SDN: Unified Networking Programming Framework

Advisor: Professor Richard Yang

Background

As Internet has grown dramatically, the scale and complexity of corporate networks have increased significantly as well. The network infrastructure of a large company like Facebook consists of large number of traditional physical network devices, which require manual configuration of each device to constitute and maintain a desired network structure. This not only is a huge burden on network engineers, but also limits the capability to customize the behavior of a network once it is set up.

Software defined network (SDN) tries to address this problem by virtualizing these physical network devices so that users can configure an entire network using a high-level programming language. In other words, SDN allows a network to customize its behaviors through centralized policies at a conceptually centralized network controller. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

Motivation

In order to tackle the aforementioned issue of traditional networks and SDN, we need a simplified and unified programming framework that can orchestrate a complex, distributed network infrastructure. Particularly, the following three domains are the most important components in building and configuring a software defined network: routing, resource allocation and security.

First, we should be able to write a centralized routing policy that can specify how each incoming or outgoing packet should be routed across all devices in a network. And the routing policy should be flexible enough to modify its routing decisions based on changes in a context or network environment. Second, the SDN model should have an efficient resource allocation capability. In other words, the model should have accurate knowledge of network resources and computational resources of an entire network so that it can serve practical needs efficiently. Lastly, we should be able to write a centralized security policy of an entire network. The security domain is closely related to routing as well since dropping potentially malicious packets or duplicating packets for network security monitoring involve customized routing policies.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.

Existing Approaches

There are a number of existing approaches that address a subset of the aforementioned three domain problems, but not all three. For example, Maple simplifies SDN programming by (1) allowing a programmer to use a standard programming language to design an arbitrary, centralized algorithm, which [is called] an algorithmic policy, to decide the behaviors of an entire network. However, challenges in managing low-level details, such as setting up and maintaining correct and efficient forwarding tables on distributed switches, often compromise this conceptual simplicity.

By having a simplified and unified programming framework that can work across all of the three domains and across all devices in the network, we will be able to realize the benefits of centralized network control and conceptual simplicity of network management.
network, and (2) providing an abstraction that the programmer-defined, centralized policy runs, conceptually, “afresh” on every packet entering a network, and hence is oblivious to the challenge of translating a high-level policy into sets of rules on distributed individual switches. However, while it provides a unified programming model to specify routing and resource allocation policies, Maple doesn’t directly address the security domain problem; if users want to install a security policy, they need to manually set up a firewall system in the network and configure the routing policy to the firewall and the security policies of the firewall separately.

On the other hand, popular network security monitoring products are mostly stand-alone systems and independent of existing SDN frameworks. For example, Bro is a stand-alone system for detecting network intruders in real-time by passively monitoring a network link over which the intruder’s traffic transits. It also offers a powerful scripting capability, called Bro script, an event-driven scripting language that provides the primary means for an organization to extend and customize Bro’s functionality, with which users can write customized security policies for the network. However, its domain is limited to security so integrating Bro into an existing SDN framework generally requires a manual configuration.

Challenges

As mentioned above, there are a number of existing systems that are specifically designed for a subset of the three domain, but none addresses all three domain problems in a unified and simplified manner. Moreover, each system has very different paradigms, so it is generally quite complex to integrate these systems together. So my programming model should provide features that allow users to incorporate these three functionalities in a simplified manner.

Another challenge in designing the unified SDN programming framework is that different network devices have different capabilities. For example, a L4 switch is able to look at attributes up to the IP layer, so the switch itself is not capable of executing a security policy that is based on L7 attributes. Moreover, keeping track of connection state and network environment is generally beyond the capability of most physical network devices. To tackle this problem, I will need to design the framework so that it can provide users with functionalities to write policies for any of the three domains regardless of what kind of devices they use in the network.

Deliverables

1. **Systematic Survey of Literature on Existing Language Models**
   
   I will investigate existing programming language models that address each of the three domains, i.e. routing, security and resource allocation. For example, Maple simplifies SDN programming by introducing an *algorithmic policy* to decide the behaviors of an entire network. And the Choate programming model allows to specify the behaviors of packets using the attributes of all network layers in a centralized way. For the security domain, there are multiple open source firewalls and IPS, such as Bro and Snort, that allow users to specify security-related policies of a network using various language paradigms. Studying the design and implementation of these languages will help understand the challenges and requirements.

---

3 Voellmy *et al.*
4 Paxson
5 Writing Bro Scripts. https://www.bro.org/sphinx/scripting/
6 Voellmy *et al.*
7 Chen *et al.*
of inventing a unified programming model, which would be the second deliverable of the project.

2. **Design of a Unified Programming Model Specification**
   I will come up with a unified and abstract programming framework that can work across those three domains and across network devices. This programming model should provide a simple and logically centralized network control, with which we can orchestrate routing, security and resource allocation policies of a complex, distributed network. Moreover, I will design this model so that some of common open source firewalls, such as Bro and Snort, could be incorporated easily into the network. For example, when a user writes a simple security policy using my programming model, it could be automatically converted to a Bro script or a Snort policy so that it could be distributed and deployed in the corresponding service. This should allow users to easily integrate those existing powerful firewalls into their network.

3. **Implementation of a Prototype of the Programming Framework**
   Finally, I will implement a prototype of the programming model. I will likely implement the model (likely as a DSL or an API) using Python and use Mininet to emulate virtual networks and test my framework.

**References**

Writing Bro Scripts. https://www.bro.org/sphinx/scripting/