计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(9):99−113 [doi: 10.15888/j.cnki.csa.008662]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于超融合云网络环境的持续丢包主动探测系统①

李德方[1,2], 古　亮[2], 闫争争[1], 陈晓帆[2]

[1](中国科学院 深圳先进技术研究院, 深圳 518055)
[2](深信服科技股份有限公司, 深圳 518055)
通信作者: 闫争争, E-mail: zz.yan@siat.ac.cn

摘　要: 业务上云在近些年已经成为趋势, 而新冠疫情也加速了这一趋势. 然而公有云并不适用于所有用户. 尤其是出于数据隐私的考虑, 很多用户尤其是政府用户更希望在后疫情时代建设他们自己的私有云或者混合云. 超融合设备 (HCI) 是达到这一目标的有效手段. 在超融合设备中, 计算、网络、存储等资源都被完全虚拟化, 传统的物理网络设备单元也被一段段代码所替代. 此外为了获得高性能的网络转发能力, 很多创新技术应运而生, 其中 DPDK 技术是其中翘楚而被广泛应用. 开发者可以利用 DPDK 技术实现多种多样地、定制化地网络转发应用. 虚拟化技术和 DPDK 技术可以大大提升设备资源的利用率以及网络转发性能, 降低大中小企业或者机构的数据中心或者私有云的构建难度和成本. 但同时高度的虚拟化也给网络运维人员带来了巨大的挑战. 这些虚拟网元对网络运维人员而言是没有实体的, 虚拟网络在运维人员看来就像一个“黑盒”. 当网络出现故障时 (如丢包), 传统的针对物理网络设备的排障手段在虚拟网络中变得不可用, 这就大大增加了网络排障的时间, 进而对业务的持续运行造成影响. 针对这种问题, 设计了一种虚拟网络持续性丢包探测系统 Flowprobe, 该系统旨在解决基于 DPDK 用户态虚拟网络的持续性丢包检测及根因定位问题. 通过该系统, 用户可以观测数据包在虚拟网络中的详细路径、经历的转发行为, 定位丢包的位置, 获知丢包的原因. 实验表明, 该系统可以针对 576 种虚拟网络持续丢包场景进行检测以及给出问题根因, 并且该系统做到了对正常转发业务的无影响, 性能测试表明, 该系统开启以后, 对用户正常业务的转发影响可以控制在 1% 以内. 该系统已经在超融合生产环境持续运行了 3 年, 帮助用户以及网络运维人员解决了诸多虚拟网络故障问题.

关键词: 虚拟化; 软件定义网络 (SDN); 云计算; 超融合; DPDK; 故障诊断; 微服务框架

引用格式: 李德方,古亮,闫争争,陈晓帆.基于超融合云网络环境的持续丢包主动探测系统.计算机系统应用,2022,31(9):99–113. http://www.c-s-a.org.cn/1003-3254/8662.html

Proactive Diagnostic System for Persistent Packet Loss in Cloud Networks Based on Hyper-converged Infrastructure

LI De-Fang[1,2], GU Liang[2], YAN Zheng-Zheng[1], CHEN Xiao-Fan[2]

[1](Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China)
[2](Sangfor Technologies, Shenzhen 518055, China)

Abstract: Moving business to cloud has been a trend recently, and COVID-19 gives a push to this trend. However, not all forms of business are suitable for public cloud computing. For the sake of data privacy, plenty of users, especially government users, prefer to build their own private cloud or hybrid cloud in the post-COVID-19 world, and hyper-converged infrastructure (HCI) is a convenient way to achieve this goal. In HCI, computing, storage, and network are all virtualized, which leads to higher resource utilization and easier way to be deployed. The network elements are no longer present as sensible hardware blocks in HCI but as lines of codes to function instead. To achieve better data forwarding performance in virtualization, many innovative technologies have risen, among which DPDK has been widely studied and applied. With DPDK, developers can customize various network forwarding applications. Virtualization and DPDK can

① 收稿时间: 2021-11-29; 修改时间: 2021-12-28; 采用时间: 2022-01-05; csa 在线出版时间: 2022-06-24

greatly improve resource utilization and network forwarding performance, reducing the difficulties and costs of building data centers or private cloud by enterprises of various scales or institutions. However, virtualization at a high level also poses great challenges to network operation and maintenance owing to the loss of physical network entities. When a virtual network suffers a failure (e.g., packet loss), the traditional diagnosis tools designed for hardware network equipment cannot fulfill the need of cause locating and analyzing, resulting in much more mean time to repair (MTTR) and business loss. Even worse, the virtual network seems like a black box to network operators, which makes the network vulnerable. To solve these problems, this study proposes a proactive diagnostic system for persistent packet loss in HCI based cloud, named Flowprobe, which aims to enable the detection and cause locating of persistent packet loss for user-space virtual networks based on DPDK. With this system, users can have a comprehensive view of the way in which the packet traverses through the virtual network, the actions that the packet has performed, the positions that suffer packet loss, and the causes resulting in the loss. Thoughtful evaluation has proven that the system can handle 576 packet loss scenarios in virtual networks. Meanwhile, it has a good performance, with the performance degradation of data forwarding not exceeding 1% when the system is functioning. The system has been deployed in the HCI production environment for about 3 years and helped solve many problems in virtual networks.

Key words: virtualization; software-defined network (SDN); cloud computing; hyper-converged infrastructure; data plane development kit (DPDK); fault diagnosis; microservice framework

1 引言

随着云计算技术的不断发展, 业务上云在各个大中小企业、政府机关、教育医疗机构等逐渐成为趋势, 并且该需求在后疫情时代变得更加紧迫. 但公有云并不适合于所有的用户和业务场景, 由于数据的保密性等原因, 利用超融合[1] 基础设施来构建私有云或者混合云是中小企业、政府机关、教育医疗机构等实现业务上云的有效手段. 超融合设备实现了计算、存储、网络等基础设施的虚拟化, 此外, 借助于网络功能虚拟化 (network function virtualization, NFV), 网络安全检测功能体也可以集成到超融合设备中. 当前国内厂商如深信服、华为、华三、SmartX 等, 国外厂商如 VMware、Nutanix、Redhat 等都提供超融合设备以及完善的基于超融合的云解决方案. 本文将关注于基于超融合构建的私有云或者混合云的网络问题以及解决方案.

借助于网络虚拟化, 传统的盒子式的硬件网络设备被一段段代码替代. 网络虚拟化的实现方式一般分为内核态实现和用户态实现 (本文涉及到的虚拟化技术默认都是基于 Linux 系统来进行实现的. Linux 系统在运行时会分为用户态空间和内核态空间). 内核态实现主要方案为借助于 Vhost-net/Virtio-net 架构[2] 和 OpenVSwitch (OVS) 来进行实现, 但其性能不高. 为了提升虚拟网络的转发性能, 业界也提出了很多转发优化方案, 如内核态的 OVS+XDP (eXpress DataPlane) 方案[3]. 其中应用最广的还是基于 DPDK (data plane development kit)[4] 的方案. DPDK 是 Intel 开发的一套数据面性能优化套件, 借助于此, 开发者可以在 Linux 用户态空间定制化地开发所需的虚拟网络转发设备, 如虚拟交换机、虚拟路由器等. 性能测试表明, 基于 DPDK 的数据面 (data plane, DP) 的转发性能相对于普通的内核转发性能可以提升 10 倍以上[5]. 因此, DPDK 技术在高性能数据转发场景得到了广泛的应用.

为了提升虚拟网络的性能, 超融合设备的网络转发面 (也即虚拟网络数据面) 一般还会应用软件定义网络 (software-defined network, SDN) 技术[6], 借助于该技术, 可以实现网络转发和网络控制功能的分离, 以及快慢路分离的转发路径[7], 大大提升了网络转发效率. 其实 OVS 即是 SDN 交换机的具体形式, 而 DPDK+OVS 的方式也是高性能虚拟网络数据面的主流开源实现方案. 本文所述虚拟交换机并不是 OVS 但是基于 DPDK 和 SDN 思想来开发的, 工作原理与 OVS 相近.

随着虚拟化技术、SDN 技术、DPDK 用户态转发技术等新技术的不断引入, 云计算中的网络 (虚拟网络) 性能也变得越来越高, 但同时也变得越来越复杂. 对于网络运维人员而言, 虚拟网络更像是一个全新的事物, 需要学习如何去管理, 但不幸的是, 应用于传统物理网络设备的很多经典工具无法满足虚拟网络排障的需求, 这就使得虚拟网络呈现出了一种“黑盒”的特

征, 网络运维人员不能获知虚拟网络的运行状态, 当虚拟网络发生故障时, 如连续长时丢包, 也就无法很快地定位出故障位置以及故障原因, 进而导致更长的故障修复时间, 更大的经济损失.

为了解决虚拟网络的排障问题, 工业界和学术界均提出了很多解决方案, 如将传统的工具进行虚拟化改造, 但更有效的方法是设计新的遥测技术 (telemetry), 如思科和华为提出的带内网络遥测 (in-band network telemetry, INT) 技术[8−10], 开源的 OpenTelemetry 技术[11], OpenTracing 技术[12] 等. 遥测技术一般分为带内遥测 (in-band telemetry) 和带外遥测 (out-band telemetry), 带内遥测获得的信息更加准确, 但性能开销比较大, 并且可扩展性也不如带外遥测. 带外遥测获得的信息不如带内遥测精确, 但性能开销一般较小, 并且具备良好的可扩展性. 本文提出的方案是一种带外遥测, 但我们做了一系列的优化, 使之能取得接近于带内要测的效果.

在设计虚拟网络故障检测工具或者系统时, 需要遵循如下 3 个原则.

高性能: 基本上任何的检测或者监控工具/系统都会产生“观察者效应”[13], 观察者效应会影响到观察结果的有效性, 观察者效应越明显, 所得到的结果也就越不准确, 也就失去了观测的意义. 这种观察者效应在虚拟化环境中会更加明显, 因而虚拟化环境是更加功能集中, 资源紧缺的, 因此, 设计的检测或者监控工具要做到高性能、低资源开销.

低干扰: 除了观察者效应的影响, 设计的检测或者监控工具/系统在运行时不应该干扰到正常的业务流量, 更不能因为故障检测导致正常业务崩溃.

信息完全性: 收集到的信息应该足够全面, 并且分析方法也要全面, 可以多角度地对收集到的信息进行分析并得出结论.

在上述原则下, 本文设计了一种虚拟网络持续性丢包探测系统: Flowprobe, 该系统可以回答以下问题:

1) 数据包在虚拟网络中的详细转发路径是什么? 是否与其路径一致.

2) 若发生丢包, 则持续性丢包的虚拟网元或者位置在哪.

3) 持续性丢包的原因是什么.

该虚拟网络持续性丢包检测系统针对基于 HCI 构建的私有云或者混合云而设计, 其主要思想是通过主动发送带标记的探测包, 虚拟网络数据面识别到这些探测包以后解析探测包的指令信息, 根据指令信息上传探测信息至控制器, 在控制器进行路径计算和丢包计算. 该系统针对的虚拟网络是基于 DPDK 来实现的, 因此, 该系统完全位于 Linux 用户空间, 运行时不会对 Linux 内核空间造成干扰. 为了使得所设计的系统符合上述原则以及解决所提出的问题, 做了如下创新设计与优化:

1) 支持定制不同类型的 IPv4/IPv6 探测包, 如 TCP、UDP、ICMP 等, 并支持探测包包头参数和包大小的定制. 支持数据包的定制可以使得探测包更加接近于用户业务包, 如此一来, 探测包经历的网络行为与业务包经历的网络行为就更加一致, 得到的探测结果也更加准确.

2) 设计了高效的探测包标记方案. 通过该方案做到了对虚拟网络数据面正常业务转发的最小影响.

3) 探测包不会由用户虚拟机发出, 也不会进入到用户虚拟机. 如此一来, 可以减小对用户虚拟机工作负载的干扰.

4) 支持探测路径的计算和展示.

5) 支持显示数据包被丢弃的原因.

通过该系统, 用户可以观测数据包在虚拟网络中的详细路径、经历的转发行为, 定位丢包的位置, 获知丢包的原因. 实验表明, 该系统可以针对 576 种虚拟网络持续丢包场景进行检测以及给出问题根因, 并且该系统做到了对正常转发业务的无影响, 性能测试表明, 该系统开启以后, 对用户正常业务的转发影响可以控制在 1% 以内. 该系统已经在超融合生产环境持续运行了 3 年, 帮助用户以及网络运维人员解决了诸多虚拟网络故障问题.

接下来的部分将对 Flowprobe 系统进行详细的介绍. 第 2 节将介绍研究背景, 针对超融合设备、SDN 网络、DPDK、网络遥测技术等进行简明的阐述. 第 3 节将介绍相关工作. 第 4 节将具体介绍 Flowprobe 的系统设计与实现. 第 5 节对 Flowprobe 系统进行了详细的功能和性能评估. 最后在第 6 节, 对工作进行了总结, 并讨论了以后的演进方向.

2 研究背景

2.1 超融合与超融合云网络

超融合是一种 IT 基础架构解决方案, 将计算、存储和网络资源整合在一个统一系统中. 超融合基础架

构由计算资源 (虚拟机)、软件定义存储和软件定义网络组成, 并使用超级资源管理器 (hypervisor) 进行统一管理和配置[14].

超融合设备可以认为是当前虚拟化技术的集大成者, 在超融合设备中会使用到计算虚拟化、网络虚拟化、存储虚拟化等多种虚拟化技术. 一般而言, 主流的开源超融合虚拟化实现方案是基于 KVM 与 QEMU 来实现的.

图 1 是 HCI 的一般架构图[15]. HCI 会集成计算、存储、网络、安全等功能, 并且可以构建平台即服务 (platform as a service, PaaS) 和基础设施即服务 (infrastructure as a service, IaaS) 平台. 在 PaaS 和 IaaS 的基础之上提供多种多样的应用服务. HCI 基础设施既包括软件也包括硬件, 因此它拥有硬件管理服务和软件定义基础设施管理服务, 通过这两种服务, 用户可以对平台资源进行管理和调度, 构建集群, 进行高可用配置等. 平台自身一般会提供 REST API 接口, 并可以与第三方软件进行集成.

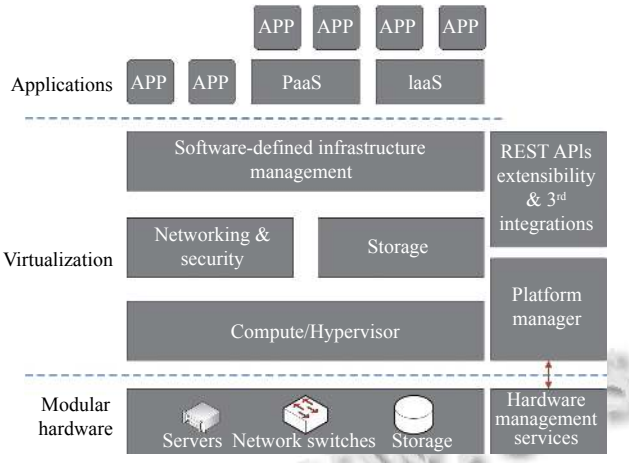

图 1 HCI 功能架构

借助于虚拟化, HCI 可以为用户提供灵活便捷的基础设施管理和配置方式. 如深信服的 HCI 设备提供丰富的网络虚拟化功能, 包括分布式虚拟交换机、虚拟路由器、分布式虚拟防火墙、虚拟负载均衡器、虚拟上网行为管理设备等, 并且提供一个“画布”, 在上面可以通过简单的拖拽、点击、连线就可以构建出一个包括上述各种虚拟网元的完整可用网络.

鉴于 HCI 的高度虚拟化和便捷的管理特性, 用户可以基于 HCI 来很方便、快捷地构建自身的私有云或者混合云环境. 图 2 是一个基于 HCI 设备搭建的简单的云网络示例. 从图中我们可以看出基于 HCI 构建的私有云或者混合云网络可以被清晰地分为两部分: 虚拟网络部分和物理网络部分. 物理网络用来连接一个个 HCI 设备. 其中在虚拟网络中, 会区分不同的租户. 当网络发生故障时, 针对物理网络, 网络运维人员可以使用丰富的排障工具包进行问题定位, 但是这些工具并不能很好地应用到虚拟网络中, 因此, 针对虚拟网络需要设计专门的排障工具或者系统.

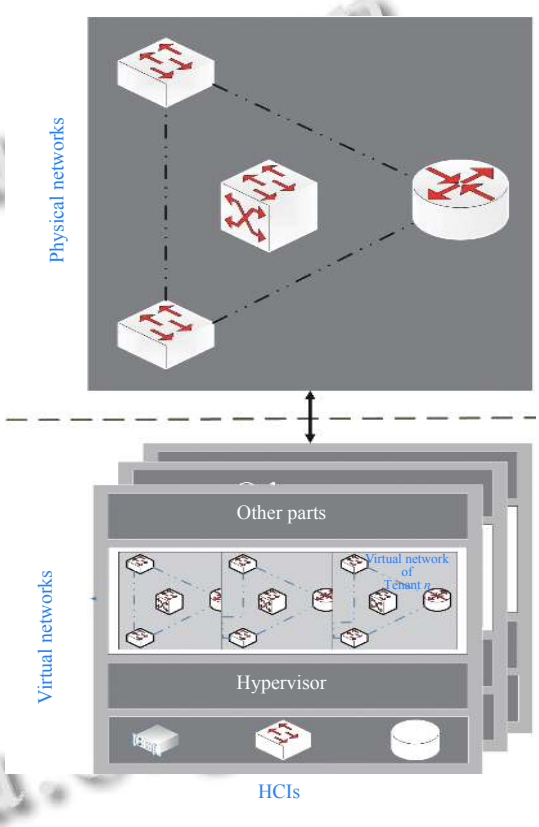

图 2 基于 HCI 构建的云网络示例

本文提出的 Flowprobe 即是针对上述 HCI 网络结构设计的持续性丢包检测与分析系统.

2.2 SDN 网络

SDN 网络有 3 大特征, 分别是[6]:

1) 集中式控制.

2) 控制功能与转发功能分离.

3) 可编程.

为了提升虚拟网络的转发性能以及方便虚拟网络的管理, HCI 内部的网络一般采用 SDN 架构. SDN 控制器也即 HCI 网络的管理平面可以获知虚拟网络的拓扑信息、配置信息等, 可以完成路由计算、网络负载均衡、设备主备切换等功能. 因此, 借助于 SDN 控制

器可以进行网络排障分析, Flowprobe 系统的路径计算即是借助于 SDN 控制器中信息来完成的.

2.3 用户态数据面与 DPDK

图 3 是用户态数据面和内核态数据面接收数据包的流向示意图 (数据包发送为相反方向). 数据包被网卡接收以后直到被应用 (图中为 VM, 即虚拟机) 处理, 需要经过宿主机也即 (HCI 内核系统) 的数据包转发.

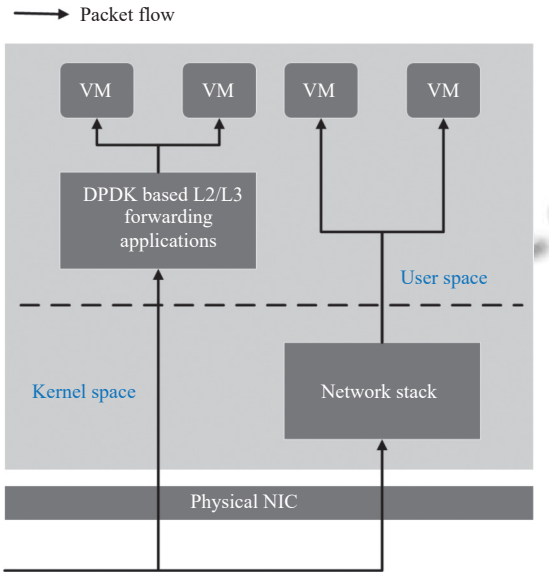

图 3 用户态数据面和内核态数据面接收数据包数据包流向

经典的网络转发是由 Linux 内核网络协议栈来完成的, 在网络协议栈中, 数据包被重新进行校验、拼接、信息统计等, 上述操作都通过的数据包然后被由内核空间再复制到用户空间, 进而传送到 VM 内. 该过程比较繁琐, 此外, 由于涉及到了数据包拷贝, 因此还会有一定的性能损耗.

用户态数据面位于 Linux 的用户空间 (user space), 图 3 中的用户态数据面是基于 DPDK 来实现的, DPDK 并非用户态数据面所必须, 但基于 DPDK 的用户态数据面可以获得非常好的性能, 并且也在生产环境中得到了广泛的应用. 数据包由物理网卡直接透传到用户态数据面, 并不会经过内核态空间, 该过程会利用到零拷贝、直接内存访问 (direct memory access, DMA) 等技术, 因此不会存在由内核空间向用户态空间进行数据拷贝的操作. 一般而言, 用户态数据面专注于 L2/L3 转发, 比较少有 L4 的处理, 因为虚拟网络的主要职责在于进行高速数据转发, 并不会有过多进行 L4 及以上层的数据包解析的需求.

DPDK 是 Intel 推出的高性能数据面开发套件. 为了取得加速数据包转发, DPDK 做了一系列的优化, 如基于轮询的收发包驱动 (poll mode driver, PMD), 使用大页内存, 增加 CPU 的亲和性, 重新设计数据包缓存结构等. 实验表明基于 DPDK 实现的网络转发面的性能可以 10 倍于普通的基于内核的网络转发面. 因此 DPDK 在生产环境中得到了广泛地应用. 此外 DPDK 应用完全运行于用户态空间, 相对于运行于内核空间的转发面会具备更加好的稳定性, 即使其发生故障也不会引起整个系统的崩溃.

基于 DPDK 的转发赋予了开发者灵活的开发能力, 使得开发者可以根据自己的需要实现数据面转发能力. 这也导致基于 DPDK 的数据转发面相对于基于内核的转发面没有那么标准统一, 增大了使用者和网络运维人员的维护成本. 事实上, 很少有使用者或者网络运维人员能理解 DPDK 技术以及基于该技术实现的转发应用的工作原理, 因为统一的、规范的资料比较少, 并且需要较深的专业背景. 基于 DPDK 开发的转发应用或者虚拟网络需要像基于内核的网络转发面一样可以有很多方便的工具来进行监控和检测, 使得使用者及网络运维人员能够直观地了解基于 DPDP 开发的网络转发应用的运行状态, 才能降低使用者的门槛, 方便他们的使用. 但这些运维工具恰恰是当前 DPDK 生态所欠缺的.

本文提出了一种针对基于 DPDK 实现的虚拟网络转发面的持续性丢包检测系统, 其目的即在于方便用户以及网络运维人员能够直观地看到数据包在虚拟网络中的详细转发路径、丢包发生位置、丢包原因等, 大大降低他们维护基于 DPDK 的虚拟网络的门槛和时间.

3 相关工作

随着虚拟化技术的深层次化地引入, 云计算网络也变得越来越复杂, 如网元数目呈指数级增长, 网元种类也越来越多. 在此背景下, 当网络出现性能下降或者故障时, 如何定位网络故障, 以及进行网络性能分析就变得十分困难. 这就使得高效、灵活的网络故障感知、定位变得越来越重要. 高效、灵活的网络运维技术可以大大提高网络的管理和维护的效率, 进而带来巨大的经济价值.

传统的网络故障检测工具, 如 ping、traceroute 等, 可以对网络状态进行一定的探查, 如查看源/目的端的网

络是否处于正常连接状态, 源到目的的简单路径信息显示等. 但这种简单地网络状态探知已经不能满足当前云计算网络, 尤其是虚拟网络的需要. 为此, 在业界和学术界对网络状态或者故障探测一直都有着广泛深入的研究.

微软提出了 Everflow 技术来解决数据中心丢包、时延的检测和根因分析问题[16]. Everflow 设计了功能强大的数据包过滤器, 允许用户设置匹配规则, 然后将这些规则下发到交换机上对指定类型的数据包进行镜像. 通过负载均衡操作将镜像的数据包分散到多个处理服务器, 提高数据分析处理的速度. 云杉网络认为高级的网络管理功能依赖于详细的网络流量分析, 为此, 他们提出了一种基于策略的网络流量镜像方案[17]. 在该方案中, 他们提出了一种创新的流量分类算法, 以加快策略的匹配速度, 同时降低性能损耗.

传统的探测手段一般都属于带外探测技术, 带外探测技术实现简单、灵活, 但获取的信息并不精确. 基于流量镜像的技术虽然可以获取到最详细的网络信息, 但性能开销大, 产生的数据量也大, 需要付出大量的计算资源去处理流量信息, 对于很多网络场景并不适用. 为了精准探测网络状态、对网络故障进行快速定位, 并在一定程度上降低信息采集时的性能损耗. 思科和华为提出了 INT 技术[8–10]. 通过 INT 技术, 网络的数据平面可以收集和上报网络的详细运行状态, 而无需进行数据包镜像. 在 INT 架构中, 数据包在其头部包含有可以被网络设备解析的遥感指令. 根据这些遥感指令, INT 使能的网络设备可以知道收集何种信息, 并将这些信息写入流经它的数据包内. INT 流量源 (traffic sources, 包括但不限于网络应用、主机网络协议栈、超级管理器、网卡、发送方交换机等) 可以将遥感指令嵌入到正常的数据包中, 或者特殊的探针包 (probe packets) 中. 在 INT 流量目的地 (traffic sinks) 中, 收集到的信息被从数据包中提取出来, 借此 INT 技术可以监控数据平面的运行状态. 经典的 INT 技术是将探测信息封装到探测包之中, 但这种方式受限于数据包的大小, 以及路径的 MTU 等, 导致这种方式的 INT 技术无法探测比较长的路径, 如基于这种技术架构实现 INT 技术的华为交换机最大只能支持 8 跳探测[18]. 为此, 华为提出了 Postcard 模式的 INT 技术[19], 即探测包仅封装探测指令, 每个 INT 使能的网元接收到探测包以后, 解析指令信息, 然后根据探测指令收集所需探测信息, 最后将探测信息额外封装到另一个数据包之中, 上传到控制器. 阿里云在自己的 Overlay 网络上实现了一套持续性丢包检测系统—vTrace[20], 该系统可以自动发现阿里云 Overlay 网络发生持续性丢包的位置以及丢包的原因, 其实现原理与华为的 Postcard 模式的 INT 技术原理类似. 文献 [21] 中, 作者针对云计算网络基于 Open vSwitch 的流表设计了一种可行的 INT 技术方案, 但该方案的性能损耗比较大.

如上所述, INT 技术会带来额外的开销, 影响网络流的处理时间, 增加了处理时延, 进而会影响应用层面的网络性能. 为此, 在文献 [22] 中, 作者提出了一种统计型 INT 技术, 它将探测信息编码到多个数据包之中, 最大可以将额外开销降低到 1 bit 每数据包. 在文献 [23] 中, 作者提出了一种基于 sketchlets 的 INT 探测系统, sketchlet 是他们精心设计的数据结构, 该数据结构可以嵌入到数据包之中, 随数据包进行转发和信息收集, 该数据结构的大小是固定的, 在转发过程中并不会一直增大, 这在很大程度上解决了 INT 技术的可扩展性和性能损耗问题.

借鉴于 INT 技术的思想, 我们针对基于超融合设备构建的云计算网络设计了一套持续性丢包检测系统, 该系统可以发现超融合虚拟网络中的持续性丢包的位置, 并给出丢包的原因. 该系统是一种带外探测技术, 但为了接近带内探测的效果, 我们支持探测包可定制, 用户可以通过参数来配置探测包, 以最大程度地接近自己的业务包, 详细设计见第 4 节.

4 系统设计与实现

4.1 系统架构

图 4 为 Flowprobe 的总体架构示意图. 由图可知, Flowprobe 是一个典型的分布式 CS (client-server) 系统, 其中包括中央控制器 (Flowprobe controller)、分布在各个 HCI 宿主机上的代理 (Flowprobe agent)、以及在每一个 HCI 转发面中的虚拟探针 (vProbe). Flowprobe controller 负责接收用户的探测请求, 处理用户的探测配置, 将探测配置和命令下发到 Flowprobe agent, 接收来自于 Flowprobe agent 的探测信息, 对探测信息进行路径的拼装, 将探测信息进行存储, 向上层应用提供 API 接口供前端或者第三方应用调用等.

Flowprobe agent 主要负责接收来自 Flowprobe controller 的探测指令, 根据指令构造探测包, 将探测包注入到 HCI 数据转发面, 并接受来自转发面 vProbe 的

探测信息, 将探测信息进行打包上传到 Flowprobe controller. Flowprobe agent 与 Flowprobe controller 之间使用 RPC (remote process call) 进行通信. vProbe 的主要作用在于识别探测包, 根据探测包中的指令信息将数据面对探测包的转发行为进行记录并上传到本地 Flowprobe agent. vProbe 与 Flowprobe agent 之间使用 UNIX socket 进行通信.

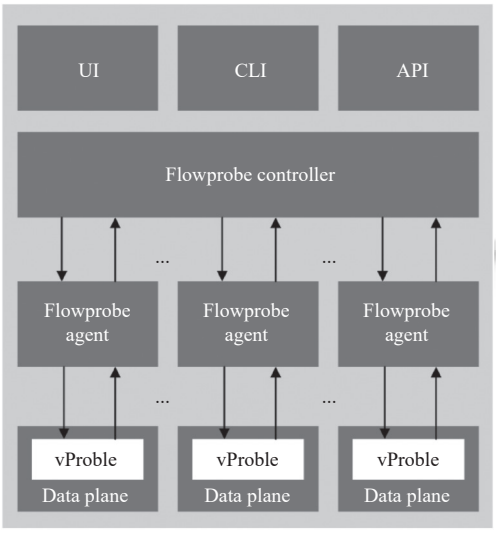

图 4 Flowprobe 总体架构示意图

图 5 是 Flowprobe 的一个探测示例, 下面将以该示例说明 Flowprobe 的工作流程.

用户从 UI (user interface) 界面配置探测包, 并开启 Flowprobe 探测功能. 随后用户的配置被向下传递到 Flowprobe controller, Flowprobe controller 对配置进行合法性检查、格式化等, 然后将探测包配置信息下发到探测发起 HCI 的 Flowprobe agent.

Flowprobe agent 根据配置信息构造出探测数据包. 探测数据包从位于用户虚拟机机 1 (VM 1) 和虚拟防火墙 (vFirewall 1) 之间的虚拟链路 (vlink) 注入, 随后探测数据包在各个网元之间被进行转发, 分别经过 vSwitch 1, vRouter 1, vSwitch 2, vRouter 2, vSwitch 3, vFirewall 2. 探测数据包经过的各个网元分别将自身的设备信息、对数据包执行的网络行为利用 UNIX socket 传递给本地 Flowprobe agent, 最后这些信息被 Flowprobe agent 进行格式化被上传到 Flowprobe controller.

Flowprobe controller 进行信息处理 (关联、去冗余等) 和路径重建以后, 将探测信息存入到数据库中. 前端或者第三方应用可以通过标准 API 或者 CLI 接口对数据库进行访问, 对结果进行可视化展示.

值得注意的是, 在探测数据包到达 vFirewall 2 时, vFirewall 2 在完成自身信息和网络行为 (送达) 的统计和上传以后, 即将探测数据包丢弃, 而并不将其转发进用户虚拟机 2 (VM 2). 由整个探测过程可知, 探测包不会由用户虚拟机发出, 也不会进入到用户虚拟机. 如此以来, 可以最大程度降低对用户虚拟机工作负载的干扰.

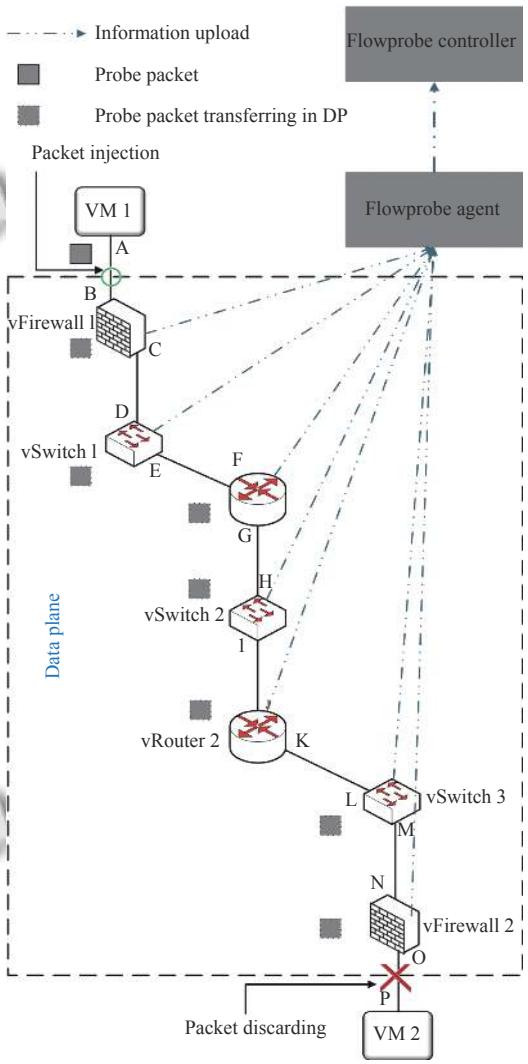

图 5 Flowprobe 系统探测示例

在接下来的第 4.2 节和第 4.3 节, 将对详细的系统设计进行阐述, 以揭示系统设计时对高性能、低干扰和信息完全性的考虑. 第 4.2 节介绍了 Flowprobe agent 和 vProbe 的设计, 第 4.3 节介绍了 Flowprobe controller 的详细架构.

4.2 Flowprobe agent 与 vProbe

Flowprobe agent 和 vProbe 的详细架构如图 6 所示.

从图中可知, Flowprobe agent 由 4 个模块组成, 分别是: 指令接收模块 (instructions receiving from controller)、探测包构造模块 (probing packet construction)、数据包注入和信息接收模块 (packet injection and information receiving) 以及探测信息上传模块 (probing information uploading). vProbes 则由两部分组成: 探测包识别与处理模块 (probing packet identification and processing)、探测信息上传模块 (probing information uploading). 下面就各个模块功能进行介绍.

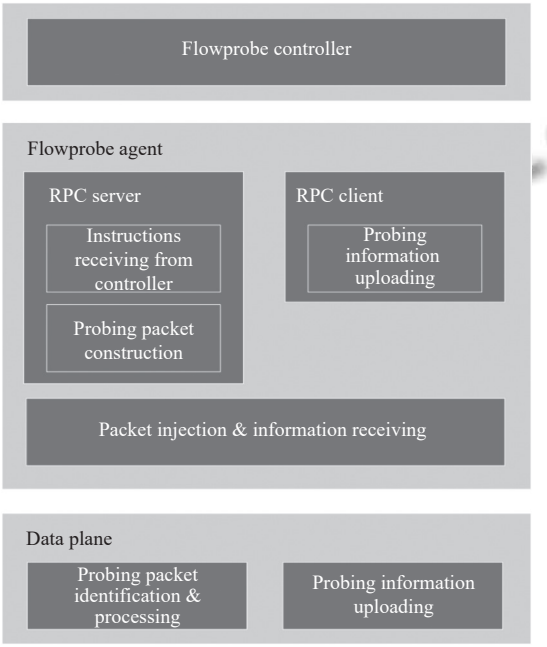

图 6 Flowprobe agent 和 vProbe 详细功能架构

指令接收模块 (instructions receiving from controller): 主要负责接收来自控制器的探测指令. 探测包构造模块根据这些指令来构造探测包. 这些指令包括探测包类型、探测包包头配置信息、探测包负载需要携带的指令信息、数据面 DPI 功能开启指令等. 当前探测包支持的数据包类型以及可配置项如表 1 所示. 通过丰富的可配置项支持, 构造的探测包可以最大程度地接近用户业务包, 从而保证探测包在虚拟网络中经历的网元行为、经过的路径等与业务包一致, 使得探测结果更加准确.

表 1 探测包类型及可配置项

探测包类型	可配置项
IP	IPv4 或者IPv6, 源IP地址, 目的IP地址, 负载大小
TCP	源端口, 目的端口
UDP	源端口, 目的端口
ICMP	类型, 代码 (code), 识别码 (identifier), 序列号

探测包构造模块 (probing packet construction): 该模块根据探测指令中的配置信息来构造探测包, 在设计系统过程中, 我们使用了开源的 Scapy 库[24] 来实现探测包的构造. 构造探测包的过程中要实现探测包的“染色”, 以使得数据面可以区分探测包与正常数据包. 数据面在高效快速地区分出探测包和正常数据包之后可以对探测包和正常业务包分别进行处理, 因此, 良好的探测包“染色”方案可以降低探测过程对数据面的转发性能的影响. 因此, 我们设计了一种创新的探测包“染色”方案来达到上述目的. 该方案将在第 4.2.1 节进行阐述.

数据包注入和信息接收模块 (packet injection and information receiving): 主要负责将探测包注入到数据面, 接收来自数据面的探测信息.

探测包识别与处理模块 (probing packet identification and processing): 该模块位于数据面, 主要区分探测包和正常业务包, 随之对探测包进行 DPI (deep packet inspection) 解析, 得到探测包负载 (payload) 中的探测指令, 根据指令收集探测包经历的网元转发信息, 并将这些信息进行格式化封装. 探测包识别和处理的详细流程将在第 4.2.2 节进行阐述.

探测信息上传模块 (probing information uploading): 该模块主要负责将探测信息向上进行传递. vProbe 中的该模块将探测信息通过 UNIX socket 传递给 Flowprobe agent. Flowprobe agent 中的该模块通过 RPC 将探测信息传递给 Flowprobe controller.

4.2.1 探测包染色方案

本小节将详细阐述探测包的“染色”方案. 如前所述, 探测包支持 IPv4 和 IPv6 两种类型, 因此, 探测包“染色”方案也分为两种. 如图 7 所示, 为 IPv4 探测包的包结构示意图. 我们选择 IPv4 包头中 DSCP (differentiated services code point, 区分服务比特) 位中的 2 个保留位 (reserved bits) 中的一个作为染色位, 具体而言, 我们选择最后一位作为染色位. 保留位的值通常为 0, 我们将其改为 1, 以标识该 IPv4 数据包位探测包.

IPv6 数据包头结构与 IPv4 的大不相同, 其中并没有 DSCP 位. 针对 IPv6 类型的探测包, 可用于标记探测包的方案有 3 种:

1) 使用特殊的 IPv6 地址.

2) 使用 IPv6 包头中 flow label 中的某一比特.

3) 使用特殊的扩展头.

但方案 1 有 IP 冲突的风险, 并且可扩展性差. 方案 2

中 flow label 位目前使用还不规范, 容易和外部物理网络起冲突. 因此, 综合可靠性和可扩展性, 我们使用 IPv6 扩展头中的比特位来作为染色标记. 具体而言, 我们使用 IPv6 扩展头中的 hop-by-hop 扩展头, 该类型扩展头的结构如图 8 中展示. Options 是可变长的, 通常必须是 64 比特的整数倍, 并且由 0 进行填充. 我们将填充位中的最后一个 0 变为 1 来作为探测包的染色位.

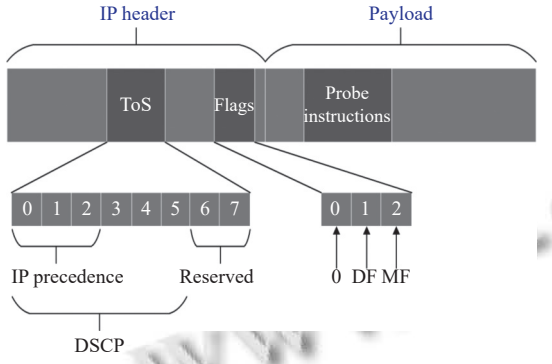

图 7 IPv4 探测包解构示意图

在 DPDK 转发中, 使用 rte_mbuf 存储数据包[25], 也即 DPDK 转发面需要首先解析 rte_mbuf, 再进行 rte_mbuf 中数据包信息的解析, 因此, 如果能提早识别出探测包 rte_mbuf, 那么就只需对探测包 rte_mbuf 进行信息解析. 图 9 是 DPDK rte_mbuf 的数据结构, 图中 mbuf struct 存储了该 mbuf 的元信息, 并且有一些自定义字段可供使用, 因此我们在 mbuf struct 中定义了一个字段来表明该 rte_mbuf 中的数据包是探测包, 可以对该 rte_mbuf 进行 DPI 操作.

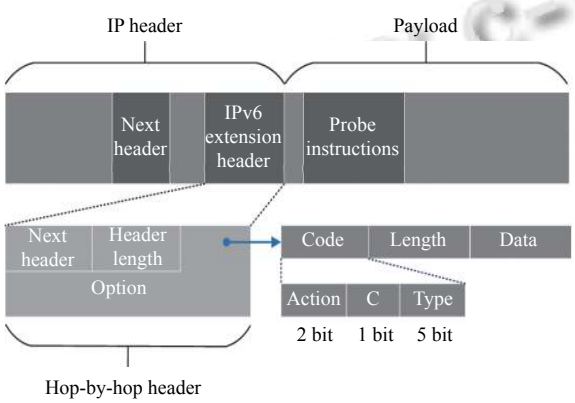

图 8 IPv6 探测包解构示意图

结合 DPDK 转发的特点, IPv4 及 IPv6 数据包的特点, Flowprobe 系统设计了上述探测包“染色”方案. 该染色方案可以使得数据面快速、准确地识别出探测包, 并在之后仅对探测包执行 DPI 操作, 以获得探测包负载中的指令信息. 这样, 最大程度地降低了对正常业务转发性能的影响.

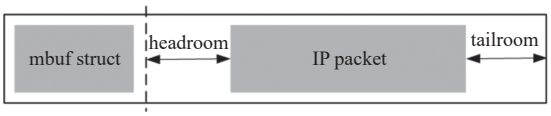

图 9 DPDK rte_mbuf 结构示意图

4.2.2 探测包识别与探测信息处理

探测包负载中包含的信息以及含义如表 2 所示.

表 2 探测包负载信息项及含义

负载信息项	字段含义
TenantID	开启探测的租户ID
TaskID	标识唯一探测的ID, 类型为UUID
DevName	探测包经过的虚拟设备名字
IfName	探测包经过的网卡名
HostIP	虚拟网络设备所在的主机IP
MsgID	消息ID, 数据面用来通知丢包的原因
PktIndex	探测包经过的虚拟网络设备的序号, 每经过一个虚拟网络设备加1
SeqNum	探测包的序列号, 用来区分多个探测包

表 2 中 MsgID 是数据包被丢弃原因的标识码, 通过查表可以得到数据包被丢弃的具体原因. PktIndex 记录了当前数据包经过的虚拟网络设备的个数, 探测数据包每经过一个虚拟网络设备该值加 1 并更新到探测包负载中的指令信息之中. 可以根据该值确定探测包经过各个虚拟网络设备的顺序, 并以此计算出探测包在转发面经历的转发路径. Flowprobe 系统在每次丢包探测过程中均会发送多个探测包, 以增强可靠性, 同时也方便计算丢包率. SeqNum 记录该探测包是一次探测中的第几个包, 该值用来区分上传的探测信息属于那个探测包.

探测包每经过一个虚拟网络设备, 位于数据面的 vProbe 即会记录该虚拟网络设备对探测包的操作行为 (actions) 以及上述探测指令信息, 然后将这些信息单独进行信息封装, 通过 UNIX socket 上传到 Flowprobe agent. 虚拟网络探测设备对探测包的行为主要包括接收、转发、送达等.

4.3 Flowprobe controller

Flowprobe 控制器的详细功能架构如图 10 所示.

由图 10 可知, Flowprobe 控制器主要由 8 个模块

组成, 各个功能模块的主要作用如下.

配置验证模块 (configuration validation): 该模块主要接收前端传递过来的探测配置, 包括探测包的配置信息, 发起探测的用户信息等, 并对这些配置信息进行合法性以及安全性校验.

接口处理模块 (interface handler): 该模块主要负责实现对外开放的接口, 如图 10 中的 UI 接口、CLI 接口、API 接口等.

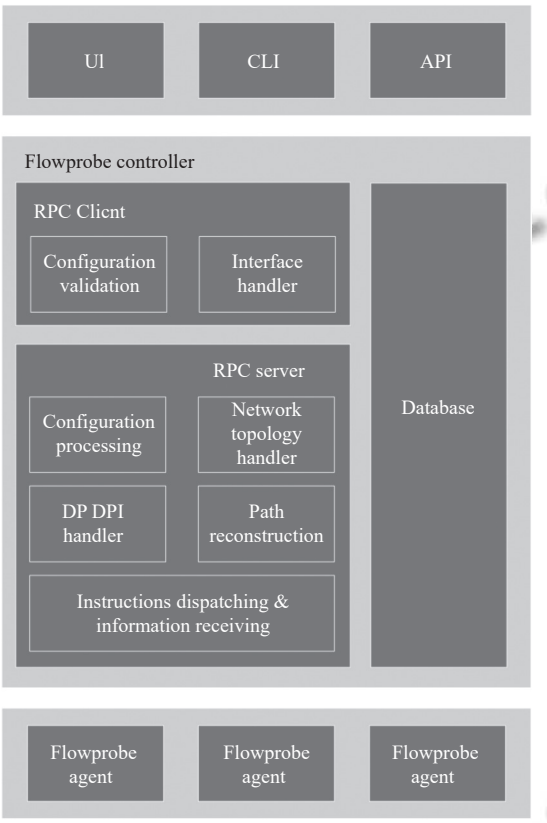

图 10 Flowprobe 控制器详细功能架构

配置处理模块 (configuration processing): 该模块主要负责对接收到的探测配置进行处理, 如格式转换、信息关联等. 从图 10 中可以看出, 配置验证和配置处理模块分别位于 RPC 客户端 (RPC client) 和 RPC 服务端 (RPC server), 这是因为 Flowprobe controller 做了异步处理设计, RPC 客户端中的配置验证模块在完成对配置的检查以后即将结果返回给前端, 若检查通过则告知用户等待探测结果, 若检查未通过则告诉用户校验输入参数并重新发起探测. 这种异步处理方式可以使得前端可以快速响应, 并实现前端展示与后端探测的解耦, 提升系统的模块化设计, 降低系统的耦合性.

网络拓扑处理模块 (network topology handler): 该模块主要负责与 SDN 控制器进行通信, 获得虚拟网络的拓扑结构, 以便于探测信息相结合进行探测路径的重建.

数据面 DPI 处理模块 (DP DPI handler): 该模块主要负责控制数据面 DPI 功能的开启和关闭. DPI 功能是会影响数据转发性能的, 因此不能长期开启, 需要在探测完成以后及时关闭数据面中的 DPI 功能.

指令分发和信息接收模块 (instructions dispatching and information receiving): 该模块负责将探测包配置信息、探测指令信息、数据面 DPI 控制指令信息等下发到各个 Flowprobe agent. 此外, 负责接收来自各个 Flowprobe agent 上传的探测信息. 该模块使用 Python 进行实现, 为了加快信息处理速度, 我们使用了 Python 的多进程多协程模型来实现高并发处理.

路径重建模块 (path reconstruction): 路径重建负责结合网络拓扑信息与上传的探测信息重建出探测包在虚拟网络中的转发路径. 路径计算方法如算法 1 所示.

算法 1. 路径计算算法

1) 根据 TenantID, TaskID 将探测信息进行分组聚合;
2) 根据 PktIndex, 虚拟网络设备转发行为, 网络拓扑信息构建虚拟网络设备的邻接矩阵;
3) 在虚拟网络设备构建的邻接矩阵上运行简单路径计算算法[26], 得到探测源虚拟机到目的虚拟机的完整路径.

值得一提的是, 在步骤 2 中, 我们根据虚拟网络设备的转发行为以及 PktIndex 来判断起始虚拟网络设备、终点虚拟网络设备以及网络设备之间的前后关系. 虚拟网络设备对数据包的行为有以下类型: 注入 (inject)、接收 (in)、转发 (out)、送达 (delivered)、丢弃 (drop). 其中执行“注入”这一行为的虚拟网络设备必是路径的首节点, 而“丢弃”和“送达”这两种行为对应的虚拟网络设备必是路径的末节点, 接收和转发行为为路径的中间节点, 对这些中间节点我们使用 PktIndex 来确定它们之间的先后顺序.

数据库 (database): 该模块主要用来保存探测配置信息、探测结果等. 前端在接收到 RPC 客户端的返回结果以后 (若通过配置检查, 则成功开启探测, RPC 客户端会返回给前端一个唯一的探测 ID, 即表 2 中的 TaskID), 会根据 TaskID 向数据库进行轮询, 一旦探测结果计算完成即可立刻返回结果.

5 系统功能和性能评估

Flowprobe 系统在深信服超融合计算产品中已经

使用了 3 年多的时间, 最新的版本已经落地到了深信服的超融合云计算产品 aCloud 之中. 该系统经过了严格的产品化测试, 针对 582 种虚拟网络持续丢包故障进行了场景化测试, 结果表明在 576 个场景中均可以实现持续性丢包检测. 接下来, 将在第 5.1 节对 Flowprobe 的功能进行验证, 在第 5.2 节对 Flowprobe 的性能进行评估. 功能验证是在一个由 3 台 HCI 设备构建的生产集群上进行的, 该集群内每一台 HCI 设备的配置是相同的, CPU 配置为: Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40 GHz, 24 核, 128 GB 内存. 性能测试是在一个额外的 HCI 主机上进行的, Flowprobe controller 和 agent 所在宿主机 CPU 配置为: Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40 GHz, 8 核, 32 GB 内存.

5.1 Flowprobe 系统功能验证

本小节将对 Flowprobe 的功能进行验证.

如图 11 所示为验证 Flowprobe 功能所用网络拓扑, 图中有 3 个虚拟机, 3 台虚拟交换机, 2 台虚拟路由器. 值得注意的是, 虚拟交换机具有分布式特性, 也即每一个虚拟交换机在每一个物理宿主机上都有一个实例, 这些实例之间使用 VXLAN 进行通信和状态同步. 图 11 中的拓扑视图由深信服超融合设备管理界面的“所画即所得”网络管理功能所得. 在该界面上可以通过对各个虚拟网元的拖动、以及连线快速构建出所需网络架构. 其中的“连通性探测”功能即是本文中所说 Flowprobe 系统在深信服超融合产品上的具体实现.

图 11 中 Centos7-Kafka、Centos7-MongDB、Centos7-Mysql 是 3 个业务虚拟机. 我们首先在网络正常的状态下由 Centos7-Mysql 向 Centos7-Kafka 发起 Flowprobe 探测 (连通性探测), 结果如图 12 所示. 从图 12 的结果中, 可以清晰地看到探测包在虚拟网络中的转发路径、各个虚拟网元的转发行为、各个虚拟网元所在的宿主机等信息.

接着, 去除边界路由器 1 中关于 Centos7-Mysql 虚拟机的路由, 再发起同样的探测, 结果如图 13 所示. 可以看到, 探测精准地定位到了出现故障的虚拟网元, 以及出现网络中断的原因: 虚拟路由器查询路由表失败.

最后, 我们恢复边界路由器 1 的路由, 将边界路由器 2 与交换机 2 的接口禁用, 再发起同样的探测, 结果如图 14 所示.

由图 14 可见, Flowprobe 系统亦可精准地定位出出问题的路由器以及网络中断的原因.

上述实验仅是 Flowprobe 可探测的持续性丢包或网络中断场景中的一小部分, 严格的产品化测试流程表明, Flowprobe 系统可以对 576 种虚拟网络持续性丢包或者网络中断场景进行探测与故障定位.

5.2 Flowprobe 系统性能评估

本小节将对 Flowprobe 系统的性能进行评估. 图 15、图 16 分别是 Flowprobe controller 和 Flowprobe agent 的 CPU 资源消耗, 由图可知, Flowprobe controller 的总体 CPU 资源消耗约在 2.5%, Flowprobe agent 的 CPU 资源利用率小于 0.2%.

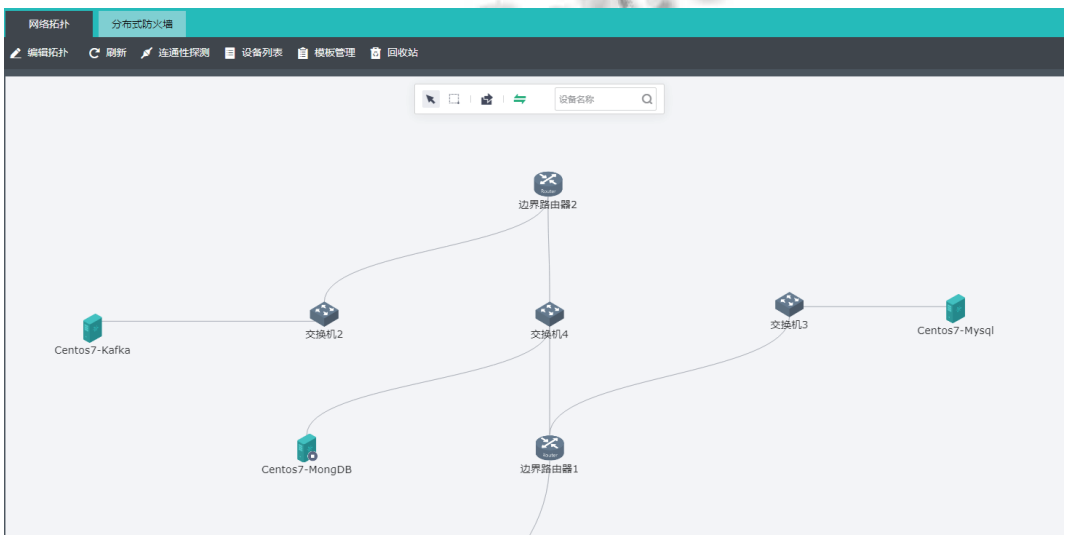

图 11 Flowprobe 功能验证拓扑

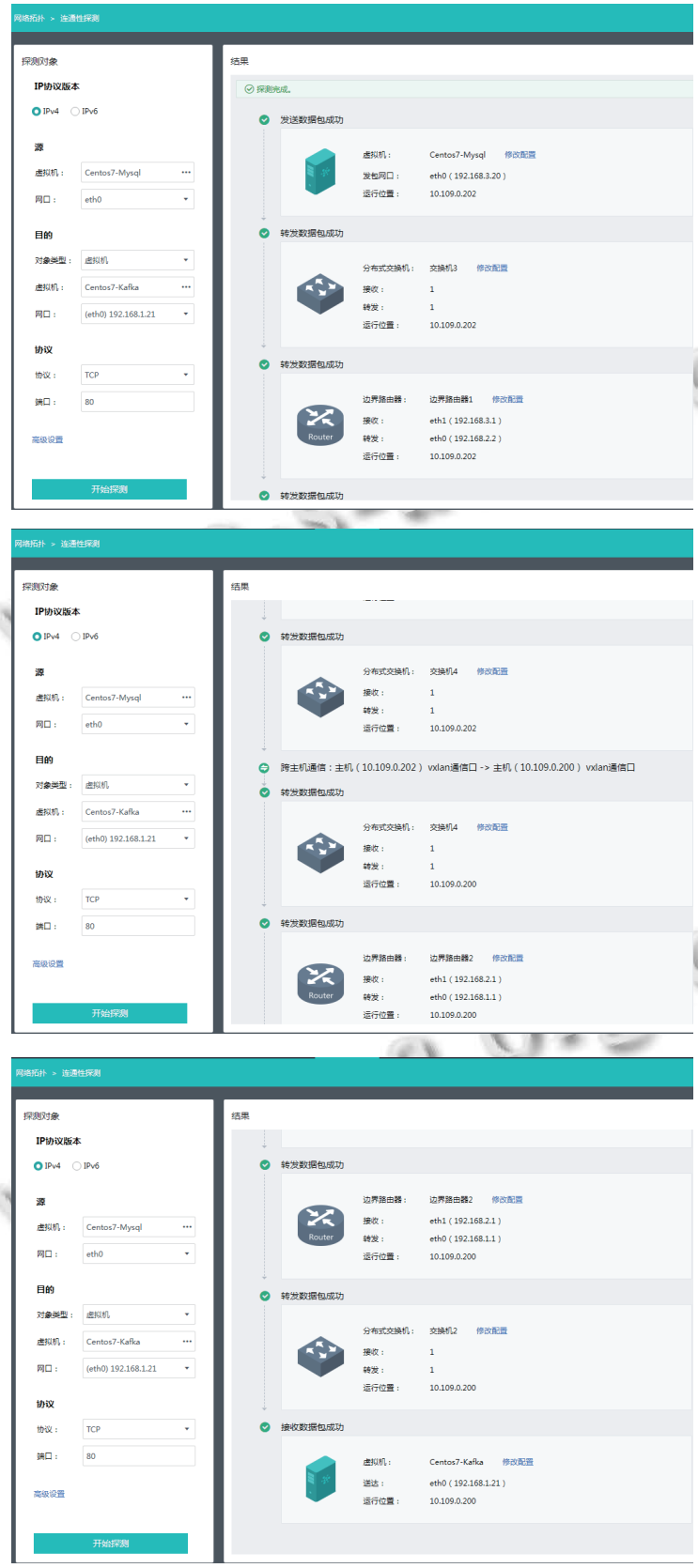

图 12 Centos7-Mysql 向 Centos7-Kafka 发起 Flowprobe 探测结果 (正常)

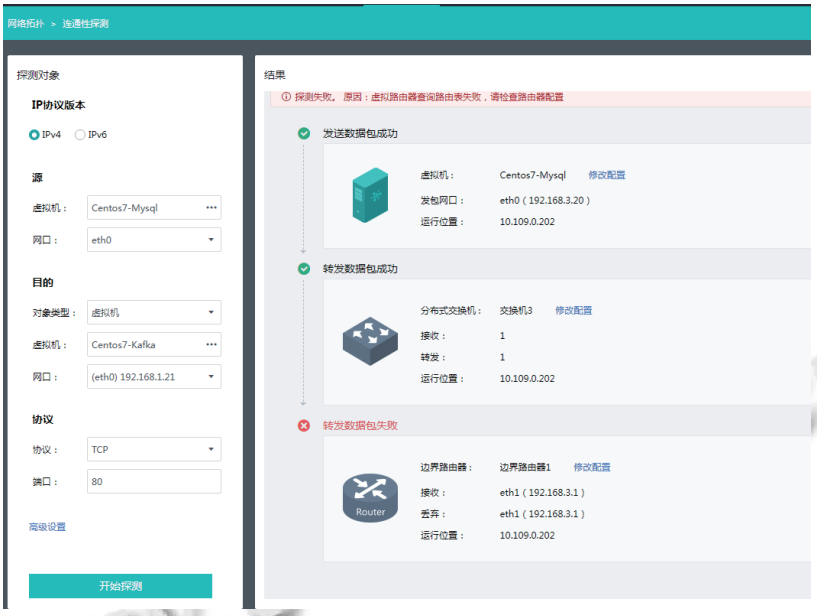

图 13 Centos7-Mysql 向 Centos7-Kafka 发起 Flowprobe 探测结果 (边界路由器 1 失败)

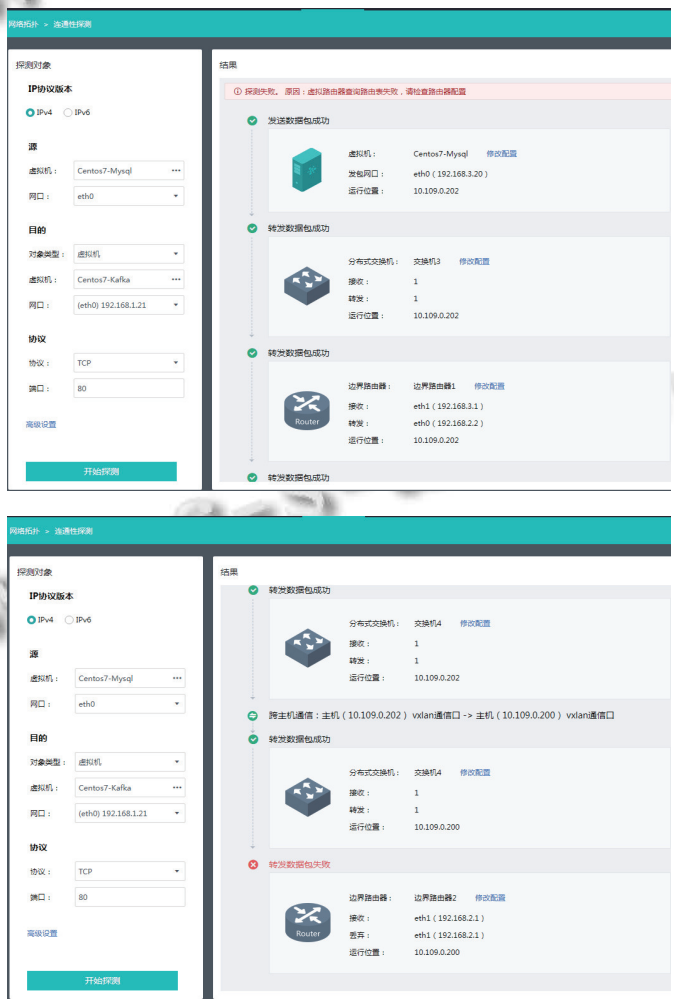

图 14 Centos7-Mysql 向 Centos7-Kafka 发起 Flowprobe 探测结果 (边界路由器 2 网口禁用)

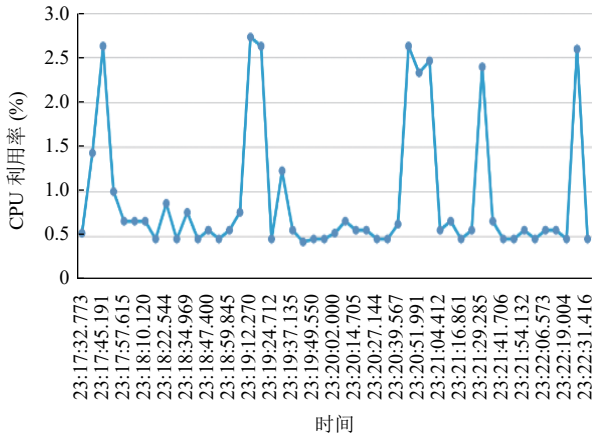

图 15 Flowprobe controller CPU 资源消耗

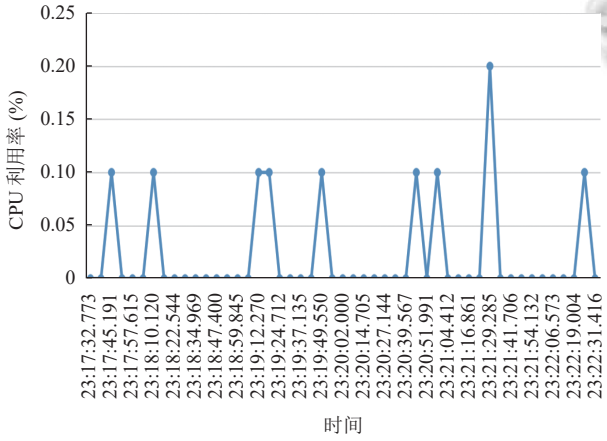

图 16 Flowprobe agent CPU 资源消耗

图 17、图 18 分别是 Flowprobe controller 和 Flowprobe agent 的内存资源消耗. 从图中可知, Flowprobe controller 和 Flowprobe agent 的内存占用分别约为 96 MB, 48 MB.

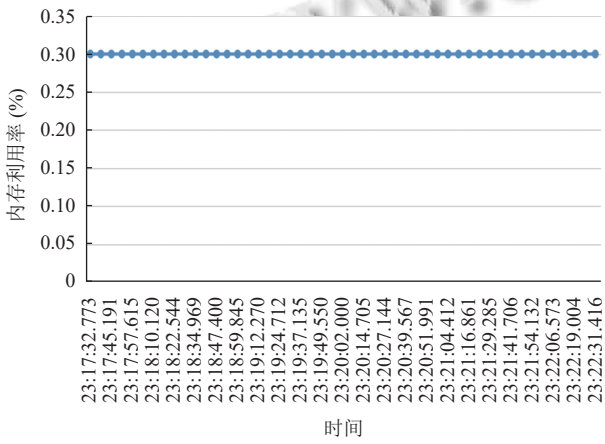

图 17 Flowprobe controller 内存资源消耗

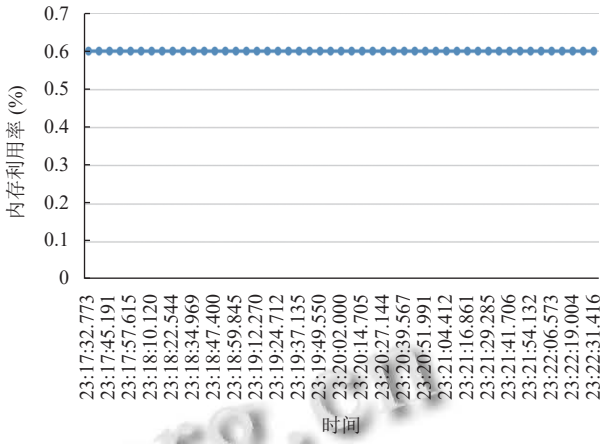

图 18 Flowprobe agent 内存资源消耗

表 3 Flowprobe 开启前后 DP 吞吐量对比

帧大小 (b)	吞吐量(Gb/s, 开启 Flowprobe前)	吞吐量(Gb/s, 开启 Flowprobe后)
64	3.5593	3.5593
256	9.8571	9.8571

除了资源消耗以外, 我们还利用思博伦根据 RFC2544 测试了 Flowprobe 功能开启前和开启后 DP 吞吐量的变化, 表 3 显示了测试结果. 由结果可见, 开启 Flowprobe 前后, DP 的最大吞吐量没有发生变化. 这其实与我们精心设计的数据包标记方案有关, 设计的标记方案最大程度上地减少了 DPI 的执行次数, 尽最大可能地降低了对 DP 正常转发性能的影响.

6 结论与展望

本文设计了一种虚拟网络持续性丢包探测系统, 该系统旨在解决基于 DPDK 用户态虚拟网络的持续性丢包检测及根因定位问题. 文中详细描述了该系统的设计思想. 为了减小探测对正常转发的影响, 我们提出了一系列的优化措施, 如精心设计的探测包标记方案, 路径重建算法等.

该系统未来将持续进行功能和性能的优化, 后续将尝试将该系统应用到安全运维领域, 实现对网络功能服务链的转发路径验证功能. 另外, 基于该系统实现虚拟网络转发过程中逐链路时延的探测等.

参考文献

1 Wikipedia. Hyper-converged infrastructure. https://en.wikipedia.org/wiki/Hyper-converged_infrastructure. (2021-12-03)[2021-12-28].

2 Adam A, Ilan A, Nadeau T. Introduction to virtio-networking and vhost-net. https://www.redhat.com/en/blog/introduction-

virtio-networking-and-vhost-net. (2019-09-09)[2021-12-28].
3 Tu W, Wei YH, Antichi G, *et al*. Revisiting the open vSwitch dataplane ten years later. Proceedings of the 2021 ACM SIGCOMM 2021 Conference. New York: Association for Computing Machinery, 2021. 245–257.
4 Intel. Data plane development kit. https://www.intel.com/content/www/us/en/communications/data-plane-development-kit.html. (2020-07-30)[2021-12-28].
5 Mellanox, Hat R. Mellanox and Red Hat deliver enhanced performance and simplicity for NFV infrastructure and agile cloud data centers. https://www.businesswire.com/news/home/20180508005635/en/Mellanox-Red-Hat-Deliver-Enhanced-Performance-Simplicity. (2018-05-08).
6 Open Networking Foundation. Software-defined networking (SDN) definition. https://opennetworking.org/sdn-definition/. (2020-06-03)[2021-12-28].
7 Open vSwitch. Implementation details. https://docs.openvswitch.org/en/latest/faq/design/. (2021-06-11)[2021-12-28].
8 Cisco. Inband network telemetry. https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/92x/programmability/guide/b-cisco-nexus-9000-series-nx-os-programmability-guide-92x/b-cisco-nexus-9000-series-nx-os-programmability-guide-92x_chapter_0100001.html. (2020-03-16).
9 Tan LZ, Su W, Zhang W, *et al*. In-band network telemetry: A survey. Computer Networks, 2021, 186: 107763. [doi: 10.1016/j.comnet.2020.107763]
10 Huawei. Overview of iFIT. https://support.huawei.com/enterprise/en/doc/EDOC1100143222/aaf8276b/overview-of-ifit. (2020-06-09)[2021-12-28].
11 OpenTelemetry. What is OpenTelemetry? https://opentelemetry.io/docs/concepts/what-is-opentelemetry/. (2021-12-21)[2021-12-28].
12 OpenTracing. What is distributed tracing? https://opentracing.io/docs/overview/what-is-tracing/.(2020-12-02)[2021-12-28].
13 Song H, Zhou T, Li ZB, *et al*. Network telemetry framework. https://tools.ietf.org/id/draft-song-opsawg-ntf-02.html. (2018-12-14).
14 Prabowo RJ, Hidayanto AN, Sandhyaduhita PI, *et al*. The determinants of user's intention to adopt hyper-converged infrastructure technologies: An integrated approach. 2018 International Conference on Information Technology Systems and Innovation (ICITSI). Bandung: IEEE, 2018. 306–311.
15 Wolf C. Hyper-converged is more than an architecture: It's a lifestyle. https://virtualizationreview.com/articles/2016/04/04/hyperconverged-infrastructure-is-a-lifestyle.aspx. (2016-04-04).
16 Zhu YB, Kang NX, Cao JX, *et al*. Packet-level telemetry in large datacenter networks. Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. London: Association for Computing Machinery, 2015. 479–491.
17 Hu XH, Xiang Y, Li YF, *et al*. Trident: Efficient and practical software network monitoring. Tsinghua Science and Technology, 2021, 26(4): 452–463. [doi: 10.26599/TST.2020.9010018]
18 Huawei. Licensing requirements and limitations for IOAM. https://support.huawei.com/enterprise/en/doc/EDOC1100138141/a36f6d78. (2021-11-06)[2021-12-28].
19 Song H, Zhou T, Li Z, *et al*. Postcard-based in-band flow data telemetry. https://tools.ietf.org/id/draft-song-ippm-postcard-based-telemetry-04.html. (2019-12-04)[2021-12-28].
20 Fang CR, Liu HY, Miao M, *et al*. VTrace: Automatic diagnostic system for persistent packet loss in cloud-scale overlay network. 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: Association for Computing Machinery, 2020. 31–43.
21 Gulenko A, Wallschläger M, Kao O. A practical implementation of in-band network telemetry in open vSwitch. 2018 IEEE 7th International Conference on Cloud Networking (CloudNet). Tokyo: IEEE, 2018. 1–4.
22 Basat RB, Ramanathan S, Li YL, *et al*. PINT: Probabilistic in-band network telemetry. 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: Association for Computing Machinery, 2020. 662–680.
23 Zhao YK, Yang KC, Liu ZR, *et al*. LightGuardian: A full-visibility, lightweight, in-band telemetry system using sketchlets. 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21). Boston: USENIX, 2021. 991–1010.
24 Scapy Project. What is scapy? https://scapy.net/. (2021-12-19)[2021-12-28].
25 Intel. Mbuf library. https://doc.dpdk.org/guides-1.8/prog_guide/mbuf_lib.html. (2014-12-20)[2021-12-28].
26 Sryheni S. Find all simple paths between two vertices in a graph. https://www.baeldung.com/cs/simple-paths-between-two-vertices. (2020-10-19).

(校对责编: 孙君艳)