Improving Network Management with Software Defined Networking

Hyojoon Kim and Nick Feamster, Georgia Institute of Technology
2013 IEEE Communications Magazine

Presented by 101062505 林瑋琮

Outline

1. Introduction
2. Background
3. Framework
4. Case study
5. Conclusion

Introduction

- Tradition network management is challenging. To operate and maintain a network, the operators must deal with low-level configuration to implement high-level policies.

- Since the network devices are gererally closed, the inflexibility of the underlying infrastructure provides few possibilities for improvement.

- A novel paradigm, software defined network (SDN), was introduced.

Introduction

- SDN advocates separating the data plane and the control plane, making switches in the data plane simple packet forwarding devices and leaving a centralized software program to control the behavior of the network.

- In this article, the author identify the problems with the current network configuration and management mechanisms, and introduce mechanisms to improve them.

Background

Difficulties in network management

1. These networks typically comprise a large number of switches, routers, firewalls, and various middleboxes.
2. To enforce various high-level policies and respond to the wide range of network events, the operators need to specify them in terms of distributed low-level configuration.
3. Network state changes continually, and operators must manually adjust network configuration in response to changing network conditions.

Background

Difficulties in network management

4. Today's networks provide little mechanism for automatically responding to such complex environment.
5. Also, today's networks typically involve integration and interconnection of many proprietary, vertically integrated devices, which limits the innovation in network management.
6. It's difficult to introduce and deploy new protocols, since updates are generally published by vendors unilaterally.

Background

Difficulties in network management

Hence, we need a advanced paradigm to manage the networks, which is software defined network.

SDN is a paradigm where a central software program, called a controller, controls the overall network behavior.

In SDN, network devices become simple packet forwarding devices (data plane), while the control logic is implemented in the controller (control plane).

Background

Software defined network

1. It is much easier to introduce new ideas into the network through simply writing a software program.
2. It introduces the benefits of a centralized approach to network configuration. Operators need not to configure all devices individually to change the behavior of the networkm but instead modify the forwarding decisions in a single central controller.

Framework

Southbound

The southbound interface refers to the interface and protocol between programmable switches (SDN-capable switches) and the software controller.

Northbound

The northbound interface determines how to express operational tasks and network policies, and also how to translate them into a form the controller can understand.

Framework

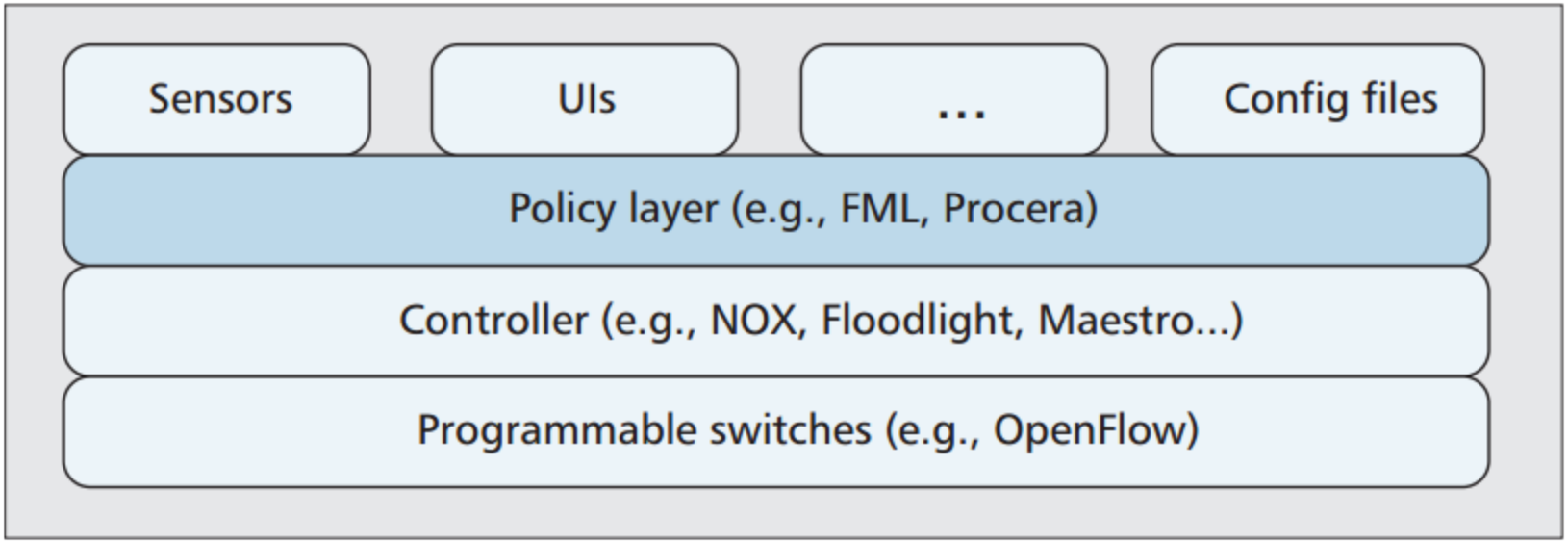

Figure 1. *Layers in software defined networking.*

Framework

OpenFlow

OpenFlow is one of the most common southbound SDN interfaces. Many vendors, including HP, NEC, NetGear, and IBM, produce OpenFlow-capable network switches.

Controller

There are a variety of OpenFlow controllers, such as NOX, Floodlight, and Maestro.

Procera

A northbound interface that provides the ability to specify and implement reactive policies.

Framework

Procera

To allow operators to implement high-level policies in an easier manner, the authors have designed and implemented Procera, an event-driven network control framework based on SDN paradigm.

It allows operators to express high-level policies with this language, and translates such polices into a set of forwarding rules, which are used to enforce the policy on the underlying network infrastructure, using OpenFlow.

Framework

Procera

To express event-driven network policies, Procera offers a set of ***control domains*** that operators can use to set certain conditions and assign appropriate packet forwarding actions corresponding to each condition.

Operators can also combine control domains to implement rich network policies, instead of relying on time or event-triggered scripts, which are error-prone.

Framework

Control domains	Examples
Time	Peak traffic hours, academic semester start date
Data usage	Amount of data usage, traffic rate
Status	Identity of device/user, distinct policy group, authentication status
Flow	Ingress port, ether src/dst, ether type, vlan id, vlan priority, IP src/dst, IP protocol, IP ToS bits, port number src/dst

Table 1. *Control domains in Procera, along with examples of how a higher-level policy can use them.*

Framework

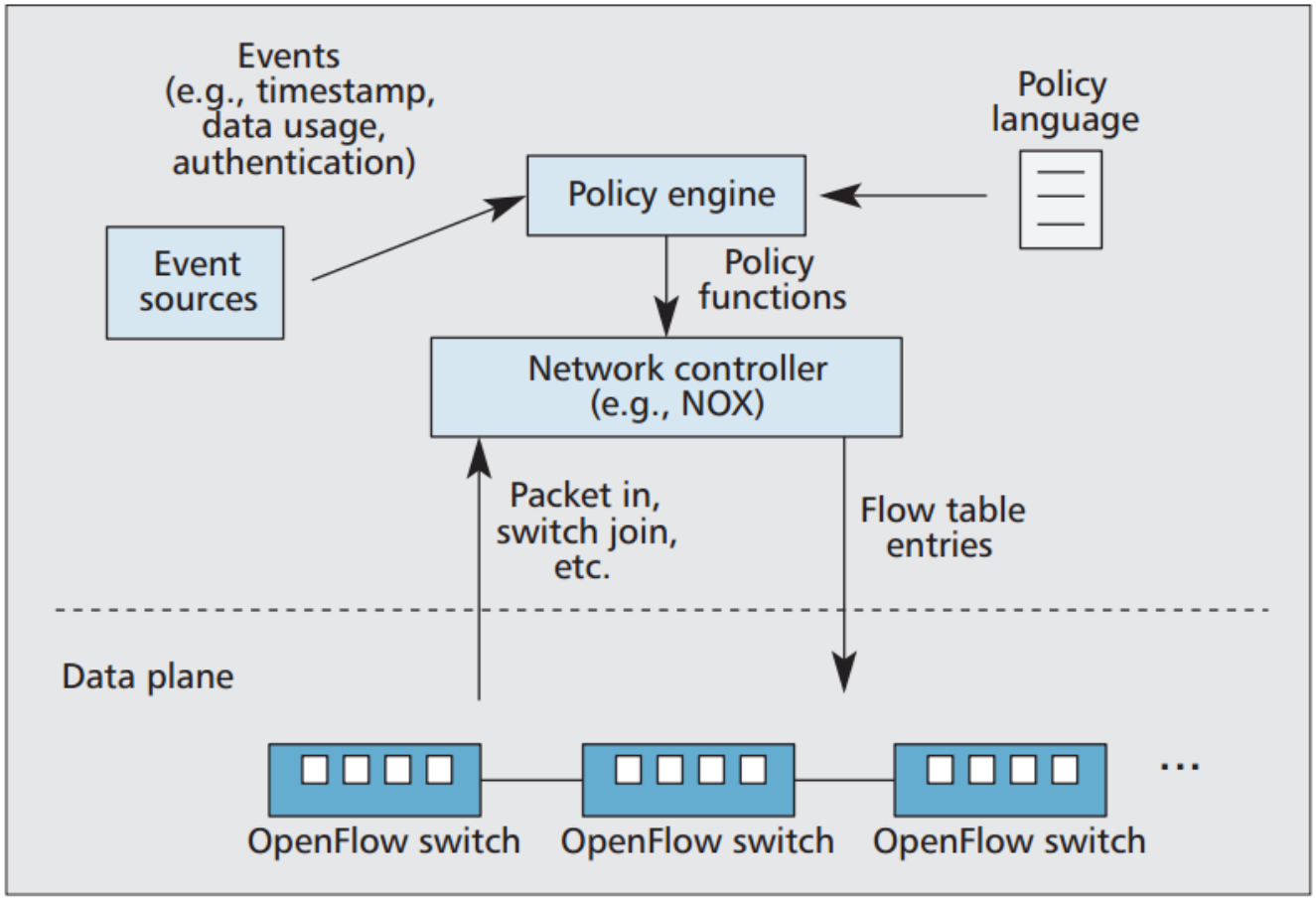

Figure 2. *Procera architecture.*

Framework

Event sources

Network components or middleboxes that can send dynamic events to the Procera controller.

Policy engine

responsible for parsing the network policy expressed with a policy language, and also processing various events that come from event sources.

Controller

Procera has a controller that makes all traffic forwarding decisions and updates low-level network switch flow-table entries according to this policy.

Case Study

The authors describe the deployment of Procera in a campus network (Georgia Institute of Technology), which is dynamic environment with many events occurring across the network.

The authors also describe the deploymeny of Procera in home networks, and show how Procera makes it easier to express various types of policies.

Case Study

Campus network

1. It requires every unregistered end-host device to undergo an authentication process.
2. After successful authentication, the device is scanned for possible vulnerabilities.
3. If none are found, the device is finally granted access to the internal network and the Internet.

Case Study

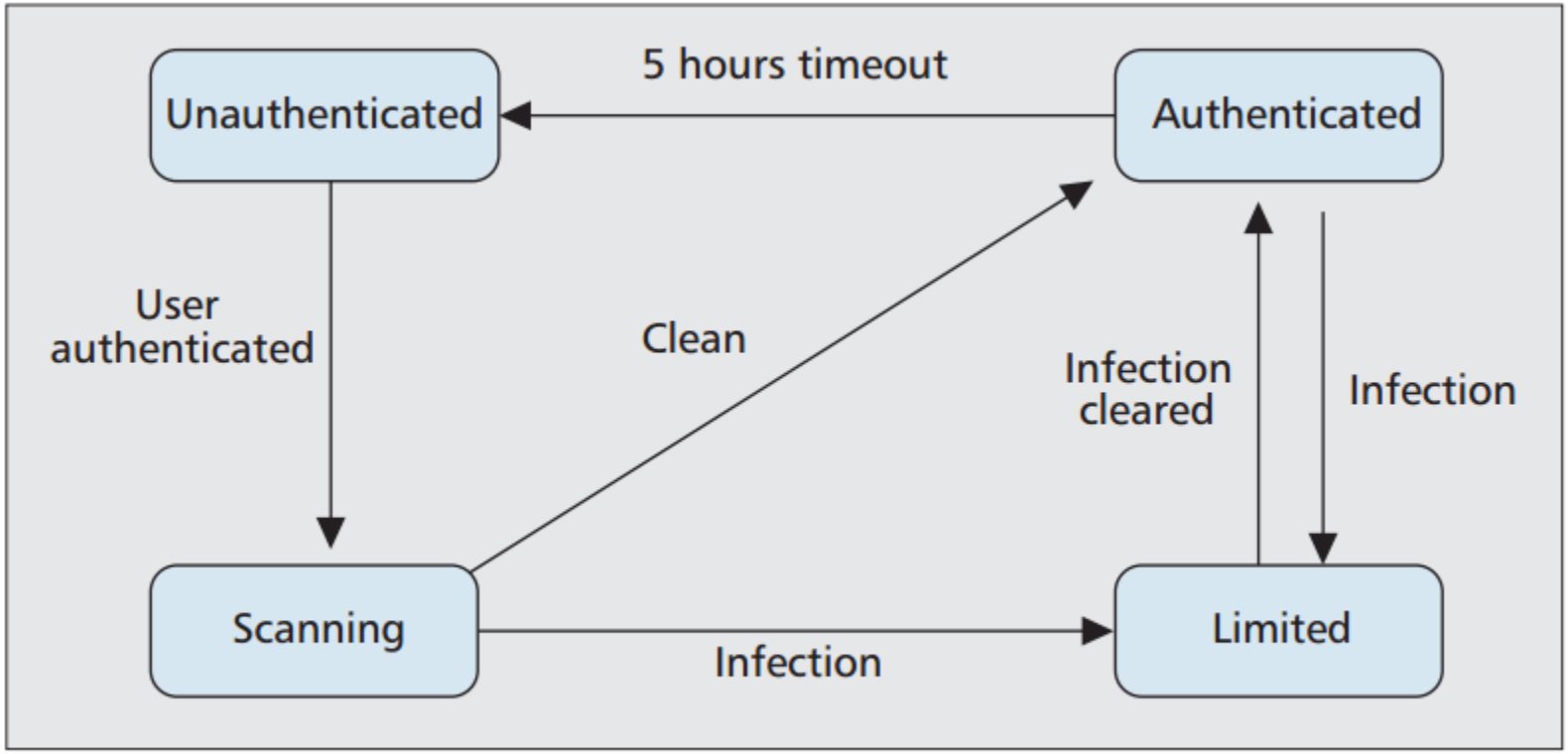

Figure 3. *Transitions and events in campus network.*

Case Study

Campus network

Implementing such complex policy with static tools like firewall rules and VLAN technology requires network operators to independently configure multiple different components, including middleboxes, management servers, and numerous ad hoc scripts. Procera significantly simplifies the expression of these types of policies.

Case Study

Campus network

The deployment spans three buildings in the Georgia Tech campus. For packet forwarding, they use five OpenFlow-capable network switches.

There are two wireless access points deployed in building 3, through which end-host devices can connect to through a broadcasted SSID.

The authentication web portal, intrusion detection system, and scanner, which are event sources, are located in the data closet in building 2.

Case Study

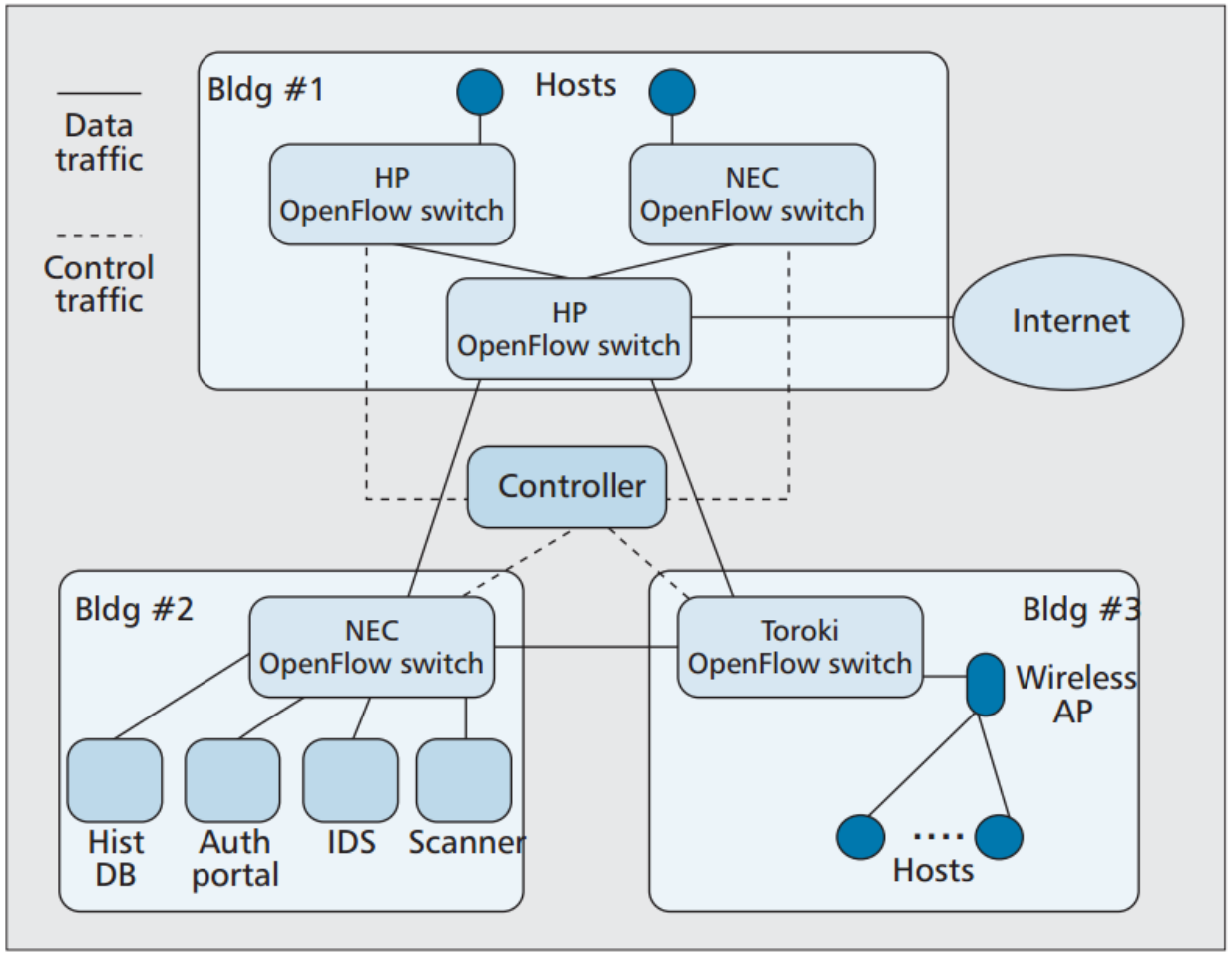

Figure 4. *Campus network deployment.*

Case Study

Home network

One of the problems about home networks is that they offer only limited visibility into home broadband performance and its overall status.

Access Internet service providers (ISPs) often want to continuously monitor the status of home networks, and ensure that customers receive their promised service. Content providers may desire to know how their traffic engineering decisions influence the home user experience.

BISmark is a collection of home gateways installed in households, a centralized management and data collection server, and multiple measurement servers deployed around the world.

Case Study

Scenario

ISPs are starting to enforce monthly bandwidth caps in home networks that limit data usage in the home. Unfortunately, home users currently lack resources to manage or even monitor their daily bandwidth usage.

The internet users need a system to monitor their data usage at a device level and to manage the bandwidth usage by allocating data usage capacity to each devices.

Through the uCap project, the authors have implemented and deployed such a system in 10 households with OpenFlow-capable wireless home gateways.

Case Study

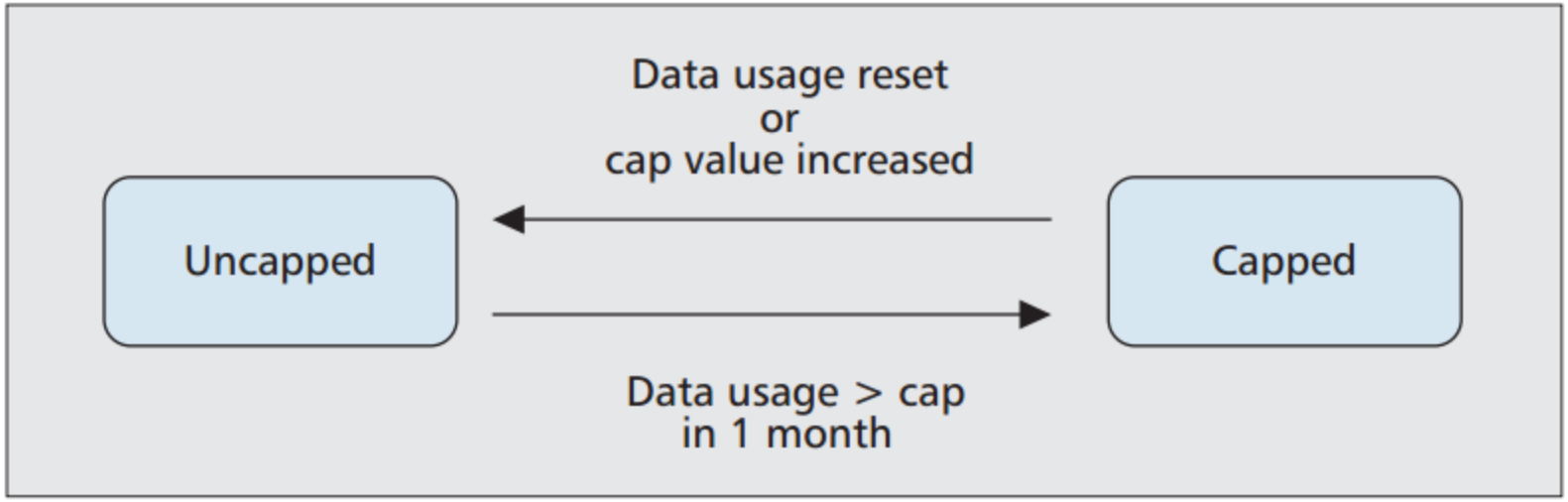

Figure 5. *States and transitions for a simple example of usage cap management in home networks.*

Case Study

Deployment status

The authors use NetGear WNDR 3700v2 and 3800 wireless routers as OpenFlow-capable forwarding devices in homes. Home users use the router as a wireless access point in the home and observe no particular difference from any normal wireless access point.

As of September 2012, 24 people and 137 devices in 10 households were using the SDN-based system. The total traffic usage of 10 households over three months has reached approximately 1140 Gbytes.

Conclusion

- Two of the main reasons that networks remain difficult to manage are continually changing network state and low-level per-device network configuration.
- SDN introduces a method to raise the level of abstraction for network configuration, providing mechanisms that automatically react to frequent and continual changes to network state.
- An event-driven network control framework, Procera, was introduced. Operators can use it to implement and enforce a reactive network policy with the high-level configuration language.
- The authors have deployed such the system in both a campus network and many home networks, show that Procera is feasible.

Conclusion

- As Procera is based on the software defined networking paradigm, it suffers from the inherent delay caused by the interaction of the control plane and the data plane. [1] [2]
- Procera has four control domains — time, data usage, authentication status, and traffic flow. However, this list of domains is incomplete.
- The trade-off between the cost of the installation and the benefits from it need to be evaluted.

References

[1] A. R. Curtis et al., “Devoflow: Scaling Flow Management for High-Performance Networks,” Proc. ACM SIGCOMM ’11, New York, NY, 2011, pp. 254–65.

[2] M. Yu et al., “Scalable Flow-Based Networking with DIFANE,” Proc. ACM Sigcomm, Aug. 2010.