CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 40 卷第 10 期 2018 年 10 月
Vol. 40, No. 10, Oct. 2018

文章编号:1007-130X(2018)10-1757-09

OpenFlow 协议 IPv6 属性测试研究*

李元平[1,3],李　华[1,2],赵俊岚[3],阮宏玮[1]

(1.内蒙古大学计算机学院,内蒙古 呼和浩特 010021;2.内蒙古大学网络中心,内蒙古 呼和浩特 010021;
3.内蒙古财经大学网络中心,内蒙古 呼和浩特 010070)

摘　要:鉴于 SDN 网络中数据转发与控制相互分离的特性,OpenFlow 协议在其南向接口中扮演着重要的角色。随着下一代互联网的发展,IPv4 可供分配的地址资源已然耗尽,瓶颈地位益发凸显。如何尽快部署 IPv6,使其服务于社会生产与生活,使得当前网络与 IPv6 网络长期共存或平滑过渡到 IPv6 网络是工业界与学术界要解决的问题。SDN 则提供了这样一个选项,其中 OpenFlow 协议是否支持 IPv6 协议便成为我们关注的重点。通过形式化方法对 OpenFlow 协议进行形式化建模,得到其非确定性有限状态机模型,在此基础上得到其测试生成树,以指导测试。同时,对于其是否支持 IPv6 进行重点关注,利用组合测试的方法,产生了 167 条测试例。完成了测试引擎的开发,支持高效的测试生成算法,以及测试执行与判定。利用此测试引擎,以上述测试例为测试输入,执行测试过程,同时进一步对测试结果进行分析,得到了定量的分析结果,符合预期要求。

关键词:NFSM;IPv6;OpenFlow;测试引擎

中图分类号:TN915.04　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2018.10.006

IPv6 property testing for OpenFlow protocol

LI Yuan-ping[1,3], LI Hua[1,2], ZHAO Jun-lan[3], RUAN Hong-wei[1]

(1. College of Computer Science, Inner Mongolia University, Hohhot 010021;
2. Center of Network and Information, Inner Mongolia University, Hohhot 010021;
3. Center of Network and Information, Inner Mongolia University of Finance and Economics, Hohhot 010070, China)

Abstract: Since the data forwarding and control are separated in SDN networks, the OpenFlow protocol plays an important role in its southbound interface. With the development of the next generation Internet, available IPv4 address resources are almost used up, and the bottleneck of IPv4 networks is highlighted. How to deploy IPv6 networks quickly, making their contribution for social production and life, and realizing long-term coexistence of current networks and IPv6 networks or a smooth transition to IPv6 networks, becomes an urgent problem for the industry and academia. SDN networks provide such an option, so verifying whether the OpenFlow protocol supports IPv6 protocol has attracted our attention. We construct a formal model for the OpenFlow protocol, and build a nondeterministic finite state machine (NFSM). In order to guide the testing, we also achieve a test generation tree. We focus on identifying whether this protocol supports the IPv6 protocol, and generate 167 test cases by a combination test method. We also develop a test engine, which supports efficient test generation algorithms, test operation and identification. We test our test engine on the test cases, and analyze the test results. The quantitative analysis results meet the expected requirements.

* **收稿日期**:2017-08-31;**修回日期**:2017-12-01
基金项目:国家自然科学基金(61862047);内蒙古自治区科技计划项目(2018MS0612);赛尔网络下一代互联网技术创新项目(NGII20150112)
通信作者:李华(cslihua@imu.edu.cn)
通信地址:010021 内蒙古呼和浩特内蒙古大学计算机学院
Address: College of Computer Science, Inner Mongolia University, Hohhot 010021, Inner Mongolia, P. R. China

Key words:nondeterministic finite state machine (NFSM);IPv6;OpenFlow;test engine

1 引言

软件定义网络 SDN(Software Defined Network)通过将网络设备控制平面与数据平面相分离,实现了网络流量的灵活控制,使网络作为管道变得更加智能。控制平面与数据平面相分离,使得网络研究者可以分别站在控制平面与数据平面的角度对网络行为进行更好的抽象,总体来说基本包括转发行为的抽象、分布式状态的抽象与网络管理的抽象。由此带来的问题便是,分离之后的控制平面如何与数据平面进行通信,感知数据平面的状态,同时将控制信息下发到数据平面以指导流量的转发,其中底层南向接口扮演着重要的角色。McKeown 等人[1]将斯坦福大学的两栋教学楼网络改造成了支持 OpenFlow 协议的 SDN 网络,提出了新的网络模式,在工业界与学术界引出一股清流。开放网络基金会 ONF(Open Networking Foundation)将 OpenFlow 协议作为 SDN 网络的南向接口标准,并不断提出新的改进版本,且得到了各个厂商与学术社区的支持。Google 在全球范围内的数据中心之间利用 SDN 控制相关流量,每个站点基于 OpenFlow 进行互联[2],使其广域网带宽利用率得到大幅度提升,降低了企业生产成本,为 OpenFlow 的发展与 SDN 的应用起到了示范效应。从近几年的情形来看,SDN 是最大的研究热点[3]。鉴于 SDN 网络的蓬勃发展,需要对其南向接口标准协议 OpenFlow 做深入研究。

数据平面的验证测试方法可以分为三大类:一类采用逻辑描述方法刻画被研究属性或者系统,然后通过模型检测、定理证明的方法检测属性是否被违背。例如,有建模采用逻辑公式、使用模型检验,借助 CTL(Computational Tree Logic)、LTL(Linear Temporal Logic)、PCTL(Probabilistic Computation Tree Logic)和 DPRL(Data Path Requirement Language)等手段描述被验证的属性(例如无环等),通过辅助工具的执行完成属性是否违背的检测。第二类将数据平面或控制平面建模为形式化模型刻画被研究对象,通过系统化搜索状态空间,验证行为和策略的一致性。Batfish 以设备配置文件、网络拓扑以及相应的环境信息为输入,逻辑构建整体网络控制平面与数据平面,利用路径搜索的方式进行相关验证,并且根据验证错误定位到配置文件的相应位置,修改之后迭代进行上述步骤进行重复验证,从而避免了因网络错误配置所导致的生产损失[4]。文献[5]提出了高层次的抽象表示 ARC (Abstract Representation for Control plane),该模型在不产生数据平面的情况下对控制平面的多个网络不变式如可达、隔离、单点故障隐患、SFC(Service Function Chain)的遍历以及多路径的等价性等属性做了验证,大幅度提高了验证的时间效率。Canini 等人[6]利用等价类减少测试输入,构建 OpenFlow 应用模型,同时以此理论为指导,进一步构建了测试工具 NICE[7],将模型检验应用到控制器、交换机和主机所构成的整体系统状态中。在符号执行和模型检测相结合的思路引领下,减少测试输入数据,对 OpenFlow 协议的应用程序进行了相关测试。测试与验证工具 Kinetic[8]针对 OpenFlow 协议应用程序,提出了模型描述语言,具体考虑到大部分的网络任务是相互独立的,结合工作流串联与并联接入的方式构建组合模型,有效减少了状态空间的规模,缓解了状态空间爆炸的问题。第三类是直接对 OpenFlow 协议进行测试测量。OFLOPS[9]提出了一种 SDN 的性能测试框架,该工作的主要目的是针对性能测试。OFTest[10]是一种黑盒测试工具,但是没有采用形式化方法进行测试例的指导生成。该工具对于 OpenFlow 协议规范 1.0 和 1.1 提供了上百个测试例。但是,由于该工具采用手工编辑测试例的形式,测试的全面性与完整性难以得到保证。

当前 OpenFlow 协议的最新版本为 1.5.1[11],稳定版本为 1.3.4。针对 SDN 交互协议 OpenFlow 的测试中,SOFT[12]采用基于形式化方法的白盒测试,提出了一种测试 OpenFlow 交换机互操作的方法。采用符号执行技术找到运行于交换机中各个 OpenFlow 代理实现的不一致性,但是该方法不能发现与规范不一致的实现错误。当前 ONF 已经给出了 OpenFlow 协议 1.3 版本的基本测试例描述说明,但是该说明同样采用手工描述的方式,没有利用形式化的手段进行测试例的生成,与 OFTest 工具相比,同存在测试完整性以及全面性难以保证的问题。文献[13]在 SDN 网络中采用基于 IPv6 流标签的转发形式,利用 IPv6 协议的流标签在每个 IPv6 流中都彼此不同的特性,将此字段作为转发凭据,减少路由的匹配字段,增加了转发性能,其思路值得借鉴。

综上,当前针对 SDN 数据平面的功能性测试,无论是白盒测试还是黑盒测试均主要采用非形式化的处理手段,测试的覆盖性无法从根本上得到保证。本文利用非确定性有限状态机模型,在对 OpenFlow 协议进行形式化建模的基础上,对 OpenFlow 协议进行一致性测试研究,同时结合当前 IPv6 的应用趋势,重点关注 OpenFlow 对 IPv6 的支持程度。

2 OpenFlow 协议形式化建模与抽象测试生成

2.1 OpenFlow 协议形式化建模

针对 OpenFlow 协议的形式化建模,可以通过多种方式,面向不同的关注点进行。我们曾通过时间着色 Petri 网的方式对 OpenFlow 协议进行形式化建模,对其中关键变迁行为赋予时间属性,并对其做出定量的分析,得到定性与定量相结合的分析验证结论[14]。本文根据 OpenFlow 协议在数据平面与控制平面中的交互特点,采用非确定性有限状态机 NFSM(Nondeterministic Finite State Machine)的形式对其进行形式化建模。通常意义上的有限状态机都是指确定性有限状态机,而确定性有限状态机只是非确定有限状态机的特殊形式。针对非确定性有限状态机的研究,我们通常关注其中的可观察部分非确定有限状态机 OPNFSM(Observed Partial Nondeterministic Finite State Machine)[15]。在得到可观察部分非确定有限状态机的基础上,对其进行化简,得到相应最小化的非确定有限状态机。关于最小化以及相应的可区分状态分别见文献[15],可观察部分非确定有限状态机的完整定义与相关的完全测试套导出算法请见文献[16]。

OpenFlow 协议描述本身不存在状态机或者其他形式化模型,出于测试的需要,使其满足一定的覆盖性要求,根据 OpenFlow 协议描述以及 Floodlight 控制器的相应源码信息[17],我们构建了 OpenFlow 协议实现的非确定性有限状态机,用于指导抽象测试例生成。需指出,由于所构建的状态机专注于控制器与交换机之间的信息交互,此部分功能是控制器与 OpenFlow 交换机的必备功能,与具体控制器实现以及具体 OpenFlow 交换机实现相独立,具备一般性特征。因此,此状态机模型对于 OpenFlow 协议的测试具有一般意义上的指导作用。此状态机主要包括两个子状态机,其中一个子状态机主要描述控制器与交换机之间相互发送 OpenFlow 协议 Hello 消息以及 FeatureRequest 消息,并且接收相应的 FeatureReply 消息,为控制器与交换机之间准备逻辑通信信道,为交互控制消息如 Packet_In 等消息做准备工作,同时负责维护此逻辑信道,周期性发送与接收 Hello 消息,使其处于活跃状态。我们称此状态机为握手有限状态机。建立逻辑信道之后,控制器与交换机之间交互同步消息与异步消息,处理相应流表信息,此状态机承担了几乎全部的交互工作,是建模的重点,我们称之为交互有限状态机。上述形式化模型分别如图 1 与图 2 所示,其中交互有限状态机相对结构复杂,节点较多,故截取部分有限状态机展示。实际测试生成过程中,上述非确定有限状态机需要合并为一个整体有限状态机使用。

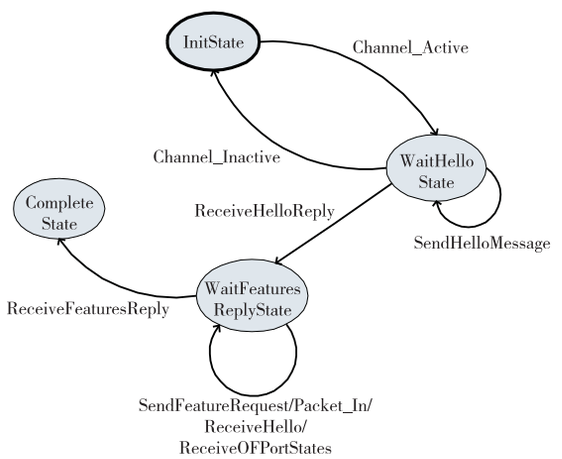


Figure 1 HandShake FSM

图 1 握手有限状态机

在得到非确定有限状态机的基础上,将其以物理文件的形式存储。本文采用 XML 的格式存储状态信息,与有限状态机模型的对应如下所示:

```
〈xs:simpleType name="FSM_Vertex_Choice"〉
  〈xs:restriction base="xs:NMTOKEN"〉
    〈xs:enumeration value="Channel_Process_InitState"/〉
      〈xs:enumeration value="WaitHelloState"/〉
      〈xs:enumeration value="WaitFeaturesReplyState"/〉
      〈xs:enumeration value="CompleteState"/〉
       〈xs:enumeration value="HandShake_InitState"/〉
        〈xs:enumeration value="WaitPortDescStatsReplyState"/〉
       〈xs:enumeration value="WaitConfigReplyState"/〉
  〈/xs:restriction〉
〈/xs:simpleType〉
```

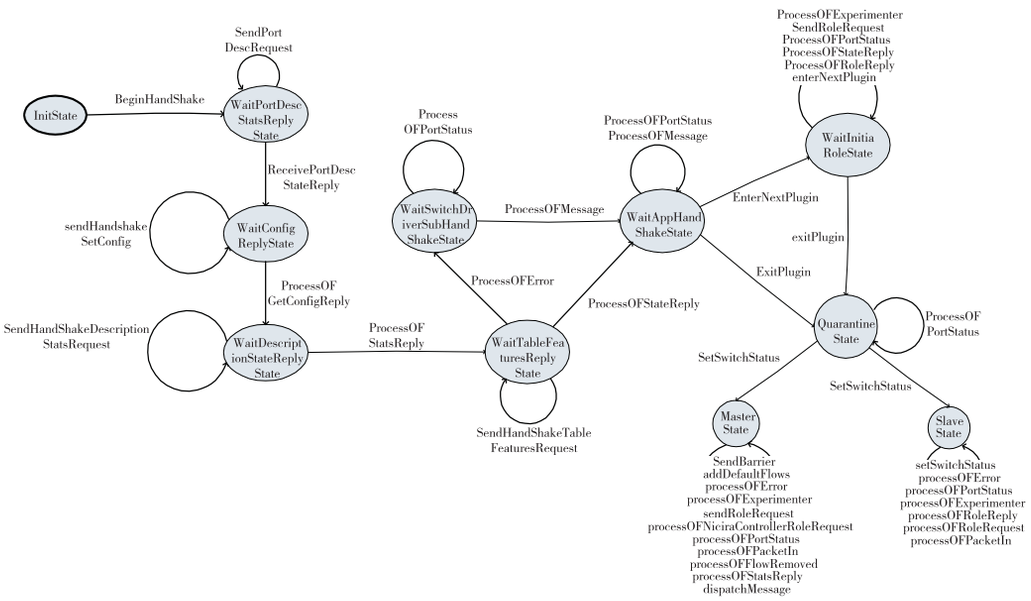


Figure 2 Interactive FSM

图 2 交互有限状态机

2.2 OpenFlow 协议抽象测试生成

在非确定性有限状态机的基础上构建测试生成树生成算法,具体描述如算法 1 所示,算法复杂度为 $O(n^2)$。本文实现了该算法,算法运行得到测试生成树,此测试生成树同样采用 XML 格式存储,其样例代码如下所示:

```
〈Tree_Node〉〈Edge_Node_Id〉e476d387-1353-4110-b00a-fb4fc398ae75〈/Edge_Node_Id〉〈Parent_Edge_Node_Id〉48801253-9904-4a13-9084-972d36fbf23f〈/Parent_Edge_Node_Id〉
〈Graph_Edge〉〈From_Graph_Vertex〉WaitFeaturesReplyState〈/From_Graph_Vertex〉〈To_From_Graph_Vertex〉CompleteState〈/To_From_Graph_Vertex〉〈Input_Data〉ReceiveFeaturesReply〈/Input_Data〉
〈Output_Data〉0〈/Output_Data〉
〈/Graph_Edge〉
〈Succeed_Edge_Node〉〈Edge_Node_Id〉972c8517-b76b-4be9-ae60-9614333c7347〈/Edge_Node_Id〉〈Edge_Node_Id〉b0c1d1a3-6592-4dea-acb6-7c10b740b026〈/Edge_Node_Id〉〈Edge_Node_Id〉8cac6616-df7f-431f-b702-ce137535915a〈/Edge_Node_Id〉
〈/Succeed_Edge_Node〉
〈/Tree_Node〉
```

算法 1 测试生成树构建算法

Input:Infomation Description About FSM。

Output:Test Generated Tree。

```
Initialize the FSM information according to the configuration file;
push the initial vertex into stack;
while(! stack.isEmpty()){
    pop the current stack node;
  if the current node has not been processed,find the corresponding edge information of this node;
      if the current edge set is null,then set state of this node to processed;
      else {process every edge;
        push tail node of every edge into stack;
        check whether this edge needs to check;
          if this edge need not to process,continue;
          else process this edge;
        if this edge is processed correctly,set the edge to generate tree branches for test;
      }
    if each edge is processed correctly,set the current node to processed;
  Goto 3;
  }
  write info into output file.
```

其最终树形结构如图 3 所示,因为得到的测试生成树结构比较大,故截取其中部分树节点进行展示。依托此测试生成树以及相应的 XML 存储文件,利用循环遍历测试生成树的方式,推导得到 53 条抽象测试例,用于指导我们的实际测试生成。同时,关于 OpenFlow 协议的流表下发部分,在完成控制器与交换机之间交互部分测试需求的基础上,我们重点关注 OpenFlow 协议对于 IPv6 协议的支持程度。借助 Floodlight 控制器的 Static Flow Pusher 模块和交换机模块的 write 方法,进行相应流表的下发处理。同时利用 netty 框架的 channel 模块进行 OpenFlow 消息的相应发送与接收。

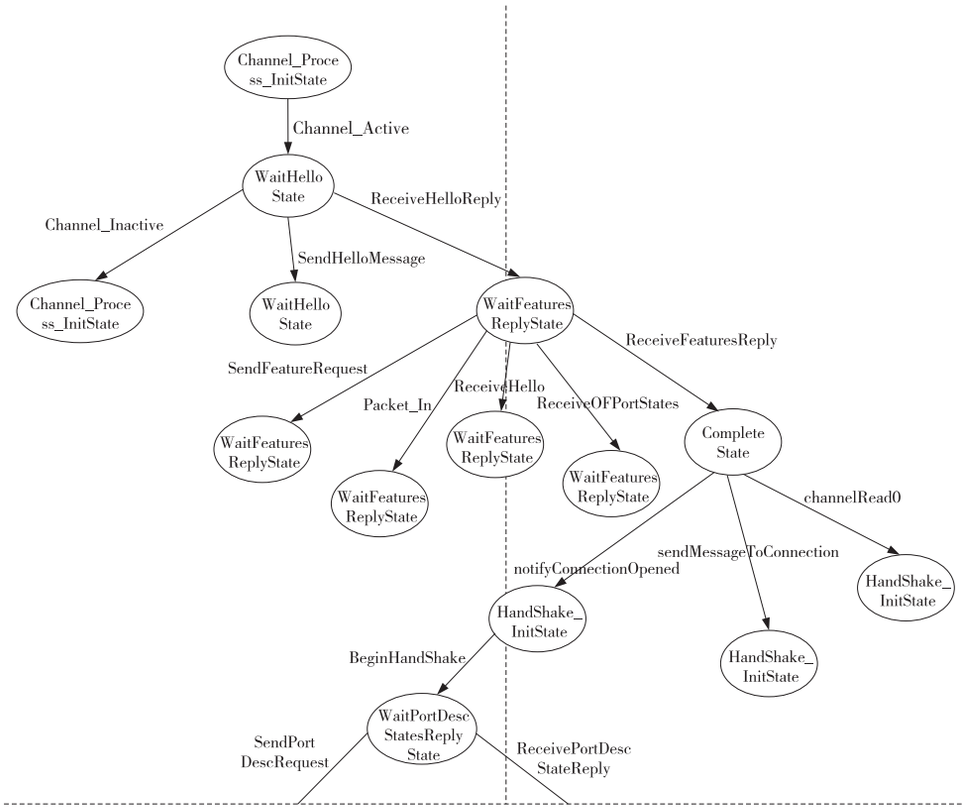


Figure 3 Test generation tree of OpenFlow protocol

图 3 OpenFlow 协议测试生成树

3 测试引擎构建、执行与结果分析

文献[18]提出了测试的主要指导原则,分别为统一的测试理论、基于模型的测试、100%的自动化以及高效的测试引擎。我们已经对 OpenFlow 协议进行了形式化建模,利用形式化模型自动遍历的方式得到抽象测试例,并在其指导之下结合测试环境生成了可执行测试例。我们将在上述工作的基础上进一步构建高效的测试引擎替代传统的手工执行方式来实际执行测试,以满足测试需求。关于测试的整体框架如图 4 所示,本文重点实现了图 4 中的测试引擎构件。本次测试工作中,实际的被测对象是 OpenFlow 协议 1.3 版本,所构建的网络拓扑图如图 5 所示。

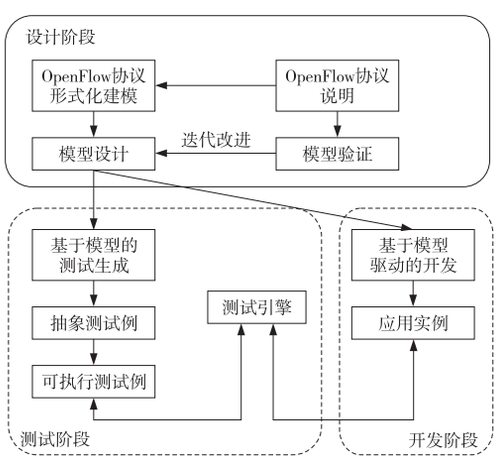


Figure 4 Test frame

图 4 测试框架

3.1 OpenFlow 协议中 IPv6 支持字段及其前置条件

根据 OpenFlow 协议的描述说明,其支持的 IPv6 相应的字段包括 IPv6 基本头部中的 IPv6 源地址、IPv6 目的地址、IPv6 流标签、IPv6 扩展头字段、ICMPv6 协议的 Type 字段与 Code 字段以及 ND 协议中的 Target 字段、Sll 字段以及 Tll 字段[11]。上述字段只有在满足一定前置条件的情况下,字段所在流表信息方可以生效,具体前置条件

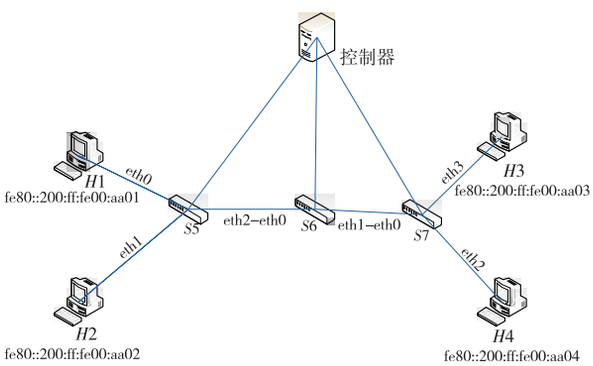


Figure 5 Network topology graph of OpenFlow protocol

图 5 OpenFlow 协议测试网络拓扑图

见表1。在测试执行过程中,该前置条件是我们需要关注的部分。如果不是前置条件所导致的失败,即为真实的测试失败,否则需变其前置条件使其更完善,重新执行测试。

Table 1 Precondition of IPv6 support fields in OpenFlow protocol

表1 OpenFlow协议中IPv6支持字段前置条件

域名称	比特位	支持掩码	前置条件
OXM_OF_IPV6_SRC	128	Yes	ETH TYPE=0x86dd
OXM_OF_IPV6_DST	128	Yes	ETH TYPE=0x86dd
OXM_OF_IPV6_FLABEL	20	Yes	ETH TYPE=0x86dd
OXM_OF_IPV6_EXTHDR	9	Yes	ETH TYPE=0x86dd
OXM_OF_ICMPV6_TYPE	8	No	IP PROTO=58
OXM_OF_ICMPV6_CODE	8	No	IP PROTO=58
OXM_OF_IPV6_ND_TARGET	128	No	ICMPV6 TYPE=135 or ICMPV6 TYPE=136
OXM_OF_IPV6_ND_SLL	48	No	ICMPV6 TYPE=135
OXM_OF_IPV6_ND_TLL	48	No	ICMPV6 TYPE=136

3.2 OpenFlow协议测试执行引擎设计

通过关于形式化与抽象测试生成部分的前导工作[14,15],并结合当前测试实际,考虑通用性以及可扩展性的需求,以XML格式文件为载体,构建得到了可执行测试文件,具体样例代码如下所示:

```
〈? xml version="1.0" encoding="UTF-8"?〉
〈xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"〉
  〈xs:element name="TestSuite"〉
    〈xs:complexType〉
      〈xs:sequence〉
        〈xs:element name="TestCase" type="TestCase_Type" minOccurs="1" maxOccurs="unbounded"/〉
      〈/xs:sequence〉
    〈/xs:complexType〉
  〈/xs:element〉
  〈xs:complexType name="TestCase_Type"〉
    〈xs:sequence〉
      〈xs:element name="TestDescription" type="xs:string" minOccurs="1" maxOccurs="1"/〉
      〈xs:element name="TestStep" type="TestStep_Type" minOccurs="1" maxOccurs="unbounded"/〉
    〈/xs:sequence〉
  〈/xs:complexType〉
  〈xs:complexType name="TestStep_Type"〉
    〈xs:sequence〉
      〈xs:element name="TestInput" type="xs:string"/〉
      〈xs:element name="TestOutput" type="xs:string"/〉
      〈xs:element name="Test_Centence" type="Test_Centence_Choice"/〉
    〈/xs:sequence〉
  〈/xs:complexType〉
  〈xs:simpleType name="Test_Centence_Choice"〉
    〈xs:restriction base="xs:NMTOKEN"〉
      〈xs:enumeration value="Pass"/〉
      〈xs:enumeration value="Failed"/〉
      〈xs:enumeration value="Unknown"/〉
    〈/xs:restriction〉
  〈/xs:simpleType〉
〈/xs:schema〉
```

在此XSD(XML Schemas Definition)文件数据结构的基础之上,本文构建了高效的测试引擎,其剖面图如图6所示。该测试引擎采用Java编写,利用Dom解析的方式,采用Log4j2、Netty以及当前流行的SSM(Spring+SpringMVC+MyBatis)等框架技术,并结合Floodlight控制器中的StaticFlow Push模块进行自定义编程。该引擎读取测试例文件内容,并利用Dom解析树进行相应解析,构建测试步。整个工作流步骤,测试引擎主导整个测试过程,采取主动测试方法,将相应测试数据发送给被测试的OpenFlow交换机。本文所进行的实验中,具体被测环境由Mininet模拟底层网络,按照自定义编程的方式,分别由若干台交换机以及主机构成。测试引擎构建不同的测试数据分别发送到上述交换机,并通过主机间连通性测试验证相应流表数据是否正确或者通过前端展示Portal的方式进行验证。测试结果可在IPv6环境下通过前端Portal进行展示,如图7所示。

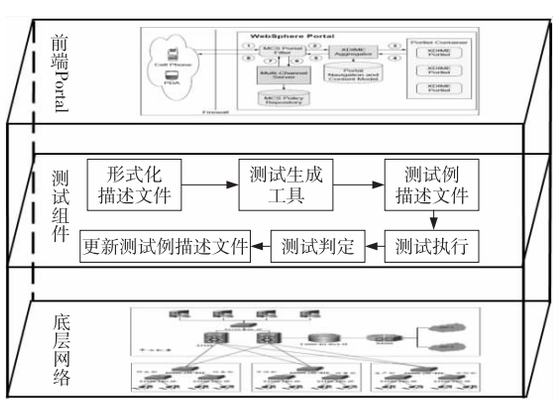


Figure 6 Profile photo of the test engine

图6 测试引擎剖面图

Figure 7 Test result

图 7 测试结果展现图

3.3 OpenFlow 协议测试结果分析

本次基于属性的测试中,更多关注 OpenFlow 协议中 IPv6 属性的相关信息,根据表 1 中的 9 个 IPv6 相关测试字段,按照组合测试的原则,进行穷举测试,理论上可以得到 511 条测试例。在保证测试效果的同时,如何减少时间与资源的相关投入,利用数据挖掘与人工智能的方式对系统进行模拟,生成相关的测试数据,同时考虑测试的重用性与回归性,是当前测试的相关研究范畴。本文按照等价类划分的原则对抽象测试例进行规约与化简,在考虑冗余性与覆盖性的前提下最终得到了 167 条测试例。上述测试例包含字段组合出现的测试例。其中各个字段以及相应测试例数量以及测试结果分布如图 8 与图 9 所示,根据我们的测试结果,OpenFlow 协议中对于常用的 IPv6 相关的字段如源地址、目的地址、流标签等的支持比较好,测试结果的正确性覆盖相对比较高,但是对于 IPv6 扩展头的支持相对薄弱。经过对测试失败的测试例进行分析,失败产生的原因基本分为 4 类,如表 2 所示。

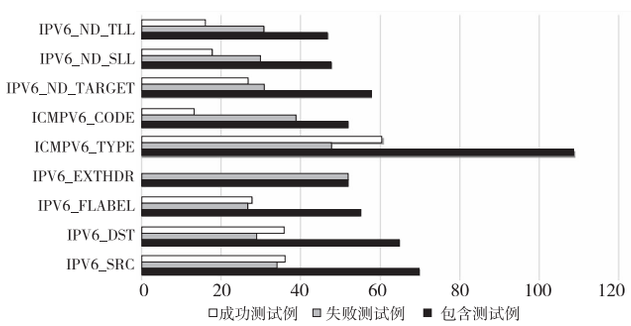


Figure 8 Testcase quantity and result distribution

图 8 测试例数量与结果分布图

其中对于经过仔细设计变更测试例,加入或更改其前置条件即可消除的错误,我们称为伪错误,上述第一类别即符合此种情况。对于包含 IPv6 扩展头字段的测试例,本次测试执行中共包含52条

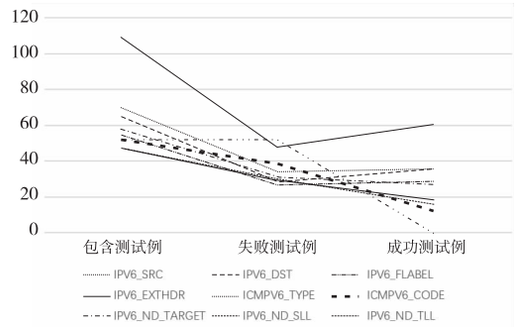


Figure 9 Testcase quantity and result distribution

图 9 测试例数量与结果分布图

Table 2 IPv6 testcase error classification and description in OpenFlow protocol

表 2 OpenFlow 协议 IPv6 测试错误类别及描述

测试错误类别	错误原因描述	错误测试例编号举例
1	前置约束条件违背	测试例 119
2	IPv6 扩展头字段影响	测试例 22 测试例 30 测试例 37
3	ICMPv6 类型与 ICMPv6 代码字段组合出现对 IPV6_ND_TARGET,IPV6_ND_SLL,IPV6_ND_TLL 等字段的影响	测试例 54 测试例 55 测试例 56 测试例 57
4	IPV6_ND_SLL 与 IPV6_ND_TLL 字段不能共存	测试例 60 测试例 89 测试例 110

可执行测试例,测试判定全部为失败,所以我们有理由相信被测实现没有实现该字段或者实现错误。对于 ICMPv6 类型与 ICMPv6 代码字段组合出现的情形,结合图 8 中的测试例分布情况,共包含 20 条这样的错误,且经过仔细设计测试例,上述错误无法消除,我们同样有理由相信,被测 OpenFlow 交换机在对于 ICMPv6 类型字段和 ICMPv6 代码字段与其它字段的组合实现时存在实现错误。第四类错误从逻辑上来说不能共存,本次测试结果也印证了此点。

综上,通过上述面向 IPv6 属性的 OpenFlow 协议测试,我们认为被测实现 OpenvSwitch(2.0.2 版本)基本支持 IPv6,但是不支持 IPv6 扩展头字段,同时在 ICMPv6 类型字段和 ICMPv6 代码字段与其它字段(如 IPV6_ND_TARGET,IPV6_ND_SLL,IPV6_ND_TLL 等)组合出现时存在实现错误。在涉及利用 OVS 2.0.2 版本实现 IPv6 网络的具体场景中,如与 OpenStack 结合构建云数据中心、与终端结合构建物联网以及其上的具体应用等场景中,需主动回避上述错误,以免对业务应用构成潜在的影响。

4 结束语

本文在对 OpenFlow 协议进行形式化建模的基础上,得到其抽象测试例,同时结合具体执行环境产生可执行测试例。我们构建了自动化的测试引擎,可以根据可执行测试例的相关描述信息自动展开测试,其测试结果可通过前端 Portal 在 IPv6 环境下集成展示。

随着 SDN 网络的发展,关于其分布式状态的位置问题逐渐演化,成为研究者关注的热点。通过对 OpenFlow 协议的测试,在了解其机制的基础上,针对不足,如何扩展其实现以支持相应状态转发,是下一步要考虑的问题。具体表现为如何扩展 OpenvSwitch 交换机,使其支持相关状态信息[19]提出抽象的描述方式,允许用户以状态机的方式描述 OpenFlow 应用,并利用其自定义的控制器将此描述转化为相应的流表信息并下发到交换机中。文献[20]提出 XFSM(eXtended Finite State Machines)的模型,同时扩充了 OpenFlow 协议,在 OpenFlow 交换机中加入新的状态数据结构,涉及到本地状态的操作,交换机依据流表匹配以及相应的状态信息决定转发行为,并更改相应的状态信息,不需要频繁地与控制器通信,减少了交互信息,提高了网络效率。上述文献为分布式状态的相应位置起到了指导作用。文献[21]提出 Pronane 语言及其相应的编译器,利用构造性证明的方法,将正则化网络策略转化为图形化内部表示,并在此基础之上结合网络拓扑生成抽象的 BGP(Border Gateway Protocol)配置,同时根据不同的网络设备映射为相应的实际 BGP 配置,其思路值得借鉴。下一步工作中,我们将从资源已知的情况下如何使效率最优的角度,考虑分布式状态的相应位置信息[22],必要的情况下,扩充 OpenFlow 的协议实现,以满足上述要求。

致谢 感谢内蒙古自治区“云计算与服务软件工程实验室”及内蒙古自治区“云计算与软件工程科技创新团队”项目的支持。

参考文献:

[1] McKeown N,Anderson T,Balakrishnan H,et al. OpenFlow:Enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review,2008,38(2):69-74.

[2] Jain S,Kumar A,Mandal S,et al. B4:Experience with a globally-deployed software defined WAN[J]. ACM SIGCOMM Computer Communication Review,2013,43(4):3-14.

[3] Su Jin-shu,Xu Cheng-cheng,Wang Xin. Prediction of network technology development and SIGCOMM 2016[J]. Communications of the CCF,2016,12(11):77-79. (in Chinese)

[4] Fogel A,Fung S,Pedrosa L,et al. A general approach to network configuration analysis[C]// Proc of NSDI,2015:469-483.

[5] Gember-Jacobson A,Viswanathan R,Akella A,et al. Fast control plane analysis using an abstract representation[C]// Proc of the 2016 Conference on ACM SIGCOMM,2016:300-313.

[6] Canini M,Kostic D,Rexford J,et al. Automating the testing of OpenFlow applications[C]// Proc of the 1st International Workshop on Rigorous Protocol Engineering (WRiPE),2011:1-6.

[7] Canini M,Venzano D,Peresini P,et al. A NICE way to test OpenFlow applications[C]// Proc of NSDI'12,2012:127-140.

[8] Kim H,Reich J,Gupta A,et al. Kinetic:Verifiable dynamic network control[C]// Proc of NSDI,2015:59-72.

[9] Rotsos C,Sarrar N,Uhlig S,et al. Oflops:An open framework for OpenFlow switch evaluation[C]// Proc of International Conference on Passive and Active Network Measurement,2012:85-95.

[10] OFTest-validating OpenFlow swtiches[EB/OL]. [2014-12-01]. http://www.projectfloodlight.org/oftest/.

[11] OpenFlow[EB/OL]. [2014-12-01]. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

[12] Kuzniar M,Peresini P,Canini M,et al. A soft way for Open Flow switch interoperability testing[C]// Proc of the 8th International Conference on Emerging Networking Experiments and Technologies,2012:265-276.

[13] Zhang Qing-qing. The research and optimization on the forwarding of OpenFlow based on IPv6 flow label[D]. Dalian:Dalian University of Technology,2015. (in Chinese)

[14] Li Yuan-ping,Li Hua,Zhao Jun-lan. The modeling research about OpenFlow protocol based TCPN[J]. Journal of Huazhong University of Science and Technology(Natural Science Edition),2016,44(S1):35-42. (in Chinese)

[15] Li Yuan-ping,Li Hua,Zhao Jun-lan. Research about FSM test sequence generation algorithm[J]. Computer Science,2016,43(S2):474-481. (in Chinese)

[16] Petrenko A,Yevtushenko N. Adaptive testing of nondeterministic systems with FSM[C]// Proc of 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering,2014:224-228.

[17] Floodlight OpenFlow controller[EB/OL]. [2014-12-01]. http://floodlight.openowhub.org/.

[18] Bertolino A. Software testing research:Achievements,challenges,dreams[C]// Proc of 2007 Future of Software Engineering,2007:85-103.

[19] Moshref M,Bhargava A,Gupta A,et al. Flow-level state transition as a new switch primitive for SDN[J]. ACM Sig-

comm Computer Communication Review,2014,44(6):61-66.

[20] Bianchi G,Bonola M,Capone A,et al. OpenState:Programming platform-independent stateful OpenFlow applications inside the switch[J]. ACM SIGCOMM Computer Communication Review,2014,44(2):44-51.

[21] Beckett R,Mahajan R,Millstein T,et al. Don't mind the gap:Bridging network-wide objectives and device-level configurations[C]// Proc of the 2016 ACM SIGCOMM,2016:328-341.

[22] Pfaff B,Pettit J,Koponen T,et al. The Design and implementation of Open vSwitch[C]// Proc of NSDI,2015:117-130.

附中文参考文献:

[3] 苏金树,徐成成,王鑫. SIGCOMM 2016 与网络技术发展预测[J]. 中国计算机学会通讯,2016,12(11):77-79.

[13] 张晴晴. 基于 IPv6 流标签的 OpenFlow 转发研究及优化[D]. 大连:大连理工大学,2015.

[14] 李元平,李华,赵俊岚. 基于时间着色 Petri 网的 OpenFlow 协议建模研究[J]. 华中科技大学学报(自然科学版),2016,44(S1):35-42.

[15] 李元平,李华,赵俊岚. 有限状态机模型测试序列生成算法研究[J]. 计算机科学,2016,43(S2):474-481.

作者简介:

李元平(1981-),男,内蒙古鄂尔多斯人,博士生,讲师,CCF 会员(73260M),研究方向为计算机网络和形式化方法。**E-mail**:liyp@imufe.edu.cn

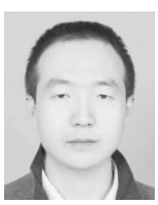
LI Yuan-ping, born in 1981, PhD candidate, lecturer, CCF member(73260M), his research interests include computer network, and formal method.

李华(1965-),女,山西忻州人,博士,教授,CCF 高级会员(E200006873S),研究方向为软件定义网络和形式化方法。**E-mail**:cslihua@imu.edu.cn

LI Hua, born in 1965, PhD, professor, CCF senior member(E200006873S), her research interests include computer network, and formal method.

赵俊岚(1963-),女,内蒙古托克托人,硕士,教授,CCF 会员(06594S),研究方向为 Web 应用技术研究,信息工程监理与信息技术。**E-mail**:zhaojunlan2007@163.com

ZHAO Jun-lan, born in 1963, MS, professor, CCF member(06594S), her research interests include Web research on the applied technology, information engineering supervision, and information technology.



阮宏玮(1981-),男,河北玉田人,博士生,讲师,CCF 会员(E200023319M),研究方向为软件定义网络和软件测试。**E-mail**:csrhw@imu.edu.cn

RUAN Hong-wei, born in 1981, PhD candidate, lecturer, CCF member(E200023319M), his research interests include software-defined networking, and software testing.