FUDP: AN SDN-BASED MECHANISM FOR CONTROLLING UDP FLOWS

Hui Hu,* Bo Liu,** Chao Hu,** Ming Chen,** Guang Cheng,*** and Changyou Xing**

Abstract

It is always a complicated problem to effectively manage and control UDP traffic on the Internet. As UDP traffic is still the chief component of the Internet, the Internet would become difficult to operate stably once network congestion had occurred. Therefore, it is necessary to explore effective control mechanisms for UDP traffic under the current IP framework. This paper introduces the concept of friendly-UDP (FUDP), which aims to give UDP flows quasi-TCP-friendly characteristics. A FUDP mechanism is designed based on software-defined networking (SDN), which controls UDP traffic through flows in a closed-loop manner. The experimental results on the test bed and real Internet trace-driven analysis show that the FUDP mechanism enables UDP to have FUDP features, which avoids network congestion in bottlenecks and allows the whole network to achieve higher throughput and greater bandwidth fairness.

Key Words

UDP flow, congestion control, TCP-friendly, SDN, fairness

1. Introduction

TCP-friendliness is a desirable attribute that helps to maintain bandwidth fairness between multiple TCP flows in two ways: one is fairness in using network resources, and the other maintains the stability of the network by avoiding network congestion. In contrast, UDP flows do not provide any support with respect to the above characteristics at all. UDP flows completely ignore the existence of other flows and grab all resources greedily to meet their own demands. Accordingly, when TCP and UDP flows are transmitted over the same bottleneck, the TCP flows can seldom obtain a fair portion of the bandwidth. The greedy behaviour of UDP flows not only pushes TCP flows to the point of starvation, but it also wastes a considerable amount of bandwidth [1], which risks the collapse of the Internet because UDP traffic is still around 20%–50% of total Internet traffic [2], [3]. Therefore, efficient management and control of UDP traffic is essential to maintaining the availability and high resource utilization of the Internet.

Both academia and industry are strongly motivated to develop an effective UDP traffic control mechanism to avoid the negative effects of UDP traffic on the network, while maintaining the fine support of UDP for some network applications. However, the proposed methods, such as adding scheduling mechanisms to routers, modifying UDP or TCP protocols, and even introducing middleboxes on the Internet, still do not address this issue well. In this paper, we propose a mechanism that confers quasi-TCP-friendly characteristics to UDP flows as mentioned above based on SDN [4]. SDN uses a programmable centralized controller to control each flow's bandwidth allocation and regulate UDP flows by defining the behaviour of forwarding tables [5], which is easily deployed in current IP networks [6], [7]. Specifically, we propose the concept of FUDP and implement a prototype system for the FUDP mechanism based on SDN and conduct a series of experiments to verify the feasibility and availability of the FUDP mechanism.

2. Related Work

In the past 20 years, the relationship between TCP and UDP has been extensively investigated in the literature. Floyd and Fall [1] first explored the issue of unfair competition between TCP and UDP in network bandwidth problems and pointed out that the Internet will be unable to maintain stable operation without control of UDP traffic. They proposed adding a scheduling mechanism [*e.g.*, weighted fair queuing (WFQ)] in routers to limit the overuse of bandwidth resources by UDP flows to maintain fairness between the TCP flows and UDP flows. Furthermore, the AQM [8], ECN [9], RCP [10], and XCP [11] technologies were proposed to improve the performance of TCP flows and network fairness. All of these mechanisms require new protocols, router hardware, or packet header formats, and so, deployment of these mechanisms has been rare.

Some of the works in the literature suggest modifying UDP to support window-based TCP, such as DCCP [12] (with CCID3 [12] control mechanism) and TEAR [13]. However, other schemes used rate-based control

* Information Center, Northwestern Polytechnical University, Xi'an, P.R. China; e-mail: huhui_email@126.com

** College of Command Information Systems, PLA University of Science and Technology, Nanjing, P.R. China; e-mail: lbo.xidian@163.com, huchaonj@126.com,mingchennj@163.com, changyouxing@126.com

*** College of Computer Science and Engineering, Southeast University, Nanjing, P.R. China; e-mail: chengguang@seu.edu.cn

Corresponding author: Hui Hu

Recommended by Dr. Xiaonv Hu

(DOI: 10.2316/Journal.206.2018.5.206-0066)

algorithms in the UDP protocol to limit the injection traffic of UDP flows, such as the Rate Adaptation Protocol [14] and Loss-Delay based Adjustment Algorithm [15], [16]. Equation-based Congestion Control [17] methods proposed controlling the sending rate of UDP flows according to a mathematical model based on the TCP sending rate, but they still face many challenges [18]. The approaches noted above are hard to deploy on the Internet because the protocol stacks of the end hosts must be modified. Other methods, such as Fast TCP [19], CUBIC [20], TCP-Illinois [21], and PCC [22], have suggested modifying the TCP protocol to let the TCP flows rapidly occupy the available bandwidth, but nevertheless, they do not effectively control UDP traffic.

UDP flows could also be adjusted by using middleboxes [23] or a traffic shaper at the network edge. However, the middleboxes would have to satisfy the following requirements: (i) they should identify and process each UDP flow in the line speed and; (ii) they should track and evaluate the network resources and all flows, and then the UDP flows can be controlled effectively. However, current middleboxes can only control specific UDP flows according to a pre-set rate and they cannot satisfy the above requirements.

3. Overview of the FUDP Mechanism

3.1 Related Concepts

Definition 1: Correlation flow *For any two flows, f_i and f_j, transmitting on path p_i and p_j, they are called the correlation flow if $l_i = l_j$, where l_i is the bottleneck link of f_i on p_i and l_j is the bottleneck link of f_j on p_j.*

Flows sharing the same bottleneck link are defined as a correlation flow set (CFS). Shared congestion (also called a shared bottleneck) detection technology [24] can identify the correlation flows, although this issue is out of the scope of this paper.

Definition 2: FUDP *In a CFS, we defined the UDP flows as FUDP if the UDP flows can fairly share the bottleneck bandwidth with the TCP flows. The mechanism that enables UDP flows to be FUDP is called the FUDP mechanism.*

Specifically, if there are n TCP flows and m UDP flows in a CFS and the available bandwidth of the bottleneck link is BW, the bandwidth derived from each flow (TCP flow f_T or UDP flow f_U) should be approximate to $BW/(n+m)$ in the stable state. Here, $BW = \sum_{i=1}^{n} B_{f_{Ti}} + \sum_{j=1}^{m} B_{f_{Uj}}$, where $B_{f_{Ti}}$ and $B_{f_{Uj}}$ stand for the bandwidth of the TCP flow f_{Ti} and the UDP flow f_{Uj}, respectively.

Without loss of generality, we now consider a typical scenario illustrated in Fig. 1. The UDP and TCP flows from the left area arrive at the right area *via* the FUDP facility and the outport is the bottleneck link. After processing the FUDP facility with the FUDP mechanism, the UDP flows arriving on the right side will become FUDP flows.

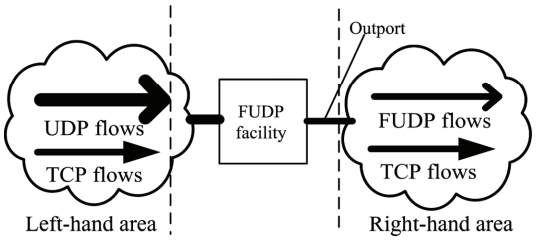

Figure 1. The runtime environment of the FUDP mechanism.

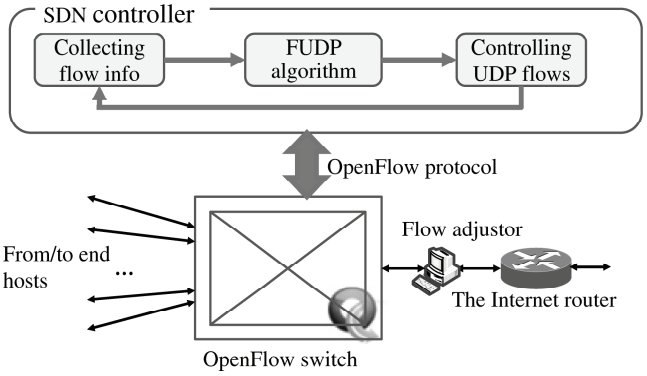

Figure 2. The closed-loop control model of the FUDP mechanism.

3.2 The Control Model for the FUDP Mechanism

We propose a closed-loop control model for the FUDP mechanism based on SDN as illustrated in Fig. 2. There are three main functional modules in the SDN controller: the flow information collection module (FIC), FUDP algorithm module (FUDPA), and UDP flow control module (UFC). These three functional modules execute periodically and form a closed-loop control process. First, the FIC collects the flow information from the flow entries in the SDN switch, where the information includes the number of the TCP/UDP flows and the bandwidth of all the flows. Second, the FUDPA calculates the deserved bandwidth for each UDP flow based on the above flow information. Third, the UFC sends the control information to the flow adjustor, and the flow adjustor controls the per-UDP-flow bandwidth concretely. The closed-loop control will continuously regulate the UDP flows if necessary and enable them to have FUDP properties.

4. Design of the FUDP Prototype System

We designed the FUDP prototype system based on SDN in this section according to the closed-loop control model for the FUDP mechanism.

4.1 Collecting Flow Information

The FIC collects the flow information through both active and passive methods. Specifically, once a new flow is transmitted through an SDN switch without any matched flow table, the SDN switch will forward the packet flow to the controller, which means the FIC can get each

flow's address information *via* a passive method. In the active method, the FIC adopts periodic polling to get the information for each flow based on the comprehensive information query system in OpenFlow [25] technology, with which the FIC can easily get the current transmission rate of each flow.

4.2 FUDP Algorithms

There are three main procedures in FUDPA to calculate the bandwidth for UDP flows for each CFS: (i) predicting the available bandwidth of the bottleneck link; (ii) establishing fairness and deciding whether to reallocate the bandwidth for each UDP flow; and (iii) calculating the bandwidth for each UDP flow.

4.2.1 Available Bandwidth Prediction

Assume that the current bandwidth sample usage for the TCP flow f_{Ti} asked for by the controller is B_{fTi}; then the total bandwidth of the TCP flows will be $W(f_T) = \sum_{i=1}^{n} B_{f_{Ti}}$. The total estimated bandwidth of the TCP flows f_T in the next period will be calculated iteratively according to

$$W_{k+1}(f_T) = (1-\alpha) \times W_k(f_T) + \alpha \times W(f_T) \qquad (1)$$

where $W_k(f_T)$ represents the total predicted bandwidth of the TCP flows in the previous period, and α is a scaling factor limited by $0 < \alpha < 1$. For a new flow, we choose $W_0(f_T) = W(f_T)$. In addition, the deviation in the TCP flows' total bandwidth can be computed according to

$$D_{k+1}(f_T) = (1-\beta) \times D_k(f_T) + \beta \times |W(f_T) - W_k(f_T)| \qquad (2)$$

where $D_k(f_T)$ represents the deviation in the TCP flows' total bandwidth in the previous period, and β is the scaling factor limited by $0 < \beta < 1$.

4.2.2 Network Fairness Estimation

We estimate the network fairness by comparing the average transmission bandwidth of TCP flows and UDP flows in a CFS. Specifically, we use $W_a(f_U)$ to represent the average bandwidth of the UDP flows, while we use $W_{au}(f_T)$ and $W_{ad}(f_T)$ to represent the upper bound and lower bound of the TCP flow average bandwidth. If $W_a(f_U) \in [W_{ad}(f_T), W_{au}(f_T)]$, we consider that all flows are in a state of fairness; otherwise, we have to adjust the bandwidth of the UDP flows to realize fairness, where $W_{au}(f_T) = \frac{1}{n}[W_{k+1}(f_T) + D_{k+1}(f_T)]$ and $W_{ad}(f_T) = \frac{1}{n}[W_{k+1}(f_T) - D_{k+1}(f_T)]$.

4.2.3 Calculating the Adjustment Parameters for UDP Flows

For the different needs of UDP applications, FUDPA uses a max–min fair [26] algorithm to allocate the bandwidth for UDP flows when $W_a(f_U) \notin [W_{ad}(f_T), W_{au}(f_T)]$ in a given CFS. First, the FUDPA calculates the available bandwidth of the CFS and then allocates the available bandwidth for each UDP flow according to the max–min fair algorithm. The available bandwidth of the bottleneck link is according to

$$BW = W_{k+1}(f_T) + \sum_{j=1}^{m} W(f_{Uj}) \qquad (3)$$

For TCP flows, their predicted bandwidths are used; for UDP flows, their distributed bandwidths are used. More details on bandwidth allocation are shown in Algorithm 1.

Algorithm 1: Bandwidth Distribution in FUDP

Input:
BW; /* the total available bandwidth of a CFS*/
$F_{s \to d} \leftarrow CFS(s \to d)$ /*each flow with its bandwidth demand B_d in the CFS*/
Output: B_f/*per-flow distributed bandwidth*/

```
1   Increasingly sort all the flows in F_{s→d} in terms of
    their B_d;
2   F* ← F_{s→d}
3   y = number(F*) /*the number of flow in the CFS*/
4   for UDP_flow f in F* do
5       B_p = min(B_d, BW/y)
6       B_f ← B_p
7       BW ← BW − B_f
8       F* ← F*\f
9       y ← y − 1
10  end for
11  return B_f
```

4.3 Controlling UDP Flows

After calculating the bandwidth for each UDP flow according to Algorithm 1, we implement a per-flow control scheme using the Linux traffic control (Linux TC) tool [27]. As shown in Fig. 2, the control unit of the FUDP mechanism is composed of an OpenFlow switch and a flow adjustor. A PC with a multiport Ethernet card installed Linux operating system is used as the flow adjustor, which executes per-flow bandwidth adjustments for UDP flows and forwards packets. A daemon program runs in the flow adjustor to monitor the control information messages from the controller. Specifically, after the FUDPA has calculated the proper transmission bandwidth for each UDP flow, the UFC encapsulates the adjustment parameters into a Packet-Out packet and sends the Packet-Out packet toward the port connected to the flow adjustor. When the flow adjustor obtains the information message, it parses the adjustment parameters and executes a series of TC commands to achieve a per-flow bandwidth adjustment for each UDP flow.

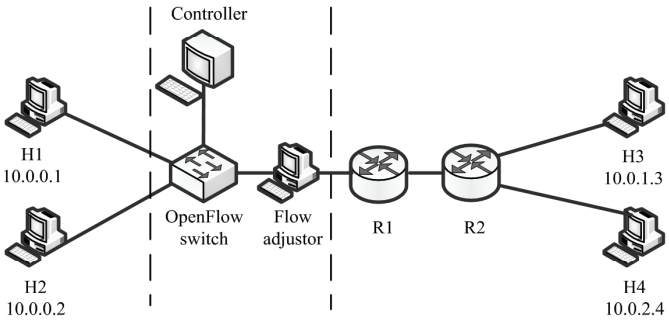

Figure 3. Experimental environment.

5. Evaluation

5.1 Methodology

5.1.1 Experimental Topology

The topology of our test bed is shown in Fig. 3, where the bandwidth of the R2-H4 link is 100 Mbps, while the other links are 1,000 Mbps. In the test bed, both R1 and R2 are Huawei AR2200 Series routers running on the OSPF protocol. A commercial Pica8 P-3297 OpenFlow switch acts as the OpenFlow switch, while six Lenovo PCs are used as the controller, flow adjustor and H1–H4 end hosts, where each Lenovo PC has 2-core Intel Core i5-3470 3.2GHz CPUs, 4GB of RAM and one 1TB 7200RPM HDD and runs the Ubuntu 12.04 LTS operating system. POX-eel [28] is employed as the controller, and both the controller and the OpenFlow switch comply with the OpenFlow1.0.0 standard [25].

5.1.2 Workloads

The Iperf traffic generator [29] is used to generate four long TCP flows and two long UDP flows as the workload in our experiments. The workloads are listed in Table 1. The sending order for the flows is according to the flow number, and there is a 50 s lag between the flows.

Table 1
Workload Information

No.	Protocol	Flow information
1	TCP	H1 → H4, lasting 400 s
2	TCP	H2 → H3, lasting 350 s
3	UDP	H1 → H4, lasting 300 s, sending bandwidth: 500 Mbps
4	UDP	H2 → H3, lasting 250 s, sending bandwidth: 400 Mbps
5	TCP	H2 → H3, lasting 200 s
6	TCP	H2 → H4, lasting 150 s

5.1.3 Method Comparison

To evaluate the performance of the FUDP mechanism effectively, we set up the IP and WFQ for two typical mechanisms in the same testing environment, where a Layer 2 switch replaced the OpenFlow switch and the flow adjustor was removed. Other setting details are as follows.

IP mechanism: Standard TCP-CUBIC is used as the baseline for our evaluation. The initial window is set to 85.3 KB, that is, the default value of the Ubuntu 12.04 LTS operating system on the end systems, while the routers adopt FIFO scheduling and drop-tail queuing.

WFQ mechanism: The end systems use the same configuration as those in the IP mechanism, but the routers adopt WFQ scheduling and drop-tail queuing. Meanwhile, two queues are set in each router's port and named q0 and q7. The TCP flows are sent into q7 while the UDP flows into q0. The weights of q0 and q7 are set to be 0.4 and 0.6, respectively.

FUDP mechanism: The design is described in Section 4 with $\alpha = 0.875$ and $\beta = 0.25$ settings and the adjustment period of the controller $t = 5$s. The configurations of the end systems and routers are the same as those in the IP mechanism.

We use per-flow throughput, a Fairness-index [30] and Power [31] as the three metrics to evaluate the performance of the FUDP mechanism. The Fairness index is defined as follows [30]: Given a set of flow bandwidths $\{w_1, w_2, \ldots, w_n\}$, their Fairness index is calculated by

$$F(w_1, w_2, \ldots, w_n) = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{n * \sum_{i=1}^{n} w_i^2} \tag{4}$$

while the Power is defined as

$$\text{Power} = \frac{\text{Throughput}}{\overline{RTT}} \tag{5}$$

where Throughput is the network throughput and $\overline{RTT}$ is the average round-trip time of the network.

5.2 Experimental Results and Discussion

In the test environment, the FUDP mechanism first assigns flows into two CFSs named Set1 and Set2 based on Concept 1. In our test environment, Set1 includes H1 → H4, H2 → H4 two TCP flows and H1 → H4 UDP flow, while Set2 includes three flows transmitting to host H3. Figures 4–6 show the per-flow throughput, Fairness index, and Power in the different mechanisms.

5.2.1 Throughput

Figure 4 illustrates the real-time per-flow throughput of the three different mechanisms. Due to the lack of a control mechanism for UDP flows in the IP mechanism, the UDP flows grab as much bandwidth as they can demand, while the TCP flows have to share the remaining bandwidth, and in particular, for H1 → H4 and H2 → H4 TCP flows, they cannot transmit any packet at all. As shown in Fig. 4(b),

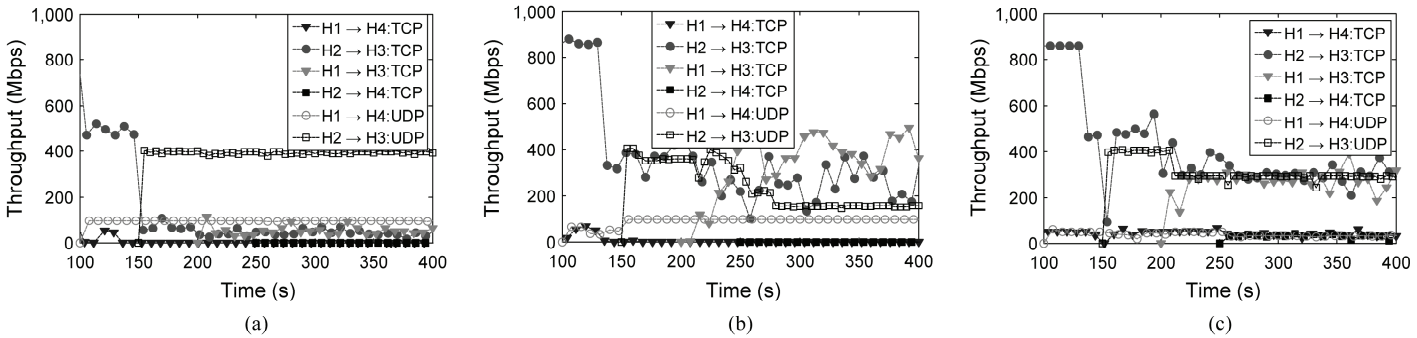

Figure 4. Per-flow throughput in different mechanisms: (a) IP, (b) WFQ and (c) FUDP.

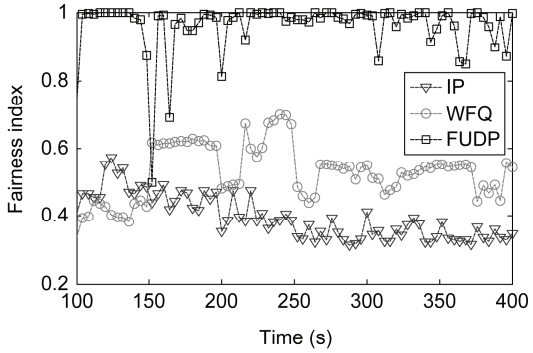

Figure 5. The Fairness condition for the different mechanisms.

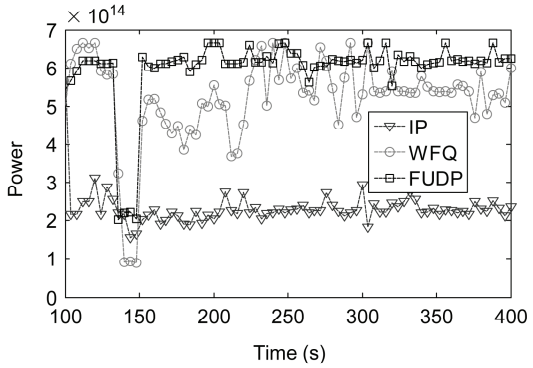

Figure 6. The Power condition for the different mechanisms.

although WFQ can restrict the total bandwidth of UDP flows to 400 Mbps as predefined, H1 → H4 and H2 → H4 TCP flows still suffer from starvation. However, the FUDP mechanism puts the flows into different CFSs, and for each CFS, the FUDP mechanism allocates bandwidth for each UDP flow based on the current workload in a closed-loop control manner, and each flow in the CFS can get a similar share of the bandwidth as shown in Fig. 4(c). In addition, the FUDP control algorithm has a good convergence, which fulfils the bandwidth adjustment for UDP flows in one adjustment period. At the same time, the FUDP mechanism demonstrates excellent stability.

5.2.2 Fairness

Figure 5 illustrates the real-time network Fairness index of the three different mechanisms. Due to the lack of means to control UDP flows, UDP flows occupy more bandwidth in the IP mechanism, and the Fairness index is merely 0.36 when the network is in the equilibrium state. In the WFQ mechanism, the Fairness index has been improved as the overall bandwidth of UDP flows is restricted to within a scheduled scope. However, the Fairness index is just 0.6, which is still not high because some TCP flows suffer from starvation. On the other hand, the overall Fairness index of the FUDP mechanism is up to 0.9 when the network lies in a stable state. The main reason is that the FUDP mechanism can adjust the bandwidths of the UDP flows according to the current network workload and control the UDP flows in the closed-loop manner.

5.2.3 Power

Figure 6 shows the real-time network power for the three different mechanisms. Both the IP and WFQ mechanisms waste bandwidth and suffer long delays due to network congestion; so, there are still large gaps in the Power compared with the FUDP mechanism. Specifically, the issue of bandwidth waste is more serious in the IP mechanism due to the lack of a UDP control method, and the Power is about 30% of this in the FUDP mechanism. The Power in WFQ fluctuates markedly, and the overall level is not high, which is mainly because the TCP flows always suffer from congestion. However, the FUDP mechanism can restrict and adjust the bandwidth of each UDP flow according to the current workload, which avoids waste of network bandwidth and reduces the congestion at the bottleneck link, thereby achieving the highest Power for the entire network.

6. Trace-driven Experiment

We tested the performance of the FUDP mechanism using a real Internet traffic trace from China Education and Research Network (CERNET) in this section. The studies in [3] show that the ratio of UDP traffic has significantly increased since 2005, and the traffic has gradually increased 20%–50% in CERNET. In particular, many UDP flows use too large a share of the bandwidth, thus making the Internet suffer from a problem of serious unfairness regarding bandwidth allocation. Table 2 presents the statistics on traffic traces from a provincial boundary link

Table 2
Traffic Trace Statistics from CERNET

Protocol	Packet	Byte	Long flow
TCP	8.9×10^7 (58.6%)	9.6×10^{10} (74.4%)	4,360
UDP	6.3×10^7 (41.4%)	3.3×10^{10} (25.6%)	1,350

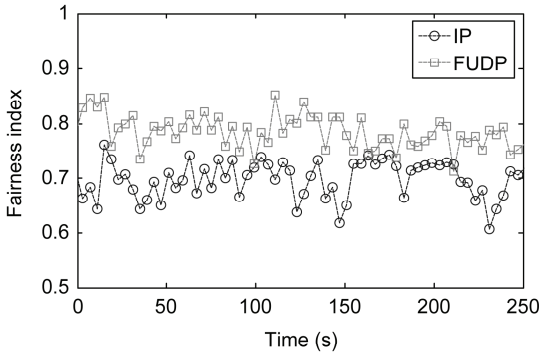

Figure 7. Fairness of the two different mechanisms.

in East China in CERNET on November 8, 2014, where a long flow is defined as a duration greater than 10 s and the average transmitting bandwidth is greater than 1 Mbps.

We simply control the long flows in the traces with the FUDP mechanism, and we evaluate the performance of the FUDP mechanism by comparing it with the IP mechanism using the metric of the Fairness index. The experimental results are shown in Fig. 7.

Overall, the FUDP mechanism improves the Fairness by 14.3% compared with the IP mechanism. Specifically, we find that about 10% of the UDP flows occupy more bandwidth than an average one in the IP mechanism, but the FUDP mechanism can control the flows' transmitting bandwidth according to real workloads and transforms them into FUDP flows, which contribute to a high level of network fairness for the whole network. Moreover, tens to hundreds of new flows come into the network simultaneously in the trace-driven experiment, but the FUDP mechanism can still efficiently control the transmission bandwidth of each UDP flow, which means that the FUDP mechanism has excellent viability.

7. Conclusion

There are serious and difficult problems in the Internet, such as the lack of fair bandwidth allocation between TCP and UDP flows and low transmission efficiency in IP networks, all of which has hindered the development of the Internet for some time. In this paper, a novel FUDP mechanism was proposed to give UDP flows quasi-TCP-Friendly properties, as a means of controlling UDP flows. The FUDP facility is a closed-loop control system based on SDN, which collects information on all the TCP/UDP flows, and computes and allocates the bandwidth of each UDP flow based on the current network workload. Both the FUDP prototype system and the trace-driven experimental results show that the FUDP mechanism can effectively control UDP traffic without modifying TCP/UDP and routers, and this improves the network throughput and avoids network congestion. Future studies will focus on how the FUDP mechanism can work in more complicated network environments and deploy the FUDP facilities in a real network.

Acknowledgement

This work is supported by the State Key Development Program for Basic Research of China (Grant No. 2012CB315806), the National Natural Science Foundation of China (Grant Nos. 61103225 and 61379149), the Jiangsu Province Natural Science Foundation of China (Grant No. BK20140070), and the Jiangsu Future Networks Innovation Institute Prospective Research Project on Future Networks (Grant No. BY2013095-1-06).

References

[1] S. Floyd and K. Fall, Promoting the use of end-to-end congestion control in the Internet, *IEEE/ACM Transactions on Networking (ToN)*, *7*(4), 1999, 458–472.

[2] D.J. Lee, B.E. Carpenter, and N. Brownlee, Observations of UDP to TCP ratio and port numbers. Internet Monitoring and Protection (ICIMP), *2010 Fifth International Conf. on IEEE*, Barcelona, Spain, 2010, 99–104.

[3] CERNET, http://iptas.edu.cn, 2014 [EB/OL].

[4] N. McKeown, Software-defined networking, *INFOCOM Keynote Talk*, *17*(2), 2009, 30–32.

[5] B. Liu, M. Chen, B. Xu, H. Hu, C. Hu, Q.Y. Zuo, and C.Y. Xing, An OpenFlow-based performance-oriented multipath forwarding scheme in datacenters, *Frontiers of Information Technology & Electronic Engineering*, *17*(7), 2016, 647–660.

[6] S. Jain, A. Kumar, S. Mandal, *et al.*, B4: Experience with a globally-deployed software defined WAN, *ACM SIGCOMM Computer Communication Review*, *43*(4), 2013, 3–14.

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

[8] R. Adams, Active queue management: A survey, *IEEE Communications Surveys & Tutorials*, *15*(3), 2013, 1425–1476.

[9] A. Kuzmanovic, The power of explicit congestion notification, *ACM SIGCOMM Computer Communication Review*, *35*(4), 2005, 61–72.

[10] N. Dukkipati, G. Gibb, N. McKeown, *et al.*, Building a RCP (rate control protocol) test network, *Hot Interconnects*, Stanford, CA, USA, 2007, 15.

[11] L.L.H. Andrew, S.H. Low, and B.P. Wydrowski, Understanding XCP: Equilibrium and fairness, *IEEE/ACM Transactions on Networking (TON)*, *17*(6), 2009, 1697–1710.

[12] S. Floyd, E. Kohler, and J. Padhye, Profile for datagram congestion control protocol (DCCP) congestion control ID 3: TCP-friendly rate control (TFRC), 2006.

[13] D.X. Wei, C. Jin, S.H. Low, *et al.*, FAST TCP: Motivation, architecture, algorithms, performance, *IEEE/ACM Transactions on Networking*, *14*(6), 2006, 1246–1259.

[14] T. Tielert, D. Jiang, Q. Chen, *et al.*, Design methodology and evaluation of rate adaptation based congestion control for vehicle safety communications, *Vehicular Networking Conf. (VNC), 2011 IEEE*, IEEE, Amsterdam, Netherlands, 2011, 116–123.

[15] P.X. Liu, M.Q.H. Meng, P.R. Liu, *et al.*, An end-to-end transmission architecture for the remote control of robots over IP networks, *IEEE/ASME Transactions on Mechatronics*, *10*(5), 2005, 560–570.

[16] L. Budzisz, R. Stanojevic, R. Shorten, *et al.*, A strategy for fair coexistence of loss and delay-based congestion control algorithms, *IEEE Communications Letters*, 2009, *13*(7), 555–557.

[17] I. Rhee and L. Xu, Limitations of equation-based congestion control, *ACM SIGCOMM Computer Communication Review. ACM*, *35*(4), 2005, 49–60.

[18] I. Rhee and L. Xu, Limitations of equation-based congestion control, *IEEE/ACM Transactions on Networking (TON)*, *15*(4), 2007, 852–865.

[19] D.X. Wei, C. Jin, S.H. Low, *et al.*, FAST TCP: Motivation, architecture, algorithms, performance, *IEEE/ACM Transactions on Networking (ToN)*, *14*(6), 2006, 1246–1259.

[20] S. Ha, I. Rhee, and L. Xu, CUBIC: A new TCP-friendly high-speed TCP variant, *ACM SIGOPS Operating Systems Review*, 42(5), 2008, 64–74.

[21] S. Liu, T. Başar, and R. Srikant, TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks, *Performance Evaluation*, *65*(6), 2008, 417–440.

[22] M. Dong, Q. Li, D. Zarchy, *et al.*, PCC: Re-architecting congestion control for consistent high performance, *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, Oakland, CA, 2015, 395–408.

[23] B. Claise, Cisco systems NetFlow services export version 9, 2004.

[24] M.S. Kim, T. Kim, Y.J. Shin, *et al.*, A wavelet-based approach to detect shared congestion, *IEEE/ACM Transactions on Networking*, *16*(4), 2008, 763–776.

[25] OpenFlow switch specification, version 1.0.0. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

[26] B. Radunović and J.Y.L. Boudec, A unified framework for max-min and min-max fairness with applications, *IEEE/ACM Transactions on Networking (TON)*, *15*(5), 2007, 1073–1083.

[27] M.S. Seddiki, M. Shahbaz, S. Donovan, *et al.*, FlowQoS: QoS for the rest of us, *Proc. Third Workshop on Hot Topics in Software Defined Networking*, ACM, 2014, 207–208.

[28] Pox-eel. [Online]. Available: https://github.com/noxrepo/pox/

[29] Iperf. [Online]. Available: https://iperf.fr/

[30] A.B. Sediq, R.H. Gohary, R. Schoenen, *et al.*, Optimal tradeoff between sum-rate efficiency and Jain's fairness index in resource allocation, *IEEE Transactions on Wireless Communications*, *12*(7), 2013, 3496–3509.

[31] L.L. Peterson and B.S. Davie, *Computer networks: A systems approach*, (Elsevier, 2007).

Biographies

Hui Hu is currently a research assistant at the Information Center at Northwestern Polytechnical University. Her research interests include network measurements and monitoring, performance evaluation, and software-defined networking.

Bo Liu is currently a Ph.D. student at PLA University of Science and Technology. His research interests include network measurements and monitoring, performance evaluation, software-defined networking, and data centre networks.

Chao Hu is currently an assistant professor at the College of Command Information Systems at PLA University of Science and Technology, Nanjing, China. His research interests include peer-to-peer multimedia communications, network modelling, and software-defined networking.

Ming Chen received his Ph.D. from the Nanjing Institute of Communication Engineering, China, in 1991. He is currently a professor at the College of Command Information Systems at PLA University of Science and Technology, Nanjing, China. He held the position of visiting professor at Columbia University in 1999. He has published extensively on network architecture, network measurements, and monitoring, performance evaluation, and distributed computing.

Guang Cheng, born in 1973, is a professor at the Computer Science and Engineering School at Southeast University and a doctoral advisor for the Doctor of Engineering program. Currently, he is the director of the Key Laboratory of Computer Network and Information Integration at Southeast University, Nanjing, China. His scientific research has focused on theoretical exploration and applied research in SDN network measurement and management, high-speed Internet and wireless on-board network measurements, network traffic measurements and analysis of big data, botnets and APT attack detection, and other areas.

Changyou Xing received a Ph.D. from PLA University of Science and Technology, Nanjing, China, in 2009. He is currently an assistant professor in the College of Command Information Systems at PLA University of Science and Technology, Nanjing, China. His research interests include peer-to-peer multimedia communications and network modelling.