On Service Replication Strategy for Service Overlay Networks

Kevin Y.K. Liu[1], John C. S. Lui[1], and Zhi-Li Zhang[2]

[1] Dept. of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong
{ykliu,cslui}@cse.cuhk.edu.hk
[2] Dept. of Computer Science, University of Minnesota, Minneapolis, MN 55455
zhzhang@cs.umn.edu

Abstract. *The service overlay network (SON) is an effective means to deliver end-to-end QoS guaranteed applications on the current Internet. In [5], authors address the bandwidth provisioning problem on a SON, specifically, in determining the appropriate amount of bandwidth capacity to purchase from various autonomous systems so as to satisfy the QoS requirements of the SON's end users and at the same time, maximizes the total revenue of operating the overlay network. In this paper, we extend the concept of the service overlay network. Since traffic demands are time varying and there may be some unexpected events which can cause a traffic surge, these will significantly increase the probability of QoS violation and will reduce the profit margin of a SON. To overcome these problems, we propose to replicate services on the service gateways so as to dynamically adapt to these traffic surges. We show that the service replication problem, in general, is intractable. We propose an efficient service replication algorithm which replicates services for a subset of traffic flows. Under our replication strategy, one does not need to increase the bandwidth capacity of underlying links and at the same time, be able to increase the average profit for the overlay network. Experiments are carried out to illustrate that replication algorithm provides higher flexibility during traffic fluctuations and can quickly find a near-optimal solution.*

Keywords: provisioning and quality assurance, quality of Service Management, Overlay networks, virtual topologies and VPN services.

1 Introduction

The Internet is being used for many different user activities, including emails, software distribution, audio/video entertainment, e-commerce, and real-time games. Although some of these applications are designed to be adaptive to available network resources, they still expect different levels of services from the network in order to have good performance. However, the primary service provided by the Internet

0-7803-8230-7/04/$20.00

is the *best-effort* service model which does not perform any service differentiation. Therefore end-to-end quality-of-service (QoS) guarantees are difficult to maintain. Another reason for the difficulty in providing end-to-end QoS guarantees is that the Internet is organized as many different autonomous systems (**AS**s) wherein each AS manages its own traffic, performance level and internal routing decisions. These autonomous systems have various bilateral business relationships (e.g., peering, provider-customer) for traffic exchange so as to maintain the Internet global connectivity. Since data traffic usually traverses multiple autonomous systems, it is difficult to establish *multi-lateral* business relationship which spans many autonomous systems. Therefore, services which need end-to-end QoS guarantees are still far from realization and the above mentioned problems hindered the deployment of many time sensitive services on the Internet.

In [5], authors advocate the notion of *service overlay network* (SON) as an effective mean to address problems of providing end-to-end services. A SON is an overlay network that spans many autonomous systems. In general, the SON purchases bandwidth with certain QoS guarantees from all **AS**s that the overlay network spans. This way, a logical end-to-end service delivery infrastructure can be built on top of the existing network infrastructure. On this logical overlay network, SON provides different types of time sensitive services (e.g., video-on-demand, video/audio multicast, VoIP, . . ., etc). SON offers these services to different users who pay the SON for using these value-added services. [11, 8, 4, 2, 12] are some examples of SON.

In deploying a service overlay network, one has to address the bandwidth provisioning problem so as to reduce the operating cost and to maximize its profit. In particular, one needs to decide the appropriate amount of bandwidth to purchase from the underlying autonomous systems so as to provide end-to-end value-added QoS sensitive services to different users and at the same time, recovering the cost of deploying an overlay network. In [5], authors formulate a mathematical model for the bandwidth provisioning problem by considering the stochastic traffic demand distribution, bandwidth costs and the level of QoS guarantees. Note that once the bandwidth provisioning is carried out, the overlay network is committed to a topology wherein each link l in the overlay network has a fixed bandwidth capacity of c_l (units in Mbps). The capacity of each link is fixed until the next bandwidth provisioning instant[3].

Since traffic is time varying and stochastic in nature, it is possible that there will be a sudden surge on traffic due to some unexpected event (e.g., a popular pay-per-view sport or musical event). This traffic surge may not be well-represented or characterized in the original measured traffic distribution that was used for the bandwidth provisioning. In this case, the allocated bandwidth for the SON may not be sufficient to provide the end-to-end QoS guarantees. This translates to lower profit for the SON operator since the operator needs to pay for the penalty for these QoS violations.

[3] In [5], the authors also address the dynamic bandwidth provisioning problem, however it is technically difficult to implement [7].

To solve this problem, we propose a methodology to replicate services on nodes of a SON so as to make the overlay network more adaptive to traffic flow variation. Rather than using the dynamic bandwidth provisioning approach as suggested in [5], we propose to use the static bandwidth provisioning cost model (which is usually more cost effective than the dynamic bandwidth provisioning cost model) and dynamically replicate services on the SON so as to reduce the operating cost and reduce the probability of violating the end-to-end QoS guarantees.

The paper is organized as follows: In Section 2, we introduce the background and the architectural framework of the service overlay network. In Section 3 we formulate the replication problem of a SON and present our replication algorithm. In Section 4 we present the numerical experiment results and show the effectiveness of the replication algorithm. Finally, we conclude our paper in Section 5.

2 Background on SON

In this section, we provide the necessary background on a service overlay network and the bandwidth provisioning problem.

Table 1 illustrates the notations used for describing the bandwidth provisioning problem.

Parameter	Remarks
$\mathcal{N}$	the set of all nodes in the SON.
$\mathcal{L}$	the set of all links in the SON.
$\mathcal{R}$	the set of all source-destination (SD) paths in the SON.
ρ_l	a non-negative random variable describing the traffic flow on link $l \in \mathcal{L}$.
$\bar{\rho}_l$	average traffic flow on link $l \in \mathcal{L}$.
c_l	allocated capacity (Mbps) on link $l \in \mathcal{L}$.
$\Phi_l(c_l)$	cost per unit of time for reserving c_l amount of bandwidth for link $l \in \mathcal{L}$.
e_r	revenue for carrying one unit of traffic flow along a SD path $r \in \mathcal{R}$.
π_r	penalty of QoS violation for one unit traffic flow on SD path $r \in \mathcal{R}$.

Table 1. Notations

A SON is a logical overlay network with a set of nodes $\mathcal{N}$ and a set of links $\mathcal{L}$. Each node in $\mathcal{N}$ is a *service gateway* which performs service-specific data forwarding and control functions. A service gateway is a physical end host on the Internet, for example, a server controlled and managed by the SON operator. A link in $\mathcal{L}$ is a logical connection between two service gateways and this logical link is an IP level path provided by the underlying autonomous system(s). The advantages of the SON architectural framework are: 1) one can purchase different bandwidth for different links in the SON; and 2) one can bypass congested peering points among ASs and thereby provide end-to-end QoS guarantees.

Bandwidth provisioning problem [5, 6, 10] is one of the major issues in designing a service overlay network. To guarantee the delivery of end-to-end services, the SON needs to purchase sufficient amount of bandwidth for each (logical) link from different underlying ASs so that certain QoS guarantees can be maintained. The bandwidth provisioning problem for a SON is to determine the appropriate amount of bandwidth to purchase for each of its links such that the QoS sensitive traffic demand for any source-destination path in $\mathcal{R}$ can be satisfied and at the same time, the total net profit of the SON is maximized.

In [5, 6], authors provide the formal mathematical framework for performing the bandwidth provisioning. Given a network topology $\mathcal{N}$ and $\mathcal{L}$, the source-destination (SD) path requirements in $\mathcal{R}$, the stochastic traffic demand $\{\rho_r\}$ for each $r \in \mathcal{R}$, and the routing method, the model [5, 6] provides an *lower bound* of expected net profit for the service overlay network.

Let r denote a path in the source-destination path set $\mathcal{R}$. Assume that the traffic demand distribution on path r is known [4] and traffic of all paths in $\mathcal{R}$ is described by the stochastic traffic demand matrix $\{\rho_r\}$, the total net income for the SON, denoted by the random variable W, can be expressed as:

$$W(\{\rho_r\}) = \sum_{r\in\mathcal{R}} e_r\rho_r - \sum_{l\in\mathcal{L}} \Phi_l(c_l) - \sum_{r\in\mathcal{R}} \pi_r\rho_r B_r(\{\rho_r\}). \quad (1)$$

where $\sum_{r\in\mathcal{R}} e_r\rho_r$ is the total revenue received by a SON for carrying $\{\rho_r\}$ traffic in $\mathcal{R}$; $\sum_{l\in\mathcal{L}} \Phi_l(c_l)$ is the total bandwidth cost that a SON must pay to all its underlying autonomous systems; $\sum_{r\in\mathcal{R}} \pi_r\rho_r B_r(\{\rho_r\})$ is the total penalty that a SON suffered when the QoS guarantees for those traffic demands are violated. The variable B_r represents the probability that QoS guarantees for the traffic along path r is violated. The problem of bandwidth provisioning can thus be formulated as the optimization of the average total net profit $E(W)$, i.e.:

$$\max_{c_l} E(W). \quad (2)$$

In other words, determining the appropriate amount of capacity $\{c_l\}$ for each link $l \in \mathcal{L}$.

To derive the lower bound on the average net profit, one additional notation is introduced: define a very small real number δ, for each SD path r, define $\hat{\rho}_r > \bar{\rho}_r$ such that $\int_{\hat{\rho}_r}^{\infty} \rho_r d\rho_r \leq \delta$. Therefore, $\Pr\{\rho_r \geq \hat{\rho}_r\} \leq \delta/\hat{\rho}_r$. This basically says that $\hat{\rho}_r$ is such that the probability the traffic demand along r exceeds $\hat{\rho}_r$ is very small, and thus negligible. Then, $E(W)$ is lower bounded by $V(\mathcal{L}, \mathcal{R})$, wherein:

$$V(\mathcal{L},\mathcal{R}) = \sum_{r\in\mathcal{R}} e_r\bar{\rho}_r - \sum_{l\in\mathcal{L}} \Phi_l(c_l) - \sum_{r\in\mathcal{R}} \pi_r\bar{\rho}_r B_r(\hat{\rho}_r) - \sum_{r\in\mathcal{R}} \pi_r\delta_r(1+\sum_{r'\neq r} \frac{\bar{\rho}_r}{\hat{\rho}_{r'}}). \quad (3)$$

Intuitively, Equation (3) can be interpreted as follows: the first term represents the earning of the SON, the second term is the cost for bandwidth provision on all the

[4] This traffic demand distribution can be obtained, for example, through long-term observation and measurement of past traffic history.

links, the third term is the QoS violation cost conditioning on that the traffic on each route is bounded by $\hat{\rho}_r$, and the last term is the QoS violation cost for those extremal traffic, i.e. the traffic exceeding the bound $\hat{\rho}_r$. Given this model, we then compute the optimal bandwidth allocation c_l^* for all link $l \in \mathcal{L}$, by solving that $\frac{\partial V}{\partial c_l} = 0$. Thus, one can find the optimal bandwidth provision of each link $l \in \mathcal{L}$ which maximizes V, the least average profit for a service overlay network.

Note that the above mentioned bandwidth provisioning is only practical in an off-line manner. That is, once bandwidth is provisioned, it cannot be changed until the next bandwidth provisioning instance. However, due to difficulties in implementation and in adjusting the multi-lateral agreements, the period between two bandwidth provisioning instants is usually quite long (e.g., days, weeks or even months). However, during this period, the traffic demand of a SON could fluctuate. This is especially true for a SON that spans a large geographical area where the time-of-day effect is significant, e.g., some part of the network is congested during rush hours, while other part of network is very lightly loaded because it is at a different time zone. Also, it is possible that there may be a surge in traffic demand due to some unexpected events, e.g., a popular pay-per-view sport or musical event that attracts many users. The variation of traffic flow will increase the QoS violation probability B_r. Therefore, it is crucial for the SON to have the adaptive capability to traffic flow fluctuation. In this paper, we propose to *dynamically replicate services* within a SON so as to reduce the traffic demands on "overloaded" links and to maximize the net income of an SON operator.

3 Replication in SON

Since the time scale of two consecutive bandwidth provisioning instances is generally quite long, while the traffic demand could change during this period due to some unexpected events, therefore, it is crucial for an overlay network to have the adaptive ability to such traffic demand fluctuation. One way to solve this problem via the static bandwidth provisioning method is to provision more bandwidth for each link in the SON (e.g., having a smaller value of δ and larger value of $\hat{\rho}_r$ in Equation (3)). However, the drawback of this approach is that one has to pay a much higher cost for bandwidth provisioning. In this paper, we propose a service replication approach which makes the SON more flexible and adaptive to traffic variation without purchasing extra bandwidth resource from the underlying autonomous systems.

Note that *service gateway* inside a SON is a network host managed by the SON operator. The service gateway has sufficient storage space and processing power to perform the basic packet forwarding function as well as some service-specific functions (e.g., video-on-demand service). The replication strategies make use of these service gateways and extend their functionalities. Therefore, each service gateway can be a potential server and deliver the content to users in the SON. In the following, we present the service replication problem.

Given the source-destination paths in $\mathcal{R}$, the stochastic traffic demands $\{\rho_r\}$ for all $r \in \mathcal{R}$, one can choose a set of demands in $\mathcal{R}$ to replicate. An SD path $r \in \mathcal{R}$

consists of a source node s_r, a destination node d_r, and its stochastic traffic demand ρ_r along the path r. It is important to point out that a destination node may consist of a large number of users, i.e., a set of users within the same network edge who wants to receive a video-on-demand service. In the following context, we use γ to denote one *replication* event. We also introduce the following notations:

$loc(\gamma)$: the node which γ chooses to install the replicated service.
$target(\gamma)$: the SD path that γ chooses to replicate.
$path(\gamma)$: the new path taken by γ to deliver the replicated service.
$\beta(\gamma)$: the fraction of traffic we need to shift from $target(\gamma)$ onto $path(\gamma)$.

Consider Figure 1, suppose the replication event γ is for a SD path $r \in \mathcal{R}$ and we choose node $i \in \mathcal{N}$ to install the replicated service, then $target(\gamma) = r$ and $loc(\gamma) = i$. Let the average traffic demand on r be $\bar{\rho}_r$. After the replication process, the traffic demand on r will decrease because $loc(\gamma)$ is serving some of the clients in r. Therefore, the average traffic demand on r after the replication process is $\bar{\rho}_r(1 - \beta(\gamma))$. The replication process γ will create a new $path(\gamma)$ with source node in $loc(\gamma)$ and destination node in d_r for the replicated service. The traffic on this new path needs to deliver, on the average, $\beta(\gamma)\bar{\rho}_r$ amount of traffic to a set of users in r.

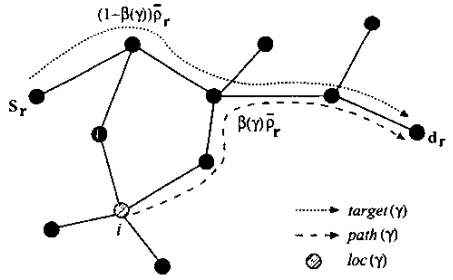

Fig. 1. Illustration of replication event in a SON.

In general, each node $i \in \mathcal{N}$ may only be able to support a finite set of services. For example, if path r_a carries service a while path r_b carries service b. It is possible that a certain node i can only provide service a but not b due to some storage or processing constraint. Therefore this node i can target for path r_a, but not r_b. So we denote S_i as the set of paths that node i can target for.

Let $\mathcal{D}$ denote the set of all replication events γ. Let $\mathcal{R}'$ denote the set of all source-destination paths of the SON after the replication events $\mathcal{D}$. The single source, single destination replication is to find a set of replication events $\mathcal{D}$ which maximizes the increase in (the lower bound V of) the total net income $E(W)$ of SON by performing

service replication, i.e., to maximize the following objective function [5] :

$$\begin{aligned} &\max_{\mathcal{D}} V(\mathcal{R}') - V(\mathcal{R}) \\ \text{subject to:} \quad &loc(\gamma) \in \mathcal{N}, \\ &target(\gamma) \in S_{loc(\gamma)}, \\ &0 \le \beta(\gamma) \le 1 \end{aligned}$$

Since the replication will not change the sum of all the traffic demands for the SD paths in $\mathcal{R}$ and the total bandwidth cost, we have,

$$\begin{aligned} V(\mathcal{R}') - V(\mathcal{R}) = &\sum_{r\in\mathcal{R}} \pi_r \bar{\rho}_r B_r(\hat{\rho}_r) - \sum_{r\in\mathcal{R}'} \pi_r \bar{\rho}_r B_r(\hat{\rho}_r) \\ &+ \sum_{r\in\mathcal{R}} \pi_r \delta_r \left(1 + \sum_{r'\neq r} \frac{\bar{\rho}_r}{\hat{\rho}_{r'}}\right) - \sum_{r\in\mathcal{R}'} \pi_r \delta_r \left(1 + \sum_{r'\neq r} \frac{\bar{\rho}_r}{\hat{\rho}_{r'}}\right) \quad (4) \end{aligned}$$

Theorem 1. *The time complexity to solve the optimal replication problem using exhaustive search is* $O(|\mathcal{N}|^{|\mathcal{R}|})$

Proof: Denote the set of all *service gateways* as N, and $N \subseteq \mathcal{N}$. Each service gateway i has a set of supported services $S_i \subseteq \mathcal{R}$. Therefore, in the worst case, when $N = \mathcal{N}$ and $S_i = \mathcal{R}$, the exhaustive search must try all the possible replications, therefore we have $|\mathcal{N}|^{|\mathcal{R}|}$ choices. ∎

As shown in Theorem 1, the problem is in general intractable. Therefore we propose the following heuristic algorithm. Numerical examples will be given in the next section to illustrate the effectiveness of our approach.

The motivation of our replication is as follows. Note that although the replication strategy *cannot* alter the capacity of each link in SON, it may change and divert part of traffic demands from some of the *highly congested* links and redirect them to a replicated server. A *key observation* is that for a given link l, the total net income is *more sensitive* to the change of total traffic demand on this link *if* $-\frac{\partial V}{\partial \rho_l}$ *is large*. In other words, a small decrease in the traffic demand on link l can cause a large drop in the total net income of the SON. This is an idea similar to the optimal routing problem in [3] Therefore we focus on those links for which $-\frac{\partial V}{\partial \rho_l}$ is large and attempt to reduce the traffic demands on these links by service replication. In deciding which path r to select for service replication, we introduce the following notion.

[5] We assume the cost of setting-up the replicated server is a small constant overhead which does not depend on the traffi c demand, so it will not influence the total cost in the long run.

Definition 1. *Let a path r having $n \geq 1$ links $l_1, l_2, \ldots, l_n$. The "negative first derivative sum" (NFDS) of the path r is*

$$NFDS(r) = -\sum_{i=1}^{n} \frac{\partial V}{\partial \rho_{l_i}}.$$

In deciding which path to replicate, we choose a path r that has the *most negative* NFDS value.

To determine which node (i.e., service gateway) to place the replication, we adopt the following strategy. For all the traffic going to a certain user, they must go through the link connecting that user to the SON (the "last-mile" link). So to place a replication whose target is path r, we only consider *those nodes along path r*. The rationale for this approach is that if one sets up replication on nodes not along the path in r, it will increase the traffic demands on other links (which may in turn increase the probability of violating the QoS requirements for those links). Therefore, our replication strategy only targets those nodes along path r. Figure 2 illustrates this strategy. In

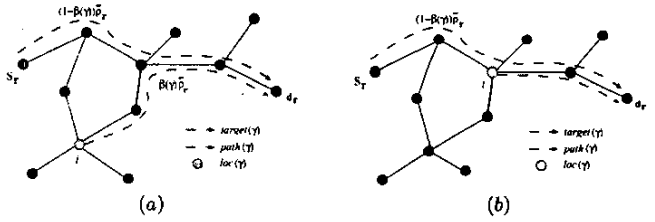

Fig. 2. Illustration of replication in SON. (a) replicating node $loc(\gamma)$ is not along the path $target(\gamma)$; (b) replicating node $loc(\gamma)$ is along the path $target(\gamma)$.

Figure 2(a), the replication γ takes on the $path(\gamma)$ and we need to increase the traffic on the first two links along $path(\gamma)$. On the other hand, Figure 2(b) illustrates that a replication at node i does not increase any extra traffic in the SON.

Based on these observations, the replication algorithm is shown in Figure 3. Our replication algorithm uses a *greedy* iterative method to compute one replication in each iteration. Steps 1-4 create a replication event γ based on the NFDS(r) for all $r \in \mathcal{R}$. Once we determine which SD path r to replicate, we need to determine the fraction of traffic shift from $target(\gamma)$ to $path(\gamma)$. From theorem 2, we know $\beta(\gamma)$ can be determined if B_l is known. Step 9 tests whether the current replication will benefit the SON. If there is any benefit, it will continue to the next iteration, else it will stop and output the total income as well as the set of replication events.

Replication Algorithm:

1. **FOR EACH** $r \in \mathcal{R}$, Compute NFDS(r)
2. Choose $r^* \in \mathcal{R}$ which has the *largest* NFDS(r)
3. Choose the last possible node $i \in \mathcal{N}$ along path r^* such that $r \in S_i$
4. Create replication γ, such that $loc(\gamma) = i$, $target(\gamma) = r^*$
5. Determine the optimal traffi c split $\beta(\gamma)$
6. $\mathcal{D} \leftarrow \mathcal{D} \cup \{\gamma\}$
7. $\mathcal{R}' \leftarrow \mathcal{R} \cup \{path(\gamma)\}$
8. Compute the lower bound average profi t $V(\mathcal{R})$
9. **IF** $(V(\mathcal{R}') > V(\mathcal{R}))$ **THEN**
10. $\quad \mathcal{R} \leftarrow \mathcal{R}'$
11. $\quad V(\mathcal{R}) \leftarrow V(\mathcal{R}')$
12. $\quad$ **GOTO** 1
13. **ELSE**
14. $\quad$ **RETURN** $V(\mathcal{R})$ and $\mathcal{D}$
15. **ENDIF**

Fig. 3. Pseudocode of the replication algorithm.

Theorem 2. *Given particular form of* B_l, $\beta(\gamma)$ *can be determined for each* $\gamma \in \mathcal{D}$ *in time* $O(|\mathcal{L}|)$.

Due to the page limit, we omit the details of the proof here, which is given in [9]. ∎

Also, our replication algorithm has the following computational complexity.

Theorem 3. *Our replication algorithm has time complexity of* $O(|\mathcal{R}||\mathcal{L}|)$.

Proof: In each iteration of our algorithm, step 1-4 can be finished in $O(|\mathcal{L}|)$ time, from Theorem 2, we can also finish step 5 in time $O(|\mathcal{L}|)$, and all the others steps only $O(1)$ time each. Since one replication serves for one SD path r. In the worst case, there are $|\mathcal{R}|$ replications. Therefore, the total time of our algorithm at the worst case is $O(|\mathcal{R}||\mathcal{L}|)$. ∎

4 Numerical Experiments

In this section we perform numerical studies for different network settings to evaluate the proposed heuristic algorithm. The first experiment shows the performance of our algorithm and the second experiment tests the scalability of our replication algorithm.

Experiment 1: (Comparison of our replication algorithm with the exhaustive search approach): In this experiment, we analyze the performance of our replication algorithm by comparing our solution with the *optimal* solution obtained by performing exhaustive search.

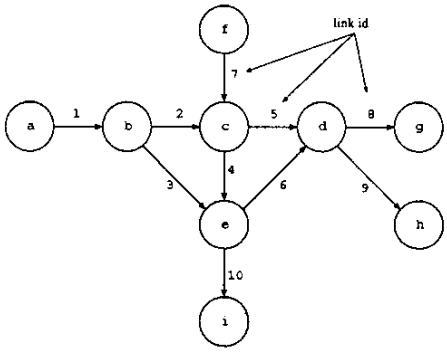

Fig. 4. Topology for experiment 1.

Figure 4 shows the SON topology used in this experiment. Table 2 shows the traffic demands of each SD requirements and links on each SD path. The bandwidth to be allocated on each link is obtained by using the static bandwidth provisioning model and shown in Table 3.

	Src-Dest	Links Used	Demand
r_1	$a \rightarrow g$	1-2-5-8	100
r_2	$a \rightarrow h$	1-3-6-9	300
r_3	$a \rightarrow i$	1-3-10	100
r_4	$f \rightarrow g$	7-5-8	200
r_5	$f \rightarrow i$	7-4-10	300

Table 2. SD paths and the traffi c demands for Experiment 1.

The expected total net income using the static bandwidth provisioning model is 4358.8. Now suppose that the traffic demands of all SD paths increase 10%. This increase in traffic demands will result in larger QoS violation penalty, and consequently the total income will be reduced to 2608.5 *without service replication*. This is a significant loss (40%) of profit to the SON.

By using our replication strategy, this situation can be considerably improved even if we only allow a single replication. Assuming that nodes b, c, d, e can accommodate any replication, thus, altogether we have 20 possible replications as illustrated in Table 4. For example, the first entry in the table implies that if we replicate path r_1 on node b, then the total net income is 2467.5. Obviously, the optimal replica-

Link ID	2	4,5,6,8,9	3,10	1,7
ρ_l	100	300	400	500
c_l^*	175	467	607	746

Table 3. Bandwidth provisioned on each link.

tion γ^* is such that $loc(\gamma) = d$, and $target(\gamma) = r_2$,namely, placing the replicated server on node d for SD path 2 will yield a total income equal to 3058.1, which reduce the loss of proft from 40% to 30%.

node	r_1	r_2	r_3	r_4	r_5
b	2467.5	2737.2	2467.5	2569.6	2614.9
c	2487.0	2670.0	2444.9	2702.4	2798.4
d	2587.7	**3058.1**	2315.7	2822.0	2583.7
e	2354.2	2908.9	2637.9	2593.4	2930.7

Table 4. All possible results for a single replication. The number in boldface (3058.1) is the optimal solution.

Using our replication algorithm, the NFDS computed for SD path r_1 to r_5 are: 9.0541, 11.2258, 8.3201, 7.6974 and 7.2453, respectively. So r_2 has the largest NFDS, and node d is the last node along the path $a \rightarrow b \rightarrow e \rightarrow d \rightarrow h$. Therefore, it will lead us to do replication for r_2 at node d, that is $loc(\gamma) = d$, and $target(\gamma) = 2$ which is same as the optimal replication γ^* as illustrated in Table 4.

If we allow up to two replications, then all the combinations of possible replications are shown in Table 5. The optimal solution is $\mathcal{D}^* = \{\gamma_1^*, \gamma_2^*\}$, where $loc(\gamma_1^*) = d$, $target(\gamma_1^*) = r_2$ and $loc(\gamma_2^*) = d$, $target(\gamma_2^*) = r_5$. The resulting net income is 3422.7.

Using our replication algorithm, we choose replication γ_1, (which is the same as γ_1^*) in the first iteration. After that, we evaluate the NFDS of each paths again, and now the value for r_1 to r_5 becomes: 7.2309, 6.8087, 4.8771, 8.2094 and 7.9012, respectively. Thus, r_4 will be chosen, and the last node along the path $f \rightarrow c \rightarrow d \rightarrow g$ path is the node d. Therefore, in our second iteration $loc(\gamma_2) = d, target(\gamma_2) = r_4$. The resulting net income is 3331.0.

Although our result is not the same as the optimal solution, it is still close to the optimal. In fact, it is the second best among all the possible choices. This implies that our replication algorithm is very efficient in placing the *near-optimal* replicated server.

Experiment 2: (Scalability Analysis): In this experiment we use the topology generator "BRITE" [1] to generate a more realistic SON network with 40 nodes and 72 links. We randomly create 50 SD paths for this SON as the services request by

replication nodes	paths r_1, r_2	paths r_1, r_3	paths r_1, r_4	paths r_1, r_5	paths r_2, r_3	paths r_2, r_4	paths r_2, r_5	paths r_3, r_4	paths r_3, r_5	paths r_4, r_5
(b, b)	2504.2	2189.2	2421.3	2418.2	2504.2	2702.0	2669.0	2421.5	2418.0	2401.4
(b, c)	2437.0	2167.5	2590.3	2717.2	2482.5	2871.2	2969.5	2590.3	2717.2	2585.0
(b, d)	2825.1	2187.5	2729.0	2502.5	2504.2	3010.1	2754.7	2728.9	2502.5	2585.0
(b, e)	2675.9	2360.4	2421.1	2849.5	2675.4	2701.1	3101.7	2421.0	2849.5	2717.2
(c, b)	2523.7	2208.7	2468.5	2437.7	2437.0	2712.4	2718.6	2277.2	2482.3	2534.2
(c, c)	2437.0	2187.0	2609.8	2736.7	2244.1	2712.4	2810.7	2446.0	2572.9	2717.7
(c, d)	2844.6	2207.0	2758.8	2522.0	2437.0	2860.0	2687.5	2584.6	2479.9	2717.7
(c, e)	2695.4	2380.0	2450.9	2869.0	2437.0	2860.0	2942.9	2159.8	3019.9	2850.0
(d, b)	2624.6	2309.6	2543.1	2586.6	2825.1	3022.9	3198.3	2305.8	2381.9	2653.8
(d, c)	2509.1	2287.9	2758.8	2885.7	2632.1	3192.2	3290.4	2438.5	2565.4	2837.3
(d, d)	2945.5	2307.7	2758.8	2622.7	2825.1	3331.0*	3075.6	2558.1	2350.7	2837.3
(d, e)	2796.3	2480.9	2450.9	3017.9	2825.1	3331.0	**3422.7**	2329.6	2697.7	2969.6
(e, b)	2390.1	2075.1	2309.6	2515.1	2675.9	2873.7	3080.2	2591.9	2482.3	2509.7
(e, c)	2509.1	1937.0	2525.3	2652.1	2482.9	3042.9	3141.2	2760.7	2887.6	2608.8
(e, d)	2945.5	2074.1	2525.3	2389.1	2675.9	3181.8	2926.4	2899.3	2672.9	2608.8
(e, e)	2445.1	2130.0	2378.8	2784.4	2675.9	2754.8	3273.4	2474.4	3019.9	2741.1

Table 5. Total income after all possible two replication events. The bold faced value (3422.7) is the optimal total income after the exhaustive search. The underlined value (3331.0) is the total income obtained by our replication algorithm, which is the second best among all possible choices.

users, and each of them has a traffic demand $\bar{\rho}_r = 400$. The link capacity are given as $c_l = 1.2\bar{\rho}_l$. The shortest-path routing is used here. Figure 5 shows the topology of this network. The total net income of this network before replication is 101210.

Table 6 compares the result after each iteration of our replication algorithm versus the optimal result that is obtained by enumerating all the possible replications. Since the search space of the true optimal solution will grow exponentially as the number of replications grows, it is not feasible to use the enumeration method to find the optimal solution. So the optimal result at iteration i is obtained by enumerating *only* the possible choices at that iteration.

Iteration	Replication Algorithm	Exhaustive Search
1	106320	106320
2	108950	108950
3	109370	109480
4	109700	110500

Table 6. Comparison of expected total income of our algorithm with exhaustive search.

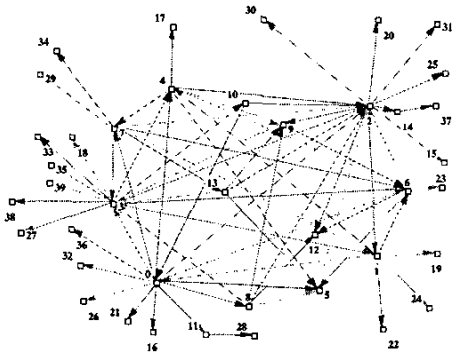

Fig. 5. A SON network with 40 nodes and 72 links.

In Table 6, we observe that for the first two iterations, our replication algorithm generates a result that has the same total income as the one produced by the exhaustive search. Even at the 3rd and 4th iterations, our result is still very close to the optimal result.

Iteration	CPU Time 1	CPU Time 2
1	0.09	82.43
2	0.11	83.32
3	0.11	84.52
4	0.10	88.28

Table 7. CPU Time 1 is the CPU time used in our replication algorithm; CPU Time 2 is the time used to compute optimal result.

Another important thing to note is the time complexity of our algorithm. Since in the real SON network, the replication process must be done dynamically to adapt to the traffic demand surge, we need an efficient algorithm to determine the right replication event in a *timely* manner. Table 7 shows the CPU time used by our algorithm and the time used by the exhaustive search. Our algorithm has much better performance, and it can still find a near-optimal result.

5 Conclusions

We have studied the bandwidth provisioning problem in a service overlay network (SON), in particular, the static bandwidth provisioning model proposed by Zhang et al [5, 6]. We extend this model by applying our replication strategy.

Since traffic demands are time varying and there may be some unexpected events which can cause a traffic surge, these will significantly increase the probability of QoS violation and will reduce the profit margin of a SON. To overcome these problems, we propose to replicate services on the service gateways so as dynamically adapt to these traffic surges. We propose an efficient service replication algorithm which replicates services for a subset of traffic flows. Under our replication strategy, one does not need to increase the bandwidth capacity of underlying links and at the same time, be able to increase the average profit for the overlay network. Our replication algorithm works well in a wide range of network settings. It can cope with dynamic flow change on links so as to reduce the QoS violation cost of the network and at the same time, able to find the appropriate path to replicate.

References

1. BRITE. http://www.cs.bu.edu/brite/.
2. MIT Resilient Overlay Networks (RON) Project. http://nms.lcs.mit.edu/ron/.
3. D. Bertsekas and R. Gallager. *Data Networks*, chapter 5.5, pages 451–455. Prentice Hall, 2nd edition, 1992.
4. Y. Chu, S. G. Gao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *ACM SIGCOMM 2001*, Apr. 2001.
5. Z. Duan, Z.-L. Zhang, and Y. T. Hou. Service Overlay Networks: SLAs, QoS and Bandwidth Provisioning. In *IEEE 10th International Conference on Network Protocols (ICNP'02)*, Paris, France, Nov. 2002.
6. Z. Duan, Z.-L. Zhang, and Y. T. Hou. Service Overlay Networks: SLAs, QoS and bandwidth provisioning. Technical report, Computer Science Department, University of Minnesota, Feb. 2002.
7. J. Jannotti. *Network Layer Support for Overlay Networks*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, Aug. 2002.
8. J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr. Overcast: Reliable Multicasting with an Overlay Network. In *the Fourth Symposium on Operating System Design and Implementation (OSDI)*, pages 197–212, Oct. 2000.
9. K. Y. Liu, J. C. Lui, and Z.-L. Zhang. On service replication strategy for service overlay networks. Technical Report CS-TR-2003-09, Dept. of Computer Science and Engineering, The Chinese University of Hong Kong, May 2003.
10. D. Mitra and Q. Wang. Stochastic traffi c engineering, with applications to network revenue management. In *IEEE Infocom 2003*, San Francisco, USA, 2003.
11. J. Touch. Dynamic Internet Overlay Deployment and Management Using the X-Bone. *Computer Networks*, pages 117–135, July 2001.
12. J. D. Touch, Y.-S. Wang, and L. Eggert. Virtual Internets. Technical Report ISI-TR-2002-558, Information Sciences Institute, July 2002.