not well explored
How to Solve these Issues

- Keep adding ad-hoc patches to resolve issues inherited from the TCP/IP architecture

Or

- Build a management framework on top of a new network architecture
  - Solve the communication problem in a more fundamental and structured way
  - Without the shortcomings of the TCP/IP architecture

Our solution: RINA Management Architecture
RINA: Recursive InterNetwork Architecture

- RINA is clean slate network architecture
- Networking is Inter-Process Communication (IPC) and only IPC
- Distributed IPC Facility (DIF) is a collection of IPC processes with shared states
- DIF is the layer that provides communication service
- Separation of mechanisms and policies
- DIF is policy based and dynamically instantiated
- Provide better support for mobility, security and manageability

![Diagram of RINA architecture with layers and processes](image)
RINA Management Architecture

- RINA API simplifies network programming
  - QoS is explicitly supported by RINA API
  - Eliminate the distinction between Southbound and Northbound API
  - Unified API for both network administrators and end-users

- DIF is a network management block
  - DIF defines the scope, and it can be viewed as a virtual private transport network that can span single or multiple domains
  - Policies can be easily configured for different DIFs over different scopes at different levels

- Deployment of RINA
  - Physical nodes support recursive process management as RINA views networking as only inter-process communication
  - RINA can also be deployed in the presence of legacy physical devices through a shim layer (overlay), which forms the base case for RINA’s recursive communication
Summary

- SDN Management Architecture
- SDN Management Layer: Design Requirements and Open Issues
- Better Network Management through RINA architecture
Thank you!

For more details about RINA, please visit us @ http://csr.bu.edu/rina