HiQoS: An SDN-Based Multipath QoS Solution

YAN Jinyao[1], ZHANG Hailong[2], SHUAI Qianjun[1], LIU Bo[1], GUO Xiao[1]

[1]Computer and Network Center, Communication University of China, Beijing 100024, P. R. China
[2] School of Information Engineering, Communication University of China, Beijing 100024, P. R. China

Abstract: Today's Internet architecture provides only "best effort" services, thus it cannot guarantee quality of service (QoS) for applications. Software Defined Network (SDN) is a new approach to computer networking that separates control plane and forwarding planes, and has the advantage of centralized control and programmability. In this paper, we propose HiQoS that provides QoS guarantees using SDN. Moreover, HiQoS makes use of multiple paths between source and destination and queuing mechanisms to guarantee QoS for different types of traffic. Experimental results show that our HiQoS scheme can reduce delay and increase throughput to guarantee QoS. Very importantly, HiQoS recovers from link failure very quickly by rerouting traffic from failed path to other available path.

Keywords: SDN; QoS; multipath; OpenFlow

I. INTRODUCTION

Over the years, researchers have been studying IP Quality of Service (QoS) and have proposed a series of theories and solutions of QoS. QoS provides performance guarantees for applications in terms of bandwidth, delay and packet loss etc. It is very important for many applications, particularly for multimedia applications, such as VoIP and IPTV. However, today's Internet architecture still provides only "best effort" services that cannot provide effective guarantees on quality of service. In the process of the development of QoS technology, the Internet Engineering Task Force (IETF) has explored several QoS architectures such as IntServ [1] and Diffserv [2], but none has been successfully and globally deployed. This is partly because these QoS architectures are built on top of a distributed hop-by-hop routing architecture without a global view. Instead of IP QoS and guarantees, overlay QoS for multimedia applications may provide some level of quality of service to improve the multimedia communication on the top of the best services[3][4][5][6][7]. By using fast label switching, Multi-Protocol Label Switching (MPLS) [8] provides a reliable solution to improve QoS performance. However, it is configured statically without real-time reconfiguration ability and adaptability [19].

Current QoS solutions, such as IntServ, may cost too much, or cannot provide an end to end QoS guarantee (e.g. DiffServ), or provides only soft and less efficient QoS such as application layer QoS. Software Defined Networking (SDN) [9] provides a possible way to solve these problems and paves the way for new QoS frameworks. SDN is a new approach for computer networking to manage network services with a global view and real time programmability through abstraction of lower level functionality. It separates the control plane and the forwarding plane of the traditional

network architecture. The controller running on the control plane can achieve the control objective for data forwarding through managing the switches' flow tables in a centralized way. SDN requires southbound interface for the control plane to communicate with the data plane. OpenFlow [10] [11] protocol is by far the most well-known southbound interface. OpenFlow switches can collect traffic information, such as sending and receiving data for each port, flow table and queue. The controller can be used to check the statistics maintained on switches, by sending a query message and perform traffic management according to these statistics.

In this paper, we propose HiQoS - a new QoS solution based on SDN technology. Importantly, HiQoS exploits multiple paths between the source and destination, and makes use of queuing mechanisms to achieve bandwidth guarantee for different types of traffic. Experimental results show that our HiQoS reduces the server response time and delay, increases the throughput of the system and its resiliency to link failure. It can timely reroute traffic from a failed path to other available paths.

The remainder of this paper is organized as follows. The second section introduces related work of QoS technologies, by describing the existing technologies of QoS and recently proposed SDN based QoS approaches. The third section presents our HiQoS solution, introduces the details of its design and implementation. We conduct experiments in the fourth section to show the performance of HiQoS in terms of delay and response time of the server, the throughput of the system and the recovery time to path failure. The fifth section concludes this paper.

II. Related Work

Researchers have been working on QoS support in the network for more than two decades. QoS solutions proposed in the past include the Integrated Services (IntServ) model, the Differentiated Services (DiffServ) model and some application layer QoS approaches such as OverQoS[3].

The IntServ was proposed to provide end to end QoS guarantee using the Resource Reservation Protocol (RSVP) [12] reserving resources such as bandwidth for flows along the path in the network. It is possible to be deployed in small-scale networks and capable of providing relatively high quality QoS. But as the number of flows increases and the network scale gradually expands, the resource consumption of each router will also increase very fast due to its complexity. This will make routers become the performance bottleneck and reduce the QoS performance of the whole network. DiffServ classifies traffic into different types of services and quality. It only guarantees the QoS for each group of traffic in one DiffServ domain, thereby reduces the complexity of implementation of this model, however, without the end to end QoS guarantee.

Both IntServ and DiffServ have not been deployed widely in the Internet. As a result, some overlay or application layer QoS approaches have been proposed to improve the quality of multimedia applications such as OverQoS [3] and media- and TCP- friendly congestion control algorithm in [7]. These application layer approaches provide only some level of quality on the top of best effort service instead of QoS guarantee. Content delivery network (CDN) [13] [14] is another approach

The authors propose HiQoS that provides QoS guarantees using SDN. Moreover, HiQoS makes use of multiple paths between source and destination and queuing mechanisms to guarantee QoS for different types of traffic.

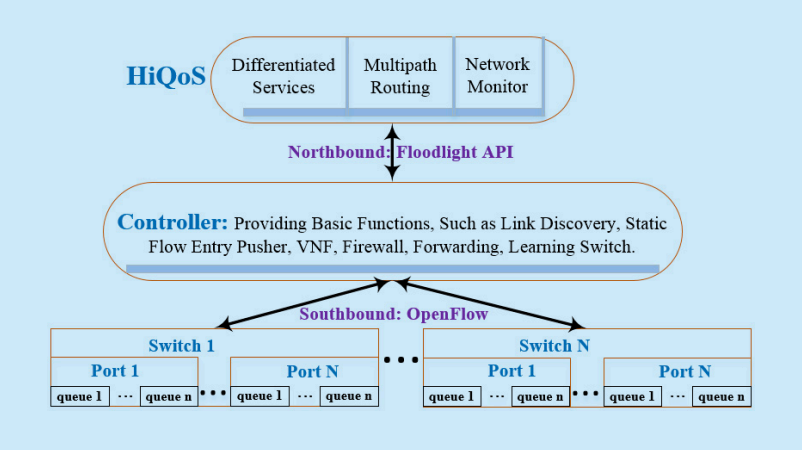

Fig.1 *Reference architecture of HiQoS*

to provide high performance and to improve the QoS to serve content to end-users with large distributed systems of servers deployed across the Internet. However, any intermediate or last-mile user network are fully out of the control of CDN servers.

Recently, some new QoS approaches based on SDN have been proposed. Govindarajan et al. [15] proposed a QoS Controller, the Q-Ctrl, with five major operations including queue creation, QoS flow addition, QoS flow modification, QoS flow deletion and queue deletion. Cinvalar et al. [16] proposed an optimization model to improve packet routing in OpenFlow enabled networks. Such a model considers delay and packet loss. Their optimization model uses linear programming to compute a QoS path for video traffic and a shortest path for best effort traffic. Kim et al. [17] propose a network QoS control framework based on SDN, allowing programming QoS parameters on network devices. However, the authors do not describe the extensions implemented on SDN networks and the performance of the framework is not evaluated. Egilmez et al. [18] [19] proposed an OpenFlow protocol based controller, namely OpenQoS to achieve end-to-end QoS for multimedia-based applications. It is extended to a distributed architecture in [20]. OpenQoS classifies the traffic into data and multimedia flows. The multimedia flows are following the dynamic QoS guaranteed routing algorithm, whereas the other data flows are following the shortest routing path algorithm. Unlike another OpenFlow-based QoS framework in [17], OpenQoS does not use the queue mechanisms that OpenFlow provideds to realize QoS guarantees for different service types. When the traffic of certain types exceeds the threshold, it can still cause link congestion. In our proposed HiQoS, the controller does not only use queue mechanisms to achieve bandwidth guarantee for different types of traffic, but also realize multipath routing to make use of available bandwidth in the network. This idea of multipath routing follows up on our previous work on SDN-based ECMP (Equal Cost Multipath Routing) algorithm for data center networks [21]. HiQoS aims to achieve end-to-end QoS for different types of applications such as web traffic and multimedia streaming.

III. HiQoS

In this section, we describe the detailed design of our proposed HiQoS. HiQoS uses SDN technology to achieve multipath QoS with differentiated services. HiQoS is mainly divided into two components: the differentiated service component and the multipath routing component. In the differentiated services component, the controller distinguishes different types of services by identifying the IP address of the source node and provide different bandwidth guarantees to different services through queuing mechanisms on the SDN switches. The multipath routing component finds multiple paths meeting certain QoS constraints between the source node and the destination node, and calculates the optimal path for each flow by real time monitoring of the network state.

The reference architecture of HiQoS is shown in Figure 1. It consists of an SDN controller in the middle, HiQoS as an SDN application, and the OpenFlow switches of the network. The SDN controller performs the basic functions such as link discovery, network statistics, and security checks and translates the requirements from the SDN application (i.e. HiQoS) to appropriate OpenFlow rules that then installes on the OpenFlow switches. HiQoS implements our proposed multipath QoS solution mainly with the differentiated services component and the multipath routing component. Switches line up incoming packets into queues and forward them to the appropriate port based on the flow tables and queue policies ruled by HiQoS through the SDN controller.

In the following, we describe the details of the differentiated services component and the multipath routing component in HiQoS.

3.1 The differentiated services component

In the implementation of the differentiated services component, one task is to classify the application data for different QoS level of services. In HiQoS, we classify user traffic into video stream with high bandwidth requirements, interactive audio/video with low delay requirements and normal data stream as best effort services [29]. Commonly used techniques to differentiate flows include methods that are: a) based on source IP address; b) based on source port; c) based on MAC address; d) based on Type of Service (ToS) bits; e) traffic class header field in MPLS. In particular, we may differentiate data flows by setting the Match Fields (Source Port, Vlan, Ethertype, Source IP Address, Destination IP Address) of flow entries in OpenFlow switches. We use the method based on the source IP address in our first implementation of differentiated services as the controller can quickly query the source IP address information from the received packet. More complicated methods could also be used.

To be able to provide bandwidth guarantee for each particular type of applications, we use queuing mechanisms provided by the OpenFlow protocol [11]. An OpenFlow switch provides limited QoS support through a simple queuing mechanism. One or more queues can be attached to a port and then be used to forward packets via it. Packets forwarded to a specific queue will be processed according to that queue's configuration (e.g. minimum rate or maximum rate).

In HiQoS, the different types of data packets are forwarded to different queues. Data are transmitted in the queue according to the presently available queue bandwidth. This makes the normal data flow to not affect the real-time video streaming or real-time audio stream, and there is also no interference between real-time video stream and real-time audio stream.

The use of queuing mechanisms guarantees network bandwidth for different types of applications. However, when the total traffic of a certain type application is larger than the bandwidth allocated for its queue, the queue congestion will occur that may result in the loss of data packets and retransmissions.

3.2 The multipath routing component

In order to avoid the packet congestion mentioned above, we have introduced multipath routing technology into HiQoS to make use of multiple available paths and their additional bandwidth.

Here, we modified the Dijkstra algorithm [22], enabling it to calculate multiple paths that satisfy certain QoS constraints. These paths are stored in a HashMap and the controller will periodically check the network state of these paths. When a packet arrives at an OpenFlow switch that doesn't have a matching flow entry for it, it is first sent to the controller. According to the current network state (including queues and bandwidth utilization), the controller will selects an optimal path from the multiple available paths in the network and push flow entries to the OpenFlow switches for packet forwarding. Such QoS solution ensures that the data packets are always forwarded to the current optimal path selected by the multipath routing component. Therefore, HiQoS makes use of multiple available paths and their bandwidth.

3.2.1 Calculation of multipath

For the multipath calculation, we characterize the routing topology as a weighted graph $G<V, E>$ [23]. V is the set of nodes in the network. E is the set of edges that connect any two nodes. Each edge e has a link cost value L_e which is defined in equation (1).

$$L_e = \alpha \cdot price(e) + \beta \cdot dist(e) - \gamma \cdot bw(e) \quad (1)$$

where $price(e)$ is a combination value estimated by the price for using the link, the link stability and robustness. $dist(e)$ and $bw(e)$ are the physical distance and the bandwidth of the link respectively. α, β and γ are the weights of the cost respectively. We define all the paths from a source node to a destination node as

a set $P=\{p_1,p_2,\ldots, p_{4k},\ldots\}$. For example, for a flow from node s to destination node z in Figure 2, here $s,z \in V$,$z \in V$, we can find that there are 4 paths $\{p_1:(e_1e_4e_7e_8e_9), p_2:(e_1e_5e_9), p_3:(e_2e_9), p_4:(e_3e_6)\}$. The total cost of each path is defined in equation (2).

$$L_{pk} = \sum_{e \in p} L_e \quad (2)$$

In this experiment, we set the threshold of the path cost value as L_0. The paths whose total cost value satisfies the equation (3) are chosen as candidate path.

$$L_{pk} \le L_0 \quad (3)$$

The pseudo code of our proposed HiQoS multipath routing algorithm is shown in Table I. We modified the Dijkstra algorithm with our proposed algorithm of multipath calculation satisfying certain QoS constraints.

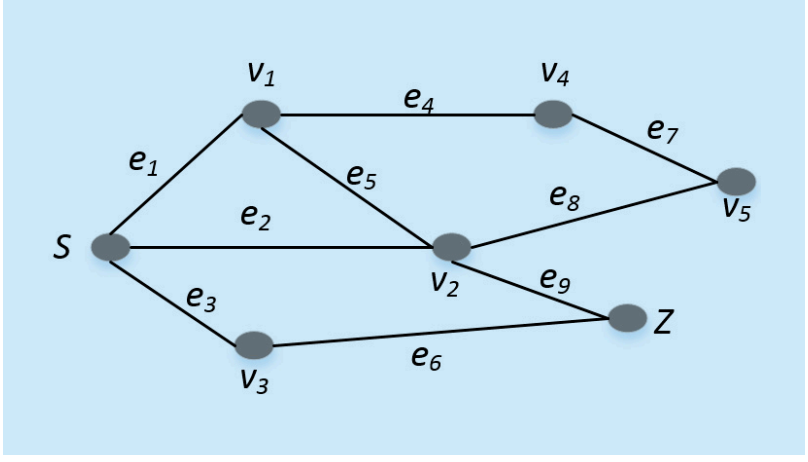

Fig.2 *Graph for multipath calculation*

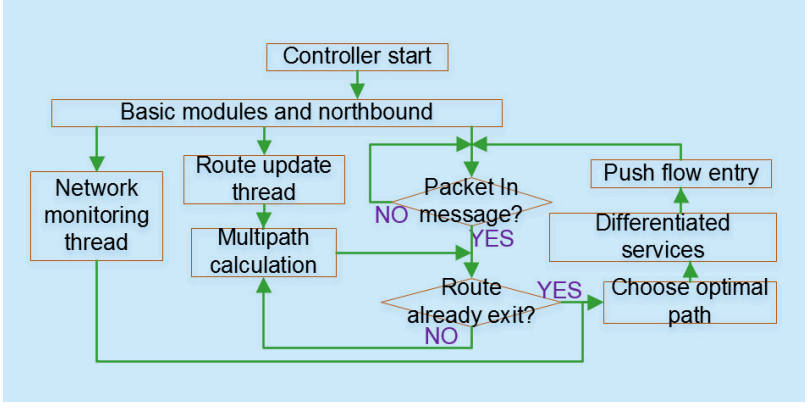

Fig.3 *Work flow of HiQoS*

3.2.2 Optimal route selection

In HiQoS, the path with the minimal bandwidth utilization of a queue will be selected as the optimal path for a new flow. The controller periodically measures the bandwidth utilization of each queue along the path $bwu(p_k,q_i)$. Where q_i is the queue for a specific type i of traffic of path p_k, $bw(q_i)$ is the bandwidth of queue i , $T(q_i)$ is the number of transmitted packets of the queue i, and Δt is the interval time of measurement.

$$bwu(p_k, q_j) = T(q_i)/(\Delta t \cdot bw(q_i)) \quad (4)$$

The higher utilization of queues, the more congestion in the path. We select for an incoming flow a path that its bandwidth utilization of the queues is the minimum compared to other paths. In this way, HiQoS selects the optimal path to avoid congestion for received data packets. This enables the system to achieve QoS guarantees that will reduce the delay and the response time of the server, increase the throughput of the system.

In our implementation, the controller periodically measures the transmitted packets of each queue in the paths, and calculates the optimal path. Then the Controller pushes flow entries to the OpenFlow switches. We also set an aging flag that represents the state of the data transmission. With this aging flag, the controller deletes flow entries and multipath cache list after some amount of time of the data transmission. In our implementation, the Idle Timeout of flow entries is set to 5 second in HiQoS. Figure 3 shows the work flow of HiQoS with optimal path selection for flow data.

IV. Experiments and Performance Results

In this section, we conduct experiments to exam the performance of our proposed SDN-based multipath QoS solution (i.e. HiQoS) in terms of the server response time (the delay of transmission), system throughput and the resilience to path failure.

4.1 Settings of the experiments

We conduct experiments with Mininet on Linux host. Mininet [24] is a network simulator that can conveniently create a network topology with virtual OpenFlow switches. We use the Floodlight [25] as the SDN controller. The experimental network topology is shown in Figure 4.

For the performance comparison, we designed another two routing schemes which called LiQoS and MiQoS. LiQoS is single-path solution without differentiated services support at switches. MiQoS is also single-path solution with differentiated services support at switches to provide bandwidth guarantee by queue mechanisms.

- LiQoS:

LiQoS selects the path that has the minimum total link L_{pk} for data transmission. The OpenFlow switches only forward traffic in "best effort" way without bandwidth guarantee. This scheme simulates the traditional network transmission that has no QoS guarantee, which causes interference between the various types of data flows.

- MiQoS:

MiQoS also chooses the path which has the minimum L_{pk}, but OpenFlow switches classify flows and queue up for different types of services. In our experiments, the OpenFlow switch has two queuing buffers for each link port. Different type of traffic will be transmited through different queues. For example in our experiments, traffic from server 1 is queued and forwarded through queue 1, while traffic from server 2 gets forwarded through queue 2.

- HiQoS:

As we described above, HiQoS transmits data through multiple paths and provide QoS guarantee through queue mechanisms. The queuing parameters are set the same as in MiQoS. Above all, the controller periodically inquires the total number of bytes sent from each queue and chooses an optional path from the multiple available paths for forwarding data from a new flow. In MiQoS, when the traffic of one server increases, the queue will experience congestion. But in HiQoS, data

Table I *The pseudo code of calculating multiple paths*

```
Initialization
use HashMap<> to store previous node previous<>, link cost costs<> and initialization
Calculate the previous<> of each switch
while queue is not empty:
    get the first node from queue
    if node equals to destination switch:
        break
    end if
    for each switch connected to this node:
        calculate the total cost between this switch to source switch
        if this switch has been checked:
            continue
        end if
        if the value of total cost satisfy our limitation:
            add this switch to previous<>
            costs<> = minimum{ cost<>, total cost}
            put this switch node into the queue
        end if
    end for
end while
Generate the multiple paths
set a list to store information of the path
set destination switch as current node
function generate-multiple-path(routes, src, dst, current, previous, list):
    if current node equals to source switch:
        path = new Route<>
        add this path to multiple path cache
        return
    end if
    for each link connectecd to the current node:
        add this link to the list
        generate-multiple-path (routes, src, dst, current, previous, list)
        remove two nodes of this link
    end for
end function
```

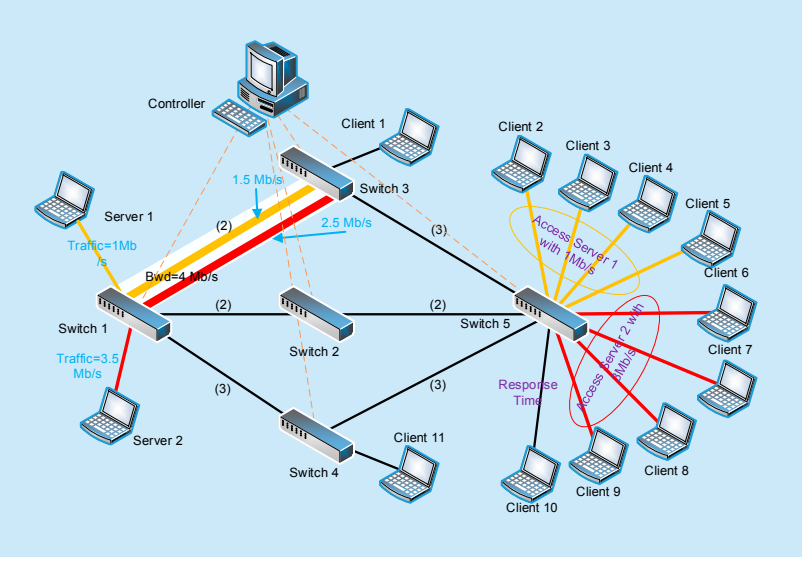

Fig.4 *The experiment topology*

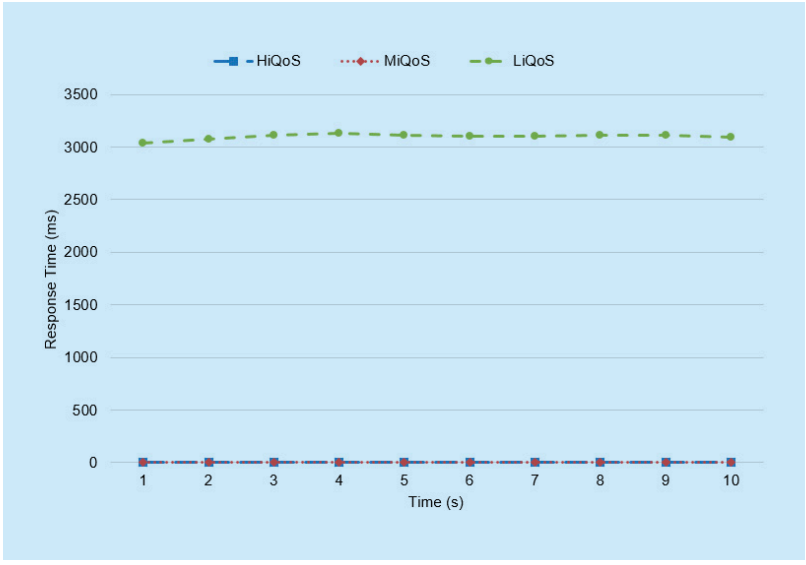

Fig.5 *The response time of server 1*

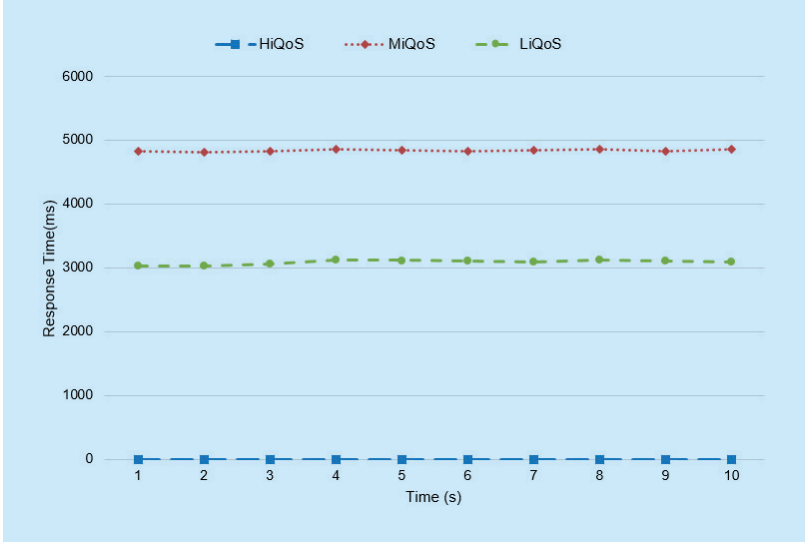

Fig.6 *The response time of server 2*

are transmitted through multiple paths, which indirectly increases the queue bandwidth, thus enhance the overall performance of the system.

The experiment topology shown in Figure 4 has 5 switches, 2 servers and 11 clients. The controller controls the switches through the OpenFlow protocol. Two servers provide different types of services to users respectively. Server 1 provides users with real-time video streaming, and server 2 provides users with a generic data sharing service. The clients randomly access these two servers. There are multiple paths between Switch 1 and Switch 5. Values in parentheses represent the link cost between nodes. HiQoS uses Equations (1), (2), (3) to calculate the multiple paths. The value of L_0 is set to 10 in our experiments. In Figure 4, *Bwd* means the total bandwidth of the link while Traffic means the total access traffic of each server.

When the system is running LiQoS, traffic is transmitted through a single path without queues for differentiated services. When the system is running MiQoS, traffic is transmitted through a single path too, however switches classify flows and queue up for different types of services. HiQoS not only uses the queue buffer as shown in Figure 4, but also uses the multiple paths for better performance.

4.2 Performance and results

In the following, we conduct three series of experiments to show the performance in terms of the server response time (the delay of transmission from server to client), and system throughput respectively.

4.2.1 Response time and delay

In Figure 4, the total bandwidth of the link is 4 Mbps with two queues. The bandwidth of queue 1 is 1.5 Mbps while queue 2 is 2.5 Mbps. We assume that the server 1's priority is higher for real time media services and it requires a certain bandwidth guarantee via queue 1. Traffic packets from server 2 don't require bandwidth guarantee, thus transmit packets via queue 2. Multiple clients are accessing server 1 and server 2 respectively. The total throughput of the traffic of server1 is 1 Mbps of media streaming, while server 2 is with 3.5 Mbps of data traffic. We use Ping to measure the response time of server 1 and server 2 which is represent to the delay of data transmission from server to clients. This experiment can verify the ability of HiQoS to provide bandwidth guarantee to different types of services. We record the response time of each server for 10s and show the obtained measurements in Figure 5 and Figure 6.

From Figure 5 we can see that the response time is very short in HiQoS and MiQoS, because we use queues to provide bandwidth guarantee for server 1. But in LiQoS, the response time of server 1 is very large since the data packets are transmit through the same one queue, where the total traffic exceeds the link bandwidth and leads to link congestion.

Figure 6 shows the response time of server 2. In HiQoS, by utilizing multiple paths, traffic from server 2 is distributed in the three paths to eliminate congestion, so the response time is also very small. But in MiQoS and LiQoS, the response time of server 2 is very large due to the congestion of the path. Figure 6 also shows that the response time in MiQoS is greater than LiQoS when congestion occurs. This is because in MiQoS, the bandwidth of the queue for traffic from server 2 is 2.5 Mbps which is less than 4 Mbps of total bandwidth in LiQoS.

In another experiment, we change the traffic load of server 2 using Iperf [26], and then check the performance of server 1 whose QoS for its flows should be guaranteed. The traffic of server 1 is still 1 Mbps. We use the Ping tool to test the average response time for each server. The simulation results are shown in Figure 7 and Figure 8.

Figure 7 and Figure 8 show that server 1 obtains bandwidth guarantee and its response time is not affected by the traffic of server 2 when the system is running HiQoS and MiQoS. As there is no differentiated services and bandwidth guarantee in LiQoS, the response time of server 1 and server 2 increased rapidly when the total traffic exceeds the link bandwidth. As we can see from Figure 8, when the traffic of server 2 exceeds the bandwidth of queue 2 in MiQoS and LiQoS, the response time of server 2 starts to rise. In Figure 8, the response time of HiQoS is very small. This is because HiQoS uses multiple paths for data transmission. The total bandwidth of HiQoS is three times the single path. So the traffic of 4.5 Mbps does not exceed the total bandwidth of the network.

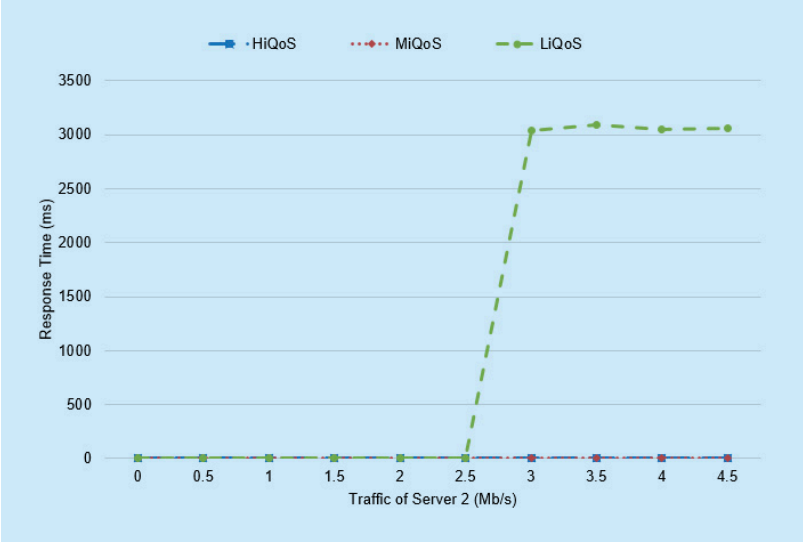

Fig.7 *The response time of server 1*

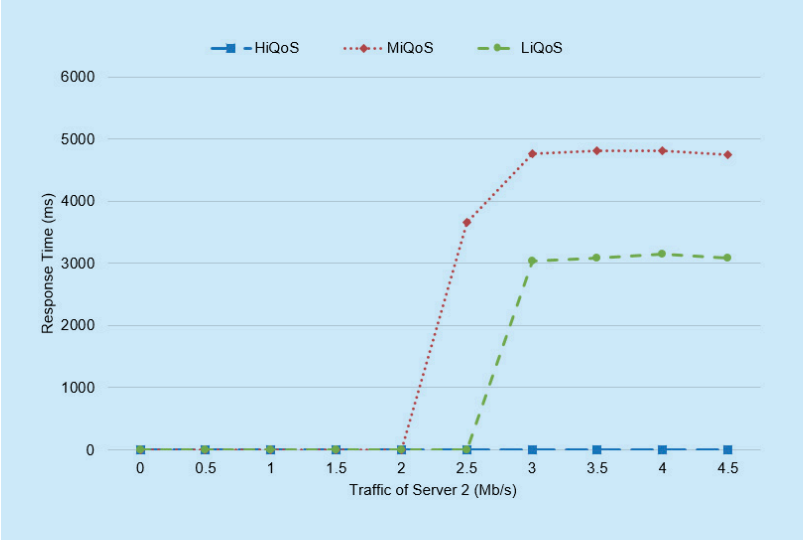

Fig.8 *The response time of server 2*

4.2.2 Throughput

This experiment is to test the system throughput when using different QoS solutions. The experiment topology is the same as in previous experiments. Multiple clients are accessing server 1 and server 2. We increase traffic to exceed the bandwidth of the network, so that we can measure the good throughput of each QoS solution. We periodically measure the throughput of switch 1 every 5 seconds to calculate the average throughput of the system.

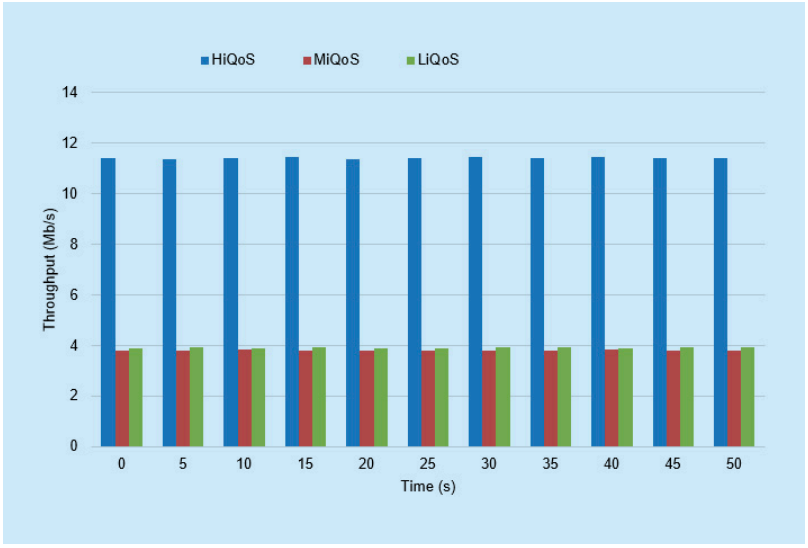

Fig.9 *The throughput of our system*

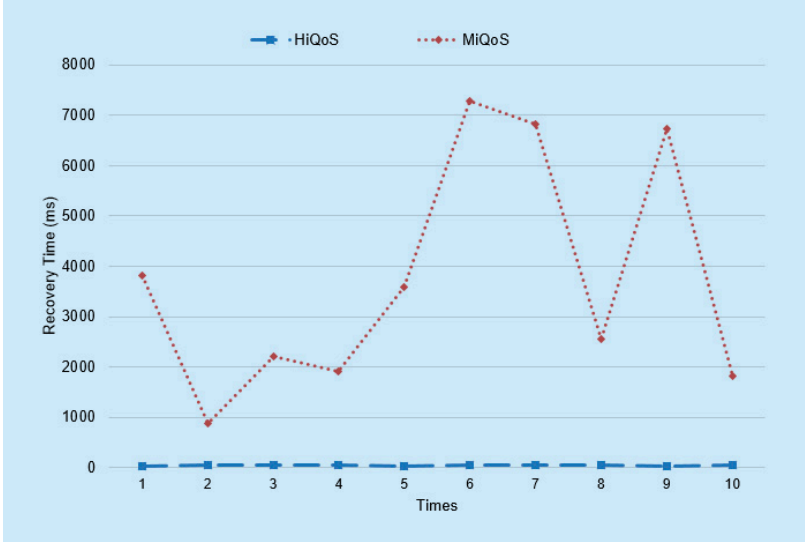

Fig.10 *The recovery time of our system*

The simulation results are shown in Figure 9.

Figure 9 shows that the throughput of MiQoS and LiQoS are about 3.8 Mps which is close to the path bandwidth of 4 Mbps. HiQoS has the maximum throughput, reaching approximately 11.4 Mbps, which is about three times more than when compared with the other two solutions. This is because HiQoS can transmit data via multiple paths, which indirectly increases the total bandwidth of the system and also increase the system throughput.

4.2.3 Resilience to path failure

In the following, we conduct experiments to show the performance of the resilience to path failure.

Unexpected disasters and human misbehavior can cause network outages, thus the recovery time of the network is a key factor for QoS. In order to verify the performance of HiQoS in disaster, we designed an experiment with link failure events. In this experiment, the link from switch 1 to switch 3 was suddenly disconnected at a certain moment and hence the service was interrupted by this link failure. We measure the recovery time of services from server 1 with HiQoS and MiQoS running respectively.

We set the interval time for routing update to 10 seconds as the rerouting time and recovery time are typically from several seconds to several minutes both for OSPF and BGP[27] [28].

Figure 10 shows that the recovery time of HiQoS is much less than that of MiQoS. With HiQoS, server 1 can resume services immediately. This is because HiQoS stores and uses multiple paths while MiQoS has only single path for flows. When a path is disconnected, HiQoS immediately chooses a path from several other paths for data transmission. MiQoS has to recalculate the route for broken paths at the time of a route update. HiQoS improves the resilience performance through multiple paths, and thus it is more practical than MiQoS.

We conclude from above experiments that HiQoS significantly improves the overall performance of the system. HiQoS introduces differentiated services and increases bandwidth usages for different types of services. Moreover, HiQoS recovers from link failure very quickly, thus it is resilient to disasters.

V. Conclusions

In this paper, we proposed a HiQoS solution based on OpenFlow technology. HiQoS takes advantages of differentiated services to

guarantee bandwidth, and applies multipath routing to make use of the available capacity in the network. The experimental results show that the HiQoS solution reduces the delay of data transmission and the response time of servers, improve the throughput of the system. Very importantly, our proposed HiQoS timely reroutes traffic from a failed path to other available paths.

Acknowledgements

This work was supported partly by NSFC (National Natural Science Foundation of China) under grant No.61371191 and No.61472389.

References

[1] R. Braden, D. Clark and S. Shenker. "Integrated Services in the Internet Architecture: an Overview". IETF RFC1633, (1994) June, pp. 2-3.

[2] S. Blake, D. Black, M. Carlson, et al. "An Architecture for Differentiated Service". IETF FC2475, (1998) December, pp. 2-9.

[3] Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, et al. "OverQoS: offering Internet QoS using overlays", SIGCOMM Comput. Commun. Rev., Vol. 33, No. 1. (January 2003), pp. 11-16.

[4] Dejan Kostic, Adolfo Rodriguez, Jeannie Albrech, et al. "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh". Proceedings of the 19th ACM Symposium on Operating System Principles, October 2003.

[5] Sally Floyd, Mark Handley, Jitendra Padhye, et al. "Equation-Based Congestion Control for Unicast Applications". August 2000. ACM SIGCOMM 2000.

[6] Jinyao Yan, Wolfgang Mühlbauer, Bernhard Plattner, "Analytical Framework for Improving the Quality of Streaming Over TCP". IEEE Transactions on Multimedia 14(6): 1579-1590 (2012).

[7] Jinyao Yan, M. May, K. Katrinis, et al. "Media- and TCP-Friendly Congestion Control for Scalable Video Streams", IEEE Transactions on Multimedia, vol.8, No.2, Apr. 2006.

[8] E. Rosenetal. "Multiprotocol Label Switching Architecture". IETF RFC3031, (2001) January, pp. 2-3.

[9] Thomas D. Nadeau and Ken Gray, Editor, "SDN: Software Defined Networks", O'Reilly Media, Inc, USA (2013).

[10] N. McKeown, T. Anderson, H. Balakrishnan, et al. "OpenFlow: Enabling Innovation in Campus Networks". ACM SIGCOMM Computer Communication Review, (2008) April 2; New York, USA.

[11] OpenFlow Switch Specification, Version 1.1, http://www.openflow.org/wp/documents.

[12] Zhang, L., Deering, S., Estrin, D., et al."RSVP: A New Resource ReSerVation Protocol", IEEE Network, September 1993.

[13] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun, "The Akamai Network: A Platform for High-Performance Internet Applications", ACM SIGOPS Operating Systems Review, vol. 44, no. 3, July 2010.

[14] Vakali, A.; and Pallis, G. (2003). "Content Delivery Networks: Status and Trends". IEEE Internet Computing 7 (6): 68–74.

[15] Kannan Govindarajan, Kong Chee Meng, Hong Ong, et al. "Realizing the Quality of Service (QoS) in Software-Defined Networking (SDN) based Cloud Infrastructure". In Proceedings of IEEE International Conference on Information and Communication Technology, (2014) May 28-30; Bandung, Indonesia.

[16] Civanlar S, Parlakisik M, Tekalp A M, et al. "A Qos-Enabled Openflow Environment for Scalable Video Streaming". In Proceedings of IEEE GLOBECOM Workshops, (2010) December 6-10; Miami, USA.

[17] W. Kim, P. Sharma, J. Lee, et al. "Automated and Scalable QoS Control for Network Convergence". Proceedings of the 2010 internet network management conference on Research on enterprise networking, (2010) April 27; Berkeley, USA.

[18] H. E. Egilmez, S. T. Dane, K. T. Bagci, et al. "OpenQoS: An OpenFlow Controller Design for Multimedia Delivery with End-to-End Quality of Service over Software-Defined Networks". In Signal & Information Processing Association Annual Summit and Conference, (2012) December 3-6; Hollywood, USA.

[19] H.E. Egilmez, S. Civanlar, A.M. Tekalp, "An Optimization Framework for QoS-Enabled Adaptive Video Streaming over OpenFlow Networks", IEEE Trans. on Multimedia, 2013.

[20] Egilmez, H.E. "Distributed QoS Architectures for Multimedia Streaming over Software Defined Networks," Multimedia, IEEE Transactions on, vol.16, no.6, pp.1597, 1609, Oct. 2014.

[21] Hailong Zhang, Xiao Guo, Jinyao Yan, et al. "SDN-Based ECMP Algorithm for Data Center Networks". IEEE ComComAp 2014, Beijing China. Oct., 2014.

[22] Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs". Numerische Mathematik 1: 269–271.

[23] John M. Harris, Jeffry L. Hirst, Michael J. Mossinghoff, "Combinatorics and Graph Theory". 2nd Edition, New York, NY: Springer, 2008.

[24] Mininet. http://mininet.org.

[25] Project Floodlight. http://www.projectfloodlight.org/floodlitght.

[26] Iperf: http://sourceforge.net/projects/iperf/.

[27] Sahoo, A.; Kant, K.; Mohapatra, P., "Characteri-

zation of BGP Recovery Time under Large-Scale Failures," Communications, 2006. ICC '06. IEEE International Conference on, vol.2, no., pp.949, 954, June 2006.
[28] Watari, M., Hei, Y., Ano, S., et al. "OSPF-Based Fast Reroute for BGP Link Failures," Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE, vol., no., pp.1, 7, Nov. 30 2009-Dec. 4 2009.
[29] J. Babiarz, K. Chan, F. Baker, "Guidelines for Diff-Serv Service Classes", RFC 4594, August 2006.

Biographies

YAN Jinyao, received the B.S. degree from Tianjin University, the M.S. and Doctor (in engineering) degrees from Beijing Broadcasting Institute, the Ph.D (in science) degree from Swiss Federal Institute of Technology (ETH Zurich). Since 2010, he has been a Professor at Communication University of China, Beijing, P.R.China. He was a guest professor in communication system group at ETH Zurich 2012-2013. His research interests are in the areas of future network, multimedia communication, and cloud computing and network security. Email: jyan@cuc.edu.cn

ZHANG Hailong, was born in 1988. Hi is studying information network technology in Communication University of China, working toward Master degree. His current research interests include SDN, network security. Email: zhlcuc@163.com

SHUAI Qianjun, received the M.Sc. and Ph.D. degrees in electronic engineering and communication system engineering from Communication University of China in 2004 and 2012, respectively. From Jan. 2012 to Jan. 2013, she was a visiting scholar with Electrical & Computer Engineering department at New Jersey Institute of Technology. She is currently an associate professor of Computer and Network Center at Communication University of China. Her research interests are in the area of broadband access network, cloud computing and resource assignment algorithms.

LIU Bo, was born in 1978. He received Master degree from China University of Petroleum (UPC). He is a network engineer at Communication University of China.