Optimization of Server Locations in Server Migration Service

Yukinobu Fukushima
The Graduate School of Natural Science and Technology
Okayama University
Okayama, Japan
fukusima@okayama-u.ac.jp

Tutomu Murase
Cloud System Research Laboratories
NEC Corporation
Kanagawa, Japan
t-murase@ap.jp.nec.com

Tokumi Yokohira
The Graduate School of Natural Science and Technology
Okayama University
Okayama, Japan
yokohira@okayama-u.ac.jp

Tatsuya Suda
University Netgroup Inc.
Irvine, USA
tatsuyasuda@gmail.com

***Abstract*—In server migration service (SMS), a work place (WP) refers to a computer that runs a virtual machine, and a server refers to a virtual machine that runs a server-side application of a network application (NW-App). In SMS, WPs are deployed at various locations in a network, and servers may migrate between WPs towards the users of the NW-App to achieve better QoS for the users. In SMS, an SMS provider tries to provide an NW-App provider with a certain level of QoS that they agree upon, and if the SMS provider fails, it pays penalty (e.g., reimbursement of a part of service charges to users) depending on the degree and length of the QoS violation. Thus, the SMS provider is incentivized to migrate servers between WPs to satisfy the agreed-upon QoS level to reduce the penalty. On the other hand, an SMS provider also needs to be moderate in performing server migrations to avoid degradation of network QoS (i.e., QoS of background traffic). This is because a server is typically large in size and the server generates a large amount of traffic when it migrates, resulting in increasing delay and loss for its background traffic in a network. This paper formulates an integer-programming model for the off-line server locations decision (i.e., when and to which WP server should migrate) where the penalty associated with NW-App's QoS violations is minimized, keeping the number and distance of server migrations below a given level. This paper also compares the minimum penalty obtained through solving the integer-programming model against the penalty obtained with a greedy on-line server locations decision algorithm, which migrates a server to a WP that minimizes the current penalty with no consideration of the penalty that will arise in the future. Numerical examples show that the integer-programming model achieves 36% to 49% lower penalty than the greedy algorithm when the degradation of network QoS is little acceptable.**

Keywords-cloud computing*; *server migration service*; *integer-programming model*; *penalty

I. INTRODUCTION

Cloud computing [1] is emerging as a new computing paradigm. As one of its service models, IaaS (Infrastructure as a Service) cloud service (e.g., Amazon EC2 [2]) is attracting attention from the cloud research community. In IaaS cloud service, customers may operate their virtual machines (VMs) at IaaS cloud service provider's data center on demand with little initial capital investment and operation complexity.

In IaaS cloud service, the location of a VM is fixed at an IaaS provider's data center. In supporting highly interactive network applications (NW-Apps) such as network games application that consists of one or more server-side applications and client-side applications, if the QoS of the NW-App's communication degrades because of some reasons (e.g., there is a significant physical distance between a server-side application and its client-side applications), it is difficult for an IaaS provider to provide a NW-App with a desired level of QoS such as low delay and high throughput.

QoS of NW-Apps in IaaS cloud service may improve by adopting the server migration service (SMS) [3] (also referred to as micro data centers [4]). In SMS, work places (WPs) that run virtual machines are deployed at various locations, and a virtual machine (server) that runs a server-side application of an NW-App can migrate between WPs towards the users of the NW-Apps to achieve better QoS for the users.

In SMS, an SMS provider tries to provide an NW-App provider with a certain level of QoS that they agree upon, and if the SMS provider fails, it pays penalty (e.g., reimbursement of a part of service charges to users) depending on the degree and length of the QoS violation. Thus, the SMS provider is incentivized to migrate servers between WPs to satisfy the agreed-upon QoS level to reduce the penalty. On the other hand, an SMS provider also needs to be moderate in performing server migrations to avoid degradation of network QoS (i.e., QoS of background traffic). This is because a server is typically large in size (e.g., the storage size of a VM in IaaS can be a few hundreds of gigabytes) and the server generates a large amount of traffic when it migrates, resulting in increasing delay and loss for its

 ISBN: 978-1-61208-256-1

background traffic in a network.

This paper formulates an integer-programming model for the off-line server locations decision (i.e., when and to which WP server should migrate) where the penalty associated with NW-App's QoS violations is minimized, keeping the number and distance of server migrations below a given level. This paper also compares the minimum penalty obtained through solving the integer-programming model against the penalty obtained with a greedy on-line server locations decision algorithm, which migrates a server to a WP that minimizes the current penalty with no consideration of the penalty that will arise in the future.

Previous work related to server migration service includes server migration within a single data center [5] and migration of databases (DBs) [6]. The paper [5] proposes an algorithm which enables a server dynamically migrate among different physical hosts within a single data center according to their workloads. The algorithm proposed in [5] reduces the number of servers required to achieve a given server response time. However, the paper focuses on server migration within a single data center and does not need to consider degradation of network QoS due to the server migration. Our paper considers server migration across a network and considers degradation of network QoS due to the server migration unlike the paper [5]. The paper [6] proposes a DB-migration scheduling algorithm and achieves the shortest communication time between DB servers and their clients by optimally determining locations of DB servers for a given query sequence. However, degradation of network QoS due to a DB server migration is not considered. Our paper considers degradation of network QoS due to the server migration.

The rest of the paper is organized as follows. Section II explains an NW-App model and a network model as well as the server migration service that are considered in this paper. Section III describes how server locations are determined and formulates it as an integer programming model. In Section IV, the numerical examples are presented. Section V concludes the paper.

II. A MODEL FOR THE SERVER MIGRATION SERVICE

A. A model for a network application

As shown in Fig. 1, a NW-App consists of one or more server-side applications and client-side applications, and the former run on a virtual machine (hereafter referred to as "server") that is operated at WPs, while the latter run on a user-terminal (hereafter referred to as "client") such as a note PC and a smart phone. It is assumed that a server runs a single server-side application of NW-App and when a server-side application needs to migrate to the new WP, the server that runs the application also migrates to the new WP. It is assumed that server S_i ($i = 1, 2, \cdots, n$) communicates with a pre-determined set of clients C_i^j ($i = 1, 2, \cdots, n$, and $j = 1, 2, \cdots, m_i$). If a NW-App operates multiple servers for some purpose such as load balancing, server S_i also communicates with other servers S_ks ($k \neq i$) in the same NW-App. It is assumed that clients do not communicate with other clients.

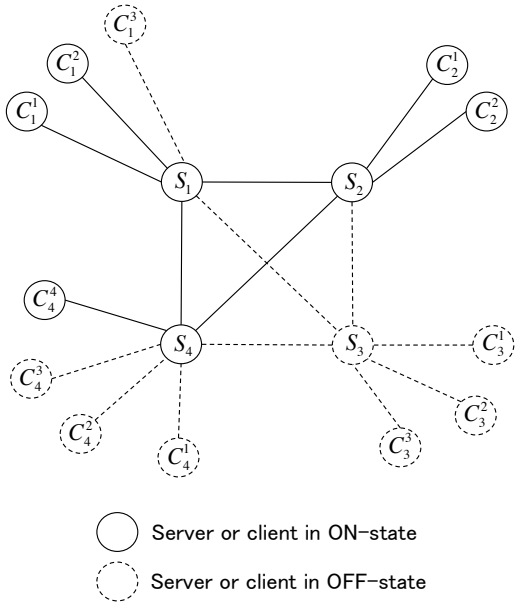

Figure 1. Communications in a network application

A server and a client each has two states, ON-state (i.e., running a server/client-side application) and OFF-state (i.e., not running a server/client-side application). The communication described above among servers and clients only occur when they are in ON-state. A client changes from OFF-state to ON-state when it starts to execute the client-side application, and it returns to OFF-state when its execution completes. A server is in ON-state only when at least one of its clients is in ON-state. For example, S_4 is in ON-state because its client C_4^4 is in ON-state, and S_3 is in OFF-state because all of its clients are in OFF-state.

B. A network model

Fig. 2 shows an example of a network considered in this paper. In Fig. 2, $R_1 \sim R_6$ are routers. Client C_i^j is connected to a router, and client-router association does not change throughout the time period of the interest of this paper. A work place (WP) is a physical computer and can operate a VM (server) that run a server-side application of a NW-App. A server in ON-state uses the resources of a WP (e.g., CPU and memory) for communication. Thus, the number of ON-state servers in a WP should be bounded due to the limitation of the resources (e.g., CPU time and memory space) of a WP. This paper assumes that the number of ON-state servers that a WP can support is bounded at a pre-determined threshold (refereed to as the capacity of a WP).

C. Server migration service

SMS can be classified into two service models: integrated model and overlay model depending on whether an SMS provider owns a network or not. In an integrated model, an

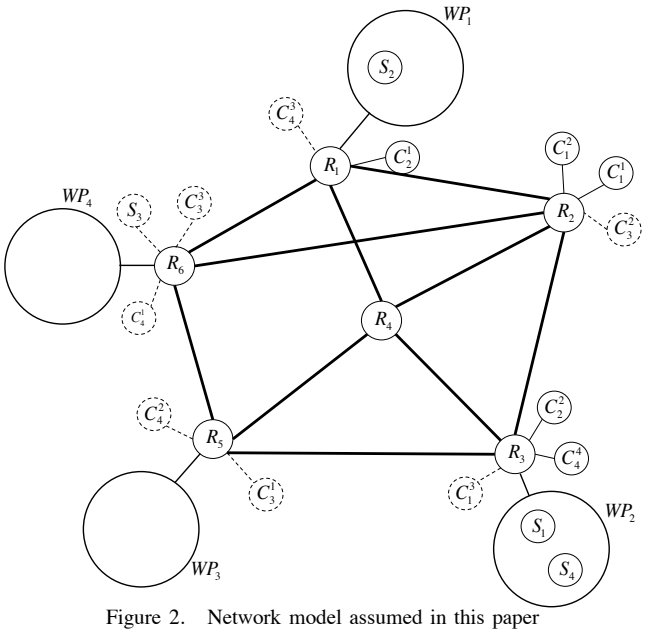

Figure 2. Network model assumed in this paper

SMS provider owns WPs and a network that connects among the WPs (e.g., carrier cloud with SMS), and consequently the objectives of the SMS provider will be 1) providing NW-Apps with good QoS and 2) keeping the network QoS good. In an overlay model, on the other hand, an SMS provider owns WPs only and rents network capacity from network operators to achieve the reachability among WPs, and consequently the objectives of the SMS provider will be 1) providing NW-Apps with good QoS and 2) reducing the network rental fees paid for network operators. Although our server locations decision model can cope with both the models, this paper focuses on the integrated model.

Fig. 3 shows SMS based on the integrated model. The intended customers of the SMS are NW-App developers who hope to run their NW-Apps with the desired level of QoS. In the SMS, prior to the start of the service, an NW-App developer and the SMS provider make an agreement regarding NW-App's QoS (such as communication delays between a server and a client, communication delays between servers, throughputs, and packet loss) of the service that the SMS provider provides the NW-App provider with. Based on the agreement, the NW-App developer pays fees to the SMS provider, and the SMS provider provides the NW-App developer with its service. If the NW-App's QoS is violated, the SMS provider pays the penalty for the NW-App developer. The penalty will be calculated based on the degree of the NW-App's QoS violation and its duration (i.e., how long the NW-App's QoS had been violated).

In order to provide QoS specified in the agreement and minimize the penalty, an SMS provider may migrate servers among WPs. Server migrations may be performed when the NW-App's QoS is violated (reactive migration) or in order to prevent the QoS from being violated (proactive migration). When a server migration is performed, the server

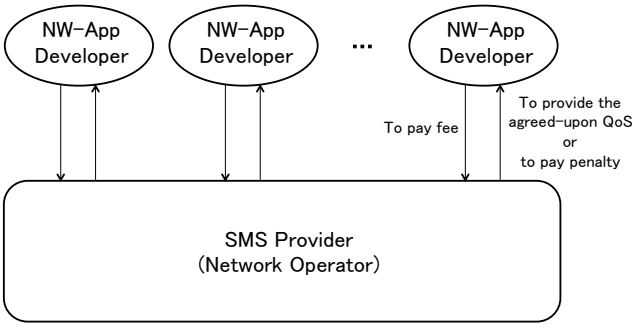

Figure 3. Server migration service based on the integrated model

needs to be migrated from a WP to another WP, creating additional traffic and resulting in possible QoS degradation of the network. Thus, in performing server migration, the SMS provider also needs to minimize the network QoS degradation caused by the additional traffic associated with server migration.

To decide when and to which WP servers migrate so that both the penalty and the network QoS degradation are minimized in SMS, two server locations decision approaches can be considered: off-line approach and on-line approach. An off-line approach makes decisions of server locations based on client ON/OFF state transitions in the past, present and future. It can be applied to NW-Apps whose clients show regular and easily predictable ON/OFF state transitions. The off-line approach achieves the optimal performance when the predicted state transitions are correct, while it can result in great performance degradation when the transitions are wrongly predicted. An on-line approach makes decisions of server locations without client ON/OFF state transitions in the future. It can be applied to any NW-Apps and achieves reasonable performance. As the first step for realizing the optimal SMS, this paper proposes an off-line server locations decision approach. Our approach is useful for NW-Apps with predictable client state transitions, and also serves as a benchmark for on-line approaches that cope with NW-Apps with unpredictable client state transitions.

III. Off-line approach for server locations decision

As discussed in Section II, it is important to minimize both the penalty associated with NW-App's QoS violation and the network QoS degradation caused by the additional traffic associated with server migrations. This section first considers the server locations decision as a simple discrete-time model. Then this section formulates an integer-programming model for the discrete-time server locations decision.

A. Discrete-time server locations decision

In order to simplify modeling of server locations decision, we consider a discrete-time model where state-transitions and server locations decision (i.e., when and to which WP servers should migrate) occur at discrete-time instances. This

model is referred to as *the discrete-time server locations decision* in the rest of the paper.

In the model, time is slotted, and server and client status changes at the boundary of slots, and the server locations are also determined at the boundary of slots as depicted in Fig. 4. It is assumed that, once the new location (WP) is determined for a server, the server migrates to the WP instantaneously, i.e., there is no network delay associated with moving the server to the new WP. In calculating NW-App's QoS provided by the SMS provider and also the network QoS degradation, it is assumed that those QoSs are static within a slot, and they change only at the boundaries of a slot. These assumptions make it relatively easier to calculate NW-App's QoS that the SMS provider provides and the network QoS degradation that servers create when they migrate.

Because time slots are artificially introduced in the discrete-time model to approximate continuous time, it is important to carefully determine the size of a time-slot. When a slot is large, the deviation in the calculated NW-App's QoS and the network QoS degradation from the actual values will be large. When a slot size is small, frequency of the server locations decision increases, and consequently complexity (e.g., CPU time and memory space) for determining server locations will be high. The optimal length of a time slot is beyond the scope of this paper.

The total penalty refers to the sum of the penalties that arise in all time-slots. In order to calculate the total penalty, the penalty in a single slot is first calculated. The penalty in each slot depends on the NW-Apps' QoSs of communications between servers and their clients, as well as communications between servers. For each communication, *a penalty function* calculates the penalty based on the degree of the NW-App's QoS violation and its duration. In our model, any form of the function may be adopted under the agreement between the SMS provider and the NW-App developer.

The degree of network QoS degradation due to a server migration may be calculated based on factors such as the size of a server and the number of hops that a server takes to move to the new WP. *A network QoS degradation function* calculates the degree of the network QoS degradation. In our model, any form of the function may be adopted by the SMS provider. In order to keep the network QoS degradation below a predetermined level, the sum of the degrees of the network QoS degradations needs to be kept below a predetermined upper bound in a given period of time (i.e., called a network QoS degradation window) that consists of a given number of consecutive time-slots. For example, if the size of a network QoS degradation window is three (slots) and if the upper bound on the sum of the degrees of network QoS degradations in a window is ten in Fig. 4, the sum of the degrees of the network QoS degradations in every window must be less than or equal to ten.

B. Model formulation

Given a sequence of ON/OFF state transitions of all clients, the optimal server locations in every time-slot should be decided so that the total penalty is minimized while keeping the network QoS degradation below a predetermined level. Our model for the server locations decision is based on an integer programming and is described below.

- Parameters (Constants)
  - T : A set of consecutive slots (= $\{0, 1, 2, \cdots N\}$). Slots 1 to N are included in the time period where the SMS provider migrates servers while slot 0 stands for expressing the initial locations (WPs) of servers.
  - S : A set of servers (= $\{1, 2, \cdots n\}$).
  - L : A set of WPs (= $\{1, 2, \cdots r\}$).
  - R_i : A set of clients that server i supports.
  - C_i : Capacity of WP i (i.e., the number of servers that WP i can support).
  - Q_i^t : A binary constant that is equal to 1, if client i is in ON-state in slot t, and 0, otherwise.
  - P_{ij}^t : Penalty when server i stays at WP j in slot t (the penalty function calculates P_{ij}^t using Q_i^t).
  - W : A set of network QoS degradation windows (= $\{\{1, 2, \cdots, p\}, \{2, 3, \cdots, p+1\}, \cdots, \{N-p+1, N-p+2, \cdots, N\}\}$ where p is the length (slots) of the window).
  - U : The upper bound on the sum of the degrees of the network QoS degradations for a given network QoS degradation window.
  - I_{ij} : The degree of network QoS degradation when a server migrates from WP i to WP j (the network QoS degradation function calculates I_{ij}).
  - F_i : Initial location (WP) of server i.
- Variables
  - s_{ij}^t : A binary variable that is equal to 1, if server i stays at WP j in slot t, and 0, otherwise.
  - o_i^t : A binary variable that is equal to 1, if server i is in ON-state in slot t, and 0, otherwise.

Using the parameters and variables defined above, our model becomes to minimize the objective function (1) subject to (2)–(7) below.

- Objective function: To minimize the total penalty.

$$\text{minimize} \quad \sum_{t \in T} \sum_{i \in S} \sum_{j \in L} P_{ij}^t s_{ij}^t \tag{1}$$

- Constraints
  - A server's location must fall within all WPs.

$$\sum_{j \in L} s_{ij}^t = 1 \quad \forall i \in S, \forall t \in T \tag{2}$$

  - The number of servers that reside on a WP must

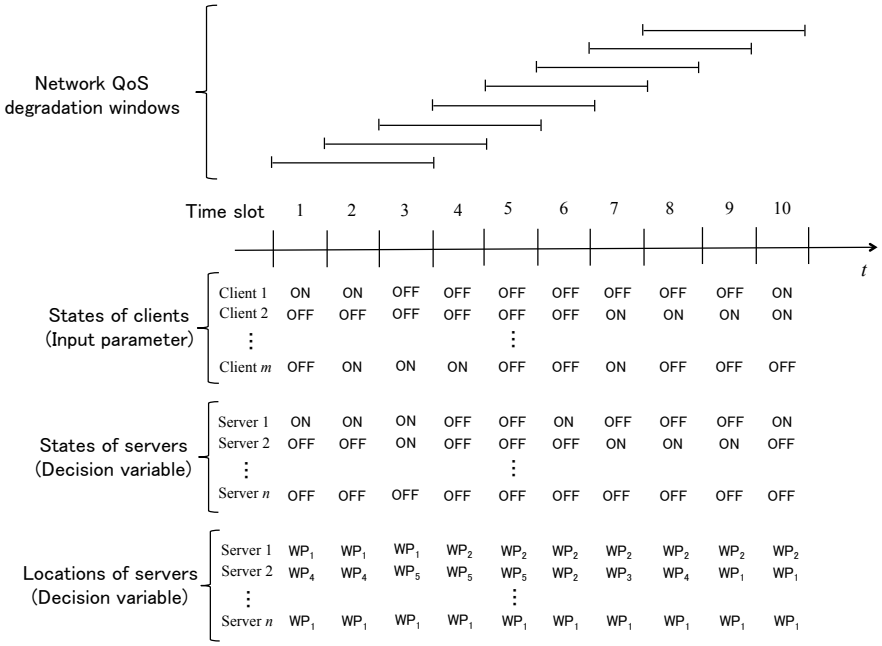

Figure 4. An example of discrete-time off-line server locations decision.

be less than or equal to the WP's capacity.

$$\sum_{i \in S} o_i^t s_{ij}^t \leq C_j \quad \forall j \in L, \forall t \in T \tag{3}$$

– Sum of the degrees of network QoS degradations during an network QoS degradation window must be less than or equal to the predetermined upper bound.

$$\sum_{t \in w} \sum_{i \in S} \sum_{j,k \in L} s_{ij}^{t-1} s_{ik}^t I_{jk} \leq U \quad \forall w \in W \tag{4}$$

– A server is in ON-state, if one or more of its clients is in ON-state.

$$o_i^t \geq Q_j^t \quad \forall j \in R_i, \forall i \in S, \forall t \in T \tag{5}$$

– A server's initial WP is given.

$$s_{ij}^0 = 1 \quad j = F_i, \forall i \in S \tag{6}$$

$$s_{ij}^0 = 0 \quad j \neq F_i, \forall i \in S \tag{7}$$

IV. Numerical examples

In this section, we obtain the optimal performance with our integer programming model in section III, and compare it with the performance obtained through a simple greedy on-line algorithm.

A. Parameter settings

With the simple greedy on-line algorithm that we consider in this paper, a server selects the WP with the minimum penalty in the current slot from the candidate WPs that satisfy the network QoS degradation constraint. When there are multiple such candidates with the same minimum penalty, a server migrates to the WP that yields the minimum network QoS degradation when the server migrates to the new WP.

We use 14-node NSFNET (Fig. 5) as the network model. Every router is equipped with one WP with the capacity of one server. The propagation delays of links in this network varies between 1.4 and 11.2 [ms]. In the numerical examples in this section, it is assumed that the delay on each link is dominated by the propagation delay of the link, and the packet transmission time (i.e., packet length divided by channel speed of the link) and the queueing delay at a router are negligible. Negligible packet transmission time is realistic and justified, because channel speed of the link is huge and getting huger. So is negligible queueing delay at a router, because it is reported that a very small buffer (e.g., a buffer for 10–20 packets) is enough for core routers to achieve high TCP throughput [7]. Small buffer yields negligible queueing delay at routers.

In the scenario considered in this numerical result section, there is only one NW-App consisting of one server and 14 clients. 14 clients are uniformly distributed over 14 routers (i.e., one client per router). The SMS provider and the NW-App developer agree that end-to-end delay between the server and each client must be smaller than or equal to 10 ms. It is assumed that clients follow the exponential ON/OFF model where ON-state period and OFF-state period follow the exponential distributions with means μ_{ON} = 2 (slots) and μ_{OFF} = 10 (slots), respectively. The duration of the time period the SMS provider performs server migrations is set to 10 (slots). We consider a total of 100 ON/OFF-state transitions of clients to obtain the average of total penalties. IBM ILOG CPLEX Optimizer [8] is used to solve our integer programming model.

 ISBN: 978-1-61208-256-1

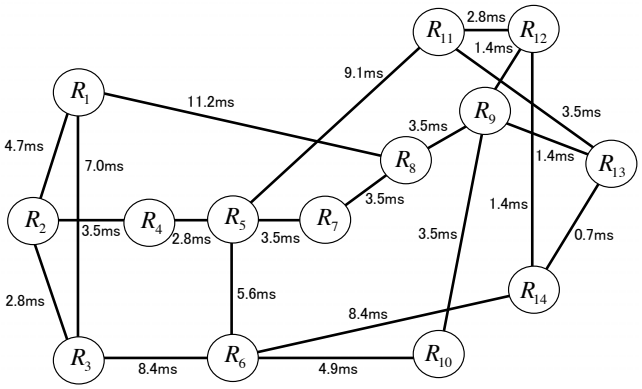

Figure 5. NSFNET

It is assumed that the penalty of the NW-App's QoS violation in one time-slot is proportional to the difference between the end-to-end delay between the server and a client and the predetermined threshold (i.e., 10 ms), i.e., the penalty function is α times the difference, where α is a constant. Note that the difference is regarded as zero when the end-to-end delay is smaller than or equal to the threshold value. In the numerical examples in section IV.B, we set the value of α to ten. As for the network QoS degradation function, we simply consider that the degree of network QoS degradation is equal to the number of hops that a server takes to move to the new WP (e.g., the server's migration on 2-hop counts route causes the network QoS degradation of two). We set the size of the network QoS degradation window to three (slots). It is assumed that a server and a client communicate using the shortest hop path between them. It is also assumed that the server migrates to the new WP using the shortest hop path. In the numerical result examples shown in section IV.B, the upper bound (U) on the sum of the network QoS degradation in a window varies one to ten.

B. Results

Figs. 6 and 7 depict the average total penalty as a function of the upper bound on the sum of the network QoS degradation in a window (U) with 95% confidential interval, when the server's initial locations are set to WP 1 connected to R_1 and WP 5 connected to R_5, respectively.

These figures show that the average total penalties of both our model and the greedy algorithm decrease as U increases. This is because the larger U enables the server to migrate more frequently and/or to the new WP that are further over more number of hops. Consequently, there is a larger possibility of a server finding the new WP that either avoids or reduces the penalty.

Figs. 6 and 7 show that, when U is less than or equal to three (when the degradation of network QoS is little acceptable), our model achieves 36% to 49% lower penalty than the greedy algorithm.

When U is larger than or equal to four, both our model and the greedy algorithm show nearly identical penalty. This is explained as follows. When the value of U is large, namely, when there is no tight upper bound on the network QoS degradation, even if the server migrate to almost any WP, it still meet the network QoS degradation constraint. As a result, with the greedy algorithm, a server often migrates to the WP with the minimum penalty, resulting in nearly identical total penalty as with our model.

We next explore the influence of the server's initial location on the average total penalty. Figs. 8 and 9 depict the average total delays of our model and the greedy algorithm as a function of the server's initial WP. In the figures, the larger the value of U becomes, the smaller the difference of the average total penalties among different server's initial WPs. This is because the larger U leads to extending a server's moving range limited by its initial WP.

V. Conclusion and future work

In this paper, we formulated an integer programming model for a discrete-time off-line server locations decision in the server migration service, and derived the optimal server locations. Numerical examples showed that 1) our model achieves 36% to 49% smaller penalty than the greedy on-line algorithm when the degradation of network QoS due to server migrations is little acceptable and 2) the greedy on-line algorithm can achieve the optimal or near optimal performance when an upper bound on the network QoS degradation is large (i.e., when the network QoS degradation constraint virtually does not exist).

Our future work includes 1) investigation of the optimal length of a time-slot and 2) design of an on-line server locations decision algorithm that achieves a performance close to our integer programming model because the computational complexity of our integer programming model can be large in a practical situation.

References

[1] M. Armbrust et al., "A view of cloud computing," Communications of the ACM, vol. 53, Apr. 2010, pp. 50–58.

[2] "Amazon EC2." http://aws.amazon.com/ec2 [retrieved: Jan. 2013].

[3] A. Yamanaka, Y. Fukushima, T. Murase, T. Yokohira, and T. Suda, "Destination selection algorithm in a server migration service," in Proceedings of the 7th International Conference on Future Internet Technologies (CFI), Sept. 2012, pp. 15–20.

[4] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," in Proceedings of ACM SIGCOMM'08, Jan. 2009, pp. 68–73.

[5] S. Ranjan, J. Rolia, H. Fu, and E. Knightly, "QoS-driven server migration for Internet data centers," in Proceedings of tenth International Workshop on Quality of Service (IWQoS), May 2002, pp. 3–12.

[6] T. Hara, M. Tsukamoto, and S. Nishio, "A scheduling method of database migration for WAN environments," in Proceedings of Brazilian Symposium on Database (SBBD), Oct. 1999, pp. 125–136.

[7] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: Routers with very small buffers," SIGCOMM Computer Communication review, vol. 35, July 2005, pp. 83–90.

[8] "IBM ILOG CPLEX Optimizer." http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ [retrieved: Jan. 2013].

 ISBN: 978-1-61208-256-1

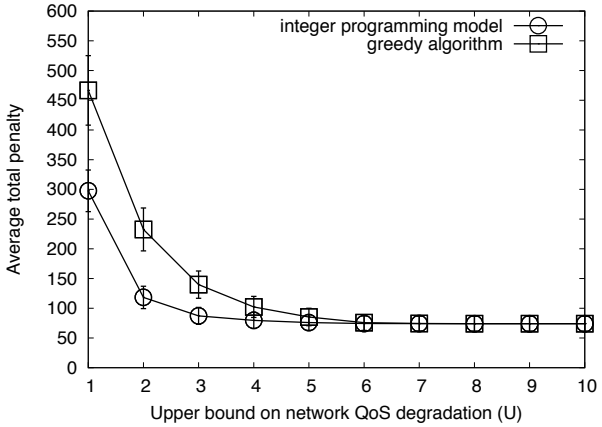

Figure 6. Average total penalty (server's initial WP: WP 1).

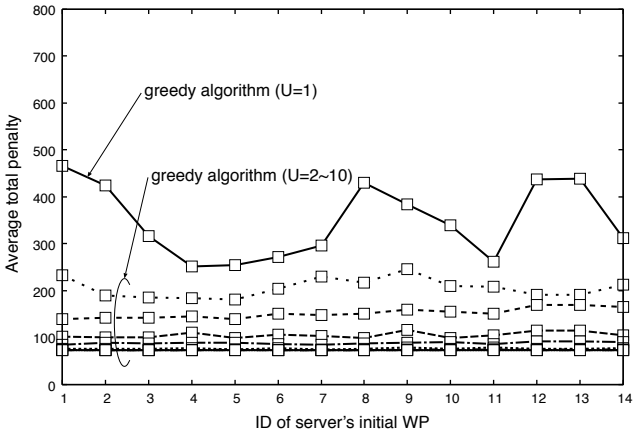

Figure 9. Average total penalty as a function of server's initial WP (greedy algorithm).

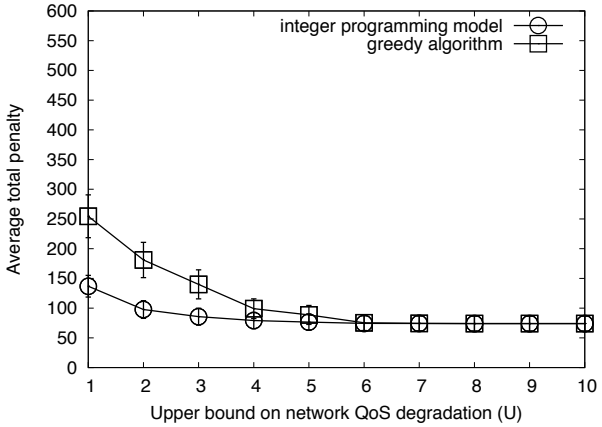

Figure 7. Average total penalty (server's initial WP: WP 5).

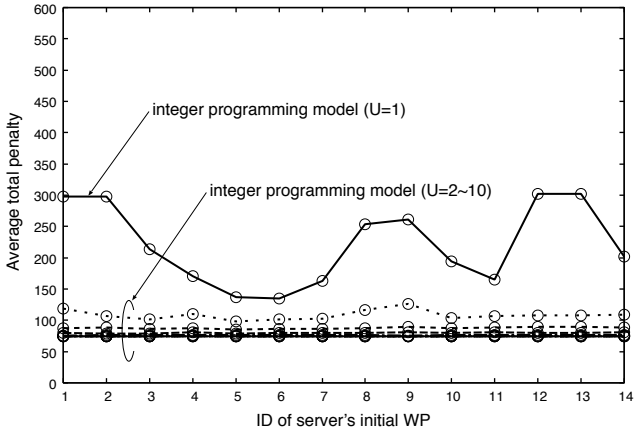

Figure 8. Average total penalty as a function of server's initial WP (integer programming model).