文章编号: 1007-5321(2015)05-0042-05 DOI: 10.13190/j.jbupt.2015.05.007

基于 OpenFlow 的链路故障诊断方法

程 光[1,2], 王玉祥[1,2], 胡一非[1,2], 郭晓军[1,2]
(1. 东南大学 计算机科学与工程学院, 南京 211189; 2. 东南大学 计算机网络和信息集成教育部重点实验室, 南京 211189)

摘要: 为了实现对软件定义网络(SDN)环境下链路故障的快速诊断及定位,提出了一种基于 OpenFlow 的链路故障诊断机制. 利用 SDN 控制平面与转发平面分离的特性,使用 SDN 南向接口协议——OpenFlow 对网络各个节点进行信息采集,从拓扑管理、链路丢包、链路带宽、链路拥塞 4 个方面进行链路故障的诊断. 实验结果表明,该方法能够准确获取网络链路的性能参数. 此外,结合网络拓扑信息,能够对网络故障点进行精确的定位.

关 键 词: 故障诊断; 故障定位; 拓扑管理; OpenFlow
中图分类号: TN929.53 **文献标志码:** A

Network Link Fault Diagnosis Based on OpenFlow

CHENG Guang[1,2], WANG Yu-xiang[1,2], HU Yi-fei[1,2], GUO Xiao-jun[1,2]
(1. School of Computer Science and Engineering, Southeast University, Nanjing 211189, China; 2. Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing 211189, China)

Abstract: To rapidly diagnose and locate link failure in software-defined network (SDN), an OpenFlow based link failure diagnosis mechanism was proposed. Making full use of SDN's feature that the control plane decouple from the data plane, the mechanism collects information from every node in the network using the OpenFlow protocol, and then analyzes the information for topology management, link packet loss, link bandwidth and link congestion to diagnose link failure. Simulation proves that the mechanism can accurately measure link performance metrics; besides, with topology information, link failure can be precisely located.

Key words: fault diagnosis; fault location; topology management; OpenFlow

链路故障诊断对许多网络应用和协议有着重要的作用,端到端的接入控制、服务的动态选择保证等都需要链路故障诊断的支持. 然而,随着计算机网络规模越来越庞大,结构越来越复杂,网络中的故障诊断变得愈发困难.

传统的网络设备将控制平面和数据平面紧密耦合,每个网络设备都有自己的控制平面,使得网络成为一个分布式的系统,而且,网络内部也不允许随意放置监测点,这就使得被动测量方式所能获取的网络信息受到了限制. 因此,在传统网络中,网络性能故障诊断所需的信息多采用主动测量的方式获取. 然而,主动测量增加了网络的潜在负载,尤其是未经仔细设计的测量会对网络造成较大的影响;并且,主动测量获取的信息多为网络整条路径上的性能信息,并不完全反映某段链路的性能信息.

软件定义网络(SDN, software-defined network)作为一种新型的网络架构[1],将网络设备的控制层面与转发层面相分离,使得原本的网络黑盒变得透明可编程. 笔者提出一种 SDN 环境下基于 OpenFlow 的链路故障诊断方法. 该方法利用 SDN 集中

收稿日期: 2014-12-10
基金项目: 江苏省未来网络创新研究院未来网络前瞻性研究项目(BY2013095-5-03); 国家高技术研究发展计划(863 计划)项目(2015AA015603); 江苏省"六大人才高峰"高层次人才项目(2011-DZ024)
作者简介: 程 光(1973—),男,教授,博士生导师; 王玉祥(1990—),男,硕士生,E-mail: yxwang@njnet.edu.cn.

管制和自我监控的特点,将主动测量和被动测量相结合,通过主动测量获取网络的拓扑信息,通过被动测量收集网络设备上的统计信息、监测网络拓扑的变化,并根据收集的信息进行网络链路物理故障及性能故障的诊断和定位.

1　相关研究

链路故障诊断的基础在于网络信息参数的获取,关键在于故障的诊断和定位. 近年来,关于如何利用 SDN 的特点进行网络测量的研究很多.

Tootoonchian 等[2] 提出了一种基于 OpenFlow 的网络流量矩阵估算系统,该系统通过获取交换机上的统计信息来推算 SDN 的流量矩阵,这样既可以精确地获得流量信息,又不会引入过大的额外负载. 但该系统在选路上对控制器有特定要求,而且获取的只有流量信息.

Ofpeck[3] 是一种测量 OpenFlow 网络状态和性能的系统,该系统运行在与 OpenFlow 交换机相连的用户主机上,使用主动测量的方法,测量与主机相连交换机的流表项建立时间,还可以从用户的角度测量指定 IP 的往返延迟和丢包率,以及 Web 服务器的 wget 延迟. 但是,该方法所测参数的准确性较差.

Rasley 等[4] 开发了一种新型的测量平台 planck,该平台使用 SDN 集中控制的特点,使用过载端口镜像和高速包处理方法提供毫秒级的网络监控,相较于传统网络监测方法,该方法将监测效率提高了 11 ~ 18 倍. 但是,由于过载使用端口,使得监测端口过多地占用交换机缓存,造成交换机上的丢包率增加、延迟减小.

Jarschel 等[5] 研究了 SDN 测量中的准确性和潜在的资源过载问题,从带宽测量、延迟测量两个方面,将基于 SDN 的网络测量和基于数据包追踪的网络测量进行比较.

从上述已有文献来看,SDN 测量方面,主要集中在网络性能参数获取方面,但关于 SDN 中故障诊断的研究则相对较少.

2　系统设计

控制器是 SDN 的核心,网络中的拓扑管理、路由选择、负载平衡等均由控制器负责实现. 因此,故障诊断方法主要在控制器上实现,该方法包括 3 个模块: 拓扑管理模块、网络测量模块和故障诊断模块.

2.1　拓扑管理模块

拓扑管理模块主要采用主动测量的方式进行网络全局拓扑的感知. 在 OpenFlow 协议中[7],控制器在和 OpenFlow 交换机进行握手时,交换机向控制器发送自己的状态信息,包括数据平面 ID、端口状态等,但这些信息并不包括网络拓扑信息. 为了获取 OpenFlow 的网络拓扑信息,通过链路层发现协议(LLDP, link layer discovery protocol) 进行拓扑发现. 如图 1 所示,控制器将 sw1 的数据平面 ID(datapath_id) 以及端口号(port2) 封装到 LLDP 数据包中,然后通过 packet_out 消息指示 sw1 将该 LLDP 数据包通过端口 port2 发送出去,当该数据包到达 sw2 的 port1 端口后,会触发 sw2 发送 packet_in 消息给控制器,该 packet_in 消息中包含了由 sw1 的 port2 发送过来的 LLDP 数据包,控制器从收到的 packet_in 消息中可以解析得到该 packet_in 消息是由 sw2 的 port1 触发的,同时从 packet_in 的 LLDP 数据包中可以解析出该数据包是从 sw1 的 port2 发送过来的,此时控制器就可确定 sw1 的 port2 端口和 sw2 的 port1 端口是直连的. 同理,控制器也可以指示 sw2 从其 port1 端口发送 LLDP 数据包,这样控制器就可确定 sw2 的 port1 端口和 sw1 的 port2 端口是直连的. 由此,控制器完全获取了一条链路((sw1,port2) ,(sw2,port1)) 的连接信息. 若推广到整个网络,控制器可以指示从所有交换机的所有端口发送 LLDP 数据包,从而获取全网的拓扑信息.

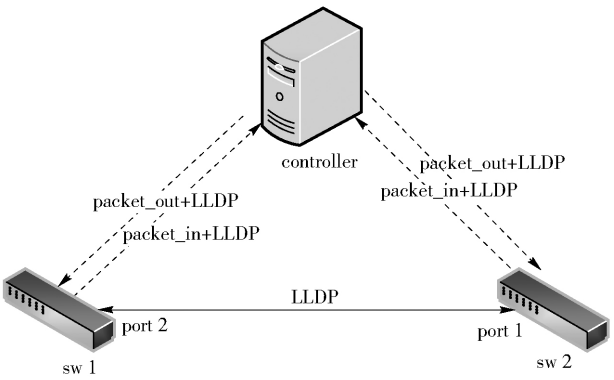

图 1　拓扑发现

2.2　网络测量模块

网络测量模块主要测量带宽、丢包率和拥塞度 3 个网络性能参数,其中带宽可以反映链路的流量情况,丢包率和拥塞度可以反映链路的拥塞情况.

2.2.1　带宽测量

带宽的测量采用被动的方式进行,OpenFlow 交换机上针对每条流、每个端口、每个队列维护计数器.

带宽的测量可以利用交换机上维护的端口信息进行，由控制器向链路两端的交换机发送 ofp_port_status_request[6] 消息，询问构成链路的端口上的统计信息，交换机收到查询请求后返回 ofp_port_status[6] 消息，如图2所示. ofp_port_status 消息包含了对应端口的全部统计信息，而带宽的计算需要使用 rx_bytes 和 tx_bytes 统计信息. 设 t_1 时刻收集到的(sw1,port2)的 rx_bytes 为 b_1，t_2 时刻收集到的(sw1,port2)的 rx_bytes 为 b_2，则(sw1,port2)端口发送流量为 $B_{tra}=(b_2-b_1)/(t_2-t_1)$；同理，可以得到(sw2,port1)的接收流量为 B_{rec}. 则链路((sw1,port2),(sw2,port1))在(sw1,sw2)方向的流量为 $\min\{B_{tra},B_{rec}\}$.

```
struct ofp_port_status {
    uint16_t    port_no;
    uint8_t     pad[6];
    uint64_t    rx_packets; /* 接收数据包个数* /
    uint64_t    tx_packets; /* 发送数据包个数* /
    uint64_t    rx_bytes;   /* 接收数据包字节数* /
    uint64_t    tx_bytes;   /* 发送数据包字节数* /
    uint64_t    rx_dropped;
    uint64_t    tx_dropped;
    uint64_t    rx_errors;
    uint64_t    tx_errors;
    uint64_t    rx_frame_err;
    uint64_t    rx_over_err;
    uint64_t    rx_crc_err;
    uint64_t    collisions;
};
```

图2 ofp_port_status 结构体

2.2.2 丢包率测量

链路丢包率测量与链路带宽测量类似，也是采取被动测量的方式，通过控制器发送端口状态查询消息，获取交换机返回的端口状态消息，从中读取指定端口的发送数据包数目以及接收数据包数目，然后利用读取到的参数进行链路丢包率的计算. 如图1所示，分别读取 sw1 交换机 port2 端口的 tx_packets 和 sw2 交换机 port1 端口的 rx_packets，则链路((sw1,port2),(sw2,port1))在(sw1,sw2)方向的丢包率为(tx_packets - rx_packets)/tx_packets.

2.2.3 拥塞度测量

链路拥塞度的测量主要利用链路端口的发送数据包统计参数和接收数据包统计参数计算. 如图1所示，链路((sw1,port2),(sw2,port1))的拥塞度 C 可利用带宽测量模块计算的端口流量参数计算. 设端口(sw1,port2)的发送流量为 B_{tra}，端口(sw2,port1)的接收流量为 B_{rec}，则链路的拥塞度为 $C=B_{tra}/B_{rec}$. 在链路没有出现拥塞的情况下，C 的值为1；链路出现拥塞，C 的值会大于1，并且会随着链路拥塞程度的增加而变大.

2.3 故障诊断模块

故障诊断模块是结合拓扑管理模块与网络测量模块的信息来进行故障的诊断和定位的，所诊断的故障类型包括交换机端口物理故障以及网络链路性能故障. 其中，交换机端口的故障诊断主要借助于 ofp_port_status 消息，OpenFlow 协议中规定当交换机的端口状态发生改变时，交换机会向控制器发送 ofp_port_status 消息，该消息中包含 reason 字段，如图3所示，通过解析其中的 reason 字段可以获得链路变化的具体原因. 同时，当收到 ofp_port_status 消息时，可以再次进行网络拓扑的发现，以此来更新网络拓扑信息.

```
enum ofp_port_reason {
    OFPPR_ADD = 0,
    OFPPR_DELETE = 1,.
    OFPPR_MODIFY = 2,
};
```

图3 ofp_port_reason 字段结构

对于链路的拥塞故障，新建一个 Floodlight[7] 控制器模块，从拓扑管理模块读取实时拓扑信息，以5 s一次的频率向现有拓扑中的每个端口发送 ofp_port_status_request 消息，获得每个端口上发送、接收、丢包等各项指标信息并存储备用. 每当获得新的数据，与上一条数据进行比较运算，获得最新5 s内链路状况. 在此过程中并不会遍历网络中所有的端口，这得益于在拓扑管理中已经获得了整体网络拓扑结构，从而在进行信息采集时只需要访问相关链路上的端口，大大节省了程序开销. 同时，将测量得到的链路带宽、链路丢包率以及链路拥塞度相关联，当链路丢包率大于0，链路拥塞度大于1时，说明链路出现了拥塞，此时的链路带宽值为链路的瓶颈带宽.

3 实验结果与分析

为了验证方法的正确性，使用 Mininet 进行仿真实验，构建如图4所示的网络拓扑，控制器选择 Big Switch 公司的 Floodlight 控制器. Floodlight 的核心架构是功能模块化，在 Floodlight 基础上进行开发设计所需的功能模块.

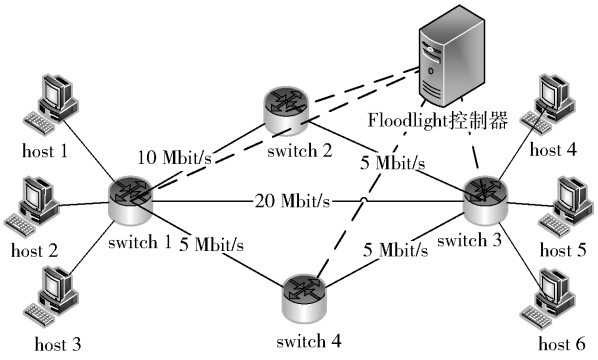

图 4　实验网络拓扑图

3.1　链路物理故障诊断

链路物理故障主要使用拓扑管理模块获取的信息进行诊断. 在 Mininet 下建立如图 4 所示的拓扑结构,使用 mn 命令的 custom 和 topo 参数来指定自定义的网络拓扑. 同时,使用 Mininet 的(link node1 node2 down) 命令断开 switch1 和 switch2 之间的连接、switch4 和 switch3 之间的连接,以此来检测链路物理故障诊断功能.

表 1 为测试开始时的初始拓扑,0 表示交换机之间没有链路,1 表示交换机之间存在链路,测试结果与图 4 搭建的拓扑结构完全相同. 表 2 为断开拓扑中的 switch1-switch2 链路和 switch3-switch4 链路之后的结果,拓扑矩阵的(sw1,sw2) 、(sw2,sw1) (sw3,sw4) 、(sw4,sw3) 位置已经变成了 0. 实验表明,故障诊断模块能够准确地定位网络中的链路物理故障.

表 1　初始拓扑　　Mbit/s

交换机	sw1	sw2	sw3	sw4
sw1	0	1	1	1
sw2	1	0	1	0
sw3	1	1	0	1
sw4	1	0	1	0

表 2　断开后拓扑　　Mbit/s

交换机	sw1	sw2	sw3	sw4
sw1	0	0	1	1
sw2	0	0	1	0
sw3	1	1	0	0
sw4	1	0	0	0

3.2　链路带宽测量

链路的带宽主要使用链路流量接收端口的统计参数计算,从静态带宽测试及动态带宽测试两个方面来评估该方法的准确性,并将测试结果与 iperf 工具的测试结果相对比. 其中静态带宽测试使用 Mininet 中的 mininet. link 配置图 4 所示的链路带宽. 同时,在控制器端向 4 台交换机下发流表,使得 host1 的流量按照 host1→witch1→switch2→switch3→host4 的路径进行转发,host2 的流量按照 host2→switch1→switch3→host5 的路径进行转发,host3 的流量按照 host3→switch1→switch4→switch3→host6 的路径进行转发. 然后,使用 iperf 生成测试流量,在 host1、host2、host3 中运行 iperf 客户端,host4、host5、host6 上运行 iperf 服务器端,测试使用用户数据报协议(UDP, user datagram protocol) 流量,host1 产生 10 Mbit/s 测试流量,host2 产生 15 Mbit/s 测试流量,host3 产生 5 Mbit/s 测试流量.

静态带宽测试的结果如表 3 所示. 因为 switch2-switch3 之间的带宽限制,虽然在 switch1→switch2→switch3 注入 10 Mbit/s 的流量,但是测量得到的 switch2 - switch3 之间只有 5 Mbit/s,而且,其他链路的带宽测试结果均与实际情况相符.

表 3　带宽测试结果　　Mbit/s

交换机	sw1	sw2	sw3	sw4
sw1	0	10. 05	15. 56	5. 04
sw2	0	0	5. 04	0
sw3	0	0	0	0
sw4	0	0	5. 04	0

动态带宽测试使用 host2→switch1→switch3→host5 链路,测试流量为 UDP 流量,测试工具使用 iperf,初始时向链路注入 2 Mbit/s 的流量,然后以每次 3Mbit/s 的增幅增长到 8 Mbit/s,每个流量值持续 60 s,链路带宽的测试结果如图 5 所示.

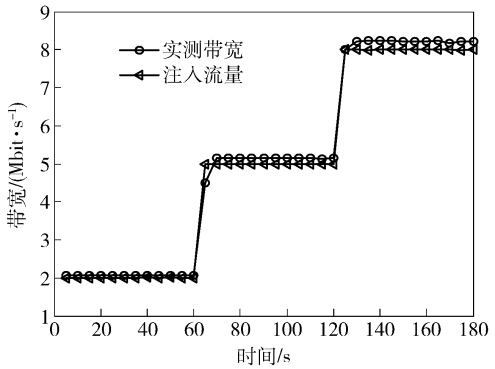

图 5　链路带宽实时监测

从监测结果可以看出,0 ~ 60 s 链路带宽的监测值为 2 Mbit/s,60 ~ 120 s 链路带宽的监测值为 5 Mbit/s,120 ~ 180 s 链路带宽的监测值为 8 Mbit/s. 这些链路带

宽的监测值与 iperf 工具的注入流量基本一致，说明所提方法能够及时准确地监测链路带宽的变化.

3.3 链路丢包率测量

链路的丢包率可根据链路两个端口的发送数据包字节数以及接收数据包字节数来计算. 为了验证该计算方法的准确性，选取 host2→switch1→switch3→host5 作为测试链路，将链路带宽设置为 8 Mbit/s，测试流量使用 UDP 流量，初始流量大小为 5 Mbit/s，然后以 3 Mbit/s 的增幅增长到 14 Mbit/s，每个流量值持续 60 s. 实验过程中，使用 2.2.2 节所述方法计算链路丢包率，并将计算结果与 iperf 工具实测的丢包率相对比，具体的实验结果如图 6 所示.

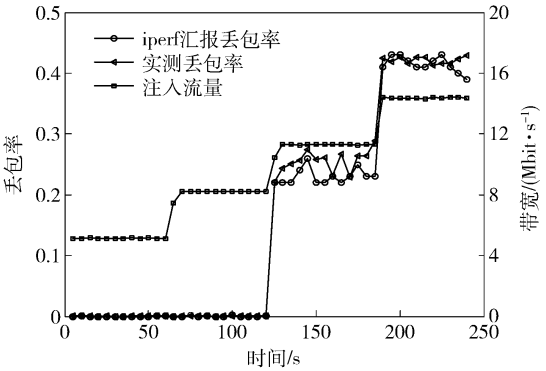

图6 链路丢包测试

从测试结果可以看出，在 0～60 s 和 60～120 s 链路的注入流量分别 5 Mbit/s 和 8 Mbit/s，均没有超过链路的设定带宽，因此链路上也没有出现丢包. 而 120～180 s 和 180～240 s 时，链路的注入流量分别是 11 Mbit/s 和 14 Mbit/s，该带宽值均超过了链路的设置带宽. 此时链路出现丢包，通过 2.2.2 节所述方法计算的丢包率与 iperf 测量的丢包率基本吻合.

3.4 链路拥塞检测

链路拥塞检测是通过将链路流量发送端的带宽、接收端的带宽以及链路丢包率相结合，从而检测链路的拥塞情况和链路的瓶颈带宽. 链路拥塞的检测选取 host2→switch1→switch3→host5 作为测试链路，测试流量为 UDP 流量，链路带宽设为8 Mbit/s，测试流量的初始值为 5 Mbit/s，然后以3 Mbit/s 的增幅增加到 14 Mbit/s，每个流量值持续 60 s，具体实验结果如图 7 所示.

从测试结果可以看出，在 0～120 s，注入的流量没有超过链路的设定带宽，链路没有出现丢包，实测带宽与注入的流量值基本相同，在这种情况下，链路的拥塞度为 1，即链路并没有出现拥塞. 在 120～240 s，注入的流量超过了链路的设定带宽，从链路的丢包率可以看出，链路开始出现丢包；从链路拥塞度可以看出，此时拥塞度大于 1，链路开始出现拥塞. 同时，在 120～240 s，实测的链路带宽一直保持在 8 Mbit/s，并没有随着注入流量的增大而变大，说明该条链路的瓶颈带宽值为 8 Mbit/s，这与实验设置的值一致，说明通过将丢包率、拥塞度和链路带宽 3 个测度相结合，能够准确地进行链路拥塞的检测，而且能够测得链路的瓶颈带宽值.

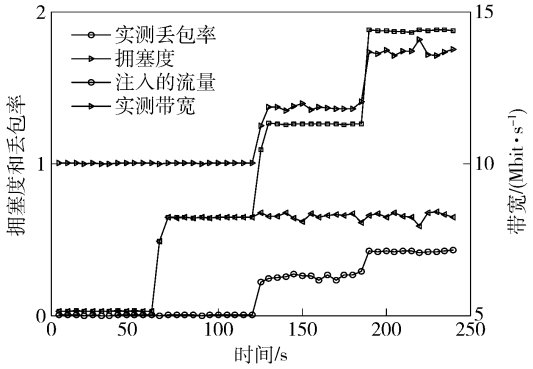

图7 拥塞链路检测

4 结束语

随着网络规模愈加庞大及其结构越来越为复杂，迫切需要有效的网络故障链路诊断方法. 笔者提出的基于 OpenFlow 的网络故障诊断方法能够准确、快速地测量网络链路上的性能参数，根据测得的链路参数信息，可以准确定位网络的故障链路. 但实验是在小范围网络上进行的，没有考虑大规模网络中的效率及准确性问题，下一步需要在大规模网络中测试本方法，以验证方法的准确性及高效性.

参考文献:

[1] 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.

[2] Tootoonchian A, Ghobadi M, Ganjali Y. OpenTM: traffic matrix estimator for OpenFlow networks [C] //Passive and Active Measurement, Zurich: Springer Berlin Heidelberg, 2010: 201-210.

[3] McKeown. Ofpeck [EB/OL]. Canlifornia: WordPress, 2011 [2014-10-20] http://www.openflow.org/wk/index.php/Ofpeck.

(下转第53页)

[9] Kaushik P，Jain A. DroidCheck：android malware detection by behavioral techniques and honeypot [J]. International Journal of Computer Science and Mobile Computing (IJCSMC)，2015，4(4)：829-834.

[10] 诸葛建伟，韩心慧，周勇林，等. HoneyBow：一个基于高交互式蜜罐技术的恶意代码自动捕获器 [J]. 通信学报，2007，28(12)：8-13.

[11] Levchenko K，Pitsillidis A，Chachra N，et al. Click trajectories：End-to-end analysis of the spam value chain [C] //2011 IEEE Symposium on Security and Privacy (SP). 2011 IEEE. Berkeley：IEEE Press，2011：431-446.

[12] Mohammed M，Chan H A，Ventura N，et al. An automated signature generation approach for polymorphic worms using principal component analysis [J]. Int'l Journal for Information Security Research (IJISR)，2011，1(1)：45-52.

[13] Dynamips project [EB/OL]. 2007. http：// www. ipflow. utc. fr/index. php/Cisco_7200_Simulator.

[14] Newsome J，Song D. Dynamic taint analysis for automatic detection，analysis，and signature generation of exploits on commodity software [EB/OL]. 2005. http：// web. eecs. umich edu/ ~ mahlke/courses/583f12/reading/newsome-ndss05. pdf.

[15] Guilfanov I. IDA fast library identification and recognition technology (FLIRT technology)：in-depth [EB/OL]. (2012-02-27) [2012-03-11]. http：// www. hex-rays，com/products/ida/tech/flirt/in _ depth. shtml，2012.

[16] Erdélyi G. IDAPython：User scripting for a complex application [D]. EVTEK：University of Applied Sciences，2008.

(上接第 46 页)

[4] Rasley J，Stephens B，Dixon C，et al. Planck：millisecond-scale monitoring and control for commodity networks [C] //Proceedings of the 2014 ACM conference on SIGCOMM. New York：ACM Press，2014：407-418.

[5] Jarschel M，Zinner T，Hohn T，et al. On the accuracy of leveraging SDN for passive network measurements [C] //. In Proceedings of 2013 Telecommunication Networks and Applications Conference (ATNAC'13). Christchurch：IEEE Press，2013：. 41-46.

[6] McKeown OpenFlow Wiki [EB/OL]. Canlifornia：Word Press，2011 [2014-10-20] http：//archive. openflow. org/wk/index. php/OpenFlow_Wiki.

[7] Rich Lane. Floodlight [EB/OL]. California：Project Floodlight，2013 [2014-10-20] http：//www. projectfloodlight. org/floodlight/.