ATLANTIS PRESS
Advances in Computer Science Research, volume 71
4th International Conference on Machinery, Materials and Information Technology Applications (ICMMITA 2016)

Multi-Layer Modeling of OpenFlow based on EFSM

ZhiHao Zhang[1, a], DongMing Yuan[2, b] and HeFei Hu[2, c]

[1, 2]School of Electronic Engineering, Beijing University of post and telecommunication, Beijing 100876, China

[a]bupt_zhangzhihao@163.com, [b]yuandm@bupt.edu.cn

Keywords: OpenFlow protocol, Software-Defined network, Extended Finite State Machine(EFSM), test generation.

Abstract. The OpenFlow protocol, as the main south protocol for Software-Defined Network, is researched by many people. This paper proposed a new notion of Muitil-Layer Extended Finite State Machine(MEFSM) to model OpenFlow protocol and derive the executable testing sequence base on it. Using Hierarchical method, we assemble test sequences of each layer. And then we using a global variable to avoid cloned test sequences. Experimental result show that our model can avoid state explosion in some way.

1. Introduction

Software defined network (SDN) has been researched by more and more people, both in industry and academic. It aims to make more innovations in network by decoupling data plane and control plane. However, owing to different understanding of protocol, devices/implementations produced by different suppliers may meet kinds of faults (or exceptions) while try to communicate with each other. In order to ensure consistency, devices/implementations must be full tested before coming to the market. Protocol consistency testing is the primary challenging, both in data plane and control plane, many methods or models are proposed to try to solve it.

OpenFlow is a communication protocol that permit control plane interactive with data plane, it allows direct access and manipulation of the forwarding plane of network devices after the network controllers get information from forwarding plane. At present OpenFlow is maintained by the Open Network Foundation (ONF) and is the main communication protocol of South Bound Interface of software defined network. However, as a primary protocol, vendors and research would have different implementation/understanding of it, that would cause unpredictable problems while communicate with each other. For this point, protocol consistency testing before connecting devices from other vendors is significant. Few papers give attention to this problem.

Black-box testing is an important method of software testing which reveal the correctness of an implementation without expose its internal structures or codes. This method of test is widely used to test whether implementation coincide with protocol. The researcher designs valid or invalid inputs and assemble them different types of test sequences to determine the correct output, always accompany with test oracle (expect output). In the past, researchers need to generate appropriate test sequences manually, it's inefficiently and error prone while the system grows huge. So Extend finite-state machine (EFSM) was adopt to solve this situation. It provides puritanical pattern to describe a system and new approaches for functional test of protocol and system. In this paper, we aim to propose an appropriate black-box testing model to represent OpenFlow. In the view of fact that most internal implementation or codes of communication switch is not available, we hope that our method would help to solve that problem in some way. We define a new notion of Mutil-Layer Extended Finite State Machine (MEFSM), which divided conventional Extended Finite State Machine into multiple layers according to pipelining processing mode. It is more suitable to describe OpenFlow potocol, because we ignore detail within each step of pipeline processing, that could avoid state explosion.

One of prominent features of Extended finite state machine (EFSM) is that it can't only represents data flow but also control flow, which make EFSM popular in modeling giant system.

however, because of the existence of context variable between different states, a specific transition path may be infeasible. In some System, infeasible transition path could account for a large portion of all transition path [1], some papers give useful algorithm to solve this problem. In our model, we prevent infeasible transition path come into being by attach a specific signal label in different pipeline processing.

The latest edition of OpenFlow pipeline of every OpenFlow switch contains at most hundreds of flow tables, each flow table containing multiple flow entries. An Inorder Traversal Algorithm can easily cause states explosion, of which many transition path may infeasible. Alternatively, in our model, we combine pipeline processing with multiple layer method relieve this situation efficiently. Based on OpenFlow process mode, we translate it into MEFSM. Then, we generate test sequences on account of this model, the result shows that it can reduce states explosion to some extent. By using a processing label, we can efficiently avoid generate infeasible transition path.

The reminder of this paper is organized as follows. The related work stated in section II. Section III introduces the EFSM model, based on it, we describe the multi-layer Extended Finite State Machine. In section IV, we generate our test sequences and shows the experiments result. Lastly, we conclude our work in section V.

2.Related works

In this section, we show some works about EFSM model in test sequences generation and SDN formal model and verification separately.

2.1 EFSM in test sequences generation

There are many related studies on EFSM model on the test sequence generation. Most of them focus on generate test case automatic. Jie Zhang and Rui Yang [2] introduce a new approach to generate test cases automatically for given transition paths of an EFSM model. Then, they use scatter search algorithm to search for test data that can trigger given transition path. In their executable EFSM model, a new notion Fe = (TTP/GTP) *100 is used to determine whether the current initial input variable values can trigger a better transition path. Mathias Landh¨außer, Walter F. Tichy [3] present an approach that generates useful test clones automatically. In their research, they found that clone test cases always take an undeniable proportion of all test cases. And then use an automatic detection of analogs to reduce the number of “boiler-plate” test. However, the method may fail for some reasons.

Xiaofei Zhou, Ruilian Zhao and Feng You [4] propose a method to generate test data with multi-population genetic algorithm(MPGA) based EFSM model. Due to the existence of the context variables, automated test generation on EFSM is difficult. In their method, the key problem is the parameter setting of MPGA, which would directly influence the efficiency of test data generation. They use a simple ‘rules of thumb’ approach to find an optimal parameter setting of MPGA. In their paper, they simple study a single EFSM model – ATM, and they did not mention the effect of diversity with different combination parameters on the test generation of EFSM model by MPGA.

2.2 OpenFlow protocol modeling

Actually, OpenFlow has become the most important south interface protocol in SDN. Michael Jarschel and Simon Oechsner[5] based on measuremnets of switching times of OpenFLow hardware ,they derive a basic model to model OpenFlow architecture. In their work, the model is limited to a single switch per controller, whereas OpenFlow allows multiple controllers connect to multiple switches. Natali Ruchansky, Davide Proserpio proposed a formal modeling of the OpenFlow switch using Alloy. They use model enumeration (Alloy and Alloy Analyzer) to model OpenFlow-capable in order to provide a proof of correctness of the OpenFlow Switch Specification. However, no results are provided in this paper.

Jiangyuan Yao and Zhiliang Wang [6] proposed a systematic Black-Box testing modeling of SDN data plane. In their work, a new model, Pipelined Extended Finite State Machine(Pi-EFSM), was adopted to specify the multiple-level pipeline of SDN data plane. Based on this model, they present a 3-phase test generation approach, at last, test sequences of different scale and coverage are generated. In this paper, only three tables were used to generated testing sequences, in fact, in

OpenFlow protocol, hundreds of tables may exist, in this situation more states should considerate, and it would probably lead to state explosion.

Currently, few papers force on the modeling of OpenFlow protocol. Li Hua and He Nan [7] proposed an OpenFLow Modeling based on hierarchical CPN (Coloured Petri Nets). In this paper, they model the OpenFlow protocol from three aspects: OpenFlow switch, channel and controller. However, they ignored the links them.

3. Formal Modeling

In this section, wo will firstly introduce EFSM model. Based on it, we proposed our MEFSM model, which would help generate testing sequences more efficiently. In our method, the multiple layer separates the inner processing procedure and the linking between different model. Because of the same procedure in each table. So in testing sequences generation we ignore internal implementation in them.

3.1 EFSM model

An extended finite state machine(EFSM) is evolved from FSM. However, what is different from FSM is that EFSM can not only presents data flow but also controlling flow in a system. The EFSM model is formally described as a 6-tuple $M=\{S, S_0, V,I,O,T\}$ where

- S is a finite set of symbolic states
- $S_0 \in S$ is the initial state of the system, as the entrance of the model
- V is a set of finite variable
- I is a set of finite inputs
- O is a set of finite outputs
- T is s set of finite transition paths

In a system, a state is a situation or condition of a system during which some invariant holds or waits for some external event. Guards are Boolean conditions of extended state variables which enable or disable certain operations. They are usually immediate consequence of using extended state variables. An event may be parametric which conveys quantitative information. Actions are performed when an event is dispatched.

In the transition path set T, each path $t \in T$ is also a 6-tuple, $(t_s, t_t, t_g, t_o, t_a, t_i)$ where $t_s \in S$ is the state of the transition, $t_t \in S$ is the target state of the transition, $t_i \in I$ is the input event, t_g is an expression or null which is called guard, t_o is the output symbols.

A simple EFSM example are given below:

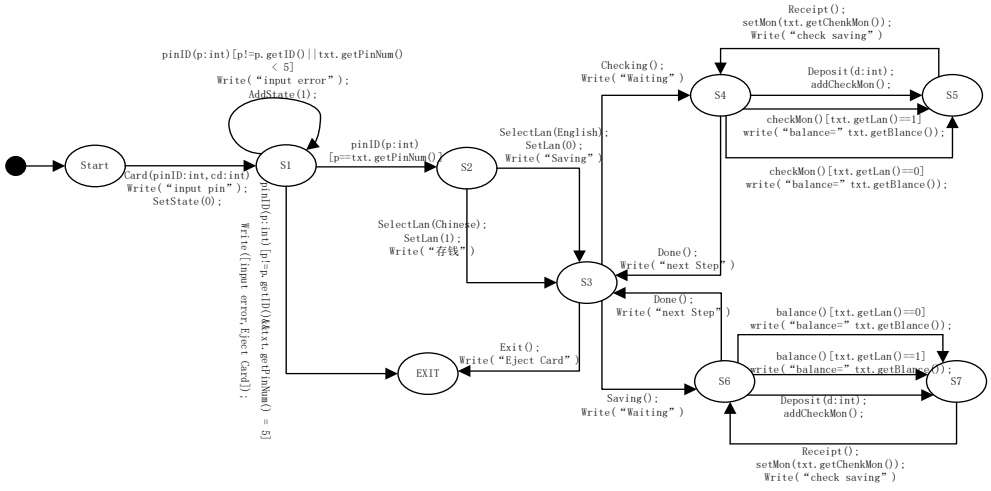

Fig 3.1 EFSM of ATM

3.2 Outer modeling of EFSM

The newest OpenFlow defined at most hundreds of tables, each table holds several flows which decided how one input data flow should be processed. It is easily causes states explosion because of the huge number of flow table. However, if appropriate model is token, the difference between

different table can be ignored. What more, when take in account of pipeline process of data packet. We avoid state explosion in some way.

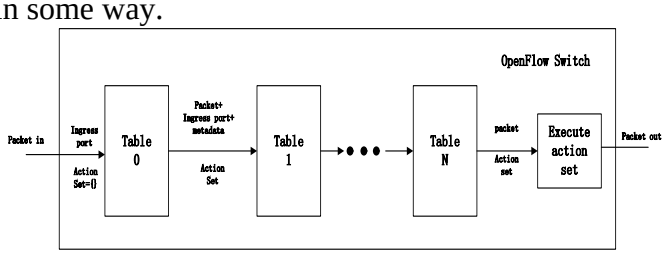

Fig 3.2 process of data packet in OpenFlow

When a packet is arrived, it first come into table 0, it is the entrance of all packets, in each table, firstly, the packet finds highest-priority matching flow entry, then it applies instruction into action bucket, which includes three kinds of instruction. (1) modify packet or update match fields; (2) update action set; (3) update metadata. finally, send match data and action set to next table. In each table, the packet follows the step above. so we can hide the process internal, when generate testing sequences, firstly, we find a solid transition path, then we apply our internal path of each table into one single transition path. By this way, we could ensure the transition path is not a cloned sequence [3], and it is easy to generate different length of testing sequences. In our simulation, we can infer that by this model, state explosion is avoid in some way. The picture 3 below show the outer layer of MEFSM.

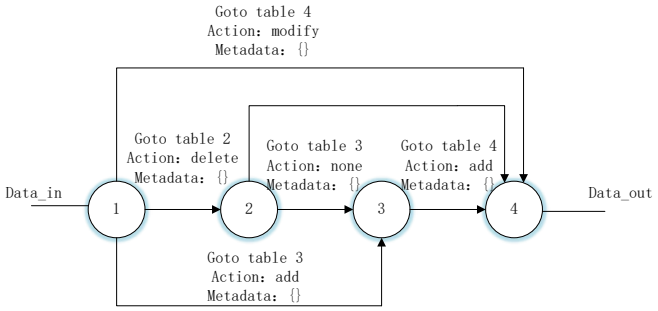

Fig 3.3 The brief example of the MEFSM outer model

In this outer layer, we force on the packet walk through between different tables. When a packet get inside, it firstly go into table 0, then after compare table flows inside of the table 0, it can choose to go to table which may be any table whose table number is bigger, or it may just drop or send to the controller. If the packet goes to another table, it would do the same thing until the packet get out of the pipeline. At last, the actions in the action bucket are implemented.

3.3 Internal modeling of MEFSM

The internal model of the MEFSM reveal the operation to an input packet. When a packet goes into a table, the packet will follow the instructions that match them in the table. Each flow table contains of flow entries, they are match fields, priority, counters, instructions, timeouts and cookies. A flow table entry is identified by its match field and the priority: the match field and priority taken together identify a unique flow entry in the flow table. The flow entry that wildcards all fields and has priority equal to 0 is called the table-miss flow entry. The figure shows the match processing.

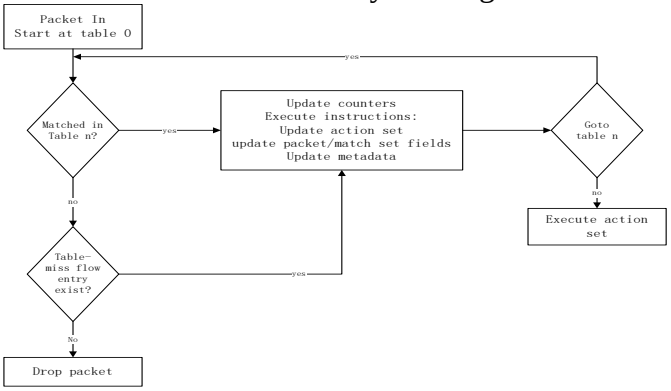

Fig 3.4 The match processing in OpenFlow Switch

Once receive a packet, the OpenFlow Switch performs the function above. It starts through compare the table lookup in the first flow table, and then based on the instructions, it may perform table lookups in other flow tables or drop. We need notice that if a packet does not match any flow table, then how to process this packet. In fact, each flow table must support a table-miss flow entry to process table miss. If a packet unmatched in the flow table, depends on design, it would be sent to the controller, drop or directed to a subsequent table. When a flow entry is expired, it would be removed from flow tables in three ways, either at the request of the controller, via the switch flow expiry mechanism, or via the optional switch eviction mechanism.

The counters are maintained for each flow table, flow entry, port, queue, group bucket, meter and meter band. OpenFlow-compliant counters may be implemented in software and maintained by polling hardware counter with more limited ranges.

Each flow entry contains a set of instructions that are executed when a packet matches the entry. Those instructions result in changes to the packet, action set and/or pipeline processing.

An action set is associated with each packet. This set is empty by default. A flow entry can modify the action set using a Write-Action instruction or a Clear-Action instruction associated with a particular math. When the instruction set of a flow entry does not contain a Goto-Table instruction, pipeline processing stops and the actions in the action set of the packet are executed.

There is a link between switch and controller, A switch would connects with one controller or more, that depends on how to deploy the network. Of-config protocol is designed to deal with the linking between them. So in our paper, our model ignores the connection of them. We just regard them as a port in the OpenFlow switch.

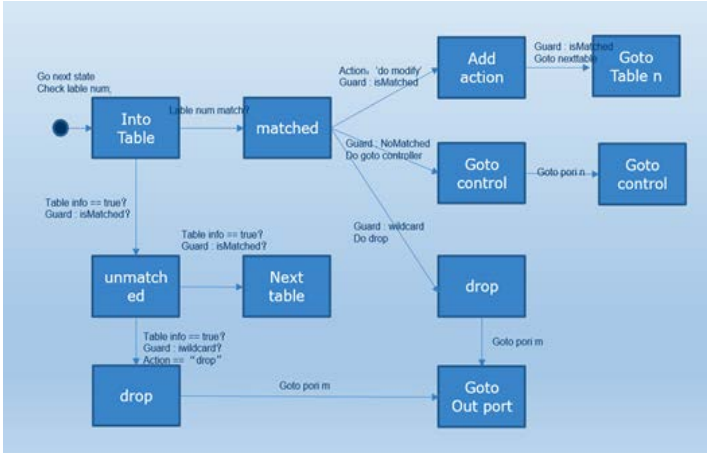

Fig 3.5 Inner modeling of OpenFlow data processing

In our model, we define a global variable to ensure the number of the table which the packet is getting through. When a packet arrives the entrance of each table, the first thing we do is tag the table number to it. By this way, can we efficiently avoid generating negative test sequence.

4. Experimental Study

In this section, a detailed empirical study is presented to validate the presented approach. The effectiveness is evaluated in terms of the correctness and the time the model generates the test sequences in compare with other traditional model.

In our algorithm, the input is our MEFSM model, we generate our test consequence by two steps, firstly, we choose a feasible way to get through all flow tables. To traverse all possible data path, we using Dijkstra algorithm to achieve that, but before each explore of new node in Dijkstra algorithm, we compare current table number to the previous table number. If the current table number is smaller, the node is blocked. By this way, the cloned test sequence can be avoid in some way. Secondly, in each table, they have the same action in fact, basing the outer test sequence that generated in the previous step, we add each internal path into the outer path. one thing to note is that when add each internal path, the conflict should be avoid. Finally, after each test sequence is assembled, a complete, effective and streamlined test sequence list is generated.

Table1 generate of test sequence

	Data Gragh			Component Machines	
coverage rule	Path	edges	vertexes	states	transition
coverage rate	100%	100%	100%	100%	100%

5. Conclusion

In this paper, we propose a new formal model to generate test sequences for OpenFlow protocol. Our approach has many advantages. Firstly, the MEFSM model is suitable for the pipeline processing with global variable. Secondly, we know that the table number in a OpenFlow switch may as many as hundreds, we abstract each table process flow, by this way, the states explosion can be eliminated in some way. Finally,By using a global variable, we can stop search data path once the wrong path appeared. It can help generates test sequence more efficiently. However, in our model, we ignore the link between the switch and controller, the Of-config is designed to solve it. In the future, we will do more research in that aspect.

6. References

[1] Wong S, Ooi C Y, Yuan W H, et al. Feasible transition path generation for EFSM-based system testing[C]// IEEE International Symposium on Circuits and Systems. IEEE, 2013:1724-1727.
[2] Zhang J, Yang R, Chen Z, et al. Automated EFSM-based test case generation with scatter search[C]// International Workshop on Automation of Software Test. IEEE Press, 2012:76-82.
[3] Landh&#, Er M, Tichy W F. Automated test-case generation by cloning[C]// International Workshop on Automation of Software Test. 2012:83-88.
[4] Zhou X, Zhao R, You F. EFSM-based test data generation with Multi-Population Genetic Algorithm[C]// IEEE International Conference on Software Engineering and Service Science. 2014:925-928.
[5] Jarschel M, Oechsner S, Schlosser D, et al. Modeling and performance evaluation of an OpenFlow architecture[C]// International Teletraffic Congress. International Teletraffic Congress, 2011:1 - 7.
[6] Yao J, Wang Z, Yin X, et al. Formal Modeling and Systematic Black-Box Testing of SDN Data Plane[C]// IEEE, International Conference on Network Protocols. IEEE, 2014:179-190.
[7] Dong L, Li H, He N, et al. Testing OpenFlow interaction property based on hierarchy CPN[C]// IEEE International Conference on Network Protocols. 2013:1-2.
[8] OpenFlow Switch Specification Version 1.3.1, 2012.9.6.
[9] http://yuba.stanford.edu/foswiki/bin/view/Open Flow/Mininet.
[10] “White Paper (Software-Defined Networking: The New Norm for Networks),” 2012. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-library/whitepapers