Dynamic Network Provisioning for Time-Varying Traffic

Vicky Sharma, Koushik Kar, Richard La, and Leandros Tassiulas

Abstract: **Abstract: In this paper, we address the question of dynamic network provisioning for time-varying traffic rates, with the objective of maximizing the system throughput. We assume that the network is capable of providing bandwidth guaranteed traffic tunnels for an ingress-egress pair and present an approach that (1) updates the tunnel routes and (2) adjusts the tunnel bandwidths, in an incremental, adaptive manner, based on the variations in the incoming traffic. First, we consider a simpler scenario where tunnel routes are fixed, and present an approach for adjusting the tunnel bandwidths dynamically. We show, through simulations, that our dynamic bandwidth assignment algorithm significantly outperforms the optimal static bandwidth provisioning policy, and yields a performance close to that of the optimal dynamic bandwidth provisioning policy. We also propose an adaptive route update algorithm, which can be used in conjunction with our dynamic bandwidth assignment policy, and leads to further improvement in the overall system performance.**

Index Terms: **Bandwidth re-assignment, dynamic traffic engineering, tunnel re-routing.**

I. INTRODUCTION

The Internet provides the basic networking infrastructure for a multitude of applications. In addition to the traditional data traffic, it is being used for voice and multimedia traffic, and is expected to provide support for a wide range of real-time applications in near future. Such applications require certain Quality-of-Service (QoS), which usually translates to a certain minimum or average bandwidth guarantee from the network. Such requirements have spurred the development of models and approaches aimed at providing bandwidth guarantees to users, even with interruptions in underlying network infrastructure. Virtual Private Networks (VPN) are also fueling these availability and reliability demands, and often require the establishment of dedicated (traffic) tunnels across the network to meet the desired QoS requirements. Here a tunnel refers to a virtual path with a certain reserved bandwidth, routed through a path available between the ingress and egress nodes in the logical network.

The extension of IP routing protocols to provide new and scalable routing capabilities through Multiprotocol Label Switching (MPLS) [1], [5], [13], has emerged as one of the key frameworks to enable the required services mentioned above. MPLS could be used to establish bandwidth-guaranteed virtual tunnels (also known as Label Switched Paths (LSPs)) between an ingress (source) node and egress (destination) node, and packets can be routed from the ingress to the egress through these tunnels. Once these bandwidth-guaranteed tunnels are established, real-time voice and multimedia traffic can be carried over these tunnels while guaranteeing a certain level of QoS.

In recent years, there has been a significant interest in traffic engineering issues related to such bandwidth-guaranteed tunnels (see [7], [9], for instance). Most of the recent literature on this topic is, however, associated with set-up or tear-down questions for these tunnels, and static tunnel bandwidth and route assignments. Here, tunnel bandwidths and routes are typically determined at set-up time, and remain fixed throughout its lifetime. Significant variations in the traffic profiles is then handled by establishing new tunnels, or removing existing ones. However, resource constraints can limit the number of tunnels that can be set up between an ingress and an egress once all available resources are reserved by existing tunnels. In such a scenario, a more intelligent algorithm should try to reassign the bandwidths to the existing tunnels or even reroute some of the existing tunnels through alternate paths with more resources available in face of a significant change in traffic profiles. We refer the readers to [3], [12] for a few examples on dealing with time-varying traffic in the context of MPLS.

In this paper, we consider a scenario where traffic is carried from the ingress to the egress node through bandwidth-guaranteed tunnels. The traffic requires a certain level of QoS, which can be translated to a minimum bandwidth guarantee. If this bandwidth is not available between the corresponding ingress-egress pair at the time of the connection arrival, the connection is rejected with no queueing. From the system perspective, we want to choose the tunnel routes and their bandwidths so as to minimize the connection rejection rate, i.e., the fraction of connections rejected. This is equivalent to maximizing the system throughput in our formulation. The optimal bandwidth assignment policy must take into account the traffic rates between different ingress-egress pairs. Since the traffic rates could vary over time and are often unpredictable, it follows that a good network provisioning policy must be dynamic, and adaptively adjust the tunnel bandwidths and/or routes based on online measurement of the traffic rates. For practical feasibility, such adaptive, online tunnel dimensioning and routing policies must also be computationally simple. In addition, since traffic rates typically vary gradually and rerouting existing tunnel may disrupt on-going traffic and applications, an algorithm that gradually reconfigures the network in an incremental manner with time-varying traffic is more desirable than an algorithm that completely reconfigures the network periodically (e.g., recomputing the optimal static policy periodically based on the current traffic rates). Hence, in this paper we are interested in designing a suite of algorithms that will dynamically reconfigure the tunnels by changing their routes and assigned bandwidth.

In this work, we present simple, yet effective, network provisioning algorithms for time-varying traffic patterns. Firstly,

Manuscript received May 15, 2007; approved for publication by Bijan Jabbari, Lead Guest Editor, Nov 19, 2007.

V. Sharma and K. Kar are with the ECSE Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA (email: sharmv@rpi.edu, kark@rpi.edu).

R. La is with the ECE Department, University of Maryland, College Park, MD 20742, USA (email: hyongla@eng.umd.edu).

L. Tassiulas is with the CET Department, University of Thessaly, Volos, Greece (email: leandros@uth.gr).

we consider a scenario where tunnel bandwidths are adjustable, but tunnel re-routing is costly, and hence tunnel routes are kept fixed. In this scenario, we present an adaptive tunnel bandwidth assignment policy that re-allocates bandwidths between the fixed-route tunnels based on the dynamic traffic variations. We show, through simulations, that our incremental bandwidth re-assignment policy converges and yields the optimal tunnel bandwidth allocations for stable traffic patterns. We demonstrate that our approach significantly outperforms any static bandwidth assignment policy, and its performance is fairly close to that of the optimal dynamic bandwidth assignment policy. Next we consider a scenario where tunnel re-routing is feasible, and present an adaptive, incremental tunnel re-routing policy that can be used in conjunction with our bandwidth re-allocation policy. Simulation results indicate that the performance of our integrated tunnel bandwidth and route assignment algorithm is very close to that of the optimal dynamic network provisioning policy, while incurring significantly less computational overhead.

Several recent works have considered the question of dynamic bandwidth provisioning, and are worth mentioning in this context. The authors in [2] propose the TEAM architecture for LSP set-up, bandwidth allocation and routing. The proposed scheme measures the LSP traffic and employs Kalman filtering to arrive at an estimate of optimal number of active connections and the bandwidth to be reserved. In [6], the authors propose a fuzzy logic approach to allocate bandwidth to incoming requests. This work assumes a diffserv-enabled MPLS network with three main service classes (premium, assured, best effort) and proposes a fuzzy logic table to allocate rates to different classes as requests arrive in the network. The authors in [11] formulate the resource allocation problem for virtual paths (LSPs in MPLS). The authors construct an approximate model for a single link case, and derive the optimal allocation in a closed form, based on this model. The optimal allocation rule for the single-link case is then used as a building block to develop a solution for a general network. It is worth noting, however, that these existing works only consider the bandwidth provisioning question in isolation; we, on the other hand, consider the route assignment question as well. Our dynamic bandwidth assignment algorithm differs from the existing approaches, and integrates nicely with our route assignment procedure to provide a complete dynamic network provisioning solution.

This paper is organized as follows. In the next section, we formulate the network provisioning problem as an optimization question, and outline the solution requirements and basic approach. In Section III, we assume that the tunnel routes are fixed, and present and evaluate a dynamic bandwidth assignment algorithm for fixed-route tunnels. In Section IV, we present our adaptive tunnel routing algorithm, and evaluate the performance of our integrated tunnel routing and dimensioning algorithm. We conclude in Section V.

II. THE OPTIMAL NETWORK PROVISIONING PROBLEM

In this section, we pose our network provisioning question as a non-linear, mixed integer programming problem. We argue why solving this problem directly is practically infeasible, describe the solution requirements, and motivate and outline our basic solution approach.

A. System Model and Notation

Consider a network represented by a directed graph $G(V, E)$ with link capacities $c_{i,j}\,,\ (i,j) \in E$, where all link capacities are assumed integral. Let $K \subseteq V \times V$ denote the set of ingress-egress pairs of the network; the ingress and egress nodes of an ingress-egress pair $k \in K$ are denoted by s_k and d_k, respectively. We denote the set of tunnels used by ingress-egress pair $k \in K$ by L_k, and $L = \cup_{k \in K} L_k$ is the set of all tunnels. In practice, the number of tunnels between an ingress-egress pair will be limited, for example, by the computational and/or storage complexity of maintaining these tunnels. For ingress-egress pair k, let $\bar{L}_k$ represent this pre-specified upper limit. In the rest of this paper, we assume $|L_k| = \bar{L}_k$, i.e., we always use the maximum number of tunnels. This does not result in a loss of generality, since the assigned bandwidth to some of these tunnels may be set to zero, allowing the use of a fewer number of tunnels when desired. The bandwidth assigned to a tunnel $l \in L$ of ingress-egress pair k is denoted by b_l (assumed integral), and the total bandwidth allocated to ingress-egress pair k is $B_k = \sum_{l \in L_k} b_l$. Let R_l denote the set of all available loop-free routes (paths) between the ingress and egress nodes of tunnel l. Let $x_{l,r}$ be a 0/1 variable denoting whether tunnel l is routed over route (path) r or not. Assume that the set of links on route r is denoted by Ω_r.

We assume, for simplicity of exposition and analysis, that each incoming connection request requires one unit of bandwidth. However, our approach can be easily extended to the case where the bandwidth requirements of different connections may be different. The connection arrival process for ingress-egress pair k is assumed to be a Poisson process with rate λ_k. Each connection lasts for a random interval of time, and the connection durations are exponentially distributed. For ingress-egress pair k, the mean connection duration is $(1/\mu_k)$. The connection arrival processes of all ingress-egress pairs are assumed to be mutually independent. Let $\rho_k = \frac{\lambda_k}{\mu_k}$ and $\phi_k(\rho_k, B_k)$ denote the (average) fraction of ingress-egress pair k connections that are rejected due to insufficient bandwidth. Using the above assumptions and M/M/n/n queueing results [10], we get

$$\phi_k(\rho_k, B_k) \quad = \quad \frac{1}{B_k!} \frac{\rho_k^{B_k}}{\sum_{k'=0}^{B_k} \frac{\rho_k^{k'}}{k'!}}\,. \tag{1}$$

The overall rejection rate, aggregated over all ingress-egress pairs, is given by

$$\Phi \quad = \quad \sum_{k \in K} \alpha_k \phi_k(\rho_k, B_k)\,, \tag{2}$$

where $\alpha_k,\ k \in K$, are given by

$$\alpha_k \quad = \quad \frac{\lambda_k}{\sum_{k' \in K} \lambda_{k'}}\,. \tag{3}$$

B. Formulation

Maximizing the system throughput in this case is equivalent to minimizing the overall connection rejection rate. The objec-

tive of the bandwidth reallocation algorithm is given by

$$\text{minimize } \Phi,$$

subject to

$$\sum_{l \in L} \sum_{r \in R_l : (i,j) \in \Omega_r} b_l \, x_{l,r} \leq c_{i,j}, \ \forall (i,j) \in E, \quad (4)$$

$$b_l \geq 0, \ \forall l \in L, \quad (5)$$

$$\sum_{r \in R_l} x_{l,p} = 1, \ \forall l \in L, \quad (6)$$

$$x_{l,r} \in \{0,1\}, \ \forall r \in R_l, \ \forall l \in L. \quad (7)$$

Note that the the expression $\sum_{l \in L} \sum_{r \in R_l : (i,j) \in \Omega_r} b_l \, x_{l,r}$ in (4) denotes the total bandwidth allocated on link (i,j). Therefore, (4) represents link capacity constraints. Constraints in (5) are the non-negativity constraints of the bandwidths, and (6)-(7) describe the constraint that a tunnel must be routed over a single route (path).

In the above formulation, the variables b_l, $x_{l,r}$ are the decision variables that need to be chosen optimally. The traffic parameters λ_k and ρ_k are assumed to be fixed (and known).

C. Solution Requirements and Basic Approach

Ideally, we would like to find the optimal routes and bandwidths of the tunnels by solving the optimization problem posed above. Such an approach is feasible and efficient if the traffic rates do not vary significantly with time, in which case the network provisioning problem can be solved off-line once the traffic rates are known (or estimated). However, when traffic varies with time, a static route/bandwidth assignment policy will be inefficient in general (as we demonstrate later in this paper). One solution to the problem may be to re-solve the optimization problem whenever the traffic matrix changes significantly. Such an approach is practically infeasible for two reasons. Firstly, it may require solving the complex mixed integer optimization problem frequently, which incurs a large computational overhead. Secondly, even if the optimal tunnel routes (bandwidths) are computed, the new routes (bandwidths) can be very different from the current ones, and therefore, updating the current set of routes (bandwidths) to the new routes (bandwidths) could involve significant delay and communication costs, and at the same time disrupt much of on-going traffic that needs to be rerouted.

For the reasons stated above, when the traffic is time-varying an *incremental* solution is preferred. In such an approach, the bandwidth assignment and route selection algorithms are run periodically (time-driven) or whenever the traffic changes significantly (event-driven), but the bandwidths and/or the routes of *only a few* tunnels are updated at a time. By limiting the number of bandwidth and routing updates that can take place each time the algorithm is executed, the overhead associated with the dynamic network reconfiguration can be kept within a manageable level and we can minimize the disruption to existing traffic. In this paper we take this approach and propose integrated bandwidth and route update algorithms that react to time varying traffic. We demonstrate through extensive simulations that even a small number of appropriately chosen bandwidth and route updates could improve the system performance significantly, and in many cases achieve near-optimal network performance.

III. DYNAMIC BANDWIDTH RE-ASSIGNMENT FOR TIME-VARYING TRAFFIC

In this section, we consider a scenario where the routes of the tunnels are assumed fixed, and investigate the problem of developing an efficient bandwidth re-assignment algorithm that re-allocates bandwidths of these fixed-route tunnels in an incremental, adaptive manner. The reason for considering this simpler, fixed-route scenario is two-fold. Firstly, tunnel re-routing can involve a much higher overhead compared to simple bandwidth re-assignment. Therefore, tunnel re-routing can often occur at a longer time-scale (possibly based on the long-term time average traffic rates) in comparison to adaptive bandwidth reprovisioning of existing tunnels based on short-term traffic variations. Therefore, in certain practical scenarios, dynamic tunnel bandwidth assignment questions can be considered orthogonal to tunnel route selection issues, and can be studied separately. Secondly, even in scenarios where joint bandwidth and route reassignments are feasible, study of the bandwidth re-assignment problem in isolation provides valuable insights to the more complex problem of joint bandwidth and route re-assignment.

A. The Optimal Bandwidth Assignment Problem

In the network model described in Section II, we assume that the route of each tunnel is fixed. Let the set of links on the route (path) of a tunnel $l \in L$ be denoted by A_l. Then the objective of the optimization problem is given by (1)-(3), and the constraints are

$$\sum_{l \in L : (i,j) \in A_l} b_l \leq c_{i,j}, \ \forall (i,j) \in E, \quad (8)$$

$$b_l \geq 0, \ \forall l \in L. \quad (9)$$

Note that our objective function is a non-linear function of the tunnel bandwidths, and the above problem is a complex non-linear optimization problem. As mentioned earlier, we are interested in designing an incremental bandwidth assignment algorithm that tries to achieve the same objective of minimizing the connection rejection rate *without* having to explicitly solve the optimization problem posed above.

B. Solution Approach

At an intuitive level, our solution approach can be described as follows. Our algorithm is run periodically every T time units, where T is an adjustable system parameter. Each time the algorithm is executed, it repeats the following procedure up to M times, where M is also an adjustable system parameters. In each of M iterations, the algorithm first identifies a candidate tunnel whose bandwidth may be incremented by one unit. This candidate tunnel is chosen in a greedy manner, based on a 'profit function', described in details later in this section. Intuitively, the 'profit function' for a tunnel is a measure of the improvement in the system-wide connection rejection rate when the bandwidth of the tunnel is incremented by unit amount (while possibly decrementing the bandwidth of other tunnels in order

to free up the needed bandwidth for the candidate tunnel). Once a tunnel is selected for a bandwidth increment, the algorithm determines whether or not it is necessary to reduce the bandwidths of some other tunnels that share links with the chosen tunnel by one unit. If so, the algorithm identifies a set of tunnels which a unit bandwidth is taken away from and is given to the candidate tunnel, and computes the change in the overall system rejection rate, which is defined to the profit function of the candidate tunnel. If the overall rejection rate decreases, then the bandwidth update takes place. Otherwise, the algorithm does nothing and proceeds to the next iteration and identifies another candidate tunnel if possible. Note that the profit function depends on the traffic parameters, and reflects the relative importance of adding bandwidth to a tunnel under the current traffic parameters. Therefore, as the traffic rates vary over time, our algorithm tries to dynamically allocate more bandwidth to tunnels assigned to ingress-egress pairs that are more heavily loaded, while taking bandwidth away from less loaded ingress-egress pairs, so as to improve the system-wide connection rejection rate.

Next we describe how the profit function is defined in details. For any $k \in K$, define Δ_k^+ and Δ_k^- as follows:

$$\Delta_k^+ = \phi_k(\rho_k, B_k) - \phi_k(\rho_k, B_k + 1) , \quad (10)$$

$$\Delta_k^- = \phi_k(\rho_k, B_k - 1) - \phi_k(\rho_k, B_k) . \quad (11)$$

Note that Δ_k^+ represents the decrease in the connection rejection rate when the bandwidth allocated to ingress-egress pair k is incremented by one unit, from B_k to $B_k + 1$. Similarly, Δ_k^- represents the increase in the connection rejection rate when the bandwidth allocated to ingress-egress pair k is decremented by one unit, from B_k to $B_k - 1$.

Let $\kappa(l)$ denote the ingress-egress pair for tunnel l, i.e., $\kappa(l) = \{k \in K : l \in L_k\}$. Consider a tunnel $l \in L$, and let us assume that the bandwidth of the tunnel is incremented from b_l to $b_l + 1$. This results in an increase in the overall bandwidth allocated to ingress-egress pair $\kappa(l)$ from $B_{\kappa(l)}$ to $B_{\kappa(l)} + 1$. Therefore, the addition of one unit of bandwidth to tunnel l results in a decrease in the connection rejection rate of ingress-egress pair $\kappa(l)$ by $\Delta_{\kappa(l)}^+$.

Note, however, that in order to increment tunnel l's bandwidth allocation, we may need to decrement the bandwidth allocated to some other tunnels. To see this, let us define $\hat{c}_{i,j}$ to be the available capacity of a link $(i, j) \in E$, given by

$$\hat{c}_{i,j} = c_{i,j} - \sum_{l \in L:(i,j) \in A_l} b_l . \quad (12)$$

Clearly, $\hat{c}_{i,j}$ represents the capacity of link (i, j) minus the aggregate bandwidth that has been allocated to different tunnels sharing link (i, j). Let $\hat{A}_l \subseteq A_l$ denote the set of links on tunnels l's route that have zero available capacities, i.e., $\hat{A}_l = \{(i, j) \in A_l : \hat{c}_{i,j} = 0\}$. If $\hat{A}_l = \phi$, then it is possible to increment tunnel l's bandwidth without reducing the bandwidth of other tunnels. Otherwise, the unit bandwidth added to tunnel l must be "taken away" from some other tunnel(s) using the links in $\hat{A}_l$. Let N_l be the set of tunnels whose bandwidth is decremented by one in order to increase bandwidth allocation of tunnel $l \in L_k$ by unity. Clearly, the tunnels in N_l must "cover" all the links in $\hat{A}_l$, i.e., each link in $\hat{A}_l$ must be a part of the route of some tunnel in N_l. The bandwidth allocation of each tunnel in N_l is decremented by unity, and therefore, the overall increase in the connection rejection rate due to these bandwidth decrements is given by

$$\sum_{l' \in N_l} \alpha_{\kappa(l')} \Delta_{\kappa(l')}^- , \quad (13)$$

where Δ_k^- are defined by (11).

Note that there can be multiple sets of tunnels that cover $\hat{A}_l$. In that case, we choose the set N_l that minimizes the increment in the overall rejection rate given in (13).

The profit function of tunnel $l \in L_k$, denoted by P_l, is given by

$$P_l = \Delta_{\kappa(l)}^+ - \min_{N_l} \sum_{l' \in N_l} \alpha_{\kappa(l')} \Delta_{\kappa(l')}^- , \quad (14)$$

where Δ_k^+ and Δ_k^- are defined in (10) and (11), respectively. The term $\min_{N_l} \sum_{l' \in N_l} \alpha_{\kappa(l)} \Delta_{\kappa(l)}^-$ represents a *minimum weighted set-cover problem* [4], [8], where we are interested in finding a set of tunnels that cover all the links in $\hat{A}_l$ and yields the smallest increase in the aggregate rejection rate when a unit bandwidth is taken away from them. This set-cover problem is NP-hard [4]. In our simulations, we will use a greedy approximation to this set cover problem, which is described below. For the unweighted set cover problem, this greedy approximation algorithm is known to guarantee a logarithmic approximation factor [8].

Our algorithm can now be described as follows:

Bandwidth_Re-assignment_Algorithm (BRA)

For $t = T, 2T, 3T, ...$, do the following:

For $i = 1, 2, ..., M$, execute the following steps:

1. Determine profit for every tunnel $l \in L$ using (14), using the **Greedy_Link_Cover** algorithm to compute $\min_{N_l} \sum_{l' \in N_l} \alpha_{\kappa(l)} \Delta_{\kappa(l)}^-$ approximately. The average traffic arrival rates in the last T time units are used in these calculations.
2. Pick the tunnel with the maximum profit. Let this tunnel be l. If the profit $P_l > 0$, go to the next step; otherwise, exit the loop.
3. Increment the bandwidth of the tunnel l by one unit. Decrement the bandwidth of each tunnel $l' \in N_l$ by one unit, where the set N_l is computed using the **Greedy_Link_Cover** algorithm.

Greedy_Link_Cover (GLC)

1. *Initialization:* Find the set of all tunnels which cover least one link in $\hat{A}_l$. Let this set of tunnels be $\tilde{N}_l$. Set $\tilde{A}_l = \hat{A}_l$, and $N_l = \phi$.
2. Do the following until $\tilde{A}_l = \phi$:
   (a) For each tunnel $l' \in \tilde{N}_l$:
   i. Compute $n_{l'}$, the number of links in $\tilde{A}_l$ that are covered by l'.
   ii. Compute the weight function $w_{l'} = n_{l'}/(\alpha_{\kappa(l')} \Delta_{\kappa(l')}^-)$.
   (b) Pick the tunnel with the maximum weight function. Let this tunnel be l''. Then,
   i. Include l'' in N_l.

ii. Remove l'' from $\tilde{N}_l$.
iii. Remove all the links covered by l'' from $\tilde{A}_l$.

C. Example

As an example, consider the network topology shown in Figure 1, which consists of two ingress-egress pairs, each with one tunnel. Let us denote the two ingress-egress pairs, and the corresponding tunnels, by 0 and 1.

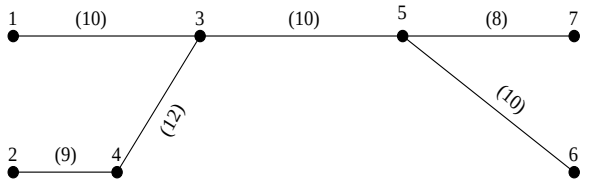

Fig. 1. Example Network 1. (The numbers shown in ( ) across the links represent the link capacities.)

The ingress and egress nodes as well as the route of the tunnel for each ingress-egress pair are as follows: $s_0 = 1,\ d_0 = 7,\ A_0 = \{(1,3),(3,5),(5,7)\};\ s_1 = 2,\ d_1 = 6,\ A_1 = \{(2,4),(4,3),(3,5),(5,6)\}$.

Let $b_0 = B_0 = b_1 = B_1 = 5$. Let $\mu_0 = \mu_1 = 1$ and $\lambda_0 = 3, \lambda_1 = 1$. Thus $\rho_0 = 3$, $\rho_1 = 1$, and

$$\begin{aligned}
\alpha_0 &= \frac{\lambda_0}{\lambda_0+\lambda_1} = \frac{3}{4},\\
\alpha_1 &= \frac{\lambda_1}{\lambda_0+\lambda_1} = \frac{1}{4},\\
\phi_0(\rho_0,5) &= \frac{1}{5!}\frac{(\rho_0)^5}{\sum_{k=0}^{5}\frac{(\rho_0)^k}{k!}} = 0.11,\\
\phi_1(\rho_1,5) &= \frac{1}{5!}\frac{(\rho_1)^5}{\sum_{k=0}^{5}\frac{(\rho_1)^k}{k!}} = 0.03,
\end{aligned}$$

$$\begin{aligned}
\Phi &= \alpha_0\phi_0(\rho_0,B_0)+\alpha_1\phi_1(\rho_1,B_1) = 0.0833,\\
\Delta^+(0) &= \phi_0(\rho_0,B_0)-\phi_0(\rho_0,B_0+1) = 0.0578,\\
\Delta^+(1) &= \phi_1(\rho_1,B_1)-\phi_1(\rho_1,B_1+1) = 0.0025,\\
\Delta^-(0) &= \phi_0(\rho_0,B_0-1)-\phi_0(\rho_0,B_0),\\
\Delta^-(1) &= \phi_1(\rho_1,B_1-1)-\phi_1(\rho_1,B_1),\\
P_0 &= \frac{3}{4}\times\Delta^+(0)-\frac{1}{4}\times\Delta^-(1) = 0.0403,\\
P_1 &= \frac{1}{4}\times\Delta^+(1)-\frac{3}{4}\times\Delta^-(0) = -0.071.
\end{aligned}$$

Since the profit of tunnel 0 (of ingress-egress pair 0) is positive and greater than profit of tunnel 1 (of ingress-egress pair 1), we select the former for bandwidth increment. Thus we increment b_0 by 1, and decrement b_1 by the same amount. Now we have new overall rejection rate, $\Phi^{'} = \frac{3}{4}\times 0.0522 + \frac{1}{4}.0154 = 0.04325$; hence $\Phi^{'} < \Phi$. Hence, incrementing the bandwidth of route 0 using our dynamic bandwidth re-assignment algorithm results in a reduction in overall connection rejection rate. If $\mu_0 = \mu_1 = 1$ and $\lambda_0 = 3, \lambda_1 = 3$, in this case it is easily observed that $P_0 = P_1 = 0$. Hence, no tunnel will be chosen for bandwidth increment in this case, and the tunnel bandwidth assignments remain unchanged, as expected.

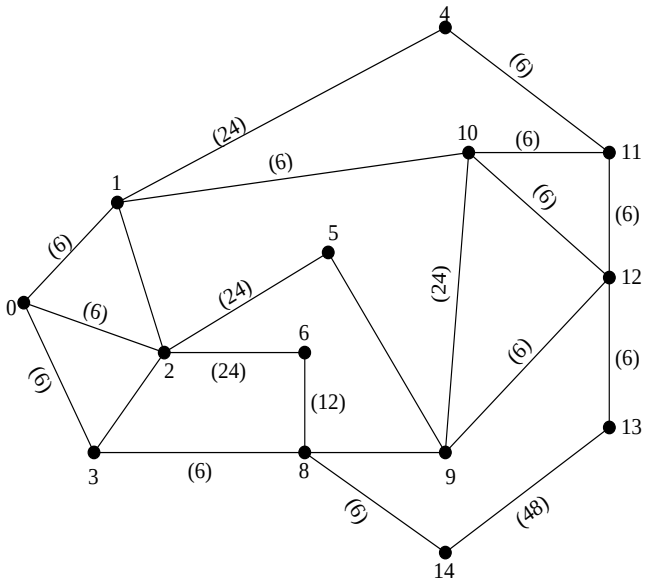

Fig. 2. Example Network 2. (The numbers shown in ( ) across the links represent the link capacities.)

D. Simulation Results

D.1 Example Network

We present the simulation results of the proposed algorithm based on the example network shown in Figure 2. There are four ingress-egress pairs in the network, and the ingress and egress nodes of these pairs are as follows: $s_0 = 0$, $d_0 = 12$, $s_1 = 4$, $d_1 = 8$, $s_2 = 3$, $d_2 = 1$, $s_3 = 4$, $d_3 = 14$. Ingress-egress pairs 0 and 2 have three tunnels each, while ingress-egress pairs 1 and 3 have two tunnels each.

D.2 Traffic Profiles Used

Simulations were carried out for three different traffic profiles. The connection arrival processes are Poisson processes, and the connection durations are exponentially distributed. The mean connection duration is fixed at one unit time. Traffic rate of an ingress-egress pair is varied by changing the arrival rate of the corresponding arrival process (assumed to be Poisson).

The traffic profiles used are as follows:

1. *Profile A:* This is a step profile as shown in Figure 3. The traffic patterns of all ingress-egress pairs are the same. We use this traffic profile to show that if the traffic rates remain stable (i.e., do not vary significantly) over a period of time, then our algorithm tends to converge and yield the optimal rates for the traffic profile.

2. *Profile B:* This traffic profile shown in Figure 4, is more realistic, and is generated by adding a noise to smooth sinusoidal variations. The noise is modeled as a *brownian bridge process* [14]. Note that the traffic profile varies from one ingress-egress pair to another. Note that in the figure, the traffic pattern is shown only for 0-23 hours; this traffic pattern is repeated every 24 hours. We use this profile to compare the performance of our algorithm with that of the optimal static and dynamic bandwidth allocations.

For each of the profiles, a total of 10,000 arrival and departure events were simulated, and results are averaged over 25 sample

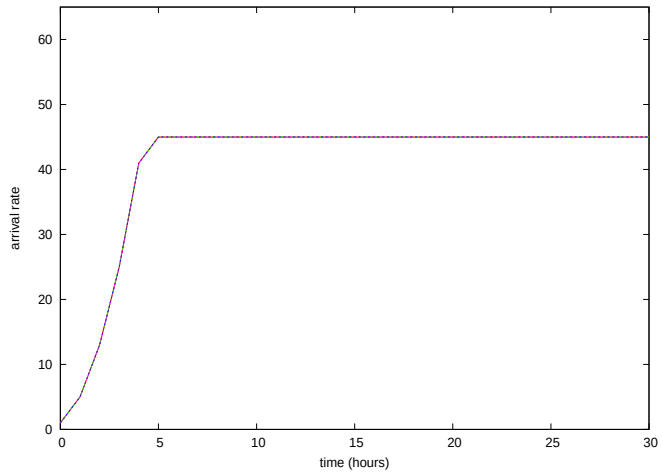

Fig. 3. Traffic profile A. The vertical axis plots the rate of poisson arrivals for an ingress-egress pair.

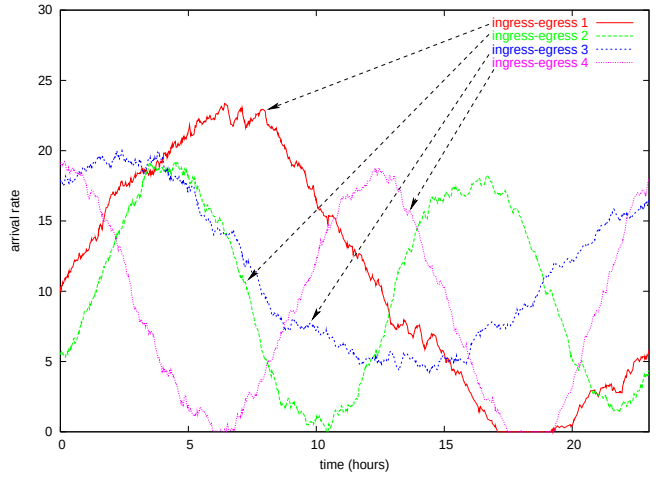

Fig. 4. Traffic Profile B. The vertical axis plots the rate of poisson arrivals for an ingress-egress pair.

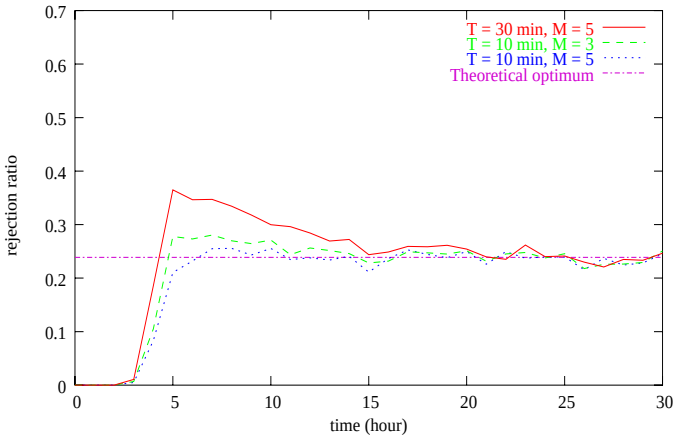

Fig. 5. Convergence of achieved rejection rate to the theoretical optimum.

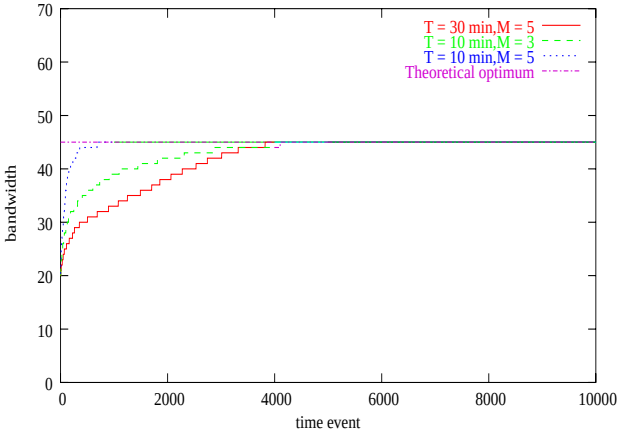

Fig. 6. Convergence of the allocated bandwidth to the optimum, for ingress-egress pair 1.

paths.

D.3 Results and Discussions

Results for profile A:

Figures 5-6 show the simulations results for traffic profile A. Figure 5 shows that the average connection rejection rate for our algorithm gradually converges to the optimal rejection rate (computed using the Erlang-B formula). Figure 6 demonstrates the convergence of the allocated bandwidths to their optimal values. Here, the optimal bandwidth allocations are computed by numerically solving the nonlinear optimization problem posed in Section II. From the plots, we see that as expected, a smaller value of the update interval T results in faster convergence. Similarly, a larger value of M (the maximum number of iterations the bandwidth update procedure is executed) also results in faster convergence.

Results for profile B: In this case, the fraction of connections rejected (rejection rate) is plotted as a function of the average traffic load, for different values of the simulation parameters T and M. Figures 7 and 8 show the corresponding simulation results. In the figure, the 'static optimal policy' case corresponds to the policy where a static bandwidth allocation is used. In this policy, the optimal bandwidth allocations are computed offline, based on the time-averaged traffic rates, and these bandwidth assignments are used throughout the simulation run. We see that our algorithm performs significantly better than this static policy. This is because our policy takes into account the dynamic nature of the traffic profile. The 'dynamic optimal policy' case refers to a policy that recomputes the optimal bandwidth allocations every hour (based on the current traffic arrival rates), by solving the optimization problem posed in Section III-A. In our case, we used the MATLAB optimization toolbox to compute the optimum bandwidth assignments, for both the static and dynamic policies. Our plots demonstrate that our incremental algorithm achieves performance very close to that of the optimal dynamic bandwidth assignment policy. Moreover the computational complexity of our algorithm is considerably less than that of the latter.

In Figure 7 we see that, for a fixed T, the performance improves as M increases as one would intuitively expect. Similarly, Figure 8 shows that for a fixed M, the performance improves with decreasing value of T, as expected. At intermediate values of the traffic load, our algorithm is observed to perform about 10-15% better than the static optimal policy, and is within 5% of that of the optimal dynamic policy.

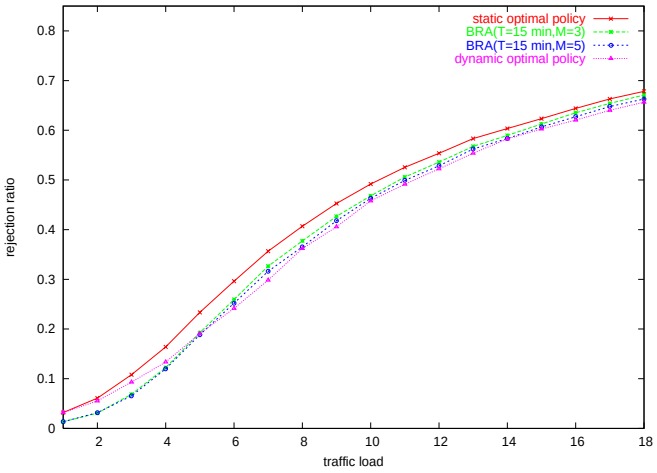

Fig. 7. Rejection rate vs. average traffic load for our algorithm (for T = 15 minutes and M = 3,5), and the static and dynamic optimal policies.

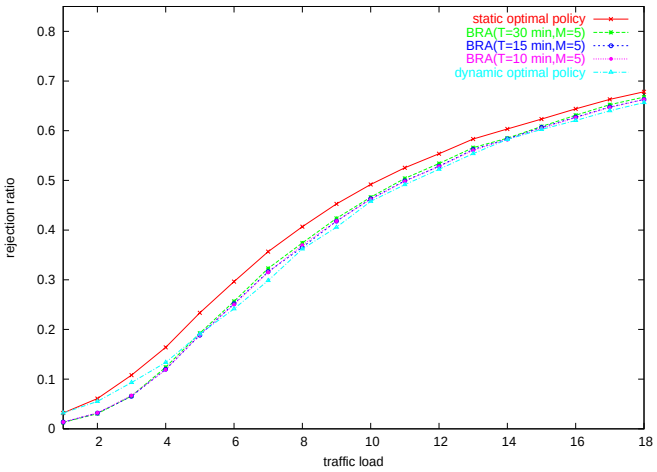

Fig. 8. Rejection rate vs. average traffic load for our algorithm (for T = 10, 15, 30 minutes and M = 5), and the static and dynamic optimal policies.

IV. DYNAMIC TUNNEL RE-ROUTING FOR TIME-VARYING TRAFFIC

In this section, we consider a scenario where tunnels are re-routable, and present an adaptive, incremental route re-assignment algorithm for time varying traffic. We then demonstrate that the inclusion of our tunnel re-routing algorithm into our dynamic network provisioning algorithm leads to a significant increase in the system throughput, over what can be achieved only by bandwidth re-assignment of tunnels with fixed routes. We also show that the performance of our integrated route and bandwidth assignment algorithms is fairly close to that of the optimal dynamic network provisioning policy.

A. Motivation

Although our bandwidth assignment algorithm presented in the previous section performs close to optimal dynamic bandwidth assignment policy, the connection rejection rate achieved by our algorithm depends significantly on the choice of the (fixed) routes. We illustrate this by means of an example, which also demonstrates the importance of dynamic tunnel re-routing in achieving improved system throughput.

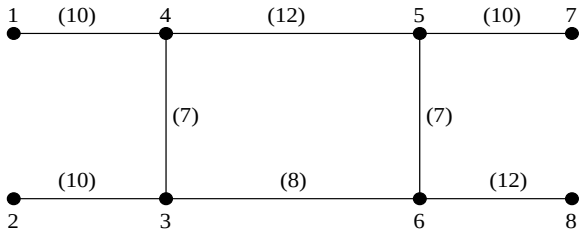

Fig. 9. Example Network 3. (The numbers shown in ( ) across the links represent the link capacities.)

Consider the network shown in Figure 9. Assume that there are two ingress-egress pairs, with one tunnel each. Let us denote the two ingress-egress pairs, and the corresponding tunnels, by 0 and 1. The ingress and egress nodes, and the tunnel routes and bandwidths are as follows: $s_0 = 1$, $d_1 = 7$, $A_0 = \{(1,4),(4,5),(5,7)\}$, $b_0 = B_0 = 6$; $s_1 = 2$, $d_2 = 8$, $A_1 = \{(2,3),(3,4),(4,5),(5,6),(6,8)\}$, $b_1 = B_1 = 5$.

Assume $\mu_0 = \mu_1 = 1$, and $\lambda_0 = \lambda_1 = 3$. Therefore $\rho_0 = \rho_1 = 3$. In this scenario, it can be easily observed that the profit functions for the tunnels are zero. Therefore, this is the optimal bandwidth allocation for the two tunnels, and the optimal rejection rate can be computed to be 0.052. Therefore, using bandwidth adjustments only (while keeping routes fixed), we would not be able to reduce the rejection rate below 0.052. However, note that if tunnel 2 is re-routed to route $A_2' = \{(2,3),(3,6),(6,8)\}$, then bandwidths of 10 and 8 could be assigned to tunnels 0 and 1, respectively. This reduces the overall connection rejection rate to 0.0044, which is smaller than our previous rejection rate by an order of magnitude. Therefore, by combining tunnel re-routing and bandwidth re-assignment intelligently, we could achieve a much better throughput than what is achievable by fixed-route bandwidth re-assignment. This motivates us to look for simple but efficient tunnel re-routing strategies.

B. Solution Approach

At an intuitive level, our solution approach can be viewed as follows. Our algorithm is run periodically after every T' time units, where T' is an adjustable system parameter. In each run of the algorithm, a few tunnels are re-routed. We assume that tunnel re-routing preserves the bandwidth of the re-routed tunnel. Thus, if the tunnel to be re-routed is l, then our tunnel re-routing algorithm keeps the tunnel bandwidth unchanged at b_l after the tunnel is rerouted. Clearly, this re-routing requires reserving a bandwidth of b_l on the new route and will 'free up' the same amount of bandwidth in the current route of the tunnel. If the new route does not currently have enough available capacity to accommodate tunnel l, then we decrement the assigned bandwidths of some appropriately chosen tunnels that share one or more links with this route, by the necessary amount. This freed up bandwidth is then used to accommodate the re-routed tunnel. Also, the bandwidth b_l freed up on the original route of the tun-

nel is used to increment the bandwidths of other tunnels sharing the links in the route .

From the above discussion, note that although the bandwidth of the re-routed tunnel l is kept the same during the re-routing process, the bandwidth of some tunnels other than l might change due to the re-routing. Bandwidths of some of the tunnels sharing links on the original route of tunnel l may increase, resulting in a decrease in the rejection rate for the corresponding ingress-egress pairs, thereby improving the system throughput. On the other hand, bandwidths of some of the tunnels sharing links with the new route of tunnel l may decrease, leading to an increase in the rejection rate for the corresponding ingress-egress pairs, thereby reducing the system throughput. Clearly, in order to enhance the system performance we need to choose the tunnel l such that the throughput improvement due to the former factor outweighs the throughput reduction due to the latter factor, yielding a positive gain in system throughput.

Viewed intuitively, to maximize throughput improvement due to re-routing, we want to choose a tunnel (for re-routing) such that it is currently using the "most congested" route, and move it to the "least congested" route. Our tunnel re-routing algorithm is based on this intuition. In the following, we outline how the notions of the most and least congested routes can be formally captured in terms of the given problem parameters.

Towards this end, lets us first quantify the gain (loss) in the overall system throughput when one unit of bandwidth is made available on (taken away from) any route r. Note that if one unit of bandwidth is made available on each of the links in route r, this extra bandwidth can be used by some tunnels using the links in r. With the available link capacities defined as in (12), let $\hat{\mathcal{A}}_r$ denote the set of links on route r that have zero available capacities (before the extra unit bandwidth is made available). A tunnel is said to be *bottlenecked by route* r if the all links with zero available capacities in the tunnel's route belong to $\hat{\mathcal{A}}_r$. Clearly, when an extra unit of bandwidth is made available on route r, this bandwidth can be used to increase the bandwidth assigned to a tunnel bottlenecked by route r. Let $\hat{L}_r$ denote the set of all tunnels bottlenecked by route r. Let $S_r \subseteq \hat{L}_r$ denote the set of tunnels whose bandwidth is incremented by one unit, when the extra unit of bandwidth becomes available on route r. Note that the overall reduction in the connection rejection rate due to the bandwidth increments of the tunnels in S_r, is given by

$$\sum_{l' \in S_r} \alpha_{\kappa(l')} \, \Delta^{+}_{\kappa(l')} \, ,$$

where the terms Δ^{+}_{k} are defined by (10). Note that there can be multiple choices for S_r, and we choose the set S_r that maximizes the above expression. Thus, Γ_r, the overall *gain* in the system throughput due to the availability of a unit of bandwidth on route r, is given by

$$\Gamma_r \quad = \quad \max_{S_r} \sum_{l' \in S_r} \alpha_{\kappa(l')} \, \Delta^{+}_{\kappa(l')} \, . \tag{15}$$

If the amount of extra bandwidth available on route r is δ units, then using a first order approximation, the overall gain in the system throughput is approximated by [1]

$$\delta \, \Gamma_r \quad = \quad \delta \max_{S_r} \sum_{l' \in S_r} \alpha_{\kappa(l')} \, \Delta^{+}_{\kappa(l')} \, .$$

Now consider the case where one unit of bandwidth needs to be freed up on route r, i.e., the bandwidth on each of the links on route r needs to be decremented by one unit. If a link on route r is not in $\hat{\mathcal{A}}_r$, this bandwidth can be taken away from link's available capacity (which is non-zero). However, for links in $\hat{\mathcal{A}}_r$ (which have zero available capacities), this bandwidth reduction must be carried out by taking away an equivalent amount of bandwidth from tunnels that use these links. Let T_r be the set of tunnels whose bandwidth is decremented in order to free up one unit of bandwidth on route r. The overall increase in the connection rejection rate due to the bandwidth decrements of the tunnels in T_r is given by

$$\sum_{l' \in T_r} \alpha_{\kappa(l')} \, \Delta^{-}_{\kappa(l')} \, ,$$

where the terms Δ^{-}_{k} are defined by (11). Note that there can be multiple choices for T_r, and we choose the set T_r that minimizes the above expression. Thus, Λ_r, the overall *loss* in the system throughput due to the reduction of one unit of bandwidth on route r, is expressed by

$$\Lambda_r \quad = \quad \min_{T_r} \sum_{l' \in T_r} \alpha_{\kappa(l')} \, \Delta^{-}_{\kappa(l')} \, . \tag{16}$$

If the amount of bandwidth reduction on route r is δ units, then using a first order approximation, the overall loss in the system throughput is approximated by

$$\delta \, \Lambda_r \quad = \quad \delta \min_{T_r} \sum_{l' \in T_r} \alpha_{\kappa(l')} \, \Delta^{-}_{\kappa(l')} \, .$$

Next we comment on how the sets S_r and T_r can be chosen optimally. The maximization $\max_{S_r} \sum_{l' \in S_r} \alpha_{\kappa(l')} \, \Delta^{+}_{\kappa(l')}$ corresponds to the *maximum weighted set-packing problem* [8]. Here the links in $\hat{\mathcal{A}}_r$ correspond to the elements, and each tunnel in $\hat{L}_r$ corresponds of a set consisting of one or more of these elements. With this mapping, S_r corresponds to a collection of disjoint sets. In the weighted set-packing problem, we are required to choose this collection of disjoint sets so that the total weight of the sets chosen is maximized. If the weight of a tunnel $l' \in \hat{L}_r$ is set to $\alpha_{\kappa(l')} \, \Delta^{+}_{\kappa(l')}$, then the problem $\max_{S_r} \sum_{l' \in S_r} \alpha_{\kappa(l')} \, \Delta^{+}_{\kappa(l')}$ corresponds to the maximum weighted set-packing problem. The maximum weighted set-packing problem is NP-hard [8]. In our re-routing scheme, we use a greedy approximation to this problem, which is similar in nature to the **Greedy_Link_Cover** algorithm, and is described below as the **Greedy_Tunnel_Packing** algorithm.

The minimization $\min_{T_l} \sum_{l' \in T_l} \alpha_{\kappa(l')} \, \Delta^{-}_{\kappa(l')}$ represents a minimum weighted set-cover problem, as already argued in

[1] Note that in discussion here, we are making an implicit, simplifying assumption that the network links not included in $\hat{\mathcal{A}}_r$ have enough available capacities for accommodating a simultaneous unit bandwidth increment for all tunnels in S_r. Clearly, if the bandwidth changes are done in units of δ and δ is sufficiently small, this assumption holds.

Section III. This problem being NP-hard, we can use **Greedy_Link_Cover**, a greedy approximation algorithm for the problem, as described in Section III in this paper.

In our re-routing algorithm, we first compute the gain functions (given by (15)) on the routes of all existing tunnels in the network. Let l represent the tunnel that is chosen for re-routing, and $\hat{r}(l)$ denote its current route. Then l is chosen so that it achieves the maximum gain $\Gamma_{\hat{r}(l)}$, over all tunnels that currently exist in the network. The new route, $\tilde{r}(l)$, is chosen so that it minimizes the loss function (given by (16)) over all possible routes of ingress-egress pair $\kappa(l)$ (we discuss later how this minimization can be done in a computationally efficient manner). Tunnel l is now re-routed from $\hat{r}(l)$ to $\tilde{r}(l)$. Thus, bandwidth b_l is now taken away from tunnel l, and this bandwidth is assigned to other tunnels using the links in $\hat{r}(l)$ based on the algorithm **Greedy_Tunnel_Packing**. Also, tunnel l is allocated an equal amount of bandwidth on route $\tilde{r}(l)$, and this bandwidth is obtained from other tunnels using the links in $\tilde{r}(l)$ based on the algorithm **Greedy_Link_Cover** (or from the solution of our route-finding algorithm, as described later). In practice, the process of moving the bandwidth allocation of tunnel l from route $\hat{r}(l)$ to $\tilde{r}(l)$ can be done in a gradual manner, to avoid re-routing of existing traffic as much as possible. Also, note that in each run of our re-routing procedure (which is invoked once in every T' time units), up to M' tunnels are re-routed, where M' is an adjustable parameter.

Our algorithm is described below:

Tunnel_Re-routing_Algorithm (TRA)

For $t = T', 2T', 3T', ...$, do the following:

For $i = 1, 2, ..., M'$, execute the following steps:

1. Compute the gain function for every tunnel in the network using (15) and approximating the maximum weighted set-packing problem by the **Greedy_Tunnel_Packing** algorithm. The average traffic arrival rates in the last T' time units are used in these calculations.
2. Pick the tunnel with the maximum gain function. Let this tunnel be $l \in L$.
3. Find the route between ingress-egress pair $\kappa(l)$ with the minimum loss function (16) using the **Min-Loss_Route_Algorithm** (see below). This is the new route, $\tilde{r}(l)$.
4. Reduce the bandwidth of tunnel l on the old route $\hat{r}(l)$ to zero, and assign this bandwidth to links in $S_{\hat{r}(l)}$ computed by the **Greedy_Tunnel_Packing** algorithm in step 1. Obtain the optimum $T_{\tilde{r}(l)}$ from the solution of the **Min-Loss_Route_Algorithm** algorithm, found in step 3 (alternatively, compute $T_{\tilde{r}(l)}$ approximately using the **Greedy_Link_Cover** algorithm on route $\tilde{r}(l)$). Assign a bandwidth b_l to tunnel l on the new route $\tilde{r}(l)$ by taking away equivalent amounts of bandwidth from the tunnels in $T_{\tilde{r}(l)}$.

Greedy_Tunnel_Packing (GTP)

1. *Initialization:* Find $\hat{L}_r$, the set of all tunnels that are bottlenecked by r. Set $\tilde{\mathcal{A}}_r = \hat{\mathcal{A}}_r$, $\tilde{L}_r = \hat{L}_r$ and $S_{\hat{r}} = \phi$.
2. Do the following until $\tilde{\mathcal{A}}_l = \phi$:
   (a) For each tunnel $l' \in \tilde{L}_r$:
   i. Compute $n_{l'}$, the number of links in $\tilde{\mathcal{A}}_l$ that are covered by l'.
   ii. Compute the weight function $w_{l'} = \alpha_{\kappa(l')}\ \Delta^{+}_{\kappa(l')})/n_{l'}$.
   (b) Pick the tunnel from $\tilde{L}_r$ that has the maximum weight function. Let this tunnel be l''. Then,
   i. Include l'' in S_r.
   ii. Remove l'' from $\tilde{L}_r$.
   iii. Remove all the links covered by l'' from $\tilde{\mathcal{A}}_r$.

Min-Loss_Route_Algorithm (MRA)

1. From the given directed network graph $G = (V, E)$, and the set of tunnels L, form a graph $G' = (V, E')$ in the following way:
   (a) For each tunnel $l' \in L$, do the following:
For each node v on tunnel l''s route:
   Add directed edges of cost $\alpha_{\kappa(l)}\ \Delta^{-}_{\kappa(l)}$ to *all downstream nodes* of v on tunnel l''s route.
   (b) For any two nodes u and v, remove all the edges from u to v except for the minimum-weight edge from u to v.
2. Run Dijkstra's algorithm to compute the minimum-weight path from the ingress node to the egress node of the pair $\kappa(l)$. This is the new route (min-loss route) for tunnel l, $\tilde{r}(l)$.

It can be verified that the min-weight path in the graph G' corresponds to the min-loss route in G, and vice versa. Moreover, note that the **Min-Loss_Route_Algorithm** also provides the optimal set of tunnels that cover the links in the new route. To see this, note that each edge in G' is mapped to a tunnel in G. The optimal set of tunnels covering links in the new route is thus obtained by picking the tunnels in G that correspond to the edges in the min-weight path in G'. Note that the complexity of the **Min-Loss_Route_Algorithm** is polynomial in the network size, although it depends on the total number of tunnels, like the other algorithms discussed before. Algorithmically, this represents an interesting scenario where the optimal path based on a cost-metric can be obtained in polynomial time, whereas the (seemingly simpler) problem of finding the cost of a given path is NP-hard in general (since the later problem corresponds to the set-cover problem). Note that if the new route is computed by some other approach, then **Greedy_Link_Cover** can be used to find a (possibly non-optimal) set of tunnels covering the links in the new route.

C. Simulation Results

For simulation experiments, we use the same network as shown in Figure 2, and traffic profile B. In this section, we compare the performance of our joint route and bandwidth reassignment policy (BRA+TRA) against that of our fixed-route bandwidth re-assignment algorithm (BRA). In addition, we also compare the performance of (BRA+TRA) against that of the optimal dynamic network provisioning policy. The optimal network provisioning policy can be obtained by solving the nonlinear mixed-integer programming problem posed in Section II. In a dynamic scenario, this problem need to be solved at regular intervals, or whenever the traffic pattern changes significantly. Solving mixed-integer programs is computationally expensive. Therefore, we solve a relaxed version of the mixed-integer program instead. In this relaxed problem, the constraint on the maximum number of tunnels in use for an ingress-egress pair is relaxed, thus reducing the problem to a continuous flow optimization problem with a non-linear objective function. Solving this problem provides a set of routes (the number of routes could be

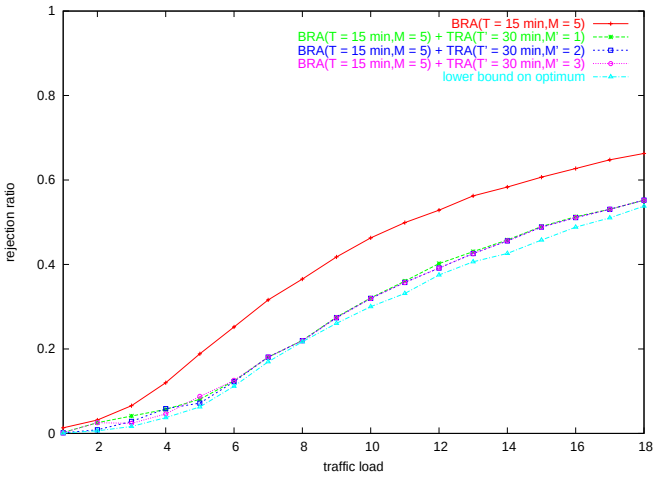

Fig. 10. Rejection rate vs. average traffic load for our algorithms (BRA and (BRA+TRA)), and the dynamic optimal policy.

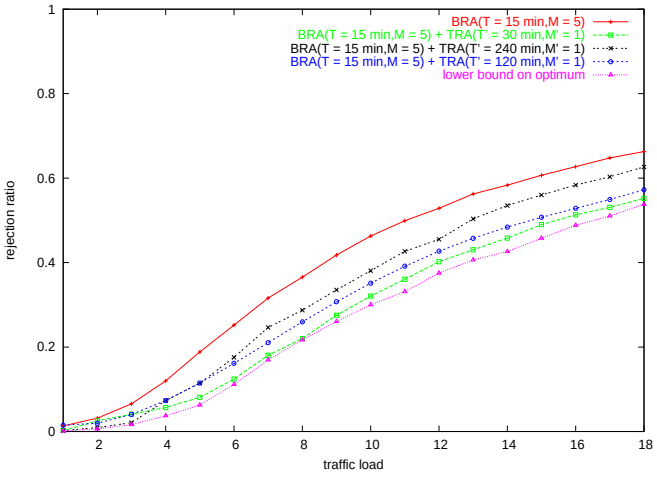

Fig. 11. Rejection rate vs. average traffic load for our algorithms (BRA and (BRA+TRA)), and the dynamic optimal policy.

much larger than the pre-specified upper limit on the number of tunnels), and their bandwidth assignments. In our simulations, this optimization process is run every hour, based on the current traffic matrix, and the set of tunnel routes and bandwidths obtained from the relaxed optimization problem are used over the next hour period. Clearly, the rejection rate of this 'relaxed algorithm' is expected to provide a lower bound on the rejection rate of the optimal dynamic network provisioning policy, as defined in Section II. In the simulation results shown in this section, we use this lower bound as a characterization of the performance of the optimal network provisioning policy.

Figures 10 and 11 plot the connection rejection rate as a function of the average traffic load, for different values of the simulation parameters T, M, T', and M'. The traffic profile used is Profile B, as shown in Figure 4. For the TRA algorithm, we assume the upper-limit on the number of tunnels is three for ingress-egress pairs 0, 2, and two for ingress-egress pairs 1, 3.

From Figure 10, we see that the performance of (BRA+TRA) is significantly better than that of only BRA. Moreover, even when our tunnel re-routing is done infrequently (once in 30 mins

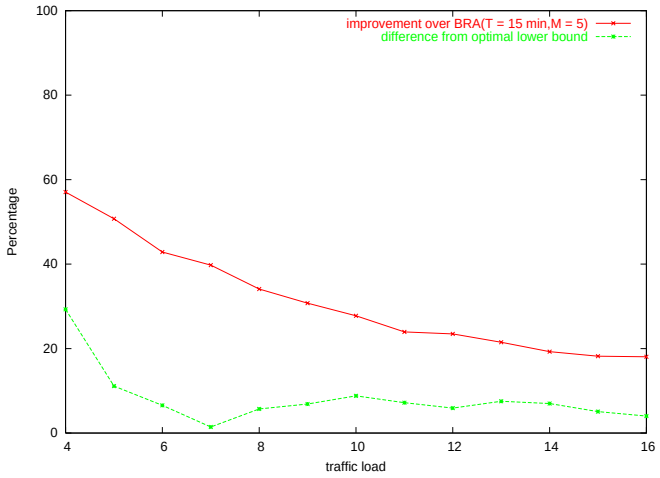

Fig. 12. Performance of (BRA+TRA) (T' = 30 minutes and M' = 1) with respect to BRA and the lower bound on the optimum performance.

in this case), and only for a few routes at a time (once, twice or thrice in each run of TRA), the performance of (BRA+TRA) is very close to that of the best achievable performance. This suggests that our incremental tunnel re-routing and bandwidth reassignment scheme performs close to the optimal dynamic network provisioning policy, while incurring a much smaller overhead of computation and communication. Moreover, the figure shows that the performance improvement of the cases $M' = 2, 3$ over $M' = 1$ is only marginal. Therefore, only a small number of routing updates are needed at the decision times for a good performance, and a large number of simultaneous tunnel re-routes do not lead to any significant additional performance improvement.

Figure 11 shows the variation of the performance with respect to the re-routing interval T', when M' is kept fixed at 1. As expected, the plots show that the performance of our scheme improves as the interval between successive runs of our re-routing algorithm, T', is reduced. Figure 12 shows the percentage difference in the performance of (BRA+TRA) w.r.t. that of BRA and the lower bound on the optimum.

V. CONCLUDING REMARKS

In this paper, we have presented a dynamic network provisioning algorithm that adaptively allocates bandwidths and routes to tunnels so as to maximize the overall connection acceptance rate. Since our algorithm takes into account dynamic variations in the traffic profiles, it significantly outperforms the best static bandwidth allocation policy. Moreover, although our algorithm is incremental in nature and simple to implement, it achieves a performance fairly close to that of the optimal dynamic network provisioning policy (which may not be practically feasible for most networks).

ACKNOWLEDGMENTS

The authors would like to thank Samir Khuller of the University of Maryland, College Park, for his help on the design and proof of correctness of the **Min-Loss_Route_Algorithm**.

REFERENCES

[1] "Requirement for traffic engineering over MPLS," IETF RFC 2702.

[2] T. Anjali, C. Scoglio and J. C. de Oliveira, "New MPLS Network Management Techniques Based on Adaptive Learning," *IEEE Transactions on Neural Networks*, Vol. 16, No. 5, September 2005.

[3] S. Bessler, "Label switched paths re-configuration under time-varying traffic conditions," ITC Specialist Seminar, Wurzburg, July 2002.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

[5] B. Davie and Y. Rekter, *MPLS - Technology and Applications*, Academic Press 2000.

[6] N. Md. Din and N. Fisal, "Dynamic Resource Allocation of IP Traffic for a Diffserv-MPLS Interface using Fuzzy Logic," *Proc. 9th Asia Pacific Conference on Communications (APCC)*, 2003.

[7] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," *Proc. IEEE INFOCOM 2001*, Anchorage, March 2001.

[8] D. Hochbaum (Ed.), *Approximation Algorithms for NP-Hard Problems*, Brooks Cole Publishers, 1996.

[9] K. Kar, M. Kodialam, T. V. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications," *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 12, December 2000.

[10] L. Kleinrock, *Queueing Systems (Vol. II)*, John Wiley & Sons, 1975.

[11] H. Levy, T. Mendelson and G. Goren, "Dynamic Allocation of Resources to Virtual Path Agents," *IEEE/ACM Transactions on Networking*, Vol. 12, Issue 4, August 2004.

[12] F. Ricciato, S. Salsano, A. Belmonte, M. Listanti, "Offline configuration of a MPLS over WDM network under time-varying offered traffic," *Proc. IEEE INFOCOM 2001*, Anchorage, AK, March 2001.

[13] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol label switching architecture," IETF RFC 3031.

[14] S. Ross, *Stochastic Processes*, Wiley, 1995.

Vicky Sharma received the M.S. degree in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, USA in 2006, and is currently pursuing the Ph.D. degree at the same institute. He received the B.Tech. in Electrical Engineering from the Indian Institute of Technology, Kanpur, India in 2002. His research interests include traffic engineering, wireless transport layer design and medium access control.

Koushik Kar received the Ph.D. and M.S. degrees in Electrical & Computer Engineering from the University of Maryland, College Park, USA in 2002 and 1999, respectively. He received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Kanpur, in 1997. Since 2002, he has been an assistant professor in the Electrical, Computer & Systems Engineering department at Rensselaer Polytechnic Institute, Troy, NY. His research interests include performance optimization questions in ad-hoc and sensor networks, traffic engineering, congestion control and multicasting. Dr. Kar received the CAREER Award from the National Science Foundation in 2005.

Richard La received his B.S.E.E. from the University of Maryland, College Park, USA in 1994 and M.S. and Ph.D. degrees in Electrical Engineering from the University of California, Berkeley, USA in 1997 and 2000, respectively. From 2000 to 2001 he was with the Mathematics of Communication Networks group at Motorola Inc,. Since 2001 he has been on the faculty of the Department of Electrical and Computer Engineering at the University of Maryland, where he is currently Associate Professor.

Leandros Tassiulas obtained the Diploma in Electrical Engineering from Aristotle University of Thessaloniki, Greece in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Maryland, College Park, USA in 1989 and 1991 respectively. He is currently a Professor in the Dept. of Computer and Telecommunications Engineering, University of Thessaly, Greece. He has held positions as Assistant Professor at Polytechnic University, New York, USA (1991-95), Assistant and Associate Professor at University of Maryland, College Park, USA (1995-2002), and Professor at University of Ioannina, Greece (1999-2002). His research interests are in the field of computer and communication networks with emphasis on architectures and protocols of wireless systems, sensor networks, high-speed internet, satellite communications and stochastic network optimization. Dr. Tassiulas received a National Science Foundation (NSF) Research Initiation Award in 1992, an NSF CAREER Award in 1995 an Office of Naval Research, Young Investigator Award in 1997 a Bodosaki Foundation award in 1999 and the INFOCOM '94 best paper award.