3.3.3 Assertion Language[32]

Assertion Language 是为了方便网络验证而提出的高级逻辑型语言.SDN 中虽然可以有全网的视图以及全网设备的行为记录,但是现有的 SDN 中的网络验证大部分都是基于静态的网络配置或是在数据平面进行分析验证,所以无法根据上层应用逻辑的动态变化而做出实时性的验证.在这种情况下,Assertion Language 便是能够对数据平面不断变化的网络做出验证的一种语言,它不仅能够做出可达性、是否存在循环以及切片是否分离这种一般性的验证,还可以利用本文提出的语言做出“有状态的防火墙”,“在 MAC 学习的过程中减少控制器的开销”这样的验证.

Assertion Language 可以针对变化的网络做出实时性的验证,采用数据平面的事件作为驱动的方式,利用正则表达式描述不同种类的报文的路径特征,采用逻辑型编程模型.但与其他语言不同的是,本文提出的语言不是为了抽象现有北向接口而提出来的,而是专门针对网络验证的高级接口.属于为了实现 SDN 中的某个新功能而做出的北向控制平面的抽象.

3.4 北向接口中的命令式语言

命令式语言由于更强调如何去实现一个过程,而不是强调实现什么过程,所以更加注重怎么做网络策略.前几种 SDN 北向接口的语言都重在抽象出一个接口,供编程者用它去获取自己需要的策略资源.而命令式语言更加关注的是让编程者去想如何实现这个算法策略.命令式语言的典型代表便是 Pyretic.

3.4.1 Pyretic[16]

Pyretic 也是对 SDN 北向接口语言的抽象,是 Frenetic 编程语言的 python 子类.Pyretic 可以让网络编程者和管理者写出模块化的网络应用,它基于 python,实时的编译系统可以将利用 pyretic 写出的程序转化为流表规则下发到交换机中,Pyretic 也是仅支持 OpenFlow 设备.

Pyretic 处理报文,对于编程者来说,屏蔽了底层 OpenFlow 规则被使用的细节,利用高级语言编写转换函数,把 packetin 报文转化为 packetout 报文,同时编译出相应的无冲突的流表规则.Pyretic 支持模块化编程,支持报文各个字段层次的或与非的操作,支持网络虚拟化,还可以利用事件的监听回调机制处理动态的策略等等.

3.5 北向接口中的面向对象编程语言

在 SDN 的北向接口的高级编程语言中,有一类语言专门为了实现网络虚拟化的功能,它们的编程模式更像面向对象的编程模型,即每一个对象都应该能够接受数据、处理数据并将数据传达给其他对象.这种编程模式的好处是可以将网络中的元素抽象为一个个对象,灵活且利于维护.SDN 中这类语言的典型代表有上文提到的 Pyretic,还有文献[33].

3.5.1 Splendid Isolation[33]

特定种类的流量彼此分离是很多网络操作得以正确进行的前提,例如高等院校要保护学生的记录,情报局的认证系统的流量要与常规流量分离,数据中心的租户之间的流量要分离等等.但是,在当今网络虚拟化的机制中,有很多机制都是临时的,比如 VLAN(虚拟局域网)虽然可以隔离处理网络中不同类型的报文,但加重了原本复杂的网络配置;防火墙虽然可以阻止特定类型的报文进入特定的网络切片中,但需要在控制层加入 hypervisor(例如 Flowvisor[41])并且需要可信.总之,没有一种方案能够提供一个完全满意的隔离机制,而且这些机制都不能提供一个验证网络是否被有效隔离的功能.

基于以上问题,文献[33]提出了一个可以允许多个网络程序并行运行的网络切片机制,同一个物理拓扑可以被多个应用程序所用,它可以实现流量隔离、物理隔离以及控制隔离.通过这种语言编写的程序易读,并且对用户友好,采用网络隔离的抽象接口,可以编写简单、灵活的程序,进而把物理网络分为多个虚拟切片,再在这个切片上编写应用程序.

3.5.2 Network Control Language(简称 NCL)[34]

NCL 是由华为公司提出的北向接口的高级语言,它基于 Java,编程者同样不用关心设备的实现细节,只关注应用的需求.由于已将 NCL 嵌入到 Java 中,所以用户可以直接用 Java 编写网络应用,NCL 中的编译器可以将策

略编译为数据平面的低级语言.利用 NCL 面向对象的语法模式可以完成如下功能:(1) 方便地定义自己的虚拟网络;(2) 时间监听与回调机制可以处理网络中的协议和通知;(3) 可以应对上层的拓扑查询等应用.

NCL 的系统结构如图 6 所示.

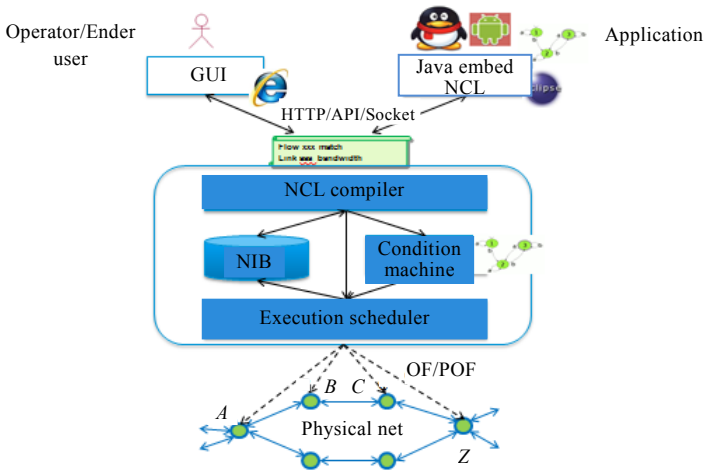

Fig.6 NCL architecture[34]

图 6 NCL 系统结构[34]

3.6 小 结

以上便是北向接口中的高级语言,总结北向接口中的各种编程语言,它们的共同点都是为了给网络编程带来简捷,通过抽象底层硬件设备细节,抽象流表生成及下发细节,抽象网络配置的细节,简化编程者的工作,使得网络管理员甚至是一般的网络参与者都能配置自己需要的网络策略,进而指导网络的行为.同时,某些北向接口的语言还可以提供网络验证、兼容各种网络中间盒设备、网络虚拟化等特性,使得 SDN 的可编程特性得到效益的最大化.下一节将对北向接口中的高级语言进行横向比较.

4 北向接口语言的横向比较

经过了上一节对于北向接口高级语言的分类和详细介绍,下面将对各个分类的语言进行一个横向的比较.

对于高级语言和低级语言而言,高级语言有如下优势:

(1) 屏蔽底层设备接口细节,方便编程者只关注网络策略的算法逻辑;

(2) 支持并行的应用同时运行,不用编程者手动解决流量冲突问题;

(3) 模块化的编程方便代码的复用;

(4) 物理网络可以被切分为隔离好的虚拟网络,方便路由等策略的部署.

而针对高级语言中的声明式语言、命令式语言以及面向对象语言,在不同的应用场景下分别有不同的优势.函数型语言中的 Frenetic,HFT,PANE,Maple,NetCore 更适合处理对于转发、防火墙等需要处理报文的应用;对于网络验证、编写有状态的防火墙的应用来说,Flog,Frenetic,FlowLog,Assertion Language 有明显的优势;对于网络虚拟化以及需要对底层流量做隔离的应用来说,面向对象的编程语言 Splendid Isolation,pyretic 有明显的优势.表 3 给出了在各个分类下的编程语言的详细比较.

由表 3 可以看出,从是否引入新的网络功能这个分类标准来看,对于需要进行网络验证、模型检查的网络,Flog、FlowLog、Merlin、Assertion Language 等都存在优势,而对于需要作网络切片、流量隔离的网络来说,Splendid Isolation 和 NCL 是比较不错的选择.同时,如果需要在网络上部署多个协议,可以采用 HFT、PANE、Maple、pyretic 等能够有效解决流表冲突的语言.

从语言的实现机制上,对于嵌入现有语言的北向接口语言来说,可以利用现有语言丰富的库以及复杂的功

能,对于编程者来说也能相对容易些.与此相对应,对于自定义语法规则的语言来说,更容易填写新的功能以及针对网络需求优化编译策略.

Table 3 Programming languages in each category

表 3 各个分类下的编程语言

编程语言	语言形式	编程模型	是否引入新功能	实现机制	主要面向的应用
NOX,POX 提供的北向 API	低级语言	命令式	直接利用设备提供的接口编程	内嵌于 Java 或 Python	符合相应 OpenFlow 版本的网络应用
tinyNBI	低级语言	命令式	兼容所有版本的 OpenFlow 协议	自定义语法规则	兼容现有 5 个版本 OpenFlow 的应用
FML	高级语言	交互式	简化协议部署过程	基于非递归的 DATALOG 语言	企业网络配置管理,包括 ACL、NAT、VLAN 等
Nettle	高级语言	交互式、函数型	网络事件的驱动转化为流驱动,可以处理动态协议	嵌入到 Haskell 语言中	支持 OpenFlow 交换机的网络.面向的应用未作明确限定,可支持动态路由、流量工程、安全策略、负载均衡等应用
Procera	高级语言	交互式、函数型	更好地处理网络中的交互行为	自定义语法规则	家庭网络和企业网络等的网络控制策略
Frenetic	高级语言	函数型	更加简化的报文转发模型,同时支持网络拓扑的改变以及网络验证等功能	扩展 OCaml 语法规则	未明确限定应用场景,可支持网络监测、路由、负载均衡等应用.可支持源码层面的网络验证等功能
HFT	高级语言	函数型、逻辑型	把策略转化为有优先级的流表,有效避免流表冲突	自定义语法规则	面向具有层次化策略的网络应用,能够解决策略冲突
PANE	高级语言	函数型、逻辑型	对 HFT 的扩展,可以给网络参与者分配权限,让他们参与网络配置	自定义语法规则	面向具有层次化策略的网络应用,能够解决策略冲突
Maple	高级语言	函数型	优化策略生成流表的过程并提供一个高效、多核的调度器	自定义语法规则	对具体的应用场景未作限定
NetCore	高级语言	函数型	改进 Frenetic 的编译算法,更高效地编译报文转发策略	基于 Frenetic	未明确限定应用场景,可支持网络监测、路由、负载均衡等应用
FlowLog	高级语言	函数型	更加方便地进行网络验证	自定义语法规则	面向模式检查、网络验证等应用
FatTire	高级语言	函数型	利用正则表达式提高网络的容错性	自定义语法规则	链路故障的发现以及容错处理等应用
NetKAT	高级语言	函数型	拥有完备的数学理论,可以进行网络验证	自定义语法规则	对具体的应用场景未作限定
Flog	高级语言	逻辑型	可进行模型检查、动态验证、编写有状态的中间盒	自定义语法规则	面向模型检查、动态验证等
Merlin	高级语言	逻辑型	支持将网络权限向下分发	自定义语法规则	对具体的应用场景未作限定,可鉴别流量类型
Assertion language	高级语言	逻辑型	在应用的级别作网络验证	基于 python	面向数据平面的网络验证
pyretic	高级语言	函数型、命令式	支持协议策略的并行化编写	基于 python	对具体的应用场景未作限定
Splendid isolation	高级语言	面向对象	提供网络切片功能	基于 python	主要面向网络虚拟化、网络隔离、网络切片等应用
NCL	高级语言	面向对象	支持网络切片、事件驱动以及网络资源请求功能	基于 Java	主要面向网络虚拟化、网络隔离、网络切片等应用

在语言的表达能力上,高级语言由于对细节的封装程度高,所以表达性要强于低级语言,更加易读,并且方便学习,所以对于用户也更加友好.在实现形式上,基本上所有的高级语言都由两部分组成:上层的高级语言以及下层的编译器,这一点基本类似.在运行效率上,编译器的效率决定了整体的运行效率,在高级语言中,NetCore 和 Maple 等在编译器上作了很多有效的优化,所以运行效率相对较高.

5 SDN 北向接口语言研究方向的展望

SDN 作为一种新型的网络体系结构,用可编程的交换机代替了传统网络中的交换机、路由器、防火墙等设备,通过集中的控制器去管理网络.SDN 的上层应用可以利用北向接口获取网络信息并做出网络策略指导下层设备的行为.这样的架构给网络的可编程性带来可能,但是基于 SDN 的编程却并不简单.针对现在 OpenFlow 给出的接口过于低级带来的各种问题,SDN 北向接口中的高级语言不断涌现,它们都在不同的应用层面上给网络编程者带来方便,克服了低级语言存在的种种问题.

但是目前针对 SDN 北向接口的研究工作还处于起步阶段,尚未发展成熟,仍有很多值得研究的问题,未来可进一步研究的相关重要方向包括:

(1) 北向接口还没有形成一个统一的标准.针对现在控制器多样化的情况,需要制定一套适用于定义北向接口的通用语义,这样不仅编程者有一个标准的部署应用的流程,开发控制器的人也可以利用这些已标准化的语义去开发各自的北向接口.

(2) 各个北向接口的编程语言都是针对特定场景而提出的,由于语言特性的限制,也只能固定简化某一类网络应用.如果能有一种较为通用的语言可以解决绝大多数场景下的问题,不拘束于语言的特性,则可以极大地提高编程效率.

(3) 对于一些常用的网络应用,北向接口语言可以提供一套常用功能函数库的支持,以提高 SDN 网络中的编程效率.

(4) 目前北向接口高级语言框架中编译器的性能还有待优化.例如,在提高编译生成流表的效率、尽量减少编译生成流表的优先级等方面,还有很大的提升空间.

(5) 现有的很多语言还仅仅支持单一控制器,如果能让这类语言自动部署运行在分布式控制器中,则可以有效提高控制器的性能以及整个网络的鲁棒性.

(6) 现有的北向接口语言往往只能支持有限的新功能,如何将多种新功能集成到同一个北向接口语言中,有待进一步加以研究.

(7) 对于可以处理网络链路错误配置的北向接口语言(如 FatTire 等)来说,如何能够发现交换机层级的错误以及如何让链路进行自动恢复也是未来的研究工作.

(8) 对于支持网络隔离和虚拟化的北向接口语言(如 Splendid Isolation),现在只限于在拓扑上进行网络切片,未来可以考虑针对网络带宽等更多的网络资源进行切片.

6 总 结

本文是一篇针对软件定义网络中北向接口语言的综述,为了简化软件定义网络中的编程模型,给网络策略的开发和部署提供便捷,近年来,北向接口的语言纷纷涌现出来,在这些语言中,根据不同的分类标准,又可以划分为不同的层次,本文详细介绍了学术界和工业界现有的各类北向接口的编程语言,同时对各种语言进行了横向比较,并对未来北向接口的研究工作进行了展望.

References:

[1] ONF Market Education Committee. Software-Defined Networking: The New Norm for Networks. ONF White Paper. Palo Alto: Open Networking Foundation, 2012.

[2] Nunes BAA, Mendonca M, Nguyen XN, Obraczka K, Turletti T. A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communication Surveys & Tutorials, 2014,16(3):1617−1634. [doi: 10.1109/SURV.2014.012214.00180]

[3] Kreutz D, Ramos FMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-Defined networking: A comprehensive survey. Proc. of the IEEE, 2015,103(1):14−76. [doi: 10.1109/JPROC.2014.2371999]

[4] Hu F, Hao Q, Bao K. A survey on software defined networking (SDN) and OpenFlow: From concept to implementation communications surveys & tutorials. IEEE Communication Surveys & Tutorials, 2014,16(4):2181−2606. [doi: 10.1109/COMST.2014.2326417]

[5] Zuo QY, Chen M, Zhao GS, Xing CY, Zhang GM, Jiang PC. Research on OpenFlow-based SDN technologies. Ruan Jian Xue Bao/Journal of Software, 2013,24(5):1078−1097 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4390.htm [doi: 10.3724/SP.J.1001.2013.04390]

[6] Gude N, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N, Shenker S. NOX: Towards an operating system for networks. Computer Communication Review, 2008,38(3):105−110. [doi: 10.1145/1384609.1384625]

[7] McCauley M. POX, 2012.

[8] Floodlight Is A Java-Based OpenFlow Controller, 2012.

[9] https://openflow.stanford.edu/display/Beacon/Home

[10] http://www.opendaylight.org/

[11] http://www.juniper.net/us/en/dm/sdn/

[12] Mckeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008,38(2):69−74. [doi: 10.1145/1355734.1355746]

[13] Bosshart P, Dan D, Gibb G, Martin I, McKeown N, Rexford J, Schlesinger C, Talayco D, Vahdat A, Varghese G, Walker D. P4: Programming protocol-independent packet processors. Computer Communication Review, 2014,44(3):87−95. [doi: 10.1145/2656877.2656890]

[14] Song H. Protocol-Oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane. In: Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN 2013. Hong Kong: ACM, 2013. 127−132. [doi: 10.1145/2491185.2491190]

[15] Casey CJ, Sutton A, Sprintson A. tinyNBI: Distilling an API from essential OpenFlow abstractions. In: Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN 2014. Chicago: ACM, 2014. 37−42. [doi: 10.1145/2620728.2620757]

[16] Monsanto C, Reich J, Foster N, Rexford J, Walker D. Composing software-defined networks. In: Proc. of the 10th USENIX Conf. on Networked Systems Design and Implementation, ser. NSDI 2013. Berkeley: USENIX Association, 2013. 1−14.

[17] Laird A. The four major programming paradigms topic paper #17. Computer Science, 2009. http://www.alexlaird.com/content/uploads/2009/05/topicpaper17-thefourmajorprogrammingparadigms.pdf

[18] Coenen F. Characteristics of declarative programming languages. 1999. http://cgi.csc.liv.ac.uk/~frans/OldLectures/2CS24/declarative.html

[19] Voellmy A, Hudak P. Nettle: Taking the sting out of programming network routers. In: Proc. of the PADL 2011. Berlin, Heidelberg: Springer-Verlag, 2011. 235−249. [doi: 10.1007/978-3-642-18378-2_19]

[20] Hinrichs TL, Gude NS, Casado M, Mitchell JC, Shenker S. Practical declarative network management. In: Proc. of the WREN 2009. Barcelonan: ACM, 2009. 1−10. [doi: 10.1145/1592681.1592683]

[21] Voellmy A, Kim H, Feamster N. Procera: A language for high-level reactive network control. In: Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN 2012. Helsinki: ACM, 2012. 43−48. [doi: 10.1145/2342441.2342451]

[22] Reitblatt M, Canini M, Guha A, Foster N. Fattire: Declarative fault tolerance for software defined networks. In: Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN 2013. Hong Kong: ACM, 2013. 109−114. [doi: 10.1145/2491185.2491187]

[23] Nelson T, Guha A, Dougherty DJ, Fisler K, Krishnamurthi S. A balance of power: Expressive, analyzable controller programming. In: Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN 2013. Hong Kong: ACM, 2013. 79−84. [doi: 10.1145/2491185.2491201]

[24] Foster N, Harrison R, Freedman MJ, Monsanto C, Rexford J, Story A, Walker D. Frenetic: A network programming language. In: Proc. of the ICPF 2011. Tokyo: ACM, 2011. 279−291. [doi: 10.1145/2034773.2034812]

[25] Foster N, Guha A, Reitblatt M, Story A, Freedman MJ, Katta N.P, Monsanto C, Reich J, Rexford J, Schlesinger C, Walker D, Harrison MR. Languages for software-defined networks. IEEE Communications Magazine, 2013,51(2):128−134. [doi: 10.1109/MCOM.2013.6461197]

[26] Ferguson AD, Guha A, Liang C, Fonseca R, Krishnamurthi S. Hierarchical policies for software defined networks. In: Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN 2012. Helsinki: ACM, 2012. 37−42. [doi: 10.1145/2342441.2342450]

[27] Voellmy A, Wang J, Yang YR, Ford B, Hudak P. Maple: Simplifying SDN programming using algorithmic policies. In: Proc. of the SIGCOMM 2013. Hong Kong: ACM, 2013. 87−98. [doi: 10.1145/2486001.2486030]

[28] Monsanto C, Foster N, Harrison R, Walker D. A compiler and run-time system for network programming languages. In: Proc. of the POPL 2012. Philadelphia: ACM, 2012,47(1):217−230. [doi: 10.1145/2103656.2103685]

[29] Guha A, Reitblatt M, Foster N. Machine-Verified network controllers. In: Proc. of the PLDI 2013. Seattle: ACM, 2013. 483−494. [doi: 10.1145/2491956.2462178]

[30] Katta NP, Rexford J, Walker D. Logic programming for software-defined networks. In: Proc. of the ACM SIGPLAN Workshop on Cross-Model Language Design and Implementation, ser. XLDI. 2012. https://www.cs.princeton.edu/~dpw/papers/xldi-2012.pdf

[31] Soule R, Basu S, Kleinberg R, Sirer EG, Foster N. Managing the network with Merlin. In: Proc. of the 12th ACM Workshop on Hot Topics in Networks (HotNets-XII). College Park, 2013. 1−7. [doi: 10.1145/2535771.2535792]

[32] Beckett R, Zou XK, Zhang SY, Malik S, Rexford J, Walker D. An assertion language for debugging SDN applications. In: Proc. of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN 2014. Chicago: ACM, 2014. 91−96. [doi: 10.1145/2620728.2620743]

[33] Gutz S, Story A, Schlesinger C, Foster N. Splendid isolation: A slice abstraction for software-defined networks. In: Proc. of the 1st Workshop on Hot Topics in Software Defined Networks, ser. HotSDN 2012. Helsinki: ACM, 2012. 79−84. [doi: 10.1145/2342441.2342458]

[34] Network Control Language(NCL) Software Defined Networking Research Group. 2014. https://datatracker.ietf.org/meeting/90/agenda/sdnrg/

[35] Anderson CJ, Foster N, Guha A, Jeannin JB, Kozen D, Schlesinger C, Walker D. NetKAT: Semantic foundations for networks. In: Proc. of the POPL 2014. San Diego: ACM, 2014. 113−126. [doi: 10.1145/2535838.2535862]

[36] Ferguson AD, Guha A, Liang C, Fonseca R, Krishnamurthi S. Participatory networking: An API for application control of SDNs. In: Proc. of the SIGCOMM 2013. Hong Kong: ACM, 2013. 327−338. [doi: 10.1145/2486001.2486003]

[37] Pfenning F. Logic programming lecture 26 datalog. 2006. http://www.cs.cmu.edu/~fp/courses/lp/

[38] Oliveira N, Pereira MJV, Henriques PR, Cruz D. Domain specific languages: A theoretical survey. In: Proc. of the 3rd Compilers, Programming Languages, Related Technologies and Applications (CoRTA 2009). 2009. http://alfa.di.uminho.pt/~danieladacruz/CoRTA09DSLsurveyvf.pdf

[39] Bird R. Introduction to Functional Programming Using Haskell. New York: Prentice Hall, 1998.

[40] Kazemian P, Varghese G, McKeown N. Header space analysis: Static checking for networks. In: Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation, ser. NSDI 2012. Lombard: USENIX Association, 2012. 9. https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/kazemian

[41] Sherwood R, Gibb G, Yap KK, Appenzeller G, Casado M, McKeown N, Parulkar G. FlowVisor: A network virtualization layer. Technical Report, OPENFLOW-TR-2009-01, OpenFlow Consortium, 2009. 1−14.

附中文参考文献:

[5] 左青云,陈鸣,赵广松,邢长友,张国敏,蒋培成.基于 OpenFlow 的 SDN 技术.软件学报,2013,24(5):1078−1097. http://www.jos.org.cn/1000-9825/4390.htm [doi: 10.3724/SP.J.1001.2013.04390]

于洋(1989－),女,吉林松原人,硕士生,主要研究领域为软件定义网络.

王之梁(1978－),男,博士,副研究员,主要研究领域为网络协议测试与形式化方法,互联网体系结构与协议,软件定义网络.

毕军(1972－),男,博士,研究员,博士生导师,CCF 杰出会员,主要研究领域为互联网体系结构与协议,未来互联网,软件定义网络,互联网路由.

施新刚(1980－),男,博士,高级工程师,主要研究领域为网络测量,互联网体系结构与协议,互联网路由.

尹霞(1972－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为互联网体系结构与协议,网络协议测试与形式化方法,互联网路由.