基于模型检测的OpenFlow多交换机数据包转发协议的分析与验证

朱 革[1,2] 曾国荪[1] 丁春玲[3] 王 伟[1]
(同济大学计算机科学及技术系 上海 201804)[1] (高效能服务器和存储技术国家重点实验室 济南 250101)[2]
(同济大学化学系 上海 201804)[3]

摘 要 OpenFlow协议是SDN网络中控制平面与数据转发平面之间进行交互的规范与标准,其正确性将直接影响到整个网络功能的实现。通过模型检测技术实现一种验证OpenFlow协议正确性的形式化方法。首先提取OpenFlow协议的核心子协议,即OpenFlow多交换机数据包转发协议作为验证的实例;然后运用协议行为自动机对该子协议进行形式化建模,并且通过时态逻辑描述协议需要进行验证的性质;最后给出算法验证协议模型是否满足给定的性质要求,以此检测OpenFlow协议是否存在正确性漏洞,以便对其进行修正。

关键词 OpenFlow协议,模型检测,协议行为自动机,时态逻辑
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.10.014

Analysis and Verification for OpenFlow Multi-switch Protocol Based on Model Checking

ZHU Ge[1,2] ZENG Guo-sun[1] DING Chun-ling[3] WANG Wei[1]
(Department of Computer Science and Technology, Tongji University, Shanghai 201804, China)[1]
(State Key Laboratory of High-end Server & Storage Technology, Jinan 250101, China)[2]
(Department of Chemistry, Tongji University, Shanghai 201804, China)[3]

Abstract In software defined networking (SDN), OpenFlow protocol is the communication standard between the control plane and forwarding plane. Its validity and rationality will influence the performance of the whole network. This paper focused on a formal method to verify the correctness of the OpenFlow protocol by model checking. Firstly, a key protocol called OpenFlow multi-switch protocol was proposed, which is regarded as an important example. Then, a basic model for this protocol is built by protocol action automata machine. The temporary logic is used to describe some major properties that this protocol has. A model checking algorithm is given to verify these properties. Finally, the special analysis and verification for OpenFlow multi-switch protocol are conducted so that the existing faults hidden in this protocol can be found and modified.

Keywords OpenFlow protocol, Model checking, Protocol action automata machine, Temporary logic

1 引言

经过30多年的高速发展,互联网已经从最初满足简单Internet服务的"尽力而为"网络,逐步发展成能够提供涵盖文本、语音、视频等多媒体业务的融合网络。网络功能的扩展与结构的复杂化,使得传统基于IP的简洁网络架构日益臃肿且越来越无法满足高效、灵活的业务承载需求。软件定义网络(Software Defined Networking, SDN)技术是一种新型的网络解决方案,其将网络的控制平面与数据平面分离的理念为网络的发展提供了新的可能。然而,集中化控制虽然给网络带来了诸多好处,但也使得网络暴露在更多的错误风险之中。作为SDN网络控制平面与数据平面之间的通信规范,OpenFlow协议定义了控制器与OpenFlow交换机间交互的数据包的结构与传输逻辑,其正确性与可靠性直接决定了SDN网络功能的准确实现。如何对OpenFlow协议乃至整个SDN网络的可靠性与安全性进行检测已经成为当前SDN领域的研究热点。

针对上述问题,Rotsos等人[1]提出了一个开源的通用框架OFLOPS来测试OpenFlow交换机,以评估OpenFlow交换机的性能与瓶颈。Canini[2]结合模型检测与符号化验证的概念,提出了一个能够自动快速地测试SDN应用的工具NICE(No bugs In Controller Execution),通过搜索遍历可能的执行路径检测控制器应用程序的准确性。建立在NICE的基础上,OpenFlow测试环境OFTEN[3]实现了实验模拟环境

到稿日期:2015-09-26 返修日期:2016-02-26 本文受863863项目(2009AA012201),国家自然科学基金项目(61272107,61202173,61103068),上海市优秀学科带头人计划项目(10XD1404400),华为创新研究计划项目(IRP-2013-12-03),高效能服务器和存储技术国家重点实验室开放基金项目(2014HSSA10)资助。

朱 革(1991-),男,硕士生,主要研究方向为SDN网络、OpenFlow协议、模型检测;**曾国荪**(1964-),男,博士,教授,博士生导师,主要研究方向为并行计算、可信软件设计、信息安全;**丁春玲**(1965-),女,硕士,高级工程师,主要研究方向为计算机硬件技术、计算机软件及计算机应用;**王 伟**(1976-),男,博士,副教授,主要研究方向为并行计算、分布式系统。

与真实网络设备之间的交互，将测试工具应用到了现实网络之中。石颖[4]应用 Pomela 语言对简化的 SDN 网络进行了形式化建模，并通过模型检测工具 SPIN 对网络控制器上的典型应用程序 pyswitch 进行了形式化验证。OFTest[5]是开源平台上一个基于 Python 的测试 OpenFlow 规范的框架和测试套件，已经成功测试了 OpenFlow 协议 1.0 与 1.1 版本的基础功能。Khurshid 等人[6]通过介于 SDN 网络控制层与数据传输层的 VeriFlow，达到了在极小的时延下对 SDN 网络中全网不变量的检测，实现了在不影响网络正常工作的情况下验证数据传输策略的一致性的目的。FlowChecker[7]将策略编码为二进制决策图，来检测交换机内部流表配置的准确性、不同交换机与控制器之间的一致性以及数据包的可达性与安全性。倪海铜[8]结合协议一致性测试标准，以模块化和树形管理模型为指导，设计并实现了协议的一致性测试集与测试系统，从而验证了 SDN 网络的实际执行是否与 OpenFlow 协议的标准相一致。

综上所述，前人的研究更多的是在实际或仿真的网络环境中，通过不同的检测工具模拟数据包的传输，以便对基于 OpenFlow 协议的 SDN 应用进行检测，验证 SDN 网络的实际功能与 OpenFlow 协议规范是否一致。但没有剖析具体的检测的方法，或是针对 OpenFlow 协议本身的逻辑过程进行分析验证；与此同时，仿真环境的不稳定性以及其与实际生产环境的差异性都将影响检测结果的可信度。随着 SDN 网络功能的不断扩展，OpenFlow 协议的复杂度也会随之增大，仿真或是人工分析与验证的难度和代价会变得巨大。本文基于模型检测技术，提出一种对 OpenFlow 协议的执行逻辑进行分析与验证的形式化方法。应用协议行为自动机对 OpenFlow 协议的通信主体及其行为进行建模，用时态逻辑公式刻画需要进行验证的协议性质，最后通过相应的模型检测算法验证 OpenFlow 协议是否满足用户描述的性质需求，以此达到通过形式化方法检测 OpenFlow 协议执行逻辑是否满足用户定义的正确性与可靠性要求的目的。

2 SDN 网络与 OpenFlow 协议

SDN(Software Defined Networking)又称为可编程网络，是指将网络设备控制平面从嵌入式节点中独立出来，部署到软件平台，由通过软件驱动的中央控制器集中化控制整个网络功能的网络架构。用户可以通过软件定义网络的拓扑、资源分配方案以及数据包的转发策略。OpenFlow 协议定义了控制器与底层 OpenFlow 交换机进行信息交互的 API 接口，控制器可以通过生成满足 OpenFlow 协议的流表项，并将其下发安装到相应的 OpenFlow 交换机来实现数据包的转发控制。

2.1 SDN 网络的组成和工作原理

SDN 网络主要是由若干控制器和由这些控制器进行控制管理的交换机集群组成。每一个交换机上有多个流表，而一个流表又由多条流表项构成。每一条流表项包含匹配字段(match fields)、优先级(priority)、计数器(counters)、操作(instructions)等字段。传输到 OpenFlow 交换机上的数据包通过与流表项上的匹配字段进行匹配来实现与特定流表项的关联，并根据该流表项的操作实现相应的转发策略。

数据包在 SDN 网络中的传输过程如下：OpenFlow 交换机接收到数据包后会对数据包包头进行解析，然后将解析得到的包头匹配域与第一个流表中的流表项按照优先级进行匹配。如果匹配成功，那么表项对应的 counters 会被更新，同时相应的 instructions 将被触发，其会对数据包进行指定的修改，将 action(action 是 OpenFlow 交换机可能对数据包执行的动作，包括将数据包发送到 OpenFlow 交换机的某一端口、修改数据包的包头或者是更改数据包的状态等)加入到数据包的 action set 中，或者是将数据包转发到下一个指定流表进行匹配；如果匹配失败，而流表中配置了通配匹配域的 table-miss 表项(优先级最低)，其默认指示 OpenFlow 交换机会将数据包发送给控制器(用户也可以自定义 table-miss 表项的 instructions)，由控制器制定转发策略。控制器会将转发策略以流表项的形式下发到传输路径上的各个 OpenFlow 交换机中，并将该数据包返还给其源交换机，最后数据包会按照新安装的转发策略进行传输。如果流表中没有配置 table-miss 表项，那么数据包会被直接丢弃。

2.2 OpenFlow 多交换机数据包转发协议

OpenFlow 协议是实现 SDN 网络的控制平面与数据平面之间通信的南向接口标准，从整个 OpenFlow 协议标准来看，可以将其分为两个部分：数据结构部分，定义了控制器与 OpenFlow 交换机之间交互信息的格式；逻辑交互部分，定义了交互信息的传输顺序与交互逻辑，其也将是本文分析与验证的重点。

结合实际应用场景，从应用较广的 OpenFlow specification 1.3.4 中提取了 OpenFlow 协议中的核心交互逻辑，并将其命名为 OpenFlow 多交换机数据包转发协议，以此作为 OpenFlow 协议验证测试的范例。

OpenFlow 多交换机数据包转发协议定义了数据包在由单控制器与多 OpenFlow 交换机组成的 SDN 网络中传输的规则与标准。其描述了 SDN 网络中从控制器与 OpenFlow 交换机建立连接并初始化流表，到数据包在 OpenFlow 交换机与控制器之间的交互，直至完成数据包传输的整个过程。

如图 1 所示，控制器在与 OpenFlow 交换机建立连接后，首先会根据用户定义的规则对交换机上的转发策略进行初始化，通常情况下是在交换机的流表中插入 table-miss 表项或者是根据需求插入用户自定义的初始表项。当终端 Host1 向终端 Host2 发送数据包，数据包到达 OpenFlow 交换机后会先与其上的流表项进行匹配，如果匹配成功，那么数据包将按照匹配的策略进行传输；如果匹配失败，OpenFlow 交换机会将数据包以 Packet-in 消息包的格式转发给控制器，由控制器制定相应的转发策略。控制器将设计的转发策略封装成 Flow-Mod 消息包下发到策略指定的数据包传输路径上的各个 OpenFlow 交换机，并以流表项的形式添加到特定的流表；同时，控制器会将交换机提交的数据包以 Packet-out 消息包的形式返还给源 OpenFlow 交换机。数据包到达源 OpenFlow 交换机上后与交换机上新安装的流表项进行匹配，经过流表项相关的一些列操作后被转发给下一跳交换机。数据包经过多交换机的传输，到达指定接收终端 Host2，完成交付。此外，用户也可以根据应用需求通过控制器主动修改指定 OpenFlow 交换机上的流表项，或是添加新的流表项以实现网络转发策略的自定义。

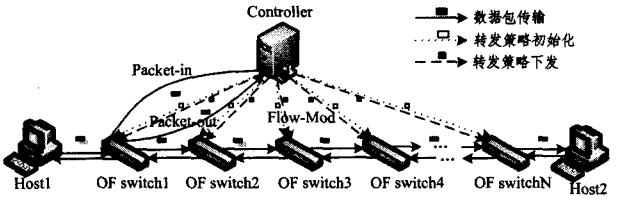

图 1　OpenFlow 多交换机数据包转发协议流程

2.3　OpenFlow 协议正确性要求

如需完成 SDN 网络的实时准确传输，实现网络的可靠性，OpenFlow 协议需要满足的最基本的必要条件（性质）如下。

（1）可达性：OpenFlow 交换机能够根据交换机上的路由转发策略，将数据包准确传输到指定的目的节点。

（2）无环性：保证 OpenFlow 网路中不存在无效的环路。

（3）一致性：根据应用的对象，可以将 OpenFlow 协议的一致性分为数据包一致性与数据流一致性。

数据包一致性是指数据包在传输过程中需要保证在各个交换机上匹配的转发策略一致，即如果数据包在第一个交换机上匹配成功的转发策略为 P_1，那么其在传输过程中都需要按照 P_1 进行传输。数据流一致性是指数据流中的所有数据包需要按照同一传输策略进行传输，即如果数据流中的第一个数据包按照转发策略 P_2 进行传输，那么这个数据流中的数据包都必须按照 P_2 进行传输。

3　模型检测技术

模型检测（model checking）是一种应用于有限状态系统的自动验证技术，它最早由 Clarke 和 Emerson 以及 Quielle 和 Sifakis 在 1981 年分别提出，主要通过显式状态搜索或隐式不动点计算来验证有穷状态并发系统的模态/命题性质。

模型检测的基本思想[10]是用状态迁移图（S）表示有限状态系统，用时序逻辑公式（F）描述系统的性质。这样“系统是否具有所期望的性质”就转化为数学问题“状态迁移图 S 是否满足公式 F ?”，用公式表示为 $S \mid = F$。对有限状态系统，这个问题是可判定的，可以用计算机程序在有限时间内自动遍历整个有限状态系统来确定时态逻辑公式的正确性。

3.1　OpenFlow 协议系统的建模

系统建模就是对需要检验的系统进行合理且无歧义的抽象，将其描述为有限状态迁移系统，这也是模型检测的第一步。在系统建模时，首先，需要获得系统的状态、系统的状态是对系统在某一固定时刻的具体参数数值的描述；其次，需要得到两个不同的状态之间的转换关系；最后，由不同的系统状态、状态之间的转换关系以及一系列约束条件组成整个有限状态迁移系统。本文结合实际需求提出协议行为自动机，如定义 1 所示。

定义 1（协议行为自动机）　$M=\langle S,S_0,R,AP,L\rangle$ 为一个五元组，其中，S 是系统状态的有限集合；$S_0 \in S$ 是系统的初始状态集合；$R \in S \times S$ 是状态之间的转换关系；AP 是所有原子命题与它的否定命题的集合；$L: S \rightarrow 2^{AP}$ 是一个标记函数，该函数把状态 $s \in S$ 映射为在 s 上为真的原子命题的集合，该集合是 AP 的一个子集。

可以通过协议行为自动机 M 来描述有限状态迁移系统，其可以看成是一个有标记的、有根的有向图。S 表示图的节点集合，S_0 是根节点的集合，R 表示边的集合，L 是节点上的标记。图 2 示出的协议行为自动机是一个简单的状态迁移系统，$S=\{s_0,s_1,s_2,s_3\}$，$S_0=\{s_0\}$，$AP=\{a,b,c,\neg a,\neg b,\neg c\}$，$R=\{\langle s_0,s_1\rangle,\langle s_0,s_2\rangle,\langle s_2,s_0\rangle,\langle s_1,s_3\rangle,\langle s_3,s_1\rangle,\langle s_3,s_2\rangle\}$，$L(s_0)=\{a\}$，$L(s_1)=\{a,b,c\}$，$L(s_2)=\{a,c\}$，$L(s_3)=\{\neg a,\neg b,\neg c\}$。

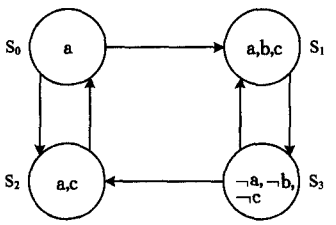

图 2　协议行为自动机

3.2　OpenFlow 协议性质的刻画

时态逻辑[11]CTL* 是非经典逻辑的一个分支，其公式由原子命题、布尔运算符、路径量词以及跟在其后的时态算子构成。路径量词有 **A** 和 **E**，**A** 表示总是（always）；**E** 表示存在（exit）。时态算子标记了路径上状态满足的性质的顺序，时态算子有 **X**（next）算子、**F**（future）算子、**G**（global）算子以及 **U**（until）算子。假设 p、q 为原子命题，那么：

Xp：路径上当前状态的下一个状态满足 p；

Fp：路径上当前状态后的某一状态满足 p；

Gp：路径上的所有状态满足 p；

p**U**q：路径上在满足 q 的状态之前的所有状态都满足 p；

AGp：所有路径上的所有状态都满足 p；

AFp：所有路径上总存在满足 p 的状态；

EFp：存在某一路径使得该路径上的某一状态满足 p；

时态逻辑公式有两种类型：状态公式（表示其在某一状态上为真）与路径公式（表示其在某一路径上为真）。其规则定义如定义 2、定义 3 所示。

定义 2（状态公式语法）　设 AP 为原子命题集合[12,13]，

如果 $p \in AP$，那么 p 是一个状态公式；

如果 f 和 g 为状态公式，那么 $\neg f$，$f \wedge g$ 和 $f \vee g$ 也是状态公式；

如果 f 是路径公式，那么 **E**f 和 **A**f 为状态公式。

定义 3（路径公式语法）　路径公式的语法为状态公式的语法加上下面两条规则[12,13]：

如果 f 是状态公式，那么 f 也一定是一个路径公式；

如果 f 和 g 是路径公式，那么 $\neg f$，$f \wedge g$，$f \vee g$，**X**f，**F**f，**G**f 和 f**U**g 也都是路径公式。

定义 4（时态逻辑的语义）　设 $p \in AP$ 为原子命题，f_1 和 f_2 为状态公式，g_1 和 g_2 为路径公式，$M=\langle S,S_0,R,AP,L\rangle$ 是一个协议行为自动机，$\pi=s_0 s_1 s_2 \cdots$ 是协议行为自动机 M 中由 s_0 开始的一条无限长的状态序列组成的路径，π^i 表示从状态 s_i 开始的路径 π 的后缀。$M,s \mid = f_1$ 表示状态公式 f_1 在 M 中的状态 s 上为真；如果 g 是一个路径公式，那么 $M,\pi \mid = g_1$ 表示路径公式 g_1 在路径 π 成立[12,13]。

（1）$M,s \mid = p$ iff $p \in L(s)$。

（2）$M,s \mid = \neg f_1$ iff $M,s \mid \neq \neg f_1$。

（3）$M,s \mid = f_1 \wedge f_2$ iff $M,s \mid = f_1$ 并且 $M,s \mid = f_2$。

（4）$M,s \mid = f_1 \vee f_2$ iff $M,s \mid = f_1$ 或者 $M,s \mid = f_2$。

(5)$M,s|=\mathbf{E}g_1$ iff 存在从状态 s 开始的一条路径 π 满足 $M,\pi|=g_1$。

(6)$M,s|=\mathbf{A}g_1$ iff 对于所有从状态 s 开始的路径 π 满足 $M,\pi|=g_1$。

(7)$M,\pi|=f_1$ iff 状态 s 是路径 π 的第一个状态，并且满足 $M,s|=f_1$。

(8)$M,\pi|=\mathbf{X}g_1$ iff $M,\pi^1|=g_1$。

(9)$M,\pi|=\mathbf{F}g_1$ iff 存在 $k\geqslant 0$ 满足 $M,\pi^k|=g_1$。

(10)$M,\pi|=\mathbf{G}g_1$ iff 对所有 $i\geqslant 0$ 满足 $M,\pi^i|=g_1$。

(11)$M,\pi|=g_1\mathbf{U}g_2$ iff 存在 $k\geqslant 0$ 满足 $M,\pi^k|=g_2$ 并且对于所有的 $0\leqslant j<k$，满足 $M,\pi^j|=g_1$。

(12)…

4 OpenFlow 多交换机数据包转发协议的具体分析验证

4.1 协议行为自动机模型

OpenFlow 多交换机数据包转发协议中，参与协议的主体集合为：{控制器 C，OpenFlow 交换机 S}。控制器和 OpenFlow 交换机的不同状态的组合形成了状态迁移系统中的状态集合，用一个圆圈表示，而协议的不同行为使得状态节点之间完成相互转换。

本文规定，主体之间通过安全信道传递消息 m，"?"代表接收，"!"代表发送；$S_i(i=1,2,3,\cdots)$表示控制器制定的数据包传输路径上的各个交换机；$P_i(i=1,2,3,\cdots)$表示 OpenFlow 交换机上 S_i 已有的转发策略，在网络启动时即为初始化策略；NP 表示 NewPolicy，即控制器制定并下发的新的转发策略。

为了便于分析理解，本文选取单控制器、双交换机的 SDN 网络进行建模，用协议行为自动机来刻画 OpenFlow 多交换机数据包转发协议，得到的协议行为自动机如图 3 所示。根据协议中的不同主体，OpenFlow 多交换机数据转发协议模型中有 7 个原子命题{$C\ wait$，$S_1\gg P_1$，$S_1\gg NP$，$S_1\ wait$，$S_2\gg P_2$，$S_2\gg NP$，$S_1\ wait$}。以状态节点 5 为例，状态节点 5 表示交换机 S_1 接收到数据包时的状态，当前控制器 C 处于等待接收数据包状态，交换机 S_1 与交换机 S_2 都已安装各自的初始化策略 P_1 与 P_2，S_1 处于数据包处理状态，S_2 处于等待接收状态。

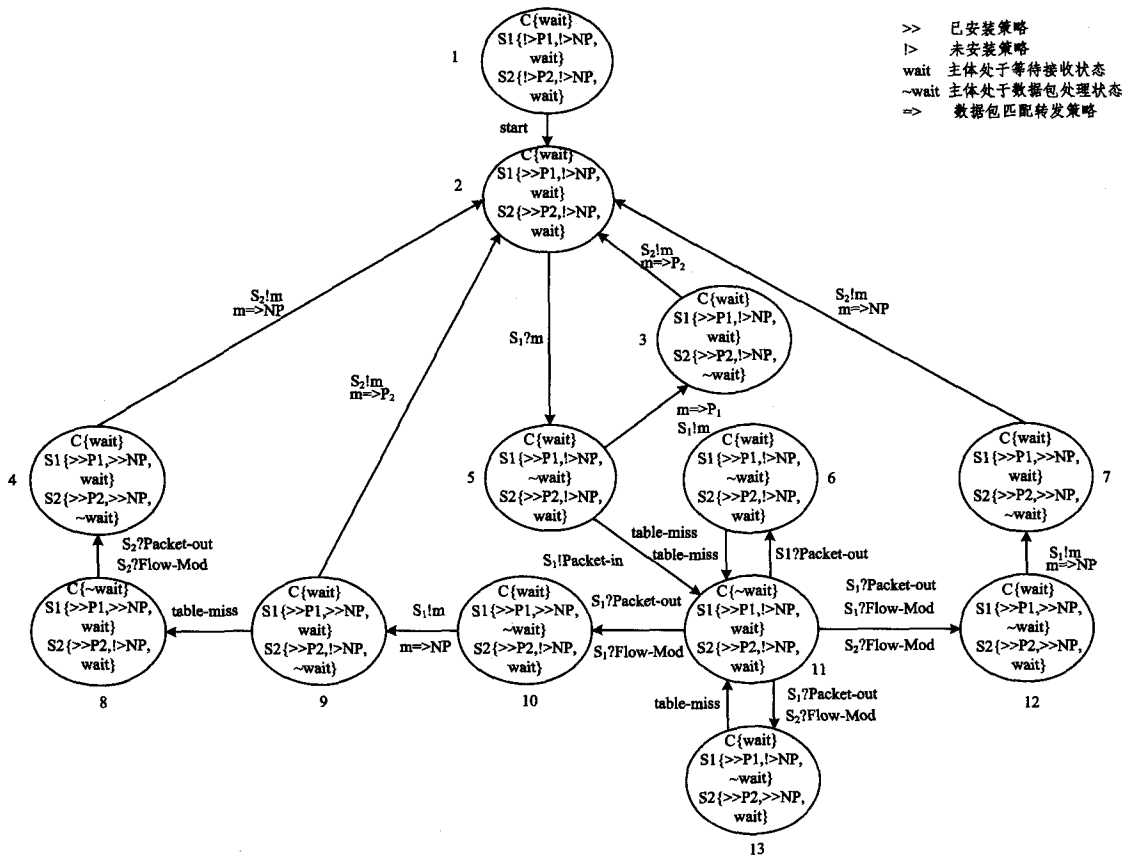

图 3 OpenFlow 多交换机数据转发协议模型

4.2 协议性质描述

本文通过时态逻辑将 OpenFlow 协议实现正确性需要满足的部分必要性质描述如下。

(1)SDN 网络中的 OpenFlow 交换机在接收到数据包后总能将数据包转发出去，即 OpenFlow 交换机 S_1 因为接收到数据包 m 状态变为 $S_1\sim wait\wedge S_2\ wait$ 后，总能将数据包转发出去而回到状态 $S_1\ wait\wedge S_2\ wait$。

$\mathbf{AG}(C\ wait\wedge S_1\sim wait\wedge S_2\ wait\rightarrow\mathbf{AF}(C\ wait\wedge S_1\ wait\wedge S_2\ wait))$

(2)控制器能正确响应交换机发来的请求，即 OpenFlow 交换机 S_1 或 S_2 向控制器提交 Packet-in 消息包后，最终总是能收到 C 制定的新策略 NP 和 Packet-out。

$\mathbf{AG}(C\sim wait\rightarrow\mathbf{AG}((S_1\gg NP\wedge S_1\sim wait)\vee(S_2\gg NP\wedge S_2\sim wait))$

(3)同一个数据包在不同的交换机上采用的转发策略一致，即如果 OpenFlow 交换机 S_1 上装有 NP，并且数据包根据 NP 策略进行转发，那么 S_2 也应该按照 NP 策略对 m 进行转发。

$\mathbf{AG}(S_1 \gg NP \rightarrow \mathbf{AF}\ S_2 \gg NP)$

(4)网络中不存在回路，即 OpenFlow 交换机 S_2 不会将数据包转发给它的前一跳交换机 S_1。

$\mathbf{AG}(S_1\ wait \wedge S_2 \sim wait \rightarrow \mathbf{AG} \neg (S_1 \sim wait \wedge S_2\ wait))$

(5)其它性质。类似 OpenFlow 协议正确性要求还可罗列出许多，在此不再赘述。

4.3 OpenFlow 协议的模型检测算法

模型检测算法就是在给定协议行为自动机 M，以及满足描述待验证的性质的时态逻辑公式 F 的情况下，找出 M 中所有满足 F 的状态的集合，即 $S_f=\{s_i \mid s_i$ 是 M 中的路径 r 上的状态，且 $M,s_i \mid = F\}$。在给定协议行为自动机时会给出初始状态的集合 S_0，根据不动点理论[14]，如果对于 $\forall s \in S_0$ 都有 $s \in S_f$，那么就认为系统满足所要验证的性质；否则不满足。

O. Grumberg 等[12]在他们的著作《Model Checking》中详细介绍了 Labelling 算法。其核心思想是对于给定的协议行为自动机 M 和待验证的时态逻辑公式 f，从 f 的最小子公式出发(常量命题 true，false，原子命题)，通过状态迁移图的遍历，标记满足该子公式的状态，然后将子公式向外扩展获得上一层子公式，并重复上述标记的步骤，直到获得所有被标记了 f 的状态的集合，也就是满足性质 f 的所有状态的集合。本文将其运用到下文给出的 CheckPCTL 算法中。

根据时态逻辑的语义，可以得到以下等价关系：$\mathbf{AF}f \equiv \neg\mathbf{EX}(\neg f)$，$\mathbf{EF}f \equiv \mathbf{E}[\text{true}\ \mathbf{U}\ f]$，$\mathbf{AG}\ f \equiv \neg\mathbf{EF}(\neg f)$，$\mathbf{AF}\ f \equiv \neg\mathbf{EG}(\neg f)$，$\mathbf{A}[f\ \mathbf{U} g] \equiv \neg\mathbf{E}[\neg g\ \mathbf{U}(\neg f \wedge \neg g)] \wedge \neg\mathbf{EG}\neg g \cdots$

因而，可以用$\{\neg, \vee, \mathbf{EX}, \mathbf{EU}, \mathbf{EG}\}$5 种 CTL 基本算子对协议行为自动机中的状态进行标记。

基于以上概念，给出 $\mathbf{EX}f$，$\mathbf{E}(f\ \mathbf{U}\ g)$，$\mathbf{EG}\ f$ 的标记过程，以及 CheckPCTL 算法的核心过程。

算法 1 CheckEX(f)

```
输入:T:={s|f∈label(s)}
输出:R:={s|EX(f)∈label(s)}
CheckEX( f )
    T:={s|f(label(s)};
    while T≠∅ do
        choose s∈T;
          T:=T\{s};
        for all t such that R(t,s) do
          if EX f∉label(t) then
            label(t):=label(t)∪{EX f};
    return R:={s|EX(f)∈label(s)};
```

算法 2 CheckEU(f,g)

```
输入:T:={s|g∈label(s)}
输出:R:={s|EU(f,g)∈label(s)}
CheckEU(f,g )
    T:={s|g∈label(s)};
    for all s∈T
    do label(s):=label(s)∪{E[f U g]};
    while T≠∅ do
        choose s∈T;
        T:=T\{s};
        for all t such that R(t,s) do
            if E[f U g]∉label(t)
              and f∈label(t) then
                label(t):=label(t)∪{E[f U g]};
                T:=T∪{t};
    return R:={s|EU(f,g)∈label(s)};
```

算子 EG 的标记过程要复杂许多，有一种基于将有向图分解为非平凡强连通分量(Nontrivial Strongly Connected Component，NSCC)的标记过程更为有效。强连通分量(SCC) C 是指一个有向图的最大子图，图内的任意两个节点都可以通过图内的边互相连通；而非平凡是指该子图含有不只一个节点或者是含有一个带反身边的节点。

假设协议行为自动机 $M'=\langle S', S_0', R', AP', L'\rangle$ 为去掉协议行为自动机 M 中所有不能使 CTL 公式 f 为真的状态得到的，因而 $S'=\{s \in S \mid M,s \mid = f\}$，$R'=R|_{S' \times S'}$，$L'=L|_{S'}$. $M,s \mid = \mathbf{EG}f$ 当且仅当 s 满足以下两个条件：

(1)$s \in S'$；

(2)M' 中存在一条从状态节点 s 到有向图(S',R')的非平凡强连通分量中的某一个状态节点 t 的路径。

算法 3 CheckEG(f)

```
输入:T:=∪c∈SCC{s|s∈C}
输出:R:={s|EG(f)∈label(s)}
CheckEG(f)
    S':={s|f∈label(s)};
    SCC:={C|C是S'的非平凡SCC};
T:= ∪c∈SCC{s|s∈C};
for all s∈T do label(s):=label(s)∪{EG f};
    while T≠∅ do
        choose s∈T ;
            T:=T\{s};
        for all t such that t∈S' and R(t,s) do
            if EG f∉label(t) then
                label(t):=label(t)∪{EG f };
                T:=T∪{t};
    return R:={s|EG(f)∈label(s)};
```

综合以上 3 个标记算法，给出 CheckPCTL 算法，其主要思想是根据性质 f 的类型对协议行为自动机中的节点进行标记，核心执行过程如下。

算法 4 CheckPCTL(f)

```
输入:S//协议行为自动机中的状态节点集合
输出: R:={s|f∈label(s)}
CheckPCTL(f)
    case:
    f is true:for all s∈S do
                if true∉label(s) then
                label(s):=label(s)∪{true};
    f is atomic:for all s such that s∈S
                  and f∈L(s) do
                if atomic∉label(s) then
                label(s):=label(s)∪{atomic};
    f is ¬g :for all s such that s∈S
                and g∉L(s) do
                if ¬g∉label(s) then
                label(s):=label(s)∪{¬g};
    f is g1 ∨ g2:for all s such that s∈S
                  and (g1 or g2)∈L(s) do
                if g1 ∨ g2∉L(s) then
                label(s):=label(s)∪{g1 ∨ g2};
f is EXg:CheckPCTL(g);
```

```
        CheckEX(g);
f is E(g1 U g2):CheckPCTL(g1);
                CheckPCTL(g2);
                CheckEU(g1,g2);
f is EG g:CheckPCTL(g);
        CheckEG(g);
```

5 OpenFlow 协议验证结果

参照 OpenFlow 多交换机数据包转发协议模型，本文通过 CheckPCTL 算法对 4.2 节中描述的协议性质进行逐一验证。这里需要注意的是，在本文建立的协议行为自动机模型中，节点 11 同节点 6 以及节点 13 之间分别存在一个循环，这是因为同是控制器发出的 Packet-out 数据包可能先于转发策略 Flow-Mod 到达 OpenFlow 交换机，但是在实际情况中这样的回路的循环次数是有限的，因为转发策略终会到达交换机，一旦交换机上安装了新策略，循环就会结束。故而为了保证验证的准确性，本文在验证过程中选择忽略节点 6 与节点 13 这两个特殊节点。

以性质 3 的验证为例：$\mathbf{AG}(S_1 \gg NP \rightarrow \mathbf{AF}\ S_2 \gg NP)$ 等价于时态逻辑公式 $\neg\mathbf{EF}(S_1 \gg NP \wedge \mathbf{EG}(S_2 ! > NP))$，这里的 $\mathbf{EF}\ f$ 可以认为是 $\mathbf{E}(\text{true}\ \mathbf{U}\ f)$ 的简化形式。验证的步骤如下。

Step1　从初始状态节点 1 出发，用 $S(g)$ 表示满足子公式 g 的状态的集合。$S(S_1 \gg NP)=\{4,7,8,9,10,12\}$；$S(S_2 ! > NP)=\{1,2,3,5,8,9,10,11\}$。

Step2　为了计算 $S(\mathbf{EG}(S_2 ! > NP))$，需要先获得 $S'=S(S_2 ! > NP)$ 的非平凡强连通分量。$SCC=\{2,3,5,9,10,11\}$。根据算法 CheckEG 得 $S(\mathbf{EG}(S_2 ! > NP))=\{1,2,3,5,9,10,11\}$。

Step3　计算 $S(S_1 \gg NP \wedge \mathbf{EG}(S_2 ! > NP))=\{9,10\}$。

Step4　从 $T=S(S_1 \gg NP \wedge \mathbf{EG}(S_2 ! > NP))$ 计算 $S(\mathbf{EF}(S_1 \gg NP \wedge \mathbf{EG}(S_2 ! > NP)))$，得到 $S(\mathbf{EF}(S_1 \gg NP \wedge \mathbf{EG}(S_2 ! > NP)))=\{1,2,3,4,5,7,8,9,10,11,12\}$。

Step5　最后可知 $S(\neg\mathbf{EF}(S_1 \gg NP \wedge \mathbf{EG}(S_2 ! > NP)))=\emptyset$。

因为初始状态 1 并不包含在空集中，所以最终得到行为自动机描述的状态迁移系统并不满足协议性质 $\mathbf{AG}(S_1 \gg NP \rightarrow \mathbf{AF}\ S_2 \gg NP)$，即 OpenFlow 协议不满足协议的一致性需求。根据 SDN 网络的特点进行分析，这是网络的传输存在时延造成的。SDN 网络中，由于控制平面与数据平面的分离，网络转发策略的制定与下发往往需要经过控制器与交换机之间的通信，这就为 SDN 网络引入了额外的传输时延。如图 4 所示，假设 Switch1 按照新策略 NP 来转发数据包，当其将数据包转发给 Switch2 时，如果控制器下发的 NP 由于网络传输时延还没有到达 Switch2，且 Switch2 上原有的策略 P_2 也匹配数据包，那么就会出现数据包在 Switch1 上按照 NP 进行转发，在 Switch2 上按照 P_2 进行转发的转发策略不一致的现象。

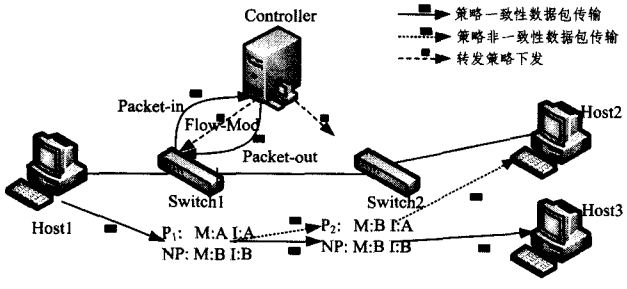

图 4　传输时延导致的转发策略不一致示意图

同理，本文通过上述算法验证得到 $\mathbf{AG}(C\ wait \wedge S_1 \sim wait \wedge S_2\ wait \rightarrow \mathbf{AF}(C\ wait \wedge S_1\ wait \wedge S_2\ wait))$，$\mathbf{AG}(C \sim wait \rightarrow \mathbf{AG}((S_1 \gg NP \wedge S_1 \sim wait) \vee (S_2 \gg NP \wedge S_2 \sim wait))$ 与 $\mathbf{AG}(S_1\ wait \wedge S_2 \sim wait \rightarrow \mathbf{AG}\neg(S_1 \sim wait \wedge S_2\ wait))$，这 3 条 OpenFlow 协议正确性必要条件在用协议行为自动机表示的 OpenFlow 多交换机数据包转发模型中均获得满足。由此说明 SDN 网络中，控制器总是能完成对交换机发来的请求的响应，并管理交换机集群完成数据包的传输，且网络满足无环性的基本网络性质要求。

结束语　本文基于对软件定义网络(SDN)架构的研究，针对 SDN 网络中控制器与交换机进行数据交互的规范——OpenFlow 协议，提出了一种通过模型检测技术对用户描述的协议正确性要求进行分析与验证的方法。首先提取 OpenFlow 协议的核心子协议——OpenFlow 多交换机数据包转发协议；然后提出协议行为自动机的概念，通过协议行为自动机对 OpenFlow 协议的核心逻辑进行了形式化建模；同时，运用时态逻辑对协议的正确性需要满足的部分必要条件(性质)进行了形式化描述；最后给出了一种基于图遍历标记思想的模型检测算法 CheckPCTL，并通过算法对 OpenFlow 协议是否满足上述描述的协议性质进行了验证。

与其他基于实验仿真模拟的协议验证方法相比，本文提出的 OpenFlow 协议形式化验证方法通过模型检测的思想对用户描述的协议性质进行了准确验证，不仅降低了搭建实验模拟环境带来的验证开销，同时规避了仿真环境的不稳定性以及其与实际生产环境的差异性对验证结果带来的影响。运用本文提出的验证方法，用户可以对自定义的 OpenFlow 网络环境进行建模，结合实际网络环境需求通过时态逻辑描述当前 OpenFlow 网络需要满足的基本正确性要求，在 OpenFlow 网络的设计阶段即可实现对网络需要满足的基本性质的验证，降低了 OpenFlow 网络设计与部署过程中的时间与成本开销。

从最初的 OpenFlow specification 1.0 到目前最新的 OpenFlow specification 1.5，OpenFlow 协议还在不断地更新完善，其中存在的大部分漏洞都获得了很好的解决，我们将不断跟进对新版本 OpenFlow 协议的验证工作。同时，本文提出的方法目前只能结合网络模型对用户描述的协议性质进行验证，如何建立系统且完备的 OpenFlow 协议正确性需求库，并将其运用到完整的 OpenFlow 协议进行检测验证将是我们下一步研究工作的关注点。

参 考 文 献

[1] Rotsos C, Sarrar N, Uhlig S, et al. OFLOPS: An open framework for OpenFlow switch evaluation[M]//Passive and Active Measurement. Springer Berlin Heidelberg, 2012: 85-95

[2] Canini M, Venzano D, Peresini P, et al. A NICE Way to Test OpenFlow Applications[C]//NSDI. 2012, 12: 127-140

[3] Kuźniar M, Canini M, Kostić D. OFTEN testing OpenFlow networks[C]//2012 European Workshop on Software Defined Networking (EWSDN). IEEE, 2012: 54-60

[4] Shi Y. The formal verification of openfiow protocol using SPIN [J]. Network & Computer Security, 2014, 3(9): 38-43(in Chinese)
石颖. 利用 SPIN 实现对 OpenFlow 协议的形式化验证[J]. 计算机安全, 2014, 3(9): 38-43

[5] OFTest[OL]. http://www.projectfloodlight.org/oftest

[6] Khurshid A, Zhou W, Caesar M, et al. Veriflow: verifying network-wide invariants in real time[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 467-472

[7] Al-Shaer E, Al-Haj S. FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures[C]// Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration. ACM, 2010: 37-44

[8] Ni H T. Design and Implementation of the Conformance Test System for OpenFlow Protocol[D]. Beijing: Beijing Jiaotong University, 2015(in Chinese)
倪海铜. OpenFlow 协议一致性测试系统设计与实现[D]. 北京: 北京交通大学, 2015

[9] OpenFlow Switch Consortium. OpenFlow Switch Specification Version1. 3. 4 (2014) [OL]. https://www. opennetworking. org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1. 3. 4. pdf

[10] Lin H M, Zhang W H. Model checking: Theories, techniques and applications[J]. Chinese Journal of Electronics, 2002, 30(S1): 1907-1912(in Chinese)
林惠民, 张文辉. 模型检测: 理论, 方法与应用[J]. 电子学报, 2002, 30(12): 1907-1912

[11] Emerson E A. Temporal and modal logic[M]. MIT Press Cambridge, 1995

[12] Clarke E M, Grumberg O, Peled D. Model checking[M]. MIT Press, 1999

[13] Zhang Y Z. Research on symbolic model-checking algorithm [D]. Jilin: Jilin University, 2009(in Chinese)
张衍志. 符号化模型检测算法的研究[D]. 吉林: 吉林大学, 2009

[14] Granas A, Dugundji J. Fixed point theory [M]. Springer Science & Business Media, 2013

(上接第 65 页)

据集上运行的 iForest 算法的结果可以看作期望值，而在流环境下，每个数据集的 AUC 值却相对较低，这说明在机器学习领域中对整体样本分布的认知程度直接影响着探测器的性能。

其次，从上述 δ 参数在不同值情况下的 AUC 值分析可知，对于不同的数据集，由于样本的进化特征不同，不可能有统一的 δ 值用来指导集成探测器的更新，因此实际应用中需要根据历史数据及其在线学习理论，掌握针对具体应用的流数据产生机理，学习其进化特征。

结束语 本文基于异常数据的本质特征和在线集成学习理论，提出了一种基于隔离森林的在线流数据异常检测方法。4 个标准数据集的实验结果表明，提出的方法能获得较好结果。但需要进一步对该算法中的一些关键参数(如窗口尺寸、集成规模等参数的设置)进行研究以提高算法的性能。其次，在构建隔离树时，属性及其属性值的随机选择没有考虑样本属性值的分布情况，后续将进一步对其展开研究。

参 考 文 献

[1] Gupta M, Gao J, Aggarwal C C, et al. Outlier Detection for Temporal Data: A Survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(9): 2250-2267

[2] Zheng Li-ming, Zou Peng, Jia Yan. How to Extract and Train the Classifier in Traffic Anomaly Detection System[J]. Chinese Journal of Computers, 2012, 35(4): 719-730(in Chinese)
郑黎明, 邹鹏, 贾焰. 网络流量异常检测中分类器的提取与训练方法研究[J]. 计算机学报, 2012, 35(4): 719-730

[3] Shen M X, Liu D R, Shann S H. Outlier detection from vehicle trajectories to discover roaming events[J]. Information Sciences, 2015, 294: 242-254

[4] Lu You, Li Wei, Luo Jun-zhou, et al. A Network User's Abnormal Behavior Detection Approach Based on Selective Collaborative Learning[J]. Chinese Journal of Computers, 2014, 37(1): 28-41(in Chinese)
陆悠, 李伟, 罗军舟, 等, 一种基于选择性协同学习的网络用户异常行为检测方法[J]. 计算机学报, 2014, 37(1): 28-41

[5] Srinivasan U. Anomalies Detection in Healthcare Services[J]. It Professional, 2014, 16(6): 12-15

[6] Zhou Dong-hua, Wei Mu-heng, Si Xiao-sheng. A Survey on Anomaly Detection, Life Prediction and Maintenance Decision for Industrial Processes[J]. Acta Automatica Sinica, 2013, 39(6): 711-722(in Chinese)
周东华, 魏慕恒, 司小胜. 工业过程异常检测、寿命预测与维修决策的研究进展[J]. 自动化学报, 2013, 39(6): 711-722

[7] Tan S C, Ting K M, Liu T F. Fast anomaly detection for streaming data[C]// Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence. AAAI Press, 2011, 2: 1511-1516

[8] Widmer G, Kubat M. Learning in the presence of concept drift and hidden contexts[J]. Machine Learning, 1996, 23(1): 69-101

[9] Li D, Liu S L, Zhang H L. A negative selection algorithm with online adaptive learning under small samples for anomaly detection[J]. Neurocomputing, 2015, 149: 515-525

[10] Liu F T, Ting K M, Zhou Z H. Isolation-Based Anomaly Detection[J]. ACM Transactions on Knowledge Discovery from Data, 2012, 6(1): 1-39

[11] Daneshpazhouh A, Sami A. Entropybased outlier detection using semi-supervised approach with few positive examples[J]. Pattern Recognition Letters, 2014, 49: 77-84

[12] Quinn J A, Sugiyama M. A least-squares approach to anomaly detection in static and sequential data[J]. Pattern Recognition Letters, 2014, 40: 36-40

[13] He H, Chen S, Li K, et al. Incremental learning from stream data[J]. IEEE Transactions on Neural Networks and Learning Systems, 2011, 22(12): 1901-1914

[14] Ando S, Thanomphongphan T, Seki Y, et al. Ensemble anomaly detection from multi-resolution trajectory features [J]. Data Mining and Knowledge Discovery, 2015, 29(1): 39-83

[15] Minku L L, Yao X. DDD: A New Ensemble Approach for Dealing with Concept Drift[J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(4): 619-633

[16] Chang W C, Cho C W. Online Boosting for Vehicle Detection [J]. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, 2010, 40(3): 892-902

[17] UCI Machine Learning Repository [OL]. http://archive. ics. uci. edu/ml/datasets. html

[18] Hand D J, Tillr J. A simple generalisation of the area under the ROC curve for multiple class classification problems[J]. Machine Learning, 2001, 45(2): 171-186