Dynamic Priority-adjustment for Real-time Flows in Software-defined Networks

Namwon An*, Taejin Ha*, Kyung-Joon Park†, and Hyuk Lim*
* Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea
Email: {nwan, tjha, hlim}@gist.ac.kr
† Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, Republic of Korea
Email: kjp@dgist.ac.kr

***Abstract*—Network traffic can be divided into delay-sensitive and delay-tolerant traffic. Each delay-sensitive real-time flow has a different end-to-end delay deadline, and has to be delivered by that deadline. In general, real-time flows are to be processed prior to delay-tolerant flows, and each real-time flow may has the relative priority depending on its relative importance of real-time flows. A software-defined networking (SDN) makes it possible to monitor and control flows dynamically in the centralized view of the overall network. In this paper, we propose an priority-adjustment algorithm to guarantee the different delay deadlines of real-time flows in SDN-based networks. Compared to other algorithms, simulation results indicate that the number of real-time flows that satisfy their deadlines is highest in the proposed algorithm.**

***Index Terms*—Real-time, software-defined networking, priority adjustment**

I. Introduction

As computer networks have grown rapidly over recent decades, they have become one of the most important parts in modern society. From huge data centers and cloud systems to small portable devices such as a cell-phone, wearable devices, and tablet PCs, almost every device is now connected to the Internet. Network traffic is divided into real-time and non-real-time traffic. Real-time traffic has an end-to-end delay deadline, and it has to be delivered from source to destination by that deadline. There has been research into guaranteeing such delay deadlines for real-time traffic, but it remains a challenging problem.

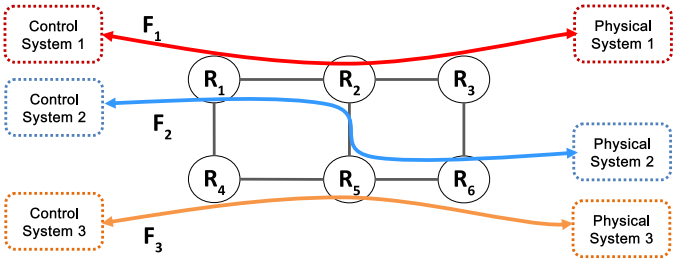
For the Internet, there has been research into guaranteeing a variety of QoS requirements of users, such as IntServ and DiffServ frameworks [1]. With IntServ, network resources for each real-time flow are reserved in advance at each router the flow passes [2]. However, this requires too many resource-reservation connections for each flow. In addition, as the number of flows increases, the overhead of reserving and returning resources increases and the scalability becomes very poor [3]. With DiffServ, real-time flows are serviced at each router according to their DS field priorities, but DiffServ still provides the best-effort service at each router and does not guarantee the end-to-end QoS [4].

Software-defined networking (SDN) is a new networking paradigm to separate the control and data planes in a network [5]. SDN is composed of infrastructure and control layers. In the infrastructure layer, SDN switches are connected over a data plane for user-data delivery, in which each SDN switch is a simple network device for forwarding network packets according to its forwarding table [6]. In the control layer, an SDN control software is connected with each SDN switch over the control plane, and it monitors each SDN switch and manages its forwarding table by using data-plane interfaces, such as ForCES and OpenFlow [7] [8]. Such centralized management and control of SDN switches provides a new way to design network functionality [3]. In traditional networks, the functionality for network management has been designed in the distributed view of each network device. However, with SDN, the functionality can be designed in the centralized view of the overall network and implemented based on the SDN control software. It then becomes possible to handle issues that have not been solvable in traditional networks [9].

In this paper, we propose two dynamic priority-adjustment schemes to satisfy delay deadlines for real-time applications on SDNs. Because an SDN enables the precise configuration and control of each SDN switch in a centralized manner, it is possible to guarantee the deadline constraints by dynamically adjusting the priority of the real-time traffic flows on the switches that the flows pass through. The first priority scheme assigns a unique flow priority to each flow. On the other hand, in the second scheme, each flow can have a different priority on the switches that the flow passes through. Through simulations, we evaluate the performance of our two priority-adjustment schemes against other schemes.

II. Motivation

We consider an example of a real-time network in Figure 1(a), which consists of real-time control systems, and physical systems that are controlled by each real-time control system. There are three real-time flows between the real-time control and the physical systems. We assume that the paths of the real-time flows are determined in advance and do not change later. Each real-time flow has its own end-to-end delay deadline, and its priority is determined by priority-adjustment methods. Figure 1(b) shows an NS-2 simulation results. The earliest deadline first (EDF)-based priority-adjustment method makes the packet that is closest to its deadline relayed first at each switch. Under a per-flow priority-adjustment method, a flow-specific priority is given to the flows in order to guarantee



(a) Real-time network

	No priority	EDF-based	Per-flow priority adjustment		Per-router priority adjustment		Deadline
	Delay	Delay	Priority	Delay	Priority	Delay	
F_1	43 ms	41 ms	2	57 ms	2 @ R_1, R_2, R_3	43 ms	50 ms
F_2	69 ms	67 ms	1	43 ms	2 @ R_1, R_2 1 @ R_5, R_6	56 ms	60 ms
F_3	41 ms	45 ms	2	57 ms	2 @ R_4, R_5, R_6	54 ms	60 ms

(b) End-to-end delay and deadline

Fig. 1. Example of real-time network with an end-to-end delay constraint.

their delay deadlines. Under a per-router priority adjustment method, each flow can have a different priority at each router.

The deadlines of the flows are given as 50 ms, 60 ms, and 60 ms, respectively. The path of F_2 is the longest, and its delay is 69 ms without a priority-adjustment, which fails to satisfy its deadline of 60 ms. In the EDF-based priority-adjustment method, F_1 has a shorter remaining time to the deadline than F_2 at R_1 and R_2. F_1 is therefore processed with a higher priority, but this does not mean that all packets of F_1 are always transmitted prior to the packets of F_2. If a F_2 packet stays at a router for a long time, it gets closer to its deadline and has a high priority. At F_5 and F_6, F_2 is processed with a high priority, but F_2 eventually fails to satisfy its deadline of 60 ms.

In the per-flow priority-adjustment method, a higher priority is given to F_2 to guarantee its deadline, and the delay of F_2 becomes 43 ms, which is very low and satisfies its deadline of 60 ms. The delays of F_1 and F_3 are each increased to 57 ms, but again F_1 fails to satisfy its deadline of 50 ms. In the per-router priority-adjustment method, a higher priority is given to F_2 only at R_5 and R_6. The F_3 delay becomes 54 ms, which still satisfies its deadline of 60 ms. The F_2 delay becomes 56 ms, which is greater than the 43 ms in the per-flow priority-adjustment method, but which satisfies its deadline of 50 ms. As a result, all the deadlines are satisfied.

Whether or not the delay deadlines of real-time flows are guaranteed is determined by how their priorities are assigned and adjusted. In this paper, we propose priority-adjustment schemes that assign and adjust the priorities of real-time flows dynamically in SDN-based real-time networks to guarantee their various delay deadlines.

III. Related Work

There has been research to guarantee the delay deadline or end-to-end QoS of real-time flows. Guck and Kellerer [3] proposed a deterministic network model in which the delay of each priority queue is calculated using network calculus [10]. It finds possible paths of each flow by a simple greedy algorithm, and the authors compared these paths with the optimal paths obtained by a mixed-integer program. Wu *et al.* [11] proposed a resource-allocation scheme based on the central resource database of the SDN controller. When a flow is admitted, the SDN controller finds a routing path that uses network resources efficiently, based on the delay bound of the priority queues and the transmission delay of each link. In order to calculate delay bounds for each priority queue, a rate-controlled static-priority (RCSP) queueing service discipline is adapted in each router. Tomovic *et al.* [12] proposed a new SDN framework and design for QoS provisioning, in which a bandwidth of each link is reserved for guaranteeing the deadline of each flow. Based on the available bandwidths of links, the shortest path to minimize the delay is found by using the Dijkstra algorithm. In [13], Akella and Kaiqi considered the network of a cloud carrier in which the requests of cloud users are processed. A cloud provider manages the network, and Open vSwitch is running at each switch. In order to guarantee the QoS requirements of multiple cloud users, a bandwidth that is sufficient to guarantee QoS requirement is assigned at each switch using the QoS policy of Open vSwitch. Ongaro *et al.* [14] provides a mathematical model to find the shortest path for each flow with constraints. It is based on Integer Linear Programming (ILP), and it routes all flows so as to minimize the sum of delays or packet drop rate based on the available bandwidth of each link. It assumes that a program, which is implemented according to the QoS architecture, monitors the network delay or packet drop rates in each link continuously.

Most of the existing work focuses on how to route real-time flows to guarantee delay deadlines based on the available bandwidth or delay of each link. However, such routing strategies require a great deal of computation to find a path to guarantee the deadlines of real-time flows, and the scalability is very poor. This paper tackles the question of how to adjust the priorities of flows to guarantee their deadlines.

IV. Proposed Algorithm

A. System Model

Let us consider an SDN-based real-time networks in Figure 2, which is composed of an SDN controller, SDN-enabled switches, real-time control systems, and physical systems. Each physical system is controlled by its own control system in real time, and the real-time flow between them has an end-to-end delay deadline, which differs according to the characteristics of the physical systems. An SDN-enabled switch divides network traffic into real-time and non-real-time traffic, and prioritizes the real-time traffic by priority queueing. Each real-time flow has different priorities at each switch along its path.

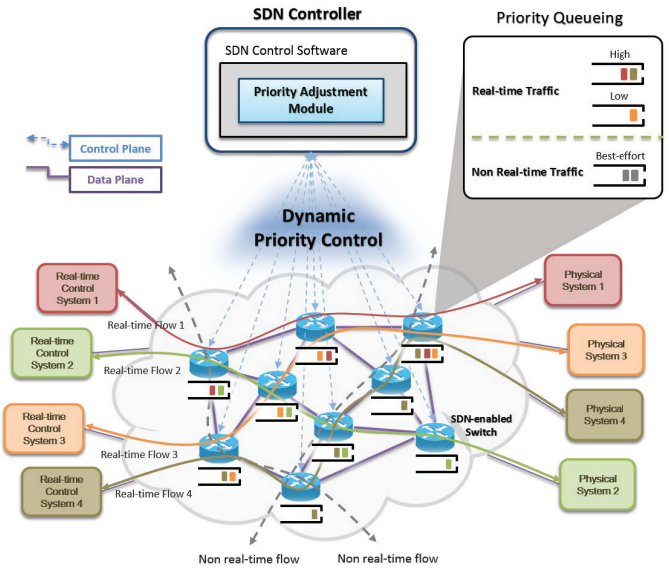


Fig. 2. Architecture of an SDN-based real-time networks.

At each switch, packets are enqueued into a queue according to its priority. Among queues, strict priority queueing (SPQ) is applied, which implies that packets of a low-priority queue are never processed before a high-priority queue becomes empty [15]. We assume that either of a real-time control system or a physical system reports its deadline to the SDN controller before establishing the connection between them, and also reports the average end-to-end delay periodically according to the time interval that the SDN controller determined.

Suppose that there are m SDN-enabled switches denoted by $S = \{s_1, s_2, ..., s_m\}$, n real-time flows denoted by $F = \{f_1, f_2, ..., f_n\}$, and q priority queues at each switch, in which queue 1 has the highest priority and queue q has the lowest priority. Let the priority of flow $f \in F$ at switch $s \in S$ be $x_{f,s}$. At every switch $s \in S$, $x_{f,s}$ of flow f is denoted by a priority vector $X_f = [x_{f,1}, x_{f,2}, \ldots, x_{f,m}]^T$ in which, if flow f is passing switch s, $x_{f,s}$ has a priority value from 1 to q, and otherwise 0. The priority vectors X_f for every flow $f \in F$ are denoted by a priority matrix:

$$M = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{pmatrix}. \quad (1)$$

Let d_f denote the measured end-to-end delay of flow $f \in F$, and let κ_f denote the end-to-end deadline for flow f. Instead of κ_f, we use a pseudo deadline, namely $D_f = \alpha \times \kappa_f$ for $0 < \alpha \leq 1$, in order to compensate for the fluctuation of delays.

B. Priority-adjustment operations

One of basic operations for priority adjustment is to increase or decrease the priorities of flows dynamically to respond to such fluctuations and guarantee each flow's delay deadline. Let $\hat{d}_f = \frac{d_f}{D_f}$ be the normalized delay for each flow $f \in F$.

TABLE I
SYSTEM PARAMETERS.

Parameter	Description
m	Number of SDN-enabled switches
S	A set of SDN-enabled switches
n	Number of real-time flows
F	A set of real-time flows
q	Number of priority queues per switch
$x_{f,s}$	Priority of flow f at switch s
d_f	End-to-end delay of flow f
κ_f	Real end-to-end deadline of flow f
α	Similarity between real and pseud end-to-end deadlines
D_f	Pseudo end-to-end deadline of flow f, $D_f \leftarrow \alpha \times \kappa_f$

We use the normalized delay as a metric to select the best or worst flow. For the priority increment, we select a flow that has the highest normalized delay. For the priority decrement, conversely, we select a flow that has the lowest normalized delay.

We find a priority matrix M^* that minimizes the summation of the normalized end-to-end delays $\hat{d}_f$ for all $f \in F$, satisfying their end-to-end delay deadlines D_f:

$$M^* = \arg\min_{M} \sum_{f \in F} \hat{d}_f \quad (2)$$
$$\text{subject to } \hat{d}_f \leq 1 \text{ for } \forall f \in F.$$

When there are m switches, n real-time flows, and q priorities, the size of the search space for M is q^{mn}, which is too high to be computed in practice for a large network. Usually, the priority-adjustment algorithms are executed in SDN control software periodically, and the priorities of real-time flows are increased or decreased at each execution to respond to the delay variances. Therefore, it is urgently required that the complexity of the priority-adjustment algorithms should be kept as low as possible.

C. Priority-adjustment algorithms

We propose two greedy algorithms for the priority-adjustment: a) a per-flow priority-adjustment scheme; b) a per-router priority-adjustment scheme. In a nutshell, in order to guarantee the different delay deadlines of real-time flows, the priority-adjustment algorithms assign and adjust the priority of a real-time flow, based on its deadline and measured end-to-end delay.

1) Per-flow priority-adjustment scheme: The per-flow priority-adjustment scheme assigns a unique flow priority to each flow, as shown in Figure 3. In this case, the flow has the same priority at every switch, i.e., $x_{i,1} = \cdots = x_{i,m}$ for $\forall n$ in (1). This constraint is included in the optimization in (2). The per-flow priority-adjustment scheme is divided into two parts. First, it selects the worst of the flows that violate their deadlines, and then it increases the flow's priority to allow it to satisfy its deadline. In the decrement step, conversely, it selects the best of the flows that satisfy their deadlines, and decreases its priority.

Algorithm 1 shows the detailed procedure of the per-flow priority-adjustment algorithm, which consists of loops to

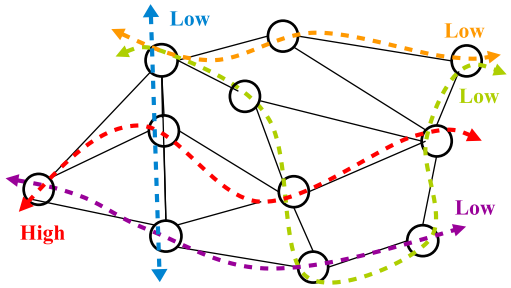


Fig. 3. Per-flow priority-adjustment scheme.

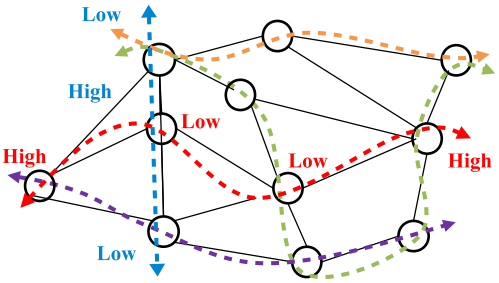


Fig. 4. Per-router priority-adjustment scheme.

Algorithm 1 Per-flow priority-adjustment algorithm.

$G \leftarrow$ all flows
Measure d_f for all $f \in G$
while $D_f \prec d_f$ for $\exists f \in G$ **do**
  $f_{max} \leftarrow \arg\max_{f \in G}(\frac{d_f}{D_f})$
  $p_{f_{max}} \leftarrow 1$
  Measure d_f for all $f \in G$
  for $k = 1 \rightarrow q - 1$ **do**
    if $d_f \preceq D_f$ for $\forall f \in G$ and $p_f = k$ **then**
      Break
    end if
    while $D_f \prec d_f$ for $\exists f \in G$ and $p_f = k$ **do**
      $f_{min} \leftarrow \arg\min_{f \in G \text{ and } p_f = k}(\frac{d_f}{D_f})$
      $p_{f_{min}} \leftarrow k + 1$
      Measure d_f for all $f \in G$
    end while
  end for
  $G \leftarrow G - \{f_{max}\}$
end while

increase and decrease priority. The first loop checks whether there is a flow that violates its deadline, and it repeats until every flow of G satisfies its deadline. We set the priority of flow $f \in F$ as p_f, which means that $p_f = x_{f,s}$ for all $s \in S$. Within the loop, a flow f_{max} with the highest normalized delay is selected from G, and the flow's priority $p_{f_{max}}$ is set to the highest priority of 1. Since the priority of flow f_{max} is changed to the highest priority of 1, the existing flows that have priority 1 may violate their deadlines. So, the for-loop checks whether there is a flow that violates its deadline at each priority level from 1 to $q - 1$. If the deadline of every flow is satisfied at a priority level, the for-loop finishes; otherwise it repeats to decreases the priority of a flow with the lowest normalized delay until every flow deadline is satisfied at priority level.

The overall process of this algorithm is similar to a lazy bubble sort, because it only adjusts priorities when there is a flow that violates its deadline. From the highest to the lowest priority level, flows are compared according to the normalized delay at each priority level, and their priorities are decreased.

2) Per-router priority-adjustment scheme: The per-router priority-adjustment scheme is the same as the per-flow priority-adjustment scheme except that different priorities are given at each switch for a selected flow, as shown in Figure 4. One of the switches that the selected flow passes through is selected for priority adjustment. For each selected switch, the priority of a selected flow is increased or decreased.

Algorithm 2 Per-router priority adjustment algorithm.

$G \leftarrow$ all flows
Measure d_f for all $f \in G$
while $D_f \prec d_f$ for $\exists f \in G$ **do**
  $f_{max} \leftarrow \arg\max_{f \in G}(\frac{d_f}{D_f})$
  $P \leftarrow \{s | s$ is each switch flow f_{max} passes$\}$
  $x_{f_{max},s} \leftarrow 1$ for all $s \in P$
  Measure d_f for all $f \in G$
  for $k = 1 \rightarrow q - 1$ **do**
    $P \leftarrow \{s | s$ is a switch that flow f_{max} passes$\}$
    if $d_f \preceq D_f$ for $\forall f \in G$ and $s \in P$ and $x_{f,s} = k$ **then**
      Break
    end if
    while $|P| \neq 0$ **do**
      $s_{target} \leftarrow$ choose a switch $s \in P$ randomly
      while $D_f \prec d_f$ for $\exists f \in G$ and $x_{f,s_{target}} = k$ **do**
        $f_{min} \leftarrow \arg\min_{f \in G \text{ and } x_{f,s_{target}} = k}(\frac{d_f}{D_f})$
        $x_{f_{min},s_{target}} \leftarrow k + 1$
        Measure d_f for all $f \in G$
      end while
      $P \leftarrow P - \{s_{target}\}$
    end while
  end for
  $G \leftarrow G - \{f_{max}\}$
end while

Algorithm 2 shows the detailed procedure of the per-router priority-adjustment algorithm. In this algorithm, different priorities for a flow are given at each switch. After a flow f with the highest normalized delay is selected from G, flow f's priority is set to the highest priority at each switch of flow f's path P. Unlike the per-flow priority-adjustment algorithm, the priority is adjusted only at switches of path P. At a priority level, if all flows satisfy their deadlines at each switch $s \in P$, the for-loop finishes.

The per-router adjustment algorithm adjusts the priority at each switch, and essentially it can do more fine-grained priority-adjustment than the per-flow priority adjustment algorithm to guarantee the delay deadlines of flows.

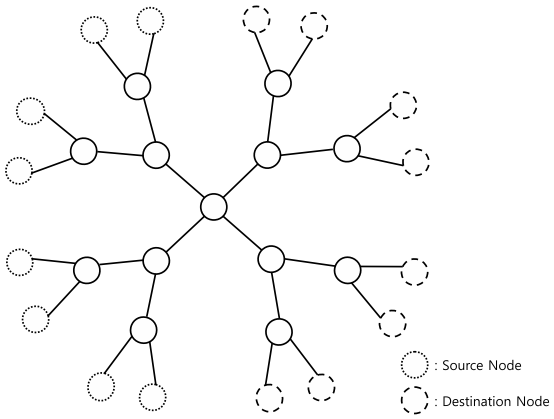


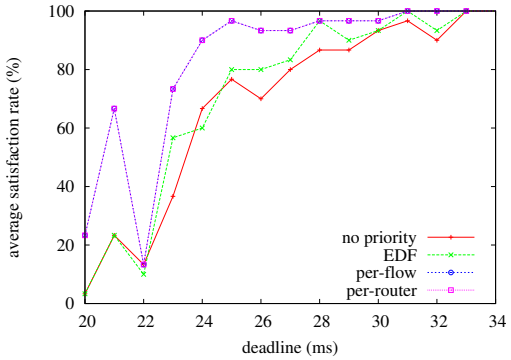
Fig. 5. Tree-type mesh network.

V. Simulation

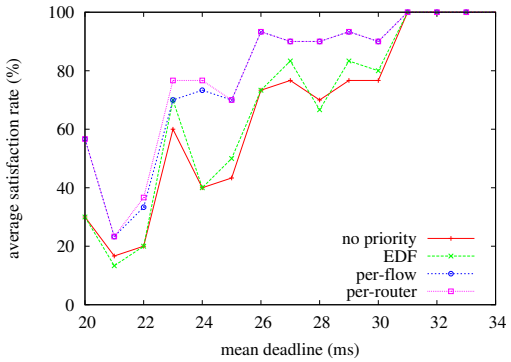
In order to evaluate the performance of our proposed algorithms, we conducted NS-2 simulations in which each output port of a node consists of 5 drop-tail queues. A class-based queueing (CBQ) discipline was used for the priority queueing. We assume that the maximum bandwidth utilization of each link does not go beyond 0.99, and that the propagation delay of each link is 1 us. We set α to 0.99 for all flows. We define and use the satisfaction rate as a performance index of our proposed algorithms, which is the percentage of flows that satisfy their delay deadlines in all flows. We compare the satisfaction rate of our proposed algorithms with the EDF-based priority-adjustment algorithm, which gives a higher priority to the packets that are closer to its deadline at the current time [16].

We conducted NS-2 simulations in the tree-type mesh network of Figure 5. When the number of flows is 10, 20 and 30, the source and destination of each flow are chosen randomly from the leaf nodes in Figure 5. In the simulations, we consider two cases in which every flow either has the same deadline or each flow has a random deadline for a given mean deadline.

Figure 6 shows the average satisfaction rates of the per-flow, per-router, and EDF-based priority-adjustment algorithms for 10 flows that have the same deadline or different deadlines with a Gaussian distribution with standard deviation of 0.001. We set the rate of each flow to 50. For each deadline, the algorithms were repeated three times, and the satisfaction rates were averaged. The per-flow and per-router algorithms were run three times continuously for a repetition. In Figure 6(a), there are deadline points for which the satisfaction rate of the EDF-based algorithm is lower than no priority adjustment. The per-flow and per-router algorithms are always higher than or equal to no priority adjustment and the EDF-based algorithm in terms of average satisfaction rate. In Figure 6(b), the average satisfaction rates of the per-flow and per-router algorithms exceed the EDF-based algorithm, and at several deadlines the per-router algorithm is higher than the per-flow algorithm because the former algorithm controls the priorities of flows



(a) Constant deadline



(b) Random deadline

Fig. 6. Average satisfaction rate of 10 flows.

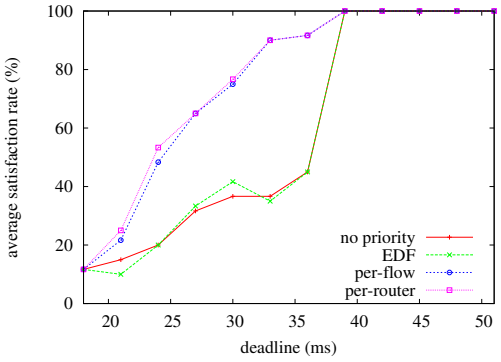
at each router in a more fine-grained way.

In Figure 7, we set the rate of each flow to 25, and the average satisfaction rates of the per-flow and per-router algorithms show a higher performance than the EDF-based algorithm. For both of equal and different deadlines, the per-router algorithm shows a higher performance than the per-flow algorithm at several deadlines. As the number of flows increases, the per-router algorithm has a higher number of cases for which the priorities of flows can be assigned and adjusted at each switch, and it shows a better performance than the per-flow algorithm.

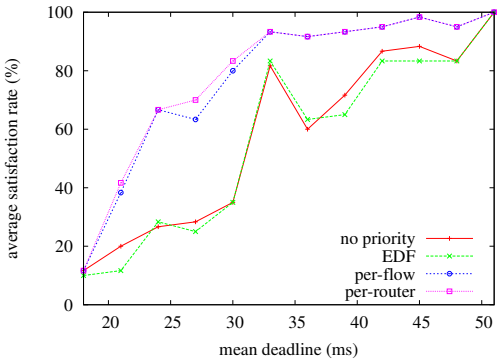
We set the rate of each flow to 20 in Figure 8. When compared to the cases of 10 and 20 flows, there is a higher difference of average satisfaction rates between our proposed algorithms and the EDF-based algorithm.

VI. Conclusion

In order to guarantee the different delay deadlines of real-time flows, we proposed two priority-adjustment algorithms: per-flow and per-router priority. In SDN-based networks, the algorithms run on the SDN controller periodically and adjust the priorities of real-time flows dynamically based on delay measurement. In the per-flow priority-adjustment algorithm, priorities are given to each flow. This algorithm changes the priority of the flow that has the highest normalized delay to the highest priority, and then updates the priorities of all the other flows from the highest to the lowest priority level like

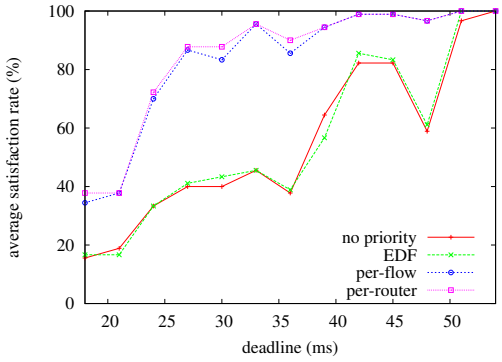


(a) Constant deadline

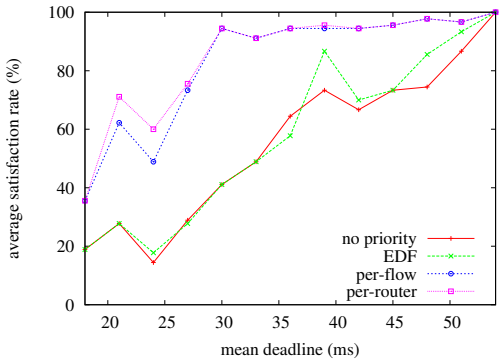


(b) Random deadline

Fig. 7. Average satisfaction rate of 20 flows.



(a) Constant deadline



(b) Random deadline

Fig. 8. Average satisfaction rate of 30 flows.

a bubble sort. In the per-router priority-adjustment algorithm, different priorities for a flow are given at each switch. We evaluated and compared these two algorithms with the EDF-based priority-adjustment algorithm in terms of satisfaction rate as the deadlines of flows increase. The results indicate that these algorithms show higher satisfaction rates than those of other algorithms.

REFERENCES

[1] W. Zhao, D. Olshefski, and H. G. schulzrinne, "Internet quality of service: An overview" *Columbia University Academic Commons*, 2000.

[2] R. Braden, D. Clark, and S. Shenker, "RFC1633: Integrated services in the Internet architecture: An overview", Jun. 1994.

[3] J.W. Guck and W. Kellerer, "Achieving end-to-end real-time quality of service with software defined networking", *IEEE Cloud Networking (CloudNet)*, pp. 70-76, Luxembourg, Oct. 2014.

[4] K. Nichols, S. Blake, F. Baker, and D. Black, "RFC2474: Definition of the differentiated services field (DS Field) in the IPv4 and IPv6 headers", 1998.

[5] Open Networking Foundation, "Software-defined networking: The new norm for networks", https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[6] B.A.A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of software-defined networking: Past, present, and future of programmable networks", *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, Feb. 2014.

[7] A. Doria, J. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "RFC5810: Forwarding and control element separation (ForCES) protocol specification", Mar. 2010.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks", *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, Apr. 2008.

[9] S. Sezer, S. Scott-Hayward, P.K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks", *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36-43, Jul. 2013.

[10] J.-Y. Le Boudec, and P. Thiran, "Network calculus: A theory of deterministic queueing systems for the internet", *Springer*, 2001.

[11] M. Wu, M. Zhao, and H. Yu, "Dynamic priority queue mapping for QoS routing in software defined networks", *United States Patent Application Publication*, Mar. 2015.

[12] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning", *Telecommunication Forum Telfor (TELFOR)*, pp. 111-114, Belgrade, Nov. 2014.

[13] A. V. Akella and X. Kaiqi, "Quality of service (QoS)-guaranteed network resource allocation via software defined networking (SDN)", *IEEE International Conference on Dependable, Autonomic and Secure Computing*, pp. 7-13, Dalian, Aug. 2014.

[14] F. Ongaro, E. Cerqueira, L. Foschini, A. Corradi, and M. Gerla, "Enhancing the quality level support for real-time multimedia applications in software-defined networks", *International Conference on Computing, Networking and Communications (ICNC)*, pp. 16-19, Garden Grove, Feb. 2015.

[15] Y. Qian, Z. Lu, and Q. Dou, "QoS Scheduling for NoCs: Strict priority queing versus weighted round robin", *IEEE International Conference on Computer Design (ICCD)*, pp. 52-59, Amsterdam, Oct. 2010.

[16] P. Pedreiras and L. Almeida, "EDF message scheduling on controller area network", *Computing and Control Engineering Journal*, vol. 13, no. 4, pp. 163-170, Aug. 2002.