基于分类的软件定义网络流表更新一致性方案

周 烨 杨 旭 李 勇* 苏 厉 金德鹏 曾烈光
(清华大学电子工程系 北京 100084)

摘 要：软件定义网络是近年来的研究热点，流表更新问题则是其中的一个重要问题。该文提出基于分类的流表更新一致性方案，在保证更新一致性的同时，具有通用性强、有效减轻控制负载等特点。文中引入逻辑证明来验证所提方案能够保证流表更新一致性。多个场景的仿真结果显示，与相关研究相比，该文方案有更好的通用性，更新时间基本一致，能有效降低控制负载。

关键词：软件定义网络；流表更新；流表更新一致性
中图分类号：TP393.0 **文献标识码**： A **文章编号**：1009-5896(2013)07-1746-07
DOI: 10.3724/SP.J.1146.2012.01431

Classification Based Consistent Flow Update Scheme in Software Defined Network

Zhou Ye Yang Xu Li Yong Su Li Jin De-peng Zeng Lie-guang
(*Department of Electronic Engineering*, *Tsinghua University*, *Beijing* 100084, *China*)

Abstract: Software Defined Network (SDN) becomes a hot topic in recent years, in which flow update is one of important issues. In this paper, a classification based consistent flow update scheme is proposed, which has the advantage of strong generality and efficiently lightening working load on controller. The logic synthesis is also used to demonstrate how the scheme maintains consistency in the whole updating process. Simulation results in multiple scenarios show that the proposed scheme achieves better generality than existing schemes in related work. In addition, the scheme reduces significantly working load on controller while only costs nearly the same time in the update process as compared schemes.

Key words: Software Defined Network (SDN); Flow update; Consistent flow update

1 引言

软件定义网络(Software Defined Networking, SDN)是近年来提出的一种新型网络体系结构，实现数据平面和控制平面的解耦[1]。在数据平面中，交换机等网络核心设备负责数据包的处理；在控制平面中，存在集中式控制器，与各个交换机通过控制链路相连，通过控制交换机中的流表项来指导数据包的处理。另外，SDN 还具有虚拟化、可编程等特性[2]。

近年来，SDN 正在逐渐成为学术界和业界的研究热点。一方面，SDN 在目前主流网络创新实验环境中广泛应用，例如 GENI[3], FIRE[4], AKARI[5], TUNIE[6]等。另一方面，SDN 在数据中心等热点研究领域中也有丰富应用场景，例如流量均衡、集中式管理等[7]。然而，SDN 很多关键技术目前尚未有成熟解决方案，需要研究者进行深入研究。

与现有互联网类似，网络状态的变化在 SDN 中也频繁出现，例如路由切换、流量均衡、网络设备维护等。这些网络状态的变化，需要以更新交换机中流表项的方式，来实现对数据包的不同处理。SDN 流表更新问题，即是从旧规则更新到新规则，需要在多个交换机上用一套新规则的流表项来替换旧规则的流表项。在流表更新过程中，任意数据包或者在每个交换机上只按照旧规则中的流表项处理，或者在每个交换机上只按照新规则中的流表项处理，而不能在不同交换机上分别受旧、新规则中流表项的处理，这就是流表更新一致性问题。

目前，针对流表更新一致性的研究刚刚起步，研究者已经提出若干方案，并提出若干评价流表更新一致性方案的指标，例如通用性、隔离性、更新时间、控制负载等[8−10]。

文献[8]提出基于额外标签(VLAN)的流表更新一致性方案，核心思想是以额外的数据标签(VLAN)

2012-11-08 收到，2013-03-15 改回
国家 973 计划项目(2013CB3291005)，国家自然科学基金(61171065, 61021001)，国家 863 计划项目(2013AA010601, 2013AA010605)和中兴通讯股份有限公司产学研合作基金资助课题
*通信作者：李勇 liyong07@tsinghua.edu.cn

来区分新、旧两套规则。首先，在旧规则的所有流表项中，添加匹配信息，即 VLAN=0；其次，修改对应数据包的包头信息，将 VLAN 值设为 0，从而按照旧规则进行处理；再次，在新规则的所有流表项中，添加匹配信息，即 VLAN=1，再将新规则的流表项写入对应交换机；然后，当新规则的所有流表项成功写入后，修改数据包的包头信息，将 VLAN 值从 0 更新到 1，从而按照新规则进行处理；最后，删除旧规则的流表项。

文献[9]提出基于中间规则的流表更新一致性方案，核心思想是构建一组中间规则：只保存同时出现在两套规则中的共同流表项，对于其他流表项则将相应数据包上传给控制器缓存。首先，在各个交换机中写入中间规则的流表项，全部成功写入后等待一个全网最长端到端网络延时，之后删除旧规则的流表项；其次，在各个交换机中写入新规则的流表项，并将之前上传给控制器缓存的数据包发回数据平面，再等待一个全网最长端到端网络延时，之后删除中间规则的流表项。

文献[10]针对 SDN 网络中虚拟机迁移引发的流表更新场景，简单提出一种更新顺序，假设虚拟机从 s 迁移至 d，则所有需要更新流表项的交换机，按离 d 的距离进行升序排列，并依次进行更新。然而，该文献并未证明所提方案能保持流表更新一致性。

综上，这些已有研究方案各有明显不足，例如通用性差、控制负载严重等。本文提出基于分类的流表更新一致性方案，利用 SDN 的集中控制特性，首先对待更新流表的交换机、两套流表规则进行分类，然后根据分类结果进行流表更新。多个场景的仿真结果显示，与已有研究方案相比，本文方案具有较强的通用性，在更新时间基本一致的同时，有效降低控制负载。

2　SDN 流表更新一致性

在 SDN 中，控制器通过对交换机中的流表项进行更新，实现 SDN 中网络状态的变化。具体地，需要在多个交换机上进行多条流表项的更新，即从旧规则更新到新规则，这就是流表更新问题。然而，由于数据平面的分布式特性，新规则的所有流表项可能无法同时生效，进而可能引起数据包的错误处理。为了保证数据包在更新过程中被正确处理，研究者提出流表更新一致性，具体定义如下：任意数据包在传输过程中，或者只按旧规则的流表项处理，或者只按新规则的流表处理，而不能在不同交换机上依次按旧、新规则中的流表项处理[8−10]。

在 SDN 的原有流表更新机制中，控制器依次向多个交换机写入新规则对应的流表项，由于这些流表项的生效时间有先后，数据包可能依次按旧、新规则中的流表项处理，即不能保证更新一致性。为了保证一致性，控制器也可为每次更新事件选择一个特定的更新顺序，逐一写入新的流表项，然而这种方案有如下不足：(1)每次更新时都需重新选择一个特定的更新顺序，增加管理复杂度，且不具有通用性；(2)可扩展性较差，随着网络规模的扩大、待更新流表项数量的增多，选择更新顺序的工作量将急剧增大。可见，上述更新方案都不能高效、通用地解决流表更新一致性问题。另外，已有研究中也尚未有较为成熟的更新方案。因此，针对该问题进行深入研究，有重要意义。

3　SDN 网络建模

图 1 示出一个 SDN 网络拓扑示意图，控制平面由集中式控制器和控制链路组成，数据平面可视为无向图 $G=(V,E)$，交换机的集合为 $V=\{a,b,c,d,e,f,g,h,i,j\}$，物理链路的集合为 E。控制器通过向各个交换机写入、修改、删除流表项来实现对数据平面的控制。当进行流表更新时，记旧、新规则的流表项集合分别为 R_1, R_2，记需要进行流表更新的交换机集合为 $V_0 \subseteq V$。

由于 SDN 采取集中控制方式，控制器能够根据触发流表更新的具体事件，对受流表更新影响的交换机、两套流表规则进行细分。首先，将 V_0 中交换机细分为 2 类：(1)初始交换机 $V_1 \subseteq V_0$，假设数据包按新规则处理，经过 V_0 中的第 1 个交换机记为初始交换机。(2)后继交换机 $V_2 \subseteq V_0$，即 V_0 中除 V_1 之外的其它交换机。其次，将 R_1，R_2 的流表项细分为 4 类：(1)待新增流表 R_a，控制器对新数据包制定的处理规则；(2)待修改流表 R_m，控制器对已有数据包的处理规则修改；(3)待删除流表 R_d，当 R_2 全部生效后，控制器需要删除 R_1 中的相应流表；(4)共同流表 R_s，即在两套流表中都存在的流表。

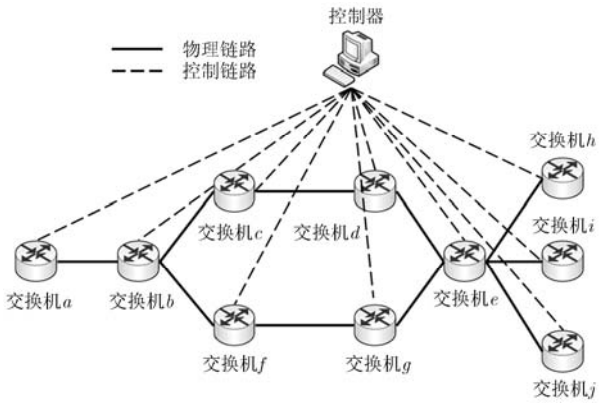


图 1 SDN 网络拓扑示意图

4 基于分类的流表更新一致性方案

本文提出一种基于分类的更新方案，即根据V_0中 2 类交换机及R_1和R_2中 4 类流表项的不同特性，依次进行流表更新，旨在保证更新一致性的同时，有效减轻控制器负载、提高方案的通用性、不修改数据包包头信息。在本节，首先详细介绍所提方案，然后从逻辑上证明该方案能保证更新一致性，接着描述 2 个实施示例，最后将该方案与相关研究进行比较。

4.1 方案描述

如图 2 所示，本文提出的方案分为 5 个阶段，分别介绍如下：

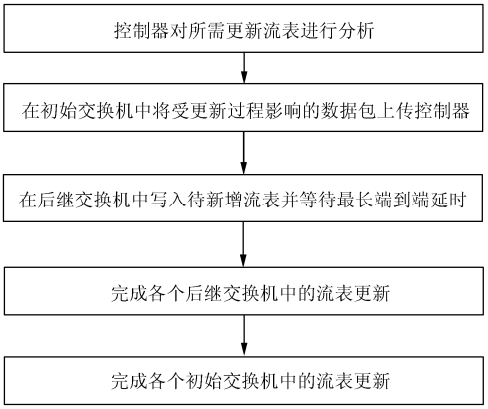


图 2 基于一致性的 SDN 网络流表更新方案

第 1 阶段 控制器对两套流表规则和交换机进行分类。控制器完成对新、旧两套流表的分析，并细分为R_a,R_m,R_d和R_s；控制器明确需要进行流表更新的交换机V_0，并细分为V_1和V_2。

第 2 阶段 在初始交换机中将受更新过程影响的数据包上传给控制器。在更新过程中，除了与R_s相关的数据包，其它数据包的传输都受影响。在本阶段，控制器向各个初始交换机发出控制指令：只保留R_s，同时将其它流表对应的数据包上传给控制器，控制器对这些上传的数据包进行缓存。本阶段结束时，V_1各个交换机上受更新过程影响的数据包都已上传给控制器。

第 3 阶段 在后继交换机中写入待新增流表，并等待全网端到端最长延时。在本阶段，并行进行 2 项操作：(1)在V_2各个交换机上写入待新增流表，由于此时初始交换机尚未进行流表更新，因此后继交换机中写入的流表项不会生效。(2)在本阶段开始之前，网络中仍然有数据包按照旧规则处理，因此需等待一个全网的端到端最长延时，以使这些数据包全部处理完毕。本阶段结束时，V_2各个交换机上已写入待新增流表，并且R_1中除R_s之外的其它流表项都不发挥作用。

第 4 阶段 完成各个后继交换机的流表更新。具体地，控制器向各个后继交换机上写入待修改流表，并删除待删除流表。本阶段结束时，各个后继交换机已完成流表更新。

第 5 阶段 完成各个初始交换机的流表更新。具体地，控制器向各个初始交换机上写入待新增流表、待修改流表，并删除待删除流表。完成这些更新之后，不再将数据包上传给控制器，并将控制器缓存的数据包发回网络中按照新规则处理。本阶段结束时，各个初始交换机已完成流表更新。

至此，流表更新过程结束，新规则的流表项已经写入各个交换机，数据包按照新规则进行处理。下面，再从时序的角度描述上述方案，不失一般性，假设V_1,V_2中各包含 2 个交换机。同时，假设在t_0时刻，控制器已经完成对待更新流表、交换机的分类，即已完成第 1 阶段。

(1)在t_0时刻开始第 2 阶段，控制器向V_1中 2 个交换机依次发送控制指令。这 2 个交换机分别从t_1,$t_2(t_1<t_2)$时刻起，开始将相应数据包上传给控制器。那么，第 2 阶段在t_2时刻结束。在t_2时刻之前，仍然有数据包从V_1流向V_2；在t_2时刻之后，不再有数据包从V_1流向V_2。

(2)在t_2时刻开始第 3 阶段，控制器向V_2中 2 个交换机依次发送控制指令，以写入待新增流表；同时，等待一个端到端最长延时t_d，以使t_2时刻之前流向V_2交换机的数据包按照旧流表完成处理过程。假设V_2中 2 个交换机分别在t_3,$t_4(t_3<t_4)$时刻成功写入待新增流表，那么，第 3 阶段在$t_5=\max\{t_4,t_2+t_d\}$时刻结束。

(3)在t_5时刻开始第 4 阶段，控制器向V_2中 2 个交换机依次发送控制指令，进行流表更新。假设V_2中 2 个交换机分别在t_6,$t_7(t_6<t_7)$时刻成功完成上述操作，那么第 4 阶段在t_7时刻结束。

(4)在t_7时刻开始第 5 阶段，控制器向V_1中 2 个交换机依次发送控制指令，进行流表更新。假设V_1中 2 个交换机分别在t_8,$t_9(t_8<t_9)$时刻成功完成上述操作，那么，第 5 阶段在t_9时刻结束。从t_9时刻开始，所有数据包都按新规则进行处理。

由上述时序描述可知，本文所提方案能够保持流表更新一致性：从t_2时刻开始，相应数据包从初始交换机上传给控制器；而t_2时刻之前的数据包继续按照旧流表处理，并于t_5时刻之前全部处理完毕；从t_5到t_8时刻，相应数据包仍然从各初始交换机上传给控制器；V_1中交换机分别在t_8,t_9时刻完成流表

更新，之后数据包按新规则处理；在 t_8 与 t_9 时刻之间，一个初始交换机完成流表更新，并按新规则处理数据包，另一个则继续将数据包上传给控制器，也不影响流表更新的一致性。

4.2 一致性的逻辑证明

除了从时序的角度来验证流表更新一致性之外，本文还借鉴相关研究中的思路，对一致性进行逻辑证明。在进行逻辑证明之前，首先描述一些相关符号标记：将交换机 s 对数据包 p 进行处理的过程看作一个函数 $f^s: X \to Y$，其中，X 是数据包的包头内容，Y 是交换机对数据包进行的处理，例如转发、丢包等[11]。将交换机 s 基于 R_1, R_2 对数据包进行处理的函数分别记为 f_1^s, f_2^s。当数据包经过多个交换机，例如 $s \to t \to u \to v$，用函数 $F(p) = f^v(f^u(f^t(f^s(p))))$ 来表示整个处理过程，为了区分旧、新规则，分别记为 F_1, F_2。将数据包从某个交换机 i 开始的后继处理函数记为 $F^i(p)$，例如 $F^s(p) = f^v(f^u(f^t(p)))$。如果数据包在处理过程中上传到控制器，则记 $F(p) = c$。那么，流表更新一致性问题有如下形式：当流表从 R_1 更新到 R_2，任一数据包 p 的处理函数 $F(p)$ 或者表示为 $F_1(p)$，或者表示为 $F_2(p)$。下面证明，本文提出的方案能够保持流表更新一致性。

证明　首先证明，从 t_0 至 t_5 时刻任一数据包 p 的处理函数 $F(p) = F_1(p)$ 或者 $F(p) = c$。假设数据包 p 在网络中依次经过的交换机序列为 $s_1 \to s_2 \to \cdots \to s_n$，根据方案描述，从 t_0 至 t_5 时刻存在 2 种可能情形：(1)数据包 p 经过每个交换机，最终成功传输完毕，那么对于每个交换机 $s_i (1 \le i \le n)$，$f^{s_i}(p) = f_1^{s_i}(p)$，即都按照 R_1 处理，因此 $F(p) = F_1(p)$；(2)数据包 p 没有成功传输完毕，即在初始交换机中被上传给控制器，因此 $F(p) = c$。综上，从 t_0 至 t_5 时刻 $F(p) = F_1(p)$ 或者 $F(p) = c$。

其次证明，从 t_5 至 t_9 时刻任一数据包 p 的处理函数 $F(p)$ 最终都可以表示为 $F_2(p)$。截至 t_5 时刻，网络中已有的数据包，都按照 R_1 处理完毕；从 t_5 时刻开始，数据包继续从初始交换机上传至控制器。截至 t_7 时刻，各个后继交换机完成流表更新；从 t_7 至 t_9 时刻，控制器开始更新各个初始交换机上的流表，更新成功后不再将数据包上传给控制器。因此，从 t_5 时刻至 t_9 时刻，任一数据包 p 首先被上传至控制器，最终被发回网络按照 R_2 处理，即 $F(p) = F_2(p)$。

综上，任一数据包 p 截至 t_5 时刻，$F(p) = F_1(p)$，或者 $F(p) = c$。(1)如果 $F(p) = F_1(p)$，那么该数据包已经传输完毕；(2)如果 $F(p) = c$，数据包上传至控制器，但是最终会发回网络按照 R_2 处理，则 $F(p) = F_2(p)$。那么，在整个流表更新过程中，$F(p) = F_1(p)$ 或者 $F(p) = F_2(p)$，即能够保持流表更新一致性。　　证毕

4.3 实施示例

以图 1 所示的网络拓扑为例，在其中分别进行路由切换、流量均衡。

(1)路由切换　图 3 示出一个路由切换的示例，从 $a \to b \to c \to d \to e$ 切换到 $a \to b \to f \to g \to e$，则 $V_0 = \{b,c,d,f,g\}, V_1 = \{b\}, V_2 = \{c,d,f,g\}$。流表更新内容如下：在 b 上，修改一条流表；在 c 和 d 上，删除旧路由的流表项；在 f 和 g 上，添加新路由的流表项。根据本文所提方案，首先将旧路由中从 a 发到 b 的数据包全部上传给控制器；其次，在 f 和 g 上写入新增流表，同时等待一个 b 到 e 的延时，以使原路由中的所有数据包传输完毕；然后，在 c 和 d 上删除旧路由的流表项，在 b 中修改流表；最后，将之前上传给控制器的数据包重新发回 b，按照新路由进行转发。

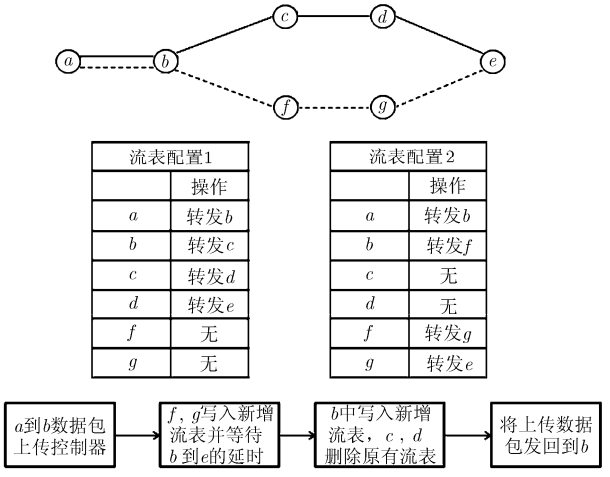


图 3 路由切换实施示例

(2)流量均衡　图 4 示出一个流量均衡的示例，涉及 $V_0 = \{e,h,i,j\}$ 这 4 个交换机，$V_1 = \{e\}$，$V_2 = \{h,i,j\}$。流表更新内容如下：在 e 上，修改现有的 3 条流表，分别对应 3 个后继交换机；在 h, i, j 上各修改一条流表。根据本文所提方案，首先将从 e 发到 h, i, j 的数据包全部上传给控制器；接着，等待一个网络延时，以使正在按照旧规则处理的数据包全部传输完毕；然后，在 h, i, j 上分别写入待修改流表；最后，在 e 中写入待修改流表，并将之前上传给控制器的数据包重新发回 e，按照新规则进行转发。

4.4 与相关研究的比较

研究者对流表更新一致性方案提出如下评价指

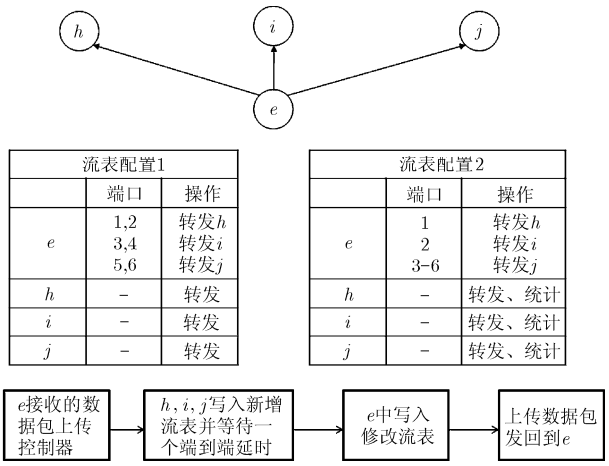


图 4 流量均衡实施示例

标[8–10]：通用性，即考虑方案的适用范围；隔离性，即考虑是否可能对其它控制行为造成影响；更新时间，即实现更新过程的耗费时间；控制负载，即考虑给控制平面带来的额外负载。由于文献[10]未证明所提方案能保证流表更新一致性，在本节只将本文所提方案与文献[8]、文献[9]的方案进行比较，如表 1 所示。

表 1 本文所提方案与相关研究比较

方案	通用性	隔离性	更新时间	控制负载
本文	强	良好	长	轻
文献[8]	差	较差	短	无
文献[9]	强	良好	长	重

(1)通用性　文献[8]方案使用 VLAN 标签来区分两套流表，需要将 VLAN 信息添加为流表项匹配信息的一部分；如果新旧两套规则中已将 VLAN 信息作为流表项匹配信息，则该方案失效。另外，文献[8]方案需要同时更改所有数据包的 VLAN 标签，这要求只有 1 个初始交换机，如果有多个初始交换机，则该方案也失效。因此，文献[8]方案通用性差。本文所提方案和文献[9]方案由于不更改流表项的匹配信息，而只是将数据包上传控制器，因此通用性较强。

(2)隔离性　在流表更新过程结束后，文献[8]方案修改了数据包的包头信息，即修改 VLAN 标签，这可能导致数据包被错误处理，因此隔离性较差。本文所提方案和文献[9]方案不修改数据包包头信息，隔离性良好。

(3)更新时间　由于文献[9]方案引入一套中间规则，流表更新时间要长于文献[8]方案。本文所提方案为了减轻控制负载，分阶段对不同交换机进行流表更新，流表更新时间近似等于文献[9]方案。

(4)控制负载　即在更新过程中上传给控制器的数据包。文献[8]方案不会带来额外的控制负载，本文所提方案和文献[9]方案由于需要将数据包上传给控制器，从而有额外控制负载。与文献[9]相比，本文对控制负载进行优化。

综上，文献[8]方案受限于通用性和隔离性，而文献[9]方案和本文所提方案则相对更优。同时，本文所提方案对控制负载进行优化，所需更新时间基本一致，因此又优于文献[9]方案。

5 仿真分析

基于图 1 的网络拓扑，首先在单个初始交换机的场景中进行仿真，其次在多个初始交换机的场景中进行仿真。在本部分，记文献[8] 、文献[9]所提方案分别为方案 1、方案 2。重要的仿真环境参数设置如下：每条物理链路的延时，服从 0 到 100 个时间单位的均匀分布；控制器向交换机中成功写入流表项的延时，服从 0 到 10 个时间单位的均匀分布；数据包到达服从泊松分布，速率为 n 。

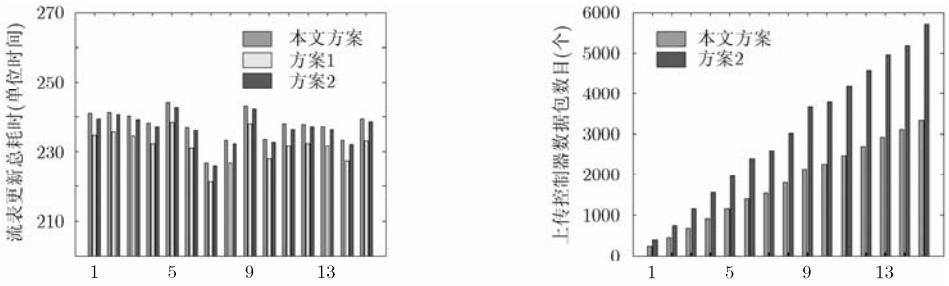
5.1 单个初始交换机的场景

将第 3 节的 2 个实施示例(路由切换、流量均衡)结合，则 $V_1=\{b\}$；在此场景中，本文方案、方案 1、方案 2 都适用，因此对这 3 个方案的更新时间、控制负载进行分析。根据数据包到达速率 n 的不同，设置多组仿真场景，每组进行 100 次重复试验，并以试验结果的平均值作为仿真结果，如图 5 所示：3 个方案所需更新时间基本相近，方案 1 略好于本文方案、方案 2；方案 1 没有控制负载，本文方案的控制负载远低于方案 2，均能实现约 40%的降低。

各个方案所需流表更新时间主要受物理链路延时、流表项写入延时的影响。由于物理拓扑固定，物理链路延时相同。然而，为了实现更新，各个方案写入的流表项数量不同，因此流表项写入延时不同。在更新过程中，方案 1 总计写入 $|R_1|+|R_2|+2$ 条，方案 2 需写入 $|R_1|+|R_2|-|R_s|$ 条，本文方案仅需写入 $|V_1|+|R_2|-|R_s|$ 条。可见，本文方案需要写入的流表项数量最少。如果 $|R_s|=0$，$|R_1|=|R_2|$，则本文方案写入流表项仅约为方案 1、方案 2 的一半。由于控制链路的通信带宽有限，随着两套规则中流表项数量的增加，本文方案将因只需写入较少流表项，从而降低流表项写入延时，进而降低总更新时间。

5.2 多个初始交换机的场景

考虑 2 个数据流同时进行路由切换，第 1 个数



(a)流表更新时间　　(b)控制负载

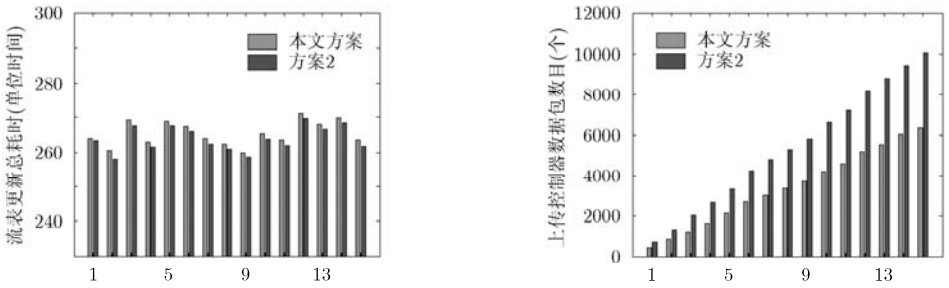
图 5 单个初始交换机场景的仿真结果

据流从 $a \to b \to c \to d \to e$ 到 $a \to b \to f \to g \to e$，第 2 个数据流从 $e \to g \to f \to b \to a$ 到 $e \to d \to c \to b \to a$，则 $V_1 = \{b,e\}$。在此场景中，由于存在 2 个初始交换机，方案 1 无法同时对 2 个数据流的 VLAN 标签进行更改，因此失效，而本文方案和方案 2 则继续适用。因此，接下来仅对本文方案和方案 2 的更新时间、控制负载进行分析。同样地，根据数据包到达速率 n 的不同，设置多组仿真场景，仿真结果如图 6 所示：本文方案更新时间与方案 2 基本一致，控制负载降低约 40%。

综上，本文方案和方案 2 适用于单个、多个初始交换机的场景，而方案 1 仅适用于单个初始交换机的场景，可见本文方案和方案 2 有更强的通用性。另外，与方案 2 相比，本文方案在更新时间基本一致的同时，能够有效降低控制负载，可见本文方案优于方案 2。

6　结束语

本文提出基于分类的 SDN 网络流表更新一致性方案，在保证流表更新一致性的同时，能够保证较好的通用性、隔离性，且有效降低控制负载。仿真结果表明，与方案 1 相比，本文方案有更好的通用性；与方案 2 相比，本文方案保持更新时间基本一致，同时有效降低控制负载约 40%。目前，针对 SDN 流表更新一致性的研究刚刚起步，值得进一步深入研究。本文所提方案是一种通用的流表更新方案，未来可以针对某个具体应用进行相关优化，例如，针对基于 SDN 的数据中心网络中的流量均衡场景，设计高效的流表更新一致性方案。



(a)流表更新时间　　(b)控制负载

图 6 多个初始交换机场景的仿真结果

参 考 文 献

[1] McKeown N. Software-defined networking[OL]. http://klamath.stanford.edu/~nickm/talks/infocom_brazil_2009_v1-1.pdf. 2012.10.

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

[3] GENI. GENI: Global Environment for Network Innovations [OL]. http://www.geni.net/. 2012.10.

[4] FIRE. FIRE: Future Internet Research and Experimentation [OL]. http://cordis.europa.eu/fp7/ict/fire/. 2012.10.

[5] AKARI. AKARI architecture design project[OL]. http://akari-project.nict.go.jp/eng/index2.htm. 2012.10.

[6] 周烨, 李勇, 苏厉, 等. 基于虚拟化的网络创新实验环境研究[J]. 电子学报, 2012, 40(11): 2152–2157.
Zhou Ye, Li Yong, Su Li, *et al.*. Research of network

innovation experimental environment based on network virtualization[J]. *Acta Electronica Sinica*, 2012, 40(11): 2152–2157.

[7] Jarschel M and Pries R. An openflow-based energy-efficient data center approach[C]. Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, New York, 2012: 87–88.

[8] Reitblatt M, Foster N, Rexford J, *et al.*. Abstractions for network update[C]. Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, New York, 2012: 323–334.

[9] McGeer R. A safe, efficient update protocol for openFlow networks[C]. Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, New York, 2012: 61–66.

[10] Ghorbani S and Caesar M. Walk the line: consistent network updates with bandwidth guarantees[C]. Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, New York, 2012: 67–72.

[11] McGeer R. Verification of switching network properties using satisfiability[C]. Proceedings of IEEE ICC, Ottawa, 2012: 6638–6644.

周 烨：男，1986年生，博士生，研究方向为网络虚拟化、下一代网络、SDN.

李 勇：男，1985年生，博士后，研究方向为下一代网络、移动容迟网络、移动机会网络、SDN等.

曾烈光：男，1947年生，教授，研究方向为通信网、ASIC设计、片上网络、下一代网络、SDN等.