doi:10.11887/j.cn.201806010 http://journal.nudt.edu.cn

用户意图的网络边缘主动计算方法*

吕高锋,孙志刚,李 韬
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:互联网端到端的设计原则没有区分端系统和网络间接口,使得用户分组处理期望无法告知网络,降低了网络效能。首次提出用户转发意图概念,端系统凭此向网络通告其意图,并借助网络边缘计算能力得以执行,实现用户意图在网络节点转发决策中的主动使能,构成新型主动表意网络i-ECAN。主动表意网络可以在视频直播加速、物联网数据融合、端系统安全认证等场景中得到应用。基于CORE模拟器构建网络演示环境,对无线网络切换和基于主动表意网络的无线网络切换的延迟进行比较,证明了主动表意网络机制的合理性和可行性。主动表意网络扩展了新的端系统和网络之间的接口,为人-网协同提供了基础。

关键词:转发意图;主动表意;边缘计算
中图分类号:TP393 **文献标志码**:A **文章编号**:1001-2486(2018)06-068-07

Active computing method in network edge for user intention

LYU Gaofeng, *SUN Zhigang*, *LI Tao*
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: The internet principle of "end-end" cannot distinguish host-network interface, so the intent of the user cannot be aware by the network, which decreases the efficiency of the transmission. The novel edge computing-based active network especially for user intent, i-ECAN, was proposed, and the concept of the user forwarding intent was designed, which was utilized to inform the network the intent directly by the end system and was executed on the basis of the resource of edge computing. The i-ECAN can embody the user's intention in the forwarding decision of network nodes and be used in mobile communication, video broadcast, and data fusion in internet of things. In the wireless network based on CORE simulator, the switching time of normal wireless network and wireless network based on i-ECAN was compared, which proves the rationality and the feasibility of i-ECAN. The i-ECAN extends the interface between the end system and the network, which enforces the collaboration between machines and networks.

Key words: forwarding intention; active network for user intention; edge computing

目前,网络对于应用而言是数据传输管道,而应用对于网络而言则是无差别的数据分组,这种网络-端系统透明的设计原则支持了网络应用的快速发展,也支撑了网络规模的不断扩展[1]。为了提高网络传输效率、降低网络运维成本,研究人员提出多种应用感知方式,其中一种是网络被动分析应用(如应用代理、L4转发等),用来优化基于目的地址的转发策略,另一种是应用主动探测网络(如P2P和应用层组播(Application Layer Multicast, ALM)等),由端系统选择目标或调整发送速率。网络(运营商)和应用(用户)以私有信息来决策,试图实现各自利益最大化。网络博弈并没有实现双赢[2],有时候还造成了双方利益损失,如BT文件共享。网络(运营商)和应用(用户)缺乏沟通渠道,缺少共享信息,从而做出不合理的转发决策和路径选择,最终导致BT边缘化。

软件定义网络(Software Defined Network, SDN)采用了转发和控制分离的思想[3-5],提供了管理员和网络之间的控制接口,但还是没有考虑用户和网络之间的接口。Fabric[6]认为目前互联网没有区分Host-Network和Packet-Network接口,如仅仅在报文中通过服务类型(Type of Service, ToS)域向网络通告用户服务质量需求,故采用了Host-Network、Packet-Network和Operator-Network接口等概念。多协议标签交换(Multi-Protocol Label Switch, MPLS)采用了转发等价类和标签的概念,而标签是网络管理者在构建虚拟私有网络(Virtual Private Nework, VPN)等隧道时的一种管理控制分组的方式。基于应用-网络接口,应用能够认识网络,合理分配资源,更

* **收稿日期**:2017-10-24
基金项目:国家自然科学基金资助项目(61702538,61301236)
作者简介:吕高锋(1980—),男,陕西扶风人,副研究员,博士,E-mail:gflv@163.com

高效地利用网络资源,实现快速转发。网络需要根据用户需求来提前规划,并合理分离资源,避免网络热点的形成等。另外,主动网络[7-8]采用了用户自定义代码,然而,网络运营商并不能根据用户的表述来进行决策,只是被动执行。

针对上述问题,本文提出了显式地表示端系统或用户意图的方法,用高级语言对用户的意图进行抽象,网络来理解意图,并根据本地策略,在意图的指导下做出用户友好的分组转发决策。用户主动表意不仅能够向网络提供需求信息,还能够支撑运营商根据自己的策略进行适当的调整,而不是照搬规范执行。

与主动网络相比,用户主动表意不是通过命令式直接在网络节点中运行。主动网络以直接命令方式运行,对运营商来说是黑盒子,用户占用多少资源无法提前预估,也无法在线干预用户执行行为,不利于运营商的管理。而主动表意提供用户或端系统的意图,网络运营商根据策略再做出用户友好的决策,另外,用户或端系统的意图采用高级语言形式表示,在网络运营商处汇聚,运营商可以进行多目标的融合、冲突目标的消解等,从而做出全局优化。

1　相关研究

互联网发展关键问题是下一步技术路线选择,SDN 避开直接解决这个问题,而是以转发与控制分离的松耦合架构来支撑网络规模扩展和控制能力的提升,特别是网络功能虚拟化(Nework Function Virtualization, NFV)概念提出后,其网络功能实现又得到增强。SDN 技术实际上扩展了网络纵向管理员与网络之间的接口。而在网络横向,Fabric 采用了边缘和核心分离,扩展了端系统和网络的接口,但是还没有到人 - 网协同的程度。网络横向接口的定义和认可,与纵向接口定义的意义同等重要,将推动网络功能和应用的快速发展。

1.1　网络开放可编程

思科和华为等商用路由器、交换机网络设备厂商根据协议标准从硬件平台到软件系统全部研发,给网络运营商提供了管理配置接口,即保证网络设备能够互连、协议能够互通,从而实现组网运行。然而,传统网络设备功能难以升级,为了支持新的协议,只能由设备厂商来设计开发,并测试部署,网络运维成本高。

斯坦福大学 Clean Slate 项目提出了转发与控制分离的思想,设计了网络领域 x86 指令集,即数据平面抽象 Openflow[3]。在软件定义网络架构下,网络设备只需实现数据平面分组转发处理,并提供编程接口,网络控制器运行网络协议对网络设备进行配置管理。软件定义方式促进了 Pica8、Big Switch 等白盒网络设备的研制,以及 Floodlight、开放网络操作系统(Open Network Operation System, ONOS)等开源控制器的设计。

然而,对于负载均衡器、防火墙、入侵检测和安全防御等网间设备来说,其网络分组处理行为复杂,管理策略各不相同,无法抽象成统一的模型,不能用同一个硬件平台来实现[9]。随着中央处理器(Central Processing Unit, CPU)性能的提高、系统总线带宽的提升,基于虚拟机来运行复杂网络功能,即网络功能虚拟化成为可能[10]。NFV 将网络功能与平台实现解耦,将网络功能以软件形式实现,将网络设备硬件平台化。NFV 简化了网间设备的设计开发和管理,并大幅降低了网络运营商的运维成本[11]。在网络边缘,NFV 能够为智能终端提供计算和存储资源,形成移动边缘计算[12-13](Mobile Edge Computing, MEC)。

1.2　网络领域高级语言编程

在 SDN 架构下,白盒交换机“white box”以低廉的成本为网络运营商提供了足够大的灵活性,获得了快速发展。网络运营商和服务提供商等根据自己需求也研制白盒交换机,如 Facebook 和 Google 等。服务提供商不再采购通用交换机,而是采用自顶向下的设计方法,由应用驱动硬件设计,即根据需求来定制硬件。

Barefoot 提出了一种协议独立的交换架构[14],采用多级流水架构,每级集成了可配置的解析器、可配置的匹配 - 动作引擎,通过灵活配置,实现不同的协议和不同的网络功能[15]。为此,其还设计了一种面向网络领域的高级编程语言 P4[16],设计了解析器、表、动作和控制流等关键字词。利用 P4 很容易编写程序,实现对 Barefoot 交换芯片的配置,从而实现不同的功能[17]。P4 不局限于 Barefoot 交换芯片,有研究者基于 P4 设计开发了软件交换机[18]。

现场可编程门阵列[19](Field Programmable Gate Array, FPGA)也是一种可重构的芯片。Xilinx 和 Altera 作为 FPGA 的主要厂商,也纷纷推出了高级编程语言编译器 PX[20] 和 OpenCL[21],能够把高级语言程序映射到 FPGA 硬件逻辑,简化并加速基于 FPGA 的网络功能的设计开发。Microsoft 在 ClickNP[22]项目中基于 FPGA 实现了基本的网络功能,如最长前缀匹配、Hash 查表、三

态内容寻址存储器(Ternary Content Addressable Memory, TCAM)和校验等,认为性能可满足高性能网络处理要求。

1.3 主动网络

Tennenhouse 等于 1996 年提出主动网络概念[7],在网络设备中集成执行环境来运行虚拟机,对到达的主动分组进行解释执行。在不改变网络整体架构的情况,用户通过部分主动节点获得网络提供的自定义的服务,从而满足用户的特殊需求。基于主动网络模型,可以实现可靠组播服务、服务质量保障机制和流量缓存机制等。

主动网络通过主动分组携带的代码在执行环境中运行,对分组进行特殊处理,实现增值服务。一方面,主动分组代码执行时间不可预估,可能是简单的报文头校验或者查表,也可能是复杂的视频流编解码程序,这会对需要线速处理的网络节点造成很大影响,故厂商支持的积极性不高;另一方面,随着网络协议的发展,网络节点功能越来越完善,基本可以满足用户数据传输需求。因此,主动网络最终没有得到部署。

在 20 多年前,主动网络已经采用了执行环境的概念来运行虚拟机,是最早的网络虚拟化方法,与当前网络功能虚拟化的概念非常相似,只是网络功能扩展的提供者不同,前者是端系统或用户,而后者是网络运营商。

1.4 用户意图及其与网络的接口

互联网中端系统将分组交付给网络,网络逐跳转发到目的端系统。应用和网络之间透明,既保证了应用有很好的移植性,又保证了网络有良好的扩展性。应用和网络之间透明也带来诸多问题,如:应用无法探测和诊断网络故障、网络无法对应用传输做出合理的优化。Fabric[6]采用的分组协议头只是分组和网络转发之间的接口,网络管理协议是管理员和网络之间的接口,没有设计用户和网络之间接口。Fabric 主要将网络分解为边缘和核心,两者分别控制,网络核心是交换矩阵,边缘网络通过各种服务原语接入网络核心。

软件定义网络中控制器提供了网络数据平面抽象,为网络应用提供了控制平台。ONOS 控制器提出了意图的概念,将网络应用的意图转化为管理策略,再通过控制器将管理策略转换为数据平面的转发规则。

2 主动表意网络模型设计

为了表示用户或端系统转发意图,设计了转发意图原语,以数据分组隐式携带或专有协议显式通告给网络。在网络中以边缘计算存储等资源为基础,通过解释器执行转发意图,来影响分组处理流程和转发决策,形成基于边缘计算的主动表意网络模型(Edge Computing-based Active Network for user intention, i-ECAN)。

2.1 用户转发意图表示

转发意图表示端系统或用户的分组转发的期望,由基本意图原语构成。

转发意图中基本原语主要分为三类:①用户需求型,主要表示分组处理方法,如优先级、带宽、延迟和丢弃阈值等期望,是对基本分组转发处理的修正;以及用户数据的深度处理,如数据压缩、内容缓存和更新等,是网络面向用户提供的服务。②网络属性获取型,如延迟记录、队列长度和跳数等网络状态,向用户提供类似于管理员的简单网监机制。基本的用户需求型和网络属性获取型转发意图可以以意图原语集合和参数的形式描述。通过定义的意图原语直接与网络交互,指明分组处理方式,参数可采用偏移长度值(Offset Length Value, OLV)形式来表示,其指明了分组选项。③服务型,以高级语言形式进行描述。目前,可以采用 P4 语言规范,P4 是一种面向网络领域的高级语言[16],可以定义协议、分组处理和控制流等。由于转发意图涉及分组处理状态,还需要对 P4 语言进行扩展,使之支持状态的处理。与主动网络相比,P4 语言的执行最终由网络中解释器决定,行为是可预估的。

2.2 主动表意网络模型

主动表意网络模型由三层组成,如图 1 所示。

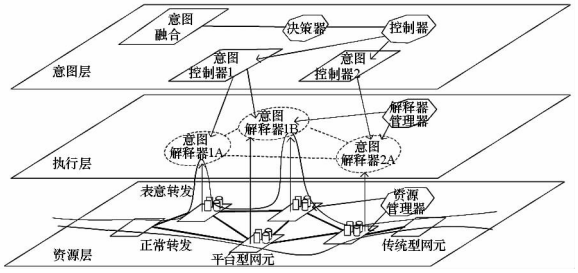

图 1 主动表意网络模型

Fig. 1 Model of i-ECAN

资源层:主要包括计算、存储和网络等资源,是基本的分组处理和转发意图解释执行的基础设施。

执行层:主要包括解释器,解释执行转发意图,并对基本的分组处理流程和转发决策进行局部控制。

意图层:转发意图的控制层,判断用户的转发

意图是否被执行。

主动表意网络中,用户或端系统向网络(控制器)通告以高级语言形式描述的转发意图,端系统和用户的意图可在数据分组中携带,也可通过专有协议方式发送给控制器。网络接收到报文后,根据控制器决策,直接运行或者部分执行报文的转发意图,从而在转发决策中体现用户意愿。对于传统网络设备,不理解转发意图选项,可以不做处理。

2.3　原语解释及执行

转发意图解释器:依据网元计算能力,对转发意图中原语进行解释执行,从而达到或部分完成端系统或用户的分组处理需求。

解释器根据转发意图中基本原语进行执行,可以是软件程序,也可以是软硬件协同处理模块。类似于主动可编程网络,报文经过基本的网络处理后,解释器从报文中获取转发意图原语集并执行。执行结果可以是直接转发,也可以作为正常转发的决策依据,从而在转发过程中嵌入端系统或用户的意图。

对于以软件程序为载体的解释器,资源层提供的计算、存储和网络等资源为其运行提供平台支撑。类似于 Java 在虚拟机解释执行,转发意图也可以在网络功能解释器中执行。目前,可以借用 P4 语言中行为模型[17](Behavioral-Model,BM),增加软件交换机用户定义的操作来实现转发意图。在报文处理之前,根据转发意图对行为模型进行配置,然后,软件交换机再对报文进行处理,实现用户定义的功能。

在执行层多个转发意图解释器并行执行。为了保证解释器安全可靠运行,可以采用虚拟机方式,在每个虚拟机中运行解释器(该方法安全性好、性能较低),也可以采用容器方式。如图 2 所示,回调微程序对分组进行原语级处理。Docker 等容器中运行解释器,运行空间有一定的隔离,性能损失较少。

对于基于硬件模块的解释器,硬件处理引擎可以作为微程序的加速引擎,对软件处理中部分功能进行硬件加速处理,通过软硬件协同实现转发意图。另外,可重构硬件架构[14-15],如协议无关交换架构(Protocol-Independent Switch Architecture,PISA),也可作为最基本的数据平面,通过高级语言进行定制和扩展功能。

当采用类似于行为模型和 PISA 的架构时,需要事先进行配置。可以先缓存可重构解析器的配置,当报文到达时经过匹配查找直接进入已经配置好的解释器。解释器在处理报文的过程中可以修改 metadata 以及标志位来表示报文处理状态,以供后续解释器处理。

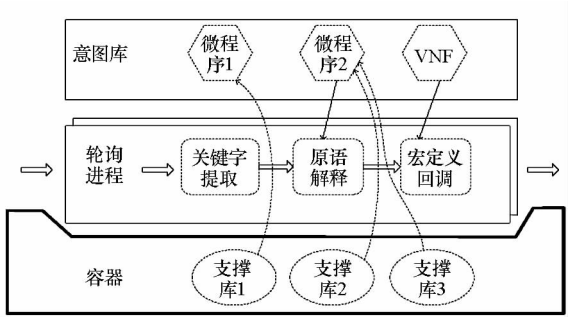

图 2　转发意图原语处理方法

Fig. 2　Execution of forwarding intention

由于意图语言描述能力的制约,用户的表意程序不再是任意的二进制程序,另外,网络应用 L2 – 7 层处理流程基本固定,因此用户意图原语集的合法性验证将会更简单。解释器负责对转发表意程序的正确性进行检查,在有限的关键字和控制流下,保证用户的转发表意程序不会引起系统性错误,不会影响其他用户程序的执行。另外,解释器还统计了用户程序的执行时间,确保流的处理时间不超过预约的配额,不会独占资源。

2.4　主动计算网元

主动表意网络资源层包含通用的传统型网元和平台型网元。

平台型网元:主动表意网络重要组成,不仅具有基本的网络转发功能,还具有计算和存储等资源,支持用户转发意图原语解释和执行。

传统型网元:已有的传统网络设备,主要负责网络分组的转发处理。

主动表意网络采用边缘和核心分离的设计思想。平台型网元通常部署在网络边缘,与端系统所在子网直连,作为第一跳来处理用户转发意图,实现用户期望的操作。传统型网元部署在网络核心,作为交换矩阵,主要实现平台型网元的互连互通。

在主动表意网络中,平台型网元提供了计算(含加速器等)、存储和网络等异构资源,主要采用计算机虚拟化技术[9],如虚拟机和容器技术,对资源进行抽象,将异构资源转化为能力池,并以资源片的形式提供给用户使用。

由于平台型网元位于边缘,主动表意网络形成边缘计算能力[11-12],提供给网络应用使用。平台型网元增强了本地的网络处理功能,同时,还与已有网络兼容,不影响其他用户正常使用,支持渐进式部署。

2.5 网络管理控制

主动表意网络的管理控制是通过三层管理单元垂直管理来实现的,分别是资源层的资源管理器、执行层的解释管理器以及意图层的控制器。

1)控制器:收集到转发意图,并根据功能测试结果判断是否给转发意图分配资源。

转发意图在网元中的执行过程中占用网元的资源,改变了传统的转发模型。当网元接收到携带转发意图的报文时,如果不支持,则按照默认的通用转发模型;如果支持转发意图处理,但是还未给转发意图分配资源,则需要将转发意图的请求发送给意图层控制器,让控制器决策是否执行该转发意图。

控制器根据资源层提供的能力,决定是否分配解释器来执行用户的转发意图。在资源层和执行层解释器功能都满足的情况下,控制器给该转发意图分配解释器,该流后续报文都将通过解释器来执行。转发意图的执行依赖于 SDN 控制器获取全局信息,需将解释器下载到执行平面进行执行。

2)解释管理器:根据网络控制器的命令,加载或释放解释器,对解释器的运行状态进行监控,进行解释器全生命周期管理。

根据用户协议,特别是对于性能和服务质量等需求的约定,解释管理器一方面可根据实时监测的性能结果,动态复制解释器,通过并行处理方式实现用户解释器执行性能的提高;另一方面,也可以根据流量,动态撤销解释器,实时缩减处理能力,节省资源。

3)资源管理器:对平台型网元的资源使用状态进行管理控制。

主动表意网络模型中资源层由传统型网元和平台型网元构成。传统网络设备和平台型网元都可以通过简单网络管理协议进行管理。另外,资源管理器主要是向控制器反馈资源层的状态以及资源的利用率等信息。

资源管理器还需要和通用网络设备管理器进行协调。在资源层,平台型网元的流量是通过传统网络设备导入的,因此,在流转发和路由上,资源管理器要和通用网络设备交互,根据网络拓扑配置相应的转发路径。

3 主动表意网络设计及应用开发

3.1 基于 FAST 的主动表意网络节点设计

在主动表意网络中,平台型网元不仅具有一定的计算能力支持应用加速,还能够根据用户转发意图对转发进行定制。由于基于软件的解释器处理耗费资源且时间长,为了提高性能,可以采用 NFV 中的加速方法[10],如单根虚拟化技术和数据平面开发套件(Data-Plane Development Kit, DPDK)等。

平台型网元处理过程中,涉及硬件加速器、软件解释器和操作系统容器等多方处理逻辑。解释器在网络开放开发平台 FAST(FPGA accelerated switch technology)上采用 Labelcast 处理模型来实现,将分组处理中通用功能下载到 FAST 平台的硬件进行处理。基于 FAST 平台 FPGA 硬件可以对解释器中公共的通用的分组处理进行加速,如分组解析、元组匹配、协议分析等处理。硬件处理后,在分组中增加 metadata 表示处理结果,解释器再根据硬件处理中间结果运行微程序来完成后续处理,如图 3 所示。

在网元处理过程中,一种加速方式是转发意图中宏定义映射到二进制代码,在虚拟机或容器中执行,这需要将报文重定向到虚拟机或容器。在 FAST 平台中,硬件加速器和 CPU 之间实现了轻量级直接存储器访问(Direct Memory Access, DMA)机制,实现数据快速收发,并支持在解释器和容器之间快速交换。

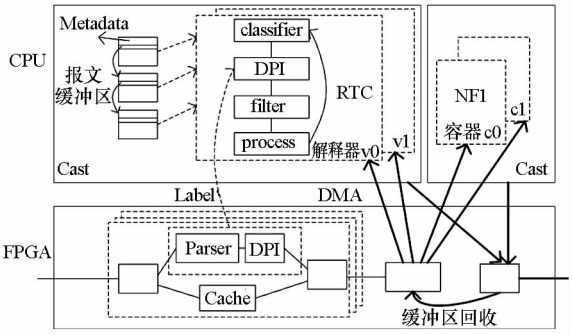

图 3 转发意图原语处理加速模型

Fig. 3 Acceleration of forwarding intention

另一种加速方式是硬件解释器方法,即是在 FAST 硬件直接实现可重构的网络分组处理器,作为解释器,加速转发意图的执行。

3.2 应用开发模型

在主动表意网络模型中,为应用开发提供了向网络表示转发意图的接口,用户可以将自己的需求通过该接口发送给网络。在 socket 层增加了转发意图应用编程接口(Application Programming Interface, API),如图 4 所示。

通过转发意图 API,用户的位置管理、状态控制、通信协议、内容处理方式和 QoS 等需求以转

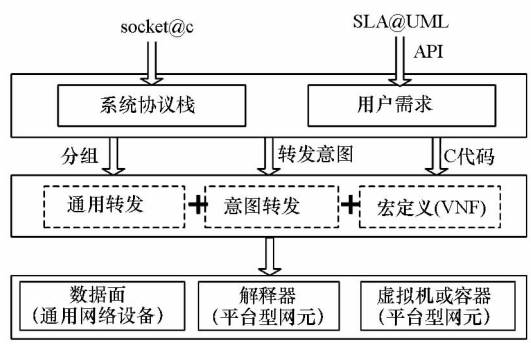

图 4　主动表意网络应用开发模型

Fig. 4　Application development for i-ECAN

发意图原语集的形式嵌入到分组中,网元通过解析分组,获取用户转发意图,在转发过程中根据转发意图来决策;或通过专用协议帧通告给网络,控制器配置解释器。另外,部分定制的转发意图以宏定义方式实现,需要经过网络运营商认证后,分配网络功能标识,才加载到网元中。

3.3　演示与验证

与 OMNet + + 相比,波音公司开发的 Linux 平台开放研究模拟器(Common Open Research Emulator, CORE)中每个节点都具有宿主机的协议栈,很容易开发网络微程序,实现边缘计算功能,因此,基于 CORE 模拟器构建了由两个无线路由器、一台控制器和多个移动终端构成的小型网络试验环境,如图 5 所示。通过研究移动终端在两个无线路由器之间做往返式移动的过程中,移动终端的无线链路切换时间和无线网络传输带宽,来说明主动表意网络对无线网络性能的提升。

移动终端距离无线路由器变远,无线信号则变弱,当无线网络断开后,移动终端再搜索新的无线网络信号。在无线路由器切换过程中,有断开网络服务或无线链路较差的情况,影响服务质量。而在主动表意网络中,移动终端通过转发意图原语向控制器通告位置坐标(x_1, y_1),控制器计算移动终端和不同无线路由器(x_2, y_2)之间的距离$sqrt[(x_1-x_2)^2+(y_1-y_2)^2]$,判断无线网络信号强弱,来控制移动终端主动连接无线路由器。试验表明,该方法减少了移动终端被动切换时间。

如果无线路由器之间没有覆盖区域重叠,那么移动终端只能断开再连接,两种切换方式没有任何差别。如果路由器之间覆盖区域有重叠,控制器可以根据无线路由器和移动终端之间距离远近来选择信号较好的无线路由器,从而避免掉线后再连接时出现的服务断开情况。

4　主动表意网络功能演化

主动表意网络中,转发意图原语一方面是用户意图的体现,可作为网元转发决策的重要依据,另一方面其解释执行过程融入网元分组处理流程中,直接扩展了网络分组处理功能。在 SDN 全局视图等信息支持下,转发意图原语以一种更直接更简单的方式改变了已有网络功能。主动表意网络的转发意图原语和解释器可动态扩展,支持网络功能演化。

4.1　转发意图原语扩展

随着应用的发展,网络功能越来越复杂,表意语言要求可扩展,主要采用原语和宏定义两个层次来扩展。

1)原语级扩展:根据用户需求扩展新的意图原语,待控制器认可后再部署应用。

转发意图原语主要针对用户对网络处理的期望,与高级语言需要支持通用数据处理不同,其主要是定制的面向领域的分组处理。从用户表意和隐私保护两个方面,从位置和状态管理、通信协

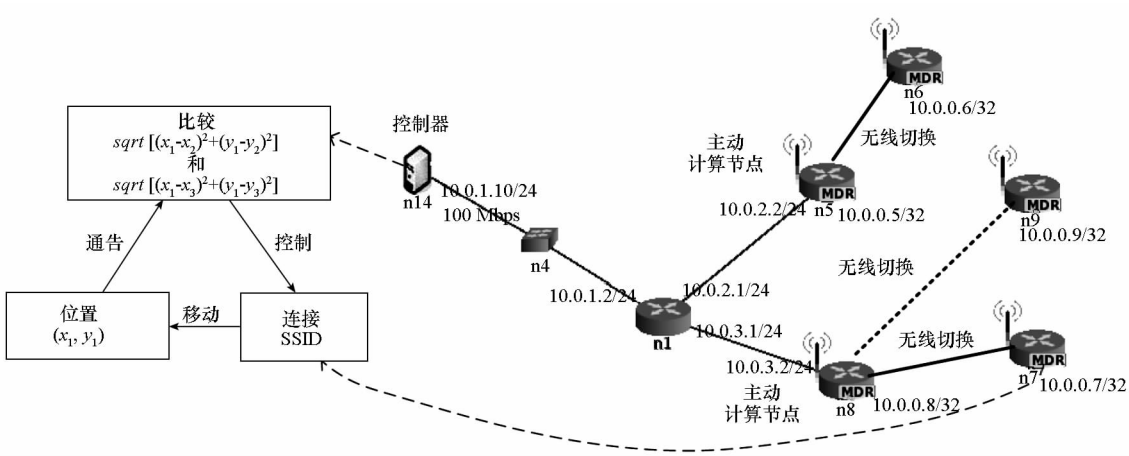

图 5　基于主动表意网络的无线链路切换控制

Fig. 5　Wireless switching in i-ECAN

议、服务质量需求、内容处理方式等多个角度可以扩展转发意图原语。

2)宏定义扩展:通过网络运营商事先向平台型网元注入代码,通过报文中携带的宏来索引并执行该代码。

在原语级扩展未得到运营商认可之前,用户可以申请边缘计算资源注入宏定义代码。运营商可以对宏定义代码进行验证和检查,确保符合执行的性能和资源占用率等约束。宏定义扩展时,源代码与转发意图程序不同,不再在解释器中执行,而是在单独的容器中运行,防止宏定义代码对解释器的破坏。宏定义代码类似于网络功能虚拟化 NFV 中虚拟网络功能(Virtual Network Function, VNF)程序,也是经过网络运营商验证的,按需部署服务用户的需求。

4.2 解释器功能扩展

主动表意网络解释器有两种——软件解释器和硬件可重构执行器,一般主要针对软件解释器进行扩展。在软件解释器中,首先分析转发意图原语程序语法,再将转发意图原语直接映射到微程序,在容器中执行。为了扩展转发意图原语和解释器,用户向控制器提交需求描述、转发意图原语、原语对应的微程序以及编译运行支撑库。控制器对功能的普及性进行评判,若该功能接受程度高,则作为原语增加到解释器中。控制器配置转发意图原语和微程序的映射关系,即回调函数,同时,将微程序以及相关库和以前的容器进行编译,形成新的解释器。

5 结论

本文扩展了用户和网络接口,支持主动表意,用户直接以转发意图原语形式向网络通告信息,实现人-网的协同。以边缘计算为基础,用户意图在网络边缘执行,对用户数据后续处理方式或流程等产生影响。转发意图原语主动向网络以显式方式通告用户的位置、QoS、协议、内容和状态等信息,使得网络直接掌握用户信息,从而做出比较合理高效的转发决策。同时,用户的隐私等信息也可以通过转发意图原语进行许可,既能得到利用,同时还能进行有效保护。

参考文献(References)

[1] Saltzer J H, Reed D P, Clark D D, et al. End-to-end arguments in system design[J]. ACM Transactions on Computer Systems, 1984, 2(4): 277-288.

[2] Clark D D, Wroclawski J, Sollins K R, et al. Tussle in cyberspace: defining tomorrow's internet[J]. IEEE/ACM Transactions on Networking, 2005, 13(3): 462-475.

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

[4] Nunes B A, Mendonca M, Nguyen X N. A survey of software-defined networking: past, present, and future of programmable networks[J]. IEEE Communications Surveys and Tutorials, 2014, 16(3): 1617-1634.

[5] Sivaraman A, Winstein K, Subramanian S, et al. No silver bullet: extending SDN to the data plane[C]//Proceeding of Hot Topics in Networks, 2013: 140-146.

[6] Casado M, Koponen T, Shenker S, et al. Fabric: a retrospective on evolving SDN[C]//Proceedings of the First Workshop on Hot Topics in Software Defined Networks, 2012: 85-90.

[7] Tennenhouse D L, Wetherall D J, et al. Towards an active network architecture[J]. ACM SIGCOMM Computer Communication Review, 1996, 26(2): 5-17.

[8] Tennenhouse D L, Smith J M, Sincoskie W D, et al. A survey of active network research[J]. IEEE Communications Magazine, 1997, 35(1): 80-86.

[9] de Simone L, Lanzaro A, Cotroneo D, et al. Network function virtualization: challenges and directions for reliability assurance[C]//Proceeding of International Symposium on Software Reliability Engineering, 2014: 37-42.

[10] Mijumbi R, Serrat J, Gorricho J, et al. Network function virtualization: state-of-the-art and research challenges[J]. IEEE Communications Surveys and Tutorials, 2016, 18(1): 236-262.

[11] Li H X, Shou G C, Hu Y H, et al. Mobile edge computing: progress and challenges[C]//Proceeding of IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, 2016: 83-84.

[12] Sabella D, Vaillant A, Kuure P. Mobile-edge computing architecture: the role of MEC in the internet of things[J]. IEEE Consumer Electronics Magazine, 2016, 5(4): 84-91.

[13] Salman O, Elhij I H, Kayssi A, et al. Edge computing enabling the Internet of Things[C]//Proceeding of the Internet of Things, 2015: 603-608.

[14] Jose L, Yan L S, Varghese G, et al. Compiling packet programs to reconfigurable switches[C]// Proceeding of Networked Systems Design and Implementation, 2015: 103-115.

[15] Bosshart P, Gibb G, Kim H S, et al. Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN[C]//Proceedings of the ACM SIGCOMM Conference, 2013: 99-110.

[16] Sivaraman A, Kim C, Krishnamoorthy R, et al. DC. p4: programming the forwarding plane of a data-center switch[C]// Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015: 9-18.

[17] Bosshart P, Daly D, Gibb G, et al. P4: programming protocol independent packet processors[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95.

[18] Shahbaz M, Choi S, Pfaff B, et al. PISCES: a programmable, protocol-independent software switch[C]// Proceedings of ACM SIGCOMM Conference, 2016: 525-538.

[19] Lockwood J W, McKeown N, Watson G, et al. NetFPGA—an open platform for gigabit-rate network switching and routing[C]// Proceeding of International Conference on Microelectronic Systems Education, 2007: 160-161.

[20] Xilinx. Vivado design suite 2017[EB/OL]. [2017-04-19]. http://www.xilinx.com/.

[21] Altera. Altera SDK for OpenCL 15.0[EB/OL]. [2016-02-01]. http://www.altera.com/.

[22] Li B J, Tan K, Luo L Y, et al. ClickNP: highly flexible and high-performance network processing with reconfigurable hardware[C]//Proceedings of ACM SIGCOMM Conference, 2016: 1-14.