软件定义网络一致性协同更新算法

于倡和,兰巨龙,胡宇翔
(国家数字交换系统工程技术研究中心,河南郑州 450002)

摘 要: 为实现软件定义网络的一致性更新,本文提出一种协同利用分段路由、顺序更新、两步复制三种机制的更新算法. 算法首先启用分段路由机制,尝试用现有路径规则拼接待更新数据流的最终路径,并根据最终路径是否能由现有规则拼接,将数据流分为可拼接与不可拼接两种. 对于可拼接流,分段路由可将最终路径信息封装入数据包包头,使得数据包能立即沿最终路径转发. 对于不可拼接流,算法计算最长一致性更新序列,并按照此序列依次更新节点,最后利用两步复制机制来完成剩余未更新节点的更新. 并且经实验验证,算法比之前研究提出的算法不仅消耗更少的三态内容寻址存储器的空间资源,并且有更好的适用性与稳定性.

关键词: 一致性更新; 分段路由; OpenFlow
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2018)10-2341-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.10.005

Synergetic Consistent Update Algorithm for SDN Networks

YU Chang-he, LAN Ju-long, HU Yu-xiang
(*National Digital Switching System Engineering & Technological Research Center, Zhengzhou, Henan* 450002, *China*)

Abstract: To achieve consistent update in software defined network, a consistent update algorithm which combines segment routing, two-phase commit and node scheduling mechanism is proposed in this work. The algorithm first leverages segment routing mechanism, which attempts to splice the final path with existing paths. According to whether the final path can be spliced by the existing paths, the algorithm divides flows into either segmentable flows or flows that are not segmentable. For the segmentable flow, segment routing mechanism can encapsulate the final path information into the packet header so that packets can be forwarded immediately along the final path. For the flows not segmentable, the algorithm calculates the longest consistent update sequences for them, and updates nodes in accordance with these sequences. Finally, the algorithm uses the two-phase commit mechanism to complete the update of the remaining nodes. We verified its performance by experiments and the outcome illustrates that our algorithm not only requires less additional ternary content addressable memory resources but also has better performance stability and applicability than prior techniques.

Key words: consistent update; segment routing; OpenFlow

1 引言

因网络状态的动态变化特性,规则更新是软件定义网络(Software Defined Network, SDN)中的常见现象. 运营商可通过网络更新来调整路由、更改访问权限以完成网络维护、漏洞修补等运维操作. 而转发信息表(Forward Information Base, FIB)的更改是网络更新的核心,用新的 FIB 规则替代旧有的 FIB 规则是更新过程的本质. 网络更新是一个渐进的增量过程,在全网更新完成前,不同交换机的配置可能是不一致的,这可能造成网络环路、丢包等一系列异常,所以如何维护更新一致性,保障网络的正常运行,避免各种异常与错误是网络更新的关键.

更新的一致性,便是确保每个流的转发要么采用初始规则,要么采用最终规则,没有第三种选择. 而关于 SDN 网络的一致性更新,之前已经有了大量的研究工作. 文献[1~5]基于顺序更新(ordered scheduling)机制,即按照特定约束下求得的节点序列来更新网络. 虽然该机制不会引入额外的三态内容寻址存储器(Ternary Content Addressable Memory, TCAM)资源消耗,但这

收稿日期:2017-11-20;修回日期:2018-06-02;责任编辑:李勇锋
基金项目:国家自然科学基金(No. 61502530);国家 863 高技术研究发展计划(No. 2015AA016102)

种机制的强一致性更新序列未必一定存在[1],故而只能保证全网策略维护、网络无环、无拥塞等弱一致性更新[2~5]. 文献[6,7]基于两步复制机制(two-phase commit),其核心在于让相关节点同时维护初始与最终两种规则,并通过入口节点为相关数据流所加的不同标签来决定所采取的转发规则,这类算法一般会消耗大量的 TCAM 资源. 除此之外,文献[8]中提出的 GPIA + FED 算法是协同利用顺序更新机制与两步复制机制来完成网络更新. GPIA + FED 算法虽然可以很好的完成网络更新,并减少更新过程的 TCAM 资源冗余. 但是当 ordered scheduling 机制失效或者所求最长一致性更新序列的节点数目远小于全网待更新的节点数目时,算法所需的额外 TCAM 资源逼近 two-phase commit 机制,性能迅速恶化. 综上,之前一致性更新算法一般受限于适用性与资源开销.

为了实现 SDN 网络的普适低冗余一致性更新,本文协同利用分段路由[9],ordered scheduling 与 two-phase commit 等机制来更新网络,提出了 FCUA (Fast Cooperative Consistent Update Algorithm)算法. 其中,分段路由可以通过现有路径拼接的方式来尝试构建待更新数据流的最终路径,若最终路径可拼接,则可以通过为数据包包头加装对应段标识符(segment identifier,SID)的方式来指导数据包沿最终路径转发,这里 SID 是由各拼接段末尾节点的 IP 构成. 相应的,若最终路径不可拼接,则可以利用 ordered scheduling 与 two-phase commit 等机制来完成网络更新. FCUA 算法具体架构在第二章中有详细论述,并且实验证明该算法可以在保持更新过程强一致性的提前下,极大的减少 TCAM 的额外消耗,并且明显提升了算法的性能稳定性与适用性.

2 算法设计

我们提出的 FCUA 算法通过三步更新来实现 SDN 网络的一致性更新,第一步通过运行基于分段路由机制的 CASR (Consistent update Algorithm using Segment Routing)算法,利用分段路由为可拼接数据流安装相应的 SID,在保证一致性的前提下,使得数据流可以立即沿着新路径传输,并快速更新节点. 因不可拼接流可能存在,单一的 CASR 算法可能无法完成全网更新. 所以在第二步在 CASR 处理的基础上,运行基于 ordered scheduling 机制的 NOUA (Node Ordering Update Algorithm)算法,计算最长一致性更新序列,按照该序列更新节点. 如果网络中尚有待更新节点,则启用 two-phase commit 机制,通过规则复制的方式来完成最后的更新.

2.1 CASR 算法设计

分段路由更新机制的性能与网络中可拼接的数据流比例直接相关,但甄别可拼接流并分配合适的段标识符序列(SID List,SL)是困难的. 因现有路径规则有指数多种拼接方式,且最终路径可以同时有多个可行的拼接集,如图 1,路径 $<u\ v\ z\ w\ d>$ 可以有拼接集 $<uv\ vz\ zw\ wd>$ 也可以按 $<uvzw\ wd>$ 的方式拼接,显然,第二种拼接方式需要安装的 SL 短,引入的额外带宽小,所以如何找到所有的可拼接流并且分配最优的 SL 是利用分段路由完成网络更新的关键.

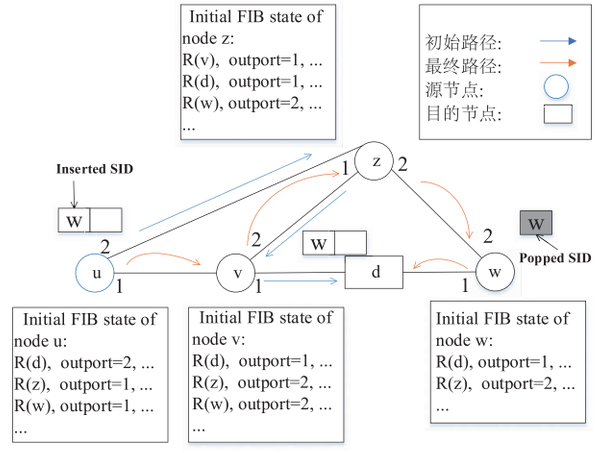

图1 分段路由在网络更新中的应用

由于各个数据流之间相互独立,各数据流的 SL 的寻找与优化问题可利用分治法思想独立求解. 而在 CASR 算法中,我们将求解某一数据流 SL 的问题转化为一个 0 - 1 整数线性规划(Integer Linear Programming,ILP)问题,通过求解所建立的模型,我们可以判断出该数据流是否可拼接,如果模型有解,则数据流可拼接,并得到最优的 SL.

算法 1 详细的记述了 CASR 算法的核心思想. 根据网络的初始与最终配置,算法可以计算出发生变化需要更新的数据流集合(行 2). 确定待更新数据流集后,依次遍历待更新的数据流,对数据流分别建立相应的 ILP 模型并求解,便可判断出数据流是否可拼接,如果可拼接,则模型的解便是最优的拼接集. 对于任意待更新数据流,我们首先查找其最终路径所有可能的拼接段在现有路径集中是否存在,并形成集合 C(行 5),而 L 为数据流最终路径的跳数,seg 代表当前模型下,数据流的拼接集中拼接段数量的最大值,随后求解 ILP 模型,若模型无解,将 seg 的值加一重新建模求解,直到模型有解或者 seg 的值超出界限(行 8 - 11),如果最终模型有解,则存储该流的 SID,如果模型最终无解,则标记该流不可拼接(行 12 - 17). 模型建立与求解过程本质上是将搜索拼接段组成新路径的过程建模为一个 0 - 1 规划,并求解的过程,当拼接段数目的上界 L 与候选拼接段集 C 已知后,我们建立 ILP 模型,目标为式(1)而约束为式(2) ~ (5). 优化目标是让布尔变量 $flag$ 取得最

大值,而该变量代表流 f 是否可拼接,而在约束条件中,布尔变量 $w_f^{x,y}$ 代表拼接段 $c_{x,y}$ 是否被选中用于拼接流 f 的新路径,而布尔变量 $a_l^{x,y}$ 与 b_l^f 分别代表边 l 是否在拼接段 $c_{x,y}$ 与流 f 中,而式(3)约束拼接路径中必须含有最终路径中的所有边,而 L_f 与 L_f^* 代表流 f 最终路径的跳数与拼接成的流路径边数. 显然,式(4)是为保证拼接路径与流最终路径的长度相同. 最后一条约束是关于拼接段数量的,被选取的拼接段数量必须小于等于本轮求解所限定的拼接段的最大值.

$$\text{Maximize}: flag \tag{1}$$

$$\text{Subject to}: \tag{2}$$

$$\forall c_{x,y} \in C: w_f^{x,y} \in \{0,1\}, flag \in \{0,1\}$$

$$\forall l \in P_f: \sum_{c_{x,y} \in C_{s,d}} a_l^{x,y} \times w_f^{x,y} \leq b_l^f \tag{3}$$

$$L_f \times flag = L_f^* \tag{4}$$

$$\sum_{c_{x,y} \in C} w_f^{x,y} \leq seg \tag{5}$$

算法 1　CASR:利用分段路由机制更新网络

输入:节点集合 N, 初始配置 G_i, 最终配置 G_f, 目的节点集 D

输出:SL 集 S, 当前路径 P_c, 未更新数据流集 *unsafepair*.

```
1:  S={} #全网待更新数据流 SL 集
2:  flow=compute_changing_flow(G_i, G_f, D) #待更新数据流
3:  for f in flow:
4:      S_f= {} #数据流 f 的 SL
5:      C=find(f, P_i, P_f) #候选拼接段集合
6:      seg=2
7:      L=length(P_f) #最终路径的跳数
8:      while S_f=∅ and seg<L:
9:          S_f =solveILP()
10:         seg= seg +1
11:     end loop
12:     if S_f≠∅ :
13:         S.add(S_f)
14:         P_c=update_network (f, G_i, G_f)
15:     else:
16:         unsafepair.add (f)
17:     end if
18: end for
19: return S, P_c, unsafepair
```

2.2　NOUA 算法设计

因为并不是所有的数据流都是可拼接的,对于不可拼接的数据流,我们采取 NOUA 算法来进行更新. 在 CASR 算法更新的基础上,NOUA 算法采用 ordered scheduling 机制来计算最长一致性更新序列. 而在本质上,NOUA 是一种贪婪算法.

算法2　NOUA:计算最长一致性更新节点序列

输入:节点集合 N, 初始配置 G_i, 最终配置 G_f, 目的节点集 D, 未更新数据流 *unsafepair*.

输出:更新序列 *nl*, 未更新数据流集 *unsafepair*, 当前路径集 P_c.

```
1:  U=calculate_updated_node (N, unsafepair) #已更新节点集
2:  while len(U) < len(N):
3:      C=set(N). difference(U) , node_candidates=set(C)
4:      P_c=contruct_current_paths(N, G_i, G_f, D, U)
5:      for n in C:
6:          P=contruct_current_paths(N, G_i, G_f, D, U∪{n})
7:          if not safety_check(N, G_i, G_f, D, P)
8:              node_candidates.remove (n)
9:          end if
10:     end for
11:     F=list(node_candidates)
12:     if len(F)==0:
13:         break
14:     (nl, U)=update (F, nl, U)
15: end loop
16: unsafepair=store_unsafe (N, G_i, G_f, D, U)
17: return (P_c, unsafepair, nl)
```

算法 2 描述了 NOUA 算法的设计架构,首先,算法根据 CASR 算法输出的未更新数据流集 *unsafepair* 构建已更新节点集 U(行 1). 在所有节点完成一致性更新或者没有可以一致性更新的候选节点之前,算法会将所有未更新节点作为一致性更新序列的候选节点(行 2 – 3),并依次进行更新一致性检测. 即从候选节点集中随机挑选一个节点,在假设该节点已经被更新的条件下,计算当前路径集(行5 – 6). 随后,函数 *safety_check* 通过检测该节点更新后各节点到目的节点的路径是否有既不是初始路径又不是最终路径的情况来确定节点的更新一致性. 如果节点无法进行一致性更新则移除该节点(行 7 – 9),一轮节点遍历结束后,判断所剩候选节点数目是否为 0,若为零则终止循环. 若数目非零,则更新最长一致性更新序列与已更新节点的信息(行 12 – 14),并再进行循环条件判定,直到所有节点完成更新或者已经没有节点可以进行一致性更新. 循环结束后,根据已更新节点信息,重新计算未更新数据流集合(行 16). 对于剩余无法通过 NOUA 算法一致性更新的节

点,我们最后通过 two-phase-commit 机制来完成一致性更新.

3 性能测试与分析

我们将通过实际的拓扑网络验证我们算法的效能,并与之前的技术方法进行比较. 在 3.1 节主要叙述我们的验证环境与方法,后续章节将算法运行结果与当前主流与最新的技术方法进行比较.

3.1 实验方法

出于普适性与现实性考虑,我们采用实际的开源拓扑图数据 Rocketfuel topologies[10] 作为数据集,具体信息如表 1 所示,并以此设计了两套实验方案. 方案一为单目的节点的网络更新情景,即更新全网其他节点到单个目的节点的路径,用以仿真网络的细粒度更新诸如虚拟机迁移等. 在实际操作中,我们随机选取拓扑中任一节点作为目的节点. 方案二是多节点的网络更新,目的是为了模拟大规模流量迁移的情形,其中待更新数据流可以是全网任意两个节点间的任意流.

在实际仿真中,我们计算初始拓扑的最短路径作为初始路径,然后随机改变全网 10% ,40% 的链路的权值,变化后权值可取初始拓扑链路权值范围内的任意值,并在改变后的拓扑上重新运行最短路径算法,返回的最短路径作为最终的新路径. 为了验证算法的优越性,我们将算法分别与最新提出的 GPIA + FED 算法[8]、FLUS 算法[11] 以及基于传统 two-phase commit 机制的算法[6] 进行比较,为了保证实验数据的有效性,我们将实验重复 200 次,并将数据取平均值.

表 1 实验拓扑数据集

拓扑	AS1221	AS1239	AS3967	AS3257
节点数目	104	315	79	121
边数	306	1944	294	656

3.2 实验结果

在实验中我们主要关注算法对 TCAM 额外开销的节省、算法时间成本等性能参数,特别地,因算法引入了分段路由机制,SL 的长度分布也是测度额外流量负担的直观指标,所以本节最后我们也对 SL 的长度分布进行讨论.

3.2.1 TCAM 资源节省

为了比较各个算法对额外 TCAM 资源的需求程度,我们将以相同情况下 two-phase commit 机制消耗的额外 TCAM 资源作为基准,计算其他算法对 TCAM 资源的消耗,并与 two-phase commit 机制进行对比. 定义算法的 TCAM 冗余削减为 $(N_{TPC} - N_x)/N_{TPC}$,其中 N_{TPC} 为 two-phase commit 机制更新网络需要消耗的 TCAM 规则条目,N_x 为相应算法在相同条件下更新网络消耗的 TCAM 规则条目,如 N_{FCUA} 代表 FCUA 算法更新网络需要消耗的 TCAM 规则条目.

实验结果在图 2 中以互补累计分布函数(CCDF)的形式展示,图中曲线的任意点 (x,y) 代表该算法相对 two-phase commit 机制,在 $y*100\%$ 的实验中至少节省 $x*100\%$ 的规则条目. 实验结果表明,相较于其他算法,FCUA 在整体上可以实现更大比例的 TCAM 资源节省,特别是在节省的最小值上. 如在图 2(*a*) 的拓扑 1221 中,FCUA 至少可以削减 75% 的额外 TCAM 冗余,比 FLUS,GPIA + FED 至少提高 15% 与 65%.

值得注意的是,因为 FCUA 需要在入口节点添加相应的分段路由规则,无法实现零冗余的更新,而

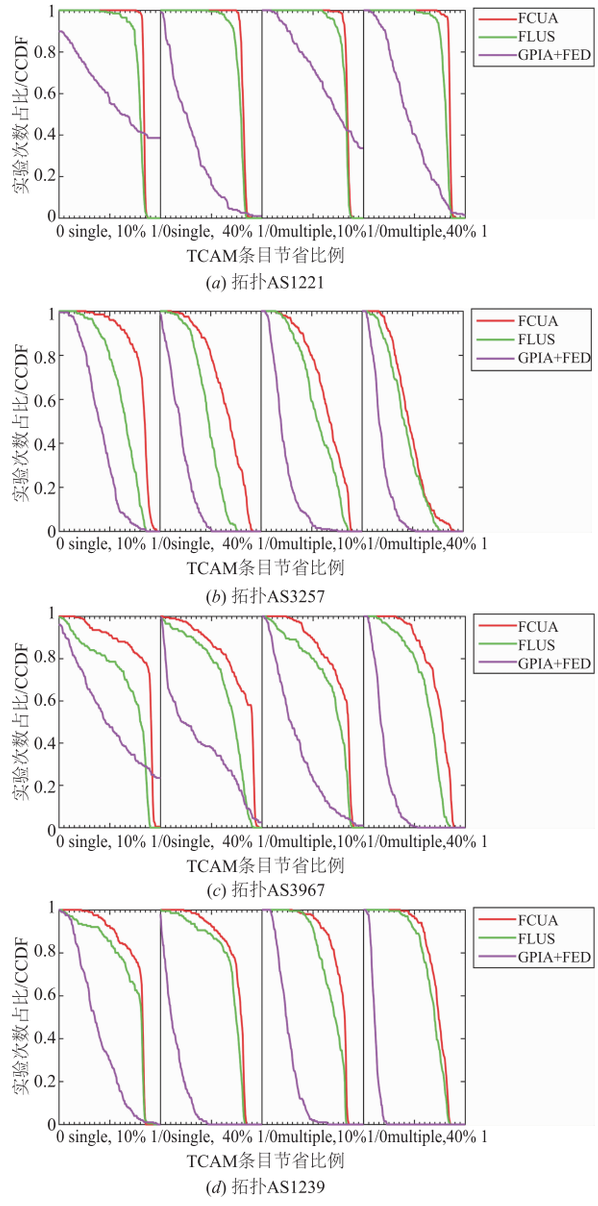

图2 TCAM资源节省的互补累计分布图

GPIA + FED 在小规模的流量更新时,有可能实现零冗余的更新. 但是这种零冗余更新不够可靠,因为在相同条件下未实现零冗余更新的实验中,算法依旧造成了很高的 TCAM 额外消耗,并且在部分实验中 GPIA 算法失效,只能完全采用基于两步复制的机制的 FED 来完成更新,如图 2(*a*)与图 2(*c*)中单目的节点,10% 链路变化条件下算法的实验结果. 最后,在所有实验拓扑中,随着实验环境的复杂与更新流量规模的增大,GPIA + FED 算法的性能明显恶化,而 FCUA 算法的性能基本保持不变.

实验结果证明 FCUA 算法在小规模流量变化更新时,具有更好的性能稳定性,在大规模流量更新时候,性能更佳更适用. 这是由于分段路由机制与顺序更新机制可以互为补充,形成一种比单一机制性能更加优秀且稳定的协同更新机制,当单一更新机制更新效果不佳时,另一机制尚能有效更新剩余节点,以保证网络中的绝大部分结点在启用 two-phase commit 机制前得到更新,从而有效削减 TCAM 资源冗余开销,拥有更加稳定的算法性能.

3.2.2　更新时间

算法运行环境为 Ubuntu 14.04 系统,硬件环境配置为 2.2 GHz Intel Xeon E5-2407 CPU 与 16GB 的 DDR3 内存. 我们分别计算 200 次算法运行时间的平均值作为反映算法时间成本的指标. 实验结果如表 2 所示,为了简洁,表 2 仅仅显示算法在最复杂的拓扑 AS1239 的运行时间.

表 2　算法运行时间

实验条件	FCUA	FLUS	GPIA + FED
单目的节点,10% 链路改变	7.912s	7.714s	5.714s
单目的节点,40% 链路改变	7.629s	7.582s	5.523s
多目的节点,10% 链路改变	7.920s	7.917s	5.518s
多目的节点,40% 链路改变	8.963s	8.898s	6.077s

实验结果表明,FCUA 算法的运行时间比 FLUS 算法与 GPIA + FED 算法要略长,这是由于 FCUA 算法在启用 two-phase commit 之前需要经过两步机制的处理,而 FLUS 算法与 GPIA + FED 算法只采取一步处理,但这仅仅是运行时间的线性增加,对于大部分应用来讲是可以接受的.

3.2.3　SL 的长度分布

分段路由机制引入的 SL 带来了额外的带宽消耗,故 SL 的长度是测度所需额外带宽的直接指标. 为了简洁,我们只在表 3 中列出了最复杂实验拓扑 1239 在各种实验条件下的 SL 长度分布. 结果显示,至少 44.04% 的数据流只需要一个 SID 便可以完成更新,而绝大部分的数据流(≥93.39%)可以在 SL 长度小于等于 3 的情况下完成更新,而在 SID 等于 6 时,所有数据流都可以完成更新. 所以说,FCUA 算法只需要引入很小的流量负担便可以完成更新过程.

表 3　SID 序列的长度分布

实验条件	SL = 1	SL≤ 2	SL≤ 3	SL≤ 4	SL ≤ 5	SL≤6
单目的节点,10% 的链路权值改变	62.56%	90.85%	97.99%	99.73%	99.98%	100%
单目的节点,40% 的链路权值改变	44.04%	77.52%	93.39%	98.73%	99.80%	100%
多目的节点,10% 的链路权值改变	71.11%	93.84%	99.04%	99.82%	99.99%	100%
多目的节点,40% 的链路权值改变	51.19%	80.92%	94.34%	98.60%	99.74%	100%

4　结论

在本文中,我们针对 SDN 网络一致性更新提出一种多机制协同的普适算法. 实验表明该算法具有良好的更新效能,可以在保证更新过程的强一致性的前提下,大量减少更新时的 TCAM 冗余,并且适用性好,算法性能稳定,对保证网络安全与正常运行有较大意义.

参考文献

[1] MCCLURG J, HOJJAT H, FOSTER N. Efficient synthesis of network updates[J]. ACM Sigplan Notices, 2015, 50(6): 196 – 207.

[2] WANG W, HE W, SU J, et al. Cupid: Congestion-free consistent data plane update in software defined networks[A]. Proceedings of the 35th Annual IEEE International Conference on Computer Communications[C]. San Francisco: IEEE, 2016. 1 – 9.

[3] LUO S, YU H, LUO L, et al. Arrange your network updates as you wish[A]. Proceedings of IFIP Networking Conference and Workshops[C]. Vienna: IEEE, 2016. 10 – 18

[4] FÖRSTER K T, MAHAJAN R, WATTENHOFER R. Consistent updates in software defined networks: on dependencies, loop freedom, and blackholes[A]. Proceedings of IFIP Networking Conference and Workshops[C]. Vienna: IEEE, 2016. 1 – 9.

[5] VISSICCHIO S, CITTADINI L. Safe, efficient, and robust SDN updates by combining rule replacements and additions[J]. IEEE/ACM Transactions on Networking, 2017, 25(5): 3102 – 3115.

[6] REITBLATT M, FOSTER N, REXFORD J, et al. Abstrac-

tions for network update[J]. ACM SIGCOMM Computer Communication Review,2012,42(4):323 – 334.
[7] KATTA N P,REXFORD J,WALKER D. Incremental consistent updates[A]. Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking[C]. Hong Kong:ACM,2013. 49 – 54.
[8] VISSICCHIO S,VANBEVER L,CITTADINI L,et al. Safe Update of Hybrid SDN Networks[J]. IEEE/ACM Transactions on Networking,2017,25(3):1649 – 1662.
[9] FILSFILS C,NAINAR N K,PIGNATARO C,et al. The segment routing architecture[A]. Proceedings of Global Communications Conference[C]. San Diego:IEEE,2015. 1 – 6.
[10] Spring N,Mahajan R,Wetherall D. Measuring ISP topologies with rocketfuel[J]. IEEE/ACM Transactions on Networking,2004,12(1):2 – 16.
[11] LUO L,YU H,LUO S,et al. Achieving fast and lightweight SDN updates with segment routing[A]. Proceedings of Global Communications Conference[C]. Washington:IEEE,2017. 1 – 6.

作者简介

于倡和 男,1993 年 10 月出生,山东青岛人,2016 年毕业于上海交通大学,现国家数字交换系统工程技术研究中心在读硕士,主要研究方向为新型网络体系结构与网络安全.
E-mail:Yu_Changhe@ hotmail. com

兰巨龙 男,1962 年出生,河北张北人,国家数字交换系统工程技术研究中心教授,博士生导师,主要研究方向为新型网络体系结构与网络安全.