

PAPER • OPEN ACCESS

Model Checking the OpenFlow Protocol Using SPIN

To cite this article: Huiling Shi *et al* 2019 *IOP Conf. Ser.: Earth Environ. Sci.* **234** 012071

View the [article online](#) for updates and enhancements.



IOP | ebooks™

Bringing you innovative digital publishing with leading voices to create your essential collection of books in STEM research.

Start exploring the **collection** - download the first chapter of every title for free.

Model Checking the OpenFlow Protocol Using SPIN

Huiling Shi*, Wei Zhang and Xinchang Zhang

Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan)
Qilu University of Technology (Shandong Academy of Sciences), Jinan, China
shihl@sdas.org

Abstract. Software-Defined Networking (SDN) is a new architecture of separation of logical control and data forwarding. Although this separation has brought many benefits to the network, it also exposes the network to more risks. OpenFlow protocol is a communication protocol between them. The correct behavior of OpenFlow protocol is critical to SDN. Model checking technology has been widely applied to the protocol verification. This paper presents the process of model checking the OpenFlow protocol using SPIN. First, we describe an abstract formal model of OpenFlow protocol. And then we translate the model with PROMELA which is a model description language. Finally we apply the model checker (SPIN) to verify properties. In the process, it is realized that to pay more attention to the delay of SDN is more efficient when updating OpenFlow protocol version.

1. Introduction

Software Defined Networking (SDN) has attracted increasing attention due to the flexibility, programmability, and simplicity of management. The basic idea of SDN is separation of logical control and data forwarding. Although this separation has brought many benefits to the network, it also exposes the network to more error risks. OpenFlow protocol is communication specification between control plane and data plane. It is critical to the correctness and safety of SDN.

There is some research work for OpenFlow testing[1][2]. Previous research is mainly in the actual or simulated network environment, through different detection tools to simulate the transmission of data packets. However, there is no analysis of the logical process of the OpenFlow protocol itself.

With the complexity of OpenFlow protocol, it is necessary to formalize the validation of the OpenFlow protocol to find defects at the early design stage. [3] analyzes the process of model checking. [4] applies CSP and PAT to model. [5] constructs the CPN model of OpenFlow protocol. In [6] model checker tools are VERSA and UPPAAL, and in [7] model checker is PAT. [8] introduces a model checking system FLOVER.

However there is still not much modeling work for OpenFlow especially using SPIN. In this paper, we present a formal method for analyzing and validating the execution logic of OpenFlow protocol based on model checking. First analyse the packet processing mechanism of OpenFlow protocol, and build formal PROMELA models. Then we definite properties and set Assertion. Finally, the OpenFlow protocol is verified through the corresponding model checker tool SPIN.



2. Formal Modeling of OpenFlow Protocol

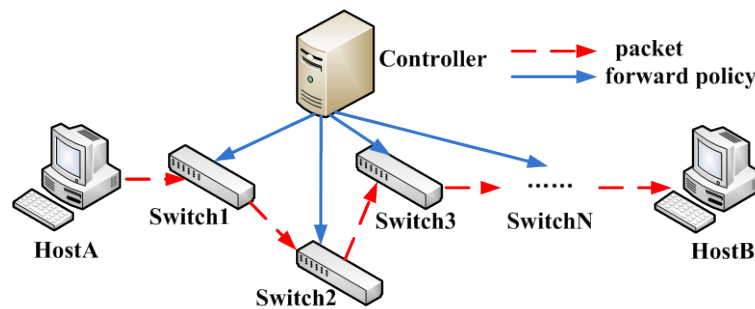


Figure 1. Overview of SDN

2.1. Packet Processing Mechanism

OpenFlow protocol can be divided into the data structure definition and the logical interaction between controllers and switches. The logical interaction is the focus of analysis and verification in this paper. For simplicity, SDN consists of single controller and multi switches as shown in Fig1.

Packet processing mechanism is as follows. When the terminal HostA sends the packet to the terminal HostB, the packet arrives the switch1, the switch1 parses the packet header first, and then matches the packet header with the flow table item according to the priority. If the match is successful, the specified field of the flow table is modified and the packet is operated according to the forwarding strategy of the flow table, including forwarding packets to the next port. If the matching fails and the Table-miss table entry is configured in the flow table, the packets are sent to the controller in the "Packet-in" format, and the controller makes a new forwarding strategy and encapsulates it as the "Flow-Mod" format to each switch on the transmission path, and the new forwarding strategy is sent down to each switch on the forwarding path. The packet is returned to the source switch in the "Packet-out" format. If the match fails and the table-miss table item is not configured in the flow table, the packet is discarded.

2.2. PROMELA Model

Model checking is widely used for the verification of protocols. The most famous model checker is SPIN. SPIN accepts design specifications written in the verification language PROMELA [9].

Combined with the practical application scenarios, we have simplified the size and details of SDN on the basis of guaranteeing the logic integrity of OpenFlow protocol. After simplification, the SDN is composed of two hosts, one controller and two switches. The message flow between them is shown in Fig 2. Arrows indicate the direction of message flow.

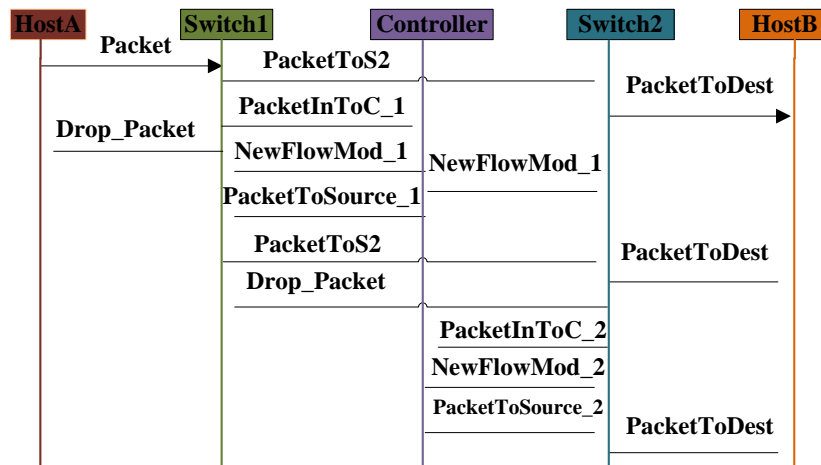


Figure 2. Message flow

Messages are defined as “mtype = {Packet, PacketToS2, PacketInToC_1, ..., PacketToDest}”. Channels are declared like “chan SwitchToHost=[0]{mtype}”.

According to the analysis above, we have abstracted five processes, named HostA , Switch1, Controller, Switch2, HostB. The fragment of PROMELA code is shown as follow.

```

proctype Switch1(.....)
{
  do
    ::HostAToSwitch1?Packet-> /*Receiving a Packet*/
    if
      ::Switch1ToSwitch2!PacketToS2->
        /*Match */
        {
          MatchFlow_1=true;
          goto end;
        }
      ::Switch1ToController!PacketInToC_1->
        /*Mismatch*/
        {
          MatchFlow_1=false;

          do
            /*Waiting for NewFlowMod*/
            ::ControllerToSwitch1?NewFlowMod_1->
              {
                UpdateFow_1=true;
                Switch1ToSwitch2!PacketToS2;
                goto end;
              }
          od
        }
      ::Switch1ToHostA!Drop_Packet->
        .....
    fi;
    ::Switch2ToSwitch1?Drop_Packet->
      .....
  od;
}
  
```

```

end:skip;
}

```

2.3. Properties Definition and Assertion

Then We analyze whether OpenFlow protocol satisfies the following properties. (a) Accessibility : When a packet is sent to the network and matches the forwarding policy, the packet must be sent to the destination. (b) There is no loop in SDN.

SPIN verifies properties that we are interested in against PROMELA models by setting assertion. In this case, we insert the assertion "assert((MatchFlow_1==true) || ((MatchFlow_1==false) && (UpdateFlow_2==true)))" after Switch2 receives packet from Switch1 in order to verify whether delay is or not considered. MatchFlow-1 and UpdateFlow_2 are global variables. They are valued in the processes according to the change of state.

3. Verification With Spin

Let us now observe whether the properties are or not satisfied in the analyzed promela model of OpenFlow protocol with JSpin 4.7 (based on SPIN version 4.3.0.) Without delay of SDN, all properties are satisfied. Because the control plane and data plane are separated, there could be that the delay[10] from Controller to Switch2 is longer than the sum of the delay from Controller to Switch1 and the Delay from Switch1 to Switch2. If the new rule could be not installed in Switch2 prior to the packet arriving from Switch1 to Switch2, error can occur, shown as in Fig.3.

```

JSpin Version 4.7
File Edit Spin Convert Options Settings Output SpinSpider Help LTL formula
Open Check Random Interactive Guided Weak fairness Safety Verify Stop Translate Clear Load SpinSpider Maximize

bool testaaa=false;
31 int a=1;
32 int b=2;
33 proctype HostA()
34 {
35   HostAToSwitch1?Packet->
36   if
37     ::Switch1ToHostA?Drop_Packet->printf("failure!");
38     ::Switch1ToHostA?Ok_Packet->printf("OK");
39   fi;
40 }
41 proctype Switch1()
42 {
43   do
44     :: HostAToSwitch1?Packet->
45       /*Receiving a Packet */
46     if
47       ::Switch1ToSwitch2!PacketToS2->
48       {
49         /*match in flow table */
50         MatchFlow_1=true;
51         /*symbol matched */
52         goto end;
53       }
54       ::Switch1ToController!PacketInToC_1->

```

```

pan: assertion violated (n=2) (at depth 86)
pan: wrote count.pml.trail
(Spin Version 4.3.0 -- 22 June 2007)
Warning: Search not completed
+ Partial Order Reduction
Full statespace search for:
  never claim - (none specified)
  assertion violations +
  cycle checks - (disabled by -DSAFETY)
  invalid end states +
State-vector 20 byte, depth reached 90, *** errors: 1 ***
  94488 states, stored
  36511 states, matched
  130999 transitions (= stored+matched)
  0 atomic steps
hash conflicts: 2423 (resolved)
Stats on memory usage (in Megabytes):
  2.646 equivalent memory usage for states (stored*(State-vector + overhead))
  2.021 actual memory usage for states (compression: 76.39%)
  State-vector as stored = 13 byte + 8 byte overhead
  2.097 memory used for hash table (-w19)
  0.064 memory used for DFS stack (-m2000)
  0.037 memory lost to fragmentation
  4.145 total actual memory usage

bin\spin.exe -a -v openFlow20181023.pml ... done!
bin\spin.exe -a openFlow20181023.pml ... done!
c:\mingw\bin\gcc.exe -DSAFETY -o pan pan.c ... done!
C:\jspin\jspin-examples\pan -m2000 -X ... done!
Saved openFlow20181023.pml
Saved openFlow20181023.pml
bin\spin.exe -a -v openFlow20181023.pml ... done!
bin\spin.exe -a openFlow20181023.pml ... done!

```

Figure 3. Results of verification taking into account delay

4. Conclusion

In this paper, we give an approach to verify OpenFlow protocol. First we analyse the packet processing mechanism of OpenFlow protocol, and build formal PROMELA models. Then we define properties and set assertion. Finally we apply model checker SPIN to verify. We realize that it is more efficient to pay more attention to the delay of SDN when updating OpenFlow protocol version.

In the future, we will model SDN with multiple controllers using model checker SPIN, and verify more properties.

5. Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61472230 and No.61802233) and Shandong Provincial Natural Science Foundation of China (No. ZR2016YL001, No. ZR2016YL004 , No. ZR2016YL008 and ZR2018MF010).

6. References

- [1] Lin, Y. D., Lai, Y. K., Wang, C. Y., Lai, Y. C.: OFBench: Performance Test Suite on Openflow Switches. IEEE Systems Journal, PP(99), 1--11(2017).
- [2] Sindhura, B., Shobha, K. R.: Implementation and Testing of Openflow Switch Using FPGA. In: IEEE 8th International Conference on Computing, Communication and Networking Technologies, pp.1-5. IEEE Computer Society, Delhi(2017).
- [3] Tkachova, O., Saad, I.: Method for OpenFlow Protocol Verification. In: 2015 2th International Scientific-Practical Conference, Problems of Infocommunications Science and Technology, pp.139-140. IEEE, Kharkiv(2015).
- [4] Wang, H., Zhu, H., Xiao, L., Xie, W., Lu, G. : Modeling and Verifying OpenFlow Scheduled Bundle Mechanism Using CSP. In: IEEE 42th International Conference on Computer Software and Applications, pp.376-381. IEEE Computer Society, Tokyo (2018).
- [5] Ruan, H., Wang, L., Yang, X., Dong, L., Li, H.: OpenFlow Modeling Based on CPN for Evolution Consideration and Executable Test Case Generation. In: IEEE 41th Annual Computer Software and Applications Conference, pp.72-77. IEEE Computer Society, Torino (2017).
- [6] Kang, M., Kang, E. Y., Hwang, D. Y., Kim, B. J., Nam, K. H., Shin, M. K., Choi, J.Y.: Formal Modeling and Verification of SDN-OpenFlow. In: IEEE 6th International Conference on Software Testing, Verification and Validation, pp.481-482. IEEE Computer Society (2013).
- [7] Xiao, L., Xiang, S., Zhuy, H. :Modeling and Verifying SDN with Multiple Controllers. In: SAC 2018, pp.419-422. ACM , Pau (2018).
- [8] Son, S., Shin, S., Yegneswaran, V., Porras, P. :Model Checking Invariant Security Properties in OpenFlow. In: IEEE International Conference on Communications, pp.1974-1979. IEEE(2013).
- [9] Holzmann, G.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley Professional(2004).
- [10] Zhang, W., Zhang, X., Shi, H., Zhou, L.: An Efficient Latency Monitoring Scheme in Software Defined Networks. Future Generation Computer Systems. 83,303--309(2018).