SDN 网络中端到端 QoS 控制机制的研究

叶云东[1]，曹争[1]

（1.东南大学计算机科学与工程学院，南京，211189）

摘　要：软件定义网络(Software Defined Network)是一种新型网络架构，目前已经成为网络研究领域内的一大热点。QoS 技术通过一定的控制机制保障对于传输质量有一定要求的数据流的服务质量。本文的研究目标为分析传统网络中 QoS 保障机制，并结合 SDN 网络（基于 OpenFlow）控制与数据平面分离，集中控制以及可编程的特点，设计并实现一套 SDN 网络中的 QoS 控制机制。

关键词：软件定义网络；QoS；OpenFlow；

Research on End-to-End QoS Control Mechanism In the SDN Networking

Ye Yundong[1], Cao Zheng[1]

(1. School of Computer Science and Engineering, Southeast University, Nanjing, 211189)

Abstract: Software Defined Networking (SDN) is an innovated networking architecture, and nowadays it has become a hot topic of the networking research field. The technology of Quality of Service (QoS) guarantees the service quality of the flow which have certain requirements with the help of some control mechanism. This aim of this article is to design and implement a QoS control mechanism in the SDN Networking combining the traditional QoS control mechanism and the characteristics of SDN (based on Openflow).

Key words: Software Defined Networking; QoS; Openflow;

软件定义网络（Software Defined Networking, SDN）是一种新型的网络架构，起源于斯坦福大学 Nick Mckeown 教授团队的 Clean Slate 项目[1]。Clean Slate 项目的目标是“重塑互联网”，使未来的网络能够克服如今传统网络架构的限制，不断的吸收新的技术，提供新的应用与服务。2005 年，Albert Greenberg 等人的 4D 项目[2]提出了将控制平面从数据平面分离的新架构。2006 年，Martin Casado 等人以 4D 为基础，提出了一个逻辑上集中控制的用于企业网的管理架构 SANE[3]。Martin Casado 等人在 2007 年又对 SANE 进行功能扩充，提出 Ethane[4]，能够对流进行更细粒度的控制与管理。Nick Mckeown 等人于 2008 年，提出了 Openflow 的概念，并将该技术用于校园网络[5],并于 2009 年，在基于 Openflow 的基础上提出了软件定义网络（SDN）的概念。2011 年，在 Nick Mckeown 教授等人的推动下，一些服务提供商成立了开放网络基金会（Open Networking Foundation， ONF）这样一个非营利性的组织，来商业化，标准化和促进生产网络中使用 Openflow[6]，推动 SDN 的发展。

ONF 定义的 SDN 架构基于 Openflow 将网络设备的控制平面与数据平面进行分离，采用集中式控制的方法，具有可编程的特点，能够对网络流量进行灵活的控制。

服务质量（Quality of Service, QoS）是一项为指定的网络通信提供更好的服务能力的基础技术。该项技术对于关键应用以及多媒体应用十分的重要。随着互联网技术的快速发展，各种新型互联网应用与服务（IP 电话，视频会议等）不断地涌现，这些业务往往对于网络带宽有着一定的要求，并对时延，丢包等网络性能参数比较敏感，需要网络为传输过程提供 QoS 保障。

本文通过 OpenFlow 技术结合传统网络中的 QoS 保障机制，对 SDN 网络中的 QoS 问题进行研究，设计并实现一个 SDN 网络中的端到端 QoS 控

作者简介：叶云东，（1991-），男，硕士研究生，E-mail: ydye@njnet.edu.cn；曹争，（1958-），男，副教授，E-mail: zcao@njnet.edu.cn

制系统。

1 技术背景

1.1 SDN 与 OpenFLow 概述

SDN 架构具有三大特点：可编程、控制平面与数据平面的分离以及集中式控制。与传统的网络架构相比，SDN 使得网络管理人员能够通过编程的手段以一种更灵活的方式来对通过网络的流进行管理与控制。集中式的控制与全局性的视角，使的传统网络中的一些技术得到了改变的机会。

OpenFlow 是当前 SDN 架构中最主流的南向接口之一，用来描述控制器与交换机之间交互所用的信息标准。运行 OpenFlow 协议的交换机，则被称为 OpenFlow 交换机。OpenFlow 交换机是基于 OpenFlow 的 SDN 网络核心部件，主要承担数据的转发。在 OpenFlow 交换机中有流表，流表中的各个表项用于匹配不同特征的流，并根据流表项中定义的不同动作，对相应的流采取不同的动作。

1.2 QoS 概述

到目前为止，国际互联网工程组（IETF）已经提出了多种 QoS 服务模型。综合服务[7]（IntServ）通过使用资源预留协议[8]（RSVP）来提供 QoS 保障，在该模型中，端到端的网络资源被保留，因此能够提供严格的 QoS 保障，但是这种模型实现复杂，而且网络中所有节点必须支持 RSVP 协议，并维护流的状态和交换信令信息，因此代价极大。区分服务[9](DiffServ)则在网络边界对流进行分类和标记，网络设备则根据流的不同类型，执行相应的逐跳行为（Per Hop Behavior， PHB）。区分服务模型实现简单，扩展性较好，需要维护的状态信息较少。但是，由于该模型对网络中的流按照类别来对待，粒度较粗，不能很好的提供某一条流的 QoS 保障。

1.3 Ryu

SDN 控制器起到连接底层网络设备与上层应用的作用，在 SDN 网络中承担着关键角色。控制器通过南向接口，对网络中的设备进行集中控制与管理，同时通过北向接口，想上层应用开放编程接口。

目前，已经出现多种 SDN 开源控制器，Ryu[10]便是一种常用 SDN 控制器。Ryu 完全使用 Python 实现，目前由 NTT 实验室进行支持。Ryu 通过提供给用户统一的 REST API，使用户能够基于该框架开发自己的网络管理 APP。

2 系统功能结构

SDN 网络具有集中控制与可编程的特点，本文利用这些特点与传统网络中的 QoS 控制机制结合，设计实现了一个端到端的 QoS 控制系统。其系统功能结构如图 1 所示。

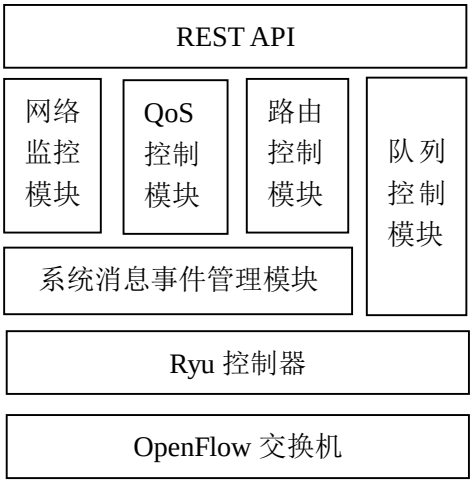

图 1 系统功能结构

Openflow 协议对于 QoS 的支持有两种方案，一种是 Meter，另一种则是使用队列。Meter 表虽然能对流进行更加细粒度的控制，但是由于这是一个 OpenFlow 协议可选项，因此目前只有少数 OpenFlow 交换机支持这一特性。队列机制则通过对 OpenFlow 交换机端口设置队列，采用不同的 QoS 策略，对流进行灵活的控制。本系统采用了队列的方案，因此设计了一个队列配置模块。

系统向上层用户提供了 REST API，用于预约网络的资源，并向系统提供 QoS 需求。同时也提供了查询流状态的接口，使用户可以及时获取流在网络中的相关信息。

系统中所有的消息以及事件都通过消息事件管理模块进行处理。比如说路由模块在求得路径之后需通过该模块下发流表项调整消息（Modify Flow Entry Message），网络监控模块需要采集网络信息的消息（Echo Request, Port Statistics Request 等）也通过该模块下发。OpenFlow 交换机发送给控制器的 Packet In 消息也首先由该模块接收。

本系统主要的功能模块为网络监控模块，路由

管理模块，QoS 控制模块。这三个模块相互协作，并依靠消息事件模块的帮助，对网内的流起到端到端的 QoS 保障。

3 核心功能设计与实现

3.1 网络监控模块

网络监控模块，主要对网络资源进行监控，并采集相应的性能参数，为路由管理模块与 QoS 控制模块起到帮助。本模块尝试完全使用 OpenFlow 协议实现，对网络拓扑，带宽，时延以及丢包进行监控。

网络拓扑结构通过 LLDP 报文进行获取，Ryu 控制器提供了一个 API，能够通过 LLDP 报文获取网络拓扑以及交换机和端口的信息。

带宽与丢包率的计算则是通过发送 Port Statistics Request 消息至 OpenFlow 交换机，接收 Port Statistic Reply 消息，来获取需要的数据，并计算相应的测度。

对于时延，本系统采用主动测量的方法。首先向 OpenFlow 交换机预先下发用于测量时延的流表项，之后通过周期性的下发报文，并在报文的数据中打入时间戳，然后通过计算获得时延。该部分使用了 OpenFlow 协议中的 Echo Request，Echo Reply 以及 Packet Out 消息。时延测量的流程如图 2 所示，交换机 s1 至交换机 s2 的时延 $T_{s1\to s2}$ 如 (1) 所示。

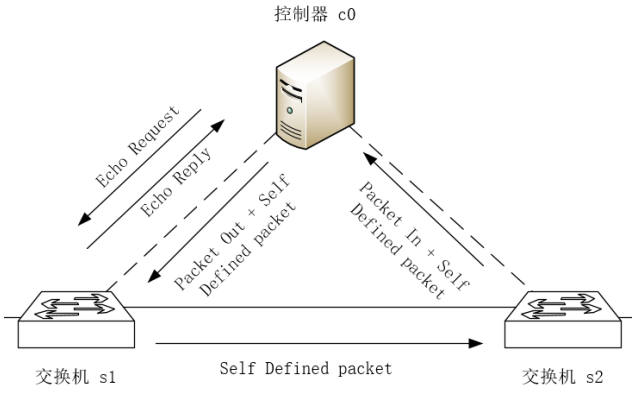

图 2 时延测量示意图

$$T_{s1\to s2} = T_{c0\to s1\to s2\to c0} - (T_{c0\to s1} + T_{c0\to s2})/2 \quad (1)$$

3.2 路由控制模块

路由控制模块，通过获取网络监控模块的测度以及拓扑信息，对于网络有一个全局的视角。当 QoS 控制模块向路由控制模块请求计算一条端到端的路径时，本模块会根据当前网络中的状态信息，以及相应的 QoS 需求，利用遗传算法进行路由计算。

将网络视为一张由 N 个节点 E 条有向边构成的有向图$\mathrm{G(N,E)}$，每一个节点的编号即为网络中的交换机编号，有向边也与网络中的链路一一对应。一条有向边具有 3 个测度d_1, d_2, d_3，他们分别表示带宽，时延与丢包率，这三个测度由网络监控模块提供，并实时更新。

本模块将一条路径上每一个交换机的编号按照顺序排列作为一个染色体。当需要搜索一条端到端的路径时，使用随机深度优先搜索。并且一条染色体 i 的适应度根据公式 (2) 进行计算。

$$F_i = \sum_{j=2}^{n} \left[\frac{S_{ij}}{\sum_{k=1}^{m} S_{kj}} \right] / (n-1) \quad (2)$$

上式中，n 表示测度的种类，m 表示一代染色体的数目，S_{ij}表示第 i 条染色体所表示的路径第 j 个测度的测度和。S_{ij}的求和公式如 (3) 所示，其中d_{ikj}表示第 i 条染色体所表示的路径上第 k 条边的测度 j 的大小。

$$S_{ij} = \sum_{k=1}^{p} d_{ikj} \quad (3)$$

由于丢包率是一个乘性参数应此需要使用公式 (4) 转换为加性参数，且还需要加上一个修正参数 1 进行结果修正，防止转换后的测度是一个负数。

$$d'_3 = \log(d_3 + 1) \quad (4)$$

带宽则是一个瓶颈性能参数，应此在对图进行预处理阶段，就将不满足带宽要求的路径从有向图中去除，所以在计算适应度参数的时候，只使用了时延与丢包率。

本模块的算法流程如下所示。

1) 接收 QoS 管理模块的需求，如 QoS 需求以及源地址与目的地址。
2) 根据带宽需求，将有向图 $\mathrm{G(N,E)}$ 中带宽不满足要求的边删除。
3) 根据算法设定的种群数量，初始化第一代染色体。
4) 选择：每次随机选择两条染色体，并选取其中适应度较高的一条染色体，该操作进行两次。
5) 交叉：若在选择操作选中的染色体拥有交叉节点，则进行交叉操作。
6) 反复进行选择和交叉直至新一代染色体的数量达到设定的种群数。

7) 变异：对于新一代染色体，分别根据变异率进行变异操作，即从一个节点开始重新随机生产路径，产生一条新的染色体。
8) 选取新一代染色体中适应度最差的一条染色体，并用上一代染色体中适应度最好的一条染色体进行替换。
9) 返回步骤 3），直至一条可行路径被发现。

3.3 QoS 控制模块

本模块利用了 SDN 网络的特点即集中控制，设计了并实现了端到端的 QoS 控制机制。

本模块向上接收来自 REST API 发来的用户请求，并根据请求的内容做出响应。若是一条端到端的 QoS 传输请求，则会向路由控制模块请求传输路径的计算，根据最终计算的结果决定接受请求还是进行拒绝。此外，该模块对网络中有 QoS 需求的流进行监控，当监控结果显示当前流的传输质量已经无法达到预设的需求时，则会向路由控制模块请求重新计算一条端到端的传输路径，之后进行传输路径的切换。本模块结构如图 3 所示。

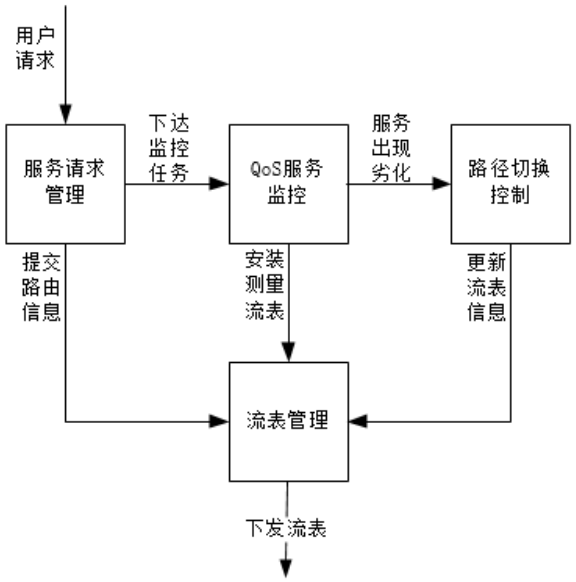

图 3 QoS 控制模块结构图

本模块需要实现传输路径切换的算法，与传统网络中依靠端口队列的队列控制算法来做拥塞避免不同，本系统尝试利用 SDN 网络的特点，通过对 QoS 服务实时监控并在服务劣化时通过调度的方法来做 QoS 控制。然而，在进行传输路径切换时，需要处理新老流表同时存在的问题，且需要按照一定的顺序下发流表项，避免新流表项进入网络设备后，对报文的传输产生影响，而产生不必要的丢包。

以图 4 中的路径切换方案为例，在这个例子中需要将传输路径[S1, S2, S3, S4, S5]切换为[S1, S2, S6, S4, S5]。算法的执行过程如下。

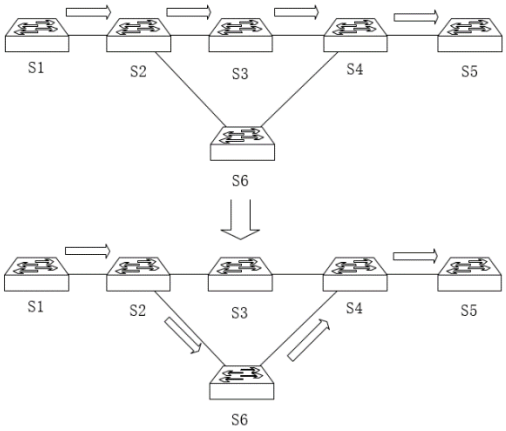

图 4 路径切换算法示意图

1） 向 OpenFlow 交换机下发新路径流表，下发流表项的顺序从后往前，在此例中新流表项的下发顺序为 S5，S4，S6，S2，S1。下发的新流表项的优先级需高于老流表项。
2） 对于同时存在新老流表项的 OpenFlow 交换机先将老流表项删除，由于新流表项优先级高于老流表项，老流表项此时已经废弃。此例中删除 S1，S2，S4，S5 四个 OpenFlow 交换机中的老流表项。
3） 当老路径一部分流表项被删除后，老路径被切分成一段段独立路径，此例中只剩一段路径[S3, S4]，其中 S4 没有老路径的流表项。对于这种剩余流表项，为了防止删除过程中路径上仍然存在报文，本系统采用人为构造最后一个报文的方法，在控制器中根据当前流的特征构造一个特殊报文，用于该特殊报文的流表项下发至路径中，此例为[S2, S3, S4]，其中最后一跳会将报文送回控制器。下发流表项完成后，下发报文。
4） 当控制器接收到一个特殊报文时，就将对应的老路径流表项与用于特殊报文转发的流表项一并删除。
5） 当老路径流表项删除完成后，此算法结束。

这种路径切换方案，能够尽量避免即将删除流表项的路径上，仍然存在处于传输过程中的报文，

防止由于路径的切换，造成的瞬时丢包率上升的问题。

4 系统测试

本系统使用 Mininet 进行实验，实验拓扑如图 4 所示。

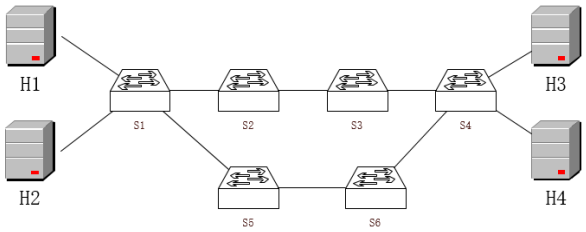

图 4 实验拓扑

利用 Iperf 产生数据流，对本系统进行测试。图 5 为网络监控模块测试的结果。对 QoS 管理模块的测试，首先测试同一条链路上同时存在 QoS 流与普通流时，QoS 流的带宽能够得到保障。再对路径切换功能进行测试，使用专为实验设置的 REST API，人为触发路径切换，对比了使用切换算法与不使用切换算法的切换，结果显示，不使用切换算法的路径切换过程中，由于失去了流表项的匹配，会有一部分报文被送入控制器，这对于系统负载和 QoS 流的服务十分不利，造成了时延的上升和服务的劣化。而使用了切换算法的路径切换，则没有该情况的发生。

Link(src switch, dst switch)	Link Capacity (Mbps)	Free Bandwidth(Mbps)	Latency(ms)	Loss
(1,2)	10000.0	9906.83940566	0.185489654541	0
(6,4)	10000.0	9999.9325648	4.59325313568	0.0
(3,2)	10000.0	9906.83940566	0	0.0
(4,6)	10000.0	9999.9325648	4.05323505402	0.0
(5,6)	10000.0	9999.93260514	0.348329544067	0.0
(2,1)	10000.0	9906.83940566	1.9633768988	0.000337268128162
(1,5)	10000.0	9999.93260514	2.11942195892	0.0
(2,3)	10000.0	9906.83940566	0	0.0
(4,3)	10000.0	9906.83937206	3.51369380951	0
(5,1)	10000.0	9999.93260514	0	0.0
(3,4)	10000.0	9906.83937206	2.22146511078	0.000179689906561
(6,5)	10000.0	9999.93260514	2.39300727844	0.0

图 5 网络性能监控

5 结束语

SDN 是一种崭新的网络架构，本文的主要工作是利用 SDN 的特点，设计并实现了一套 SDN 控制器层面的 QoS 控制系统，用来保证端到断的 QoS 需求，利用 SDN 控制器全局性的视角进行调度，从而保证服务质量。

参考文献

[1] Clean Slate. http://cleanslate.stanford.edu/index.php

[2] Greenberg A, Hjalmtysson G, Maltz D A, et al. A clean slate 4D approach to network control and management[J]. Computer Communication Review, 2005, 35(5): 41-54.

[3] Casado M, Garfinkel T, Akella A, et al. SANE: a protection architecture for enterprise networks[C]. usenix security symposium, 2006.

[4] Casado M, Freedman M J, Pettit J, et al. Ethane: taking control of Casado M, Freedman M J, Pettit J, et al. Ethane: taking control of the enterprise[J]. acm special interest group on data communication, 2007, 37(4): 1-12.

[5] Mckeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks[J]. acm special interest group on data communication, 2008, 38(2): 69-74.

[6] Open Networking Foundation. OpenFlow Switch Specification Ver 1.3.5 [s]. 2015.

[7] Braden R, Clark D, Shenker S. Integrated Services in the Internet Architecture: an Overview[J]. Ietf Rfc, 1994, 96(2):i-i.

[8] Zhang, Lixia, Deering, Stephen, Estrin, Deborah, et al. RSVP: a new resource ReSerVation Protocol[J]. Network IEEE, 1993, 7(5):8-18.

[9] Blake S, Black D, Carlson M, et al. An Architecture for Dierentiated Services[C]// Rfc. 1998:págs. 98-100.

[10] Ryu [EB/OL]. http://osrg.github.io/ryu/.