Modelling and Analysing SDN Controller

Christian Attiogbé

Séminaire AeLoS
Mercredi 19 Avril 2018

LS2N LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Plan

LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Software-Defined Networking

Software-defined networking (SDN) makes it possible to **control an entire network in software**, by writing programs that tailor network behavior to suit specific applications and environments.

SDN limitations

Unfortunately, developing correct SDN programs is easier said than done. SDN programmers today must deal with several complications.

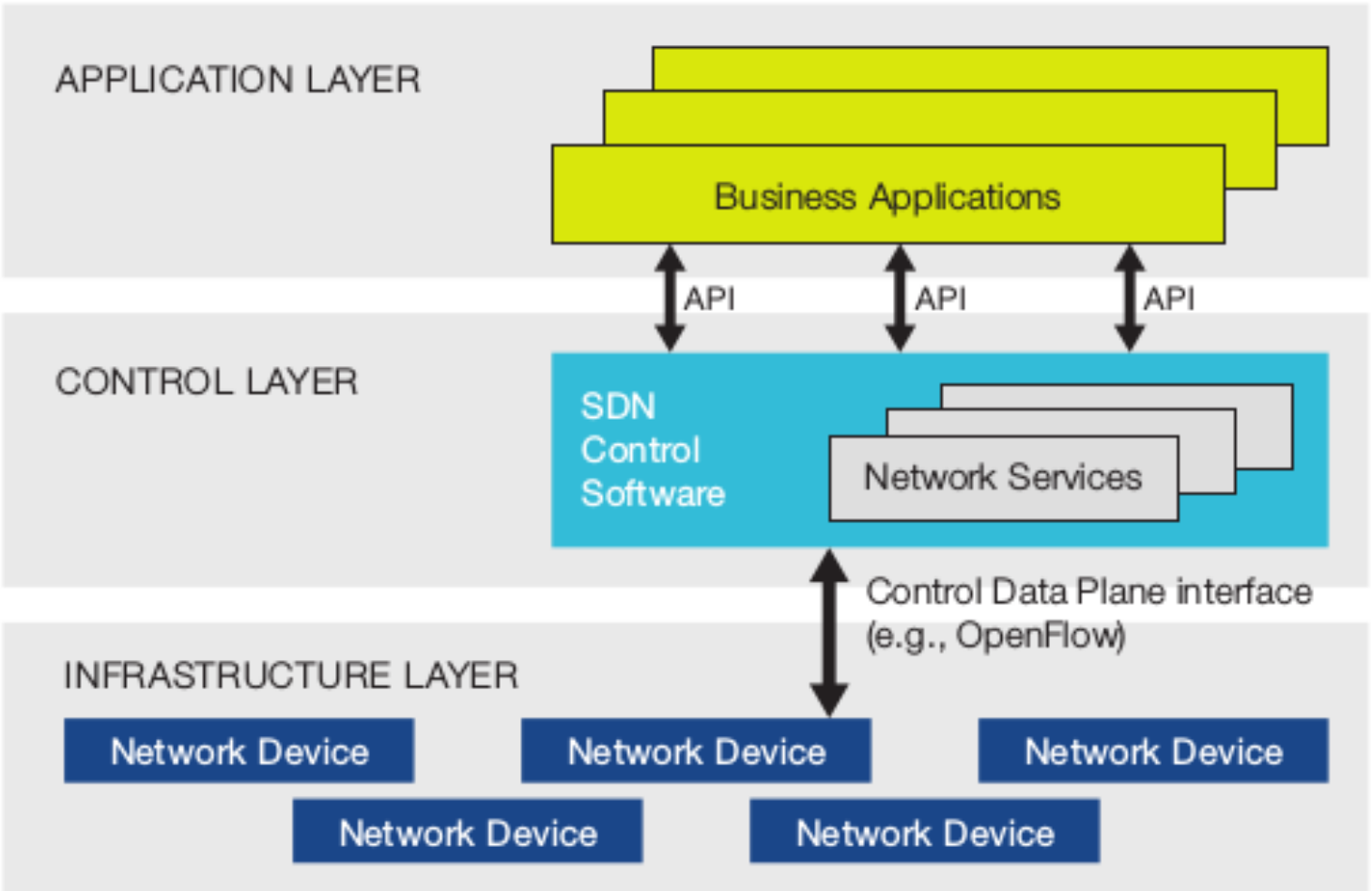
Guha & Reitblatt & Foster, Cornell, *Machine-verified network controllers*. In PLDI, 2013

Aim of SDN

To provide :

- open interfaces enabling development of software that can control the connectivity provided by a set of network resources and the flow of network traffic through them,
- possible **inspection and modification of traffic** that may be performed in the network.

Overview of SDN Architecture



OpenFlow (OF) Initiative

OpenFlow Specification :

- OpenFlow is the **most popular SDN platform**
- But OpenFlow has informal specification

Several works dedicated to the formalization of (parts of) the specification :

- ACSR (Algebra of Communicating Shard Resources), 2012
- Featherweight OpenFlow (2013, ... Coq)
- . . .

Plan

OpenFlow Network

- An OpenFlow network provides a powerful, scalable infrastructure that can be used to tailor networks to specific tasks.
- All network design and planning beyond physical connectivity can be done in software, specifically in the design of the OpenFlow controller program.
- This allows for significantly richer network processing and routing functionality, but also introduces the possibility thatlogical or design errors could lead to unsafe or incorrect behavior.

Example of SDN Invariants

Basic network properties

- No loop
- No blackhole (e.g., packet loss)

SDN-specific properties

- OF rule consistency between multiple applications
- Dynamic info/statistics consistency (e.g., flow, port, QoS, etc.)
- Consistency with legacy protocols (e.g., STP)

Issues

- SDN platforms such as OpenFlow force programmers to use a low-level API to express high-level intentions, which makes reasoning about SDN unnecessarily hard.
- While network programming bugs may be tolerated in best-effort systems or those serving non-critical needs, many cyber-physical systems need iron-clad guarantees that a network routing mission-critical information will always (or with very high probability) meet its safety and correctness requirements.
  (verificare - R Skowyra, A. Lapets, A. Bestavros, A. Kfoury, 2012)
- OpenFlow does not provide these guarantees ; in its present form, not suitable for scalable network design, in cyber-physical realm, etc.

Plan

Motivations

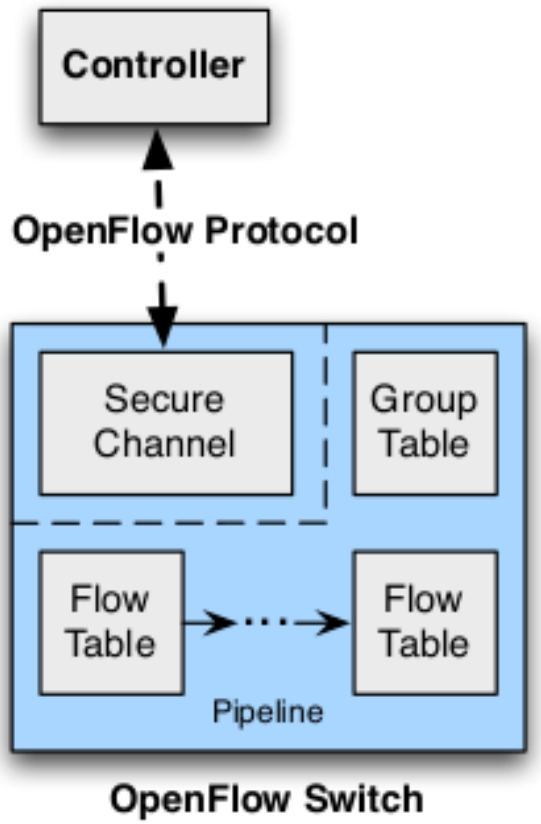
- Reliable Software-Hardware systems
- Correct-by-construction instead of bug detection.

Vision

An extensible event-based model to support property verification and tools for high-level abstractions with SDN + Refinement to specific codes.

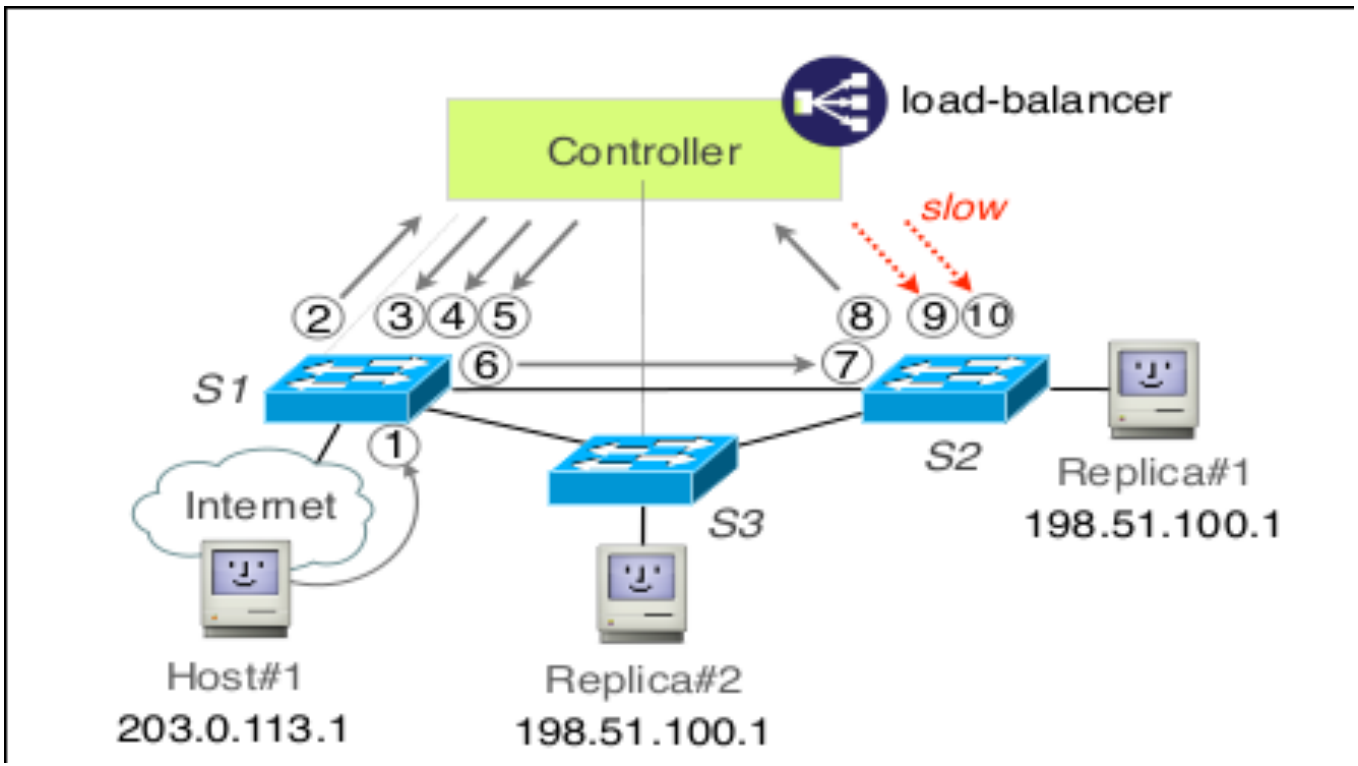
Plan

Overview of the controller



(source : The Open Networling Foundation, 2012)

General View of Interactions



(source : SDNRacer, PLDI2016)

Modelling Abstractions

1. Interaction between **Switches** and **Controller** : each one has a behaviour
   Controller : $\{sendMsg, receivePacket, cdots\}$
   Switch : $\{receivePacket, forwardPacket, addEntry, \cdots\}$
2. Exchange of **messages** and **packets**
3. Event-based view

The switch and the controller interact via *OpenFlow protocol*

Controller

- The controller goal : to compute, maintain and populate the **forwarding table of each SDN switch** in the network.
- The controller can **add, update, and delete flow entries** in Flow Tables, both reactively (in response to packets) and proactively.

Observed events : add, update, delete, ...

Switch

- Each OpenFlow Switch consists of **one or more Flow Tables** and a group table, which perform **packet lookups and forwarding**, and an OpenFlow channel to an external controller.
- Each Table contains **several entries** to be matched with incoming packets
- Each Table Entry contains **headers and body**.

  set of packet processing rules

Forwarding actions include **sending** the packet to the controller or to a given output port.

Flow Table

- Each flow entry consists of match fields (boolean predicate), counters, and a set of instructions to apply to matching packets.

Instructions associated with each flow entry either contain actions or modify pipeline processing.
Actions included in instructions describe packet forwarding, packet modification and group table processing.

Flow Table

Main components of a flow entry in a flow table :

Match fields	Priority	Counters	Instructions	Timeouts	Cookie

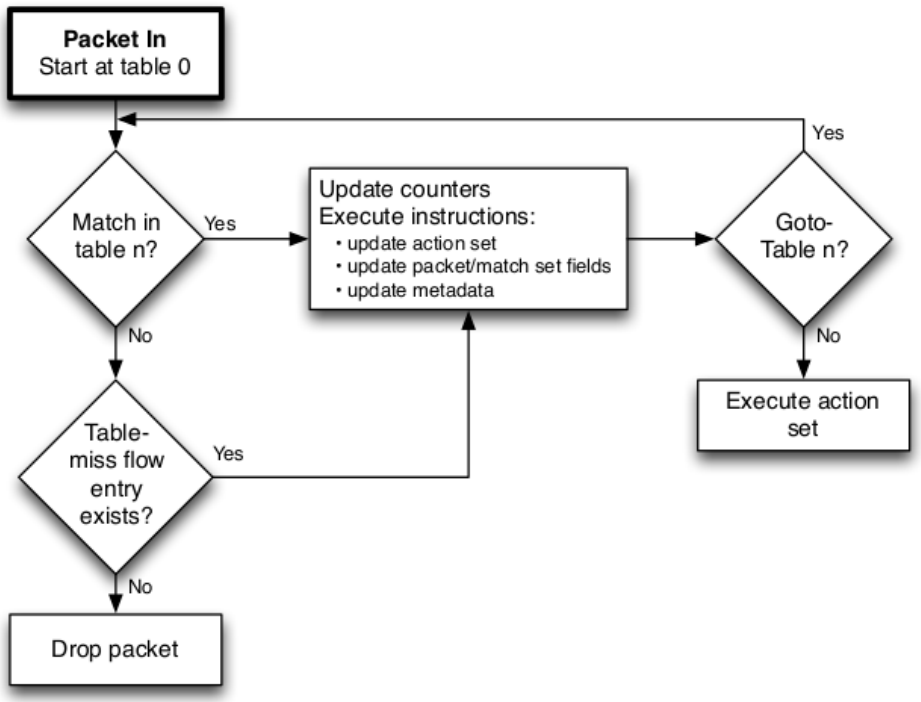
match fields : to match against packets ; in port, packet headers, optional metadata
priority : matching precedence of the flow entry
. . .

Matching packets using Table

Upon a packet reception in a switch

- Find highest-priority matching flow entry
- Apply related Actions
  - Modify packet, update match fields
  - Update action set (clear, write, ...)
  - Update metadata
- Send match data and action set to next Table

Matching packets using Table



If a matching entry is found, the **instructions associated with the specific flow entry are executed**. If no match is found, the packet may be **forwarded** to the controller, **dropped**, or may continue to the **next flow table**.

(source : The Open Networling Foundation, 2012)

LS2N LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Concurrency Issues

- asynchronous environment : forwarding packets
- read events, write events
- some non-commuting events (read, write in tables)
- Update consistency violation : packets forwarded an old version and a new version of the forwarding table ! (it should be one or the other).
- ...

Modelling with Event-B

- Abstract Model (exchange of messages+packets)
  Model = Behaviour of **Controller + Switches[Tables]**
- Refinements
  - with reordering of messages : to avoid consistency violation
  - delay forwarding or reading, appropriate message reordering
- Define a semantics for ordering the events
  (through strengthening of the event guards)
- two categories of messages : commuting ones, non-commuting ones

Event-B + Rodin

Context :

- voir dans rodin

Abstract Model :

- voir dans rodin