ADAPTIVE BANDWIDTH MANAGEMENT AND QoS PROVISIONING IN IPVPNs

C. Hota,[*] S. Jha,[**] and G. Raghurama[***]

Abstract

An IP Virtual Private Network (VPN) uses a major share of physical resources of a network to satisfy customer's demand for secure connectivity and Quality of Service (QoS) over the Internet. Service Level Agreements (SLAs) are often used to provide bandwidth-guaranteed VPNs on networks that do not support reservation. To meet these SLAs, service providers overprovision the bandwidth allocation. This is effective, but not economic and does not enforce compliance by the customer, with potentially adverse consequences for charging and congestion control mechanisms. This article proposes an agent-based approach to dynamically adjust the allocated bandwidth as per the user's requests so that the VPNs that are carrying real time or multimedia traffic can be allocated with required amount of bandwidth. The Agent processes acting on behalf of VPN users may decrease their allocated capacity if the users underuse the allocated quota so that the Service provider can satisfy few additional demands. We propose distributed bandwidth resizing algorithms for optimizing inter-VPN and intra-VPN bandwidth allocations. This leads to an increased number of VPN connections and better utilization of network resources. The simulation results of the proposed adaptive algorithms show efficient utilization of network bandwidth among the VPN users.

Key Words

Virtual private networks, bandwidth allocation, Quality of Service, agents, user satisfaction

1. Introduction

A Virtual Private Network (VPN) is usually created through the provision of private connectivity on public network infrastructure [1]. A network-based IP-VPN uses Internet Service Provider's (ISP's) core routers to build the VPN [2]. Quality of Service (QoS) guarantee is becoming a significant challenge for the VPN service providers as VPN users want to have real time applications such as IP telephony, interactive games, teleconferencing, videos and audios etc., over their VPN connections [3].

The VPNs have traditionally been deployed for reasons of economy of scale, but have either been statically defined, requiring manual configuration, or else unable to offer any QoS guarantees. In order to achieve the statistical multiplexing in the providers' network, and thus increase the utility of the underlying network, the bandwidth allocated to each tunnel should be dynamically adjusted [4, 5]. The user may not be able to specify traffic matrix accurately in the Service Level Agreement (SLA). Hence, the admission control and tunnel setup must be supplemented with finer levels of dynamic control over the bandwidth allocation to maximize the network usage, and user satisfaction. Our approach is different from the bandwidth management approach discussed in literature [5–7] where, the resource management task is solely done by the edge device.

There are two different ways for bandwidth management: one is bandwidth reallocation and the other is path rerouting. As an example, Figs. 1–4 show reallocations and path rerouting. Fig. 1 shows the initial tunnel set ups. The spare capacities available over L1 and L2 are 2 Mbps (12-5-5), and 20 Mbps (30-10) respectively. Assume that there are no tunnels currently provisioned over links L3 and L4. Fig. 2 and 3 depict separate instances of bandwidth reallocations. In Fig. 2, T1's additional 2 Mbps demand is given from the available spare capacity. However, in Fig. 3, T1's requirement is satisfied by releasing 2 Mbps

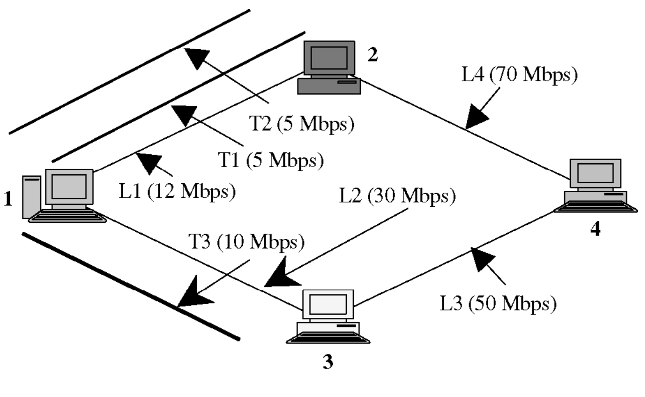

Figure 1. Initial tunnel setups and link capacities.

[*] Computer Science and Information Systems Group, Birla Institute of Technology and Science, Pilani 333031, India; e-mail: c_hota@bits-pilani.ac.in
[**] School of Computer Science & Engineering, University of New South Wales, Sydney, NSW 2052, Australia; e-mail: sjha@cse.unsw.edu.au
[***] Electrical and Electronics Engineering Group, Birla Institute of Technology and Science, Pilani 333031, India; e-mail: graghu@bits-pilani.ac.in

Recommended by Prof. Mohamed Ould-Khaoua
(paper no. 202-2223)

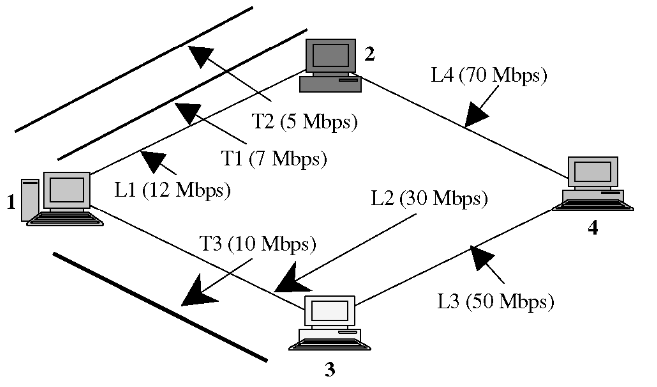

Figure 2. Bandwidth reallocation when T1 needs 2 Mbps additional capacity.

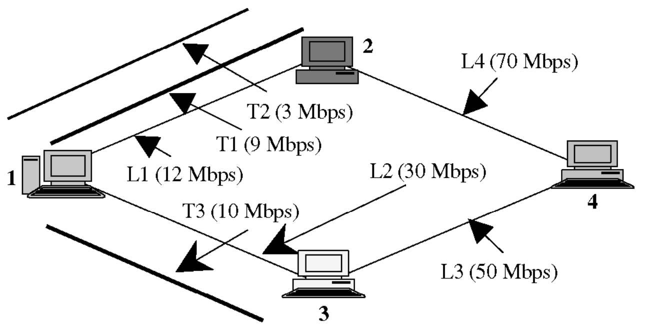

Figure 3. Bandwidth reallocation when T1 needs 4 Mbps additional capacity.

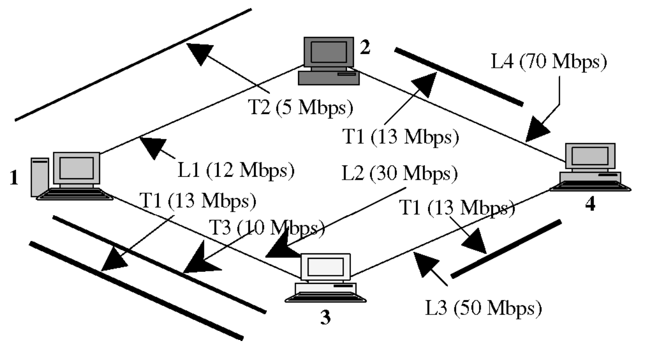

Figure 4. Bandwidth path rerouting when T1 needs 8 Mbps additional capacity.

from T2 and the spare capacity of 2 Mbps. Of course, the assumption here is that T2 is underused. Fig. 4 depicts the path rerouting scenario where, T1's requirement to reach at node 2 starting from node 1 was met by an alternate route over L2, L3 and L4. This is because the resources available over L1 were not sufficient.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we define the system model. In Section 4, we describe algorithms for adaptive bandwidth management that describe how the additional amount of bandwidth is computed by the Agent process (AP), and how Bandwidth Managers (BMs) compute the tunnel path and manage the requests of these agents. In Section 5, we report experimental results and finally, in Section 6, we conclude and discuss future work.

2. Related Work

Mitra and Ziedins [6] have proposed a hierarchical virtual partitioning scheme where they optimally allocate bandwidth on a single link. Network state and traffic load decide priority of a VPN. They do not consider the bandwidth allocation in the core network. Garg and Saran [7] have proposed a stochastic fair sharing approach for bandwidth management. They have considered normalized usage of a fraction of link bandwidth. The capacities are dynamically modified, i.e., increased upon session arrival and decreased on completion. Duffield *et al.* [8] propose a hose model that analyses the need for aggregating the traffic that belongs to a single VPN. They reserve capacity in the core network, where the capacity needs to be dynamically allocated. They used statistical multiplexing to decrease aggregate bandwidth requirements. Later, they resized the initial allocation on the basis of online measurements. Lin *et al.* [9] have proposed virtual path bandwidth resizing algorithm, routing and rerouting algorithms for managing virtual paths in ATM networks. They have considered the bandwidth utilization and the different degrees of urgency between increase and decrease of the bandwidth. Chan *et al.* [10] have shown that the network utilization can be improved by managing the bandwidth dynamically as per the arrival rate of the user requests. Saito [11] compares static and dynamic resource allocation mechanisms in ATM networks and shows that dynamic allocation is promising for situations where a priori reference model is unclear. Pitsillides *et al.* [12] have proposed hybrid schemes that combine classical constrained and genetic algorithms for solving efficiently the bandwidth allocation for virtual paths problem.

Most of the above schemes have considered the hierarchical traffic differentiation methodology for managing bandwidth in VPNs. They do not consider the utility functions suitable to VPN users which we have addressed in this paper. Also the algorithms proposed here use an agent-based distributed approach. This distributed solution outperforms the centralized solutions earlier proposed for dynamically managing the bandwidth in VPNs. An initial version of this paper appears in [13].

3. System Model

The VPN tunnel path from source to destination is computed using hill climbing shortest path heuristics that is better from [14] in terms of time taken to reach at the destination. Here, it uses link cost as a metric for selecting a better link from many available. Algorithm 2 in Annexure describes this heuristics. We use one AP for each user at the Customer Premises Equipment (CPE) device and BMs at all the ISP domains including the CPE. The AP is created by the BM for each user of the tunnel that is responsible for monitoring the bandwidth usage of the corresponding user. If it is necessary to resize the bandwidth currently allocated to a user in the VPN, then the corresponding AP sends a request to the local BM. The BM in turn sends this to all the BMs in the path of the tunnel that was earlier computed. A distributed consensus protocol is run by all the BMs to reach at an agreement (Algorithm 5 in Annexure).

For better understanding let us look at an example. Let the link capacity of a link be C units. Let the utility function for every user 'i' in the tunnel be denoted by $u_i(b)$ that is computed by the AP using following equation:

$$u_i(b) = \frac{R}{b_i} \tag{1}$$

Here, R is the traffic rate and b_i is the allocated bandwidth.

Let the single link serve several VPN tunnels. Let there be a partitioning parameter α for each tunnel 'i'. The BM computes the bandwidth that is to be allocated to each tunnel using the equation given below:

$$C_i = \alpha_i * C \text{ where } 0 < \alpha_i \leq 1 \tag{2}$$

The spare capacity over the link with N number of tunnels is given as:

$$SC = C - \sum_{i=1}^{N} C_i \tag{3}$$

Let each tunnel 'i' support n_i number of users. The BM allocates initially b_i amount of bandwidth to the ith user in tunnel 'i' as:

$$b_i = \frac{C_i}{n_i} \tag{4}$$

Using (1) and (4) AP computes the utility function for every user in the tunnel.

Two thresholds (Low and High) are defined for the utility function of any user. The following heuristics are used by APs to send a request to the (BMs):

- If utilization is greater than or equal to high, then an increase (for additional δ amount) request is sent.
- If utilization is lower than or equal to low, a request is sent to decrease the bandwidth.

Upon receipt of a request, the local BM broadcasts a request message to all the local APs asking for their utility functions. It also sends this bandwidth increase or decrease message to all the BMs in the tunnel path. BM computes the total utility of the tunnel with 'n_i' number of users by using:

$$U_t(b) = \sum_{i=1}^{n_i} u_i(b) \tag{5}$$

Every tunnel also has an upper threshold, i.e., Max_$U_t(b)$. BM uses the conditions given in the following equation for making a decision regarding reallocation or rerouting.

$$b_i = \left\{ \begin{array}{l} b_i + \delta, \text{ if } U_t(b) \leq \text{Max }\_U_t(b) \text{ \& } \delta < SC \\ b_i + \delta_{victim}, \text{ if } U_t(b) \leq \text{Max }\_U_t(b), \delta \\ \quad > SC \text{ \& } \delta \in V \\ (((\alpha_i + \sigma) * C)/n_i) + \delta_{victim}), \\ \text{if Max}\_U_t(b) < U_t(b), \delta < SC, 0 \\ \quad < \sigma \leq 1 \text{ \& } \delta \in V \\ ReCompute\ TunnelPath\ Otherwise \end{array} \right\} \tag{6}$$

The above equation depicts four different cases of bandwidth management. Here, δ is the additional amount requested. Requested amount is allocated from spare if we have spare capacity available in the link and current utilization is under the upper bound. This is characterized by the condition 1. If spare capacity is not available, and tunnel utilization is not exceeding the threshold, then choose a victim, release the required amount of bandwidth from it, and give this to the requesting process. This is formulated in condition 2. Third case shows that utilization has gone up, but the spare is sufficient to handle the request, then first resize the tunnel bandwidth by changing α and then add an additional δ requested. For this case as well choose a victim, and decrease its bandwidth by an appropriate amount. Fourth case shows that the spare is not available and also the utilization of the tunnel has gone up. This case is of bandwidth path rerouting. To handle this, find out another shortest path between source and destination of the tunnel and reallocate bandwidth to it.

4. Proposed Algorithms

The detailed distributed adaptive algorithms are given in the Annexure. We describe here the relevant data structures used and explain with an example, the applications of these algorithms. The data structures that are used in all these algorithms are described as below.

- DQ: A queue to hold the deferred tunnel paths for bandwidth path rerouting.
- ActiveList: A linked list that has an entry for every active user in the tunnel path.
- SLA: The service level agreement that is to be supported for every source destination pair.
- TS_{ij}: A set that contains the tunnels that are formed over a single link l_{ij}.
- Vote[ ]: A Boolean array.
- P: A set consisting of tunnels that form a path between a source destination pair.

An application of algorithms is explained using Fig. 5. Let the shortest tunnel paths computed by Step 2 of Algorithm 1 for both the branches be S1 $\rightarrow$ C1 $\rightarrow$ C3 $\rightarrow$ C4 $\rightarrow$ C7 $\rightarrow$ D, and S2 $\rightarrow$ C2 $\rightarrow$ C1 $\rightarrow$ C3 $\rightarrow$ C4 $\rightarrow$ C7 $\rightarrow$ D. Step 3 computes number of tunnels over each link. DQ is a deferred queue to hold the tunnels which are reformed. Step 4 computes initial capacity for each tunnel in the link. Step 5 computes initial capacity of each

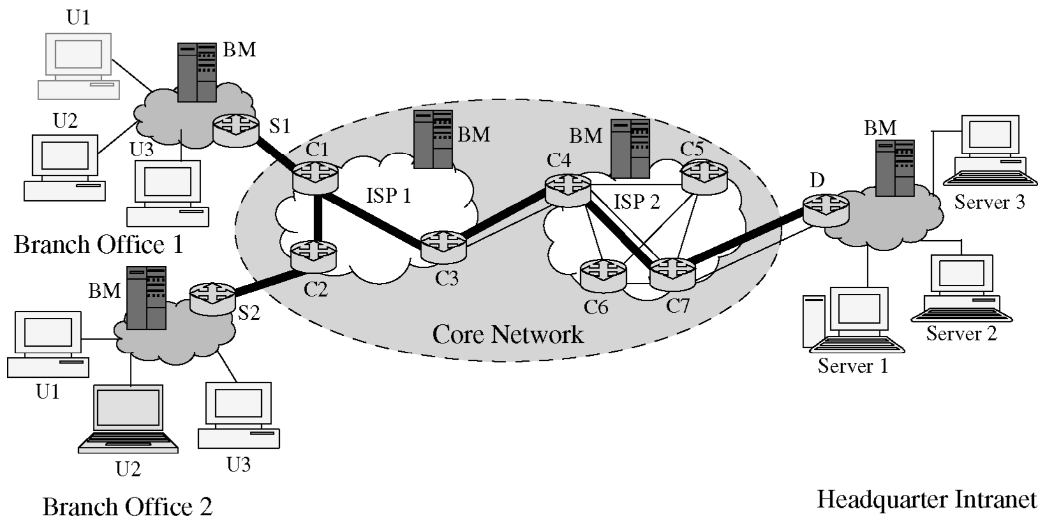

Figure 5. Intranet VPN scenario with BMs.

user in the tunnel. Step 6 computes the spare capacity on the link. Step 7 is the main part of this algorithm where the BM interacts with APs and peer BMs. When the first packet comes from a user, the network element sends new user request to local BM that creates AP, makes an entry of this active user in a Linked List called $ActiveList$. BM sends the initial bandwidth allocated to the AP. If the message received is for bandwidth increase from any of the earlier existing APs then it collects the utilities of each user in the tunnel by using Algorithm 3 and 4. BM then decides to either resize the capacity allocated to another user or the capacity allocated to the tunnel. The victim agent is selected by using a greedy approach for simplicity reasons. The victim AP is the process that has lowest utilization among others. This victim process can only be selected for a maximum number of times as defined in the variable limit. This is done for avoiding starvation of selecting VPN users that underuse their capacity over a long period of time. Once a victim agent is selected, spare capacity plus the remaining requirement is taken from the victim agent. On the other hand, if all the users in a single tunnel are efficiently utilizing their allocated quota, then tunnel capacities are resized by changing the partitioning parameter (α) and then allocating the required amount of bandwidth to meet the increasing demand of the current user. If neither of the above said conditions are true then entire tunnel path is recomputed and again the same set of steps are repeated expecting that we will get a path with higher bandwidth availability. Before a BM decides to satisfy the increased demand of a dynamic request, it first ensures with other BMs in the path regarding whether they have sufficient capacity available. This is done by running a consensus protocol, i.e., Algorithm 5 given in Annexure. For a bandwidth increase request, if the first BM (leader) gets yes reply from all the followers then consensus is reached. If the request is for termination, then it terminates the agent and increases the spare by the current allocation of the terminating user. Algorithm 3 models the behaviour of an AP that monitors the user activity. It computes the user utilization. If utilization is greater than the SLA defined upper bound, it computes η that is an incremental step, using exponential averaging defined in Algorithm 4. If the user utilization is below the lower bound, it requests for a decrease in bandwidth. This is based on a deterministic value, as we do not need to predict the future samples in this case.

5. Simulation Results

We implemented our algorithms in a distributed platform using Unix threads and communication application programming interfaces (APIs). Each BM forks one thread for a single user. These threads communicate with the BM using message send and message receive primitives. The communication between peer BMs in different domains is also done using send, and receive APIs. The synchronization between threads and the BM; between BMs is done by Unix conditional variables and Mutexes. The network topology is input to the algorithm using a two dimensional matrix, where the row, column represent a link and the cross section represents the cost over the link. The BMs are tied to the concerned node in the topology by a TCP/IP socket. Each user process is created as a foreground process, and BM, and APs are created as background processes. User processes communicate with Agent and BMs using socket API. Here, user processes act as packet generators.

The results of simulation runs are as shown in Figs. 6–8. The adaptive bandwidth management is done by the BM at the CPE device. Fig. 6 shows that with less number of tunnels over a link, not much benefit is achieved from bandwidth utilization point of view. For example, the curve is linear up to 50 users for number of tunnels equal to 1 and 2 over a link. But, we do get utilization better for this range (0 to 50) if number of tunnels is more (e.g., 9 and 13). For large number of users, numbers of tunnels do not play any role in reducing the bandwidth utilization significantly. The behaviour is nearly the same. This is shown in the graph for number of users equal to 50 and above. This behaviour is resulted because of the fact that we have to reroute many user packets from the earlier established tunnels when the number of users is large. But, for links with more tunnels still we can get a better performance than the links with less number of tunnels. Fig. 7 shows that irrespective of number of tunnels, up to some value of traffic rate (30 kbps), utilization decreases very slowly. But as we increase the traffic rate, for less number of tunnels the utilization increases where as for more number of tunnels, it decreases sharply. This is because when the average traffic rate is low, BM allocates

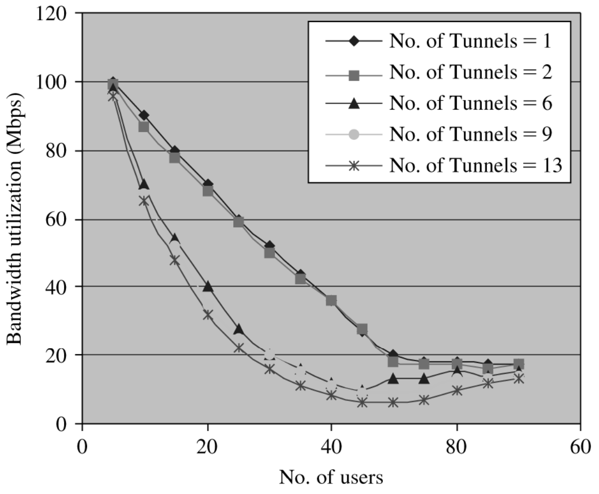

Figure 6. Utilization of a link with number of users.

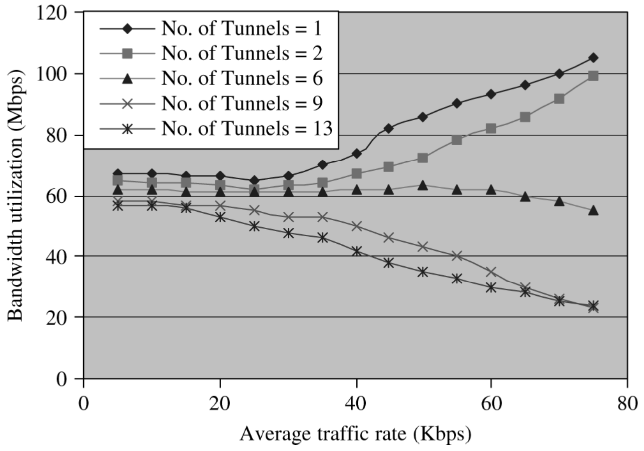

Figure 7. Utilization of a link with average traffic rate.

a major share of bandwidth to users, and when the average rate is high, the reallocation happens frequently. The more the number of tunnels on a link more are the reallocations. Fig. 8 shows that with less number of tunnels, the Request Blocking Rate (RBR) increases linearly with increase in average traffic rate. But with more number of tunnels in a link, the RBR drops significantly. This is because, with high average traffic rate and more number of tunnels, chances of getting victim agents is larger. These victim APs are those that underuse their capacity.

6. Conclusion

In this paper, we have proposed adaptive algorithms for bandwidth resizing for IPVPNs using software agents. The adaptive algorithms are designed keeping in view the efficient network utilization (ISP's) and the different degrees of urgency between the users. Our algorithms are user oriented. The way the link bandwidth or the bandwidth allocated to tunnels over the link is partitioned depends on the user's utility. Each sudden change in the users requirement is addressed by a linear change in the partitioning parameter (α). By using the approach suggested here VPN service providers can satisfy more number of users with a quality of service guarantee and also at the same time they can improve upon their revenue.

The proposed algorithms are scalable. They work fine with a decent number of user connections per tunnel. The workload of BMs can be distributed by either using broker architecture or by using distributed scheduling approaches. We plan to investigate these in our future work.

Acknowledgement

This work is supported by AVCC India Exchange programme grant, The University of New South Wales, Sydney, Australia. We thank anonymous reviewers for many helpful comments on the paper.

References

[1] D. McDysan, *VPN applications guide: Real solutions for enterprise networks* (USA: John Wiley & Sons, 2000).

[2] T. Erlebach & M. Ruegg, Optimal bandwidth reservation in hose-model VPNs with multipath routing, *Proc. 23rd Annual Joint Conf. of the IEEE Computer and Communication Societies* (*INFOCOM 2004*), Hong Kong, China, 2004, 2275–2282.

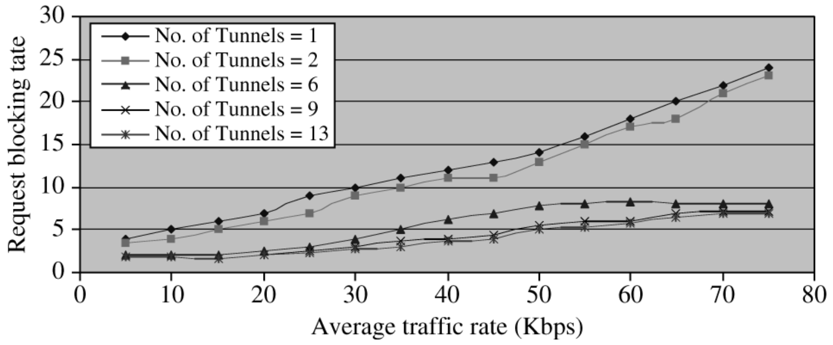

Figure 8. Request blocking rate with average traffic rate.

[3] L. Yuan, C.K. Tham, & A.L. Ananda, A probing approach for effective distributed resource reservation, *Proc. 2nd International Workshop on QoS-IP*, Milan, Italy, in M. A. Marsan, G. Corazza, M. Listanti, & A. Roveri (Eds.), Lecture Notes in Computer Science 2601 (London, UK: Springer-Verlag, 2003), 672–688.

[4] A. Juttner, I. Szabo, & A. Szentesi, On bandwidth efficiency of the hose resource management model in VPNs, *Proc. 22nd Annual Joint Conf. of the IEEE Computer and Communications Societies* (*INFOCOM 2003*), San Franciso, CA, USA, 2003, 386–395.

[5] S.H. RHEE & T. Konstantopoulos, Dynamic capacity resizing for fair bandwidth sharing in VPNs, *IEICE Transactions on Communications*, *Vol. E86-B, No.5*, 2003, 1625–1631.

[6] D. Mitra & I. Ziedins, Hierarchical virtual partitioning: Algorithms for virtual private networking, *Proc. IEEE Global Telecommunications Conference* (*GLOBECOMM 1997*), Phoenix, Arizona, USA, 1997, 1784–1791.

[7] R. Garg & H. Saran, Fair bandwidth sharing among virtual networks: A capacity resizing approach, *Proc. IEEE Computer Communications and Networking* (*INFOCOM 2000*), Tel Aviv, Israel, 2000, 255–264.

[8] N.G. Duffield, P. Goyal, A. Greenberg, P. Mishra, & K.K. Ramakrishnan, A flexible model for resource management in virtual private networks, *Proc. ACM SIGCOMM*, ACM Press, New York, USA, 1999, 95–108.

[9] Y. Lin, W. Su, & C. Lo, Virtual Path Management in ATM Networks, *Proc. IEEE International Conference on Communications* (*ICC*), Dallas, USA, 1996, 642–646.

[10] T. Chan, W. Lau, & V. Li, A measurement based congestion alarm for self similar traffic, *Proc. IEEE International Conf. on Communications* (*ICC 2001*), Helsinki, Finland, 2001, 1528–1533.

[11] H. Saito, Dynamic resource allocation in ATM networks, *IEEE Communications Magazine*, May 1997, 146–153.

[12] A. Pitsillides, C. Patichis, A. Sekercioglu, G. Stylianou, & A. Vasilakos, Bandwidth allocation for VP (BAVP): An investigation of performance of classical constrained and evolutionary programming optimization techniques, *Computer Communications*, *25*, 2002, 1443–1453.

[13] C. Hota, S.K. Jha, & G. Raghurama, Distributed dynamic resource management in IPVPNs to guarantee quality of service, *Proc. Coordinated Quality of Service in Distributed Systems (COQODS) Workshop*, Singapore, 2004, 414–419.

[14] C. Hota & G. Raghurama, A heuristic algorithm for QoS path computation in a Virtual Private Network, *Proc. International Conf. on Information Technology* (*CIT 2003*), Bhubaneswar, India, 2003, 19–24.

Biographies

Chittaranjan Hota received his bachelors degree from Amravati University, M.S. in 1990, received his masters degree from TIET (Deemed), Patiala in 1998, and Ph.D. degree from Birla Institute of Technology and Science (Deemed), Pilani in 2006. From 1990 to 2000 he worked as a faculty member at Amravati University, India. Since 2000, he has been a faculty member at Birla Institute of Technology & Science, Pilani, where he is currently an Assistant Professor in Computer Science & Information Systems group. His research interests include traffic engineering and resource management in computer networks, distributed systems, and network security. He is a life member of ISTE, India.

Sanjay Jha received his Ph.D. from University of Technology, Sydney, Australia. He is currently a Professor in the Networking group at School of Computer Science and Engineering, University of New South Wales, Sydney. His research cover a wide range of topics in networking including Wireless Sensor Networks, Adhoc wireless networks, Quality of Service (QoS) in IP Networks, and Programmable networks. He is the principal author of the book Engineering Internet QoS and a co-editor of the book Wireless Sensor Networks: A Systems Perspective. He is a member of IEEE.

G. Raghurama received his masters degree from Indian Institute of Technology, Madras in 1980, received his Ph.D. degree from Indian Institute of Science, Bangalore in 1985. After a year of postdoctoral work at IISc, Bangalore, he joined Electrical, Electronics and Instrumentation group at BITS, Pilani in 1987 where he has been involved in teaching, research, and administration. He is currently the Deputy Director (Academic) at Birla Institute of Technology and Science, Pilani. His research interests include interconnection networks, and telecommunication network management.

Annexure

Algorithm 1 Dynamic Bandwidth Management
Input: Network Graph, Source Destination Pairs, and Maximum Utilization of each Tunnel (MAX_UT).
Output: Initial Bandwidth allocated to each user, and dynamic increase or decrease of the bandwidth allocated to active users.

```
Step 1. Initializations: C ← Link Capacity, $U_T$ ← 0, MAX_UT ← Maximum Utilization, $TS_{ij} \leftarrow \phi$, P ← $\phi$, DQ ← $\phi$, Limit ← Max number of times a process can be selected as victim, Count [victim] ← 0.

Step 2. Compute Tunnel paths for all the SLAs' to be supported.
For every $SLA_i$ to be supported do
  $T_i = ShortestPath$ (source, destination);                    //Algorithm 2
  $P = P \cup T_i$;   Endfor;

Step 3. Compute the set of tunnels that use a single link.
For every link $l_{ij}$ on Service Provider's domain do
  Select an element 's' from $P \cup DQ$;
  If ($l_{ij} \in s$) then $TS_{ij} \leftarrow TS_{ij} \cup s$;   Endfor;

Step 4. Compute the initial bandwidth to be allocated to $Tunnel_i$ of link $l_{ij}$.
For every $Tunnel_i$ in $TS_{ij}$ do
  $C_i = \alpha_i * C$;   Endfor;

Step 5. Compute the initial bandwidth to be allocated to $user_i$ with $n_i$ number of users in $Tunnel_i$
For every $user_i$ in $Tunnel_i$ do
  $b_i = C_i / n_i$;   Endfor;

Step 6. Compute the spare capacity of the link $l_{ij}$
      $$SC = C - \sum_{i=1}^{n_i} C_i$$

Step 7. When a new user packet arrives, Bandwidth Manager (BM) creates an agent process (AP), sends the initial bandwidth, and runs in an infinite loop.
While (True) do
 ReceivePacket;
 If (NewUser) then                                          // Connection Establishment
  $i$ = QueueInWhichPacketArrives;
  If (ExistsinActiveList ($i$) == false) then
    AddtoActiveList ($i$);
    Increment SizeofActiveList;
    Fork AgentProcess ($i$);
    Send BandwidthAllocated ($b_i$) to Agent Process;
 Endif;
Else                                                        // Dynamic Bandwidth Management
  If (BandwidthIncreaseRequest) then
    Run ConsensusProtocol;                                  // Algorithm 5
    While SizeofActiveList $\neq$ NULL do
      Get Element (i) from ActiveList;
      Send RequestMsgforUtility to Agent Process;
      $U_u$ = Receive UserUtility;                            //Algorithm 3
      $U_T = U_T + U_u$;
    Endwhile;
    If ($U_T \leq$ MAX_$UT$) then
      If ($\delta \leq SC$) then
        If (ConsensusOK) then
          $SC \leftarrow SC - \delta$;
         $b_i = b_i + \delta$;
```

```
            Send BandwidthAllocated ($b_i$) to AP;
        Else
         $SC = SC + b_i$;
         Remove RequestfromActiveList;
         Decrement SizeofActiveList;
         Kill AgentProcess ($i$);
         $T_i \leftarrow$ Recompute the Path;                         // Algorithm 2
         DQ $\leftarrow$ DQ $\cup T_i$; Go to step 3;
        Endif;
      Else
                  Victim $\leftarrow$ ProcessWithLowestUtilization;
                  Count [victim] $\leftarrow$ Count [victim] + 1;
                  If (Count [victim] $\geq$ Limit)
                  Victim $\leftarrow$ ProcessWithNextLowestUtilization;
                  $b_{victim} = b_{victim} - \delta + SC$;
                  $b_i = b_i + \delta$;
                  $SC \leftarrow 0$;
                  Send BandwidthAllocated ($b_i$) to AP;
      Endif;
    Else
          If ($\delta < SC$) then
                  Victim $\leftarrow$ TunnelWithLowestUtilization;
                  Count [victim] = Count [victim] + 1;
                  If (Count [victim] $\geq$ Limit)
                           Victim $\leftarrow$ TunnelWithNextLowestUtilization;
                  $b_{victim} = ((((\alpha_{victim} - \sigma) * C)/n_i) - \delta)$;
                  $b_i = ((((\alpha_i + \sigma) * C)/n_i) + \delta)$;
                  Send BandwidthAllocated ($b_i$) to AP;
          Else
                  $SC \leftarrow SC + b_i$;
                  Remove RequestfromActiveList;
                  Decrement SizeofActiveList;
                  Kill AgentProcess($i$);
                  $T_i \leftarrow$ Recompute Tunnel Path              // Algorithm 2
                  DQ $\leftarrow$ DQ $\cup T_i$; Go to step 3;
          Endif;
        Endif;
      Else
         If (BandwidthDecreaseRequest) then
          For each BM in the TunnelPath
            Send BandwidthDecreaseRequestMessage;
          $SC = SC + \delta$;
         Else                                                  // Termination of the connection
          Remove RequestfromActiveList;
          Decrement SizeofActiveList;
          Kill AgentProcess ($i$);
          $SC = SC + b_i$;
         Endif;
      Endif;
    Endif;
Endwhile;
```

Algorithm 2 Shortest Tunnel Path using Hill Climbing
Input: Network Graph, Source, and Destination Pairs.
Output: Tunnel Path (T) for every Source Destination Pair.

```
Step 1. Initializations: $T \leftarrow \phi$, $I \leftarrow$ Source (S), D $\leftarrow$ Destination.
Step 2.
While (I $\neq$ D) do
  i. Select neighbor (N) along the path with least cost;
  ii. T $\leftarrow$ I $\cup$ N;
  iii. I $\leftarrow$ N;
Endwhile;
```

Algorithm 3 Algorithm for computing User Utility and the bandwidth increase or decrease (Agent Process)
Input: Maximum User Utility (MAX_UU), Minimum User Utility (MIN_UU), and Allocated Bandwidth (b_i).
Output: The current User utility or delta, the increase or decrease in the allocated bandwidth.

```
Step 1. Initializations: User utility ($U_u$) $\leftarrow 0$
Step 2. While (TRUE) do
  Receive message from Bandwidth Manager;
  If (Message type is BandwidthAllocated) then
      $b_{\text{i}}$ = bandwidth;
  Else
    If (connection is active) then
      Rate ($R$) $\leftarrow$ Token Bucket Parameters;
      $U_u = R/b_i$;
        Send User utility ($U_u$) to BM;
    Else
        Send ConnectionCloseRequest;
    Endif;
  Endif;
  If ($U_u >$ MAX_UU) then
        $\eta =$ Incr ( ) ;                  // Algorithm 4
        $\delta = \eta * R$;
        Send BandwidthIncreaseRequest of $\delta$ to BM;
  Else
    If ($U_u <$ MIN_UU) then
     $\delta = 0.8 * b_i$;
     Send BandwidthDecreaseRequest of $\delta$ to BM;
    Endif;
  Endif;
Endwhile;
```

Algorithm 4 Computing η using averaging approach.
Input: User utilizations at different time intervals (U_u)
Output: Return η

```
Step 1. Let [($t_1$, $t_2$), ($t_2$, $t_3$), ...], be the size of the observing window and [($U_u^1$, $U_u^2$), ($U_u^2$, $U_u^3$), ...] be the user utilizations at those times.
Step 2. Let $\eta_p$ and $\eta_c$ be the slope of incoming traffic in the previous and current intervals respectively.
Step 3. Compute $\eta_p = (U_u^{n-1} - U_u^{n-2})/(t_{n-1} - t_{n-2})$
Step 4. Compute $\eta_c = (U_u^{n} - U_u^{n-1})/(t_{n} - t_{n-1})$
Step 5. Compute $\eta = \rho * \eta_c + (1-\rho) * \eta_p$, where (o $< \rho < 1$)
Step 6. Return $\eta$
```

Algorithm 5 Consensus Protocol

```
Step 1. Initializations: Leader $\leftarrow$ Source BM, Follower Set (FS) $\leftarrow$ BMs in the tunnel path, Majority $\leftarrow 0$, Vote [tunneli] $\leftarrow$ False, $SC_{tunneli} \leftarrow$ Spare Capacity of tunneli.
Step 2. Leader broadcasts VoteRequest to all the elements of FS.
Step 3. For every element of FS
          If ((Vote [tunneli] == False) &&
            ($SC_{tunneli} > \delta$))
            Send Consensus to the Leader;
            Vote [tunneli] = True;
Step 4. For every Consensus message from a Follower, Leader computes majority = majority + 1.
Step 5. If (majority == Length (FS) then Leader broadcasts ConsensusOK message to all the BMs in FS.
```