On Bandwidth Efficiency of the Hose Resource Management Model in Virtual Private Networks

Alpár Jüttner*†, István Szabó* and Áron Szentesi*
*Traffic Analysis and Network Performance Laboratory, Ericsson Research, Budapest, Hungary.
E-mail: {alpar.juttner, istvan.szabo, aron.szentesi}@eth.ericsson.se
†Computer Networks Laboratory, Eötvös University, Budapest,
Pázmány Péter sétány 1/A, H-1117 Hungary, E-mail: alpar@cs.elte.hu

Abstract— **The hose resource provisioning model promises to provide an easy–to–use characterization framework for Virtual Private Network service offerings. Significant research effort has recently been spent on proposing new algorithms for provisioning cost–optimal networks specified according to this new model. However, a detailed comparison of the bandwidth requirement for networks designed based on the hose model and networks designed based on the traditional pipe model has not been performed. The first contribution of this paper is a detailed comparison of the bandwidth needs of the two models assuming a range of network sizes and network topologies. This numerical evaluation required efficient calculation methods for determining resource allocation based on the hose model parameters, therefore, a linear programming based formulation is also presented for this purpose. The second contribution is the calculation of a lower bound for the hose based realization. This lower bound is very useful in evaluating the two models given that the problem of provisioning a minimal cost network based on the hose model specification can only approximately be solved in polynomial time.**

I. Introduction

Virtual Private Network (VPN) service offerings are playing an increasing role in the revenue stream of network operators. Operators need a flexible and bandwidth efficient model to be able to support a wide variety of customer needs in terms of capacity, network topology and communication patterns. Traditionally, resource provisioning for virtual private networks is done in a way that traffic demand is specified for each node pair belonging to the VPN and resources are reserved for point–to–point pipes between these VPN endpoints. Duffield et al. in [1] propose a different resource provisioning model they call the "hose model". In the case of the hose model, there is no need for a complete traffic matrix, but only the total amount of traffic which a node injects into the network and the total amount of traffic which it receives from the network shall be specified (in fact, the customer's interface at a network node — providing access to all other nodes in the network — is called a hose, and a VPN is constructed of as many hoses as many nodes it has). In [1] they argued that the hose model has key advantages compared to the traditional model (in their terminology, the "pipe model"): ease of specification, flexibility and multiplexing gain.

Besides the aforementioned undoubtable benefits of the hose model, there is one cornerstone issue that primarily determines the feasibility of using this resource provisioning technique in practice: *bandwidth efficiency*. Recalling the basic concept of [1], in case of static resource provisioning, the service provider has to permanently allocate resources in the backbone network that can accommodate all the "worst case" traffic patterns the hose specification allows for. This can give rise to some strong concerns whether how much overprovisioning does this mean in reality.

One can argue — as it is done in [1] — that dynamic reallocation can improve bandwidth efficiency a lot, however, this is a rather complex and unproven technology. First and foremost, a new signaling protocol is needed which is capable of carrying hose–specific state information in order to allow dynamic resource allocation for the hoses. Not just a widely accepted standard protocol is missing here, but — to our best knowledge — there is not even a first research proposal for such a protocol. Therefore, in the rest of the paper we concentrate on static provisioning as the only currently viable alternative.

The main question that immediately comes into the mind of operators considering to introduce the hose model in their networks is how does it relate to the traditional "pipe" solution in terms of bandwidth efficiency. The amount of network resources required for implementing a hose based VPN is a result of three main factors:

1) Statistical multiplexing.
2) Resource requirement of dimensioning for the "worst–case" traffic distribution allowed by the hose parameters.
3) Gain achieved by dynamic capacity reallocation.

In [1] factors 1 and 3 are thoroughly evaluated, although the analysis of the effect of statistical multiplexing is confined to the access links. However, up to our best knowledge, factor 2 — although, as we will see, it has a really significant implication on the resulting bandwidth efficiency — has never been evaluated in detail, and, especially, it has not been compared to the pipe approach. Therefore, the main contribution of our paper is a thorough analysis and comparison of the resource requirements of VPNs based on the two models, assuming different network sizes, network topologies, traffic demands, and different realization alternatives.

The remainder of this paper is organized as follows. Section II presents a short survey on the related literature. Section III presents our network and traffic model as well as the realization alternatives we investigate. Section IV presents a linear programming based solution for calculating resource requirements of arbitrary-topology VPNs based on the hose model, and also a lower bound for the total resource requirement of a hose-based VPN. Section V evaluates the bandwidth need for realizing a VPN based on the hose and the pipe model using randomly generated networks and traffic matrices. Finally, Section VI summarizes the results and lists items for future work.

II. Related work

The main concept of the hose model (i. e., the characterization of user traffic as per-node aggregate ingress and egress traffic volumes instead of a traffic matrix containing point–to–point demands for each node pair) has long been present in the literature under the theory of "nonblocking networks". For example, Fingerhut et. al. in an earlier paper [2] present a network design methodology based on just the same resource provisioning concept as the hose model. The paper also gives a short literature survey on the issue.

As we already mentioned in the Section I, Duffield et al. in [1] were the first to propose this concept for provisioning IP Virtual Private Networks. In their paper an analysis on the bandwidth efficiency of the hose model is presented. The evaluation is based on trace driven simulations of traffic derived from a voice network and from a large corporate network. However, their network–level analysis is limited to a single topology (a simplified, 12–node approximation of AT&T's continental backbone) and they provide numerical results for network–wide capacity demand for different hose realizations, but no comparison to the pipe model is presented. Another problem is that comparison of the bandwidth efficiency of the traditional pipe and the new hose model — which, in our opinion, is the main question here — is limited to the access links, although the real overprovisioning required by the hose model will be present within the backbone network.

The paper from Duffield et al. [1] inspired research work on developing algorithms for designing minimum cost networks based on hose specifications. Kumar et al. in [3] argued that optimal cost solution for hose realizations shall be based on tree topology, and they proved that the general problem with asymmetric hoses (different amount of traffic sent and received by the hose) and constrained link capacities is $\mathcal{NP}$-hard. Consequently, they provided a couple of approximation algorithms with provable performance bounds. Numerical results presented in the paper concern only the hose case, no comparison is provided with the pipe model. The latest important contribution in this field improves the tree–based hose realization by proposing restoration algorithms [4].

III. Network and traffic model

In order to facilitate easy calculations, we defined a simple network and traffic model that contain only the most important parameters needed for dimensioning. We represent the network configuration by a directed graph $G = (V, E)$ (all edges are bidirectional, they in fact represent two links with two distinct capacities). Traditionally, subscriber traffic is represented by a traffic matrix $T = \{t_{ij}\}$, describing the required bandwidth between each VPN endpoint pair. In case of the hose model, we only need to define the maximal ingress traffic (t_n^i) and the maximal egress traffic (t_n^o) at each site, regardless of the exact traffic split. From the "traditional" traffic description, hose model parameters can be calculated as summing up rows/columns in the traffic matrix (this way, we get the set of "smallest" necessary hoses that can accommodate the given traffic matrix). Furthermore, we assume (arbitrary) fixed routing in the network. Expressed in mathematical terms, for each pair of nodes u, v we are given a vector $r_{uv} \in \mathbb{R}^E$ determining the traffic route from u to v that is, for a link $e \in E$ the value $r_{uv}(e)$ shows how much portion of the traffic from u to v goes through the link e. Restricting $r_{uv}(e)$ to the integer set of $\{0, 1\}$ we can model single path routing. Enabling $r_{uv}(e)$ to take any real value from the interval $[0, 1]$ makes our notation capable of modeling an arbitrary load sharing mechanism (e. g. Equal Cost Multipath (ECMP) in the Open Shortest Path First (OSPF) routing protocol [5]).

A. Dimensioning alternatives

In the following we list the most important dimensioning approaches (for a more detailed explanation of the different approaches we refer the interested reader to [1]) and shortly outline the calculation of the corresponding resource requirements. For simplicity, we adopt the naming convention of [1].

1) Customer–pipe model (or, shortly, the pipe model): This approach models the "traditional" way of provisioning Virtual Private Networks. Traffic between each (i, j) pair of customer access points is carried through "customer–pipes" (i. e., point–to–point connections with a given pre–allocated capacity). Each customer–pipe is dimensioned according to the corresponding traffic matrix element t_{ij}. Calculating the total resource allocation is simply to add up the traffic matrix capacities at each link, of course considering only those origin–destination site pairs whose route traverses this particular link (determined by the routing scheme $r_{uv}(e)$). The routing pattern can be arbitrary, however, if we consider shortest paths between each node pair, we can observe that this method provides optimal[1] dimensioning (disregarding the possible benefits of dynamic re–allocation and statistical multiplexing). Later on this will enable us to use the total capacity obtained with the customer–pipe model as a reference to calculate the relative extra capacity needed by any other dimensioning approach.

2) Provider–pipe approach: This is probably the simplest way of dimensioning that can ensure that the resulting capacities will fulfill the requirements of the hose model based traffic characterization. Here, we allocate a "provider–pipe" for the worst–case traffic value between each node pair (i. e., the minimum of t_k^i and t_l^o for a node pair (k, l)) along the path between nodes k and l and simply add up the capacity of these pipes on each link in order to calculate the total VPN capacity requirement on the link. Since here we allocate the worst–case capacity between each node pair independently from

[1]W. r. t. the total capacity allocated for the VPN in the network.

other capacity demands, we cannot exploit the interrelation between the hose constraints. Therefore, as we will see later at the numerical evaluation, this approach will be the least bandwidth–effective.

3) Hose–specific state: The basis for improving the bad capacity utilization of the Provider–pipe approach is that instead of adding up worst–case pipes on each link, we exploit the relation of the hose parameters on the hose level. This means that on links carrying multiple pipes originating from a single ingress node (or destined to a single egress node) we do not simply add up the pipe capacities as in the previous case, but we allocate the minimum of the sum and the respective hose parameter of the ingress (or egress) node. In this approach, we look at a whole hose at a time (i. e., traffic originating at a single site and destined to all other sites and vice versa).

4) VPN–specific state: The third alternative for dimensioning a VPN according to the hose model is to calculate the worst case traffic of each link by considering all hose model parameters in the network. The idea is similar to the calculation of the "Hose–specific state" approach, however, the calculation of the capacity requirements on the links is much more complex (there are much more constraints). In Section IV we present a linear programming formulation for the calculation of the capacity requirements on the links.

It is obvious that if we assume the same routing pattern, the following relation holds between the different hose model variants:

$$BW_{provider-pipe} \geq BW_{hose-specific} \geq BW_{vpn-specific}$$

for the bandwidth of each link and, of course, in total as well.

5) Tree routing: The routing pattern has an important effect on dimensioning. By letting the traffic take routes other than the shortest paths, we can obtain solutions with significantly better resource utilization than in the case of simple shortest path routing. In [3], the authors consider tree–topology VPNs, i. e., all the connections are restricted to use a tree subset of the original topology. It is proven that the optimal tree can be found by computing the shortest path tree for each nodes, and choosing the best one from among these trees. For simplicity, in our investigations we will use arbitrary shortest path trees (link weights are set to 1) for this purpose. As also explained in [3], since in any tree each link cuts the VPN into two disjoint parts, we can very easily calculate the capacity of a link in this tree, according to the given set of hose parameters. We simply take the minimum of:

- The total ingress capacity of the sites that are on the "source side" of the link,
- The total egress capacity of the sites that are on the "destination side" of the link.

However, since the routing in this case is constrained to use a tree, the path length can be significantly higher than in case of the VPN–specific state version, which might give rise to delay concerns. Also, there will be extra capacity required to implement the necessary redundancy [4].

IV. Linear programming based capacity calculation

In this section we present two linear programming formulations. The first formulation computes the smallest necessary link capacities considering a given network topology, a set of hose parameters and a system of routing paths[2]. The second formulation provides a lower bound on the total required capacity for a given network topology and set of hose parameters. This bound can be explicitly calculated by our linear programming formulation in case of smaller–size networks. The meaning of this lower bound is that it is impossible to go below this number with any kind of hose model–based dimensioning, using any link capacity allocation and any kind of routing (including load sharing).

At the end of this section we also outline possible further research directions, applying our formulation for network topology and routing optimization.

A. Calculation of link capacities for the "VPN-specific state" case

Based on the notations of Section III, the necessary capacity for link e can be calculated as:

$$\max \sum_{u,v \in V} r_{uv}(e) x_{uv} \tag{1a}$$

subject to

$$x_{uv} \geq 0 \qquad \forall u, v \in V \tag{1b}$$

$$\sum_{v \in V} x_{uv} \leq t_u^o \qquad \forall u \in V \tag{1c}$$

$$\sum_{u \in V} x_{uv} \leq t_v^i \qquad \forall v \in V \tag{1d}$$

$$\tag{1e}$$

Here the variable x_{uv} indicates the amount of the traffic that goes from u to v in a certain traffic situation. Inequalities (1c) and (1d) force the traffic situation to fulfill the ingress and egress traffic limits and we find maximum possible traffic on the link e.

Since the above formulation is a pure linear program with $|V|^2$ variables and $2|V|$ constraints, it can be easily solved with any LP-solver software library (in our numerical evaluations we used the software package `lp_solve` [6]). Of course we have to repeat this process for each edge $e \in E$ in order to get the total necessary bandwidth.

In case when the speed of calculation is of great importance, an alternative method can be used instead of using a general LP library. Namely, the above linear program can also be formalized as a maximum cost flow problem. For a certain link e we define the graph $H_e := (\{s, t\} \cup V_1 \cup V_2, E_1 \cup E_2 \cup E_3)$ (see Fig. 1), where V_1 and V_2 are two disjoint copies of the

[2]Note that our formulation is able to handle load sharing (such as ECMP) as well as single path routing.

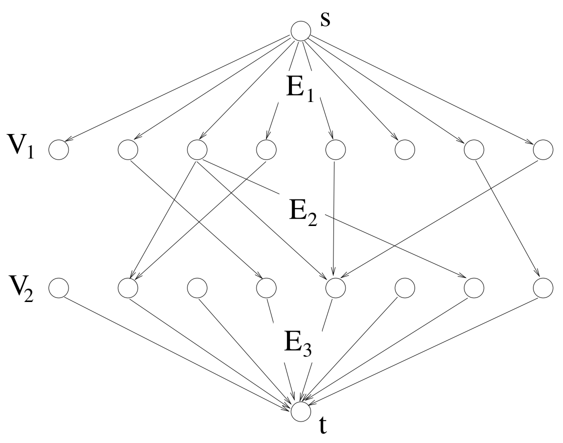

Fig. 1. Flow formulation

set V and

$$E_1 := \{(s, v) : v \in V_1\} \tag{2a}$$
$$E_2 := \{(u_1, v_2) : u_1 \in V_1, v_2 \in V_2, r_{uv}(e) > 0\} \tag{2b}$$
$$E_3 := \{(v, t) : v \in V_2\}. \tag{2c}$$

The capacities of the edges $(s, v_1) \in E_1$ and $(v_2, t) \in E_3$ are t_v^o and t_v^i, respectively. The edges in E_2 have infinite capacities. The cost of each edge $(u_1, v_2) \in E_2$ is r_{uv}^e. All other edges have 0 cost.

It can be seen as well, that the optimum value of the above maximum cost s-t flow problem is just the worst case load of the link e in the hose model.

B. Lower bound

The cut condition is used in order to get a lower bound to the original problem.

For a set $X \subseteq V$, let

$$\delta(X) := \{(uv) \in E : u \in X, v \notin X\},$$

the set of the edges leaving the node set X.

It is clear that for any set $X \subset V$, all the traffic from X to $V \setminus X$ will use at least one edge from $\delta(X)$, so, the sum of the capacities of the edges in $\delta(X)$ must be at least $\min(\sum_{v\in X} t_v^o, \sum_{v\in V\setminus X} t_v^i)$. Thus a solution of the following linear program gives a lower bound of the sum of the necessary capacities irrespectively of the used routing method:

$$\min \sum_{e\in E} x_e \tag{3a}$$

subject to

$$x_e \geq 0 \qquad \forall e \in E \tag{3b}$$
$$\sum_{e\in\delta(X)} x_e \geq \min(\sum_{v\in X} t_v^o, \sum_{v\in V\setminus X} t_v^i) \qquad \forall X \subsetneq V, X \neq \emptyset \tag{3c}$$

Unfortunately, this formulation has a huge number of constraints. In fact, since graph G is symmetric (i.e. for each link $e \in E$ there exists another link $e' \in T$ between the same nodes but in the opposite direction), the variables of the oppositely directed edges can be merged together, thus the number of variables and constraints can be halved. In order to give this formulation, let $\overline{E}$ be the set of undirected edges corresponding to the edges in E. For a set $X \subseteq V$ let $d(X)$ denote the set of undirected edges having exactly one endpoint in X. Finally, let $s \in V$ be an arbitrary fixed node. With these notations the lower bound calculation transforms to the following linear program:

$$\min \sum_{e\in\overline{E}} x_e \tag{4a}$$

subject to

$$x_e \geq 0 \qquad e \in \overline{E} \tag{4b}$$
$$\sum_{e\in d(X)} x_e \geq \min(\sum_{v\in X} t_v^o, \sum_{v\in V\setminus X} t_v^i) + \min(\sum_{v\in X} t_v^i, \sum_{v\in V\setminus X} t_v^o) \qquad \forall X \subsetneq V, s \in X \tag{4c}$$

To solve this linear program, we use a simple row generation method. In each iteration we have a linear program that is a subprogram of the above one. We start with an empty program only with the non-negativity constraints. Then in each iteration we find the optimal solution x to the current problem. After this we check whether x is a feasible solution to (4). If it does not satisfy all of the constraints (4c), we add some of the failed inequalities to the stored linear program, and we iterate this until the found vector x is a feasible solution to (4).

The above simple method makes it possible to compute a theoretical lower bound of the total link capacities up to 22 nodes on a Sun Ultra-5 workstation.

C. Network optimization

Although in this paper the focus is not on network optimization but only the comparison of different resource allocation methods, we mention that the fast method presented in Section IV-A for computing the necessary link capacities of a certain network topology and routing configuration makes it possible to develop an efficient algorithm for optimizing the routing or the network topology.

These algorithms can be based on several kinds of metaheuristics such as Simulated Annealing [7], [8], Genetic Algorithm [9] or Tabu Search [10]. All these are general optimization methods that require a subroutine to compute the cost of a given feasible solution and they are able to find a solution with low cost by starting with one or more feasible solutions and heuristically varying them in several iterations. In the terminology of metaheuristics the set of feasible solutions is called “state space”. Of course, it depends on the exact optimization task.

For example, if the task is to find a network topology with minimal total bandwidth (or with minimal cost where the cost of a link is a function of its bandwidth), and the routing is along the minimal hop paths, then each element of the state space consists of a possible set of installed links. Such a set determines the routing, thus the required bandwidths of the links can be computed using the algorithm of Section IV-A.

As another example, in case of optimization of the routing in OSPF, the network topology is given, the routing is along the shortest paths according to the weights on the links and the optimization task is to find those weights for the links that minimize the total bandwidth, or the total link overload in the worst case if the capacities of the links are given. In this case the elements of the state space are the possible configurations of the administrative weights.

V. Numerical results

Probably the most important characteristic that determines the usability of the hose model is the bandwidth efficiency — the numerous other advantages (ease of specification, flexibility) will probably never pay off if the new model requires significant overprovisioning compared to traditional resource provisioning algorithms. As we have outlined in the Introduction, bandwidth efficiency is the result of three main factors:

1) Statistical multiplexing.
2) Resource requirement of dimensioning for the "worst–case" traffic distribution allowed by the hose parameters.
3) Gain achieved by dynamic capacity reallocation.

We devote Sections V-B and V-C to the thorough numerical evaluation of factor 2. In addition, we present further analysis on factor 1 (statistical multiplexing) in Section V-D in order to provide further insight to the applicability of the hose model.

A. Methodology and definitions

Evaluation of factor 2 (i. e., the extra capacity needed to dimension for all the worst case traffic situations allowed by the hose model) can be done as the calculation of bandwidth volumes (we want to compare the required bandwidth reservations), therefore, there is no need to do any detailed network simulation; it is perfectly enough if we stay at the level of the graph–theoretical model defined in Section III. On the other hand, factor 1 will need a more detailed traffic description, which will be presented in Section V-D.

Our goal when trying to characterize the required bandwidth reservation of the hose model was to be able to compare its total network capacity requirement to the customer–pipe model. Naturally, we wanted to have "comparable" network examples with "comparable" traffic volumes when doing the evaluation, which is not trivial if we recall that the main difference of the hose and the customer–pipe dimensioning models is in the traffic description. Our solution for this was the following:

1) We generate a traffic matrix $T = \{t_{ij}\}$ containing point–to–point bandwidth values required to dimension according to the customer–pipe model.
2) As a "comparable" hose characterization we calculate the hose parameters as the respective sums of the rows and columns of T ($t_n^i = \sum_{k=1}^{N} t_{nk}$, $t_n^o = \sum_{k=1}^{N} t_{kn}$). As we have already mentioned in Section III, this way we calculate the smallest possible hose that can accommodate traffic matrix T (and, of course, many others).

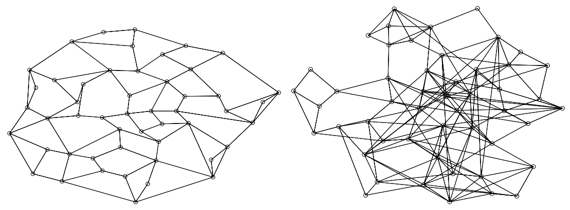

Fig. 2. Typical test topologies: sparse (link/node=2.5) and dense (link/node=6) — 50 nodes

After dimensioning the same network according to these inputs with both models (considering any of the different realization alternatives of the hose model, see Section III), we calculate the ratio between the resource requirements of the hose and the customer–pipe models, which we consider a good indicator of the required extra capacity. In the following we call this ratio as the *overprovisioning factor*. The benefit of using this measure is twofold:

1) It describes the required extra link capacity in an easy–to–understand way.
2) It eliminates the dependence on the total traffic volume (of course, dependence on traffic distribution still remains).

Of course, the overprovisioning factor defined this way will depend very much on the selection of the traffic matrix T. We tried to eliminate this dependence using a statistical approach: calculating a large number (a couple of hundreds) of randomly generated T's for each topology, we took the average of the resulting overprovisioning factors and used this value for comparison. The generation of the random T matrices was simply done by generating each t_{ij} element of T separately as an independent, uniformly distributed random variable. This way, the resulting hose parameters will be rather asymmetric (t_n^i and t_n^o can be very different).

Our initial tests have shown that the (average) overprovisioning factor will depend also very much on the network size and the type of topology. Therefore, we have calculated this value for a series of networks of different sizes and topology types. This, however, implies that we had to generate random topologies. We have used our own Random Graph Generator described in [11]. We have developed this generator utilizing some of the results published recently in this area, e. g. [12].

B. Dependence of the overprovisioning factor on the network size

In our first test the main goal was to capture the dependence of the overprovisioning factor on the number of nodes in the network. However, in order not to neglect the effect of the "density" of the network topology (i.e., the ratio of the number of links and nodes in the network), in the first series of tests we have used two topology types. The first (sparse) one with a link/node ratio of 2.5, and a second (dense) one with a link/node ratio of 6 (Fig. 2).

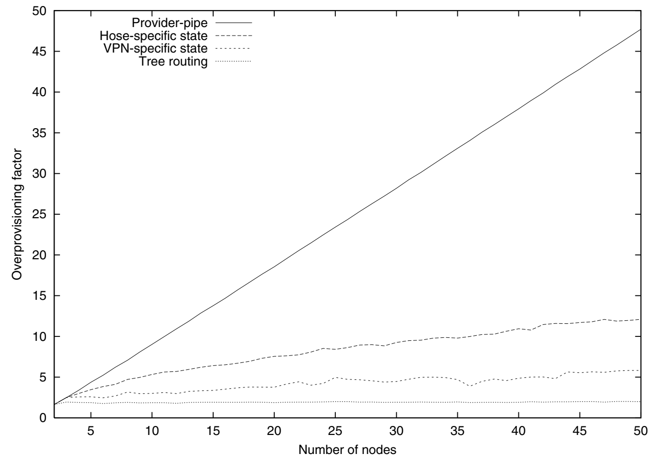

Fig. 3. Dependence of the overprovisioning factor on the network size (sparse topology type)

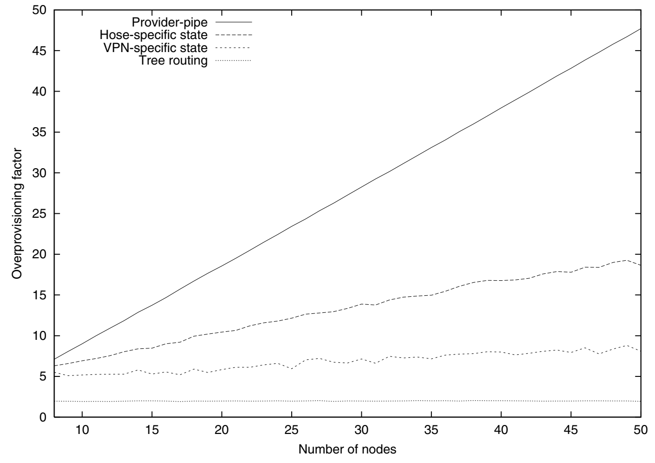

Fig. 4. Dependence of the overprovisioning factor on the network size (dense topology type)

As it is obvious from the charts Fig. 3 and Fig. 4, the "provider–pipe" and the "hose–specific state" versions of the hose model are practically useless, since the overprovisioning factor increases linearly with the network size[3]. In the following we don't waste any more words on them.

Instead, we have magnified the results of the "VPN–specific state" and the "Tree routing" versions in a single chart (Fig. 5).

We can observe that the overprovisioning factor of the "VPN–specific state" version is in the range of 2.5–6 for the sparse and 5–9 for the dense topology type. This means that for example in case of a 30 site network we need 4.5–7 times the capacity of the traditional customer–pipe network to dimension it with the hose model. We can also note that the overprovisioning factor stays below 2 in all cases if we restrict the routing to a tree. Moreover, the average overprovisioning factor for the "VPN–specific state" version increases with the

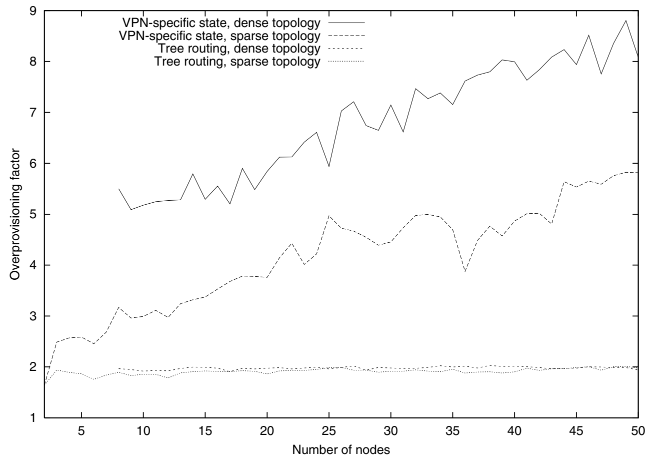

Fig. 5. Dependence of the overprovisioning factor on the network size (both topology types)

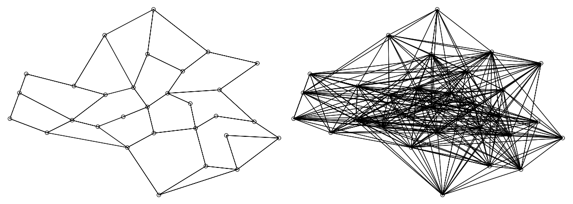

Fig. 6. Topologies with the least and the most number of links

network size, while it is more or less constant if we use tree routing.

C. Dependence of the overprovisioning factor on the network density

We can also note from Fig. 5 that in case of the "VPN–specific state" version the overprovisioning factor also increases with the density of the topology. We decided to do some tests also to investigate this issue. We have generated a series of networks with the same number of nodes but increasing number of links. We have investigated networks with 30 nodes. The number and location of the sites were the same in all test cases, but we have gradually increased the number of links. The topology with the least number of links had a link/node ratio of 3, while the topology with the most number of links had a link/node ratio of 19 (Fig. 6).

Fig. 7 shows how the overprovisioning factor of the "VPN–specific state" version increases with the topology density.

The overprovisioning factor for the "Tree routing" version is, however, constant with respect to the number of links[4] (Fig. 8).

Our main observation is, therefore, that the average overprovisioning factor for the "VPN–specific state" version increases with the topology density, while it is more or less constant for the "Tree routing" version.

[3]This is no surprise since they implement the same specification as the "VPN–specific state" version, however, with less efficient resource allocation, as also pointed out in [1].

[4]Of course, this is not surprising since in this case we use only a tree subset of the topology, therefore the additional links will not affect the results very much.

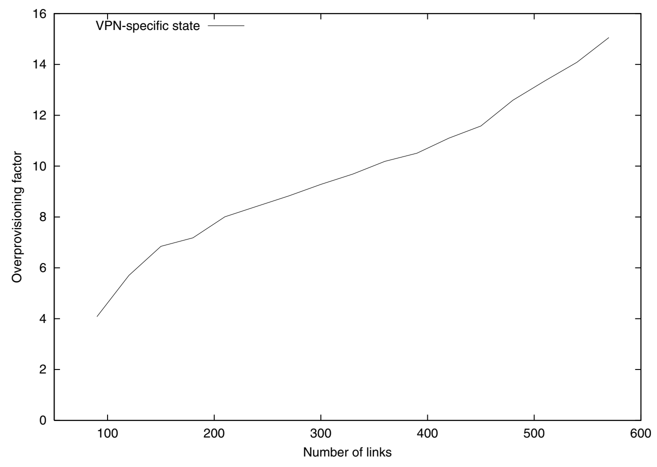

Fig. 7. Overprovisioning factor of the "VPN–specific state" version as the function of the number of links

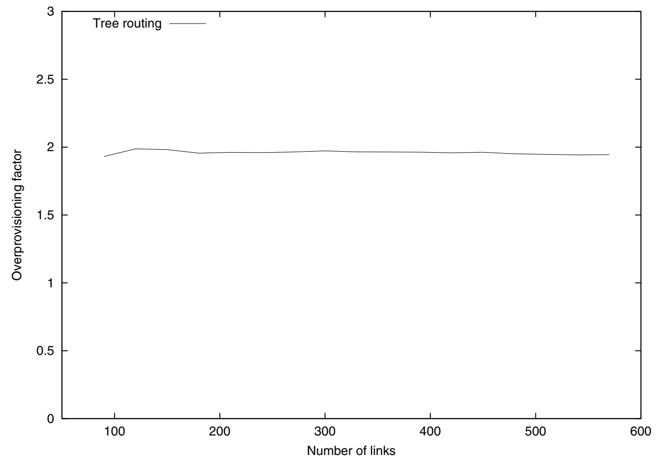

Fig. 8. Overprovisioning factor of the "Tree routing" version as the function of the number of links

In our final test we tested the dependence of the overprovisioning factor on the number of links in case of a smaller network. Our goal with this test was twofold:

1) We wanted to calculate our lower bound formula (it only works on smaller networks).
2) We wanted to analyze the effect of establishing additional links in the network (here, instead of having different networks with increasing topology density, we added links one–by–one to the *same* network).

The topology with the least number of links and the one with the most number of links can be seen in Fig. 9.

Fig. 10 depicts the total resource requirement of the "VPN–specific state" version, the "Tree routing" version, as well as the lower bound and the customer–pipe model in a single chart. Results are similar to what we found earlier in the larger network.

The overprovisioning factors (Fig. 11) look also similar to Fig. 7 and Fig. 8.

The most important observation is that (unlike in the case of

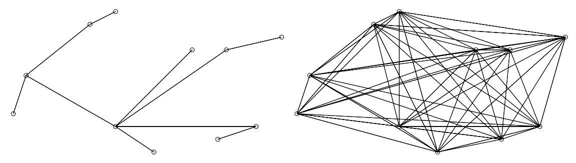

Fig. 9. Topology with the least and the with the most number of links (tree vs. full mesh)

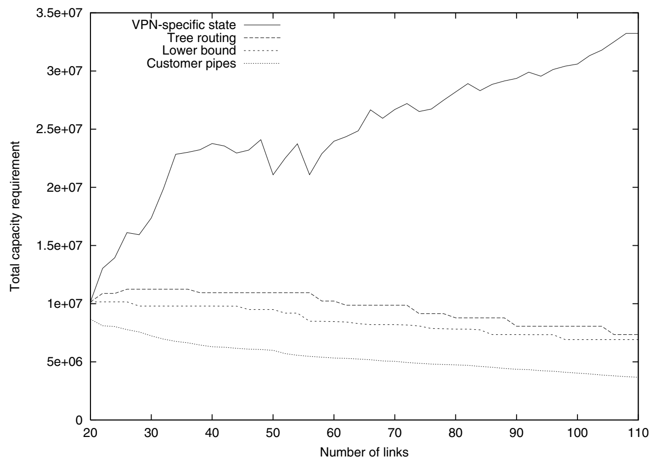

Fig. 10. Dependence of total resource requirement on the number of links in the network (small example)

the customer–pipe model) in most cases we do not do any good to the network bandwidth efficiency by introducing new links in case of the "VPN–specific state" hose version (note that in case of the "Tree routing" hose version adding new links has no practical meaning because always only a tree subset is used). This behavior can be better understood if we recall the fundamental difference between the traffic description of the two models:

- In case of the customer–pipe model, if we notice a significant traffic increase between two particular sites, it can be reasonable to introduce a direct link between those two sites in order to eliminate congestion.
- In case of the hose model, however, we do not have information on the traffic distribution on a site–pair basis. We can only notice that the total ingress traffic at a site has increased. This, however, gives us no direct clue where to put a new link.

The main conclusion of this last test is that in case of the hose model the addition of new links to the topology should be done along connectivity, reliability, QoS etc. reasons, but definitely not for increasing bandwidth efficiency.

D. On the flexibility of the hose model

As we have outlined in the beginning, probably the most important benefit of the hose model is its flexibility to changes in the traffic matrix. It means that traffic from a hose endpoint can arbitrarily be distributed among different destinations provided that the aggregate traffic demand does not exceed the hose specification. In case of the pipe model, the network

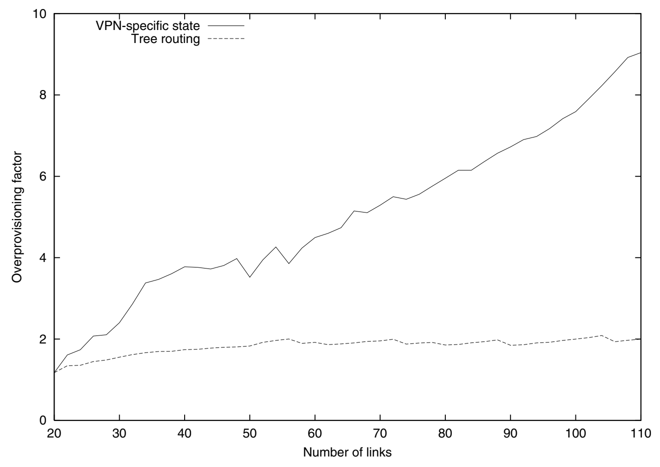

Fig. 11. Dependence of the overprovisioning factor on the number of links in the network (small example)

can only satisfy the quality demands of the offered traffic if the traffic from a particular endpoint conforms to a predefined distribution among the destinations. A deviation from the traffic distribution assumed during the provisioning process can have different consequences depending on the way traffic is handled in the network:

1) If the traffic is handled at packet level, an unexpected increase over a given path results in larger than expected packet loss and may also result in larger than expected packet delay due to increased buffer occupancy.
2) If the abstraction of "call" is used in the network, and there is some sort of call admission control exercised by the VPN endpoints for the calls destined to other VPN endpoints, then the result of a deviation from the originally assumed traffic distribution will be an unexpectedly high blocking probability.

An important question is if this flexibility can compensate for the price of the required extra capacity. Generally and comprehensively investigating the behavior of the network in the above two cases seem to be a daunting task. There are many legitimate combination of traffic mixes, network topologies and traffic source models.

There is a special case, which is relatively easy to characterize, and due to its increasing importance, the results are edifying. More and more corporations are establishing IP virtual private networks for carrying voice (telephony) traffic between their premises at different geographical locations, more and more operators are offering IP telephony services, and incumbent telecommunication service providers are also willing to carry voice traffic between the IP telephony gateways using IP VPN technology over a multiservice IP backbone network. Of course, these telephony VPNs can be provisioned using either the pipe or the hose model. There is a widely accepted method for calculating capacity requirement for telephony traffic, which has been used for many years in Public Switched Telephony Networks and it can also be used for IP telephony systems [13]. The offered traffic shall be specified in units of Erlang, and the expected blocking probability shall also be given. The Erlang-B formula can then provide the number of channels required. The bandwidth demand for a given route can finally be calculated by multiplying the number of channels with the bandwidth demand of a single voice call. In case of the the pipe model the offered traffic in Erlang is given for each pair of nodes in the VPN, which enable the calculation of the bandwidth demand between each node pair. In case of the hose model, the total traffic in Erlang injected into the network by a VPN endpoint is given, and the Erlang formula is used to calculate the total bandwidth that can be injected into the network by the endpoints.

In order to quantify the robustness of the two dimensioning methods, we assume a network carrying voice traffic only and use the Erlang formula for dimensioning and blocking probability calculations.

The actions taking place when an end user initiates a new call can be followed in Figure 12. Note that the message names appearing in the figure are for illustration purposes only, they don't imply the use of any particular call control protocol.

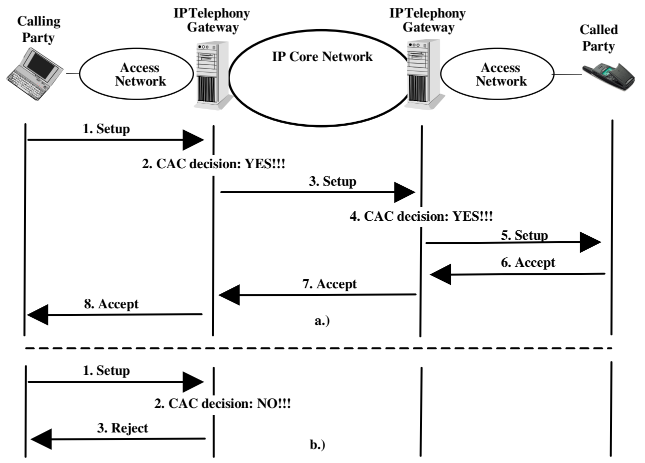

Fig. 12. Sequence of actions for successful (a.) and unsuccessful (b.) call establishment

A call setup message travels through the access network of the calling party, and arrives at the first IP telephony gateway. The gateway derives the address of the remote gateway from the address of the called party received in the call setup message. It compares the resource requirement of the new call to the resources available in its egress hose (to the resources available in its pipe towards the selected remote gateway). If there is space for the new call, it is accepted, and the call establishment message is forwarded to the remote gateway (Step 3 of Figure 12.a). If the resource limit of the hose/pipe would be exhausted, the new call is rejected (Step 3 of Figure 12.b). When making the call admission control decision (Step 4), the second gateway uses its own egress hose/pipe parameters for making the call admission control decision for the reverse direction.

The calculation is as follows:

1) We generate a network topology, a traffic matrix and the

hose model parameters as we did in the previous section.

2) We dimension the links by the pipe method. Obviously, the resulting network capacities will be able to carry the original traffic matrix with the grade of service guarantees assumed at the dimensioning.
3) Now we randomly generate a new traffic matrix that is still within the original hose model parameters.
4) Then we try to enter the new traffic matrix to the network. We calculate the blocking probabilities assuming the pipe capacities are according to the dimensioning in Step 2.
5) We go back to step 3 until we obtain a good statistical average on the blocking probability.
6) We repeat the calculation of extra blocking using different factors of pipe overdimensioning compared to the original resource allocation of Step 2.

This way we can compare the amount of blocking caused by the changing traffic conditions in the pipe model compared to the unaltered blocking rate provided by the hose model. It is very difficult to come up with a reasonable model about the variation of the traffic matrix. We considered the following three methods:

- When generating the randomly altered traffic matrix, the maximum change in the traffic volume to a particular destination is maximized by a fraction of the original traffic in this direction. In this case, an overprovisioning factor of 2 will result in no extra blocking given any change of the traffic matrix. Since this overprovisioning factor is around the same level as the overprovisioning requirement of the best known hose realization, there is no point in running simulations for finer characterization of this case.
- When generating the altered traffic matrix, the capacity requirement in a given direction is maximized by a fraction of the total load, and it does not depend on the network size.
- When generating the altered traffic matrix, the capacity requirement in a given direction is maximized by a fraction of the total load, and this maximum value is scaled down as the network size increases.

Calculation results using case 2 and in case 3 are shown in Fig. 13 and Fig. 14. Note that the hose model as well as the default (step 2) dimensioning for the pipes has been calculated for providing blocking probability of 0.001.

In Fig. 13 we maximized the load of a single pipe as $1/3$ of the total load generated by a specific node. Of course, when the network size increases this leads to more and more unequal traffic distributions. This results in a significant increase in blocking probabilities compared to the hose case. Observe that even an overprovisioning factor of 5.5 can not prevent a significant increase in blocking probabilities in case of large networks dimensioned according to the pipe model.

Taking a look at Fig. 14 reveals that assuming less stressing traffic distributions leads to much nicer behavior. This chart depicts the case when the load to a specific site is maximized

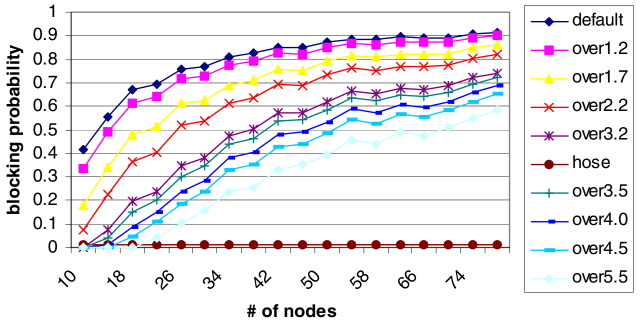

Fig. 13. Blocking probability due to inaccurate traffic model estimation assuming different overprovisioning factors (case2)

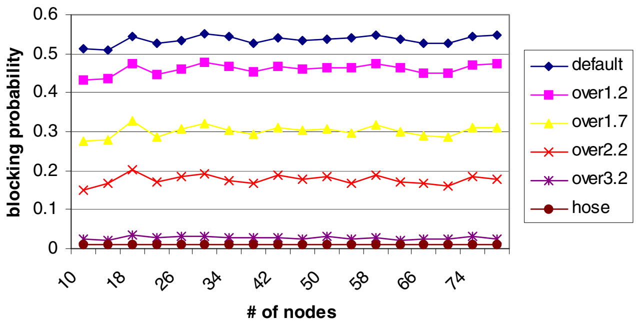

Fig. 14. Blocking probability due to inaccurate traffic model estimation assuming different overprovisioning factors (case3)

as: $total\_load \cdot 4/network\_size$. We can see that an overprovisioning factor of 3.2 which is significantly less than the overprovisioning requirement of the VPN-specific state based realization of the hose model renders the blocking performance comparable to that of the hose model.

We must emphasize that all these tests were based upon certain assumptions about the distribution of the traffic between the peer client nodes. Another limitation of these results is that we assume that the hose parameters can precisely be determined a priori[5]. Of course, if the users generate more traffic the hose model also results in larger blocking probabilities.

VI. Conclusions

In this paper we investigated the bandwidth efficiency of the recently proposed hose virtual private network provisioning model. We derived numerical results for comparing the bandwidth requirement of different realization alternatives of the hose model with the traditional pipe model, assuming a wide range of network sizes and topologies.

As far as bandwidth efficiency is concerned we note that the average overprovisioning factor increases with the network size even for the VPN–specific state realization, which is the most sophisticated realization of the hose model using

[5]This seems to be a realistic assumption. For example, the number of echo cancellers or the voice coding capacity of the codecs in the IP telephony gateways can be used for calculating a strict upper bound of the amount of traffic injected by the node.

shortest path routing. Furthermore, we observed that the overprovisioning factor is more or less constant for tree routing based hose realizations. We further observe that the average overprovisioning factor for the VPN–specific state hose realization increases with the topology density, while it is again basically constant for for tree routing based hose realizations and that the addition of new links to the topology does not improve bandwidth efficiency in case of the VPN–specific state realization.

The experiments with randomly altered traffic matrices revealed that a significant amount of extra blocking is incurred by the pipe model if extreme traffic distributions between the peer nodes are also considered. However, when limiting somehow the level of uncertainty in the traffic matrix, modest level of overprovisioning (a factor of 2-3) can eliminate the extra blocking.

Our main conclusion can be stated as that the hose model in general is a viable alternative to the traditional pipe model in all cases when the uncertainty of the traffic distribution is high. The overprovisioning of the hose model can be well offset by the benefits it delivers in terms of flexibility, reduced blocking, decreased traffic loss, ease of specification.

The numerical results provided in this paper makes the picture about the hose model more complete. Together with the other recent results available in the literature, an operator can now decide in each and every case whether to opt for the pipe model or for the hose model. Interesting theoretical questions, however, still remain for future work. Such a study item is to find an upper bound of the overprovisioning factor or to show that such an upper bound does not exist.

ACKNOWLEDGMENTS

The authors would like to thank Peter Dahlberg, Lars Westberg and Balázs Szviatovszki for many useful discussions.

REFERENCES

[1] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. V. der Merwe, "A flexible model for resource management in virtual private networks," in *ACM Sigcomm*, San Diego, California, USA, August 1999.

[2] J. A. Fingerhut, S. Suri, and J. S. Turner, "Designing least-cost non-blocking broadband networks," *Journal of Algorithms*, vol. 24, no. 2, pp. 287–309, August 1997.

[3] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, "Algorithms for provisioning virtual private networks in the hose model," in *ACM Sigcomm*, Cambridge, Massachusetts, USA, August 2001.

[4] G. Italiano, R. Rastogi, and B. Yener, "Restoration algorithms for virtual private networks in the hose model," in *IEEE Infocom*, New York, New York, USA, June 2002.

[5] J. Moy, "OSPF version 2," Internet Engineering Task Force, Request for Comments (Proposed Standard) 2328, Apr. 1998.

[6] M. Berkelaar and J. Dirks, "lp_solve 2.2," `ftp://ftp.es.ele.tue.nl/pub/lp_solve`.

[7] B. Hajek, "A tutorial survey of theory and applications of simulated annealing," in *Proceedings of 24th Conference on Decision and Control*, Ft. Lauderdale, Florida, USA, December 1985.

[8] B. Hajek, "Cooling schedules for optimal annealing," *Mathematics of Operation Research*, vol. 13, no. 2, May 1988.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Readings, MA, USA, 1989.

[10] F. Glover and M. Laguna, *Tabu Search*. Kluwer Academic, 1997.

[11] D. Orincsay and B. Józsa, "Random graph generator (for telecommunication networks)," *Ericsson Technical Report*, April 1999.

[12] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of IEEE Infocom '96*, San Francisco, CA., USA, March 1996.

[13] *http://www.erlang.com.*