doi:10.11959/j.issn.1000-436x.2017223

OpenFlow 中基于精确时间戳的延迟测量方法

邱欣逸[1,2]，李俊[1]，周建二[1]，李菁菁[1]
（1. 中国科学院计算机网络信息中心，北京 100190；2. 中国科学院大学，北京 100049）

摘 要：目前的 OpenFlow 延迟测量方法占用较多的网络资源，且测量精度较差。根据 SDN 集中控制的特性，在 OpenFlow 网络中设计一种基于准确时间戳的延迟测量方法。使用控制器动态计算出任意 2 个交换机之间的多条可达路径，只需发送一个探测分组就可以测量出多条路径上的延迟，减少了网络资源的消耗。同时，为了提高准确性，将时间戳的获取从控制器转移到 OpenFlow 交换机中。仿真实验证明，与其他测量方法相比，该方法的测量精度与稳定性都有较大提升。

关键词：OpenFlow；软件定义网络；延迟测量；时间戳
中图分类号：TP393 **文献标识码**：A

New delay measurement method based on accurate timestamp in OpenFlow

QIU Xin-yi[1,2], LI Jun[1], ZHOU Jian-er[1], LI Jing-jing[1]
(1. Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China;
2. University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: At present, delay measurement methods in OpenFlow network have the disadvantage of excessive network resources and poor measurement accuracy. DeMon, an active mechanism to measure the delay of multiple paths between any two switch based on the controllable feature of individual traffic flow provided in OpenFlow was proposed. DeMon required only one probe packet to be send from controller, which was excepted to reduce the operational cost. Moreover, DeMon used OpenFlow switch instead of controller to get the timestamp of probe packet, making the measurement accuracy and stability have been greatly improved compared with other monitoring techniques in the OpenFlow network.
Key words: OpenFlow, software defined network, latency monitoring, timestamp

1 引言

网络延迟是网络故障分析和流量调度的重要依据[1]。传统的网络延迟测量分为主动与被动 2 种测量方式。被动测量因为涉及设备安装、时间同步等一系列问题，大大降低了方法的可扩展性，在大规模网络中部署较为困难。而像 ping、traceroute 等传统的主动测量方法[2]或 pingMesh 工具[3]，对于路径的选择依赖于传统分布式路由协议的计算结果，以致用户无法按指定的路径进行延迟测量[1]。同时，选择的路径会随着路由协议的变化而发生改变，这也增加了延迟测量的不确定性[4,5]。在传统技术中，为了能全方位地检测网络情况，通常需要增加大量额外的设备[6,7]，这对运用商而言，将带来巨大的开销[8]。

近几年，SDN（software defined network）[9]技术不断发展，Google、腾讯、华为等业界主流公司已有基于 SDN 的实际落地方案。SDN 中提出转控分离的新思路，将路由决策从路由器转移到了控制器当中，且路由方式也由分布式转为集中式计算，这为用户指定特定路线进行延迟测量提供了可能。在 SDN 中，OpenFlow[10]协议已经成为事实上的标准，但目前，OpenFlow 协议中的延迟测量方法尚未成熟。

针对 OpenFlow 中延迟测量问题，国内外学者

收稿日期：2017-04-28；**修回日期**：2017-10-28
基金项目：国家自然科学基金资助项目（No.61672490）
Foundation Item: The National Natural Science Foundation of China(No.61672490)

提出的解决方案大体分为 2 种。第一种方案完全依赖于控制器，控制器需要完成路线计算、下发探测分组、记录时间戳、计算延迟等多种任务。在第二种方案中，控制器只需完成路线计算和下发探测分组等部分步骤，而获取时间戳和延迟计算等对时间精度要求较高的行为则转移到额外的设备上。对于第一种方案，控制器产生的结果难以保证稳定与准确。当控制器需要处理的事物增多时，测量的结果会产生较大的波动。而对于第二种方案，额外的设备会带来较大的开销，并且该方案要求交换机与测量设备之间的线路延迟不能有较大变化。这 2 种方案对网络环境都有较高要求，具有一定局限性。

本文所提方案结合了上述 2 种方案的优点，将获取时间戳等时间敏感的动作转移到交换机上，保证了控制器在业务繁忙时也能计算出正确结果。同时，采用以链路为单位的延迟计算方案，保证每条链路不被重复测量，控制器只需发送一个探测分组就能完成多条路径的测量。通过计算单条链路的往返延迟，避免了多台设备之间的时间同步问题，为用户提供了灵活、准确的延迟测量方案。和 GRAMI 等测量方法相比，减少了测量误差和方差。

2　相关工作

随着 SDN 技术的发展，对 OpenFlow 网络中延迟测量的研究，引起了越来越多人的关注。文献[11]首先提出基于控制器进行延迟测量。该方案先将用户输入的路径信息分解为一个链路集合，控制器依次向每个链路的起始节点发送探测分组，并接收链路终点返回的探测分组，再将发送与接收探测分组的时间差，减去控制器到链路起始交换机之间的往返延迟，以得到该链路的延迟[11]。这一方案可以在不改变现有网络行为的基础上进行延迟测量，但是由于该方案属于单向时延测量方法，存在时间同步问题。解决单向时延测量中时间同步问题的方法可分为硬件精度改进和数学分析 2 种[12~15]。硬件精度改进以 Google 为例，在 Spanner 项目[15]中通过在数据中心接入原子钟和 GPS 接收器保证了各设备之间的时钟同步。但是该方法价格昂贵，无法在现网中实施。而用数学分析方法进行时间同步存在一定的误差，并且需要长时间、全网地进行数据搜集，对网络造成一定负担，无法适应大规模、远距离网络。文献[11]提出的方法中需要控制器向每条链路的起始交换机发送探测分组，给控制器带来较大的负担。控制器是整个 SDN 中的最繁忙的设备，随着功能的增多，已经成为了整个网络的瓶颈[16]。所以使用控制器来获取时间戳会带来较大误差，且难以有效消除。在文献[11]的结论中也发现，由控制器进行时间戳的获取，并进行计算，得出的结果波动较大，单次测量的准确性无法得到保障[11]。

为了计算 OpenFlow 全网延迟，文献[17]提出了 OpenNetMon。该方案与文献[11]提出的方案类似，是通过控制器获取时间戳来计算往返延迟。但 OpenNetMon 不再要求为路径中的每一段链路都发送一个探测分组，而是直接通过计算探测分组到达起点与终点交换机的时间差来获得整条路径延迟[17]。为了获得全网中每条路径的延迟，OpenNetMon 通过轮询的方式进行统计，并根据结果计算出网络的分组丢失率和吞吐量。这一方案减少了测量某条具体路径时需要发送的探测分组的数量，但是没有从根本上解决控制器成为瓶颈的问题，并且发送大量探测分组进行延迟测量为网络带来了不小负担。文献[18]发现，为每条可能线路发送不同的探测分组进行延迟测量，会使网络的分组丢失率上升 5%以上。

Atary 等[1]提出了 GRAMI（GRAnular RTT monitoring infrastructure）解决方案。将探测分组的发送和延迟的计算转移到 MP（monitoring point）中，控制器只进行路径计算，减轻了控制器的工作负担。当交换机接收到探测分组以后不管是不是终点交换机都向 MP 返回探测分组，MP 根据每个路由器返回的探测分组计算出网络中各个链路的延迟。但该方法需要部署新的设备，而且在大型网络中，返回的探测分组需要经过大量的交换机才能到达 MP，这也为延迟的测量带来了很大的不确定性，为测量的结果带来了较大的误差。

以上方法主要针对单条路径或全网路径进行延迟测量，不太适合在网络较为复杂、各链路延迟波动较大的场景。用户在对网络进行分析，或当系统对路径进行选择时，往往希望可以得到指定 2 个交换机之间多条可能路径的延迟，以进行对比和判断，从而获得最优结果。为了解决上述问题，本文提出 DeMon（delay monitoring），用于测量指定交换机之间前 K 条最短路径的延迟。在传统测量延迟方案中，每条路径被当作不可分割的整体进行测量，这样做会导致某些链路被重复测量，并且不利于将路径进行合并，增加了网络资源的消耗。在本方案中，延迟的测量是以链路为单位的，只需测量

出各路径所覆盖的链路的延迟情况，就可以将其进行合并计算以获得各个路径的延迟。控制器只需发送一个探测分组，即可完成各路径的测量，有效减少网络资源的消耗。同时为了使延迟测量更为准确和稳定，将时间戳的获取从控制器转移到了 OpenFlow 交换机当中，解决了因控制器繁忙而造成的误差。此外，DeMon 采用双向时延测量方法，避免了因单向时延测量所带来的时钟同步问题。同时使用 SDN 技术，指定探测分组的往返路径，很好地解决了双向时延测量中因往返路径不一致而带来的测量误差。通过 Mininet 对实验拓扑进行模拟，经过上千次的实验发现，和 GRAMI 等方案相比，DeMon 可将延迟测量误差缩小为原误差的$\frac{1}{3}$，方差减少为原误差的$\frac{1}{5}$。

通过使用 DeMon 测量方法，可以为故障检测和流量调度带来极大的便利。当网络中分组丢失或时延大幅增加时，网络管理人员利用 DeMon 多路径测量的特性可以快速找出任意两点之间的瓶颈链路，定位故障设备。当网络人员需要对指定两点间的流量调度进行优化时，可以定期调用 DeMon，对两点间的多条路径进行延迟监控，及时调整路由策略。

3 DeMon 系统介绍

DeMon 的工作主要分为路线计算以及探测延迟 2 个部分。路线计算的基本步骤如下。首先，利用控制器中存储的网络拓扑信息，计算出用户指定的两交换机之间的可能路径，接着，将这些路径进行合并，最后，控制器将合并好的路径生成对应的流表项下发至 OpenFlow 交换机中。探测延迟部分的基本步骤为控制器向起始交换机发送探测分组，探测分组到达 OpenFlow 交换机之后，交换机为探测分组打上进入交换机的时间戳。接着，根据流表项将探测分组复制、发送给下游的交换机和控制器，并在发送探测分组之前，为每个探测分组打上离开交换机的时间戳。控制器根据返回来的探测分组，计算出每条链路的延迟。最后，将链路延迟进行组合，计算出每条路径的延迟，并返回给用户。下面详细介绍相关步骤。

3.1 路线计算

路线计算是进行延迟测量的基础。控制器根据用户指定的交换机，对测量进行预处理，以便探测分组可以沿着指定的线路前进。其基本步骤如下。1) 多路径获取；2) 多路径合并；3) 流表项生成与下发。

3.1.1 多路径获取

控制器根据网络的拓扑信息，采用 Yen 改进算法[19,20]计算出起点与终点交换机之间的前 K 条最短路径，该算法在最坏情况下的时间复杂度为 $O(Kn(m + n\log n))$[19]。为避免路径计算与探测延迟上消耗过多的资源，当获得的路径数超过一定阈值（本实验设置为 16 跳）时，将不再进行计算。图 1 展示的是 s1~s7 之间前 3 条最短路径。

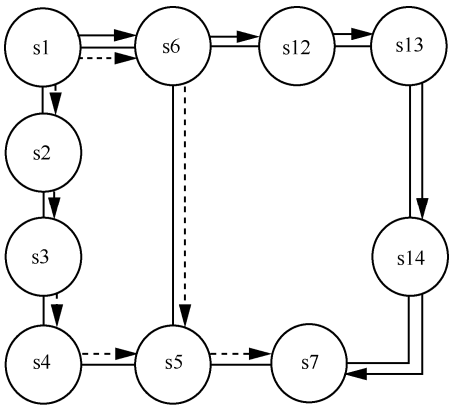


图 1 删除算法结果

3.1.2 多路径合并

由图 1 可知，在计算得出的 3 条路径中，有部分链路被多条路径经过。在传统计算延迟的方法中，控制器要为每一条路径发送不同的探测分组，以获得各路径的往返延迟。这就意味着交换机要为经过它的每一条路径增加 2 个流表项，以达到探测往返延迟的目的（如表 1 所示）。而在 DeMon 中，一条路径的延迟被转换为路径中每个链路延迟的和，即当控制器获得了某条路径中各段链路的延迟，就可以为用户计算出这条路径总的延迟，而不必关心这些链路的延迟是否由同一个探测分组探测所得（如图 2 所示）。

表 1 未合并时 S6 交换机需新增流表项

ID	匹配域	指令集
1	SOURCEIP:0.0.0.1 IN_PORT:1	OUTPUT:2
2	SOURCE_IP:0.0.0.1 IN_PORT:2	OUTPUT:CONTROLLER; OUTPUT:1
3	SOURCE_IP:0.0.0.2 IN_PORT:1	OUTPUT:3
4	SOURCE_IP:0.0.0.2 IN_PORT:3	OUTPUT:CONTROLLER; OUTPUT:1

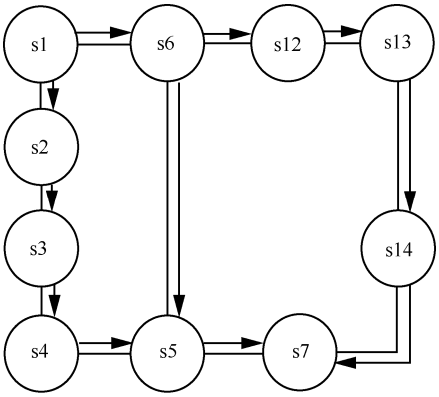


图 2　合并路线结果

DeMon 利用多路径合并的优化算法，确保在进行多路径的延迟测量时，每段路径都只有一个探测分组经过，避免了链路的重复测量，减少了 OpenFlow 交换机中流表的数目和网络中探测分组的数量。

3.1.3　生成流表项

合并完路径后，控制器根据系统合并路径的结果为各交换机生成对应的流表项。

在 Open Flow 交换机中，匹配规则被存储在流表中，每一条规则对应一个流表项。流表项结构如表 2 所示，当数据分组进入交换机后，根据匹配域与优先级选择对应的流表项，进而根据流表项中的指令集完成对数据分组的处理。

表 2　　流表项结构

字段	说明
匹配域	将数据分组与其比较，获得对应的规则
优先级	指定规则的优先级（数字越小优先级越大）
计数器	记录规则被匹配的次数
指令集	记录对数据分组的处理动作
超时	规则的生命周期

在流表项的生成过程中，会遇到 4 种不同情况的交换机。

1) 一进一出型，交换机在计算好的线路中只拥有一个入口和一个出口。

2) 一进多出型，交换机在计算好的线路中拥有一个入口和多个出口。

3) 多进单出型，交换机在计算好的线路中拥有多个入口和一个出口。

4) 多进多出型，交换机在计算好的线路中拥有多个入口和多个出口。

第 4 种情况其实是第 2 种与第 3 种的组合，所以，在本文中只讨论前 3 种情况。

根据图 3 展示的交换机端口链接情况，可以为每种交换机生成对应的流表项。

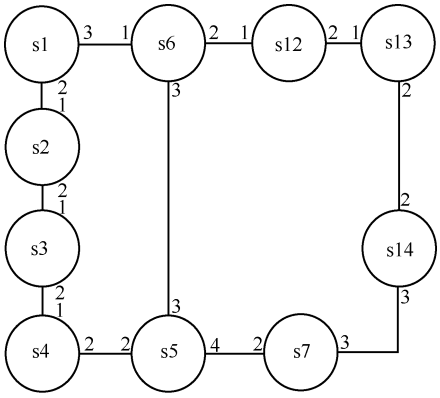


图 3　拓扑

对于第一种情况的交换机而言，只需要为其添加 2 个流表项，如表 3 所示。

表 3　一进一出型交换机（S2）新增流表项

ID	匹配域	指令集
1	IN_PORT:1	OUTPUT:CONTROLLER,OUTPUT:2 OUTPUT:IN_PORT
2	IN_PORT:2	OUTPUT:CONTROLLER

当一进一出型交换机收到来自上游交换机发送的探测分组时，将探测分组进行复制发送给下游交换机、控制器并返回给上游交换机以便计算往返延迟。

当交换机收到来自下游的交换机时，将其发送给控制器，控制器根据此交换机发送回来的 2 个探测分组的时间差计算出链路的延迟。

对于一进多出的情况，需要添加的流表项如表 4 所示。

表 4　一进多出型交换机（S6）新增流表项

ID	匹配域	指令集
1	IN_PORT:1	OUTPUT:CONTROLEER,OUTPUT:2 OUTPUT:3, OUTPUT:IN_PORT
2	IN_PORT:2	OUTPUT:CONTROLLER
3	IN_PORT:3	OUTPUT:CONTROLLER

第 2 种交换机处理的方法与第 1 种类似，只是要为每个下游交换机创建单独的流表项，以便控制器可以计算下游每段链路的延迟。

针对多进单出型交换机添加的流表项如表5所示。

表 5　多进单出型交换机（S5）新增流表项

ID	匹配域	指令集
1	IN_PORT:3	OUTPUT:CONTROLLER,OUROUT:4 OUTPUT:IN_PORT
2	IN_PORT:2	OUTPUT:IN_PORT
3	IN_PORT:4	OUTPUT:CONTROLLER

对于此种交换机，需要事先指定一个主上游交换机（本实验中将最短路径中的上游路由器指定为主上游交换机），只有当接受到该交换机发送的探测分组时，才会将探测分组进行复制发送给下游交换机和控制器。如果收到的是非主上游交换机发送来的探测分组，仅将探测分组返回给上游交换机，不再向下发送，避免链路被重复计算延迟。

DeMon 使用的路径合并优化算法，使交换机增加的流表项个数只与该交换机相邻的交换机个数有关，而与经过该交换机的路径个数无关。以 s6 为例，其流表项从原来的 4 个（如表 1 所示）减少到了目前的 3 个（如表 4 所示），并且优化强度会随着经过该交换机路径个数的增加而不断增加。由此可知，DeMon 对流表产生的负担很小，能有效适用于复杂网络。

3.2　探测延迟

控制器根据预先计算出的路径向起始的 OpenFlow 交换机发送探测分组。探测分组到达起始交换机之后，处理流程如图 4 所示。

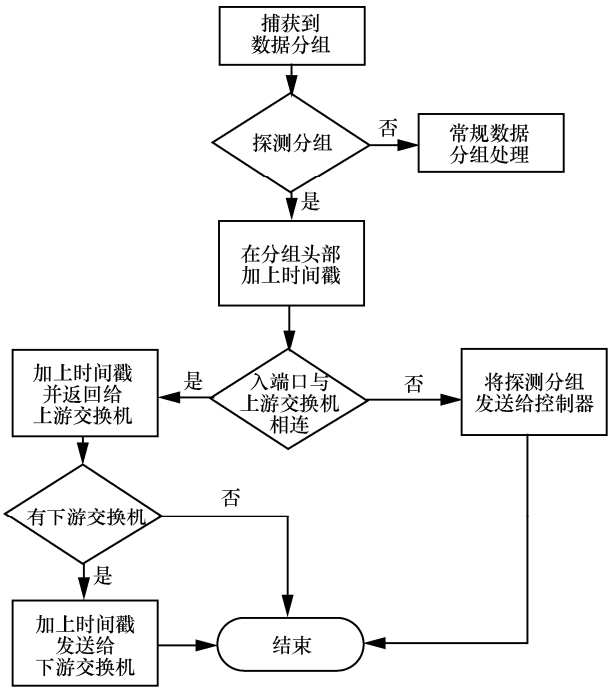


图 4　交换机处理探测分组流程

当数据分组进入交换机后，交换机对其特征进行判断，如果判断结果为探测分组，则为其在头部打上进入交换机的时间戳，并为其找到对应的流表项，根据流表项中的指令对探测分组进行操作。当探测分组要离开交换机时，交换机还会为其在头部打上离开交换机的时间戳。进出时间戳的获取，能有效避免因探测分组在交换机中处理所产生的误差，提高了整体的测量精度。

探测分组在网络中的流向如图 5 所示（控制器

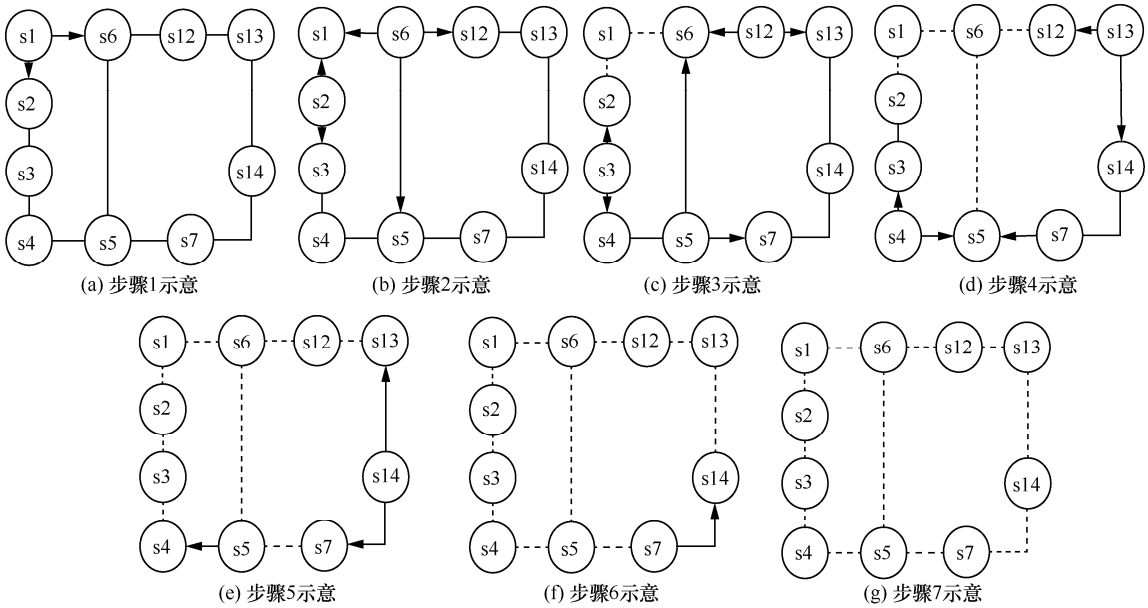


图 5　探测分组流向

未画出）。其中，实线表示未进行延迟测量的链路，箭头表示测量延迟时探测分组的流向，虚线表示已经测量完延迟的链路。

步骤 1，如图 5(a)所示，起始交换机接收到控制器发送的探测分组后，根据事先设置的流表项，将探测分组打上时间戳，发送给下游交换机 s2 和 s6。步骤 2，s2 和 s6 收到 s1 发送的探测分组后，通过查询流表项，找到下游交换机，并将探测分组发送给下游交换机、s1 以及控制器（如图 5(b)所示）。步骤 3，s1 收到下游交换机 s2 和 s6 返回的探测分组后，将其打上时间戳发送给控制器，以此来计算 s1~s2 以及 s1~s6 链路的延迟（如图 5(c)所示）。步骤 4~步骤 7，下游交换机按照上述步骤完成探测分组（如图 5(d)~图 5(g)）的发送。当探测分组到达终点交换机后，因无下游交换机需要其发送探测分组，终点交换机直接将打上时间戳的探测分组发送给控制器和上游交换机。

根据上述过程可知，当探测分组到达某条链路的上游交换机时，交换机会将探测分组复制一份发送给控制器。当该链路的下游交换机将探测分组返回给上游交换机时，上游交换机再将探测分组复制一份，发送给控制器。所以对于每条链路，控制器将收到 2 个探测分组。相关变量定义如表 6 所示。

表 6　　变量说明

符号	说明
T_{SL}	探测分组离开上游交换机的时间
T_{EE}	探测分组进入下游交换机的时间
T_{EL}	探测分组离开下游交换机的时间
T_{SE}	探测分组返回上游交换机的时间

得到链路的延迟式为

$$delay = T_{SE} - T_{SL} - (T_{EL} - T_{EE}) \tag{1}$$

$T_{SE} - T_{SL}$ 计算的是链路的往返延迟加上探测分组在下游交换机中的处理延迟。

$T_{EL} - T_{EE}$ 计算的是探测分组在下游交换机中处理的延迟。

3.3　探测分组丢失处理

在实际探测场景中，可能出现因背景流量过大而导致探测分组丢失，致使控制器只能获得部分链路的延迟数据。针对上述情况，本文方案的处理流程如图 6 所示。

当控制器接收到上一跳交换机返回的探测分组后，将启动一个定时器。如果在 200 ms（Linux 操作系统中将超时重传 RTO 设置为 200 ms）内没有接收到下一跳交换机返回的探测分组，则认为该探测分组丢失，控制器将向丢失探测分组的交换机再次发送探测分组。当对同一交换机的重复发送次数超过阈值（实验中设置为 3 次）后，认为该链路不可达，并标明为不可达链路，方便用户对瓶颈链路进行分析。

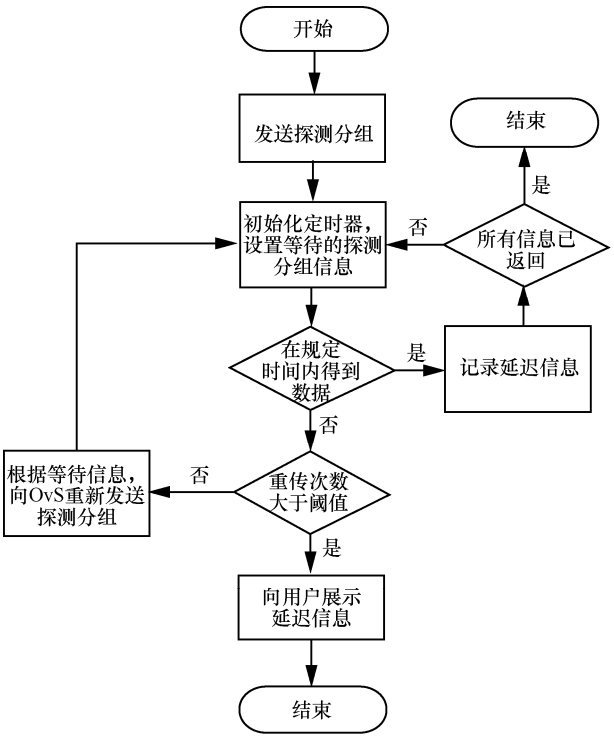


图 6　探测分组丢失处理流程

4　性能仿真与分析

本实验基于 Ubuntu14.04 操作系统，采用 Mininet2.2 完成网络拓扑的仿真。Mininet 是由一些虚拟的终端节点（end-hosts）、交换机、路由器连接而成的一个网络仿真器，使用轻量级的虚拟化技术模拟真实网络环境。

采用的控制器是 ONOS1.8.0[21]。ONOS 是由 ON.Lab 发布，首个面向运行商业务场景的开源 SDN 控制器平台。由业界顶级运营商和设备供应商所主导，具有高可靠性、安全、可扩展等多个优点。为了能够高效准确地处理延迟测量功能，本实验在 ONOS 中增加了探测分组路径计算模块、探测分组下发模块以及延迟计算模块。

采用的交换机是 Open vSwitch（OvS）2.6.1，支持 OpenFlow 协议。为了使获得的进程时间戳更

为准确，在 OvS 的 ovs_vport_receive 函数（数据分组接收函数）以及 ovs_vport_send 函数（数据分组发送函数）内获得进入与离开交换机的时间戳，并将其加入到探测分组的头部。

4.1 测量路径计算与合并

本实验基于多个拓扑验证计算两点间可达路径所需时间，拓扑来自 Topology Zoo。为了能够真实地反映 DeMon 的计算性能，针对每种拓扑进行 100 次实验，每次实验中随机选取 2 个点作为源节点与目的节点进行延迟计算。为了避免计算资源的过度消耗，将两点间可达路径的数量上限设置为 100 条。

表 7 展示在不同类型的拓扑下，DeMon 计算任意两点间可达路径并将路径进行合并所消耗的时间。从结果可以看出，虽然 DeMon 消耗的时间会随着可达路径数的增加而小幅增加，但总体而言 DeMon 都能够快速完成路径计算，平均时间不超过 20 ms。

4.2 链路延迟测量

本实验中链路延迟指的是探测分组经过某 2 个相邻交换机之间链路的往返时间。实验拓扑采用的是 ATT North America，如图 7 所示。该拓扑具有结构比较复杂，任意 2 个交换机之间存在多条可达路径的特点，方便进行多路径延迟测量的研究。实验中使用 Mininet 初始化每一条链路的延迟。为了便于阅读，以下选取 s1~s2 这一链路的数据进行展示，将 DeMon 与 GRAMI、OpenNetMon 方案在此链路上的测量结果进行比较。OpennetMon 将线路计算、探测分组下发以及延迟计算等多种功能集中到控制器当中，通过计算探测分组到达控制器的时间差计算线路的延迟。而 GRAMI 将延迟计算等后续工作放入到额外的设备 MP 中，控制器中只完成线路计算以及探测分组下发的工作，通过探测分组到达 MP 的时间差计算具体链路的延迟。

图 8 展示的是 50 次连续 20 s 测量延迟的平均结果。图 9 展示的是连续测量 100 次并重复 5 组所

表 7 路径计算结果

拓扑名称	交换机数	链路数	多路径平均计算时间/ms	多路径最长计算时间/ms	多路径合并平均时间/ms	平均可达路径数
GetNet	7	8	1	3	0.066	2
AboveNet	21	27	6	17	0.126	28
科技网	18	31	8	24	0.588	74
GARR	36	44	7	19	0.384	42
ATT	25	54	13	31	0.691	100
CESNet	43	54	10	28	0.641	89
ForthNet	62	62	1	2	0.055	1

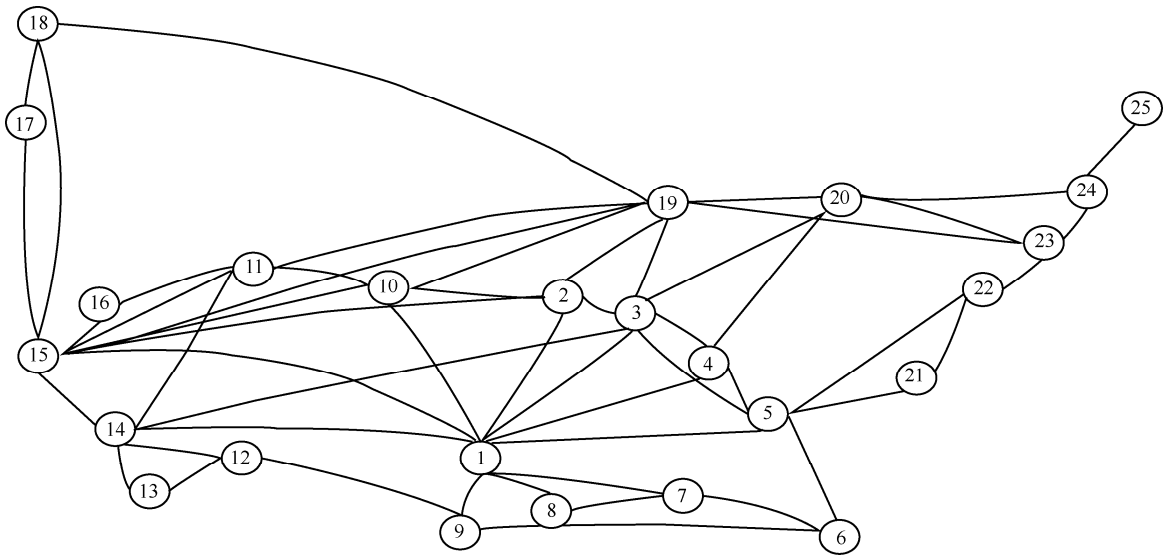


图 7 实验拓扑（ATT）

计算出来的方差。从图 8 可以看出 DeMon 测量出的结果与真实值最为接近，其误差缩小为其他 2 种方法的$\frac{1}{3}$。从图 9 可以看出，DeMon 所测得的延迟非常稳定。与之相反，通过控制器获得时间戳，并以此计算链路延迟的 OpenNetMon 所得到的标准差是 DeMon 标准差的 5 倍。OpenNetMon 波动较大的主要原因是其控制器不断地响应各种交换机的请求，导致数据分组获取时间戳获取的时间点难以保证，从而使测量结果波动较大。这一缺陷会随着控制器要处理事件的增加而愈加严重。由此可以看出，在复杂网络中使用控制器获得时间戳，并以此计算延迟的方法是无法提供稳定且准确的测量结果。而 DeMon 将时间戳获取的任务，转移到了 OvS 交换机中，使获取时间戳的时间不再受控制器性能的影响，较为稳定。

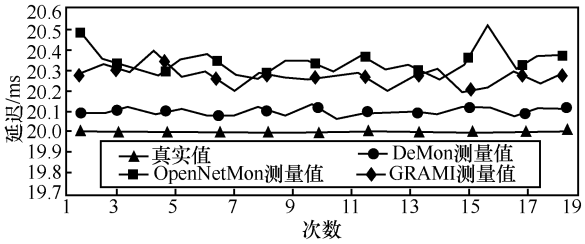


图 8　链路延迟测量结果对比

在图 9 中可以发现 GRAMI 的稳定性与 DeMon 并没有太大的差别。但值得注意的是，GRAMI 是用 MP（GRAMI 中专门测量误差的设备）来计算路径延迟。因为 MP 的个数有限，一个 MP 要完成对多条链路的延迟测量，以致无法保证每个链路的起始交换机都能与 MP 直接连接，所以 GRAMI 的测量结果依赖于链路起始交换机到 MP 之间流量的情况。如果在该交换机与 MP 之间的路径发生拥塞，那么 GRAMI 所测量的结果将会产生巨大误差。

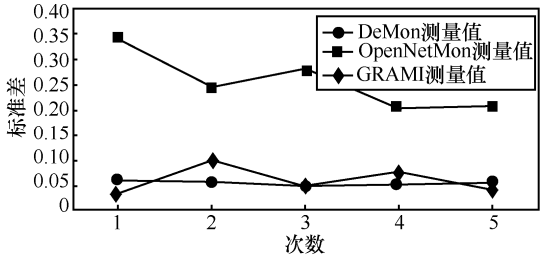


图 9　结果方差对比

为了展示背景流量对 GRAMI 所带来的影响。周期性地随机挑选一条不需要进行延迟测量的链路，并为其注入 20 MB 的流量。实验结果如图 10 所示。当选中的链路为交换机到 MP 所需经过的链路时，GRAMI 的测量误差增加了 20 倍左右，而 DeMon 所测量的结果并没有任何影响。可以看出，GRAMI 的实验是建立在网络中各条链路都有稳定延迟的前提下。但在现实网络中，这个前提并不能很好满足，将会影响其稳定性。相比于 GRAMI，DeMon 并不对其他链路的延迟有任何要求。这使 DeMon 的结果更为准确，适用范围更加广泛。

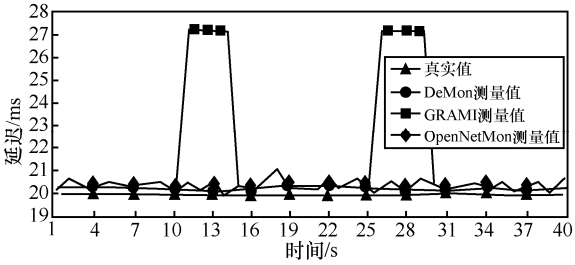


图 10　引入背景流量结果对比

图 11 展示的是在 20 s 的时间内，GRAMI、DeMon 与 OpenNetMon 在路径 s1→s2→s10→s11→s16 测量结果的对比。可以看出测量结果的误差随着经过交换机数量的增加而增加，但 DeMon 依旧是误差最小且最为稳定的。

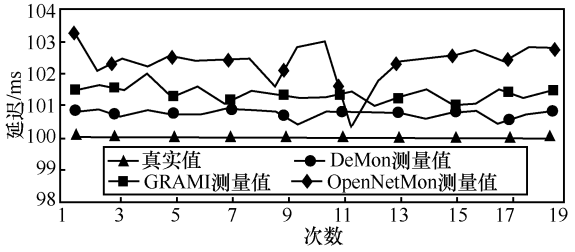


图 11　路径延迟测量结果对比

4.3　链路延迟的快速感知

为了测试 DeMon 对链路延迟变化感知的快慢程度，使用 Mininet 动态选择拓扑中具有不同延迟的链路的延迟。当延迟改变时，真实延迟与 DeMon 测量的延迟间的关系如图 12 所示。

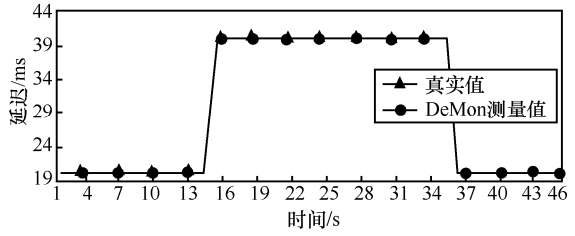


图 12　动态选择链路结果

可以看出，DeMon 对链路延迟的变化非常敏感，可以快速地探测到链路的延迟，并将其反馈给用户。

通过上述实验，可以看出，DeMon 在控制器繁忙、网络流量复杂等情况下能表现出较好的稳定性，并且能快速适应网络延迟的变化，提供可靠的测量服务。

4.4 延迟测量开销

DeMon 在探测阶段的开销主要分为 3 个部分，第一部分是控制器对返回探测分组的延迟计算开销，第二部分是 OvS 中因增加流表项而产生的流表匹配开销，第三部分是因发送探测分组而产生的流量开销。

4.4.1 控制器延迟计算开销

当探测分组返回控制器后，控制器会对探测分组进行分析，计算每段链路的延迟，并将得到的链路延迟信息进行组合，得到各条路径的总延迟，同时返回给用户。

为了获得控制器延迟计算所产生的开销，本文分析了单条链路延迟计算所需的平均时间，以及组合路径延迟所需的时间。表 8 为不同拓扑下控制器计算路径延迟所产生的开销。

表 8 控制器延迟开销

拓扑名称	单链路平均计算开销/ms	路径延迟计算开销/ms
GetNet	0.059	0.133
AboveNet	0.032	0.211
科技网	0.033	0.223
GARR	0.041	0.143
ATT	0.023	0.422
CESNet	0.035	0.372
ForthNet	0.066	0.125

从表 8 可以看出，单条链路的延迟计算开销是很小，不超过 0.1 ms，保证了 DeMon 在多链路测量环境中依旧能保持较高的性能。同时从表 8 可以看出，虽然组合成路径延迟的处理时间会随着路径数量的增多而有所增加，但依然没有超过 0.5 ms。

4.4.2 流表匹配开销

DeMon 在进行延迟测量时，会在交换机中增加流表项以达到控制探测分组行为的目的。通过对表 7 中各种拓扑进行分析并进行多次多路径测量，发现每个交换机中平均增加的流表项个数为 2~4 个，最多不超过 10 个。表 9 展示的是交换机中流表项数量与规则匹配时间之间的关系。

表 9 规则匹配开销

流表项总个数	探测相关流表项个数	OvS 处理时间/ms
4	0	0.211
6	2	0.234
7	3	0.254
8	4	0.272
9	5	0.274
10	6	0.322
11	7	0.327
12	8	0.347
13	9	0.355
14	10	0.376

从表 9 可以看出，DeMon 对交换机带来的开销是很小的。在最坏的情况下（增加了 10 个流表项），OvS 处理数据分组的时间也仅增加了 0.165 ms。

4.4.3 流量开销

DeMon 采用路径合并优化算法，能保证在多路径的情况下，每段链路只有一个探测分组经过（在实验中发送的探测分组大小为 72 B），很好地减少了探测分组对网络中流量的影响。通过实验证明，每次探测中每段链路只有 144 B 大小的探测分组经过。

5 结束语

本文采用了以链路为单位的测量方法，对任意两点间的多条路径进行延迟测量。不仅大大减少了交换机中流表项的个数，提高交换机的匹配速度，而且控制器只需发送一个探测分组就可以完成多条路径的测量，有效减少了网络中探测分组的数量，避免了网络资源的过度消耗，并通过上千次的实验证明，DeMon 可以有效避免因控制器繁忙、网络中部分拥塞等因素所带来的测量误差。为用户在 OpenFlow 网络中提供了一个快速、准确、稳定延迟测量方法。下一步需要开展的工作是将 OpenFlow 网络与真实网络相结合，实现不同类型网络间的延迟测量。

参考文献：

[1] ATARY A, BREMLER-BARR A. Efficient round-trip time monitoring in OpenFlow networks[C]// IEEE INFOCOM 2016-IEEE Conference on Computer Communications. 2016:1-9.

[2] ZENG H, KAZEMIAN P, VARGHESE G, et al. A survey on network troubleshooting[C]//Technical Report Stanford/TR12-HPNG-061012. 2012.

[3] GUO C, YUAN L, XIANG D, et al. Pingmesh: a large-scale system

for data center network latency measurement and analysis[C]//The 2015 ACM Conference on Special Interest Group on Data Communication. 2015: 139-152.

[4] PELSSER C, CITTADINI L, VISSICCHIO S, et al. From paris to tokyo: On the suitability of ping to measure latency[C]//The 2013 conference on Internet measurement conference. 2013. 427-432.

[5] CROVELLA M. KRISHNAMURTHY B. Internet measurement: infrastructure, traffic and applications[M]. John Wiley & Sons, Inc., 2006.

[6] HU N, LI L, MAO Z, et al. Locating Internet bottlenecks: Algorithms, measurements, and implications[C]//ACM SIGCOMM. 2004.

[7] MAHAJAN R, SPRING N, WETHERALL D, et al. User-level Internet path diagnosis[C]//ACM SOSP. 2003.

[8] TACHIBANA A, ANO S, TSURU M. A large-scale network diagnosis system based on user-cooperative active measurements[J]. Int. J. Space-Based and Situated Computing, 2013, 3(2): 69-82.

[9] MCKEOWN N. Software-Defined networking[C]//INFOCOM. 2009.

[10] DAVE T. Openflow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[11] PHEMIUS K, BOUET M. Monitoring latency with OpenFlow[C]// International Conference on Network and Service Management. 2013:122-125.

[12] JOHANNESSEN S. Time synchronization in a local area network[J]. IEEE Control Syst. Mag., 2004, 24(2): 61-69.

[13] COOKLEV T, EIDSON J C, PAKDAMAN A. An implementation of IEEE 1588 over IEEE 802.11b for synchronization of wireless local area network nodes[J]IEEE Trans. Instrum. Meas., 2007, 56(5): 1632-1639.

[14] 王俊峰, 杨建华, 周虹霞,等. 单向延迟测量中时钟动态性检测算法[J]. 软件学报, 2004, 15(4):584-593.
WANG J F, YANG J H, ZHOU H X, et al. Detecting clock dynamics in one-way delay measurement[J]. Journal of Software, 2004, 15(4): 584-593.

[15] CORBETT J C, DEAN J, EPSTEIN M, et al. Spanner: Google's globally distributed database[J]. ACM Transactions on Computer Systems, 2013, 31(3):8.

[16] WANG A, GUO Y, HAO F, et al. Scotch: Elastically scaling up sdn control-plane using vswitch based overlay[C]//The 10th ACM International on Conference on emerging Networking Experiments and Technologies. 2014, 403-414.

[17] ADRICHEM N L M V, DOERR C, KUIPERS F A. OpenNetMon: network monitoring in OpenFlow software-defined networks[C]// Network Operations and Management Symposium. 2014:1-8.

[18] SHIBUYA M, TACHIBANA A, HASEGAWA T. Efficient Performance diagnosis in OpenFlow networks based on active measurements[J]. 2014: 268-273.

[19] MARTINS E Q V, PASCOAL M M B. A new implementation of Yen's ranking loopless paths algorithm[J]. 4OR, 2003, 1(2):121-133.

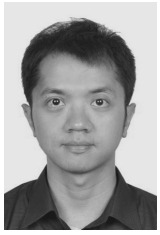
[20] YEN J Y. Finding the *K* shortest loopless paths IN a network[J]. Management Science, 1971, 17(11):712-716.

[21] BERDE P, GEROLA M, HART J, et al. ONOS: towards an open, distributed SDN OS[C]//The Workshop on Hot Topics in Software Defined NETWORKING. 2014:1-6.

作者简介：

邱欣逸（1992-），女，福建龙岩人，中国科学院计算机网络信息中心硕士生，主要研究方向为未来互联网。

李俊（1968-），男，安徽桐城人，中国科学院计算机网络信息中心研究员、副总工、博士生导师，主要研究方向为未来互联网、网络安全等。



周建二（1986-），男，广西蒙山人。中国科学院计算机网络信息中心助理研究员。主要研究方向为云计算、传输优化、网络测量和未来互联网体系结构。

李菁菁（1981-），男，湖北武汉人。中国科学院计算机网络信息中心高级工程师，主要研究方向为网络管理技术、云网融合、软件定义广域网和智能网络。