Journal of Computer Applications
计算机应用, 2020, 40(6): 1751 - 1754
ISSN 1001-9081
CODEN JYIIDU
2020-06-10
http://www.joca.cn

文章编号:1001-9081(2020)06-1751-04
DOI:10.11772/j.issn.1001-9081.2019101712

软件定义网络中控制数据平面一致性的验证

朱梦迪*,束永安
(安徽大学 计算机科学与技术学院,合肥 230601)
(*通信作者电子邮箱2524098773@qq.com)

摘　要:针对软件定义网络(SDN)中控制层网络策略与数据层流规则不一致的问题,提出了验证控制与数据平面的一致性(VeriC)检测模型。首先,通过交换机上的VeriC管道实现数据包处理子系统的功能:对数据包进行采样,采样数据包经过交换机时,对其中的标签字段进行更新;然后,更新完成后将标签值发送到服务器并保存在实际标签值组;最后,将实际标签值组与已保存的正确标签值组一起发送到验证子系统,进行一致性验证,若不通过,则进一步将两组标签值发送到定位子系统,找出流表项发生错误的交换机。通过ns-3模拟器生成一个含有4 Pod的胖树拓扑,在其中VeriC的一致性检测和故障交换机定位的准确度高于VeriDP,并且VeriC的总体性能高于2MVeri模型。理论分析和仿真结果表明,VeriC检测模型不仅能够进行一致性检测,对错误交换机进行精确定位,而且定位故障交换机所用的时间少于对比检测模型。

关键词:软件定义网络;控制平面;数据层;一致性;检测
中图分类号:TP393.0　　**文献标志码**:A

Verification of control-data plane consistency in software defined network

ZHU Mengdi*, SHU Yong'an
(*School of Computer Science and Technology*, *Anhui University*, *Hefei Anhui* 230601, *China*)

Abstract: Aiming at the problem of inconsistency between the network policies of control layer and flow rules of data layer in Software Defined Network (SDN), a detection model for Verifying control-data plane Consistency (VeriC) was proposed. Firstly, the function of the packet processing subsystem was realized through the VeriC pipeline on the switch, and the function is sampling the data packet, and updating the tag field in the sampled data packet when the packet passing through the switch. Then, after the update was completed, the tag values were sent to the server and stored in the real tag value group. Finally, the real tag value group and the stored correct tag value group were sent to the verification subsystem to perform the consistency verification. As it failed, the two groups of tag values were sent to the localization subsystem to locate the switch with flow table entry error. A fat tree topology with 4 Pod was generated by ns-3 simulator, where the accuracies of consistency detection and faulty machine location of VeriC are higher than those of VeriDP, and the overall performance of VeriC is higher than that of 2MVeri model. Theoretical analysis and simulation results show that VeriC detection model can not only perform consistency detection and accurately locate the faulty switch, but also take shorter time to locate the faulty switch compared to other comparison detection models.

Key words: Software Defined Network (SDN); control plane; data plane; consistency; detection

0　引言

软件定义网络(Software Defined Network, SDN)是一种将控制平面与数据平面分离的新型网络架构。控制平面通过北向接口协议集中管理网络应用程序,通过南向接口协议管控数据平面所有硬件设备的转发行为[1]。

与传统网络不同的是,SDN是一个流规则驱动的网络[2]。网络应用程序、操作人员可以制定网络策略,对应接口会将其转化成一组逻辑流规则,然后控制器通过北向接口将逻辑流规则部署到交换机内,最后网络设备根据交换机内的流表项对数据包进行处理,实现对应的网络策略。

在SDN中,目前有以下几方面原因,会导致数据包的转发行为与控制层流规则不一致。OpenFlow协议[3]的Barrier命令不能百分之百保证控制器下发到数据层的每个流规则全部可以正确安装[4-6];网络故障中的很大一部分原因是由于数据层的硬件问题及软件错误[7];SDN的控制平面与数据平面之间采用异步通信发送信息,一些网络更新命令(如OpenFlow协议的FlowMod命令)会使得交换机在更新流表时存在延迟,无法及时更新[5,8];最后,网络管理员的错误配置,和攻击者使用一些工具(如ONIE(http://onie.org/)),均有可能修改交换机内部的流表项,导致数据层没有实现相应的网络策略。

为了解决控制层与数据层流规则不一致的问题,文献[9-13]均提出各自的方法,用于检测多个应用程序引起的网络策略冲突,但是这些方法只能处理控制层流规则冲突的问题,却无法检测到数据层的流量转发是否符合控制层的策略要求。因此,文献[14]将数据层的可执行文件作为自制的检测工具

收稿日期:2019-10-11;**修回日期**:2020-01-12;**录用日期**:2020-01-13。　**基金项目**:安徽省自然科学基金资助项目(1408085MF125)。
作者简介:朱梦迪(1993—),女,安徽宿州人,硕士研究生,主要研究方向:软件定义网络;　束永安(1966—),男,安徽舒城人,教授,博士,主要研究方向:无线传感网、软件定义网络、下一代互联网。

的输入，用于检测数据平面是否不合理的网络配置，但检测机制无法对网络状态进行实时监测；所以，文献[15]使用探测数据包，对数据平面的信息进行收集和记录，可以动态监控转发链路上是否出现故障交换机，但该种方法探测数据包的生成时间较长，无法快速地发现数据层的错误。

继而，文献[16]提出 VeriDP 检测模型，该模型截取数据流中的数据包，在采样数据包中添加标签字段，无需单独生成探测数据包，但该方法的缺点在于控制平面需要提前保存所有正确的转发路径信息，过于消耗内存资源。文献[17]提出 2MVeri 模型，但算法时间复杂度相对较高，系统开销相对较大。因而，本文提出了验证控制与数据平面的一致性(Verifying control-data plane Consistency，VeriC)检测模型，不仅解决了内存消耗大与过滤器误报的问题，又在准确找出发生错误的交换机的前提下降低系统开销。

1 VeriC检测模型系统设计

VeriC 检测模型由 VeriC 模型对应服务器中的验证子系统、定位子系统以及交换机的数据包处理子系统共同实现。控制器和交换机通过 OpenFlow 协议交互的信息，均会被拦截并保存在模型服务器内。

VeriC 模型借助管道技术实现数据包处理子系统的功能，由交换机的快速路径技术完成，并且与 OpenFlow 所用的管道是分离的[16]。VeriC 在入口交换机上对数据包进行采样，在数据包的转发过程中对数据包进行标记，每经过一个交换机，更新采样数据包标签字段，随后将更新后的标签值发送到 VeriC 服务器，分别存储在上报的标签值 $PT_g[i]$(rePorted Tag)和上报布隆过滤器值 BF_p(reported Bloom Filter)。

VeriC 模型对通过南向接口协议交互的消息进行拦截，当被采样数据包在出口交换机标记动作完成后，模型服务器根据初始二维向量、正确的转发表、交换机 ID 以及每个交换机对应的二维矩阵，计算出一组正确的标签值 CT_g(Corrected Tag)和正确的布隆过滤器值 BF_c(corrected Bloom Filter)。每组流规则对应 $CT_g[i]$ 和 BF_c，$PT_g[i]$ 和 BF_p 均会被发送到验证子系统进行一致性检测；若存在异常，标签值组被进一步发送到到定位子系统，对故障的交换机进行定位。VeriC 模型结构如图 1 所示。

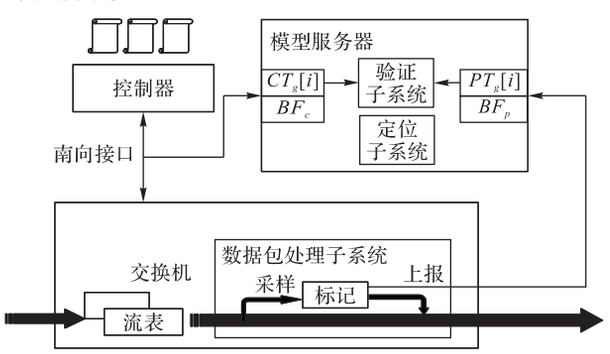

图 1 VeriC模型结构

Fig. 1 VeriC model structure

2 详细设计

2.1 数据包处理系统

2.1.1 数据包采样

在数据包处理系统中，首先要对交换机中数据包进行采样[17]。在入口交换机，使用数据包字段中的一个位来记录数据包是否被采样，设最近一次采样时间为 T_n^f，采样间隔为 T_s^f。若入口交换机在时间 t 收到数据包，数据包的采样时间满足式(1)，则该数据包即为采样数据包：

$$t - T_n^f \geqslant T_s^f \tag{1}$$

2.1.2 生成交换机矩阵

数据层交换机在初始化时，控制器会为其分配一个受限制的二维矩阵，用于对交换机进行标记[17]。为了使标签信息能够准确代表数据包的转发路径，在为每台交换机分配二维矩阵时，需满足以下要求：

1)二维矩阵中的四个元素均为素数，降低矩阵的重复率；

2)相邻交换机是不可交换矩阵，即相邻交换机的矩阵信息不满足矩阵乘法交换律；

3)交换机的二维矩阵不可以相同。

2.1.3 字符注释

为了便于描述，对本文涉及的符号进行注释，如表 1 所示。

表 1 符号注释

Tab. 1 Notations and related meanings

符号	含义
$\boldsymbol{M}$	交换机的二维矩阵值
ID	交换机的 ID
$(\boldsymbol{v}_1,\boldsymbol{v}_2)$	分配给采样数据包的二维向量值
BF_c	控制器中存储的正确 BF 值
BF_p	采样数据包上传到控制器 BF 值
$CT_g[i]$	控制器中存储的地址为 i 的正确标签值 CT_g
$PT_g[i]$	上传到控制器的地址为 i 的实际标签值 PT_g
tx	数组 CT_g 的存储个数
ty	数组 PT_g 的存储个数

2.1.4 标记采样数据包

对于每个采样数据包，会在包头中插入 BF 字段和一个二维向量字段 $(\boldsymbol{v}_1,\boldsymbol{v}_2)$。其中，BF 字段的初始值为 0(见算法 1 第 1)~2)行)，表示采样数据包经过交换机 ID 的集合。采样数据包每经过一台交换机，BF 字段就会将新交换机的 ID 存储到集合中，当数据包到达出口交换机或者要执行删除操作前，将 BF 字段发送到控制平面进行保存(见算法 1 第 10)~12)行)。

控制器基于采样数据包的性质，首先为二维向量字段 $(\boldsymbol{v}_1,\boldsymbol{v}_2)$ 分配初始值(见算法 1 第 3)~5)行)，并将初始值上传到控制层。在控制层中，使用〈inport, outport, header〉作为索引，存储到同一采样数据包的标签值集合，其中 inport 是采样数据包的入端口编号，outport 是采样数据包的出端口编号，header 是采样数据包的包头信息；然后，当采样数据包经过交换机时，二维向量会与交换机对应的二维矩阵相乘，运算值会重新存储在二维向量字段中(见算法 1 第 6)行)。二维向量字段每次完成更新后，会将标签字段发送到模型服务器进行保存(见算法 1 第 7)~9)行)。数据包处理算法描述如下。

算法 1 数据包处理算法(Packet processing algorithm)。

输入：$\boldsymbol{M}$，ID，$(\boldsymbol{v}_1,\boldsymbol{v}_2)$，$BF$；

```
1)  if the input port is an edge port then
2)    BF=0;
3)    assign (v1,v2) to the packet based on flow;
                    //BF字段初始值为0,基于流为数据包分配值
4)    if the sampled packet comes from a new flow then
                                  //采样数据包来自新的数据流
5)      send(v1,v2) to the controller;
                      //将(v1,v2)发送给控制器作为初始标签值
6)  BF=BF∪BF(ID);(v1,v2)=(v1,v2)*M;
                      //每经过一台交换机,就对数据包进行更新
7)  if tagging has completed then
```

//标签更新完成后

8)　send (v_1,v_2) to the controller;

9)　PT_g=(v_1,v_2);

10) if the port is an edge port OR drop the packet then

11)　send BF to the controller plane;

12)　BF_p=BF;

//如果出端口是边缘端口或删除数据包,

//将BF发送到控制层,存储为BF_p;

13) end if

2.2　一致性验证以及故障定位

2.2.1　一致性验证

当采样数据包在出口交换机完成转发后,会在控制层检测控制平面与数据平面之间流规则的一致性。如果服务器中存储的BF_p与BF_c相同,且二维向量标签值$PT_g[ty-1]$与控制器中的正确标签值$CT_g[tx-1]$相等;那么,数据包的实际转发路径与控制器中期望数据包的转发路径一致(见算法2第1)~2)行)。

若控制器中BF_p与BF_c标签值不同,而tx与ty的值相等,$PT_g[ty-1]$与$CT_g[tx-1]$也相等,则要考虑字段BF的误报情况:从BF字段值初始值开始比较,假设在第i台交换机,发现BF_p与BF_c不一致,检查$PT_g[i]$与$CT_g[i]$或$PT_g[i-1]$与$CT_g[i-1]$是否相等。若相等,表示第i台交换机在更新BF字段时,出现了误报的情况,则将正确的值更新到$BF_p[i]$中;若有一组值不相等,直接返回False(见算法2第3)~12)行)。验证算法描述如下。

算法2　验证算法(Verification algorithm)。

输入　BF_c,BF_p,CT_g,PT_g,tx,ty;

输出　返回consistent或者inconsistent。

1)　if (BF_p==BF_c && $PT_g[ty-1]$==$CT_g[tx-1]$) then

2)　　return True;

//若BF_p与PT_g字段均与服务器中保存的相等,则返回True

3)　else if (BF_p! =BF_c && ty==tx && $PT_g[ty-1]$==$CT_g[tx-1]$)

//考虑BF字段的误报情况

4)　　for (i=1;i<ty;i++)

5)　　if ($BF[i]$!=$BF_c[i]$ && $PT_g[i-1]$==$CT_g[i-1]$ && $PT_g[i]$==$CT_g[i]$) then

6)　　$BF_p[i]$=$BF_c[i]$;

//用正确的标签值替换误报值

7)　　if i==ty-1 then

8)　　return True;

//当i=ty-1时,返回True

9)　　else

10)　　return False;

11) else　　//其他情况,则返回False

12)　return False;

13) end if

2.2.2　故障交换机定位

在验证算法中,对于返回False的标签组,会进入故障定位算法,查找发生错误的交换机。当$PT_g[i]$与$CT_g[i]$不一致,并且$BF_p[i]$与$BF_c[i]$也不相等时,i对应的交换机即为故障位置(见算法3第1)~4)行)。对于字段BF的误报情况,已经在验证算法中实现修正,所以PT_g中的标签值基本不会出现错误情况。定位算法描述如下。

算法3　定位算法(Localization algorithm)。

输入　BF_p,CT_g,PT_g,ty;

输出　返回i。

1)　if (BF_p && PT_g)　　//BF_p与PT_g均存在时

2)　　for(i=0;i<ty;i++)

3)　　if ($PT_g[i]$! =$CT_g[i]$ && $BF_p[i]$! =$BF_c[i]$)

4)　　return i;

//PT_g与CT_g首次不等时,i为故障交换机

5)　end if

3　实验实现与结果分析

3.1　实验设置

本文通过ns-3模拟器(https: //www. nsnam. org/)对VeriC模型进行了验证。ns-3是一个离散事件网络模拟器,可以在单个计算机上为用户提供模拟实验。本文实验计算机的配置如下:Intel Core i7 CPU 3. 6 GHz处理器,16 GB内存,Ubuntu 16. 04操作系统。

本文在ns-3模拟器中仿真一个k值为4的胖树结构,胖树拓扑中包含16台主机和20台交换机。在仿真实验中,通过式(2)来设置BF字段的存储位数。

$$g_i(x) = h_1(x) + ih_2(x);\ i = 0, 1, 2 \tag{2}$$

本文使用$g_i(x)$中的前四个比特位来设置16位的BF,在ns-3模拟器中通过UniformRandomVariableClass来生成并分配给采样数据包的二维初始向量值和每台交换机的二维矩阵值[17]。

3.2　结果分析

网络拓扑构建完成之后,首先使16台主机实现互通,完成交换机上流表项的全部安装。然后随机选择一些互通的主机对,对于每一组被选中的主机对,在流量转发的路径中随机抽取一台交换机,修改交换机内部存储的流表项,观察实验结果。已知数据层的转发路径与控制平面的不一致,通过对比VeriC的验证结果,发现采样数据包上传的实际标签值与控制平面中保存的预期标签值不一致。

本文通过相对假阴性率和绝对假阴性率来展示VeriC的实验结果。对交换机中的流表项进行错误设置。m是设置错误流表项之后,交换机生成的数据包数量。m_1是设置错误流表项之后,从入口交换机到达目的地数据包数量。m_2是设置错误流表项之后,从入口交换机到达目的地,并且报文中的实际标签值与正确标签值一致的数据包数量。绝对假阴性率为m_2/m,相对假阴性率为m_2/m_1。

本文将VeriC、2MVeri、VeriDP三种模型的相对假阴性率和绝对假阴性率进行了对比[16-17],VeriC检测模型的相对假阴性率和绝对假阴性率不会因为Bloom过滤器位数的增加而改变这是由于VeriC检测模型,并不仅仅依据Bloom过滤器中保存的交换机ID值,进行一致性检测,而是根据每组流规则对应$CT_g[i]$和BF_c,$PT_g[i]$和BF_p的值,检测数据层的转发行为是否与控制层一致。在两组标签值的共同作用下,消除了Bloom过滤器的误报情况,使得一致性检测更加准确,实验结果如图2和图3所示。

VeriC模型的定位算法,与交换机接口数、端口对的数量无关。假设一个网络中(inport,outport)对的数量为n,网络流量数为m,每台交换机的端口数为p,转发路径表中的最大跳数为h,三种检测模型的运算开销如表2所示。

由表2可知,使用三种检测模型定位故障交换机时,VeriDP的内存开销和算法的时间复杂度均最高,而2MVeri、VeriC模型不需要构建路径表。2MVeri算法的时间复杂度为是$O(1)+O(ph)$,VeriC算法仅为$2O(h)$,2MVeri模型通过矩阵的逆,定位故障交换机,当一组流规则转发的交换机数和端口数越多,算法的时间复杂度也就越高。VeriC检测模型验证算法的时间复杂度大于2MVeri验证算法,但综合考虑,VeriC模型算法的复杂度总和是最小的。

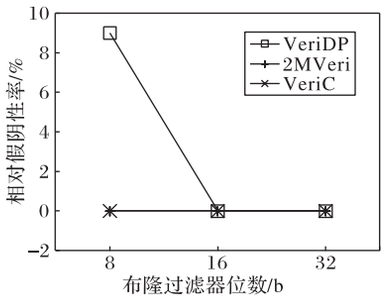

图2 相对假阴性率

Fig. 2 Relative false negative rate

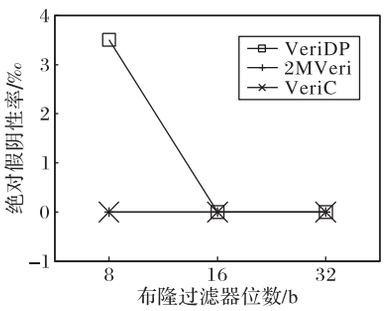

图3 绝对假阴性率

Fig. 3 Absolute false negative rate

表2 三种方法运算开销

Tab. 2 Computational overheads of three methods

模型	路径表	验证算法	定位算法
VeriDP	$O(nmh)$	$O(n)+O(m)$	$O(ph)(O(n)+O(m)+O(h))$
2MVeri	0	$O(1)$	$O(ph)$
VeriC	0	$O(h)$	$O(h)$

4 结语

本文提出了VeriC检测模型,用于检测控制平面与数据层之间流规则的一致性。VeriC检测模型通过对交换机的数据包进行采样,修改采样数据包,记录数据包的转发信息,完成控制平面与数据层的一致性检测,最终找出故障交换机。目前,已经在胖树结构上完成了仿真实验,实验证明VeriC模型能100%准确地进行故障定位。

VeriC模型虽然能够完成一致性检测,但是基本条件是控制平面流规则符合一致性检查。当多个应用程序同时对网络进行操作时,生成的流规则之间可能会存在冲突。所以,未来会对控制平面流规则的一致性进行检测。

参考文献(References)

[1] 范恂毅,张晓和. 新一代SDN:VMware NSX网络原理与实践[M]. 北京:人民邮电出版社, 2016:9-12. (FAN X Y, ZHANG X H. The New Generation SDN: VMware NSX network Principles and Practices[M]. Beijing: Posts and Telecom Press, 2016:9-12.)

[2] 黄韬,刘江,魏亮,等. 软件定义网络核心原理与应用实践[J]. 互联网天地, 2015(1): 95-95. (HUANG T, LIU J, WEI L, et al. SDN core principles and application practice[J]. China Internet, 2015(1): 95-95.)

[3] OpenFlow switch specification version 1.5.1[EB/OL]. [2019-01-10]. https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdnassl.com/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[4] BU K, WEN X, YANG B, et al. Is every flow on the right track?: inspect SDN forwarding with RuleScope[C]// Proceedings of the 35th Annual IEEE International Conference on Computer Communications. Piscataway: IEEE, 2016: 1-9.

[5] KUŹNIAR M, PEREŠÍNI P, KOSTIĆD. What you need to know about SDN flow tables[C]// Proceedings of the 2015 International Conference on Passive and Active Network Measurement, LNCS 8995. Cham: Springer, 2015: 347-359.

[6] ROTSOS C, SARRAR N, UHLIG S, et al. OFLOPS: an open framework for OpenFlow switch evaluation[C]// Proceedings of the 2012 International Conference on Passive and Active Network Measurement, LNCS 7192. Berlin: Springer, 2012: 85-95.

[7] ZENG H, KAZEMIAN P, VARGHESE G, et al. Automatic test packet generation[C]// Proceedings of the 2012 International Conference on Emerging Networking Experiments and Technologies. New York: ACM, 2012: 241-252.

[8] SHUKLA A, SCHMID S, FELDMANN A, et al. Towards transiently secure updates in asynchronous SDNs[C]// Proceedings ofthe 2016 ACM SIGCOMM Conference. New York: ACM, 2016:597-598.

[9] 李嘉麒. 软件定义网络控制平面一致性与负载均衡研究[D]. 北京:北京工业大学,2018:13-17. (LI J Q. Research on consistency and load balancing on software-defined networking control plane[D]. Beijing: Beijing University of Technology, 2018:13-17.)

[10] ZHOU W, CROFT J, LIU B, et al. NEAt: network error auto-correct[C]// Proceedings of the 2017 Symposium on SDN Research. New York: ACM, 2017: 157-163.

[11] SON S, SHIN S, YEGNESWARAN V, et al. Model checking invariant security properties in OpenFlow[C]// Proceedings of the 2013 IEEE International Conference on Communications. Piscataway: IEEE, 2013: 1974-1979.

[12] WU Y, CHEN A, HAEBERLEN A, et al. Automated bug removal for software-defined network[C]// Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2017: 719-733.

[13] DOBRESCU M, ARGYRAKI K. Software dataplane verification[C]// Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2014: 101-114.

[14] TSENG F H, CHANG K D, LIAO S C, et al. sPing: a user-centred debugging mechanism for software defined networks[J]. IET Networks, 2017, 6(2): 39-46.

[15] ZHANG H, LUMEZANU C, RHEE J, et al. Enabling layer 2 pathlet tracing through context encoding in software-defined networking[C]// Proceedings of the 3rd Workshop on Hot topics in Software Defined Networking. New York: ACM, 2014: 169-174.

[16] ZHANG P, LI H, HU C, et al. Mind the gap: monitoring the control-data plane consistency in software defined networks[C]// Proceedings of the 12th International Conference on Emerging Networking Experiments and Technologies. New York: ACM, 2016: 19-33.

[17] LEI K, LI K, HUANG J, et al. Measuring the control-data plane consistency in software defined networking[C]// Proceedings of the 2018 IEEE International Conference on Communications. Piscataway: IEEE, 2018: 1-7.

This work is partially supported by Anhui Provincial Natural Science Foundation (1408085MF125).

ZHU Mengdi, born in 1993, M. S. candidate. Her research interests include software defined network.

SHU Yong'an, born in 1966, Ph. D., professor. His research interests include wireless sensor network, software defined network, next-generation internet.