次世代ネットワークにおける 最低帯域保証サービスのためのプロビジョニングアルゴリズム

A Novel Provisioning Algorithm for Minimum Bandwidth Assurance Services in Future Networks

嶋村　昌義*
Masayoshi Shimamura

Email: shimamura.m.ad@m.titech.ac.jp

インターネットにおいて、アプリケーションサービス事業者の観点では、通信帯域が保証されることが望ましい。一方で、ネットワーク事業者の観点ではネットワークの利用効率を向上できることが望ましい。そのため、有限な資源である帯域をどのように割り当てるのかが重要な課題となる。特に、インターネットのトラフィックはバースト性が高く、静的な帯域割当は非効率となるため、効率的な帯域割当手法が必要となる。本稿では、最低帯域保証サービスを実現するための、プロビジョニングアルゴリズムを提案し、その有効性を示す。

The traditional virtual private network (VPN), which provides best-effort or static bandwidth allocation service, does not provide sufficient support for bursty Internet traffic. To support bursty traffic, a VPN provider can offer minimum throughput assurance (MTA) service to customers. MTA service provides higher throughput predictability than best-effort VPN service. Although there are many proposed network architectures for MTA service, certain parameters should be decided offline as part of the provisioning process. Determining adequate values of such parameters is necessary for quality of service control of bursty Internet traffic by providers. The difficulty in such provisioning is to meet the minimum throughput requirements in any active state matrices. We propose a provisioning algorithm that uses nonlinear programming for MTA service. We then numerically demonstrate the proposed algorithm and its performance.

*: 獨協大学経済学部非常勤講師

1. Introduction

Virtual private networks (VPNs) are used by many organizations as private information networks connecting distant sites at costs lower than those possible with a network of leased lines; however, VPNs have difficulty meeting quality of service (QoS) requirements because they must share network resources among all users. To overcome this problem and satisfy some of the QoS requirements, a framework called provider-provisioned VPN (PPVPN) was developed by VPN providers. Because customers contract with the provider offering PPVPN, the provider must adequately allocate limited network resources and satisfy requirements for its customers.

The current PPVPN does not support bursty Internet traffic well because it allocates network resources to customers statically. If VPN providers guarantee customers a static amount of bandwidth, the result is low bandwidth utilization. One way of supporting bursty Internet traffic is to assure the minimum throughput to customers. With minimum throughput assurance (MTA) service, a customer can attain substantially higher throughput than an agreed minimum throughput during periods of non-congestion. During periods of congestion, the minimum agreed throughput is provided as an average over a certain span of time. Therefore, MTA service can provide higher throughput predictability than best-effort VPN service.

To further describe the MTA service, we illustrate throughput allocation scenarios in Figure 1. In the figure, aggregated traffic from site A1 to A2 and from site B1 to B2 are each assigned 50 Mb/s as a minimum throughput. We denote macro flow as aggregated traffic. Figure 1(a) depicts a scenario with no congestion, i.e., the macro flow from site B1 to B2 is idle. In this case, the macro flow from site A1 to A2 can obtain more than the minimum throughput. Conversely, the throughput for the macro flow from site A1 to A2 is only 50 Mb/s when the link is congested, as illustrated in Figure 1(b); however, even in the congestion scenario, the MTA service can provide the minimum throughput for any macro flows.

Much research on MTA service has been performed. Clark et al. proposed random early drop routers with in/out bit (RIO) [1] to allocate capacity for best-effort VPN service. According to their performance evaluation, RIO can allocate capacity with slight effect on round-trip time (RTT), which is considered an important milestone in MTA service. Fabrega et al. proposed a guaranteed minimum throughput service for TCP flows using measurement-based admission control [2]. Several MTA service methods have also been proposed based on available bit rate (ABR) service of asynchronous transfer mode (ATM) networks. Li et al. proposed a core-stateless congestion avoidance scheme for IP networks [3]. Lee et al. proposed a weighted proportional fair rate allocation (WPFRA) method for differentiated service (DiffServ) on the Internet [4]. Yokoyama et al. proposed a one-way version of the WPFRA method to improve its performance [5]. Shimamura et al. proposed an extension of the WPFRA method [6] for the VPN hose model [7] which is a VPN model to utilize the network resource efficiently.

To apply the aforementioned methods, certain parameters should be decided offline. For example, RIO requires a target rate for each TCP micro/macro flow, and the WPFRA method requires the weight for each macro flow; however, no parameter determination algorithm has been proposed. The difficulty in achieving such a determination algorithm is to meet the minimum throughput requirements in any active state matrix (More detailed illustration of an active state matrix is given in Figure 4 within Section 3.2, in which each element of the matrix represents an active/idle state of traffic from a source to a destination.

In this paper, we determine adequate parameters for the MTA service, and we define this as a provisioning. Although determination algorithms for network topology and active queue management have been proposed [8, 9], they cannot satisfy the minimum required throughput of every customer in MTA service when active state matrices change dynamically, i.e., elements of the active state matrix may be idle or may frequently switch from active to idle states. Therefore, these algorithms cannot be applied to the provisioning process for the MTA service.

We propose a provisioning algorithm that uses mathematical programming, in particular, nonlinear programming (NLP), for providing MTA service in PPVPNs. Mathematical programming is widely and effectively used for network provisioning [10]. The proposed algorithm adequately distributes the limited network bandwidth to all customers within the given constraints.

Following this Introduction, this paper is organized as follows. We describe the WPFRA method in Section 2. Section 3 presents our proposed provisioning algorithm for the MTA service. We demonstrate the effectiveness of our approach with quantitative results in Section 4 and describe the difference between our studies and related studies in Section 5. Finally, we offer our conclusions in Section 6.

2. Weight proportional fair rate allocation method

In the Introduction, we argued that MTA service improves the throughput predictability of bursty traffic. To implement the MTA service, we set three requirements: 1) meet the minimum throughput requirement from customers in any active state matrix, 2) fairly distribute the spare bandwidth, and 3) keep high network resource utilization to efficiently use limited network resources.

The WPFRA method and its variants are approaches to the MTA service that has a potential to meet the three requirements above. By distributing spare bandwidth based on weight values, the WPFRA method provides weighted proportional fairness. Before using the WPFRA method, the VPN provider should decide the weight values for all combinations of source and destination sites. Deciding the weight values to satisfy the minimum throughput requirement (requirement (1) above) remains a challenge. Determining the weight values involves finding a common matrix of values that satisfy the minimum throughput requirement for all active state matrices.

Before we propose a common weight matrix determination algorithm to satisfy minimum throughput requirements for all elements of the active state matrix, we first describe the WPFRA method. The WPFRA method can distribute the available bandwidth in proportion to the weight value of each macro flow. Edge and core routers form a VPN provider's network and provide feedback-driven traffic control to utilize the network bandwidth efficiently.

Ingress and core routers periodically measure the amount of arriving traffic and calculate a fair share rate r_f that indicates the amount of allocated capacity for a macro flow with a weight of one. A fair share rate is calculated for each output link in a router. Egress edge routers periodically send a notification packet to each ingress edge router, and core routers update the value of r_f inside the notification packet if it is smaller than the current value stored in the packet. Ingress routers obtain the r_f from the received notification packet and set the value to the explicit rate ER, which indicates that the allocated throughput for a macro flow has a weight of one. Finally, they calculate the allocated throughput as ER multiplied by the customer's weight.

To illustrate how the WPFRA method determines the allocated throughput, we model the calculation process on Ref. [4] utilizing an algorithm depicted in Figure 2.

To illustrate this algorithm, we provide the example depicted in Figure 3. In this example, there are four sites: customer A has two sites, A1 and A2, and customer B has two sites, B1 and B2. The traffic of customer A (B) originates at A1 (B1) and is destined for A2 (B2). As described in the figure, the values $w_{(A1, A2)}$ and $w_{(B1, B2)}$ are assigned weights of 1 and 2, respectively. The numbers associated with links in the figure indicate the amount of bandwidth for each link. If $C_{(I1, C1)}$ is larger than 20, $B_{(A1, A2)}$ and $B_{(B1, B2)}$ become 20 and 40, respectively. Because $C_{(I1, C1)}$ is limited to 10, throughput calculation is difficult.

Suppose customers A and B are both active, meaning both are producing traffic. Then, $r_{f(I1, C1)}$, $r_{f(I2, C1)}$, and $r_{f(C1, E1)}$ converge at 10, 45, and 25, respectively, based on the above algorithm. As a result, $ER_{(I1, E1)}$ becomes 10 and $ER_{(I2, E1)}$ becomes 25. Since $w_{(A1, A2)}$ and $w_{(B1, B2)}$ are 1 and 2, respectively, $B_{(A1, A2)}$ becomes 10 and $B_{(B1, B2)}$ becomes 50. These results are summarized in Table 1. The first two columns depict the active state field, a value of 1 indicating an active state, a value of 0 indicating an idle state.

As described above, the WPFRA method determines the explicit rate ER and the allocated bandwidth B of each customer, and these values are calculated from the topology, active state matrix, and weight matrix. The network topology is generally assumed to be constant during a long time, whereas the active state and weight matrices are dynamically changing. Therefore, the provider needs to determine a weight matrix that can accommodate any active state matrix.

3. Weight determination algorithm

In the previous section, we described the WPFRA method and the difficulty of determining adequate parameters for the WPFRA method. In this section, we propose the weight determination algorithm for allocating sufficient network resource to every customer using MTA service. The proposed algorithm is divided into two sub-algorithms: a weight matrix determination algorithm for each active state matrix; and a common weight matrix determination algorithm for every active state matrix.

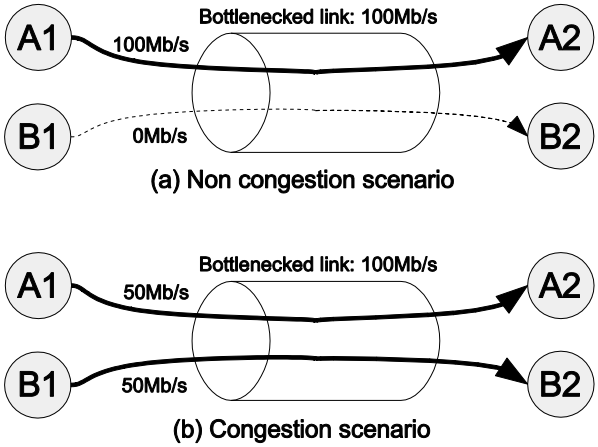


Figure 1 Throughput allocation in MTA service

Initialization

Let C be the link capacity and S be the spare bandwidth in the current iteration, which is set to

$$S_{(s,t)} \Leftarrow C_{(s,t)},$$

where (s,t) is a link directly connecting source node s and target node t.

Step 1

Let $W_{(s,t)}$ be the sum of each weight value corresponding to all macro flows passing through a link (s,t) and $r_{f_{(s,t)}}$ be the allocated capacity for a macro flow with the weight of one in the link. Then, undetermined $r_{f_{(s,t)}}$ is given by

$$r_{f_{(s,t)}} \Leftarrow \frac{S_{(s,t)}}{W_{(s,t)}}.$$

Step 2

After $r_{f_{(s,t)}}$ is calculated, explicit rate $\mathrm{ER}_{(i,e)}$ is calculated by

$$\mathrm{ER}_{(i,e)} \Leftarrow \min(r_{f_{(s,t)}}),$$

where (i,e) is a path between ingress and egress routers that contains link (s,t).

Step 3

Let w be the previously assigned weight and B be the allocated capacity of each customer given by

$$B_{(m,n)} \Leftarrow \mathrm{ER}_{(i,e)} \cdot w_{(m,n)}, \quad (1)$$

where (m,n) is a path between source and destination sites that contains path (i,e).

Step 4

Then, $S_{(s,t)}$ and $W_{(s,t)}$ are updated by

$$S_{(s,t)} \Leftarrow S_{(s,t)} - B_{(m,n)}, \quad (2)$$
$$W_{(s,t)} \Leftarrow W_{(s,t)} - w_{(m,n)},$$

where path (m,n) also contains link (s,t).

Step 5

Finally, the calculation is terminated if all $W_{(s,t)}$ values become zero: i.e., all $r_{f_{(s,t)}}$, $\mathrm{ER}_{(i,e)}$, and $B_{(m,n)}$ are determined. If not, go back to Step 1.

Figure 2 WPFRA algorithm (Conventional algorithm)

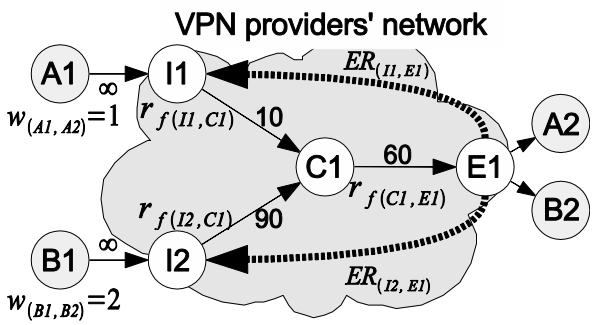


Figure 3 Example of throughput calculation: Network topology

Table 1 Throughput calculation results in Figure 2

Active state		$B_{(A1,A2)}$	$B_{(B1,B2)}$
A1 → A2	B1 → B2	($\mathrm{ER}_{(I1,E1)}$)	($\mathrm{ER}_{(I2,E1)}$)
1	1	10 (10)	50 (25)
1	0	10 (10)	0 (50)
0	1	0 (10)	60 (30)
0	0	0 (10)	0 (60)

Table 2 Example of weights calculated using NLP

Active state			Calc. weights			Alloc. bw.			Obj.	Util.
X	Y	Z	X	Y	Z	X	Y	Z	func.	
1	1	1	1	1	3	20	20	60	6.7	1.0
1	0	1	2	1	3	40	0	60	-13.3	1.0
0	1	1	1	2	3	0	40	60	-13.3	1.0
1	1	0	1	1	1	50	50	0	-8.3	1.0

Single bottlenecked link: 100

Required minimum throughput: $(X, Y, Z) = (10, 20, 60)$

3.1 Weight matrix determination algorithm

We first develop an algorithm to calculate adequate weight values based on network topology, a minimum throughput matrix, and an active state matrix. The provider needs to allocate sufficient bandwidth satisfying the minimum throughput of every customer using limited network resources. We assume these constraints as part of a mathematical programming problem in which the allocated throughput and spare bandwidth can be calculated using Eqs. (1) and (2. Let B_{ave} be the average throughput of all customers, B_{var} be the variance of the average throughput of all customers, M be the required throughput of customers, and U be the link utilization. Note that X represents a binary variable defined as follows: Set 0 in a case of idle state and set 1 in a case of active state. The objective function and constraints are formulated as follows:

Objective Function

* B_{ave} - B_{var}

Constraints

* $B_{(m,n)} \geq M_{(m,n)} \cdot X_{(m,n)}$
* $U_{(s,t)} \leq 1$
* $w_{(m,n)} \geq 1$

We adopt the objective function based on the evaluation index in Ref. [11], which specifies that

each customer can obtain high throughput and the allocated throughput among customers is balanced. Regarding alternative objective functions, we can alternatively utilize the functions maximizing the link utilization of all links in the network or maximizing the total throughput of all customers.

The allocated throughput, spare bandwidth, and utilization are all represented by polynomials. Thus, this mathematical programming can be classified as nonlinear programming (NLP). Table 2 shows an example of calculated weights. In this table, the first three columns represent an active state matrix, the second three columns represent a calculated weight matrix using NLP, the third three columns represent the allocated bandwidth calculated from the weight matrix, the fourth column represents the resulting value of the objective function, and the fifth column represents the link utilization. In the given example, three customers (X, Y, Z) share a single bottlenecked link. The required minimum throughput of (X, Y, Z) is (10, 20, 60), and the bandwidth of the bottlenecked link is 100. The calculated weight values are listed for all active state matrices, i.e., (X, Y, Z) = (1, 1, 1), (X, Y, Z) = (1, 0, 1), and so on. All resulting allocated throughput values are equal to or greater than the respective minimum throughput requirements.

3.2 Common weight matrix determination algorithm

As we saw in Tables 1 and 2, the allocated throughput is affected by the active state matrix. An illustration of this effect is shown in Figure 4. The active state matrix represents an active/idle state of traffic from a source to a destination site, i.e., the values of 0 and 1 represent idle and active states, respectively. The minimum throughput matrix describes each customer's minimum throughput requirements from source site *i* to destination site *j*. The zero element of the active state matrix results in a zero value in the minimum throughput matrix at the same element in the active state matrix.

The weight matrix consists of weight values used by the WPFRA method. When we fix the active state matrix, we can calculate a weight matrix W_k using NLP, however, if we change the active state matrix, the weight matrix $W_{k'}$ also changes. Since dynamic changes to the weight matrix is difficult implement, we aim to calculate a common weight matrix that meets the minimum throughput requirement for any active state matrix. Therefore, we need a strategy to find such a weight matrix.

To derive a common weight matrix, we iteratively use NLP. Since NLP requires an initial weight matrix, the previously derived weight matrix for an active state matrix can be used as the initial NLP weight matrix with the input of another active state matrix. We hypothesize that this iteration of NLP converges on the common weight matrix and meet the minimum throughput requirements. Based on this hypothesis, we define a basic strategy to obtain the common weight matrix detailed as follows:

Iteration: Previous output values (i.e., the weight matrix calculated by NLP) are utilized as an initial weight matrix in the next step.

One distance shift: First, we define the number of elements that differ between two active state matrices as a distance. Through the iteration of active state matrices, we need to shift an active state matrix to other active state matrices. To avoid drastic shifts in active state matrices, we shift the current active state matrix to the next active state matrix by a distance of 1. For example, suppose an active state matrix is (0, 1, 1), where 0 and 1 represent idle and active states, respectively, we set the next active state matrix to (0, 1, 0) because only the third element in these two active state matrices differ, thus the difference is 1.

Normalization: The weights represent a proportional ratio of the allocated bandwidth, i.e., the calculated weight matrix *(a, b, c)* is equivalent to *(ka, kb, kc)*, where *k* is a constant number. The allocated bandwidth calculated by these two weight matrices are the same. Therefore, we divide the weight matrix into collision groups, then normalize the grouped elements of the weight matrix, such as *(ka, kb, kc)/k → (a, b, c)*, in each collision group.

Stabilization: An element of the weight matrix corresponding to the zero element in the active state matrix can be selected for any value. To avoid needless change in weight values in such cases, we define constraints for such elements, i.e., we select the value obtained in the previous iteration.

Search domain reduction: To avoid irrelevant searching, we set lower and upper limits on the variables of the objective function. The lower limit is 1, derived directly from the constraints; the upper limit is $(C_{min} - M_{min})/M_{min}$, where C_{min} represents the minimum value of the link bandwidth in the topology and M_{min} represents the minimum value of the minimum required throughputs. To explain the reason of the upper limit, we suppose that the following: 1) multiple customers share the link with capacity C_{min} in the network, 2) one customer requires the minimum required throughput M_{min}, 3) the minimum weight value 1 is assigned to this customer, and 4) the total weight value α is assigned to other customers. In this case, because the limited link capacity C_{min} should be shared among multiple customers, the following equation $1/(1+\alpha) \cdot C_{min} \geq$

M_{min} (i.e., $\alpha \leq (C_{min} - M_{min})/M_{min}$) should be satisfied.

We construct our common weight determination algorithm based on the above strategy and illustrate it in Figure 5 using pseudocode. In our algorithm, we determine a representative weight matrix with values selected from a large number of calculated weight matrices utilizing a convergence index, which is defined as the maximum ratio of the number of common weight matrices to the number of all weight matrices. We call this index the batched convergence index.

We show example behavior of the proposed algorithm in the context of Table 3. We suppose the required minimum throughput matrix of customers (X, Y, Z) is (10, 20, 60), respectively. There is a single bottleneck link with link capacity 100. In the first cycle of the proposed algorithm, after initializing the weight matrix to (1,1,1), four weight matrices (1,2,3), (1,1,3), (1,1,3), and (2,1,3) are calculated via NLP. Note that we skip non-congestion active state matrices because the weight matrices in such active state matrices can be arbitrary values. We denote such skipped values by (-,-,-).

The highest distribution of batched tuples of (1, 1, 3) is 0.5. Therefore, we select the weight matrix (1,1,3) as the representative weight matrix in the first cycle. This representative weight matrix is utilized as the initial weight matrix in the second cycle. Similar to the first cycle, four weight matrices are calculated via NLP, and the representative weight matrix (1,1,3) is selected in the second cycle. In this example, the weight matrix (1,1,3) is suitable to every active state matrix.

Users of our provisioning algorithm should specify the required minimum throughput for each customer site so that the weight matrix satisfying minimum throughput requirements can be obtained. The fundamental calculation complexity is $l\prod_{k=1}^{u} 2^{v_k(v_k-1)}$, where u is the number of customers, v is the number of sites per customer, and l is the number of calculation cycles.

```
LoopNLP(iteration){
   InitAllWeights(weights);
   SetInputValue(topology, reqbw);
   for ( i=0; i<iteration; i++ ) {
     foreach (pattern) {
       weights = RunNLP(topology,
         pattern, reqbw, weights);
     }
     normalization(pattern, weights);
   }
 }

 RunNLP(t, p, b, w){
   foreach (p) {
     if( active[i] == 0 ){
       # Stabilize value of w[i]
       SetConstraint(w[i]);
     }
   }
   SetSearchDomain(t, b);
   ans = SolveNLP(t, p, b, w);

   return ans;
 }
end{alltt}

 begin{alltt}
 InitAllWeights(weights){
   weights = (1, 1, ..., 1);
 }

normalization(pattern, weights){
   group = divide_group(pattern);

   foreach (group) {
      normalized_w[i] =
          weights/min(weights);
   }
   return normalized_w[i];
 }
```

Figure 5 Pseudocode of the weight determination algorithm

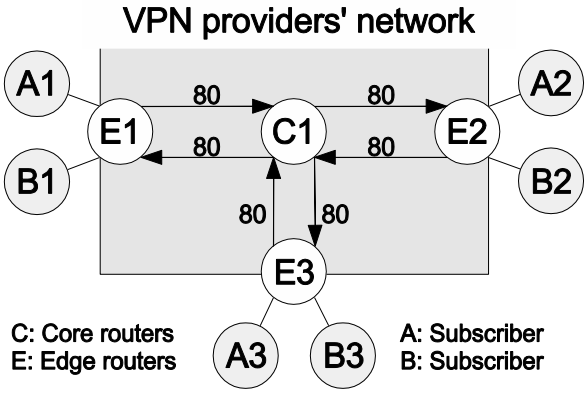


Figure 6 Experimental topology

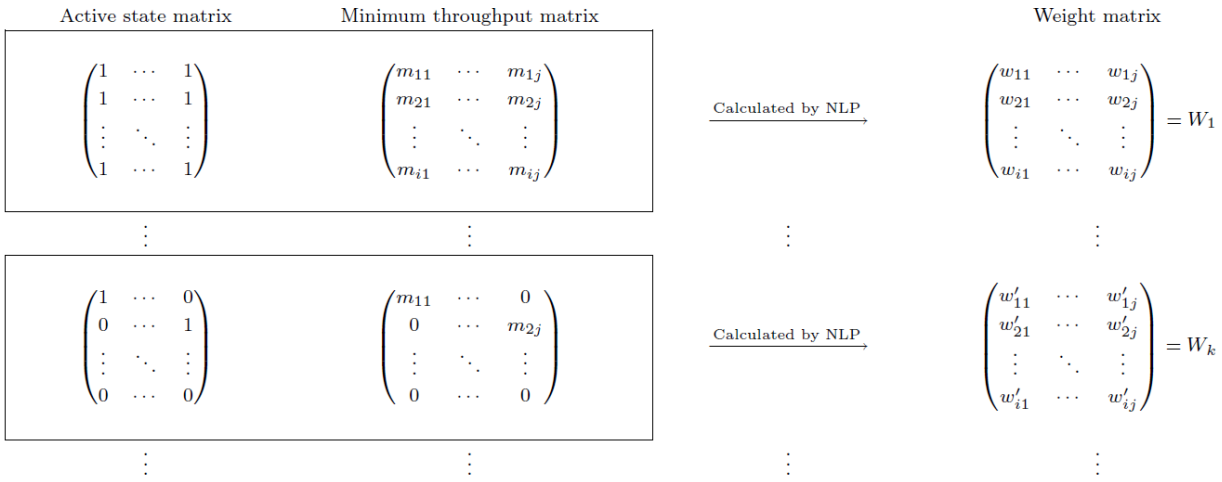


Figure 4 Common weight matrix

Table 3 Example behavior of the proposed algorithm

First cycle

Initial weight matrix in first cycle: $(X, Y, Z) = (1, 1, 1)$

Active state			Calc. weight			Alloc. bw.			Obj.	Util.
X	Y	Z	X	Y	Z	X	Y	Z	func.	
0	0	1	–	–	–	–	–	–	–	–
0	1	1	1	2	3	0	40	60	-13.3	1.0
0	1	0	–	–	–	–	–	–	–	–
1	1	0	1	1	3	50	50	0	-8.3	1.0
1	1	1	1	1	3	20	20	60	6.7	1.0
1	0	1	2	1	3	40	0	60	-13.3	1.0
1	0	0	–	–	–	–	–	–	–	–

Rep. weight matrix in 1st cycle: $(A, B, C) = (1, 1, 3)$

Second cycle

Initial weight matrix in second cycle: $(X, Y, Z) = (1, 1, 3)$

Active state			Calc. weight			Alloc. bw.			Obj.	Util.
X	Y	Z	X	Y	Z	X	Y	Z	func.	
0	0	1	–	–	–	–	–	–	–	–
0	1	1	2	2	3	0	40	60	-13.3	1.0
0	1	0	–	–	–	–	–	–	–	–
1	1	0	1	1	3	50	50	0	-8.3	1.0
1	1	1	1	1	3	20	20	60	6.7	1.0
1	0	1	2	1	3	40	0	60	-13.3	1.0
1	0	0	–	–	–	–	–	–	–	–

Rep. weight matrix in 2nd cycle: $(A, B, C) = (1, 1, 3)$

Single bottlenecked link: 100
Required minimum throughput: $(X, Y, Z) = (10, 20, 60)$

4. Numerical demonstration

In the previous section, we presented our common weight matrix determination algorithm, which is based on the hypothesis that weight matrices converge through iterations of NLP. In this section, we demonstrate that the proposed algorithm converges on a common weight matrix that meets the minimum throughput requirements. We also investigate the impact of imbalanced minimum throughput requirements and whether links have spare bandwidth available.

We use the network topology illustrated in Figure 6 containing one core and three edge routers, which together form a star topology with a link capacity of 80 on all links. All edge routers behave as both ingress and egress routers, i.e., traffic is bidirectional between each of a particular customer's sites. There are two customers (A, B), each of whom has three sites (A1, A2, A3) and (B1, B2, B3). Customer sites are connected to respective edge routers.

In our numerical demonstration, we vary the required minimum throughput to investigate the impact of the imbalance ratio and the amount of the minimum throughput as follows:

(a) $A_{(x,y)} : A_{(x,z)} : B_{(x,y)} : B_{(x,z)} = 20 : 20 : 20 : 20$
(b) $A_{(x,y)} : A_{(x,z)} : B_{(x,y)} : B_{(x,z)} = 10 : 10 : 10 : 10$
(c) $A_{(x,y)} : A_{(x,z)} : B_{(x,y)} : B_{(x,z)} = 30 : 10 : 10 : 30$
(d) $A_{(x,y)} : A_{(x,z)} : B_{(x,y)} : B_{(x,z)} = 15 : 5 : 5 : 15.$

There is spare bandwidth in scenarios (b) and (d); further, the required minimum throughput values are imbalanced in scenarios (c) and (d).

To evaluate the proposed algorithm, we define the following two evaluation indices: (1) the success rate, which is the number of cases satisfying the minimum required throughput using the representative weight matrix in each active state matrix; and (2) the average link bandwidth defined as the average of the ratio of the total throughput to the total capacity of active links. We calculate these values using the representative weight matrix calculated using our proposed algorithm.

Figure 7 shows results for the four required minimum throughput scenarios listed above. The *x*-axis represents the amount of processing from 1 to 10 iterations. The *y*-axis represents the resulting values of the batched convergence index, the success rate, and the average total link utilization.

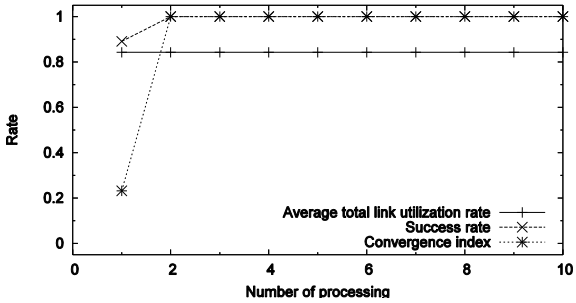
As the number of iterations increases, the batched convergence index converges, indicating that the representative weight matrix is, in fact, determined through such iterations. The average total link utilization remains constant at approximately 0.85,

although the number of iterations increases. This shows that the proposed algorithm retains the fundamental advantage of mathematical programming: i.e., the proposed algorithm can keep the overall average total link utilization under the given constrained, even as the number of iterations increases.

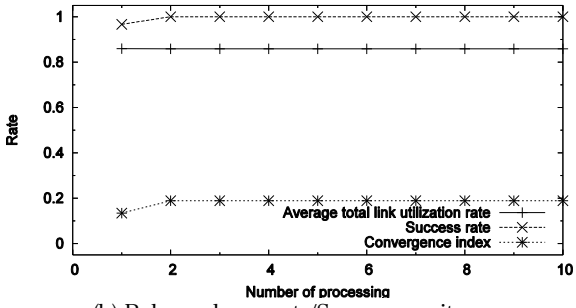
We found that the success rate progressively converges to 1.0 for all required throughput, showing that the proposed algorithm can determine the common weight matrix from the required minimum throughput matrix.

Next, we describe the impact of spare capacity on the success rate. Under spare capacity scenarios (b) and (d), the batched convergence index and the success rate is substantially lower than that of scenarios (a) and (c); this occurs because the calculated weights vary over a wide range due to the high latitude of values.

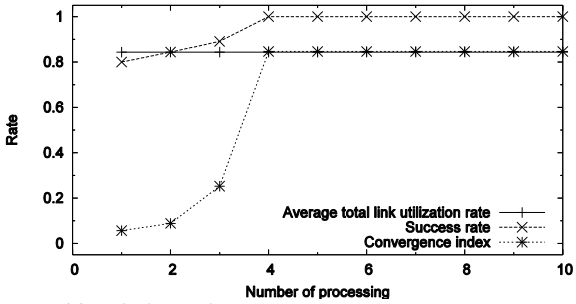
Finally, we discuss the impact on the success rate of imbalanced required throughput among sites. Our proposed algorithm requires a relatively larger number of iterations for the success rate to converge in imbalanced request scenarios (c) and (d); this occurs because of the complicated network topology and requests, although it also demonstrates the effectiveness of our iterative algorithm.



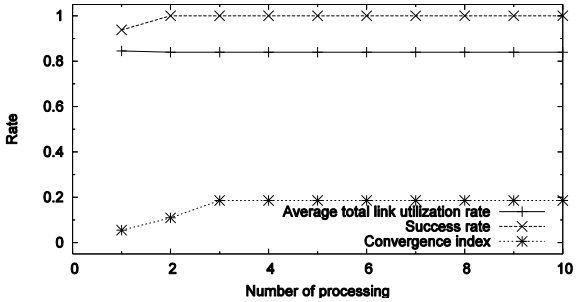
(a) Balanced requests/No spare capacity



(b) Balanced requests/Spare capacity



(c) Imbalanced requests/No spare capacity



(d) Imbalanced requests/Spare capacity

Figure 7 Success rate of the batched convergence index

5. Related work

In this section, we describe the difference between our proposed algorithm and two conventional provisioning algorithms.

The first conventional algorithm [8] can determine adequate network topology for the VPN hose model [7]. It can reduce the required bandwidth and achieve high bandwidth efficiency. Although this algorithm is helpful in constructing the network topology, it cannot achieve the MTA service because the algorithm cannot allocate network resources to the customers adequately.

The second conventional algorithm [9] adjusts parameters of a weight fair queuing (WFQ) scheduler. This algorithm achieves fairness for a single output link at the WFQ-supported router; however, it cannot support allocating available bandwidth across the entire network. Therefore, this algorithm cannot adapt to provide the MTA service.

6. Conclusion

To satisfy minimum throughput requirements of VPN customers, we proposed a provisioning algorithm that supports the MTA service. Our proposed algorithm is a necessary tool for QoS traffic control by providers. In our proposed algorithm, we first created an NLP to find a weight matrix for each active state matrix. Then we proposed a common weight matrix determination algorithm based on the hypothesis of convergence

in elements of the iteration of the NLP. Through preliminary performance evaluations, we showed indications of possibilities, i.e., the proposed algorithm can converge on a common weight matrix using the convergence index. We also evaluated the impact on convergence of the imbalanced amount of required throughput and the presence of spare bandwidth. In future work, to concentrate the convergence index more precisely and rapidly, we will focus on a pruning algorithm and consider highly correlated convergence indices. Furthermore, we attempt to apply the proposed algorithm to virtual network resource embedding problem in network virtualization environment [12].

References

(1) D. Clark and W. Fang: “Explicit allocation of best-effort packet delivery service”, IEEE/ACM Trans. Networking, vol. 6, no. 4, pp. 362-373 (1998).
(2) L. Fabrega, T. Jove, P. Vila and J. Marzo: “A guaranteed minimum throughput service for TCP flows using measurement-based admission control”, International Journal of Communication Systems, vol. 20, no. 1, pp. 43-63 (2007).
(3) J. Li and J. Liang: “A novel core-stateless ABR-like congestion avoidance scheme in IP networks”, International Journal of Communication Systems, vol. 18, no. 5, pp. 427-447 (2005).
(4) C. Lee, C. Chen and Y. Chen: “Weighted proportional fair rate allocations in a differentiated services network”, IEICE Trans. Communications, vol. E85-B, no. 1, pp. 116-128 (2002).
(5) T. Yokoyama, K. Iida, H. Koga and S. Yamaguchi: “Proposal for adaptive bandwidth allocation using one-way feedback control for MPLS networks”, IEICE Trans. Communications, vol. E90-B, no. 12, pp. 3530-3540 (2007).
(6) M. Shimamura, K. Iida, H. Koga, Y. Kadobayashi and S. Yamaguchi: “Hose bandwidth allocation method to achieve minimum throughput assurance service for provider provisioned VPNs”, IPSJ Journal, vol. 49, no. 12, pp. 3967-3984 (2008).
(7) N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan and J. Merwe: “Resource management with hoses: point-to-cloud services for virtual private networks”, IEEE/ACM Trans. Networking, vol. 10, no. 5, pp. 679-692 (2002).
(8) Y. Liu, Y. Sun and M. Chen: “MTRA: An on-line hose-model VPN provisioning algorithm”, Telecommunication Systems, vol. 31, no. 4, pp. 379-398 (2006).
(9) R. Liao and A. Campbell: “Dynamic core provisioning for quantitative differentiated services”, IEEE/ACM Transactions on Networking, vol. 12, no. 3, pp. 429-442 (2004).
(10) J. Kenningtona, E. Olinicka and G. Spirideb: “Basic mathematical programming models for capacity allocation in mesh-based survivable networks”, Omega, vol. 35, no. 6, pp. 629-644 (2007).
(11) T. Ishida, K. Ueda and T. Yakoh: “Fairness and utilization in multipath network flow optimization”, Proc. IEEE International Conference on Industrial Informatics 2006, pp. 1096-1101 (2006).
(12) N. Chowdhury and R. Boutaba: “A survey of network virtualization”, Computer Networks, vol. 54, no. 5, pp. 862-876 (2010).

(2012 年 9 月 21 日受付)
(2012 年 12 月 19 日採録)