拟态路由器 BGP 代理的设计实现与形式化验证

张进[1]，葛强[1,2]，徐伟海[1,2]，江逸茗[3]，马海龙[3]，于洪涛[3]

（1. 紫金山实验室内生安全研究中心，江苏 南京 211100；2. 东南大学网络空间安全学院，江苏 南京 211100；
3. 信息工程大学信息技术研究所，河南 郑州 450000）

摘 要：为确保拟态路由器中协议代理等“拟态括号”关键组件的安全性和功能正确性，设计实现了边界网关协议（BGP）代理，并采用形式化方法对 BGP 代理的安全性和功能正确性进行了验证。BGP 代理通过监听邻居路由器与主执行体之间的 BGP 会话报文，模拟邻居路由器与从执行体展开 BGP 会话，实现各执行体的 BGP 状态一致。采用 VeriFast 定理证明器，编写基于分离逻辑的形式化规约，证明程序不会出现空指针引用等内存安全问题，并验证了 BGP 代理各功能模块实现的高级属性符合功能规约。BGP 代理实现与验证代码量之比约为 1.8:1，程序设计实现和程序验证所耗费的时间之比约为 1:3。经过形式化验证的 BGP 代理处理 100 000 条 BGP 路由所花费的时间为 0.16 s，约为未经过验证的 BGP 代理的 7 倍。研究工作为应用形式化方法证明拟态防御设备与系统中关键组件的安全性和功能正确性提供了参考。

关键词：拟态防御；边界网关协议；路由器；形式化验证

中图分类号：TP393

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2023065

Design, implementation and formal verification of BGP proxy for mimic router

ZHANG Jin[1], GE Qiang[1,2], XU Weihai[1,2], JIANG Yiming[3], MA Hailong[3], YU Hongtao[3]

1. The Endogenous Security Research Centre, Purple Mountain Laboratories, Nanjing 211100, China
2. School of Cyber Science and Engineering, Southeast University, Nanjing 211100, China
3. Institute of Information Technology, Information Engineering University, Zhengzhou 450000, China

Abstract: To ensure the safety and correctness of the critical ‘mimic bracket’ components such as protocol proxies of mimic routers, a BGP (border gateway protocol) proxy was designed and implemented, and formal methods were applied to verify the safety and correctness of the BGP proxy. The BGP packets communicated between the peer routers and the master actor were monitored by the BGP proxy. The BGP sessions with the slave actors on behalf of peer routers were established, ensuring the consistency of the BGP protocol states for all actors. The formal specification of the BGP proxy was written based on separation logic. The VeriFast theorem prover was used to prove that the program had no memory safety problems such as null pointer reference. Furthermore, the formal verification of high-level attributes of each module in BGP proxy was also conducted to strictly ensure that the implementation met the specification. The implementation to proof code ratio of BGP proxy is about 1.8:1, and the implementation to proof labor hour ratio is about 1:3. The formally verified BGP proxy consume 0.16 seconds to process 100 000 BGP routes, which is about 7 times as long as the unverified one. Works done provide a reference for applying formal methods to verify the safety and correctness of critical components in mimic defense equipment and systems.

Keywords: mimic defense, BGP, router, formal verification

收稿日期：2022–12–06；**修回日期**：2023–02–27

基金项目：国家重点研发计划基金资助项目（No.2022YFB2901400）

Foundation Item: The National Key Research and Development Program of China (No.2022YFB2901400)

0 引言

近年来，路由器安全问题层出不穷[1-2]。攻击者可通过向受影响的设备发送恶意请求利用漏洞，从而在操作系统上执行任意命令。从功能划分的角度，路由器可分为管理面、控制面和数据转发面。相对于数据转发面，管理面和控制面功能更复杂，代码量更大，攻击面更大。针对路由器自身的安全问题，马海龙等[3]提出了一种基于拟态防御动态异构冗余（DHR, dynamic heterogeneous redundancy）架构[4]的拟态路由器（DHR2, DHR based router），如图1所示。拟态路由器采用多个功能等价，但实现方式上异构的管理面和控制面执行体、多个异构执行体并行工作，输出结果相互比对，达到发现潜在攻击的目的。

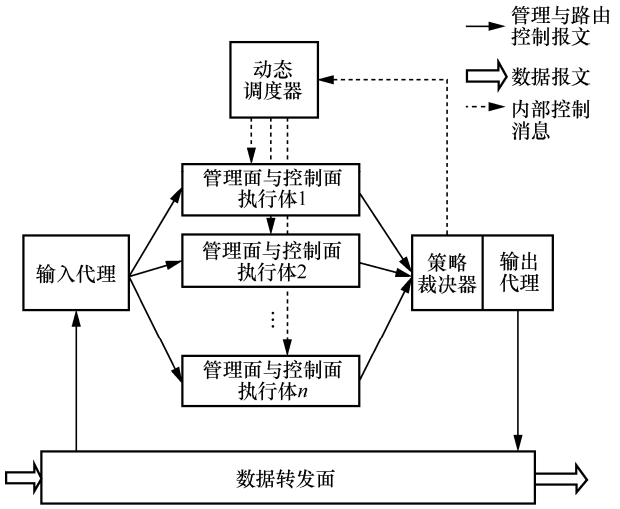


图1 基于DHR的拟态路由器

图1所示的拟态路由器中，除了功能等价的异构执行体外，其他关键组件还有协议代理（包括输入代理和输出代理）、策略裁决器、动态调度器等，这些组件在DHR架构的防护能力之外，也被称为拟态括号组件[3]。作为拟态路由器中异构执行体与外界交互的桥梁，以及执行体调度控制的中枢，拟态括号组件的安全性是拟态路由器整体安全性的前提和基础。虽然与执行体相比，拟态括号组件的攻击面极小，但是考虑到其在系统中的重要地位，仍然有必要采用严格的方法尽力提升拟态括号组件的安全性。

本文通过形式化验证方法来提高拟态括号组件的安全性，设计实现了拟态路由器边界网关协议（BGP, border gateway protocol）代理，并选择了一种基于分离逻辑和符号执行的形式化验证技术，证明了所实现的BGP代理的内存安全性和功能正确性（实现符合规约）。BGP代理实现代码与验证代码量之比约为1.8:1，程序设计实现和程序验证的时间之比约为1:3。经过形式化验证的BGP代理处理100 000条BGP路由所花费的时间约为未经过验证的BGP代理的7倍。总体而言，虽然形式化验证需要花费一定的人力成本，并对程序性能造成一定的负面影响，但是总体代价在可接受的范围之内，从提升系统整体安全性的角度而言，所付出的代价是值得的。

1 相关工作

1.1 网络功能模块的形式化验证

网络功能模块的形式化验证是指采用形式化方法证明网络功能模块具备一定的性质（如有界执行时间、内存安全性、功能正确性等），其本质是传统的软件形式化验证理论与方法在网络领域的具体应用。

Dobrescu等[5]针对Click编写的软件数据平面，提出了一种形式化验证方法，该方法基于组合符号执行的原理，充分利用软件数据平面的流水线结构、静态内存分配等特点，减少符号执行的搜索路径数量，提高验证速度。上述方法的缺点是对数据结构的处理较为简单，既不验证数据结构本身，也不验证程序是否正确使用了它们，因此，其只能验证程序的低级属性，包括崩溃避免、有界执行和报文过滤，但无法验证程序的高级功能属性。

Zhang等[6]针对Click编写的网络功能模块，提出了支持高级属性验证的工具Gravel。Gravel首先尝试对程序进行符号执行，而对于符号执行无法完全验证的有状态元素，Gravel使用可满足性模理论（SMT, satisfiability modulo theory）编码[7]的抽象数据类型代替元素状态，利用一阶求解器进行求解。上述方法导致Gravel自动化程度低且可扩展性差，没有深厚的形式化知识储备的开发者无法使用Gravel对所开发的模块的验证。由于Gravel针对Click编写的网络功能模块，且扩展难度较大，因此无法直接应用于拟态路由器BGP代理的自动化验证。

Zaostrovnykh等[8]提出了一种可形式化验证的网络地址转换（NAT, network address translator）——

VigNAT。VigNAT 将程序分为无状态和有状态的两类组件，前者通过符号执行完成验证；后者通过人工辅助的定理证明进行验证，定理证明过程的难度和工作量较大，证明代码和实现代码之比达 10:1。VigNAT 不仅可验证崩溃避免、内存安全等程序的低级属性，同时也能验证 VigNAT 在语义层面符合 RFC 3022 规范。

在 VigNAT 的基础上，Zaostrovnykh 等[9]提出了网络功能模块的自动化验证工具 Vigor。相对于 VigNAT，Vigor 采用更高层次的抽象，将无法进行符号执行的程序组件封装在专门的库（libVig）中，并由形式化方法专家通过定理证明的方式对 libVig 进行验证。此外，Vigor 还可验证网络功能模块所调用的数据平面开发工具包（DPDK, data plane development kit）、网卡驱动等底层软件栈。当前，Vigor 可对文献[9]中给出的 5 个简单的网络功能模块以及类似模块进行自动化验证，但如需验证更多功能模块，则需要对 libVig 作进一步扩展并实现相应接口，完成上述工作需要具备深厚的形式化方法知识，并深入理解 libVig。由于拟态路由器 BGP 代理和文献[9]所验证的网络功能差异较大，因此无法直接使用 Vigor 对 BGP 代理进行自动化验证。

文献[5-6,8-9]均为针对 C 语言实现的网络功能。近年来，有学者提出了针对 P4 语言实现的网络功能的形式化验证方法，如 P4V[10]、bf4[11]、Aquila[12]等。本文采用 C 语言实现 BGP 代理。

如上文所述，现有的网络功能模块形式化验证工具[6,9]无法直接应用于 BGP 代理的验证。Gravel[6]为针对 Click 编写的网络功能模块，与其相比，Vigor[9]的验证对象更具一般性。因此，本文参考 Vigor，人工编写分离逻辑的形式化规约，使用 VeriFast[13]完成程序和规约到证明条件的转换，并采用自动定理证明器 Z3[14]完成定理的证明。相比于扩展 Vigor 使之能够支持 BGP 代理的验证，本文采用的技术方法工作量更小，开发自由度更高，且验证的维度也更广。

1.2　拟态防御

网络空间内生安全拟态防御理论提出至今，已有多项将拟态防御的思想应用于具体网络设备与系统研制的成功实践[15]。马海龙等[3]提出的拟态路由器[3]采用多个异构的控制面和管理面执行体，通过对各个执行体计算得出的路由表项进行裁决比对，达到发现异常、阻断攻击的目的。宋克等[16]提出的拟态交换机系统结构能有效抵御基于未知后门和漏洞的网络攻击。欧阳玲等[17]依据拟态防御 DHR 架构与编码信道的对应关系，给出面向不同应用领域的拟态括号的构建方式，并在拟态交换机的实际场景中进行了实现和测试。

目前，拟态防御方面的研究工作大多为拟态防御思想在具体设备与系统中的实现与优化。在拟态防御的 DHR 架构中，系统攻击面主要集中于协议代理、表决器、调度器等拟态括号组件。对于如何提升拟态括号组件的安全性和功能正确性尚且缺乏研究。本文将形式化方法应用于拟态路由器协议代理的设计实现与形式化验证，证明了所实现的 BGP 代理的内存安全性和功能正确性。本文所采用的技术方法可为其他拟态括号组件的设计实现与形式化验证提供参考。

2　BGP 代理设计

2.1　总体架构

基于 DHR2 模型，本文设计实现了 BGP 代理。BGP 代理在拟态路由器系统中的位置如图 2 所示。在 DHR2 架构中，执行体分为主执行体和从执行体 2 种，BGP 代理通过监听并转发邻居路由器与主执行体之间的通信内容，模拟邻居路由器与从执行体进行 BGP 会话。具体来说，BGP 代理应具有以下功能：1) 处理收到的邻居路由器 BGP 报文，处理过程包括对报文进行分类、对比、转发、存储等；2) 模拟邻居路由器与从执行体通信；3) 管理从执行体所有 BGP 会话的状态；4) 支持新上线从执行体的状态同步。

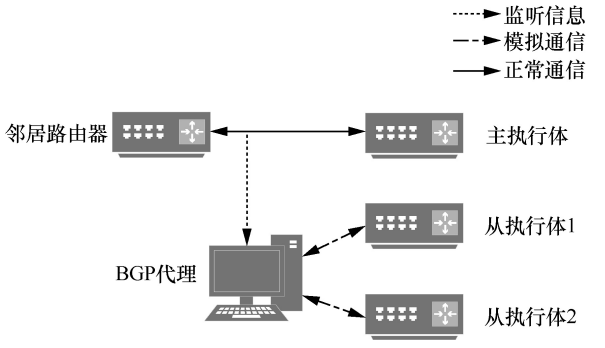


图 2　BGP 代理在拟态路由器系统中的位置

BGP 代理功能模块包括初始化模块、执行体信息配置模块、执行体状态同步模块、邻居信息配置模块、报文顶层处理模块。BGP 状态管理模块是报文顶层处理模块的一个子模块。BGP 代理功能模块架构如图 3 所示。

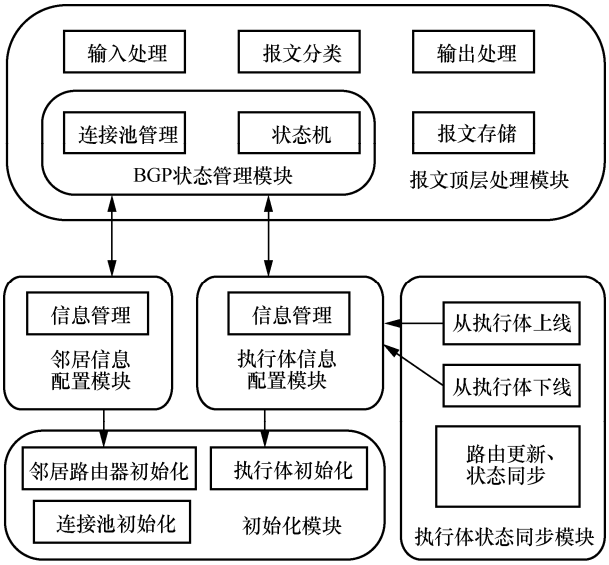


图 3　BGP 代理功能模块架构

初始化模块：程序启动时运行该模块，完成邻居路由器、执行体、连接池的信息初始化。

执行体信息配置模块：管理从执行体的状态、MAC 地址、IP 地址等信息。

执行体状态同步模块：完成从执行体轮换的操作，负责管理从执行体上线、下线等行为，同时完成从执行体上线之后的路由更新、状态同步。

邻居信息配置模块：管理邻居路由器的 IP 地址、端口等信息。

报文顶层处理模块：对输入报文进行分类，并仅对邻居路由器发送给主执行体的报文以及从执行体发送给协议代理的报文进行处理。

BGP 状态管理模块：报文顶层处理模块的一个子模块，管理与所有从执行体之间的 BGP 连接，负责维持每条连接的状态。

初始化模块、报文顶层处理模块（含 BGP 状态管理模块）、执行体状态同步模块为主要模块，接下来详细介绍其具体设计。

2.2　初始化模块

初始化模块中，BGP 代理从信息配置模块读取邻居路由器和从执行体的配置信息，包括 IP 地址、MAC 地址、自治系统（AS，autonomous system）号等，根据这些信息分别生成集合 Peers 和集合 Slaves，集合中的元素 Peer 和 Slave 分别表示具体路由器和从执行体。此外，初始化一个空的连接池，BGP 代理模拟邻居路由器与从执行体之间建立的连接都将添加到连接池中。初始化流程如图 4 所示。

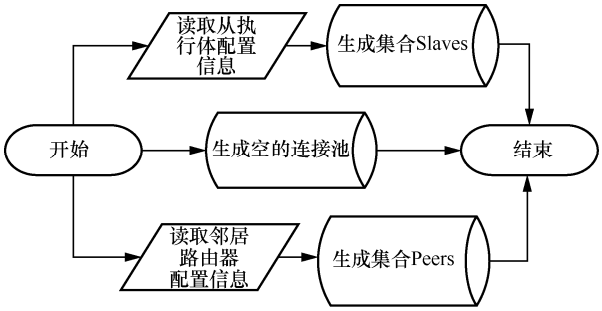


图 4　初始化流程

2.3　报文顶层处理模块

报文顶层处理模块中，BGP 代理对输入的报文先进行分类，再根据分类结果采取对应操作，报文处理流程如图 5 所示。

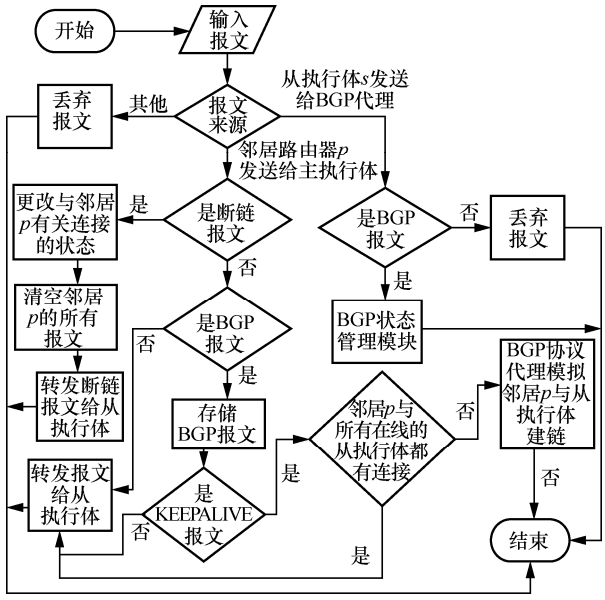


图 5　报文处理流程

BGP 代理收到报文后，首先判断该报文的来源，对两类输入流量做处理，一类是邻居路由器发送给主执行体的报文，此类报文通过监听主执行体的网卡获取，另一类是从执行体发送给 BGP 代理的报文，此类报文从 BGP 代理的网卡上抓取；对于其他报文，统一丢弃不做处理。当监听到邻居路由器 p 发送给主执行体的报文时，首先对其进行报文分类。如果是断链报文，如传输控制协议（TCP）的 FIN 报文或 BGP 的 NOTIFICATION 报文，则根据邻居路由器 p 的 IP 和端口信息，在连接池中找到与邻居路由器 p 相关的所有连接，更改其状态为 Wait_Close。然后，在集合 Peers 中找到邻居路由器 p 所对应的 Peer，清空 Peer 中存储的所有报文。最后，向从执行体转发该断链报文。如果是 BGP 报

文的 OPEN、KEEPALIVE、UPDATE 报文，则先将报文存储在邻居路由器 *p* 对应的 Peer 对象中，再进一步判断是否为 KEEPALIVE 报文。如果是 KEEPALIVE 报文，则说明邻居路由器 *p* 与主执行体之间建立了会话，如果在连接池中能找到在线的从执行体 *s* 和 *p* 之间的连接并且连接状态为 Established，直接转发 KEEPALIVE 报文即可；如果没找到连接，则 BGP 代理需要模拟邻居路由器 *p* 与从执行 *s* 建立会话，并将新建的连接放入连接池中。如果找到的连接状态不为 Established，则不做处理。

当收到从执行体 *s* 发送给 BGP 代理的报文时，如果是 BGP 报文，则在连接池中找到对应的连接，更新连接的状态机，并由状态机决定给从执行体 *s* 回复哪种类型的报文。如果不是 BGP 报文，则直接丢弃即可。

BGP 状态管理模块。BGP 代理在模拟邻居路由器与从执行体会话的过程中，对每个 BGP 会话都维持了一个独有的协议状态机。所有会话连接的状态均保存在连接池中，采用<源 IP，源端口，目的 IP，目的端口>四元组为连接标识。各连接的状态转换流程如图 6 所示。

BGP 代理收到从执行体 *s* 发来的报文时，首先对报文进行分类，对于除 BGP 的 OPEN 报文、KEEPALIVE 报文和 NOTIFICATION 报文外的其他报文，直接丢弃不做回复；对于 NOTIFICATION 报文，从连接池中删除对应连接；对于 KEEPALIVE 报文或 OPEN 报文，根据连接当前的状态进行相应处理。此外，BGP 代理不会生成报文，在与从执行体会话过程中，发送的都是 *p* 和主执行体之间会话过程中，BGP 代理监听并存储下来的报文。根据从执行体发送过来的报文的目的 IP，BGP 代理在 Peers 中找到 *p* 的对象 Peer，Peer 中存储了最新的 OPEN 报文、KEEPALIVE 报文以及所有的 UPDATE 报文，BGP 代理使用这些报文与从执行体会话，这样可以保证从执行体和主执行体之间的状态一致。

在整个状态机变化的过程中，以下两点需要特

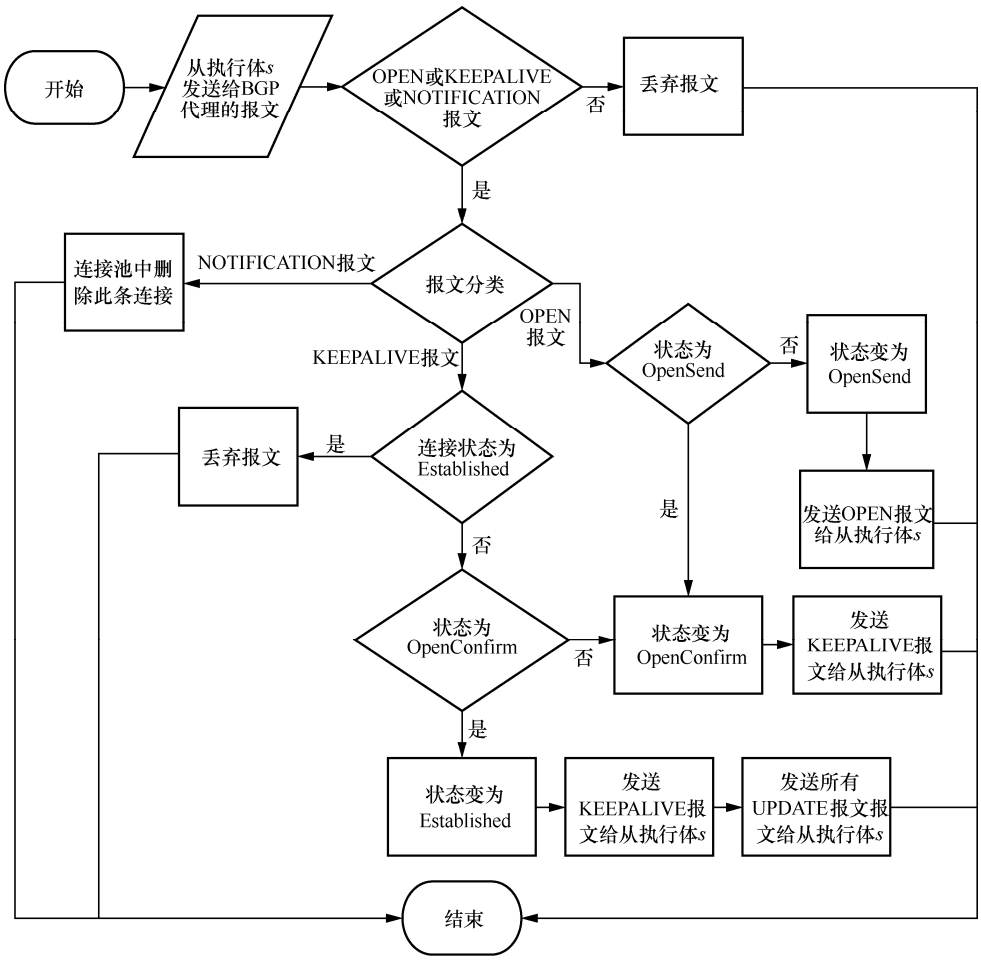


图 6　各连接的状态转换流程

别注意。1) BGP 代理不会主动给从执行体发送报文，这是因为 BGP 代理本质上是模拟邻居路由器与从执行体通信，那么只有监听到邻居路由器发送给主执行体的报文时，才会被动地转发该报文。2) 与从执行体建立会话时，因为要模拟邻居路由器和从执行体进行通信，所以发送的都是邻居路由器和主执行体建立会话时监听并存储下来的报文。

2.4　执行体状态同步模块

BGP 代理的关键功能之一就是支持从执行体的轮换，当一台从执行体下线时，BGP 代理会在连接池中删除其相关的所有连接。而当一台从执行体上线时，BGP 代理会模拟所有和主执行体已建立会话连接的邻居路由器，与新上线的从执行体建立会话连接，并将新上线执行体的 BGP 状态与主执行体完成同步。

从执行体轮换流程如图 7 所示。当收到从执行体 *s* 上线信息时，更改 *s* 的状态为在线，BGP 代理会模拟所有在线的邻居路由器与从执行体 *s* 建立会话，并在建立会话之后会将邻居路由器所存储的 UPDATE 报文全部发送给从执行体 *s*，以完成与主执行体和其他从执行体的状态同步。当收到从执行体 *s* 下线信息时，更改 *s* 的状态为离线并在连接池中删除所有跟 *s* 相关的连接。

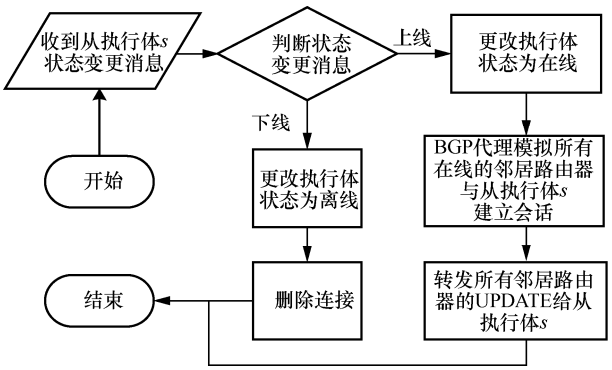


图 7 从执行体轮换流程

3　BGP 代理实现与形式化验证

3.1　BGP 代理实现

BGP 工作于应用层，其运输层采用 TCP，为了尽可能地降低程序的耦合性，便于程序验证和代码复用，本文同步开展了 TCP 代理的设计实现与形式化验证工作[18]。BGP 代理及其底层支撑模块的关系如图 8 所示，其中，操作系统采用 Ubuntu20.04，网卡抓包、发包库采用 Python 的 Scapy 库。

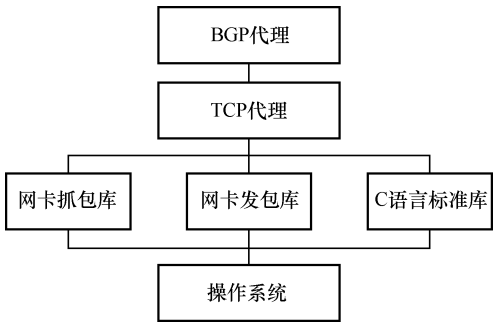


图 8　BGP 代理及其底层支撑模块的关系

本文的研究工作为后续针对拟态路由器其他网络功能的形式化验证提供了一个切实可行的参考方法。

3.2　形式化验证方法概述

本文系统的形式化验证借助 VeriFast[13]完成，VeriFast 可用于形式化验证 C 语言和 Java 语言程序，验证的过程基于符号执行[19]、分离逻辑[20]和 Z3[14]SMT 定理证明器，VeriFast 自身的可靠性证明已被 Coq 检查[21]。具体证明结构如图 9 所示。

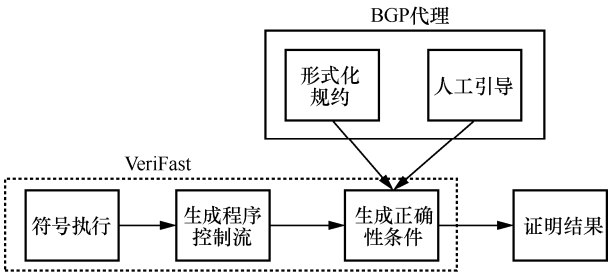


图 9　证明结构

使用者可以以注释的方式，给函数编写前置条件和后置条件，函数调用之间仅使用前置条件和后置条件来传递状态，如图 10 所示，这能有效缓解符号执行带来的状态爆炸问题。证明过程从函数的前置条件描述的符号状态开始，检查语句访问的每个内存位置的符号状态中是否存在权限，根据程序语句更新符号状态，并在函数返回时检查最终的符号状态是否满足函数的后置条件。

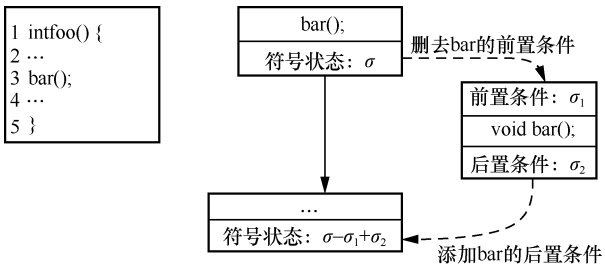


图 10　函数调用之间仅使用前置条件和后置条件传递状态

表 1　　**内存安全性验证方法**

特性名称	验证方法
内存泄露	每个函数堆上的内存块只有 2 个结果，一个在函数中被释放，另一个被添加到后置条件中返回上层函数。如果一块内存既没有在函数中被释放，也没有作为后置条件的一部分返回上层函数，那么该函数的验证就会报错，错误显示有内存块泄露
空指针引用	对于未在函数内赋值或判断非空以及前置条件中未明确描述非空的指针，符号执行会生成空指针的路径继续运行，从而导致验证不通过
指针多次释放	程序每次申请动态内存时，会将该内存的符号值放入函数堆中，释放时删除。如果一块内存已经被释放后再次被释放，验证会因为在函数堆中找不到该块内存的符号值而报错
指针访问的内存安全	程序申请动态内存时，会保存内存的符号值，用以详细描述该内存。通过检查当前函数是否拥有对这块内存的访问权限检查指针的解引用操作是否正确

本文形式化验证的目标为低级属性，即程序内存安全性，并在此基础上完成高级属性，即程序功能正确性（实现符合规约）的验证。

3.3　BGP 代理内存安全性验证

本文的内存安全性验证方法如表 1 所示。VeriFast 给程序模拟了一个函数堆，用于记录动态内存的生成、释放、传递。为了保证 BGP 代理的内存安全性，在实现与验证的过程中需不断调整代码和形式化规约，以便通过验证。

3.4　BGP 代理功能正确性验证

功能正确性的验证主要包括 4 个功能模块的形式化验证：初始化模块、报文顶层处理模块、BGP 状态管理模块以及执行体状态同步模块。

初始化模块的功能是生成邻居路由器集合 Peers 和从执行体集合 Slaves，以及对连接池完成初始化操作。该模块已形式化验证的功能特性包括：1) 堆上应有 Peers、Slaves 和连接池对应的内存块；2) 连接池中连接数为空；3) Peers 和 Slaves 中每个元素的 IP 值与配置文件中的信息一致；4) Slaves 中每个元素的状态都为在线，Peers 中每个元素的状态都为离线。

报文顶层处理模块已形式化验证的功能特性包括：1) 对于不同类型的报文，程序调用了正确的行为函数，通过在每个行为函数的前置条件中加入对 BGP 报文类型的限定来实现；2) 每个行为函数对于指定的输入，输出符合预期，即函数的功能正确，其中，不同的行为函数有不同的规约条件。例如，在报文存储的函数中，本文形式化规约并证明了 UPDATE 报文存储成功，且存储报文的链表长度加 1，具体如图 11 所示；此外，还证明了报文丢弃函数，每次运行该函数之后，堆上的报文对应的内存块及时被移除。

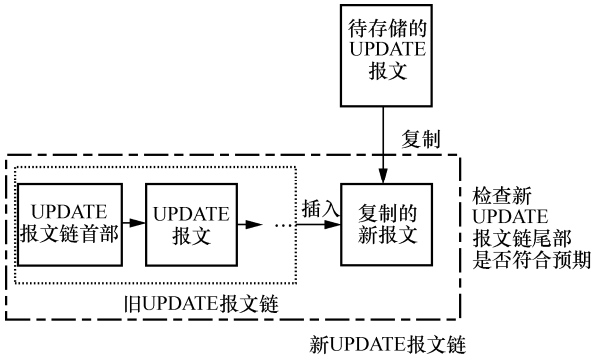


图 11　UPDATE 报文存储

BGP 状态管理模块的形式化规约最主要的是精确地定义了各 BGP 会话的状态转换逻辑，该模块已形式化验证的功能特性包括：1)运行发送 OPEN 报文的函数后，连接状态为 OpenSend；2) 发送 KEEPALIVE 报文的函数，如果前置条件中连接状态为 OpenSend，则后置条件中连接状态为 OpenConfirm，如果前置条件中连接状态为 OpenConfirm，则后置条件中连接状态为 Established。

执行体状态同步模块主要管理从执行体上下线和实现 BGP 状态同步。该模块已形式化验证的功能特性包括：1) 运行上线函数之后，从执行体状态为在线；2) 运行下线函数之后，从执行体状态为离线；3) 从执行体 *s* 上线后，能将所有邻居路由器的 UPDATE 报文发给 *s*。

3.5　形式化验证结果分析

本文对 BGP 代理书写的形式化规约已通过 VeriFast 定理证明器的检查。已形式化验证的各模块功能特性如表 2 所示。

代码开发及形式化验证工作量如表 3 所示，在开发的过程中，初始化模块、报文顶层处理模块（除 BGP 状态管理模块）、BGP 状态管理模块、

表 2 已形式化验证的各模块功能特性

模块名称	已验证的功能特性
初始化模块	1) 堆上应有 Peers、Slaves 和连接池对应的内存块； 2) 连接池中连接数为空； 3) Peers 和 Slaves 中每个元素的 IP 值与配置文件中的信息一致； 4) Slaves 中每个元素的状态都为在线，Peers 中每个元素的状态都为离线
报文顶层处理模块	1) 对于不同类型的报文，程序调用的行为函数正确； 2) 每个行为函数对于指定的输入，输出符合预期，即函数的功能正确
BGP 状态管理模块	1) 运行发送 OPEN 报文的函数后，判断连接状态为 OpenSend； 2) 发送 KEEPALIVE 报文的函数中，如果前置条件中连接状态为 OpenSend，则后置条件应该判断连接状态为 OpenConfirm，如果前置条件中连接状态为 OpenConfirm，则后置条件判断连接状态为 Established
执行体状态同步模块	1) 运行上线函数之后，从执行体状态为在线； 2) 运行下线函数之后，从执行体状态为离线； 3) 从执行体 *s* 上线后，能将所有邻居路由器的 UPDATE 报文发给 *s*

表 3 代码开发及形式化验证工作量

模块	C 程序代码行数	验证代码行数	验证代码行数/C 程序代码行数×100%（保留两位小数）	总行数
初始化模块	74	43	58%	117
报文顶层处理模块（除 BGP 状态管理模块）	377	217	57%	594
BGP 状态管理模块	489	301	61%	790
执行体状态同步模块	40	27	68%	67
其他部分	298	120	40%	418
总计	1 278	708	55%	1 986

执行体状态同步模块的 C 程序代码行数分别为 74 行、377 行、489 行、40 行，验证代码行数分别为 43 行、217 行、301 行、27 行。整体来看，BGP 代理的验证代码行数与 C 程序代码行数的比例为 55%，但在实际开发过程中，验证所需的人力、时间约是开发所需人力、时间的 3 倍。由于形式化验证排除了程序代码的内存安全问题，并证明了模块功能正确性，因此为形式化验证所花费的代价是值得的。BGP 代理的实现和验证代码详见文献[22]。

4 实验分析

4.1 实验环境构建

本文基于 eve-ng 4.0.1-86-PRO 平台搭建了拟态路由器模拟开发环境，实验设备均采用 PC 镜像实现，运行 Ubuntu 20.04 操作系统。邻居路由器 Peer、主执行体 Master 和从执行体（$Slave_1$、$Slave_2$）运行开源的 FRR（free range routing）路由协议栈。BGP 代理运行本文实现的程序模块 BGP-Proxy。实验设备通过 Open vSwitch 互联，设备信息如表 4 所示。

表 4 设备信息

设备名	IP 地址	本地 AS 号	配置的邻居 AS 号
Peer	192.168.180.129/24	129	130
Master	192.168.180.130/24	130	129
$Slave_1$	192.168.180.131/24	131	129
$Slave_2$	192.168.180.132/24	132	129
BGP-Proxy	192.168.180.133/24	—	—

本节实验的目的是检验系统的功能完整性和性能。功能完整性实验重点测试建立会话、路由发布、从执行体轮换这 3 个过程。性能测试重点对比与非验证版 BGP-Proxy 对于路由处理过程所需的时间的差异。

此外，可以关注路由处理实验中，BGP-Proxy 在面对大量的路由信息时，路由信息是否正确地分发，程序是否会崩溃以及最后 Master、$Slave_1$、$Slave_2$ 三者的路由表信息是否一致。

4.2 功能测试

4.2.1 *建立会话*

Master、$Slave_1$、$Slave_2$ 的 BGP 邻居信息如图 12 所示。从图 12 可以看出，其 BGP 邻居信息都显示 Peer 的 BGP 状态为 Established。

```
BGP neighbor is 192.168.180.129, remote AS 129, local AS 130, external link
Hostname: ubuntu
  BGP version 4, remote router ID 192.168.180.129, local router ID 192.168.180.130
  BGP state = Established, up for 02:59:16
  Last read 00:00:15, Last write 00:00:15
  Hold time is 180, keepalive interval is 60 seconds
BGP neighbor is 192.168.180.129, remote AS 129, local AS 131, external link
Hostname: ubuntu
  BGP version 4, remote router ID 192.168.180.129, local router ID 192.168.180.131
  BGP state = Established, up for 02:58:02
  Last read 00:00:18, Last write 00:00:02
  Hold time is 180, keepalive interval is 60 seconds
BGP neighbor is 192.168.180.129, remote AS 129, local AS 132, external link
Hostname: ubuntu
  BGP version 4, remote router ID 192.168.180.129, local router ID 192.168.180.132
  BGP state = Established, up for 03:46:02
  Last read 00:00:15, Last write 00:00:02
  Hold time is 180, keepalive interval is 60 seconds
```

图 12　Master、Slave$_1$、Slave$_2$ 的 BGP 邻居信息

4.2.2　路由发布

在 Peer 上连续发布 3 条新的路由信息，可以发现 Master、Slave$_1$、Slave$_2$ 都收到了对应的 UPDATE 报文。路由更新报文和路由表信息如图 13 和图 14 所示。通过抓取到的报文可以看到，3 个设备均完成了路由信息的同步。

```
192.168.180.129    192.168.180.130    BGP    UPDATE Message
192.168.180.129    192.168.180.130    BGP    UPDATE Message
192.168.180.129    192.168.180.130    BGP    UPDATE Message
192.168.180.129    192.168.180.131    BGP    UPDATE Message
192.168.180.129    192.168.180.131    BGP    UPDATE Message
192.168.180.129    192.168.180.131    BGP    UPDATE Message
192.168.180.129    192.168.180.132    BGP    UPDATE Message
192.168.180.129    192.168.180.132    BGP    UPDATE Message
192.168.180.129    192.168.180.132    BGP    UPDATE Message
```

图 13　Master、Slave$_1$、Slave$_2$ 的路由更新报文

```
K>* 0.0.0.0/0 [0/100] via 192.168.180.2, ens33, 00:04:27
B>* 3.3.3.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:06:06
B>* 4.4.4.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:05:33
B>* 5.5.5.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:05:21
K>* 169.254.0.0/16 [0/1000] is directly connected, ens33, 00:23:03
C>* 192.168.180.0/24 is directly connected, ens33, 00:23:03
K>* 0.0.0.0/0 [0/100] via 192.168.180.2, ens33, 00:03:37
B>* 3.3.3.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:53:40
B>* 4.4.4.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:53:07
B>* 5.5.5.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:52:55
K>* 169.254.0.0/16 [0/1000] is directly connected, ens33, 01:48:47
C>* 192.168.180.0/24 is directly connected, ens33, 01:48:47
K>* 0.0.0.0/0 [0/100] via 192.168.180.2, ens33, 00:18:34
B>* 3.3.3.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:43:01
B>* 4.4.4.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:42:28
B>* 5.5.5.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:42:16
K>* 169.254.0.0/16 [0/1000] is directly connected, ens33, 01:38:04
C>* 192.168.180.0/24 is directly connected, ens33, 01:38:04
```

图 14　Master、Slave$_1$、Slave$_2$ 的路由表信息

4.2.3　从执行体轮换

Slave$_1$ 下线之后，Slave$_1$ 中原先的路由信息都不再显示，如图 15 所示。

```
K>* 0.0.0.0/0 [0/100] via 192.168.180.2, ens33, 00:08:58
K>* 169.254.0.0/16 [0/1000] is directly connected, ens33, 01:54:08
C>* 192.168.180.0/24 is directly connected, ens33, 01:54:08
```

图 15　Slave$_1$ 下线后的路由表

将 Slave$_1$ 重新上线，可以发现 BGP-Proxy 会将 Peer 发送的 UPDATE 报文再次发送给 Slave$_1$，以完成与 Master 和 Slave$_2$ 的状态同步，路由表如图 16 所示。

```
K>* 0.0.0.0/0 [0/100] via 192.168.180.2, ens33, 00:04:44
B>* 3.3.3.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:00:58
B>* 4.4.4.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:00:58
B>* 5.5.5.0/24 [200/0] via 192.168.180.129, ens33, weight 1, 00:00:58
K>* 169.254.0.0/16 [0/1000] is directly connected, ens33, 02:23:50
C>* 192.168.180.0/24 is directly connected, ens33, 02:23:50
```

图 16　Slave$_1$ 重新上线后的路由表

4.3　性能测试

Master 下线后重新上线，触发 Peer 与 Master 重新完成 BGP 会话的操作，进而触发 BGP-Proxy 同样与 Slave$_1$、Slave$_2$ 完成 BGP 会话。统计当 Peer 中分别有 1 000、10 000、100 000 条路由信息时，BGP-Proxy 处理所有路由信息所需时间 R、Slave$_1$、Slave$_2$ 完成整个 BGP 会话所需时间 B（包括处理路由信息所花费的时间），以及除路由信息处理时间外 BGP 会话中其他步骤所需的时间 O，其中 B=R+O。考虑到统计时间受网络影响较大，因此对于上述 3 种情况都进行 10 次实验，结果如表 5 所示。其中，T_i（i=1～10）表示第 i 次实验的结果，A 表示 10 次实验的平均数。

从表 5 可知，随着路由数量的增加，经过验证的 BGP 代理对路由处理所需的时间以及完成整个 BGP 会话所需的时间都随之增加。但是可以发现，除了路由处理时间外，其他部分的时间都无明显增加。这是由于路由数量增加时，随之增加的只有 BGP 会话中 UPDATE 报文的数量以及 UPDATE 报文的长度，这部分报文的增加直接影响了 BGP-Proxy 对路由信息的处理速度。TCP 建链、BGP

表 5　　经过验证的 BGP-Proxy 对于 1 000、10 000、100 000 条路由信息的处理时间以及 BGP 会话时间

场景	步骤	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	A
1 000 条路由	R	0.010 6	0.012 9	0.016 7	0.017 0	0.014 5	0.018 2	0.017 2	0.015 2	0.016 4	0.014 1	0.015 3
	B	0.027 6	0.021 6	0.023 9	0.023 8	0.023 7	0.029 4	0.030 1	0.025 5	0.026 1	0.023 1	0.025 5
	O	0.017 0	0.008 7	0.007 2	0.006 8	0.009 2	0.011 2	0.012 9	0.010 3	0.009 7	0.009 0	0.010 2
10 000 条路由	R	0.079 7	0.105 5	0.035 1	0.054 0	0.053 8	0.062 6	0.058 7	0.066 1	0.055 7	0.063 2	0.063 4
	B	0.097 5	0.126 7	0.059 8	0.076 4	0.077 7	0.090 1	0.082 6	0.089 2	0.073 4	0.087 8	0.086 1
	O	0.017 8	0.021 2	0.024 7	0.022 4	0.023 9	0.027 5	0.023 9	0.023 1	0.017 7	0.024 6	0.022 7
100 000 条路由	R	0.193 9	0.091 4	0.203 1	0.196 8	0.071 0	0.195 0	0.151 5	0.083 4	0.087 9	0.105 7	0.138 0
	B	0.216 0	0.100 3	0.219 3	0.207 3	0.075 4	0.206 7	0.216 5	0.091 9	0.102 7	0.139 6	0.157 6
	O	0.022 1	0.008 9	0.016 2	0.010 5	0.004 4	0.011 7	0.065 0	0.008 5	0.014 8	0.033 9	0.019 6

交换 OPEN 报文等时间主要受网络波动的影响，并不受路由数量增加的影响。

未经过验证的 BGP-Proxy 的时间开销如表 6 所示。随着路由信息的增加（1 000 条、10 000 条、100 000 条），未经过验证的 BGP-Proxy 对路由器的处理时间以及完成 BGP 会话所需的时间也随之增加，但同验证版 BGP-Proxy 一样，完成整个 BGP 会话中除了路由处理时间外其他部分的时间都无明显增加。

对比表 5 和表 6 可以发现，当邻居路由器分别有 1 000、10 000、100 000 条路由时，对 *R*、*B* 和 *O*，未经过验证的 BGP-Proxy 都少于经过验证的 BGP-Proxy。当路由信息为 1 000 条和 10 000 条时，经过验证的 BGP-Proxy 在整个 BGP 会话的处理时间分别维持在 0.025 s 和 0.08 s 左右；当路由信息达到 100 000 条时，经过验证的 BGP-Proxy 完成整个 BGP 会话所需时间约是未经过验证的 BGP-Proxy 的 7 倍。其主要原因是，为了完成形式化验证，需要在编码的过程中进行妥协。例如，用链表代替哈希表，用递归代替循环，从而对程序性能造成了负面影响。但从绝对值来看，即使路由信息达到 100 000 条，经过验证的 BGP-Proxy 的处理时间也仍在毫秒级别，完成整个 BGP 会话时间也未超过 0.2 s，仍然保持高效性。

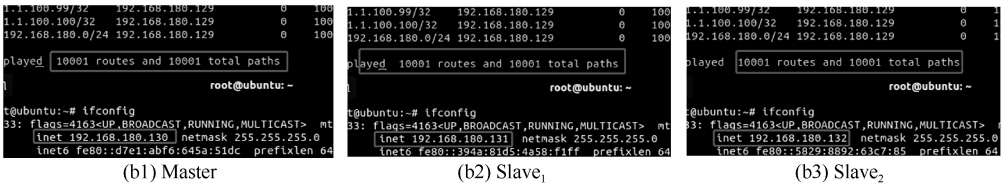
本节实验结束时，Master、Slave_1、Slave_2 的路由表信息如图 17 所示，即使当路由信息达到 100 000 条时，BGP-Proxy 仍运行良好，维持了 Master、Slave_1、Slave_2 三者路由表信息一致。这充分体现了本文所实现的 BGP 代理具有良好的稳健性。

表 6　未经过验证的 BGP-Proxy 对 1 000、10 000、100 000 条路由信息的处理时间以及 BGP 会话时间

场景	步骤	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	A
1 000 条路由	*R*	0.000 9	0.000 9	0.008 9	0.001 1	0.001 3	0.000 6	0.002 2	0.000 9	0.000 8	0.001 8	0.002 0
	B	0.007 8	0.009 2	0.013 7	0.015 4	0.005 4	0.006 6	0.008 7	0.010 3	0.004 7	0.007 2	0.008 9
	O	0.006 9	0.008 3	0.004 8	0.014 3	0.004 1	0.006 0	0.006 5	0.009 4	0.003 9	0.005 4	0.007 0
10 000 条路由	*R*	0.005 6	0.006 3	0.008 0	0.009 6	0.008 2	0.005 1	0.005 4	0.006 3	0.007 3	0.008 4	0.007 0
	B	0.008 8	0.016 0	0.012 2	0.011 0	0.016 7	0.008 1	0.009 1	0.016 3	0.013 3	0.015 8	0.012 7
	O	0.003 2	0.009 7	0.004 2	0.001 4	0.008 5	0.003 0	0.003 7	0.010 0	0.006 0	0.007 4	0.005 7
100 000 条路由	*R*	0.025 1	0.019 8	0.023 8	0.021 1	0.020 7	0.018 4	0.022 6	0.017 7	0.018 2	0.020 2	0.020 8
	B	0.032 3	0.025 7	0.030 6	0.027 3	0.027 0	0.026 6	0.031 3	0.026 5	0.025 3	0.027 4	0.028 0
	O	0.007 2	0.005 9	0.006 8	0.006 2	0.006 3	0.008 2	0.008 7	0.008 8	0.007 1	0.007 2	0.007 2

(a1) Master　(a2) Slave_1　(a3) Slave_2

(a) 当Peer有1 000条路由信息时，完成状态同步后的路由表



(b1) Master　(b2) Slave_1　(b3) Slave_2

(b) 当Peer有10 000条路由信息时，完成状态同步后的路由表

(c1) Master　(c2) Slave_1　(c3) Slave_2

(c) 当Peer有100 000条路由信息时，完成状态同步后的路由表

图 17　完成状态同步后，Master、Slave_1、Slave_2 的路由表信息

5　结束语

拟态路由器基于拟态防御 DHR 架构设计，具有比传统路由器更高的安全性，但目前尚且缺乏对拟态路由器安全非常重要的拟态括号组件的安全性和功能正确性进行严格验证的研究工作。形式化方法作为软件工程领域中保证软件正确性的一种严格方法，当前在安全攸关的网络功能模块开发过程中愈发得到重视。本文将形式化验证技术应用于拟态路由器中拟态括号组件之一的 BGP 代理，完成了 BGP 代理的设计实现，并利用 VeriFast 对 BGP 代理进行了形式化验证，证明了其内存安全性，以及各个模块的功能正确性（实现符合规约）。BGP 代理实现共 1 986 行代码，其中，实现代码与验证代码量之比约为 1.8:1，完成程序设计实现和程序验证所花费的时间之比约为 1:3。实验结果表明，所实现的 BGP 代理运行结果良好。总体而言，为了获得高度安全性所付出的代价是值得的。

本文为拟态防御设备与系统中拟态括号组件的形式化验证提供了方法参考。下一步考虑将验证方法推广到拟态路由器的裁决器、调度器等关键模块的设计实现中，进一步提升拟态路由器整体安全能力。

参考文献：

[1] ISLAM S, PAPASTERGIOU S, KALOGERAKI E M, et al. Cyberattack path generation and prioritisation for securing healthcare systems[J]. Applied Sciences, 2022, 12(9): 4443.

[2] Cisco. Cisco small business RV series routers vulnerabilities[R]. 2022.

[3] 马海龙, 伊鹏, 江逸茗, 等. 基于动态异构冗余机制的路由器拟态防御体系结构[J]. 信息安全学报, 2017, 2(1): 29-42.
MA H L, YI P, JIANG Y M, et al. Dynamic heterogeneous redundancy based router architecture with mimic defenses[J]. Journal of Cyber Security, 2017, 2(1): 29-42.

[4] 邬江兴. 网络空间拟态防御研究[J]. 信息安全学报, 2016, 1(4): 1-10.
WU J X. Research on cyber mimic defense[J]. Journal of Cyber Security, 2016, 1(4): 1-10.

[5] DOBRESCU M, ARGYRAKI K. Software dataplane verification[C]//Proceedings of 11th Symposium on Networked Systems Design and Implementation. New York: ACM Press, 2014: 101-114.

[6] ZHANG K, ZHUO D, AKELLA A, et al. Automated verification of customizable middlebox properties with gravel[C]//Proceedings of 17th USENIX Symposium on Networked Systems Design and Implementation. New York: ACM Press, 2020: 221-239.

[7] MOURA D L, BJORNER N. Satisfiability modulo theories: introduction and applications[J]. Communications of the ACM, 2011, 54(9): 69-77.

[8] ZAOSTROVNYKH A, PIRELLI S, PEDROSA L, et al. A formally verified NAT[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. New York: ACM Press, 2017: 141-154.

[9] ZAOSTROVNYKH A, PIRELLI S, IYER R, et al. Verifying software network functions with no verification expertise[C]//Proceedings of the 27th ACM Symposium on Operating Systems Principles. New York: ACM Press, 2019: 275-290.

[10] LIU J, HALLAHAN W, SCHLESINGER C, et al. P4V: practical verification for programmable data planes[C]//Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. New York: ACM Press, 2018: 490-503.

[11] DUMITRESCU D, STOENESCU R, NEGREANU L, et al. BF4: towards bug-free P4 programs[C]//Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM Press, 2020: 571-585.

[12] TIAN B C, GAO J Q, LIU M Q, et al. Aquila: a practically usable verification system for production-scale programmable data planes[C]//Proceedings of the 2021 ACM SIGCOMM 2021 Conference. New York: ACM Press, 2021: 17-32.

[13] JACOBS B, PIESSENS F. The VeriFast program verifier[R]. 2008.

[14] MOURA D L, BJØRNER N. Z3: an efficient SMT solver[C]//Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2008: 337-340.

[15] 邬江兴. 网络空间内生安全发展范式[J]. 中国科学: 信息科学, 2022, 52(2): 189-204.
WU J X. Development paradigms of cyberspace endogenous safety and security[J]. Scientia Sinica (Informationis), 2022, 52(2): 189-204.

[16] 宋克, 刘勤让, 魏帅, 等. 基于拟态防御的以太网交换机内生安全体系结构[J]. 通信学报, 2020, 41(5): 18-26.
SONG K, LIU Q R, WEI S, et al. Endogenous security architecture of Ethernet switch based on mimic defense[J]. Journal on Communications, 2020, 41(5): 18-26.

[17] 欧阳玲, 贺磊, 宋克, 等. 基于编码信道理论的拟态括号设计方法[J]. 信息工程大学学报, 2021, 22(4): 456-461.
OUYANG L, HE L, SONG K, et al. Design method of mimicry brackets based on coding channel theory [J]. Journal of Information Engineering University, 2021, 22(4): 456-461.

[18] 金希文, 葛强, 张进, 等. 拟态路由器 TCP 代理设计实现与形式化

验证研究[D]. 南京: 东南大学, 2022.
JIN X W, GE Q, ZHANG J, et al. Design, implementation and formal verification of TCP proxy in mimic defense router[D]. Nanjing: Southeast University, 2022.

[19] BALDONI R, COPPA E, D'ELIA D C, et al. A survey of symbolic execution techniques[J]. ACM Computing Surveys, 2019, 51(3): 1-39.

[20] REYNOLDS J C. Separation logic: a logic for shared mutable data structures[C]//Proceedings 17th Annual IEEE Symposium on Logic in Computer Science. Piscataway: IEEE Press, 2002: 55-74.

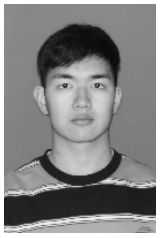
[21] JACOBS B, VOGELS F, PIESSENS F. Featherweight VeriFast[J]. Logical Methods in Computer Science, 2015, 11(3): 1-57.

[22] GE Q. BGP proxy source code [R]. 2023.

[作者简介]

张进（1979– ），男，江苏镇江人，博士，紫金山实验室高级工程师，主要研究方向为宽带信息网络、网络安全、软硬件协同设计。

葛强（1997– ），男，浙江台州人，东南大学硕士生，主要研究方向为网络安全。



徐伟海（1999– ），男，安徽黄山人，东南大学硕士生，主要研究方向为网络安全。

江逸茗（1984– ），男，河南郑州人，博士，信息工程大学副研究员，主要研究方向为新型网络体系结构、网络空间安全防御、云计算、虚拟化技术。

马海龙（1980– ），男，山东聊城人，博士，信息工程大学研究员，主要研究方向为创新网络体系、网络安全管控、路由工程等。

于洪涛（1970– ），男，辽宁丹东人，博士，信息工程大学研究员，主要研究方向为网络空间安全。