Doctoral Dissertation

A Study on Resource Allocation Schemes in Edge Cloud Computing

Yusuke Ito

March, 2019

Graduate School of Environmental Engineering
The University of Kitakyushu

Preface

Cooperation between edge and cloud computing is expected as a novel network architecture for supporting people's daily lives and work. It provides delay-tolerant services, including e-mail and file sharing, provided by cloud servers as well as delay-sensitive services, including augmented reality and autonomous driving, provided by edge servers. When computing resources of edge servers are insufficient, these servers must cooperate with cloud servers to provide such services due to limited resources. This cooperation between edge and cloud computing enables users to enjoy various services.

In edge and cloud computing, users expect to receive a wide variety of services with different requirements in terms of transmission rates on various locations, so there are important issues regarding efficient service provision to meet user requirements independently of locations: (1) fair service provision in cloud computing, (2) service provision according to user requirements in edge computing, and (3) efficient transmission in edge and cloud computing.

To address these issues, approaches to provide fair and efficient sharing of network resources among flows have been reported; however, these issues have not yet been solved. One representative approach is using active queue management algorithms applied to intermediate routers. This approach can provide fair and efficient sharing of network resources among flows by preferentially discarding packets of a high rate flow, but it is difficult to apply it to edge and cloud computing where bottleneck links shared by all flows do not exist.

The main objective of this dissertation is proposing efficient schemes to allocate and use network resources among flows to solve issues (1)–(3) in edge and cloud computing and show the effectiveness of the proposed schemes through simulation evaluations.

In Chapter 1, I give the background, objective, and organization of this disser-

tation.

In Chapter 2, I give an overview of TCP to provide a better understanding of this dissertation and introduce related work.

In Chapters 3 and 4, I discuss an efficient method to fairly allocate bandwidth among users in cloud computing regarding issue (1). In Chapter 3, I propose a bandwidth allocation scheme based on collectable information to improve fairness in data center (DC) networks. I evaluated the characteristics of this scheme through simulation by focusing on the effect of allocation factor α used to adjust the transmission rate of each flow. The simulation results indicate that the proposed scheme achieves higher fairness and throughput performance than that using conventional TCP by setting α to slightly larger than 1. In Chapter 4, I discuss extending the proposed scheme to reallocate unutilized bandwidth caused by congestion outside DC networks to other competing flows to improve link utilization in these networks and evaluated the performance of the proposed scheme through simulation by focusing on how to set parameter values (e.g., notification interval time for flow information) in more detail. Simulation results indicate that the proposed scheme can improve fairness and link utilization in DC networks by setting appropriate parameter values.

In Chapters 5 and 6, I discuss an efficient method to allocate bandwidth according to user requirements in edge computing regarding issue (2). In Chapter 5, I propose a bandwidth allocation scheme based on collectable information to meet requirements of each flow in edge computing. The simulation results indicate that this proposed scheme can allocate transmission rates among flows according to their requirements independently of each flow's round-trip time as well as achieve high link utilization. However, this scheme may degrade link utilization in access networks if it reallocates an unutilized bandwidth to flows through paths with insufficient bandwidth. In Chapter 6, I therefore discuss extending the proposed scheme to allocate bandwidth based on residual bandwidth information to improve link utilization in edge computing. The simulation results indicate that this proposed scheme achieves higher fairness than conventional schemes as well as high link utilization in edge computing.

In Chapter 7, I discuss a method to provide efficient transmission in edge and cloud computing regarding issue (3). When users receive services from cloud

servers in edge and cloud computing, they cannot use bandwidth effectively due to the long distance between users and cloud servers. This is because TCP, which is commonly used for reliable communication, controls transmission rates on an end-to-end basis. It significantly degrades the service quality for such users. I therefore propose flow splitting and aggregation schemes to improve the throughput of each flow in edge and cloud computing. From the simulation evaluations, I indicate that these proposed schemes can improve throughput performance by flow aggregation and flow control functions.

In Chapter 8, I conclude this study and address future work. The integration of the proposed schemes can help provide high-quality services to users according to the requirements of each user independently of user location. I hope that this dissertation will be helpful for further study in this field.

Acknowledgements

I would like to acknowledge the support and encouragement received from a number of peoples for several years.

First of all, I wish to express my sincere appreciation to Associate Professor Hiroyuki Koga of the University of Kitakyushu. His constant encouragement, guidance through this research, invaluable discussions and advice have greatly helped in accomplishing the research. I also thank him for his careful reading of all papers on the research.

I would also like to express my gratitude to Associate Professor Katsuyoshi Iida of Hokkaido University for his valuable comments, time and help in completing the research.

I wish to thank Professor Takeshi Ikenaga of the Kyushu Institute of Technology for valuable comments, time and help in completing the research. His steady support has greatly helped my study.

I am very grateful to Professor Masayuki Sato, Professor Takashi Sato, Professor Seung-Woo Lee, and Associate Professor Yasushi Yamazaki of the University of Kitakyushu for their advice and comments.

I extend thanks to Ms. Yurino Sato, Ms. Mei Yoshino, Mr. Takahiro Kawano, and all other members of the Network Engineering Research Group at the University of Kitakyushu for their kindly supports and valuable discussions.

Finally, my greatest appreciation goes to my family. They perpetually helped me whenever I faced various problems. Their long standing has enable me to complete my degree.

Contents

List of Figures

List of Tables

1 Introduction

Cloud computing [1–3] has permeated people's daily lives and work as a social infrastructure. In cloud computing, users can enjoy various services, such as e-mail and file sharing, provided by cloud servers anytime and anywhere (Fig. 1.1). Traditionally, cloud servers are located in data centers (DCs) and far from users. The long distance between users and cloud servers may significantly degrade service quality.

Edge computing [4–6] has been proposed to solve this problem (Fig. 1.2). The key idea of edge computing is to deploy small cloud-like entities called "edge servers" at the edge of networks, which is similar to cloudlet [7], fog computing [8], follow me cloud [9], and small cell cloud [10]. Edge servers are located close to user locations, so they can provide lower latency and faster communication services compared to cloud servers. This enables new services to be provided such as augmented reality and autonomous driving. Such services are typically resource hungry and demand intensive computation and high energy consumption. However, computing resources and battery lives of mobile devices are limited, so users offload tasks to edge servers. Edge servers execute task offloaded and return to the users. Task offloading allows users to enjoy such services in addition to traditional cloud services, including e-mail and file sharing, without depending on the limitations of mobile devices.

1.1 Edge and cloud computing

Although edge computing can provide lower latency and faster services compared to cloud computing, it cannot accommodate all user demands due to the resource limitations of edge servers. Edge computing and cloud computing are hence used jointly for accommodating all user demands, and their roles differ. The main role

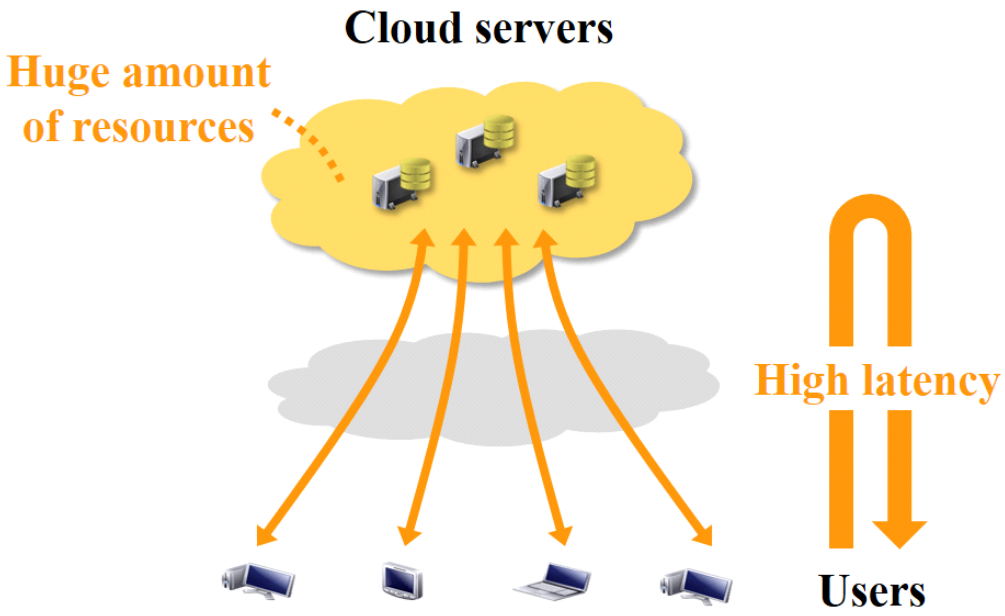


Figure 1.1: Cloud computing

of edge computing is to provide delay-sensitive services, including augmented reality and autonomous driving, provided by edge servers. When the computing resources of edge servers are insufficient, edge servers need to be supported by cloud servers. In this situation, edge servers cooperate with cloud servers to provide such services. On the other hand, the main role of cloud computing is to provide delay-tolerant services, including e-mail and file sharing, provided by cloud servers.

1.2 Issues with edge and cloud computing

Edge and cloud computing are accessed from users with different locations and requirements in terms of transmission rates through networks, so it is important to provide efficient sharing of network resources among users according to the requirements of each user independently of user location to meet their requirements. For example, network resources should be fairly shared among users with

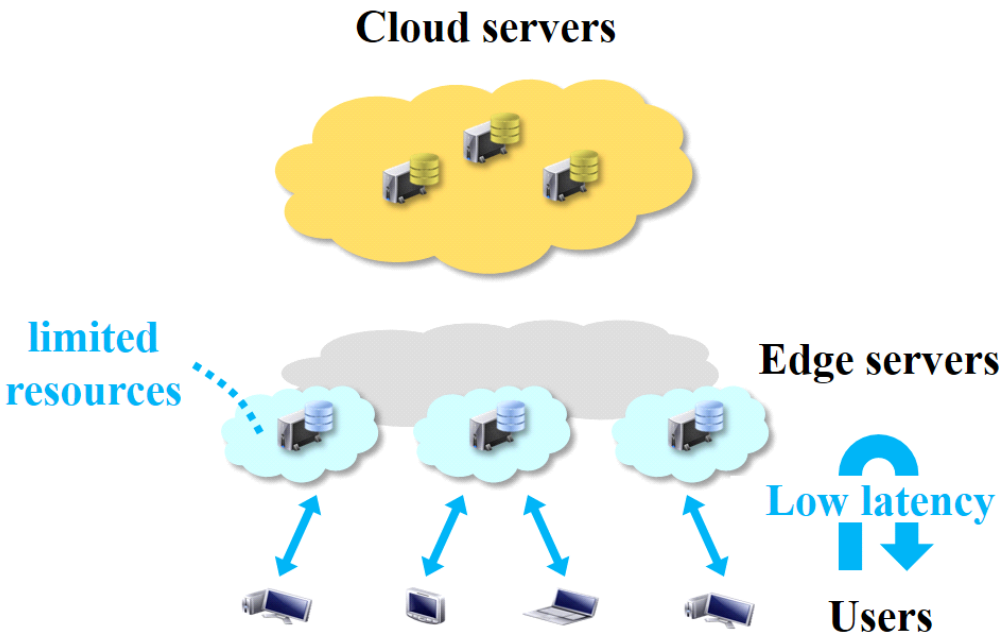


Figure 1.2: Edge computing

the same requirements and different locations. In this context, the following are important issues.

(1) Fair service provision in cloud computing
In cloud computing, cloud servers provide delay-tolerant services to users who are located at different distances from them, so communication flows with different round-trip times (RTTs) coexist. In this situation, network resources cannot be fairly shared among these flows.

(2) Service provision according to user requirements in edge computing
In edge computing, edge servers cooperate with cloud servers to provide delay-sensitive services to users when computing resources of servers are insufficient. Namely, communication flows with different RTTs coexist. In this situation, network resources cannot be fairly shared among these flows. Although these flows have different requirements in terms of transmission rates, sharing of network resources among them according to their

requirements is not ensured since the Internet is composed of "best-effort" networks.

(3) Efficient transmission in edge and cloud computing
When users communicate with cloud servers to access services in edge and cloud computing, they cannot use network resources effectively due to the long distance between users and cloud servers.

These issues are caused by TCP's congestion control when users communicate with corresponding servers using TCP as a data transmission protocol. Since TCP's congestion control is applied according to the RTT, the transmission rate for a flow with a short RTT is effectively increased compared with that of a flow with a long RTT. Moreover, the transmission rate of a flow with a long RTT cannot be effectively increased.

To solve these problems, two approaches to provide fair and efficient sharing of network resources among flows have been proposed. The first approach involves using active queue management (AQM) algorithms applied to intermediate routers. This approach preferentially discards packets belonging to a high rate flow. The second one involves router-assisted congestion control algorithms. In this approach, routers notify senders of available bandwidth information, and the senders adjust its transmission rate based on the received available bandwidth information from the routers. These approaches can provide fair and efficient sharing of network resources among flows if the flows share a bottleneck link, but it is difficult to apply them if bottleneck links shared by all flows do not exist.

Approaches to manage computing and network resources in edge and cloud computing have been proposed. These approaches allocate computing and network resources among users for task offloading and migration based on the requirements of each user and available computing and network resources. They can effectively manage computing and network resources if bottleneck links exist in wireless networks, but they do not work well if bottleneck links exist in wired networks. Namely, issues (1)–(3) in edge and cloud computing have not yet been solved.

1.3 Overview of this dissertation

In this dissertation, I propose efficient schemes to allocate and use network resources to solve issues (1)–(3) in edge and cloud computing. Through simulation evaluations, I show the effectiveness of the proposed schemes in edge and cloud computing.

In Chapter 2, I give an overview of TCP to provide a better understanding of this dissertation and introduce related work.

In Chapters 3 and 4, I discuss an efficient method to fairly allocate bandwidth among users in cloud computing regarding issue (1). In Chapter 3, I propose a bandwidth allocation scheme based on collectable information to improve fairness in DC networks and discuss the evaluation of the proposed scheme's fundamental characteristics by focusing on the effect of allocation factor α used to adjust the transmission rate of each flow. In Chapter 4, I discuss extending the proposed scheme to reallocate unutilized bandwidth caused by congestion outside DC networks to other competing flows to improve link utilization in these networks and discuss the evaluation of the proposed scheme's performance by focusing on how to set parameter values (e.g., notification interval time for flow information) in more detail.

In Chapters 5 and 6, I discuss an efficient method to allocate bandwidth according to user requirements in edge computing regarding issue (2). In Chapter 5, I propose a bandwidth allocation scheme based on collectable information to meet the requirements of each flow in edge computing and discuss the evaluation of the proposed scheme's performance in detail. In Chapter 6, I discuss extending this scheme to allocate bandwidth based on residual bandwidth information to improve link utilization in edge computing.

In Chapter 7, I discuss a method to provide efficient transmission in edge and cloud computing regarding issue (3). When users receive services from cloud servers in edge and cloud computing, they cannot use bandwidth effectively due to the long distance between them and cloud servers. This is because TCP, which is commonly used for reliable communication, controls transmission rates on an end-to-end basis. This significantly degrades the service quality for such users. Therefore, I propose flow splitting and aggregation schemes to improve the throughput of each flow in edge and cloud computing.

In Chapter 8, I conclude this study and address future work.

The results discussed in Chapter 3 are mainly taken from [14], Chapter 4 from [16], Chapter 5 from [17], Chapter 6 from [18], and Chapter 7 from [19].

2 Related Works

In this chapter, I describe an overview of TCP to give a better understanding of this dissertation and introduce related works.

2.1 Transmission control protocol (TCP)

In edge and cloud computing, users can enjoy various services provided by edge and cloud servers at anytime and anywhere. TCP is commonly used for enjoying such services as a reliable data transmission protocol. TCP estimates network conditions based on packet losses and then adjusts the window size of each flow according to the estimated network conditions. The window size is set to the minimum of *cwnd* and *awnd*, where *cwnd* is a TCP state variable that limits the amount of data that the TCP sender can send into the network before receiving an acknowledgement (ACK) message and *awnd* is a variable that advertises the amount of data that the TCP receiver can receive. *cwnd* is increased by one segment for every RTT as shown in Fig. 2.1. Note that the growing speed of window size depends on RTT. If a packet is lost due to network congestion, the lost packet is retransmitted and *cwnd* is decreased. Through these procedures, TCP enables efficient and reliable communication.

However, TCP cannot provide fair and efficient sharing of network resources among flows with different RTTs. This problem is caused by TCP's congestion control mechanisms. Since TCP's congestion control is applied according to RTTs, the transmission rate for a flow with a short RTT is effectively increased compared with that of a flow with a long RTT. It may cause RTT unfairness problem [11, 12] as shown in Fig. 2.2. Moreover, the transmission rate of a flow with a long RTT cannot be effectively increased.

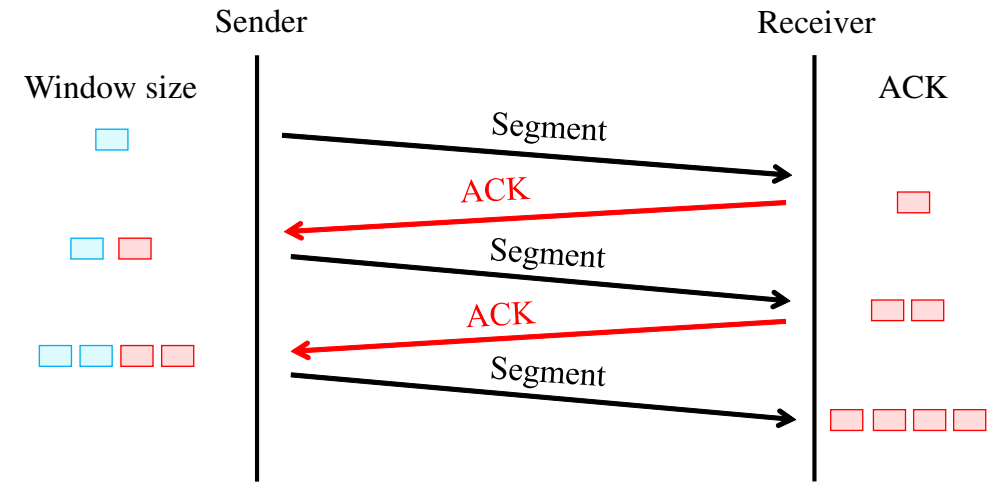


Figure 2.1: Variation of window size

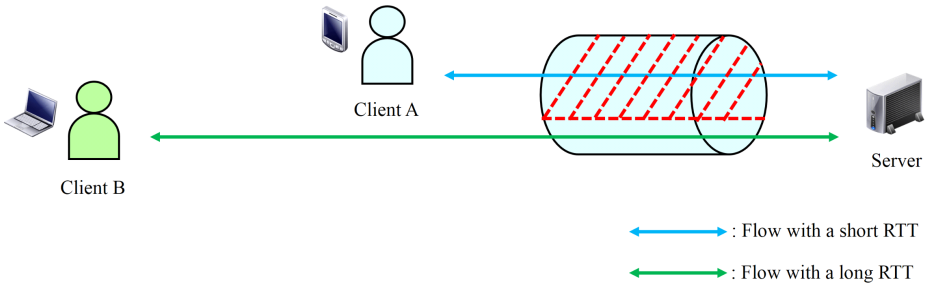


Figure 2.2: RTT unfairness problem

2.2 Congestion control algorithm in DC networks

One of the important issues in DC networks is to improve fairness among flows [20]. Several studies report fairness issues in DC networks such as TCP Outcast problem [21]. For example, data center TCP (DCTCP) [22] and SAB [23] have been proposed to solve this problem. DCTCP employs a threshold on switches and informs the TCP sender about congestion condition by using a marking algorithm when the switches' buffer size exceeds the threshold. SAB notifies

the TCP sender of information about available bandwidth at switches in DC networks. Since these studies focus only on fairness among competing flows at one output port of switches in DC networks, they cannot solve problems mentioned in Sec. 1.2.

2.3 Active queue management (AQM) algorithm

In the past, many kinds of AQM algorithms (Fig. 2.3) have been proposed for use on routers to improve fairness among flows. A typical AQM algorithm is random early detection (RED) [24]. This approach sets maximum and minimum queue lengths and drops packets when the current queue length is larger than the set maximum queue length. This enables fair communication among competing flows by preferentially discarding packets which belong to a high rate flow. However, it is difficult to determine an appropriate maximum and minimum queue length because the appropriate values depend on network conditions such as the number of flows and delay time. Unlike RED, the CHOKe AQM algorithm does not require parameter settings [25]. This approach randomly selects one packet from the output queue on routers when the router receives a packet. If the source addresses of the received and selected packets are the same, both packets should be dropped. CHOKeW [26] and CHOKeR [27], which are improved versions of the CHOKe algorithm, adjust the packet dropping rate of each flow based on service classes and network conditions. These AQM algorithms on routers can improve fairness among competing flows if the flows share a bottleneck link, but it is difficult to apply them if bottleneck links shared by all flows do not exist.

2.4 Router-assisted congestion control algorithm

A typical router-assisted congestion control algorithm is the eXplicit Control Protocol (XCP) [28]. In XCP, routers provide TCP senders with information about available bandwidth. When a TCP sender receives this information, it

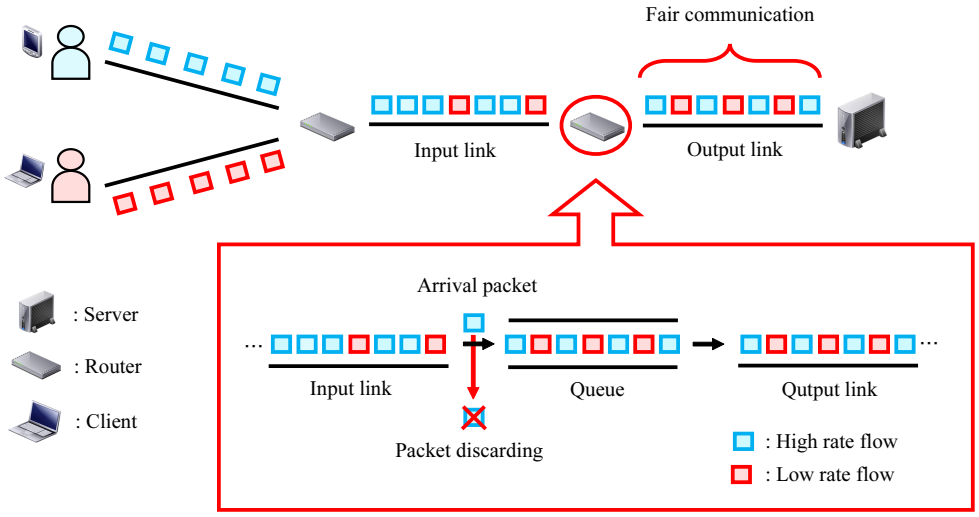


Figure 2.3: Active queue management (AQM)

adjusts a congestion window size based on the received information. Another router-assisted congestion control algorithm is proposed by L. Kalampoukas et al. [29]. This approach modifies a TCP receiver's advertised window size in packets forwarding to TCP senders on intermediate routers. These approaches can improve fairness among competing flows. However, the fairness among flows will be extremely degraded if the flows do not share the same bottleneck router providing the TCP sender with the information about available bandwidth.

2.5 Resource management in edge and cloud computing

Several approaches to manage computing and network resources in edge and cloud computing have been reported. In this section, I introduce some of these studies: task offloading, resource allocation, and migration.

2.5.1 Task offloading schemes

Task offloading schemes have been proposed by many researchers. For example, a task offloading scheme has been proposed to meet delay requirements of delay-sensitive applications [30]. This scheme preferentially offloads tasks for delay-sensitive applications to edge servers near users, while other tasks for delay-tolerant applications are offloaded to distant cloud servers. A SDN-based task offloading scheme has been proposed to minimize computation delays and save energy consumption of mobile devices [31]. By deploying a controller in access networks, this scheme can obtain global information about mobile devices, edge servers, and tasks; therefore, it enables optimal task offloading of mobile devices. A new concept of task caching has been introduced to further improve the quality of mobile applications in edge computing [32]. Since each edge server can cache offloaded tasks from users, users do not have to offload tasks to edge servers if they are already cached on them.

2.5.2 Resource allocation schemes

The authors in [33] addressed a problem of joint allocation of radio and computing resources. The authors in [34] considered a problem of joint task offloading and resource allocation in a multi-user environment. These works focus only an edge computing environment where a single edge server exists. On the other hand, the authors in [35] considered a problem of joint task offloading and resource allocation in a multi-cell and multi-server edge computing environment. They formulated the problem as a mixed integer non-linear program (MINLP) and decompose the problem into two problems: (i) a resource allocation (RA) problem and (ii) a task Offloading (TO) problem. They addressed the RA problem by using convex and quasi-convex optimization techniques and proposed a novel heuristic algorithm to address the TO problem.

2.5.3 Migration schemes

Migration schemes in edge computing can be classified into two schemes. The first one is Markov decision process (MDP) based service migration schemes, including one-dimensional MDP and two-dimensional MDP [36, 37]. The key idea of them

is to formulate the mobility-driven service migration problem as a MDP. Such a formulation is suitable where the user mobility follows or can be approximated by a mobility model that can be described by a Markov chain. The second one is time window based migration schemes [38] which deals with the migration problem from another point of view. These schemes search the optimal service placement sequence that minimizes the average cost over a given time window. In these schemes, a look-ahead window is defined as a time period in the future that can be predicted.

2.5.4 Problem

These approaches can efficiently manage computing and network resources if bottleneck links exist in wireless networks, but they do not work well if bottleneck links exist in wired networks. In this dissertation, I focus on how to allocate and utilize wired network resources among flows rather than wireless network resources (i.e., radio resources) among them.

3 A Bandwidth Allocation Scheme to Improve Fairness in Cloud Computing

3.1 Introduction

Users can now connect to the Internet through a wide variety of access networks and communication terminals such as smart phones or PCs. This has created a great demand for cloud computing which allows users to enjoy various Internet services provided by data centers (DCs) [39] anytime, anywhere. Major services available through cloud computing include file sharing or transaction services which commonly use TCP as a reliable data transmission protocol.

In cloud computing, the problem arises that a user's quality of service is unfair among users who are different distances from DCs. This problem is caused by how TCP controls congestion. TCP cannot exactly identify the conditions of other flows, so it estimates an available bandwidth based on packet losses. Since congestion control is applied according to round-trip time (RTT), the transmission rate for a user with a short RTT is effectively increased compared with that of a user having a long RTT. Fairness in terms of throughput among flows with different RTTs needs to be improved.

To improve fairness among flows, many kinds of the active queue management (AQM) technologies applied on routers have been proposed. These technologies realize fair communication among flows, but the processing cost on routers increases enormously as the number of flows increases.

In this study, I propose a bandwidth allocation scheme based on collectable information in DC networks to provide fair communication service to each user.

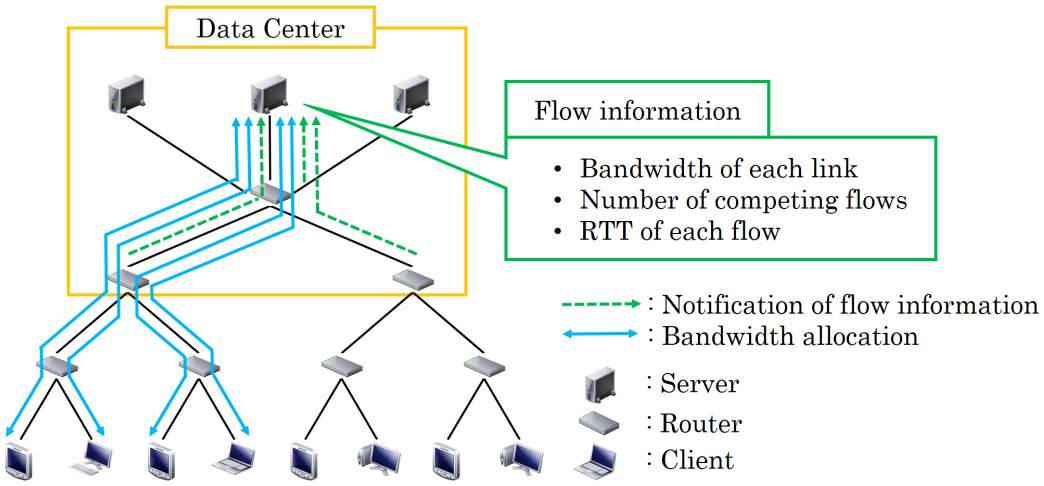


Figure 3.1: Proposed Scheme

This scheme collects flow information including the bandwidth of each link, the number of competing flows, and the RTT of each flow from routers and servers in DC networks, and then fairly allocates transmission rates among flows based on the collected information. I show the effectiveness of this approach through simulation evaluations.

3.2 Proposed scheme

The proposed scheme fairly allocates transmission rates among flows based on collectable information in DC networks as shown in Fig. 3.1. Flow information such as the bandwidth of each link, the number of competing flows, and the RTT of each flow is collected by UCCA [40], which uniformly manages the congestion information in the networks. The RTT of each flow can be measured using TCP's timestamp option. If the server does not send any data to the client, it periodically sends probe packets to measure RTT. Routers in DC networks periodically inform servers of the number of existing flows, which can be measured by extending the OpenFlow architecture [41]. The servers identify bottleneck links based on the information about the number of existing flows received from the routers and the bandwidth of each link, and then calculate the transmission rate which should

be allocated to each flow by dividing the bandwidth of the bottleneck link by the number of competing flows. Note that I focus on a case where bottleneck links exist in DC networks to allocate resources in DC networks to each flow fairly and effectively, so I will attack a case where bottleneck links exist outside DC networks in future work.

To allocate transmission rates to TCP flows with window-based congestion control, the bandwidth delay product of each flow as the allocated window size should be calculated based on the bandwidth which should be allocated and the RTT of each flow. This scheme employs an allocation factor α and the allocated window size is multiplied by α. For downwards communication from a server to a client, the server can simply allocate the calculated window size to each flow. On the other hand, for upwards communication from a client to a server, the server needs to inform the client of the allocated window size. This scheme uses TCP's advertised window size to notify the client of the window size. This bandwidth allocation enables fair communication among flows which have different RTTs.

3.3 Simulation model

To investigate the efficiency of the proposed scheme, I evaluated it through simulation using Network Simulator ns-3 [43] after its implementation. Figure 3.2 shows the simulation topology. In this simulation, there are two client groups A and B, and these groups have access links with different delay times. The clients of each group communicate with the server using TCP NewReno. The propagation delay time of access links is set to a uniform random number which ranges from 2 to 6 ms for GroupA or from 10 to 16 ms for GroupB. The propagation delay time of other links is set to 1 ms. The bandwidth of all links is set to 200 Mb/s. The allocation factor α varies from 1.0 to 1.8. The interval time to notify the flow information is set to 0.05 ms.

I focus on two evaluation scenarios: join and leave scenarios. 25 clients of each group communicate with the server after the simulation starts. At 12 s after the simulation starts, one client of each group starts communication in the join scenario, while stops communication in the leave scenario.

I evaluate Jain's fairness index [44] and total throughput for the proposed

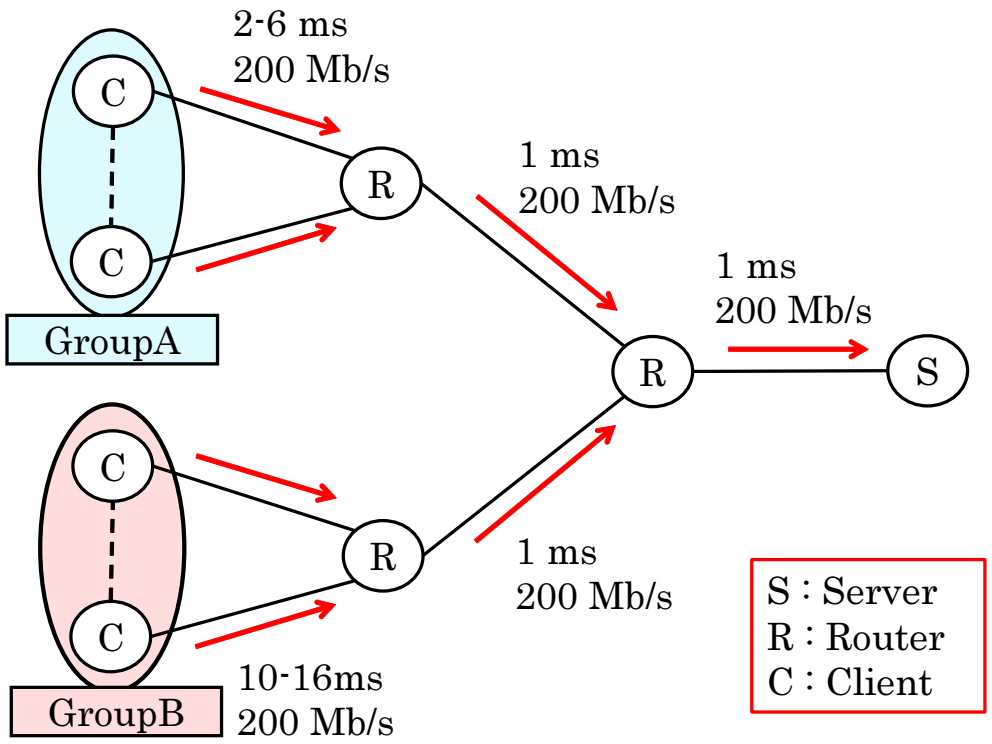


Figure 3.2: Simulation topology

scheme and conventional TCP. The fairness index is defined as Eq. (3.1), where x is the throughput of each flow and n is the number of existing flows.

$$f(x_1, x_2, x_3, \ldots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2} \tag{3.1}$$

Namely, fairness is higher as the index gets closer to 1. I evaluate the fairness index and total throughput averaged over 10 trials, which are calculated at intervals of 0.1 s.

3.4 Simulation results

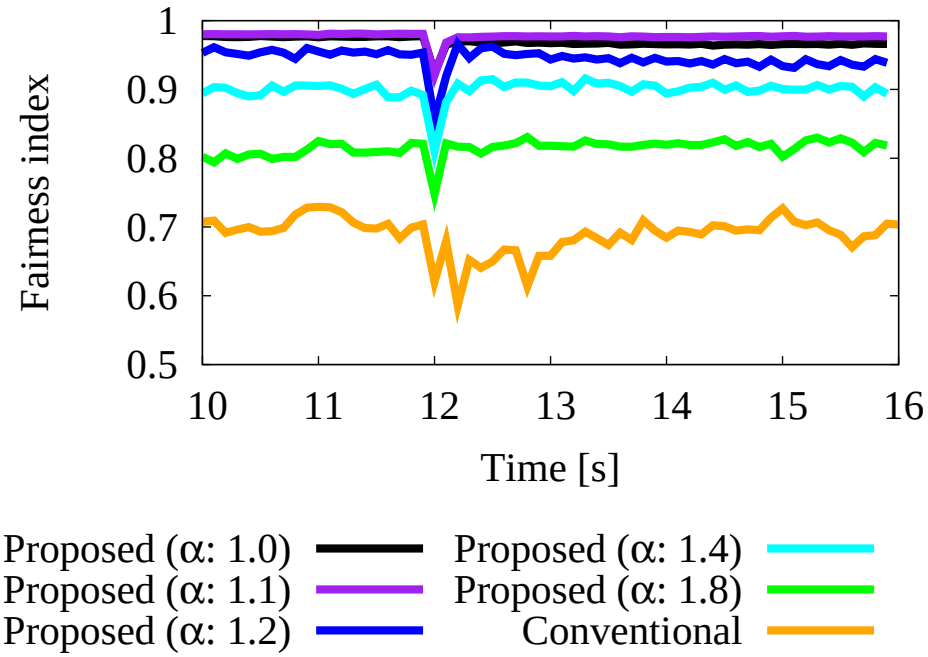
I now show simulation results and discuss the effectiveness of the proposed scheme as compared with the conventional scheme. Figures 3.3(a) and 3.3(b) show the fairness index and total throughput, respectively, of the proposed and conventional schemes in the join scenario. The proposed scheme achieves a higher fairness index than that of the conventional scheme. This is because the proposed scheme allocates an adequate window size to each flow according to its RTT. In particular, the proposed scheme with a small α achieves good fairness, but it degrades the total throughput. Otherwise, the proposed scheme with a large α achieves poor fairness and high throughput. Namely, the proposed scheme has a trade-off relationship between the fairness and total throughput for α. In this simulation, the proposed scheme with α of 1.2 enables good fairness and high total throughput.

In the leave scenario, the fairness index and total throughput of the proposed and conventional schemes are shown in Figs. 3.4(a) and 3.4(b), respectively. Similarly in the join scenario, the proposed scheme achieves a higher fairness index as compared with the conventional scheme. On the other hand, the proposed scheme with a small α drastically degrades the total throughput. This is because the proposed scheme only utilizes the allocated bandwidth, so the impact on the total throughput in the leave scenario is larger than that in the join scenario. However, the proposed scheme with α of 1.2 achieves good fairness and high total throughput like the join scenario. Consequently, the proposed scheme enables fair communication among flows which have different RTTs as well as high total throughput by setting the adequate value of α (slightly larger than 1 in this simulation).

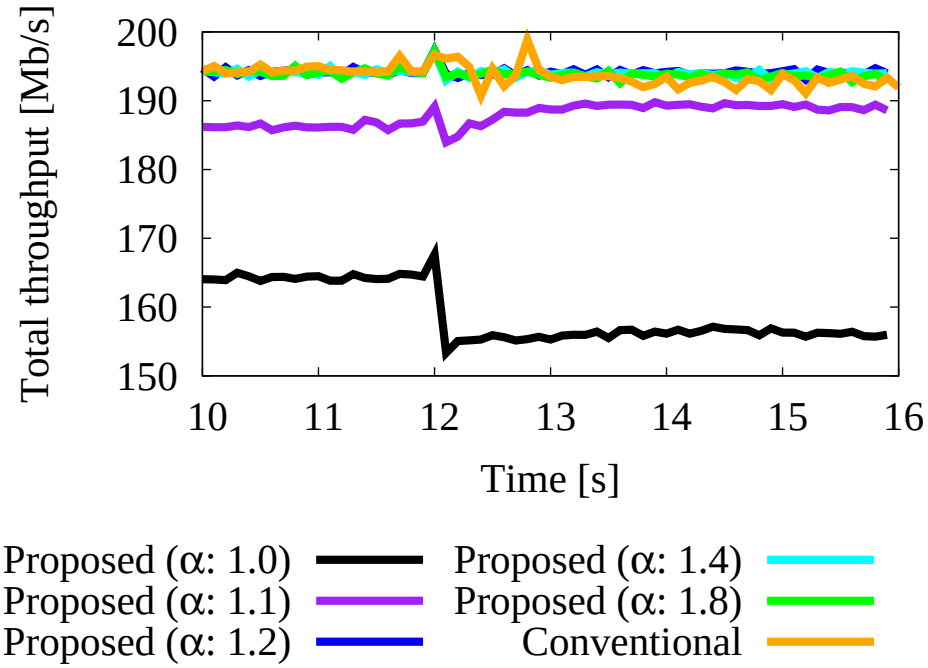
3.5 Conclusion

This proposed bandwidth allocation scheme uses collectable information in DC networks to provide fair communication service to each user. Simulation evaluations have indicated that this scheme with the adequate allocation factor enables higher fairness and throughput performance than can be achieved with conven-

tional TCP. In future work, I will consider the most effective way to dynamically adjust the allocation factor as well as to fairly allocate link bandwidth in DC networks to each user even when access networks are bottlenecks.

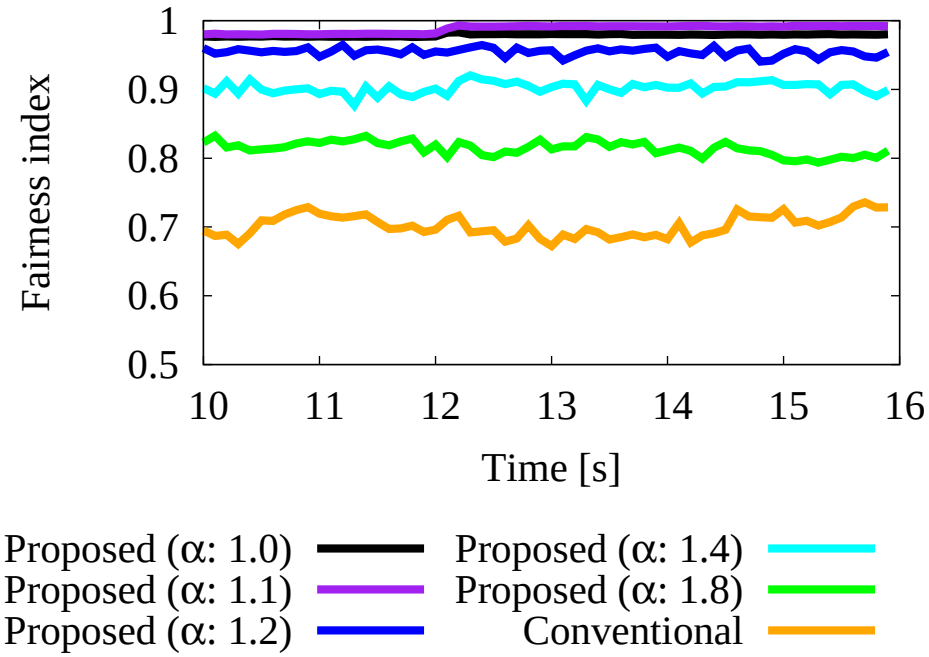


(a) Fairness index

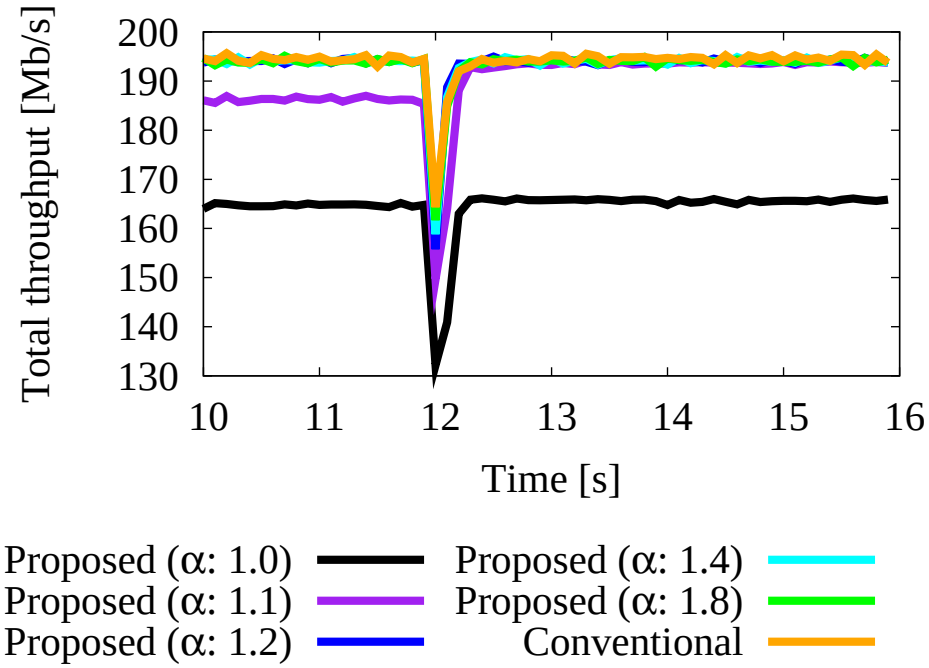


(b) Total throughput

Figure 3.3: Join scenario



(a) Fairness index



(b) Total throughput

Figure 3.4: Leave scenario

4 A Bandwidth Allocation Scheme to Improve Fairness and Link Utilization in Cloud Computing

4.1 Introduction

Users can now connect to the Internet through a wide variety of access networks and communication terminals such as smart phones or PCs. This has brought a great demand for cloud computing, which allows users to enjoy various Internet services provided by data centers (DCs) [39] anytime, anywhere. In the Internet of Things (IoT), cloud computing has been more attractive and important since users can enjoy IoT services using various sensor information collected by DCs. Major services available through cloud computing include file sharing or transaction services which commonly use TCP as a reliable data transmission protocol.

In cloud computing, a problem arises that a user's quality of service in terms of transmission rate unfairly depends on the user's distance from DCs. This problem is caused by how TCP controls congestion. TCP cannot exactly identify the conditions of other flows, so it estimates an available bandwidth based on packet losses. Since congestion control is applied according to round-trip time (RTT), the transmission rate for a user with a short RTT is effectively increased compared with that of a user having a long RTT. Fairness in terms of throughput among flows with different RTTs as well as link utilization in DC networks needs to be improved.

There are two approaches to improving fairness among flows. The first approach is to apply active queue management (AQM) technologies on routers. This approach realizes fair communication among flows by preferentially discarding packets which belong to a high-rate flow. The second approach is to use router-assisted congestion control algorithms. In this approach, routers provide the TCP sender with information about available bandwidth and the TCP sender adjusts the congestion window based on the information received from routers. These approaches enable more fairness than the original TCP. However, the fairness among competing flows will be extremely degraded if the flows do not share the same bottleneck router that uses the AQM technologies or provides the TCP sender with the information about available bandwidth.

In this study, I propose a bandwidth allocation scheme based on collectable information to improve fairness and link utilization in DC networks. This scheme collects flow information including the bandwidth of each link, the number of competing flows, the RTT of each flow, and the actual throughput of each flow from routers and servers in DC networks, and then fairly allocates transmission rates among flows based on the collected information. In addition, this scheme reallocates unutilized bandwidth to other competing flows in DC networks when bottleneck links exist outside of the DC networks. I show the effectiveness of this approach through simulation evaluations.

The rest of this chapter is organized as follows. In Section 4.2, I propose a bandwidth allocation scheme based on collectable information to improve link utilization in DC networks as well as fairness among flows. I describe simulation environment and results in Section 4.3 and 4.4, respectively. I conclude in Section 4.5.

4.2 Proposed scheme

As mentioned in Section 4.1, fairness among flows with different RTTs will be extremely degraded because TCP does not all the available bandwidth to be accurately estimated. To solve this problem, I propose a scheme that fairly allocates transmission rates among flows based on collectable information in DC networks. This scheme identifies bottleneck links based on flow information collected from

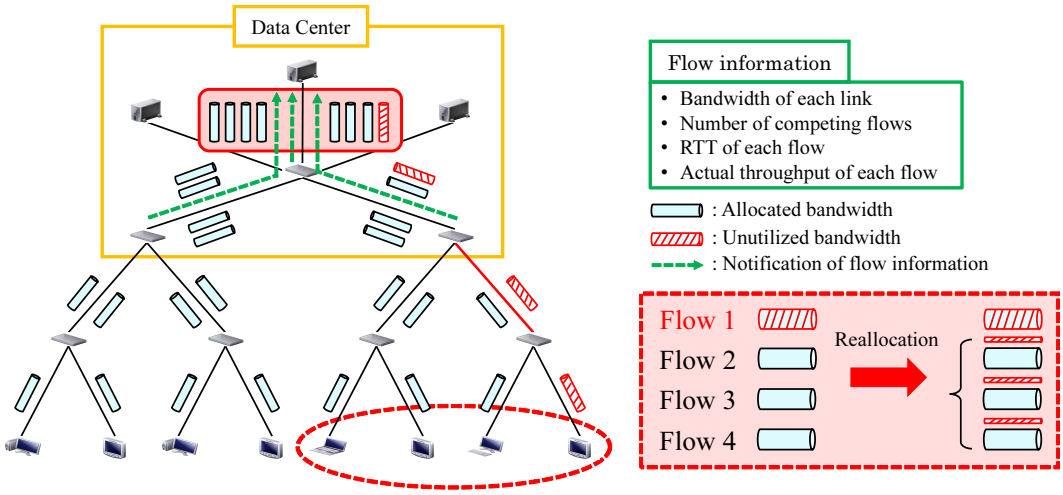


Figure 4.1: Overview of bandwidth allocation scheme

routers and servers in DC networks, and then calculates the transmission rate that should be allocated to each flow by dividing the bandwidth of a bottleneck link by the number of competing flows. In addition, this scheme reallocates any unutilized bandwidth caused by congestion outside DC networks to other competing flows to improve link utilization in DC networks when bottleneck links exist outside DC networks, as shown in Fig. 4.1. Namely, I propose two bandwidth allocation schemes to improve fairness and link utilization in DC networks. They are the proposed (no reallocation) and (reallocation) schemes, which only allocates a fair transmission rate to each flow and additionally allocates unutilized bandwidth to competing flows, respectively. Flow information such as the bandwidth of each link, the number of competing flows, the RTT of each flow, and the actual throughput of each flow is collected by a Unified Central Congestion Control Architecture (UC^3) [40], which uniformly manages the congestion information in the networks, as shown in Fig. 4.2. Namely, this scheme can adapt to network environment where is managed by an organization such as DC networks because it is easy to collect flow information from their routers, although it is hard to collect flow information from any routers in the Internet.

I describe here the operation of the bandwidth allocation scheme shown in Fig. 4.3. The server measures the RTT and actual throughput R_{th} of each flow

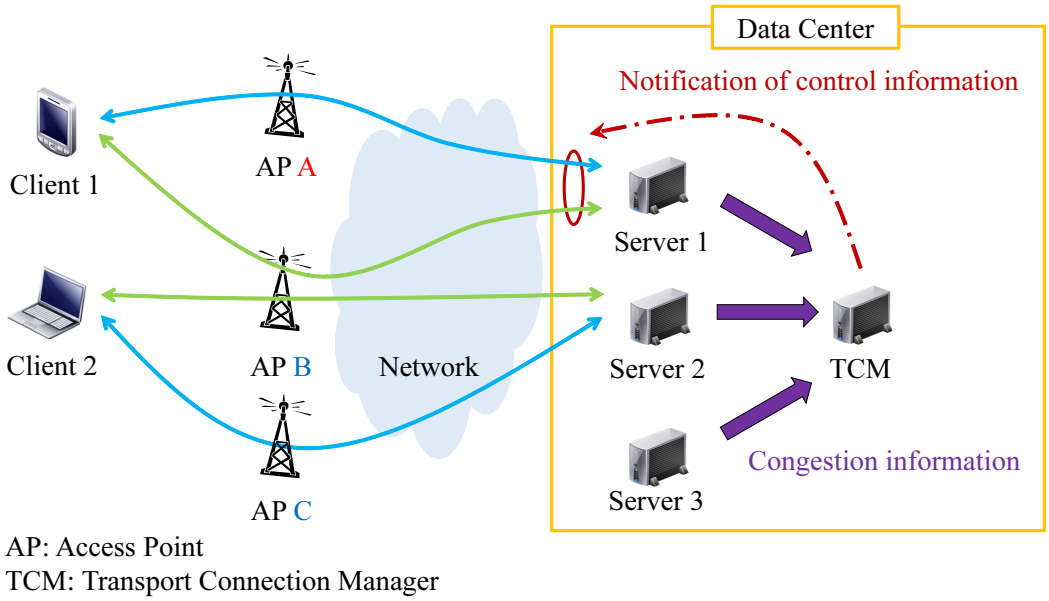


Figure 4.2: Unified central congestion control architecture

and then calculates the exponential moving average E_{th} of each flow's R_{th} by $\alpha R_{th} + (1 - \alpha)E_{th}$, where α is a weighting factor. The RTT of each flow can be measured through normal TCP operation. If the server does not send any data to the client, it periodically sends probe packets to measure RTT. I then move onto operations of routers. This scheme requires that routers in DC networks must have a special function that periodically informs the number of existing flows. There are many ways to enable this. For example, OpenFlow-enabled routers can receive a "read state" message that requests to send statistic information like the number of existing flows [41]. Another example is IP Flow Information Export (IPFIX), which enables to send statistic information like the number of flows [42]. The server identifies bottleneck links based on information about the number of existing flows received from the routers and the bandwidth of each link, and then it calculates the transmission rate which should be allocated to each flow by dividing the bandwidth of the bottleneck link by the number of competing flows. If E_{th} of each flow multiplied by a threshold factor β is smaller than the allocated transmission rate, the server calculates the transmission rate which

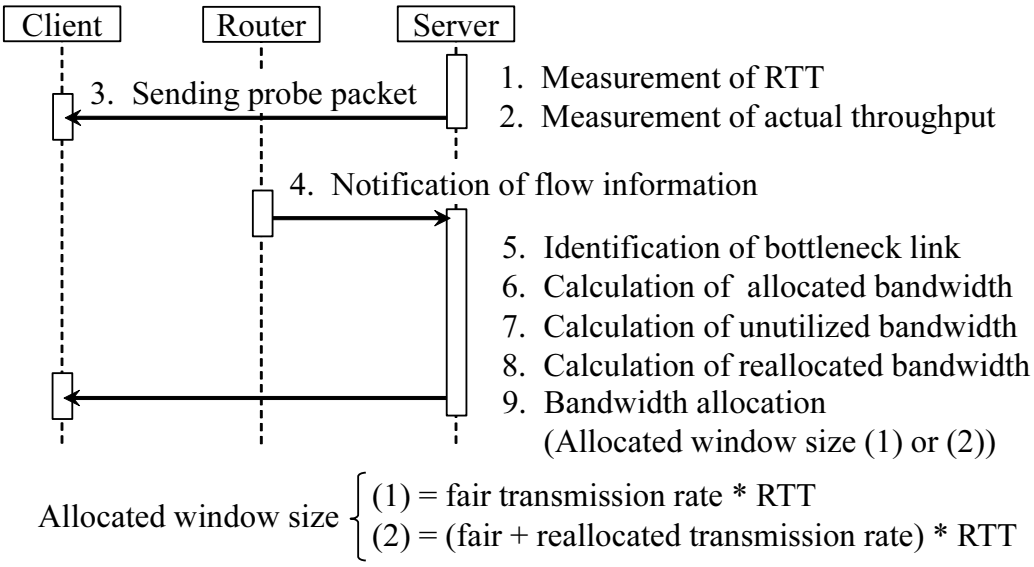


Figure 4.3: Operation of bandwidth allocation scheme

should be reallocated to other competing flows. The reallocated transmission rate is calculated by dividing the total unutilized bandwidth of flows through the same edge router (ingress routers in DC networks) by the number of other competing flows which fully utilize the allocated transmission rate. Namely, the allocated transmission rate for each flow which cannot fully utilize the allocation is the available bandwidth fairly divided by the number of flows. On the other hand, the allocated transmission rate for each flow which fully utilizes the allocation is added to the reallocated transmission rate. These bandwidth allocation procedures are periodically performed every time the flow information about the number of competing flows and the actual throughput of each flow is updated. Namely, the allocated transmission rates converge to adequate values even when network conditions change.

To allocate transmission rates to TCP flows with window-based congestion control, the bandwidth delay product (BDP) of each flow that is used as the allocated window size should be calculated based on the bandwidth which should be allocated and the RTT of each flow. When the BDP of each flow cannot be divided by the maximum segment size (MSS), the allocated transmission rate

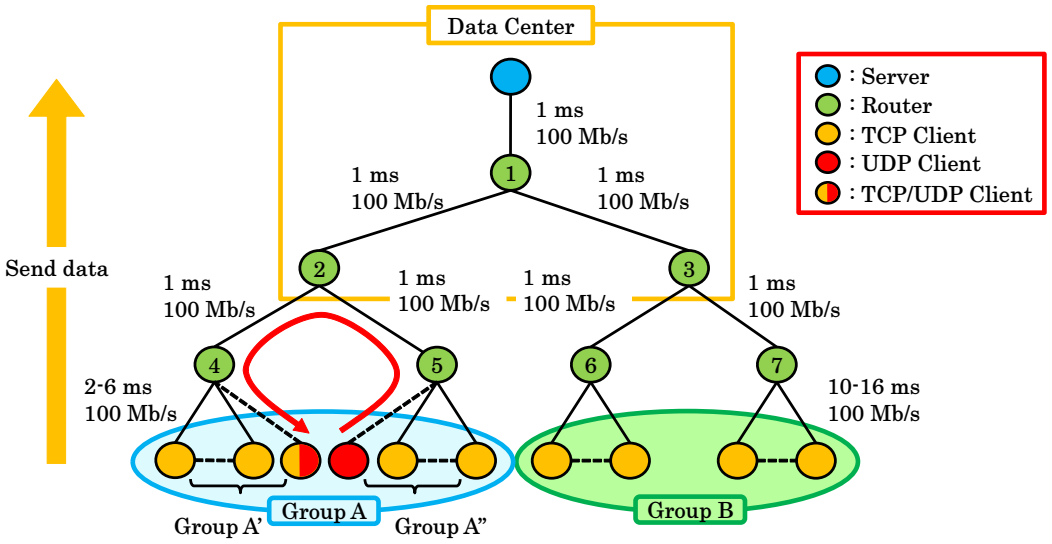


Figure 4.4: Simulation model

becomes a little bit lower due to the remainder. This may degrade link utilization in DC networks. To prevent this, I consider three ways to control the allocated window size: round-up, round, and round-down methods. In each method, the BDP allocated to each flow is respectively rounded up, rounded, and rounded down by the MSS units. If the BDP allocated to each flow is rounded up by the MSS units, the total allocated transmission rate will exceed the bandwidth of bottleneck links, although this effect can be sufficiently absorbed by routers' buffer. It will also result in the convergence to the appropriate amount of allocated bandwidth.

For downwards communication from a server to a client, the server can simply allocate the calculated window size to each flow. On the other hand, for upwards communication from a client to a server, the server needs to inform the client of the allocated window size. This scheme uses TCP's advertised window size to notify the client of the window size. This bandwidth allocation enables fair communication among flows which have different RTTs as well as high link utilization in DC networks.

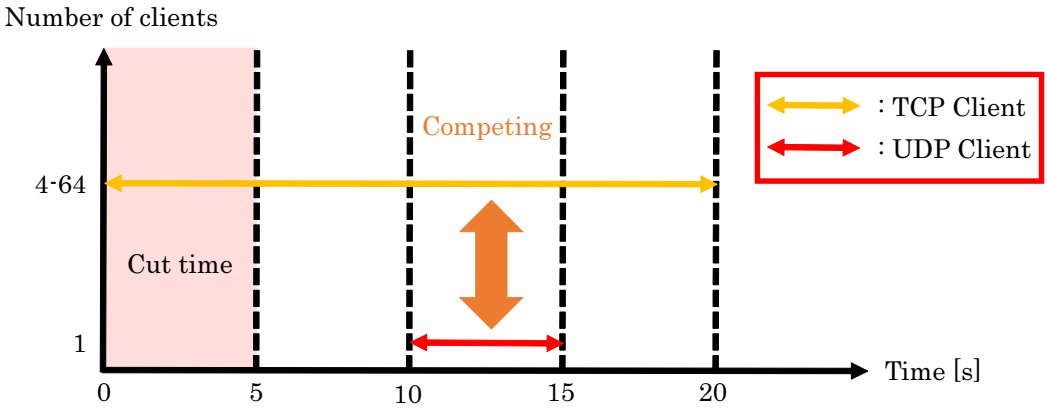
Table 4.1: Simulation parameters

TCP algorithm	TCP NewReno
Measurement interval time for actual throughput of each flow	0.1 [s]
Notification interval time for flow information	$0.005 \sim 0.5$ [s]
Weighting factor α	$0.1 \sim 1.0$
Threshold factor β	$0.1 \sim 1.0$
Buffer size on routers	200 [packet]
Number of TCP clients	$4 \sim 64$
Number of UDP clients	2
Transmission rate of UDP client	85 [Mb/s]
Number of trials	10

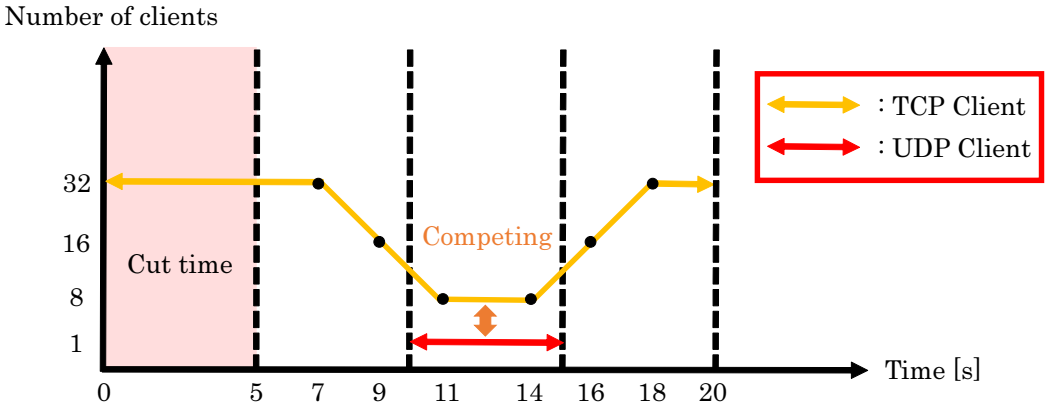
4.3 Simulation model

To investigate the effectiveness of the proposed schemes, I evaluated them through simulation using Network Simulator ns-3 [43] after their implementation. I implemented functions to measure the number of competing flows and actual throughput of each flow, to identify the bottleneck links, to calculate the transmission rates allocated to each flow, and to report the flow information in this simulator.

The proposed scheme allocates an adequate bandwidth to each flow based on the information including RTT obtained by end-to-end measurement and the number of flows obtained from routers. I evaluate the performance of this scheme focusing on the effect of such information in this simulation. Figure 4.4 shows the simulation topology. In this simulation, there are two client groups A and B, and these groups have access links with different delay times. In addition, Group A consists of two groups A' and A". The clients of each group continuously send data to the server using TCP NewReno. Note that proposed scheme works independently of TCP congestion control algorithms, so I employ TCP NewReno as a simple typical TCP. I assume that Nodes R2 and R3 are edge routers in DC networks. To realize an environment where bottleneck links exist outside DC



(a) Fixed scenario



(b) Variable scenario

Figure 4.5: Simulation scenario

networks, two UDP clients are also located in Group A so that TCP and UDP flows coexist on the link between Nodes R2 and R5. The propagation delay time of access links is set to a uniform random number which ranges from 2 to 6 ms for Group A or from 10 to 16 ms for Group B. The propagation delay time of other links is set to 1 ms. The bandwidth of all links is set to 100 Mb/s. Other simulation parameters are summarized in Table 4.1.

4.3.1 Simulation scenario

I perform two simulation scenarios; the fixed and variable scenarios about the number of TCP clients. These scenarios are illustrated in Fig. 4.5. In the fixed scenario, the fixed number (4–64) of TCP clients communicate with the server after the simulation starts. In the variable scenario, 32 TCP clients communicate with the server after the simulation starts. 4 TCP clients of each group stop to communicate with the server at intervals of 2 s during the period from 7 to 11 s, and 4 TCP clients of each group start to communicate with the server at intervals of 2 s during the period from 14 to 18 s. In both scenarios, at 10 seconds after the simulation starts, one UDP client of Group A" begins to send data to one UDP client of Group A' and then stops sending at 15 seconds. I ignore the first 5 seconds.

4.3.2 Evaluation indices

I evaluated Jain's fairness index [44] and total throughput for the proposed scheme and conventional TCP. The fairness index is defined as Eq. (4.1), where x is the throughput of each flow and n is the number of existing flows. Fairness is higher as the index gets closer to 1.

$$f(x_1, x_2, x_3, \ldots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2} \tag{4.1}$$

The fairness index and total throughput are calculated at intervals of 0.1 s.

4.3.3 Comparison schemes

I evaluate the performance of the proposed schemes (Proposed (no reallocation) and Proposed (reallocation)) compared with that of conventional TCP (Conventional) with the averaged fairness index and total throughput. The Proposed (no reallocation) scheme allocates a fair transmission rate to each flow, while the Proposed (reallocation) scheme additionally allocates unutilized bandwidth to competing flows which fully utilizes the allocated bandwidth.

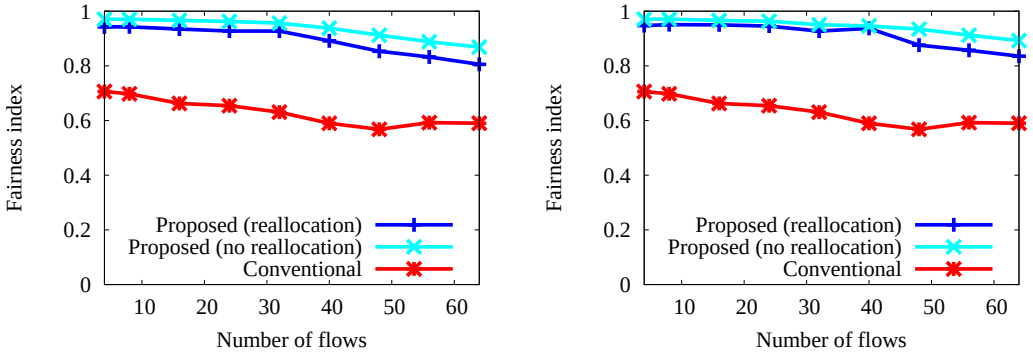
4.4 Simulation results

In this section, I now show simulation results and discuss the effectiveness of the proposed schemes as compared with the conventional scheme. In the fixed scenario about the number of TCP clients, I investigate the effect of the number of TCP clients and the notification interval time for flow information in Sections 4.4.1 and 4.4.2, respectively. In the variable scenario about the number of TCP clients, I also evaluate the effect of the weighting factor α and threshold factor β in Sections 4.4.3 and 4.4.4, respectively.

4.4.1 Effect of the number of TCP clients

Figures 4.6 and 4.7 show the fairness index and total throughput of the proposed and conventional schemes, respectively, when the number of TCP clients varies from 4 to 64. Under the proposed (reallocation) scheme with "Round-down" method, queue length on Router 1 is also shown as functions of time when the number of flows varies from 4 to 64 in Fig. 4.8. Here, the notification interval time for flow information is set to 0.05 s, α is set to 0.4, and β is set to 0.8. In these figures, "Round-up", "Round", and "Round-down" represent the methods to control the allocated window size.

In Fig. 4.6, the proposed schemes achieve higher fairness than the conventional scheme. This is because the proposed schemes allocate an adequate window size to each flow according to its RTT. In Fig. 4.7, the proposed (reallocation) scheme achieves higher total throughput than the proposed (no reallocation) scheme. This is because the proposed (reallocation) scheme effectively reallocates the unutilized bandwidth from flows which cannot fully utilize their allocated bandwidth to other competing flows which can fully utilize the allocated bandwidth. Moreover, the proposed schemes with the "Round" and "Round-down" methods obtain lower total throughput than the conventional scheme, while the proposed (reallocation) scheme with the "Round-up" method comprehensively achieves total throughput that is higher or almost the same as for the conventional scheme regardless of the number of TCP clients. This is because the "Round" and "Round-down" methods might not allocate transmission rates satisfactorily when the BDP of each flow cannot be divided by the MSS. When



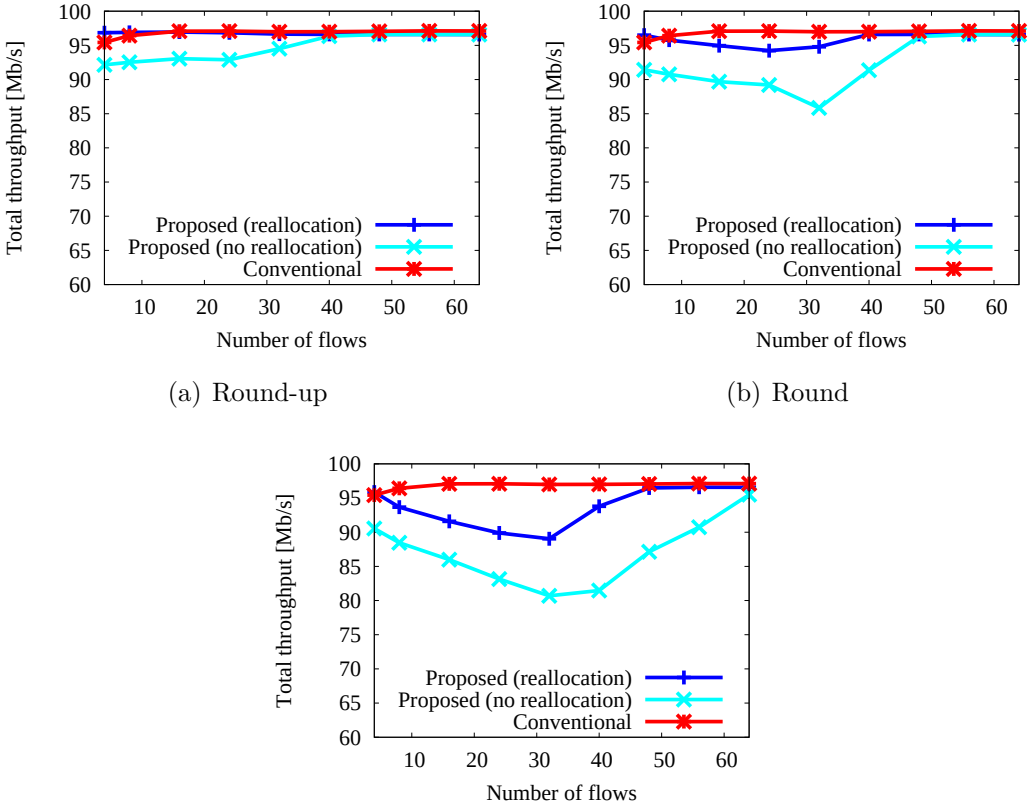
(a) Round-up

(b) Round

(c) Round-down

Figure 4.6: Effect of the number of TCP clients: Fairness index

the number of flows is small (4 to 32), the total throughput of the proposed schemes with "Round" and "Round-down" methods decreases as the number of flows increases. This is because these methods allocate smaller window size to each flow than BDP and the total unallocated bandwidth of all flows increases as the number of flows increases. On the other hand, when the number of flows is large (32 to 64), the total throughput of the proposed schemes with "Round" and "Round-down" methods increases as the number of flows increases. A large number of flows will cause simultaneous receiving of packets on routers, so that it will increase the queue length on routers as shown in Fig. 4.8. It causes longer RTT as well as larger allocated window size of each flow. As a result, the total



(a) Round-up

(b) Round

(c) Round-down

Figure 4.7: Effect of the number of TCP clients: Total throughput

allocated bandwidth of all flows increases as the number of flows increases.

To analyze this phenomenon more deeply, I investigate the fairness index and total throughput performance of each group. The fairness index and total throughput for each group as functions of time are respectively shown in Figs. 4.9 and 4.10 for the proposed and conventional schemes, and those for all groups are shown in Fig. 4.11. Here, the proposed schemes employ the "Round-up" method, the number of TCP clients is set to 32, the notification interval time for flow information is set to 0.05 s, α is set to 0.4, and β is set to 0.8. In Group A", both schemes degrade the fairness index during the period from 10 to 15 s due to a heavy UDP flow as shown in Fig. 4.9(b), although the proposed schemes achieve

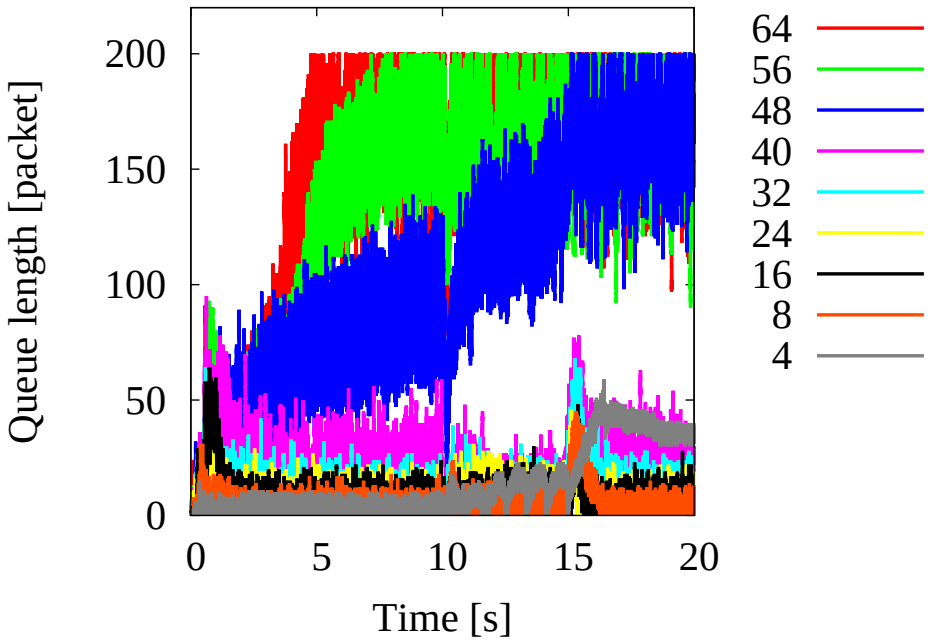
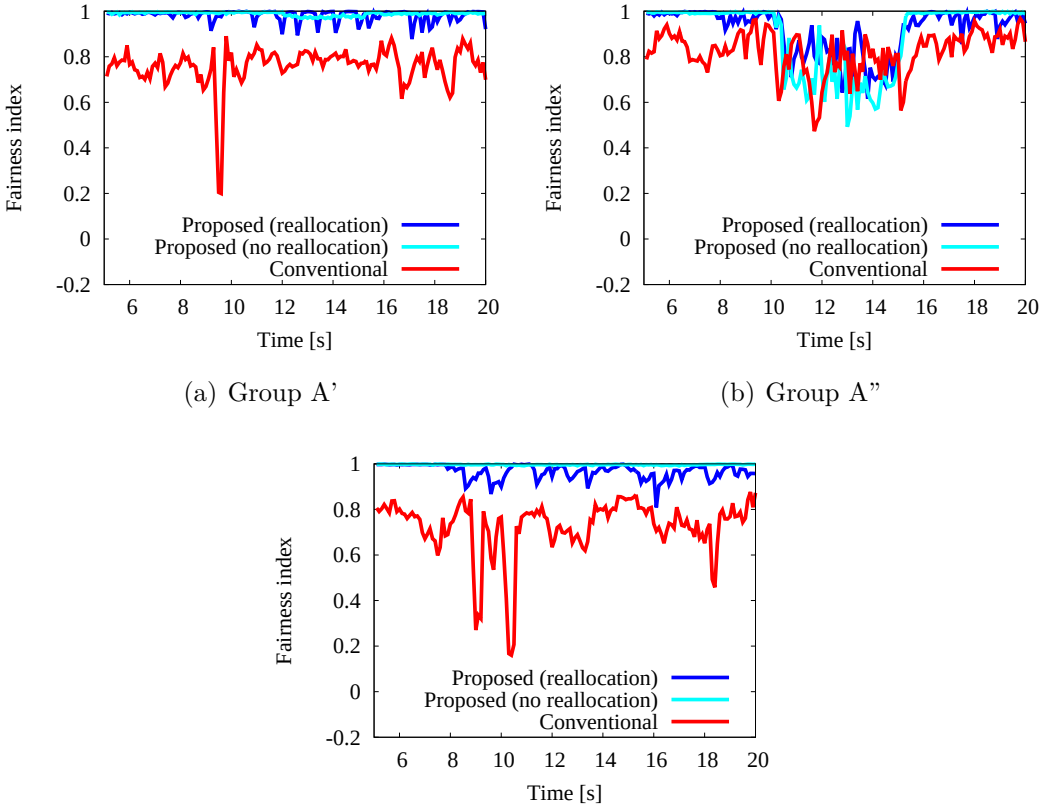


Figure 4.8: Effect of the number of TCP clients: Queue length

a good fairness index for other groups as shown in Fig. 4.9(a) and 4.9(c). The proposed schemes thus improve fairness in all groups as shown in Fig. 4.11(a). On the other hand, the proposed (no reallocation) scheme degrades the total throughput during the period from 10 to 15 s due to a heavy UDP flow, while the proposed (reallocation) scheme improves the total throughput (to almost the same as that of the conventional scheme) as shown in Fig. 4.11(b). Clearly, the proposed (reallocation) scheme can effectively reallocate the unutilized bandwidth in GroupA" flows to other group flows as shown in Fig. 4.10. Consequently, the proposed (reallocation) scheme achieves fair communication among flows which have different RTTs as well as high link utilization in DC networks.



(a) Group A'

(b) Group A"

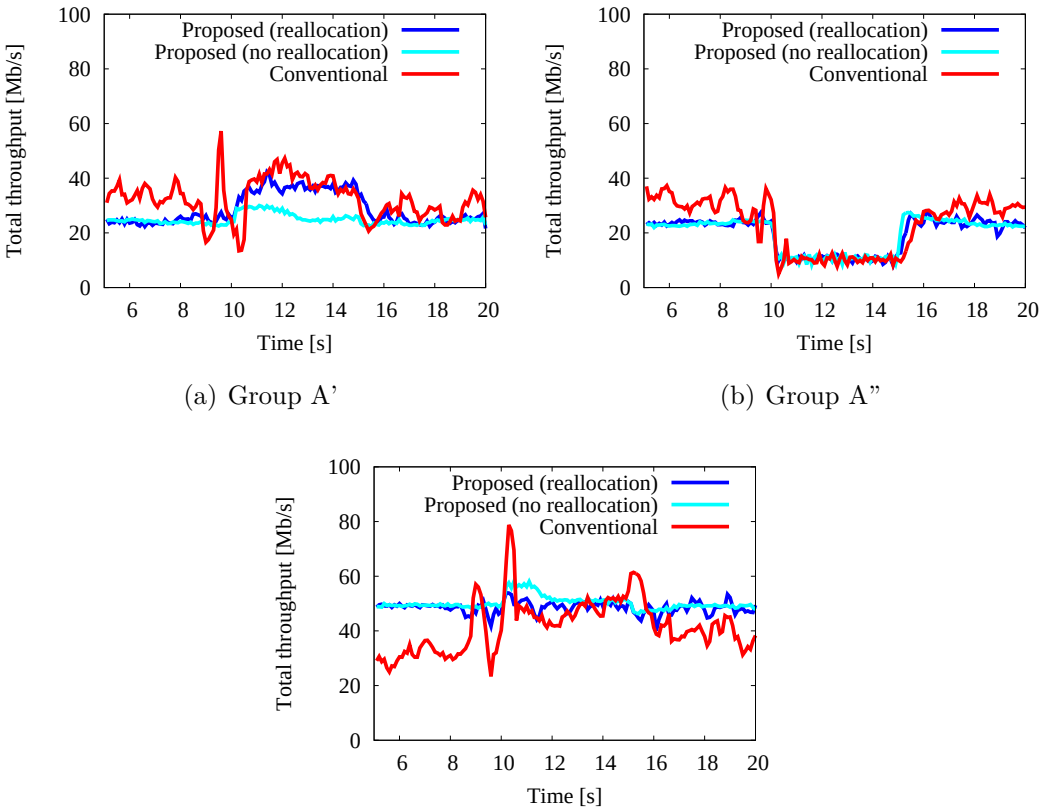
(c) Group B

Figure 4.9: Effect of the number of TCP clients: Fairness index of each group

4.4.2 Effect of notification interval time for flow information

Figures 4.12(a) and 4.12(b) respectively show the fairness index and total throughput of the proposed and conventional schemes when the notification interval time for flow information varies from 0.005 to 0.5 s. Here, the number of TCP clients is set to 32, α is set to 0.4, and β is set to 0.8.

In Fig. 4.12(a), the proposed schemes achieve a higher fairness index than the conventional scheme regardless of the notification interval time for flow information. On the other hand, the total throughput of the proposed (reallocation) scheme with the "Round-up" method is lower than that of the conventional



(a) Group A'

(b) Group A"

(c) Group B

Figure 4.10: Effect of the number of TCP clients: Total throughput of each group

scheme in particular when the notification interval time for flow information is from 0.005 to 0.01 s, as shown in Fig. 4.12(b). When the notification interval time for flow information is too short, whether the allocated bandwidth is fully utilized cannot be correctly determined since such a short notification interval time is less than the measurement interval time for the actual throughput of each flow. This causes unnecessary retransmissions as well as throughput degradation. When the notification interval time for flow information is from 0.05 to 0.5 s, the proposed (reallocation) scheme with the "Round-up" method achieves total throughput equal to that of the conventional scheme. Consequently, when the notification interval time for flow information is set to longer than 1 RTT,

the proposed (reallocation) scheme with the "Round-up" method achieves good fairness and total throughput.

4.4.3 Effect of weighting factor α

Figures 4.13(a) and 4.13(b) respectively show the fairness index and total throughput for the proposed and conventional schemes when the weighting factor α varies from 0.1 to 1.0. Here, the notification interval time for flow information is set to 0.05 s and β is set to 0.8.

In Fig. 4.13(a), as α approaches 0, the fairness index of the proposed (reallocation) scheme slightly decreases, while the total throughput does not change. This is because the proposed (reallocation) scheme allocates transmission rates to each flow using out-of-date flow information about the actual throughput of each flow. When the flow information is too old, the proposed (reallocation) scheme will reallocate unutilized bandwidth to other flows even if the flow can fully utilize the allocated transmission rate. This degrades fairness among flows. On the other hand, as α approaches 1, the fairness index and total throughput of the proposed (reallocation) scheme decreases. This is because the allocated transmission rate of each flow is not stable. Overall, the appropriate value of α is from 0.4 to 0.6 in this simulation.

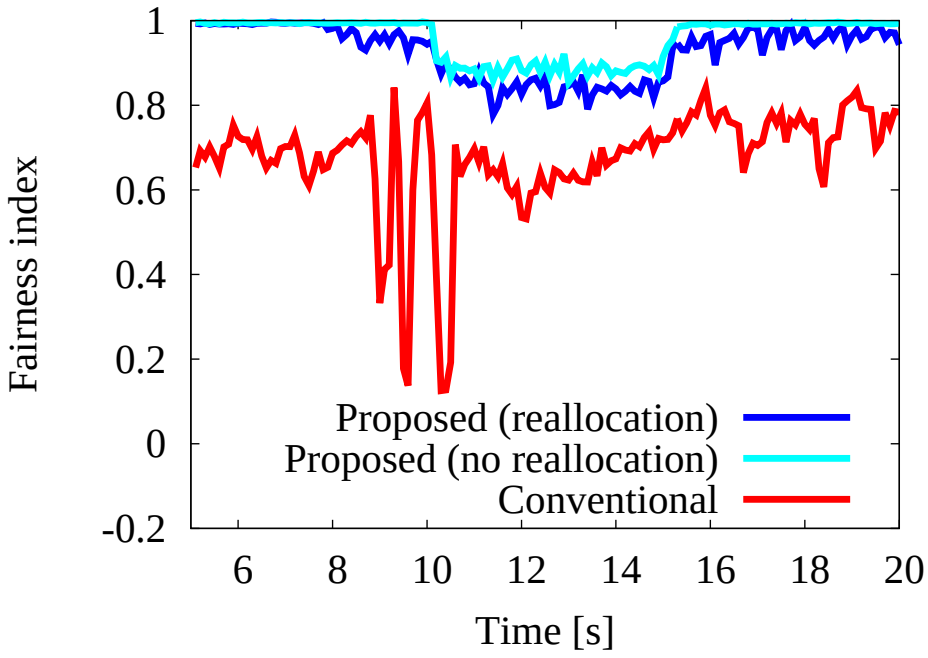
4.4.4 Effect of threshold factor β

Figures 4.14(a) and 4.14(b) respectively show the fairness index and total throughput of the proposed and conventional schemes when the threshold factor β varies from 0.1 to 1.0. Here, the notification interval time for flow information is set to 0.05 s and α is set to 0.4.

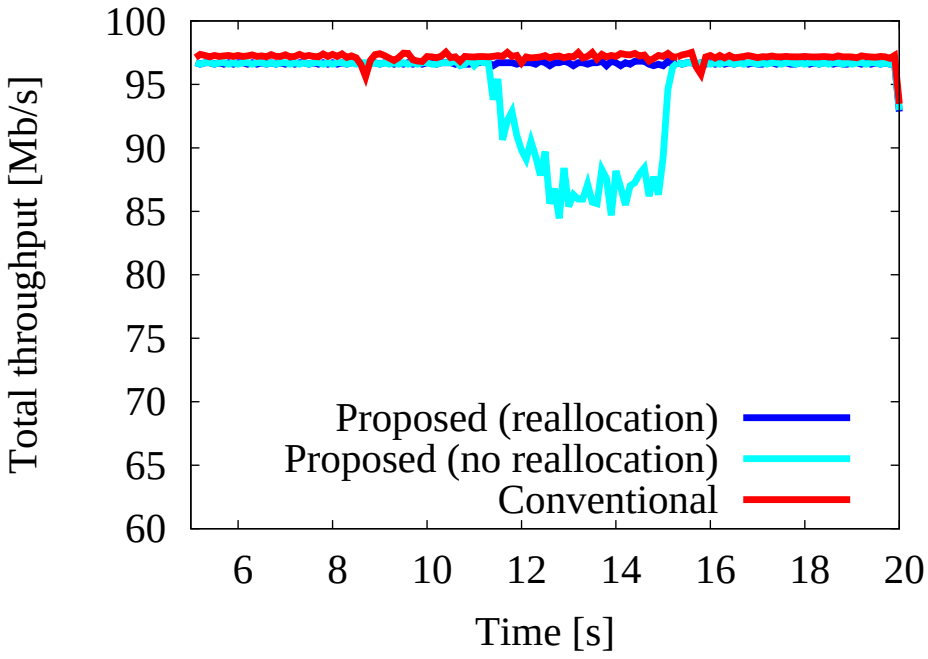
As β increases, the total throughput of the proposed (reallocation) scheme significantly increases, while the fairness index decreases. In particular, the fairness index of the proposed (reallocation) scheme is extremely degraded when β is larger than 0.8. This is because the allocated transmission rate for each flow is not stable when β is set to 1.0. Consequently, the proposed (reallocation) scheme achieves a high fairness index as well as total throughput almost the same as that of the conventional scheme when β is set from 0.6 to 0.8 in this environment.

4.5 Conclusion

In this study, I have proposed a bandwidth allocation scheme based on collectable information to improve fairness and link utilization in DC networks. This scheme collects flow information including the bandwidth of each link, the number of competing flows, the RTT of each flow, and the actual throughput of each flow from routers and servers in DC networks, and then it fairly allocates transmission rates among flows based on the collected information. In addition, this scheme reallocates unutilized bandwidth to other competing flows in DC networks when bottleneck links exist outside of the DC networks. Simulations indicated that the proposed scheme enables fair communication among flows which have different RTTs as well as high link utilization in DC networks by setting appropriate parameter values.

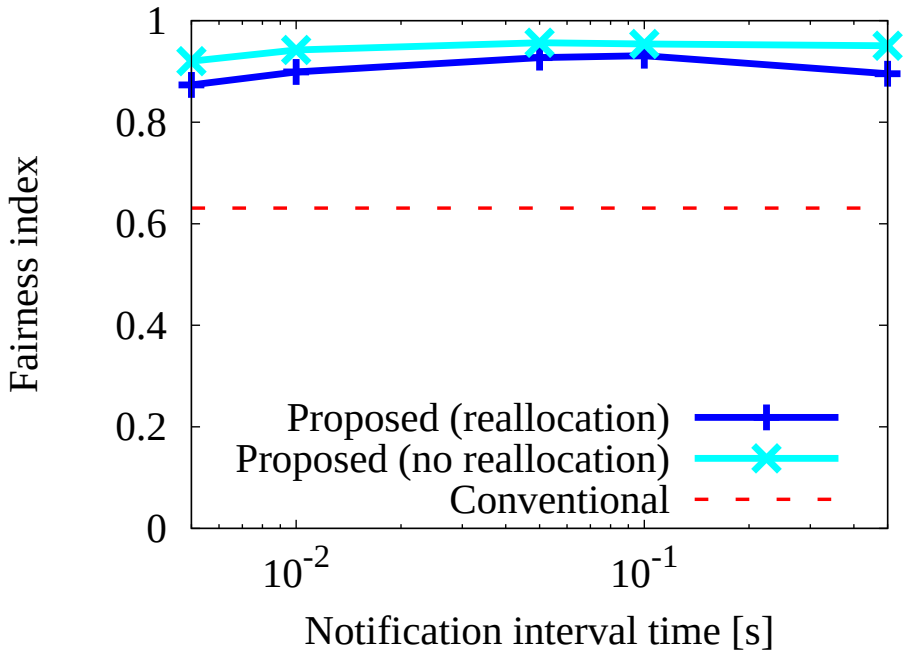


(a) Fairness index

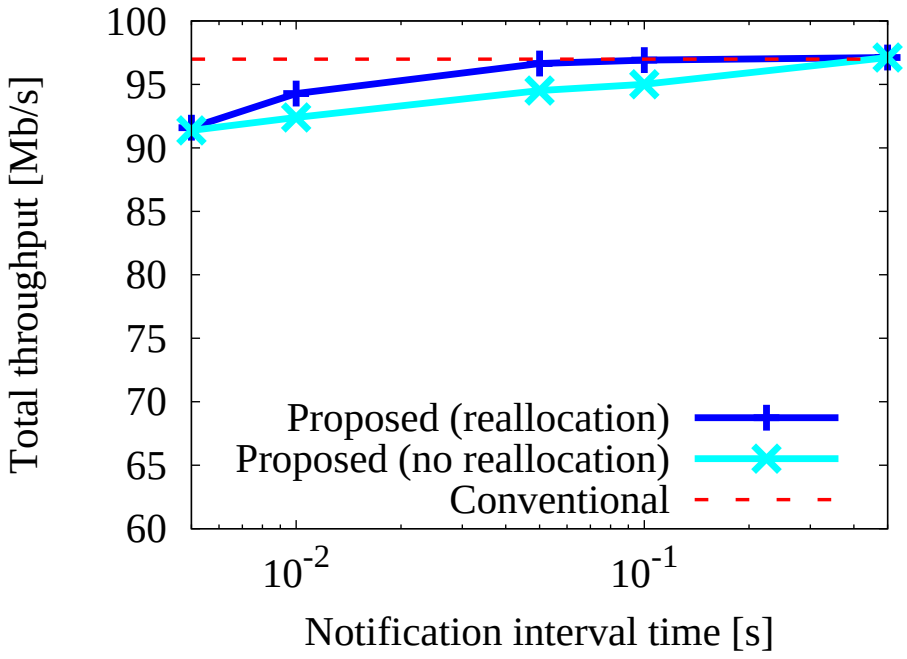


(b) Total throughput

Figure 4.11: Effect of the number of TCP clients: All groups

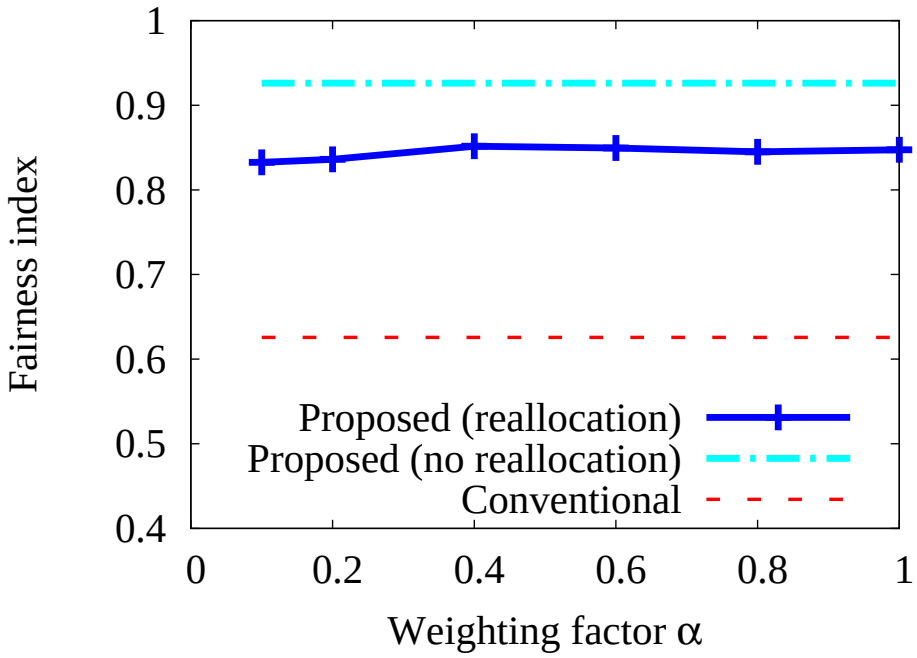


(a) Fairness index

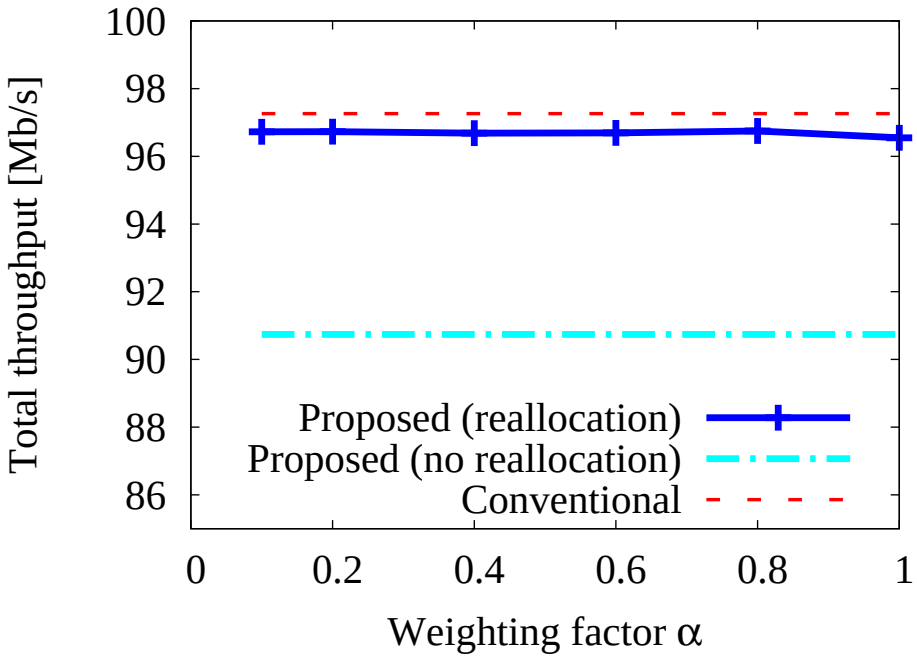


(b) Total throughput

Figure 4.12: Effect of notification interval time

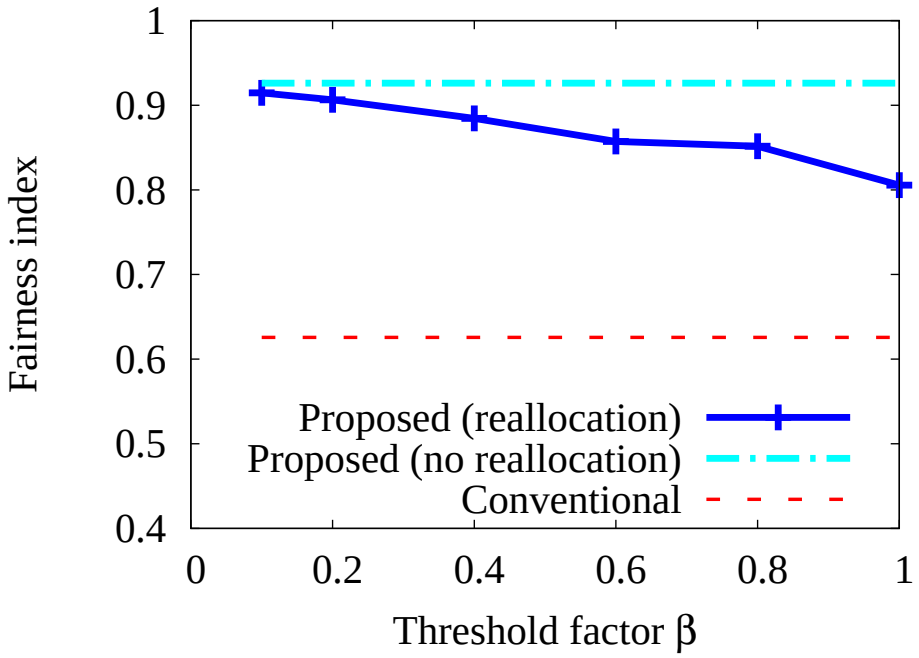


(a) Fairness index

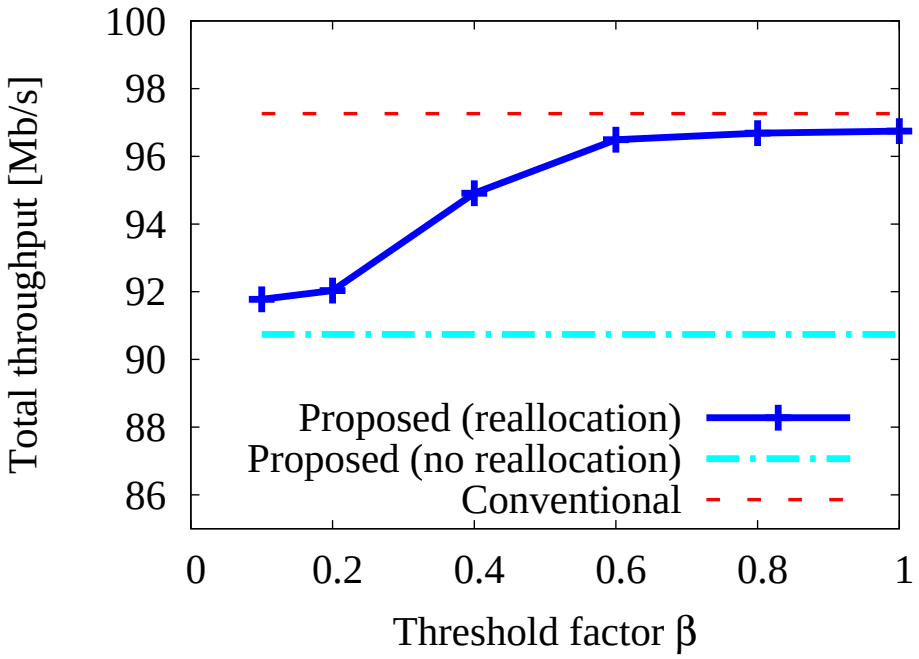


(b) Total throughput

Figure 4.13: Effect of weighting factor α



(a) Fairness index



(b) Total throughput

Figure 4.14: Effect of threshold factor β

5 A Bandwidth Allocation Scheme to Meet Flow Requirements in Edge Computing

5.1 Introduction

Users can now connect to the Internet through a wide variety of access networks and communication terminals such as smart phones or PCs. This has created a great demand for cloud computing, which allows users to enjoy various Internet services provided by data centers (DCs) [39] at anytime and anywhere. A major issue regarding cloud computing, however, is communication latency because DCs will often be far from users. A long distance between users and DCs degrades the quality of latency-sensitive services. To solve this cloud-computing distance problem, edge computing, which locates edge servers on access networks, has been proposed [45]. The edge servers have computation and storage capabilities in addition to their packet forwarding capability, and these capabilities allow them to provide lower latency and higher-speed communication services compared to what is possible solely through DCs.

Edge computing will enable future low latency services, such as augmented reality and automatic driving, while DCs will continue to provide conventional services, such as file sharing and streaming delivery. In other words, various services with different round-trip times (RTTs) will coexist in edge computing as illustrated in Fig. 5.1. In this situation, an RTT-unfairness problem arises because such services commonly use TCP as a reliable data transmission protocol. Since

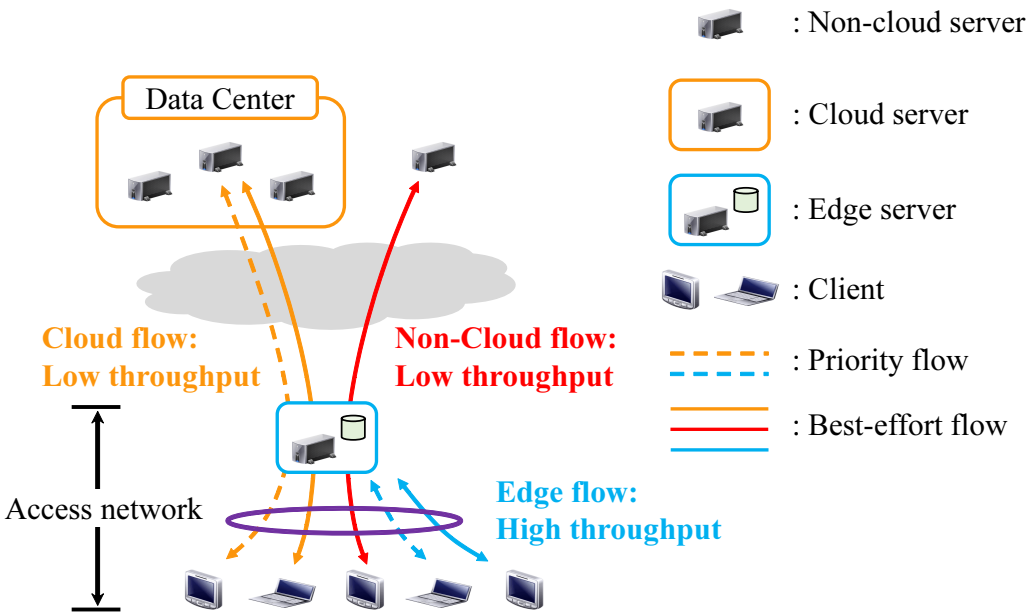


Figure 5.1: Unfairness problem in edge computing

TCP's congestion control is applied according to RTT, the transmission rate for a flow with a short RTT is effectively increased compared with that of a flow with a long RTT. Moreover, the wide variety of services in edge computing will have different requirements, including their required transmission rates. Therefore, the throughput of each flow will need to be controlled according to its requirements and independently of RTT.

In this context, I previously proposed a bandwidth allocation scheme based on collectable information in DC networks to improve fairness and link utilization [13–16]. This previous study tackles the RTT-unfairness problem in cloud computing. In the study reported here, I propose an extended bandwidth allocation scheme based on collectable information to meet the requirements of each flow in edge computing. This scheme collects flow information — including the bandwidth of each link, the number of competing flows, the RTT of each flow, and the actual throughput of each flow — and then allocates transmission rates for each flow based on the collected information. In addition, this scheme reallocates unutilized bandwidth to other competing flows in access networks when

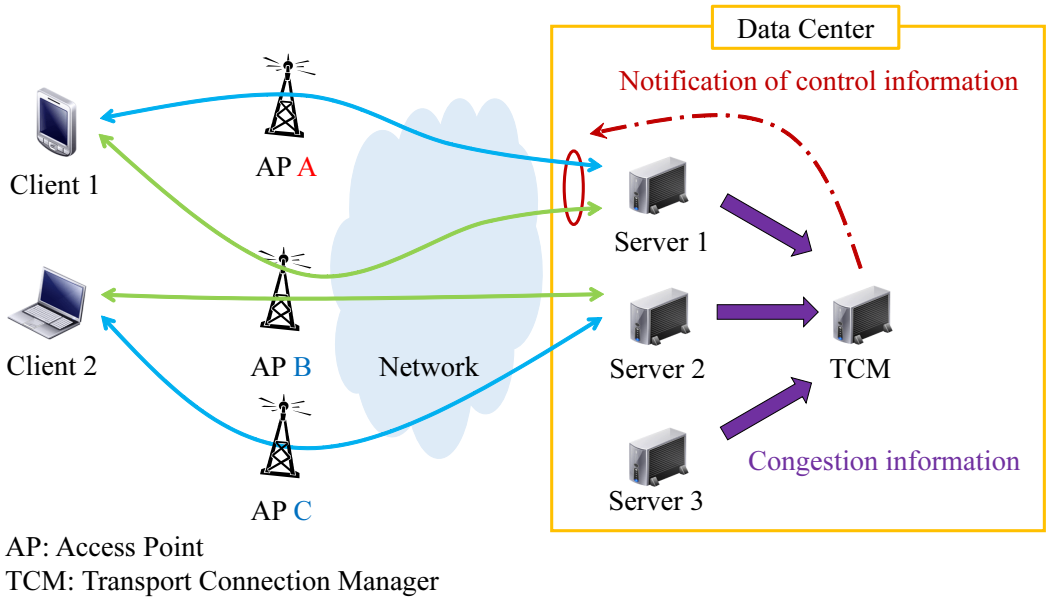


Figure 5.2: Unified central congestion control architecture

bottleneck links exist in core networks. I show the effectiveness of this approach through simulation evaluations.

5.2 Proposed scheme

I propose a bandwidth allocation scheme based on collectable information to meet the requirements of each flow in edge computing. Since the proposed scheme is an extension of this previous work, I will first explain the key points of the previous work, and then give an overview and explain the operation of the proposed scheme.

5.2.1 Previous work

I previously proposed a bandwidth allocation scheme based on collectable information in DC networks to improve fairness and link utilization. This scheme fairly allocates transmission rates among flows based on collectable information in DC networks. In addition, this scheme reallocates any unutilized bandwidth

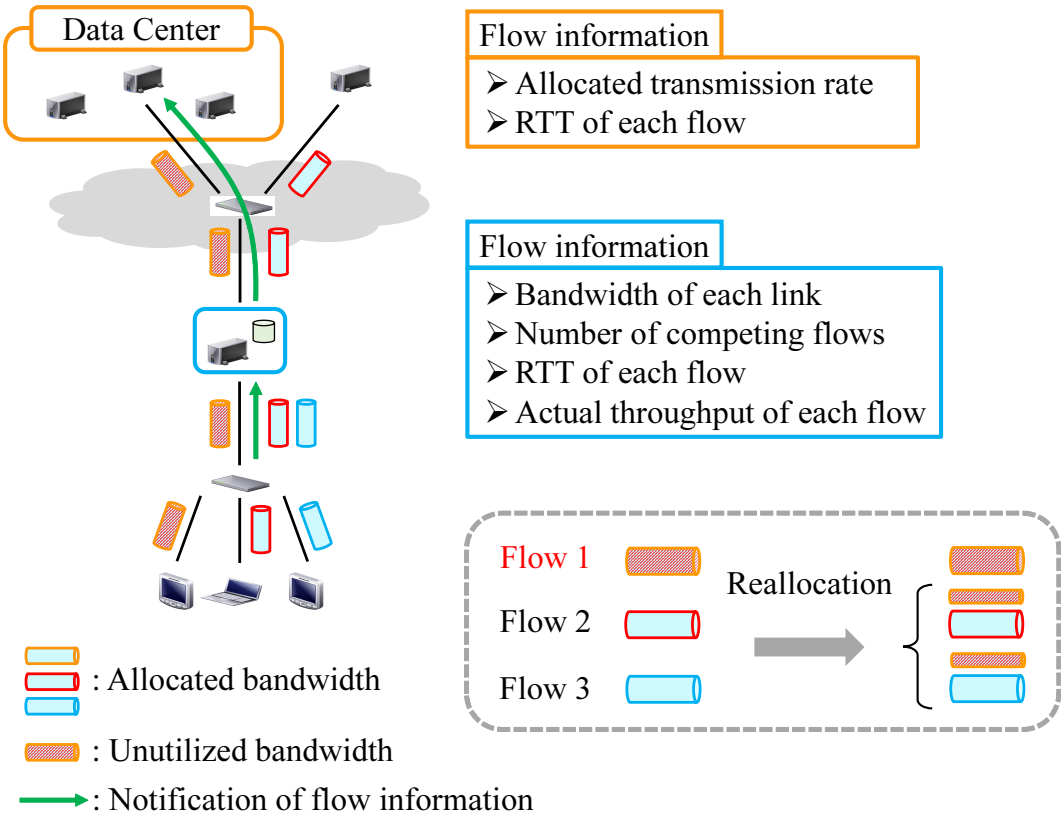


Figure 5.3: Overview of proposed scheme

caused by congestion outside DC networks to other competing flows to improve link utilization in DC networks when bottleneck links exist outside the DC networks. Flow information such as the bandwidth of each link, the number of competing flows, the RTT of each flow, and the actual throughput of each flow is collected by a Unified Central Congestion Control Architecture (UC^3) [40], which uniformly manages the congestion information in the networks, as shown in Fig. 5.2. This scheme identifies the bottleneck link based on the flow information collected from routers and servers in DC networks and then calculates the transmission rate which should be allocated to each flow by dividing the bandwidth of the bottleneck link by the number of competing flows. Namely, this scheme enables fair communication among flows which have different RTTs as

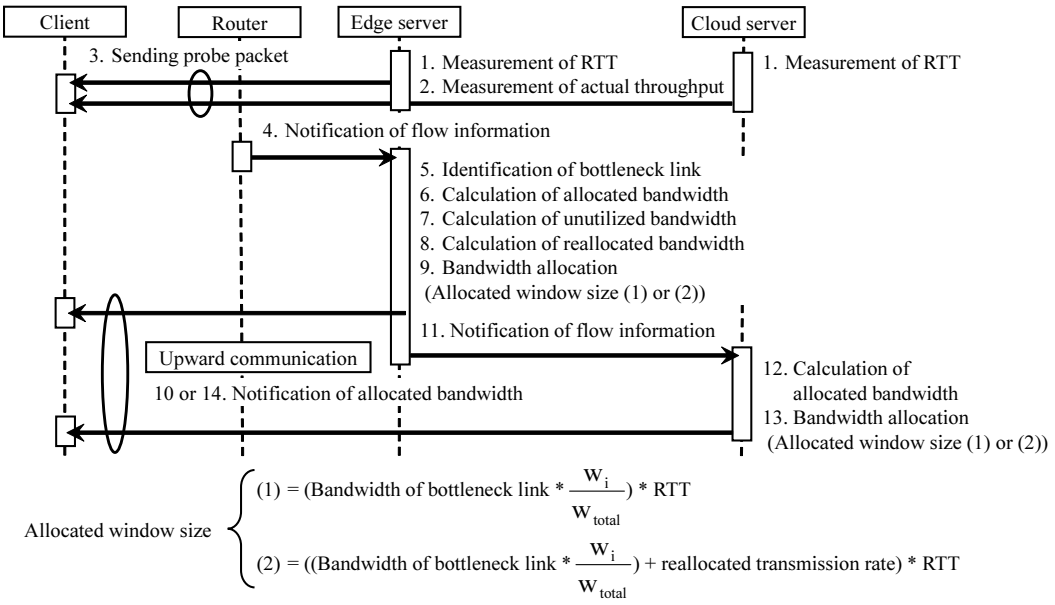


Figure 5.4: Operation of proposed scheme

well as high link utilization in cloud computing.

5.2.2 Overview of the proposed scheme

In edge computing, users can receive latency-sensitive services from edge servers in addition to conventional services from cloud servers. That is, various flows with different RTTs, including edge flows with a short RTT and cloud flows with a long RTT, will coexist. In addition, each flow commonly has different requirements in terms of transmission rates. Therefore, I need to control the transmission rate of both edge and cloud flows in consideration of service classes. To do this, I propose a bandwidth allocation scheme based on collectable information in edge and cloud servers to meet the requirements of each flow by extending this previous scheme. This scheme collects flow information — such as the bandwidth of each link, the number of competing flows, the RTT of each flow, and the actual throughput of each flow — and then allocates transmission rates among flows according to their requirements based on the collected information (Fig. 5.3). In addition, this scheme reallocates unutilized bandwidth to other competing flows in access

networks when bottleneck links exist in core networks.

5.2.3 Operation of the proposed scheme

I describe here the operation of the proposed scheme shown in Fig. 5.4. The edge server measures the RTT of each edge flow and actual throughput of each flow, while the cloud server measures the RTT of each cloud flow. This measurement is done through normal TCP operation. If the edge or cloud server does not send any data to a client, it periodically sends probe packets to measure RTT. The edge server also measures the RTT of each non-cloud flow which communicate with servers not in DC networks. This can be done based on the conventional RTT estimation technique [46]. Routers in access networks periodically inform edge servers of the number of existing flows, which can be measured by extending the OpenFlow architecture [41]. The edge server identifies the bottleneck link based on the information about the number of existing flows received from the routers and the bandwidth of each link, and then calculates the transmission rate which should be allocated to each flow by multiplying the bandwidth of the bottleneck link by w_i / w_{total} which means the number of competing flows with respect to service classes, where w_i is a weight value for flow i and w_{total} is a total weight value of competing flows. Note that w_i will be decided in advance. If actual throughput of each flow multiplied by a threshold factor α is smaller than the allocated transmission rate, the edge server calculates the transmission rate which should be reallocated to other competing flows. The reallocated transmission rate is calculated by dividing the total unutilized bandwidth of flows by the number of other competing flows which fully utilize the allocated transmission rate. Namely, the allocated transmission rate for each flow which cannot fully utilize the allocation is the bandwidth of the bottleneck link multiplied by w_i / w_{total}. On the other hand, the allocated transmission rate for each flow which fully utilizes the allocation is added to the reallocated transmission rate. The edge server adds flow information about the calculated transmission rates to the received packets from the client to the cloud server in order to inform the cloud server of this information.

To allocate transmission rates to TCP flows with window-based congestion control, the bandwidth delay product (BDP) of each flow should be calculated

as the allocated window size based on the bandwidth which should be allocated and the RTT of each flow. When the BDP of each flow cannot be divided by the maximum segment size (MSS), the allocated transmission rate becomes a little bit low, so the BDP allocated to each flow is rounded up by the MSS units. For downwards communication from an edge or cloud server to a client, the server can simply allocate the calculated window size to each flow. On the other hand, for upwards communication from a client to an edge or cloud server, the server needs to inform the client of the allocated window size. This scheme uses TCP's advertised window size to notify the client of the window size. For non-cloud flows, the edge server rewrites the window size field of the received ACK packets from the client by the allocated window size and then forwards them to the non-cloud server based on a snooping technique [47]. Through these procedures in edge computing, this scheme can allocate an adequate transmission rate to each flow according to its requirements independently of RTT.

5.3 Simulation model

To investigate the efficiency of the proposed scheme in edge computing, I evaluated it after its implementation through simulation using Network Simulator ns-3 [43]. I implemented functions to measure the number of competing flows and actual throughput of each flow, to identify the bottleneck links, to calculate the transmission rates allocated to each flow, and to report the flow information in this simulator.

Figure 5.5 shows the simulation topology. In this simulation, each client continuously sends data to the corresponding servers using TCP NewReno after the simulation starts. Note that proposed scheme works independently of TCP congestion control algorithms, so I employ TCP NewReno as a simple typical TCP. The propagation delay time of access links is set to a uniform random number which ranges from 0.5 to 1 ms for the links between clients and routers or 2 ms for the link between a router and an edge server, while the bandwidth of access links is set to 50 Mb/s. The core links have a propagation delay time of 5 ms and a bandwidth of 100 Mb/s. Other simulation parameters are summarized in Table 5.1.

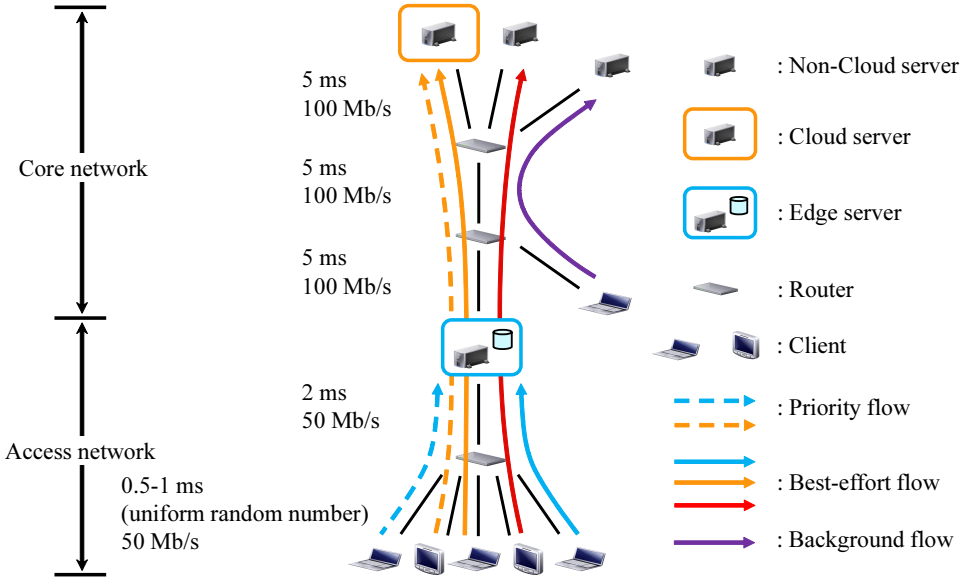


Figure 5.5: Simulation model

5.3.1 Simulation scenario

I explain here the simulation scenario. In this simulation, 5 TCP clients communicate with the server after the simulation starts. At 5 seconds after the simulation starts, one background flow joins through routers in core networks and then leaves at 10 seconds. Namely, the core link becomes a bottleneck during this period from 5 to 10 s, while the access link becomes a bottleneck in the other periods.

5.3.2 Evaluation indices

I evaluated throughput performance for the proposed scheme and conventional TCP as well as accuracy of proposed bandwidth allocation. For throughput performance, I investigate throughput of each flow and total throughput of all flows. These throughputs are calculated at intervals of 1.0 s. On the other hand, for the accuracy of proposed bandwidth allocation, I investigate an average error rate for target or allocated transmission rates. The target transmission rate is the transmission rate that should be originally allocated according to a weight

Table 5.1: Simulation parameters

TCP algorithm	NewReno
Measurement interval time for actual throughput of each flow	0.1 [s]
Notification interval time for flow information	$0.005 \sim 0.5$ [s]
Threshold factor α	$0.5 \sim 1.0$
Weight value of the best-effort flow	1
Weight value of the priority flow	2
Transmission rate of background flow	85 [Mb/s]
Simulation time	15 [s]
Number of trials	10

value of each flow, while the allocated transmission rate includes the reallocated one in addition to the target one. The error rate ER_i of each flow i is defined as Eq. (5.1), where x_i and $\hat{x_i}$ are the target or allocated transmission rates and the actual throughput of flow i, respectively. The ER_i is calculated at intervals of 1.0 s and averaged during the simulation time.

$$ER_i = \left| \frac{x_i - \hat{x_i}}{x_i} \right| \tag{5.1}$$

The average error rate AER is calculated as Eq. (5.2), where n is the number of competing flows on the same link.

$$AER = \frac{\sum_{i=1}^{n} ER_i}{n} \tag{5.2}$$

The accuracy of proposed bandwidth allocation is higher as the AER gets closer to 0.

5.3.3 Comparison schemes

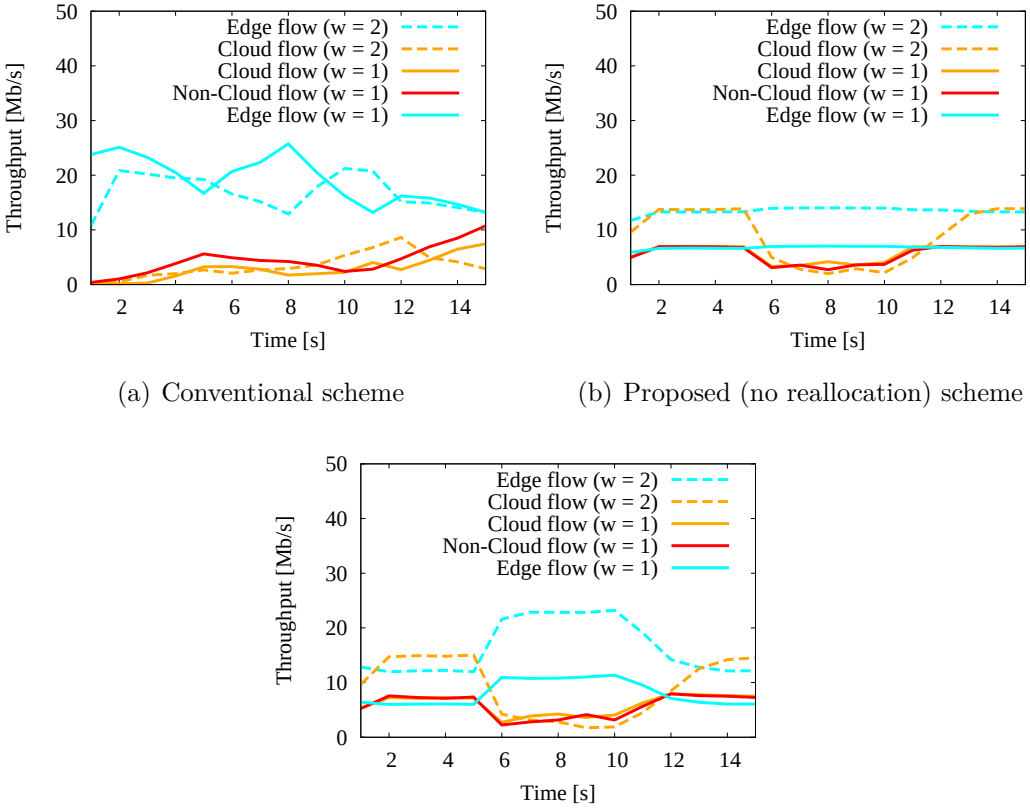
I evaluate the performance of the proposed schemes (Proposed (no reallocation) and Proposed (reallocation)) compared with that of conventional TCP (Conventional) with the throughput performance. The Proposed (no reallocation) scheme allocates a transmission rate to each flow according to its weight value, while the Proposed (reallocation) scheme additionally allocates unutilized bandwidth to competing flows which fully utilize the allocated bandwidth.

5.4 Simulation results

In this section, I now show simulation results and discuss the effectiveness of the proposed schemes as compared with the conventional scheme. I first investigate the throughput performance of the proposed and conventional schemes and then discuss the effect of the parameters of proposed schemes, such as the notification interval time for flow information and threshold factor α.

5.4.1 Throughput performance

Figures 5.6 and 5.7 respectively show the throughput performance as a function of the time when the proposed and conventional schemes are applied. Here, the notification interval time for flow information is set to 0.05 s and α is set to 0.95. In Fig. 5.6, the edge flows with a short RTT achieves higher throughput than the cloud and non-cloud flows with a long RTT under the conventional scheme. This is because TCP's congestion control depends on RTT. On the other hand, under the proposed scheme, each flow obtains the suitable throughput according to its weight value independently of RTT. This result indicates that the proposed scheme can allocate transmission rates among flows according to their requirements independently of each flow's RTT. In Fig. 5.7, the proposed (reallocation) scheme has significantly improved the total throughput as compared with the proposed (no reallocation) scheme during the period when bottleneck links exist in core networks. The proposed (reallocation) scheme precisely allocates the calculated BDP to each flow as the allocated transmission rates based on the bandwidth which should be allocated and the measured RTT of each flow. It



(a) Conventional scheme

(b) Proposed (no reallocation) scheme

(c) Proposed (reallocation) scheme

Figure 5.6: Throughput of each flow

causes a remaining bandwidth which cannot be fully utilized. This problem can be solved by allocating a little bit large BDP to each flow and will be discussed in future work.

5.4.2 Effect of notification interval time

Figures 5.8(a) and 5.8(b) respectively show the average error rate for allocated transmission rate of proposed (reallocation) scheme and the total throughput of proposed and conventional schemes when the notification interval time for flow information varies from 0.005 to 1.0 s. Here, α is set to 0.95. When the notification interval time for flow information is smaller than 0.1 s, the proposed

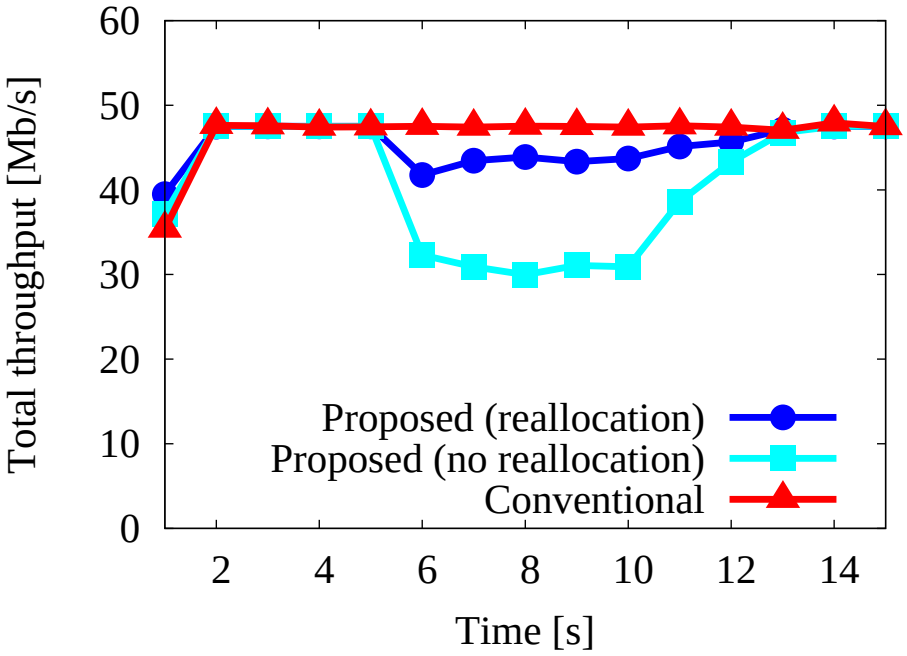


Figure 5.7: Total throughput of all flows

(reallocation) scheme achieves low average error rate as well as total throughput almost the same as that of conventional scheme, as shown in Fig. 5.8. On the other hand, when the notification interval time for flow information is larger than 0.1 s, the average error rate of the proposed (reallocation) scheme is extremely high although high total throughput is achieved. Consequently, the proposed (reallocation) scheme enables accurate bandwidth allocation among flows as well as efficient transmission by setting the notification interval time for flow information to appropriate small value.

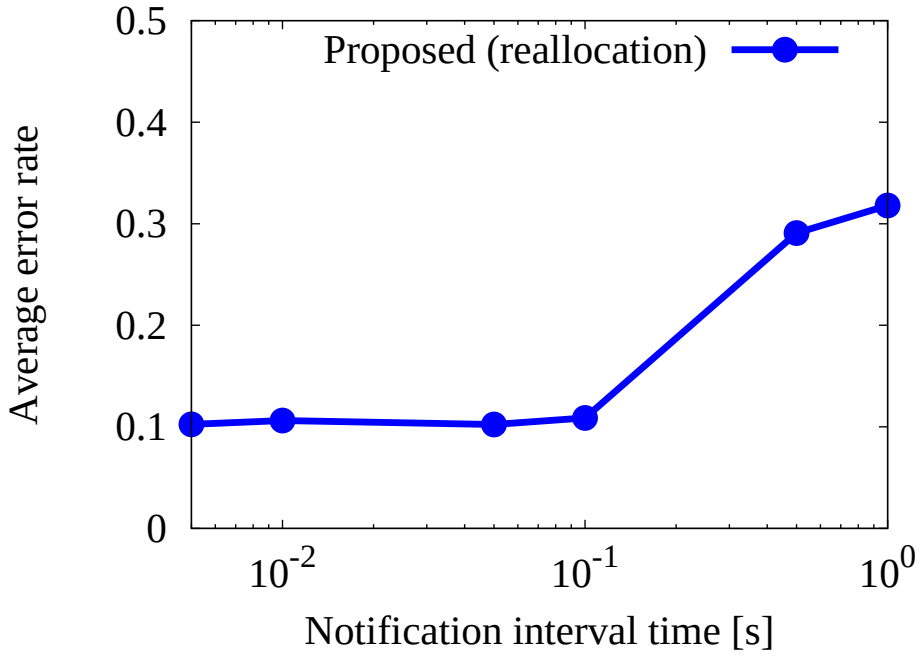
5.4.3 Effect of threshold factor α

Figures 5.9(a) and 5.9(b) respectively show the average error rate for target transmission rate of proposed (reallocation) scheme and the total throughput of proposed and conventional schemes when the threshold factor α varies from 0.5 to 1.0. Here, the notification interval time for flow information is set to 0.05 s. In Fig. 5.9, the proposed (reallocation) scheme achieves better average error rate

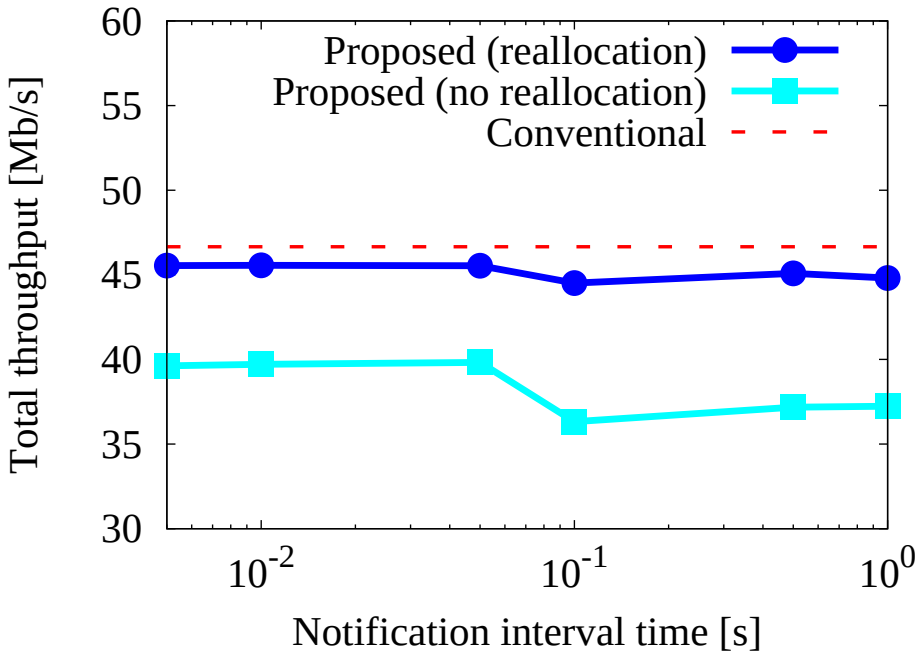
and lower total throughput as α is smaller, while it achieves worse average error rate and higher total throughput as α is larger. Namely, the proposed (reallocation) scheme has a trade-off relationship between the average error rate and total throughput for α. In this simulation, α should be set to approximately 0.85 if I consider the allocated transmission rate gets closer to the target one, while it should be set to approximately 0.95 if the total throughput is emphasized. Consequently, the proposed (reallocation) scheme achieves low average error rate and high total throughput by setting α to slightly smaller than 1.

5.5 Conclusion

In this study, I have proposed a bandwidth allocation scheme based on collectable information to meet the requirements of each flow in edge computing. This scheme collects flow information including the bandwidth of each link, the number of competing flows, the RTT of each flow, and the actual throughput of each flow from routers in access networks, and then it allocates transmission rates for each flow based on the collected information. In addition, this scheme reallocates unutilized bandwidth to other competing flows in access networks when bottleneck links exist in core networks. Simulation evaluations have indicated that the proposed scheme can allocate transmission rates among flows according to their requirements independently of each flow's RTT as well as achieve high link utilization. In future work, I will consider an effective algorithm to allocate unutilized bandwidth to other competing flows in a more realistic environment.

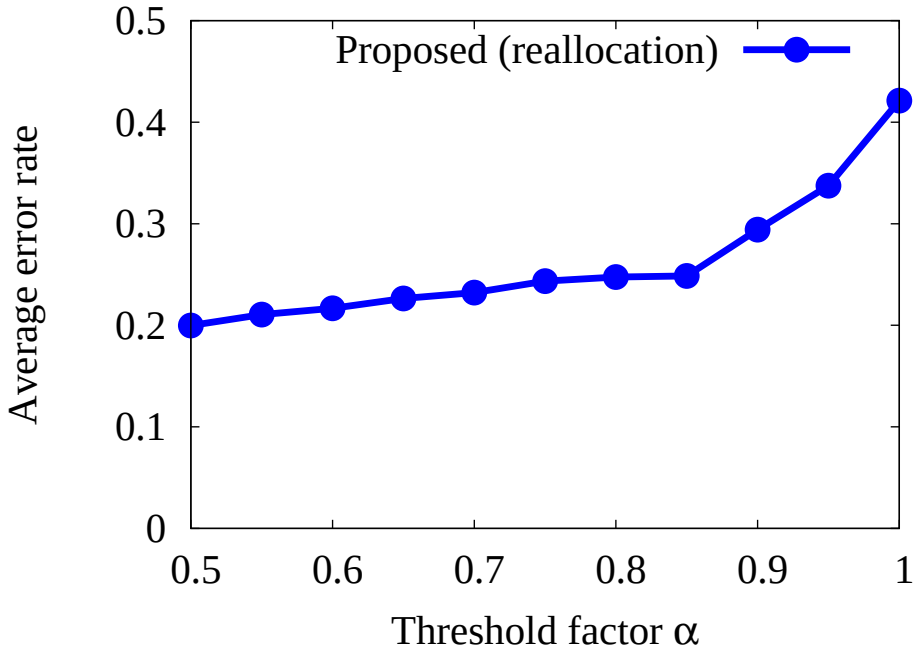


(a) Average error rate for allocated transmission rate

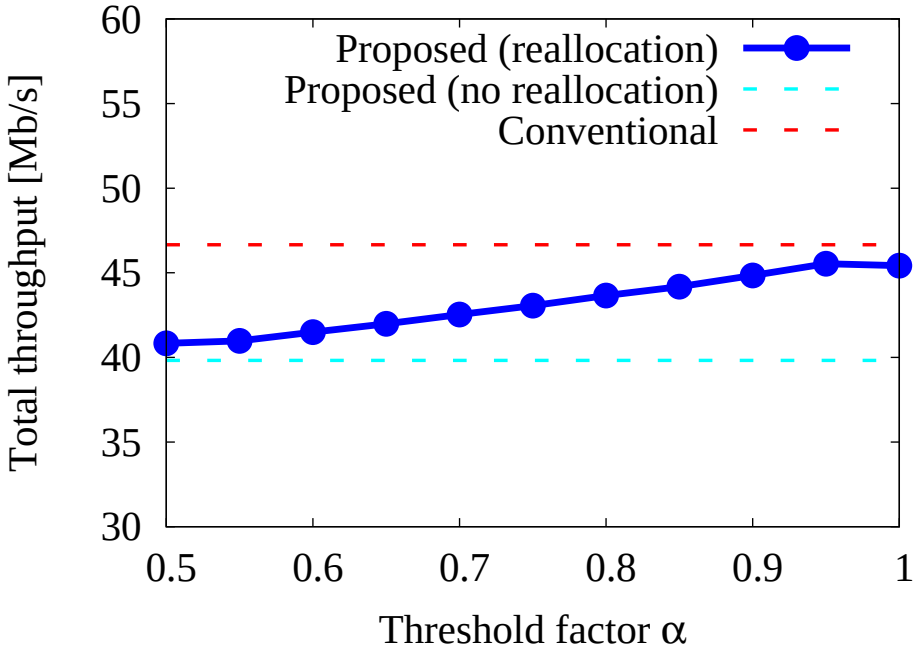


(b) Total throughput

Figure 5.8: Effect of notification interval time



(a) Average error rate for target transmission rate



(b) Total throughput

Figure 5.9: Effect of threshold factor α

6 A Bandwidth Allocation Scheme Based on Residual Bandwidth Information in Edge Computing

6.1 Introduction

Cloud computing, which enables users to enjoy various Internet services provided by data centers (DCs) at anytime and anywhere, has attracted much attention. A major issue regarding cloud computing, however, is communication latency because DCs will often be far from users. A long distance between users and DCs degrades the quality of latency-sensitive services. To solve this cloud-computing distance problem, edge computing, which locates edge servers on access networks, has been proposed [45]. The edge servers have computation and storage capabilities in addition to their packet forwarding capability, and these capabilities allow them to provide lower latency and higher-speed communication services compared to what is possible solely through DCs.

Edge computing will enable future low latency services, such as augmented reality and automatic driving, while DCs will continue to provide conventional services, such as file sharing and streaming delivery. In other words, various services with different round-trip times (RTTs), including edge flows with a short RTT and cloud flows with a long RTT, will coexist in edge computing. In this situation, an RTT-unfairness problem arises because such services commonly use TCP as a reliable data transmission protocol. Since TCP's congestion control is applied according to RTT, the transmission rate for a flow with a short RTT is

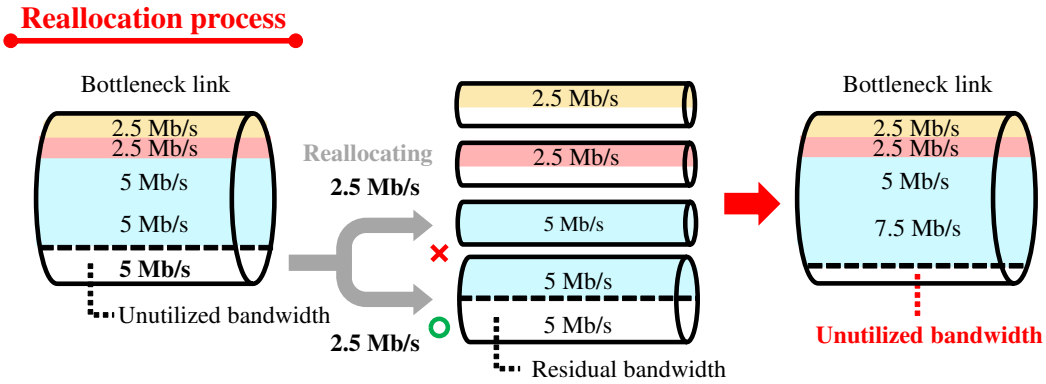


Figure 6.1: Degradation of link utilization in access networks

effectively increased compared with that of a flow with a long RTT. Moreover, the wide variety of services in edge computing will have different requirements, including their required transmission rates. Therefore, the throughput of each flow will need to be controlled according to its requirements and independently of RTT.

Several studies on edge computing technology have been reported [30, 48], but these studies have mainly focused on schemes to effectively use edge servers in edge computing. On the other hand, to generally improve fairness among flows with different RTTs, active queue management (AQM) algorithms to be applied on intermediate routers have been proposed [24–26, 49]. These AQM algorithms on routers can improve fairness among competing flows if the flows share a bottleneck link, but it is difficult to apply them in edge computing where such bottleneck links shared by all flows do not exist.

To meet the requirements of each flow in edge computing, I previously proposed a bandwidth allocation scheme based on collectable information and showed the effectiveness of this approach in edge computing [16, 17]. However, this scheme may degrade link utilization in access networks if it reallocates an unutilized bandwidth to flows through paths without a sufficient bandwidth as shown in Fig. 6.1. In this study, I propose a bandwidth allocation scheme based on residual bandwidth information to improve link utilization in edge computing. I show the effectiveness of this approach through simulation evaluations.

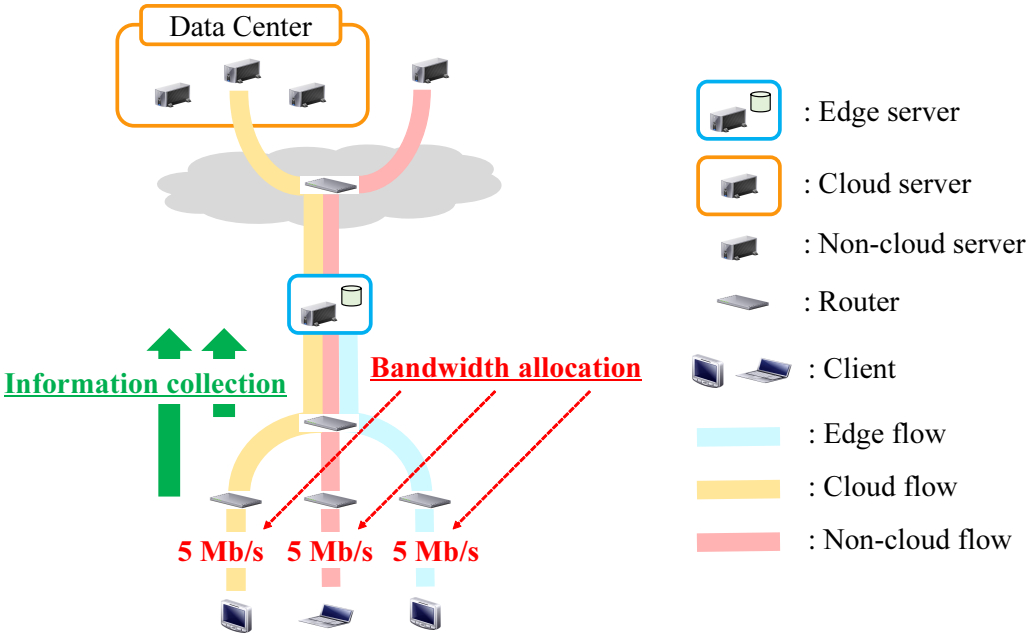


Figure 6.2: Overview of proposed scheme

6.2 Proposed scheme

This scheme collects flow information — such as the bandwidth of each link, the number of competing flows, the RTT of each flow, the actual throughput of each flow, and the residual bandwidth of each link — and then allocates transmission rates among flows according to their requirements based on the collected information as shown in Fig. 6.2. In addition, this scheme reallocates unutilized bandwidth to other competing flows with residual bandwidth in access networks when bottleneck links exist in core networks (Fig. 6.3).

I describe here the operation of the bandwidth allocation scheme. The edge server measures the RTT and actual throughput of each flow. This measurement is done through normal TCP operation. For cloud and non-cloud flows, RTT can be measured based on the conventional RTT estimation technique [46]. Edge routers in access networks periodically send a packet with information about the number of existing flows and the residual bandwidth of each link to edge servers. When an edge router in access networks receives the packet from lower routers,

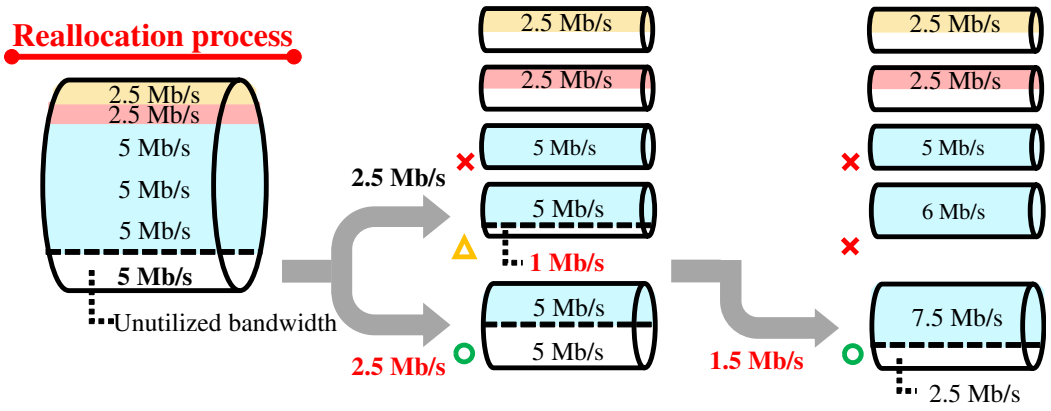


Figure 6.3: Bandwidth allocation based on residual bandwidth information

it rewrites the residual bandwidth information in the received packet to the measured one if the measured residual bandwidth is smaller than the received one, and then forwards the packet to upper routers. The edge server identifies the bottleneck link based on the information about the number of existing flows received from the routers and the bandwidth of each link, and then calculates the transmission rate which should be allocated to each flow by multiplying the bandwidth of the bottleneck link by w_i / w_{total} which means the number of competing flows with respect to service classes, where w_i is a weight value for flow i and w_{total} is a total weight value of competing flows. Note that w_i will be decided in advance. If actual throughput of each flow multiplied by a threshold factor α is smaller than the allocated transmission rate, the edge server calculates the transmission rate which should be reallocated to other competing flows. First, the edge server calculates the total reallocated transmission rate among competing flows through the same edge router in access networks. The total reallocated transmission rate for each edge router is calculated by multiplying the total unutilized bandwidth by w'_{edge} / w'_{total}, where w'_{total} is a total weight value of competing flows which fully utilize the allocated transmission rate and w'_{edge} is a total weight value of competing flows through each edge router among them. It is limited to the residual bandwidth if it exceeds it. Second, the reallocated transmission rate for each flow is calculated by multiplying the total reallocated transmission rate by w'_i

/ w'_{edge}, where w'_i is a weight value for flow i which fully utilizes the allocated transmission rate. These procedures will be repeated until the total unutilized bandwidth or residual bandwidth is exhausted. Namely, the allocated transmission rate for each flow which cannot fully utilize the allocation is the bandwidth of the bottleneck link multiplied by w_i / w_{total}. On the other hand, the allocated transmission rate for each flow which fully utilizes the allocation is added to the reallocated transmission rate.

To allocate transmission rates to TCP flows with window-based congestion control, the bandwidth delay product (BDP) of each flow should be calculated as the allocated window size based on the bandwidth which should be allocated and the RTT of each flow. When the BDP of each flow cannot be divided by the maximum segment size (MSS), the allocated transmission rate becomes a little bit low, so the BDP allocated to each flow is rounded up by the MSS units. This scheme employs an allocation factor β and the allocated window size is multiplied by β. For downwards communication from an edge server to a client, the edge server can simply allocate the calculated window size to each flow. On the other hand, for upwards communication from a client to an edge server, the edge server needs to inform the client of the allocated window size. This scheme uses TCP's advertised window size to notify the client of the window size. For cloud or non-cloud flows, the edge server rewrites the window size field of the received ACK packets from the cloud or non-cloud server by the allocated window size and then forwards them to the client based on a snooping technique [47]. Through these procedures in edge computing, this scheme can allocate an adequate transmission rate to each flow according to its requirements independently of RTT as well as high link utilization in access networks.

6.3 Simulation model

To investigate the efficiency of the proposed scheme in edge computing, I evaluated it after its implementation through simulation using Network Simulator ns-3 [43]. In this study, I focus on the fundamental performance in simple edge computing environment where various flows with different RTTs coexist and have the same requirements in terms of transmission rates. Figure 6.4 shows the sim-

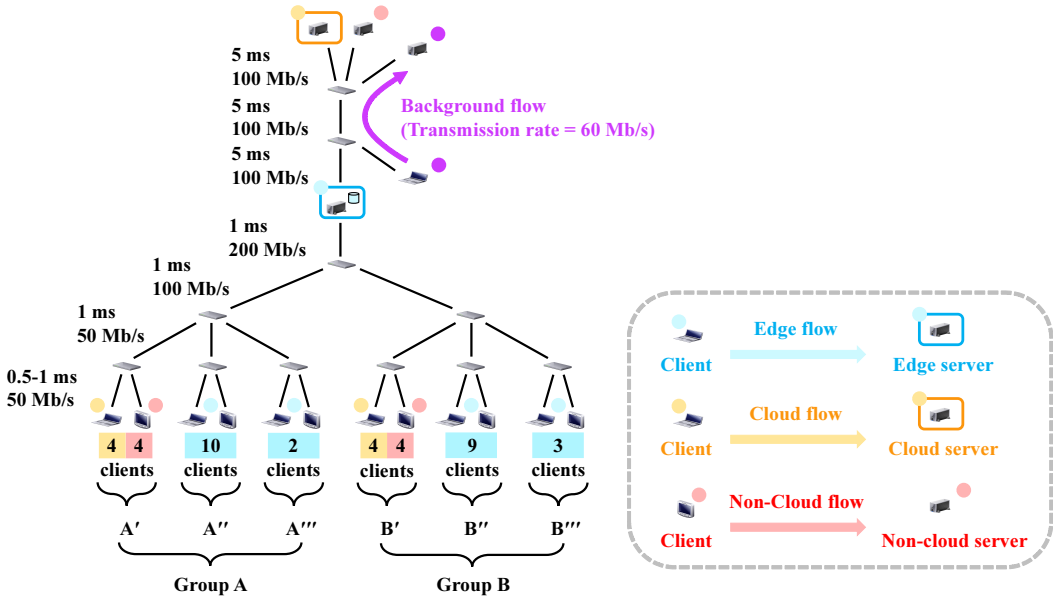


Figure 6.4: Simulation model

ulation topology and Table 6.1 summarizes simulation parameters. In this simulation, there are two groups of clients (Group A and B) to consider a situation where multiple edge routers which should be reallocated an unutilized bandwidth exist. Each client continuously sends data to the corresponding servers using TCP NewReno after the simulation starts. At 5 seconds after the simulation starts, one background flow joins through routers in core networks and then leaves at 10 seconds. Namely, the core link becomes a bottleneck during this period from 5 to 10 s, while the access link becomes a bottleneck in the other periods. I evaluated Jain's fairness index [44] and total throughput of each group for the proposed and conventional schemes. The fairness index and total throughput are calculated at intervals of 1.0 s.

6.4 Simulation results

Table 6.2 summarizes the fairness index and the total throughput of each group when the proposed and conventional schemes are applied. Here, the notification

Table 6.1: Simulation parameters

TCP algorithm	NewReno
Transmission rate of background flow	60 [Mb/s]
Notification interval time for flow information	0.05 [s]
Threshold factor α	0.8
Allocation factor β	1.3
Weight value of all flows	1
Simulation time	15 [s]
Number of trials	10

Table 6.2: Fairness index and total throughput

		Conventional	Conventional (reallocation) [17]	Proposed (reallocation+residual)
Group A	Fairness index	0.71	0.86	0.83
	Total throughput [Mb/s]	93.44	90.93	93.97
Group B	Fairness index	0.71	0.90	0.87
	Total throughput [Mb/s]	93.63	92.85	94.47

interval time for flow information is set to 0.05 s, the threshold factor α is set to 0.8, the allocation factor β is set to 1.3, and the weight value of all flows is set to 1. From Table 6.2, the proposed (reallocation+residual) scheme achieves higher total throughput than the other schemes as well as fairness index almost the same as that of the conventional (reallocation) scheme in both groups. In Group A, the proposed (reallocation+residual) scheme effectively reallocates unutilized bandwidth caused by congestion on cloud and non-cloud flows (Group A$'$) to competing edge flows through the same edge router with sufficient residual bandwidth (Group A$'''$). In Group B, the proposed (reallocation+residual) scheme reallocates unutilized bandwidth to competing edge flows through the same edge router (Group B$''$ and B$'''$). The reallocated bandwidth of Group B$''$ flows is limited to the residual bandwidth, so that the remaining unutilized bandwidth is reallocated to Group B$'''$ flows which have sufficient residual bandwidth.

6.5 Conclusion

In this study, I have proposed a bandwidth allocation scheme based on residual bandwidth information to improve link utilization in edge computing. This scheme collects residual bandwidth information from routers in access networks and then reallocates unutilized bandwidth to other competing flows with an sufficient residual bandwidth in access networks based on collected information when bottleneck links exist in core networks. Simulation evaluations have indicated that the proposed scheme achieves higher fairness than the conventional schemes as well as high link utilization in edge computing. In future work, I will investigate the effect of parameters such as the number of TCP clients, delay time, and the transmission rate of background flow on the performance of the proposed scheme.

7 Flow Splitting and Aggregation Schemes to Improve Throughput Performance in Edge and Cloud Computing

7.1 Introduction

With the development of mobile devices and wireless networks, a demand for low-latency mobile applications has been growing exponentially. For such mobile applications, edge computing [4, 5] is expected, which locates edge servers with limited computing and storage resources at the edge of networks. In edge computing, users offload a task (e.g., program execution) to edge servers, since computing resources and batteries of mobile devices are limited. The edge servers execute the offloaded tasks and return the results of them to users. Through these procedures, users can enjoy various applications without depending on the limitations of mobile devices.

To achieve high quality mobile applications, several studies on task offloading in edge computing have been reported. A task offloading scheme proposed in these studies selects an appropriate edge server from available ones according to related information (e.g., computing resources of edge servers, communication delays between users and edge servers, and energy consumption of mobile devices), and then offloads tasks from users to it; therefore, this scheme can minimize delays and save energy consumption for task offloading. However, when an edge server's load is too heavy, a large number of tasks will be offloaded to distant cloud servers. In that case, a long distance between users and cloud servers significantly

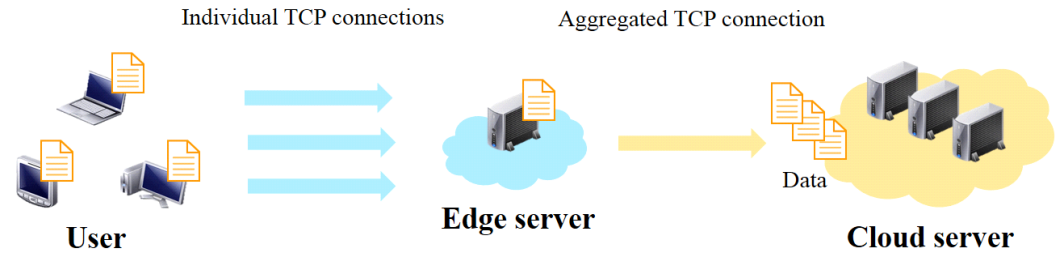
degrades the quality of mobile applications. This is because users often use TCP for reliable communication, which has congestion control mechanisms according to round-trip times (RTTs). To achieve high quality mobile applications in edge computing, the communication delay time needed to offload tasks to cloud servers should also be improved.

In this study, I propose flow splitting and aggregation schemes to improve offload delay to cloud servers as well as provide high quality mobile applications in edge computing. This scheme splits TCP connections between users and cloud servers at an intermediate edge server, and then aggregates TCP connections between the edge server and cloud servers. I show the effectiveness of this scheme through simulation evaluations.

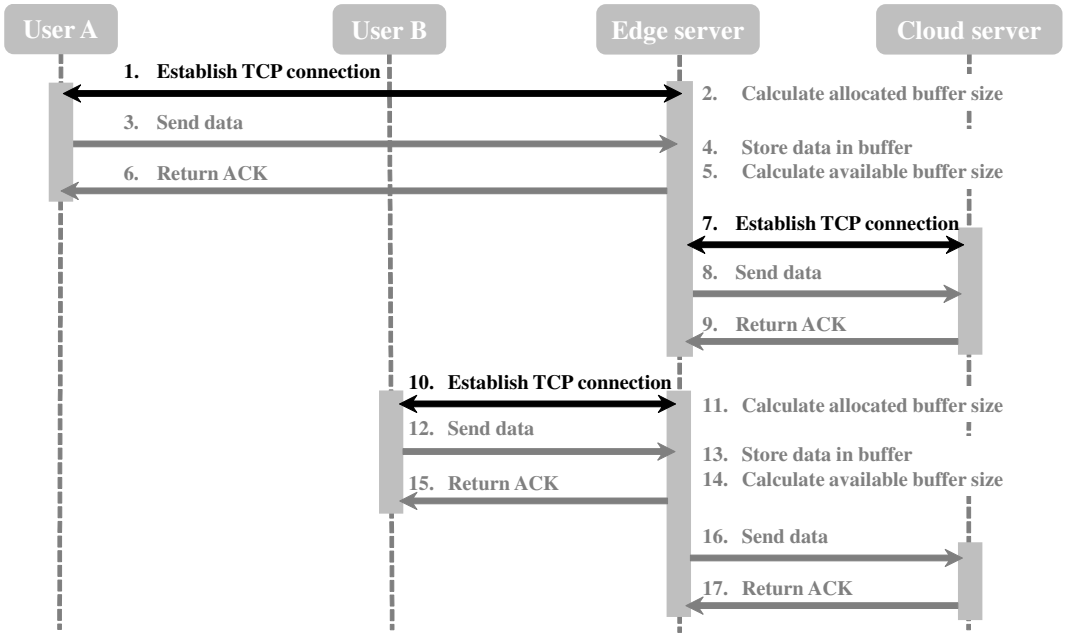
7.2 Proposed scheme

This scheme splits TCP connections between users and cloud servers at an intermediate edge server and then aggregates TCP connections between the edge server and cloud servers as shown in Fig. 7.1(a). To avoid a buffer overflow on the edge server due to bursty data transmission from users, this scheme employs flow control mechanism for individual TCP connections between the edge server and users. This scheme controls the individual and aggregated TCP connections based on flow splitting and aggregation methods [50–52] which are well-known approaches to improve TCP performance.

Here, I describe the operation of this scheme as shown in Fig. 7.1(b). Users offloading tasks to cloud servers first establish TCP connection with an intermediate edge server and send data to the edge server. The edge server receiving data from users returns ACK messages to users and stores the received data in its buffer. If TCP connection between the edge server and cloud servers has not been established yet, the edge server establishes TCP connection with the cloud server and forwards the stored data to it; otherwise, the stored data are forwarded to the cloud server through already established TCP connection. Note that users can increase transmission rates effectively because this scheme has shorter response time of ACK messages than conventional TCP by flow splitting. To avoid a buffer overflow due to bursty data transmission from users, the edge server allocates an



(a) Overview



(b) Operation

Figure 7.1: Flow splitting and aggregation schemes

individual buffer and notifies an available buffer size to each user. The "allocated" buffer size for each user is calculated by dividing the total buffer size on the edge server by the number of individual TCP connections, which is updated when the edge server receives SYN or FIN messages from users. The "available" buffer size for each user is determined based on current buffer utilization and notified to each user with TCP's advertised window size. Through these flow splitting and aggregation, this scheme can improve the communication delay needed to offload

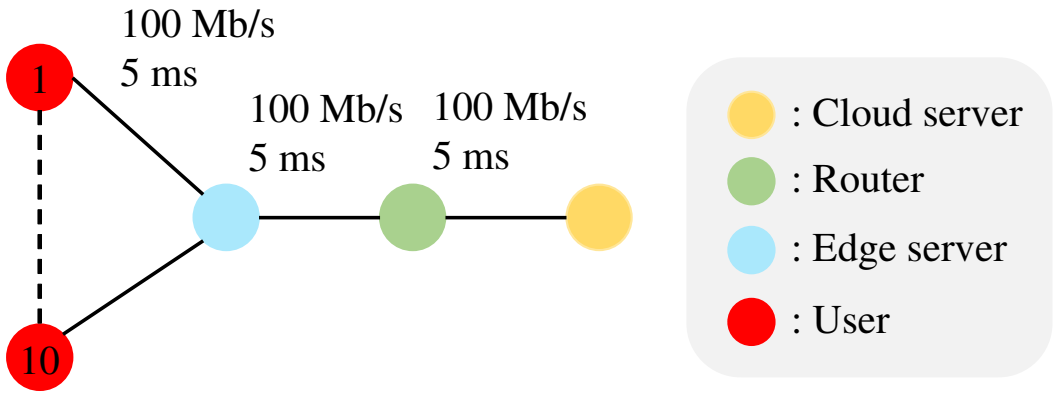


Figure 7.2: Simulation topology

Table 7.1: Simulation parameters

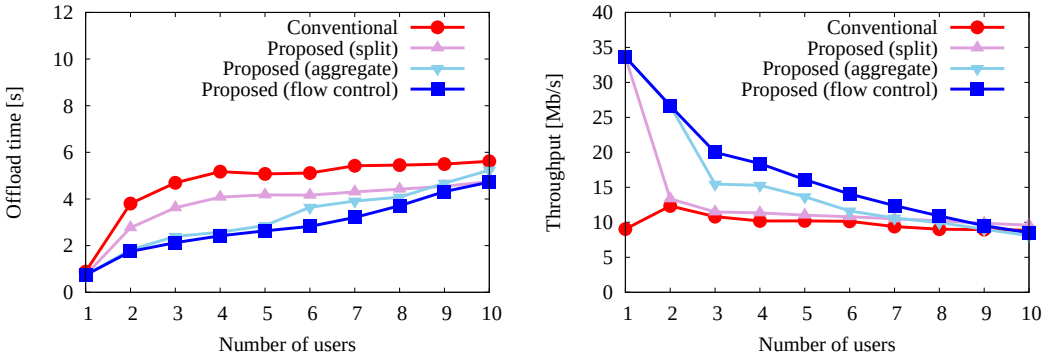
TCP algorithm	TCP NewReno
Number of users	1–10
Buffer size on edge server	100–1000 [KB]
Data size	1–10 [MB]
Segment size	1000 [Byte]
Number of trials	10

tasks from users to cloud servers in edge computing.

7.3 Simulation model

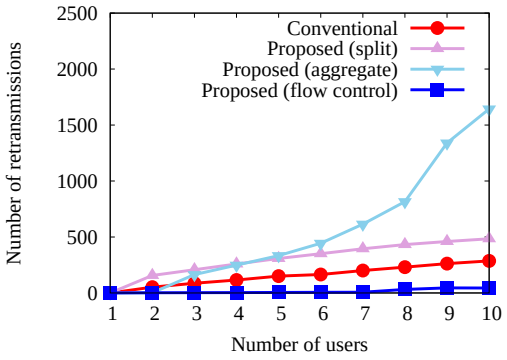
To investigate the efficiency of the proposed scheme in edge computing, I evaluated it after its implementation through simulation using Network Simulator ns-3 [43]. I implemented functions to split TCP connections between users and cloud servers at an intermediate edge server, to aggregate TCP connections between the edge server and cloud servers, and to perform flow control based on allocated buffer sizes for each TCP connection in this simulator.

Figure 7.2 shows the simulation topology. To investigate fundamental charac-



(a) Offload time

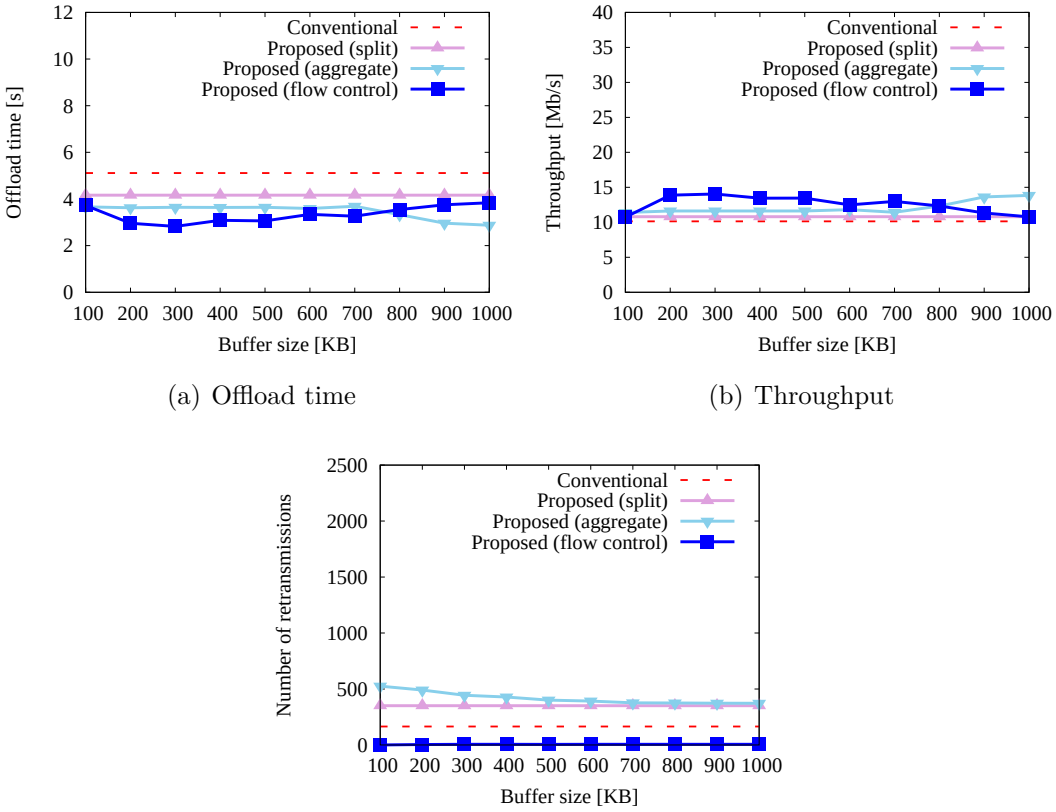
(b) Throughput



(c) Number of retransmissions

Figure 7.3: Effect of the number of users

teristics of the flow splitting and aggregation schemes in edge computing, I use a simple binary tree topology. Under the conventional scheme (original TCP), the edge server has only forwarding and routing functions, i.e., works as original router, while it has the functions including flow splitting, aggregation, and flow control in addition to the original functions under the proposed scheme. Each user sends data to the cloud server by using TCP NewReno. The start time of data transmission for each user is randomly set in a range from 0 to 0.1 s which follows uniform distribution. The bandwidth and propagation delay time of all links are set to 100 Mb/s and 5 ms, respectively. Other simulation parameters are summarized in Table 7.1.



(a) Offload time

(b) Throughput

(c) Number of retransmissions

Figure 7.4: Effect of buffer size

I compare the performance of proposed schemes (proposed (split), proposed (aggregate), and proposed (flow control)) to that of conventional scheme (original TCP) by focusing on offload time for all users, throughput of each user, and the number of retransmissions. The proposed (split) scheme splits TCP connections between users and cloud servers at the edge server. The proposed (aggregate) scheme aggregates TCP connections between the edge server and cloud servers in addition to flow splitting. The proposed (flow control) scheme performs flow control to avoid a buffer overflow on the edge server in addition to flow splitting and aggregation. Note that I ignore the time needed to execute offloaded tasks and return the results of them to users so that I focus on the performance of task

offloading from users to the cloud servers in terms of communication delay.

7.4 Simulation results

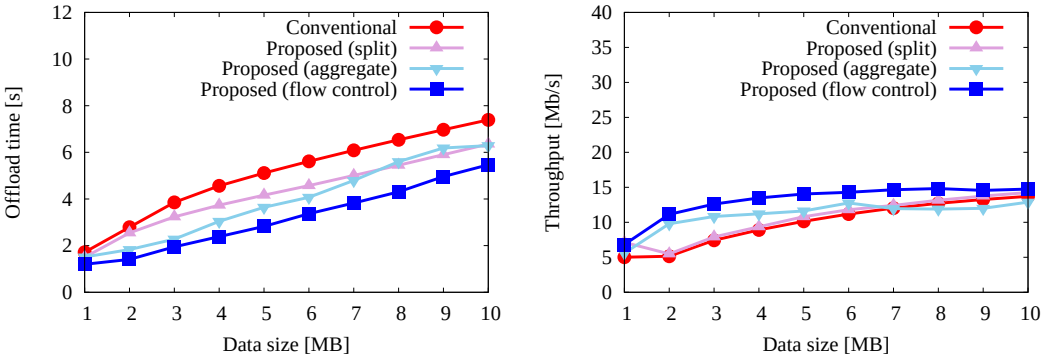
In this section, I now show simulation results and discuss the effectiveness of the proposed schemes as compared with the conventional scheme. I investigate the effect of the number of users, buffer size on the edge server, and offloaded data size.

7.4.1 Effect of the number of users

Figure 7.3 shows the offload time for all users, throughput of each user, and the number of retransmissions of the proposed and conventional schemes, respectively, when the number of users varies from 1 to 10. Here, the data size is set to 5 MB and the buffer size on edge server is set to 300 KB.

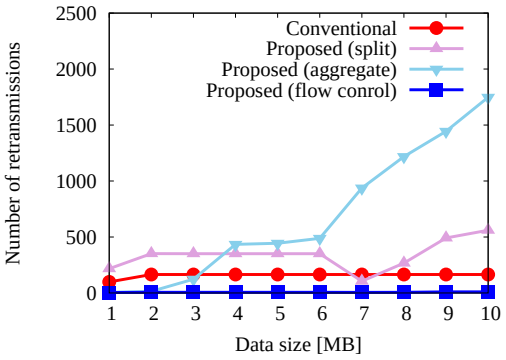
From Fig. 7.3, the proposed schemes achieve shorter offload time and higher throughput than the conventional scheme regardless of the number of users. This is because the proposed schemes can effectively increase transmission rates for each user by splitting TCP connections between the users and the cloud server. However, the offload time of proposed (split) and (aggregate) schemes are longer than that of the proposed (flow control) scheme. I describe the reasons for this phenomenon in the following.

First, I consider the difference between the proposed (split) and (flow control) schemes. When the number of users is larger than 1, the proposed (flow control) scheme establishes one TCP connection between the edge server and the cloud server (i.e., TCP connections between the edge server and the cloud server are aggregated). On the other hand, the proposed (split) scheme needs to establish one or more TCP connections between the edge server and the cloud server in such case. Namely, the proposed (split) scheme takes a longer time to establish multiple TCP connections than the proposed (flow control) scheme. Moreover, the proposed (split) scheme causes a larger number of retransmissions than the proposed (flow control) scheme because multiple flows compete in core networks under the proposed (split) scheme. From these reasons, the offload time of the proposed (split) scheme is longer than that of the proposed (flow control) scheme.



(a) Offload time

(b) Throughput



(c) Number of retransmissions

Figure 7.5: Effect of data size

Next, the difference between the proposed (aggregate) and (flow control) schemes is considered. When the number of users is smaller than 6, the offload time of proposed (aggregate) and (flow control) schemes is almost the same. However, the proposed (aggregate) scheme takes a longer offload time than the proposed (flow control) scheme when the number of users is larger than 5. This is because the proposed (aggregate) scheme exponentially increases the number of retransmissions due to buffer overflows on the edge server as the number of users increases as shown in Fig. 7.3(c).

7.4.2 Effect of buffer size

Figure 7.4 shows the offload time for all users, throughput of each user, and the number of retransmissions of the proposed and conventional schemes, respectively, when the buffer size varies from 100 to 1000 KB. Here, the number of users is set to 6 and the data size is set to 3 MB.

From Fig. 7.4, when the buffer size is too small ($<$ 200 KB), the proposed (flow control) scheme attains the poor performance in terms of the offload time and throughput because it cannot utilize the effect of flow splitting and aggregation due to too small allocated buffer size on the edge server. Similarly, when the buffer size is too large ($>$ 500 KB), the proposed (flow control) scheme degrades the performance in terms of the offload time and throughput. This is because the number of retransmissions increases slightly as the buffer size increases. Consequently, the proposed (flow control) scheme achieves the best performance when the adequate buffer size on the edge server (200–500 KB in this simulation) is employed.

7.4.3 Effect of data size

Figure 7.5 shows the offload time for all users, throughput of each user, and the number of retransmissions of the proposed and conventional schemes, respectively, when the data size varies from 1 to 10 MB. Here, the number of users is set to 6 and the buffer size on edge server is set to 300 KB.

From Fig. 7.5, although the offload time of all schemes proportionately increases as the data size increases, the proposed (flow control) scheme achieves shorter offload time and higher throughput than other schemes due to a small number of retransmissions regardless of the data size.

From the above results, I show that the proposed schemes can improve the offload time and throughput performance by flow splitting, aggregation, and flow control as compared with the conventional scheme. In particular, the proposed (flow control) scheme with the adequate buffer size on edge server achieves the shortest offload time as well as highest throughput.

7.5 Conclusion

In edge computing, users wish to tasks to edge servers due to limited computing resources and batteries of mobile devices. Edge servers execute offloaded tasks and return the results of them to users. Users can thus enjoy various mobile applications without depending on the limitations of mobile devices. However, when the edge server's load is too heavy, a large number of tasks will be offloaded to distant cloud servers. In that case, a long distance between users and cloud servers significantly degrades the quality of mobile applications. To prevent this problem, I proposed flow splitting and aggregation schemes to improve offload delay to cloud server in edge computing. This scheme splits TCP connections between users and cloud servers at the edge server, and then aggregates TCP connections between the edge server and cloud servers. To avoid a buffer overflow on the edge server due to bursty data transmission from users, this scheme also employs flow control mechanism. Through simulation evaluations, I have indicated that the proposed schemes can improve offload time as well as throughput performance by the effect of flow aggregation and flow control functions. In future work, I will evaluate the characteristics of proposed scheme in detail in more practical environment.

8 Conclusion

In this dissertation, I proposed efficient schemes to allocate and use network resources among flows to solve issues (1)–(3) in edge and cloud computing, as mentioned in Chapter 1. Through simulation evaluations, I showed the effectiveness of the proposed schemes in edge and cloud computing.

In Chapter 1, I introduced the background, objective, and organization of this dissertation.

In Chapter 2, I gave an overview of TCP to provide a better understanding of this dissertation and introduced related work.

In Chapters 3 and 4, I discussed an efficient method to fairly allocate bandwidth among users in cloud computing regarding issue (1). In Chapter 3, I proposed a bandwidth allocation scheme based on collectable information to improve fairness in DC networks. This scheme collects flow information including the bandwidth of each link, number of competing flows, and RTT of each flow from routers and servers in DC networks then fairly allocates transmission rates among flows based on the collected information. I evaluated the fundamental characteristics of the proposed scheme by focusing on the effect of allocation factor α used to adjust the transmission rate of each flow. The simulation results indicate that the proposed scheme enables higher fairness and throughput performance than that using conventional TCP by setting α to slightly larger than 1. In Chapter 4, I discussed extending the proposed scheme to reallocate unutilized bandwidth caused by congestion outside DC networks to other competing flows to improve link utilization in DC networks. This scheme collects flow information including the bandwidth of each link, number of competing flows, RTT of each flow, and actual throughput of each flow from routers and servers in DC networks then fairly allocates transmission rates among flows based on the collected information. This scheme also reallocates unutilized bandwidth to other competing flows

in DC networks when bottleneck links exist outside these networks. I evaluated the performance of the proposed scheme through simulation by focusing on how to set parameter values (e.g., notification interval time for flow information) in more detail. The simulation results indicate that this scheme enables fair communication among flows with different RTTs as well as high link utilization in DC networks by setting appropriate parameter values.

In Chapters 5 and 6, I discussed an efficient method to allocate bandwidth according to user requirements in edge computing regarding issue (2). In Chapter 5, I proposed a bandwidth allocation scheme based on collectable information to meet the requirements of each flow in edge computing. This scheme collects flow information including the bandwidth of each link, number of competing flows, RTT of each flow, and actual throughput of each flow from routers in access networks. It then allocates transmission rates for each flow based on the collected information. This scheme also reallocates unutilized bandwidth to other competing flows in access networks when bottleneck links exist in core networks. The simulation results indicate that the proposed scheme can allocate transmission rates among flows according to their requirements independently of each flow's RTT as well as achieve high link utilization. In Chapter 6, I discussed extending the proposed scheme to allocate bandwidth based on residual bandwidth information to improve link utilization in edge computing. This scheme collects residual bandwidth information from routers in access networks then reallocates unutilized bandwidth to other competing flows with a sufficient residual bandwidth in access networks based on collected information when bottleneck links exist in core networks. The simulation results indicate that the proposed scheme achieves higher fairness than conventional schemes as well as high link utilization in edge computing.

In Chapter 7, I discussed a method to provide efficient transmission in edge and cloud computing and proposed flow splitting and aggregation schemes to improve the throughput of each flow in edge and cloud computing regarding issue (3). The splitting scheme splits the TCP connections between users and cloud servers at the edge server, and the aggregation scheme aggregates the TCP connections between the edge server and cloud servers. To avoid a buffer overflow on the edge server due to bursty data transmission from users, these schemes also use

a flow control mechanism. Through simulation evaluations, I indicated that the proposed schemes can improve throughput performance by flow aggregation and flow control functions.

In this dissertation, I solved some issues in edge and cloud computing. The integration of the proposed schemes can help provide high-quality services to users according to the requirements of each user independently of his/her location. However, the following issues remain for future work. In edge and cloud computing, computation tasks will be offloaded to distant cloud servers when an edge server's load is too heavy. This task offloading can reduce execution time. However, a long transmission time is required due to the long distance, so the offload delay may become a bottleneck. To improve offload delay, each edge server should cooperate with other edge servers. For future work, I will consider an efficient method to cooperate among edge servers to improve offload delay in edge and cloud computing.

References

[1] Y. Xu and S. Mao, "A survey of mobile cloud computing for rich media applications," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 46–53, Jun. 2013. DOI:10.1109/MWC.2013.6549282

[2] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," *Proc. IEEE AINA2010*, pp. 27–33, Apr. 2010. DOI:10.1109/AINA.2010.187

[3] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, Apr. 2014. DOI:10.1109/TPDS.2014.2316834

[4] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, Mar. 2017. DOI:10.1109/COMST.2017.2705720

[5] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, Nov. 2017. DOI:10.1109/ACCESS.2017.2778504

[6] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, Feb. 2017. DOI:10.1109/ACCESS.2017.2665971

[7] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," *IEEE Pervasive Computing*, vol. 12, no. 4, pp. 40–49, Oct. 2013. DOI:10.1109/MPRV.2013.77

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of thins," *Proc. ACM MCC2012*, pp. 13–16, Aug. 2012. DOI:10.1145/2342509.2342513

[9] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," *Proc. IEEE GLOBECOM2013*, pp. 1291–1296, Dec. 2013. DOI:10.1109/GLOCOM.2013.6831252

[10] Z. Becvar, J. Plachy, and P. Mach, "Path selection using handover in mobile networks with cloud-enabled small cells," *Proc. IEEE PIMRC2014*, pp. 1480–1485, Sep. 2014. DOI:10.1109/PIMRC.2014.7136402

[11] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," *Proc. IEEE INFOCOM2004*, pp. 2514–2524, Mar. 2004. DOI:10.1109/INFCOM.2004.1354672

[12] K. Ogura, Y. Nemoto, Z. Su, and J. Katto, "A new TCP congestion control supporting RTT-fairness," *IEICE Transactions on Information and Systems*, vol. E95-D, no. 2, pp. 523–531, Feb. 2012. DOI:10.1587/transinf.E95.D.523

[13] Y. Ito, H. Koga, and K. Iida, "A bandwidth allocation scheme to improve fairness in data center networks," *Proc. ACM CoNEXT2014*, pp. 7–8, Dec. 2014. DOI:10.1145/2680821.2680834

[14] Y. Ito, H. Koga, and K. Iida, "A bandwidth allocation scheme to improve fairness in data center networks," *IEICE Communications Express*, vol. 5, no. 5, pp. 129–134, May 2016. DOI:10.1587/comex.2016XBL005

[15] Y. Ito, H. Koga, and K. Iida, "A bandwidth reallocation scheme to improve fairness and link utilization in data center networks," *Proc. IEEE PerCom2016*, pp. 65–68, Mar. 2016. DOI:10.1109/PERCOMW.2016.7457064

[16] Y. Ito, H. Koga, and K. Iida, "A bandwidth allocation scheme to improve fairness and link utilization in data center networks," *IEICE Transactions on Communications*, vol. E101-B, no. 3, pp. 679-687, Mar. 2018 DOI:10.1587/transcom.2017NRP0008

[17] Y. Ito, H. Koga, and K. Iida, "A bandwidth allocation scheme to meet flow requirements in mobile edge computing," *Proc. IEEE CloudNet2017*, pp. 114-118, Sep. 2017. DOI:10.1109/CloudNet.2017.8071541

[18] Y. Ito, H. Koga, and K. Iida, "A bandwidth allocation scheme based on residual bandwidth information in mobile edge computing," *Proc. ACM/IFIP/USENIX MECC2017*, 3 pages, Dec. 2017. DOI:10.1145/3152360.3152364

[19] Y. Ito and H. Koga, "Improving offload delay using flow splitting and aggregation in edge computing," *Proc. IEEE CCNC2019*, 2 pages, Jan. 2019.

[20] P. Sreekumari and J. Jung, "Transport protocols for data center networks: A survey of issues, solutions and challenges," *Springer Photonic Network Communications*, vol. 31, no. 1, pp. 122–128, Feb. 2015. DOI:10.1007/s11107-015-0550-y

[21] P. Prakash, A. Dixit, Y. C. Hu, and R. Kompella, "The TCP outcast problem: Exposing unfairness in data center networks," *Proc. USENIX NSDI2012*, pp. 413–426, Apr. 2012.

[22] M. Alizadeh et al., "Data center TCP (DCTCP)," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 63–74, Oct. 2010. DOI:10.1145/1851275.1851192

[23] J. Zhang, F. Ren, X. Yue, R. Shu, and C. Lin, "Sharing bandwidth by allocating switch buffer in data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 39–51, Jan. 2014. DOI:10.1109/JSAC.2014.140105

[24] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993. DOI:10.1109/90.251892

[25] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation," *Proc. IEEE INFOCOM2000*, pp. 942–951, Mar. 2000. DOI:10.1109/INFCOM.2000.832269

[26] S. Wen, Y. Fang, and H. Sun, "Differentiated bandwidth allocation with TCP protection in core routers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 1, pp. 34–47, Jan. 2009. DOI:10.1109/TPDS.2008.71

[27] L. Lu, H. Du, and R. P. Liu, "CHOKeR: A novel AQM algorithm with proportional bandwidth allocation and TCP protection," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 637–644, Feb. 2014. DOI: 10.1109/TII.2013.2278618

[28] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 89–102, Oct. 2002. DOI:10.1145/964725.633035

[29] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Explicit window adaptation: A method to enhance TCP performance," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 338–350, Jun. 2002. DOI: 10.1109/TNET.2002.1012366

[30] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and Internet cloud for delay-aware mobile cloud computing," *Proc. IEEE GLOBECOM2015 Workshops*, 6 pages, Dec. 2015. DOI:10.1109/GLOCOMW.2015.7414063

[31] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, Mar. 2018. DOI:10.1109/JSAC.2018.2815360

[32] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, Mar. 2018. DOI:10.1109/ACCESS.2018.2805798

[33] S. Sardellitti, G. Scutari, and S. Barossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, Jun. 2015. DOI:10.1109/TSIPN.2015.2448520

[34] X. Lyu, H. Tian, P. Zhang, and C. Sengul, "Multi-user joint task offloading and resources optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017. DOI:10.1109/TVT.2016.2593486

[35] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, Jan. 2019. DOI:10.1109/TVT.2018.2881191

[36] A. Ksentini, T. Taleb, and M. Chen, "A markov decision process-based service migration procedure for follow me cloud," *Proc. IEEE ICC2014*, pp. 1350–1354, Jun. 2014. DOI:10.1109/ICC.2014.6883509

[37] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," *Proc. IEEE MILCOM2014*, pp. 835–840, Oct. 2014. DOI:10.1109/MILCOM.2014.145

[38] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service migration placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017. DOI:10.1109/TPDS.2016.2604814

[39] J. Zhang, F. Ren, and C. Lin, "Survey on transport control in data center networks," *IEEE Network Magazine*, vol. 27, no. 4, pp. 22–26, Jul./Aug. 2013. DOI:10.1109/MNET.2013.6574661

[40] K. Kusuhata, I. Kaneko, K. Iida, H. Koga, and M. Shimamura, "A Unified congestion control architecture design to improve heterogeneous wireless network efficiency and accommodate traffic by various rich applications," *Proc. IFIP HET-NETs2013*, 5 pages, Nov. 2013.

[41] N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Apr. 2008. DOI:10.1145/1355734.1355746

[42] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information," *IETF RFC7011*, Sep. 2013.

[43] The ns-3 network simulator, http://www.nsnam.org/.

[44] R. Jain, D. M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared systems," *DEC Research Report*, TR-301, Sep. 1984.

[45] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," *Proc. IEEE ISCO2016*, 8 pages, Jan. 2016. DOI:10.1109/ISCO.2016.7727082

[46] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP connection characteristics through passive measurements," *Proc. IEEE INFOCOM2004*, pp. 1582–1592, Mar. 2004. DOI:10.1109/INFCOM.2004.1354571

[47] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, "Improving TCP/IP performance over wireless networks," *ACM MobiCom1995*, pp. 2–11, Nov. 1995. DOI:10.1145/215530.215544

[48] J. Plachy, Z. Becvar, and P. Mach, "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network," *Elsevier Computer Networks*, vol. 108, pp. 357–370, Oct. 2016. DOI:10.1016/j.comnet.2016.09.005

[49] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, Aug. 1995. DOI:10.1109/90.413212

[50] R. Chakravorty, S. Katti, I. Pratt, and J. Crowcroft, "Using TCP flow-aggregation to enhance data experience of cellular wireless users," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 6, pp. 1190–1204, Jun. 2005. DOI:10.1109/JSAC.2005.845628

[51] T. Khalifa, A. Abdrabou, K. Naik, M. Alsabaan, A. Nayak, and N. Goel, "Split- and aggregated-transmission control protocol (SA-TCP) for smart

power grid," *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 381–391, Jan. 2014. DOI:10.1109/TSG.2013.2279756

[52] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt, "Flow aggregation for enhanced TCP over wide-area wireless," *Proc. IEEE INFOCOM2003*, pp. 1754–1764, Mar./Apr. 2003. DOI:10.1109/INFCOM.2003.1209198